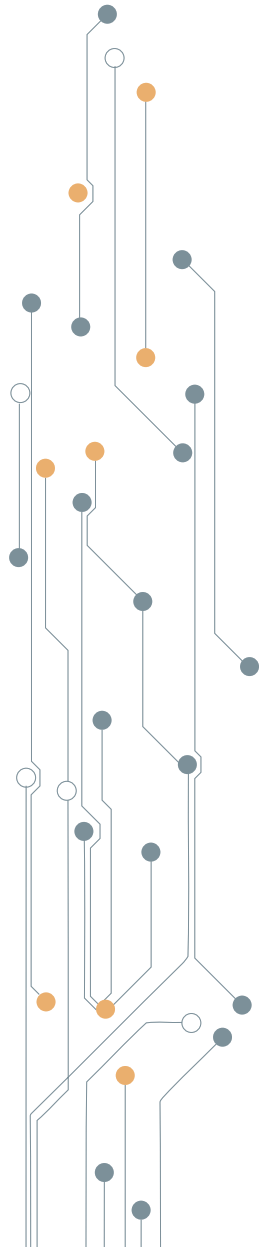
Creación de páginas web dinámicas

Telefónica
EDUCACIÓN DIGITAL

Índice



1. Creación de páginas web dinámicas

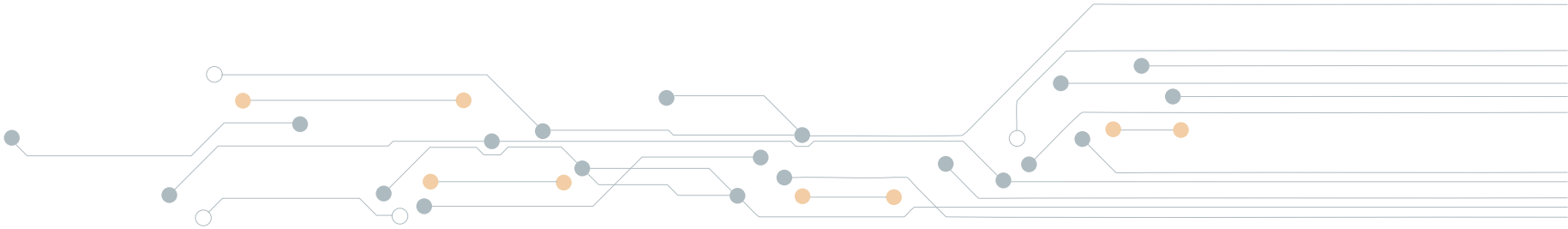
Una página web dinámica es una página que se actualiza conforme el usuario va haciendo peticiones, navegando por la página o actualizando su contenido. Suelen venir cargadas de alto contenido visual, opciones para discapacitados o aprendizaje de las elecciones que ha ido tomando el usuario.

En contra a lo que ocurre con las páginas estáticas, en las que su contenido e información se encuentra predeterminado, en las páginas dinámicas la información va apareciendo según el ciclo de vida del usuario en la aplicación.

Esto se hace posible porque una página dinámica tiene asociada una aplicación web o una Base de Datos desde la que se permite visualizar el contenido.

Para la creación de este tipo de páginas deberán utilizarse etiquetas HTML y algún lenguaje de programación que se ejecute tanto del "lado servidor" como del "lado cliente".

Los lenguajes utilizados para la creación de este tipo de páginas son principalmente: *ASP, PHP, JSP*, pero, sobre todo, mucho *Javascript (JS).*



