

DISEÑO WEB RESPONSIVO

I - Introduccion

Fue en 2007 cuando el mercado tecnológico se revolucionó con el lanzamiento de iPhone, el cual ha marcado un antes y un después, apareciendo el término de smartphones. La competencia no se detuvo y Android ha estado firmemente compitiendo contra Apple, llevando su sistema operativo a smartphones, portátiles, netbooks, tablets, televisiones y otros dispositivos de características técnicas disimiles. Posteriormente Apple lanzó nuevos productos como pueden ser iPod Touch, o unos años después iPad, los cuales tenían la capacidad de navegar por la red como cualquier equipo de escritorio.

Responsive Web Design: Interfaces Web Adaptables al dispositivo empleando HTML5 y CSS3 Algo cambió en aquel entonces y se buscaron soluciones para poder dar soporte a todas estas plataformas y las que posteriormente vendrían. Frente a esto se reaccionó lanzando webs con dos dominios, creando versiones móviles detectando el tipo de dispositivo que se conectaba, se intentaron adaptar webs a través de sistemas de grids, pero la solución final aparece con el término Responsive Web Design. El término Responsive Web Design se acuña en la búsqueda de estas soluciones, concretamente la idea nació en el año 2008 por el consorcio W3C en su recomendación "Mobile Web Best Practices" [4] que definieron el concepto "OneWeb", que se basa en la idea de realizar un diseño web para todos y accesible desde cualquier dispositivo, buscando que la experiencia del usuario se enriquezca.

"One Web" significa proporcionar, en la medida de lo posible, la misma información y servicios disponibles para todos los usuarios, independientemente del dispositivo que estos empleen. Esto no significa que el mismo contenido esté disponible en la misma representación en todos los dispositivos. El contexto del empleo del dispositivo móvil, las distintas variedades de las capacidades del dispositivo o problemas de ancho de banda en la red afectan a la representación. Además de estas razones, ciertos servicios y contenidos son más adecuados en ciertos contextos a usuarios específicos. Sin embargo, "One Web" se opone a la idea de separar la versión móvil de la de escritorio, simplemente debe adaptarse al contexto.

La aparición y establecimiento de HTML5, CSS3 y JavaScript en el mundo web ha dado la posibilidad de que la web móvil consiga adoptar las mismas funcionalidades y obtenga la misma importancia que la web de escritorio. El concepto Responsive Web Design aplicado con estas tecnologías tiene el objetivo de reacomodar los elementos de una web para adaptarlos a la resolución de pantalla del dispositivo, mediante la utilización de grids fluidos, imágenes flexibles, media-queries en la hoja de estilo CSS y posibles modificaciones en la disposición del contenido. Además, con una versión única en HTML y CSS se cubren todas las resoluciones de pantalla, evitando los problemas de usabilidad que presentan los sitios web de ancho fijo

Sin embargo el término Responsive Web Design no depende únicamente de un código como tal, es también usabilidad, conlleva diseñar una experiencia para el usuario donde el contenido es lo más importante y es lo que el diseñador debe tener cuenta, este término quiere aportar una filosofía para afrontar el diseño de una página web. Todas estas son las principales razones de por qué el término Responsive Web Design está ganando popularidad e importancia en nuestros días. Se tratarán de conocer técnicas y buenas prácticas que se deben de tener en cuenta a la hora de implementar una interfaz de usuario que se base en las ideas principales de Responsive Web Design.

Además actualmente se encuentra en auge el término de web semántica, que nos aporta HTML5, dando a nuestra estructura significado semántico, enriqueciendo con metadatos nuestros contenidos, con el fin de jerarquizar y clasificar la información haciendo que éstas sean más fácilmente

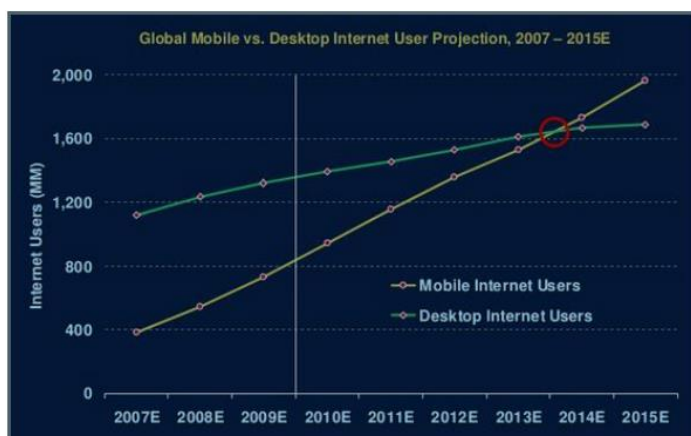
interpretables por los buscadores. Ya no solo cuenta la información o el diseño en sí, sino también conseguir posicionarse aporta puntos positivos para cualquier negocio.

