08-2023

code URJC

Fundamentos de la Web

Diapositivas de teoría

Universidad Rey Juan Carlos

Micael Gallego
Correo: micael.gallego@urjc.es
Twitter: @micael_gallego

Nicolás Rodríguez
Correo: nicolas.rodriguez@urjc.es

Iván Chicano
ivan.chicano@urjc.es

Michel Maes
michel.maes@urjc.es

08-2023

code
URJC

Micael Gallego, Nicolás Rodríguez, Iván Chicano, Michel Maes

Índice de Pruebas de Evaluación

code URJC

Índice

- La asignatura se compone de 5 bloques
  - Bloque I: Introducción a la web
  - Bloque II: Tecnologías de cliente web
  - Bloque III: Tecnologías de servidor web
  - Bloque IV: Tecnologías de interactividad en el cliente web
  - Bloque V: Web semántica

08-2023

code URJC

Fundamentos de la Web

Bloque I: Introducción a la web

Tema 1.1: Historia de la web y conceptos básicos

Universidad Rey Juan Carlos

Micael Gallego
Correo: micael.gallego@urjc.es
Twitter: @micael_gallego

Nicolás Rodríguez
Correo: nicolas.rodriguez@urjc.es

Iván Chicano
ivan.chicano@urjc.es
Michel Maes
michel.maes@urjc.es

08-2023

code
URJC

Micael Gallego, Nicolás Rodríguez, Iván Chicano, Michel Maes

Universidad
Rey Juan Carlos

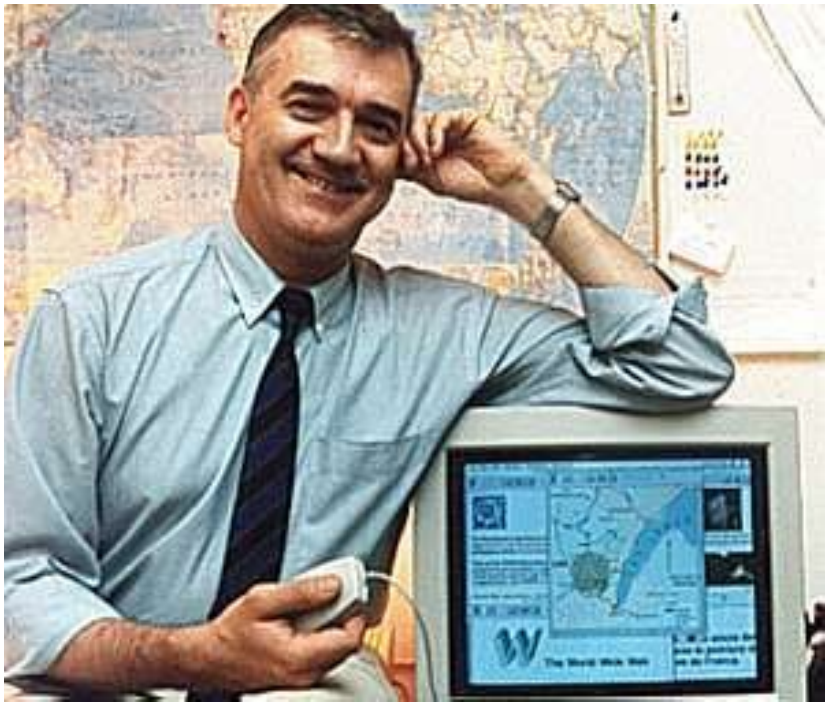
Nacimiento de la web

Nacimiento de la web

code URJC

Hace 24 años nace la web

En 1992 dos investigadores del CERN presentan la web

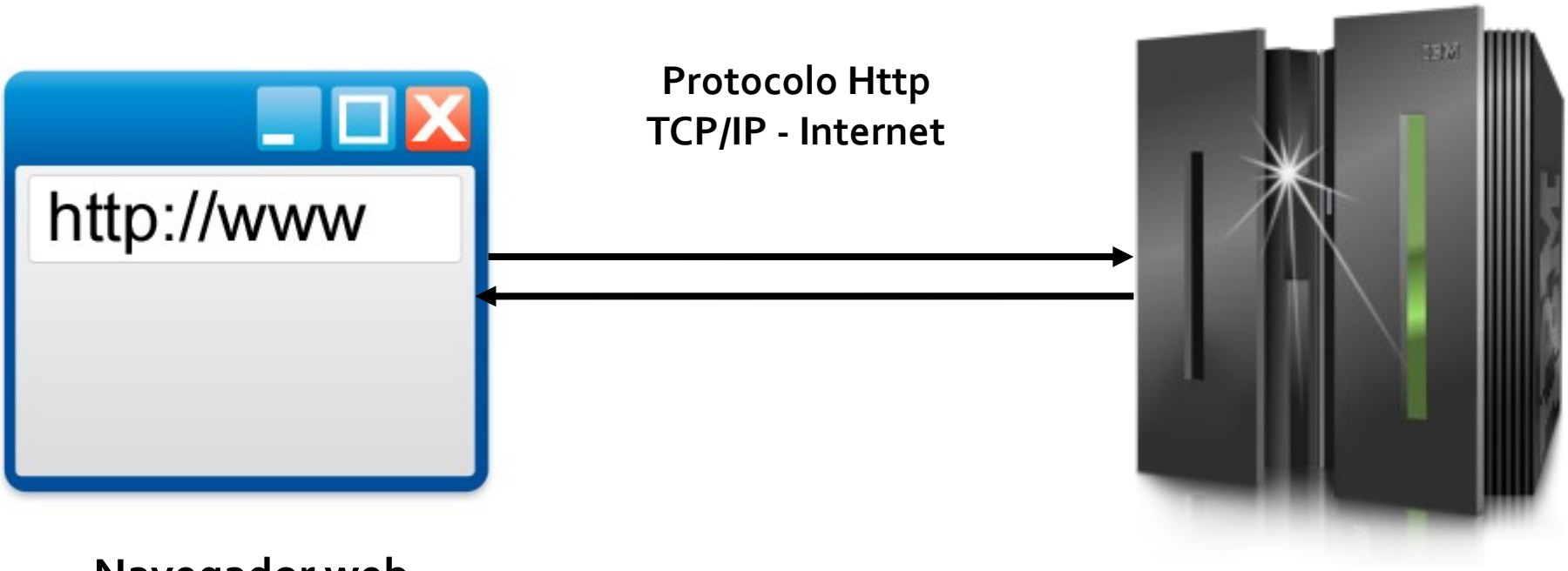


Robert Cailliau

Tim Berners-Lee

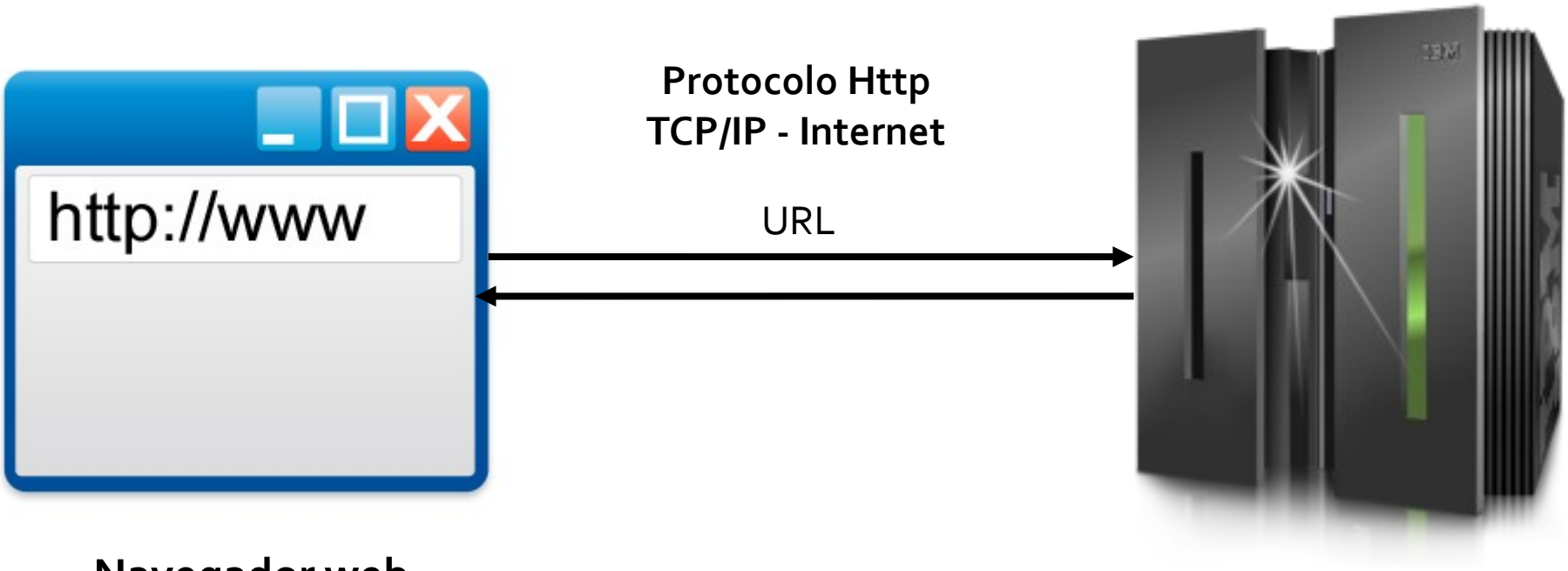
Nacimiento de la web

Elementos básicos



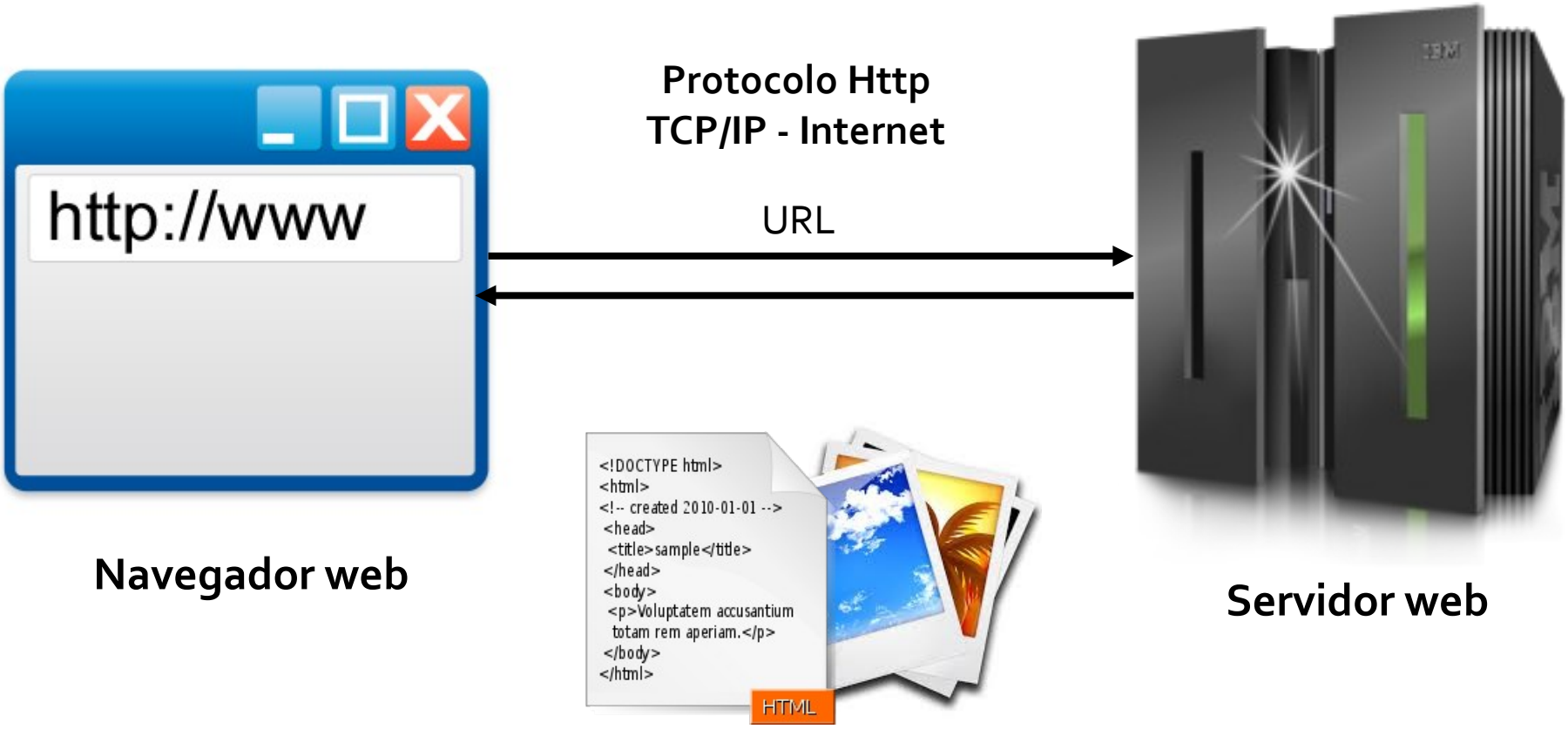
Nacimiento de la web

Elementos básicos



Nacimiento de la web

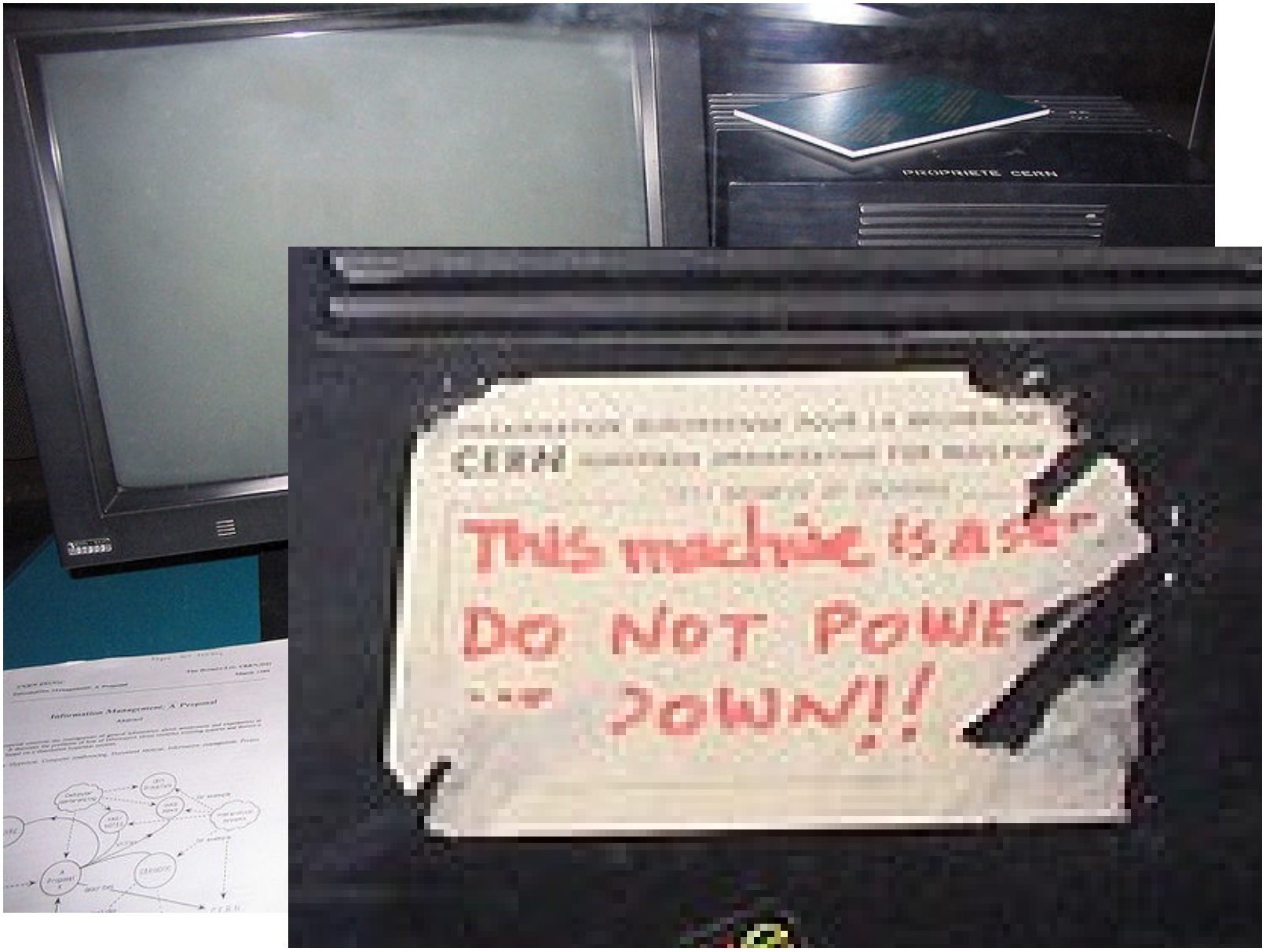
Elementos básicos



Nacimiento de la web

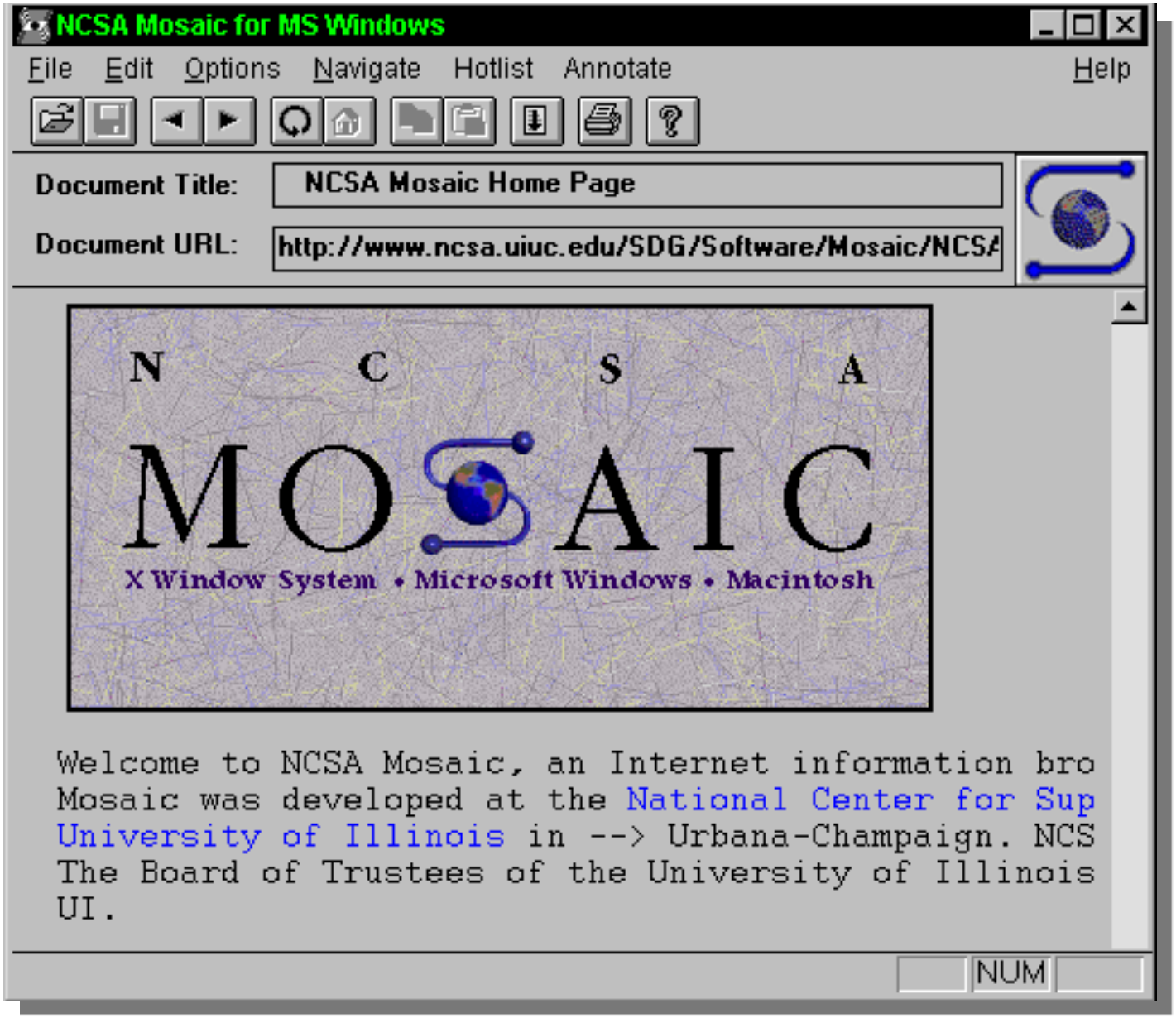
CERN
(1990)

Nacimiento de la web



CERN
(1990)

Nacimiento de la web



Mosaic (1993)

Nacimiento de la web

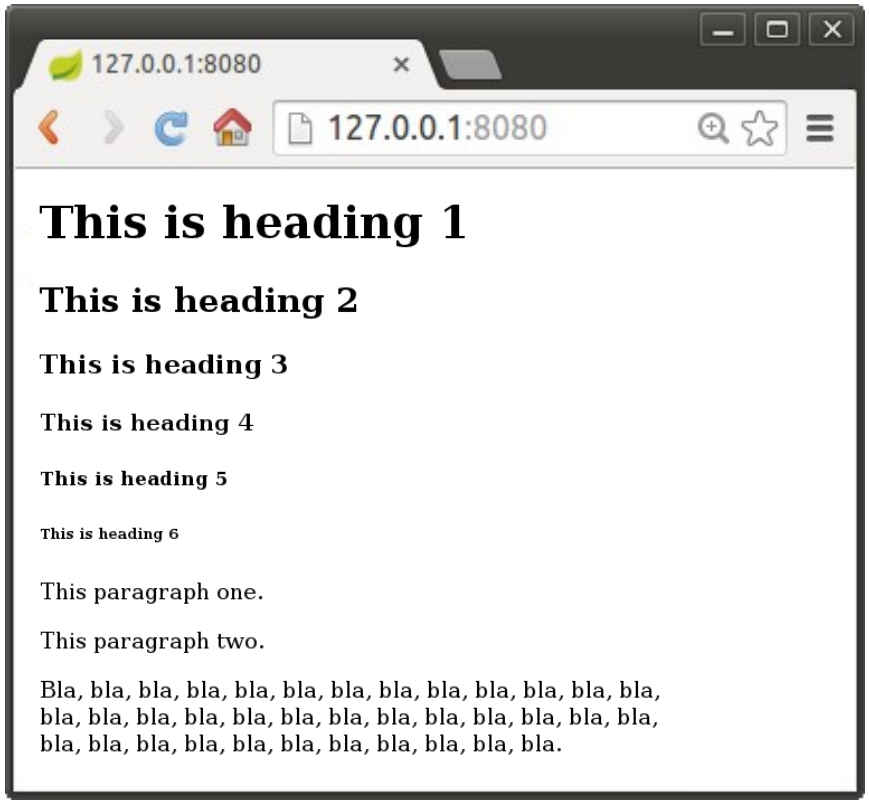
code URJC

HTML

index.html

```
<!DOCTYPE html>
<html>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>

<p>This paragraph one.</p>
<p>This paragraph two.</p>
<p>Bla, bla, bla, bla, bla, bla, bla,
bla, bla, bla, bla, bla, bla, bla, bla,
bla, bla, bla, bla, bla, bla, bla, bla,
bla, bla, bla, bla, bla, bla, bla, bla,
bla, bla, bla, bla, bla, bla.</p>

</body>
</html>
```

Visualización en el navegador



Nacimiento de la web

CSS nace a los 4 años

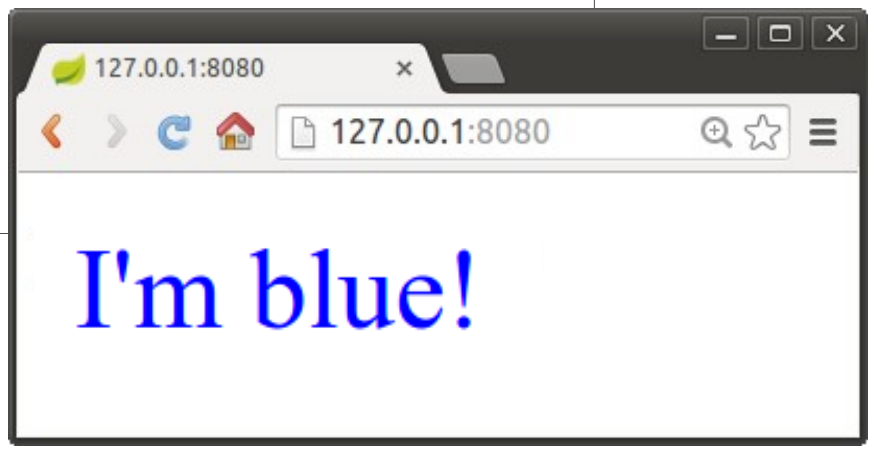
para separar contenido y estilo (1996)

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Result</title>
    <link type="text/css" rel="stylesheet" href="style.css"/>
  </head>
  <body>
    <p>I'm blue</p>
  </body>
</html>
```

style.css

```
p {
  color: blue;
}
```



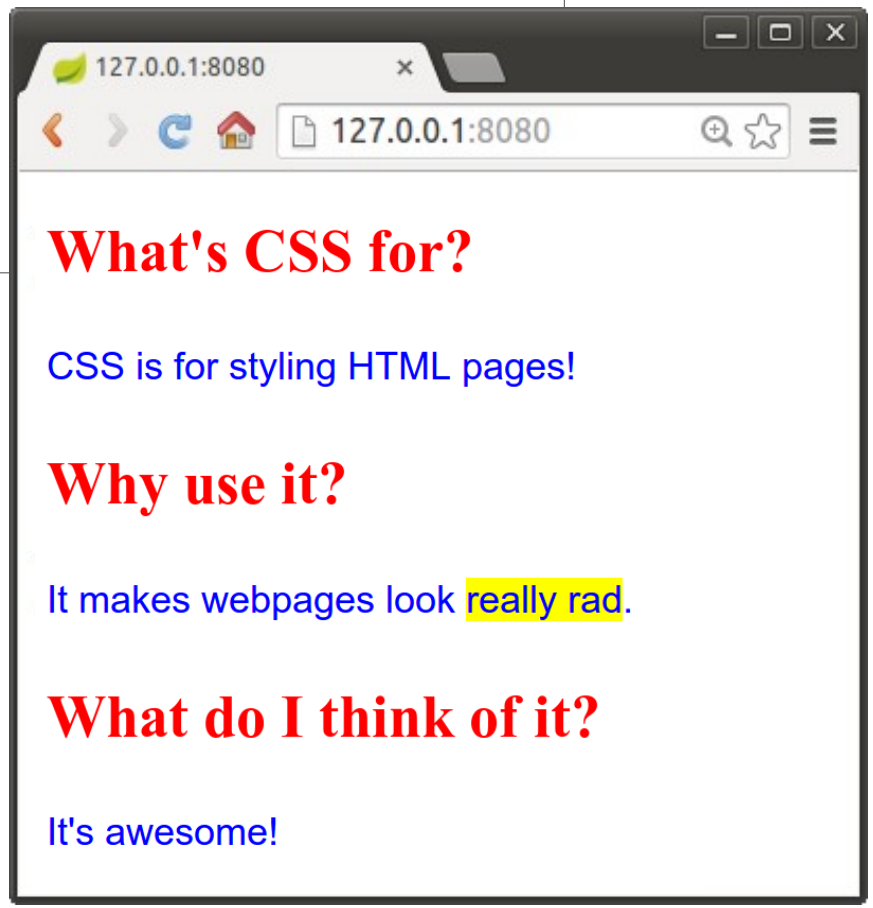
Nacimiento de la web

code URJC

CSS

```
<h3>What's CSS for?</h3>
<p>CSS is for styling HTML pages!</p>
<h3>Why use it?</h3>
<p>It makes webpages look <span>really
<h3>What do I think of it?</h3>
<p>It's awesome!</p>
```

```
p {
  font-family: Arial;
  color: blue;
  font-size: 12px;
}
h3 {
  color: red;
}
span {
  background-color: yellow;
}
```

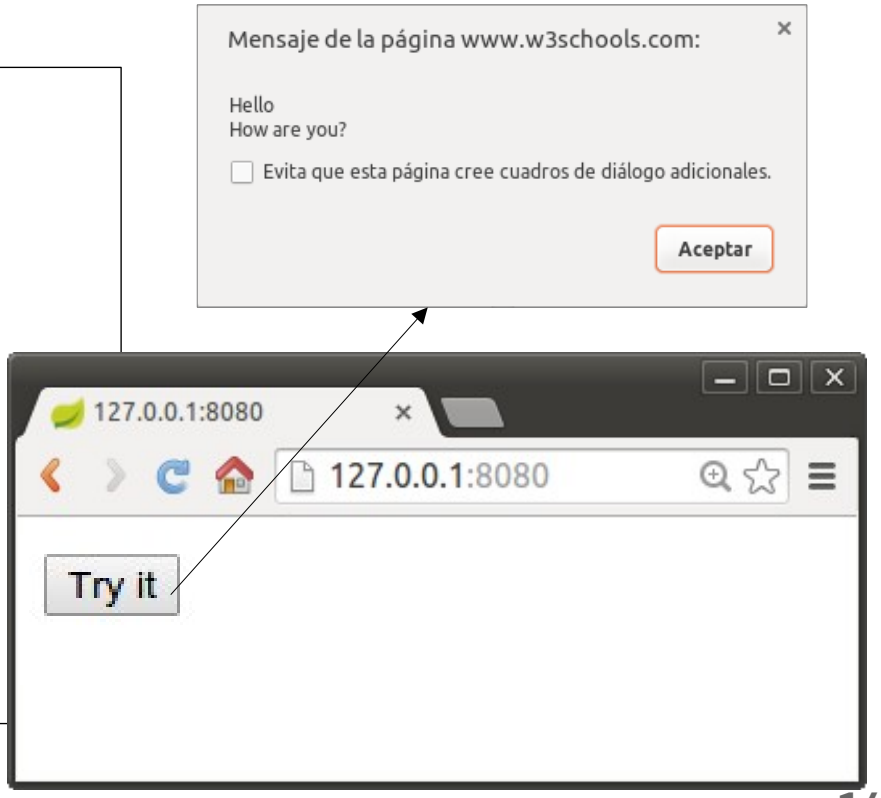


Nacimiento de la web

JavaScript nació un año antes

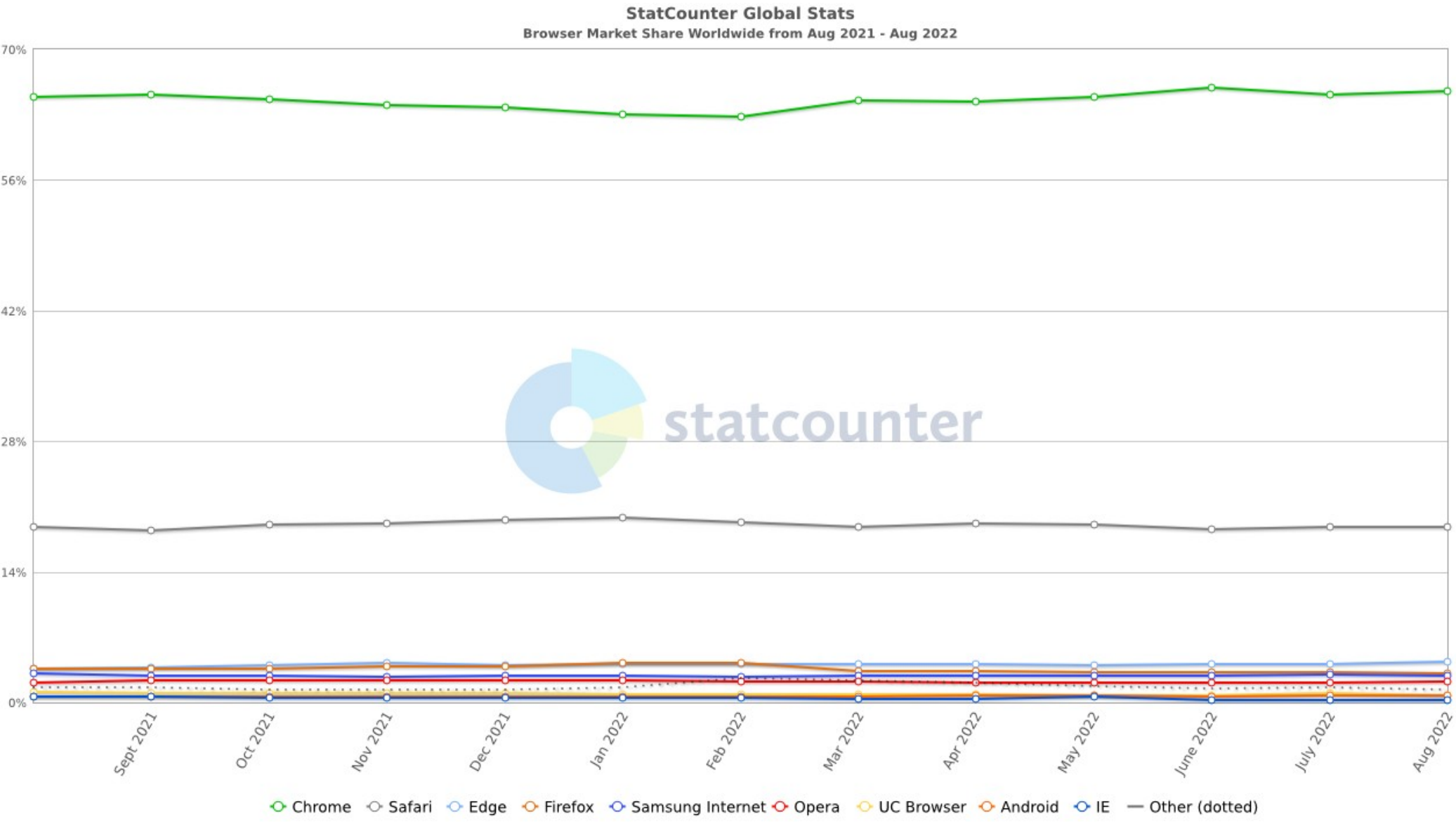
Aunque ha tenido diferentes nombres, el lenguaje apareció en el navegador web Netscape en **1995**

```
<!DOCTYPE html>
<html>
<body>

<button onclick="myFunction()">Try it</button>

<script>
  function myFunction() {
    alert("Hello\nHow are you?");
  }
</script>

</body>
</html>
```



Evolución de la web

Evolución de la web



http://gs.statcounter.com

Evolución de la web

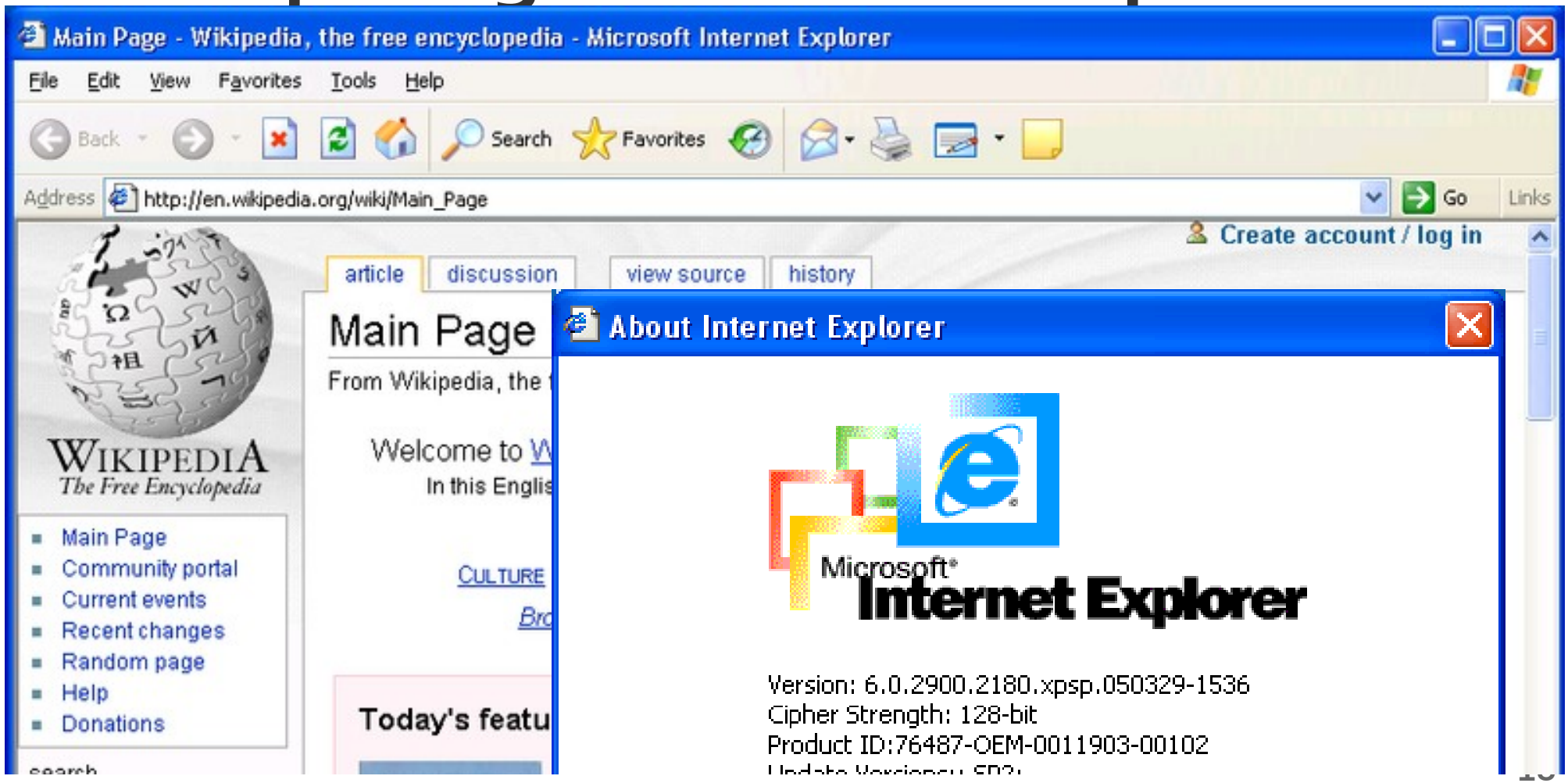
code URJC

La web ha ido creciendo

con estándares

Evolución de la web

Aunque algunos no los respetaban



Evolución de la web

code URJC

También se han usado *plugins*

no estándares y no disponibles en

todas las plataformas

Microsoft® Silverlight™

Evolución de la web

¿Y los servidores web?

Al principio los servidores web únicamente servían los ficheros del **disco duro**

(1995)

(1995)

Evolución de la web

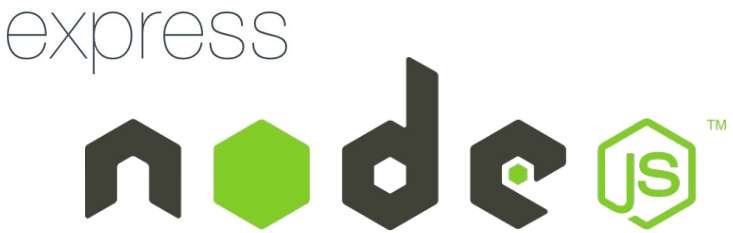
Se han transformado en aplicaciones

que generan cada página con la información que envía el **usuario** y/o la que está en la **base de datos**

Evolución de la web

code URJC

Tecnologías desarrollo web servidor



Evolución de la web

code URJC

- **Bases de datos**

  SQL (relacionales)

Evolución de la web

- **Bases de datos**

  NoSQL (no relacionales)

08-2023

code URJC

Fundamentos de la Web

Bloque I: Introducción a la web

Tema 1.2: Tecnologías y arquitecturas web

Universidad Rey Juan Carlos

Micael Gallego
Correo: micael.gallego@urjc.es
Twitter: @micael_gallego

Nicolás Rodríguez
Correo: nicolas.rodriguez@urjc.es

Iván Chicano
ivan.chicano@urjc.es

Michel Maes
michel.maes@urjc.es

08-2023

code
URJC

Micael Gallego, Nicolás Rodríguez, Iván Chicano, Michel Maes

Universidad
Rey Juan Carlos

El desarrollo web hoy

El desarrollo web hoy

code URJC

Existen varios tipos de webs

Dependiendo de cómo usan las tecnologías de **cliente** y **servidor**

Página web

Servidor estático

El servidor web sirve contenido guardado en el disco duro

Aplicación web

Servidor dinámico

El servidor web sirve contenido generado mediante código

El desarrollo web hoy

code URJC

Página web

Servidor estático

- Página web estática
- Página web interactiva

Aplicación web

Servidor dinámico

- Aplicación web con cliente estático
- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA

El desarrollo web hoy

code URJC

Página web

Servidor estático

- Página web estática
- Página web interactiva

Aplicación web

Servidor dinámico

- Aplicación web con cliente estático
- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA

El desarrollo web hoy

code URJC

- **Página web estática**
  - El navegador hace **petición** al servidor mediante **http**
  - El **servidor** transforma **URL** a ruta en **disco**
  - El **servidor** devuelve el **fichero** de disco al **navegador**
  - El navegador **visualiza** la página **HTML** con estilos CSS e imágenes (**sin JavaScript**).
  - Cuando el usuario hace **clic en un enlace**, el navegador repite el proceso con la URL del link y **recarga** por **completo** la página web

El desarrollo web hoy

- **Página web estática**
  - Se implementa con **HTML** y **CSS**
  - Se utilizan **librerías de componentes CSS**
    - Bibliotecas de componentes predefinidos
    - Diseño *responsive* (adaptativo a móviles)
    - Distribución de componentes

El desarrollo web hoy

code URJC

Página web

Servidor estático

- Página web estática
- Página web interactiva

Aplicación web

Servidor dinámico

- Aplicación web con cliente estático
- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA

El desarrollo web hoy

code URJC

- **Página web interactiva**
  - El contenido de la página web está alojado en el **disco duro del servidor** (estático)
  - El cliente es dinámico porque las páginas incluyen código **JavaScript** que se ejecuta en el **navegador**
  - Este **JavaScript** se usa para incluir **efectos gráficos**:
    - Efectos gráficos que no se pueden implementar con **CSS**
    - **Mostrar u ocultar información** en función de los elementos que se seleccionan (para documentos largos)
    - **Menús** desplegables
    - Buscadores

El desarrollo web hoy

code URJC

- **Página web interactiva**
  - Se implementan con **HTML, CSS y JavaScript**
  - Para implementar la interactividad en JavaScript se suele usa la librería **jQuery**

El desarrollo web hoy

code URJC

Página web

Servidor estático

- Página web estática
- Página web interactiva

Aplicación web

Servidor dinámico

- Aplicación web con cliente estático
- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA

El desarrollo web hoy

code URJC

- **Aplicación web con cliente estático**
  - Cuando el servidor web recibe una **petición**, dependiendo de la URL:
    - Devuelve contenido del **disco**
    - Ejecuta código para **generar el recurso dinámicamente**
  - Cuando se **ejecuta código**, normalmente se hacen **consultas a una base de datos** para recuperar la información y generar la página HTML
  - Si el usuario pulsa un link, se **recarga la página al completo**


Micael
Portal de servicios
miportal.urjc.es/portal/page/portal/portal_inicio/inicio
Inicio
Mis favoritos
Desconectar
Universidad Rey Juan Carlos
Portal de Servicios
Bienvenido/a: Micael Gallego Carrillo
Servicios Alumno
Mi Correo de Alumno - (Ayuda)
Mis datos personales
Mis notas
Mis encuestas
Mis traslados
Aula Virtual
Expediente
Mi expediente
Mi nota media del expediente
Mi progreso académico
Alumnos
Centro de atención telefónica al alumno (C.A.T.A.)
Fecha de Actualización: 06-FEB-2014
El C.A.T.A es un servicio de la Universidad Rey Juan Carlos cuyo objetivo es asistir a los alumnos y futuros alumnos en su búsqueda de información, al tiempo que asesora y orienta en las posibles cuestiones que puedan surgir en su relación con nuestra institución.
Horario del servicio:
Horario del servicio: de lunes a viernes de 9:00 a 20:00 horas
Teléfono: 91 488 93 93
Enlaces Generales
Acceso DreamSpark Premium MSDN
Enlaces Biblioteca
Recordar pin
Acceso remoto a los recursos electrónicos de la biblioteca
Enlaces Alumnos
Centro de Atención Telefónica para el Alumno (CATA)
Apple on Campus

El desarrollo web hoy

- **Aplicaciones web con JavaScript**
- Dependiendo de cómo se usa el **JavaScript** se diferencian tres tipos:
  - **Aplicación web interactiva**
  - **Aplicación web con AJAX**
  - **Aplicación web SPA**

El desarrollo web hoy

code URJC

Página web

Servidor estático

- Página web estática
- Página web interactiva

Aplicación web

Servidor dinámico

- Aplicación web con cliente estático
- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA

El desarrollo web hoy

code URJC

- **Aplicación web interactiva**
  - El JavaScript se utiliza para crear **efectos gráficos (como las páginas web interactivas)**
  - También se utiliza para **validaciones** de datos en **formularios**
  - La **gran mayoría** de las aplicaciones web disponibles en Internet son de este tipo
  - Suelen usar **jQuery**

Sign Up | Log In

create a new account

Your Email: test@test
Please use a valid email address.

Confirm Email:

Country: United States

Zip Code: 123
Please provide a valid Zip Code

Password: ●●●
Your password must be between 6-16 characters long.

Confirm Password:
This does not match the password entered above.

☐ Yes, I agree to the Terms of Use

Sign Up

Why you'll love it

- See all your accounts in one place
- Set a budget and pay down your debt
- Find the best ways to grow your money
- Stay safe and secure

El desarrollo web hoy

code URJC

Página web

Servidor estático

- Página web estática
- Página web interactiva

Aplicación web

Servidor dinámico

- Aplicación web con cliente estático
- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA

El desarrollo web hoy

code URJC

- **Aplicación web con AJAX**

  - **AJAX (*Asynchronous JavaScript And XML*)**

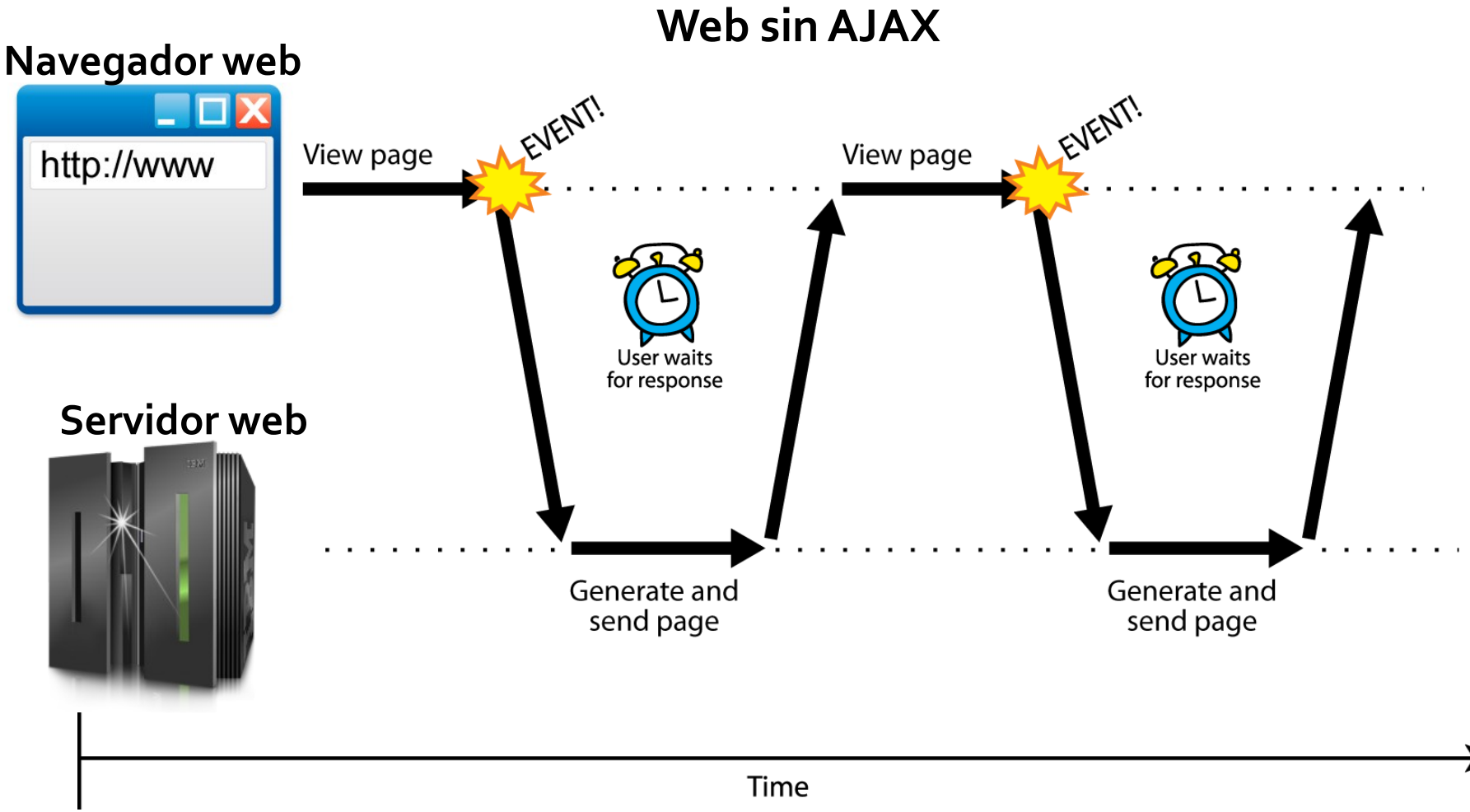
    - JavaScript se usa para no tener que **recargar completamente** la página al pulsar un link
    - Permite hacer petición al servidor web en **segundo plano** (oculta al usuario)
    - Cuando llega al navegador el **resultado de la petición**, el código JavaScript **actualiza** aquellas **partes** de la **página** necesarias

El desarrollo web hoy

code URJC

- Aplicación web con AJAX

AJAX
Asynchronous Javascript And XML

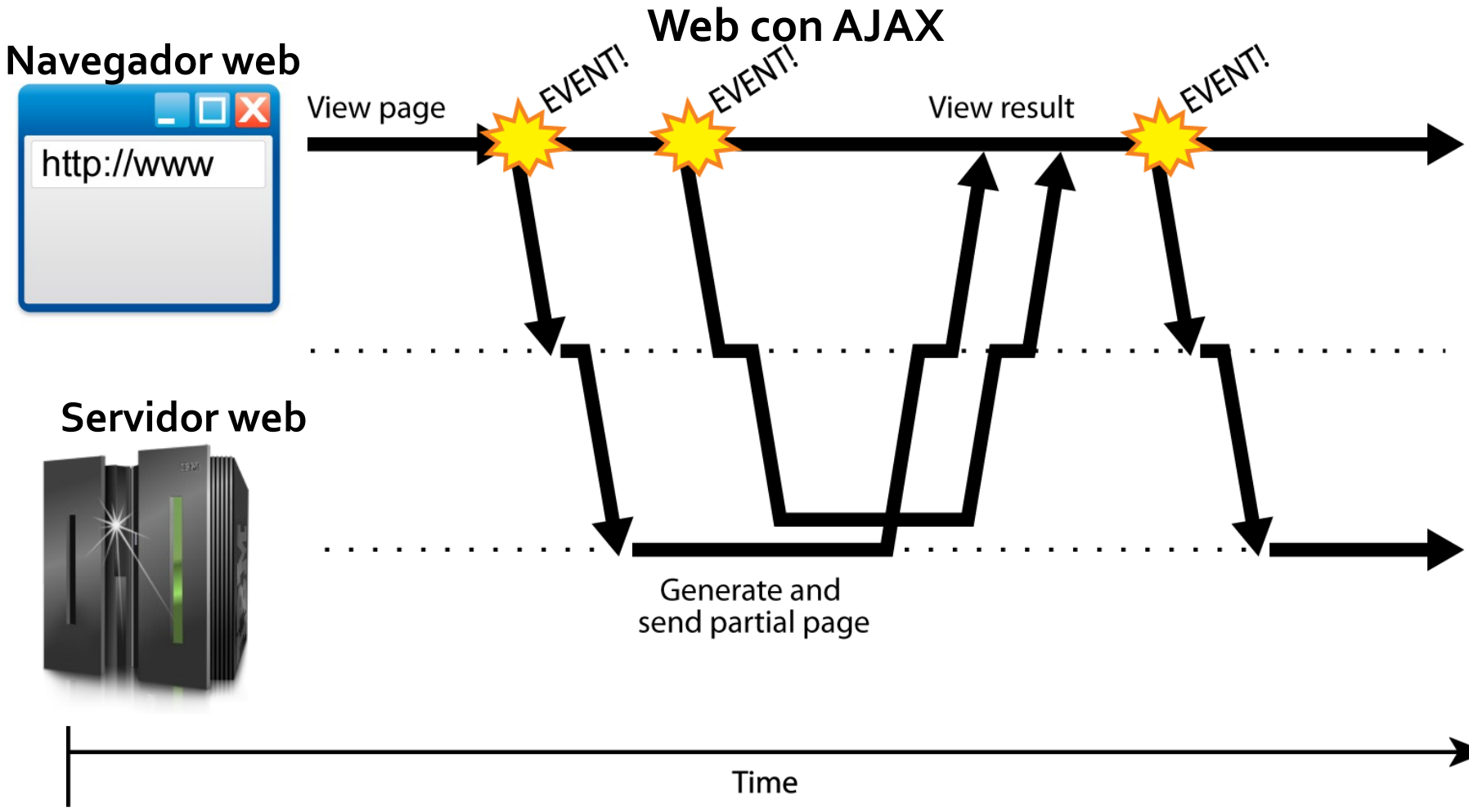


El desarrollo web hoy

code URJC

- Aplicación web con AJAX

AJAX
Asynchronous Javascript And XML



El desarrollo web hoy

code URJC

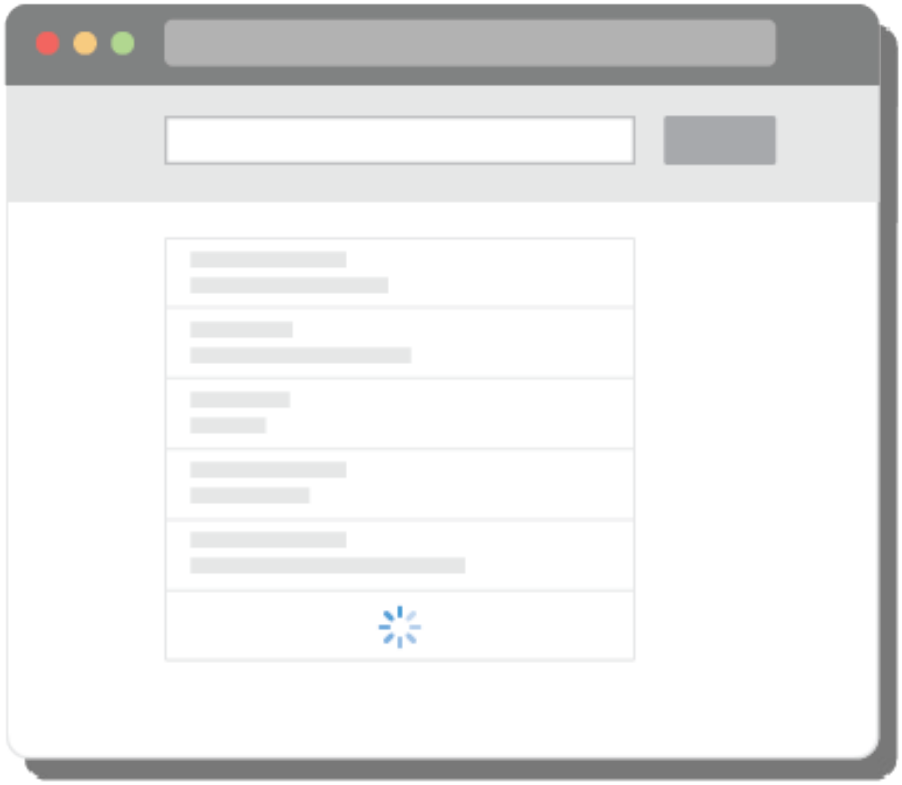
- **Aplicación web con AJAX**

  - Usar **AJAX** en una página **mejora** mucho la **experiencia de usuario**
  - No es necesario **recargar la página al completo**, sólo aquellas partes que cambian
  - Se suelen implementar con **jQuery**
  - Se puede usar para simular el efecto de **scroll infinito** y evitar los botones de anterior / siguiente

El desarrollo web hoy

code URJC

AJAX
Asynchronous Javascript And XML



Google images

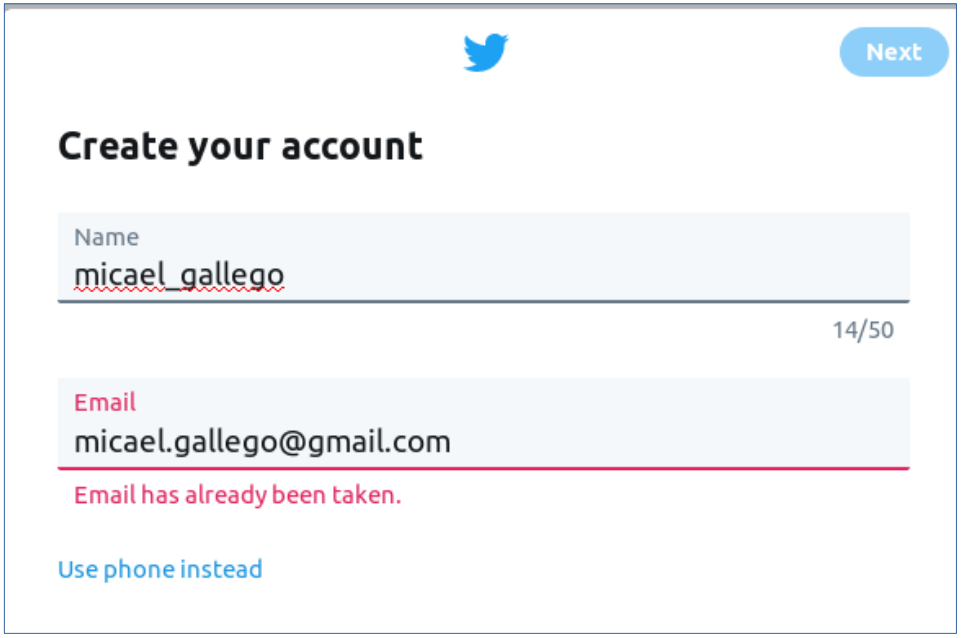
El desarrollo web hoy

code URJC

- **Aplicación web con AJAX**

AJAX Asynchronous Javascript And XML

  - Se puede dar **realimentación** al usuario antes de enviar el formulario, por ejemplo en la selección del nombre de usuario o correo si tiene que ser único



Twitter

El desarrollo web hoy

code URJC

- **Aplicación web con AJAX**

  - Cuando el código **JavaScript** hace peticiones, el servidor puede devolver:

Fragmentos de HTML	**Información estructurada**
Se incrusta directamente en la página	Se interpreta por JavaScript para modificar la página
Scroll infinito	Error de validación

El desarrollo web hoy

code URJC

- **Aplicación web con AJAX**

  - Cuando un servidor web genera **información estructurada** ante peticiones http se tiene una API REST
  - La información se representa en formato **JSON**