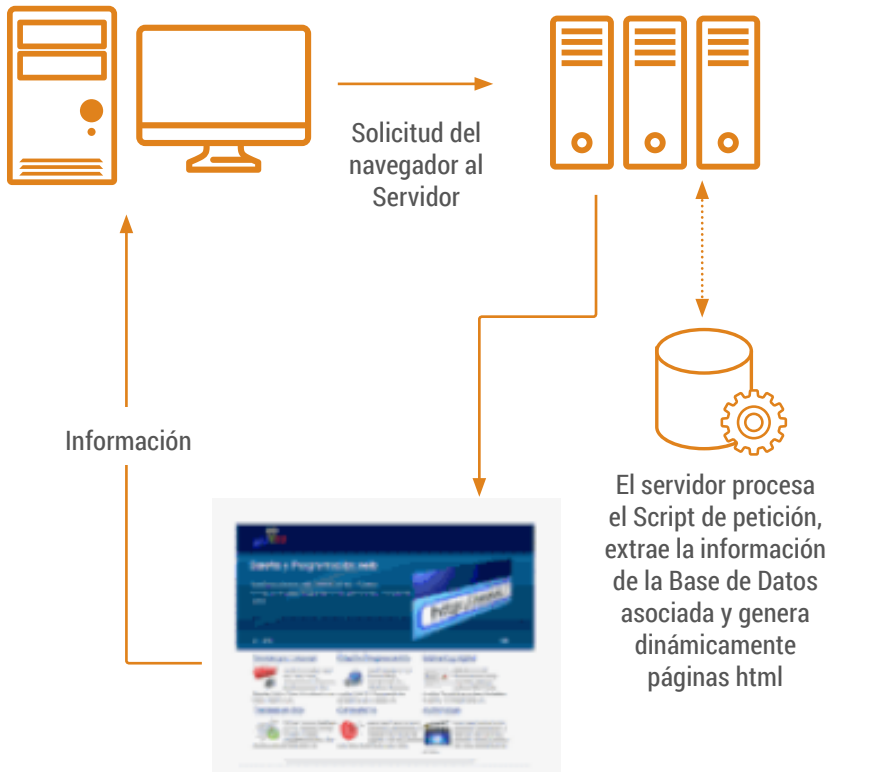
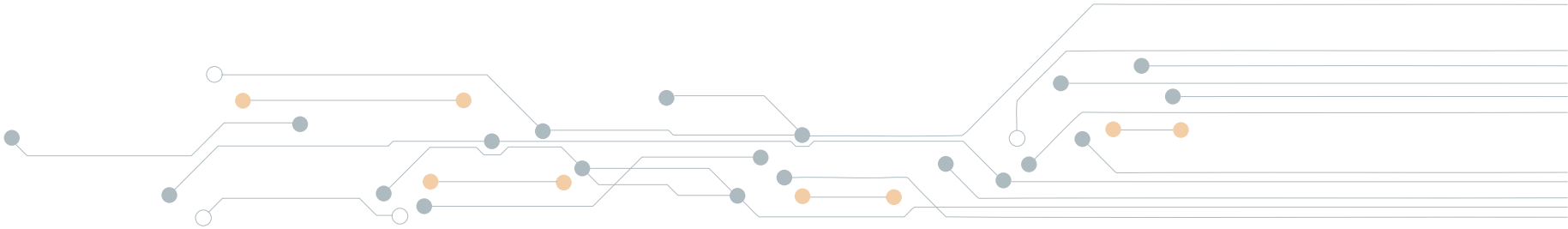


ILUSTRACIÓN 1: ESQUEMA DE FUNCIONAMIENTO DE UNA PÁGINA WEB DINÁMICA

Actualmente, Javascript (JS) ha experimentado un avance sorprendente, debido a la aparición de numerosos frameworks de desarrollo web basados en su lenguaje. En este manual trataremos algunos de ellos, como *JQuery o AngularJS.*

1.1 | Ventajas

- El proceso de actualización y creación es sumamente sencillo, sin necesidad de entrar en el servidor.
- **Gran número de funcionalidades y desarrollos** tales como bases de datos, foros, contenido dinámico, etc.
- Facilitan tener **actualizada diariamente** toda la información.
- Diferentes áreas de diferentes empresas pueden participar en la creación y el mantenimiento.
- **Dominación total sobre la administración** de todos los contenidos.
- **Contenidos fácilmente reutilizables.**
- Una **mayor interactividad** con el usuario.
- **Presentación** de contenidos en **diversos dispositivos y formatos,** como los terminales móviles.
- Los autores del contenido no requieren conocimientos técnicos.



1.2 | Desventajas

- Mayores requerimientos técnicos para su alojamiento en Servidores de pago y, por tanto, costes de alojamiento mayores.
- En algunos casos, un mayor coste de desarrollo que implican mayor cantidad de recursos en el apartado visual de la aplicación.

