Gráficos y visualización 3D

2. Introducción al desarrollo web

Boni García
Web: http://bonigarcia.github.io/
Email: boni.garcia@urjc.es

Dept. Teoría de la Señal y Comunicaciones y Sistemas Telemáticos y Computación (GSyC)
Escuela Superior De Ingeniería De Telecomunicación (ETSIT)
Universidad Rey Juan Carlos (URJC)

Índice de contenidos

Índice de contenidos

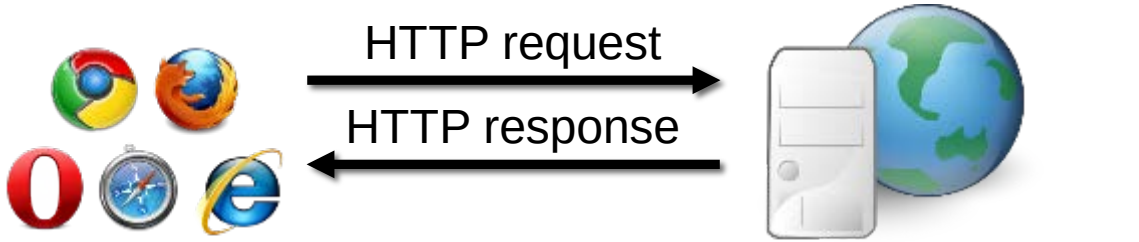
1. Introducción – ¿Qué es la Web?

- La **Web** (*World Wide Web*) es un servicio de distribución de contenidos hipertexto accesibles vía Internet
- **Internet** es un conjunto descentralizado de redes de datos interconectadas que utilizan la familia de protocolos **TCP/IP** que interconecta cientos de millones de dispositivos (hosts o sistemas terminales) en todo el mundo

Aunque la palabra *web* está en la RAE su definición no es muy precisa

1. Introducción – ¿Qué es la Web?

- La Web está basada en una arquitectura **cliente-servidor**



- El protocolo de nivel de aplicación para comunicar clientes y servidores en la Web es **HTTP** (*HyperText Transfer Protocol*)
- Las páginas se identifican con un nombre único llamado dirección web o **URL** (*Uniform Resource Locator*)

1. Introducción – ¿Qué es la Web?

- La Web nació en 1989 de mano de Tim Berners-Lee, físico del CERN (*Conseil Européen pour la Recherche Nucléaire*)
- El objetivo inicial de la WWW (*World Wide Web*) era hacer accesible la gran cantidad de información de los proyectos del CERN
  - Berners-Lee y su equipo escriben el primer servidor web y el primer programa cliente
  - Berners-Lee y su equipo crearon el HTML, el HTTP y las URL
- En 1994 nace el W3C (*World Wide Web Consortium*)

1. Introducción – Aplicaciones web

- Una **página web** es un documento electrónico escrito en **HTML** (*HyperText Markup Language*)
- Las páginas web están enlazadas a través de **hiperenlaces** (*links*)
- Mediante un navegador un usuario puede **navegar** a través de la web siguiendo los hiperenlace
- Un conjunto de páginas alojadas en el mismo servidor se suele conocer como **sitio web**
- En los comienzos de la web, todos los sitios web eran conjuntos de páginas web en forma de ficheros HTML

1. Introducción – Aplicaciones web

- Poco a poco los servidores comenzaron a ejecutar lógica para la generación de las propias páginas web u otras funciones (como por ejemplo el acceso a base de datos) naciendo así las llamadas aplicaciones web
- Una **aplicación web** es aquella aplicación que los usuarios pueden utilizar accediendo a un **servidor web** a través de **Internet** mediante un software que actúa de cliente llamado **navegador**
- Las aplicaciones web son muy populares:
  - El navegador web como cliente universal
  - Proporcionan independencia del sistema operativo
  - Facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales

