

1. Font Awesome
2. Google Fonts
3. Animaciones en Css
6. Diseño Responsive

Fuentes

En CSS podemos utilizar infinidad de fuentes o familias tipográficas.

Por defecto, solo podremos acceder a las que ofrecen los navegadores de forma nativa pero por fortuna siempre podemos agregar nuevas opciones.

Serif

Poseen terminaciones con serifa o acabado elegante.

Sans Serif

Tipografías de palo seco con terminaciones rectas o ligeramente redondeadas.

Display

Normalmente utilizadas para títulos por su capacidad de resaltar.

Mono

Cada letra mide lo mismo de ancho, suelen ser las de máquina de escribir o terminal de computadora.

Handwriting

Simulan la escritura hecha a mano.

Propiedad font-family

```
body {  
  font-family: 'Times New Roman', Times, serif;  
}
```

Se divide en varias partes:

El valor entre comillas `' '`, corresponde al **nombre** de la fuente seleccionada.

El segundo valor es el **reemplazo** en caso que no se encuentre esa fuente.

El tercero es la familia tipográfica para que cargue la fuente **por defecto del navegador** para este tipo en caso que las anteriores **no funcionen**.

Importar fuentes externas

URL:

Se importa directamente en nuestra hoja HTML mediante una etiqueta `<link href="" />` a la URL del recurso solicitado (por ejemplo Google Fonts).

```
<link rel="preconnect" href="https://fonts.googleapis.com">  
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>  
<link href="https://fonts.googleapis.com/css2?family=Merriweather+Sans:wght@300;  
400;500&display=swap" rel="stylesheet">
```


También se puede importar sobre la hoja de estilos CSS a través de la propiedad **@import**

```
@import url('https://fonts.googleapis.com/css2?family=Merriweather+Sans:wght@300;400;500&display=swap');
```

Mas Fuentes en: <https://fonts.google.com/>

Importar fuentes externas

Archivo:

Se hace mediante la propiedad @font-face de CSS.

```
1 @font-face {  
2   font-family: <un-nombre-de-fuente-remota>;  
3   src: <origen> [,<origen>]*;  
4   [font-weight: <peso>];  
5   [font-style: <estilo>];  
6 }
```

```
1 @font-face {  
2   font-family: "Bangers", sans-serif;  
3   src: url("../fonts/bangers.ttf");  
4 }
```

Iconos

Son aquellos recursos gráficos que nos ayudan a comunicar o explicar elementos en diferentes lugares de nuestro sitio.

Normalmente son en formato SVG aunque en algunos casos, se importan desde librerías que los ponen a disposición a través de etiquetas HTML.

Algunas de las librerías más conocidas son Font Awesome e Iconify.

Font Awesome

La importamos a nuestro proyecto descargando el código fuente o a través de un gestor de paquetes. Luego, la misma librería nos pone a disposición un stock de íconos que se deben usar mediante una etiqueta `<i></i>` con una clase CSS específica que mostrará el ícono.



```
<span style="font-size: 3em; color: Tomato;">
  <i class="fas fa-camera"></i>
</span>

<span style="font-size: 48px; color: Dodgerblue;">
  <i class="fas fa-camera"></i>
</span>

<span style="font-size: 3rem;">
  <span style="color: Mediumslateblue;">
    <i class="fas fa-camera"></i>
  </span>
</span>
```

Animaciones con CSS

A la hora de trabajar con animaciones se vuelve importante determinar un trigger o disparador, que será la acción que determine el inicio de una animación. Ejemplos comunes de trigger son: pasar el mouse por encima de un elemento, hacer click, hacer scroll, recargar la página, entre muchas otras acciones.

En CSS existen sentencias que nos permiten “capturar” dichas acciones en el navegador o determinadas secciones dentro de un elemento HTML, permitiendo así establecer estilos mediante selectores css especiales en función de su estado o de acciones que ocurren con respecto a los mismos.

A estos selectores especiales se los divide en dos grupos, los cuales son pseudo-clases y pseudo-elementos.

Pseudo-clases:

Harán referencia principalmente a estados o acciones comunes del navegador, así como determinado elemento/s de entre una lista en la estructura HTML, esto nos permitirá identificar a los mismos para establecer reglas CSS que den estilos a elementos en función de esas acciones o eventos. Por ejemplo, que un botón se ilumine al pasar por encima con el mouse, que una slider o switch se desplace al otro lado al hacer click sobre el mismo, etc.