```
{
  "form_validation": [
    { "id": "name", "status": "ok" },
    { "id": "email", "status": "error", "message": "Invalid format" }
  ]
}
```

El desarrollo web hoy

code URJC

Página web

Servidor estático

- Página web estática
- Página web interactiva

Aplicación web

Servidor dinámico

- Aplicación web con cliente estático
- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA

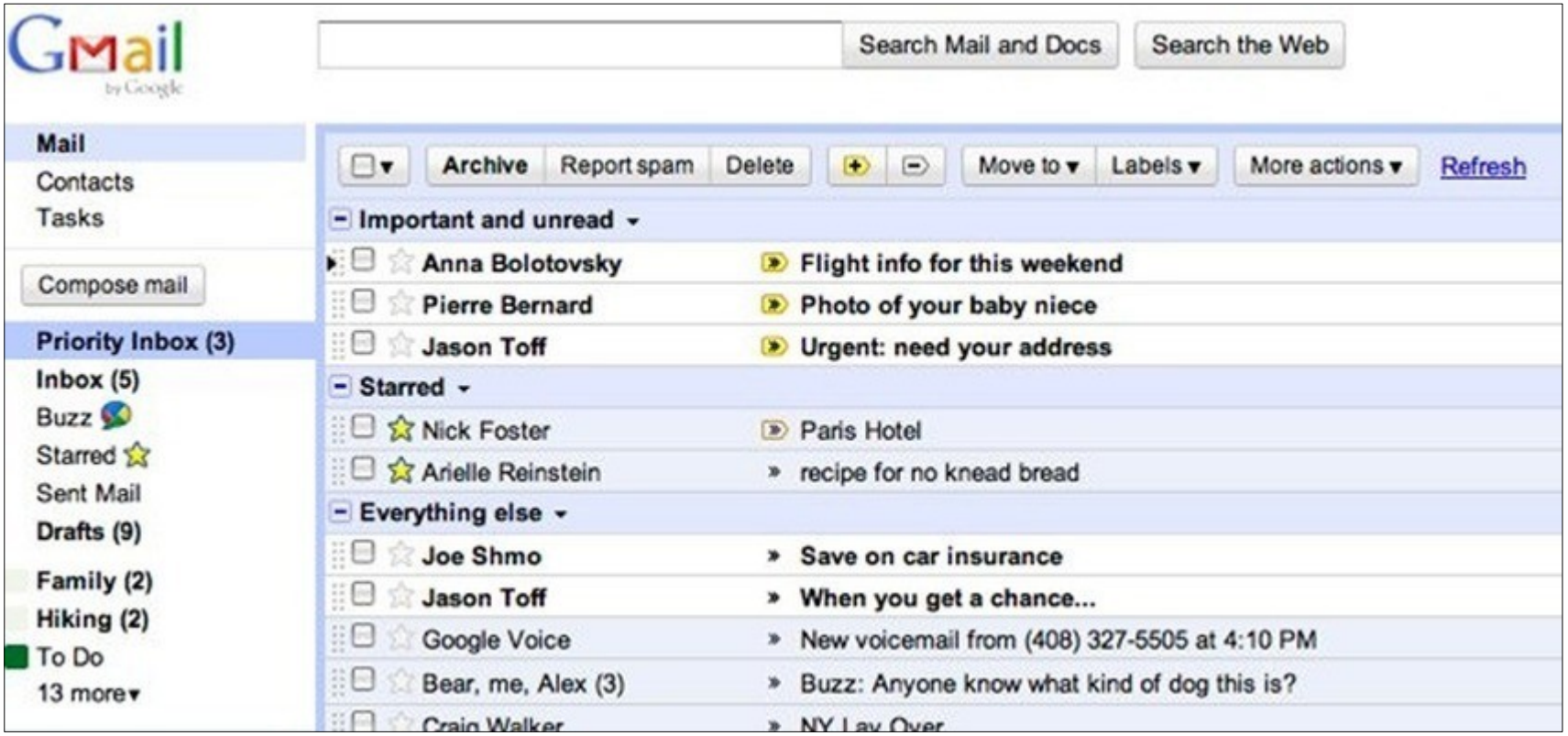
El desarrollo web hoy

code URJC

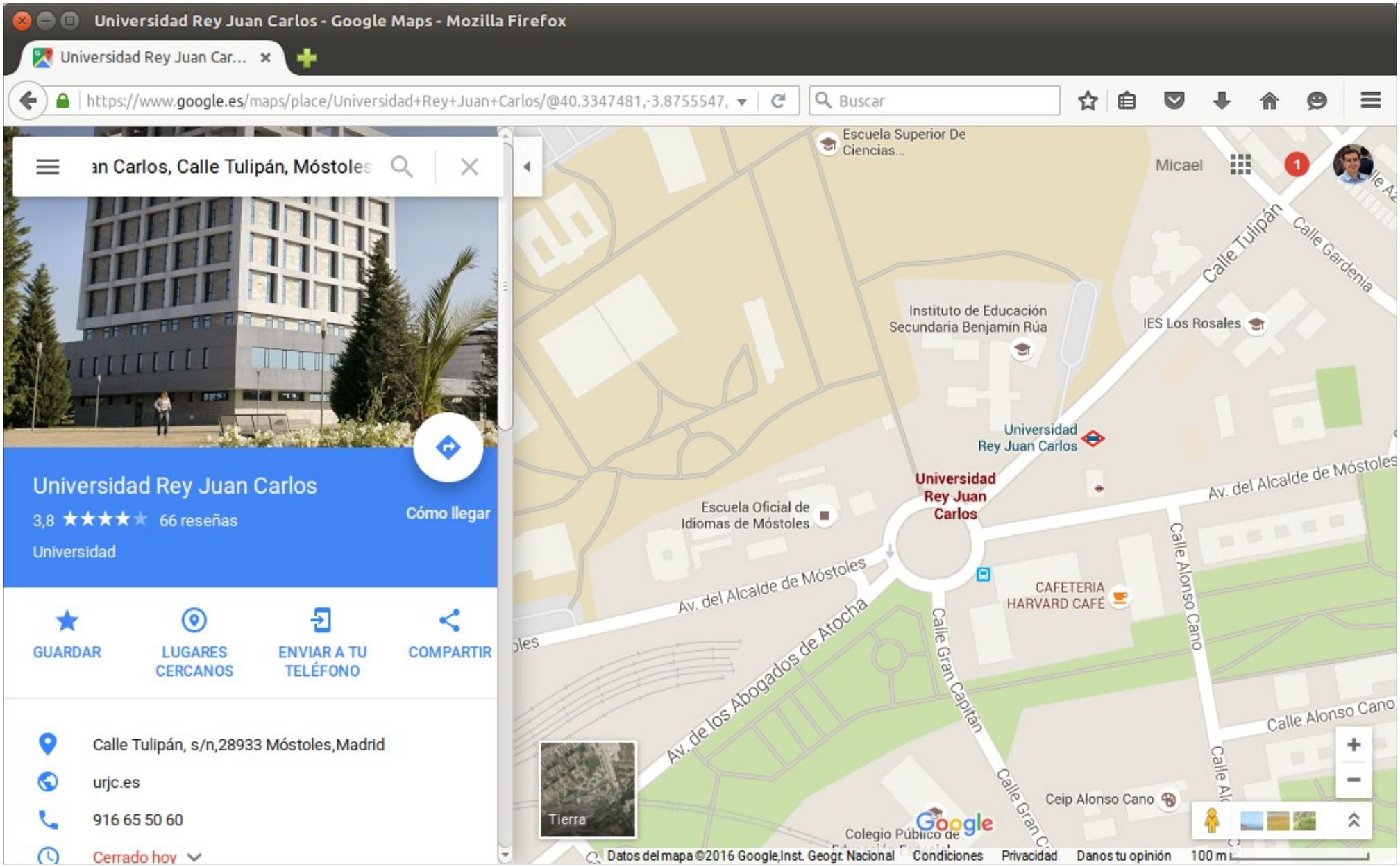
- **Aplicación web SPA**
  - **SPA (*Single Page Application*)**
  - La técnica **AJAX** se puede llevar al **extremo** y que toda la **información del servidor** sea obtenida con JavaScript en segundo plano, haciendo peticiones a la **API REST y obteniendo la información en JSON**
  - La **aplicación web** es un conjunto de **recursos HTML, CSS y JavaScript** que se cargan en el navegador al acceder a la URL principal

El desarrollo web hoy

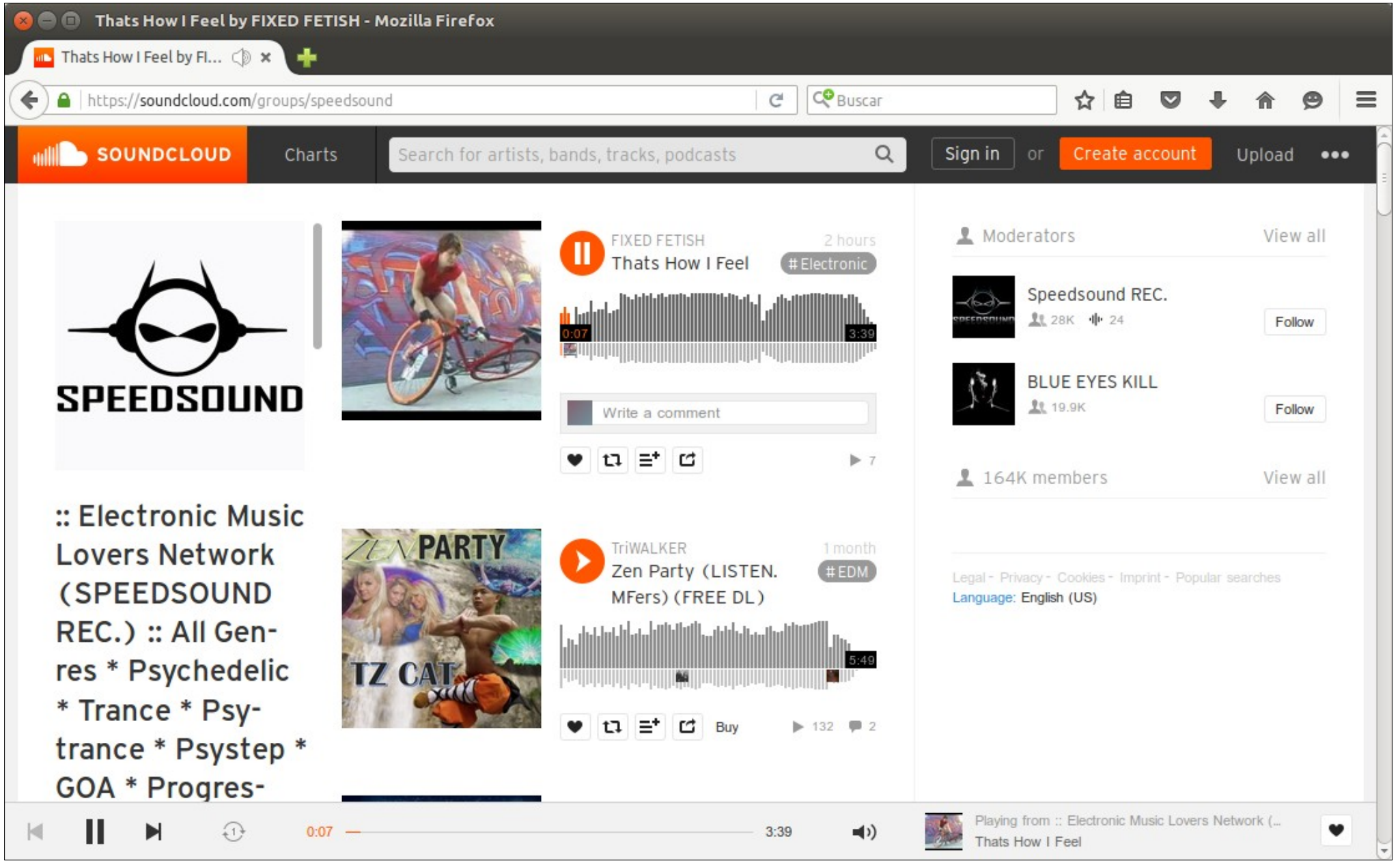
- **Aplicación web SPA**
  - Google las popularizó con **Gmail** y **Google Maps**



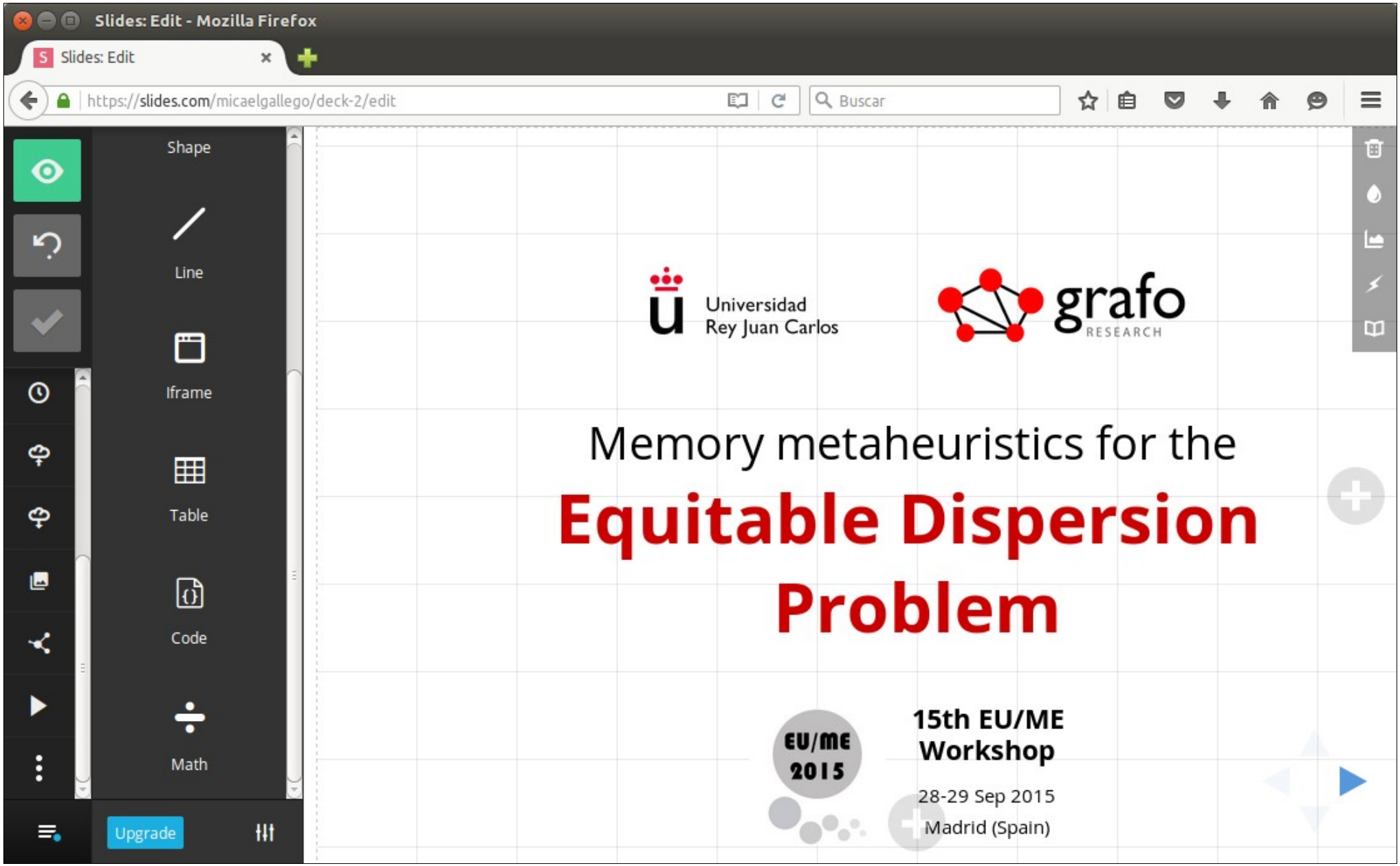
Google Maps


Universidad Rey Juan Carlos - Google Maps - Mozilla Firefox
Universidad Rey Juan Car...
https://www.google.es/maps/place/Universidad+Rey+Juan+Carlos/@40.3347481,-3.8755547,
Buscar
an Carlos, Calle Tulipán, Móstoles
Universidad Rey Juan Carlos
3,8 66 reseñas
Universidad
Cómo llegar
GUARDAR
LUGARES CERCANOS
ENVIAR A TU TELÉFONO
COMPARTIR
Calle Tulipán, s/n,28933 Móstoles,Madrid
urjc.es
916 65 50 60
Cerrado hoy
Escuela Superior De Ciencias...
Micael
Calle Tulipán
Calle Gardenia
Instituto de Educación Secundaria Benjamín Rúa
IES Los Rosales
Universidad Rey Juan Carlos
Universidad Rey Juan Carlos
Escuela Oficial de Idiomas de Móstoles
Av. del Alcalde de Móstoles
Av. de los Abogados de Atocha
Calle Gran Capitán
Calle Alonso Cano
CAFETERIA HARVARD CAFÉ
Ceip Alonso Cano
Colegio Público de
Tierra
Google
Datos del mapa ©2016 Google,Inst. Geogr. Nacional
Condiciones
Privacidad
Danos tu opinión
100 m

Soundcloud


Thats How I Feel by FIXED FETISH - Mozilla Firefox
Thats How I Feel by FI...
https://soundcloud.com/groups/speedsound
Buscar
SOUNDCLOUD
Charts
Search for artists, bands, tracks, podcasts
Sign in
or
Create account
Upload
SPEEDSOUND
:: Electronic Music Lovers Network (SPEEDSOUND REC.) :: All Genres * Psychedelic * Trance * Psytrance * Psystep * GOA * Progres-
FIXED FETISH
2 hours
Thats How I Feel
#Electronic
0:07
3:39
Write a comment
7
TriWALKER
1 month
Zen Party (LISTEN. MFers) (FREE DL)
#EDM
ZEN PARTY
TZ CAT
5:49
Buy
132
2
Moderators
View all
Speedsound REC.
28K
24
Follow
BLUE EYES KILL
19.9K
Follow
164K members
View all
Legal - Privacy - Cookies - Imprint - Popular searches
Language: English (US)
0:07
3:39
Playing from :: Electronic Music Lovers Network (...
Thats How I Feel

Slides.com


Slides: Edit - Mozilla Firefox
Slides: Edit
https://slides.com/micaelgallego/deck-2/edit
Buscar
Shape
Line
Iframe
Table
Code
Math
Upgrade
Universidad Rey Juan Carlos
grafo RESEARCH
Memory metaheuristics for the
Equitable Dispersion Problem
EU/ME 2015
15th EU/ME Workshop
28-29 Sep 2015
Madrid (Spain)

El desarrollo web hoy

code URJC

- **Aplicación web SPA**
  - En el lado cliente son **aplicaciones autónomas** que se comunican con el servidor con una **API REST o WebSockets**
  - Son las más **complejas** de implementar
  - Ofrecen una **experiencia muy interactiva y dinámica** al usuario
  - Las **tecnologías** que se usan para su desarrollo son **completamente diferentes** al resto de páginas y aplicaciones web

Desarrollo de webs SPA

code URJC

Editores / IDEs

El desarrollo web avanzado tiene sus propias herramientas

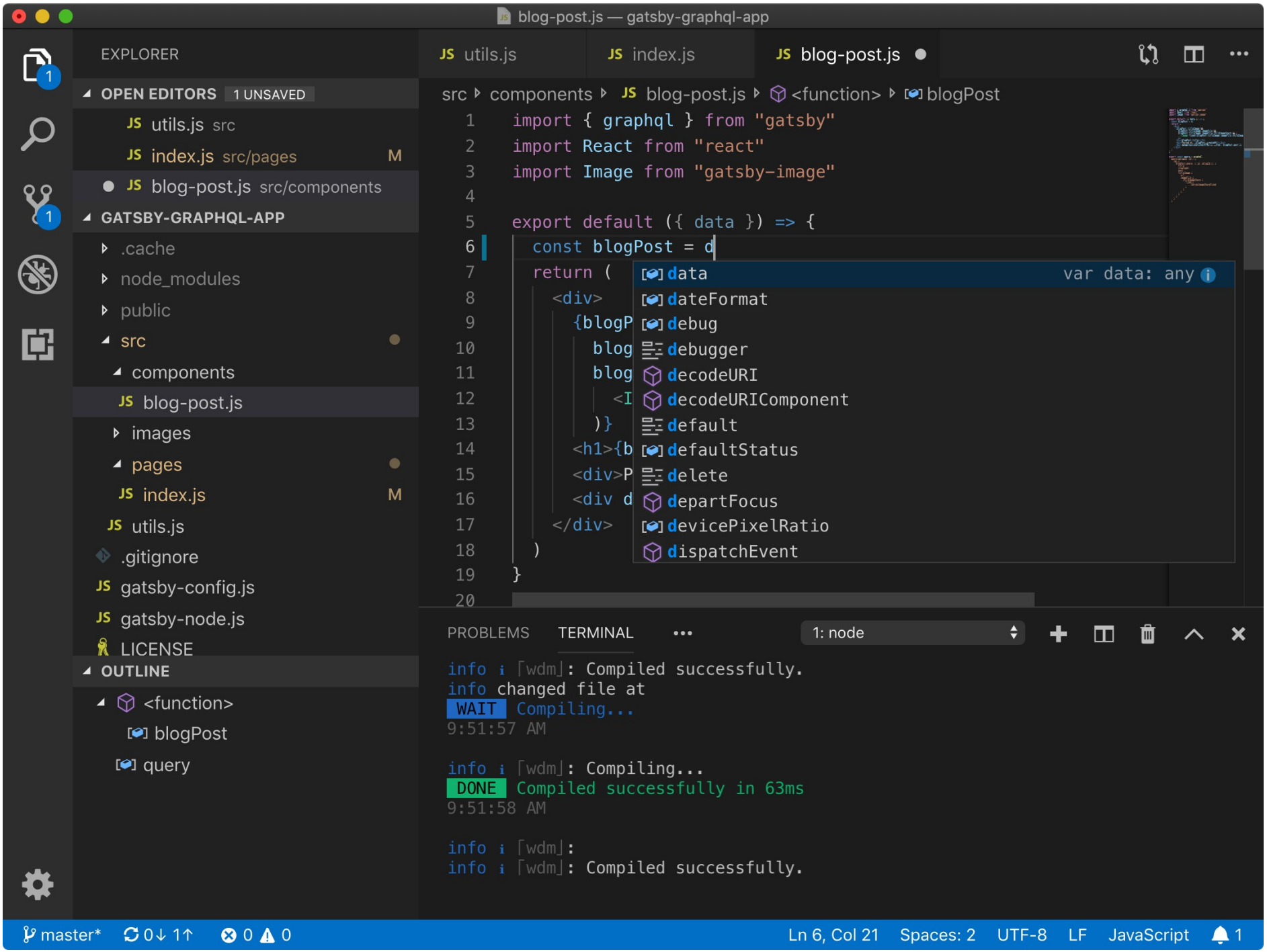
Sublime Text

Visual Studio Code

WebStorm

Brackets

Visual Studio Code



code URJC

Plataforma y gestión de paquetes

Plataforma para ejecutar aplicaciones JS fuera del navegador

Gestor de herramientas de desarrollo y librerías JavaScript

code URJC

Construcción de proyectos / empaquetado

webpack
MODULE BUNDLER

PARCEL
Blazing fast, zero configuration web application bundler

rollup.js

GRUNT

code URJC

Lenguaje programación

- Los navegadores web sólo entienden **JavaScript**
- La última versión de JavaScript (**ES6**) no está soportada por la mayoría de los navegadores
- Hay **traductores de ES6 a ES5** (versión soportada)
- Existen **lenguajes de programación** que se **traducen a JavaScript** y se ejecutan en los navegadores

code URJC

Lenguaje programación

code URJC

Lenguaje programación

code URJC

Ventajas frente a JavaScript

- Con el **tipado estático** el compilador puede verificar la **corrección** de muchas más cosas que con el tipado dinámico
- Los programas grandes son **menos propensos a errores**
- Los **IDEs** y **editores** pueden: Autocompletar, Refactorizar, Navegar a la definición
- **Muy parecido a Java y C#**

code URJC

Facilidad de adopción para JavaScripters

- Los tipos son **opcionales**
- La **inferencia de tipos** permite no tener que escribir los tipos constantemente
- En realidad es **JavaScript con más cosas**, así que todo lo conocido se puede aplicar
- Un mismo proyecto puede **combinar JS y TS**, lo que facilita migrar un proyecto existente

08-2023

code URJC

Fundamentos de la Web

Bloque I: Introducción a la web

Tema 1.3: Lenguajes de marcado

Universidad Rey Juan Carlos

Micael Gallego
Correo: micael.gallego@urjc.es
Twitter: @micael_gallego

Nicolás Rodríguez
Correo: nicolas.rodriguez@urjc.es

Iván Chicano
ivan.chicano@urjc.es
Michel Maes
michel.maes@urjc.es

08-2023

code
URJC

Micael Gallego, Nicolás Rodríguez, Iván Chicano, Michel Maes

Universidad
Rey Juan Carlos

XML

XML

- eXtensible Markup Lenguaje (XML) o Lenguaje de Marcado eXtensible
- Fue estandarizado por el consorcio WWW (W3C)
- Se actualiza y evoluciona constantemente
- XML es un lenguaje utilizado para el almacenamiento e intercambio de datos estructurados entre distintas plataformas.

XML

- Es un metalenguaje, se usa para definir otros lenguajes o dialectos XML
- Cada documento XML contiene uno o más elementos.
- Cada elemento viene delimitado por las etiquetas inicial y final
- Se basa en el marcado por etiquetas, de modo similar al HTML

XML

- Los **elementos** se pueden definir como bloques de construcción de un XML. Pueden comportarse como contenedores para texto, elementos, atributos, objetos de soporte o de todas ellas.

```
<elemento> Contenido del elemento </elemento>
```

XML

- Los **atributos** son parte de los elementos XML. Un elemento puede tener varios atributos únicos. Atributo proporciona más información acerca de elementos XML. Para ser más precisos, se definen las propiedades de los elementos. Un atributo XML es siempre un par de nombre-valor.

```
<ejemploElemento atributo="valor"> </ejemploElemento>
```

XML

- ¿Cuáles son los atributos?

```
<serie id="604">
    <name>Stranger things</name>
    <release year="2016"></release>
</serie>
```

- ¿Y los elementos?

Elemento XML

Elemento XML

- Los elementos se definen mediante etiquetas, cuyo nombre estará definido o podremos definir para adaptarlo a nuestras necesidades
- Esta es la síntesis básica que se usa para escribir un elemento en XML: <etiqueta> valor </etiqueta>
- Por ejemplo, si queremos guardar el personaje Galadriel:

```
<character> Galadriel </character>
```

Elemento XML

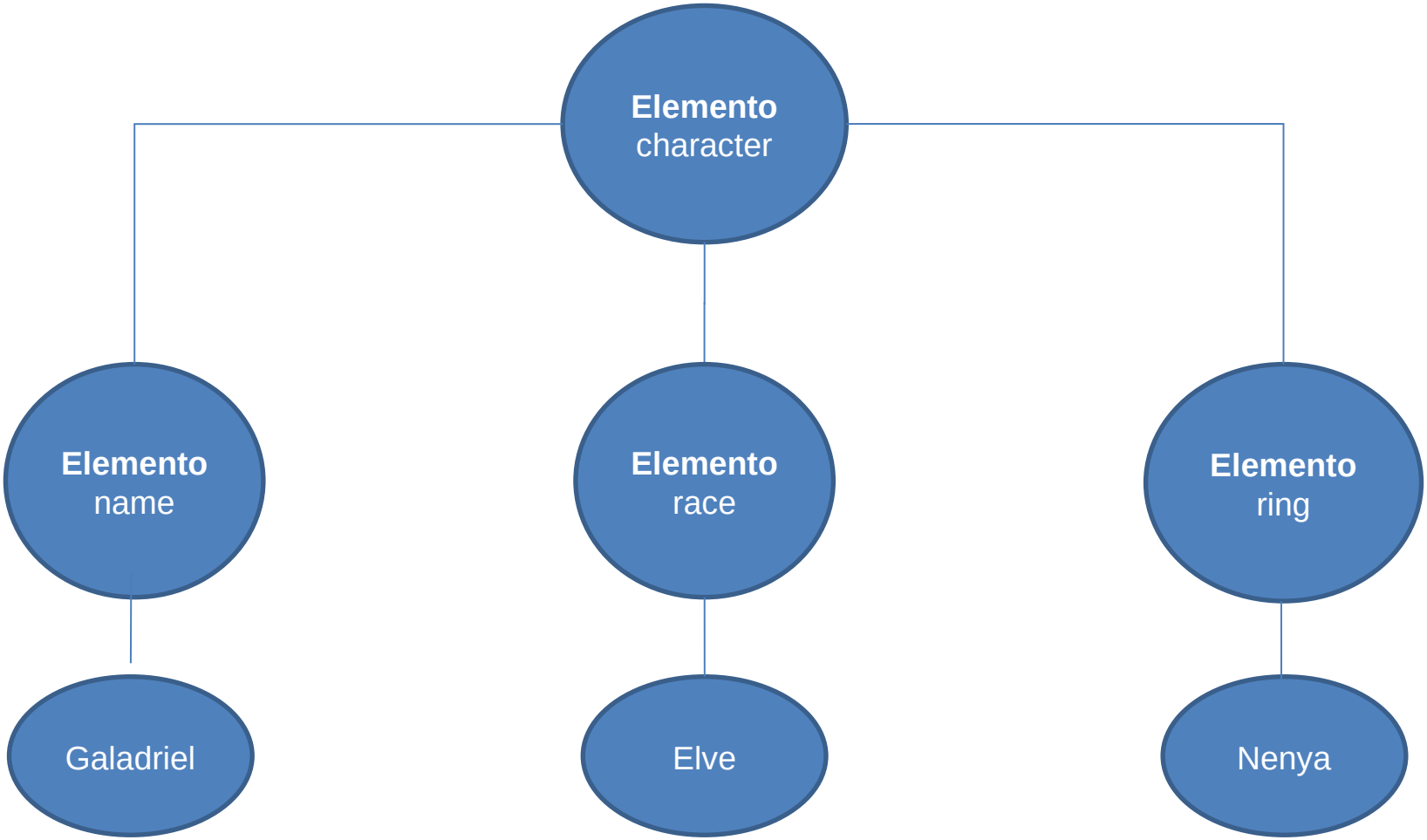
- El **nombre** del elemento va en una etiqueta, en este caso <`character`>.
- Se necesitan siempre y necesariamente dos etiquetas para definir un elemento: etiqueta de inicio y etiqueta de fin.
  - Nombre del elemento: *character*
  - Etiqueta de inicio: <`character`>
  - Etiqueta de fin: </`character`>
  - Contenido: *Galadriel*

Elemento XML

- Elementos vacíos. Si un elemento no puede contener ningún valor podemos escribir:
  - *<etiqueta></ etiqueta >*
  - *< etiqueta />*
  - Por ejemplo: *<character/>*
- Estructuras complejas:

```
<?xml version="1.0" encoding="UTF-8"?>
<character id=“01”>
        <name>Galadriel</name>
        <race>Elve</race>
        <ring>Nenya</ring>
</character>
```

Elemento XML



Elemento XML

- No está permitido:
  - El retorno de carro (intro) antes o después

```
<character>
      Galadriel </character>
```

  - Comenzar con un carácter distinto a letra o "_" (guión bajo)

```
<.character> Galadriel </character>
```

- Los caracteres usados en castellano están permitidos, pero se aconseja utilizar el código ASCII para ellos.
- Se aconseja evitar el "-" (guión) y el "." (punto)

Atributo XML

Atributo XML

- Se utilizan para proporcionar información sobre el elemento.
- Ejemplo. *Annie, conocida como the Dark Child es maga y tiene una velocidad de movimiento de 335. Su representación en un documento XML podría ser, por ejemplo:*

```
<champion>
        <speed>335</speed>
        <role>Mage</role>
        <name>Annie the Dark Child</name>
</champion>
```

Atributo XML

- No está permitido repetir nombres de atributos
- *Atributos especiales:*
  - *id: asigna un identificador único a un elemento*
  - *lang: especifica el idioma en el que se escribe (french, english, spanish,...)*

```
<article id ="a001" lang ="english">
```

Atributo XML

- Atributos *version y encoding*. Es la declaración que se indica al comienzo del documento y que comienza por "<?" y acaba por "?>"
- Version y encoding, ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
```

- Indicamos que la versión de XML utiliza es 1.0, y la codificación es la UTF-8 (que es la más ampliamente utilizada)

Atributo XML

- No es una declaración obligatoria en todos los documentos XML. Pero en caso de indicarse tiene que ir en la primera línea y sin espacios en blanco previos.
- Atributo *standalone*. Toma los valores "yes" o "no". Si su valor es "yes" indica que el documento depende de otros, como por ejemplo una DTD (Document Type Definition o Definición de Tipo de Documento) externa.

Atributo XML

- La declaración <?xml…. Es opcional, pero si se indica el atributo "version" es obligatorio, a diferencia de standalone y encoding.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

- El atributo *encoding* siempre aparecerá después de verion, y standalone siempre se define al final.
- *Comentarios en XML:*
  - *<!-- Etiqueta de inicio de comentario*
  - *--> Etiqueta de fin de comentario*

JSON

JSON

- Acrónimo de **JavaScript Object Notation**

  (extensión: .json)
- Es un formato ligero de texto para el intercambio de datos
- Independiente de lenguaje de programación
- Un fichero JSON, generalmente, será ofrecido por un servidor web y sus datos se presentarán en una página web, o como parte de una API

JSON

- Los programas necesitan enviar y recibir datos de manera sencilla, pero utilizando un formato común para estructuras complejas
- Se desea evitar construir parsers (analizadores) específicos cada vez que queremos intercambiar mensajes con el servidor
- XML es una opción válida, pero algunos no la consideran la más adecuada, por ser demasiado "pesada"

Representación en JSON

- **Value:** Puede ser un string, número, booleano, objeto u array
- **String:** Colección de cero o más caracteres Unicode (entre comillas)
- **Number:** Valor numérico (sin comillas)
- **Object:** Conjunto desordenado de pares nombre/valor
- **Array:** Colección ordenada de valores

Representación en JSON

- Los descritos: boolean, string, number, null
- Estrictamente codificación Unicode
- Por defecto, UTF-8 (se admite también UTF-16 y UTF-32)
- Un par de la siguiente forma: “nombre”:“valor”
  - `"city": "Móstoles",`

Representación en JSON

Objeto en JSON

- Conjuntos de datos separados por comas y entre llaves:

Representación en JSON

Array en JSON

- Un array se escribe entre corchetes, separando por comas

```
{
    "planet": "Tatooine",
    "population": 200000,
    "species": ["tusken", "hutt"]
}
```

JSON vs XML

code URJC

- Es más sencillo construir un parser de JSON que de XML
- En JavaScript, un texto JSON se puede analizar usando directamente la función eval(), ya que es sintaxis JS
- Este hecho ha facilitado la aceptación mayoritaria de JSON por los desarrolladores AJAX (ubicuidad de JavaScript en navegadores web)
- Cualquier librería JS (e.g. jQuery) tiene capacidad de hacerlo también

08-2023

code URJC

Fundamentos de la Web

Bloque II: Tecnologías de cliente web

Tema 2.1: HTML

Universidad Rey Juan Carlos

Micael Gallego
Correo: micael.gallego@urjc.es
Twitter: @micael_gallego

Nicolás Rodríguez
Correo: nicolas.rodriguez@urjc.es

Iván Chicano
ivan.chicano@urjc.es

Michel Maes
michel.maes@urjc.es

08-2023

code
URJC

Micael Gallego, Nicolás Rodríguez, Iván Chicano, Michel Maes

Definición

code URJC

- **Hiper Text Markup Language (HTML)**
- Es el lenguaje utilizado para crear documentos en la web
- El **navegador web** es el encargado de visualizar los documentos HTML
- Cuando un **navegador web** realiza una petición **http** a un **servidor web**, el servidor le suele devolver un documento en **HTML**

Definición

code URJC

- Este formato de documentos está muy **relacionado** con:
  - El lenguaje **CSS (Cascade style sheet)** que permite añadir estilos
  - El lenguaje **JavaScript** que permite añadir interactividad en el cliente

Definición

Documento HTML

```html
<!DOCTYPE html>
<html>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>

<p>This paragraph one.</p>
<p>This paragraph two.</p>
<p>Bla, bla, bla, bla, bla, bla, bla,
bla, bla, bla, bla, bla, bla, bla, bla,
bla, bla, bla, bla, bla, bla, bla, bla,
bla, bla, bla, bla, bla, bla, bla, bla,
bla, bla, bla, bla, bla, bla.</p>

</body>
</html>
```

Visualización en el navegador

This is heading 1

This is heading 2

This is heading 3

This is heading 4

This is heading 5

This is heading 6

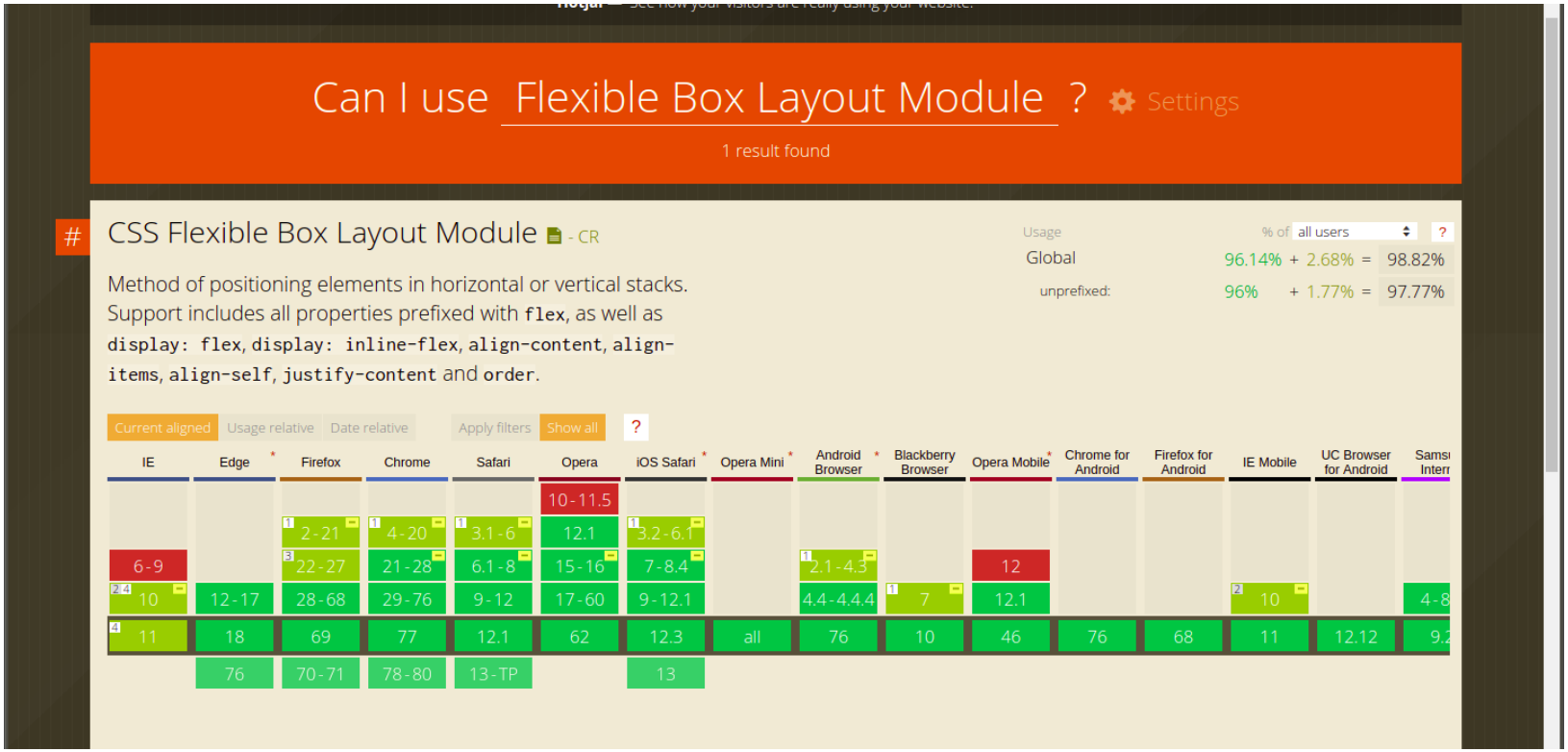
This paragraph one.

This paragraph two.

Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla.

Definición

- Soporte de funcionalidades en diferentes navegadores



https://caniuse.com/

XHTML

- **XHTML**
  - Es una variante de HTML que permite escribir documentos HTML pero como un documento válido XML
  - Un documento **XHTML** es un documento XML válido
  - Por lo general, un documento **HTML** no es un documento XML válido
  - Aunque tuvo cierta aceptación en el pasado, actualmente **no es muy usado** para escribir documentos web

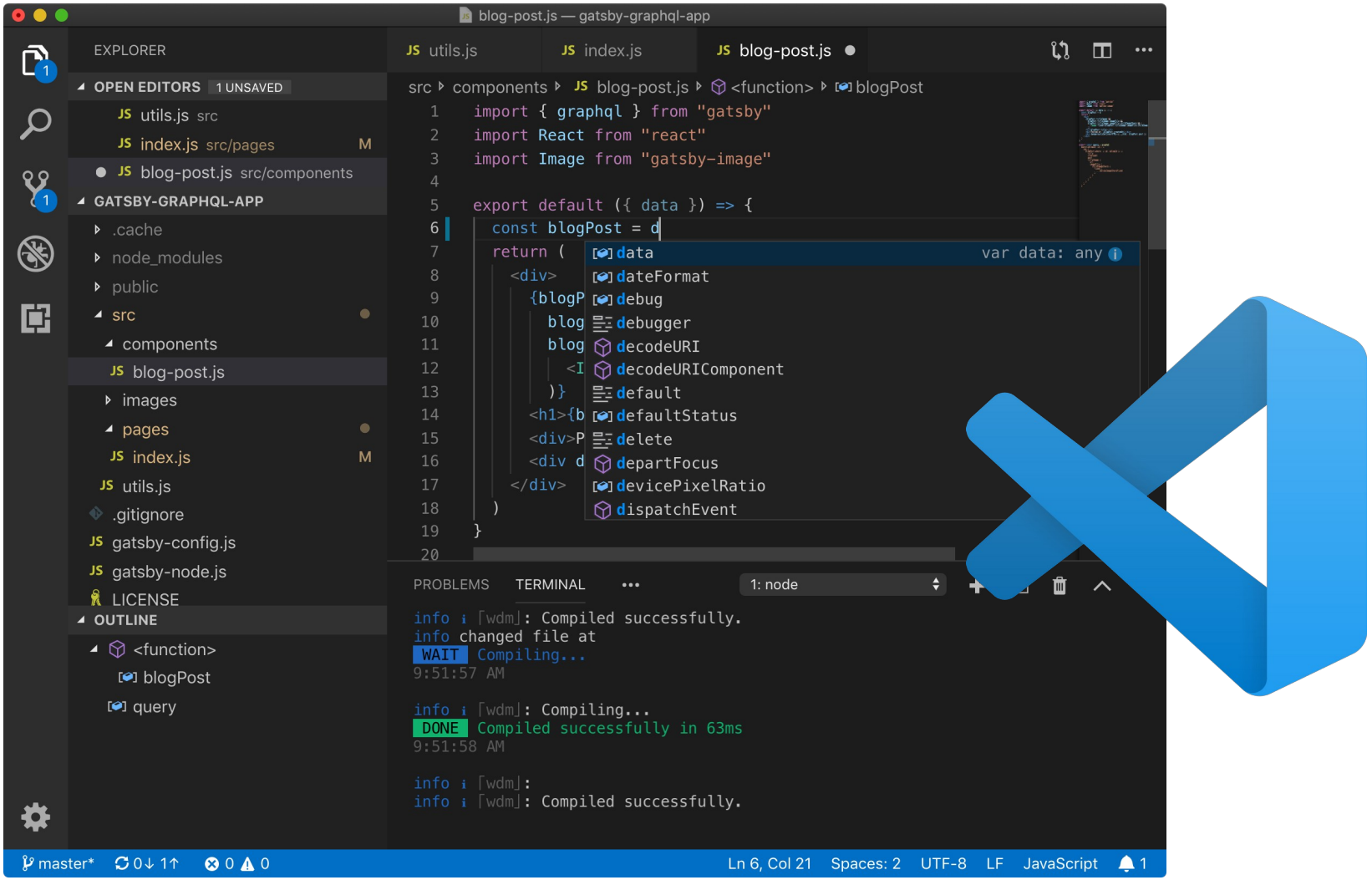
Editores

code URJC

- **Edición de un documento HTML**
  - Con un editor de textos "vitaminado":
    - Sublime Text, Atom.io, Brackets, **Visual Studio Code**
  - Con un IDE especializado en desarrollo web:
    - Netbeans, Eclipse, Webstorm
  - Con herramientas integradas en el propio navegador:
    - Chorme Developer Tools, Firefox Developer Edition
  - Con páginas web para experimentar:
    - http://jsbin.com/ http://jsfiddle.net/ http://plnkr.co/

Editores

code URJC



https://code.visualstudio.com/

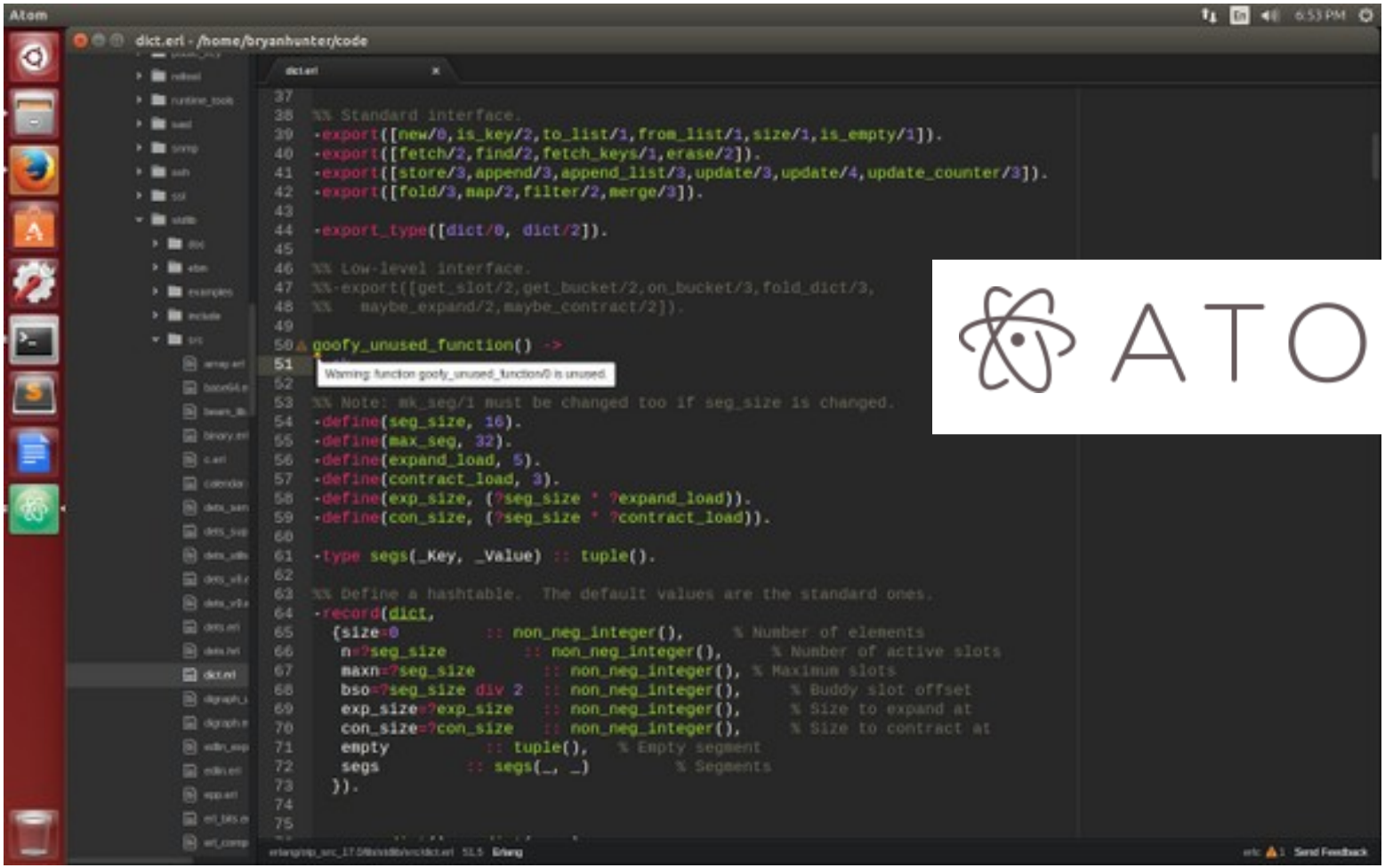
Editores

code URJC

http://www.sublimetext.com/

Editores

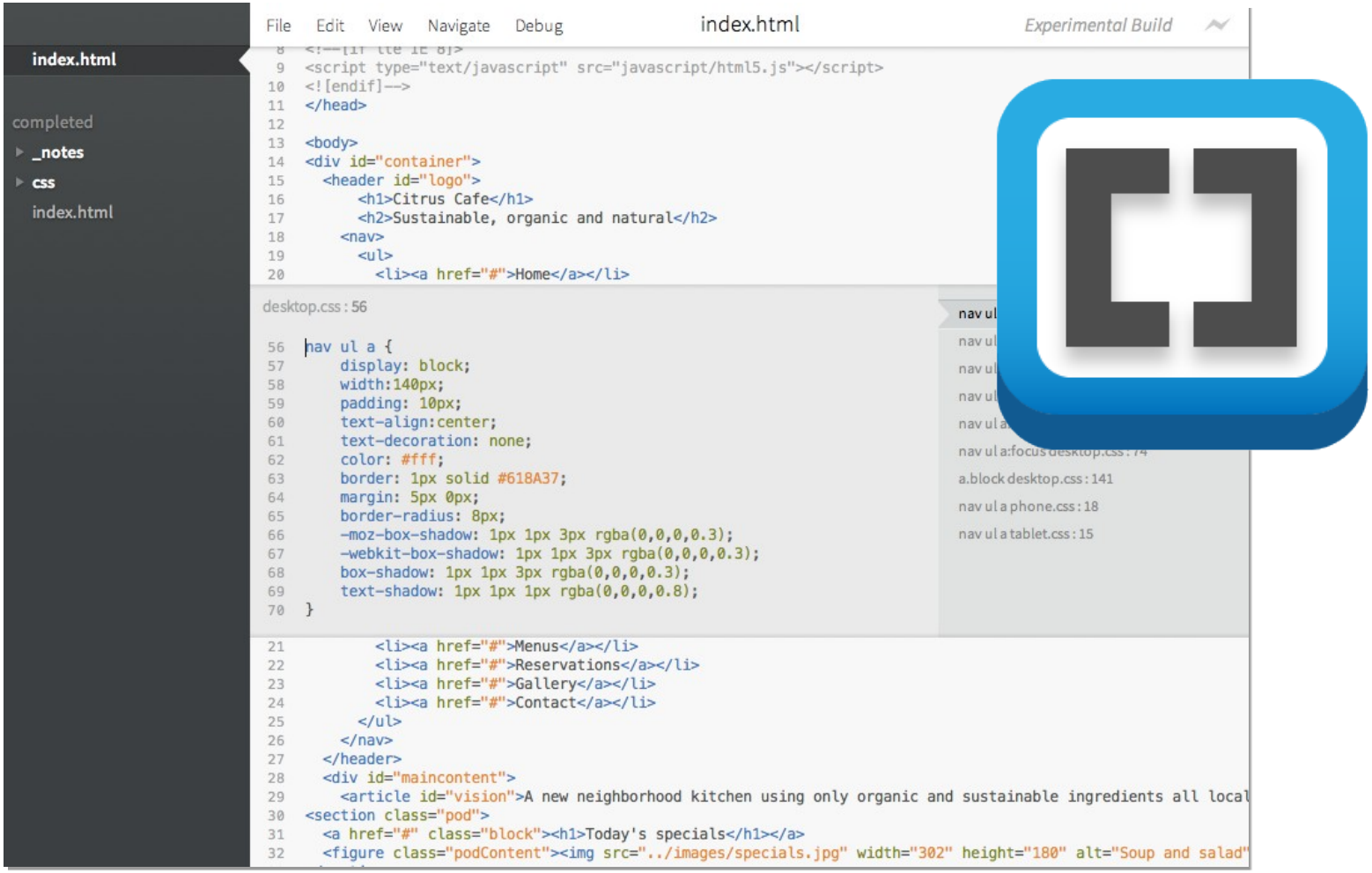
code URJC



https://atom.io/

Editor

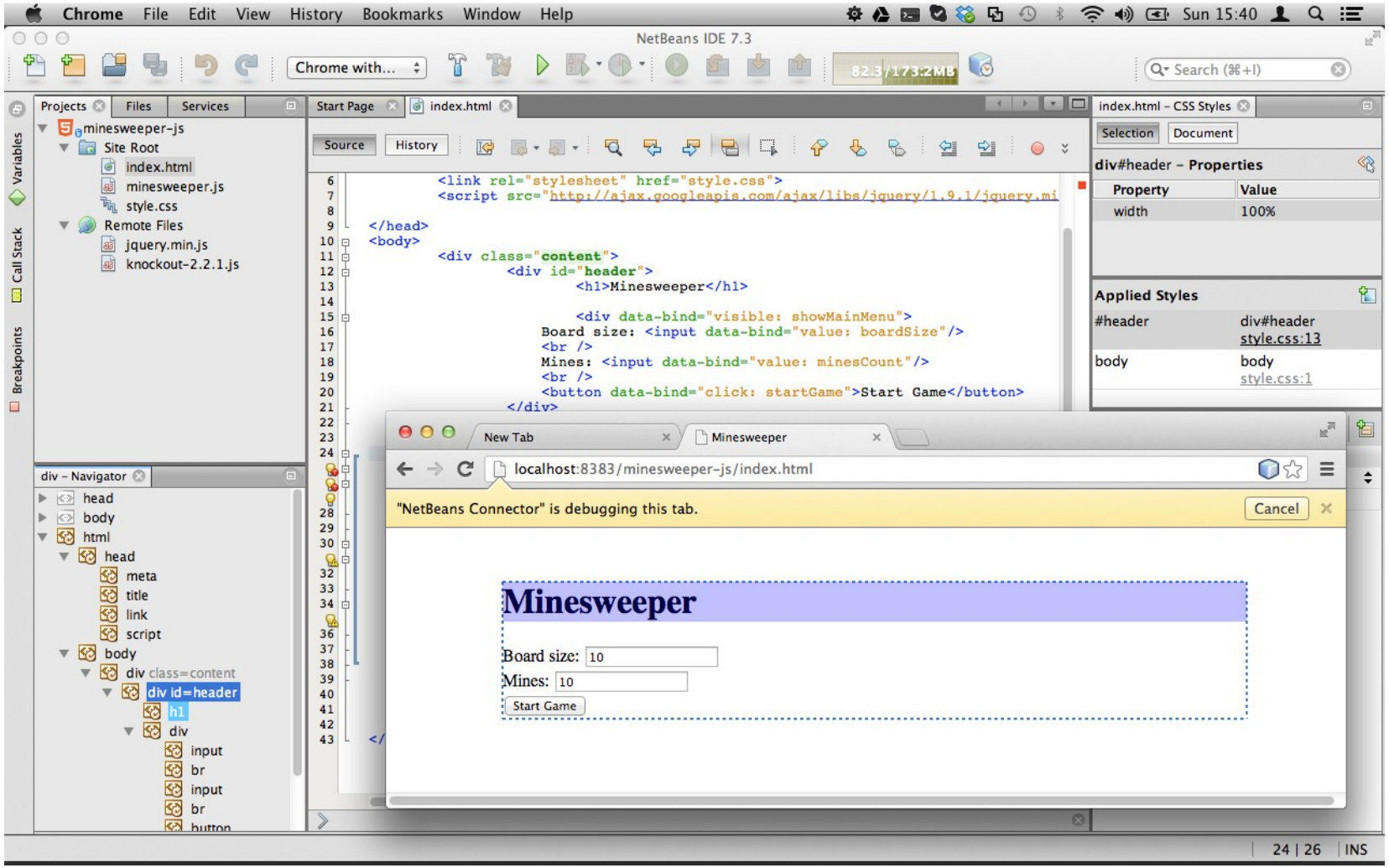
code URJC



http://brackets.io

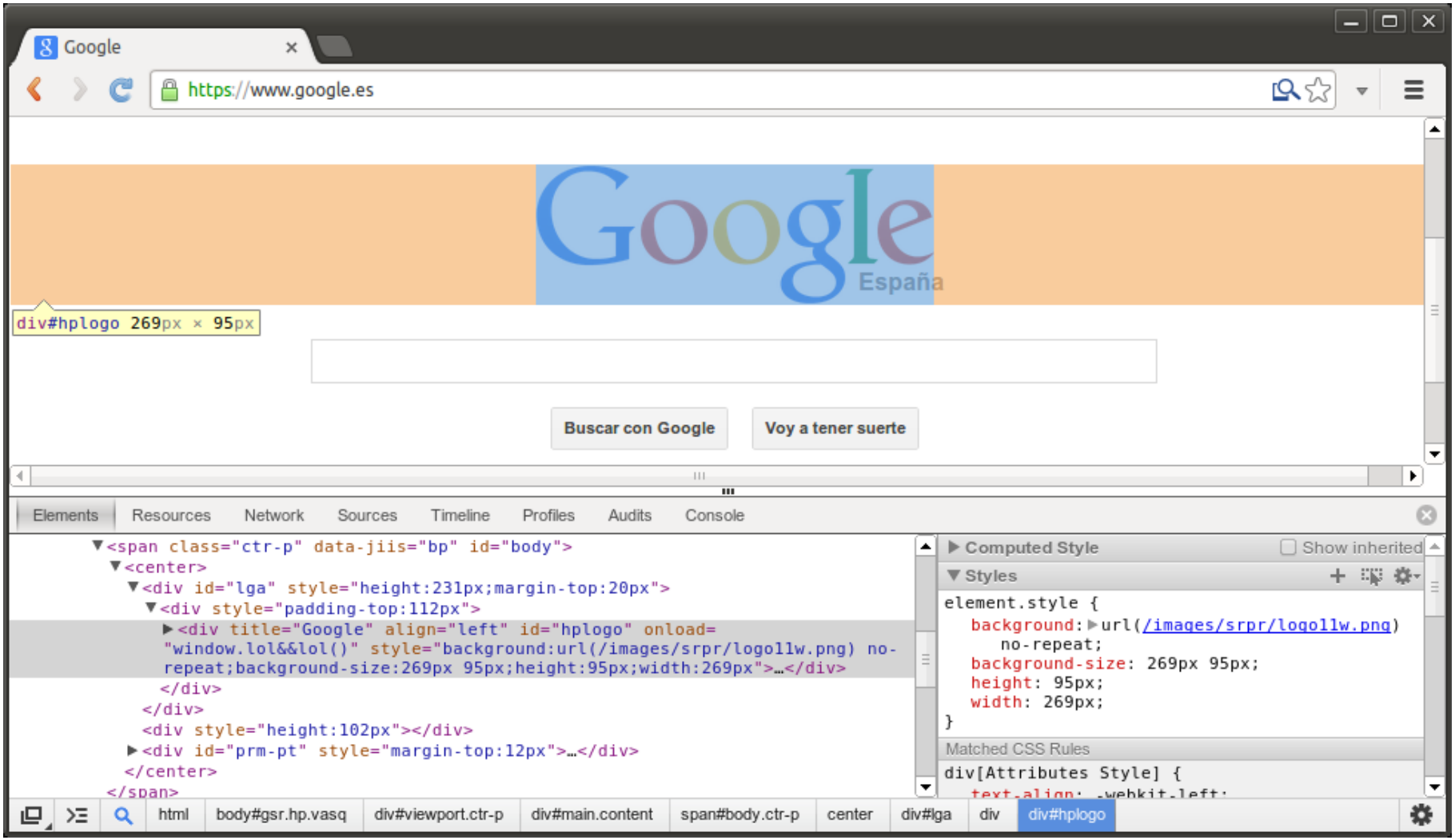
Editor

code URJC



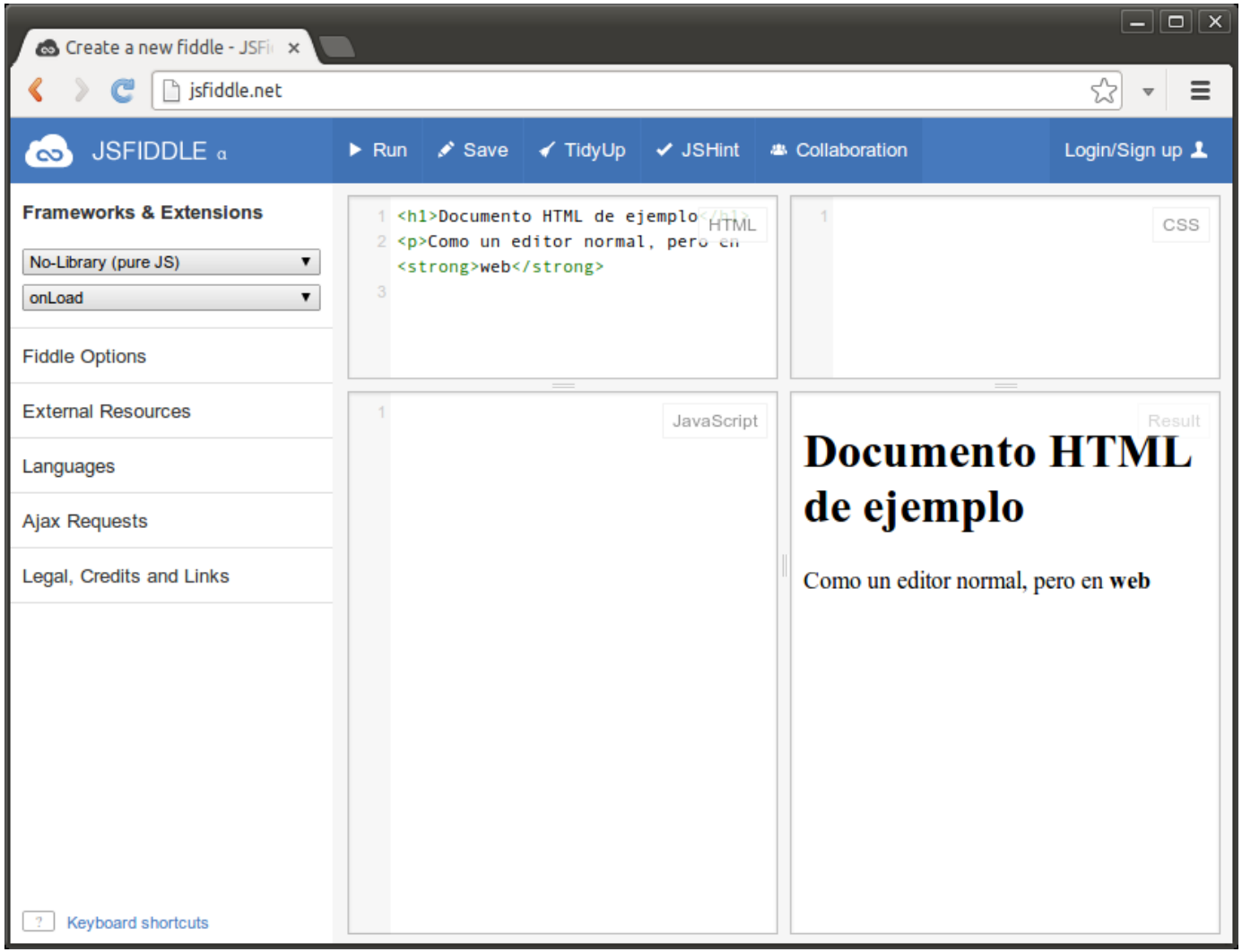
Editor

code URJC



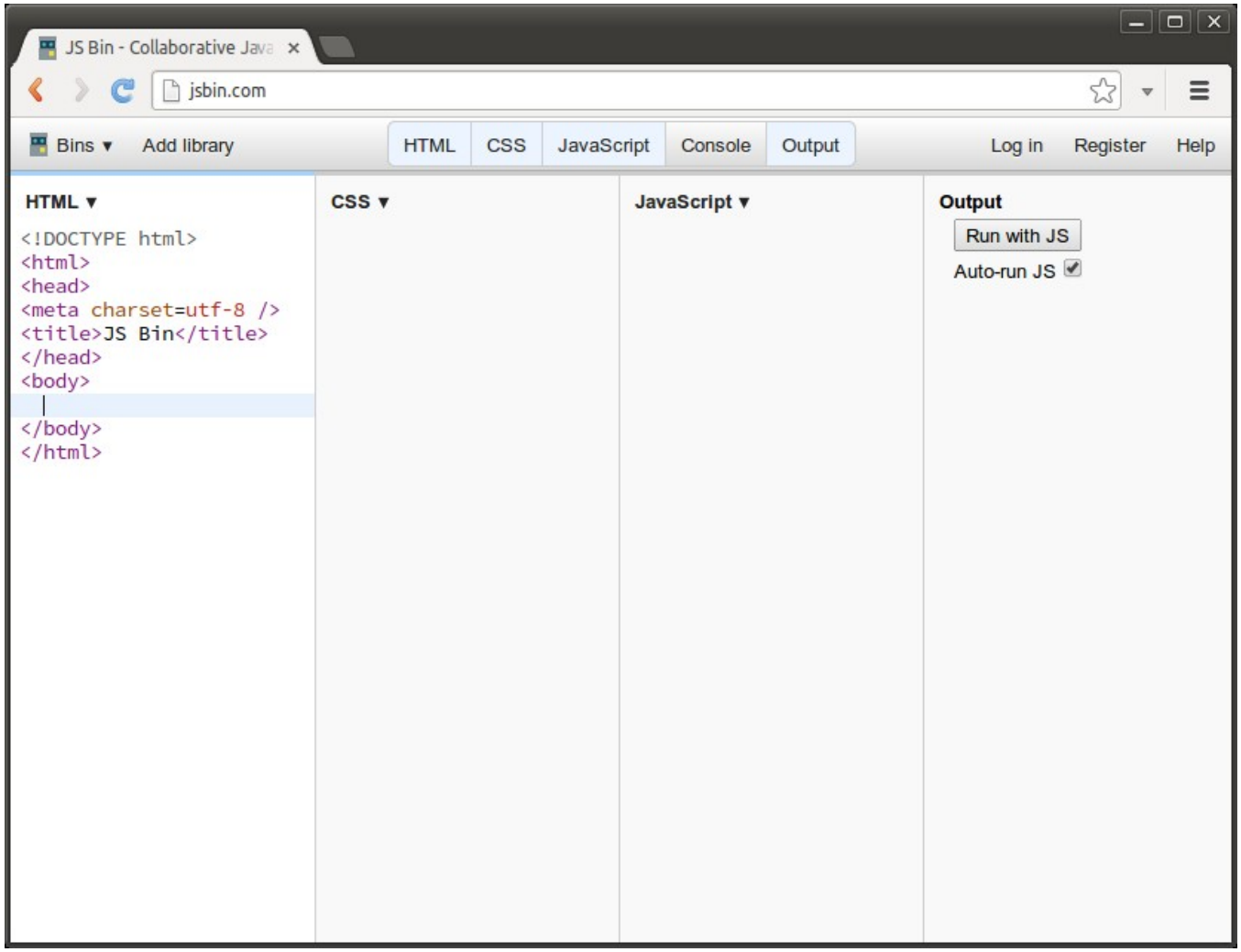
Editor

code URJC



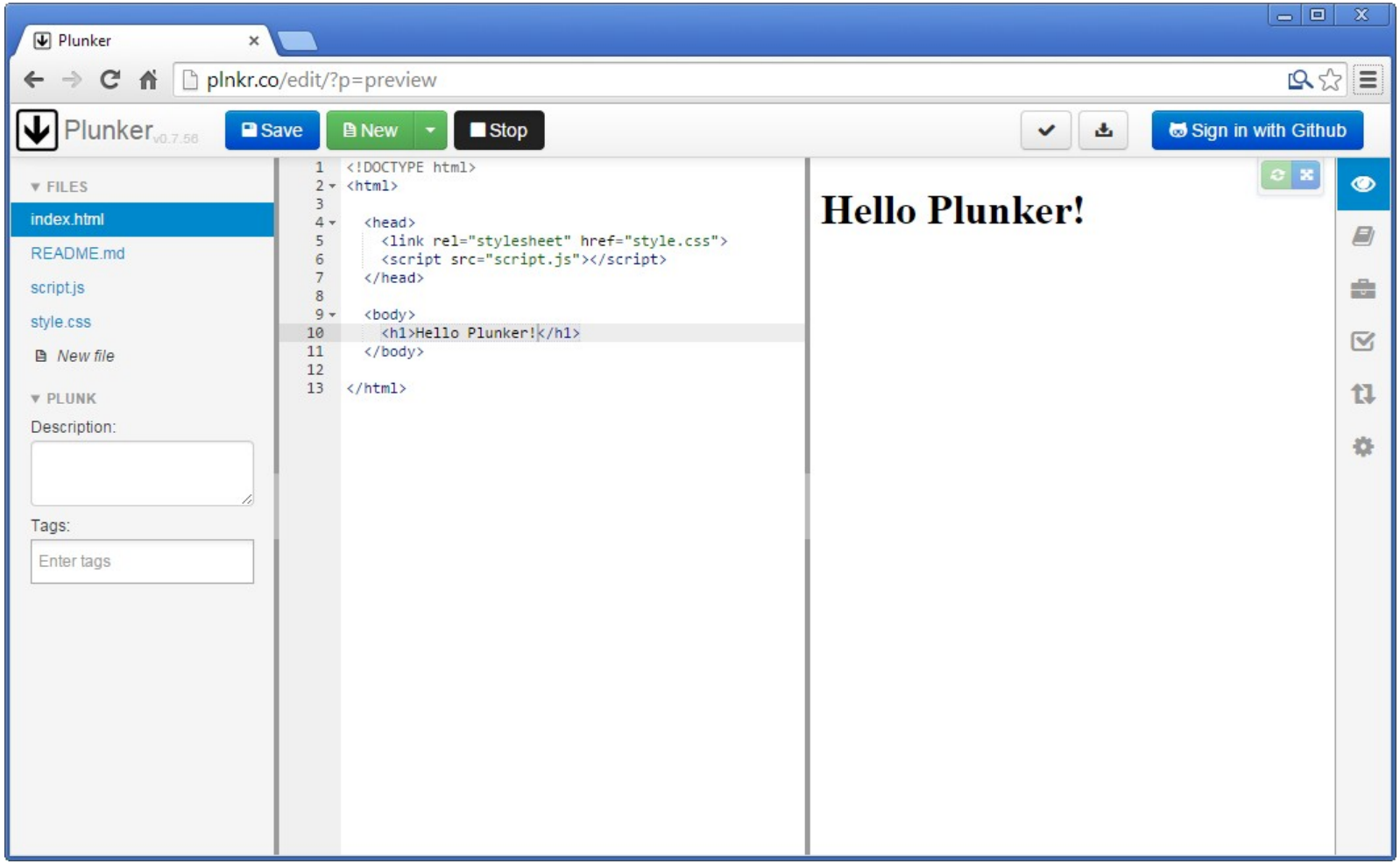
Editor

code URJC



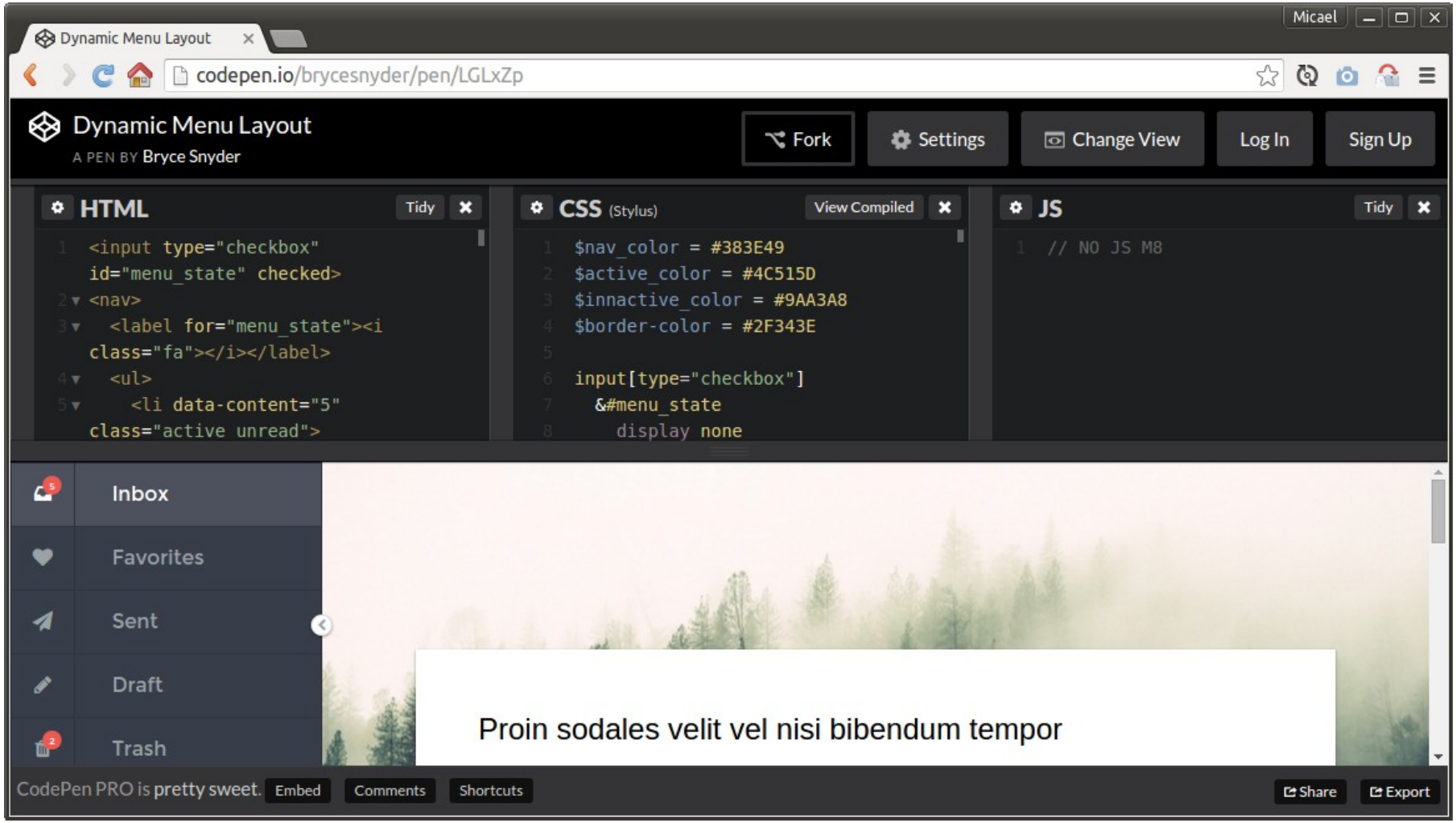
Editor

code URJC



Editor

code URJC



HTML

code URJC

- Estructura básica de un documento HTML

```
<!DOCTYPE html>
<html>
   <head>
      <title>Título de la página</title>
   </head>
   <body>
      <p>Esto es un texto <strong>resaltado</strong> y
<em>enfatizado</em></p>
   </body>
</html>
```

Se puede guardar en el disco con extensión .html

HTML

- Estructura básica de un documento HTML

```
<!DOCTYPE html>
<html>
   <head>
      <title>Título de la página</title>
   </head>
   <body>
      <p>Esto es un texto <strong>resaltado</strong> y
<em>enfatizado</em></p>
   </body>
</html>
```

Cabecera
No forma parte del contenido del documento. Título y otra información de alto nivel

Cuerpo
Contenido del documento. Se visualiza en el navegador

Se puede guardar en el disco con extensión .html

HTML

Elemento

Etiquetas HTML

- Un documento está formado por etiquetas entre ángulos (**<html>**)
- La mayoría de ellas viene en pares:
  - Etiqueta de apertura: **<h1>**
  - Etiqueta de cierre: **</h1>**
- Entre las etiquetas de apertura y de cierre se puede incluir texto que será visualizado en el navegador

```
<p>Esto es un texto
<strong>resaltado</strong>
y <em>enfatizado</em></p>
```

Esto es un texto **resaltado** y *enfatizado*

Etiquetas HTML

- **Título**
  - **<h1>…</h1>**
  - Se pueden poner hasta 6 niveles de importancia
- **Párrafos**
  - ** **Esto es un párrafo** **
- **Secciones**
  - Las secciones se usan para indicar contenidos relacionados entre sí y que tienen un título
  - Siempre deben llevar título **<h1>**

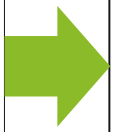
Tipos de elementos

- **Bloque: div, header, footer, section, aside, h, p, ul, li …**
  - Son elementos que comienzan en una línea nueva
  - Tratan de ocupar todo el espacio disponible
  - Pueden anidarse tantos bloques de cualquier tipo como sea necesario.

- **Texto: a, strong, em, small, del, mark, span, …**
  - Son elementos que continúan el flujo normal del texto
  - Ocupan el espacio necesario de su contenido
  - No aceptan elementos de bloque en su interior

Títulos, párrafos y secciones

```
<section>
   <h1>Postishead</h1>
   <p>Portishead are a cool band</p>
   <section>
      <h1>Dummy (album)</h1>
      <p>some info....</p>
      <p>some info....</p>
   </section>
   <section>
      <h1>Portishead (album)</h1>
      <p>some other info info....</p>
      <p>some info....</p>
   </section>
</section>
```



Postishead

Portishead are a cool band from Bristol

Dummy (album)

some info....

some info....

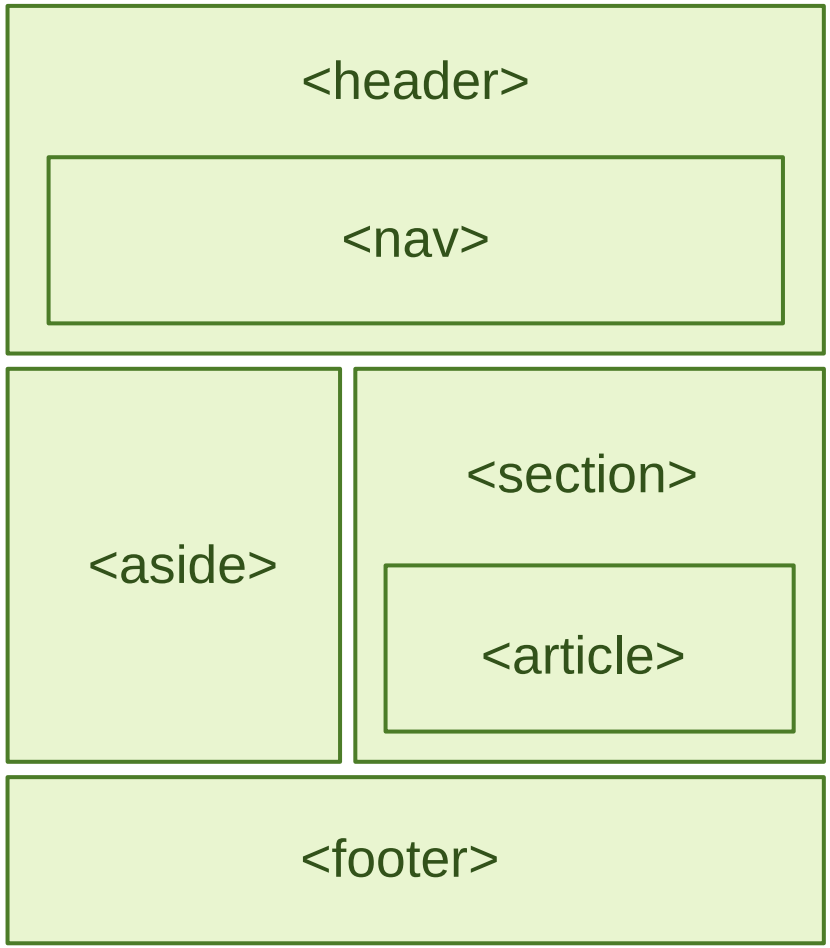
Portishead (album)

some other info info....