1. Introducción – Desarrollo web

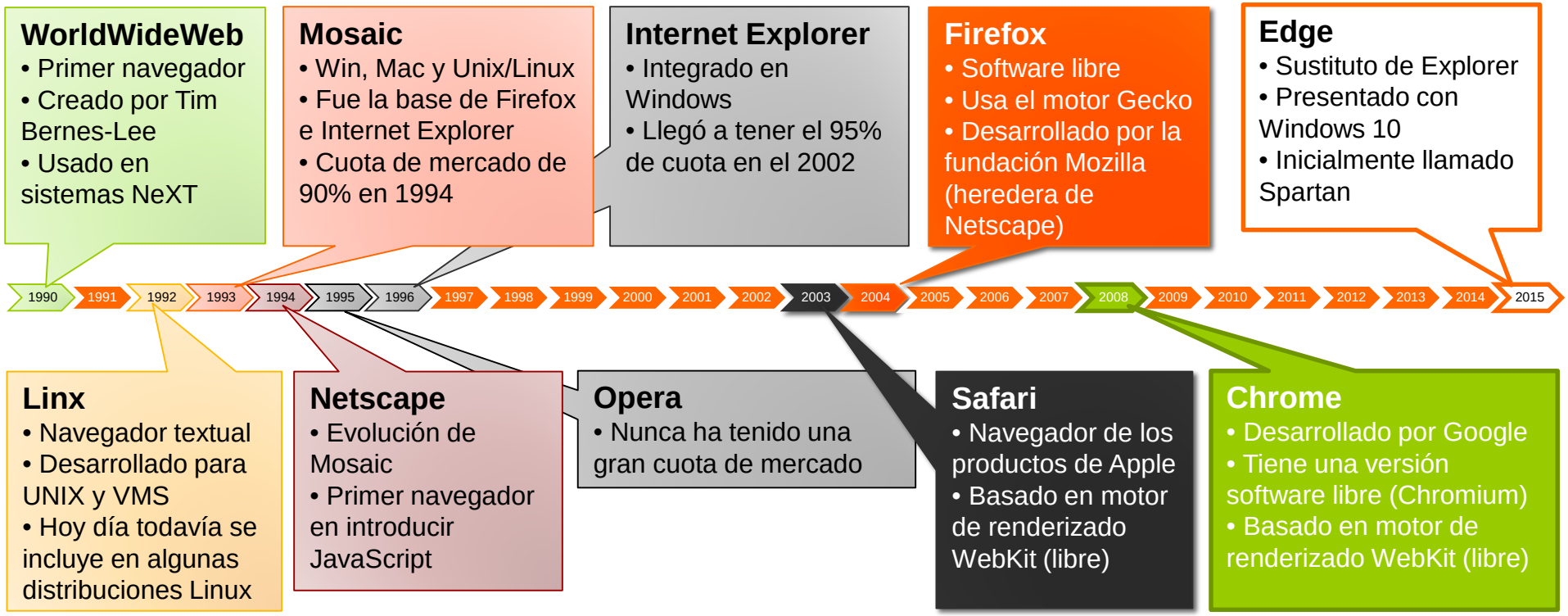
- El impacto de la Web ha propiciado la aparición de diversas tecnologías para la creación de aplicaciones web. Hay dos enfoques principales:
  1. Creación de aplicaciones web con **tecnologías de desarrollo**:
     - **Lado cliente** (*frontend*). Tecnologías que permiten crear interfaces de usuario atractivos y permiten la comunicación con el servidor. Basadas en HTML, CSS y JavaScript
     - **Lado servidor** (*backend*). Tecnologías que permiten implementar la lógica de negocio así como la integración con otros servicios (típicamente con un servidor de base datos)
  2. Creación de aplicaciones web con **sistemas gestores de contenido** (CMS, *Content Management System*)
  3. Últimamente se ha popularizado el uso de **lenguajes de marcados ligeros** para la creación de sitios web estáticos

1. Introducción – Navegadores web

- Un **navegador web** es una aplicación que se instala en el sistema que utiliza el usuario
- El usuario escribe una dirección web (**URL**, *Uniform Resource Locator*). La dirección contiene el nombre del servidor web y el nombre del recurso que se solicita
- El navegador hace una **petición** al servidor y solicita el recurso
- El navegador descarga el recurso y lo visualiza (o lo descarga si no puede hacerlo)
- Si el recurso en una página HTML, además de visualizar su contenido, descarga recursos adicionales como imágenes, estilos, etc. y los visualiza integrados en la página

