



Desarrollo de servicios en la nube con HTML5, CSS3, JavaScript-6-7-.. y node.js

Juan Quemada, DIT - UPM

Introducción al curso:

Desarrollo de servicios en la nube con HTML5, CSS, JavaScript-6-7-.. y node.js

1.	<u>Introducción al Curso</u>	<u>3</u>
2.	<u>Internet y la plataforma Web</u>	<u>10</u>
3.	<u>La plataforma Web actual: los nuevos clientes y servidores</u>	<u>15</u>
4.	<u>Introducción a JavaScript 6-7-8-9 (ES6 o ES2015, ...)</u>	<u>23</u>



Introducción al Curso

Juan Quemada, DIT - UPM

Programa MiriadaX: en Diseño de servicios en la nube



- ◆ Desarrollo en HTML5, CSS y Javascript de Apps Web, Android e IOS.

- <https://miriadax.net/web/html5mooc>

- ◆ Client-based Web Applications development: ReactJS & Angular

- <https://miriadax.net/web/client-based-web-applications-development-reactjs-angular0>

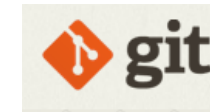


- ◆ Introducción a Linux como entorno de desarrollo de sist. software <- **prerrequisito**

- <https://miriadax.net/web/introduccion-a-linux-como-entorno-de-desarrollo-de-sistemas-software-2-edicion->

- ◆ Gestión de proyectos Software con Git y GitHub

- <https://miriadax.net/web/gitmooc>



- **prerrequisito**

- ◆ **Desarrollo de servicios en la nube con HTML5, CSS3 Javascript-6-7-.. y node.js**

- <https://miriadax.net/web/nodeMOOC>

- ◆ Proyecto de diseño con node.js, express.js y HTML5 de un portal de juegos

- <https://miriadax.net/web/portalnnodeMOOC>



El Curso

◆ Curso de nivel medio **orientado a proyecto**

- Basado en el trabajo activo de los participantes

◆ El curso incluye

- Programación con **JavaScript 6, 7, 8, 9, ..**
 - ◆ Se hace especial hincapié en las características mas novedosas
- Desarrollo de **aplicaciones de servidor** para la **Plataforma Web**
 - ◆ Se incluyen conceptos, arquitectura, librerías mas populares y ejemplos
- **Ingeniería de Software** para desarrollo de proyectos
 - ◆ Se utiliza profusamente en el curso y se deben tener conocimientos básicos

◆ **MOOCs previos** (o tener conocimientos equivalentes)

- Introducción a Linux como entorno de desarrollo de sistemas software
 - ◆ <https://miriadax.net/web/introduccion-a-linux-como-entorno-de-desarrollo-de-sistemas-software-2-edicion->
- Gestión de proyectos Software con Git y GitHub
 - ◆ <https://miriadax.net/web/gitmooc>

Estructura del curso

◆ El curso consta de **11 módulos** (de 0 a 10)

- **Esfuerzo:** 4-5 semanas de trabajo a 10-12 horas/semana
 - ◆ Conocimientos previos de programación, aunque no de JavaScript

◆ El curso estará **abierto ~6 meses**

- Todos los módulos y actividades estarán abiertas desde el primer día hasta el cierre del curso

◆ **Recomendamos realizar** los módulos en el **orden dado**

- Aunque cada participante puede realizar los módulos y actividades en el orden y el momento que desee

Estructura típica de un Módulo

- ◆ Tarea 0: Descargar transparencias y ejemplos ejecutables
 - Cuando el tema lo aconseja, se agrupan las transparencias de varios modelos en el primero
- ◆ **Micro-actividades de aprendizaje** (de 4 a 10 por módulo):
 - Explican un tema dando los conceptos clave y ejemplos explicativos
 - ◆ Constan de un **video** o **screencast** del tema (de 5 y 15 minutos)
 - ◆ Suelen tener un **test** o un **ejercicio P2P**, que puede ser obligatorio
- ◆ Ejercicios de final de módulo
 - Ejercicios principales para practicar lo explicado en el módulo
 - ◆ Son ejercicios activos donde se practica y se continua aprendiendo

Equipos y herramientas a utilizar

◆ Un **PC** o **portatil** de trabajo con su sistema operativo

- **UNIX:** Mac OS X (BSD UNIX), Linux (Ubuntu, ..),
 - ◆ Windows se puede utilizar en modo comando, pero el soporte es menos estándar

◆ Editor de textos que se use habitualmente

- Recomendados: Sublime (sublimetext.com) o ATOM (<https://atom.io/>)

◆ **IDE** - Integrated Development Environment

- **Visual Studio** (gratuito): <https://www.visualstudio.com/es/>
- **Webstorm** (<https://www.jetbrains.com/webstorm/>)
 - ◆ Gratis para estudiantes (<https://www.jetbrains.com/student/>)

◆ **Navegador:** Firefox, Chrome, Safari, Edge, ..