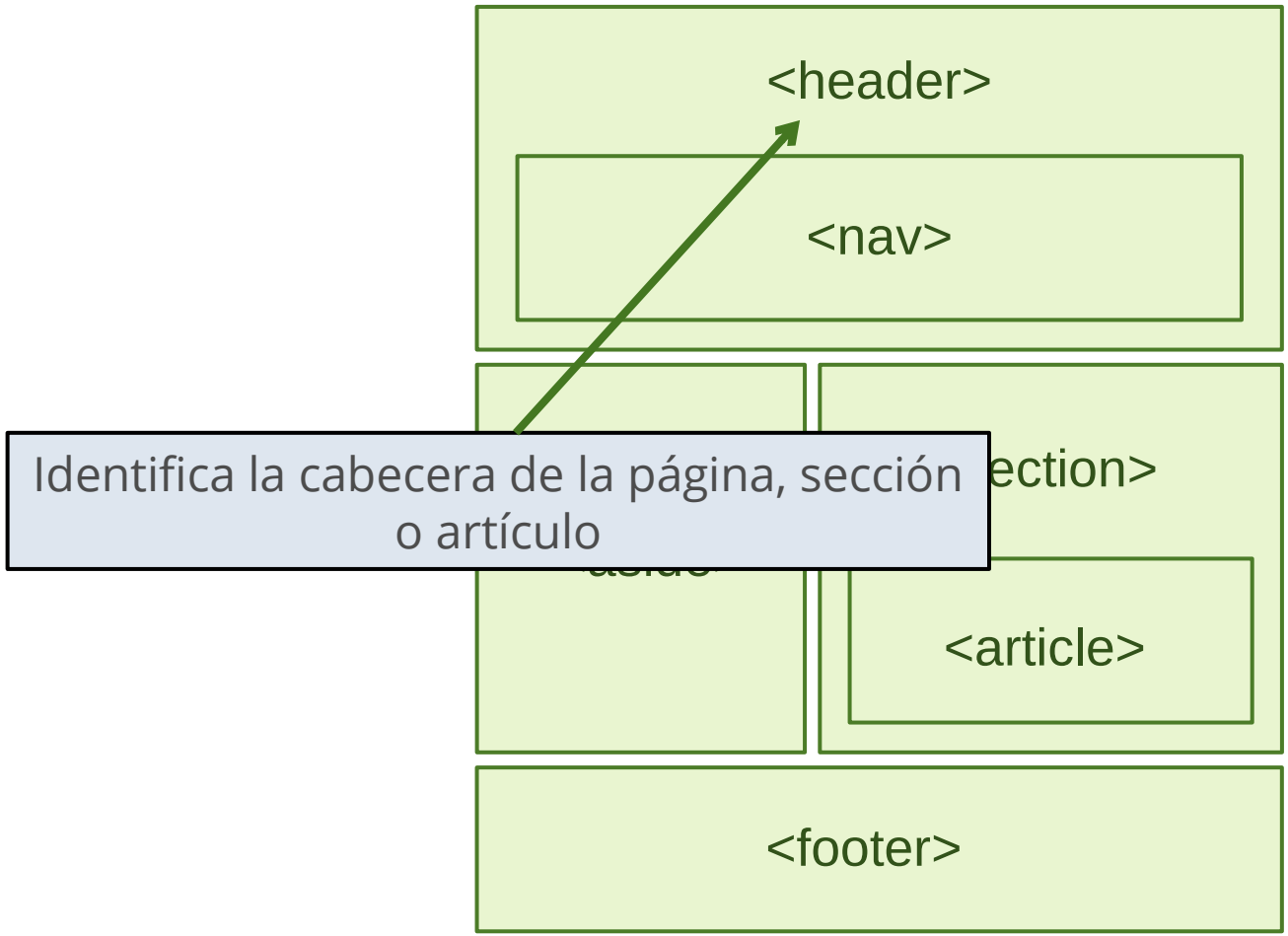
some info....

Títulos, párrafos y secciones

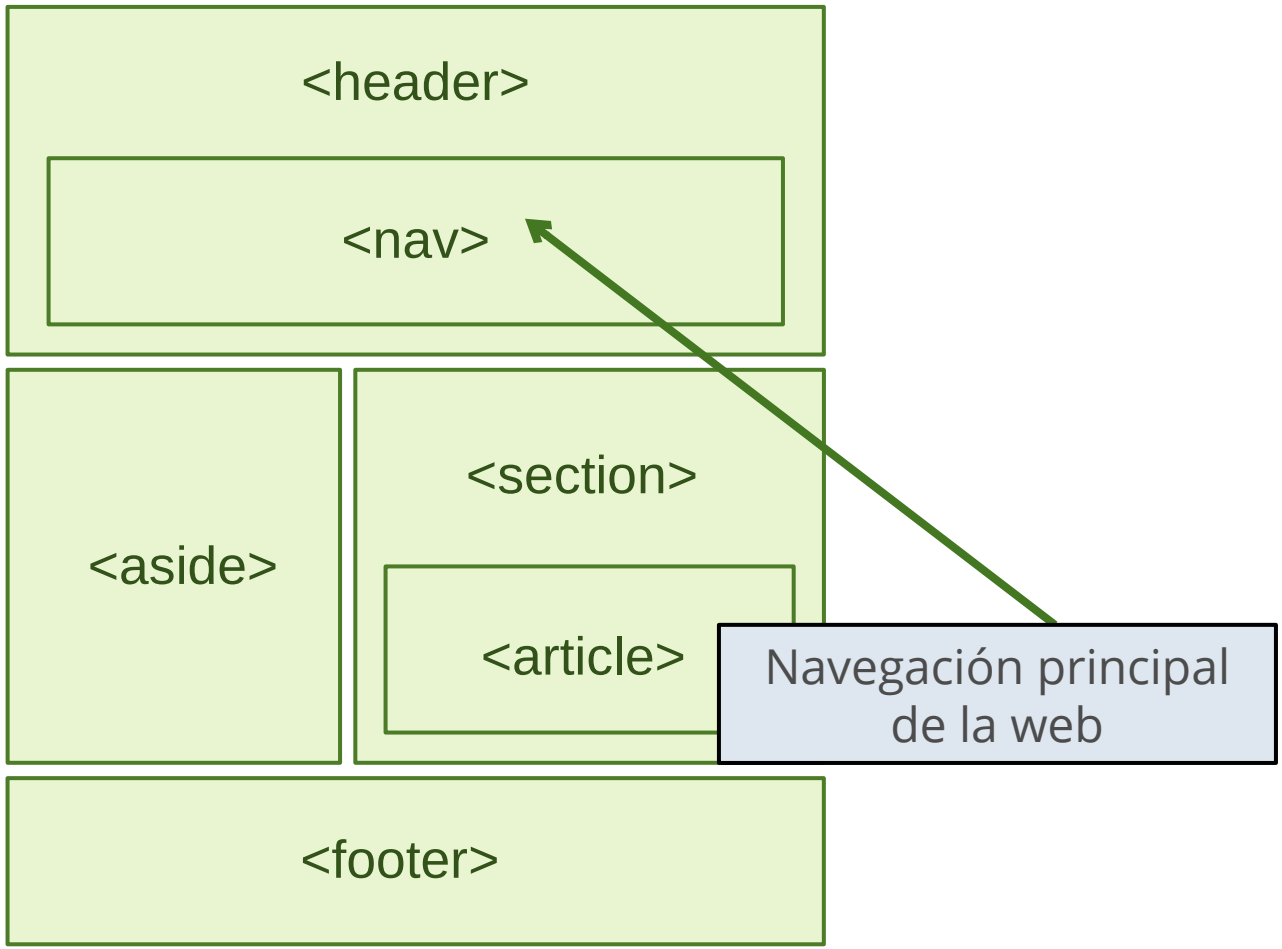
code URJC



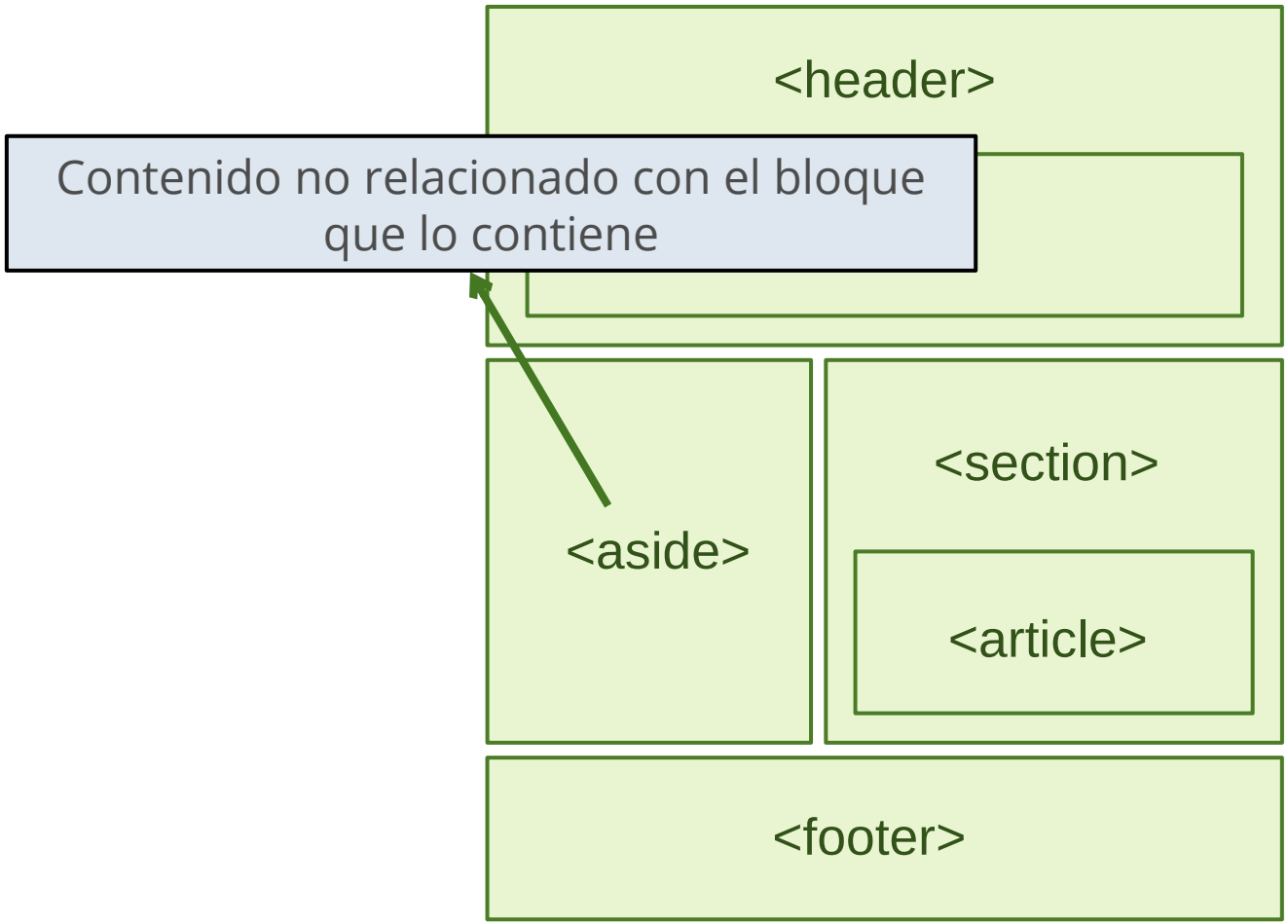
Títulos, párrafos y secciones



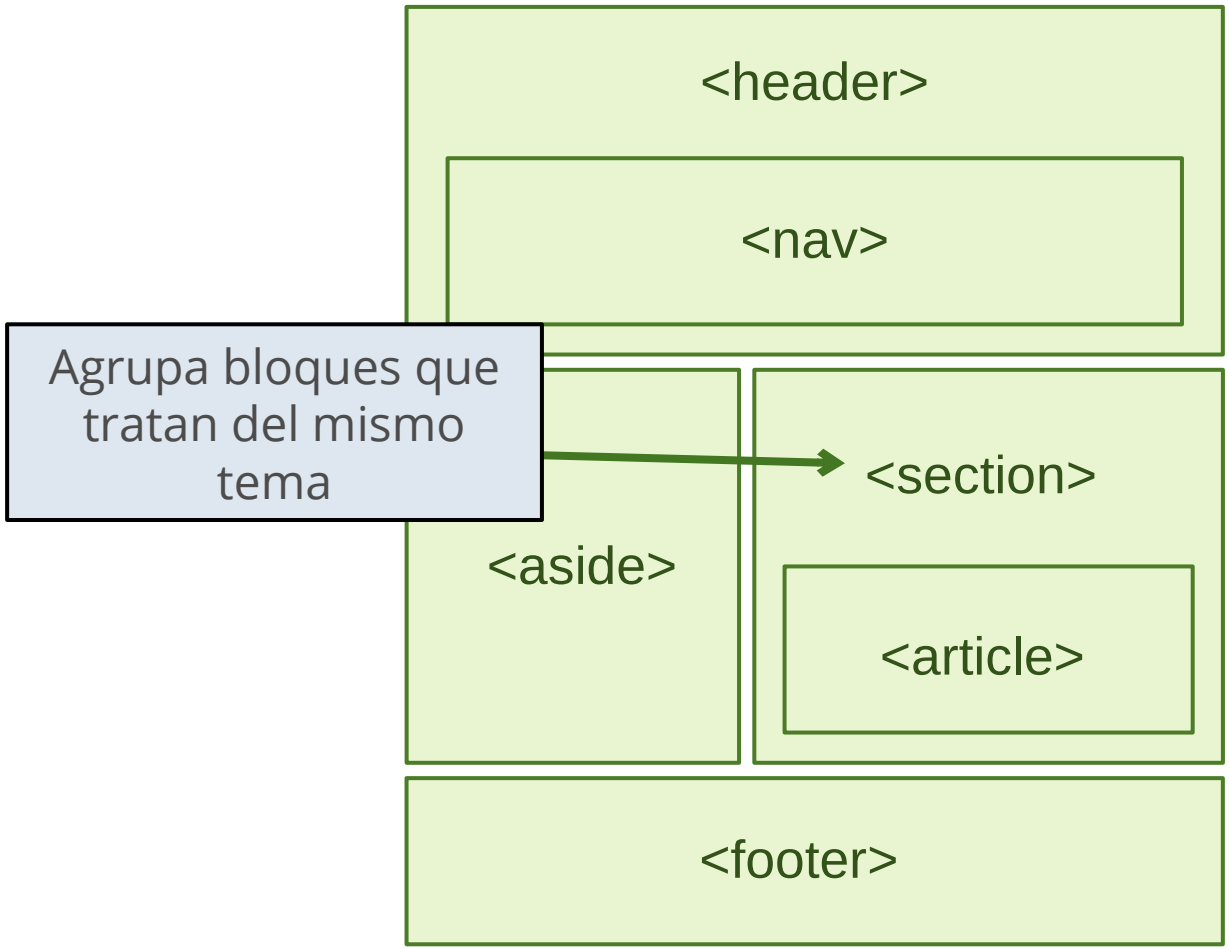
Títulos, párrafos y secciones



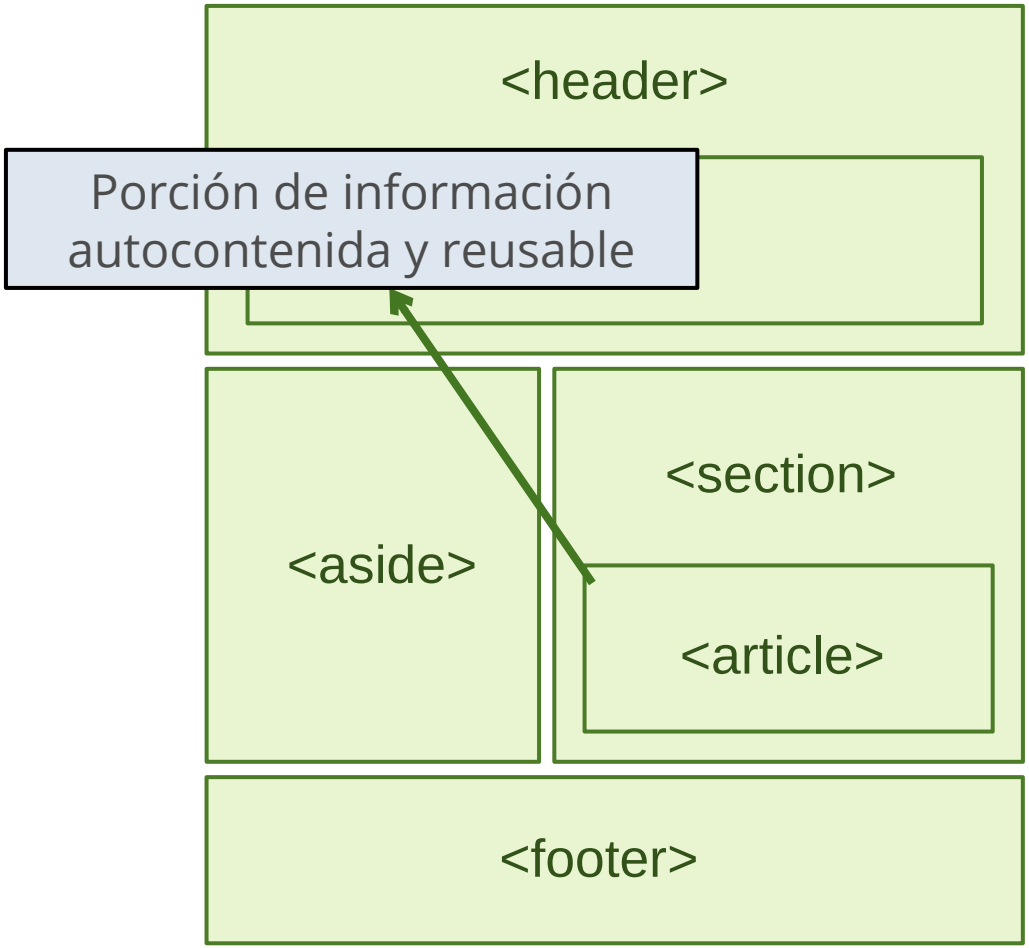
Títulos, párrafos y secciones



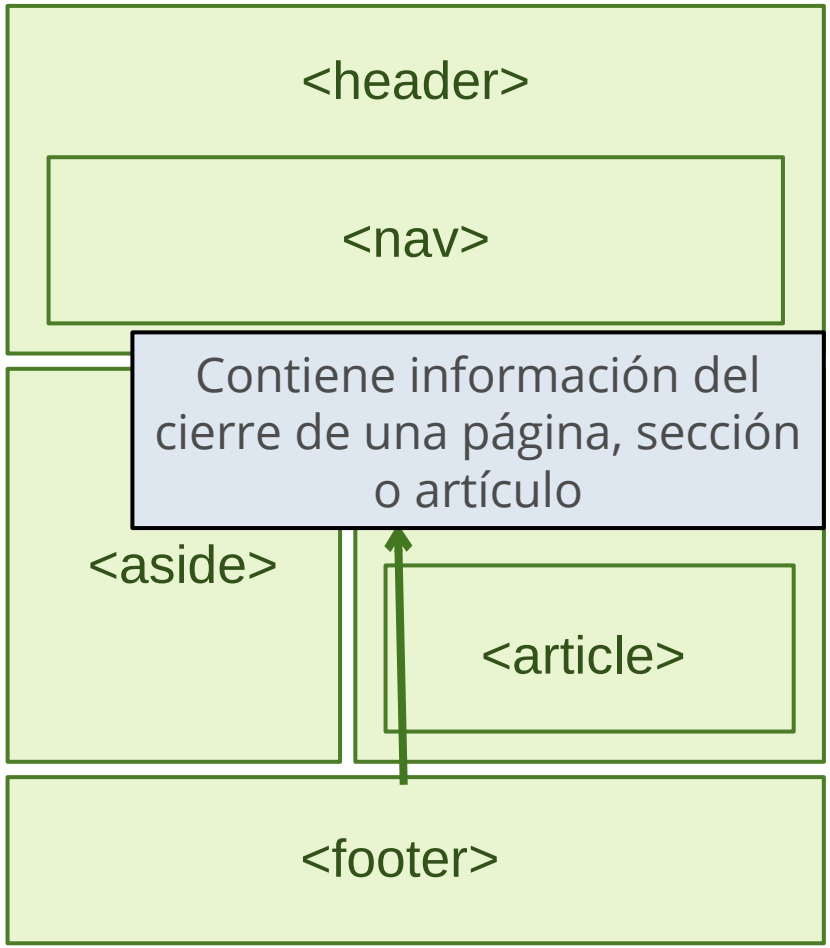
Títulos, párrafos y secciones



Títulos, párrafos y secciones



Títulos, párrafos y secciones



Títulos, párrafos y secciones

code URJC

```
<!DOCTYPE html>
<html>
   <head>
      <title>Título de la página</title>
   </head>
   <body>
      <header>
         <nav></nav>
      </header>
      <aside></aside>
      <section>
         <article></article>
      </section>
      <footer></footer>
   </body>
</html>
```

Comentarios

- Los comentarios en los documentos HTML se escriben entre **<!--** y **-->**

```
<!-- Esto es un comentario -->
<p>Esto es un texto normal</p>
<!-- Otro comentario -->
```

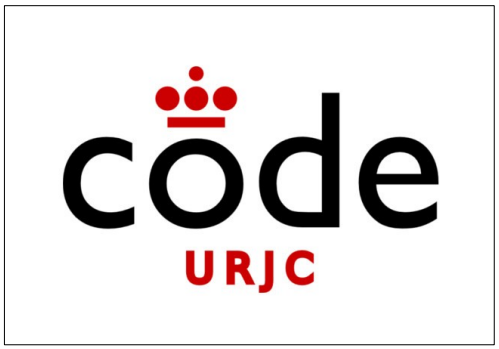
Imágenes

- Etiqueta

```
<img src="URL_IMAGEN"/>
```

Atributo de una etiqueta

```
<img src="https://www.codeurjc.es/img/codeurjc_logo.jpg" />
```



Imágenes

code URJC

- **Formatos de imágenes**
  - **PNG:** Ideal para imágenes con pocos colores diferentes o transparencias
  - **JPG:** Para imágenes con muchos colores
  - **GIF:** Imágenes animadas

Imágenes

code URJC

- **Imágenes de ejemplo**
  - Servicios online o programas que generan imágenes con la dimensión solicitada
  - Ideal para hacer diseño web sin dedicar tiempo a la creación de las imágenes con programas de diseño gráfico
  - http://lorempixel.com/
  - http://dummyimage.com/

Hiperenlaces

- Los enlaces son textos o imágenes del documento que si se pulsan se **carga** una nueva página en el **navegador**
  - Enlaces en **texto**

```
<a href="http://www.urjc.es">URJC</a>
```

  - Enlaces en **imágenes**

```
<a href="http://www.urjc.es">
   <img src="http://...">
</a>
```

Hiperenlaces

```
<!DOCTYPE html>
<html>
   <head>
      <title>URJC</title>
   </head>
   <body>
      <p>
         <a href="http://www.urjc.es">
            Universidad Rey Juan Carlos
         </a>
      </p>
         <a href="http://www.codeurjc.es">
            <img src="https://www.codeurjc.es/img/codeurjc_logo.jpg"/>
       </a>
   </body>
</html>
```

Hiperenlaces

```
<a href="https://www.urjc.es/">Enlace externo</a>

<a href="otraPagina.html">Enlace relativo</a>

<a href="about.html" target="_blank">Enlace a otra
ventana</a>

<a href="mailto:michel.maes@urjc.es?subjectEmail=Asunto
del correo&bodyEmail=Cuerpo del correo">Enviar
correo</a>
```

Listas de elementos

- **Listas ordenadas**
  - Lista: **
….
** (ordered list)
  - Elemento: **
…
** (list item)

```
<ol>
   <li>Ir a clase</li>
   <li>Hacer los deberes</li>
   <li>Estudiar</li>
   <li>Aprobar</li>
</ol>
```

1. Ir a clase
2. Hacer los deberes
3. Estudiar
4. Aprobar

Listas de elementos

- **Listas no ordenadas**
  - Lista: **
…
** (unordered list)
  - Elemento: **
…
** (list item)

```
<ul>
   <li>Ir a clase</li>
   <li>Hacer los deberes</li>
   <li>Estudiar</li>
   <li>Aprobar</li>
</ul>
```

- Ir a clase
- Hacer los deberes
- Estudiar
- Aprobar

Listas anidadas

```
<ol>
  <li>Dad's interests
     <ul>
        <li>football</li>
        <li>knitting</li>
     </ul>
  </li>
  <li>Mom's interests
    <ul>
       <li>hating football</li>
       <li>skydiving</li>
    </ul>
  </li>
</ol>
```

1. Dad's interests
    - football
    - knitting
2. Mom's interests
    - hating football
    - skydiving

Tablas

- Las tablas tienen **filas** y **celdas** en cada fila
  - Etiqueta: **<table>**….**</table>**
  - Filas (rows): **<tr>**….**</tr>**
  - Datos (data): **<td>**…**</td>**

Tablas

```
<table>
    <tr>
        <td>Nombre:</td>
        <td>Pepe</td>
    </tr>
    <tr>
        <td>Apellidos:</td>
        <td>González Díaz</td>
    </tr>
    <tr>
        <td>Fecha:</td>
        <td>13-02-2005</td>
    </tr>
</table>
```

Nombre:	Pepe
Apellidos:	González Díaz
Fecha:	13-02-2005

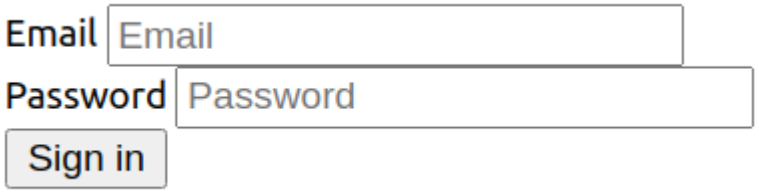
Tablas

```
<table>
   <thead>
     <tr>
       <th>Famous Monster</th>
       <th>Birth Year</th>
     </tr>
   </thead>
   <tbody>
     <tr>
       <td>King Kong</td>
       <td>1933</td>
     </tr>
     <tr>
       <td>Dracula</td>
       <td>1897</td>
     </tr>
   </tbody>
</table>
```

Famous Monster	Birth Year
King Kong	1933
Dracula	1897

Formularios

```
<form>
   <label for="inputEmailAddress">Email</label>
   <input type="email" id="inputEmail" placeholder="Email">
   <label for="inputPassword">Password</label>
   <input type="password" id="inputPassword" placeholder="Password">
   <button type="submit">Sign in</button>
</form>
```



Formularios

code URJC

- Campo de texto

```
<label for="inputEmailAddress">Email</label>
<input type="email" id="inputEmail" placeholder="Email">
```

Formularios

code URJC

- Password

```
<label for="inputPassword">Password</label>
<input type="password" id="inputPassword" placeholder="Password">
```

Password Password

Formularios

code URJC

- Tipos de campo de texto

```
<input type="button">
<input type="checkbox">
<input type="color">
<input type="date">
<input type="datetime-local">
<input type="email">
<input type="file">
<input type="hidden">
<input type="image">
<input type="month">
<input type="number">
<input type="password">
<input type="radio">
<input type="range">
<input type="reset">
<input type="search">
<input type="submit">
<input type="tel">
<input type="text">
<input type="time">
<input type="url">
<input type="week">
```

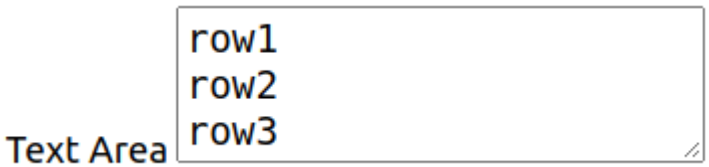
Mejor usabilidad en navegadores en móviles

Formularios

code URJC

- Área de texto

```
<label for="text-area">Text Area</label>
<textarea id="text-area" rows="3"></textarea>
```

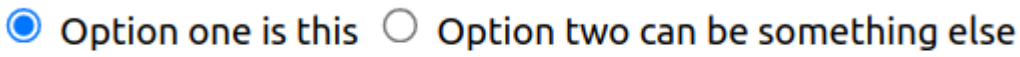


Formularios

code URJC

- Botones de radio

```
<label>
   <input type="radio" name="options-radios" id="optionsRadios1"
value="option1" checked> Option one is this
</label>
<label>
   <input type="radio" name="options-radios" id="optionsRadios2"
value="option2"> Option two can be something else
</label>
```



Formularios

- Botones de check

```
<label>
    <input type="checkbox" checked>
    Checkbox
</label>
```

Formularios

code URJC

- Selección

```
<label for="select">Selects</label>
<select id="select">
    <option value="1">1</option>
    <option value="2">2</option>
    <option value="3" selected>3</option>
    <option value="4">4</option>
    <option value="5">5</option>
</select>
```

Formularios

code URJC

- Botones

```
<form>
    <button>Cancel</button>
    <button type="reset">Reset</button>
    <button type="submit">Submit</button>
</form>
```

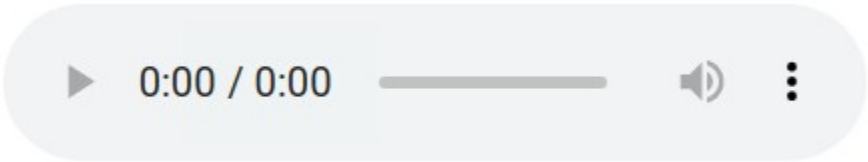
Cancel Reset Submit

Multimedia

code URJC

- Reproductor de audio

```
<audio src="song.ogg" controls autoplay loop></audio>
```



Multimedia

code URJC

- Reproductor de video

```
<video src="earth.mp4" controls autoplay loop></video>
```

Multimedia

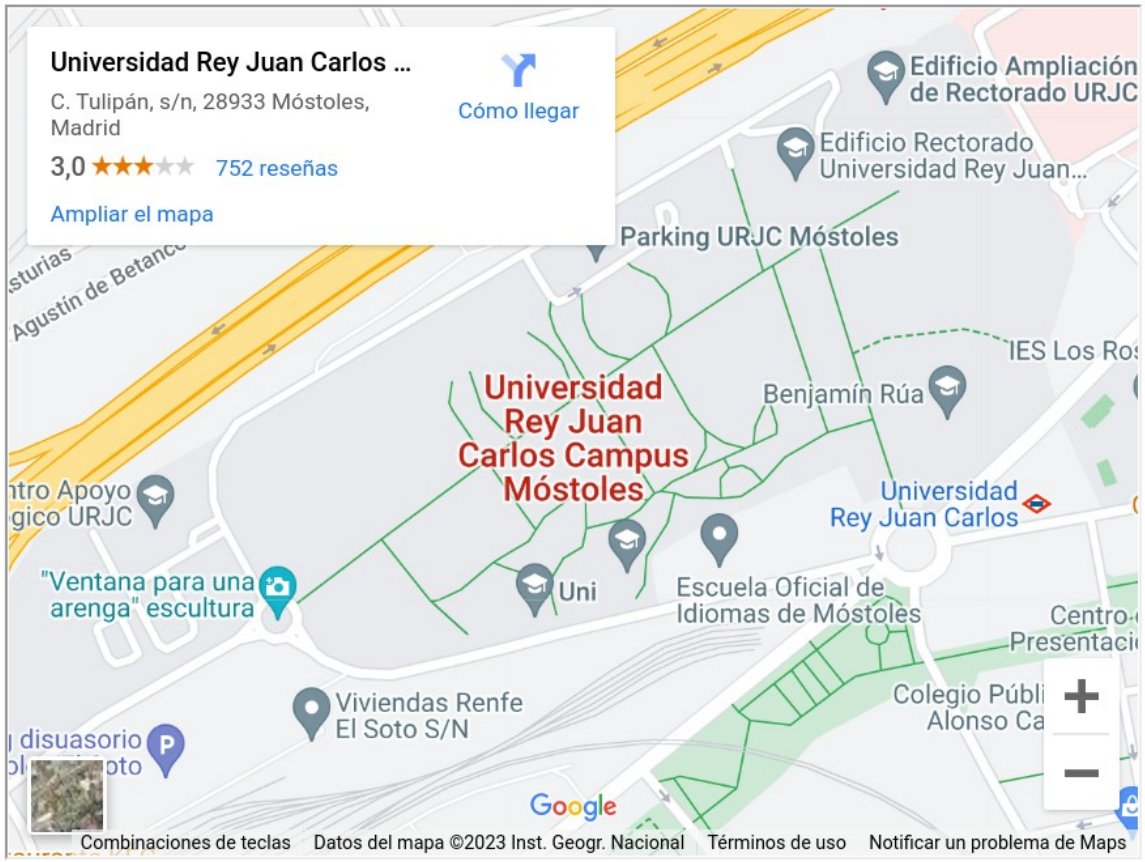
code URJC

- Mapa basado en Google Maps

```
<iframe src="https://www.google.com/maps/embed?pb=!1m18!
1m12!1m3!1d3041.311355105356!2d-3.8773975!
3d40.335438599999996!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!
4f13.1!3m3!1m2!1s0xd418e7adf68f543%3A0x841755428455a90b!
2sUniversidad%20Rey%20Juan%20Carlos%20Campus%20M
%C3%B3stoles!5e0!3m2!1ses!2ses!4v1693937155027!5m2!1ses!
2ses" width="600" height="450"></iframe>
```

Multimedia

- Mapa basado en Google Maps



Resaltar texto

- Texto **muy importante**:
  - ****Texto****
- Texto **enfatizado**:
  - ****Texto****

```
<p>
Este texto es <strong>muy importante</strong>.
Esto está <em>enfatizado</em>.
Pero esto está de <strong><em>las dos formas</em><strong>
</p>
```

Este texto es **muy importante**. Esto está *enfatizado*.
Pero esto está de ***las dos formas***

Estilos de los elementos

- El formato HTML permite indicar el contenido de forma **estructurada** y **semántica** (con significado)
- Se pueden dar **estilo** a los elementos del documento:
  - color, tamaño, tipo de letra, posición, bordes, etc…
- Los estilos se indican en el atributo **style**
- Como veremos más adelante, es preferible usar un **fichero CSS independiente** para los estilos, pero vamos a ver algunos estilos básicos

Estilos de los elementos

- Tamaño del texto

```
<p style="font-size: 10px">Texto pequeño!</p>
<p style="font-size: 20px">Texto grande!</p>
<p style="font-size: 40px">Texto super grande!</p>
```

Texto pequeño!

Texto grande!

Texto super grande!

Estilos de los elementos

- Color del texto

```
<p style="color: red">Rojo</p>
<p style="color: blue">Azul</p>
<p style="color: green">Verde</p>
```

Rojo

Azul

Verde

Estilos de los elementos

- Combinando estilos

```
<h1 style="color: green; font-size:16px">
        Big Heading
</h1>
<p style="color: red; font-size:10px">
        A giant bear and a little duck were friends
</p>
```

Big Heading

A giant bear and a little duck were friends.

Estilos de los elementos

- Color de fondo

```
<body style="background-color: green">
        <h1>Favorite Football Teams</h1>
        <ol style="background-color: yellow">
                <li>The Hawthorn Football Club</li>
                <li>San Franscisco 49ers</li>
        </ol>
</body>
```

Favorite Football Teams

1. The Hawthorn Football Club
2. San Franscisco 49ers

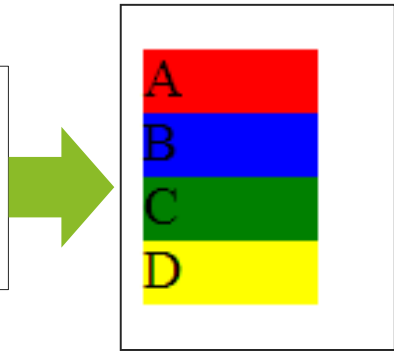
Elementos div y span

- ** ** y **** son elementos genéricos en HTML que no tienen semántica
- Se usan únicamente para seleccionar partes del documento a las que aplicar ciertos estilos
- ** ** se usa para zonas de la página
- **** se usa para palabras o frases en un texto

Elementos div y span

- ** **
  - Los elementos **div** son partes rectangulares de la página que pueden contener elementos HTML y estilo propio

```
<div style="width:50px; background-color:red">A</div>
<div style="width:50px; background-color:blue">B</div>
<div style="width:50px; background-color:green">C</div>
<div style="width:50px; background-color:yellow">D</div>
```



Elementos div y span

- ****
  - Los elementos **span** permiten marcar partes del texto para cambiarles el estilo

```
<p>This text is black, except for the word
<span style="color:red">red</span></p>
```

This text is black, except for the word red

Documentación

code URJC

- Estos son sólo algunos **elementos básicos** de HTML
- Existen muchos más elementos, algunos de ellos se han incorporado **recientemente**
- Y algunos presentes en las primeras versiones ya **no deberían usarse**
- Hay que usar siempre **documentación de referencia**

Bibliografía

- Algunos entornos de desarrollo tienen **documentación integrada** sobre HTML
- Páginas de interés:
  - Estándares:
    - http://dev.w3.org/html5/
    - http://www.w3.org/TR/html4/
  - Documentación muy completa sobre muchas tecnologías web
    - http://devdocs.io/html/
    - http://www.w3schools.com/
    - https://developer.mozilla.org/es/docs/Web/HTML
    - https://developer.mozilla.org/es/docs/HTML/HTML5

Ejercicio 1

- Crea una página HTML (sin incluir estilos)
- El contenido puede ser de cualquier temática:

  Una empresa, un grupo de música, un departamento de la universidad, un producto software, un teléfono móvil, una asociación cultural, etc…

- El contenido puede ser inventado o real (obtenido de una página de Internet)

Ejercicio 1

code URJC

- La página debe incluir, al menos:
  - Varias secciones con subsecciones
  - Fotografías/Imágenes
  - Listas de elementos
  - Una o varias tablas
  - Hiperenlaces

08-2023

code URJC

Fundamentos de la Web

Bloque II: Tecnologías de cliente web

Tema 2.2: CSS

Universidad Rey Juan Carlos

Micael Gallego
Correo: micael.gallego@urjc.es
Twitter: @micael_gallego

Nicolás Rodríguez
Correo: nicolas.rodriguez@urjc.es

Iván Chicano
ivan.chicano@urjc.es

Michel Maes
michel.maes@urjc.es

08-2023

code
URJC

Micael Gallego, Nicolás Rodríguez, Iván Chicano, Michel Maes

Definición

code URJC

- **Hojas de estilo en cascada**
- ***Cascading Style Sheets* (CSS)**
- Es un lenguaje utilizado para dar **estilo** a contenido estructurado
- Se aplica principalmente a documentos **HTML**, pero también se puede usar con otros documentos como **SVG**, **XML**, etc..

Definición

code URJC

- Con **CSS** se pueden especificar
  - **Colores:** principal, de fondo, degradados…
  - **Tipografía:** familia, tamaños, estilos…
  - ***Layout:*** Disposición de los elementos en el documento
  - **Efectos:** sombras, esquinas redondeadas…
  - **Animaciones:** en el cambio de los estilos

Definición

code URJC

- **Separación de Presentación y Contenido**
  - Aunque se puede indicar el estilo incluido en el elemento HTML (como hemos visto antes), es mejor separar el estilo del contenido
  - **Ventajas:**
    - Esto facilita la coordinación entre diseñadores/maquetadores y programadores
    - Se pueden usar distintos CSSs en función del tipo/tamaño del dispositivo
    - La página es más accesible porque el contenido está separado de la presentación del mismo

Definición

code URJC

- **Uso de CSS para dar estilo a un HTML**
  - **A)** Incluido en el elemento (atributo style)
  - **B)** Incluido en el HTML (en el HEAD)
  - **C)** En un fichero independiente (.css)

Definición

- **Uso de CSS para dar estilo a un HTML**
  - **A)** Incluido en el elemento (atributo style)

```
<!DOCTYPE html>
<html>
<head>
	<title></title>
</head>
<body>
	<p style="color:red">Red font!</p>
</body>
</html>
```

Definición

- **Uso de CSS para dar estilo a un HTML**
  - **B)** Incluido en el HTML (en el HEAD)

```
<!DOCTYPE html>
<html>
<head>
   <style>
      p {
         color: purple;
      }
   </style>
   <title>Result</title>
</head>
<body>
   <p>I'm purple!</p>
</body>
</html>
```

I'm purple!

Definición

- **Uso de CSS para dar estilo a un HTML**
  - **C)** En un fichero independiente (.css)

style.css

```
p {
    color: blue;
}
```

```
<!DOCTYPE html>
<html>
<head>
   <title>Result</title>
   <link type="text/css" rel="stylesheet" href="style.css"/>
</head>
<body>
   <p>I'm blue</p>
</body>
</html>
```

Definición

code URJC

- **Uso de CSS para dar estilo a un HTML**
  - **C)** En un fichero independiente (.css)
    - Es la forma **recomendada**
    - Un mismo fichero .css se puede aplicar a **páginas HTML diferentes**
    - Ese fichero puede estar **cacheado** por el navegador para que no se **descargue** repetidamente
    - Se pueden incluir **varios ficheros css** asociados al mismo HTML

Formato

- Un documento CSS tiene el siguiente formato:

Selector
Elemento al que aplicar el estilo identificado por las propiedades

Propiedad
Aspecto del estilo de un elemento. Fuente, color, tamaño…

```
selector {
        property: value;
        property2: value;
        property3: value;
}
selector2 {
        property1: value;
}
```

Formato

```
<h3>What's CSS for?</h3>
<p>CSS is for styling HTML pages!</p>
<h3>Why use it?</h3>
<p>It makes webpages look <span>really rad</span>.</p>
<h3>What do I think of it?</h3>
<p>It's awesome!</p>
```

```
p {
    font-family: Arial;
    color: blue;
    font-size: 12px;
}
h3 {
    color: red;
}
span {
    background-color: yellow;
}
```

Definición

code URJC

- **Hojas de estilos en "cascada"**
  - Se dice que son en **cascada** porque a los elementos se les aplican algunas de las propiedades del elemento **padre** (las heredan)
  - Si no se quieren heredar, se puede indicar un estilo particular para dicho elemento
  - Para saber si la propiedad se hereda o no hay que consultar la documentación

Definición

- **Hojas de estilos en "cascada"**
  - Como un **estilo está compuesto** de varias **propiedades** (color, tamaño, tipo de letra…), es posible que se declaren algunas nuevas pero otras se sigan **"heredando"** del elemento **padre**
  - En el ejemplo anterior, el elemento "hereda" el color azul y demás propiedades del elemento <p>

Definición

code URJC

- **Cursos online gratuitos sobre tecnologías web**
  - http://www.codecademy.com
  - https://www.codeschool.com/
  - https://lenguajecss.com/css/

*Algunos de los ejemplos de este material están basados en Codecademy

Colores

- Existen varias formas de especificar el valor de la propiedad **color** en CSS:
  - Utilizando el **nombre del color**
  - Utilizando sus componentes **RGB en formato hexadecimal**
  - Utilizando sus componente en **RGB o RGBA**

Colores

- Nombre

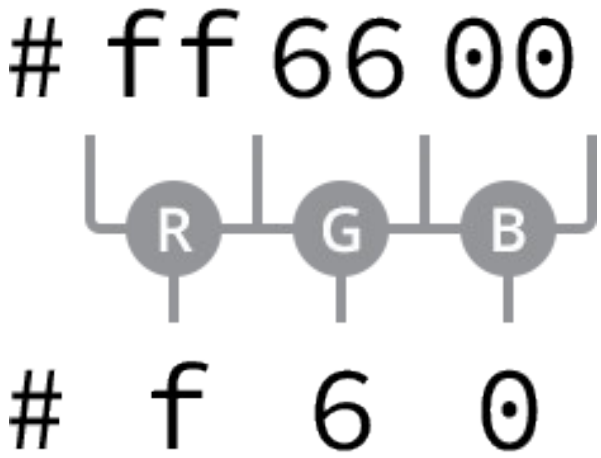
```
#id-name{
    color: red;
}
```

Named	Numeric	Color name	Hex rgb	Decimal
		black	#000000	0,0,0
		silver	#C0C0C0	192,192,192
		gray	#808080	128,128,128
		white	#FFFFFF	255,255,255
		maroon	#800000	128,0,0
		red	#FF0000	255,0,0
		purple	#800080	128,0,128
		fuchsia	#FF00FF	255,0,255
		green	#008000	0,128,0
		lime	#00FF00	0,255,0
		olive	#808000	128,128,0
		yellow	#FFFF00	255,255,0
		navy	#000080	0,0,128
		blue	#0000FF	0,0,255
		teal	#008080	0,128,128
		aqua	#00FFFF	0,255,255

Colores

- Valor hexadecimal

```
em{ color: #f00;}     /* #rgb */
em{ color: #ff0000;} /* #rrggbb */
```



Colores

code URJC

- Valor rgb/rgba

```
em{ color: rgb(255,0,0) }
em{ color: rgb(100%,0%,0%) }
em{ color: rgba(255,0,0,1) } /* 1 = SOLID */
em{ color: rgba(100%,0%,0%,0.5) } /* 50% transparent */
em{ color: rgba(255,0,0,0.1) } /* 90% transparent */
```

Colores

code URJC

- Generadores de paletas de colores
  - http://paletton.com/
  - https://color.adobe.com/es/create/color-wheel
  - Integrado en editores como Brackets

- Más información
  - http://devdocs.io/css/color_value

Tamaño de texto

code URJC

Tamaño absoluto (**px**)

```
div{ width: 100px; }
.title{ height: 50px; }
```

Tamaño relativo a su contenedor (**%**)

```
div{ width: 100%; }
.title{ height: 50%; }
```

Tamaño de texto

Tamaño absoluto (para impresión) (**pt**)

```
p{ font-size: 16pt;}
.title{ font-size: 18pt;}
```

Tamaño relativo a la fuente del elemento (**em**)

```
p{ font-size: 0.85em;}
.title{ font-size: 1.5em;}
```

Tamaño de texto

code URJC

- En general es recomendable usar **medidas relativas** en el tamaño del texto (**font-size**)
- Esto permite adaptar la página a diferentes dispositivos y resoluciones

http://devdocs.io/css/length

Tipo de letra

code URJC

- El tipo de letra (**font-family**) que se puede asociar a un texto depende de los tipos de letra del **sistema** o los que se pueden descargar de **Internet**
- Como no se puede saber qué tipos de letra están instalados o se pueden descargar, hay que indicar una **lista de tipos de letra**
- Es recomendable que el último sea un tipo de letra genérico que siempre existe: **serif, sans-serif, monospace, cursive, fantasy**

http://devdocs.io/css/font-family

Tipo de letra

```
font-family: Times, "Times New Roman", Georgia, serif;
font-family: Verdana, Arial, Helvetica, sans-serif;
font-family: "Lucida Console", Courier, monospace;
font-family: cursive;
font-family: fantasy;
```

This is an example of a serif font.

This is an example of a sans-serif font.

This is an example of a monospace font.

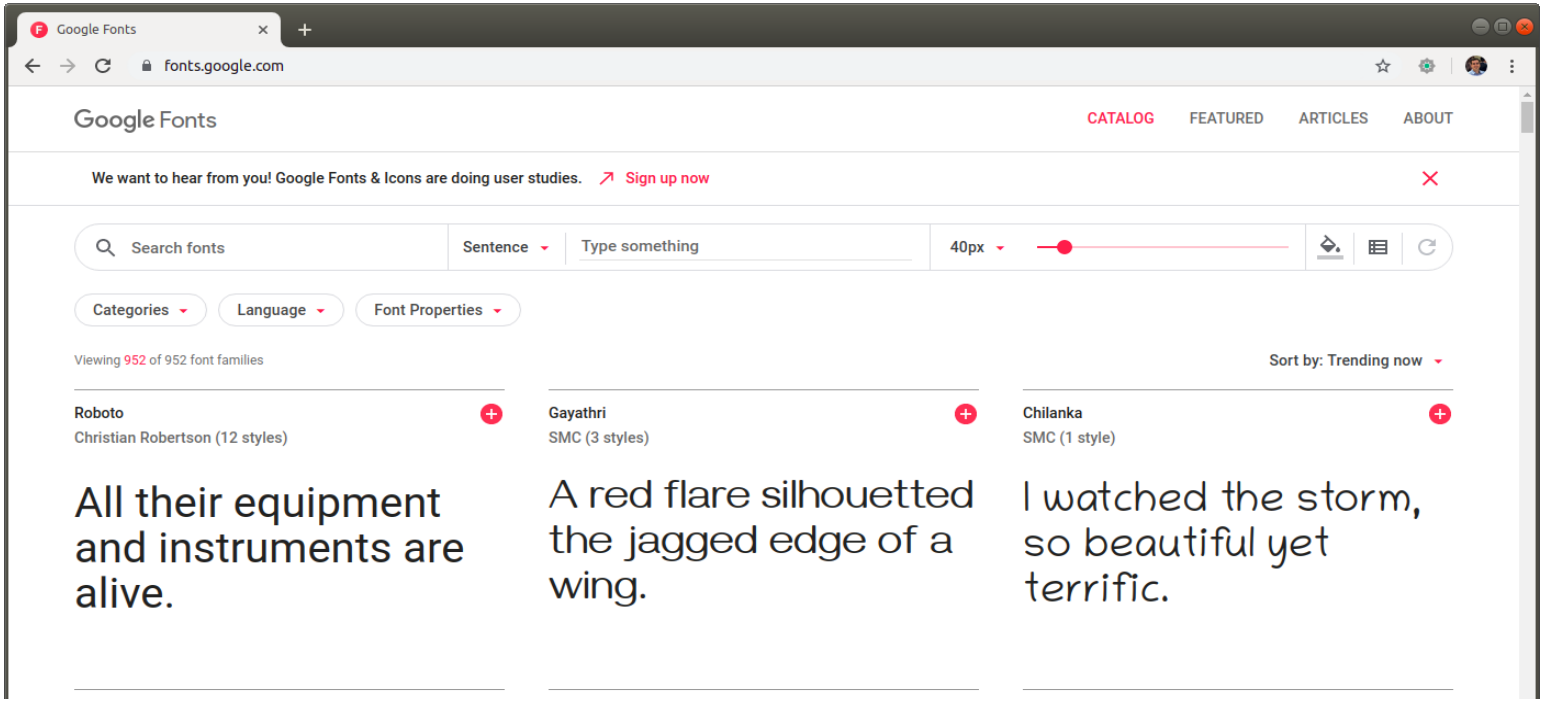
This is an example of a cursive font.

This is an example of a fantasy font.

http://devdocs.io/css/font-family

Tipo de letra

- **Google Fonts** proporciona cientos de tipos de letra que se descargan al visitar una web



https://www.google.com/fonts

Tipo de letra

- Google Fonts

```
<link href="https://fonts.googleapis.com/css?
family=Roboto" rel="stylesheet" type="text/css">
```

```
html{
        font-family: Roboto, Arial, sans-serif;
}
```

Normal 400

Grumpy wizards make toxic brew for the evil Queen and Jack.

Otros estilos de texto

code URJC

- Grosor (**font-weight**)

```
<p>Lorem ipsum...</p>
<p>Quisque...</p>
<p>Aliquam eget...</p>
```

```
html{
   font-weight: bold;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris elementum a dolor eu ultricies. Sed aliquet elit justo, non consequat arcu iaculis eu.

Quisque imperdiet tempor arcu. Pellentesque in luctus urna. Phasellus metus tellus, condimentum mollis tincidunt nec, feugiat in diam. Sed feugiat lacus eget odio imperdiet, at vehicula massa congue.

Aliquam eget ornare velit. Pellentesque fringilla convallis odio et pellentesque. Donec id viverra velit.

http://www.w3schools.com/cssref/pr_font_weight.asp

Otros estilos de texto

- Interlineado **(line-height)**

```
<p>Lorem ipsum...</p>
<p>Quisque...</p>
<p>Aliquam eget...</p>
```

```
html{
    line-height: 20pt;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris elementum a dolor eu ultricies. Sed aliquet elit justo, non consequat arcu iaculis eu.

Quisque imperdiet tempor arcu. Pellentesque in luctus urna. Phasellus metus tellus, condimentum mollis tincidunt nec, feugiat in diam. Sed feugiat lacus eget odio imperdiet, at vehicula massa congue.

Aliquam eget ornare velit. Pellentesque fringilla convallis odio et pellentesque. Donec id viverra velit.

Otros estilos de texto

code URJC

- Alineación **(text-align)**

```
<p>Lorem ipsum...</p>
<p>Quisque...</p>
<p>Aliquam eget...</p>
```

```
html{
   text-align:center;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris elementum a dolor eu ultricies. Sed aliquet elit justo, non consequat arcu iaculis eu.

Quisque imperdiet tempor arcu. Pellentesque in luctus urna. Phasellus metus tellus, condimentum mollis tincidunt nec, feugiat in diam. Sed feugiat lacus eget odio imperdiet, at vehicula massa congue.

Aliquam eget ornare velit. Pellentesque fringilla convallis odio et pellentesque. Donec id viverra velit.

Otros estilos de texto

code URJC

- Otras propiedades interesantes
  - **text-decoration**
  - **text-shadow**
  - **text-transform**
  - **letter-spacing**
  - **word-spacing**

Bordes

- Hay muchos elementos en una página HTML que pueden tener bordes
- El borde se configura con la propiedad **border**
- Esta propiedad está formada por 3 elementos:
  - **Ancho del borde:** px, em…
  - **Estilo:** none, hidden, dotted, solid, dashed…
  - **Color**

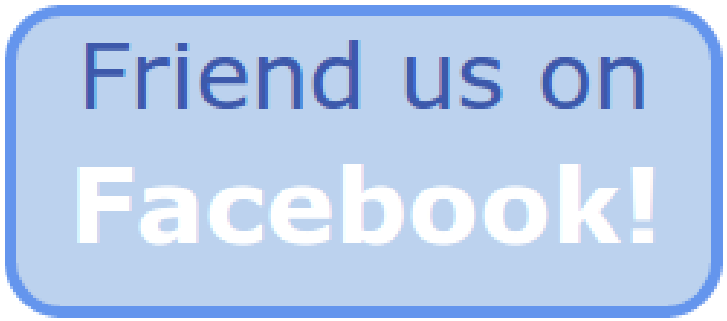
```
table {
    border: 1px solid black;
}
```

http://devdocs.io/css/border

Ejemplo: botón básico de CSS

code URJC

- Para experimentar con todo lo aprendido vamos a hacer un botón usando **CSS**
- El botón se define en **HTML** con un elemento ** **



Ejemplo: botón básico de CSS

- El botón se define en **HTML** con un elemento ** **
- Las propiedades CSS del div serán:
  - height: 50px
  - width: 120px
  - border-color: #6495ED
  - background-color: #BCD2EE
  - border-width: 2px

Ejemplo: botón básico de CSS

- Para que los bordes del botón sean **redondeados**, se usará la propiedad:
  - border-radius: 5px;
- **Centramos** el botón en la **página**. Para ello, ponemos el mismo margen a ambos lados:
  - margin: auto;
- **Centramos** el **contenido** del botón con:
  - text-align: center;

Ejemplo: botón básico de CSS

code URJC

- Se incluye un link dentro del div para que sea realmente un botón

```
<div>
  <a href="link">Friend us on <span>Facebook!</span></a>
</div>
```

- Se da estilo a los textos:

```
a {
   text-decoration: none;
   color: #3D59AB;
   font-family: Verdana, sans-serif;
}
```

```
span {
   font-weight: bold;
   font-size: 18px;
   color: #ffffff;
}
```

Ejemplo: botón básico de CSS

code URJC

button.css

```css
div {
   height: 50px;
   width: 120px;
   border: 5px solid #6495ED;
   background-color: #BCD2EE;
   border-radius: 15px;
   margin: auto;
   text-align: center;
}

a {
   text-decoration: none;
   color: #3D59AB;
   font-family: Verdana, sans-serif;
}

span {
   font-weight: bold;
   font-size: 18px;
   color: #ffffff;
}
```

button.html

```html
<!DOCTYPE html>
<html>

<head>
   <title>Facebook button</title>
   <link type="text/css"
rel="stylesheet"
href="button.css" />
</head>

<body>
   <div>
      <a href="link">Friend us on
         <span>Facebook!</span>
      </a>
   </div>
</body>

</html>
```

Ejercicio 1

code URJC

- Crea una página web con HTML y CSS con el siguiente botón

Selectores CSS

- Los **selectores** sirven para seleccionar los elementos a los que se les dará el estilo en CSS
- Cuando se usa como selector el nombre de un **elemento**, esas propiedades se aplicarán a todos los elementos de ese tipo en la página

El estilo se aplicará a todos los elementos **a** (links) de la página

```
a {
    color: #3D59AB;
}
```

- Existen otras formas de seleccionar elementos

Elementos anidados

code URJC

- Elementos que están dentro de otros elementos

```
<div>
    <div>
        <p>I like tacos!</p>
    </div>
</div>
```

El estilo se aplicará a todos los elementos p que estén dentro de un div que a su vez esté dentro de otro div

```
div > div > p {
    margin: 10px;
}
```

Elementos anidados

- Podemos seleccionar elementos dentro de otros elementos pero permitiendo que haya elementos entre medias
- Quitamos el símbolo **">"**

```
<div>
    <ul>
        <li>Leche</li>
        <li>Galletas</li>
    </ul>
</div>
```

El estilo se aplicará a todos los elementos **li** que estén dentro de un **div** aunque no sea directamente

```
div li {
    margin: 10px;
}
```

Selector comodín

- Selector * para seleccionar cualquier elemento

```
*{
    border: 2px solid black;
}
```

- Es útil cuando se quiere dar un estilo a elementos concretos (usando el nombre del elemento como selector) y se quiere configurar un **estilo para todos los demás**

Selector comodín

- Usar el selector comodín * para elementos dentro de otros elementos

Se aplica a todos los elementos dentro de un párrafo

```
p * {
    margin: 10px;
}
```

Selectores

- ¿Qué ocurre si a un mismo elemento le corresponden varias reglas CCS?

```
ul li p {
    color: red;
}
```

```
p {
    color: blue;
}
```

- Siempre se aplica el **selector más específico**

```
<p>La lista de la compra:</p>
<ul>
  <li><p>Leche</p></li>
  <li><p>Galletas</p></li>
  <li><p>Azúcar</p></li>
</ul>
```

La lista de la compra:

- Leche
- Galletas
- Azúcar

Selectores de clase e ID

- Es habitual dar un **nombre a un elemento** concreto para poder darle estilo independientemente de su tipo y su posición en el documento HTML

```
<p id="anuncio_principal">El anuncio</p>
```

- También se puede crear un **tipo de elementos**, de forma que a todos los elementos de ese tipo se les aplicará el estilo

```
<p class="cita">Lo perfecto es... </p>
```

Selectores de clase e ID

code URJC

- Definir el estilo del elemento con **id**

```
#anuncio_principal {
    color: #FF00FF;
    font-weight: bold;
}
```

- Definir el estilo de los elementos de una **clase**

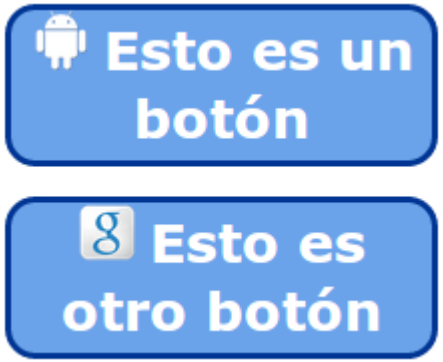
```
.cita {
    color: red;
}
```

Selectores de clase e ID

- Usando el atributo **class** podemos reutilizar los estilos de esa clase en varias partes de la página
- Asignaremos el **class** correspondiente al elemento que queramos que tenga el estilo

```
.button {
    ...
}
.button a {
    ...
}
.button img {
    ...
}
```

```
<div class="button">
<img src="http://... "/>
<a href="">Un botón</a>
</div>
<div class="button">
<img src="http://... "/>
<a href="">Otro botón</a>
</div>
```



Pseudo-clase

- El sistema de selectores permite seleccionar elementos del HTML
- También se pueden usar los selectores para aplicar estilos diferentes dependiendo del estado de un elemento (**pseudo-class**)

```
selector:pseudo-class_selector {
    property: value;
}
```

Pseudo-clase

- Por ejemplo, se puede cambiar el estilo de los links cuando el usuario pasa encima de ellos con el ratón

```
a:hover {
    color: #cc0000;
    font-weight: bold;
    text-decoration: none;
}
```

- Otros pseudo-clases de link:
  - a:link: Enlace no visitado
  - a:visited: Enlace visitado

Pseudo-clase

- Otras pseudo-clases interesantes:
  - elem:first-child: Se aplica el primer elemento hijo

```
p:first-child { color: red; }
```

  - elem:nth-child(n): Hijo n

```
p:nth-child(2) { color: red; }
```

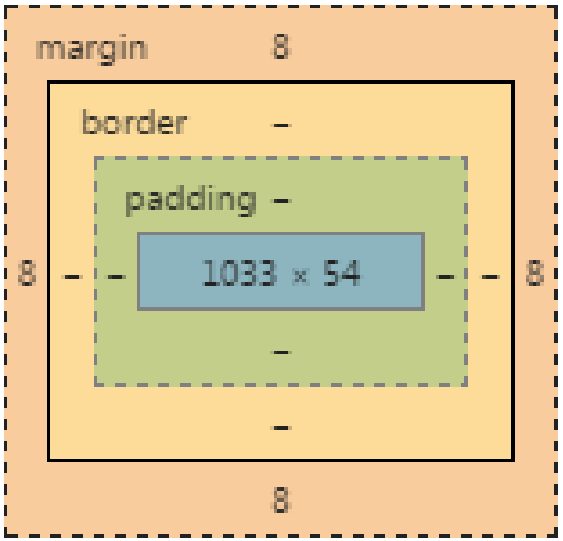
https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes

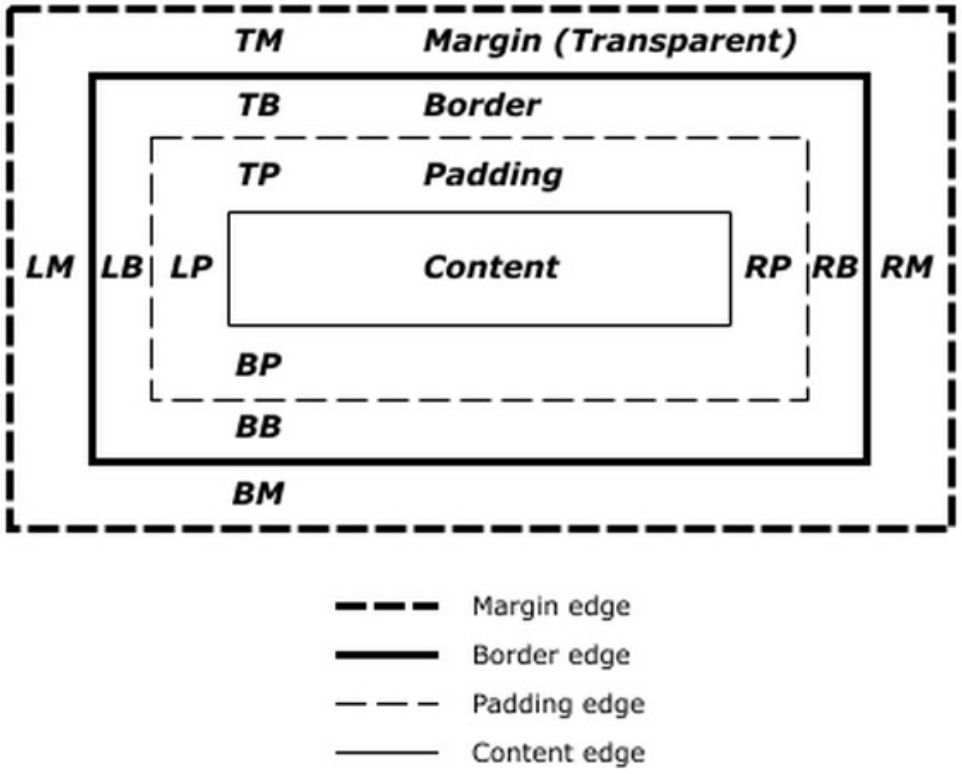
Tamaño, margen y posición

code URJC

- Por defecto, los elementos de una página se comportan como **palabras en un texto**
- Pero su tamaño, margen y posición es altamente configurable en base al **box model**
- Cada elemento está en una caja que tiene:
  - **Margen:** Espacio transparente alrededor del elemento
  - **Borde:** Línea que rodea el elemento
  - **Relleno (padding):** Relleno del elemento desde su contenido hasta el borde
  - **Contenido:** Información del propio elemento

Tamaño, margen y posición





Tamaño, margen y posición

- Por defecto, los elementos ** ** ocupan todo el ancho de la página

```
#one { background-color: red }
#two { background-color: blue }
#three { background-color: yellow }
#four { background-color: green }
```

```
<div id="one">A</div>
<div id="two">B</div>
<div id="three">C</div>
<div id="four">D</div>
```

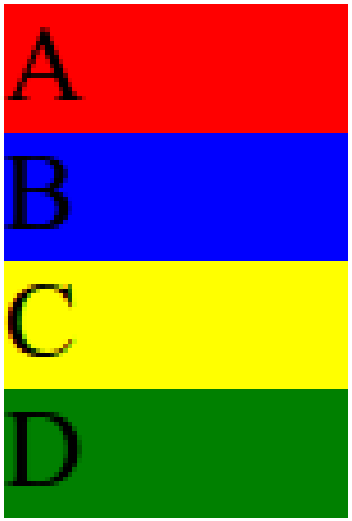
Tamaño, margen y posición

- Propiedad **display**:
  - **display: block**
    - No permite otros elementos en la misma línea (por defecto)
    - Se puede cambiar su alto y ancho
  - **display: inline-block**
    - Permite otros elementos en la misma línea
    - Se puede cambiar su alto y ancho
  - **display: inline**
    - Permite otros elementos en la misma línea
    - **NO** se puede cambiar su alto y ancho
  - **display: none**
    - Oculto

Tamaño, margen y posición

code URJC

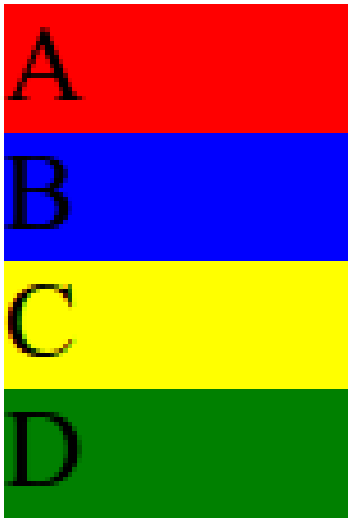
```
div {
    width: 50px;
}
```



Tamaño, margen y posición

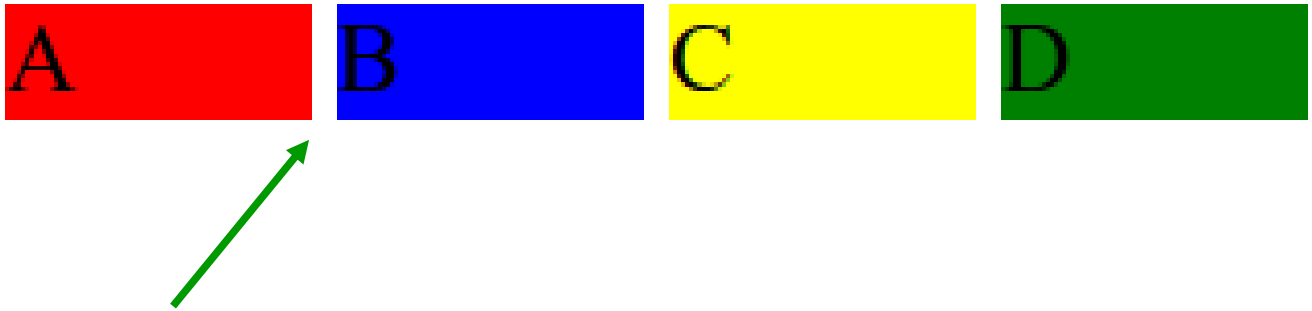
code URJC

```
div {
    width: 50px;
    display: block;
}
```



Tamaño, margen y posición

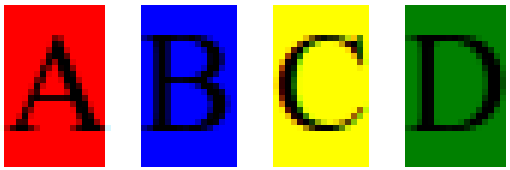
```
div {
    width: 50px;
    display: inline-block;
}
```



Este espacio se puede evitar con margin: -4px

Tamaño, margen y posición

```
div {
    width: 50px;
    display: inline;
}
```



¿Y el none?

Este espacio se puede evitar con margin: -2px

Tamaño, margen y posición

code URJC

- Propiedad **margin**:
  - Controla el espacio alrededor de un elemento
  - **margin: auto:** Pone el mismo margen a la izquierda y a la derecha. Es decir, centra el elemento horizontalmente
  - **margin-top, margin-right, margin-bottom, margin-left:** Permiten cambiar el margen individualmente a cada lado
  - **margin: 1px 2px 3px 4px:** Forma compacta de cambiar todos los márgenes a la vez (top right bottom left)
  - **margin: 10px:** El mismo margen a cada lado

Tamaño, margen y posición

code URJC

- Propiedad **border**:
  - Controla el borde del elemento
  - **border-width:** Ancho del borde
  - **border-color:** Color del borde
  - **border-style:** Estilo del borde: solid, dashed, dotted, groove…
  - **border: 4px solid #FFoooo:** Forma compacta de cambiar todas las propiedades a la vez

Tamaño, margen y posición

code URJC

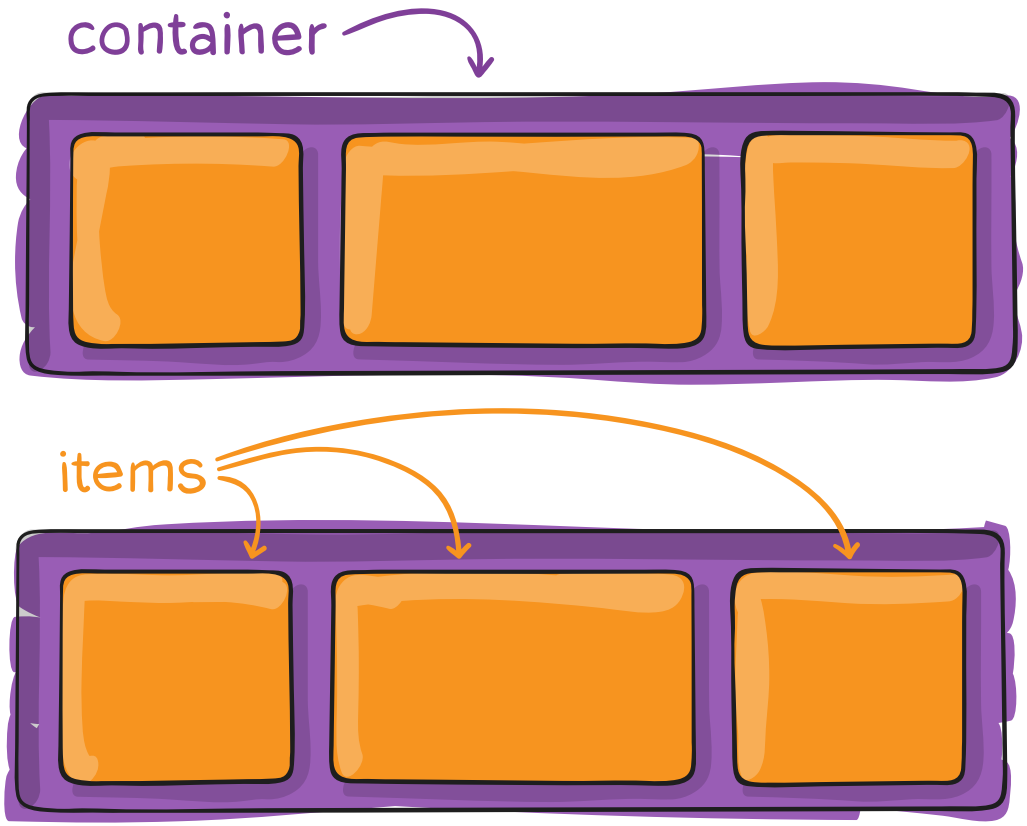
- Propiedad **padding**:
  - Rellena entre el contenido y el borde
  - **padding-top, padding-right, padding-bottom, padding-left:** Permiten cambiar el relleno individualmente a cada lado
  - **padding: 1px 2px 3px 4px:** Forma compacta de cambiar todos los rellenos a la vez
  - **padding: 10x:** Forma compacta de poner el mismo padding en todos los lados

Tamaño, margen y posición

code URJC

- El uso de **positon** y **float** está "obsoleto"
- **Flexbox** es una nueva especificación para el posicionamiento de elementos HTML mucho más flexible y potente (estructuras de una sola dimension)
- **Grid** es una especificación orientada a estructuras más comlejas que las del **Flexbox**

Flexbox

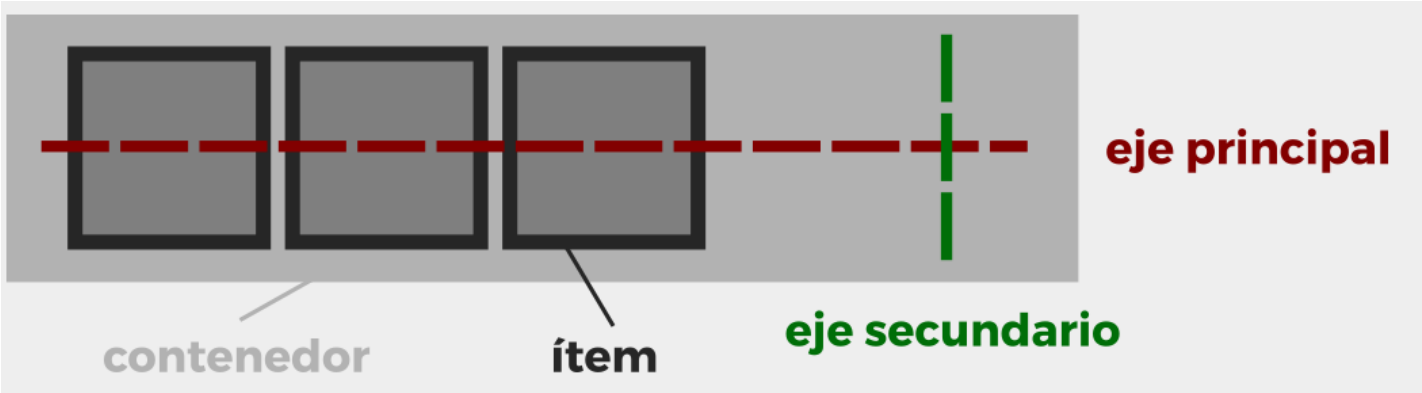


Este material se ha elaborado a partir de https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Flexbox

Especificación

- **Contenedor**: Es el elemento padre que tendrá en su interior cada uno de los ítems flexibles.
- **Eje principal**: Los contenedores flexibles tendrán una orientación principal específica. Por defecto, el eje principal del contenedor flexbox es en horizontal (en fila).



Flexbox

code URJC

Especificación

- **Eje secundario**: De la misma forma, los contenedores flexibles tendrán una orientación secundaria, perpendicular a la principal. Si la principal es en horizontal, la secundaria será en vertical (y viceversa).
- **Ítem**: Cada uno de los hijos que tendrá el contenedor en su interior.

https://lenguajecss.com/css/maquetacion-y-colocacion/flexbox/

Flexbox

Propiedades

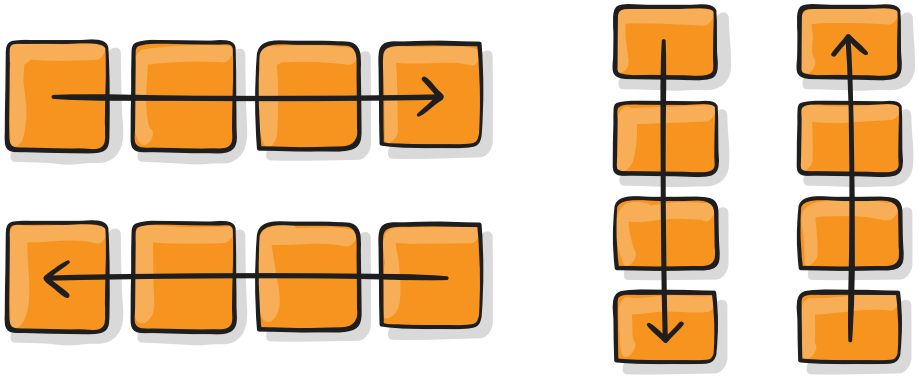
- **inline-flex**: Establece un contenedor en línea (ocupa solo el contenido).
- **flex**: Establece un contenedor en bloque (ocupa todo el ancho del padre).