II - Responsive Web Design

A nivel implementación Responsive Web Design tiene tres conceptos claves. El primero de ellos es el uso de los Media Queries que nos ofrece CSS3 permitiéndonos aplicar estilos condicionalmente teniendo en cuenta parámetros de la pantalla. El segundo se trata del diseño web fluido, se trata de layouts definidos en porcentajes que se ajustan a los anchos de la pantalla. Y por último el tercer concepto se trata de los elementos fluidos dentro de estos layouts, como son las fuentes, las imágenes o elementos multimedia. Al crear un sitio con Responsive Web Design solo necesitamos una única versión de HTML y CSS que funcionará adecuadamente en cualquier tipo de dispositivo y resolución. Con Responsive Web Design debemos de dejar de ofuscarnos en que nuestra web se vea idéntica en cada dispositivo.

III - Estadísticas

En el siguiente gráfico obtenido de un artículo de investigación de Morgan Stanley se observa el crecimiento de los usuarios de internet móvil frente a los usuarios de internet de escritorio en los últimos años en una estimación realizada:



Sin embargo, otras estadísticas avalan que este punto de ruptura donde se equilibrará el uso del internet con dispositivos móviles y con dispositivos de escritorio será más tardío, aproximadamente en 2015. Por otro lado otras estimaciones más positivas sitúan este punto más próximo y destacan que a finales de este mismo año será el momento en el que los dispositivos móviles alcancen a los de escritorio.

El concepto de Responsive Web Design se está volviendo cada vez más importante en el desarrollo de páginas web y todas las tendencias marcan que el empleo de dispositivos móviles está aumentando drásticamente. Millones de sitios web pueden estar totalmente seguros que van a recibir cada día mayor tráfico de smartphones y tablets. Lo que está claro es que en este instante el empleo de dispositivos móviles crece mucho más rápido que los dispositivos de escritorio y muy pronto el mundo móvil será el día a día. Por ello debemos prepararnos para este momento.

IV - Aplicaciones Nativas vs Aplicaciones Webs

El desarrollo de una aplicación para móvil requiere de una planificación y distintos procesos de desarrollo de software para finalmente obtener una implementación de la aplicación ideada para un dispositivo móvil. Sin embargo, antes de comenzar el desarrollo se debe de decidir cuál va a ser la forma de implementación de esta aplicación ya que existen distintas opciones, como puede ser

desarrollar una aplicación nativa para una plataforma o desarrollar una aplicación web que corra en un navegador.

Una aplicación nativa es una aplicación desarrollada para una plataforma móvil concreta instalándose directamente en el propio dispositivo. Estas aplicaciones nativas suelen estar subidas en los markets oficiales de la plataforma como puede ser la Apple App Store o el Play Store de Google.

Una *aplicación web*, es una aplicación disponible en Internet que se puede acceder a través de un navegador web de cualquier tipo de dispositivo, ya sea móvil o de escritorio. Este tipo de aplicaciones no necesitan instalarse directamente en el dispositivo.

Por otro lado, en una *aplicación nativa* cada plataforma móvil dispone de su propio entorno de desarrollo, ya que cada uno requiere de un lenguaje de programación distinto, mientras iOS emplea Objective C, Android emplea Java, mientras que las aplicaciones webs emplean únicamente HTML, CSS y Javascript funcionando en cualquier navegador. Esto implica un mayor coste para el desarrollador y la gran desventaja es que la aplicación desarrollada en una de las plataformas no será compatible con otras, debiendo portarse con un coste extra.

Las desventajas de las *aplicaciones web* principalmente se encuentra en la compatibilidad de los navegadores webs, siempre requieren conexión a internet para funcionar, aunque pueden ser instaladas localmente para su empleo online, y además muchos destacan los problemas de privacidad, pues se puede rastrear la actividad del usuario y existen mayores problemas de seguridad.

V – Técnicas

Los Mediaqueries y CSS3

La nueva versión de CSS ha evolucionado el concepto de los mediaqueries. Hasta ahora los mediaqueries habían sido empleados para mostrar la presentación en dispositivos como impresoras. Sin embargo, con esta nueva versión se emplea este concepto un paso más allá para detectar el tipo de dispositivo conectado a la web, consiguiendo la posibilidad de adaptarse al dispositivo concreto a través de distintos factores como pueden ser ancho y alto de la ventana del navegador, ancho y alto del dispositivo, la resolución del dispositivo o la orientación de la pantalla del dispositivo.

A la hora de utilizar los mediaqueries debemos de tener en cuenta el ancho de ventana del navegador y no la resolución de pantalla del dispositivo. Por ejemplo si definimos una mediaquerie como la siguiente:

```
<link rel="stylesheet" media="screen and (min-device-width: 960px)" href="960.css" />
```

El atributo min-device-width tiene en cuenta la resolución del dispositivo a la hora de ejecutar el CSS, esto quiere decir que si reducimos el ancho del navegador, seguirá mostrándose de la misma manera ya que la resolución seguirá manteniéndose, por lo que no se adaptará al nuevo ancho de ventana. Por lo tanto, lo más adecuado es parametrizar estas acciones teniendo en cuenta el ancho de ventana del navegador y no de la resolución del dispositivo.