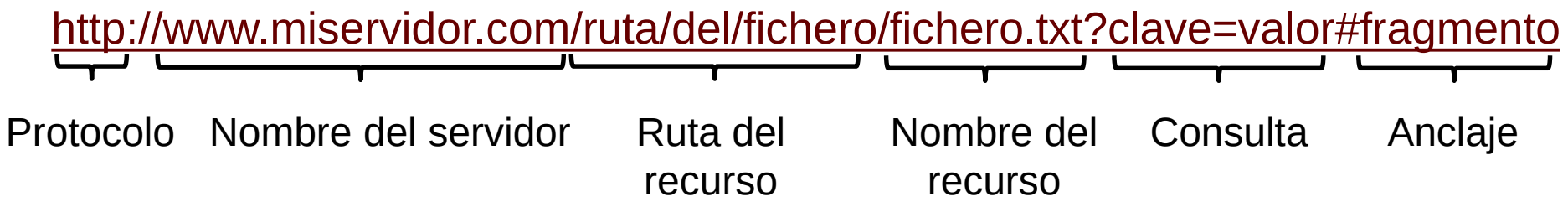
1. Introducción – Navegadores web

- Evolución de los navegadores web

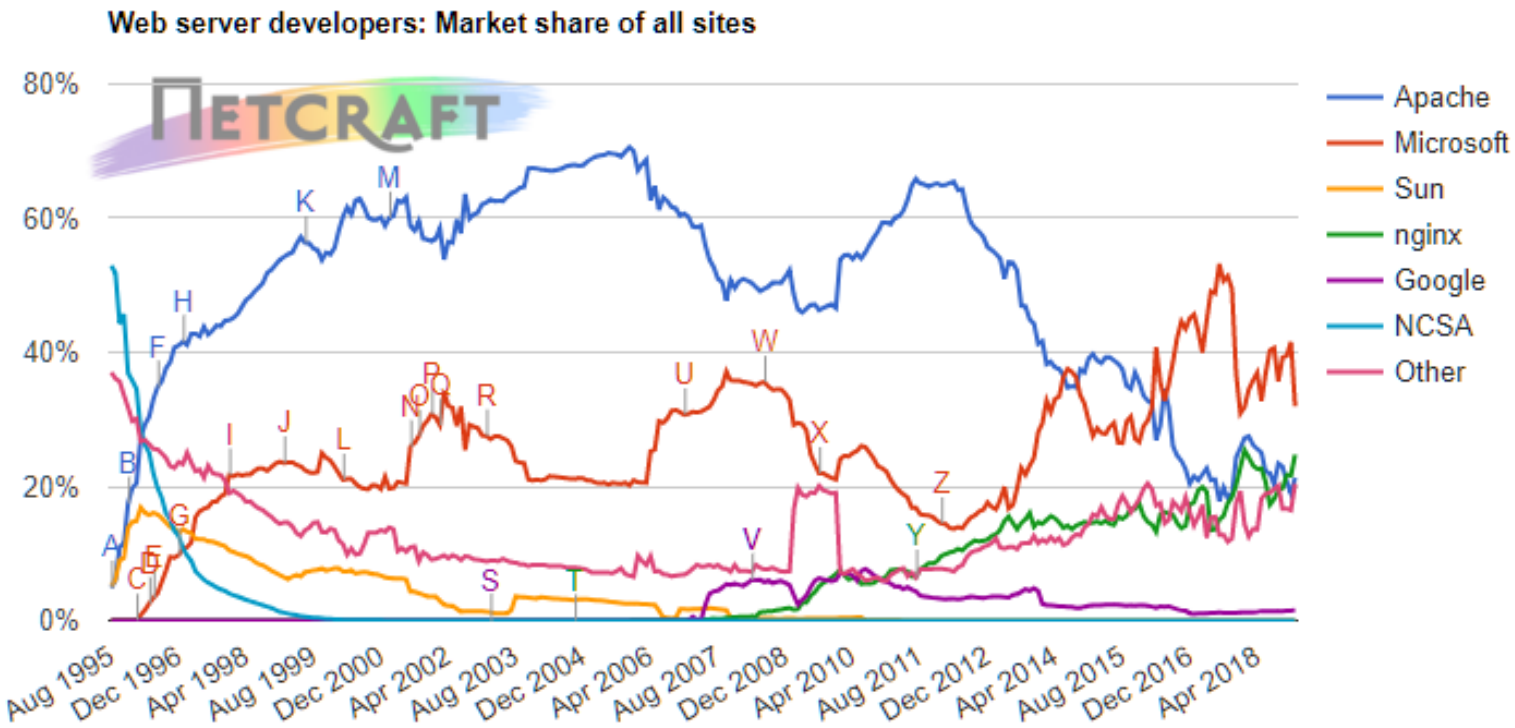


1. Introducción – Servidores web

- Un **servidor web** envía por HTTP los ficheros que tiene almacenados en su disco duro a los clientes que lo solicitan
- Puede servir cualquier tipo de fichero, aunque lo habitual son los ficheros que un navegador reconoce (html, jpg, png, pdf…)
- Cuando recibe una petición, devuelve el fichero del disco duro que se ajuste a la ruta indicada en la URLs



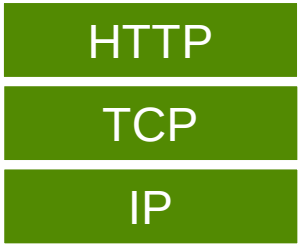
1. Introducción – Servidores web



https://news.netcraft.com/archives/2019/01/24/january-2019-web-server-survey.html