- Con sus entornos de desarrollo

◆ **GITHUB**

- Cuenta para compartir proyectos GIT (<https://github.com>)

Metodología AMMIL y Plató SAGA

- ◆ Este curso esta desarrollado con la metodología AMMIL
 - **AMMIL - Active Meaningfull Micro-Inductive Learning**
 - ◆ Video explicativo: <https://innovacioneducativa.upm.es/saga/metodologia-ammil>
- ◆ Los videos se han grabado con el plató SAGA
 - **SAGA - Sistema Autonomo de Grabación Avanzada**
 - ◆ Video explicativo: <https://innovacioneducativa.upm.es/saga/plato-saga>
- ◆ Instrucciones de configuración de un plató SAGA
 - <https://innovacioneducativa.upm.es/saga/configuracion-del-plato-saga>

Internet y la plataforma Web

Juan Quemada, DIT - UPM

Santiago Pavón, DIT - UPM

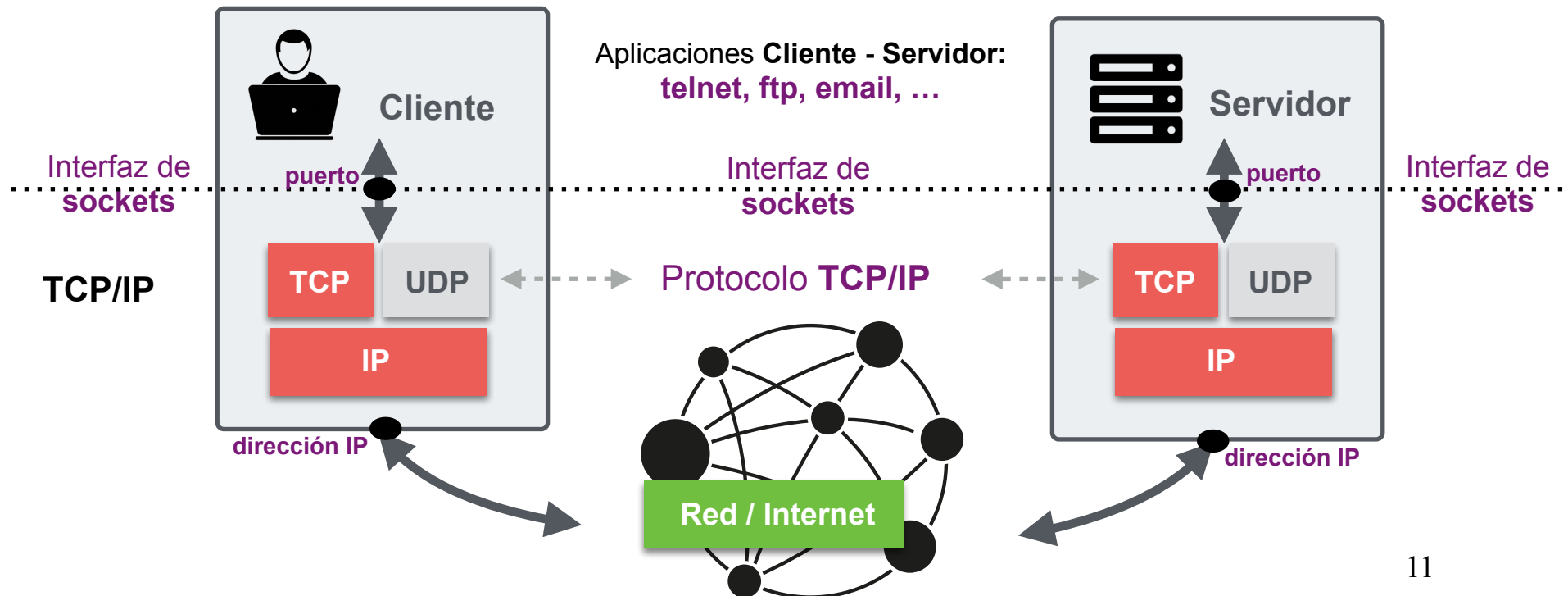
Internet y la arquitectura TCP/IP

◆ Internet empieza a operar en Arpanet el 1 de Enero 1983

- Internet conecta ordenadores a Internet con la pila de protocolos TCP/IP
 - ◆ TCP/IP soporta **aplicaciones cliente-servidor** con el **interfaz de sockets**
 - La dirección IP identifica el ordenador en Internet y el puerto identifica la aplicación dentro del ordenador