Por ejemplo tomemos la siguiente mediaquerie:

```
@media screen and (min-width: 960px) {  
/* Código CSS */  
}
```

Esta mediaquerie es una expresión condicional por el cual si se cumplen las condiciones que se especifiquen se aplicará el código CSS que se encuentre en su interior. La primera condición, en este ejemplo screen, indica el tipo de dispositivo que mostrará la salida, por ejemplo, una impresora o una

pantalla y el segundo parámetro nos indica el ancho de ventana, para este ejemplo el ancho de pantalla ha de ser como mínimo 960 pixeles para que se interprete el código.

Además se pueden incorporar varias condiciones, por ejemplo, podemos especificar un máximo y un mínimo para el ancho de pantalla:

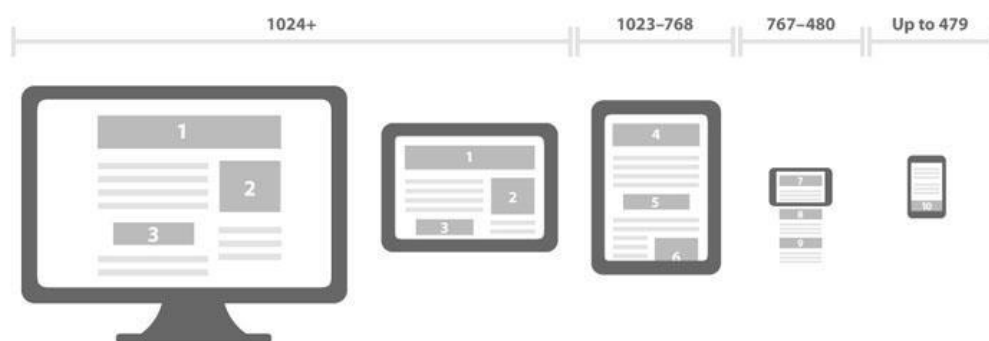
```
@media screen and (min-width: 480px) and (max-width: 960px) { }
```

Para esta mediaqueries se ejecutará el código CSS que tengo un ancho de pantalla entre 480 pixeles y 960 pixeles ambos incluidos.

Además de las características para determinar las resoluciones y anchos de pantalla, podemos determinar otros parámetros, como por ejemplo la orientación del dispositivo, importante en dispositivos móviles:

```
@media screen and (orientation: landscape) { }
```

Una vez que conocemos como crear puntos de ruptura en nuestros diseños, la pregunta que debemos de realizarnos es como determinamos en que puntos concretos debemos de crear un nuevo punto de ruptura.



La forma convencional de determinar estos saltos ha sido teniendo en cuenta anchos determinados de ciertos dispositivos, por ejemplo 320px que es el ancho del iPhone, 768px que es el ancho del iPad y 1024px soportado por la mayoría de los equipos de escritorio. Confiar en unas medidas estándar para definir unos puntos de ruptura por defecto no es realmente adecuado.

Si empezamos a definir puntos de ruptura teniendo en cuenta resoluciones de dispositivo para llegar a un concepto común, corremos un riesgo bastante alto, ya que el mercado de dispositivos está en constante crecimiento, apareciendo nuevos dispositivos con diversas características en lo que a tamaños de pantalla se refiere. Por lo tanto, es una batalla perdida, la mejor aproximación es usar la propia intuición ya que el mismo diseño pedirá cuando establecer el siguiente salto y reorganización de los elementos.

Concepto de Viewport

El viewport se define como la zona visible de un navegador, es decir se limita por el ancho del navegador. Sin embargo, todo cambia al mostrarse en un dispositivo móvil. A pesar de este tipo de pantalla, los navegadores por defecto intentan mostrar el sitio completo con el objetivo de proveer una experiencia web completa, pero esto no es totalmente adecuado.

La mayoría de los navegadores móviles escalan las páginas HTML al ancho del viewport para que se pueda ver completa en la pantalla. Sin embargo, HTML5 nos propone una meta tag para controlar

esta acción. El tag viewport nos permite decir al navegador que emplee el ancho del dispositivo como ancho de viewport y así además se puede desactivar la escala inicial por defecto del navegador. Esta meta tag se debe emplear en las cabeceras html dentro de la etiqueta <head>.

Básicamente gracias a esta función vamos a indicar como debe de comportarse el navegador a la hora de mostrar en su pantalla nuestro diseño web. Un ejemplo de uso de esta etiqueta es el siguiente:

```
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
```

Por defecto los navegadores muestran un ancho determinado de página dependiendo del terminal y el navegador, por ejemplo, en el caso de iPhone el ancho por defecto es de 980 pixeles. Esto tiene un inconveniente, porque además de mostrarse la página web reducida no se aplicarán los mediaqueries definidos. Gracias a esta etiqueta podemos establecer parámetros como el ancho, el alto o el zoom de la pantalla.