Algunos ejemplos

:visited

:hover

:first-child

Pseudo-elementos:

Harán referencia a partes específicas en relación a la estructura HTML o a secciones dentro de un elemento especificado.

Algunos ejemplos:

::before

::after

::first-letter

En algunas ocasiones necesitamos realizar animaciones en nuestro sitio o proyecto web de una manera rápida y sencilla, para esto les presento a Animate.css, una **librería CSS3 para crear animaciones** fácilmente.

Aprenderemos a trabajar con ella, primero descargamos la librería CSS3 del siguiente link:

<https://animate.style/> , dando click en el enlace “Download Animate.css”.

Una vez descargado, crearemos nuestra estructura básica en HTML y agregaremos un enlace hacia el archivo “animate.min.css” que descargamos anteriormente.

Una vez hecho esto ya podremos mostrar diferentes tipos de animaciones en nuestra página web, para esto, dentro del elemento afectado debemos agregar algunos valores en la propiedad “class” como los siguientes:

- **animated:** Para identificar el elemento a animar.
- **infinite:** Para que la animación siga un bucle infinito.
- **animación (Ej. shake):** La animación como tal, la animación “shake” puede reemplazarse por cualquiera de las más de 70 animaciones que nos ofrece la librería.

Podés ver un ejemplo de animaciones en el **video que se encuentra en el listado multimedia.**

De esta manera hemos visto que con el uso de esta librería podemos crear animaciones sin necesidad de JavaScript o conocimientos de CSS3, lo recomendable es aprender CSS3 y conocer el funcionamiento de estas animaciones por un tema de formación, optimización y para crear nuestras propias animaciones.

Diseño Responsive

Hasta hace algunos años era imprescindible utilizar el ordenador para navegar por internet; ahora en cambio, es muy probable que la mayoría de accesos se realicen desde plataformas mobile o móviles. Hoy en día todos llevamos un smartphone encima y nos comunicamos y buscamos información constantemente, por lo que se ha convertido en algo esencial optimizar los sitios web para un buen uso en estos tipos de dispositivos.

Si no sabes si tu sitio Web está optimizado para móviles, existe una herramienta online para que puedas comprobarlo.

Solo tienes que en ella ([click aquí](#)) e introducir la URL principal de tu sitio Web; en pocos segundos, te indicará si la página tiene un diseño optimizado o no.

De manera general podemos distinguir dos formas de optimización: –Responsive Web Design y –Mobile First Web.

¿En qué se diferencian Responsive Web Design y Mobile First Web?

Aunque van de la mano, hay que diferenciar entre Responsive Web Design o, lo que viene a ser lo mismo, Diseño Web Adaptativo, y Mobile First Web. Como su propio nombre indica, el Diseño Web Adaptativo es aquel capaz de adaptarse a diferentes tamaños y dispositivos, es decir, dependiendo de qué dispositivo sea en el que se cargue, tu sitio web se verá más accesible y fácil de usar. Sin embargo, lo que propone el término Mobile First es empezar a diseñar un sitio web desde la resolución más pequeña para ir creciendo y adaptando el contenido y el diseño a la resolución más grande.

Hasta ahora todos los sitios web han sido diseñados solo para equipos de sobremesa y el proceso de navegar por las webs en los teléfonos móviles era bastante incómodo, debido a una mala experiencia de usuario. Sin embargo, las tecnologías están cambiando y el principio de Mobile First se está convirtiendo en un concepto cada vez más extendido.

Responsive vs. Mobile first

¿Qué conseguimos con el principio Mobile First?

Si empezamos maquetando un sitio web para la versión de escritorio y un usuario se conecta desde un dispositivo, primero cargará todo el contenido utilizado en la primera versión, hasta cargar los recursos necesarios para móvil. Por lo tanto, lo más recomendable es empezar a maquetar para la versión más pequeña, siempre optimizando el contenido que se utilice (hojas de estilos, ficheros, imágenes...), así conseguiremos que el usuario no cargue más recursos de los necesarios, reduciendo el tiempo de carga del sitio web.