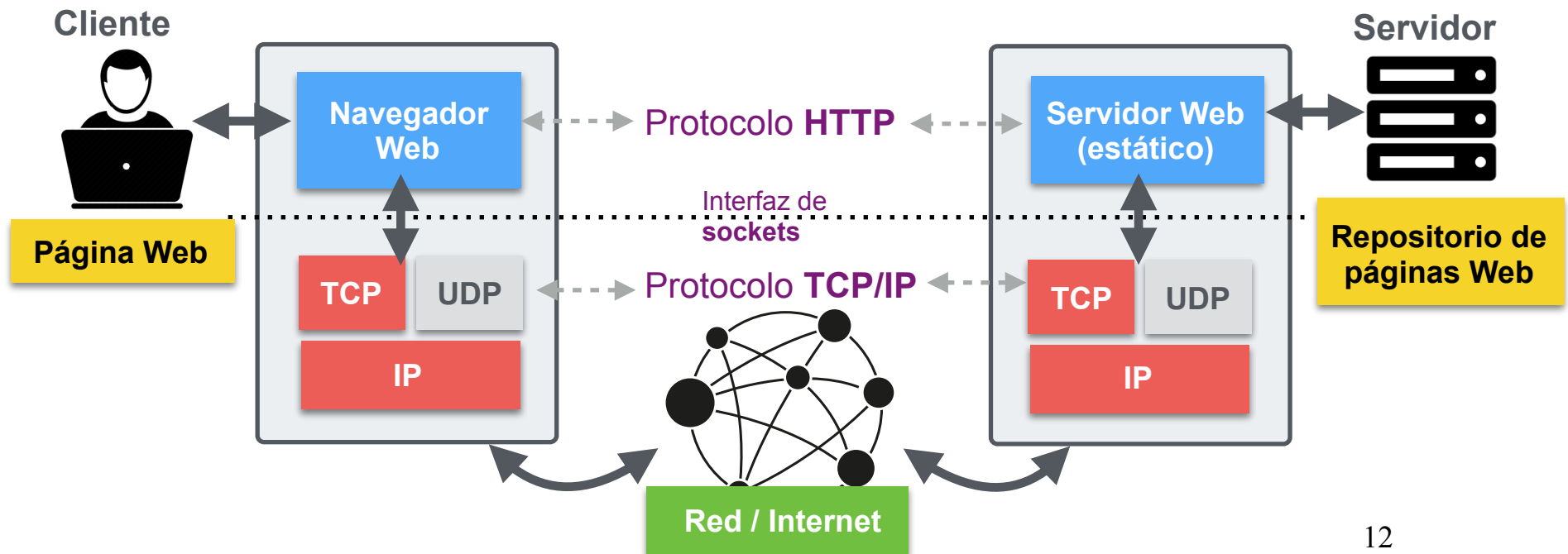
◆ Las primeras aplicaciones cliente servidor de Internet son

- telnet (terminal virtual), ftp (transferencia de ficheros), email (correo elec.), ..



La Web

- ◆ Tim Berners Lee propone en 1989 una nueva aplicación: la Web
 - Servicio de publicación de documentos hipertexto en Internet
 - ◆ Aplicación cliente (**navegador**) <--> servidor (**servidor Web estático**)
- ◆ La Web es el almacén de contenidos que necesitaba la red
 - Transforma Internet en una **"Red de distribución de contenidos"**
 - ◆ Crece continuamente -> es **descentralizada y escalable**



La Web inicial

◆ URL

- **Dirección única** a un fichero (o sección) en un servidor de Internet
 - ◆ Ejemplo: <https://en.wikipedia.org/wiki/URL>

◆ HTTP

- **Protocolo** para traer **ficheros** de un **servidor remoto**
 - ◆ Protocolo simple y *¡muy escalable!*
- El fichero se identifica con un **URL**

◆ HTML

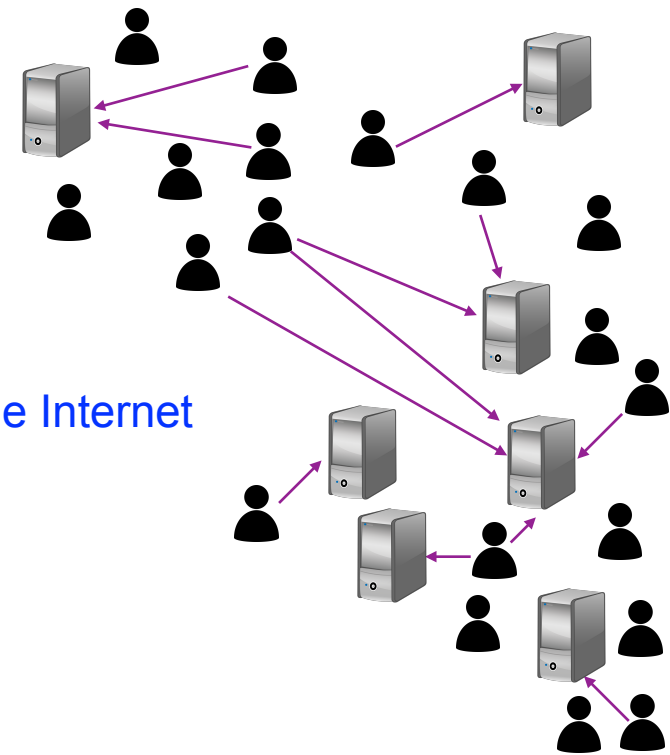
- **Lenguaje** para definir **páginas Web** (con **hiperenlaces**) para visualizar en el navegador

◆ Cliente Web (navegador)

- Programa para visualizar páginas Web (HTML) traídas de un servidor con HTTP

◆ Servidor Web estático

- Programa que sirve páginas Web (ficheros HTML) a los clientes que las solicitan