Índice de contenidos

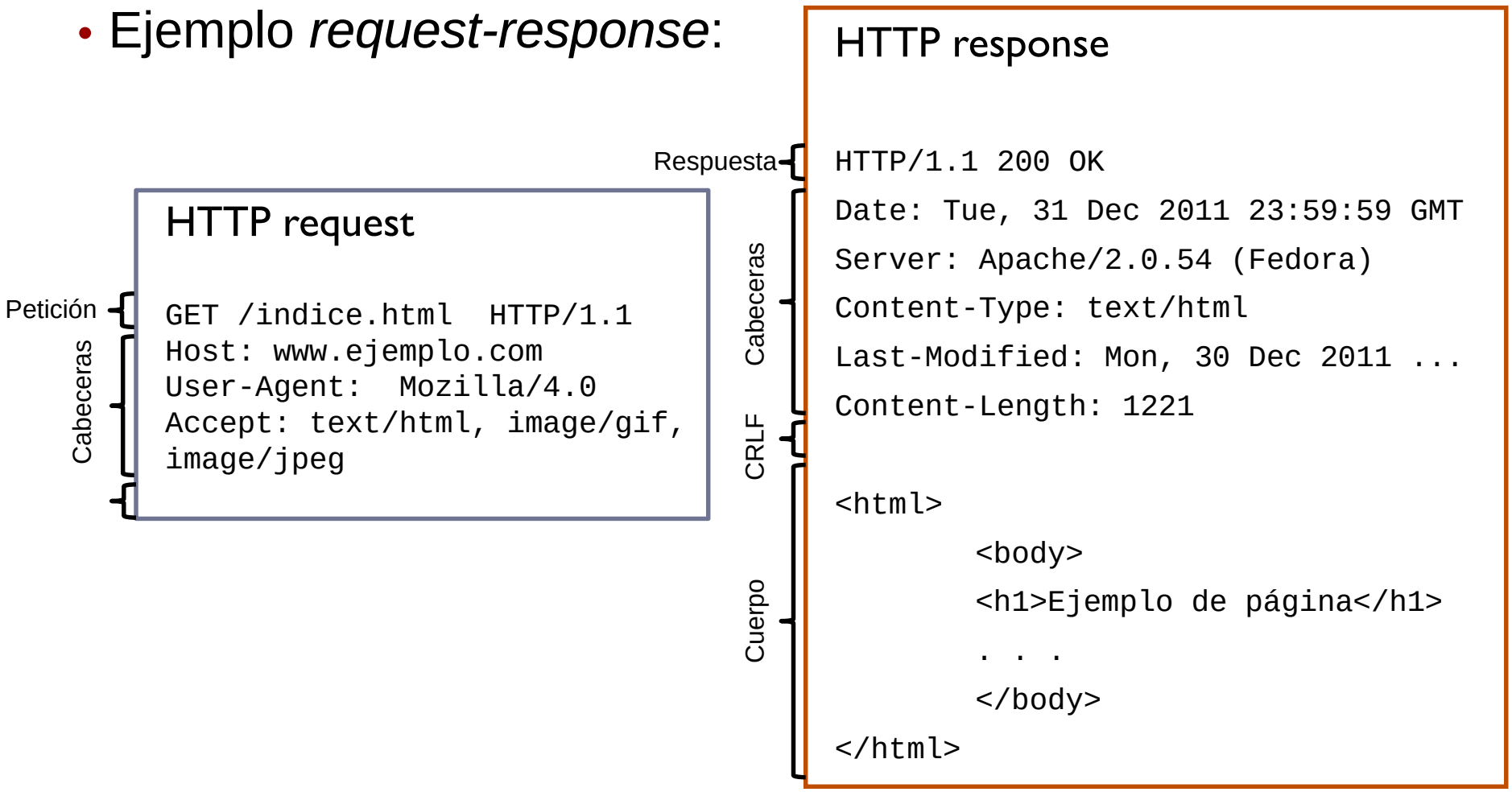
2. HTTP – Visión general



- *HyperText Transfer Protocol*, protocolo de aplicación basado en arquitectura cliente/servidor
- La versión más usada actualmente es HTTP 1.1
  - Desde 2015 ya está disponible HTTP 2.0
- Los servidores web escuchan por defecto en el puerto TCP 80 (*well-known port*)
- Todo elemento web está identificado por una **URL** (*Uniform Resource Locator*)
  - Desde 1994, en los estándares de la Internet (RFCs), el concepto de URL ha sido incorporado dentro del más general de URI (*Uniform Resource Identifier*)
- El cliente (navegador) realiza peticiones (***request***) que causan el envío de una respuesta (***response***) por parte del servidor web

2. HTTP – Visión general

- Ejemplo *request-response*:



2. HTTP – Métodos

- Los métodos en HTTP (algunas veces referido como "verbos") indican la acción que desea que se efectúe sobre el recurso identificado
- HTTP 1.1 (RFC 2616) define 8 métodos:

GET | POST | PUT | DELETE | HEAD | TRACE | OPTIONS | CONNECT

- Hay una extensión a HTTP 1.1 (RFC 5789) que define un nuevo método:

PATCH

2. HTTP – Métodos

Usados en servicios REST

- **GET**: Petición de un recurso determinado (URL)
- **POST**: Envío de datos que serán procesados por un recurso (URL)

Métodos principales

- PUT: Crea un recurso
- DELETE: Borra un recurso

Métodos normalmente deshabilitados en los servidores web

- PATCH: Solicita al servidor la modificación parcial de un recurso

- HEAD: Pide una respuesta idéntica a la que correspondería a una petición GET, pero sin el cuerpo de la respuesta. Esto es útil para conocer las cabeceras de la respuesta pero sin transportar todo el contenido.
- TRACE: Solicita al servidor que envíe de vuelta en un mensaje de respuesta con la petición enviada (servicio de *echo*). Se utiliza con fines de comprobación y diagnóstico.
- OPTIONS: Solicita al servidor los métodos admitidos para un determinado recurso. La respuesta se obtiene en la cabecera `Allow:`
- CONNECT: Se utiliza para indicar a un proxy web que establezca una conexión segura (TLS) con una máquina remota