```
.container {
        display: flex; /* or inline-flex */
}
```

Flexbox

flex-direction

- Establece el eje principal (horizontal o vertical) y la dirección



```
.container {
        display: flex;
        flex-direction: row | row-reverse | column | column-reverse;
}
```

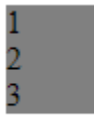
flex-direction: column

Flexbox

code URJC

```
<div class="container"> <!-- Flex container -->
   <div class="item item-1">1</div> <!-- Flex items -->
   <div class="item item-2">2</div>
   <div class="item item-3">3</div>
</div>
```

```
.container {
   display: flex;
   flex-direction: column;
   background: steelblue;
}
.item {
   background: grey;
}
```

flex-direction: column 

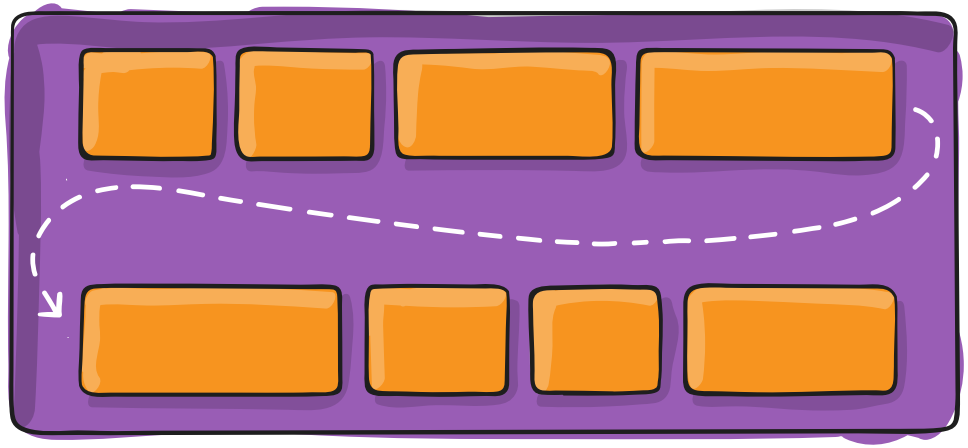
flex-direction: row 

Flexbox

code URJC

flex-wrap

- Evita o permite el desbordamiento (multilinea).



```
.container {
        display: flex;
        flex-wrap: nowrap | wrap | wrap-reverse;
}
```

Flexbox

code URJC

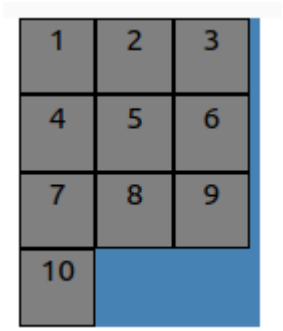
flex-wrap

- Evita o permite el desbordamiento (multilinea).
  - **nowrap**: Establece los ítems en una sola línea (no permite que se desborde el contenedor).
  - **wrap**: Establece los ítems en modo multilínea (permite que se desborde el contenedor).
  - **wrap-reverse**: Establece los ítems en modo multilínea, pero en dirección inversa.

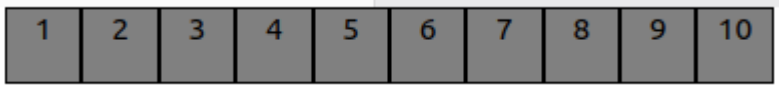
Flexbox

flex-wrap

flex-wrap: wrap;



flex-wrap: nowrap;




```
<div class="container">
   <div class="item">1</div>
   <div class="item">2</div>
   <div class="item">3</div>
   <div class="item">4</div>
   <div class="item">5</div>
   <div class="item">6</div>
   <div class="item">7</div>
   <div class="item">8</div>
   <div class="item">9</div>
   <div class="item">10</div>
</div>
```

```
.container {
   display: flex;
   flex-wrap: wrap;
   background: steelblue;
   width: 100px;
}
.item {
   background: grey;
   min-height: 30px;
   min-width: 30px;
   border: 1px solid black;
   text-align: center;
}
```

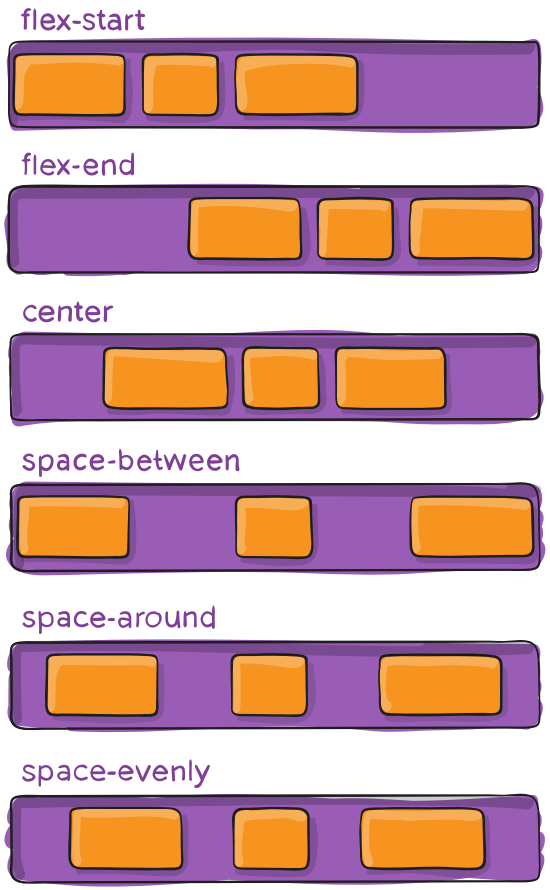
Flexbox

code URJC

Justify-content

- Se utiliza para alinear los ítems del eje principal

```
.container {
   display: flex;
   justify-content: flex-start | flex-end | center |
space-between | space-around | space-evenly | ...
}
```



Flexbox

code URJC

Justify-content

```
<div class="container">
   <div class="item">1</div>
   <div class="item">2</div>
   <div class="item">3</div>
   <div class="item">4</div>
   <div class="item">5</div>
</div>
```

```
.container {
   display: flex;
   justify-content: flex-start;
   background: steelblue;
   width: 300px;
}
.item {
   background: grey;
   min-height: 30px;
   min-width: 30px;
   border: 1px solid black;
   text-align: center;
}
```

justify-content: flex-start;

justify-content: flex-end;

justify-content: center;

justify-content: space-between;

justify-content: space-around;

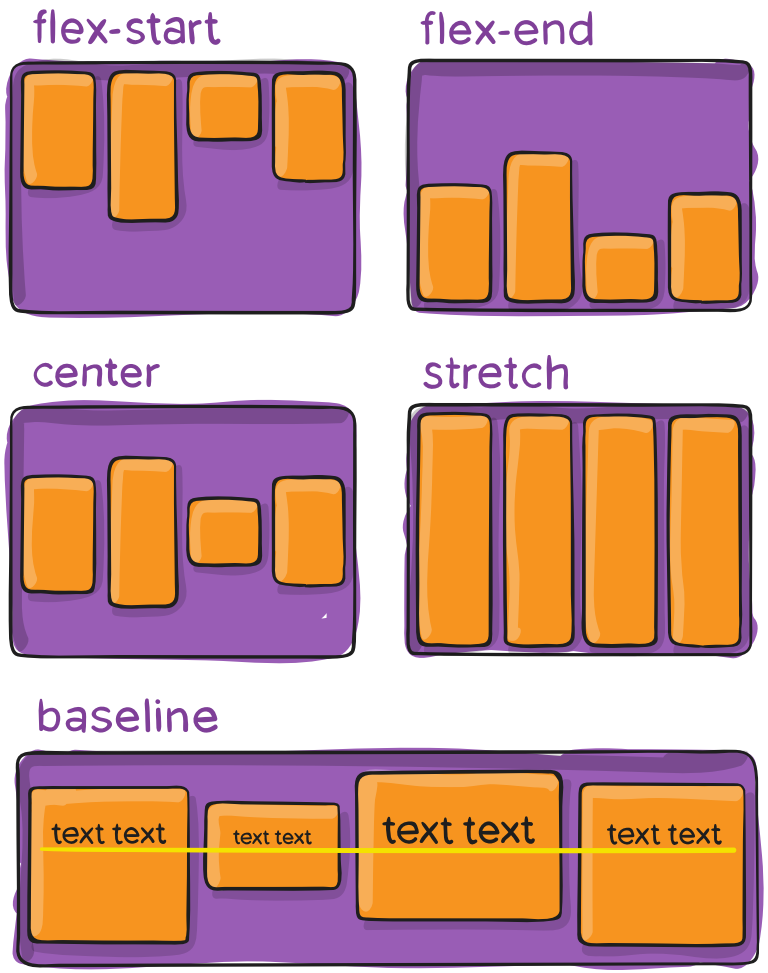
Flexbox

code URJC

align-items

- Usada para alinear los ítems del eje secundario

```
.container {
   display: flex;
   align-items: stretch | flex-start | flex-end |
center | baseline | first baseline | ….
}
```



Flexbox

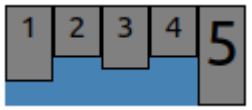
code URJC

align-items

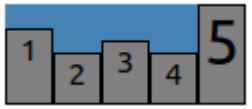
```
<div class="container">
   <div class="item">1</div>
   <div class="item">2</div>
   <div class="item">3</div>
   <div class="item">4</div>
   <div class="item">5</div>
</div>
```

```
.container {
   display: flex;
   align-items: flex-end;
   background: steelblue;
   width: 100px;
}
.item {
   background: grey;
   min-height: 20px;
   min-width: 20px;
   border: 1px solid black;
   text-align: center;
}
.item:nth-child(1) { min-height: 30px; }
.item:nth-child(3) { min-height: 25px; }
.item:nth-child(5) { min-height: 40px; height: 40px;}
```

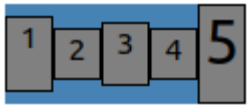
justify-content: flex-start;



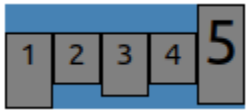
justify-content: flex-end;



justify-content: center;



justify-content: baseline;



justify-content: stretch;

Flexbox

code URJC

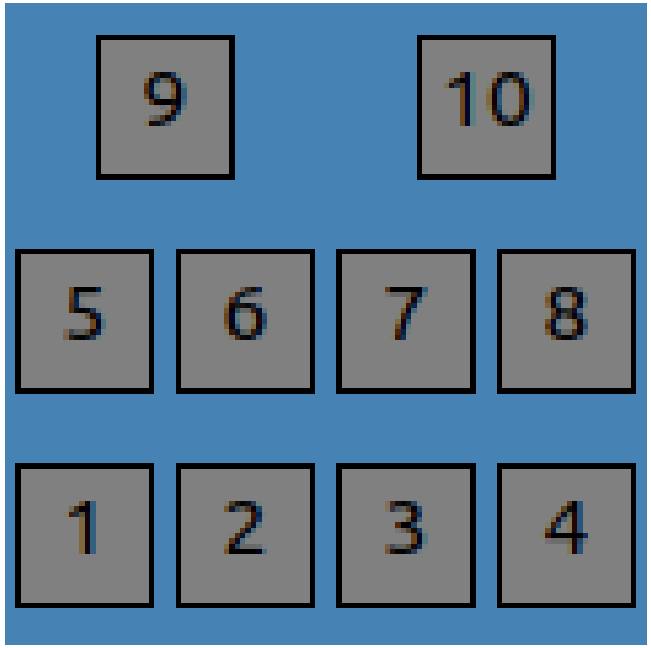
Propiedades comunes

- **flex-end:** Agrupa los ítems al final del eje principal.
- **center**: Agrupa los ítems al centro del eje principal.
- **space-between**: Distribuye los ítems dejando el máximo espacio para separarlos.
- **space-around**: Distribuye los ítems dejando el mismo espacio alrededor de ellos (izq/dcha).
- **space-evenly**: Distribuye los ítems dejando el mismo espacio (solapado) a izquierda y derecha.
- **stretch**: Estira los ítems para ocupar de forma equitativa todo el espacio.

Ejercicio 2

code URJC

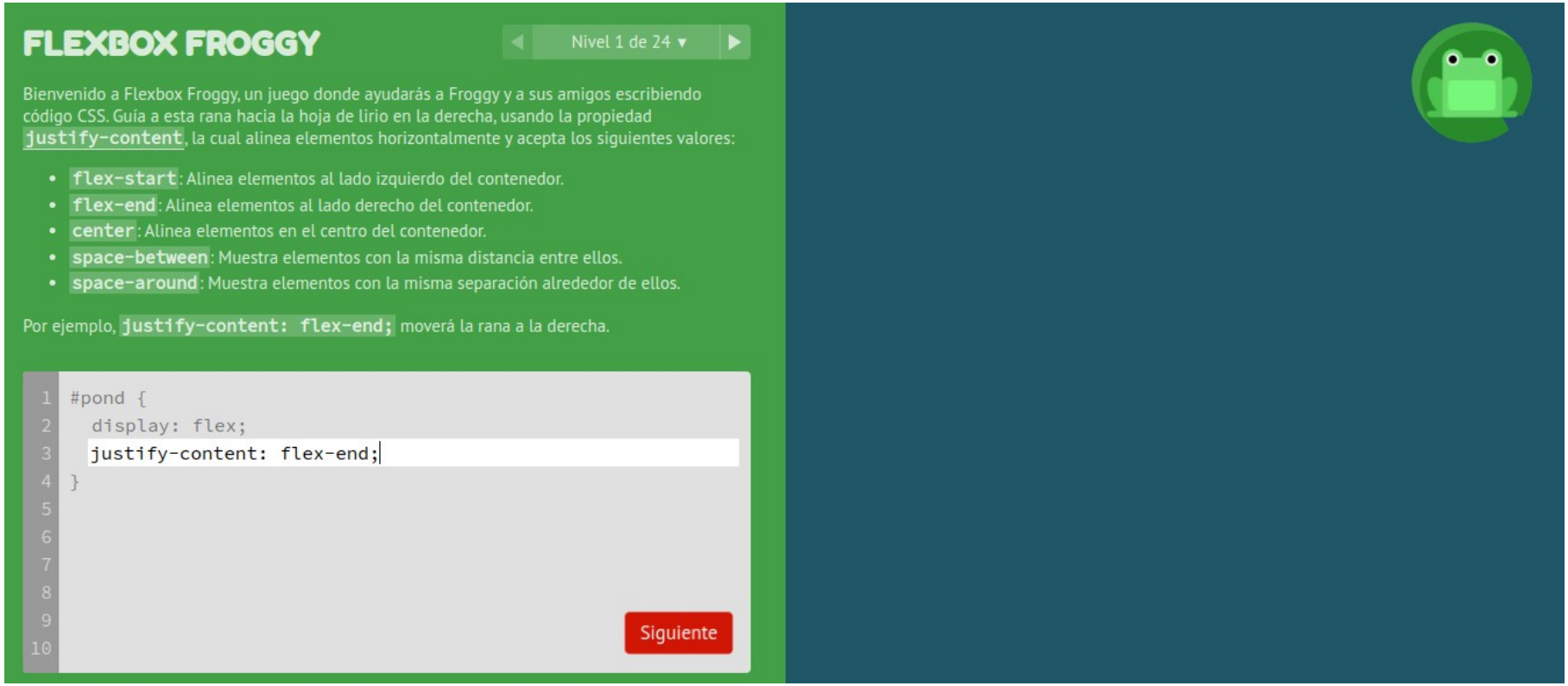
- Trata de replicar la siguiente figura utilizando solo CSS (*Flexbox)*




```
<div class="container">
   <div class="item">1</div>
   <div class="item">2</div>
   <div class="item">3</div>
   <div class="item">4</div>
   <div class="item">5</div>
   <div class="item">6</div>
   <div class="item">7</div>
   <div class="item">8</div>
   <div class="item">9</div>
   <div class="item">10</div>
</div>
```

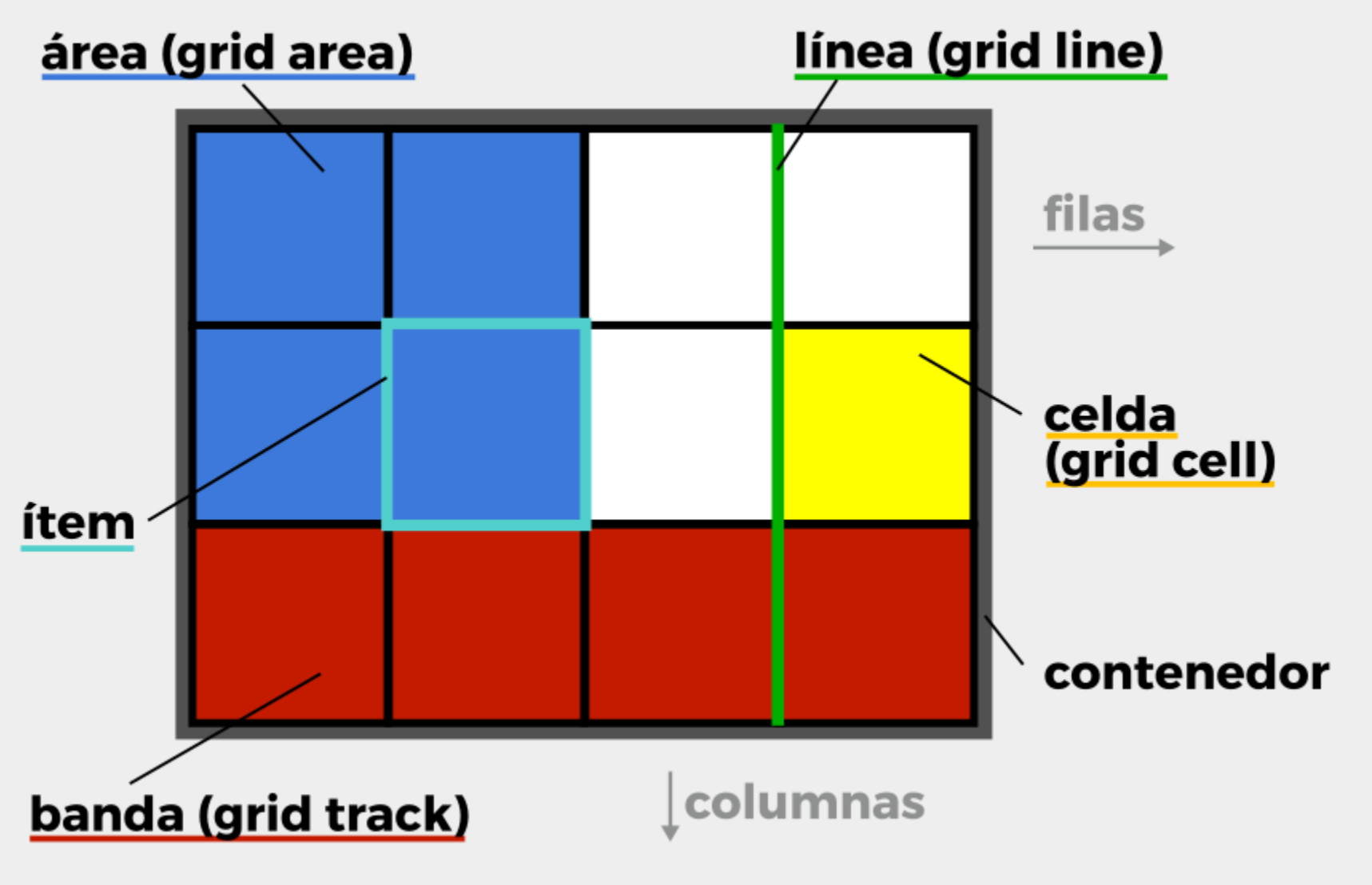
Flexbox

- Herramienta para aprender *Flexbox*



https://flexboxfroggy.com/#es

Grid



https://lenguajecss.com/css/maquetacion-y-colocacion/grid-css/

Grid

code URJC

Especificación

- **Contenedor:** El elemento padre contenedor que definirá la cuadrícula o rejilla.
- **Ítem:** Cada uno de los hijos que contiene la cuadrícula (elemento contenedor).
- **Celda (grid cell):** Cada uno de los cuadritos (unidad mínima) de la cuadrícula.
- **Area (grid area):** Región o conjunto de celdas de la cuadrícula.
- **Banda (grid track):** Banda horizontal o vertical de celdas de la cuadrícula.
- **Línea (grid line):** Separador horizontal o vertical de las celdas de la cuadrícula.

Grid

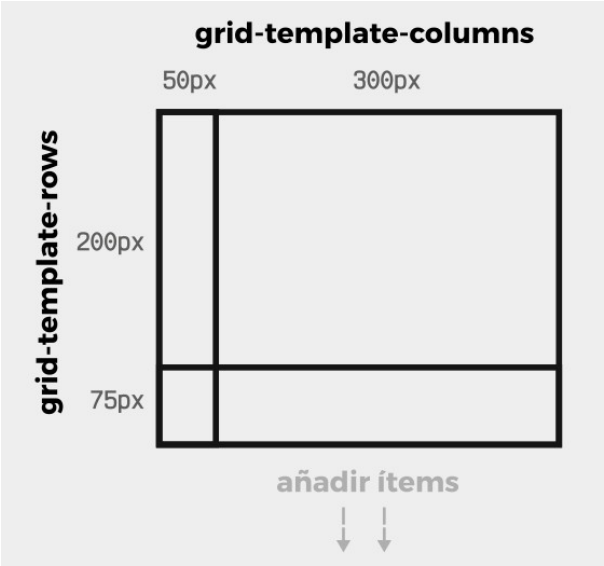
code URJC

Propiedades

- **inline-grid:** Establece una cuadrícula con ítems en línea
- **grid:** Establece una cuadrícula con ítems en bloque
- **grid-template-columns:** Establece el tamaño de cada columna (col 1, col 2…).
- **grid-template-rows:** Establece el tamaño de cada fila (fila 1, fila 2…).

Grid

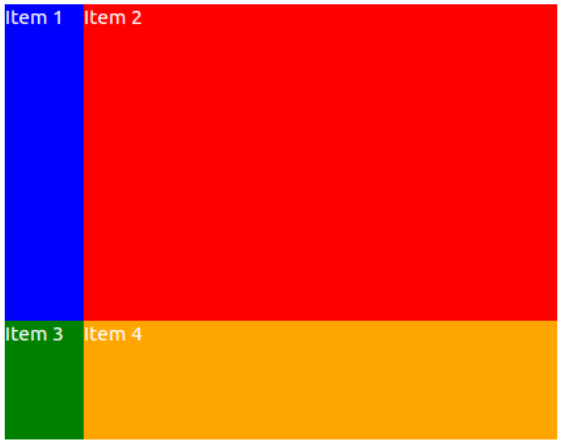
code URJC




```
.grid {
   display: grid;
   grid-template-columns: 50px 300px;
   grid-template-rows: 200px 75px;
}
.item{ color: white;}
.a { background: blue }
.b { background: red }
.c { background: green }
.d { background: orange }
```

- Dos columnas, una con 50 px y otra con 300 px de ancho
- Dos filas, una con 200 px y otra con 75 px de largo

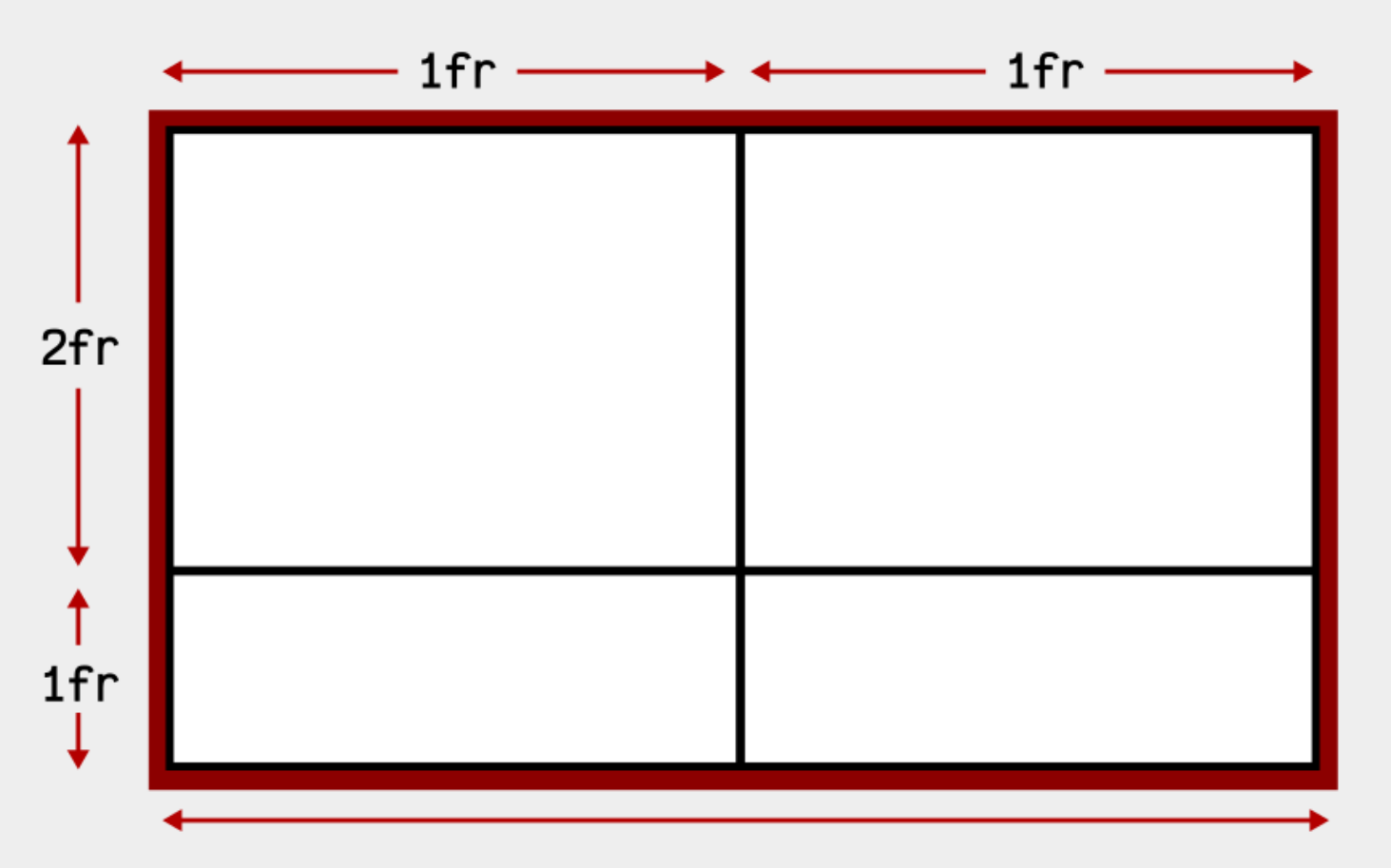
```
<div class="grid">
   <div class="item a">Item 1</div>
   <div class="item b">Item 2</div>
   <div class="item c">Item 3</div>
   <div class="item d">Item 4</div>
</div>
```



https://lenguajecss.com/css/maquetacion-y-colocacion/grid-css/

Grid

Unidad de fracción restante



- Se puede combinar con otras unidades

https://lenguajecss.com/css/maquetacion-y-colocacion/grid-css/

Grid

Filas y columnas repetitivas

- Se pueden hacer repetitivas las filas y las columnas

```
.grid {
   display: grid;
   grid-template-columns: 100px repeat(2, 50px) 200px;
   grid-template-rows: repeat(2, 50px 100px);
}
```

- En el ejemplo anterior, si añadimos nuevos items, se crearán una columna de tamaño 100 px, dos de tamaño 50, y una cuarta de tamaño 200 px
- Además, cuatro filas de tamaño 50 px y 100 px

Grid

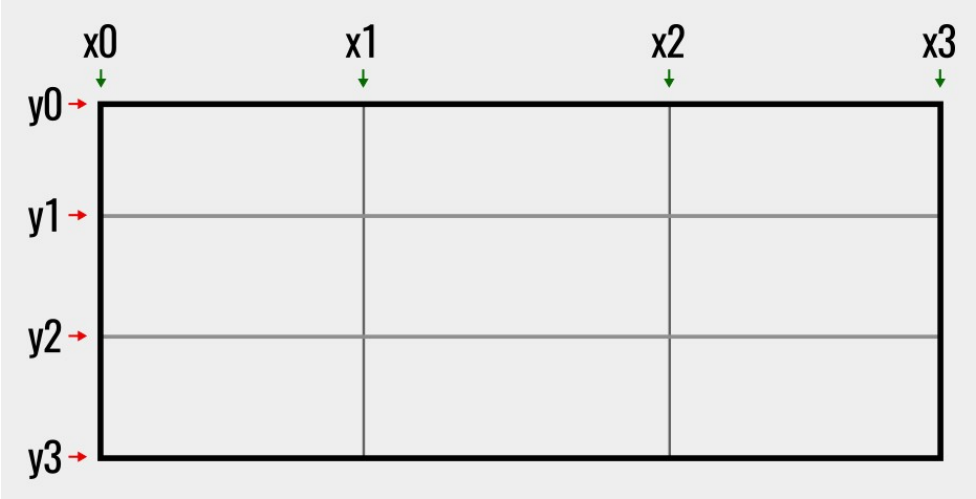
Filas y columnas repetitivas

- Se pueden hacer repetitivas las filas y las columnas

```
.grid {
   display: grid;
   grid-template-columns: 100px repeat(2, 50px) 200px;
   grid-template-rows: repeat(2, 50px 100px);
}
```

Grid

Propiedades - elementos



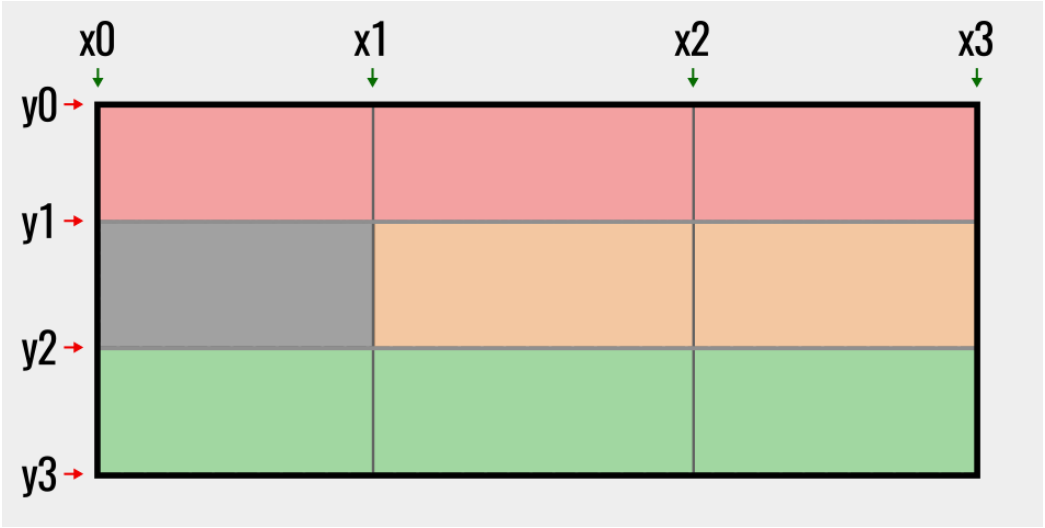

```
<div class="grid">
   <div class="header">Header</div>
   <div class="sidebar">Sidebar</div>
   <div class="content">Content</div>
   <div class="footer">Footer</div>
</div>
```

```
.grid {
   display: grid;
   grid-template-columns: [x0] 1fr [x1] 1fr [x2] 1fr [x3];
   grid-template-rows: [y0] 1fr [y1] 1fr [y2] 1fr [y3];
}
```

https://lenguajecss.com/css/maquetacion-y-colocacion/grid-css/

Grid

```
.header {
  background: darkred;
  grid-column-start: x0;
  grid-column-end: x3;
}
.sidebar {
  background: black;
  grid-row: y1 / y2;
  color: white;
}
.content {
  background: orange;
  grid-column: x1 / x3;
  grid-row: y1 / y3;
}
.footer {
  background: green;
  grid-column: x0 / x3;
  grid-row: y2;
}
```



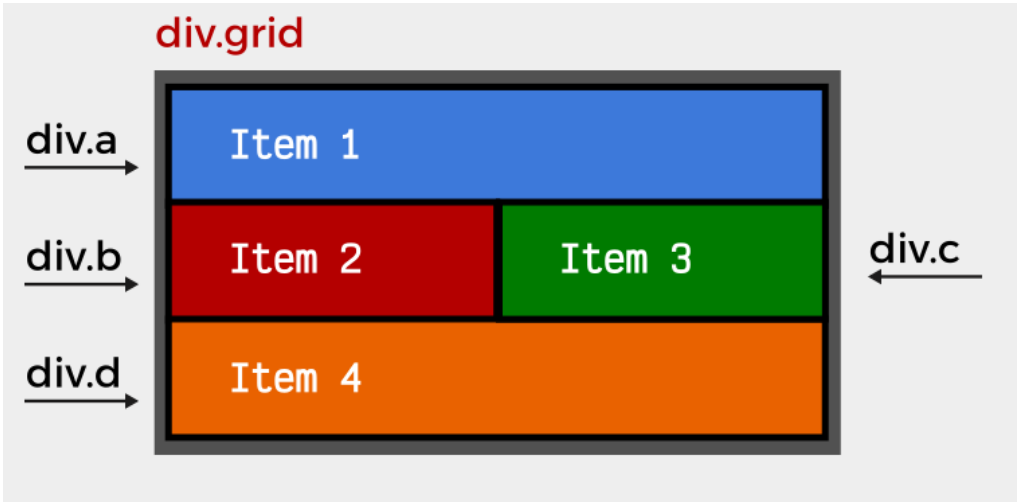
- Zona **.header** desde la columna x0 a x3.
- Zona **.sidebar** desde la fila y1 a y2.
- Zona **.content** desde la columna x1 a x3 y desde la fila y1 a y3.
- Zona **.footer** desde la columna x0 a x3 en la fila y2.

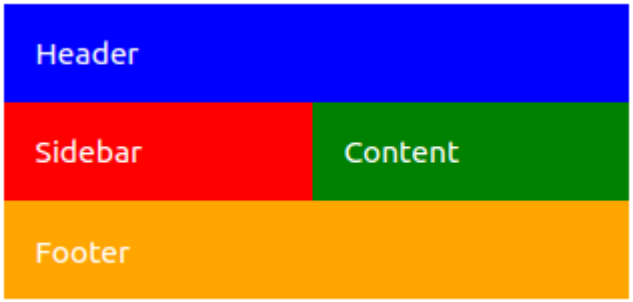
Grid

Por áreas

```
.grid {
   display: grid;
   grid-template-areas: "head head"
                        "menu main"
                        "foot foot";
}
.a { grid-area: head; background: blue }
.b { grid-area: menu; background: red }
.c { grid-area: main; background: green }
.d { grid-area: foot; background: orange }
```

```
<div class="grid">
   <div class="a">Header</div>
   <div class="b">Sidebar</div>
   <div class="c">Content</div>
   <div class="d">Footer</div>
</div>
```

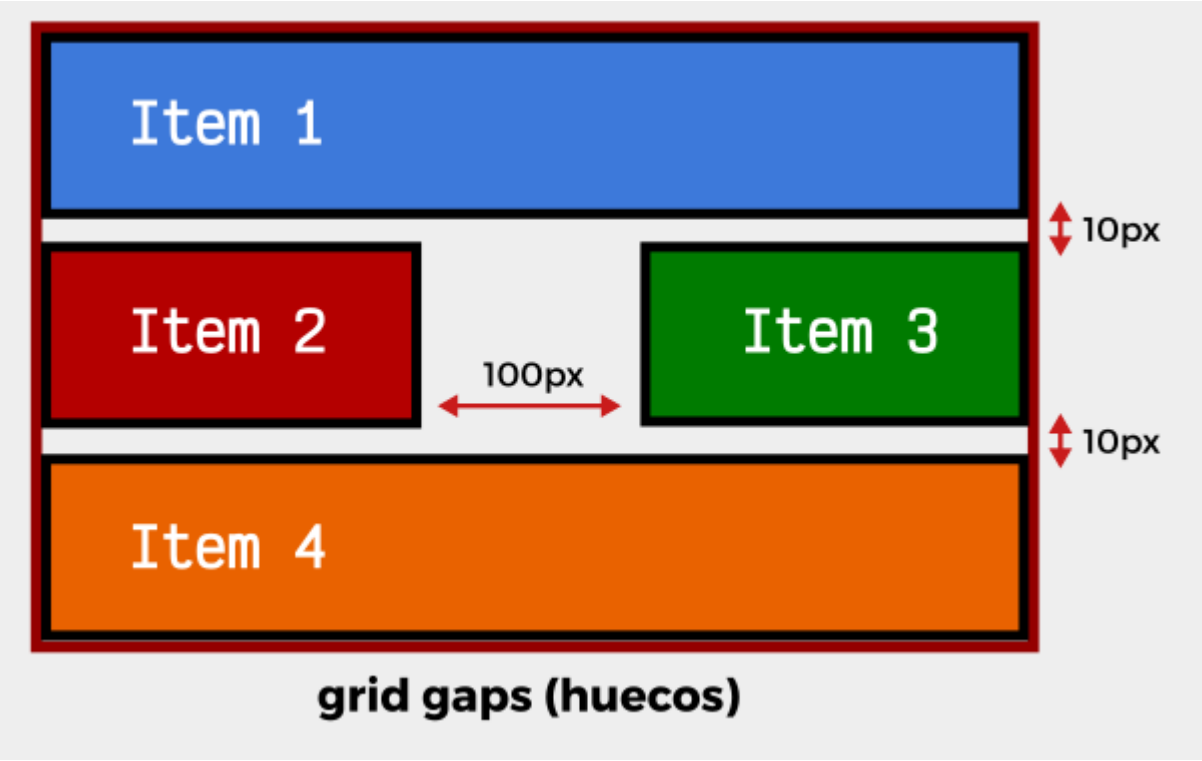




Grid

Huecos

```
.grid {
   column-gap: 100px;
   row-gap: 10px;
}
```



grid gaps (huecos)

Grid

code URJC

Posición

- **justify-items:** Distribuye los elementos en el eje horizontal (start | end | center | stretch)
- **align-items:** Distribuye los elementos en el eje vertical (start | end | center | stretch)

Distribución

- **justify-content:** start | end | center | stretch | space-around | space-between | space-evenly
- **align-content:** start | end | center | stretch | space-around | space-between | space-evenly

Grid

Elementos hijos

- **Grid-column-start:** Indica en que columna empezará el ítem de la cuadrícula.
- **grid-column-end:** Indica en que columna terminará el ítem de la cuadrícula.
- **grid-row-start:** Indica en que fila empezará el ítem de la cuadrícula.
- **grid-row-end:** Indica en que fila terminará el ítem de la cuadrícula.

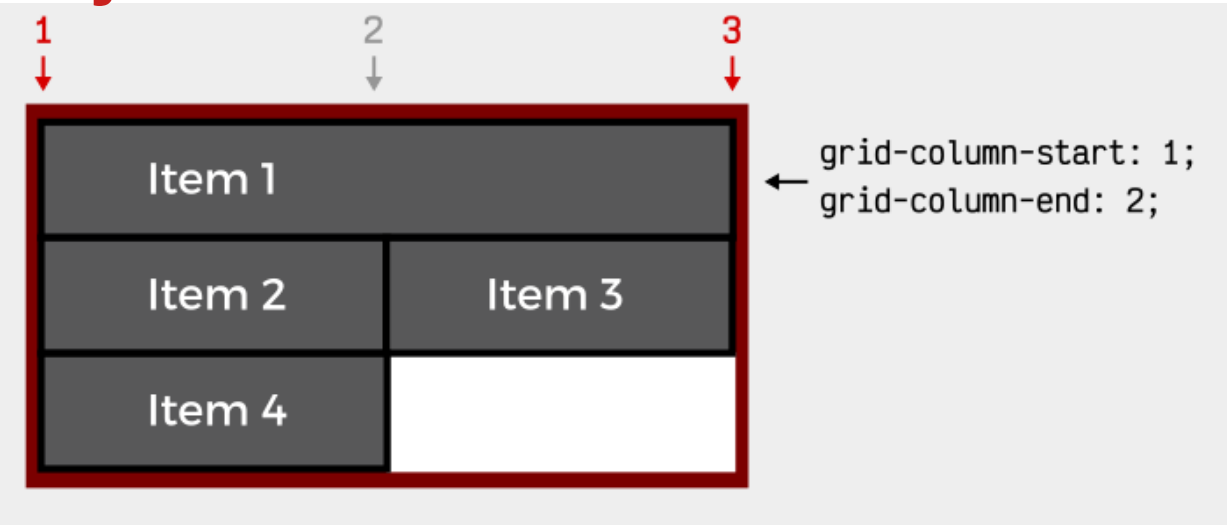
Grid

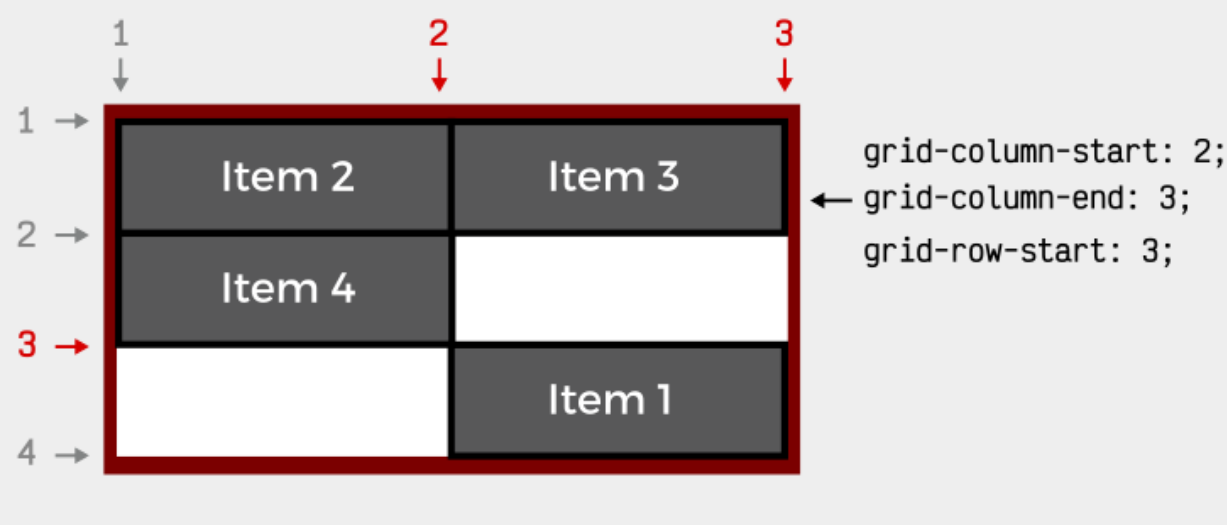
Elementos hijos


```
.grid {
   display:grid;
}
.a {
  grid-column-start: 1;
  grid-row-end: 2;
}
```

Grid

Elementos hijos



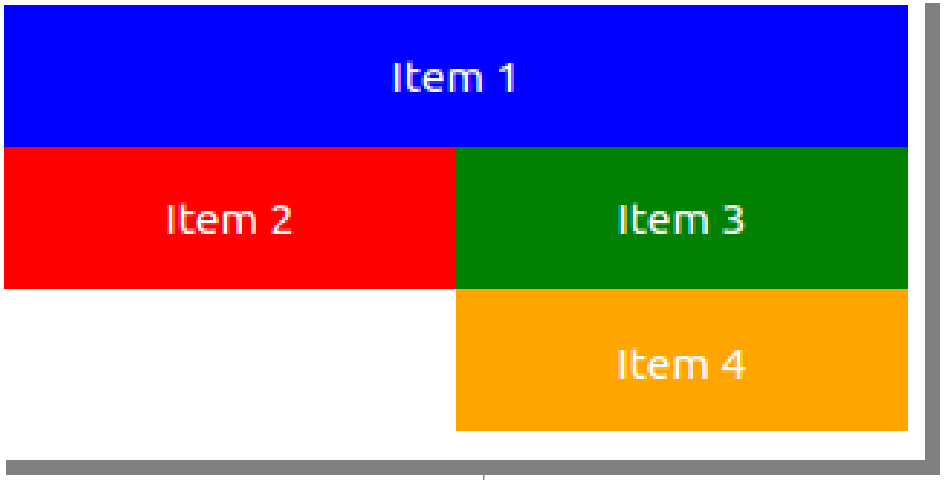


Ejercicio 3

- Trata de replicar la siguiente figura utilizando solo CSS (Grid)

```html
<div class="grid">
   <div class="item a">Item 1</div>
   <div class="item b">Item 2</div>
   <div class="item c">Item 3</div>
   <div class="item d">Item 4</div>
</div>
```

```css
.grid {
   display: grid;
}
.item{
   color: white;
   padding: 1em;
   text-align: center;
}
.a { background: blue; }
.b { background: red; }
.c { background: green; }
.d { background: orange; }
```



Grid

- Herramienta para aprender *Grid*

https://cssgridgarden.com/#es

Librerías

code URJC

- Diseñar una página completa editando el **CSS** desde cero es una tarea de **muy bajo nivel**
- Los diseñadores gráficos se especializan en **maquetación web** para crear **diseños** que los programadores podamos usar
- Existen muchas **librerías de componentes CSS** con un diseño profesional que podemos usar para crear nuestra web

Librerías

code URJC

- Algunas de las librerías más famosas

Bootstrap

http://getbootstrap.com/

http://semantic-ui.com/

Foundation
Start here, build everywhere.

http://foundation.zurb.com/

08-2023

code URJC

Fundamentos de la Web

Bloque II: Tecnologías de cliente web

Tema 2.3: Bootstrap

Universidad Rey Juan Carlos

Micael Gallego
Correo: micael.gallego@urjc.es
Twitter: @micael_gallego

Nicolás Rodríguez
Correo: nicolas.rodriguez@urjc.es

Iván Chicano
ivan.chicano@urjc.es

Michel Maes
michel.maes@urjc.es

08-2023

code
URJC

Micael Gallego, Nicolás Rodríguez, Iván Chicano, Michel Maes

Definición

code URJC

- Bootstrap es un *framework* de software libre (licencia MIT) para el diseño del lado cliente de aplicaciones web
- Fue creado en 2011 por desarrolladores de Twitter con el objetivo de fomentar la coherencia de las herramientas internas de la compañía
- Ventajas de Bootstrap:
  - Fácil de usar
  - Permite la creación de aplicaciones web adaptables (*responsive*)
  - Orientado a dispositivos móviles en primer lugar (*mobile-first*)
  - Proporciona compatibilidad entre los principales navegadores (Chrome, Firefox, Internet Explorer, Safari y Opera)

Definición

- La versión actual (septiembre de 2023): 5.3.1
- Una forma de usar Bootstrap es enlazando de la CDN (Content Delivery Network): http://www.bootstrapcdn.com/
- Para usar Bootstrap en nuestra página web insertamos las siguientes etiquetas:

Importamos el fichero **boostrap.min.css**, que incluye todas las reglas de estilo de Boostrap

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js"></script>
```

Importamos el fichero **boostrap.bundle.min.js**, que incluye código JavaScript que aporta interactividad con *algunos* componentes

¿Qué nos ofrece Boostrap?

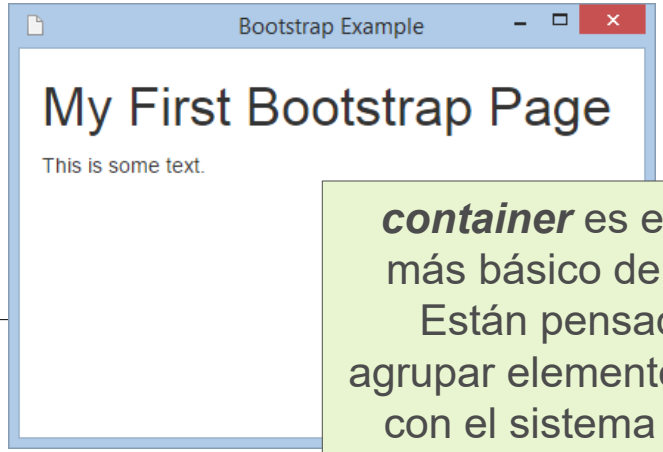
code URJC

- Layout
  - Containers
  - Grid
- Contenido
  - Tipografía
  - Tablas
  - Imágenes
- Formularios
- Componentes
  - Botones
  - Iconos
  - Etiquetas
  - Barras de progreso
  - Paginación
  - Listas
  - Navegación

Layout: Containers

```
<!DOCTYPE html>
<html lang="en">

<head>
<title>Bootstrap Example</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js"></script>
</head>

<body>
  <div class="container">
    <h1>My First Bootstrap Page</h1>
    <p>This is some text.</p>
  </div>
</body>

</html>
```

Para que la página se ajuste al ancho de pantalla del cliente



container es el elemento más básico de Boostrap. Están pensados para agrupar elementos y trabajar con el sistema de grid de Boostrap

Layout: Grid

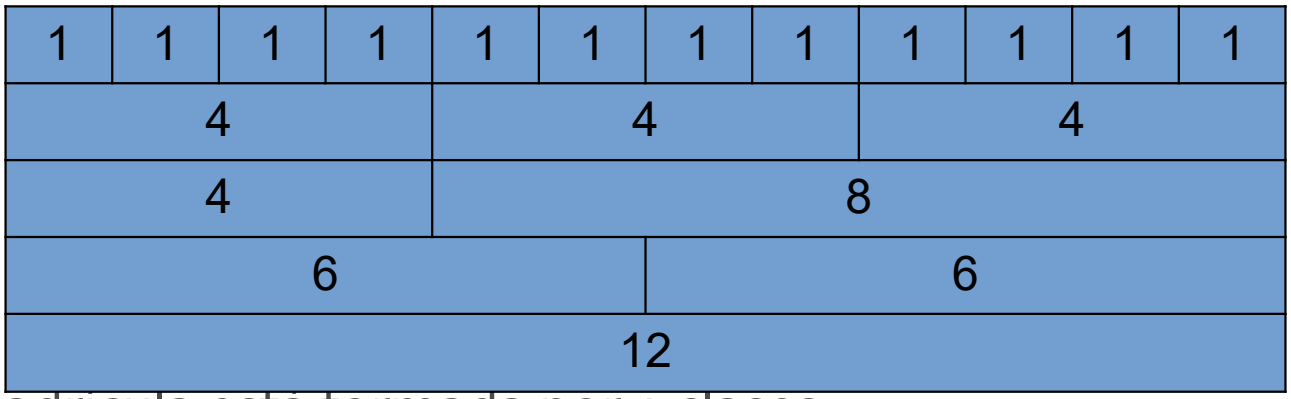
code URJC

- En Bootstrap la disposición de elementos dentro de una página (*layout*) se posiciona mediante una cuadrícula (**grid**)
- Esta cuadrícula se organiza en base a **filas** y **columnas**
  - Las filas (*rows*) deben ir contenidas siempre en un contenedor, que puede ser de dos tipos:
    - `class="container"` : ancho fijo
    - `class="container-fluid"` : ancho total
  - Las columnas (*cols*) van dentro de las filas

```
<div class="container">
  <div class="row">
    <div class="col-*-*"></div>
  </div>
  <div class="row">
    <div class="col-*-*"></div>
    <div class="col-*-*"></div>
    <div class="col-*-*"></div>
  </div>
  <div class="row">
    ...
  </div>
</div>
```

Layout: Grid

- La cuadrícula se divide en 12 columnas:



- La cuadrícula está formada por 4 clases:
  - xs: para teléfonos
  - sm: para tabletas
  - md: para escritorios
  - lg: para escritorios grandes

Este sistema es adaptable (*responsive*) en Bootstrap, de modo que se organizará de forma relativa al tamaño de la pantalla

Layout: Grid

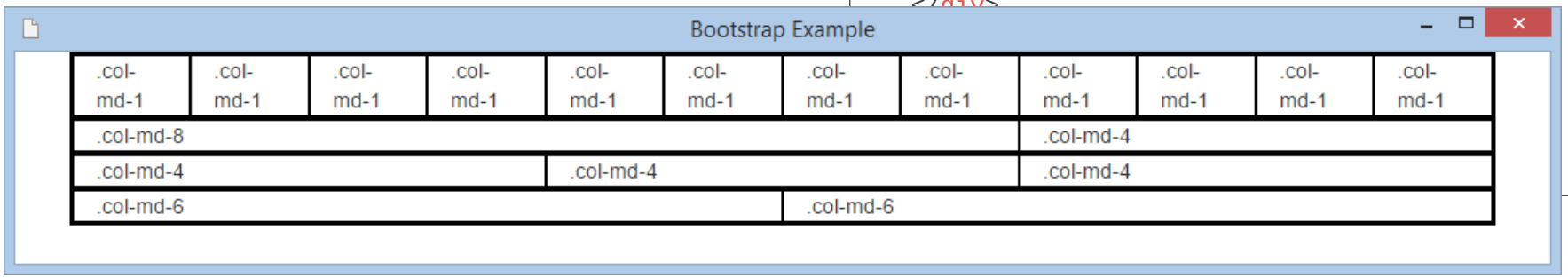
- Ejemplo de HTML usando diferentes tamaños de la rejilla de bootstrap

```
<div class="container">
   <div class="row">
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
   </div>
   <div class="row">
      <div class="col-md-8">.col-md-8</div>
      <div class="col-md-4">.col-md-4</div>
   </div>
   <div class="row">
      <div class="col-md-4">.col-md-4</div>
      <div class="col-md-4">.col-md-4</div>
      <div class="col-md-4">.col-md-4</div>
   </div>
   <div class="row">
      <div class="col-md-6">.col-md-6</div>
      <div class="col-md-6">.col-md-6</div>
      </div>
</div>
```

Layout: Grid

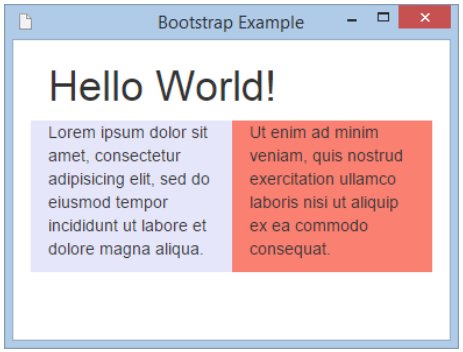
- Ejemplo de HTML usando diferentes tamaños de la rejilla de bootstrap

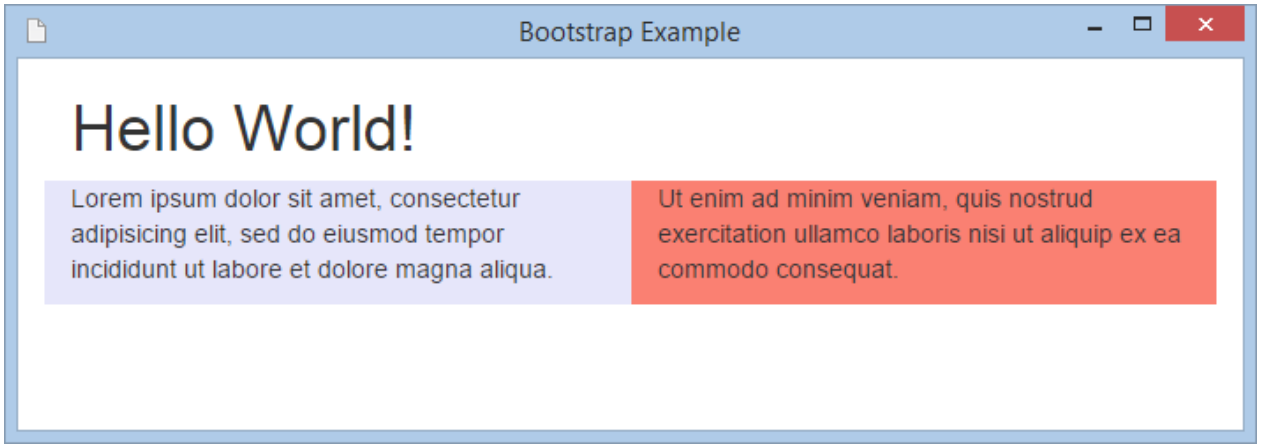
```
<div class="container">
   <div class="row">
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
      <div class="col-md-1">.col-md-1</div>
   </div>
   <div class="row">
      <div class="col-md-8">.col-md-8</div>
      <div class="col-md-4">.col-md-4</div>
   </div>
   <div class="row">
      <div class="col-md-4">.col-md-4</div>
      <div class="col-md-4">.col-md-4</div>
      <div class="col-md-4">.col-md-4</div>
```



Layout: Grid

```
<div class="container">
   <h1>Hello World!</h1>
   <div class="row">
     <div class="col-xs-6" style="background-color:lavender;">
        <p>Lorem ipsum ...</p>
     </div>
     <div class="col-xs-6" style="background-color:salmon;">
        <p>Ut enim ...</p>
     </div>
   </div>
</div>
```





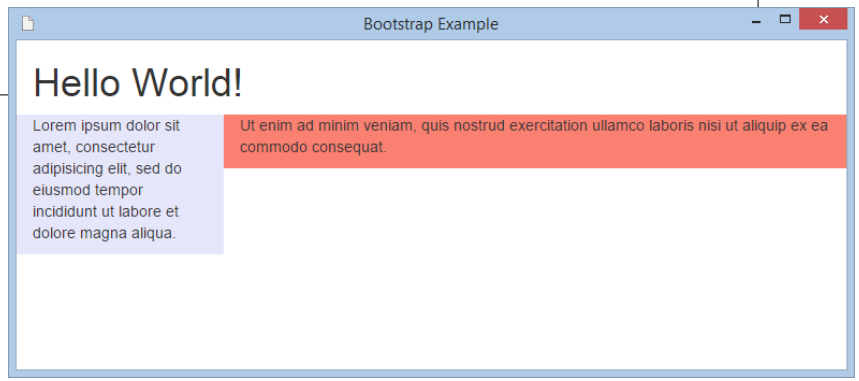
Layout: Grid

```
<div class="container-fluid">
   <h1>Hello World!</h1>
   <div class="row">
      <div class="col-sm-3 col-md-6" style="background-color:lavender;">
         <p>Lorem ipsum ...</p>
      </div>
      <div class="col-sm-9 col-md-6" style="background-color:salmon;">
         <p>Ut enim ...</p>
      </div>
   </div>
</div>
```

Hello World!

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

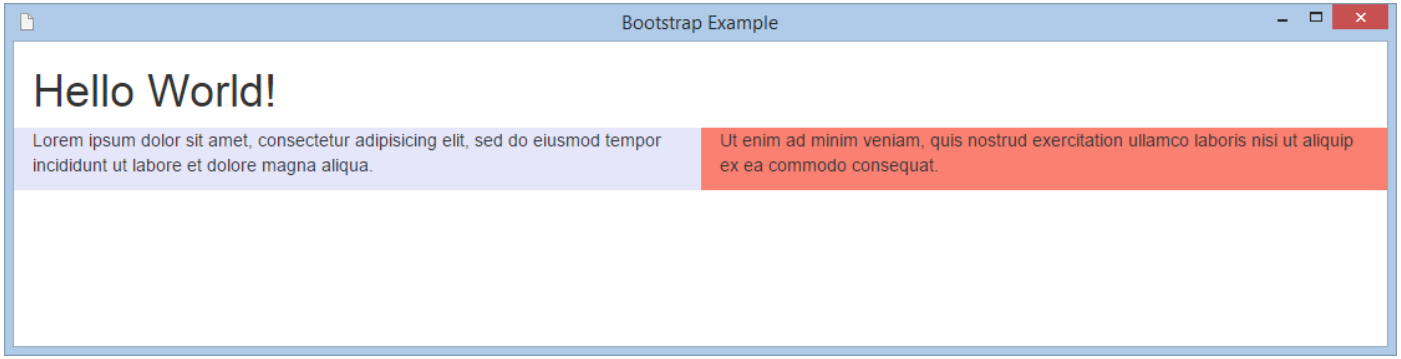
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.



Hello World!

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.



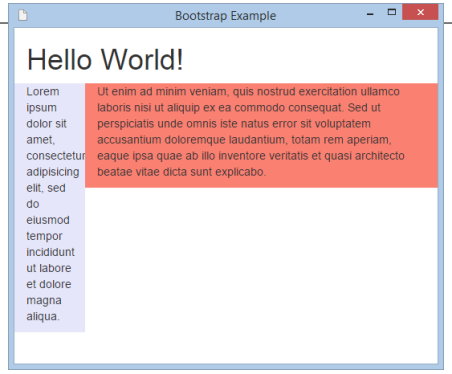
Hello World!