La Web inicial

◆ Servidor Web estático

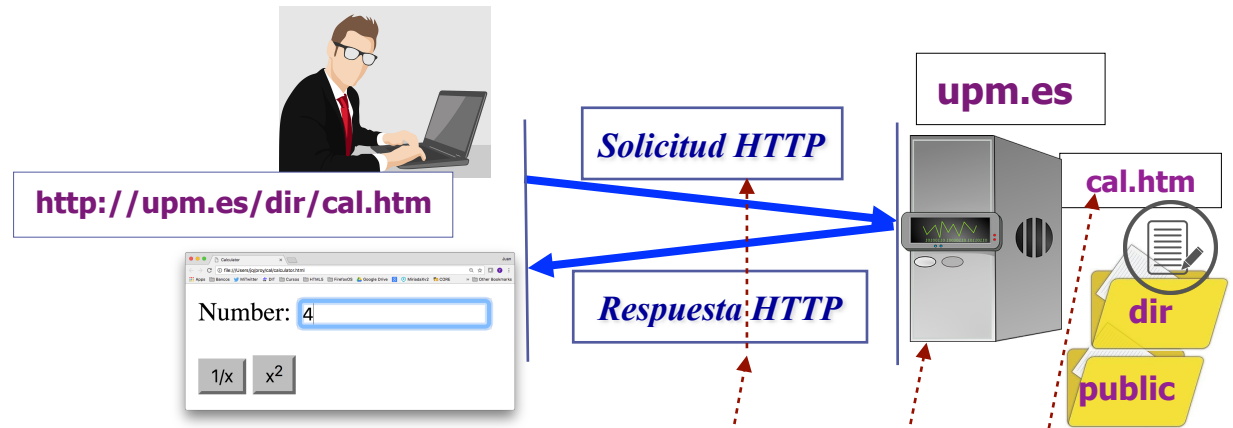
- Programa que sirve ficheros solicitados por clientes
 - ◆ Los ficheros están en un directorio de recursos (páginas Web)
 - El directorio de recursos suele ser: www, public, ..

◆ Cliente Web

- Presenta páginas Web traídas de un servidor en Internet
- El URL identifica el recurso Web: **http://upm.es/dir/cal.htm**
 - ◆ **http:** El **protocolo** de acceso al servidor (HTTP GET)
 - ◆ **upm.es:** La dirección de dominio del **servidor** que alberga la página
 - ◆ **/dir/cal.html :** La ruta al **fichero** (página Web) en el directorio de recursos del servidor

◆ La transacción HTTP vista desde el cliente:

- Establece una conexión TCP con el servidor (**upm.es**)
- Envía por la conexión una **Solicitud HTTP** con la ruta al recurso Web (**/dir/pagina.htm**)
- Recibe por la conexión la **Respuesta HTTP** con el fichero (**página Web**)
- El servidor cierra la conexión TCP



La plataforma Web actual: los nuevos clientes y servidores

Juan Quemada, DIT - UPM
Santiago Pavón, DIT - UPM

Computación distribuida y la plataforma Web

◆ Paradigma de **computación distribuida**

- Partes de un programa cooperan en un objetivo común conectados por Internet
 - ◆ Plantea múltiples retos relacionados con la concurrencia entre procesos y la comunicación entre ellos, transacciones seguras, sincronización de relojes, tolerancia a fallos de las partes, etc.
- Se han propuesto diversas plataformas: Web, CORBA, Fractal, JavaBeans, NFS, AFS, ..
 - ◆ La **plataforma Web** es el entorno más utilizado para el desarrollo de servicios en Internet

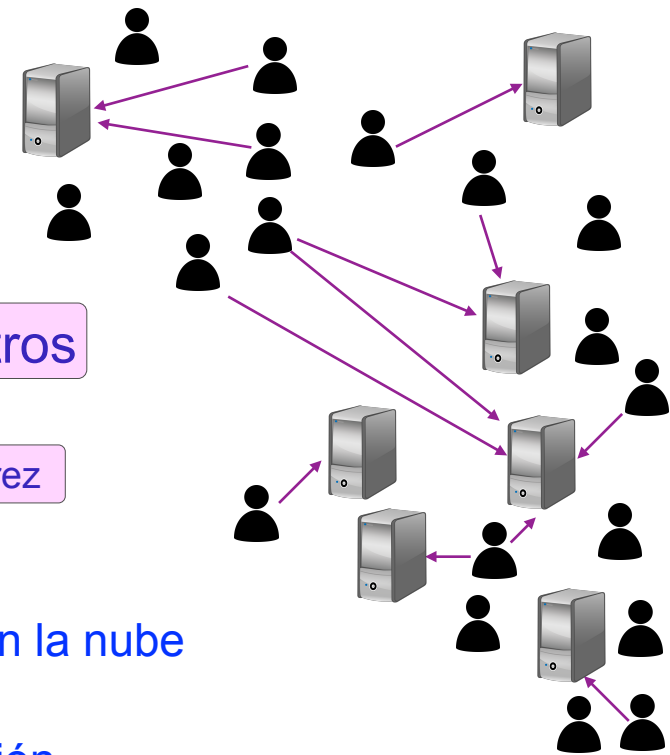
◆ La **plataforma Web**

- Arquitectura descentralizada basada en el modelo cliente <-> servidor para
 - ◆ Aplicaciones de sobremesa, teléfonos móviles u otros dispositivos
 - ◆ Servicios en la nube
 - ◆ Intranets y aplicaciones corporativas
 - ◆ Aplicaciones P2P (Pier to Pier)
 - ◆ etc.