2. El lenguaje JavaScript

JavaScript, que no debe confundirse con Java, fue creado en 10 días en mayo de 1995 por Brendan Eich, que entonces trabajaba en Netscape y ahora en Mozilla.

JavaScript no siempre fue conocido como JavaScript: el nombre original era Mocha, un nombre elegido por Marc Andreessen, fundador de Netscape.

En septiembre de 1995 el nombre fue cambiado a LiveScript, a continuación, en diciembre del mismo año, al recibir una licencia de marca de Sun, se adoptó el nombre de JavaScript. Esto fue un movimiento de marketing en ese momento, con Java que era muy popular en todo entonces.

En 1996 - 1997 JavaScript fue llevado a ECMA para labrarse una especificación estándar, que otros proveedores de navegadores entonces podrían implementar basado en el trabajo realizado en Netscape. El trabajo realizado en este período de tiempo finalmente llevó a la liberación oficial de *ECMA-262 Ed.1: ECMAScript* es el nombre de la norma oficial, siendo JavaScript la más conocida de las implementaciones. ActionScript 3 es otra aplicación bien conocida de *ECMAScript.*

El proceso de las normas continuó en ciclos, con los lanzamientos de *ECMAScript 2* en 1998 y *ECMAScript 3* en 1999, que es la línea base para el moderno JavaScript. El trabajo dirigido por Waldemar Horwat (entonces de Netscape, ahora en Google) se inició en 2000 y en un primer momento, Microsoft parecía a participar (e incluso implementar) algunas de las propuestas en su idioma JScript.net.

El próximo evento importante fue en 2005, con dos grandes acontecimientos en la historia de JavaScript. En primer lugar, Brendan Eich y Mozilla reincorporaron Ecma como miembro sin fines de lucro y el trabajo comenzó en *E4X, ECMA-357*, que venía de ex empleados de Microsoft en BEA (originalmente adquirido como Crossgain). Esto llevó a trabajar en forma conjunta con *Macromedia*, que estaban implementando E4X en *ActionScript 3.*

Así, junto con Macromedia (posteriormente adquirida por Adobe), el trabajo se reinicia en *ECMAScript 4* con el objetivo de estandarizar lo que había en AS3 y aplicarlo en SpiderMonkey. Con este fin, Adobe lanzó el "AVM2", cuyo nombre en código Tamarin, como un proyecto de código abierto. Pero Tamarin y AS3 eran demasiado diferentes de Web *JavaScript* para converger, como se dieron cuenta las partes en 2007 y 2008.

Por desgracia, todavía había confusión entre los diferentes actores; Doug Crockford - luego a Yahoo! - se unió a Microsoft en 2007 para oponerse a ECMAScript 4, lo que le llevó al 3,1 refuerzo de ECMAScript.

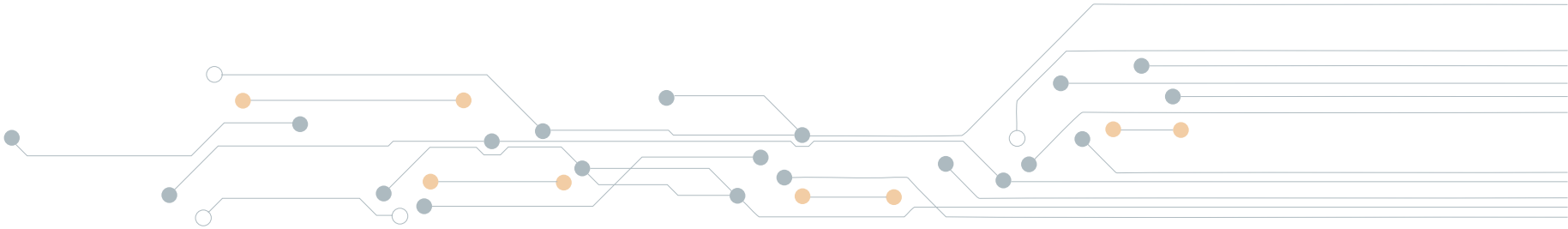
Mientras todo esto sucedía las comunidades de origen y desarrolladores abiertas se pusieron a trabajar para revolucionar lo que podría hacerse con JavaScript. Este esfuerzo de la comunidad se desató en 2005, cuando Jesse James Garrett publicó un libro blanco en el que acuñó el término *Ajax*, y describió un conjunto de tecnologías, de las cuales *JavaScript* era la columna vertebral, que se utiliza para crear aplicaciones web en las que los datos pueden ser cargados en el fondo, evitando la necesidad de cargar la página completa y como resultado: aplicaciones más dinámicas. Esto dio lugar a un periodo de renacimiento del uso de JavaScript encabezada por bibliotecas de código abierto y las comunidades que se formaron a su alrededor, con las bibliotecas como *Prototype, JQuery, Dojo, Mootools* y otros.