Fig. Establecimiento de ViewPort

La meta viewport posee una lista de parámetros clave-valor separadas por comas. Un ejemplo de las distintas propiedades aplicables son las siguientes:

```
<meta name="viewport" content="
width = [pixels | device-width ],
height = [pixels | device-height],
initial-scale = float,
minimum-scale = float,
maximum-scale = float,
user-scalable = [yes | no] ">
```

A continuación se detalla cada una de estas propiedades:

Width

Con la propiedad width podremos indicar el ancho que tomará la página renderizada en el dispositivo. Esta propiedad nos permite indicar un valor estático en pixeles para páginas con contenido fijo, pero para el objetivo de este proyecto podremos indicar el valor device-width, que representa el ancho real del dispositivo, adaptándose la web al viewport y así pudiendo llevar a cabo las media queries.

Height

La propiedad height representa al igual que la propiedad anterior una medida de la pantalla, en concreto indica el alto, aunque esta propiedad no posee demasiada utilidad funciona exactamente igual que la propiedad width, poseyendo el parámetro device-height por el cual se toma el alto de la pantalla del dispositivo.

Initial-Scale

La propiedad initial-scale nos indica el zoom inicial que se está aplicando a la página. Si has visitado alguna página desde un dispositivo móvil sin Responsive Web Design podrás ver como es necesario realizar zoom para poder leer el contenido o acercarte a cierta parte de la web. Gracias a esta propiedad podemos regular la escala de zoom. Este parámetro nos permite indicar el tanto por ciento, por lo que si ponemos un valor de 0,9 se podrá ver inicialmente un zoom del 90% respecto el ancho total.

Maximum/Minimum Scale

Al igual que con la propiedad anterior podríamos indicar el zoom por defecto inicial, con estas dos propiedades podemos indicar cuál será la escala máxima y mínima a la hora de establecer zoom.

User-Scalable

Esta propiedad está directamente relacionada con la anterior, ya que podemos controlar la acción del usuario a la hora de hacer zoom. Con un valor booleano lograremos indicar si deseamos que el usuario manipule el zoom para visualizar nuestra web.

Respecto al tema de la etiqueta viewport la comunidad web considera de malas práctica el empleo de las etiquetar user-scalable y minium-scale, ya que por ejemplo se considera un gran problema de accesibilidad a personas con problemas de visión el impedir no poder moverse en la página con total libertad o limitando el zoom.

Para aplicar responsive design:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