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Layout: Grid

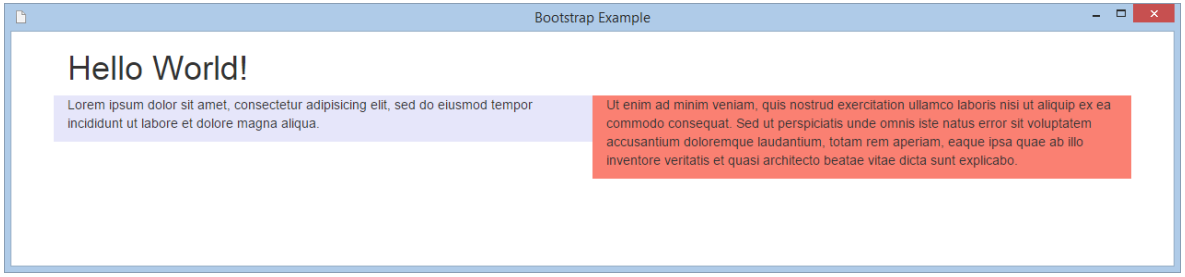
```
<div class="container">
   <h1>Hello World!</h1>
   <div class="row">
      <div class="col-xs-2 col-sm-3 col-md-4 col-lg-6" style="background-color: lavender;">
         <p>Lorem ipsum ...</p>
      </div>
      <div class="col-xs-10 col-sm-9 col-md-8 col-lg-6" style="background-color: salmon;">
         <p>Ut enim ...</p>
      </div>
   </div>
</div>
```



Hello World!

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

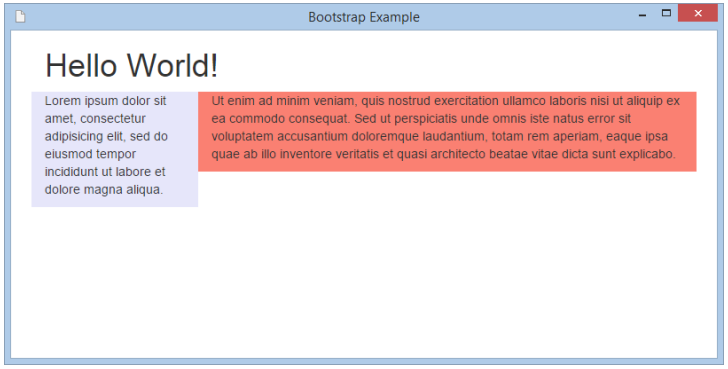
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.



Hello World!

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

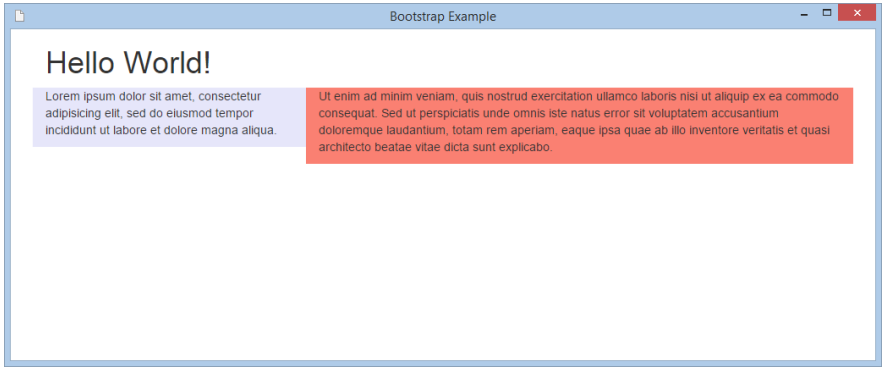
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.



Hello World!

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.



Hello World!

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Layout: Grid

code URJC

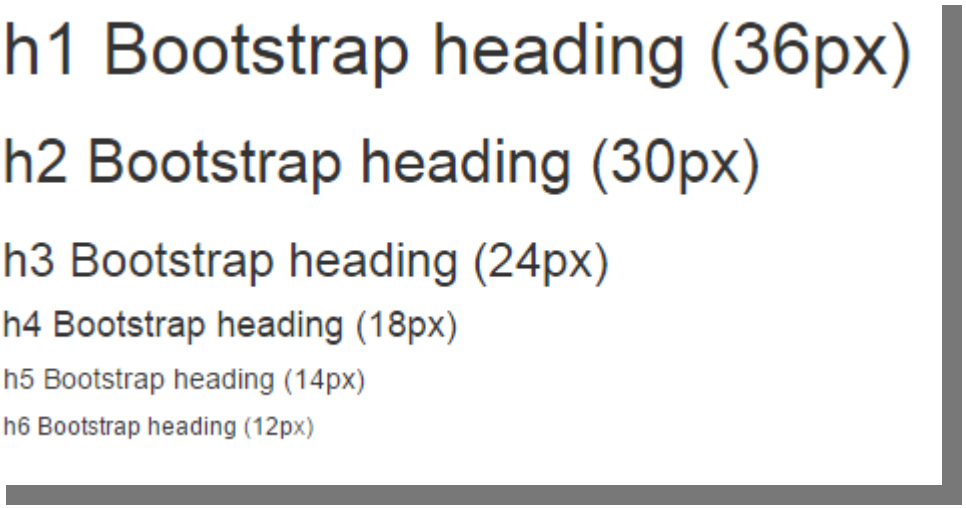
- Existen herramientas online para ver nuestra aplicación web en diferentes dispositivos/pantallas:
  - http://quirktools.com/screenfly/
  - http://www.viewlike.us/
  - http://www.infobyip.com/testwebsiteresolution.php

Contenido: Tipografía

code URJC

- Cabeceras
  - Etiquetas **h1** a **h6**

```
<div class="container">
   <h1>h1 Bootstrap heading (36px)</h1>
   <h2>h2 Bootstrap heading (30px)</h2>
   <h3>h3 Bootstrap heading (24px)</h3>
   <h4>h4 Bootstrap heading (18px)</h4>
   <h5>h5 Bootstrap heading (14px)</h5>
   <h6>h6 Bootstrap heading (12px)</h6>
</div>
```



Contenido: Tipografía

- Titulares
  - Clase **display-***:

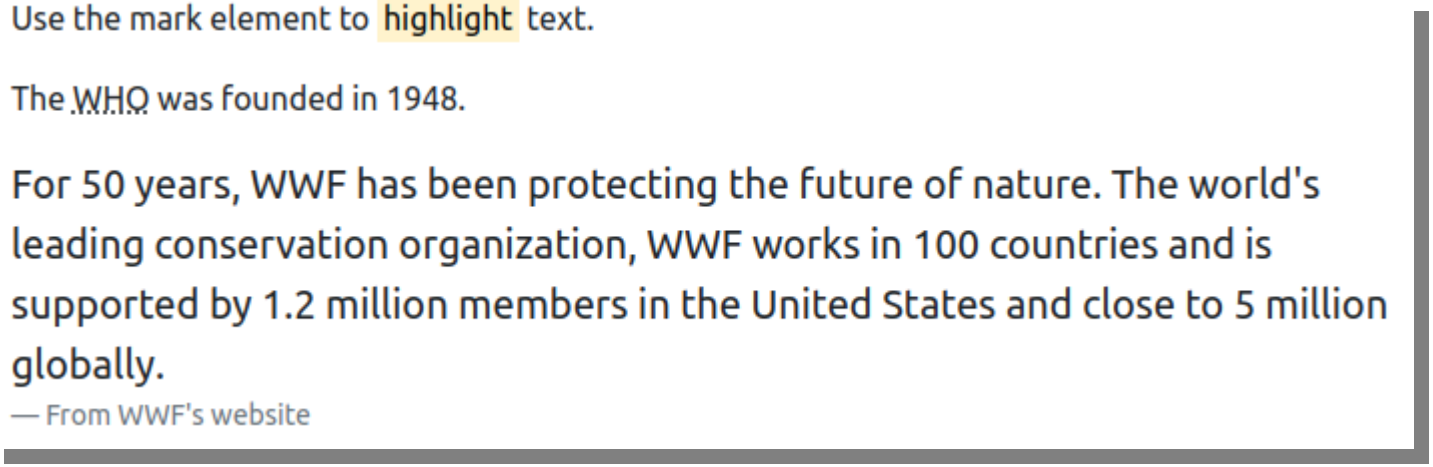
```
<div class="container">
   <h1 class="display-1">Display 1</h1>
   <h1 class="display-2">Display 2</h1>
   <h1 class="display-3">Display 3</h1>
   <h1 class="display-4">Display 4</h1>
   <h1 class="display-5">Display 5</h1>
   <h1 class="display-6">Display 6</h1>
</div>
```

Contenido: Tipografía

code URJC

- Efectos de texto
  - Etiquetas **mark**, **abbr**, **blockquote**, **footer**:

```
<p>Use the mark element to <mark>highlight</mark> text.</p>
<p>The <abbr title="World Health Organization">WHO</abbr> was founded in 1948.</p>
<figure>
   <blockquote class="blockquote">
      <p>For 50 years, WWF has been protecting the future of nature. The world's leading
conservation Organization, WWF works in 100 countries and is supported by 1.2 million members in the
United States and close to 5 million globally.</p>
   </blockquote>
   <figcaption class="blockquote-footer">From WWF's website</figcaption>
</figure>
```

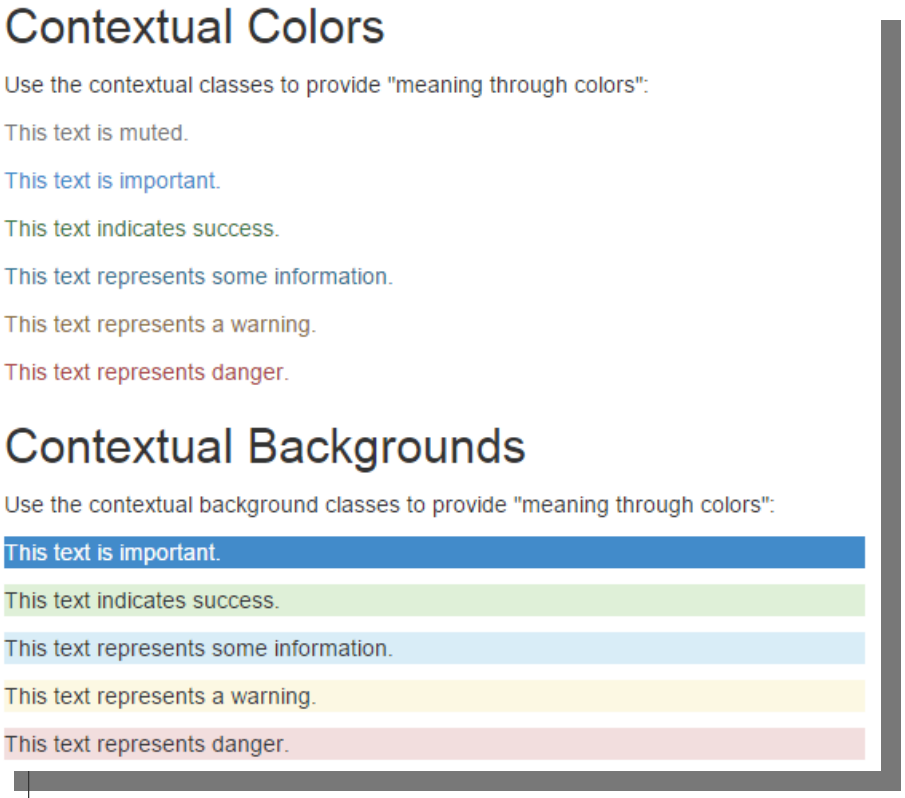


Contenido: Tipografía

code URJC

- Efectos de color
  - Clases **text-*** y **bg-***:

```
<div class="container">
   <h2>Contextual Colors</h2>
   <p>Use the contextual classes to provide "meaning through
colors":</p>
   <p class="text-muted">This text is muted.</p>
   <p class="text-primary">This text is important.</p>
   <p class="text-success">This text indicates success.</p>
   <p class="text-info">This text represents some
information.</p>
   <p class="text-warning">This text represents a warning.</p>
   <p class="text-danger">This text represents danger.</p>
   <h2>Contextual Backgrounds</h2>
   <p>Use the contextual background classes to provide "meaning
through colors":</p>
   <p class="bg-primary">This text is important.</p>
   <p class="bg-success">This text indicates success.</p>
   <p class="bg-info">This text represents some information.</p>
   <p class="bg-warning">This text represents a warning.</p>
   <p class="bg-danger">This text represents danger.</p>
</div>
```



Contenido: Tipografía

code URJC

- Efectos de código
  - Etiquetas **kbd** y **pre**:

```
<p>Use <kbd>ctrl + p</kbd> to open the Print
dialog box.</p>
<p>For multiple lines of code, use the pre
element:</p>
<pre>
Text in a pre element
is displayed in a fixed-width
font, and it preserves
both spaces and
line breaks.
</pre>
```

Use `ctrl + p` to open the Print dialog box.

For multiple lines of code, use the pre element:

```
Text in a pre element
is displayed in a fixed-width
font, and it preserves
both      spaces and
line breaks.
```

- Más efectos de texto:

  http://www.w3schools.com/bootstrap/bootstrap_ref_css_text.asp

Contenido: Tablas

code URJC

- Tabla básica
  - Clase **table**:

Basic Table

The .table class adds basic styling (light padding and only horizontal dividers) to a table:

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

```
<h2>Basic Table</h2>
<p>The .table class adds basic styling
(light padding and only horizontal
dividers) to a table:</p>
<table class="table">
   <thead>
      <tr>
         <th>Firstname</th>
         <th>Lastname</th>
         <th>Email</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>John</td>
         <td>Doe</td>
         <td>john@example.com</td>
      </tr>
      <tr>
         <td>Mary</td>
         <td>Moe</td>
         <td>mary@example.com</td>
      </tr>
      <tr>
         <td>July</td>
         <td>Dooley</td>
         <td>july@example.com</td>
      </tr>
   </tbody>
</table>
```

Contenido: Tablas

- Tabla estilo cebra
  - Clase **table-striped**:

Striped Rows

The .table-striped class adds zebra-stripes to a table:

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

```
<h2>Striped Rows</h2>
<p>The .table-striped class adds zebra-stripes to a table:</p>
<table class="table table-striped">
   <thead>
      <tr>
         <th>Firstname</th>
         <th>Lastname</th>
         <th>Email</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>John</td>
         <td>Doe</td>
         <td>john@example.com</td>
      </tr>
      <tr>
         <td>Mary</td>
         <td>Moe</td>
         <td>mary@example.com</td>
      </tr>
      <tr>
         <td>July</td>
         <td>Dooley</td>
         <td>july@example.com</td>
      </tr>
   </tbody>
</table>
```

Contenido: Tablas

code URJC

- Tabla estilo con borde
  - Clase **table-bordered**:

Bordered Table

The .table-bordered class adds borders to a table:

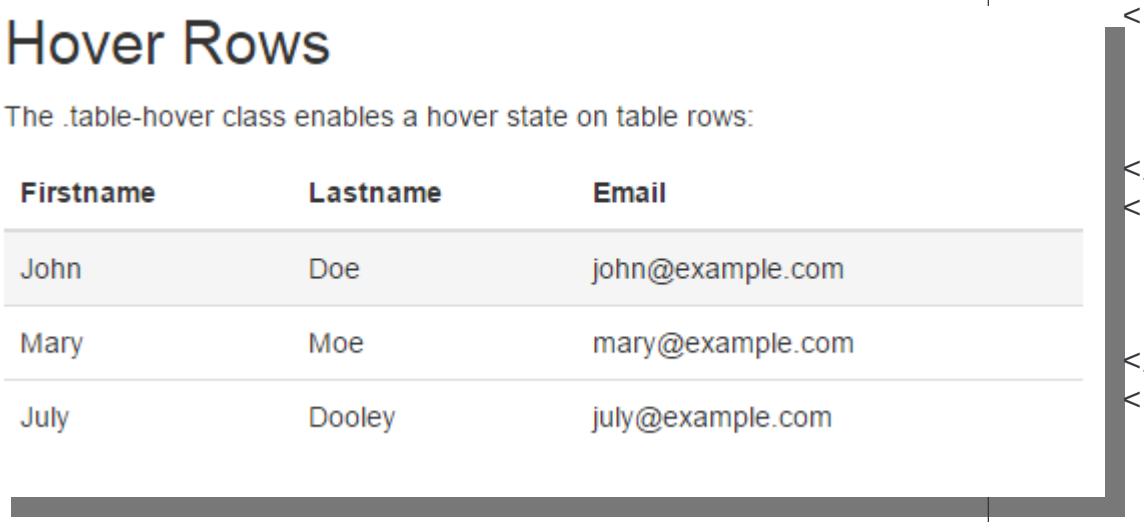
Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

```
<h2>Bordered Table</h2>
<p>The .table-bordered class adds
borders to a table:</p>
<table class="table table-bordered">
   <thead>
      <tr>
         <th>Firstname</th>
         <th>Lastname</th>
         <th>Email</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>John</td>
         <td>Doe</td>
         <td>john@example.com</td>
      </tr>
      <tr>
         <td>Mary</td>
         <td>Moe</td>
         <td>mary@example.com</td>
      </tr>
      <tr>
         <td>July</td>
         <td>Dooley</td>
         <td>july@example.com</td>
      </tr>
   </tbody>
</table>
```

Contenido: Tablas

code URJC

- Tabla con efecto *roll-over*
  - Clase **table-hover**:



Hover Rows

The .table-hover class enables a hover state on table rows:

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

```
<h2>Hover Rows</h2>
<p>The .table-hover class enables a
hover state on table rows:</p>
<table class="table table-hover">
   <thead>
      <tr>
         <th>Firstname</th>
         <th>Lastname</th>
         <th>Email</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>John</td>
         <td>Doe</td>
         <td>john@example.com</td>
      </tr>
      <tr>
         <td>Mary</td>
         <td>Moe</td>
         <td>mary@example.com</td>
      </tr>
      <tr>
         <td>July</td>
         <td>Dooley</td>
         <td>july@example.com</td>
      </tr>
   </tbody>
</table>
```

Contenido: Tablas

- Tabla condensada
  - Clase **table-condensed**:

Condensed Table

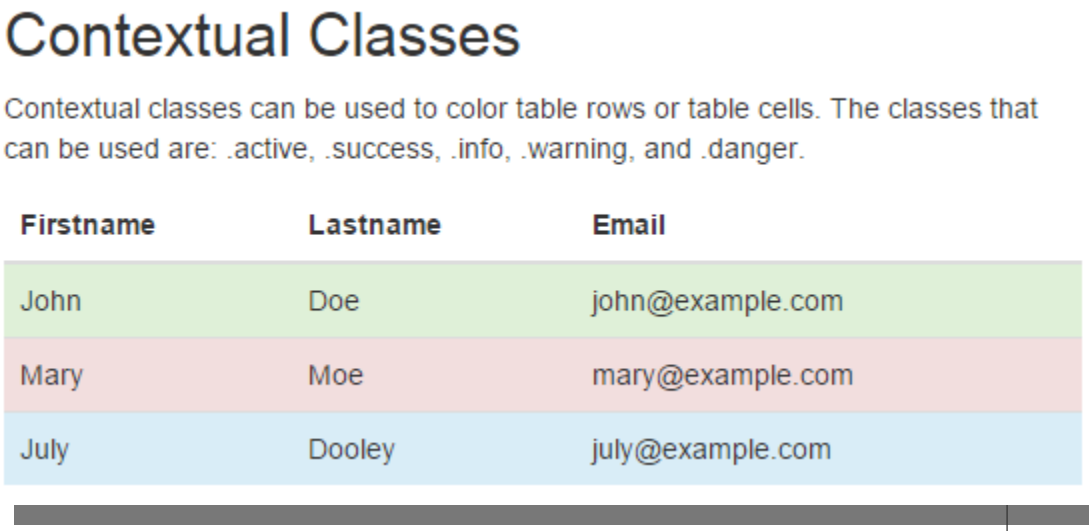
The .table-condensed class makes a table more compact by cutting cell padding in half:

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

```
<h2>Condensed Table</h2>
<p>The .table-condensed class makes a
table more compact by cutting cell
padding in half:</p>
<table class="table table-condensed">
   <thead>
      <tr>
         <th>Firstname</th>
         <th>Lastname</th>
         <th>Email</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>John</td>
         <td>Doe</td>
         <td>john@example.com</td>
      </tr>
      <tr>
         <td>Mary</td>
         <td>Moe</td>
         <td>mary@example.com</td>
      </tr>
      <tr>
         <td>July</td>
         <td>Dooley</td>
         <td>july@example.com</td>
      </tr>
   </tbody>
</table>
```

Contenido: Tablas

- Tabla con color de fondo
  - Clases de fila **success, danger e info**:



Contextual Classes

Contextual classes can be used to color table rows or table cells. The classes that can be used are: .active, .success, .info, .warning, and .danger.

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

```
<h2>Contextual Classes</h2>
<p>Contextual classes can be used to
color table rows or table cells. The
classes that can be used are: .active,
.success, .info, .warning,
and .danger.</p>
<table class="table table-condensed">
   <thead>
      <tr>
         <th>Firstname</th>
         <th>Lastname</th>
         <th>Email</th>
      </tr>
   </thead>
   <tbody>
      <tr class="success">
         <td>John</td>
         <td>Doe</td>
         <td>john@example.com</td>
      </tr>
      <tr class="danger">
         <td>Mary</td>
         <td>Moe</td>
         <td>mary@example.com</td>
      </tr>
      <tr class="info">
         <td>July</td>
         <td>Dooley</td>
         <td>july@example.com</td>
      </tr>
   </tbody>
</table>
```

Contenido: Imágenes

- Bordes redondeados
  - Clase **img-rounded**:

Rounded Corners

The .img-rounded class adds rounded corners to an image (not available in IE8):

```
<h2>Rounded Corners</h2>
<p>The .img-rounded class adds rounded
corners to an image (not available in
IE8):</p>
<img src="cinqueterre.jpg" class="img-
rounded" alt="Cinque Terre" width="304"
height="236">
```

Contenido: Imágenes

- Marco circular
  - Clase **img-circle**:

Circle

The .img-circle class shapes the image to a circle (not available in IE8):

```
<h2>Circle</h2>
<p>The .img-circle class shapes the image
to a circle (not available in IE8):</p>
<img src="cinqueterre.jpg" class="img-
circle" alt="Cinque Terre" width="304"
height="236">
```

Contenido: Imágenes

- Miniatura
  - Clase **img-thumbnail**:

Thumbnail

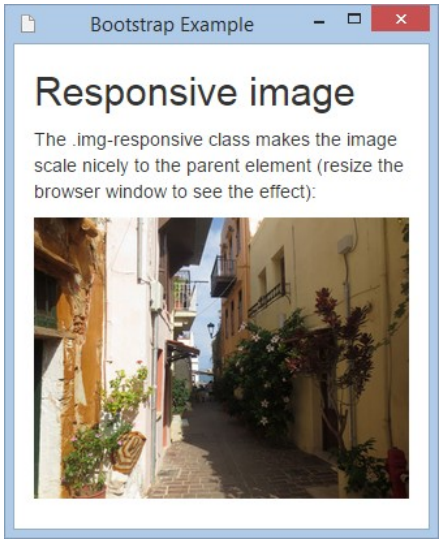
The .img-thumbnail class creates a thumbnail of the image:

```
<h2>Thumbnail</h2>
<p>The .img-thumbnail class creates a
thumbnail of the image:</p>

<img src="cinqueterre.jpg" class="img-
thumbnail" alt="Cinque Terre" width="304"
height="236">
```

Contenido: Imágenes

- Imagen adaptable
  - Clase **img-responsive**:

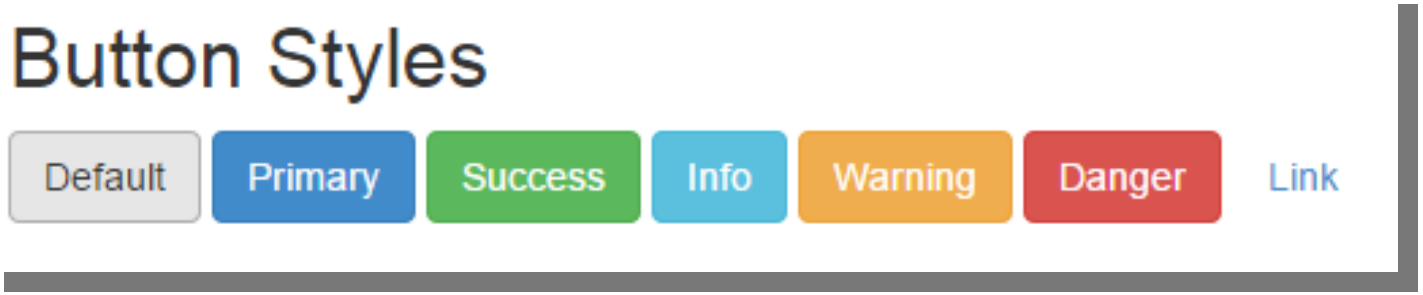

```
<h2>Responsive image</h2>
<p>The .img-responsive class makes the
image scale nicely to the parent element
(resize the browser window to see the
effect):</p>

<img class="img-responsive"
src="img_chania.jpg" alt="Chania"
width="460" height="345">
```



Componentes: Botones

- Estilos de botones
  - Clases **btn, btn-default, btn-primary, btn-success, btn-info, btn-warning, btn-danger, btn-link**:

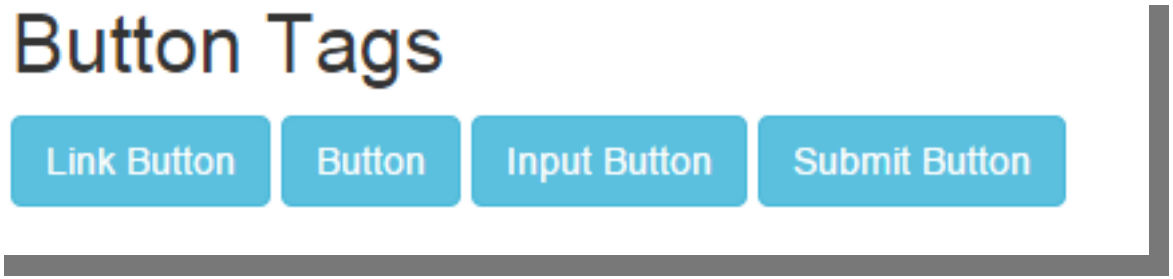
```
<h2>Button Styles</h2>
<button type="button" class="btn btn-default">Default</button>
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-link">Link</button>
```



Componentes: Botones

- Etiquetas usadas para botones
  - Las clases **btn** y **btn-*** se pueden asociar a las etiquetas **a**, **button**, **input type="button"**, **input type="submit"**:

```
<h2>Button Tags</h2>
<a href="#" class="btn btn-info" role="button">Link Button</a>
<button type="button" class="btn btn-info">Button</button>
<input type="button" class="btn btn-info" value="Input Button">
<input type="submit" class="btn btn-info" value="Submit Button">
```



Componentes: Botones

- Tamaño de botones
  - Clases **btn-lg**, **btn-md**, **btn-sm**, **btn-xs**:

```
<h2>Button Sizes</h2>
<button type="button" class="btn btn-primary btn-lg">Large</button>
<button type="button" class="btn btn-primary btn-md">Medium</button>
<button type="button" class="btn btn-primary btn-sm">Small</button>
<button type="button" class="btn btn-primary btn-xs">XSmall</button>
```

Componentes: Botones

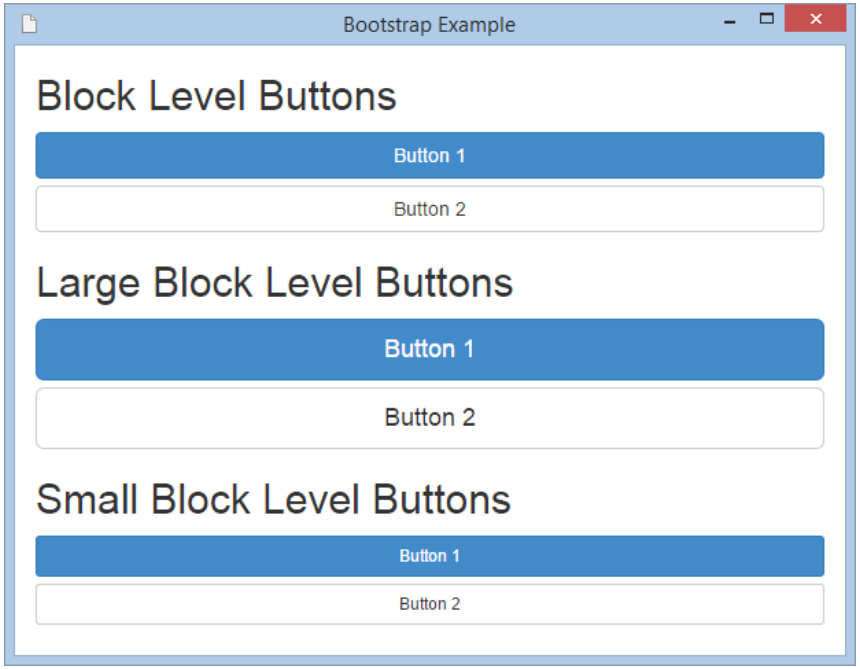
- Estado de botones
  - Clases **active**, **disabled**:

```
<h2>Button States</h2>
<button type="button" class="btn btn-primary">Primary Button</button>
<button type="button" class="btn btn-primary active">Active Primary</button>
<button type="button" class="btn btn-primary disabled">Disabled Primary</button>
```

Componentes: Botones

- Botones tamaño bloque
  - Clase **btn-block**:

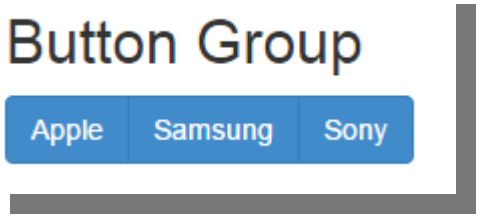
```
<h2>Block Level Buttons</h2>
<button type="button" class="btn btn-primary
btn-block">Button 1</button>
<button type="button" class="btn btn-default
btn-block">Button 2</button>

<h2>Large Block Level Buttons</h2>
<button type="button" class="btn btn-primary
btn-lg btn-block">Button 1</button>
<button type="button" class="btn btn-default
btn-lg btn-block">Button 2</button>

<h2>Small Block Level Buttons</h2>
<button type="button" class="btn btn-primary
btn-sm btn-block">Button 1</button>
<button type="button" class="btn btn-default
btn-sm btn-block">Button 2</button>
```



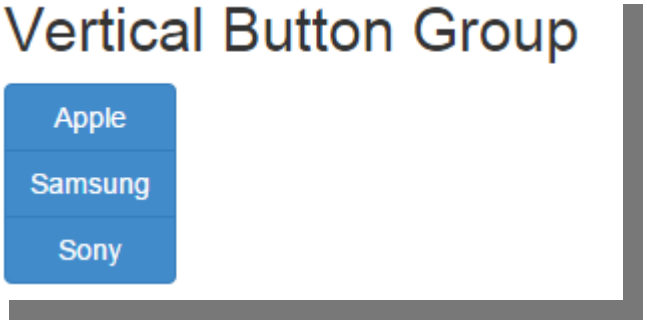
Componentes: Botones

- Grupos de botones
  - Clases **btn-group** y **btn-group-vertical**:

```
<h2>Button Group</h2>
<div class="btn-group">
<button type="button" class="btn btn-primary">Apple</button>
<button type="button" class="btn btn-primary">Samsung</button>
<button type="button" class="btn btn-primary">Sony</button>
</div>
```




```
<h2>Vertical Button Group</h2>
<div class="btn-group-vertical">
<button type="button" class="btn btn-primary">Apple</button>
<button type="button" class="btn btn-primary">Samsung</button>
<button type="button" class="btn btn-primary">Sony</button>
</div>
```

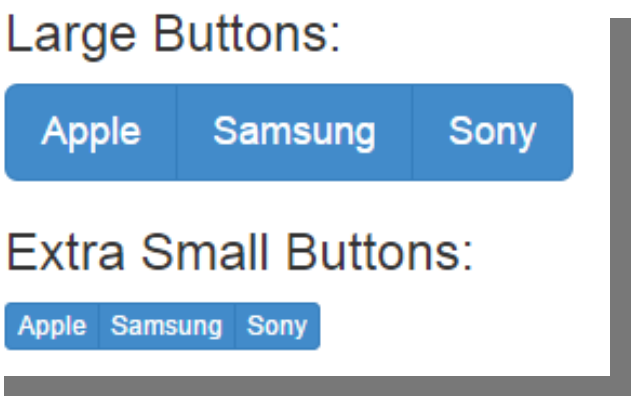


Componentes: Botones

code URJC

- Tamaño del grupos de botones
  - Clases **btn-group-lg** y **btn-group-xs**:

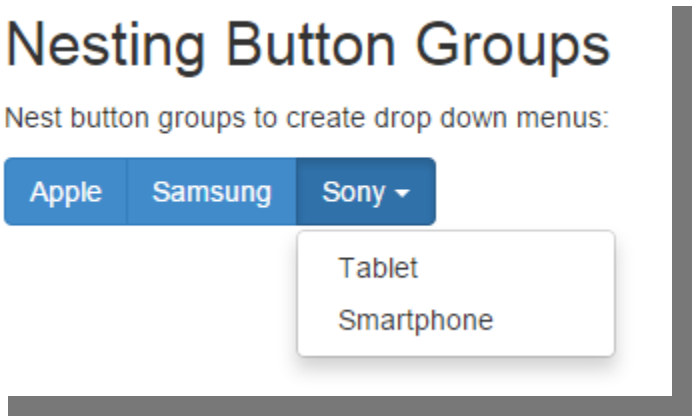
```
<h2>Button Groups - Set Sizes</h2>
<p>Add class .btn-group-* to size all buttons in a button group.</p>
<h3>Large Buttons:</h3>
<div class="btn-group btn-group-lg">
   <button type="button" class="btn btn-primary">Apple</button>
   <button type="button" class="btn btn-primary">Samsung</button>
   <button type="button" class="btn btn-primary">Sony</button>
</div>
<h3>Extra Small Buttons:</h3>
<div class="btn-group btn-group-xs">
   <button type="button" class="btn btn-primary">Apple</button>
   <button type="button" class="btn btn-primary">Samsung</button>
   <button type="button" class="btn btn-primary">Sony</button>
</div>
```



Componentes: Botones

- Botones anidados
  - Clases **dropdown-toggle** y **dropdown**:

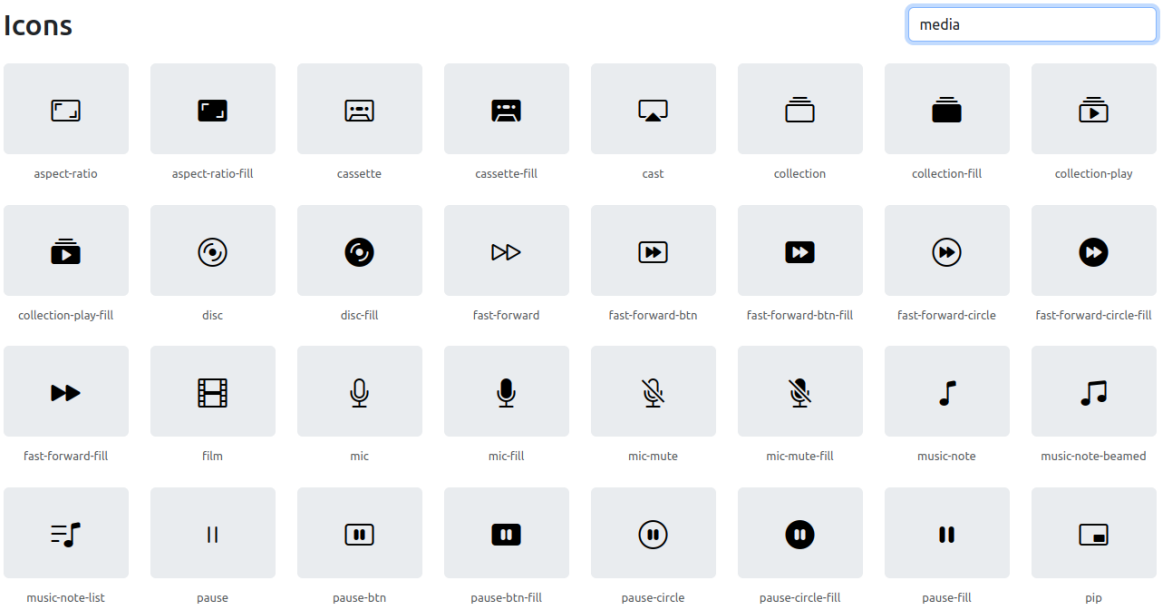
```
<h2>Nesting Button Groups</h2>
<p>Nest button groups to create drop down menus:</p>
<div class="btn-group">
   <button type="button" class="btn btn-primary">Apple</button>
   <button type="button" class="btn btn-primary">Samsung</button>
   <div class="btn-group">
      <button type="button" class="btn btn-primary dropdown-toggle"  data-toggle="dropdown">
         Sony <span class="caret"></span>
      </button>
      <ul class="dropdown-menu" role="menu">
         <li><a href="#">Tablet</a></li>
         <li><a href="#">Smartphone</a></li>
      </ul>
   </div>
</div>
```



Componentes: Iconos

- Bootstrap proporciona 1800 iconos libres
- Se pueden insertar iconos en texto, botones, formularios, etc
- Es necesario importarlos aparte:

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css">
```

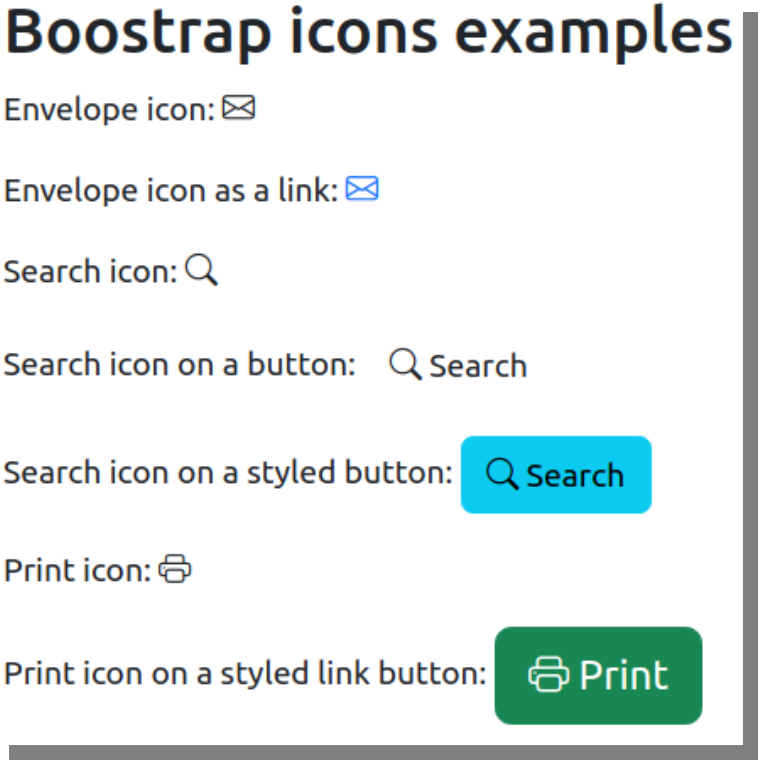


https://icons.getbootstrap.com/

Componentes: Iconos

- Se insertan las clases utilizando **bi bi-***

```
<h2>Boostrap icons examples</h2>
<p>Envelope icon: <i class="bi bi-envelope"></i> </p>
<p>Envelope icon as a link:
   <a href="#"><i class="bi bi-envelope"></i></a>
</p>
<p>Search icon: <i class="bi bi-search"></i>
<p>Search icon on a button:
   <button type="button" class="btn btn-default">
      <i class="bi bi-search"></i> Search
   </button>
</p>
<p>Search icon on a styled button:
   <button type="button" class="btn btn-info">
      <i class="bi bi-search"></i> Search
   </button>
</p>
<p>Print icon: <i class="bi bi-printer"></i></p>
<p>Print icon on a styled link button:
   <a href="#" class="btn btn-success btn-lg">
      <i class="bi bi-printer"></i> Print
   </a>
</p>
```



Componentes: Etiquetas

code URJC

- Etiquetas de colores
  - Clase **badge**

```
<h1>Example heading <span class="badge bg-primary">New</span></h1>
<h2>Example heading <span class="badge bg-secondary">New</span></h2>
<h3>Example heading <span class="badge bg-success">New</span></h3>
<h4>Example heading <span class="badge bg-danger">New</span></h4>
<h5>Example heading <span class="badge bg-warning">New</span></h5>
<h6>Example heading <span class="badge bg-info">New</span></h6>
```

Componentes: Etiquetas

- Indicadores numéricos
  - Clase **badge**

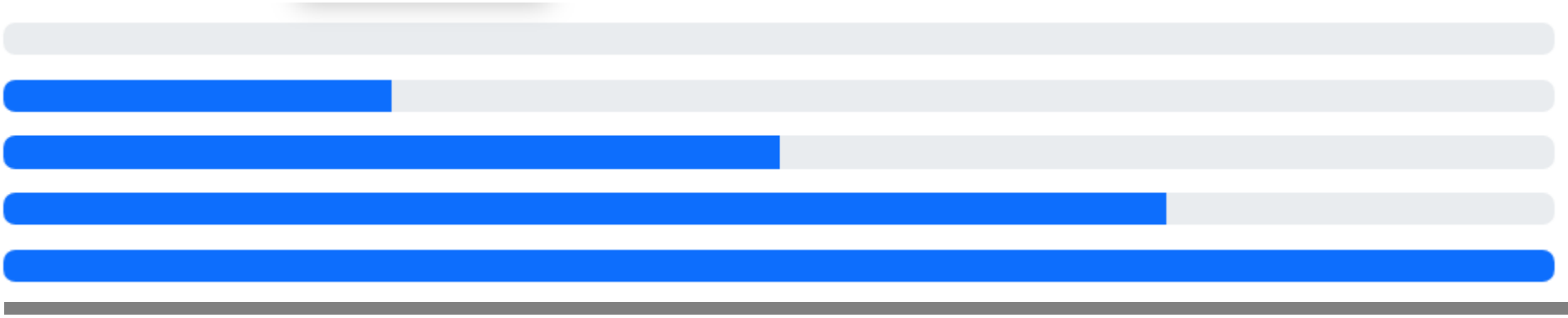
```
<h2>Badges on Buttons</h2>
<button type="button" class="btn btn-primary">Primary <span class="badge">7</span></button>
<button type="button" class="btn btn-success">Success <span class="badge">3</span></button>
<button type="button" class="btn btn-danger">Danger <span class="badge">5</span></button>

<h2>Badges on Buttons - Light style</h2>
<button type="button" class="btn btn-primary">Primary<span class="badge badge text-bg-light">7</span></button>
<button type="button" class="btn btn-success">Success<span class="badge badge text-bg-light">3</span></button>
<button type="button" class="btn btn-danger">Danger<span class="badge badge text-bg-light">5</span></button>
```

Componentes: Barras de progreso

code URJC

- Clases **progress** y **progress-bar** en elementos ** **:

```
<div class="progress" role="progressbar" aria-valuenow="0" aria-valuemin="0" aria-valuemax="100">
   <div class="progress-bar" style="width: 0%"></div>
</div>
<div class="progress" role="progressbar" aria-valuenow="25" aria-valuemin="0" aria-valuemax="100">
   <div class="progress-bar" style="width: 25%"></div>
</div>
<div class="progress" role="progressbar" aria-valuenow="50" aria-valuemin="0" aria-valuemax="100">
   <div class="progress-bar" style="width: 50%"></div>
</div>
<div class="progress" role="progressbar" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100">
   <div class="progress-bar" style="width: 75%"></div>
</div>
<div class="progress" role="progressbar" aria-valuenow="100" aria-valuemin="0" aria-valuemax="100">
   <div class="progress-bar" style="width: 100%"></div>
</div>
```

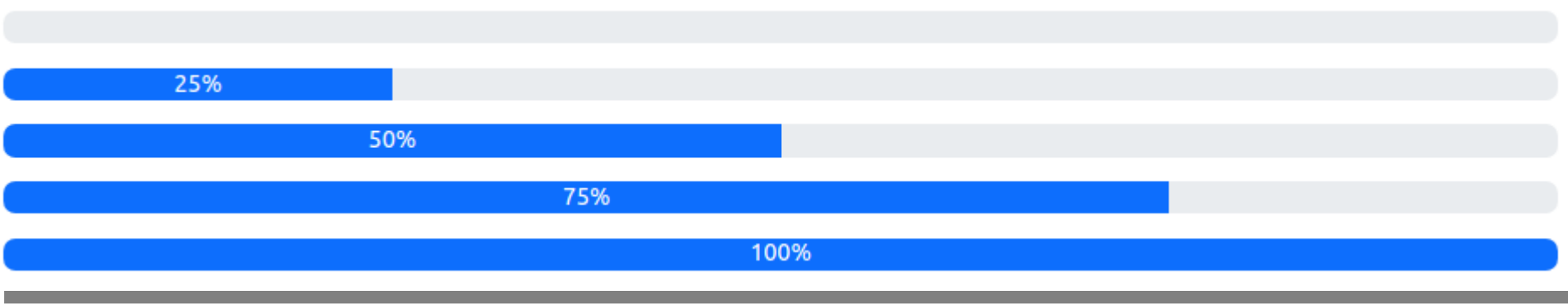


Componentes: Barras de progreso

code URJC

- Barra de progreso con texto

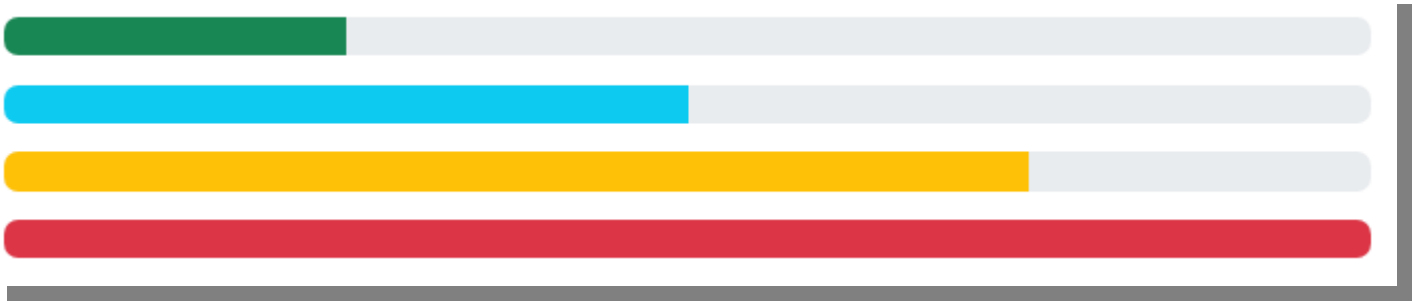
```
<div class="progress" role="progressbar" aria-valuenow="0" aria-valuemin="0" aria-valuemax="100">
   <div class="progress-bar" style="width: 0%"></div>
</div>
<div class="progress" role="progressbar" aria-valuenow="25" aria-valuemin="0" aria-valuemax="100">
   <div class="progress-bar" style="width: 25%">25%</div>
</div>
<div class="progress" role="progressbar" aria-valuenow="50" aria-valuemin="0" aria-valuemax="100">
   <div class="progress-bar" style="width: 50%">50%</div>
</div>
<div class="progress" role="progressbar" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100">
   <div class="progress-bar" style="width: 75%">75%</div>
</div>
<div class="progress" role="progressbar" aria-valuenow="100" aria-valuemin="0" aria-valuemax="100">
   <div class="progress-bar" style="width: 100%">100%</div>
</div>
```



Componentes: Barras de progreso

- Barras de progreso coloreadas: Clase **progress-bar bg-***

```
<div class="progress" role="progressbar" aria-valuenow="25" aria-valuemin="0" aria-valuemax="100">
   <div class="progress-bar bg-success" style="width: 25%"></div>
</div>
<div class="progress" role="progressbar" aria-valuenow="50" aria-valuemin="0" aria-valuemax="100">
   <div class="progress-bar bg-info" style="width: 50%"></div>
</div>
<div class="progress" role="progressbar" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100">
   <div class="progress-bar bg-warning" style="width: 75%"></div>
</div>
<div class="progress" role="progressbar" aria-valuenow="100" aria-valuemin="0" aria-valuemax="100">
   <div class="progress-bar bg-danger" style="width: 100%"></div>
</div>
```



Componentes: Barras de progreso

code URJC

- Barras compuestas
  - Clase **progress-stacked**

```
<div class="progress-stacked">
   <div class="progress" role="progressbar" aria-valuenow="15" aria-valuemin="0" aria-valuemax="100"
style="width: 15%">
      <div class="progress-bar"></div>
   </div>
   <div class="progress" role="progressbar" aria-valuenow="30" aria-valuemin="0" aria-valuemax="100"
style="width: 30%">
      <div class="progress-bar bg-success"></div>
   </div>
   <div class="progress" role="progressbar"aria-valuenow="20" aria-valuemin="0" aria-valuemax="100"
style="width: 20%">
      <div class="progress-bar bg-info"></div>
   </div>
</div>
```

Más opciones: https://getbootstrap.com/docs/5.3/components/progress/

Componentes: Paginación

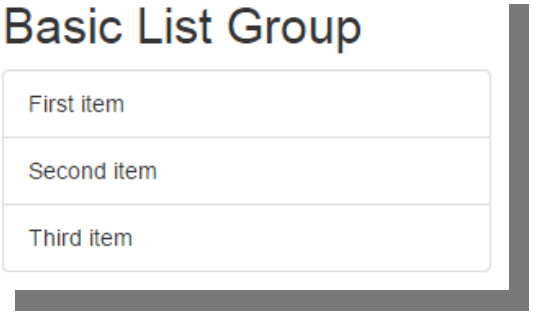
- Clases **pagination, page-item, page-link,active, disabled**:

```
<nav aria-label="...">
   <ul class="pagination">
      <li class="page-item disabled">
         <a class="page-link">Previous</a>
      </li>
      <li class="page-item"><a class="page-link" href="#">1</a></li>
      <li class="page-item active" aria-current="page">
         <a class="page-link" href="#">2</a>
      </li>
      <li class="page-item"><a class="page-link" href="#">3</a></li>
      <li class="page-item"><a class="page-link" href="#">4</a></li>
      <li class="page-item"><a class="page-link" href="#">5</a></li>
      <li class="page-item">
         <a class="page-link" href="#">Next</a>
      </li>
   </ul>
</nav>
```

Componentes: Listas

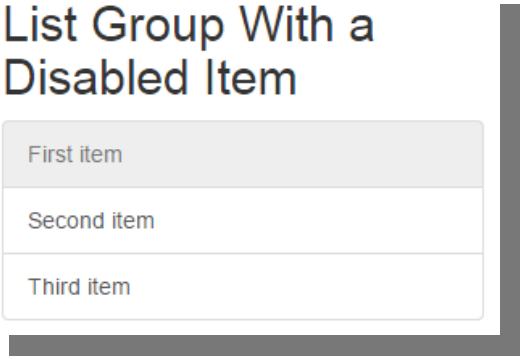
- Clases **list-group** y **list-group-item** aplicado a **ul**:

```
<h2>Basic List Group</h2>
<ul class="list-group">
   <li class="list-group-item">First item</li>
   <li class="list-group-item">Second item</li>
   <li class="list-group-item">Third item</li>
</ul>
```



- Clases **list-group** y **list-group-item** aplicado a **div** y **a**:

```
<h2>List Group With a Disabled Item</h2>
<div class="list-group">
   <a href="#" class="list-group-item disabled">First item</a>
   <a href="#" class="list-group-item">Second item</a>
   <a href="#" class="list-group-item">Third item</a>
</div>
```



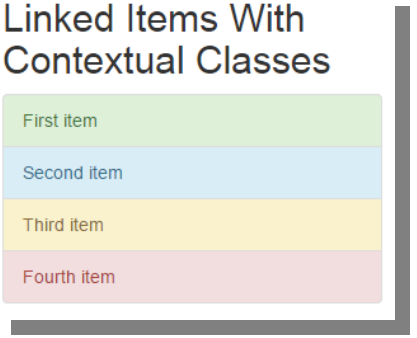
Componentes: Listas

- Clases **list-group-item-*** aplicado a **ul**:

```
<h2>List Group With Contextual Classes</h2>
<ul class="list-group">
   <li class="list-group-item list-group-item-success">First item</li>
   <li class="list-group-item list-group-item-info">Second item</li>
   <li class="list-group-item list-group-item-warning">Third item</li>
   <li class="list-group-item list-group-item-danger">Fourth item</li>
</ul>
```

- Clases **list-group-item-*** aplicado a **div** y **a**:

```
<h2>Linked Items With Contextual Classes</h2>
<div class="list-group">
   <a href="#" class="list-group-item list-group-item-success">First item</a>
   <a href="#" class="list-group-item list-group-item-info">Second item</a>
   <a href="#" class="list-group-item list-group-item-warning">Third item</a>
   <a href="#" class="list-group-item list-group-item-danger">Fourth item</a>
</div>
```



Componentes: Formularios

code URJC

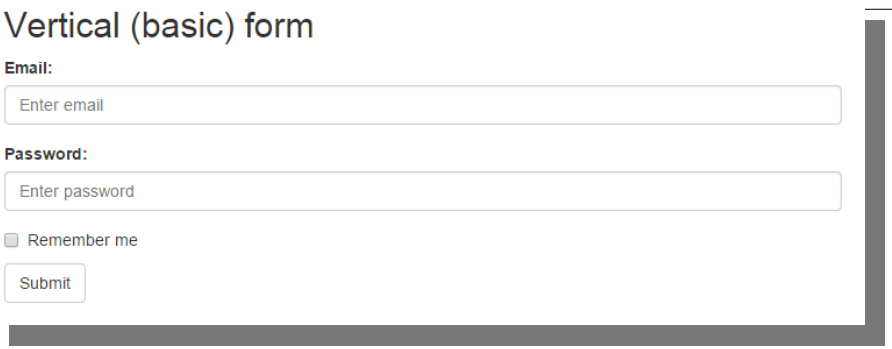
- Bootstrap proporciona tres layouts para formularios:
  - Vertical (por defecto)
  - En línea (inline)
  - Horizontal
- Se deben seguir las siguientes reglas:
  - Usar siempre **<form role="form">**
  - Agrupar elementos usando ** **
  - Usar la clase form-control en elementos de formulario textuales (**input**, **textarea**, **select**)

Componentes: Formularios

code URJC

- Formulario vertical

```
<h2>Vertical (basic) form</h2>
<form role="form">
   <div class="form-group">
      <label for="email">Email:</label>
      <input type="email" class="form-control" id="email" placeholder="Enter email">
   </div>
   <div class="form-group">
      <label for="pwd">Password:</label>
      <input type="password" class="form-control" id="pwd" placeholder="Enter password">
   </div>
   <div class="checkbox">
      <label><input type="checkbox"> Remember me</label>
   </div>
   <button type="submit" class="btn btn-default">Submit</button>
</form>
```



Componentes: Formularios

- Formulario en linea: Usando **row** y **col**

```
<div class="row">
   <div class="col">
      <input type="text" class="form-control" placeholder="First name" aria-label="First name">
   </div>
   <div class="col">
      <input type="text" class="form-control" placeholder="Last name" aria-label="Last name">
   </div>
</div>
```

Más formularios: https://getbootstrap.com/docs/5.3/forms/overview/

Componentes: Navbars

- Pestañas: Clases **navbar**, **navbar-expand**

```
<nav class="navbar navbar-expand-lg bg-body-tertiary">
   <div class="container-fluid">
      <a class="navbar-brand" href="#">Navbar</a>
      <div class="collapse navbar-collapse" id="navbarNavDropdown">
         <ul class="navbar-nav">
            <li class="nav-item">
               <a class="nav-link active" aria-current="page" href="#">Home</a>
            </li>
            <li class="nav-item">
               <a class="nav-link" href="#">Features</a>
            </li>
            <li class="nav-item">
               <a class="nav-link" href="#">Pricing</a>
            </li>
         </ul>
      </div>
   </div>
</nav>
```

Más configuración de navbar: https://getbootstrap.com/docs/5.3/components/navbar/

Personalización

- **Personalización del tema de bootstrap**
  - En el ejemplo se ha usado el **tema por defecto**
  - Se pueden hacer **temas personalizados**
  - Incluir sólo aquello que se necesita en la página (para reducir el tamaño y el tiempo de descarga)
  - Ajustando colores, diseños, etc…

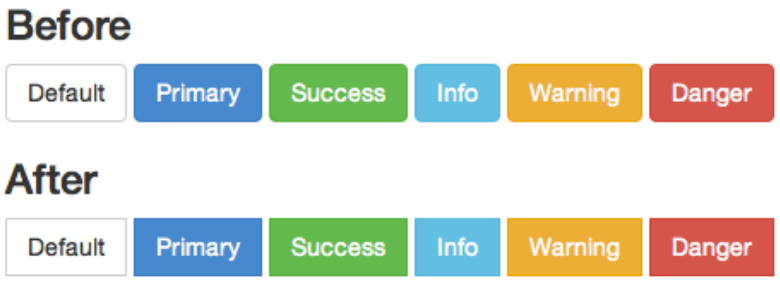
https://getbootstrap.com/docs/5.3/customize/overview/#overview

Personalización

- **Personalización del tema de bootstrap**
  - Otra forma de personalizar los temas de bootstrap es incluir primero el CSS de bootstrap y luego un CSS en el que se personalizan algunas propiedades

```
<head>
   <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
   <link rel="stylesheet" type="text/css" href="css/custom.css">
</head>
```

```
.btn {
    border-radius: 0px;
}
```

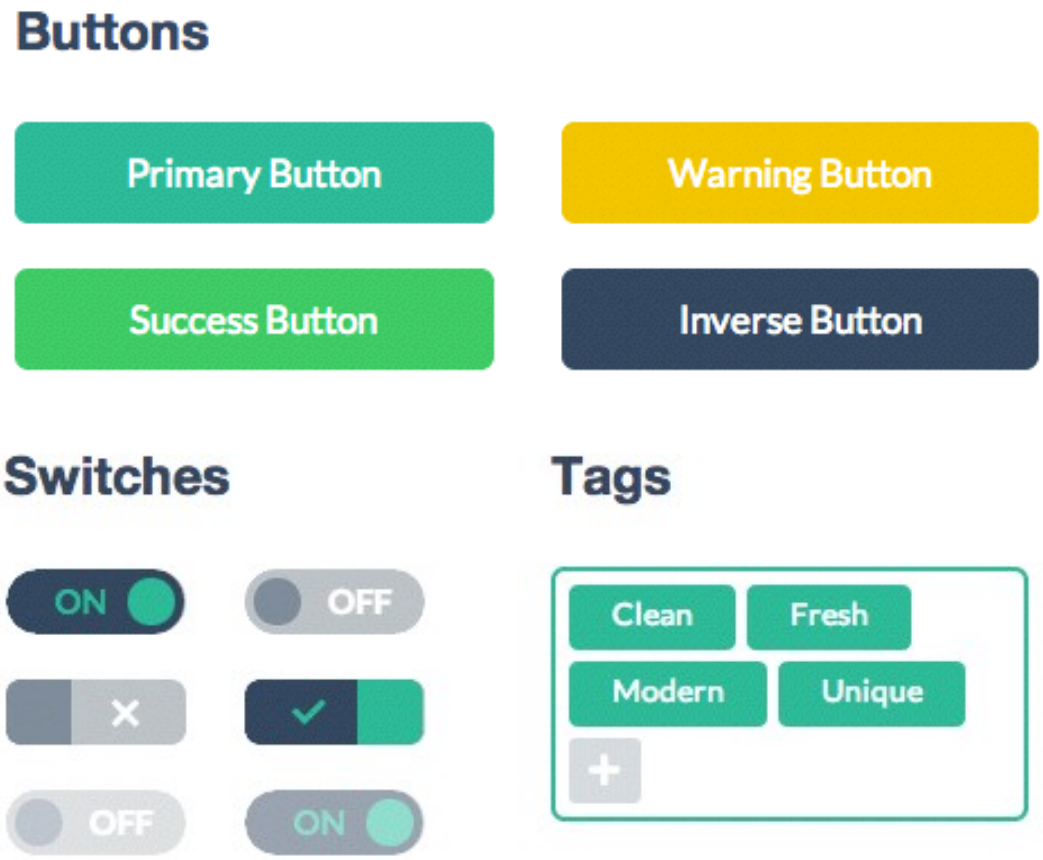


Alternativas a Boostrap

code URJC

Flat UI

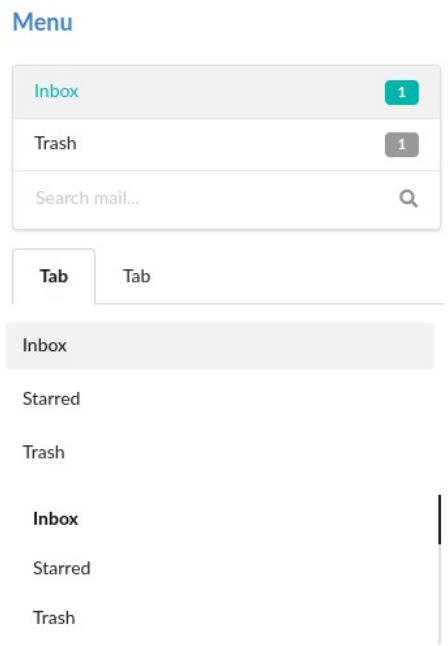
Free User Interface Kit

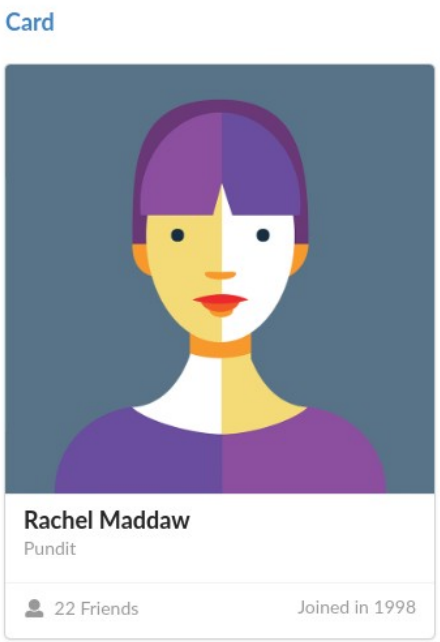


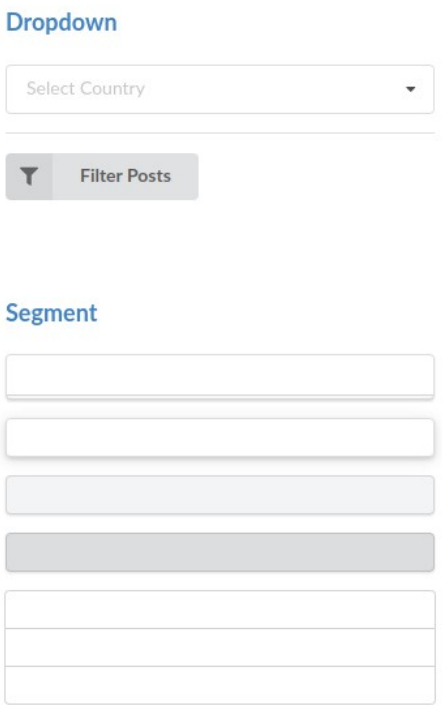
http://designmodo.github.io/Flat-UI/

Alternativas a Boostrap

code URJC





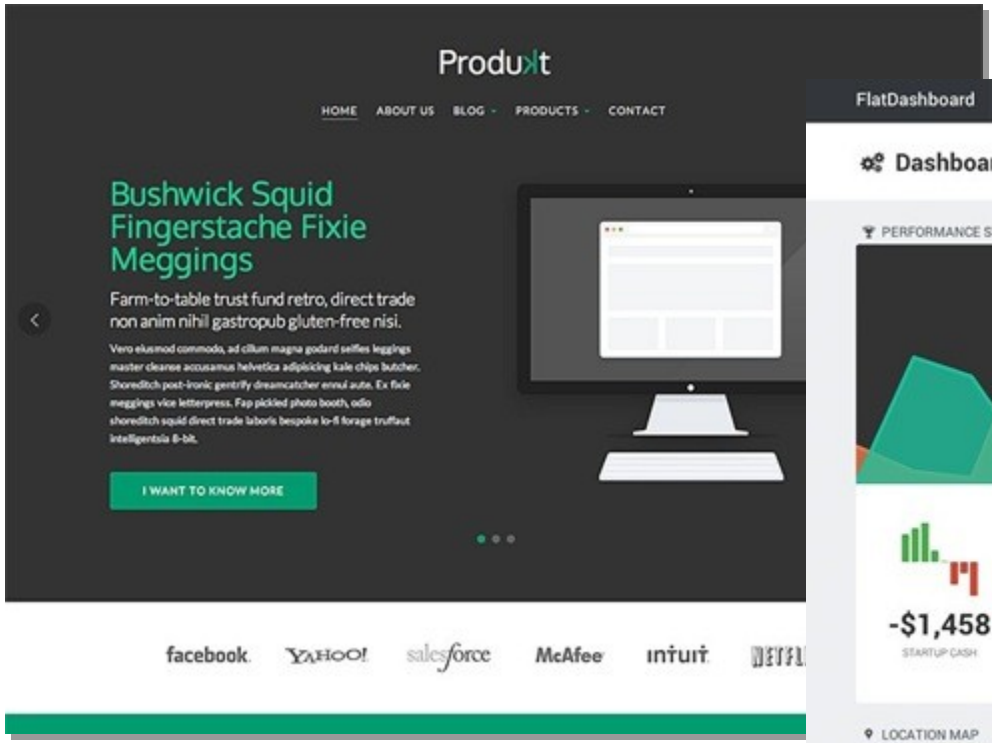


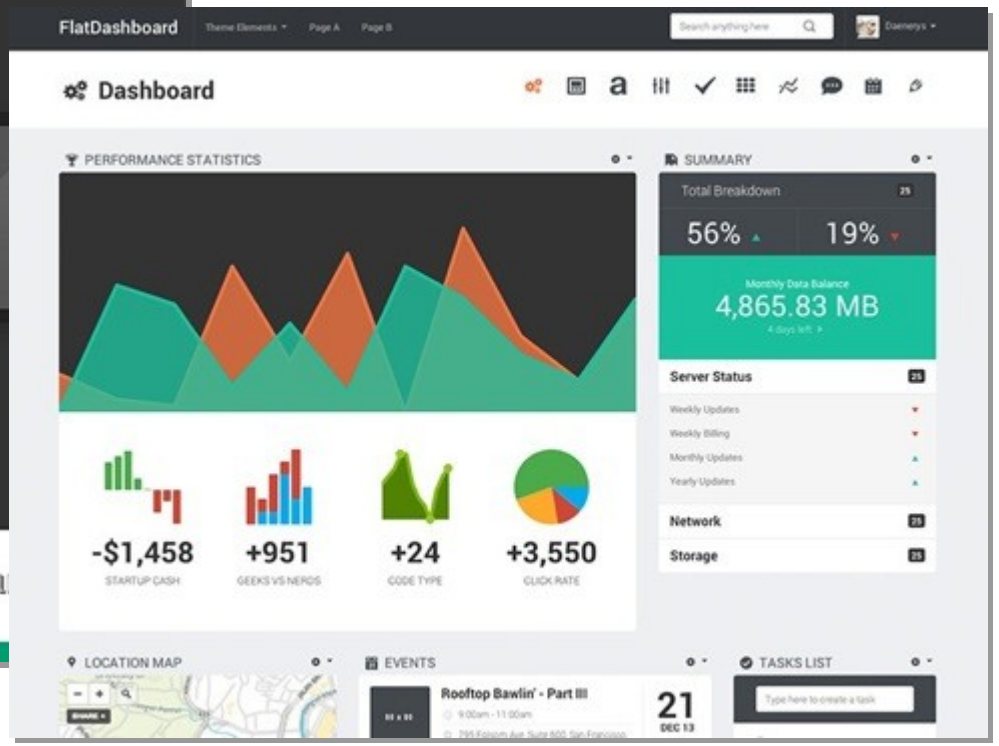
Semantic UI

https://semantic-ui.com/

Plantillas de Boostrap

- Plantillas completas





Plantillas de Boostrap

- **Plantillas completas**
  - https://getbootstrap.com/docs/5.0/examples/
  - https://wrapbootstrap.com
  - http://startbootstrap.com
  - http://bootswatch.com
  - https://bootstrapmade.com/bootstrap-5-templates/

Plantillas de Boostrap

code URJC

- Ejemplos de sitios hechos con bootstrap
  - http://expo.getbootstrap.com/
  - http://builtwithbootstrap.com/

Ejercicio 1

- Replica la siguiente página web utilizando componentes *Boostrap*
  - *Url de la imagen: https://www.codeurjc.es/img/codeurjc_logo.jpg*
  - *Será necesario customizar los estilos de Boostrap*

08-2023

code URJC

Fundamentos de la Web

Bloque III: Tecnologías de servidor web

Tema 3.1: Introducción a JavaScript y Node

Universidad Rey Juan Carlos

Micael Gallego
micael.gallego@urjc.es
@micael_gallego

Iván Chicano
ivan.chicano@urjc.es

Michel Maes
michel.maes@urjc.es

Pablo Fuente
pablo.fuente@urjc.es
@fuentepab

08-2023

code
URJC

Micael Gallego, Iván Chicano, Michel Maes, Pablo Fuente

JavaScript

code URJC

- **Introducción**
- El lenguaje JavaScript

JavaScript

- **Introducción**
  - JavaScript
  - Referencias
- El lenguaje JavaScript

JavaScript

- **Introducción**
  - **JavaScript**
  - Referencias
- El lenguaje JavaScript

JavaScript

code URJC

- Es un lenguaje de programación basado en el estándar **ECMAScript** de **ECMA** (otra organización diferente al **W3C**)
- Hay ligeras **diferencias** en la implementación de JS de los navegadores y runtimes, aunque actualmente todos son bastante **compatibles** entre sí (**en el pasado no fue así**)

http://www.ecma-international.org/

JavaScript

- **Versiones de JS**
  - JavaScript es un lenguaje que mejora cada año Al
  - principio las versiones se numeraban con un

número: ECMAScript 5 (**ES5**), ECMAScript 6 (**ES6**)

- Pero cuando se publicó ES6 decidieron llamarlo con el año: **ES2015**
- Se publica una actualización cada año

JavaScript

- **JavaScript no es Java**
  - Aunque algunos elementos de la sint recuerden a **Java**, son lenguajes **completamente diferentes**
  - El **nombre JavaScript** se eligió al publicar el lenguaje en una época en la que **Java estaba en auge** y fue principalmente por marketing (inicialmente se llamó **LiveScript**)

JavaScript

code URJC

- **Características principales** de **JavaScript**
  - **Scripting:** No necesita compilador. Se ejecuta directamente desde el código fuente usando un **Runtime**
  - **Tipado dinámico:** habitual en los lenguajes de script
  - **Funcional:** Las funciones son elementos de primer orden
  - **Orientado a objetos:** De forma más dinámica que Java

JavaScript

- **Introducción**
  - JavaScript
  - **Referencias**
- El lenguaje JavaScript

Referencias

- **Documentación**
  - https://developer.mozilla.org/es/docs/Web/JavaScript
  - https://es.javascript.info/
- **Tutoriales interactivos**
  - https://www.codecademy.com/
  - https://www.pluralsight.com/courses/javascript-getting-started

JavaScript

- **Introducción**
  - JavaScript
  - **Referencias**
- El lenguaje JavaScript

code URJC

¿Qué es Node.js?

- Es un **runtime** que permite ejecutar **JavaScript**.
- Está basado en **V8**, la máquina virtual de Google Chrome (https://v8.dev)
- Como se programa en **JavaScript**, el modelo de programación es **asíncrono** ya que las llamadas de I/O no son bloqueantes, lo que le hace muy **escalable** para **aplicaciones de red**

https://nodejs.org

code URJC

Soporte en la industria

Node.js está desarrollado bajo el paraguas de

Premium

Google

IBM

Joyent

Microsoft

Gold

GoDaddy

intel

Silver

BLOG STARTER

BrowserStack

CLOUDGREY

Coil

datree.io

Fidelity

GROUPON

hackerone

NearForm

NODESOURCE

npm

Profound Logic

Red Hat

SafetyCulture

SAUCELABS

sitepen

snyk

sourcegraph

VINCIT

yld

Node.js

code URJC

Plataforma con mucha adopción

- Tiene una **gran comunidad de desarrolladores**
- Está soportado de forma oficial en los **proveedores cloud** (como Java, Python…)
- Tiene una **gran cantidad de paquetes NPM**
- Cada vez se usa más a nivel **empresarial**
- Incorpora las últimas versiones del estándar **EcmaScript** (lenguaje y librerías muy completas)

code URJC

Tipos de aplicaciones

Con Node.js se pueden desarrollar todo tipo de aplicaciones:

- Servicios web (Aplicaciones web MVC, APIs REST)
- Herramientas por línea de comandos (CLI)
- Aplicaciones web SPA
- Aplicaciones con interfaz gráfico de usuario

code URJC

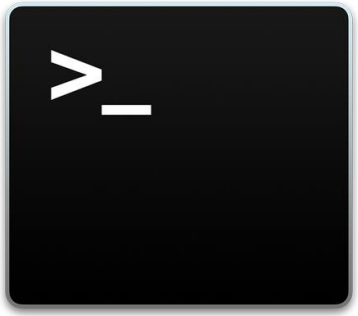
Servicios web

- Aplicaciones web MVC: Generación de HTML en servidor
- Backend para SPA
- Comunicación: API REST, WebSockets, AMQP, gRPC…
- Acceso a base de datos: Relacionales, NoSQL…
- Serverless, Microservicios…

code URJC

Herramientas por línea de comandos

- Herramientas para desarrollo:
  - Linters
  - Compiladores
  - Testing de rendimiento
- Gestión de proyectos: angular-cli, vue-cli…
- Scripts de sistemas



code URJC

Aplicaciones con interfaz gráfico de usuario

Se combina en una aplicación **Node.js** (para acceso al sistema) con un navegador web (**Chromium**) para la interfaz de usuario.

https://electronjs.org/

https://nwjs.io/

code URJC

Librerías

- En **Node.js** no se pueden usar librerías del browser como **DOM, BOM**, etc. porque no hay interfaz de usuario
- Node.js ofrece por defecto unas **mínimas librerías** para interactuar con el **sistema operativo**
- Existen muchas librerías libres disponibles en NPM, el gestor de paquetes oficial de Node

https://npmjs.com

code URJC

Librerías

Las librerías incluidas por defecto en Node.js ofrecen las siguientes funcionalidades:

- Gestión de procesos
- Gestión de datos en memoria nativa del sistema
- Acceso a la línea de comandos
- Sistema de ficheros
- Redes (Http, Sockets)
- Flujos de bytes

https://nodejs.org/api

code URJC

Versiones

Node.js ofrece 2 tipos de versiones

- Activa LTS (*Long Term Support*): Soporte extendido
- Actual (*Current*): últimas funcionalidades

En Septiembre de 2023 las versiones son

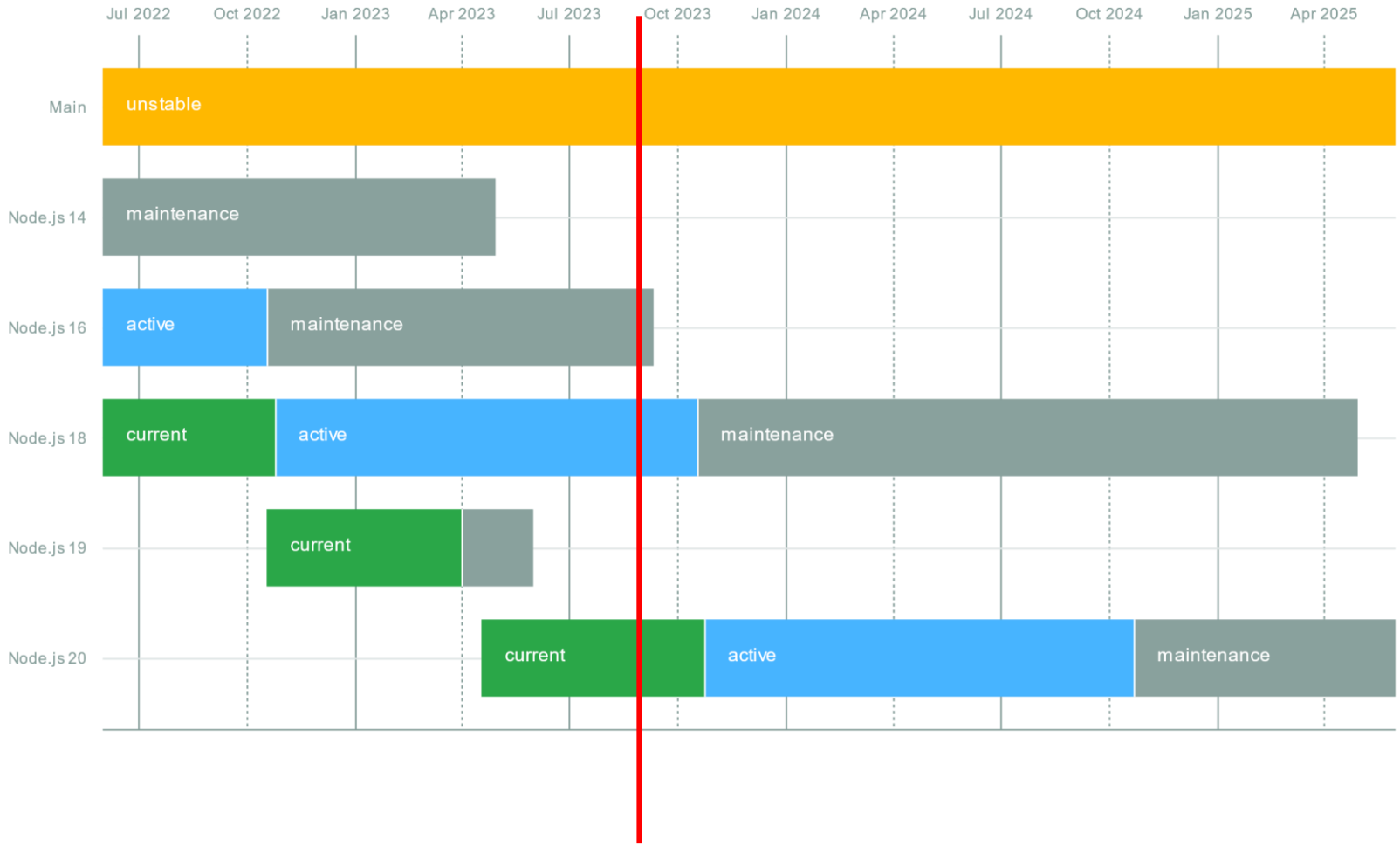
- Activa LTS: **18.17.1**
- Actual: **20.5.1**

https://github.com/nodejs/LTS#lts-schedule1

Node.js

code URJC

Versiones



code URJC

Rendimiento

- El código **JavaScript** no se puede ejecutar de forma tan eficiente como Java o C# debido a su naturaleza dinámica
- Su modelo **asíncrono**, ofrece una escalabilidad igual o superior a aplicaciones Java equivalentes que no sean asíncronas
- El **modelo de programación es mucho más sencillo** al no existir varios hilos de ejecución (no hay condiciones de carrera)
- Se considera una **solución aceptable en servicios de red** que con mucha I/O y poca algoritmia

code URJC

Instalación

https://nodejs.org/en/download

code URJC

Instalación