2. HTTP – Códigos de respuesta

- Las respuestas pueden ser del tipo:
  - 1xx Respuesta informativa. Por ejemplo:
    - 101 Cambio de protocolo
  - 2xx Operación exitosa. Por ejemplo:
    - 200 OK
  - 3xx Redirección. Por ejemplo:
    - 304 No modificado (usado como respuesta en un GET condicional → recurso en caché)
    - 307 Redirección temporal
  - 4xx Error por parte del cliente. Por ejemplo:
    - 404 No encontrado
  - 5xx Error del servidor. Por ejemplo:
    - 500 Error interno

Índice de contenidos

3. HTML – Visión general

- ***Hyper 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
- Versiones:
  - HTML 2.0. Primera versión estándar por el IETF (*Internet Engineering Task Force*) en 1995
  - HTML 4.01: 1999
  - XHTML 1.x: Es una variante de HTML que permite escribir documentos HTML pero como un documento válido XML. Aunque tuvo cierta aceptación en el pasado, actualmente no es muy usado para escribir documentos web
  - **HTML 5**: Versión estable 5.2 liberada en diciembre de 2017, versión 5.3 en desarrollo

3. HTML – Visión general

- HTML es un lenguaje de marcado, esto es, formado por **etiquetas** (*tags*)
- Las etiquetas van entre los símbolos `<` y `>`, y la mayoría va parejas:
  - Etiqueta de apertura: `<html>`
  - Etiqueta de cierre: `</html>`
- Hay algunas etiquetas que por definición no tiene etiqueta de cierre (*void elements*). Por ejemplo: `a`, `link`, `meta`, `br`, `hr`, `img`
- Las etiquetas tienen **atributos** que definen alguna característica
  - En el siguiente ejemplo, `a` es la etiqueta y `href` el atributo (que define del destino del enlace)

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

3. HTML – Formato documentos HTML

- Estructura básica de un documento HTML

Tipo de documento
Instrucción que define el tipo de la versión de HTML en la que está escrita la página

Documento
Cabecera + cuerpo

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Page title</title>
</head>
<body>
    <p>
    This is text in
    <strong>bold</strong> and
    <em>italics</em>.
    </p>
</body>
</html>
```

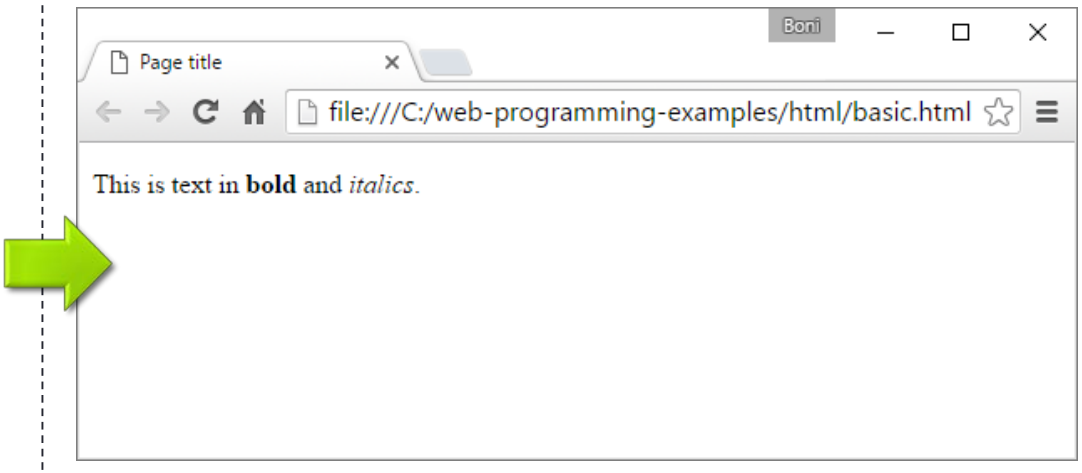
Cabecera
No forma parte del contenido del documento. Título y otra información de alto nivel

Cuerpo
Contenido del documento. Se visualiza en el navegador

3. HTML – Formato documentos HTML

- Estructura básica de un documento HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Page title</title>
</head>
<body>
    <p>
    This is text in
    <strong>bold</strong> and
    <em>italics</em>.
    </p>
</body>
</html>
```



3. HTML – Herramientas de edición

- Existen diferentes alternativas para la edición de documentos HTML:
  - Con un editor de textos:
    - Atom.io
    - Brackets
    - Sublime Text
  - Con un IDE especializado en desarrollo web:
    - Eclipse
    - IntelliJ
    - Visual Studio
  - Con editores online:
    - JSFiddle: http://jsfiddle.net/
    - JSBin: http://jsbin.com/
    - Plunker: http://plnkr.co/