◆ Este **curso** describe

- Los componentes más importantes de la plataforma Web
- Las técnicas de desarrollo de aplicaciones
- El lenguaje JavaScript para programación de aplicaciones

La plataforma Web actual



- ◆ **URL** -> Se añade la **query** para envío de parámetros
 - **Transacción** con parámetros para acceder a servicios
 - ◆ Por ejemplo: `https://upm.es/registro?nombre=José&apellido=Perez`
- ◆ **HTTP** -> **HTTP/2, WebSockets, WebRTC, ...**
 - Se añaden nuevos **protocolos** para crear aplicaciones en la nube
 - ◆ Protocolos *muy escalables*
 - Los nuevos protocolos soportan cualquier tipo de aplicación
- ◆ **HTML** -> Aplicaciones Web en **HTML, CSS y JavaScript**
 - **Aplicaciones Web** de cliente (con hiperenlaces) que se ejecutan en el navegador
- ◆ **Cliente Web (navegador)** -> Aparecen los **móviles** con sus **apps**
 - Los **clientes** web se hacen **programables**
- ◆ **Servidor Web estático** -> **Servidor Web dinámico** (programable)
 - Los **servidores** se hacen **programables** y se conectan a BBDDs

URL y URIs

◆ URL (Uniform Resource Locator)

- **Dirección** de acceso a cualquier **recurso** o **servicio** de Internet
 - ♦ Los **URLs** (RFC1738) son un caso particular de los **URIs** (Uniform Resource Identifiers, RFC3986)
 - <https://www.ietf.org/rfc/rfc1738.txt> y <https://tools.ietf.org/html/rfc3986>

scheme://user:password@host:port/path?query#fragment

◆ <http://upm.es/dir/pagina.html>

- URL Web que identifica e la página Web **/dir/pagina.html** en el servidor **upm.es**

◆ <http://upm.es:8080/dir/pagina.html>

- URL Web similar a la anterior, donde el servidor escucha en el **puerto 8080** y no en el 80 asignado a Web

◆ <http://upm.es/dir/pagina.html#p3>

- URL igual al anterior pero con **fragment** o **anchor** (ancla), que identifica el elemento con **id='p3'** en **pagina.html**

◆ <http://felix@upm.es/dir/pagina.html>

- URL Web de un recurso asociado al usuario **felix** en su cuenta en el servidor **upm.es**
 - ♦ Se recomienda enviar passwords en URLs solo con HTTPS y no con HTTP, porque es inseguro

◆ <http://upm.es/registro?id=23&nombre=José>

- URL que envía dos parámetros en la query (parámetros **id** y **nombre**)

◆ <mailto:felix@upm.es>

- URL de email que identifica el buzón del usuario **felix** en el servidor **upm.es**

