



Diseño Web Adaptativo (Flexbox)

Índice de contenidos del curso

1. [Fundamentos de RWD](#)
2. [Introducción a flexbox](#)
3. [Contenedor flex](#)
4. [Elementos flex](#)
5. [Elementos multimedia](#)
6. [Unidades de medida relativas](#)
7. [Texto en diseño adaptativo](#)
8. [Diseños Fluidos](#)
9. [Consultas de medios \(Media Queries\)](#)



Fundamentos de RWD

Hoy en día podemos decir que el Responsive Web Design responde, en líneas generales, al "Diseño web". Y es que lo importante de una web es el contenido y que los usuarios puedan acceder al mismo, sea como sea y desde donde sea.

Hablamos de diseño responsive o diseño adaptable y pensamos únicamente en las resoluciones de pantalla. Es decir, en sí vamos a acceder al sitio web desde el ordenador de mesa, desde una tablet o desde un móvil. Pero hay que tener en cuenta muchos otros factores. Estos factores que nos van a influir son los navegadores, el sistema operativo desde el que accedemos, la resolución del dispositivo...

Entendemos que **un diseño es Responsive** cuando vemos que **al agrandar o empequeñecer la ventana del navegador, el contenido se adapta** para que podamos seguir viendo, en todo momento, todo el contenido de manera correcta. Para ello, los bloques o grupos de contenidos que tenemos en pantalla se re-colocan si hace falta.

Cronología:

- 2015 ● Google penaliza a las páginas web que carecen de un diseño adaptativo
- 2013 ● Se generaliza su uso dada la necesidad de brindar una experiencia óptima a quienes se conectaban desde teléfonos móviles.
- 2010 ● Surgió cuando el diseñador gráfico Ethan Marcotte lo usa para construir una página HTML usando lenguaje de programación CCS.




Diseño adaptable

Al final se trata de hacer que nuestro contenido sea accesible.

Pensemos que, cuantos más usuarios puedan acceder, de manera sencilla y cómoda, a nuestros contenidos, más posibilidades tenemos de “*vender nuestros productos o servicios*”.

Al comienzo, se diseñaba para resoluciones de pantalla pequeñas, resoluciones de 800x600.

A medida que ha avanzado el tiempo, y han ido saliendo monitores de mayor resolución, hemos ido adaptando nuestros diseños y hemos trabajado para que nuestros sitios webs se adapten a esas nuevas resoluciones.



Llegaron los teléfonos inteligentes y los smartphones y tuvimos que volver a re-adaptarnos, creando versiones de las webs optimizadas para ciertas resoluciones.

Pero finalmente nos hemos dado cuenta de que realmente hay que ser "future friendly".

Es decir, tenemos que diseñar para cualquier tipo de sistemas, y procurando que nuestro proyecto tenga en cuenta cualquier tipo de resolución y circunstancia, de tal manera que no tengamos que modificar nada en caso de que aparezca una nueva resolución de pantalla, o tipo de dispositivo.



Qué caracteriza al RWD

- Se adapta al ancho del dispositivo, ya que puede pasar de horizontal a vertical y viceversa;
- Ofrece contenidos e imágenes con fluidez;
- Reorganiza los elementos de la web;
- Ajusta el tamaño y separación de las letras para hacer la página más legible;
- Reduce la carga visual y el tiempo de carga web;
- Usa el código media-queries de CSS3 para su programación;
- Evita los contenidos duplicados;
- Hace posible que algunos elementos modifiquen su apariencia y se eliminen otros;
- Permite compartir los contenidos de forma más rápida.



La importancia del RWD

Ofrecer una navegación excelente es indispensable, y esto lo puede garantizar una página responsiva. Esto va a favorecer que demos una buena experiencia de usuario, lo cual repercutirá, de manera favorable, en la reputación online de nuestro sitio web.

La forma en que los usuarios perciben el contenido de un sitio depende de su diseño: puedes tener los mejores temas e imágenes, pero si tu sitio web no es accesible / usable, no te va a servir de nada.

Todo esto se conoce como UX o Experiencia de Usuario.



Ventajas del RWD


Favorece el posicionamiento SEO

Desde 2015, Google penaliza a las páginas web que carecen de un diseño adaptativo. Lo propio hicieron Bing y otros motores de búsqueda.

Reduce los costos de mantenimiento

No tenemos varias webs en función de a dónde están destinadas, sino que se trata de una sola web con un diseño que se adapta.

Por lo tanto, el mantenimiento es más simple y económico. La razón es que en el momento de ejecutar cambios, estos se hacen en una sola URL y se requiere de una plantilla única para todas las plataformas.



Esta particularidad también evita los contenidos duplicados. Tenemos que tener en cuenta que Google rechaza los contenidos idénticos o duplicados, así que cuando necesites nutrir el sitio esto tampoco será un problema.

Aumenta la conversión

Hace algunos años, la experiencia del usuario no era tomada en cuenta, pero hoy sí. Estamos en un momento donde el público lo es todo. Está comprobando que, mientras mejor se sienten las personas usando un sitio web, mayores serán las posibilidades de que se conviertan en clientes.

Optimiza la velocidad de carga

No importa el equipo desde donde se navegue en una página web, el diseño hecho para adaptarse a ellos cargará más rápido. Hoy la inmediatez es una premisa, porque no solo es valorada por los usuarios, sino también por Google.



Ofrece una mejor imagen de marca

Dar una buena impresión a los usuarios, una imagen profesional, un diseño bien estructurado, es indispensable en un mundo digital donde la competencia es voraz. Como herramienta de programación, un diseño responsivo ofrece esto y más.

Es uno de los pasos para lograr el posicionamiento de la marca y hasta la fidelidad de los clientes, uno de los objetivos más difíciles de alcanzar en el marketing.