La clave es conocer el tamaño, resolución o posibles orientaciones de las pantallas en las que necesitamos mostrar nuestro contenido basándonos en los usuarios que tenemos como objetivo.

Los pilares principales del Responsive son las Media Queries y la etiqueta Viewport.

Media Queries

Las Media Queries son las herramientas fundamentales que se encargan de aplicar diferentes estilos para diferentes dispositivos, y proporcionan la mejor experiencia para cada tipo de usuario que se encuentra navegando en tu sitio web. Nacen de la necesidad de crear breakpoints o puntos de ruptura en la hoja de estilos CSS que tengas predefinida.

Permite que tu sitio Web sea manejable desde diferentes dispositivos.

Breakpoints

Si no te ha quedado muy claro, las Media Queries son un módulo de CSS que sirve para detectar el tipo de dispositivo por el que se está navegando; de esa manera el contenido consigue adaptarse al dispositivo concreto a través de las distintas condiciones que tú mismo asignas, 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. Son declaraciones lógicas que actúan dependiendo de las condiciones específicas que tú mismo declaras en la hoja de estilos. Si la premisa se cumple, se aplicaran los estilos definidos; si no, los omitirá por completo.

Hay dos formas de implementarlas:

La primera opción para poner en funcionamiento las Media Queries es a través del atributo media de la etiqueta <link>. Como sabemos, esta etiqueta es la que se usa para enlazar una hoja de estilo con un documento HTML. En ese enlace podemos especificar condiciones que deben cumplirse para que los estilos enlazados se apliquen. Debería ir dentro del

<head> de nuestro HTML.

Recuerda que la etiqueta <link> tiene esta forma:

```
<link rel="stylesheet" href="estilos.css">
```

Pues ahora simplemente agregamos el atributo media indicando la condición que se debe cumplir para que estos estilos se apliquen:

```
<link rel="stylesheet" media="only screen and (max-width: 768px)" href="estilos.css">
```

Lo que concretamente le estamos indicando es que cargue la hoja de estilos —estilos.css si se cumple que el dispositivo de salida es una pantalla, no una impresora u otro dispositivo (only screen), y si la anchura de la ventana del navegador tiene de máximo 768 píxeles (max-width: 768px).

Si se cumplen las condiciones, los estilos se mostraran correctamente, en caso contrario, los estilos se omiten por completo, y el contenido se muestra sin estilos definidos:

Carga de hojas de estilo

Cargar de esta manera las Media Queries supone un problema, y es que cada vez que queramos cargar diferentes estilos dependiendo de ciertas condiciones que queramos aplicar para distintos dispositivos, tendríamos que cargar una hoja de estilos nueva. Esto conlleva una carga más lenta de tu sitio web, ya que se hacen solicitudes HTTP adicionales, que se podrían evitar.

Hay otro sistema más recomendable para aplicar las Media Queries: basta con incluir todas las condiciones necesarias dentro de un único archivo CSS. Así, incorporaríamos la construcción `@media` seguido de las condiciones que queremos definir para cada tipo de dispositivo y donde se apliquen entre llaves `{ }` los estilos concretos para cada uno de ellos. Es la manera más aconsejable, ya que la carga es de un único archivo CSS.

La forma de incluir Media Queries dentro de la hoja de estilos CSS es la siguiente: `@media`

```
(max-width:320px){
```

```
<!-- Aquí van todos los estilos CSS -->
}
```

Esta Media Query se ejecutará sólo cuando la anchura de la ventana del navegador sea menor de 320 píxeles.

```
@media (min-width:768px){
```

```
<!-- Aquí van todos los estilos CSS -->
}
```

Esta Media Query se ejecutará sólo cuando la anchura de la ventana del navegador sea mayor de 768 píxeles.

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 (orientation: landscape) {  
  <!-- Aquí van todos los estilos CSS -->  
}
```

Portrait: orientación vertical Landscape:
orientación horizontal Portrait vs. Landscape

Operadores lógicos para las Media Queries

También se pueden combinar más de una condición en la misma Media Query para concretar todavía más un rango de resolución, mediante los operadores lógicos:

Operador and: las dos condiciones deben cumplirse para que se apliquen los estilos. Operador not: es una negación de una condición. Cuando esta condición no se cumpla, se aplicarán las media queries definidas.

Operador only: se aplican las reglas solo en el caso de que se cumpla.

Operador or: se pueden poner varias condiciones separadas por comas y en el momento que se cumpla cualquiera de ellas, se aplicarán los estilos.