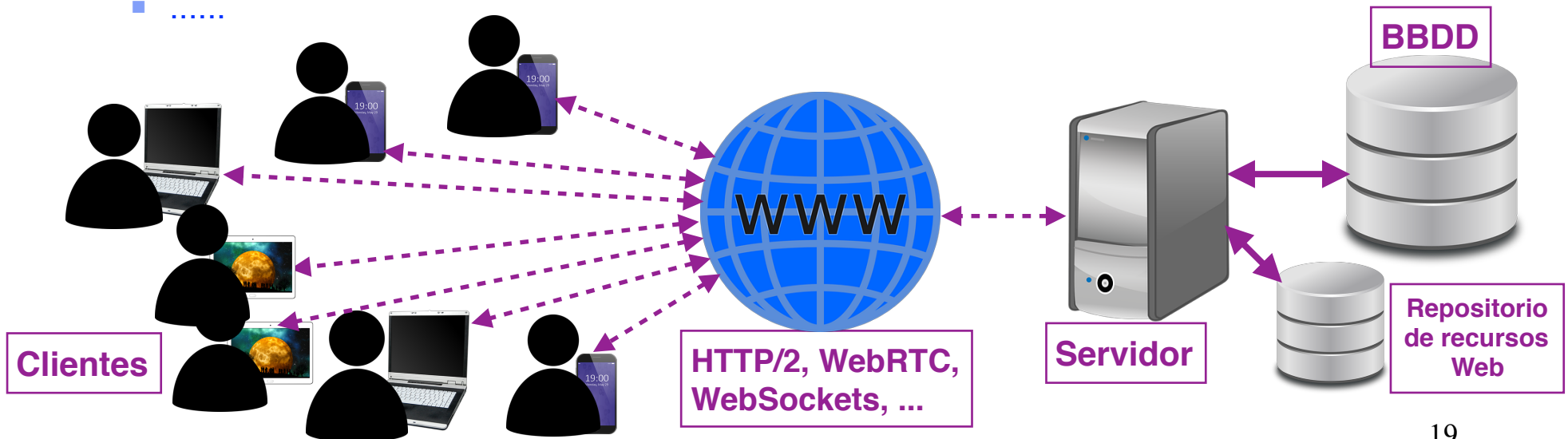
Arquitectura de 3 capas

◆ Los servicios y aplicaciones de Internet suelen tener estas 3 capas

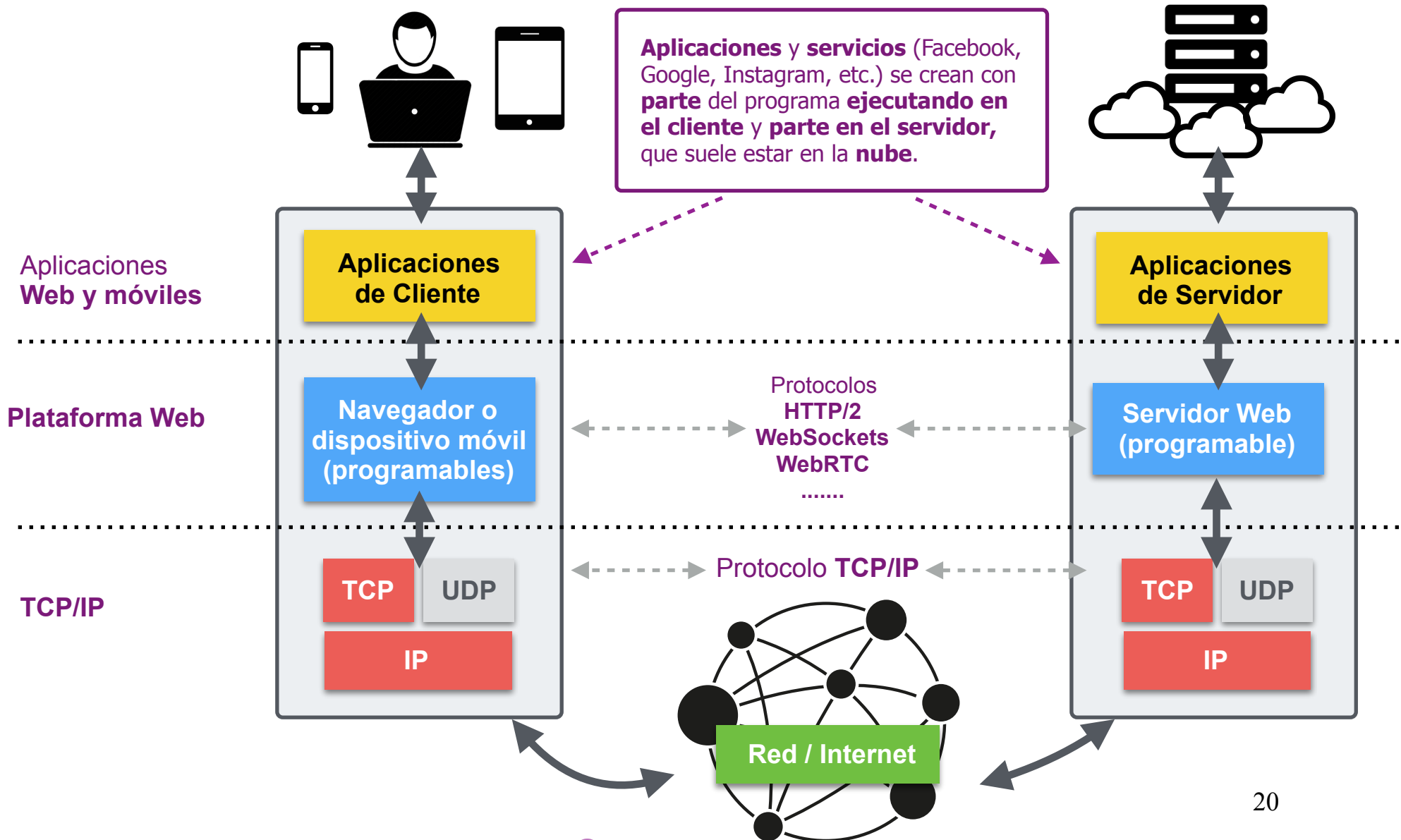
- **Cliente:** Capa de visualización y presentación con el interfaz del servicio
- **Servidor:** Capa lógica de la aplicación con las reglas de atención de peticiones
- **BBDD:** Capa de persistencia que almacena los datos en una base de datos

◆ Cliente y servidor se comunican a través de múltiples protocolos:

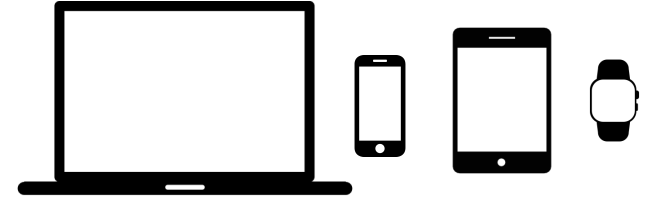
- HTTP/2: Versión 2 (actual) de HTTP es mas eficiente y con menos latencia
- Web Sockets: Para aplicaciones interactivas entre clientes
- WebRTC: Para aplicaciones de voz y video sobre IP
- Server_send events: Para envío de eventos del servidor al cliente
-



Arquitectura de la Plataforma Web



El cliente y sus aplicaciones



◆ **Dispositivos** cliente de acceso a Internet

- PCs, portátiles, tabletas, teléfonos y relojes inteligentes, etc.

◆ **Cliente: programa** que accede a servicios en Internet

- El **navegador (browser)** es el principal cliente de acceso desde un PC
- Las **apps** de los dispositivos móviles son hoy los clientes mas utilizados

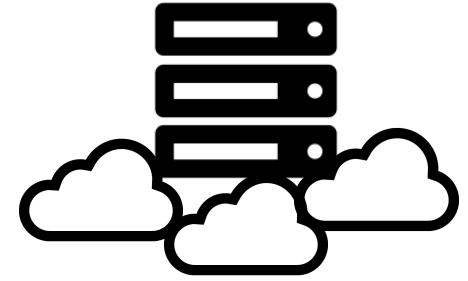
◆ **Navegadores:** Se programan en **HTML, CSS y JavaScript**

- Chrome, Firefox, Internet Explorer, Opera, Safari, ...

◆ **Aplicaciones nativas (apps):** Android, iOS-Apple, etc.