Elementos a tener en cuenta en el RWD

- **Las tipografías:** deben ser legibles pero, lo más importante es que tengan un tamaño de letra distinto en función del tamaño de la pantalla.
- **Los elementos multimedia:** imágenes, vídeos y todo tipo de gráficos, deben tener una proporción lógica con el dispositivo en el que se muestran.
- **La estructura visual de la web:** las filas, columnas, sliders, etc.
- **Los elementos de interacción,** es decir los botones de CTA, listados, menús, formularios, etc.
- **La orientación** horizontal o vertical; la web se debe visualizar correctamente en ambas, aunque el formato vertical es el predominante.
- **Los tiempos de carga:** un factor que depende en gran parte del alojamiento web que tengamos.
- **Los efectos:** Algunos efectos visuales pueden no visualizarse bien en algún dispositivo, por lo que estaremos corriendo un riesgo innecesario. Un ejemplo es el denominado efecto 'hover' al pasar el cursor por algunos elementos.



Como comprobar que nuestro diseño es RWD

Podemos hacer la comprobación con las siguientes herramientas:

- Inspeccionar elementos con Chrome, Firefox o Edge
- Instalar extensión Responsive Web Design Tester
- Instalar extensión de Chrome Windows Resizer
- Herramienta Adobe Edge Reflow: una herramienta muy eficaz para comprobar la adaptabilidad de la web

Introducción a Flexbox

En CSS, inicialmente se utilizaba el posicionamiento (*static, relative, absolute...*), los elementos en línea o en bloque (*display: inline ; display: block, y sus derivados*) o la propiedad float para realizar maquetaciones, lo que a grandes rasgos no dejaba de ser un sistema de creación de diseños bastante tosco que no encajaba con **los retos que tenemos en la actualidad: sistemas de escritorio, dispositivos móviles, múltiples resoluciones, etc...**

Flex (*también llamado flexbox*) es un sistema de elementos flexibles que viene a sustituir esa forma en la que se trabajaba hasta ahora y que **nos permite adaptar y colocar automáticamente los elementos HTML**. Esta forma de trabajar nos facilita personalizar los diseños de una página web.

Flexbox es un modelo de distribución pensado para contenido unidimensional. Se destaca por tomar varios elementos que tienen diferentes tamaños y devolver la mejor distribución para estos.

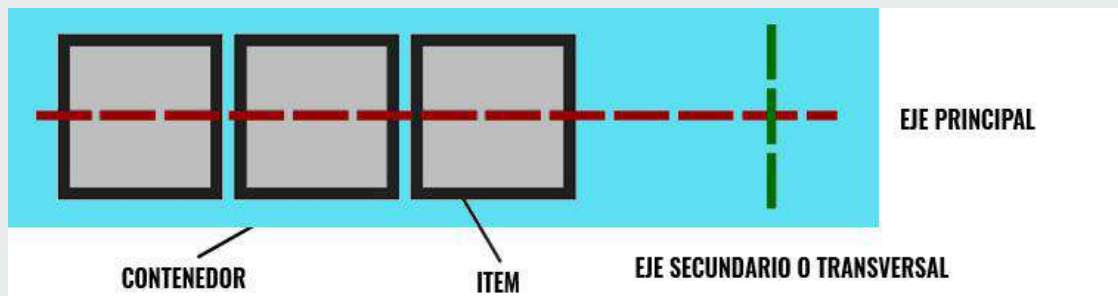
Qué nos permite hacer Flexbox

Las distribuciones flex tienen las siguientes **características**:

- Pueden mostrarse como una **fila** o una **columna**.
- **Respetan el modo de escritura** del documento.
- Son de una sola línea por defecto, pero se les puede pedir que se ajusten a varias líneas. (**flex-wrap: wrap;**)
- Los elementos en el diseño se pueden re-ordenar visualmente, sin considerar su orden en el DOM(en el código). Es decir, aunque en el código los elementos aparezcan en un orden, gracias a flex podemos invertir su orden en pantalla.
- El espacio se puede distribuir dentro de los elementos, por lo que se vuelven más grandes y más pequeños según el espacio disponible en su padre.
- El espacio se puede distribuir alrededor de los elementos y las líneas flex en una distribución envolvente.
- Los elementos en sí se pueden alinear en el eje transversal.

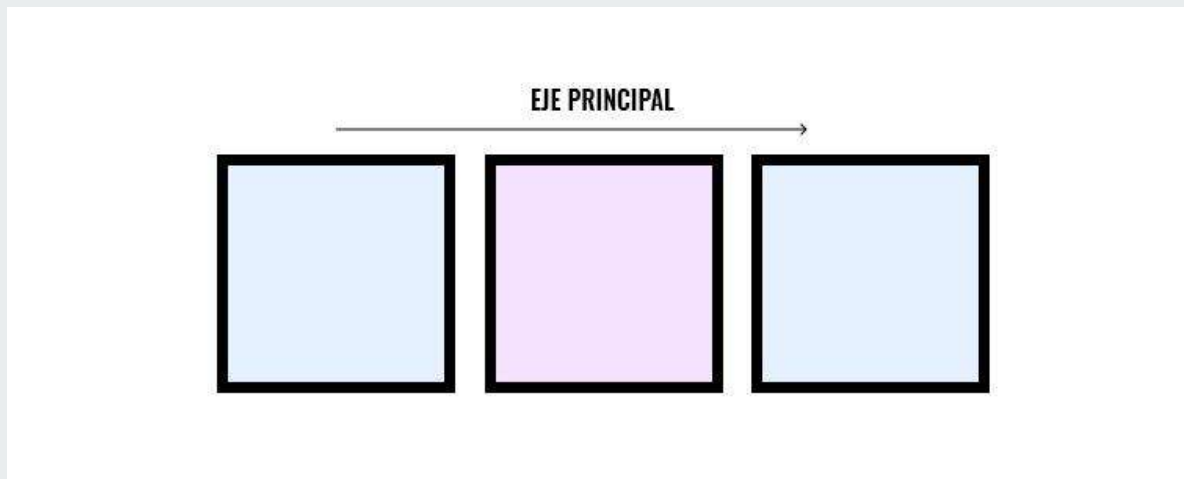
Conceptos básicos de flexbox

- **Contenedor:** Es el elemento padre que tendrá en su interior cada uno de los ítems flexibles. Por norma general, en Flex establecemos las propiedades al elemento padre.
 - **Eje principal:** Los contenedores flexibles tendrán una orientación principal específica. Por defecto, el eje principal del contenedor flex es en horizontal (*en fila*).
 - **Eje secundario:** De la misma forma, los contenedores flexibles tendrán una orientación secundaria, perpendicular a la principal. Si la principal es en horizontal, la secundaria será en vertical (*y viceversa*).
- **Ítem:** Cada uno de los hijos/elementos que encontramos en el interior del contenedor.