En julio de 2008 las partes dispares de ambos lados se reunieron en Oslo. Esto llevó a la eventual acuerdo a principios de 2009 para cambiar el nombre de *ECMAScript 3.1* a *ECMAScript 5* e impulsar el lenguaje.

Todo esto entonces nos trae a la actualidad, con *JavaScript,* que ha entrado en un nuevo y emocionante ciclo de evolución, la innovación y la normalización, con nuevos desarrollos como la plataforma nodejs, que nos permite utilizar JavaScript en el lado del servidor, y APIs de *HTML5* para controlar los medios de comunicación de los usuarios, se abren sockets web para la comunicación, obtener datos sobre las características de ubicación y disposición geográfica, y más.

En el año 2014 y principios del 2015 se considera *HTML5* y el nuevo *ECMAScript 5* como un estándar para el desarrollo de aplicaciones web.

Es un momento emocionante para aprender JavaScript.



2.1 | Pero, ¿qué es javascript?

Como ya hemos comentado en capítulos anteriores una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Básicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario hacer nada con estos programas ni tan siquiera compilarlos. Los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

Más adelante, como ya veremos, JavaScript ha traspasado el ambiente de los navegadores y ha llegado incluso al servidor.

2.2 | Incluir scripts en páginas y archivos .js

La integración entre JavaScript y HTML es muy fácil y variada, ya que hay al menos tres maneras para incluir código JavaScript en las páginas web.

ESCRIBIR JAVASCRIPT EN EL PROPIO DOCUMENTO HTML

Las sentencias JavaScript se encierran entre etiquetas, o tags, <*script*> y se incluyen en cualquier parte del documento.

Aunque es correcto incluir cualquier bloque de sentencias en cualquier zona del documento, se recomienda definir el bloque de código JavaScript dentro de la cabecera del documento (dentro de la etiqueta <*head*>)

```
<!DOCTYPE  html PUBLIC ”-//w3C//DTD XHTML 1.0
Transitional//EN” “http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd”>
<html xmlns= “http://www.w3.org/1999/xhtml”>
<head>
<meta http-equiv= “Content-Type” content= “text/
htmal; charset=iso-8859-1” />
<title>Ejemplo de código JavaScript en el propio
documento</title>
<script type= “text/javascript”>
      Alert(“un mensaje de prueba”);
</script>
</head>

<body>
<p>Un párrafo de texto. </p>
</body>
</html>
```

ILUSTRACIÓN 2: ESCRIBIR JAVASCRIPT EN EL PROPIO DOCUMENTO HTML

Para que la página HTML obtenida sea correcta, es imprescindible agregar el atributo *type* a la etiqueta *<script>*. Los valores a los que se iguala el atributo *type* están estandarizados y para el caso de JavaScript, el valor correcto es *text/javascript.*

Este método se emplea cuando se define un bloque pequeño de sentencias o cuando se quieren incluir códigos específicos en un determinado documento.

La principal desventaja es el no reutilizamiento ya que si se quiere hacer una modificación en el bloque de código, es necesario modificar todas las páginas que incluyen ese mismo bloque de código JavaScript (que habremos copiado en cada una de las respectivas páginas), por lo que no lo hace reutilizable.