En móviles conviene deshabilitar el zooming:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0,  
maximum-scale=1.0, user-scalable=no">
```

Técnicas para mostrar y ocultar contenido

Una de las técnicas empleadas en Responsive Web Design es la de ocultar o mostrar elementos según el tipo de resolución. Aunque hemos visto técnicas para transformar proporcionalmente los elementos y reorganizarlos, muchos desarrolladores emplean esta técnica para proporcionar una navegación más simple, un contenido más específico o simplemente privar de la funcionalidad a determinado dispositivo. Favorablemente CSS nos permite mostrar u ocultar elementos, pero vamos a determinar que técnica es la más apropiada.

En CSS tenemos la opción display, la cual podemos establecer en none, así el elemento HTML al que se le aplique esta propiedad se verá oculto. Podemos establecerlo tanto de forma predeterminada o asociársela a un elemento dinámicamente a través de JavaScript.

display:none

Esta técnica se emplea tanto para ocultar o mostrar contenido en resoluciones pequeñas como grandes, un ejemplo de ello es cambiar el tipo de navegación ocultando un bloque y haciendo mostrar otro.

Por otro lado, existe otra opción en CSS llamada visibility, que se puede establecer a valor hidden. Sin embargo, esta opción esconde el contenido pero el elemento permanece en la misma posición a diferencia de display:none que oculta totalmente como si no estuviera en el DOM.

Con esta habilidad para manejar la visibilidad de los elementos nos abre las puertas a distintas transformaciones, adaptar nuestros diseños, reorganizar elementos y abrir la mente del diseñador para mostrar al usuario lo que realmente se desea mostrar

VI - Rendimiento

Se recomienda incluir las hojas de estilos CSS y los documentos JS en un único fichero, ya que cada fichero necesitará una nueva petición HTTP por parte del navegador incrementando los tiempos de carga. Asimismo se recomienda que estos ficheros sean externos, y evitar incluir etiquetas CSS o styles o embeber código javascript en el HTML. La razón principal es que estos ficheros se guardan en la caché del navegador, y no es necesario volver a descargarlos, aunque esto puede ser un inconveniente mientras se está desarrollando, borrando repetidas veces la caché para probar el nuevo código.

Además del tema de la caché encapsulando código en ficheros externos, evitamos la repetición de código aligerando el documento html. Respecto al tema de aligerar el peso también es recomendable minimizar u ofuscar el código css y javascript, por ejemplo una herramienta que ayuda a esta tarea es Minify (<http://code.google.com/p/minify/>).

Se recomienda poner los enlaces a las hojas de estilos en el header ya que los navegadores por defecto no cargan la página web hasta haber obtenido las hojas de estilos manteniendo el viewport en blanco. Por otro lado, los JavaScript se recomiendan ponerlos después del body, justamente por lo contrario, para poder visualizar la página cuanto antes correctamente mientras se termina de descargar los scripts correspondientes.

VII - Frameworks para RWD

Concepto de un framework front-end

Un framework para front-end es un conjunto de conceptos y herramientas que nos facilita considerablemente el trabajo de diseñar una web proporcionándonos una base o esqueleto para nuestros nuevos diseños, como por ejemplos dándonos la posibilidad adaptarlo a distintas resoluciones y tamaños de pantalla a través de sistemas de grids u otros mecanismos.

La mayoría de los desarrollos web comparten una estructura similar, por lo tanto el objetivo de estos frameworks es proporcionar una estructura común para desarrolladores web, para agilizar el proceso de inicialización aportando reutilización de elementos básicos y repetibles.

Estos frameworks suelen consistir en una estructura de archivos y directorios de código estándar divididos en elementos html, css y javascript. En general la mayoría de estos frameworks comparten características como proporcionarnos un código css para diseñar nuestros layouts, lo que se conocen como grids o cuadrículas, suelen contener definiciones de tipografía, soluciones para el problema de las incompatibilidades de los distintos navegadores como reset css, y componentes avanzados de interfaces de usuario.