Eje principal y eje transversal

La clave para comprender flexbox es entender el concepto de eje principal y eje transversal. El eje principal lo establece la propiedad flex-direction. Si esa es “row” (fila), su eje principal está a lo largo de la fila, si es “column” (columna) su eje principal está a lo largo de la columna.





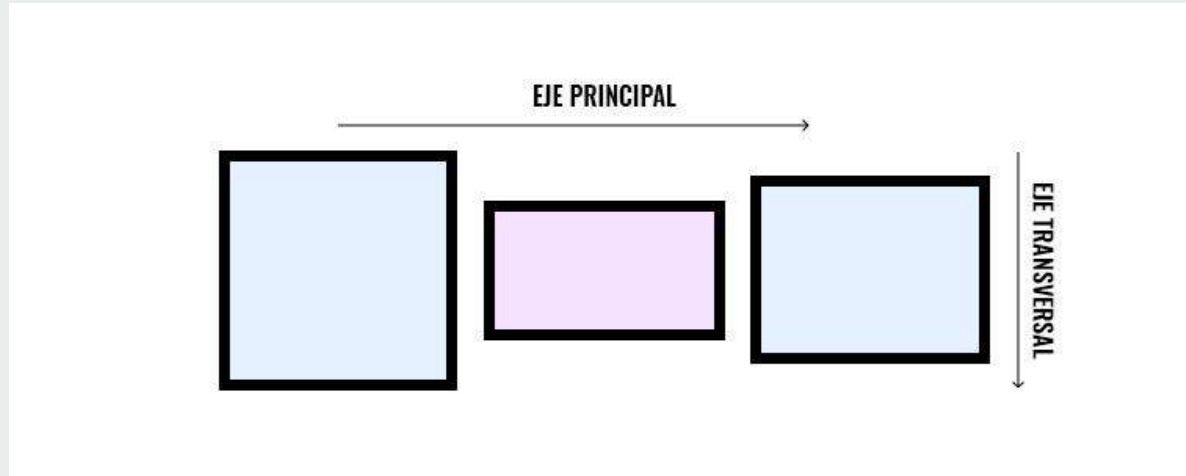
Los elementos flex se mueven como un grupo en el eje principal.

Este eje principal está definido por flex-direction, que posee cuatro posibles valores:

- row
- row-reverse
- column
- column-reverse

El eje transversal corre en la otra dirección al eje principal, por lo que si flex-direction es “row” (fila) el eje transversal corre a lo largo de la “column” (columna).

En el eje transversal podemos mover los elementos individualmente o en grupo para que se alineen entre sí y con el contenedor flex. También podríamos (si hemos envuelto los “hijos” en líneas “flex”) distribuir el espacio.



Propiedad	Valor	Descripción del ejemplo
flex-direction		Cambia la orientación del eje principal-
	row	Establece la dirección del eje principal en horizontal. flex-direction: row;
	row-reverse	Establece la dirección del eje principal en horizontal invertido. flex-direction: row-reverse;
	column	Establece la dirección del eje principal en vertical. flex-direction: column;
	column-reverse	Establece la dirección del eje principal en vertical invertido. flex-direction: column-reverse;



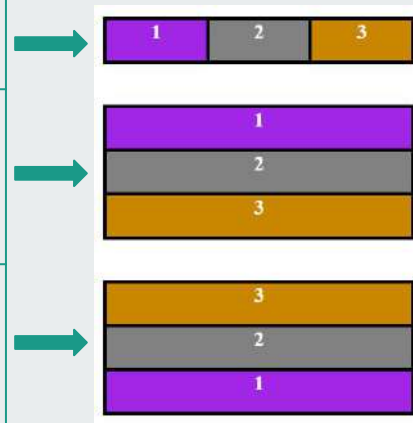


Contenedor Flexbox

En general, flex se suele utilizar para estructuras de una sola dimensión, es decir, contenedores que sólo van en una dirección. Sin embargo, existe una propiedad denominada *flex-wrap* con la que podemos especificar un comportamiento especial del contenedor.

Por defecto, si un elemento no cabe dentro de nuestro contenedor flex, los elementos se harán más pequeños (*son flexibles*) para ajustarlos al contenedor. Este es el comportamiento por defecto de un contenedor flex. Sin embargo, con la propiedad *flex-wrap* podemos cambiar este comportamiento y permitir que nuestro contenedor flex se desborde, convirtiéndose en un contenedor flex multilínea.

Propiedad	Valor	Descripción del ejemplo
flex-wrap		Evita o permite el desbordamiento (multilínea). Es decir, hace que los elementos se mantengan todos en una sola línea o que, por el contrario, ocupen varias.
	nowrap	Los ítems se ajustan para ocupar el tamaño del contenedor (no permite desbordamiento en múltiples líneas). Este sería el comportamiento por defecto (en caso de que no pongamos esta propiedad, los elementos en el navegador se comportarían igual que si pusiéramos este valor). flex-wrap: nowrap;
	wrap	Establece los ítems en modo multilínea (permite que se desborde el contenedor). flex-wrap: wrap;
	wrap-reverse	Establece los ítems en modo multilínea, pero en dirección inversa. flex-wrap: wrap-reverse;



Curiosidad: Dirección de los ejes

Podríamos combinar *flex-direction* y *flex-wrap*, de tal manera que en vez de tener que poner ambas propiedades por separado, las escribamos en una sola línea.