```
$ curl -sL https://deb.nodesource.com/setup_16.x | sudo -E bash -
$ sudo apt-get install nodejs
```

code URJC

IDE

Algunos IDEs vienen con Node.js integrado, pero se recomienda instalar Node en el sistema:

Sublime Text

Visual Studio Code

WebStorm

ATOM

Aplicaciones de consola

code URJC

- Usaremos **Visual Studio Code** para desarrollar y depurar
- Usaremos algunas **APIs** que vienen incluidas en Node.js
- Usaremos **NPM** para instalar librerías externas

Ejercicio 1

code URJC

Hello world!

Ejercicio1

- Crear una carpeta
- Crear un fichero **app.js**

```
console.log('Hello world!');
```

- Ejecutar el comando

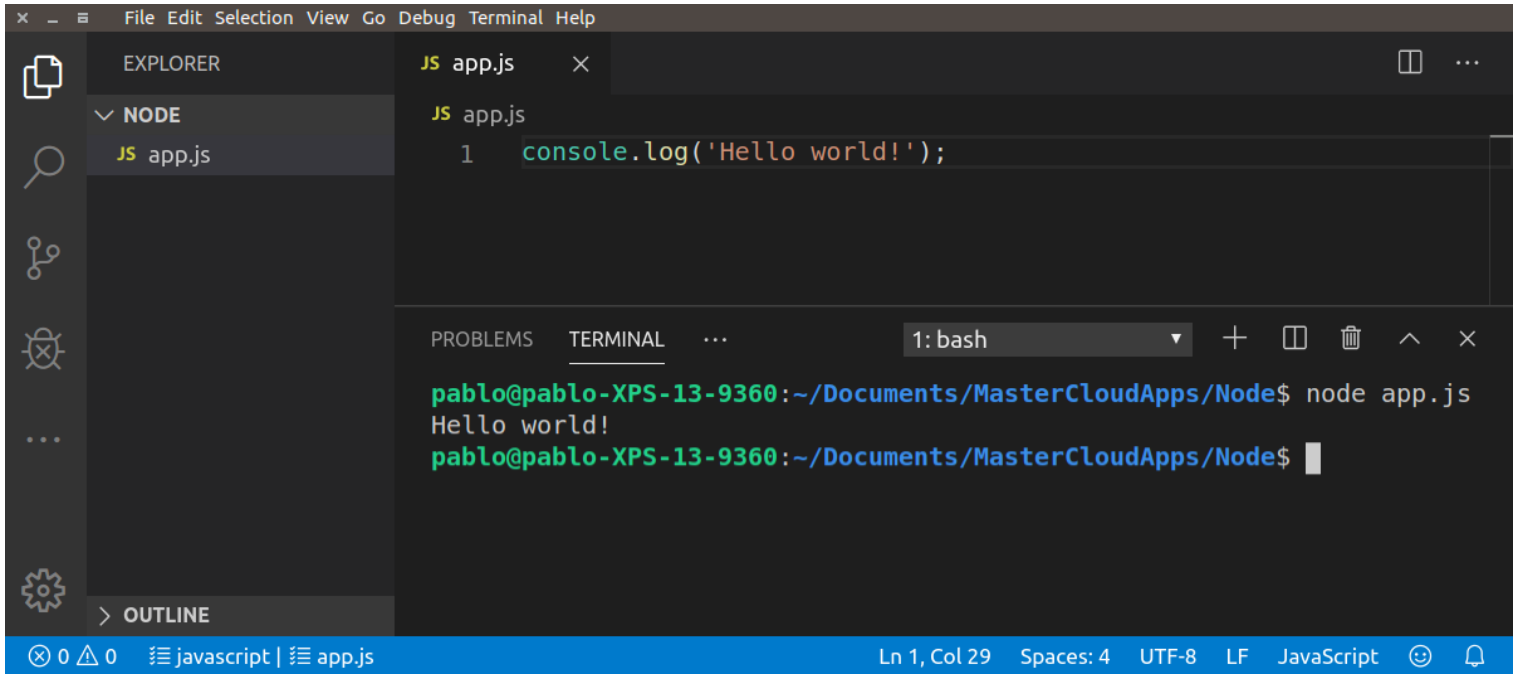
```
$ node app.js
```

Aplicaciones de consola

code URJC

Visual Studio Code

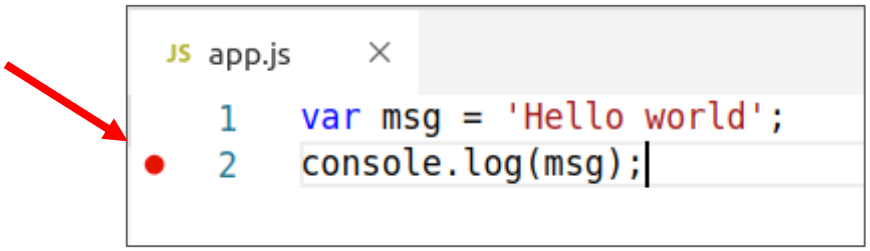
- Abrimos la carpeta con todo el código
- Ejecutamos comandos en el terminal integrado (**Ctrl+`**)



Aplicaciones de consola

Depuración en Visual Studio Code

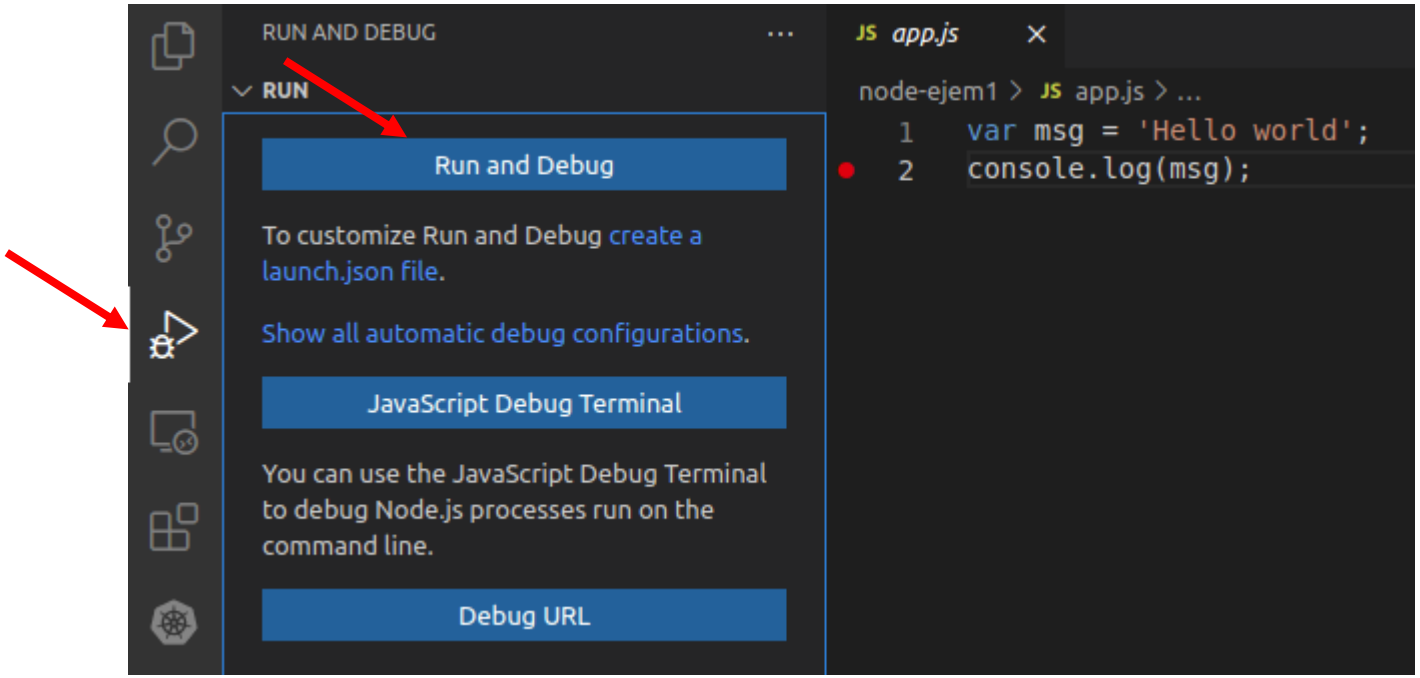
- Para poner un punto de ruptura se marca en la barra a la izquierda del código



https://code.visualstudio.com/docs/nodejs/nodejs-tutorial

Aplicaciones de consola

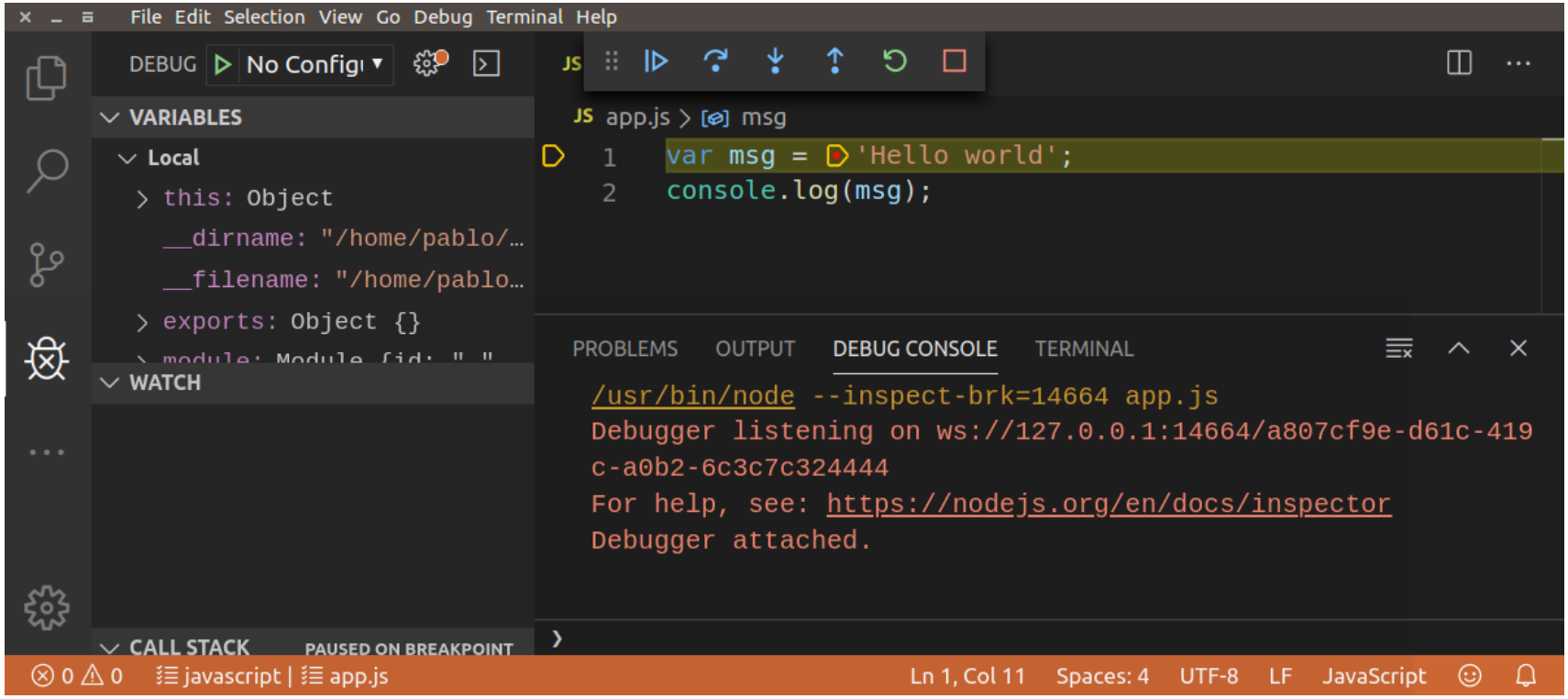
Depuración en Visual Studio Code



Aplicaciones de consola

code URJC

Depuración en Visual Studio Code



JavaScript

- Introducción
- **El lenguaje JavaScript**
  - Características del lenguaje
  - Sintaxis básica
  - Arrays
  - Sentencias de control de flujo
  - Funciones
  - Excepciones Orientación a
  - objetos

JavaScript

- Introducción
- **El lenguaje JavaScript**
  - **Características del lenguaje**
  - Sintaxis básica
  - Arrays
  - Sentencias de control de flujo
  - Funciones
  - Excepciones Orientación a
  - objetos

Características del lenguaje JS

code URJC

- Vamos a estudiar las características **más usadas** actualmente para desarrollar aplicaciones
- Hay ciertas características que ya **no se aconsejan y no veremos** (aunque te las puedes encontrar si ves código antiguo)
- Como estáis aprendiendo **Java**, haremos una comparativa entre ambos

Características del lenguaje JS

code URJC

- **Lenguaje de script (Java es compilado)**
  - No existe compilador
  - El navegador carga el código, lo analiza y lo ejecuta
  - El navegador indica tanto **errores de sintaxis** como errores de **ejecución**
- **Tipado dinámico (Java tiene tipado estático)**
  - Al declarar una variable no se indica su tipo
  - A lo largo de la ejecución del programa una misma variable puede tener **valores de diferentes tipos**

Características del lenguaje JS

- **Imperativo y estructurado (como Java)**
  - Se declaran **variables**
  - Se ejecutan las sentencias **en orden**
  - Dispone de **sentencias de control** de flujo de ejecución (if, while, for…)
  - La **sintaxis** imperativa/estructurada es muy parecida a Java y C

Características del lenguaje JS

code URJC

- **Orientado a objetos**
  - Todos los **valores son objetos** (no como en Java, que existen tipos primitivos)
  - Existe **recolector de basura** para liberar la memoria de los objetos que no se utilizan (como en Java)
  - La **orientación a objetos** puede basarse en clases o en prototipos (Java se basa en clases)
  - En tiempo de ejecución se pueden **crear objetos**, cambiar el valor de los **atributos** e invocar a **métodos** (como en Java)
  - En tiempo de ejecución se pueden **añadir y borrar atributos y métodos** (en Java no se puede)

Características del lenguaje JS

code URJC

- **Funciones**
  - Aunque sea orientado a objetos, también permite declarar **funciones independientes** (en Java todas las funciones son métodos de clases)
  - Las funciones se pueden **declarar en cualquier sitio**, asignarse a variables y pasarse como parámetro
  - Existen **funciones anónimas** (como las **expresiones lambda** de Java)
  - En JavaScript se puede implementar código inspirado en el **paradigma funcional**

JavaScript

- Introducción
- **El lenguaje JavaScript**
  - Características del lenguaje
  - **Sintaxis básica**
  - Arrays
  - Sentencias de control de flujo
  - Funciones
  - Excepciones Orientación a
  - objetos

Sintaxis básica

code URJC

- **Comentarios (como en Java)**
  - Una línea //
  - Multilínea /* ..... */
- **Delimitadores (como en Java)**
  - De bloque { }
  - De sentencia ; (opcionales)
- **Palabras reservadas**

```
abstract boolean break byte case catch char class const continue debugger default
delete do double else enum export extends false final finally float for function goto
if implements import in instanceof int interface long native new null package private
protected public return short static super switch synchronized this throw throws
transient true try typeof var volatile void while with
```

Sintaxis básica

- **Variables**
  - Es un lenguaje dinámico (con tipado **dinámico**)
  - Las variables se tienen que declarar pero **no se indica el tipo**

```
//La variable edad tendrá un valor de 34 let
edad = 34;
let encontrado = false;
```

  - La variable está disponible en el bloque en el que se declara (como en Java)

Sintaxis básica

- **Variables**
  - Una misma variable puede tener un valor numérico en un punto de ejecución y una cadena de caracteres tiempo después

```
let id;
...
id = 34;
...
id = "XDFS"
```

Sintaxis básica

code URJC

Ejemplo1

- **Variables**
  - Si las variables **no se inicializan** tienen el valor **undefined** (en vez de cero o null como en Java)

```
let hola;
console.log("Hola: " + hola);
```

```
Hola: undefined
```

  - Si las variables **no se declaran (por una equivocación)**
    - Si se intenta **leer su valor** salta un **error** (que finaliza la ejecución)
    - Si se **asigna un valor**, se crea un nuevo atributo en el objeto global (**no dan error las siguientes lecturas**)

Sintaxis básica

- **Tipos de datos básicos**
  - Son objetos (no existe distinción como en Java entre objetos y tipos primitivos)
  - **Number:** Números enteros y reales de cualquier precisión
  - **Boolean:** true o false (como en Java)

Sintaxis básica

- **Tipos de datos básicos**
  - **String**
    - Cadenas de caracteres
    - Comillas simples o dobles
    - No pueden modificarse. Inmutables (como en Java)
    - Concatenación con + (como en Java)
    - Como no hay tipo caracter, se usa un string con un único caracter

Sintaxis básica

- **Operadores en expresiones**
  - **Similares a Java**
    - Aritméticos: + - * / % (la división es siempre real)
    - Comparación números: < > <= >=
    - Lógicos: && || !
    - Comparativo: ?: (Elvis operator)
    - Modificación: ++ --
    - Asignación: = += -= *= /= %=

Sintaxis básica

Ejemplo2

- Operadores en expresiones
  - Comparación (diferente a Java)
    - Igual: === Distinto: !==
    - En strings se comporta como el equals(…) en Java
    - En arrays se comporta como == en Java

```
let uniName = "URJC";

let otherName = "UR" + "JC";

console.log(uniName === otherName) // true
```

JavaScript

code URJC

- Introducción
- **El lenguaje JavaScript**
  - Características del lenguaje
  - Sintaxis básica
  - **Arrays**
  - Sentencias de control de flujo
  - Funciones
  - Excepciones Orientación a
  - objetos

Arrays

- **Los arrays de JavaScript son parecidos a los de Java**
  - El acceso para lectura o escritura es con **[ ]**
  - Se inicializan con **[ ]** (en Java con **{ }**)
  - Tienen la propiedad **length**
  - Empiezan por **cero**

Ejemplo3

```
let numbers = [3, 4, 4,
               2];
console.log(numbers[0]);    3
                //
console.log(numbers.length)  // 4
```

Arrays

- **Los arrays de JavaScript son parecidos a los de Java**
  - La **asignación** de una variable con un array a otra variable, no copia. Ambas variables apuntan al **mismo objeto array**
  - El operador **===** compara si son el **mismo objeto** (no que tengan el contenido igual)

Ejemplo4

```
let nums = [1, 2, 3, 4]; let numsAux = nums;

numsAux[0] = -1; console.log(nums[0] === -1)

// true
console.log(numsAux === nums) // true
```

Arrays

- **Los arrays de JavaScript son parecidos a los de Java**
  - Los **arrays de varias dimensiones son arrays de arrays**. Hay que crear de forma explícita los niveles.
  - Se pueden recorrer de forma similar

Ejemplo5

```
let numbers = [3, 4, 4, 2];

 console.log(numbers[0 +  ', +
                     ]    '
             numbers[1] +  ', +
             numbers[3]);  '
             numbers[2] +  ', +
                           '
```

3,4,4,2,

Arrays

- **Pero se diferencian en algunos detalles**
  - Los arrays literales con [ ] en vez de { } y sin new

```
let empty = [];
let numbers = ['zero','one','two','three']
```

  - Un array en JS puede tener en cada posición valores de diferentes tipos mezclados (tipado dinámico)

Arrays

- **Errores de acceso en arrays**
  - La lectura de un elemento fuera de los límites devuelve **undefined**
    - En Java sería un ArrayIndexOutOfBoundsException
  - La escritura de un elemento fuera de los límites del array se permite, haciendo más grande el array y rellenando con valores **undefined** los huecos

Arrays

```
let numbers = [3, 4];
console.log(numbers[2]);

numbers[4] = 5;
console.log(numbers[4]);

 console.log(numbers[0 +  ',  +
                     ]
             numbers[1] +  ',  +
                           '
             numbers[4]);  ',  +
             numbers[2] +  '
                           '
             numbers[3] +  ',  +
                           '
```

```
undefined 5
3,4,undefined,undefined,5
```

Ejemplo6

Arrays

Ejemplo7

- **Errores de acceso en arrays**
  - Si una variable tiene el valor **undefined** y se intenta acceder a un elemento como si tuviera un array, genera un error "**Uncaught TypeError: Cannot read properties of undefined (reading '5')**"

```
let numbers;

console.log(numbers[5]);
```

  - En Java sería un NullPointerException

Arrays

- **Modificación del array**
  - Se pueden establecer elementos en posiciones no existentes y el array crece dinámicamente.
  - El método **push** añade un elemento al final del array (como el método **add** del **ArrayList** en Java)
  - Modificando la propiedad **length** se puede cambiar el tamaño del array (si se amplía, se rellena con undefined) (en Java es de sólo lectura)

Arrays

- **Modificación del array**
  - El **operador delete** borra un elemento dejando el hueco con valor **undefined**

```
delete numbers[2];
```

  - Para borrar y no dejar el hueco se usa el método **splice** indicando el índice desde el que hay que borrar y el número de elementos que borrar

```
numbers.splice(2, 1);
```

JavaScript

code URJC

- Introducción
- **El lenguaje JavaScript**
  - Características del lenguaje
  - Sintaxis básica
  - Arrays
  - **Sentencias de control de flujo**
  - Funciones Excepciones
  - Orientación a objetos
  -

Sentencias de control de flujo

code URJC

- **Sentencia if**
  - Sintaxis como en Java
  - No es obligatorio que la expresión devuelva un boolean
  - Se consideran como falsos:
    - false, null, undefined, “” (cadena vacía), 0, NaN

Sentencias de control de flujo

code URJC

Ejemplo8

```
let nota = 7;
...
if(nota < 5){ console.log('Suspenso');
} else { console.log('Aprobado');
}
```

```
let nombre = '';

if(!nombre){
  nombre = 'Sin nombre';
}
```

Sentencias de control de flujo

code URJC

Ejemplo9

- **Sentencias switch, while, do while, for**
  - Sintaxis y semántica como en Java y C

```
let animal = 'Jirafa';

switch (animal) { case 'Vaca':
case 'Jirafa':
  case 'Cerdo': console.log('Sube
    al arca'); break;
case 'Dinosaurio': default:
  console.log('No sube');
}
```

```
let array = [3,4,5,6,4,5]

for(let i=0; i<array.length; i++){
  console.log(array[i]);
}
```

JavaScript

- Introducción
- **El lenguaje JavaScript**
  - Características del lenguaje
  - Sintaxis básica
  - Arrays
  - Sentencias de control de flujo
  - **Funciones** Excepciones
  - Orientación a objetos
  -

Funciones

- **JavaScript** es un lenguaje **funcional** en el sentido de que las **funciones** son **ciudadanos de primera clase**
- Se pueden declarar con nombre

Ejemplo10

```
function imprime(param){
   console.log(param);
}

imprime(4);
```

Funciones

- Se pueden declarar **sin nombre** (**anónimas**) y asignarse a una **variable**

Ejemplo11

```
let imprime = function(param){
   console.log(param);
}

imprime(4);
```

Funciones

- **Sentencia return**
  - Cuando se ejecuta fuerza la terminación de una función (Como en Java)
  - Puede tener un valor asociado que será devuelto por la función en la que se ejecute

Ejemplo12

```
function myFunction(a, b) {
   return a * b;
}

console.log(myFunction(4, 3));
```

Funciones

- Las funciones se pueden pasar como **parámetro** a otras funciones

```
function showItems(array, hasToShow) {

  for(let i=0; i<array.length; i++){
    if(hasToShow(array[i])){
    console.log(array[i]);
  }
}
}

function isEven(number){
  return number % 2 == 0;
}

showItems([1,2,3,4,5,6], isEven);
```

Ejemplo13

```
2
4
6
```

Funciones

- Se pueden declarar funciones en el cuerpo de otras funciones

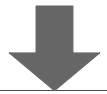
Ejemplo14

```
function showNonZeros(numbers) {

    let isNonZero = function(num) { return
        num != 0;
}

showItems(numbers, isNonZero);
}
```

Funciones

- Se pueden declarar funciones en la llamada a otras funciones

```
function showNonZeros(numbers) {

    let isNonZero = function(num) { return
        num != 0;
}

showItems(numbers, isNonZero);
}
```



```
function showNonZeros(numbers) {

    showItems(numbers, function(num) { return
        num != 0;
});
}
```

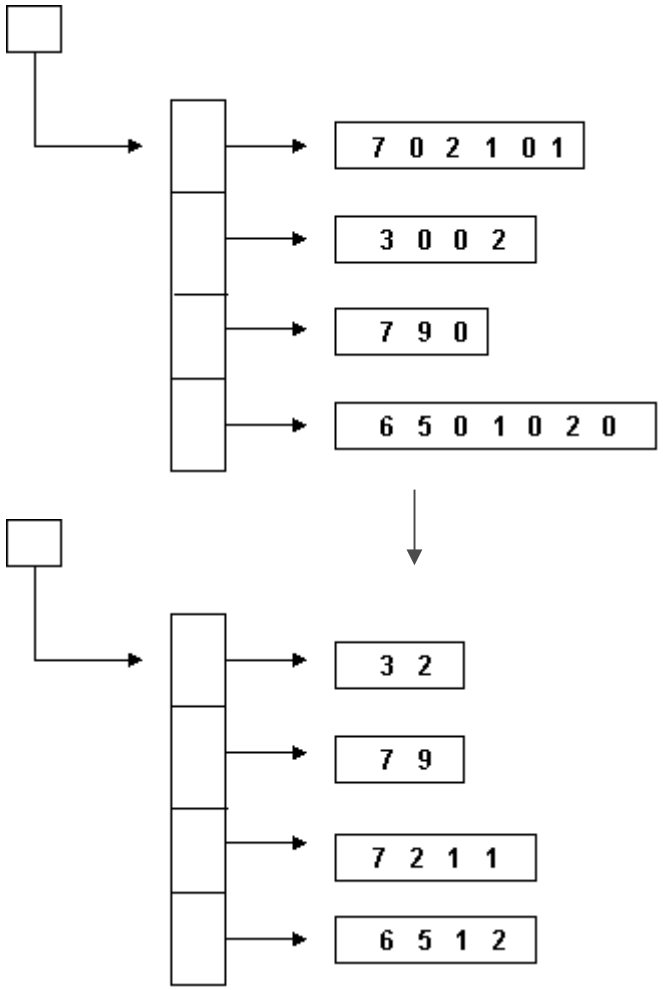
Ejemplo15

Ejercicio 2

- Crear una función que **reciba un array como parámetro** y devuelva un array de dos elementos
- El primer elemento apuntará a un **array con los números pares** del array que se pasa como parámetro
- El segundo elemento apuntará a un **array con los números impares** del array que se pasa como parámetro
- Para probar la función se implementarán varias llamadas con **diferentes arrays** y el resultado se mostrará en la **consola**

Ejercicio 3

- Crear una función que **reciba un array bidimensional, le quite los ceros y ordene las filas de menor a mayor longitud**
- Para probar la función se implementarán varias llamadas con **diferentes arrays** y el resultado se mostrará en la **consola**



Ejercicio 3

- Para ordenar los arrays por tamaño se puede usar el **algoritmo de la burbuja** (*bubble sort*)
- Ejemplo: Ordenación de un array de números (tiene que adaptarse para ordenar arrays por su tamaño)

```
function sortNumbers(numbers) {
  for (let i = 0; i < numbers.length; i++) {  for
    (let j = 0; j < numbers.length-1; j++) {
      if (numbers[j] > numbers[j + 1]) { let temp
        = numbers[j];
      numbers[j] = numbers[j + 1]; numbers[j + 1]
        = temp;
    }
  }
 }
 return numbers;
}
```

Funciones

- **Parámetros de una función**
  - Si se pasan menos parámetros de los que están en la cabecera, los que faltan toman el valor **undefined (No hay error)**

Ejemplo16

```
function imprime(param){
   console.log(param); //undefined
}

imprime();
```

Funciones

- Como pasar **menos parámetros no da error**, se puede usar para implementar parámetros **opcionales**

```
function arrayToString(array, separador){
  if(!separador){
  separador = ",";
}
let result = "";
  for(let i=0; i<array.length; i++){
    result += array[i] + separador;
  }
  return result;
}

console.log(arrayToString([1,2,3,4]));
console.log(arrayToString([1,2,3,4],"-"));
```

Ejemplo17

```
1,2,3,4,
```

```
1-2-3-4-
```

Funciones

- **Parámetros de una función**
  - Si se pasan más parámetros de los que están en la cabecera, los que sobran se **ignoran (No hay error)**

Ejemplo18

```
function imprime(param){
   console.log(param); //5
}

imprime(5, 7);
```

Funciones

- **Información accesible desde la función**
  - Parámetros y variables declaradas en la función

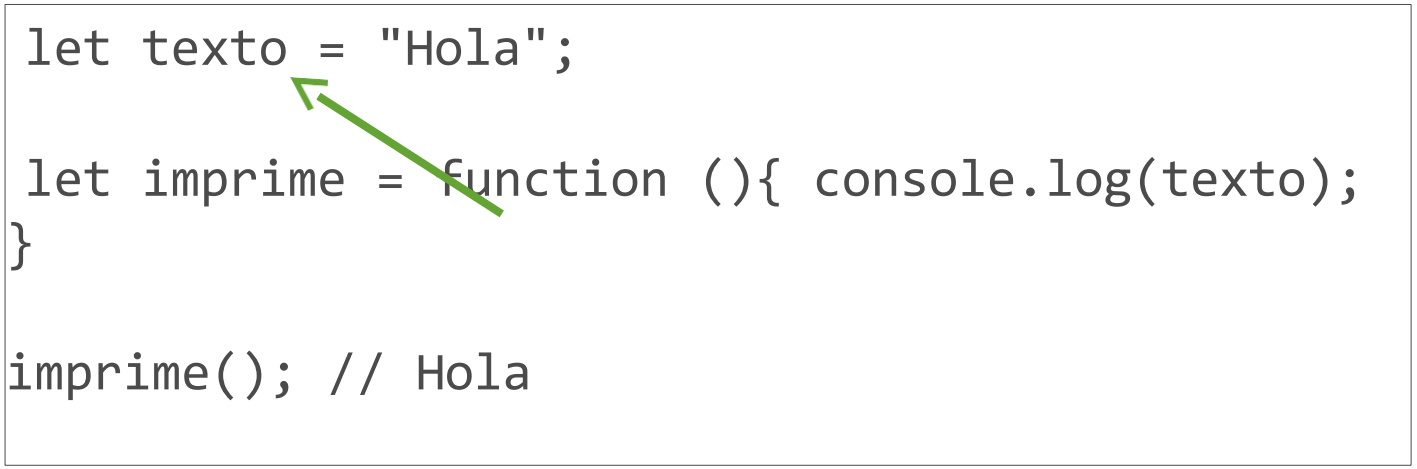
Ejemplo19

```
let imprime = function (param){ let local = 0;
console.log("param:"+param+" local:"+local);
}

imprime(4); // param:4 local:0
```

Funciones

- **Información accesible desde la función**
  - Variables accesibles en el punto en que se declara la función (ámbito léxico)

Ejemplo20



Funciones

- **Información accesible desde la función**
  - Cuando se referencia a una variable no sólo se accede a su valor, se accede a la **propia variable en sí**
  - Si se **cambia el valor** de la variable, la **función podrá leer** el nuevo valor
  - En Java las variables a las que accede una expresión lambda **no pueden cambiar de valor**

Funciones

- **Información accesible desde la función**
  - El conjunto de variables a las que tiene acceso la función se **llama cerradura o cierre (*closure*)**

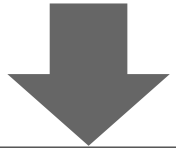
```
let texto = 'Hola' function
imprime(){
   console.log(texto)
}

imprime(); // Hola texto =

'Adios' imprime(); // Adios
```

Ejemplo21

Funciones flecha (arrow function)

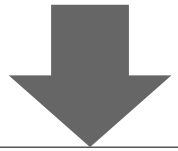
- Una forma más compacta de escribir funciones anónimas

```
let f = function (v) {
    return v + 1;
}
```



```
let f = v => v + 1;
```

```
var f = function (p1,p2) {
    console.log('Hola');
    console.log('Adios');
    return 3;
}
```



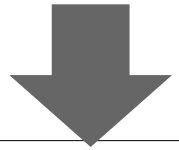
```
var f = (p1,p2) => {
    console.log('Hola');
    console.log('Adios');
    return 3;
}
```

Funciones flecha (arrow function)

- Muy usado cuando la función se declara en la lista de parámetros de otra función

Ejemplo22

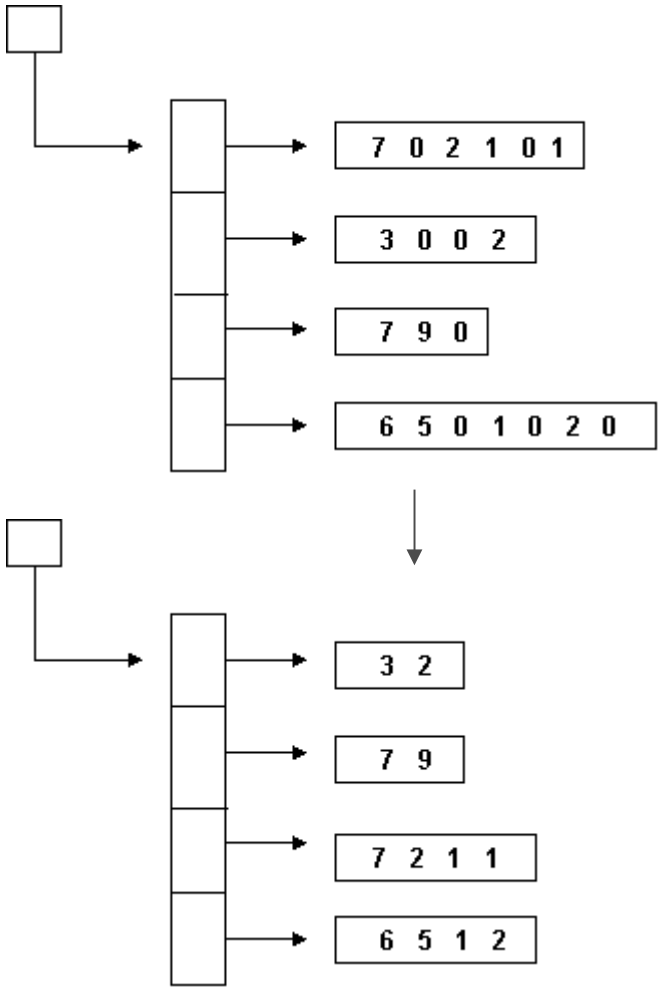
```
function showNonZeros(numbers) {

    showItems(numbers, function(num) {
        return num != 0;
});
}
```



```
function showNonZeros(numbers) {

    showItems(numbers, num => num != 0);
}
```

Ejercicio 4

- Haz que el ejercicio 3 sea configurable
- En vez de eliminar los ceros de los arrays, que elimine los números que decida el usuario pasando una función como parámetro
- Usa arrow functions en las pruebas que llaman a `quitaNumerosYOrdena(...)`



JavaScript

code URJC

- Introducción
- **El lenguaje JavaScript**
  - Características del lenguaje
  - Sintaxis básica
  - Arrays
  - Sentencias de control de flujo
  - Funciones
  - **Excepciones**
  - Orientación a objetos

Excepciones

- Funcionan igual que en **Java**
- Existe un bloque con **try catch finally**
- El operador **throw** eleva la excepción
- A diferencia de Java, se puede lanzar cualquier objeto como excepción, aunque lo recomendable es elevar un objeto con propiedades **name** y **message**

Excepciones

```
try {
   let error = ...; if (error) {
      throw "An error";
   }
   return true;

} catch (e) {

   alert (e); return false;

} finally {
   //do cleanup
}
```

JavaScript

code URJC

- Introducción
- **El lenguaje JavaScript**
  - Características del lenguaje
  - Sintaxis básica
  - Arrays
  - Sentencias de control de flujo
  - Funciones
  - Excepciones
  - **Orientación a objetos**

Orientación a Objetos

code URJC

- Hasta ahora hemos visto que **JavaScript** es un lenguaje bastante familiar en cuanto a **sintaxis** y **características:**
  - Imperativo y estructurado Tipado dinámico
  - Tres tipos básico: Number, Boolean y String
  - Arrays dinámicos
  - Recolector de basura Funciones
  -
  -

Orientación a Objetos

code URJC

- La orientación a objetos en **JavaScript** a primera vista parece **similar** a la de **Java**, pero en esencia es **muy diferente**
- La gran **mayoría de los lenguajes** de programación implementan la Orientación a Objetos con **clases** (Java, C#, C++, Python, Ruby…)
- En JavaScript se añadió el soporte de **clases** en ES2015. Hasta ese momento se usaban los **prototipos** (que no veremos nosotros)

Orientación a Objetos

- **Creación de objetos**
  - En cualquier momento se puede **crear un objeto, definir sus atributos** y asignarles **valor**

```
let empleado = {
   nombre: "Pepe",
   salario: 700
}
```

Orientación a Objetos

- **Atributos de un objeto**

Ejemplo23

```
let empleado = { nombre: "Pepe", salario:
  700
}

console.log("S:"+empleado.salario);

empleado.salario = 800;

empleado.telefono = "663232539";

console.log("T:"+empleado.telefono);
```

Orientación a Objetos

- **Métodos en un objeto**
  - Los objetos pueden tener **métodos**
  - **Los métodos son en realidad** funciones asignadas a una propiedad

```
let empleado = { nombre: "Pepe", salario: 700,
  toString: function(){
  return "N:"+ this.nombre +" S:"+ this.salario;
}
}
```

Para acceder a los atributos del objeto es necesario usar this (en Java es opcional)

Orientación a Objetos

code URJC

- **Métodos en un objeto**
  - Los métodos se **invocan** con la notación punto (como en Java)

```
//Devuelve 'Nombre:Pepe, Salario:700' let
texto = empleado.toString();
```

  - Se pueden **añadir métodos** a un objeto en cualquier momento

```
empleado.getCategoria = function(){
   return this.salario > 800 ? 'Superior':'Normal';
}
```

Orientación a Objetos

code URJC

- **Métodos en un objeto**

Ejemplo24

```
let empleado = { nombre: 'Pepe', salario: 700, toString:
  function(){
  return 'N:'+this.nombre+' S:'+this.salario;
 }
}

//Muestra N:Pepe S:700 console.log(empleado.toString());

empleado.getCategoria = function(){
  return this.salario > 800 ? 'Superior':'Normal';
}

console.log('C:'+empleado.getCategoria());
```

Orientación a Objetos

- Como se puede ver, los **objetos son muy flexibles** porque **no necesitan** un **molde** (una clase)
- Esta forma de trabajar es muy útil cuando **sólo hay un objeto** con una estructura determinada (***singleton***)
- Se usa bastante en JavaScript para crear **objetos sin métodos** que guardan información (como los **registros** o **struts**)

Ejercicio 5

- Se quiere implementar un programa en JavaScript que calcule el área y perímetro de un rectángulo
- El rectángulo se representará como un objeto con los atributos:
  - color
  - alto
  - ancho
- Y métodos:
  - area(): largo * ancho
  - perimetro(): 2 * ancho + 2 * largo

Orientación a Objetos

code URJC

- Crear varios objetos con los mismos atributos y métodos

Ejemplo25

```
let empleado = nuevoEmpleado("Pepe",700);

//Devuelve 'Nombre:Pepe, Salario:700'
console.log(empleado.toString());

//Devuelve 700 console.log(empleado.salario);
```

Orientación a Objetos

- Crear varios objetos con los mismos atributos y métodos

Ejemplo25

```
function nuevoEmpleado(nombre, salario){

   var empleado = {  nombre: nombre, salario: salario,
      toString: function(){
     return "N:"+this.nombre+" S:"+this.salario;
   }
};

return empleado;
}
```

Orientación a Objetos

code URJC

- **Objetos, null y undefined**
  - En JavaScript también se puede usar **null** como un valor válido para las variables
  - Las variables que no están inicializadas tienen el valor **undefined**
  - Usar una variable sólo si apunta a un objeto

```
let obj = null;
if(obj){
   obj.doSomething();
}
```

Devuelve true si obj no es **null** ni es **undefined**

Ejercicio 5b

- Se quiere implementar un programa en JavaScript que permita analizar un array de rectángulos
- De cada rectángulo se debe conocer:
  - Color Alto Ancho

  Área: largo * ancho

  Perímetro: 2 * ancho + 2 * largo
  -
  -

Ejercicio 5b

- Los **análisis** serán:
  - Suma total de áreas de los rectángulos
  - Suma total de perímetros de los rectángulos
  - Área media
  - Perímetro medio

- Prueba del **correcto funcionamiento**:
  - Construir un array con diferentes rectángulos
  - Ejecutar todos los análisis
  - Muestrar el resultado en la consola

Orientación a Objetos con clases

code URJC

- JavaScript también permite implementar clases (similares a las de Java)
- Se pueden crear objetos como instancia de una clase (eso hace que tenga sus métodos y atributos)
- Al igual que en Java se permite:
  - Constructor y métodos
  - Herencia de clases
  - Métodos/Atributos estáticos

Orientación a Objetos con clases

code URJC

Clase en JavaScript

```
class Empleado {



    constructor(nombre, salario){

    this.nombre = nombre; this.salario =
        salario;
}

getNombre(){
    return this.nombre;
}

toString(){
        return "Nombre:"+this.nombre+ ",
            Salario:"+this.salario;
}
}
```

Clase en Java

```
public class Empleado {

   private String nombre;
   private double salario;

   public Empleado(String nombre, double salario){

      this.nombre = nombre;
      this.salario = salario;
   }

   public String getNombre(){
      return nombre;
   }

   public String toString(){
      return "Nombre:"+nombre+
     ", Salario:"+salario;
   }
}
```

Orientación a Objetos con clases

```
class Shape {

    constructor (id, x, y) { this.id =
        id this.move(x, y)
}

move (x, y) {
    this.x = x this.y = y
}

toString() {
    return 'Shape('+this.id+')'
}
}
```

Ejemplo26

Orientación a Objetos con clases

code URJC

Ejemplo26

```
class Rectangle extends Shape {

    constructor (id, x, y, width, height) { super(id,
        x, y)
    this.width      = width this.height = height
}

toString () {
    return "Rectangle > " + super.toString()
}
}
```

Orientación a Objetos con clases

code URJC

Ejemplo26

```
class Rectangle extends Shape {
    …
    static defaultRectangle () {
        return new Rectangle("default", 0, 0, 100, 100)
    }
}

let defRectangle = Rectangle.defaultRectangle()
```

Orientación a Objetos con clases

code URJC

Métodos y atributos privados

```
class ClassWithPrivate { #privateField;
  #privateFieldWithInitializer = 42;

  #privateMethod() { This.#privateFiedl = "default";
}

static #privateStaticField;
static #privateStaticFieldWithInitializer = 42;

static #privateStaticMethod() {
  // …
}
}
```

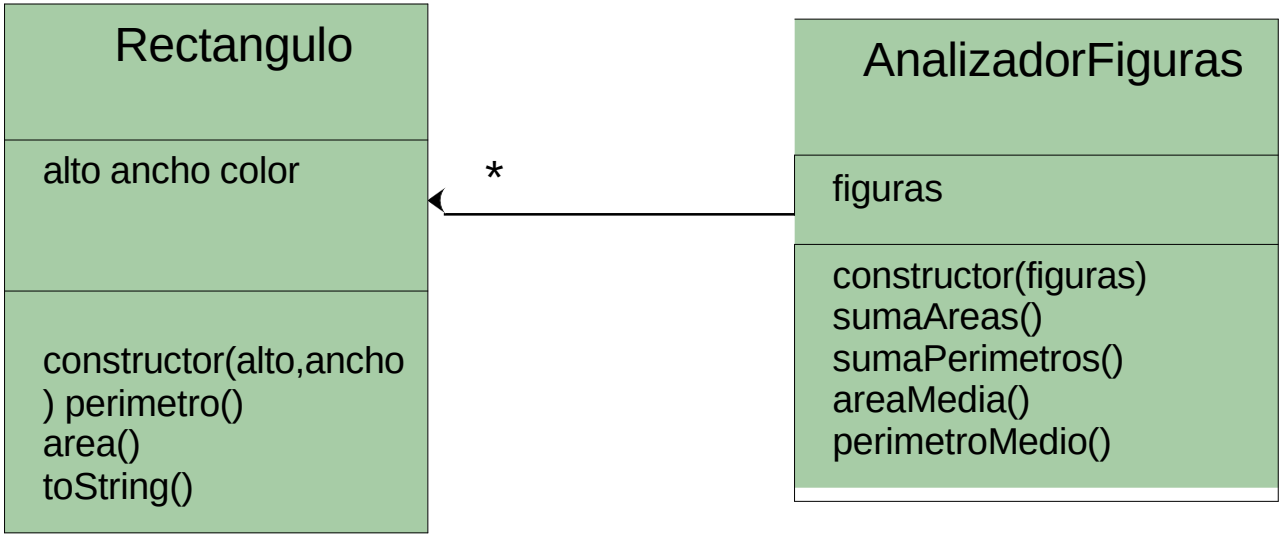
Orientación a Objetos con clases

code URJC

- En JavaScript **no existe** algo equivalente a los **interfaces de Java**
- No son necesarios con **tipado dinámico**

Ejercicio 6

- Transforma el Ejercicio 5 para que use clases
- Agrupa las funciones de análisis en la clase AnalizadorFiguras



Ejercicio 7

- Amplía el Ejercicio 6 con las siguientes funcionalidades:
  - Añadir el círculo como nuevo tipo de figura
    - Área: Math.PI * radio * radio
    - Perímetro: 2 * Math.PI * radio
  - Añadir un nuevo análisis:
    - Por cada figura saber el ratio area/perímetro
    - La media del ratio área/perímetro para todas las figuras

Ejercicio 7

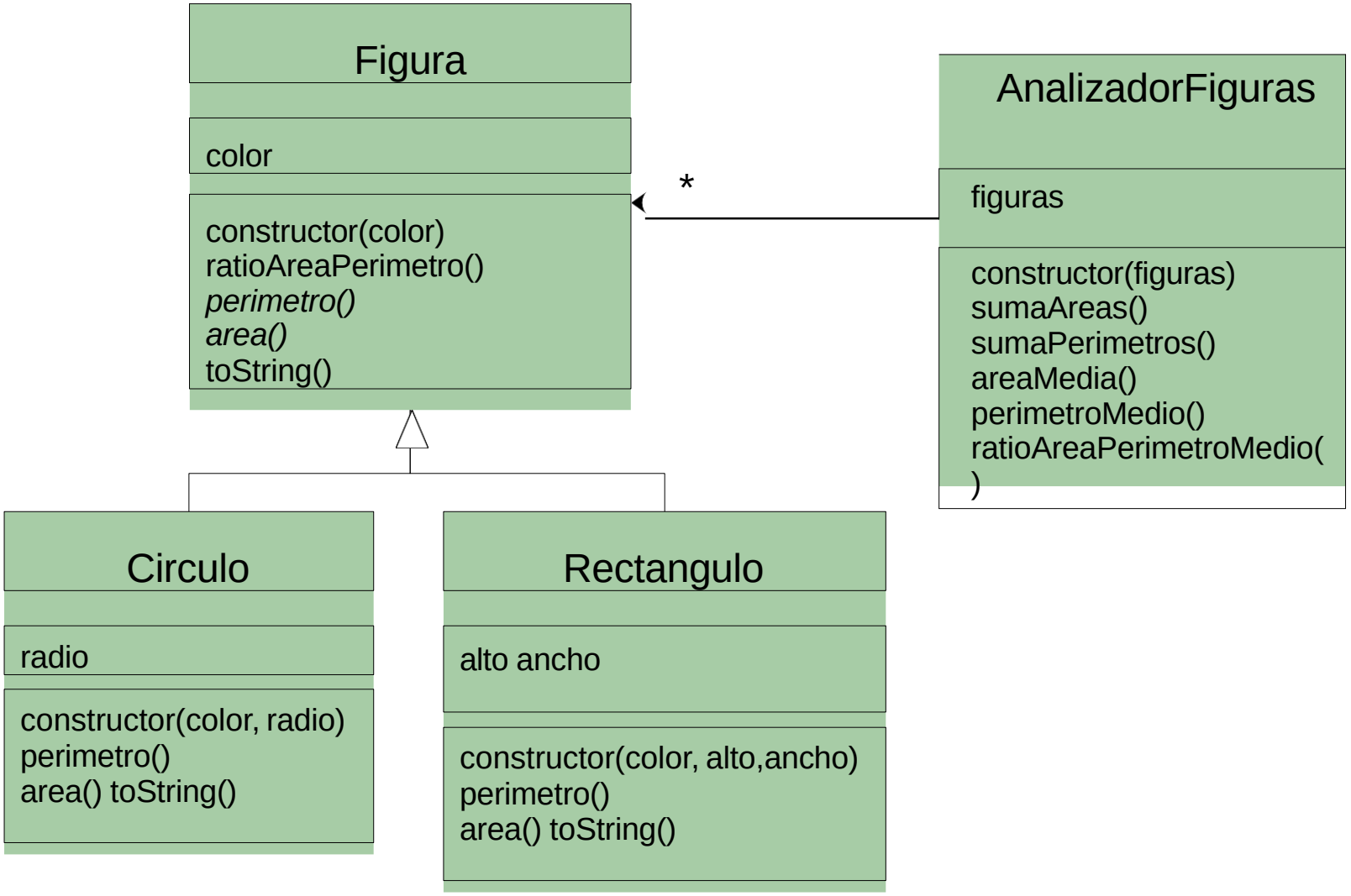


Figura
color
constructor(color)
ratioAreaPerimetro()
perimetro()
area()
toString()
*
AnalizadorFiguras
figuras
constructor(figuras)
sumaAreas()
sumaPerimetros()
areaMedia()
perimetroMedio()
ratioAreaPerimetroMedio(
)
Circulo
radio
constructor(color, radio)
perimetro()
area() toString()
Rectangulo
alto ancho
constructor(color, alto,ancho)
perimetro()
area() toString()

08-2023

code URJC

Fundamentos de la Web

Bloque III: Tecnologías de servidor web

Tema 3.2: Estructuras de datos en JavaScript

Universidad Rey Juan Carlos

Micael Gallego
Correo: micael.gallego@urjc.es
Twitter: @micael_gallego

Iván Chicano
ivan.chicano@urjc.es
Michel Maes
michel.maes@urjc.es

08-2023

code
URJC

Micael Gallego, Iván Chicano, Michel Maes

Estructuras de datos

code URJC

- **Arrays, conjuntos y mapas**
- Recorrer una estructura de datos
- Ordenación y Búsqueda

Arrays, conjuntos y mapas

code URJC

- Las **estructuras de datos** permiten almacenar **colecciones** de elementos en memoria
- Existen **varios tipos** de estructuras en función de su comportamiento
- En **JavaScript** desde **ES6** existen las siguientes estructuras de datos en la API estándar:

Array Set Map

Arrays, conjuntos y mapas

code URJC

- **Array**
  - Puede contener elementos duplicados y se puede acceder por posición
- **Set**
  - No puede tener dos o más objetos iguales
  - Se puede preguntar por la existencia de un elemento de forma rápida
- **Map**
  - Asocia valores a claves
  - El acceso del valor asociado a la clave es muy rápido

Arrays

code URJC

- Colección que **mantiene el orden de inserción** y que puede contener elementos **duplicados**
- Se accede a los elementos indicando su **posición**
- **Crece de forma dinámica.** No es necesario especificar su tamaño.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

Arrays

- Es la estructura de datos **más usada**
- Es la estructura de datos **más eficiente para la inserción** de elementos (al final)
- No obstante, **no** es muy **eficiente** para **búsquedas** (porque son secuenciales)

Arrays

code URJC

- Sintaxis especial

```
let msgArray = [];
msgArray[0] = 'Hello';
msgArray[1] = 'Bye';

console.log(msgArray[0]); // 'Hello'
console.log(msgArray[2]); // undefined
console.log(msgArray.length); // 2
```

```
let msgArray = new Array();
msgArray[0] = 'Hello';
msgArray[99] = 'world';

console.log(msgArray[2]); // undefined
console.log(msgArray.length); // 100
```

Arrays

- **push()**
  - Añade uno o más elementos al final del array y devuelve la nueva longitud del array

```
let sports = ['soccer', 'baseball'];
let total = sports.push('football', 'swim');

console.log(sports); // ['soccer', 'baseball', 'football', 'swim']
console.log(total); // 4
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/push

Arrays

- **pop()**
  - Elimina un elemento del **final** del array y lo devuelve

```
let myFish = ['angel', 'clown', 'mandarin', 'sturgeon'];

let popped = myFish.pop();

console.log(myFish); // ['angel', 'clown', 'mandarin' ]
console.log(popped); // 'sturgeon'
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/pop

Arrays

code URJC

- **shift()**
  - Elimina un elemento del **inicio** del array y lo devuelve

```
let myFish = ['angel', 'clown', 'mandarin', 'sturgeon'];

let shifted = myFish.shift();

console.log(myFish); // ['clown', 'mandarin', 'sturgeon']
console.log(shifted); // 'angel'
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/shift

Arrays

code URJC

- **splice()**
  - Añade y/o elimina elementos del array

```
var deletedItems = array.splice(start[, deleteCount[, item1[, item2[, ...]]]])
```

Borrar elementos

```
let myFish = ['angel', 'clown', 'drum', 'mandarin', 'sturgeon'];
let removed = myFish.splice(3, 1);

// removed is ["mandarin"]
// myFish is ["angel", "clown", "drum", "sturgeon"]
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice

Arrays

- **splice()**
  - Añade y/o elimina elementos del array

```
var deletedItems = array.splice(start[, deleteCount[, item1[, item2[, ...]]]])
```

Sustituir elementos

```
let myFish = ['angel', 'clown', 'drum', 'sturgeon'];
let removed = myFish.splice(2, 1, 'trumpet');

// myFish is ["angel", "clown", "trumpet", "sturgeon"]
// removed is ["drum"]
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice

Arrays

code URJC

- **splice()**
  - Añade y/o elimina elementos del array

```
var deletedItems = array.splice(start[, deleteCount[, item1[, item2[, ...]]]])
```

Añadir elementos

```
let myFish = ['angel', 'clown', 'mandarin', 'sturgeon'];
let removed = myFish.splice(2, 0, 'drum');

// myFish is ["angel", "clown", "drum", "mandarin", "sturgeon"]
// removed is [], no elements removed
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice

Conjuntos (Set)

code URJC

- No admite elementos **duplicados**
- Si se añade un elemento al conjunto y ya había otro **igual**, no se produce ningún cambio en el conjunto
- Es muy **eficiente** buscando entre sus elementos
- Pero eso hace que la **inserción** sea un poco más **costosa** que en los Arrays

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set

Conjuntos (Set)

code URJC

- **add()**
  - Añade un elemento al final de un Set

```
let mySet = new Set();
mySet.add(1);
mySet.add(5).add('some text'); // chainable

console.log(mySet); // Set [1, 5, "some text"]
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set/add

Conjuntos (Set)

code URJC

- **Para saber si un elemento es igual a otro del conjunto se usa el operador ===**
  - Dos strings son === si tienen los mismos caracteres
  - Dos objetos o arrays son === sin son "el mismo" objeto
  - Dos objetos u arrays con los mismos valores, no son ===

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set

Conjuntos (Set)

code URJC

- **size**
  - Devuelve los elementos del Set

```
let mySet = new Set();
mySet.add(1);
mySet.add(5);
mySet.add(5); //duplicated
mySet.add('some text');
mySet.add('some text'); //duplicated

let o = {a: 1, b: 2};
mySet.add(o);
mySet.add(o); //duplicated
mySet.add({a: 1, b: 2}); // same values but new object is inserted

mySet.size; // 5
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set/size

Conjuntos (Set)

code URJC

- **delete()**
  - Borra el elemento (si existe)

```
let mySet = new Set();
mySet.add('foo');

mySet.delete('bar'); // Returns false. No "bar" element found.
mySet.delete('foo'); // Returns true. Successfully removed.
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set/delete

Conjuntos (Set)

code URJC

- **Creación de Set**
  - Desde un array

```
let myArray = ['value1', 'value1', 'value2'];
let mySet = new Set(myArray);
console.log(mySet); // Set ['value1', 'value2']
```

  - Desde un string

```
let text = 'India';
let mySet = new Set(text); // Set ['I', 'n', 'd', 'i', 'a']
mySet.size; // 5
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set

Conjuntos (Set)

- **Creación de array desde Set**
  - Con Array.from(set) o spread operator

```
let mySet = new Set();
mySet.add(1);
mySet.add(3);
mySet.add(5);
mySet.add(7);

let array1 = Array.from(mySet); // [1,3,5,7]

let array2 = [...mySet]; // [1,3,5,7]
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set

Ejercicio 1

- Implementar una función **`cuentaDistintas`** que reciba un array con nombres de provincias (que podrían estar repetidas) e indique cuántas provincias sin repetir hay en ese array
- Ejecuta la función con un array de ejemplo para verificar que funciona como se espera

Mapas (Map)

code URJC

- Define una estructura de datos que asocia (**mapea**) claves con valores
- **No** permite claves repetidas (===)
- Varias claves distintas pueden estar asociadas al mismo valor (**valores repetidos**)
- La búsqueda de un valor asociado a una clave es muy **eficiente**

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map

Mapas (Map)

- **set() and get():**
  - Permite asociar un valor a una clave y recuperar el valor posteriormente

```
let coches = new Map();

let toledo = new Coche('Seat', 'Toledo', 110);
let punto = new Coche('Fiat', 'Punto', 90);

coches.set('M-1233-YYY', toledo);
coches.set('M-1234-ZZZ', punto);

let c = coches.get('M-1234-ZZZ'); // Coche ['Fiat', 'Punto', 90]
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map/get
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map/set

Mapas (Map)

code URJC

Ejemplo1

- set() and get():

```
let myMap = new Map();
myMap.set('a', { name: 'alpha'});
myMap.set('b', { name: 'beta'});
myMap.set('g', { name: 'gamma'});

console.log(myMap);

let myMap2 = new Map();
myMap2.set('a', new Date(1995,11,17));
myMap2.set('b', new Date(1920,11,17));
myMap2.set('g', new Date(2020,10,20));

console.log(myMap2);
```

Mapas (Map)

- **size:**
  - Devuelve el número de pares clave / valor

```
let myMap = new Map();
myMap.set('a', 'alpha');
myMap.set('b', 'beta');
myMap.set('g', 'gamma');

myMap.size // 3
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map/size

Mapas (Map)

- **delete():**
  - Borra una clave y su valor asociado

```
let myMap = new Map();

myMap.set('bar', 'foo');

myMap.delete('bar'); // Returns true. Successfully removed.

console.log(myMap); // Map []
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map/delete

Mapas (Map)

- **Creación de Map**
  - Desde un array

```
let kvArray = [['key1', 'value1'], ['key2', 'value2']];

//Create a map from 2D key-value Array
let myMap = new Map(kvArray);

myMap.get('key1'); // returns "value1"
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map

Mapas (Map)

- **Creación de array desde Map**
  - Con Array.from(map) o spread operator

```
let myMap = new Map();
myApp.set('k1', 'val1');
myApp.set('k2', 'val2');

let array1 = Array.from(myMap); // [['k1','val1'], ['k2','val2']]

let array2 = [...myMap]; // [['k1','val1'], ['k2','val2']]
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map

Comparativa: Array, Set y Map

code URJC

	Array	Set	Map
Tamaño	`length`	`size`	`size`
Añadir	`push(elem)` `splice(...)`	`add(elem)`	`set(key,value)`

Eliminar	`shift()`
	`pop()`
	`splice(...)`
		`delete(elem)`	`delete(key)`

Ejercicio 2

- Se pide crear una clase **`GestorAeropuertos`** que permita almacenar aeropuertos (nombre y ciudad)
- Se desea poder acceder a la información completa de un aeropuerto por su nombre de la forma más eficiente posible
- Se debe declarar como atributo de GestorAeropuertos la estructura de datos más eficiente
- Introducir varios aeropuertos y verificar el funcionamiento correcto cuando se accede a uno de ellos usando su nombre

Estructuras de datos

code URJC

- Arrays, conjuntos y mapas
- **Recorrer una estructura de datos**
- Ordenación y Búsqueda

Recorrer una estructura de datos

code URJC

- Acceder a cada elemento de una estructura de datos depende de su tipo:
  - Array
    - Acceso por posición con **bucle for**
    - Acceso **secuencial**
  - Conjunto (Set)
    - Acceso **secuencial**
  - Mapa (Map)
    - Acceso secuencial a la **colección de valores**
    - Acceso secuencial al **conjunto de claves**
    - Acceso secuencial al **conjunto de entradas**

Recorrer un Array

code URJC

- Acceso por posición con **bucle for**