```
@media only screen and (min-width:320px) and (max-width:480px){  
  <!-- Aquí van todos los estilos CSS -->  
}
```

Para esta Media Querie se mostrarán los estilos CSS cuando la anchura de la ventana del navegador sea entre 320 píxeles y 480 píxeles, ambos incluidos.

Estos son algunos de los parámetros generales que se pueden emplear a la hora de construir las condiciones en las Media Queries:

width: anchura de la ventana del navegador. height: altura de la ventana del navegador.

device-width: anchura de la resolución de pantalla. device-height: altura de la resolución de pantalla.

orientation (portrait/landscape): dispositivo en horizontal o en vertical. resolution: densidad de píxeles.

Excepto la orientación, el resto de parámetros admiten los valores `—maxll` y `—minll`.

max-width: La anchura será menor que la indicada.

min-width: La anchura será mayor que la indicada.

Algo a tener en cuenta a la hora de utilizar las Media Queries, es diferenciar entre el ancho de ventana del navegador y la resolución de la pantalla del dispositivo, es decir:

```
@media only screen and (min-device-width: 960px){  
  /* Aquí van todos los estilos CSS */  
}
```

En esta Media Querie que hemos definido, el atributo `min-device-width` se refiere a la resolución de la pantalla del dispositivo a la hora de cargar la hoja de estilos definida.

atributo min-device-width

Esto quiere decir que si reducimos el ancho del navegador, seguirá mostrándose de la misma manera, ya que la resolución de la pantalla seguirá siendo la misma y no se adaptará al nuevo ancho de la ventana del navegador (es decir, si la pantalla de nuestro móvil tiene 450 px y el navegador detecta que lo óptimo sería visualizarla con 600 px así lo hará si no usamos la meta-etiqueta Viewport).

En caso de usar los atributos para la resolución de la pantalla, la etiqueta Viewport es necesaria.

¿Qué hace la meta-etiqueta Viewport?

El Viewport es el área visual donde se plasma el contenido del documento HTML de tu sitio web. Se podría traducir como vista o ventana y nos sirve para definir qué área de pantalla está disponible al renderizar un documento, la escala/zoom que debe mostrar inicialmente. Todo ello, con parámetros que le damos a la propia etiqueta meta, separados por comas en caso de utilizar más de uno.

Prácticamente todos los navegadores de smartphones al entrar a un sitio analizan el tamaño total y lo escalan para que se muestre completo en la pantalla, este procedimiento genera muchas veces resultados inapropiados.

Por ejemplo, esta imagen mide 320 píxeles al igual que la pantalla del dispositivo, ahora bien, la imagen aparece con un tamaño inferior a causa del efecto de la escala automática.

Sin etiqueta Viewport

La escala automática se puede evitar y controlar muy fácil con el uso de este atributo Viewport: es un atributo del tag <meta> que debe incluirse entre las etiquetas <head> del documento HTML de tu sitio web:

```
<meta name="viewport" content="width=device-width"/>
```

Con solo agregar estas líneas de código, la imagen se adaptará al dispositivo:

Con etiqueta Viewport

Es posible definir un tamaño específico para el área visible del documento; muchos sitios web ajustan directamente el Viewport a 320 px para ajustar la apariencia al display vertical de un smartphone, usando un código similar a este:

```
<meta name="viewport" content="width=320"/>
```

Pero, con los diferentes equipos, dispositivos y tamaños de pantalla, definir un tamaño específico puede ser una mala práctica que puede mostrar resultados erróneos en algunos equipos o cuando el dispositivo cambia de posición. Afortunadamente podemos configurar el viewport para ajustarse dinámicamente al tamaño de cada dispositivo usando el atributo

–device-width, que es equivalente al 100% del ancho de la pantalla del dispositivo, independiente de su tamaño, posición o resolución:

```
<meta name="viewport" content="width=device-width"/>
```

El alto de la pantalla también es configurable con las mismas propiedades a través del atributo –height, aunque –salvo condiciones muy específicas– no es necesario definirlo. Esta propiedad se asignará automáticamente a través del scroll.

También podemos controlar la escala de la vista con el atributo –initial-scale–. El sitio se mostrará al doble de su tamaño original:


```
<meta name=||viewport|| content=||width=device-width; initial-scale=2—/>
```

Atributo “initial-scale”

Es posible además, limitar el tamaño al que se puede escalar el sitio con el atributo

—maximum-scale||. El siguiente ejemplo muestra el documento en escala correcta y permite ampliar (zoom) hasta al doble de su tamaño.