3. HTML – Más información

- La documentación más completa la podemos encontrar en el estándar:
  - http://www.w3.org/TR/html5/
- Hay mucha más información disponible en la Web, por ejemplo:
  - https://developer.mozilla.org/
  - https://github.com/vhf/free-programming-books
  - https://www.freecodecamp.com/
  - http://devdocs.io/
  - http://www.w3schools.com/

Índice de contenidos

4. CSS – Introducción

- 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** (*Scalable Vector Graphics*): Gráficos vectoriales
  - **XML** (*eXtensible Markup Language*): Lenguaje de marcado
- Con **CSS** se pueden especificar:
  - **Colores:** principal, de fondo, degradados, etc.
  - **Tipografía:** familia, tamaños, etc.
  - ***Layout:*** Disposición de los elementos en el documento
  - **Efectos:** sombras, esquinas redondeadas, etc.

4. CSS – 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,…

Valor
Definición del estilo. Por ejemplo, color rojo, tamaño en píxel, …

```
selector {
    property1: value1;
    property2: value2;
    property3: value2;
}
```

4. CSS – Formato

- Ejemplo:

```
<h3>What's CSS for?</h3>
<p>CSS is for styling HTML
pages!</p>
<h3>Why use it?</h3>
<p>
It makes web pages look
<span>really cool</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;
}
```

4. CSS – CSS en HTML

1. Incluido en un elemento

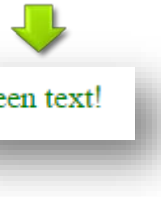
- Mediante el atributo HTML `style`:

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

2. Incluido en la página

- En la sección `head` con la etiqueta `style`:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
color: green;
}
</style>
</head>
<body>
<p>Green text!</p>
</body>
</html>
```



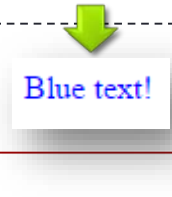
3. En fichero independiente

- Enlazado con etiqueta `link` en sección `head`:

```
<!DOCTYPE html>
<html>
<head>
<link type="text/css"
rel="stylesheet"
href="styles.css">
</head>
<body>
<p>Blue text!</p>
</body>
</html>
```

styles.css

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



Índice de contenidos

5. JavaScript – Introducción

- **JavaScript** es un lenguaje de programación de alto nivel que se puede ejecutar en los navegadores web
- Surgió en 1995, implementado en el navegador Netscape
- El nombre JavaScript se eligió al nacer en una época en la que Java estaba en auge
- JavaScript permite incorporar interactividad en el lado cliente de las aplicaciones web, por ejemplo:
  - Manipulación de las páginas web (DOM, *Document Object Model*)
  - Peticiones en segundo plano (AJAX, *Asynchronous JavaScript And XML*)
  - Generar gráficos mediante WebGL

5. JavaScript – Introducción

- El estándar que define el lenguaje JavaScript se llama **ECMAScript**
- Este estándar lo publica la organización de estandarización **ECMA** (*European Computer Manufacturers Association*)
- Llamamos JavaScript a la implementación de ECMAScript de los diferentes navegadores
- La versión mínima soportada por los navegadores modernos es ES5

5. JavaScript – Características

- **Imperativo y estructurado**
  - Se declaran **variables**
  - Se ejecutan las sentencias **en orden**
  - Dispone de **sentencias de control** de flujo de ejecución
- **Scripting:** Tradicionalmente JavaScript ha sido interpretado en el navegador
  - Esto no es cierto para todos los navegadores
  - Por ejemplo, el motor de JavaScript de Google Chrome (llamado V8) **compila** el código JavaScript a código máquina siguiendo un enfoque JIT (*just-in-time*), esto es, realizando optimizaciones al código compilado en tiempo de ejecución (cosa que un intérprete no hace)

5. JavaScript – Características

- **Tipado dinámico**
  - Al declarar una variable no es necesario definir su tipo
  - A lo largo de la ejecución del programa una misma variable puede tener valores de diferentes tipos
- **Orientado a objetos**
  - La orientación a objetos está **basada en prototipos** (se pueden crear objetos, añadir atributos y métodos en tiempo de ejecución)
  - Existe un **recolector de basura** para los objetos que no se utilizan
- **Funcional**
  - Las funciones en JavaScript son elementos de primera clase, o sea, permite declarar **funciones independientes**
  - Esas funciones se pueden declarar en cualquier sitio, asignarse a variables, pasarse como parámetro

5. JavaScript – Sintaxis básica

- Variables en JavaScript:
  - Es un lenguaje con tipado **dinámico** (las variables se declaran con `var` pero no se indica el tipo):
  - Tipos de datos:
    - `Number` (números enteros y reales de cualquier precisión)
    - `String` (Cadenas de caracteres, separadas por comillas simples o dobles)
    - `Boolean` (`true` o `false`)
  - La variable tiene como ámbito la función, no por bloque
  - Si no se inicializan, las variables tienen el valor `undefined`