```
let ciudades = ['Ciudad Real', 'Madrid', 'Valencia'];

for (let i=0; i < ciudades.length; i++) {
   let ciudad = ciudades[i];
   console.log(ciudad);
}
```

Recorrer un Array

- Acceso secuencial con **for of**

```
let ciudades = ['Ciudad Real', 'Madrid', 'Valencia'];

for (let ciudad of ciudades) {
   console.log(ciudad);
}
```

- En general es la forma preferida. Más conciso

Recorrer un Array

- Acceso secuencial con **forEach(...)**

```
let ciudades = ['Ciudad Real', 'Madrid', 'Valencia'];

ciudades.forEach(ciudad => {
   console.log(ciudad);
});
```

- Ideal cuando se usa con más operaciones funcionales

Recorrer un Conjunto

code URJC

- Acceso secuencia con **for of** o **forEach(…)**

```
let ciudades = new Set();
ciudades.add('Ciudad Real');
ciudades.add('Madrid');
ciudades.add('Valencia');

for (let ciudad of ciudades) {
    console.log(ciudad);
}

ciudades.forEach(ciudad => {
    console.log(ciudad);
});
```

Se recorren en el orden de inserción

Recorrer un Mapa

code URJC

Ejemplo2

- Formas de recorrer un mapa
  - Acceso secuencial a la **colección de valores**

```
let myMap = new Map();
myMap.set('a', 'alpha');
myMap.set('b', 'beta');
myMap.set('g', 'gamma');

for(let value of myMap.values()){
    console.log(value);
}

//alpha
//beta
//gamma
```

Recorrer un Mapa

code URJC

Ejemplo3

- Formas de recorrer un mapa
  - Acceso secuencial a la **colección de claves**

```
let myMap = new Map();
myMap.set('a', 'alpha');
myMap.set('b', 'beta');
myMap.set('g', 'gamma');

for(let key of myMap.keys()){
    console.log(key,myMap.get(key));
}

//a alpha
//b beta
//g gamma
```

Recorrer un Mapa

code URJC

Ejemplo4

- Formas de recorrer un mapa
  - Acceso secuencial a la **colección de entradas**
  - Con *destructuring* queda más conciso

```
let myMap = new Map();
myMap.set('a', 'alpha');
myMap.set('b', 'beta');
myMap.set('g', 'gamma');

for(let [key, value] of myMap){
    console.log(key,value);
}

//a alpha
//b beta
//g gamma
```

Recorrer un Mapa

code URJC

- Formas de recorrer un mapa
  - **forEach(...) por cada entrada**

```
let myMap = new Map();
myMap.set('a', 'alpha');
myMap.set('b', 'beta');
myMap.set('g', 'gamma');

myMap.forEach((key, value) => {
    console.log(key,value);
});

//a alpha
//b beta
//g gamma
```

Estructuras de datos

code URJC

- Arrays, conjuntos y mapas
- Recorrer una estructura de datos
- **Ordenación y Búsqueda**

Ordenación y Búsqueda

code URJC

Ejemplo5

- Sólo se pueden ordenar los **Arrays**
- Método **sort(…)** de Array

```
let nombres = ['Pepe','Juanin','Antonio'];

nombres.sort();

console.log(nombres); // ['Antonio', 'Juanin', 'Pepe']
```

Ordenación y Búsqueda

code URJC

- Se puede especificar el **orden de comparación**
  - Se usa una función que devuelve un valor positivo si o1 es mayor que o2. Negativo en caso contrario

Ejemplo6

Ordenar por longitud de los nombres

```
let nombres = ['Pepe','Juanin','Antonio'];

nombres.sort((s1,s2) => s1.length - s2.length);

console.log(nombres); // ['Pepe', 'Juanin', 'Antonio']
```

Ordenación y Búsqueda

code URJC

- **¿Qué es una búsqueda?**
  - **Array:** Saber la posición de un elemento (si está en el array)
  - **Conjunto:** Saber si está el elemento
  - **Map:** Saber el valor asociado a la clave (si está)

Ordenación y Búsqueda

code URJC

- **Búsqueda en conjuntos (has(...))**

```
let mySet = new Set();
mySet.add('foo');

mySet.has('foo'); // returns true
mySet.has('bar'); // returns false

let set1 = new Set();
let obj1 = {'key1': 1};
set1.add(obj1);

set1.has(obj1); // returns true
set1.has({'key1': 1}); // returns false because they are
                       // different object references
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set/has

Ordenación y Búsqueda

code URJC

- **Búsqueda en mapas (get(…) y has(…))**

```
let myMap = new Map();
myMap.set('bar', 'foo');

myMap.has('bar'); // returns true
myMap.has('baz'); // returns false
```

```
let myMap = new Map();
myMap.set('bar', 'foo');

myMap.get('bar'); // Returns "foo".
myMap.get('baz'); // Returns undefined
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set/has

Ordenación y Búsqueda

code URJC

- **Búsquedas en Arrays**
  - array.indexOf(searchElement[, fromIndex])
  - Se busca la posición del primer elemento === desde la posición indicada

```
let array = [2, 9, 9];
array.indexOf(2); // 0
array.indexOf(7); // -1
array.indexOf(9, 2); // 2
array.indexOf(2, -1); // -1
array.indexOf(2, -3); // 0
```

Ordenación y Búsqueda

code URJC

- Hay que **elegir muy bien** la estructura de datos que se utiliza en un programa
  - Arrays
    - Eficiente la inserción al final O(1)
    - Eficiente el acceso por posición O(1)
    - Ineficiente la búsqueda O(n)
  - Conjuntos
    - Eficiente la inserción O(1) (aunque menos que la lista)
    - No se puede hacer acceso por posición
    - Eficiente la búsqueda O(1)
  - Mapas
    - Igual que los conjuntos

Ordenación y Búsqueda

code URJC

	Array	Set	Map
Acceso por posición	Eficiente **O(1)**	No se puede	No se puede
Modificación	- Eficiente inserción/borrado al final **O(1)** - Ineficiente cualquier otra modificación **O(n)**	- Eficiente inserción/borrado. Pero más costosa que el Array **O(1)**	- Eficiente inserción/borrado. Pero más costosa que el Array **O(1)**
Búsqueda	Ineficiente **O(n)**	Eficiente **O(1)**	Eficiente **O(1)**

Ejercicio 3

- Implementar una aplicación que permita gestionar en memoria un conjunto de viajes de una aerolínea
- Cada viaje se representa con la ciudad origen, destino y la duración del viaje
- Se dan de alta los viajes en un gestor (clase **`GestorViajes`**)
- Al gestor de viajes se le pueden pedir:
  - Devolver un array con los viajes que tienen una determinada ciudad origen
  - Devolver un array con los viajes que tienen una determinada ciudad destino
  - Devolver un array con los viajes
  - Devolver un array con las ciudades en las que hay viajes
- Hay que conseguir el **menor tiempo de ejecución** de las consultas, aunque sean necesarias varias estructuras de datos

08-2023

code URJC

Fundamentos de la Web

Bloque III: Tecnologías de servidor web

Tema 3.3: Módulos en JavaScript

Universidad Rey Juan Carlos

Micael Gallego
Correo: micael.gallego@urjc.es
Twitter: @micael_gallego

Iván Chicano
ivan.chicano@urjc.es
Michel Maes
michel.maes@urjc.es

08-2023

code
URJC

Micael Gallego, Iván Chicano, Michel Maes

Módulos

Tipos de Módulos en Node.js

CommonsJS **ES Modules**

https://blog.logrocket.com/es-modules-in-node-today/

Módulos

Tipos de Módulos en Node.js

- Cuando se desarrolló Node, JavaScript no tenía un sistema estándar para modularizar aplicaciones
- Node diseñó su propio sistema de modularización llamado **CommonsJS**
- En **ES6 (2015)** se definió el sistema de módulos estándar llamado **ES Modules (EcmaScript Modules, ESM)**
- Node soporta ES Modules desde **13.2.0 (Nov 2019)**

Módulos

code URJC

Módulos Node (CommonsJS)

- En Node cada **fichero** es un **módulo**
- Si un módulo A quiere usar variables o funciones de un módulo B:
- El módulo B tiene que **exportar** lo que quiera hacer público

```
exports = ...
```

```
module.exports = ...
```

- El módulo A tiene que **importar** lo que quiera usar

```
var x = require('./module.js')
```

Módulos

Módulos Node (CommonsJS)

Ejemplo1

- 1) El código del módulo se ejecutar al importarlo (**require**)

hello.js

```
console.log('Hello world');
```

app.js

```
require('./hello.js');
```

app2.js

```
require('./hello');
```

La extensión del fichero se puede omitir

Módulos

Módulos Node (CommonsJS)

Ejemplo2

- 2) Para exportar **una única función anónima (default export).** Se le da nombre al importarla

bar.js

```
module.exports = function(){
    console.log('bar!');
}
```

app.js

```
var bar = require('./bar.js');
bar();
```

Módulos

Módulos Node (CommonsJS)

Ejemplo3A

- 3) Para exportar **una función con nombre**

fiz.js

```
exports.fiz = function(){
   console.log('fiz!');
}
```

app.js

```
var module = require('./fiz.js');
module.fiz();
```

Ejemplo3B

- 4) Otra forma de exportar una **función con nombre**

fiz.js

```
function fiz(){
   console.log('fiz!');
}
exports.fiz = fiz;
```

app.js

```
var fiz = require('./fiz.js').fiz;
fiz();
```

Módulos

Módulos Node (CommonsJS)

Ejemplo4

- 5) Se pueden exportar varios elementos y de diferentes tipos (funciones, objetos, valores, clases…)

utils.js

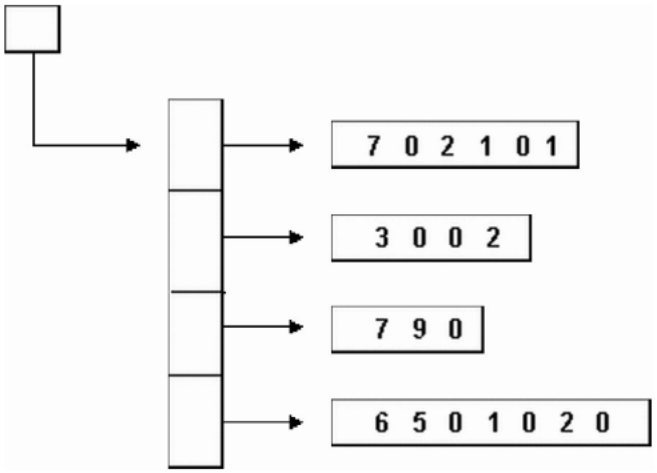
```
var obj = {
   prop1: 3,
   prop2: 'value'
}

function log(msg){
   console.log(msg);
}

class Date {
   constructor(date){
       this.date = date;
   }
}

exports.obj = obj;
exports.log = log;
exports.Date = Date;
```

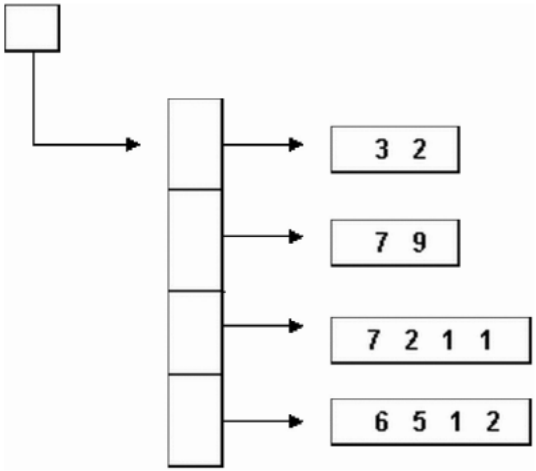
app.js

```
var utils = require('./utils.js');

utils.log('message');

console.log(utils.obj);

var date = new utils.Date('01-12-19');
```

Ejercicio 1

- Modifica el enunciado para incluir la función de quitar ceros en un módulo
- En el fichero **app.js** se usará ese módulo
- Exporta por **defecto** la función "quitaCeros" desde un módulo





Módulos

Librerías incluidas en Node

- Node expone la funcionalidad como **módulos (CommonsJS y ESM)**
- Se usa **require()** para usar esas librerías en tu módulo
- Existen algunos objetos **globales** que pueden usarse sin require()
- El objeto console está disponible **sin require**

```
console.log('Hello world!');
```

Módulos

Librerías incluidas en Node

Lectura de un fichero en Node con el módulo **File System** (fs)

```
var fs = require('fs');

fs.readFile('/home/data.txt', 'utf8', (err, contents) => {
   if (err) {
       return console.error(err);
   }
   console.log(contents);
});

console.log('After calling readFile');
```

Módulos

code URJC

Módulos NPM

- Herramienta que permite **descargar** módulos Node de la red
- Existe un **repositorio público** con módulos **software libre**
- También se pueden configurar **repositorios privados** para **módulos privados**

https://npmjs.com

Módulos

Node Package Manager (NPM)

- Una aplicación define los paquetes que necesita en un fichero llamado **package.json**

```
{
"name": "my-awesome-package",
"version": "1.0.0"
}
```

- El fichero se puede crear también con el comando

```
$ npm init
```

Módulos

Node Package Manager (NPM)

- Los paquetes de los que se depende se especifican en la sección "dependencies"

```
{
"name": "my-awesome-package",
"version": "1.0.0",
"dependencies": {
"lodash": "4.17.15"
}
}
```

- Para instalar el paquete

```
$ npm install
```

Módulos

Node Package Manager (NPM)

- Una vez instalado el paquete, ya se puede usar en el código del proyecto con "require()"

```
var lodash = require('lodash');

var output = lodash.without([1, 2, 3], 1);
console.log(output);
```

Módulos

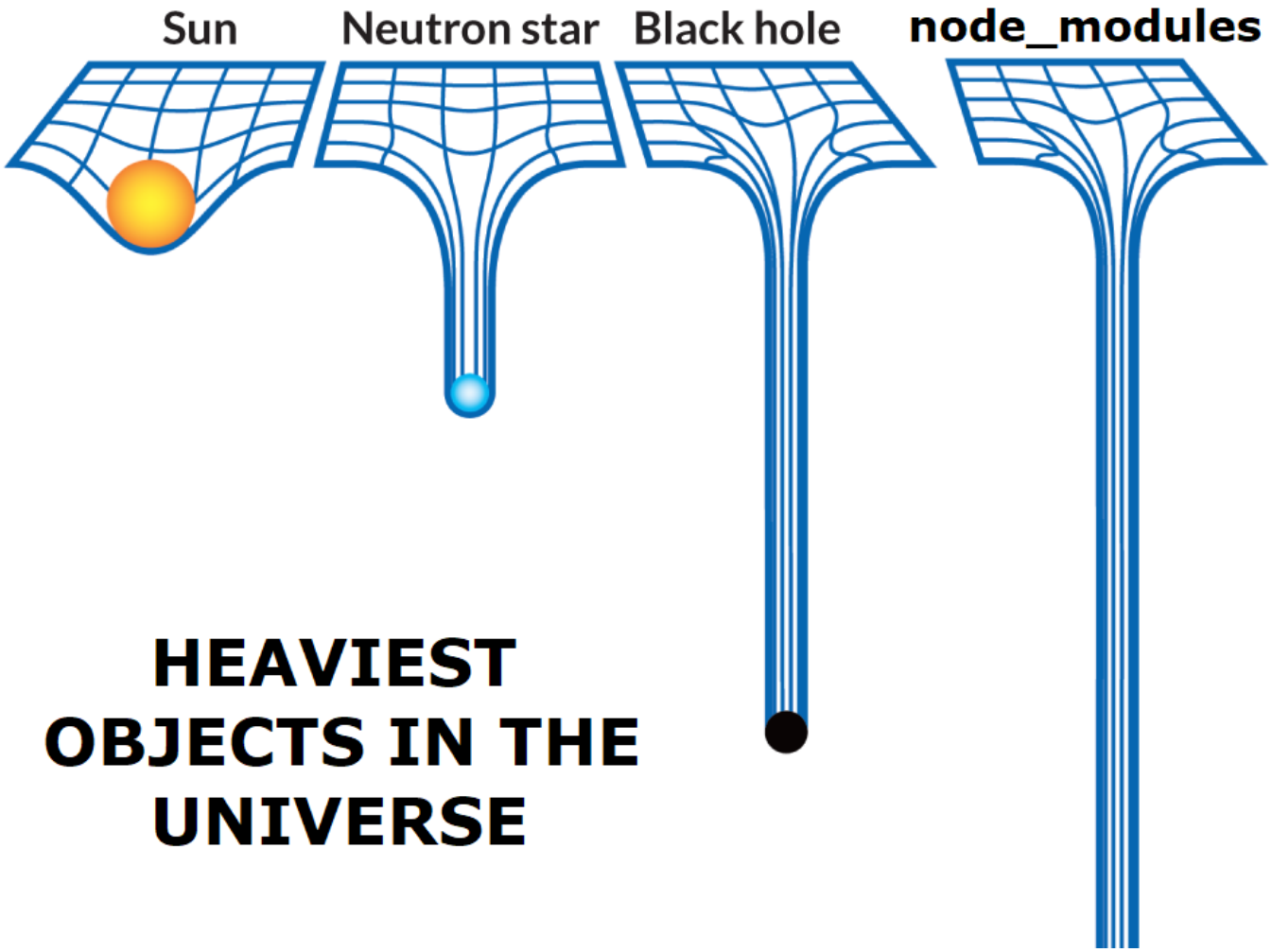
Node Package Manager (NPM)

- Los paquetes descargados se guardan en la carpeta **node_modules**
- Esta carpeta se suele incluir en el **.gitignore** porque estos paquetes no se suben al repositorio (porque se pueden descargar en cualquier momento desde la red con **npm install**)

Módulos

You have over 5000 changes in node_modules

Módulos


Sun
Neutron star
Black hole
node_modules
HEAVIEST OBJECTS IN THE UNIVERSE

Módulos

Node Package Manager (NPM)

- Se puede modificar el package.json y descargar automáticamente el paquete con

```
$ npm install --save left-pad
```

- Una ventaja es que obtiene automáticamente la última versión de la librería disponible en NPM

Módulos

Node Package Manager (NPM)

- Se pueden instalar paquetes que contienen **herramientas para desarrollo**, no para ejecutar la aplicación: gulp, webpack, browserify, angular-cli…
- Estos paquete se especifican en la sección "**devDependencies**" del package.json

```
{
"name": "my-awesome-package",
"version": "1.0.0"
"dependencies": {
"lodash": "4.17.15"
}
"devDependencies": {
"typescript": "3.7.2"
}
}
```

Módulos

Node Package Manager (NPM)

- Es habitual que algunas herramientas se instalen globalmente en el sistema, no de forma concreta en un proyecto (en linux necesitan **sudo**)

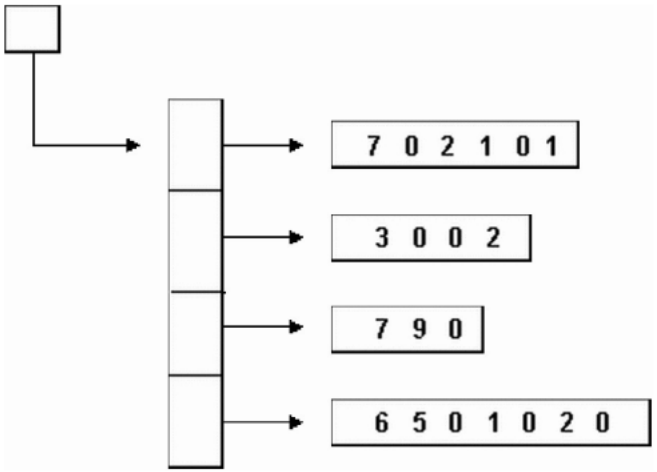
```
$ npm install -g @angular/cli
```

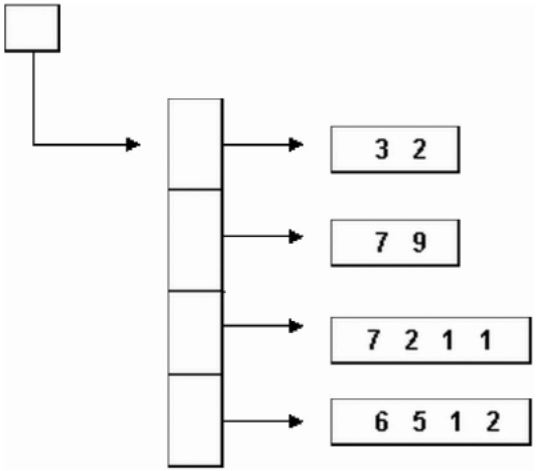
- Estas herramientas se registran de forma automática en el path y se pueden ejecutar en cualquier carpeta

```
$ ng --version
```

Ejercicio 2

- Reimplementa el ejercicio 1 para usar lodash para eliminar los ceros de los arrays





Módulos

Módulos EcmaScript (ES Modules)

- El soporte en Node se ha diseñado para que la **convivencia y transición** entre módulos **CommonJS y ES** sea lo más suave posible
- El objetivo es que una misma **aplicación** pueda tener **módulos de ambos tipos**
- Esta transición obliga a que existan **muchas situaciones** y **muchos detalles** a tener en cuenta (complejo)

Módulos

Módulos EcmaScript (ES Modules)

- Sólo estudiaremos el tipo más sencillo de aplicación:
  - Todos sus módulos ES
  - Usa módulos de la API de Node como módulos ES
  - Usa módulos de librerías externas implementados en CommonsJS

Módulos

Módulos EcmaScript (ES Modules) Ejemplo5

- Existen dos formas de implementar módulos ES
- 1) Extensión .mjs
- 2) Tipo "module" en el package.json **(preferida)**

package.json

```
{
    "name": "app",
    "version": "1.0.0",
    "type": "module"
}
```

Módulos

Módulos EcmaScript (ES Modules) Ejemplo5

- Conversión de módulos CommonsJS a ES
- 1) require() -> import

app.js

```
require('./hello.js');
```

hello.js

```
console.log('Hello world');
```

app.js

```
import './hello.js';
```

La extensión del fichero **NO** se puede omitir

Módulos

Módulos EcmaScript (ES Modules) Ejemplo5

- 2) Para exportar **una única función anónima (default export).** Se le da nombre al importarla

bar.js

```
module.exports = function(){
    console.log('bar!');
}
```

bar.js

```
export default function(){
    console.log('bar!');
}
```

app.js

```
var bar = require('./bar.js');
bar();
```

app.js

```
import bar from './bar.js';
bar();
```

Módulos

Módulos EcmaScript (ES Modules) Ejemplo5

- 3) Para exportar **una función con nombre**

fiz.js

```
exports.fiz = function(){
   console.log('fiz!');
}
```

app.js

```
var module = require('./fiz.js');
module.fiz();
```

fiz.js

```
export function fiz(){
   console.log('fiz!');
}
```

app.js

```
import * as module from './fiz.js';
module.fiz();
```

Módulos

Módulos EcmaScript (ES Modules) Ejemplo5

- 4) Otra forma de importar una **función con nombre**

fiz.js

```
exports.fiz = function(){
   console.log('fiz!');
}
```

app.js

```
var fiz = require('./fiz.js').fiz;
fiz();
```

fiz.js

```
export function fiz(){
   console.log('fiz!');
}
```

app.js

```
import { fiz } from './fiz.js';
fiz();
```

Módulos

Módulos EcmaScript (ES Modules) Ejemplo5

- 5) Se pueden exportar varios elementos y de diferentes tipos (funciones, objetos, valores, clases…)

utils.js

```
var obj = {
   prop1: 3,
   prop2: 'value'
}

function log(msg){
   console.log(msg);
}

class Date {
   constructor(date){
       this.date = date;
   }
}

exports.obj = obj;
exports.log = log;
exports.Date = Date;
```

utils.js

```
export var obj = {
   prop1: 3,
   prop2: 'value'
}

export function log(msg){
   console.log(msg);
}

export class Date {
   constructor(date){
       this.date = date;
   }
}
```

Módulos

Módulos EcmaScript (ES Modules) Ejemplo5

- 5) Se pueden exportar varios elementos y de diferentes tipos (funciones, objetos, valores, clases…)

app.js

```
var utils = require('./utils.js');

utils.log('message');

console.log(utils.obj);

var date = new utils.Date('01-12-19');
```

app.js

```
import * as utils from './utils.js';

utils.log('message');

console.log(utils.obj);

var date = new utils.Date('01-12-19');
```

Módulos

Módulos EcmaScript (ES Modules) Ejemplo5

- 5) Se pueden exportar varios elementos y de diferentes tipos (funciones, objetos, valores, clases…)

app.js

```
var utils = require('./utils.js');

utils.log('message');

console.log(utils.obj);

var date = new utils.Date('01-12-19');
```

app.js

```
import { log, obj, Date } from './utils.js';

log('message');

console.log(obj);

var date = new Date('01-12-19');
```

Módulos

Módulos EcmaScript (ES Modules) Ejemplo6

- 6) Además de los elementos con nombre, también se puede exportar un objeto "default"

utils.js

```
export var obj = {
    prop1: 3,
    prop2: 'value'
}

export function log(msg){
    console.log(msg);
}

export class Date {
    constructor(date){
        this.date = date;
    }
}

export default { obj, log, Date }
```

app.js

```
import * as utils from './utils.js';

utils.log('message');
console.log(utils.obj);
var date = new utils.Date('01-12-19');
```

app2.js

```
import utils from './utils.js';

utils.log('message');
console.log(utils.obj);
var date = new utils.Date('01-12-19');
```

Módulos

Módulos EcmaScript (ES Modules) Ejemplo7

- Las librerías de Node también se pueden importar como módulos ES

app.js

```
var lodash = require('lodash');

var output = lodash.without([1, 2, 3], 1);
console.log(output);
```

app.js

```
import lodash from 'lodash';

var output = lodash.without([1, 2, 3], 1);
console.log(output);
```

Módulos

Módulos EcmaScript (ES Modules) Ejemplo7

- No siempre se puede acceder a los elementos individuales con un **import**

app.js

```
var lodash = require('lodash');

var output = lodash.without([1, 2, 3], 1);
console.log(output);
```

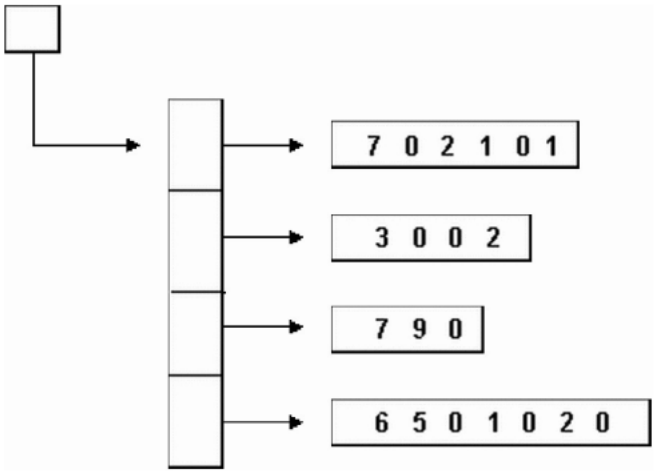
app.js

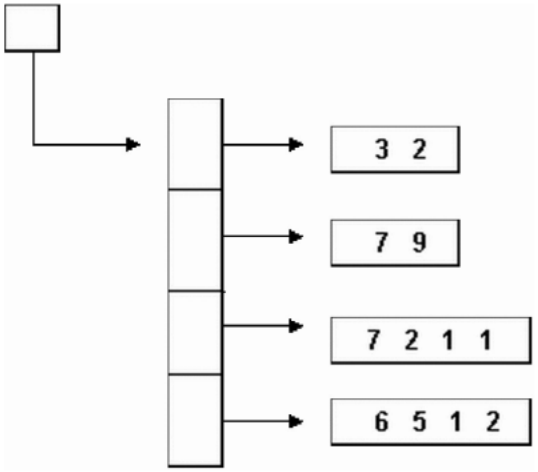
```
import { without } from 'lodash';

var output = without([1, 2, 3], 1);
console.log(output);
```

ERROR
lodash no exporta un elemento "without". Exporta un objeto "default"

Ejercicio 3

- Cambia el ejercicio 2 para que use módulos ES en vez de CommonJS





08-2023

code URJC

Fundamentos de la Web

Bloque III: Tecnologías de servidor web

Tema 3.2: Estructuras de datos en JavaScript

Universidad Rey Juan Carlos

Micael Gallego
Correo: micael.gallego@urjc.es
Twitter: @micael_gallego

Iván Chicano
ivan.chicano@urjc.es
Michel Maes
michel.maes@urjc.es

08-2023

code
URJC

Micael Gallego, Iván Chicano, Michel Maes

Aplicaciones web con Node

- Las librería estándar de Node permiten crear aplicaciones web
- Pero son muy limitadas y se suele usar el paquete NPM **express**

https://expressjs.com

Aplicaciones web con Express

Mínima aplicación web

Ejemplo1

app.js

```
import express from 'express';

const app = express();

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(3000, () => console.log('Listening on port 3000!'));
```

package.json

```
{
  "name": "web-ejem1",
  "version": "1.0.0",
  "type": "module",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

Aplicaciones web con Express

code URJC

Mínima aplicación web

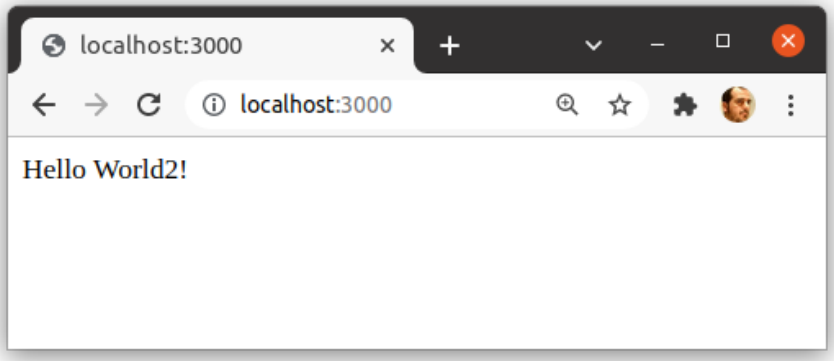
Ejemplo1

- Instalar las librerías

```
$ npm install
```

- Ejecutar la aplicación

```
$ node app.js
```



http://localhost:3000

Aplicaciones web con Express

code URJC

Mínima aplicación web

Ejemplo1

- Por cada cambio de código es necesario parar y ejecutar de nuevo la aplicación
- La herramienta **nodemon** detecta automáticamente los cambios y reinicia el servicio

```
$ sudo npm install -g nodemon
```

```
$ nodemon app.js
```

Templates y ficheros estáticos

code URJC

- Express es una librería **muy configurable**
- Es necesario configurar de forma **explícita**:
  - Servir ficheros estáticos por http
  - Procesado del body en las peticiones
  - Tecnología de plantillas

Templates y ficheros estáticos

code URJC

- **Mustache en Node**
  - Existen muchas otras tecnologías de plantillas
  - Se puede usar Mustache

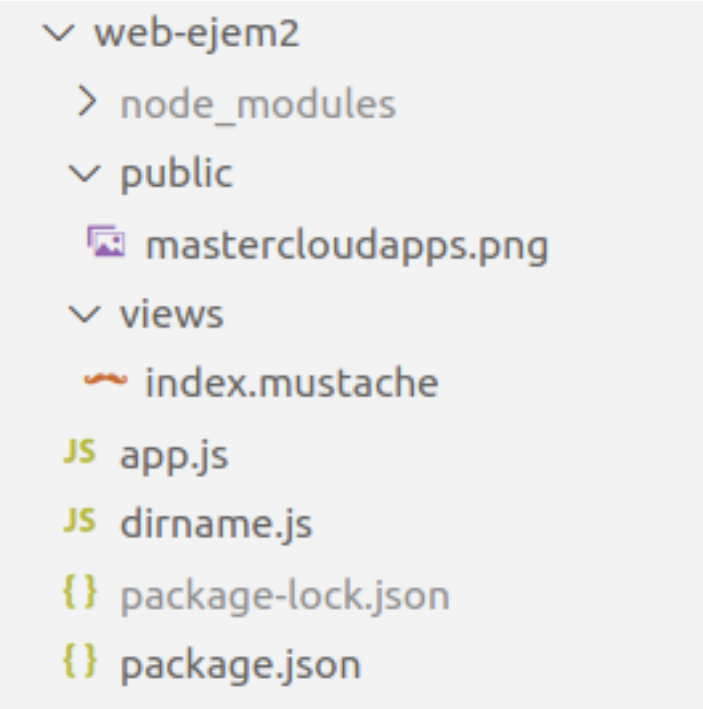
```
$ npm install --save mustache-express
```

  - Existen diferencias entre Mustache para Node y para otros lenguajes
    - P.e: No se puede usar **{{-index}}** en los bucles

http://expressjs.com/en/guide/using-template-engines.html

Templates y ficheros estáticos

Aplicación web básica con Mustache Ejemplo2



package.json

```
{
  "name": "web-ejem2",
  "version": "1.0.0",
  "type": "module",
  "dependencies": {
    "express": "^4.17.1",
    "mustache-express": "^1.3.0"
  }
}
```

Templates y ficheros estáticos

code URJC

Ejemplo2

```
import express from 'express';
import mustacheExpress from 'mustache-express';
import bodyParser from 'body-parser';
import { __dirname } from './dirname.js';

const app = express();

app.set('views', __dirname + '/views');
app.set('view engine', 'mustache');
app.engine('mustache', mustacheExpress());

app.use(bodyParser.urlencoded({ extended: true }));

app.use(express.static(__dirname + '/public'));

app.get('/', (req, res) => {
  res.render('index', {
     name: "World"
  });
});

app.listen(3000, () => console.log('Listening on port 3000!'));
```

Configuración de **Mustache**

Configuración del **analizador del body**

Configuración de **carpeta pública**

Templates y ficheros estáticos

code URJC

- **Rutas relativas**

Ejemplo2

  - Con CommonJS la variable global `__dirname` apunta a la carpeta del fichero .js
  - En ESM no existe esa variable global, pero su valor se puede obtener usando la API estándar

dirname.js

```
import { fileURLToPath } from 'url';
import { dirname } from 'path';

export const __dirname = dirname(fileURLToPath(import.meta.url));
```

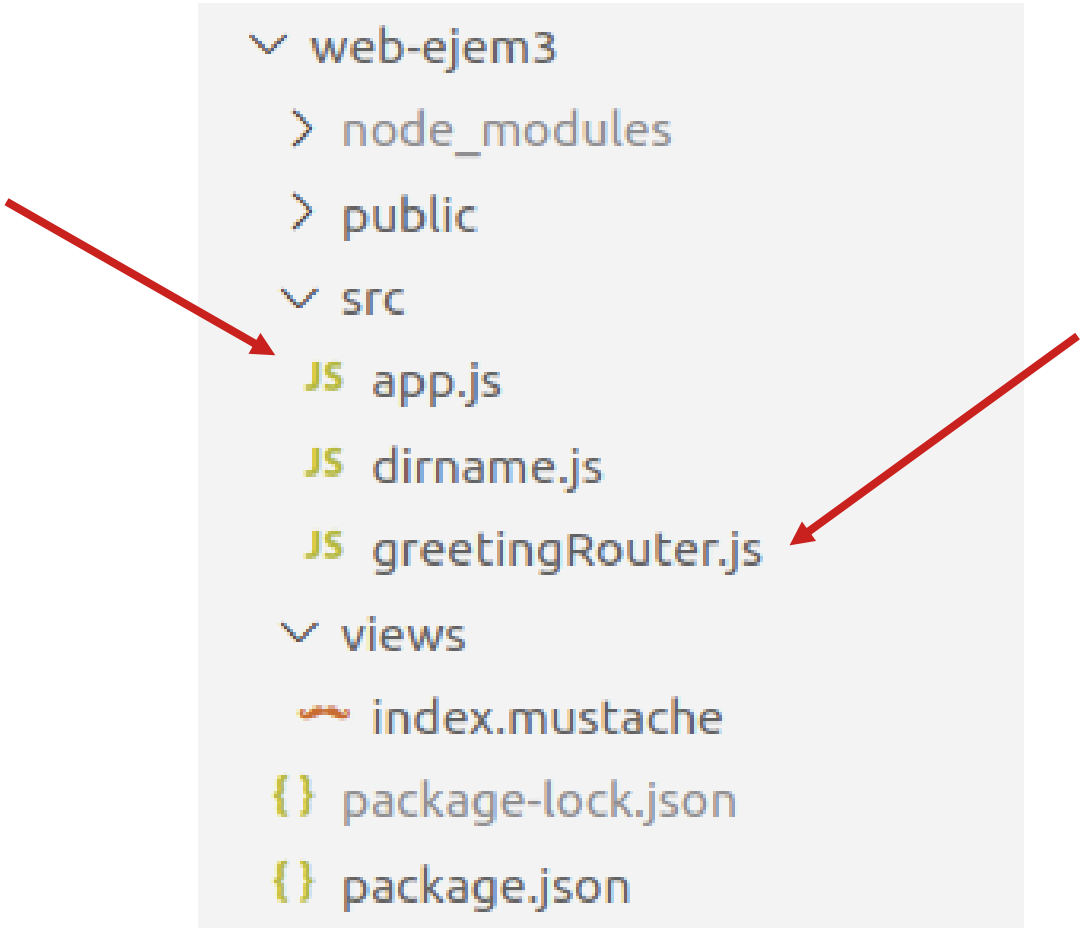
Separación en módulos

- Se recomienda dejar app.js para configurar el servidor
- La gestión de peticiones se define en otro módulo en un **Router**
- No se usa inyección de dependencias, se usan referencias entre módulos con **import**
- Los módulos se mueven a la carpeta **/src**

Separación en módulos

code URJC

Ejemplo3



Separación en módulos

Ejemplo3

src/app.js

```
import express from 'express';
import mustacheExpress from 'mustache-express';
import bodyParser from 'body-parser';
import { __dirname } from './dirname.js';
import greetingRouter from './greetingRouter.js';

const app = express();

app.set('views', __dirname + '/../views');
app.set('view engine', 'mustache');
app.engine('mustache', mustacheExpress());

app.use(bodyParser.urlencoded({ extended: true }));

app.use(express.static(__dirname + '/../public'));

app.use('/', greetingRouter);

app.listen(3000, () => console.log('Listening on port 3000!'));
```

src/greetingRouter.js

```
import express from 'express';

const router = express.Router();

router.get('/', (req, res) => {

   res.render('index', {
       name: "World"
   });
});

export default router;
```

Proceso de formularios y enlaces

code URJC

Ejemplo4

- **Acceso a los datos en el servidor**
  - Los valores se almacenan en **req.query** (formulario GET) o **req.body** (formulario POST)

public/index.html

```
<form action="greeting">
  <p>Saludar a :</p>
  <input type='text' name='userName'/>
  <input type='submit' value='Enviar'/>
</form>
```

```
router.get('/greeting', (req, res) => {

  res.render('greeting', {
    name: req.query.userName
  });
});
```

Parámetro con el valor del campo de texto del formulario

Proceso de formularios y enlaces

- Existen dos formas de enviar la información de un formulario al servidor
  - Método GET (Por defecto)

```
<form method="get" action="url_router">
  ...
</form>
```

  - Método POST

```
<form method="post" action="url_router">
  ...
</form>
```

Proceso de formularios y enlaces

code URJC

- **Existen dos formas de enviar la información de un formulario al servidor**
  - **Método GET (Por defecto)**
    - Se usa el método GET del protocol HTTP
    - El navegador incluye la información del formulario en la URL que solicita al servidor
  - **Método POST**
    - Se usa el método POST del protocol HTTP
    - El navegador incluye la información del formulario en el cuerpo de la petición (no es visible por el usuario)

Proceso de formularios y enlaces

code URJC

- Procesar diferentes métodos http

```
app.METHOD(PATH, HANDLER)
```

- Donde:
  - **app** es una instancia de express. También puede ser un router
  - **METHOD** es el método de solicitud HTTP
  - **PATH** es la ruta de acceso en el servidor
  - **HANDLER** es la función que se ejecuta cuando se acceder a la ruta

Ejercicio 1 – Tablón de mensajes

- Crear una aplicación web para gestionar un tablón de anuncios con varias páginas
- La página principal muestra los anuncios existentes (sólo nombre y asunto) y un enlace para insertar un nuevo anuncio
- Si pulsamos en la cabecera de un anuncio se navegará a una página nueva que muestre el contenido completo del anuncio

Ejercicio 1 – Tablón de mensajes

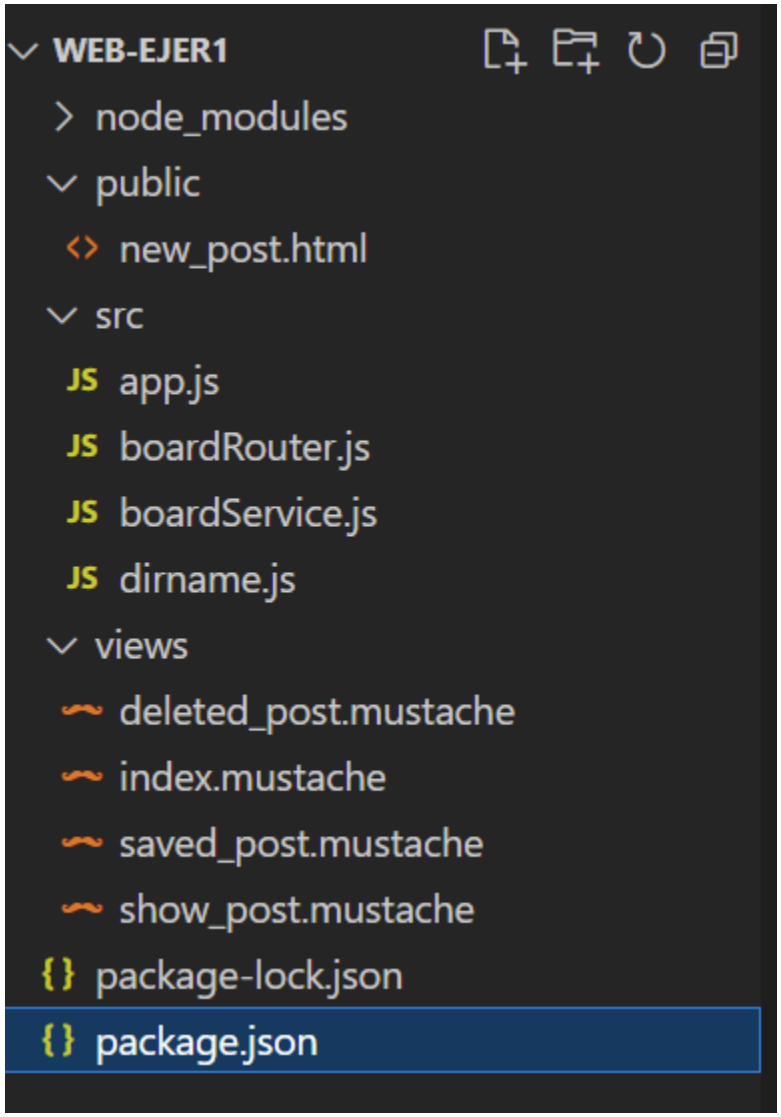
code URJC

- Si se pulsa el enlace para añadir el anuncio se navegará a una nueva página que contenga un formulario
- Al enviar el formulario se guardará el nuevo anuncio y se mostrará una página indicando que se ha insertado correctamente y un enlace para volver
- En la página del anuncio se podrá borrar

Ejercicio 1 – Tablón de mensajes

- **Implementación**
  - Se usará un único router con varias funciones (cada una atendiendo una URL diferente)
  - Se creará un módulo independiente para la gestión de los posts
  - Cada post tendrá asociado un id generado con un contador

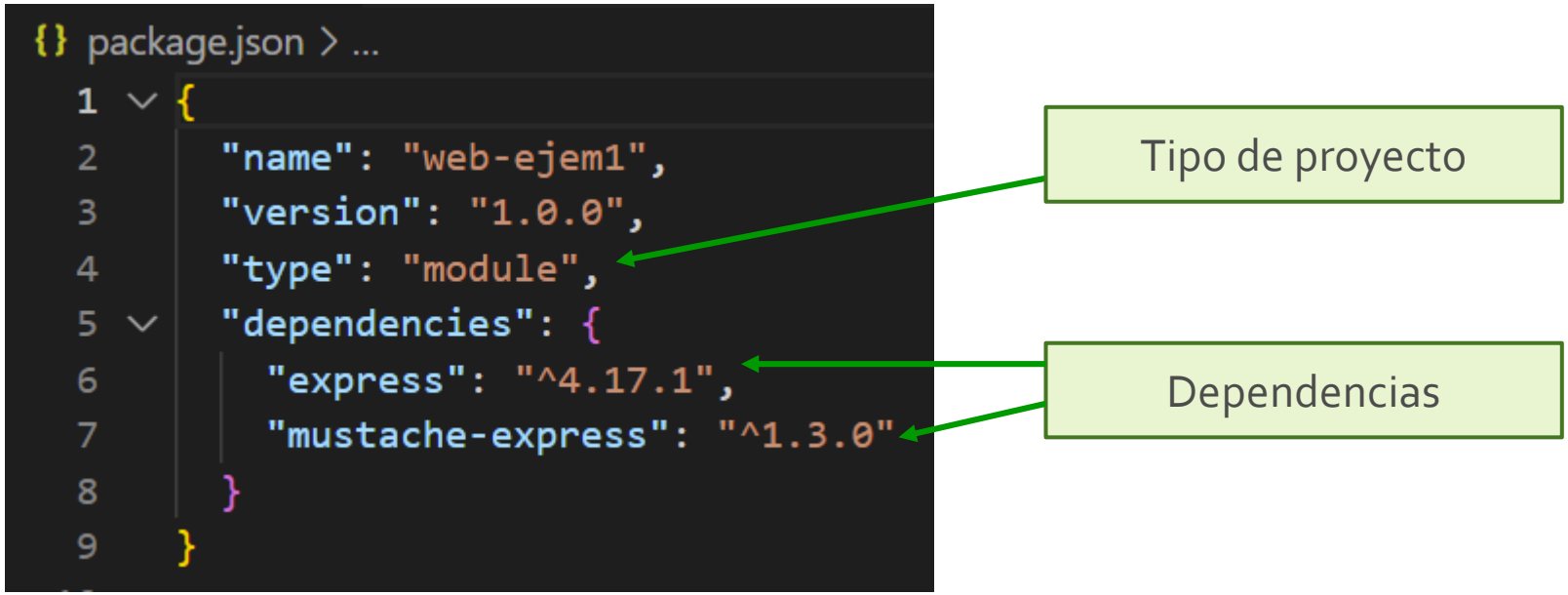
Ejercicio 1 – Tablón de mensajes



Estructura del proyecto:

- public: Carpeta con páginas accesibles para todo el mundo
- src: Código fuente del proyecto, con la funcionalidad en sí
- views: Carpeta con las páginas devueltas por los fuentes. Las plantillas con mustache siempre deberán estar en esta carpeta.

Ejercicio 1 – Tablón de mensajes



Sobre un proyecto ya creado es suficiente con:

```
$ npm install
```

```
$ node /src/app.js
```

Ejercicio 1 – Tablón de mensajes

public/new_post.html

```
<html>
<head>
    <meta charset="UTF-8"/>
</head>
<body>
    <form action="post/new" method="post">
        <p>User: </p>
        <input type='text' name='user'/>
        <p>Title:</p>
        <input type='text' name='title'/>
        <p>Text:</p>
        <textarea name='text' rows=5 cols=40></textarea>
        <input type='submit' value='Save'/>
    </form>
</body>
</html>
```

URL donde se recogen los datos

POST, dado que se envía información

Ejercicio 1 – Tablón de mensajes

src/app.js

```
import express from 'express';
import mustacheExpress from 'mustache-express';
import bodyParser from 'body-parser';
import { __dirname } from './dirname.js';
import boardRouter from './boardRouter.js';

const app = express();

app.set('views', __dirname + '/../views');
app.set('view engine', 'mustache');
app.engine('mustache', mustacheExpress());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static(__dirname + '/../public'));

app.use('/', boardRouter);

app.listen(3000, () => console.log('Listening on port 3000!'));
```

Configuración de mustache

Asociar URL – Router

Ejercicio 1 – Tablón de mensajes

src/boardRouter.js

```
import express from 'express';
import * as boardService from './boardService.js';
const router = express.Router();

router.get('/', (req, res) => {
    res.render('index', {
        posts: boardService.getPosts()
    });
});

router.post('/post/new', (req, res) => {
    let { user, title, text } = req.body;
    boardService.addPost({ user, title, text });
    res.render('saved_post');
});
```

URL – Página de inicio

Página a renderizar

URL asociada en app.js

Página a renderizar

Ejercicio 1 – Tablón de mensajes

src/boardRouter.js

```
router.get('/post/:id', (req, res) => {
    let post = boardService.getPost(req.params.id);
    res.render('show_post', { post });
});

router.get('/post/:id/delete', (req, res) => {
    boardService.deletePost(req.params.id);
    res.render('deleted_post');
});

export default router;
```

Método para obtener un elemento al completo

Método para borrar un elemento

Hay que tener en cuenta como pasa :id como parametro

Ejercicio 1 – Tablón de mensajes

src/boardService.js

```
const posts = new Map();
let nextId = 0;

addPost({
user: "Pepe", title: "Vendo moto", text: "Barata, barata" });
addPost({
user: "Juan", title: "Compro coche", text: "Pago bien" });
}
```

- Se utiliza un mapa en lugar de un lista dado que un id debe identificar de manera univocal un element
- El incremento del id, lo realizamos nosotros
- Poblamos el mapa que actúa como BBDD (aunque no hay persistencia de datos)

Ejercicio 1 – Tablón de mensajes

src/boardService.js

```
export function addPost(post) {
    let id = nextId++;
    post.id = id.toString();
    posts.set(post.id, post);
}

export function deletePost(id){
    posts.delete(id);
}

export function getPosts(){
    return [...posts.values()];
}

export function getPost(id){
    return posts.get(id);
}
```

- Se incrementa en 1 el id para el siguiente elemento, se guarda en el mapa.
- Se elimina un elemento del mapa
- Se obtiene todos los elementos del mapa.
- Se obtiene un elemento concreto del mapa (id)

Ejercicio 1 – Tablón de mensajes

code URJC

src/dirname.js

```
import { fileURLToPath } from 'url';
import { dirname } from 'path';

export const __dirname =
dirname(fileURLToPath(import.meta.url));
```

views/deleted_post.mustache

```
<html>
<body>
    <p>Post has been deleted.</p>
    <a href="/">Back</a>
</body>
</html>
```

views/saved_post.mustache

```
<html>
<body>
    <p>Post has been deleted.</p>
    <a href="/">Back</a>
</body>
</html>
```

Ejercicio 1 – Tablón de mensajes

view/show_post.mustache

```
<html>
<body>
    <p>User: {{post.user}}</p>
    <p>Title: {{post.title}}</p>
    <p>Text: {{post.text}}</

    <a href="/post/{{post.id}}/delete">Delete post</a>

    <br>

    <a href="/">Back</a>
</body>
</html>
```

{{var}} -> indica que hay una variable

{{var.campo}} -> indica campo de una variable

- Hay que tener en cuenta que post es lo que se manda al pedir renderizar la página (posts en este caso)

Ejercicio 1 – Tablón de mensajes

view/index.mustache

```
<html>
<body>
    <h1>Posts</h1>
    {{#posts}}
    <a href="post/{{id}}">{{user}} {{title}}</a><br>
    {{/posts}}

    {{^posts}}
    <p>No posts yet.</p>
    {{/posts}}

    <br>
    <a href="new_post.h
</body>
</html>
```

Región en mustache. Nos permite explorer todos los elementos de un array

Campos de la lista de elementos pasados por parámetros

Región condicional en mustache. En este caso, si no hay elementos en posts, se muestra el mensaje

08-2023

code URJC

Fundamentos de la Web

Bloque IV: Tecnologías de interactividad en el cliente web

Tema 4.1: HTML interactivo con JavaScript

Universidad Rey Juan Carlos

Micael Gallego
Correo: micael.gallego@urjc.es
Twitter: @micael_gallego

Iván Chicano
ivan.chicano@urjc.es
Michel Maes
michel.maes@urjc.es

08-2023

code
URJC

Micael Gallego, Iván Chicano, Michel Maes

JavaScript

code URJC

- Es un lenguaje de programación basado en el estándar **ECMAScript** de **ECMA** (otra organización diferente al **W3C**)
- Hay ligeras **diferencias** en la implementación de JS de los navegadores, aunque actualmente todos son bastante **compatibles** entre sí (**en el pasado no fue así**)

http://www.ecma-international.org/

JavaScript

- **Versiones de JS**
  - JavaScript es un lenguaje que mejora cada año Al
  - principio las versiones se numeraban con un

número: ECMAScript 5 (**ES5**), ECMAScript 6 (**ES6**)

- Pero cuando se publicó ES6 decidieron llamarlo con el año: **ES2015**
- Se publica una actualización cada año

JavaScript

code URJC

- **Navegadores web y versiones de JS**
  - Los navegadores web tardan un tiempo en soportar las últimas versiones de JS
  - En esta web se indica qué características soporta cada versión de cada navegador

    http://kangax.github.io/compat-table/es2016plus/

  - En clase veremos características soportadas por los navegadores más usados (Chrome, Firefox, Edge…)

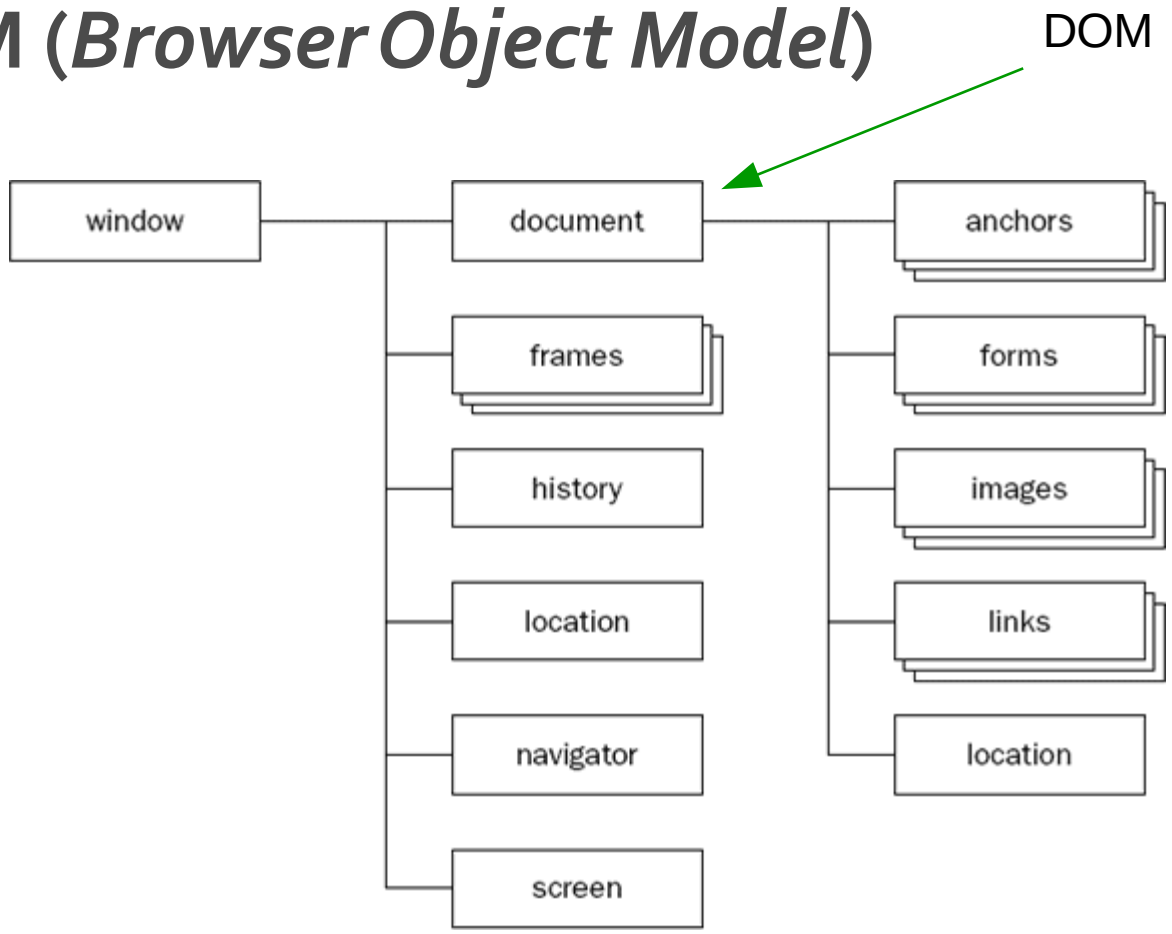
Librerías JavaScript

code URJC

- **DOM (*Document Object Model*)**
  - Biblioteca usada para manipular el documento HTML cargado en el navegador
  - Permite la gestión de eventos, insertar y eliminar elementos, cambiar propiedades..
- **BOM (*Browser Object Model*)**
  - Biblioteca usada para usar otros elementos del browser: historial, peticiones de red AJAX, etc…
  - El BOM incluye al DOM como uno de sus elementos

Librerías JavaScript

- BOM (*Browser Object Model*)



Librerías JavaScript

code URJC

- Existen multitud de **bibliotecas** (APIs) JavaScript para el desarrollo de aplicaciones
- Algunas de las más usadas son:

  - **jQuery**: es un recubrimiento de la API DOM que aporta facilidad de uso, potencia y compatibilidad entre navegadores. Se usa para gestionar el interfaz de usuario (la página web) y para peticiones ajax.
  - **underscore.js:** Librería para trabajar con estructuras de datos con un enfoque funcional. También permite gestionar plantillas (*templates*) para generar HTML partiendo de datos

Frameworks de alto nivel

- Además de bibliotecas, también existen **frameworks del alto nivel** que estructuran una aplicación de forma completa. Especialmente en **aplicaciones SPA**

Vue.js

Referencias

- **Documentación**
  - https://developer.mozilla.org/es/docs/Web/JavaScript
  - https://es.javascript.info/
- **Tutoriales interactivos**
  - https://www.codecademy.com/
  - https://www.pluralsight.com/courses/javascript-getting-started

Integración con HTML

- El código **JavaScript** se incluye en etiquetas **<script>** en el **<body>** o **<head>**
- Se puede incluir en el fichero HTML o en ficheros .js
- Se puede añadir en varios lugares (pero no es necesario)

index.html

```
<html>
  <head>
    <script>console.log("Head1");</script>
    <script src="app1.js"></script>
  </head>
  <body>
    <script>console.log("Body1");</script>
    <script src="app2.js"></script>
  </body>
</html>
```

Ejemplo1

app1.js

```
console.log("Head2");
```

app2.js

```
console.log("Body2");
```

Sintaxis básica

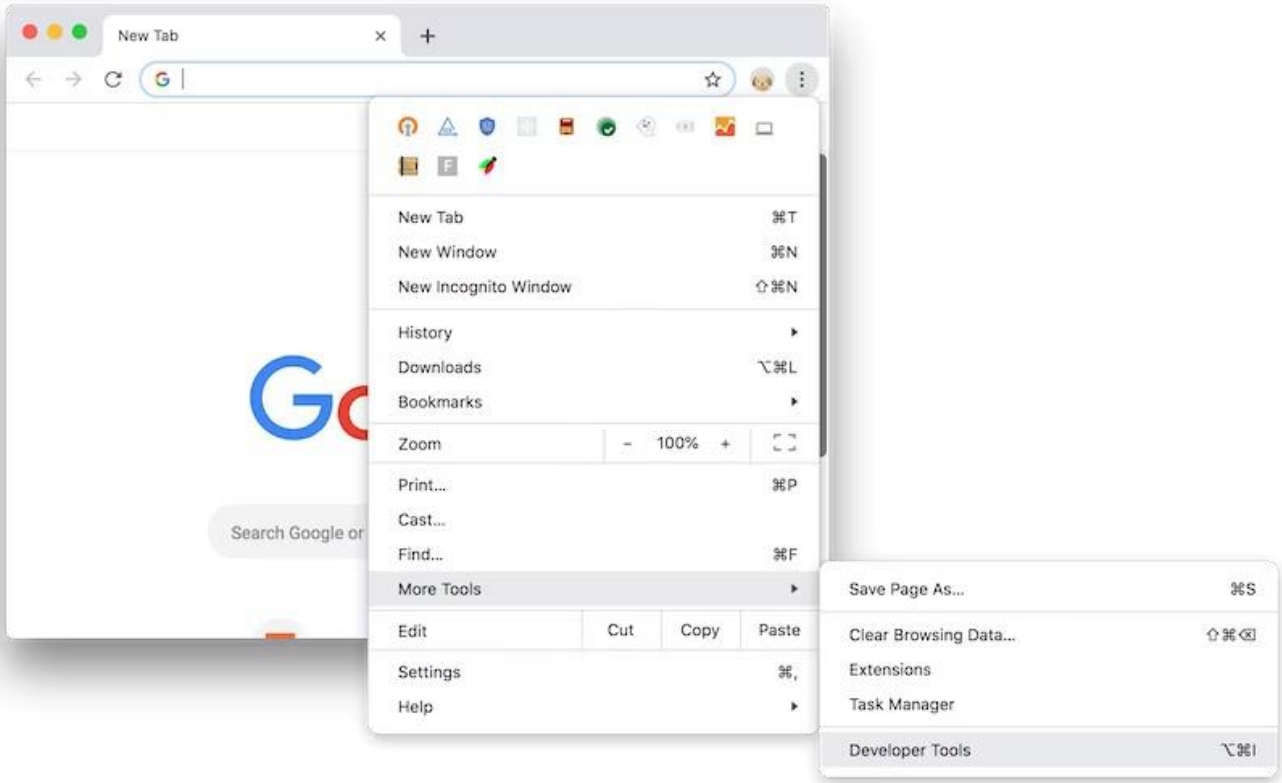
- **Mostar información desde JavaScript**
  - Escribiendo en la consola JavaScript del navegador

```
console.log('Texto');
```

  - Más adelante veremos cómo mostrar información en la página web

Sintaxis básica

- **Mostar información desde JavaScript**
  - Mostrar la consola de Google Chrome



Ejercicio 1

- Crea una página web que muestre en la consola"Hola Mundo!" al cargarse en un navegador web

HTML Interactivo

code URJC

- JavaScript se diseñó para dotar de **interactividad a las páginas HTML** cargadas en el navegador
- Se puede **modificar la página HTML** con JavaScript
  - **Cambio de CSS** de un elemento (ocultar, cambio de color…)
  - **Cambio del contenido**: Texto, imágenes, …
- Se puede **ejecutar código** JavaScript cuando el usuario interactúa con la página (**click, hover, etc…**)

Document Object Model (DOM)

code URJC

- Desde JavaScript se puede acceder al documento y al navegador
  - El **Document Object Model (DOM)** es una librería para manipular documentos HTML

https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

  - **El Browser Object Model (BOM)** permite manipular otros elementos del **navegador**

https://developer.mozilla.org/en-US/docs/Web/API/Window
https://developer.mozilla.org/en-US/docs/Web/API/Navigator
https://developer.mozilla.org/en-US/docs/Web/API/Location

Seleccionar una parte del documento

- Existe una variable **document** que apunta al documento DOM cargado en el navegador

https://developer.mozilla.org/en-US/docs/Web/API/Document

- Se pueden seleccionar partes del documento usando su id, class, name o tag

```
let image1 = document.getElementById('image1');
let citas = document.getElementsByClassName('cita');
let img2 = document.getElementByName('image2');
let links = document.getElementsByTagName('a');
```

https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById

Modificar el documento con HTML

- Para cambiar el contenido del documento cargado en el browser se selecciona un elemento del documento y se cambia su contenido HTML con la propiedad **`innerHTML`**

```
let element = document.getElementById('txt');

element.innerHTML = '<p>Nuevo texto</p>'
```

https://developer.mozilla.org/en-US/docs/Web/API/Element/innerHTML

Modificar el documento con HTML

code URJC

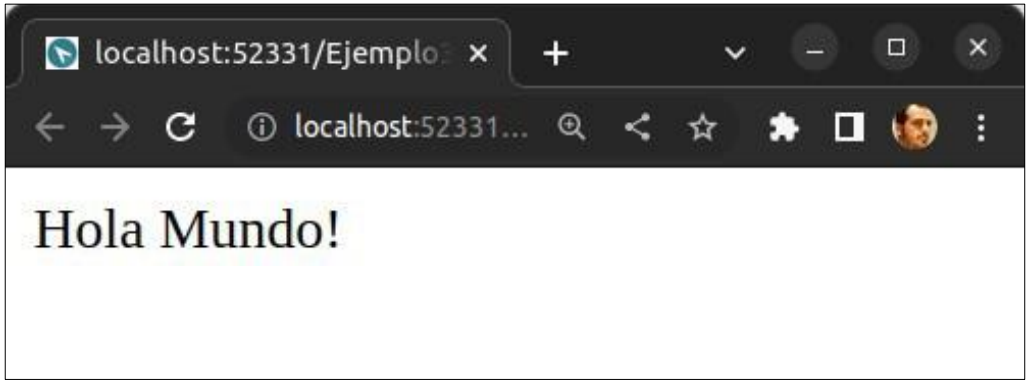
Ejemplo2

app.js

```
let content = document.getElementById('content');
content.innerHTML = '<p>Hola Mundo!</p>'
```

index.html

```
<html>
 <body>
   <div id='content'></div>
   <script src='app.js'></script>
 </body>
</html>
```



Gestión de Eventos

- Se pueden ejecutar **funciones** cuando se interactúa con elementos del documento
- Ejemplo: Mostrar una alerta al pulsar un botón

Ejemplo3

```
function alerta() {
    alert('El botón ha sido pulsado');
}
```

```
<html>
  <body>
    <script src='app.js'></script>
    <button onclick="alerta();">Botón</button>
  </body>
</html>
```

Gestión de Eventos

code URJC

Ejemplo4

app.js

```
function saluda() {
    let content = document.getElementById('content');
    content.innerHTML = '<p>Hola Mundo!</p>';
}
```

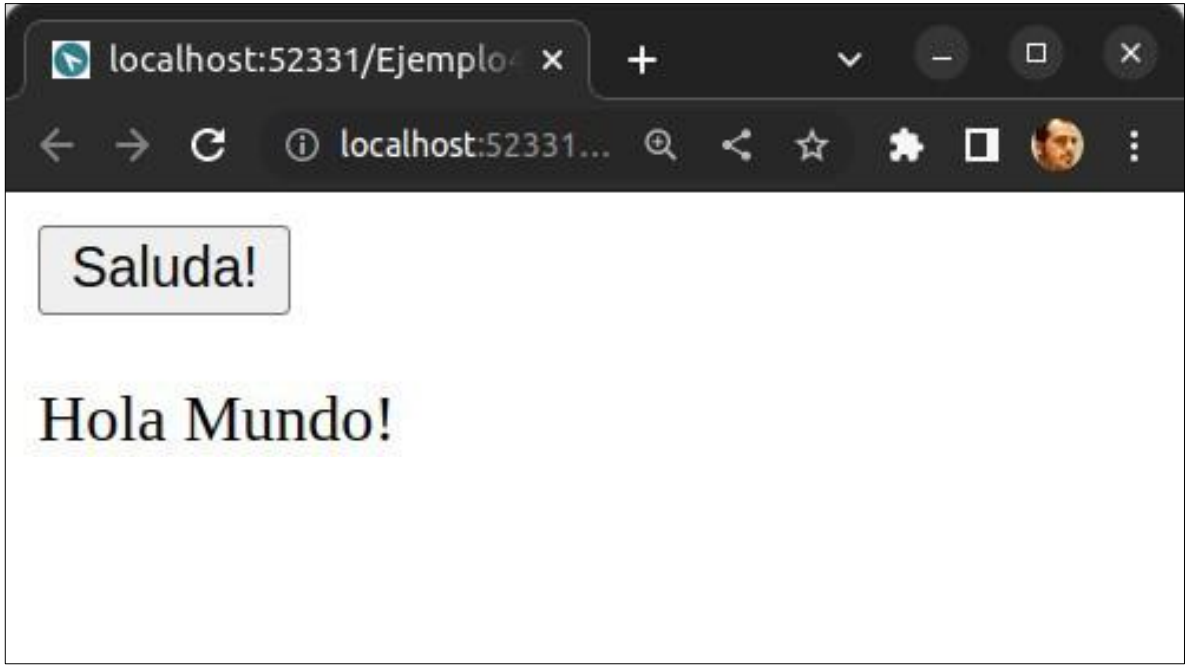
index.html

```
<html>
  <body>
    <button onclick="saluda()">Saluda!</button>
    <div id='content'></div>
    <script src='app.js'></script>
  </body>
</html>
```