DEFINIR JAVASCRIPT EN UN ARCHIVO EXTERNO A NUESTRO DOCUMENTO

Las sentencias *JavaScript* se pueden escribir en un archivo externo de tipo *JavaScript (.js)* que los documentos *HTML* enlazan mediante la etiqueta *(o tag) <script>.*

Se pueden enlazar todos los archivos JavaScript que se necesiten y cada documento HTML puede enlazar tantos ficheros JavaScript como utilice.

Aparte del atributo *type*, este método hace imprescindible definir el atributo *src*, que es el que indica la URL (o dirección) correspondiente al fichero JavaScript que se quiere unir.

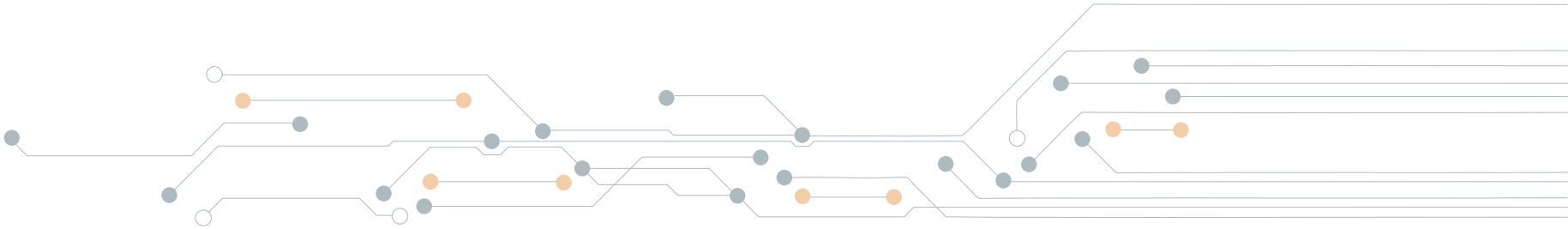
Cada etiqueta <*script*> solamente puede unir un único fichero, pero en un mismo documento se pueden incluir tantas etiquetas (o tags) <*script*> como sean necesarias, lo que hace estos archivos reutilizables.

Los ficheros de tipo JavaScript son archivos normales de texto con la extensión .js, que se pueden crear con cualquier editor de texto.

La mayor ventaja de enlazar un fichero JavaScript externo es que se simplifica el código HTML del documento, que se puede reutilizar el mismo código JavaScript en todos los documentos del sitio web y que cualquier modificación realizada en el fichero JavaScript se ve reflejada inmediatamente en todas las páginas HTML en los que está importado.

```
<!DOCTYPE html PUBLIC “-// W3//DTD XHTML 1.0
Transitional//EN” “http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd”>
<html xmlns- “http://www.w3.org/1999/xhtml”>
<head>
<meta http-equiv=”Content-Type” content=”text/html;
charset=iso-8859-1” />
<title>Ejemplo de código JavaScript en el propio
documento</title>
<script type=”text/javascript” src=”/js/codigo.
js”></script>
</head>
<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

DEFINICIÓN DE JAVASCRIPT EN UN ARCHIVO EXTERNO A NUESTRO DOCUMENTO



INCLUIR JAVASCRIPT DENTRO DE LAS PROPIAS ETIQUETAS HTML

Esta última forma es la menos utilizada, ya que consiste en crear bloques de JavaScript dentro del código HTML del documento.

La mayor desventaja de esta forma es que ensucia innecesariamente el código HTML del documento y complica el mantenimiento del bloque de código en JavaScript.

```
<!DOCTYPE html PUBLIC “-//W3C//DTD XHML 1.0 Transitional//EN” “http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”>
<html xmlns=”http://www.w3.org/1999/xhtml”>
<head>
<meta http-equiv=”Content-Type” content=”text/html;
charset=iso-8859-1” />
<title>Ejemplo de código JavaScript en el propio documento</title>
</head>

<body>
<p onclick=”alert(`Un mensaje de prueba´)”>Un párrafo de texto.</p>
</body>
</html>
```

ILUSTRACIÓN 4: INCLUIR JAVASCRIPT DENTRO DE LAS PROPIAS ETIQUETAS HTML

Telefónica

EDUCACIÓN DIGITAL