Ventajas y desventajas de emplear un framework front-end

La principal ventaja del uso de estas herramientas es la rapidez que aportan debido a que no es necesario implementar todo de una manera manual y desde cero, este tipo de frameworks nos aporta ciertas funcionalidades ya implementadas proporcionando agilidad sobre todo a la hora de maquetar nuestros diseños. Además esta agilidad será bien recibida por nuestros clientes que siempre exigen el tiempo como mayor valor.

Aportan un nivel de abstracción superior a la hora de decidir factores como los saltos necesarios de mediaqueries para cada dispositivo e incluso algunas herramientas aportan soluciones a los problemas comunes en CSS y de compatibilidad de navegadores.

Por otro lado al trabajar con un framework se pierde en libertad, ya que debemos de ajustar nuestro diseño a un grid preestablecido y a los requerimientos de la herramienta. Sin embargo, creando nuestro propio diseño de forma manual creamos un código más flexible y siempre bajo control aunque haya que invertir mayor tiempo en su desarrollo. Además el empleo de un framework requiere de una curva de aprendizaje para manejarlo.

VIII – Bootstrap

Bootstrap es un conjunto de herramientas proporcionadas por los creadores de Twitter que nos aportan distintos widgets y estilos para desarrollar con gran agilidad el front-end de nuestras aplicaciones web.

Este framework nos abstrae de las compatibilidades entre navegadores poniendo a disposición del desarrollador un conjunto de elementos como pueden ser desde formularios, botones, tablas hasta menús, alertas y otros componentes que agilizan bastante nuestro trabajo.

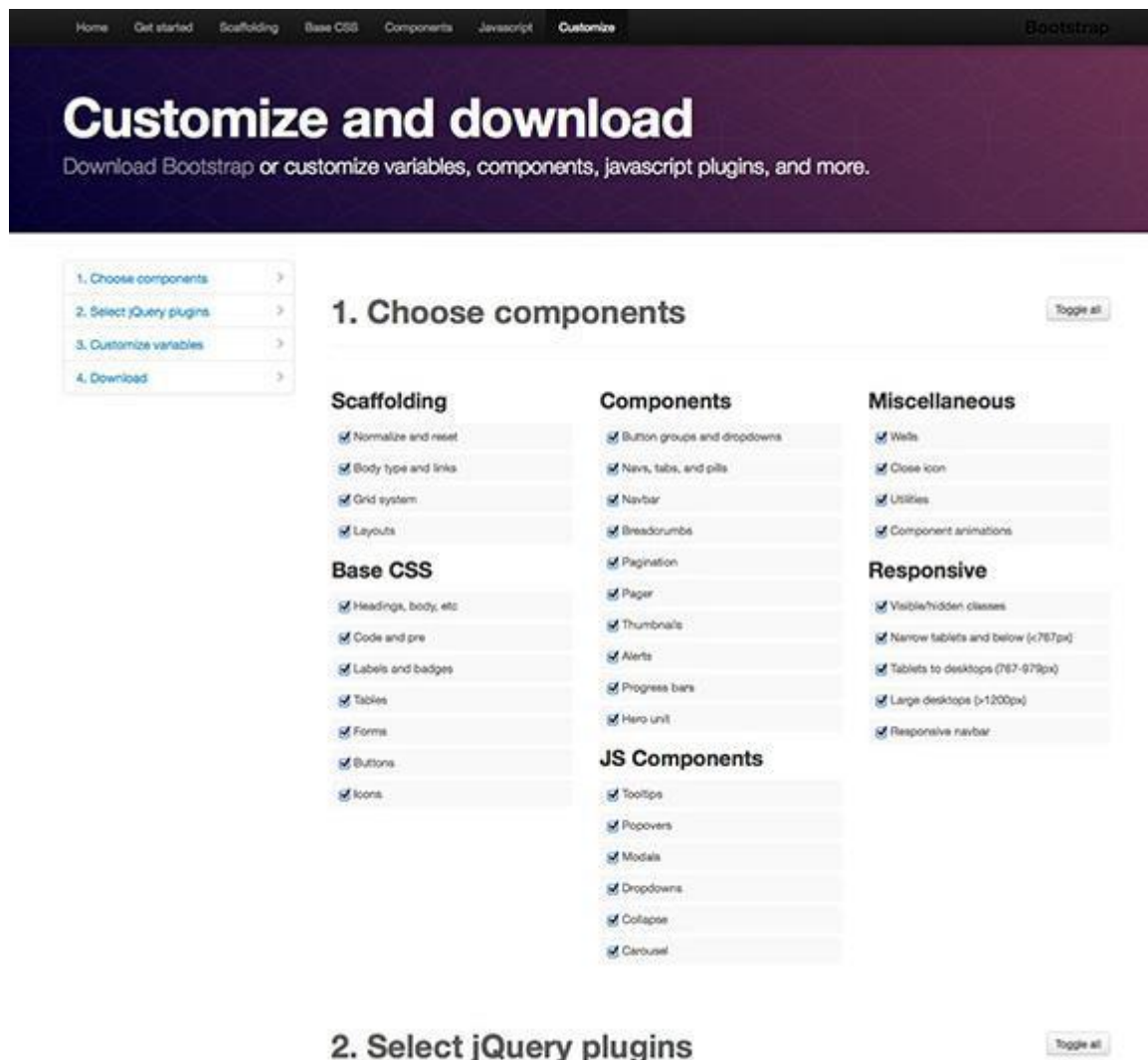
Este kit de herramientas fue un proyecto interno de Twitter que posteriormente decidieron publicar totalmente abierto a toda la comunidad web y se define a sí mismo como un framework front-end limpio, intuitivo y con gran poder para aportar rapidez y facilidad al desarrollo web. Sin embargo, el punto más importante que destaca la mayoría de la comunidad web es lo realmente fácil que es aprender a usar este framework gracias a su buena documentación y a cantidad de ejemplos que aporta.



Bootstrap nos permite descargar su conjunto de herramientas customizadas dependiendo de nuestras necesidades. Concretamente nos permiten tres opciones:

- La primera opción nos permite descargar todo el paquete de herramientas completo, ya compilado y en sus versiones minimizadas, preparado para poder incluirse en tu proyecto.
- La segunda opción también nos permite descargar todo el paquete de herramientas completas, pero en este caso con el código fuente y la documentación completa para poder ser modificada, estudiada y optimizada por los desarrolladores.

- La última opción es descargarnos los componentes del framework que se vayan a emplear con el objetivo de personalizar y optimizar las necesidades requeridas por nuestra aplicación web.

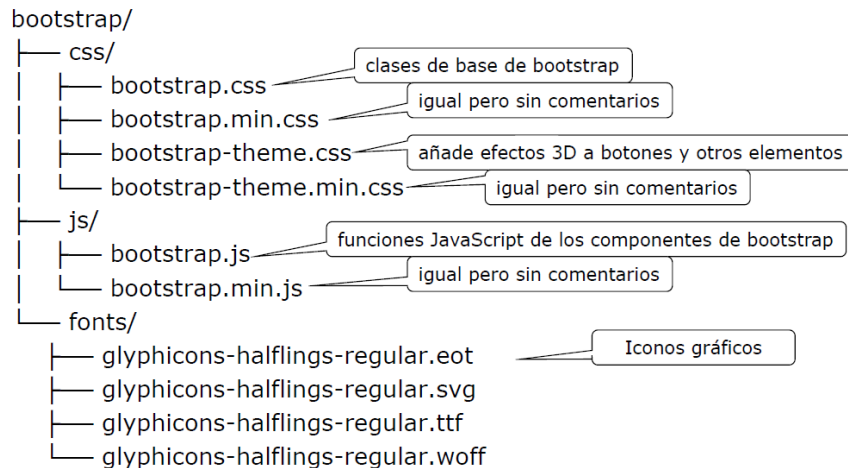


The screenshot shows the Bootstrap Customizer web application. At the top, there's a navigation bar with links: Home, Get started, Scaffolding, Base CSS, Components, Javascript, and Customize. The main heading is "Customize and download" with a subtitle "Download Bootstrap or customize variables, components, javascript plugins, and more." Below this, there's a sidebar with a list of steps: 1. Choose components, 2. Select jQuery plugins, 3. Customize variables, and 4. Download. The main content area is titled "1. Choose components" and contains three columns of checkboxes for selection. The first column, "Scaffolding", includes options like "Normalize and reset", "Body type and links", "Grid system", and "Layouts". The second column, "Base CSS", includes "Headings, body, etc.", "Code and pre", "Labels and badges", "Tables", "Forms", "Buttons", and "Icons". The third column, "Components", includes "Button groups and dropdowns", "Navs, tabs, and pills", "Navbar", "Breadcrumbs", "Pagination", "Pager", "Thumbnails", "Alerts", "Progress bars", and "Hero unit". To the right of these are "Miscellaneous" (Wells, Close icon, Utilities, Component animations) and "Responsive" (Visible/hidden classes, Narrow tablets and below, Tablets to desktops, Large desktops, Responsive navbar). A "JS Components" section at the bottom includes "Tooltips", "Popovers", "Modals", "Dropdowns", "Collapse", and "Carousel". A "Toggle all" button is present in the top right of the main content area.