```
<meta name="viewport" content="width=device-width, maximum-scale=2"/>
```

Por ultimo, está el atributo —user-scalable||, que controla los permisos de reducir/ampliar el documento. Con el siguiente código, el sitio se muestra en su escala original y no es posible cambiar el tamaño desde el dispositivo móvil (**importante mencionar que no se recomienda deshabilitar la opción de escalar el contenido**).

```
<meta name="viewport" content="width=device-width, user-scalable=no"/>
```

En general, el atributo viewport permite muchas configuraciones, pero para asegurar compatibilidad con la mayor cantidad de pantallas y navegadores móviles, se recomienda utilizar este formato como base:

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

Web optimizada para los diferentes anchos de pantalla

Para conseguir que nuestro sitio web se adapte a los diferentes anchos de pantalla, estos parámetros serán muy útiles:

Lo primero, y lo más importante es dejar de usar píxeles y usar **porcentajes** a la hora de tomar medidas (por ejemplo: width: 60%).

Que el ancho de la página sea 100% no significa que queramos que la pantalla este en una alta resolución, sino que, si queremos limitar el ancho/alto junto al máximo/mínimo del contenido, debemos usar los diferentes parámetros apropiados para ello (max-width o si quisiésemos establecer un alto máximo max-height; para establecer el mínimo sería min-width y min-height).

Las posiciones absolutas o fijas no son recomendables usarlas para posicionar contenido (menos cuando hagan falta). Lo mejor es utilizar el atributo **float** (float:left/right), es una técnica muy usada.

Hay que hacer que las imágenes y vídeos no sobresalgan de la estructura; si no, aparecerá un scroll lateral en los dispositivos móviles que descolocará totalmente el diseño.

En resumen, ¿cuál es la mejor opción para tu sitio web? La **experiencia del usuario** siempre será lo primero.

A los usuarios no les importa que versión utilices ni como estés optimizando tu sitio web; su objetivo es poder encontrar lo que buscan de manera más eficiente y rápida. Por tanto, si tu web está más centrada en el contenido, es mejor una Responsive Web Design. Pero si necesitas que el usuario interactúe mucho con la web, es mejor una **Mobile First**, ya que **cada vez se consume más información desde los dispositivos móviles**.

La conclusión es sencilla, **los clientes están yendo más rápido que las propias empresas** y estas deben adaptarse a ellos y a sus nuevas costumbres de consumo online a través de dispositivos. Esto es una solución para reducir la tasa de rebote, haciendo que el usuario pase más tiempo en la página por su **facilidad, comodidad**, y óptima visualización y lectura de los contenidos.

The background of the entire page is a light gray with a subtle pattern of binary code (0s and 1s). In the top left corner, there is a blue parallelogram containing the text '<codoa codo/>' in white and yellow. The text is stylized, with the first part in white and the second part in yellow. The rest of the page is filled with a repeating pattern of this text in a light yellow color. In the top right corner, there are several abstract shapes: a blue line, a yellow dot, a blue dot, an orange line, a blue line, and a yellow dot. In the bottom left corner, there are several abstract shapes: a yellow line, a purple circle, a blue line, an orange line, a blue line, and a yellow dot. In the bottom right corner, there is a logo for 'Agencia de Aprendizaje a lo largo de la vida' which consists of three stacked horizontal bars in purple, orange, and blue, with the text 'Agencia de Aprendizaje a lo largo de la vida' in white. The text is arranged in four lines: 'Agencia de', 'Aprendizaje', 'a lo largo', and 'de la vida'.

<codoa codo/>

Es de destacar que actualmente Google valora todas aquellas páginas web que se adaptan perfectamente a cualquier dispositivo, ya sea PC, smartphone, tablet... por ello, es necesario optimizar un sitio web de modo que cualquier usuario pueda visualizar la página sin importar el medio por el cual acceda.

Agencia de
Aprendizaje
a lo largo
de la vida