Gestión de Eventos

code URJC

Ejemplo4



Gestión de Eventos

- Documentación sobre Eventos en JavaScript y DOM

  https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events

- Listado de eventos que se pueden producir en un navegador web

  https://developer.mozilla.org/en-US/docs/Web/Events

- Evento click de un elemento HTML

  https://developer.mozilla.org/en-US/docs/Web/API/Element/click_event

Ejercicio 2

- Implementa una web que muestre los títulos de unos libros
  - Cien años de soledad
  - El señor de los anillos
  - 1984
  - ~~Un~~ mundo feliz
- Los títulos de los libros estarán almacenados en un array de JavaScript

Gestión de Eventos

- En vez de configurar el evento en el HTML, también se puede configurar en el código JavaScript con el método **addEventListener**

Ejemplo5

index.html

```
<html>
  <body>
    <button>Saluda!</button>
    <div id='content'></div>
    <script src='app.js'></script>
  </body>
</html>
```

Gestión de Eventos

- En vez de configurar el evento en el HTML, también se puede configurar en el código JavaScript con el método **addEventListener**

Ejemplo5

app.js

```
function saluda() {
    let content = document.getElementById('content');
    content.innerHTML = '<p>Hola Mundo!</p>';
}

let buttons = document.getElementsByTagName('button');

buttons[0].addEventListener('click', saluda);
```

Gestión de Eventos

- La función que se ejecuta cuando se produce un evento (*handler*) puede recibir como parámetro información del evento

Ejemplo6

app.js

```
let buttons = document.getElementsByTagName('button');

function greet(event) { console.log('Evento:', event);
console.log('Elemento origen: ', event.target);
}

buttons[0].addEventListener('click', greet);
```

https://developer.mozilla.org/en-US/docs/Web/Events/Event_handlers

https://developer.mozilla.org/en-US/docs/Web/API/Event

Carga del documento DOM

- Si el script se configura al **final del html**, se ejecuta cuando el documento ya está **cargado** en memoria

```
<html>
  <body>
    ...
    <script src='app.js'></script>
  </body>
</html>
```

- Si el script se configura en el **<head>**, se ejecuta **antes** de que el documento haya sido cargado

```
<html>
  <head>
    <script src='app.js'></script>
  </head>
  <body>
                ...
              </body>
              </html>
```

Carga del documento DOM

- Si ponemos el script en el **<head>** hay que esperar a que se haya cargado la página para insertar el contenido desde JavaScript

Ejemplo7

index.html

```
<html>
  <head>
    <script src='app.js'></script>
  </head>
  <body>
    <div id='content'></div>
  </body>
</html>
```

Carga del documento DOM

- El documento tiene el evento **DOMContentLoaded** que se ejecuta cuando el documento se ha cargado
- Si no se espera a la carga del DOM, al buscar el elemento "content" no se encontrará

Ejemplo7

app.js

```
document.addEventListener("DOMContentLoaded", function(){

    let content = document.getElementById('content');
    content.innerHTML = '<p>Hola Mundo!</p>';

});
```

Modificar los estilos

- Se puede cambiar cualquier estilo de un elemento con su propiedad **`style`**

```
let element = document.getElementById('img1');

element.style.borderWidth = width + 'px';
```

https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/style

- Ocultar y mostrar un elemento con estilos

```
let element = document.getElementById('text');

element.style.display = 'none';
...
element.style.display = 'block';
```

Ejercicio 3

- Crea una página con un botón que oculte el texto “Hola Mundo” cuando se pulse
- Si se pulsa de nuevo, se volverá a mostrar el texto “Hola Mundo”

Modificar los estilos

code URJC

- Una forma común de cambiar los estilos es añadiendo o eliminando clases CSS a un elemento HTML

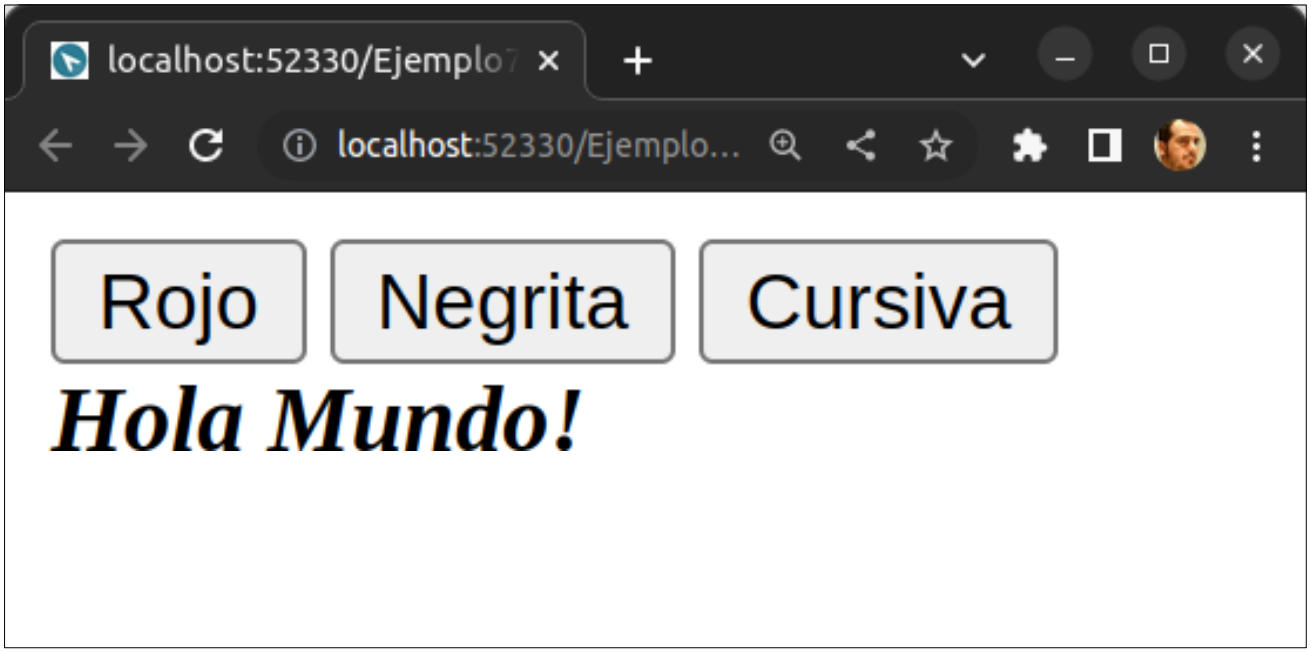
```
let content = document.getElementyById('content');
...
content.classList.remove('rojo');
...
content.classList.add('negrita');
...
content.classList.toggle('cursiva');
```

https://developer.mozilla.org/en-US/docs/Web/API/Element/classList

Modificar los estilos

- Ejemplo con botones que añaden o eliminan una clase (*toggle*) de un elemento

Ejemplo8



Modificar los estilos

- Ejemplo con botones que añaden o eliminan una clase (*toggle*) de un elemento

Ejemplo8

```
function rojo(){
  let content = documento.getElementyById('content');
  content.classList.toggle('rojo');
}
```

```
<body>
    <button onclick='rojo()'>Rojo</button>
    <div id='content'>Hola Mundo!</div>
    ...
</body>
```

```
.rojo {
    color: #F00;
}
```

Ejercicio 4

- Amplía el Ejercicio 2 para que al lado de cada libro aparezca un botón [Más info]
- Al pulsar el botón aparecerá debajo del título el autor del libro y su año de publicación
- Un elemento se puede ocultar con el estilo "display:none" y mostrarse de nuevo con "display:block"

Ejercicio 4

- Cuando el código HTML es complejo se pueden usar *template literals*
  - Se usan comilla ` (*backtick*) para multilínea
  - Se usa ${...} para las variables

```
let html =
  `<div>
    <p>${libro.titulo}</p>
    <p> ${libro.autor} (${libro.año})</p>
</div>`;
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals

Ejercicio 4

- Información libros
  - Cien años de soledad, de Gabriel García Márquez (1967)

  El señor de los anillos, de J. R. R. Tolkien (1954)

  - 1984, de George Orwell (1949)

  Un mundo feliz, de Aldous Huxley (1932)

Generación de HTML

- Existen diversas formas de generar código HTML desde código JavaScript
  - **Cadena de caracteres (*String*)**
  - Creando los nodos del árbol DOM

Generación de HTML

- Cadena de caracteres (*String*)

```
let text = "Hola Mundo!";

let content = document.getElementById("content");

content.innerHTML = "<p>" + text + "</p>";
```

Generación de HTML

- Existen diversas formas de generar código HTML desde código JavaScript
  - Cadena de caracteres (*String*)
  - **Creando los nodos del árbol DOM**

Generación de HTML

- Creando los nodos del árbol DOM

```
let text = "Hola Mundo!";

let content = document.getElementById("content");

content.innerHTML = "<p>" + text + "</p>";
```

```
let text = "Hola Mundo!";

let content = document.getElementById("content");

let p = document.createElement("p"); p.textContent =
text; content.appendChild(p);
```

Generación de HTML

- Creando los nodos del árbol DOM

Ejemplo9

```
let text = "Hola Mundo!";

let content = document.getElementById("content");

let p = document.createElement("p"); p.textContent =
text; content.appendChild(p);
```

https://developer.mozilla.org/en-US/docs/Web/API/Document/createElement

https://developer.mozilla.org/en-US/docs/Web/API/Node/textContent

https://developer.mozilla.org/en-US/docs/Web/API/Node/appendChild

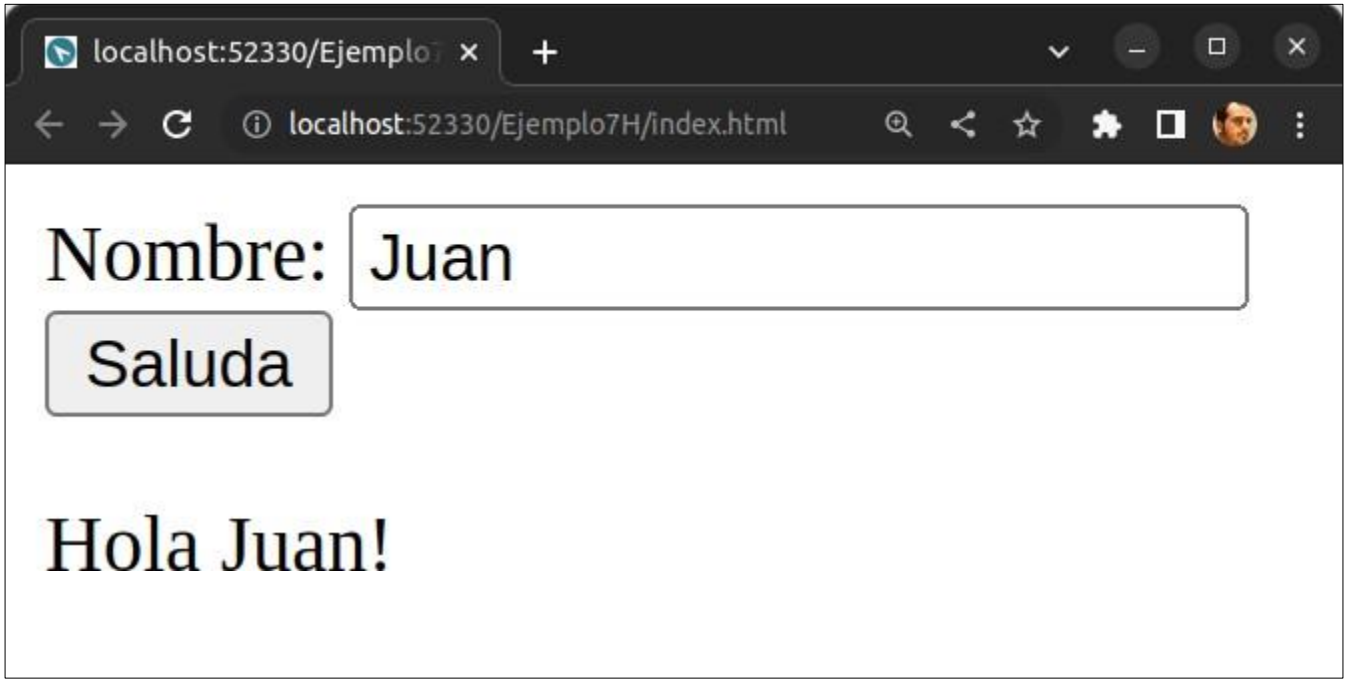
Ejercicio 5

- Actualiza el ejercicio 3 para no generar el HTML como texto y usar la API de nodos del DOM

Formularios

- Se puede obtener información de campos de formularios

Ejemplo10



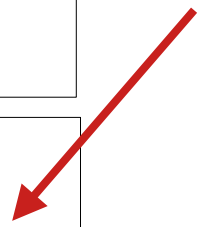
Formularios

- Se puede obtener información de campos de formularios

Ejemplo13

```
<div>
    Nombre: <input id='nombre' type='text'></input>
    <button onclick='saluda()'>Saluda</button>
</div>
<div id='content'></div>
```

```
function saluda() {

    let nombre = document.getElementById('nombre').value;

    let content = document.getElementById('content');
    content.innerHTML += '<p>Hola '+nombre+'!</p>';
}
```



https://developer.mozilla.org/en-US/docs/Web/API/HTMLInputElement#value

Ejercicio 6

- Amplía el ejercicio 5 para que se puedan dar de alta nuevos libros con un formulario

Ejercicio 7

- Crea una página que muestre el título de varios libros
- Al lado de cada título aparecerá el botón "Más info"
- Al pulsar el botón se borrará la lista de libros y aparecerá la información del libro junto con un botón "Volver a la lista"

Ejercicio 7


localhost:52330/7.1/Ejerc
localhost:52330/7.1/Ejercicio...
Librería FW
Libros
Cien años de soledad
Más info
El señor de los anillos
Más info
1984
Más info
Un mundo feliz
Más info
localhost:52330/7.1/Ejerc
localhost:52330/7.1/Ejercicio...
Librería FW
Libro: Cien años de soledad
Autor: Gabriel García Márquez
Año: 1967
Volver a la lista

Modificar el documento con HTML

Ejecutar código de un módulo

- Los navegadores modernos solo permiten ES Modules
- El código del módulo se ejecuta cuando se importa

hello.js

```
console.log('Hello world');
```

app.js

```
import './hello.js';
```

index.html

```
<html>
  <head>
    <script type="module" src="app.js">
    </script>
  </head>
  <body>
  </body>
</html>
```

08-2023

code URJC

Fundamentos de la Web

Bloque III: Tecnologías de interactividad en el cliente web

Tema 4.2: AJAX

Universidad Rey Juan Carlos

Micael Gallego
Correo: micael.gallego@urjc.es
Twitter: @micael_gallego

Iván Chicano
ivan.chicano@urjc.es
Michel Maes
michel.maes@urjc.es

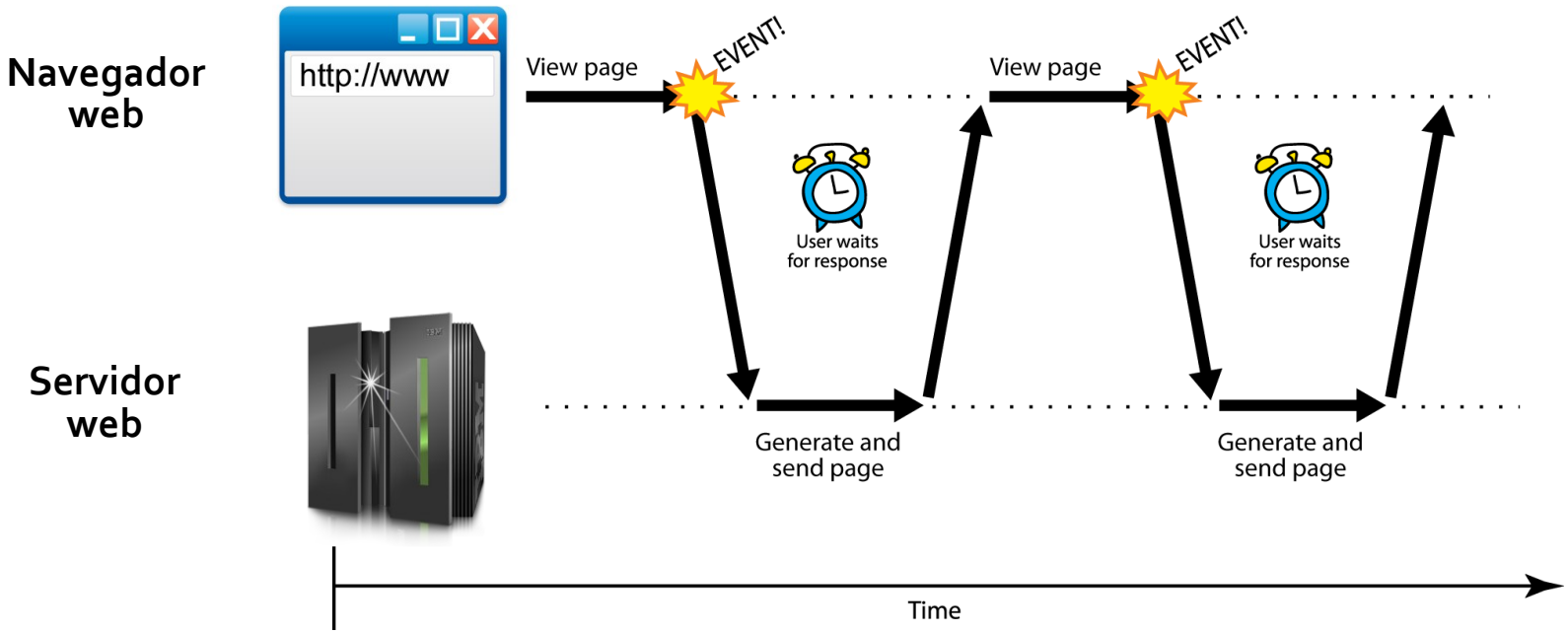
08-2023

code
URJC

Micael Gallego, Iván Chicano, Michel Maes

Introducción

- En una página web básica, cada vez que el usuario **pulsa un enlace**, en navegador **recarga completamente la página**

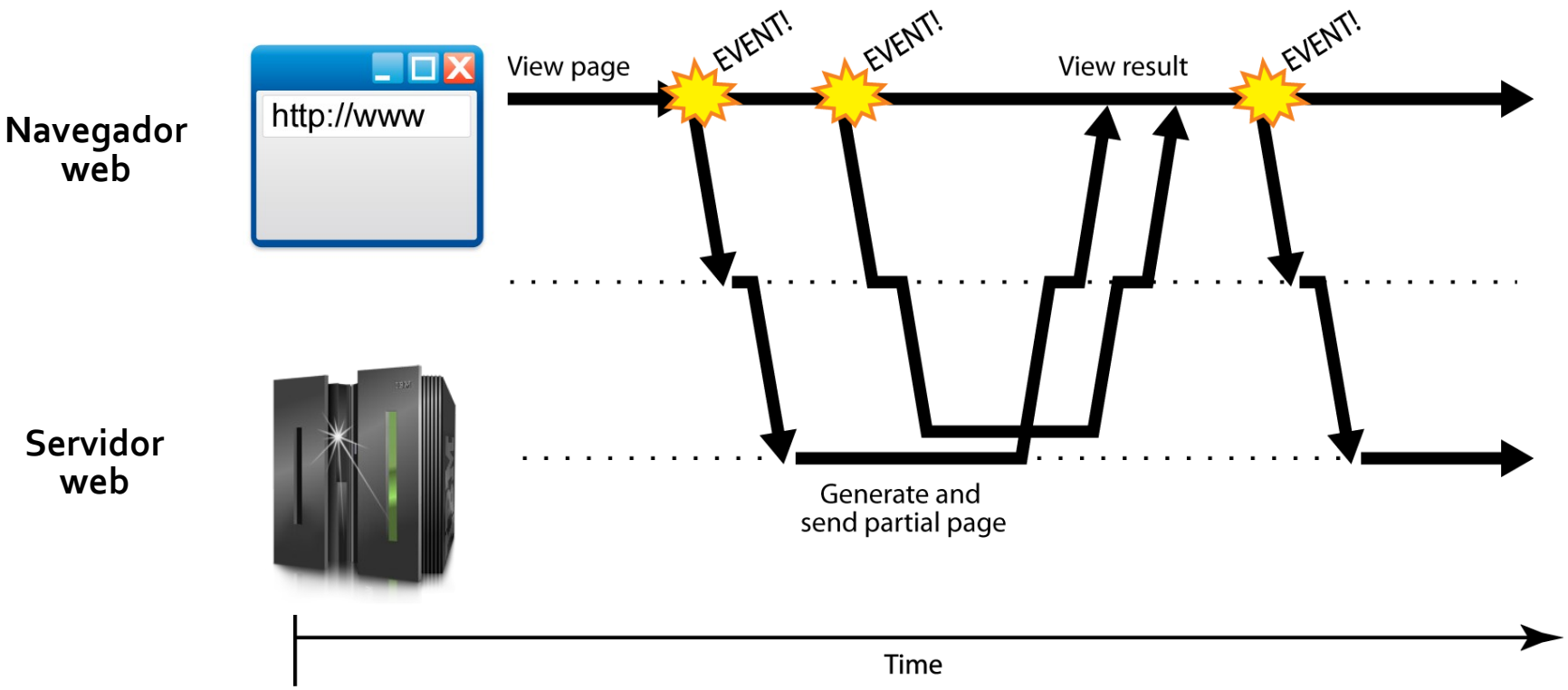


Introducción

- A veces sólo cambia **parte de la página,** pero se tiene que recargar completa (más **transferencia de datos**, mayor **tiempo de espera**…)
- **AJAX (*Asynchronous JavaScript And XML*)** es una técnica que permite actualizar únicamente parte de la página
- Desde código JavaScript que **solicita al servidor la parte nueva** y **actualiza** la página con el contenido obtenido del servidor

Introducción

- AJAX (*Asynchronous JavaScript And XML*)



Introducción

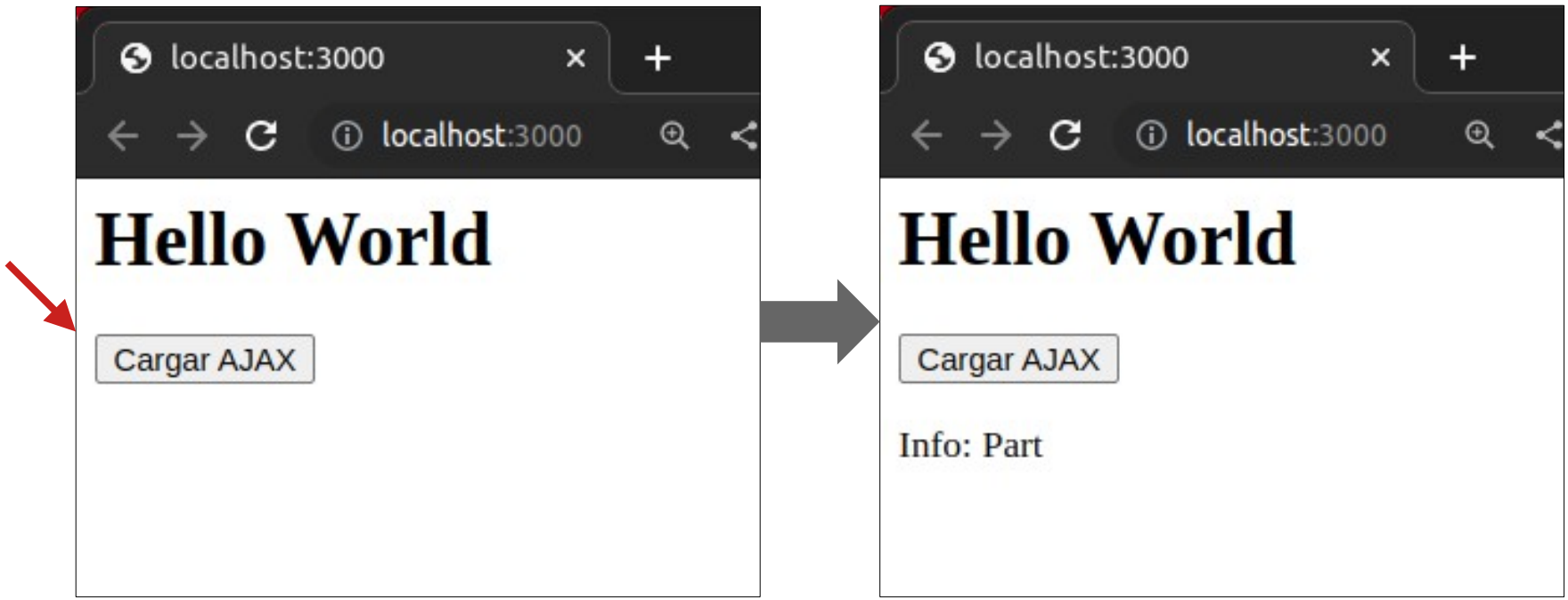
- Es la técnica que se usa para **cargar más resultados** en las búsquedas sin recargar la página (botón "**Cargar más**" o de forma **automática**)

Google images

Implementación de AJAX

code URJC

Ejemplo1



Implementación en cliente

code URJC

Ejemplo1

- Petición AJAX desde el navegador web

```
<html>

<head>
  <script src="app.js"></script>
</head>

<body>

  <h1>Hello {{name}}</h1>

  <button onclick="loadAJAX()">Cargar AJAX</button>

  <div id="content"></div>

</body>

</html>
```

Botón que ejecuta la función loadAJAX()

Implementación en cliente

code URJC

Ejemplo1

- Petición AJAX desde el navegador web

```
async function loadAJAX(){

    const response = await fetch('/pagePart');
    const pagePart = await response.text();

    const content = document.getElementById("content");
    content.innerHTML = pagePart;

}
```

Implementación en cliente

code URJC

Ejemplo1

- Petición AJAX desde el navegador web

Hace la petición al servidor de la ruta **/pagePart** en segundo plano y devuelve una respuesta (response)

```
async function loadAJAX(){

    const response = await fetch('/pagePart');
    const pagePart = await response.text();

    const content = document.getElementById("content");
    content.innerHTML = pagePart;

}
```

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Implementación en cliente

code URJC

Ejemplo1

- Petición AJAX desde el navegador web

```
async function loadAJAX(){

    const response = await fetch('/pagePart');
    const pagePart = await response.text();

    const content = document.getElementById("content");
    content.innerHTML = pagePart;

}
```

Procesa la respuesta y obtiene el **texto** que contiene

https://developer.mozilla.org/en-US/docs/Web/API/Response/text

Implementación en cliente

code URJC

Ejemplo1

- Petición AJAX desde el navegador web

```
async function loadAJAX(){

    const response = await fetch('/pagePart');
    const pagePart = await response.text();

    const content = document.getElementById("content");
    content.innerHTML = pagePart;

}
```

Se inserta el contenido cargado en la página

Implementación en cliente

code URJC

Ejemplo1

- Petición AJAX desde el navegador web

```
async function loadAJAX(){

    const response = await fetch('/pagePart');
    const pagePart = await response.text();

    const content = document.getElementById("content");
    content.innerHTML = pagePart;

}
```

Más adelante veremos qué significa **async** y **await**

Implementación en servidor

code URJC

Ejemplo1

- Generación de contenido AJAX en el servidor

router.js

```
import express from 'express';

const router = express.Router();

router.get('/', (req, res) => {

  res.render('index', {
    name: "World"
  });
});

router.get('/pagePart', (req, res) => {

  res.render('pagePart', {
    info: "Part"
  });
});

export default router;
```

pagePart.mustache

```
<p>Info: {{info}}</p>
```

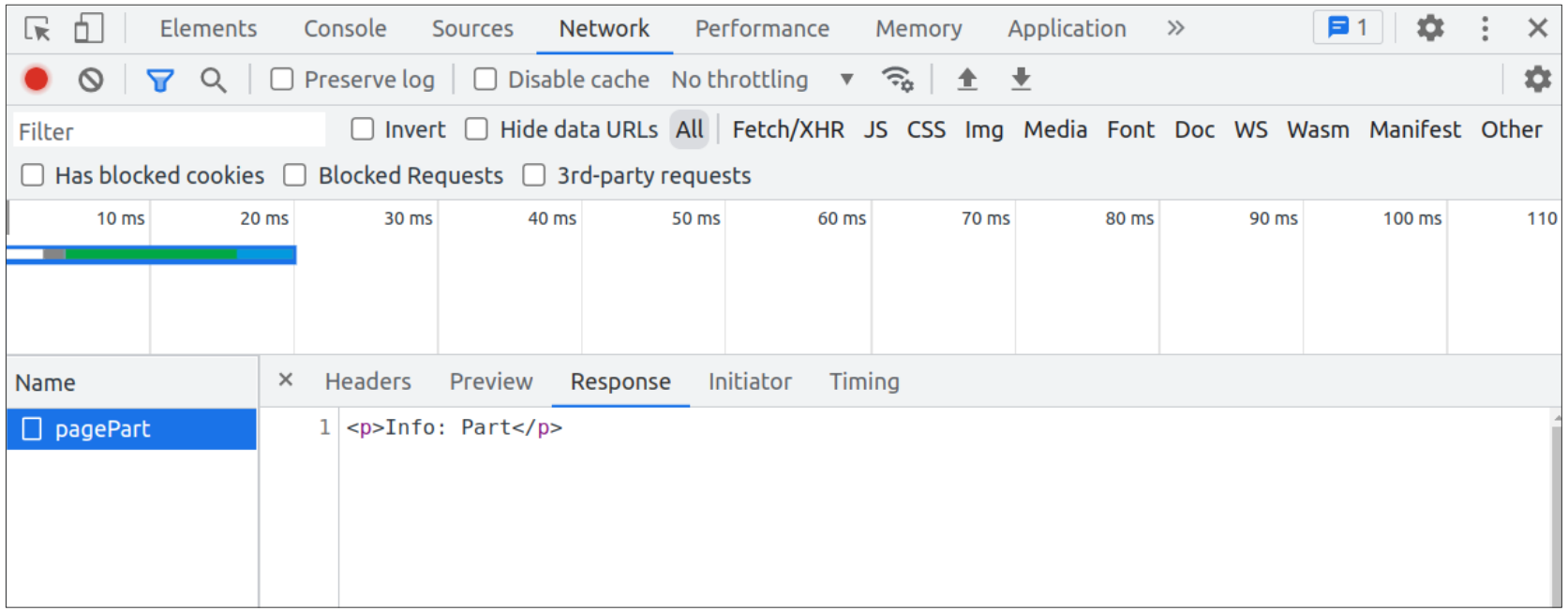
Se genera el contenido igual que las páginas completas

Depuración

code URJC

Ejemplo1

- La pestaña “Network/Red” del navegador permite **ver las peticiones** realizadas por AJAX



Ejercicio 1

code URJC

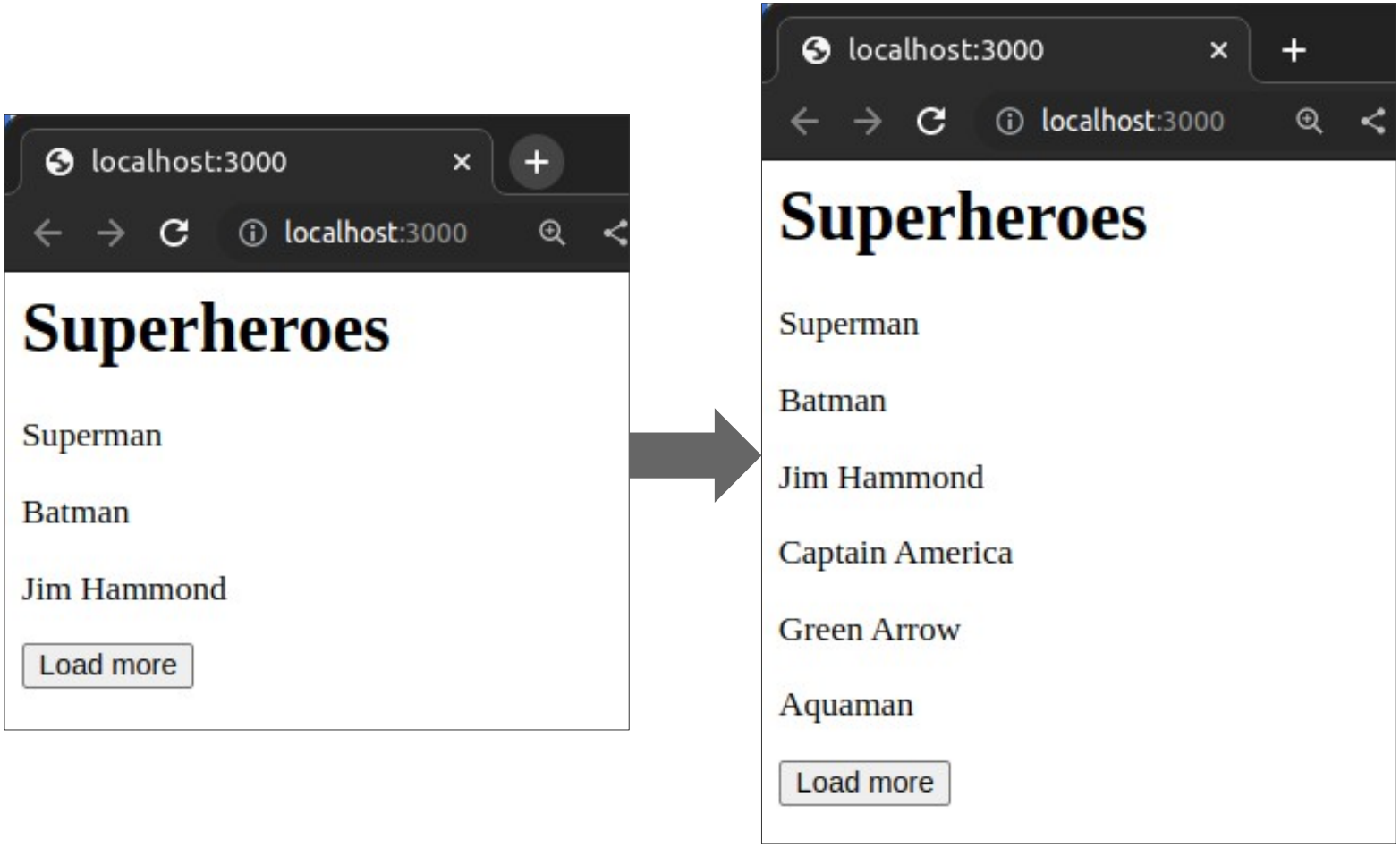
Ejercicio1

- Se tiene una web que muestra los superhéroes guardados en un mapa
- Se quiere implementar un botón de "Cargar más" con AJAX
- Los superhéroes se mostrarán de 3 en 3

Ejercicio 1

code URJC

Ejercicio1



Ejercicio 1

code URJC

Ejercicio1

- Al cargar la página se mostrarán los superhéroes 0,1 y 2.
- Cuando se pulse el botón la primera vez se cargarán los superhéroes 3, 4 y 5.
- Cuando se pulse el botón la segunda vez se cargarán los superhéroes 6, 7 y 8.
- La URL que se usa para cargar más resultados deberá indicar qué resultados cargar

Ejercicio 1

code URJC

- Se proporciona una web en la que se muestran inicialmente todos los libros

```
import express from 'express';
import { getSuperheroes } from './superheroes.js';

const router = express.Router();

router.get('/', (req, res) => {

  const superheroes = getSuperheroes();

  res.render('index', {
    superheroes: superheroes
  });
});

export default router;
```

Ejercicio 1

code URJC

Ejercicio1

- Se proporciona una web en la que se muestran inicialmente todos los libros

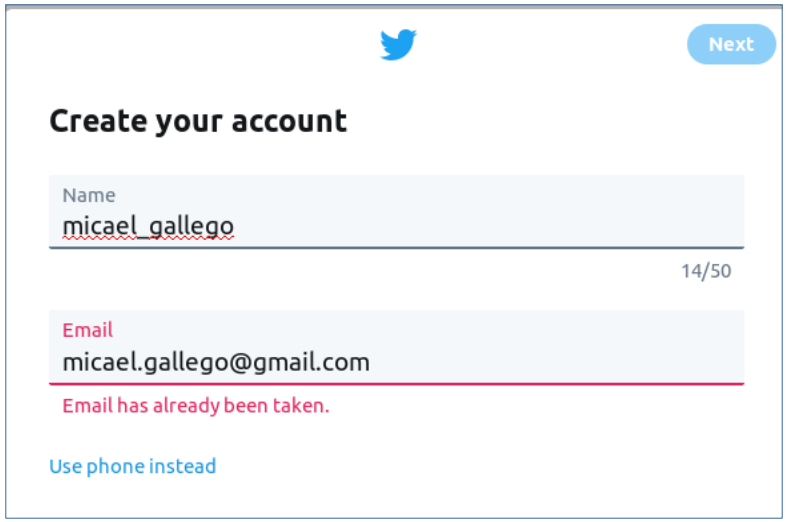
```
const superheroes = new Map();
let id = 0;

export function addSuperhero(superhero) {
  superheroes.set(id, superhero);
  superhero.id = id;
  id++;
}

export function getSuperhero(id) {
  return superheroes.get(id);
}
```

```
export function getSuperheroes(from, to) {
  let values = [...superheroes.values()];
  if (from !== undefined) {
    return values.slice(from, to);
  } else {
    return values;
  }
}

export function loadSampleData() {

  addSuperhero({ name: 'Superman' });
  addSuperhero({ name: 'Batman' });
  ...
}

loadSampleData();
```

Información estructurada con AJAX

code URJC

- AJAX también puede ser para que el código JavaScript **consulte información al servidor**
- Con esa información puede **manipular la página** como quiera (por ejemplo mostrando una alerta)



Información estructurada con AJAX

code URJC

- Cuando el código **JavaScript** hace peticiones, el servidor puede devolver:

Fragmentos de HTML	**Información estructurada**
Se incrusta directamente en la página	Se interpreta por JavaScript para modificar la página
Ej: Cargar más	Ej: Error de validación

Información estructurada con AJAX

code URJC

- Cuando se solicita información estructurada al servidor la suele generar en **formato JSON**
- También se puede devolver en **formato XML**

```
{
 validation:[
  {id:“name”,status:“ok”},
  {id:“email”,status:“error”,message:“Invalid format”}
 ]
}
```

Información estructurada con AJAX

code URJC

Ejemplo2

- Cliente

```
async function checkInfo(){

  const info = 'someInfo';

  const response = await fetch(`/checkInfo?info=${info}`);

  const infoCheck = await response.json();
  const content = document.getElementById("content");

  content.innerHTML = `
    <p>Valid: ${infoCheck.valid}</p>
    <p>Message: ${infoCheck.message}</p>`;

}
```

Información estructurada con AJAX

code URJC

Ejemplo2

- Cliente

```
async function checkInfo(){

  const info = 'someInfo';

  const response = await fetch(`/checkInfo?info=${info}`);

  const infoCheck = await response.json();
  const content = document.getElementById("content");

  content.innerHTML = `
    <p>Valid: ${infoCheck.valid}</p>
    <p>Message: ${infoCheck.message}</p>`;

}
```

La información se carga en un objeto con `response.json()`

Información estructurada con AJAX

code URJC

Ejemplo2

- Servidor

```
import express from 'express';

const router = express.Router();

...

router.get('/checkInfo', (req, res) => {

  let info = req.query.info;

  let response = {
    valid: false,
    message: `Info '${info}' not valid`
  }

  res.json(response);
});

export default router;
```

Información estructurada con AJAX

code URJC

Ejemplo2

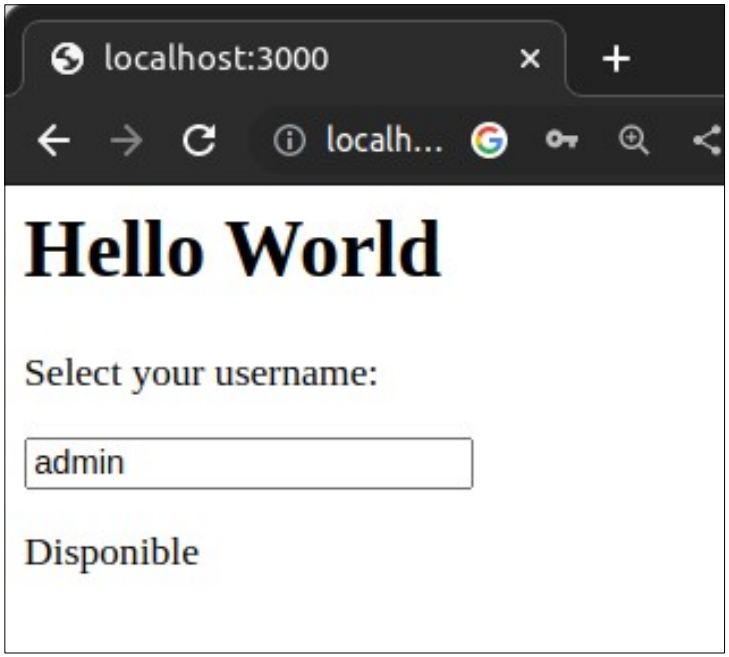
- Servidor

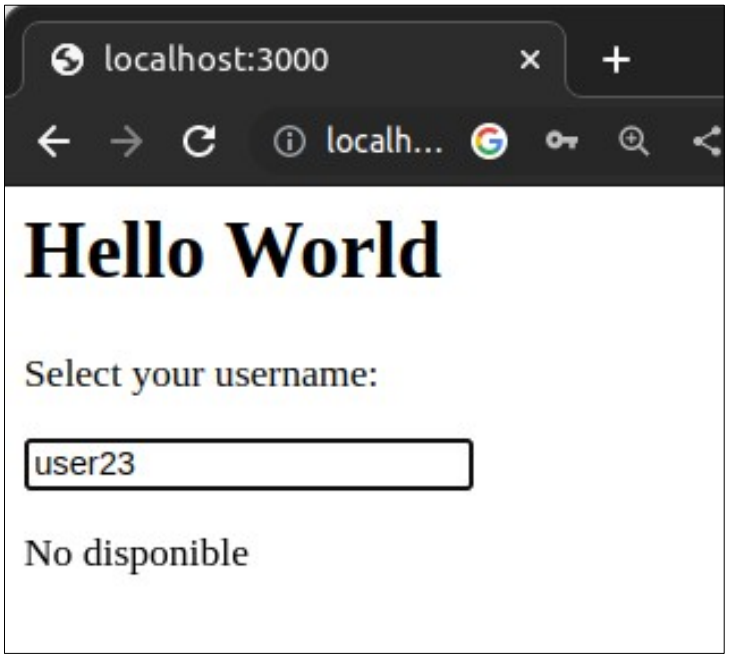
```
import express from 'express';

const router = express.Router();

...

router.get('/checkInfo', (req, res) => {

  let info = req.query.info;

  let response = {
    valid: false,
    message: `Info '${info}' not valid`
  }

  res.json(response);
});

export default router;
```

La información se envía con `res.json()`

Ejercicio 2

- Web con input de texto que consulta disponibilidad al servidor según va escribiendo el usuario





Ejercicio 2

- El evento “input” permite ejecutar una función cada vez que el texto cambia

```
<input id='username' oninput='checkUsernameAvailability()' type='text'>
```

- Nota: En las web reales sólo se hace la petición cuando el usuario deja de escribir durante un tiempo para no saturar al servidor

Async / Await

code URJC

- En programación, una función puede bloquearse o no:
  - Si sólo ejecuta **cálculos** en el procesador, **no se bloquea.**
  - Si solicita datos por red o al disco (**Entrada/Salida, IO**) y **espera** a que lleguen o simplemente se espera durante un tiempo sin hacer nada, **se bloquea.**

Async / Await

code URJC

- En **Java** (y en la mayoría de los lenguajes de programación), cuando se hace una **llamada** a una función **no se sabe si** esta función es **bloqueante o no**.

- En **JavaScript**, a las funciones **bloquantes** (por IO o por esperas), se las tiene que ejecutar de una forma **especial**.

Async / Await

- En JavaScript las funciones bloqueantes habitualmente se ejecutan con **`await`**
- Si una función llama a otra con await, la función debe declararse con **`async`** (y se convierte en bloqueante)

```
async function loadAJAX(){

    const response = await fetch('/pagePart');
    const pagePart = await response.text();

    ...
}
```

Async / Await

code URJC

- En versiones previas de JavaScript **no se podía usar await**
- Cuando las funciones bloquantes **se ejecutan sin await** devuelven un objeto **Promesa (*Promise*)**
- Este objeto tiene un método **then(…)** para definir qué código ejecutará cuando **termine el bloqueo** y esté **disponible el valor** de red o disco

Async / Await

code URJC

Con async / await

```
async function loadAJAX(){

    const response = await fetch('/pagePart');
    const pagePart = await response.text();
    ...
}
```

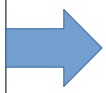
Sin async / await

```
function loadAJAX(){

    fetch('/pagePart')
       .then(response => response.text())
       .then(pagePart => { ... });
}
```

Async / Await

- Si hay código después de haber configurado la promesa se ejecutará antes de que haya llegado el valor de la petición de red

```
function loadAJAX(){

     fetch('/pagePart')
       .then(response => response.text())
       .then(pagePart => console.log('Valor recibido');

     console.log('Petición enviada');
}
```



```
Petición enviada

Valor recibido
```

Async / Await

code URJC

- El código **sin await** es un poco más **complejo** de entender y está más **limitado**.
- Pero es importante que lo conozcamos por varios motivos:
  - Se olvida poner el await
  - Documentación con promesas
  - Ejecución de funciones bloqueantes en paralelo

Async / Await

- **Se olvida poner await**
  - Tenemos que saber que en vez del valor esperado tenemos una promesa

```
function loadAJAX(){

    const response = fetch('/pagePart');
    const pagePart = response.text();
    ...
}
```

ERROR: **response** es una promesa y no tiene el método **text()**

Async / Await

code URJC

- **Documentación con promesas**
  - Como en versiones anteriores de JavaScript no se podía usar async/wait hay mucha **documentación con ejemplos con promesas**
  - Pero se puede usar async/await

```
fetch('http://example.com/movies.json')
  .then((response) => response.json())
  .then((data) => console.log(data));
```

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Async / Await

- **Ejecución de funciones bloqueantes en paralelo**
  - Se pueden procesar los resultados según están disponibles

```
async function loadAJAX1() { … }
async function loadAJAX2() { … }

loadAJAX1().then(v => console.log(v));
LoadAJAX2().then(v => console.log(v));
```

Async / Await

- **Ejecución de funciones bloqueantes en paralelo**
  - Se pueden procesar los resultados cuando han llegado todos

```
async function loadAJAX1() { … }
async function loadAJAX2() { … }

let [v1, v2] = Promise.all([loadAJAX1(), loadAJAX2()]);

console.log(v1);
console.log(v2);
```

Conclusiones

- Código JavaScript en el **navegador** puede ejecutar código JavaScript en el **servidor** (haciendo una **petición http**)
- El servidor puede **devolver HTML** (que se incrusta en la página) o **JSON** (que se procesa)
- Las funciones bloqueantes se pueden ejecutar con **await**.
- Si se ejecutan **sin await** se obtiene una **promesa**

08-2023

code URJC

Fundamentos de la Web

Bloque V: Web semántica

Tema 5: Web semántica

Universidad Rey Juan Carlos

Nicolás Rodríguez
nicolas.rodriguez@urjc.es

Iván Chicano
ivan.chicano@urjc.es
Michel Maes
michel.maes@urjc.es

08-2023

code
URJC

Nicolás Rodríguez, Iván Chicano, Michel Maes

Definición

code URJC

- Framework común que permite compartir y reusar **datos** más allá de los límites de las aplicaciones, las empresas y las comunidades.
- También conocido como Web 3.0.
- Web de datos (Web of data)

Definición

- La web semántica añade información semántica a los contenidos de la web y confiere a las máquinas la capacidad de distinguir entre significado.

Área de conocimiento

- Inteligencia Artificial (IA): sistemas que tratan de emular la forma de razonar de los humanos: aprendizaje, toma de decisiones, deducciones.
- Ingeniería del conocimiento: Disciplina que se encarga de extraer y codificar el conocimiento de expertos humanos en un determinado campo, y de utilizarlo para automatizar algunas sus tareas.

Conceptos

- Usar ontologías para especificar el significado de las anotaciones.
- En filosofía, ontología es una disciplina que estudia la naturaleza y organización de la realidad.
- En informática, una ontología proporciona un vocabulario de términos utilizados para describir y representar un área de conocimiento.

Conceptos

- Se especifican:
  - Definiciones de conceptos básicos del dominio.
  - Relaciones entre ellos (y con elementos de otros dominios).
  - Propiedades o atributos que pueden tener estos conceptos.
- Permite formar nuevos términos combinando los existentes.

Estructura

- Vocabularios.
- Taxonomías simples (clasificaciones jerárquicas de términos).
- Sistemas relacionales (definición de nuevas relaciones).
- Teorías axiomáticas y reglas de inferencia (restricciones, enunciados lógicos que permiten demostrar nuevas relaciones)

RDF

- RDF (Resource Description Framework) es la base de la mayoría de las tecnologías de Web semántica.
- Es un lenguaje estándar del Web Consortium.
- En la Web convencional la unidad fundamental son documentos que se enlazan mediante hiperenlaces.

Elementos básicos de RDF

- En RDF la unidad fundamental son recursos, que representan entidades lógicas, y se enlazan entre sí con propiedades, que representan relaciones lógicas.
- Ejemplo: “Piolín es de color amarillo”
- Tanto recursos como propiedades se identifican con URIs, facilitando así la gestión integrada de datos distribuidos en la red.

Elementos básicos de RDF

- Recursos (identificados con URIs)
- Un URI identifica un recurso, pero no necesariamente apunta a su localización.
- Corresponden a nodos en un grafo
- Ej: http://www.w3.org, http://midireccion.es/#piolin
- Propiedades (identificadas con URIs)
- Representan relaciones entre dos recursos, o entre un recurso y un valor literal.

Elementos básicos de RDF

- Corresponden a etiquetas en los arcos de un grafo.
- El nodo destino se denomina valor de la propiedad.
- Literales
- Valores concretos
- Ej. “José Pérez”, “amarillo”

Elementos básicos de RDF

- Una descripción RDF se puede ver como un grafo acíclico dirigido donde:
- Los nodos son recursos o literales (o nodos en blanco)
- Los arcos son afirmaciones o enunciados que nos permiten asociar una propiedad a un recurso.
- Las propiedades son etiquetas de los arcos.

Elementos básicos de RDF

- Una descripción RDF también se puede ver como un conjunto de tripletas (sujeto, predicado, objeto)
  (<http://midireccion/#Piolin>, <http://www.sentidos.net/#tieneColor>, "amarillo")
- Cada tripleta es una afirmación que asocia una propiedad a un recurso.

Elementos básicos de RDF

- El sujeto puede ser un recurso o un nodo en blanco
- El predicado es una propiedad
- El objeto puede ser un literal, un recurso o un nodo en blanco
- Los literales sólo pueden ser objetos (no tiene sentido asociar una propiedad a un literal)

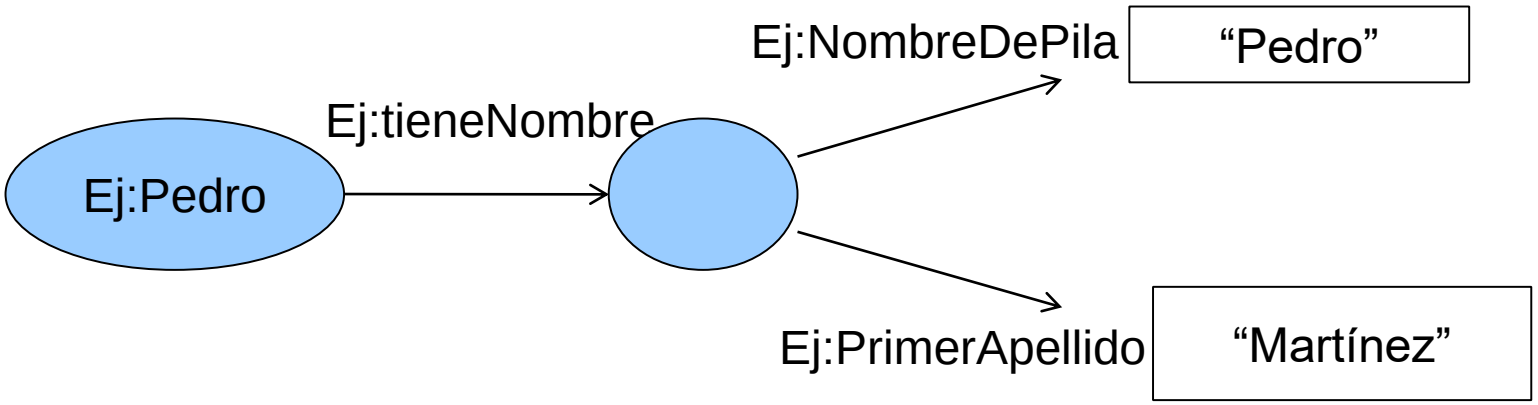
Elementos básicos de RDF

- Nodos en blanco: No tienen asociado un URI
- Pueden ser:
- Recursos sin nombre
- Nodos auxiliares para formar construcciones complejas

(<Ej:Pedro>, <Ej:tieneNombre>, _:x)

(_:x, Ej:NombreDePila, "Pedro")

(_:x, Ej:PrimerApellido, "Martínez")



Sintaxis de RDF

Notación estándar: RDF/XML

- Permite usar las herramientas XML existentes
- Análisis y comprobación sintáctica
- Transformación (con XSLT)
- Diferentes representaciones RDF
- Visualización (XHTML)
- Engorrosa
- No es trivial reconstruir el grafo RDF

https://www.w3.org/TR/rdf11-primer/#section-rdfxml

Sintaxis de RDF

Notación Turtle

- Refinamiento de N3
- Es la base de SPARQL (Lenguaje de consulta sobre RDF)
- Permite agrupar las tripletas según el sujeto
- Mecanismo para utilizar y prefijar los espacios de nombres
- Sintaxis: @prefix espacioNombres:<URI>.
  (terminado en punto)

Sintaxis de RDF

Notación Turtle

- Ejemplo:
  @prefix rdf:<http://ww.w3.org/199/02/22-rdf-sintax-ns#>.
- rdf: es el espacio de nombres por defecto (se puede omitir)

Sintaxis de RDF

Notación Turtle - Identificadores

- URI: <URI>
- Nombres: abbr:nombre, rdf:type dc:title
- Literales: “string”(@idioma)(^^tipo) “Pedro” “Hello”@en-GB “1.4”^^xs:decimal
- Abreviatura para los literales con tipo integer: 24 es la abreviatura de “24”^^xsd:integer
- Decimal: 2.4 5.67
- Nodos en blanco _:nombre

Sintaxis de RDF

Notación Turtle - Tripletas

- Simples: sujeto predicado objeto.
  (separados por espacios y terminado en punto)
  Ejemplo:
  Ej:Juan Ej:nombre “Juan”.
- Agrupación de tripletas con el mismo sujeto:
  sujeto predicado objeto ; predicado objeto.
  Ej:JuanPerez
  Ej:nombre “Juan” ; rdf:type Ej:Persona ;
  Ej:paginaWeb <http://ejemplo.org/paginaJuan>.

https://www.w3.org/TR/rdf11-primer/#section-turtle-family

RDFS

Vocabularios RDFS

- De manera similar a como XML Schema permite especificar qué tipos de atributos, elementos y relaciones entre elementos puede contener un documento XML, RDF Schema permite definir los tipos de recursos y propiedades que contiene un grafo RDF.
- No se requiere que se defina en un documento separado.

RDFS

Vocabularios RDFS

- Además de facilitar la comunicación entre herramientas, RDF Schema facilita la integración de grafos distribuidos, añade algunos elementos de significado y permite hacer algunas deducciones (inferencias)
- RDF permite establecer propiedades y asociárselas a recursos y literales pero no le atribuye significado a esas propiedades (todavía no se puede manejar automáticamente el significado).

RDFS

Clases y propiedad permitidas

- Jerarquías de clases y subclases
- Herencia entre clases
- Jerarquías de propiedades
- Restricciones de rango y de dominio
- Metadatos
- El vocabulario RDFS extiende el vocabulario RDF y está definido en el espacio de nombres rdfs:<http://www,w3.org/2000/01/rdf-schema#>

RDFS

Clases y propiedades

- Podemos agrupar recursos que tengan características comunes en categorías o clases. Para evitar la confusión con los objetos de las propiedades, a los recursos que son instancias de una clase se les suele denominar individuos.
- Para indicar que un recurso es una clase: ej:VehiculoDeCarga rdf:type rdfs:Class.
- Para indicar que un individuo pertenece a esa categoría: ej:MiFurgoneta rdf:type ej:VehiculoDeCarga.

RDFS

Clases y propiedades

- Se pueden definir jerarquías de clases.
- Para indicar que una clase es una subclase o subcategoría de otra:
  ej:VehiculoDeCarga rdfs:subClassOf ej:VehiculoAMotor.

RDFS

Clases y propiedades

- Permite hacer automáticamente inferencias (deducciones)
  "si MiFurgoneta es una instancia de VehiculoDeCarga y VehiculoDeCarga es una subclase de VehiculoAMotor entonces MiFurgoneta es una instancia de VehiculosAMotor"

RDFS

Clases y propiedades

- Como en otros ámbitos de la informática, una clase representa siempre una categoría, y una subclase una subcategoría.
- Herencia múltiple: En RDFS (y OWL) una clase puede ser subclase de varias clases no relacionadas entre sí, "heredando" las características de todas ellas.

RDFS

Clases y propiedades

- Todas las propiedades se declaran como miembros de la clase rdfs:Property.
- Se puede especificar el rango de una propiedad asociando una propiedad a una clase:
  ej:Varon rdf:type rdfs:Class.
- También se puede utilizar como rango un tipo.
  ej:edad rdfs:range xsd:integer

RDFS

Clases y propiedades

- ej:tienePadre rdf:type rdfs:Property.
  ej:tienePadre rdfs:range ej:Varón.
  Indica que los valores de la propiedad tienePadre son miembros de la clase Varon (las afirmaciones que tengan como propiedad tienePadre tienen como objeto miembros de la clase Varon)

RDFS

Clases y propiedades

- Para especificar que una propiedad se aplica a una clase se usa rdfs:domain
  ej:Libro rdf:type rdfs:Class.
  ej:tieneAutor rdf:type rdfs:Property.
  ej:tieneAutor rdfs:domain ej:Libro.
- Indica que las afirmaciones que tengan como predicado Autor tienen como sujeto instancias o individuos de la clase Libro.

RDFS

Metadatos

- Metadatos son “datos sobre los datos” que se pueden asociar a los recursos:
- rdfs:comment
  Descripción “legible” de un recurso
  ej:persona rdfs:comment
  “Una persona es un ser humano”
- rdfs:label
  Versión “legible” del nombre de un recurso
  ej:persona rdfs:label “Ser humano”

RDFS

Metadatos

- rdfs:seeAlso
  Información adicional sobre un recurso
- ej:persona rdfs:seeAlso
  <http://xmlsns.com/wordnet/1.6/Human>
- rdfs:isDefinedBy Subpropiedad de seeAlso, se usa habitualmente para enlazar a otro grafo RDFS (pero no es imprescindible)
- ej:persona rdfs:isDefinedBy
  <http://xmlns.com/wordnet/1.6/Human>

Otros lenguajes

- OWL (Web Ontology Language)
  Pemite definir clases, propiedades, individuos, restricciones y enunciados lógicos.
- También se utilizan lenguajes y herramientas no estándar.

Estándares asociados a RDF

- RDFa (RDF in annotations)
  Sintaxis de RDF para anotar HTML
- JSON-LD (JSON Linked Data)
  Lenguaje basado en JSON para representar datos enlazados.
  Se utiliza para anotar HTML.
- SPARQL
  Lenguaje de consulta sobre RDF

Estándares asociados a RDF

- LDP (Linked Data Platform)
  Patrones para construir servicios Web tipo REST para leer y escribir RDF
- GRDDL (Gleaning Resource Descriptions from Dialects of Languages)
  Técnica para extraer información expresada en RDF desde documentos XML

Anotación de HTML con información semántica

code URJC

- HTML representa la estructura de un documento, no la estructura de la información, mucho menos su semántica.
- Añadir la estructura de la información y/o enlazar los datos de los documentos con una ontología resultaría útil para muchos propósitos

Anotación de HTML con información semántica

code URJC

- Muchas páginas HTML se generan automáticamente a partir de información estructurada, como bases de datos o documentos XML, por lo que resulta bastante fácil generar también anotaciones.
- Aproximadamente la mitad de los sitios web actuales están anotados con información semántica.

Anotación de HTML con información semántica

code URJC

- En el documento HTML se anotan los datos con identificadores de recursos (individuos) y propiedades, en muchos casos relacionándolos con ontologías o vocabularios externos.
- GRDDL (Gleaning Resource Descriptions from Dialects of Languages). Mecanismo estándar del W3C para obtener RDF a partir de documentos XML (y XHTML) a base de asociarles un algoritmo de transformación, típicamente en XSLT.

Anotación de HTML con información semántica

code URJC

Atributos para anotar elementos

- Microformatos
- Microdatos
- RDFa
- Bien integrado con los datos que se muestran en el documento HTML, por lo que las anotaciones tienden a ser consistentes con los datos.
- Laborioso.

Anotación de HTML con información semántica

code URJC

Elementos flexibles con anotaciones

- Elementos meta (Open Graph, RDFa)
- Scripts (JSON-LD)
- Más fácil para el desarrollador.
- Peor integrado con los datos del documento, y más propenso a inconsistencias.

Anotación de HTML con información semántica

code URJC

- Los buscadores y asistentes utilizan las anotaciones:
  - Para ayudar a detectar si la página es relevante para el usuario.
  - Para añadir enlaces al fragmento de la página que devuelven en el resultado (fragmentos enriquecidos o rich snippets)

Anotación de HTML con información semántica

code URJC

- No las suelen utilizar para determinar el orden de los resultados, pero si pueden contribuir a que detecte nuestra página como relevante para el usuario.
- Depende del caso que a los autores de la página les interese añadir anotaciones reconocibles por los buscadores.

Anotación de HTML con información semántica

code URJC

- schema.org es un vocabulario elaborado conjuntamente por Google, Bing, Yahoo y Yandex. Admite tanto microdatos como RDFa y JSON-LD.
- Estructura basada en RDFS, añade algunas cosas como que los dominios y rangos de las propiedades pueden ser la unión de varias clases y/o tipos de datos.

Anotación de HTML con información semántica

code URJC

- schema.org es un vocabulario elaborado conjuntamente por Google, Bing, Yahoo y Yandex. Admite tanto microdatos como RDFa y JSON-LD.
- Estructura basada en RDFS, añade algunas cosas como que los dominios y rangos de las propiedades pueden ser la unión de varias clases y/o tipos de datos.

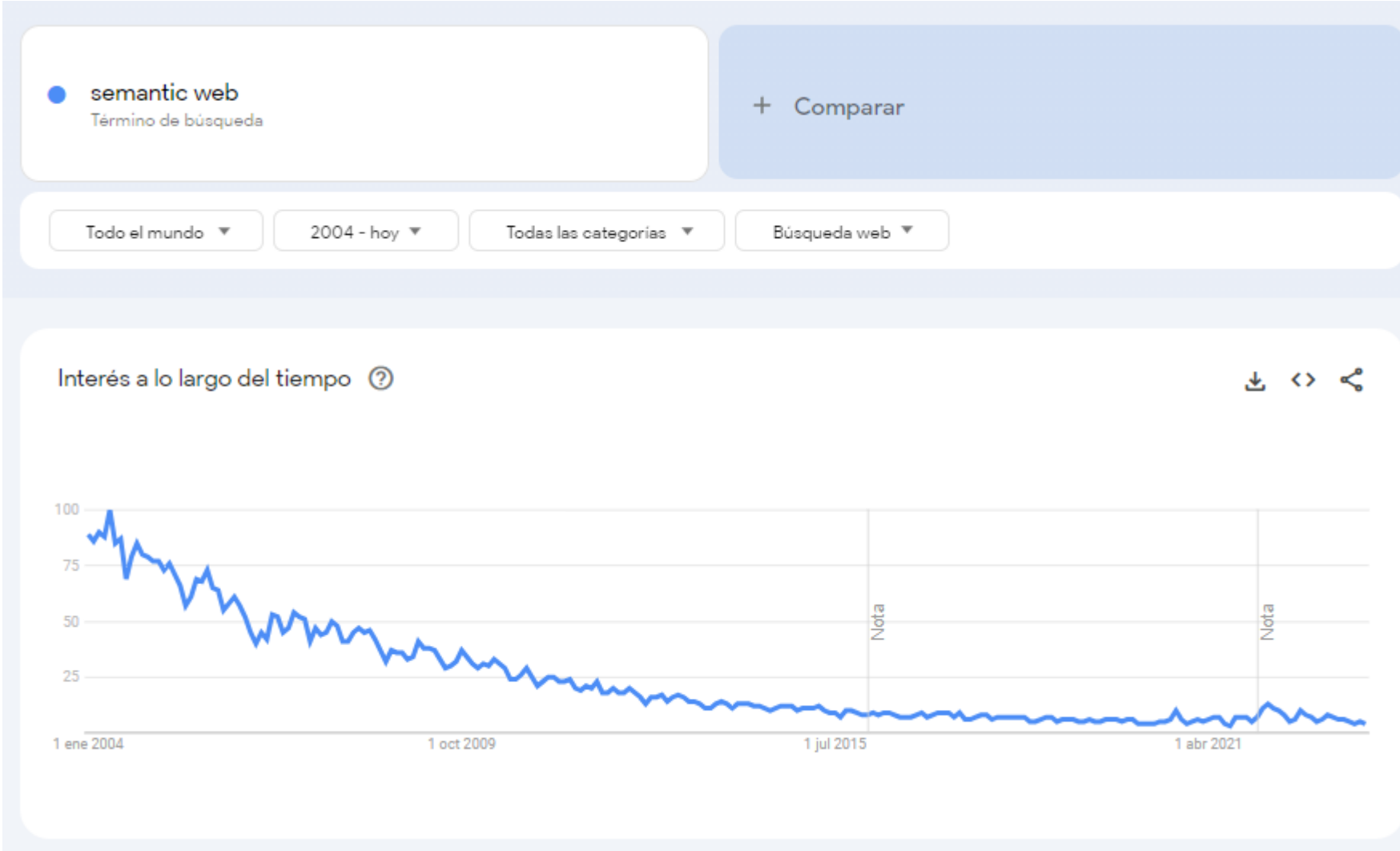
Aplicaciones

- Buscadores semánticos
- Ideas afines (http://www.ideasafines.com.ar)
  Buscador de palabras y conceptos relacionados
- Navegación (http://www.zaragoza.es/tramites/)
  Buscador de trámites del Ayuntamiento de Zaragoza
- Refinamiento de las búsquedas convencionales
- Los grandes buscadores, como Google, Bing, etc, usan técnicas de Web semántica para refinar las búsquedas y para ampliar los resultados que presentan.

Aplicaciones

- Sistemas de información emergentes.
- Sistemas de búsqueda autónomos que informan cuando encuentran algo interesante. Ejemplo: Noticias de la oficina española del Consorcio Web (http:www.w3.es/Noticias/?feed=rdf)
- Resaltar la información más relevante de enlaces, resultados de búsqueda o emails. Ejemplo: páginas web anotadas con Open Graph (FaceBook)
- https://github.com/cyocum/irish-gen

Problemas


semantic web
Término de búsqueda
+ Comparar
Todo el mundo
2004 - hoy
Todas las categorías
Búsqueda web
Interés a lo largo del tiempo
100
75
50
25
Nota
Nota
1 ene 2004
1 oct 2009
1 jul 2015
1 abr 2021

Problemas

- Transformar documentos en datos es complejo
- Lenguajes y herramientas complejos para desarrollar
  - Falta de madurez
- Preferencia por APIs Web usando JSON
- Datasets de mala calidad
- Requiere buena intención por parte de los usuarios que añaden datos
- Avances en ciencia de datos y procesamiento de lenguaje natural reduce la utilidad de la web semántica
- Alto coste en tiempo y dinero sin un beneficio **obvio**

Problemas

https://terminusdb.com/blog/the-semantic-web-is-dead/

https://news.ycombinator.com/item?id=32412803

Bibliografía

- The Semantic Web, Tim Berners-Lee, James Hendler and Ora Lassila, Scientific American, may 17, 2001 http://www.cs.umd.edu/~golbeck/LBSC690/SemanticWeb.html
- Linked Data: Evolving the Web into a Global Data Space, Tom Heath y Christian Bizer, 2011, http://linkeddatabook.com/
- A New Look at the Semantic Web. Communications of the ACM, Septembre 2016, http://cacm.acm.org/magazines/2016/9/206254-a-new-look-at-the-semantic-web/fulltext#FNC

Bibliografía

- EUCLID (European Curriculum for linked Data) http://www.euclid-project.eu/
- Beginner's Guide to RDF. Steve Baskauf. 2014 https://code.google.com/p/tdwg-rdf/wiki/Beginners
- Semantic Web for the Working Ontologist, D. Allemang, J. Hendler, Elservier/Morgan Kaufmann
- https://www.w3.org/TR/rdf11-primer/

Bibliografía

- http://schema.org/docs/full.html
- http://schema.org/docs/datamodel.html
- http://www.google.com/insidesearch/features/search/knowledge.html
  https://html.spec.whatwg.org/multipage/microdata.html
- http://skos.um.es/TR/rdfa-primer/
- http://skos.um.es/TR/2012/REC-rdfa-core-20120607/