Estructura del framework

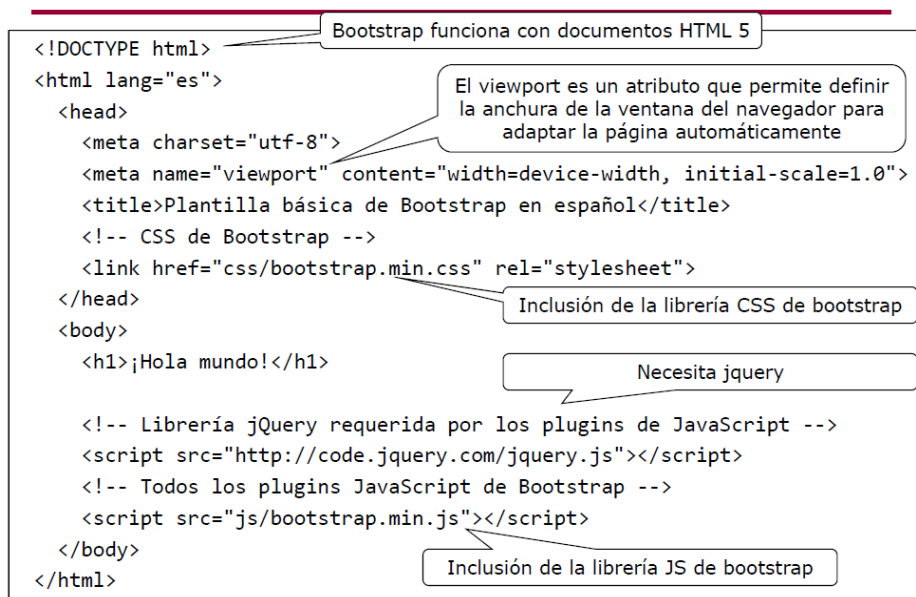
El código fuente descargado está estructurado en tres directorios con una pequeña cantidad de ficheros fácilmente reutilizables e integrables en nuestro proyecto. Concretamente nos encontraremos con los siguientes directorios:

- CSS: Esta carpeta contiene dos ficheros css más sus versiones minimizadas. Los ficheros son bootstrap.css y bootstrap-responsive.css. Estos ficheros se emplean para estilizar los elementos de la web. La versión responsive incluye todos los componentes necesarios para incluirlos en el proyecto.
- JS: Esta carpeta incluye el fichero bootstrap.js además de su versión minimizada donde se encuentra todo el código javascript necesario para el correcto funcionamiento de los widgets de bootstrap.
- FONTS: Esta carpeta incluye los sprites empleados para emplear los iconos de Bootstrap cedidos por Glyphicons.



Incluir Bootstrap en nuestra aplicación web

Una vez conocemos la estructura y los componentes que nos aporta descubramos como podemos incluir el framework en nuestros proyectos web. Básicamente debemos de incluir los archivos descargados (CSS Y JS) y en cada página html incluyéndolos dentro de las etiquetas <head> para el caso de CSS y al final del body para el caso del JS. Veamos un ejemplo



Sistema de columnas de Bootstrap

Bootstrap proporciona un sistema de columnas, también denominada cuadrícula o rejilla, bastante fácil de usar a la hora de diseñar layouts y bastante sólido tanto para layouts fijos como fluidos.

El sistema se basa en un grid de 12 columnas y cubre un ancho de 940 pixels en su versión estática y entre 724 pixels y 1170 pixels en su versión fluida.

Menú	1	2	3	4	5	6	7	8	9	10	11	12
Cabecera												
Columna 1				Contenido principal						Columna 2		

Bootstrap permite que nuestra aplicación responda adaptándose a cualquier dispositivo con un diseño fluido ocupando si es necesario el 100% de amplitud de pantalla o por otro lado que nuestra aplicación sea un diseño estático que no necesite el 100% del ancho del dispositivo.

Bootstrap implementa su layout adaptable mediante el uso de mediaqueries. La documentación nos muestra los siguientes puntos de ruptura en nuestros diseños que se corresponden con la tabla anterior:

/ Large desktop */*

@media (min-width: 1200px) { ... }

/ Portrait tablet to landscape and desktop */*

@media (min-width: 768px) and (max-width: 979px) { ... }

/ Landscape phone to portrait tablet */*

@media (max-width: 767px) { ... }

/ Landscape phones and down */ @media (max-width: 480px) { ... }*

Se pueden ocultar elementos de una página dependiendo del dispositivo con las siguientes clases (*aplicables solo para bloques*)

	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
.visible-xs	Visible	Hidden	Hidden	Hidden
.visible-sm	Hidden	Visible	Hidden	Hidden
.visible-md	Hidden	Hidden	Visible	Hidden
.visible-lg	Hidden	Hidden	Hidden	Visible
.hidden-xs	Hidden	Visible	Visible	Visible
.hidden-sm	Visible	Hidden	Visible	Visible
.hidden-md	Visible	Visible	Hidden	Visible
.hidden-lg	Visible	Visible	Visible	Hidden

Para que el sistema de grids funcione correctamente todo el contenido de la página debe incluirse dentro de un DIV aplicándole la clase CONTAINER que nos sirve un contenedor centrado en el navegador y estableciendo un ancho establecido para el contenido de la página. Dentro de este gran DIV contenedor se deben definir DIVS aplicándoles la clase ROW, que crean columnas dentro del wrapper manteniendo márgenes, padding y clear entre estas filas. Finalmente dentro de estos DIVS se implementa el diseño ocupando el espacio necesario a través de elementos que ocupan un espacio determinado.