- Se programan en entornos de desarrollo con lenguajes específicos
 - ◆ **Android** se programa en **Java**, **IOS** en **Swift**, etc
- Se programan en **JavaScript** en entornos para aplicaciones **nativas**, por ejemplo
 - ◆ **React Native**, Apache-Cordova/PhoneGap, (reutilizan el código del navegador)

El servidor y sus aplicaciones



◆ Servidor

- **Programa** proveedor de servicios a los clientes
 - ♦ Se conecta a un **puerto** de la **máquina servidora**, el servidor **Web** usa el **puerto 80** por defecto

◆ El programa **servidor** se ejecuta en una **máquina servidora**

- Una máquina servidora tiene una dirección “**conocida**” en Internet
 - ♦ La dirección esta incluida en el URL de acceso: <https://en.wikipedia.org/wiki/URL>
 - Dirección de la máquina servidora: en.wikipedia.org
- La máquinas servidoras pueden ser máquinas físicas o máquinas virtuales en la nube

◆ Servidores Web más usados: Apache, Nginx, Microsoft-IIS, etc.

- Los servidores Web integran aplicaciones en múltiples lenguajes de programación
 - ♦ **node.js** + **JavaScript**
 - ♦ Ruby on Rails
 - ♦ Django + Python
 - ♦ Spring MVC + Java
 - ♦ Zend + PHP
 - ♦ etc



JavaScript

Introducción a JavaScript 6-7-8-9 (ES6 o ES2015, ES7 o ES2016, ..)

Juan Quemada, DIT - UPM

Santiago Pavón, DIT - UPM

JavaScript

- ◆ Lenguaje de programación diseñado en 1995 por Brendan Eich
 - Para animar páginas Web y realizar aplicaciones en el Navegador Netscape
 - ◆ Hoy se ha convertido en **el lenguaje de programación** más utilizado en **Internet**
- ◆ JavaScript tiene pocos elementos, pero muy genéricos y potentes
 - Tiene literales muy expresivos, y funciones, objetos y tipado débil muy potentes
 - ◆ JavaScript tiene además algunas partes mal diseñadas, que se recomienda no utilizar
- ◆ Este **curso** se centra en la partes buenas (**Good parts**) de JavaScript
 - Libro: **JavaScript: The Good Parts**, Douglas Crockford, O'Reilly Media, 2008
- ◆ Existen herramientas para comprobar el buen uso:
 - **jslint**: <http://www.jshint.com/>
 - **eslint**: <http://eslint.org/>
 - ◆ Muchos editores incluyen estas herramientas y avisan cuando no se utilizan las partes buenas
 - Por ejemplo, ATOM, Sublime Text 3, Visual Studio, Webstorm, Eclipse, ..
- ◆ JavaScript ha sido portado a otros entornos
 - Aplicaciones de servidor, de escritorio, sistemas empujados, etc.
 - ◆ Ryan Dahl crea **node.js** en 2009 para crear **aplicaciones de servidor** con gran éxito

ECMAScript

◆ JavaScript sigue la norma ECMA-262

- Publicada por European Computer Manufacturers Association
 - ◆ <https://www.ecma-international.org/publications/standards/Ecma-262.htm>
- El lenguaje ha pasado por múltiples versiones desde que apareció
 - ◆ Las siglas ESversión o ESaño identifican las versiones, por ejemplo ES6 o ES2015 para JavaScript 6

◆ ECMA-262 está en fase de transición de **ES5** a **ES6**, **ES7**, **ES8**, **ES9**

- **ES5** - ECMAScript 5.1, ES2011 o JavaScript 5 (Jun. 2011)
 - ◆ Se desarrolló junto con HTML5 y está soportada por node.js y todos los navegadores actuales
- **ES6** - ECMAScript 6, ES2015 o JavaScript 6 (Jun. 2015)
 - ◆ Incluye muchas mejoras y está soportada por node.js y en fase de transición en los navegadores
- **ES7** (2016), **ES8** (2017), **ES9** (2018), ... se añaden mejoras anualmente

◆ Documentación

- La documentación de JavaScript 6-7-8-9 utilizada para preparar este curso es
 - ◆ **Buenos tutoriales de JavaScript 6:** <https://javascript.info> <https://t.co/nLNMwtgHV5>
 - ◆ **Listados de mejoras:** <http://es6-features.org/>, <https://github.com/lukehoban/es6features#readme>
 - ◆ **Libro online:** Exploring ES6, Axel Rauschmayer, (<http://exploringjs.com/es6/>)
- La documentación más completa de JavaScript es de Mozilla Developer Network (MDN)
 - ◆ **Guía:** <https://developer.mozilla.org/es/docs/Web/JavaScript>
 - ◆ **Guía en inglés** (mas completa): <https://developer.mozilla.org/en/docs/Web/JavaScript>

JavaScript en el Navegador

◆ JavaScript se extiende al navegador con

- Objetos de interacción con el entorno y el usuario
 - ◆ **window**
 - objeto de acceso a los elementos del navegador
 - ◆ **DOM** (Document Object Model)
 - objeto **document** de acceso a elementos la página HTML

◆ Los navegadores ejecutan programas de 2 formas

- Ejecutan **scripts** JavaScript en una página HTML
- Ejecutan sentencias paso a paso en la **consola**