De esta manera, en vez de tener que poner:

```
.container {  
    flex-direction: row;  
    flex-wrap: wrap;  
}
```

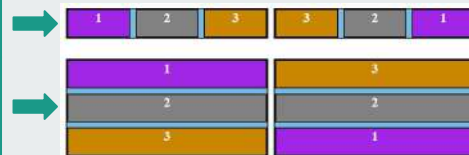
Podríamos escribir:

```
.container {  
    flex-flow: row wrap;  
}
```

Huecos / Gaps

Existen un par de propiedades que han surgido recientemente y que permiten establecer el tamaño del espacio entre ítems desde el elemento padre contenedor.

Propiedad	Valor	Descripción del ejemplo
column-gap	Normal Tamaño fijo	Espacio entre columnas (sólo funciona con <code>flex-direction: row</code>)
row-gap	Normal Tamaño fijo	Espacio entre filas (sólo funciona con <code>flex-direction: column</code>)



Solo podemos combinar tanto row-gap como column-gap cuando hemos definido la propiedad flex-wrap como wrap:

`flex-wrap: wrap;`

Los huecos sólo se aplican entre elementos, y no entre un elemento hijo y su contenedor padre.

Curiosidad: Huecos

Hay una propiedad, llamada gap, que nos sirve para indicar de una sola vez valores para las propiedades *row-gap* y *column-gap*, de forma que escribimos menos y es más cómodo en ciertas situaciones:

De esta manera, en vez de tener que poner:

```
.container {  
  row-gap: 4px;  
  column-gap: 8px;  
}
```

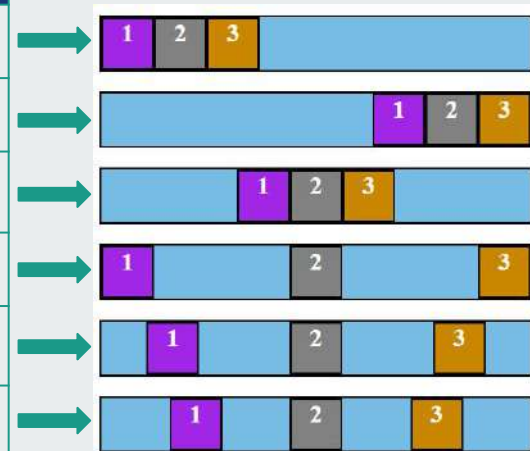
Podríamos escribir:

```
.container {  
  gap: 4px 8px;  
}
```

Propiedades de alineación

Existen varias propiedades que nos van a ayudar a alinear nuestros contenidos. La primera de ellas, actúa en el **eje principal** (en este ejemplo en el que tenemos *flex-direction: row*; este eje principal es horizontal):

Propiedad	Valor	Descripción del ejemplo
<code>justify-content</code>	<code>flex-start</code>	Agrupar los ítems al inicio del eje principal.
	<code>flex-end</code>	Agrupar los ítems al final del eje principal.
	<code>center</code>	Agrupar los ítems al centro del eje principal.
	<code>space-between</code>	Distribuye los ítems dejando espacio entre ellos.
	<code>space-around</code>	Distribuye los ítems dejando espacio alrededor de ellos.
	<code>space-evenly</code>	Distribuye como <code>space-around</code> , pero con un espacio exactamente igual alrededor de ellos.

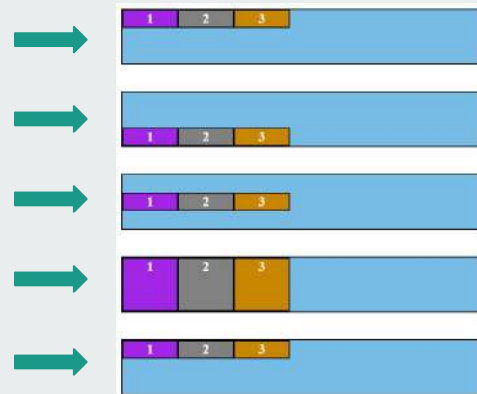


La propiedad *align-items* nos va a permitir alinear los ítems en el eje secundario del contenedor.

Esta propiedad se puede combinar con la propiedad *justify-content*.

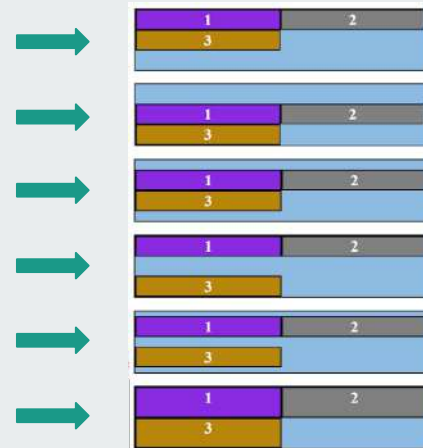
Esta propiedad tiene los siguientes valores:

Propiedad	Valor	Descripción del ejemplo
<i>align-items</i>	<i>flex-start</i>	Alinea los ítems al inicio del eje secundario.
	<i>flex-end</i>	Alinea los ítems al final del eje secundario.
	<i>center</i>	Alinea los ítems al centro del eje secundario.
	<i>stretch</i>	Alinea los ítems estirándolos de modo que cubran desde el inicio hasta el final del contenedor.
	<i>baseline</i>	Alinea los ítems en el contenedor según la base del contenido de los ítems del contenedor.



Por último, tenemos la propiedad **align-content**, que es un caso particular de align-items. Nos servirá cuando estemos tratando con un **contenedor flex multilinea** creado mediante *flex-wrap*. Recordemos que los contenedores multilinea son un tipo de contenedor en el que, cuando los ítems no caben en el ancho disponible, el eje principal se divide en múltiples líneas.

Propiedad	Valor	Descripción del ejemplo
align-content	flex-start	Agrupar los ítems al inicio del eje principal.
	flex-end	Agrupar los ítems al final del eje principal.
	center	Agrupar los ítems al centro del eje principal.
	space-between	Distribuye los ítems desde el inicio hasta el final.
	space-around	Distribuye los ítems dejando el mismo espacio a los lados de cada uno.
	stretch	Estira los ítems para ocupar de forma equitativa todo el espacio.