```
var count = 10;
var found = false;
var name = 'John';
```

5. JavaScript – Sintaxis básica

- Las funciones en JavaScript se pueden declarar mediante un nombre de función que luego será usado en la invocación:

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

func(4); // Prints 4 in the console
```

- Se pueden declarar sin nombre (anónimas) y asignarse a una variable o usarse como parámetro:

```
var func2 = function(param) {
   console.log(param);
}

func2(5); // Prints 5 in the console
```

5. JavaScript – Sintaxis básica

- El *Document Object Model* (DOM) es la API para manipular documentos HTML desde JavaScript
- Cuando se carga una página HTML en un navegador, se convierte en un objeto tipo `document`
- Todos los elementos HTML están representado por el objeto `element`
- Acceso a un elemento del documento usando el atributo `id` (el atributo `id` debería ser único en la página. Si no lo fuese, `getElementById` devuelve el primer elemento con dicho `id`):

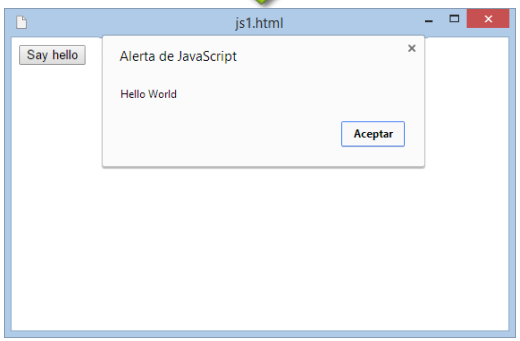
```
var obj1 = document.getElementById("objId");
```

5. JavaScript – JavaScript en HTML

1. Incluido en un elemento

- Mediante eventos HTML (*onclick*, …):

```
<!DOCTYPE html>
<html>
<body>
<button
onclick="alert('Hello
World');">Say
hello</button>
</body>
</html>
```



2. Incluido en la página

- En la sección `head` o `body` con la etiqueta `script`:

```
<!DOCTYPE html>
<html>
<head>
<script>
function hello() {
  alert('Hello World');
}
</script>
</head>
<body>
<button
onclick="hello();">Say
hello</button>
</body>
</html>
```

3. En fichero independiente

- Enlazado con etiqueta `script` en sección `head` o `body`:

```
<!DOCTYPE html>
<html>
<head>
<script
src="script.js"></script>
</head>
<body>
<button
onclick="hello();">Say
hello</button>
</body>
</html>
```

script.js

```
function hello() {
   alert('Hello World');
}
```

5. JavaScript – jQuery

- **jQuery** es una librería **JavaScript** cuyo objetivo es simplificar y unificar el código JavaScript en los diferentes navegadores
  - Unifica las diferencias que existen en los navegadores
  - Tiene una API fluida (*fluent API*) para reducir el uso de variables locales y poder escribir scripts en una línea
  - Existen funcionalidades adicionales en forma de plugins, de forma que la librería se mantiene pequeña

http://jquery.com

5. JavaScript – jQuery

- Podemos usar jQuery enlazando con una red de distribución de contenidos (CDN, *Content Delivery Network*):
  - jQuery CDN: https://code.jquery.com/

```
<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
```

  - Google CDN: https://developers.google.com/speed/libraries/devguide

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

En la versiones *minificadas* se eliminan bytes innecesarios (espacios, sangrías) para que el tamaño de la librería sea menor y se reduzca así el tiempo de carga

5. JavaScript – jQuery

- Con jQuery se seleccionan elementos HTML (*query*) y ser realizan acciones sobre estos elementos. La sintaxis básica general es:

```
$(selector).action();
```

… donde:

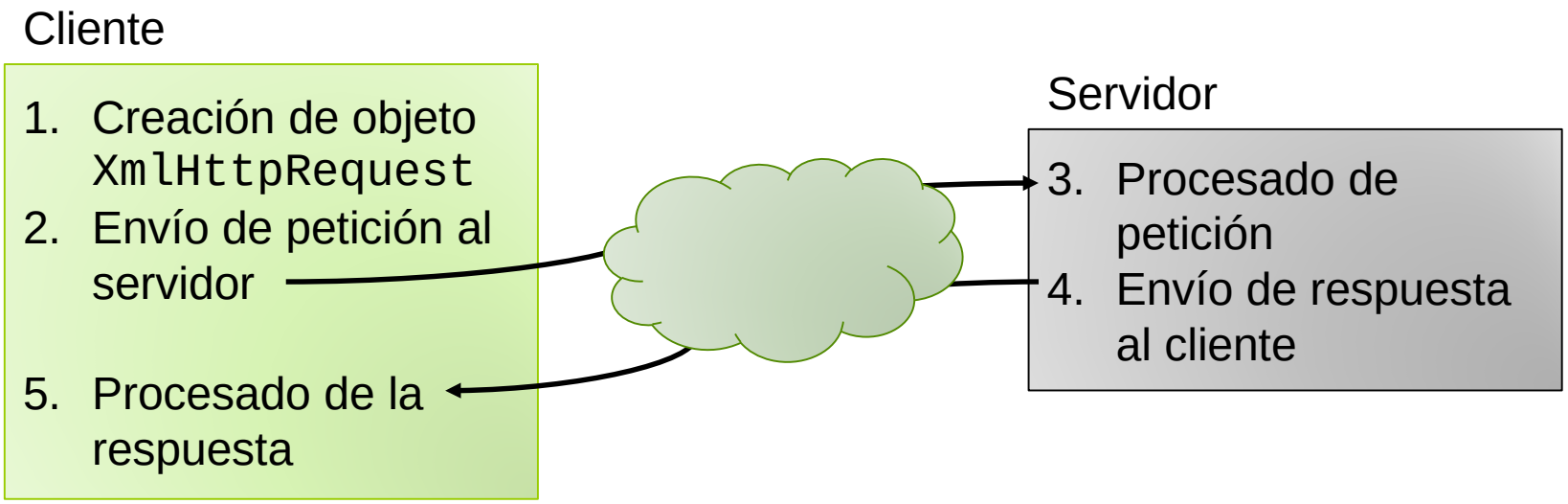
- `$` : Operador genérico para jQuery
- `(selector)` : Elemento(s) sobre los que realización la acción
- `action()` : Acción a ser ejecutada en los elementos seleccionados

- Ejemplos:

```
$("p").hide(); // hides all <p> elements
$(".test").hide(); //hides all elements with class="test"
$("#test").hide(); // hides the element with id="test"
```

5. JavaScript – AJAX

- **AJAX** (*Asynchronous JavaScript And XML*), es una técnica que nos permite hacer peticiones en segundo plano (sin recargar la página) a un servidor web
- En los navegadores modernos, se implementa a través del objeto `XMLHttpRequest`



5. JavaScript – AJAX

- Constructor del objeto `XMLHttpRequest`:

```
var xhr = new XMLHttpRequest();
```

- Métodos importantes del objeto `XMLHttpRequest`:
  - `xhr.open(method, url, async);` (especifica la petición)
    - `method` puede ser `"GET"` o `"POST"`
    - `url` es la ruta en el servidor destino
    - `async` determina el tipo de petición: síncrona (`false`) o asíncrona (`true`)
  - `xhr.send();` (envía la petición)
  - `xhr.abort();` (cancela la petición actual)

5. JavaScript – AJAX

- Constructor del objeto `XMLHttpRequest`:

```
var xhr = new XMLHttpRequest();
```

- Métodos importantes del objeto `XMLHttpRequest`:
  - `xhr.open(method, url, async);` (especifica la petición)
    - `method` puede ser `"GET"` o `"POST"`
    - `url` es la ruta en el servidor destino
    - `async` determina el tipo de petición: síncrona (`false`) o asíncrona (`true`)
  - `xhr.send();` (envía la petición)
  - `xhr.abort();` (cancela la petición actual)

5. JavaScript – AJAX

- Propiedades importantes del objeto `XMLHttpRequest`:

  `xhr.onreadystatechange` define la función que atiende a la respuesta (*callbak*)

  `xhr.readystate` (especifica el estado de la petición a nivel AJAX)

  - `0` petición no inicializada
  - `1` conexión establecida con el servidor
  - `2` petición recibida
  - `3` petición en proceso
  - `4` petición finalizada y respuesta lista

  `xhr.status` (especifica el estado de la petición a nivel HTTP)

  - `200` ok
  - `403` prohibido
  - `404` no encontrado

5. JavaScript – Política del mismo origen

- La **política del mismo origen** (*same-origin policy*) es una medida de seguridad implementada en los navegadores para evitar cargar scripts cuyo *origen* sea diferente
  - Se entiende origen como la parte inicial de la URL (combinación de protocolo + host + puerto)
- A veces esta política es demasiado restrictiva, y se puede relajar permitiendo orígenes cruzados (**CORS,** *Cross-origin Resource Sharing*) en los servidores web
  - Es un estándar del W3C que permite saltarse la política del mismo origen usando ciertas cabeceras HTTP (`Access-Control-Allow-Origin`)

https://www.w3.org/TR/cors/

5. JavaScript – Política del mismo origen

- Debido a la política del mismo origen, a veces en el desarrollo local de páginas web, es necesario usar un servidor web local para servir las páginas que estamos desarrollando (por ejemplo, al hacer peticiones AJAX)
- Algunas alternativas sencillas para usar un servidor web local:
  - Usando Python desde la línea de comandos (Shell):

```
python -m SimpleHTTPServer
```

  - Como extensión de Chrome: Web server for Chrome