◆ La adaptación de los navegadores a ES6 es lenta

- Todos los navegadores existentes deben adaptarse
 - ◆ Ver tabla de soporte: <https://kangax.github.io/compat-table/es6/>
- Existen librerías que traducen ES6 para el browser
 - ◆ Babel: <https://babeljs.io>
 - ◆ Traceur: <https://github.com/google/traceur-compiler>
 - ◆ etc.

```

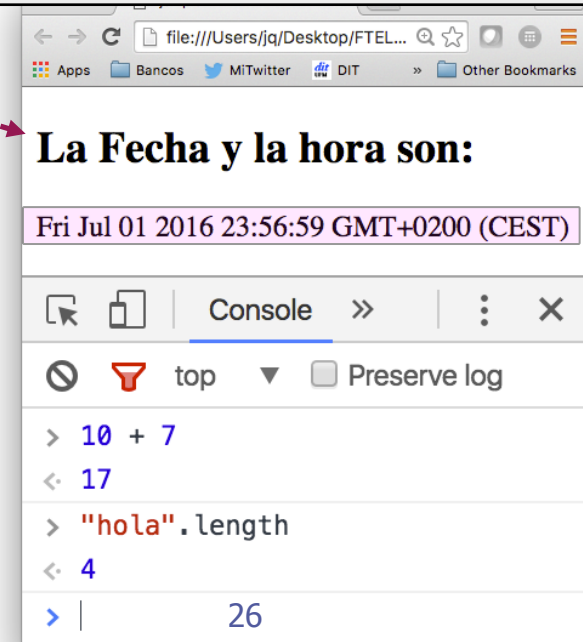
<!DOCTYPE html><html><head>
  <title>Ejemplo</title>
  <meta charset="UTF-8">
</head>

<body>
  <h2> La Fecha y la hora son:</h2>

  <div id="fecha"></div>

  <script type="text/javascript">
    document.getElementById("fecha")
      .innerHTML = new Date();
  </script>
</body>
</html>

```



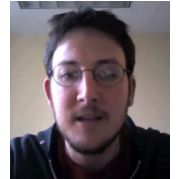
JavaScript en el Servidor: node



◆ Node añade a JavaScript el objeto **global**

■ **global** da acceso a funciones del sistema operativo

- ◆ El proceso donde se ejecuta el programa node
- ◆ El sistema de ficheros con la información
- ◆ Los elementos de entrada/salida y comunicación
- ◆ ...



◆ **node**: comando UNIX y Windows que ejecuta programas JavaScript

- Múltiples portales lo utilizan: E-bay, PayPal, LinkedIn, Netflix, Yahoo, ...

◆ **node** se porta rápidamente a ES6-7-8-9 (<https://nodejs.org/es/docs/es6/>)

- ES6-7-8-.. facilitan mucho la programación de aplicaciones

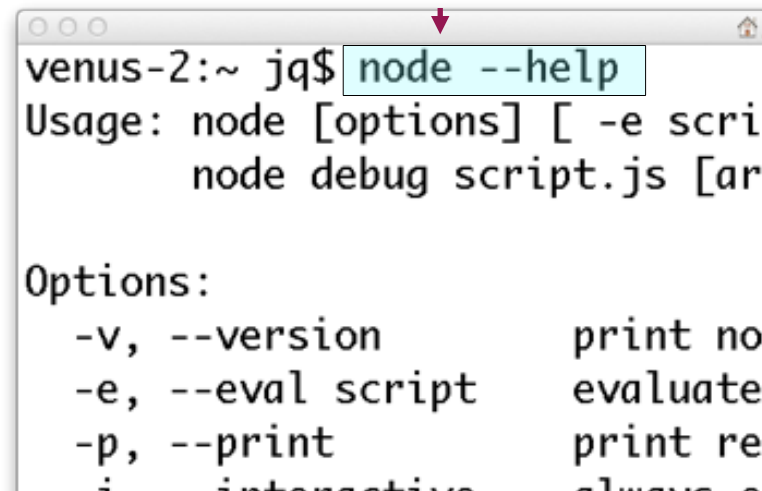
```
var express = require('express');
var path = require('path');

var app = express();

app.use(express.static(path.join(__dirname, 'public')));

app.listen(8000);
```

Ejemplo de servidor
estático de páginas
Web en node.js



```
venus-2:~ jq$ node --help
Usage: node [options] [ -e scri
node debug script.js [ar

Options:
  -v, --version          print no
  -e, --eval script      evaluate
  -p, --print            print re
  -i, --interactive      always
```

IDEs para JavaScript

◆ IDE - Integrated Development Environment

- Facilitan el desarrollo de un proyecto software
 - ◆ Edición y detección de errores sintácticos (eslint, jslint)
 - ◆ Ejecución, depuración y prueba de programas
 - ◆ Gestión de versiones
 - ◆

◆ IDEs mas populares para JavaScript

- Visual Studio
 - ◆ Descarga de versión gratuita
 - :<https://www.visualstudio.com/es/>
- Webstorm
 - ◆ Se puede solicitar una licencia gratis para uso educativo:
 - <https://www.jetbrains.com/buy/classroom/>
 - ◆ O sino la prueba gratuita de un mes:
 - <https://www.jetbrains.com/webstorm/>
- Comparación de IDEs y Editores de JavaScript:
 - ◆ Ver: <https://www.slant.co/topics/1686/~javascript-ides-or-editors>





Final del tema