Elementos Flexbox


Los elementos flexbox nos permiten crear algunos diseños de página bastante complejos. Es perfectamente aceptable configurar un elemento flexible para que también sea un contenedor flexible, de modo que los elementos secundarios también se dispongan como cajas flexibles.

Propiedades de alineación

En el caso de los elementos, podremos usar la propiedad align-self. Esta **actúa exactamente igual que align-items**, sin embargo es la primera propiedad de flex que vemos que **se utiliza sobre un ítem hijo específico** y no sobre el elemento padre contenedor.

La propiedad align-self cambia el comportamiento de align-items y lo sobrescribe con comportamientos específicos para ítems concretos que no queremos que se comporten igual que el resto.

Propiedad	Valor	Descripción del ejemplo
align-self	flex-start	Alinea los ítems al inicio del eje secundario.
	flex-end	Alinea los ítems al final del eje secundario.
	center	Alinea los ítems al centro del eje secundario.
	stretch	Alinea los ítems estirándolos de modo que cubran desde el inicio hasta el final del contenedor.
	baseline	Alinea los ítems en el contenedor según la base de los ítems.
	auto	Hereda el valor de align-items del padre (si no se ha definido, es stretch).



En este caso, podemos combinar las propiedades (y sus valores):

justify-content: flex-start flex-end center space-between space-around space-evenly

align-items: flex-start flex-end center stretch baseline

align-self: flex-start flex-end center stretch baseline auto

Propiedad orden

La propiedad *order* se trata de una propiedad mediante la cual podemos modificar y establecer un orden de los elementos mediante números.

Por defecto, todos los elementos hijos de un contenedor flex tienen establecido un *order* por defecto al valor 0, aunque no se especifique de forma explícita. Si indicamos una propiedad *order* con un valor numérico diferente, los ítems se recolocarán según dicho número, colocando antes los elementos con un número *order* más pequeño (*incluso valores negativos*) y después los elementos con números más altos.

Propiedad	Valor	Descripción del ejemplo
<i>order</i>	<i>número</i>	Número (peso) que indica el orden de aparición de los ítems.

Propiedades de flexibilidad: flex-basis

La propiedad *flex-basis* define el tamaño base por defecto que tendrán los ítems antes de aplicarle una cierta distribución de espacio. Se suele aplicar un tamaño específico (*unidades, porcentajes, etc...*), pero también se puede aplicar la palabra clave *content* que ajusta automáticamente el tamaño al contenido del elemento. Este es el valor por defecto de la propiedad.

Propiedad	Valor	Descripción del ejemplo
flex-basis	Tamaño	Tamaño base de los ítems antes de aplicar una variación.

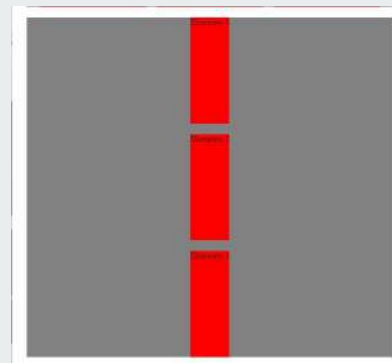
Curiosidad

En función del *flex-direction* definido en el padre, el comportamiento de *flex-basis* será diferente. Si le aplicamos un *flex-direction: column*, invertiremos los ejes primario y secundario, y *flex-basis* actuará como *height* en lugar de como *width*.

```
<div class="container">
  <div class="item">Elemento 1</div>
  <div class="item">Elemento 2</div>
  <div class="item">Elemento 3</div>
</div>
<style>
.container {
  display: flex; justify-content: center; align-items: center; background: grey;
  gap: 20px; margin: 20px;
  flex-direction: row;
}
.item { flex-basis: 200px; background: red; }
</style>
```



```
<div class="container">
  <div class="item">Elemento 1</div>
  <div class="item">Elemento 2</div>
  <div class="item">Elemento 3</div>
</div>
<style>
.container {
  display: flex; justify-content: center; align-items: center; background: grey;
  gap: 20px; margin: 20px;
  flex-direction: column;
}
.item { flex-basis: 200px; background: red; }
</style>
```



Propiedades de flexibilidad: flex-grow

La propiedad flex-grow actúa en situaciones donde:

- Hay un flex-basis definido.
- Los ítems cubren el tamaño total del contenedor flex padre.

En resumen, flex-grow establece un factor de crecimiento observando en conjunto el resto de elementos, y veremos que actúa siempre y cuando la suma del espacio de los elementos hijos no superen el 100% del contenedor padre.

Propiedad	Valor	Descripción del ejemplo
flex-basis	Tamaño	Factor de crecimiento en base a flex-basis ya definido.

Propiedades de flexibilidad: flex-shrink

La propiedad *flex-shrink* es la opuesta a la propiedad *flex-grow*. *Flex-shrink* aplica un factor de decrecimiento.

La propiedad *flex-shrink* actúa en situaciones donde:

- Hay un *flex-basis* definido.
- Los ítems no cubren el tamaño total del contenedor flex padre.

Propiedad	Valor	Descripción del ejemplo
flex-shrink	Tamaño	Factor de decrecimiento en base a flex-basis ya definido.



Elementos Multimedia

Los elementos multimedia se presentan en muchos formatos diferentes. Puede ser casi cualquier cosa que se pueda oír o ver, como imágenes, música, sonido, vídeos, discos, películas, animaciones, etc.

Las páginas web suelen contener elementos multimedia de distintos tipos y formatos.