```
<div class="container">
  <div class="row">
    <div class="col-md-4">
      <h2>Caja de 4 columnas</h2>
    </div>

    <div class="col-md-6 col-md-offset-2">
      <h2>Offset de 2 columnas y caja de 6 columnas</h2>
      <div class="row">
        <div class="col-md-8"><p>Anidado de 4</p></div>
        <div class="col-md-4"><p>Anidado de 2</p></div>
      </div>
    </div>
  </div>
</div>
```

La suma de columnas anidadas tiene que ser 12

Tipografía

Bootstrap define la tipografía para distintos tipos de texto

h1, h2, h3, h4, h5, h6: Semibold 36px, 30px, 24px, 18px, 14px, 12px

- Con <small> dentro del header se puede añadir texto más pequeño

Código Ejemplo

```
<h1>h1. Cabecera de Bootstrap <small>Texto secundario</small></h1>
```

Formularios en Bootstrap

Los formularios son uno de los elementos más importantes de los sitios y aplicaciones web. Por eso Bootstrap 3 permite diseñar formularios con aspectos muy variados y define decenas de estilos para todos los campos de formulario. Para organizar los elementos dentro de un formulario podemos utilizar las columnas para ajustar tamaños y posiciones.

Conjunto de elementos form-group:

Cada conjunto de elementos dentro de un formulario los agruparemos dentro del siguiente código <div class="form-group"> Aquí estarán nuestros elementos </div> De esta manera tendremos un pequeño margen entre un elemento y otro, más menos así:

Usuario

Contraseña

Sin la clase "form-group" se vería así

Usuario

Contraseña

Clases del formulario:

Tenemos 3 tipos de organizar nuestros formularios, horizontales (form-horizontal),en linea (form-inline) y por defecto (recién visto). Queda a creatividad y necesidad de cada uno como lo utilizarán.

form-inline

Usuario **Contraseña**

form-horizontal

Usuario

Password

Código de Ejemplo

```
<form action="#" class="form-horizontal">
<div class="form-group">
<label class="col-sm-2 control-label">Usuario</label>
<div class="col-sm-10">
<input type="text" class="form-control" placeholder="Usuario">
</div>
</div>
<div class="form-group">
<label class="col-sm-2 control-label">Password</label>
<div class="col-sm-10">
<input type="password" class="form-control" placeholder="password">
</div>
</div>
<div class="form-group">
<div class="col-sm-10 col-sm-offset-2">
```



```
<button type="submit" class="btn btn-default">Entrar</button>
</div>
</div>
</form>
```

* Es importante que utilicemos las clases form-control y control-label para darles un estilo más llamativo a los elementos.

Uso de botones

Bootstrap nos da varias alternativas para utilizar botones, nosotros nos enfocaremos en los tradicionales con y sin iconos.

Colores



Estos colores vienen por defecto en nuestras hojas de estilo de Bootstrap, utilizando las siguientes clases

```
<button class="btn btn-default" type="submit">Blanco</button>
<button class="btn btn-primary" type="submit">Azul</button>
<button class="btn btn-info" type="submit">Celeste</button>
<button class="btn btn-success" type="submit">Verde</button>
<button class="btn btn-warning" type="submit">Amarillo / Naranja</button>
<button class="btn btn-danger" type="submit">Rojo</button>
```

Iconos en botones

Por motivos de rendimiento, todos los iconos requieren de una clase CSS común para todos y de una clase CSS específica para cada uno. Para añadir un icono en cualquier elemento de la página, utiliza el siguiente código como ejemplo. Y no olvides añadir un espacio entre el icono y el texto para que se vea mejor:

```
<span class="glyphicon glyphicon-search"></span>
```



Fig. Galería de Iconos

Fuente Bibliográfica:

- URL http://librosweb.es/libro/bootstrap_3/
- Responsive Web Design – Univ. de Alcalá (Escuela Politécnica Superior).