Audio

Imaginemos que queremos añadir un audio a nuestra web, por ejemplo, un podcast o una entrevista. Para ello, deberemos escribir el código de la siguiente manera:

<audio controls>

<source src="../../ejercicio-video/media/audio.mp3"
type="audio/mp3">

<p>Tu navegador no soporta la etiqueta audio en html5.
Aquí tienes enlace al audio.</p>

</audio>



Los audios tienen las siguientes etiquetas:

Etiqueta de audio	Descripción del ejemplo
<audio>	Define el contenido sonoro
<source>	Define múltiples recursos multimedia para elementos multimedia, como <video> y <audio>

A los audios les podemos añadir los siguientes atributos:

Atributo	Descripción del atributo
autoplay	Un atributo booleano; si se especifica (incluso aunque el valor sea "false"), el sonido comenzará a reproducirse automáticamente en cuanto sea posible, sin detenerse para terminar de cargar los datos.
buffered	Un atributo que se puede leer para determinar qué intervalos de tiempo del multimedia se han almacenado en búfer.
controls	Si está presente este atributo, el navegador ofrecerá controles para permitir que el usuario controle la reproducción de audio, incluyendo volumen, búsqueda y pausar/reanudar reproducción.
src	La URL del audio que se va a insertar. Está sujeta a los Controles de acceso HTTP. Es opcional; en su lugar puedes usar el elemento source dentro del bloque de audio para especificar el audio que se va a insertar.

Existen los siguientes formatos de audio:

Formato de archivo	Tipo de media
mp3	audio/mpeg
ogg	audio/ogg
wav	audio/wav

Vídeo

Una de las maneras de añadir un vídeo a nuestra web es enlazando o embebiendo un vídeo que tengamos nosotros mismos subido a nuestro servidor.

En ese caso, los vídeos los incluiremos en código siguiendo las siguientes pautas.

Gracias al atributo controls, añadiremos controles de vídeo, como reproducción, pausa y volumen.

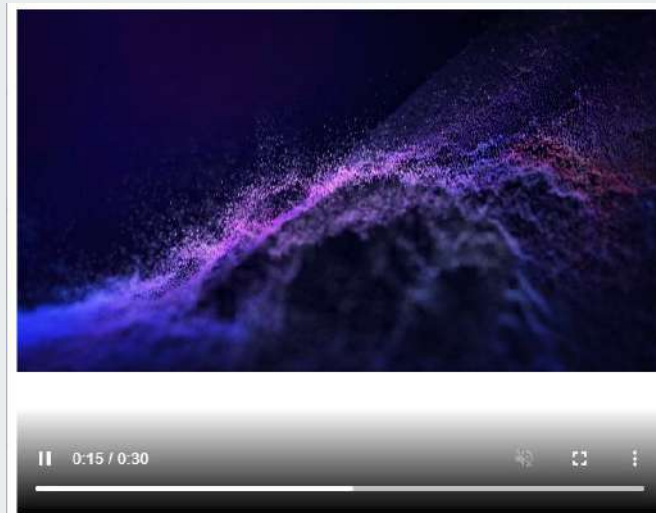
Es buena idea incluir siempre los atributos width y height. Si no establecemos la altura y la anchura, la página podría parpadear mientras se carga el vídeo.

El elemento <source> permite especificar archivos de vídeo alternativos entre los que el navegador puede elegir. El navegador utilizará el primer formato reconocido.

El texto entre las etiquetas <video> y </video> sólo se mostrará en los navegadores que no admitan el elemento <video>.

Ejemplo vídeo que se muestra como contenido

```
<video controls width="560" height="560" autoplay  
loop muted  
poster="../ejercicio-video/img/poster.jpg">  
  <source  
src="../ejercicio-video/media/video.mp4"  
type="video/mp4">  
  <p>Tu navegador no soporta HTML5 video.  
Aquí está el <a  
href="../ejercicio-video/media/video.mp4">enlace  
del video</a>.</p>  
</video>
```



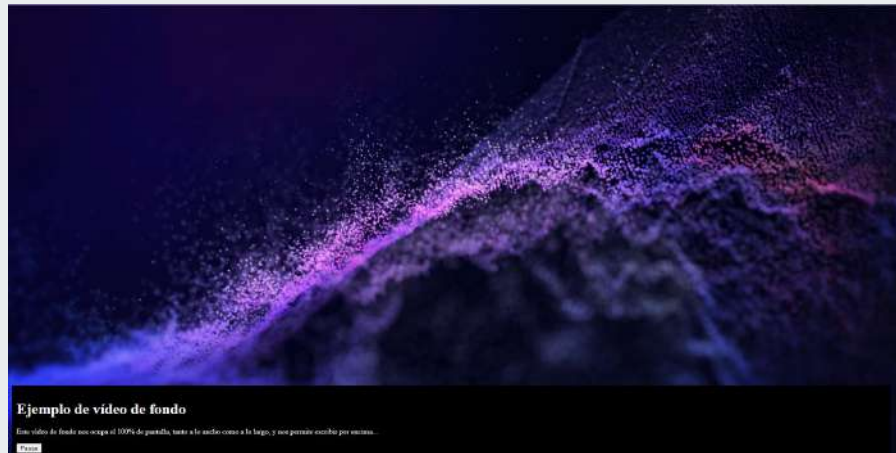
Ejemplo vídeo de youtube que se muestra como contenido

```
<iframe width="560" height="315"  
src="https://www.youtube.com/embed/Z6inVqdMpKI"  
title="YouTube video player" frameborder="0"  
allow="accelerometer; autoplay; clipboard-write;  
encrypted-media; gyroscope; picture-in-picture"  
allowfullscreen></iframe>
```



Ejemplo vídeo de fondo

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ejemplo de como añadir un video de fondo</title>
  <style>
    /* Style the video: 100% width and height to cover the entire window */
    #myVideo {
      position: fixed; right: 0; bottom: 0; min-width: 100%; min-height: 100%; max-width: 100%;
    }
    /* Add some content at the bottom of the video/page */
    .content {
      position: fixed; bottom: 0; background: rgba(0, 0, 0, 0.5); color: #f1f1f1; width: 97%; padding: 10px; border: none; background:
      #000; color: #fff; cursor: pointer;
    }
    #myBtn:hover {
      background: #ddd; color: black;
    }
  </style>
</head>
<body>
  <!-- The video -->
  <video autoplay muted loop id="myVideo">
    <source src="ejercicio-video/media/video.mp4" type="video/mp4">
  </video>
  <!-- Optional: some overlay text to describe the video -->
  <div class="content">
    <h1>Ejemplo de video de fondo</h1>
    <p>Este video de fondo nos ocupa el 100% de pantalla, tanto a lo ancho como a lo largo, y nos permite escribir por encima...</p>
    <!-- Use a button to pause/play the video with JavaScript -->
    <button id="myBtn" onclick="myFunction()">Pause</button>
  </div>
  <script>
    // Get the video
    var video = document.getElementById("myVideo");
    // Get the button
    var btn = document.getElementById("myBtn");
    // Pause and play the video, and change the button text
    function myFunction() {
      if (video.paused) {
        video.play();
        btn.innerHTML = "Pause";
      } else {
        video.pause();
        btn.innerHTML = "Play";
      }
    }
  </script>
</body>
</html>
```



A los vídeos les podemos añadir los siguientes atributos:

Atributo	Descripción del atributo
autoplay	Un atributo booleano; si se especifica, el vídeo comenzará a reproducirse automáticamente tan pronto como sea posible, sin detenerse para terminar de cargar los datos.
buffered	Un atributo que se puede leer para determinar qué intervalos de tiempo del multimedia se han almacenado en búfer.
controls	Si está presente este atributo, Gecko ofrecerá controles para permitir que el usuario controle la reproducción de video, incluyendo volumen, búsqueda y pausar/reanudar reproducción.
height	La altura del área de visualización del vídeo en píxeles CSS.
loop	Un atributo booleano; si se especifica, al alcanzar el final del video, buscaremos automáticamente hasta el principio.
poster	Una URL que indica un marco de póster para mostrar el resultado hasta que el usuario reproduzca o busque. Si este atributo no se especifica, no se muestra nada hasta que el primer cuadro está disponible, entonces se muestra el primer marco como el marco de póster.
src	La URL del vídeo que se va a insertar. Es opcional; podrás optar, en su lugar, por el elemento <u><source></u> dentro del bloque de vídeo para especificar el video que se va a incrustar.
width	La anchura del área de visualización del vídeo en píxeles CSS.

Existen los siguientes formatos de vídeo:

Formato de archivo	Tipo de media
mp4	video/mp4
WebM	video/webm
Ogg	video/ogg

Unidades de medida relativas

Las unidades absolutas son un tipo de medida fija que no cambia, que no depende de ningún otro factor. Son ideales en contextos donde las medidas no varían como pueden ser en medios impresos (documentos, impresiones, etc...), pero son unidades poco adecuadas para la web, ya que no tienen la capacidad de adaptarse a diferentes resoluciones o pantallas, que es lo que tendemos a hacer hoy en día.

Las unidades relativas son un tipo de medida mucho más potente y habitual en el CSS que creamos generalmente. Al contrario que las unidades absolutas, las unidades relativas dependen de algún otro factor (*resolución, tamaño de letra, etc...*). Tienen una curva de aprendizaje más compleja, pero son ideales para trabajar en dispositivos con diferentes tamaños, ya que son muy flexibles y versátiles.



Pixel (la unidad pixel)

La unidad píxel es una unidad absoluta de medida.

El uso de esta unidad de medida está muy extendida en el desarrollo y diseño web, y es prácticamente una de las primeras unidades de CSS que se recomienda aprender. La razón es que se trata de una unidad muy sencilla para el desarrollador: muy fácil de comprender, conocida y que nos permitirá afianzar conceptos a la vez que profundizamos en el diseño web.

Unidades de medida relativas

Unidad	Significado	Medida aproximada	Ejemplo
em	“M”	Multiplicador del tamaño font-size en ese elemento (o heredado)	1.5em * 16px = 24px
ex	“x”	Multiplicador de la mitad del tamaño font-size (altura de la x minúscula)	1ex ~ 0.5em
ch	“zero width”	Multiplicador del tamaño de ancho del carácter cero (0)	1ch ~ 1 carácter
rem	“root M”	Multiplicador del tamaño font-size del elemento raíz (<html> o similar)	
%	Porcentaje	Relativa a herencia (concretamente, al elemento padre)	50% = mitad del padre



La unidad “em”

La unidad em se utiliza para hacer referencia al tamaño actual de la fuente en ese elemento HTML. Por defecto, es un valor aproximado a 16px (*salvo que se modifique por el usuario*). De esta forma, podemos trabajar simplificando las unidades a medidas en base a ese tamaño.

La unidad “rem”

Una unidad muy interesante y práctica para tipografías es la unidad rem (*root em*). Esta unidad toma la idea de la unidad em, pero permitiendo establecer un tamaño base personalizado (*generalmente para el documento en general, utilizando html o la pseudoclase :root*). De esta forma, podemos trabajar con múltiplos del tamaño base.

Esto nos da una ventaja principal considerable: Si queremos cambiar el tamaño del texto en general, sólo tenemos que cambiar el font-size de la pseudoclase :root, puesto que el resto de unidades son factores de escalado y se modificarán todas en consecuencia al cambio del :root. Algo, sin duda, muy práctico y fácil de mantener.

Unidades de medida flexibles (viewport)

Existen unas unidades de "nueva generación" que resultan muy útiles, porque dependen del viewport (*región visible de la página web en el navegador*). Con estas unidades podemos hacer referencia a un porcentaje concreto del tamaño específico que tengamos en la ventana del navegador, independientemente de si es redimensionado o no.

Unidad	Significado	Medida aproximada
vw	viewport width	1vw = 1% ancho de navegador
vh	viewport height	1vh = 1% alto de navegador
vmin	viewport minimum	1vmin = 1% de alto o ancho (el mínimo)
vmax	viewport maximum	1vmax = 1% de alto o ancho (el máximo)

Texto en diseño adaptativo

Tipografía

Existen muchísimos tipos de fuentes y tipografías y en función de lo que queramos expresar y mostrar, utilizaremos una u otra.

Podríamos dividir las tipografías en dos grandes tipos:

- **Serifa (serifa o gracia):** incorporan unos pequeños adornos o remates en los extremos de los bordes de las letras. Muchas de estas tipografías suelen terminar su nombre en «Serif» (*con serifa*).
- **Paloseco (sans-serif):** son las opuestas a la anterior, tipografías lisas, sin adornos o remates en los extremos de los bordes de las letras. Muchas de estas tipografías suelen terminar su nombre en «Sans Serif» (*sin serifa*).

Podemos consultar online y descargarnos tipografías de diferentes bancos de fuentes.

- <https://fonts.google.com/> → De los más conocidos por el mero hecho de pertenecer a Google. Una de las ventajas de este banco de tipografías es que nos permite utilizar sus tipografías en nuestras webs sin necesidad de descargarnos esas fuentes.
- <https://www.dafont.com/es/> → Banco de tipografías en el que podremos ver y descargar, de manera gratuita, tantas tipografías como queramos / necesitemos.
- <https://fonts.adobe.com/> → Por supuesto, Adobe nos proporciona tipografías pero en este caso se trata de tipografías de pago.

Existen más bancos de tipografías pero, de primeras, con los dos primeros mencionados, que además son gratuitos, podremos trabajar sin problemas.

Previsualización de “Dafont”:

Fantasia	Aspecto extranjero	Tecno	Gótico	Básico	Script	Dingbats	Códigos de barras	Holiday	
Animación Tribal, Cómico Groovy Old School Ricos Cute Desgastado Distorsionado Destrozado Terror	Rusgo, Hielo Decoración Máquina de escribir Stencil, Army Bingo Iniciales Cuadrícula Varios	Chino, Japonés Arabe Mokiano Romano, Griego Ruso Varios	Cuadrado LCD Cruzada Rociada Varios	Medieval Moderno Celta Iniciales Varios	Sans serif Serif Ancho fino Varios	Caligrafía Escalón Manuscrito Brocha, Pincel Grafista Grafista Old School Varios	Aliens Animales Rolo Arboreo Runes, Eñico Escalón Fantástico Terror Juegos Formas	Códigos de barras Neurótica Deporte Carnes Pascua Halloween TV, Cine Logos Sexy Ejército Música Varios	San Valentín Pascua Halloween Navidad Varios

Básico > Sans serif

1 2 3 4 5 6 7 8 9 10 ... 103

Mostrar variantes Mediano Popularidad Enviar Más opciones

Lemon Milk

Nombre de la tipografía

CURSO WORDPRESS

Previsualización del texto que hemos puesto de ejemplo. Si hacemos click en el nombre, nos llevará a la página de detalle de la tipografía

15,456,158 descargas (18,794 años) 76 comentarios Donar al autor

Descargar

Donar al autor

Coolvetica de Typodermic Fonts

Curso WordPress

9,120,994 descargas (5,577 años) 131 comentarios 100% Gratis - 5 ficheros

Descargar

Donar al autor

Bebas Neue de Dharma Type

CURSO WORDPRESS

20,891,406 descargas (3,481 años) 76 comentarios Dominio público / GPL / OFL - 2 ficheros

Descargar

Donar al autor

Categorías y formatos de tipografías

Existen diferentes **formatos de tipografía**. Normalmente veremos (y, sobre todo cuando nos descarguemos una fuente) el formato ttf.

Pero cuando trabajemos en una página web, con tipografías especiales, tendremos que tener en cuenta que no todos los navegadores son capaces de interpretar todas las tipografías y deberemos subir la misma fuente / tipografía en varios formatos.

Los formatos a tener en cuenta serán:

*.ttf ; *.otf ; *.eot ; *.woff ; *.woff2

Mientras se trabaje **en local** (en nuestro ordenador) solo instalaremos el formato de fuente que nos hayamos descargado (no necesitaremos disponer de todos los formatos).

Para hacer esto, tendremos que:

1. Descargar la tipografía que deseemos
2. Descomprimir la carpeta de la tipografía que nos hayamos descargado
3. Copiar el / los archivos que formen dicha tipografía
4. Ir a la carpeta C: → Windows → Fonts
5. En esa carpeta “Fonts” será donde peguemos nuestra tipografía

A tener en cuenta que si queremos poder usar esa tipografía en un programa, ese programa debe estar cerrado mientras hacemos la operación de pegar la tipografía en nuestra carpeta “*fonts*”.

Una vez pegada la tipografía en la carpeta “*fonts*”, esta fuente o tipografía estará disponible, en nuestro ordenador, para ser utilizada en cualquier programa que tengamos instalado.

Cuando estemos online, si no hemos usado una tipografía de Google Fonts y estamos usando una tipografía especial propia, deberemos tener en cuenta que deberemos subir esa tipografía al servidor al igual que cualquier otro recurso (como imágenes, vídeos, etc).

La forma de “llamar” a estas tipografías desde nuestra hoja de estilos (o CSS) para poder usarlas será añadiendo al inicio de nuestra hoja de estilos, los siguiente:

```
@font-face{font-family:'Futura T';src:  
url('https://xxxxxxxxx.com/fonts/Futurat.ttf') format('ttf'),  
url('https://xxxxxxxxx.com/fonts/Futurat.otf') format('otf'),  
url('https://xxxxxxxxx.com/fonts/Futurat.eot') format('eot'),  
url('https://xxxxxxxxx.com/fonts/Futurat.woff') format('woff'),  
url('https://xxxxxxxxx.com/fonts/Futurat.woff2') format('woff2');  
font-weight:500;font-style:normal;font-display:swap}
```

Repetiremos cada una de estas llamadas tantas veces como estilos y grosores de fuente tengamos.

Siguiendo el ejemplo anterior, si tuviéramos tres versiones de la anterior tipografía, en el CSS quedaría de la siguiente manera:

```
@font-face{font-family:'Futura T';src:
url('https://xxxxxxxx.com/fonts/futurat.ttf') format('ttf'),
url('https://xxxxxxxx.com/fonts/Futurat.otf') format('otf'),
url('https://xxxxxxxx.com/fonts/Futurat.eot') format('eot'),
url('https://xxxxxxxx.com/fonts/Futurat.woff') format('woff'),
url('https://xxxxxxxx.com/fonts/Futurat.woff2') format('woff2');
font-weight:500;font-style:normal;font-display:swap}
@font-face{font-family:'FuturaT-Bold';src:
url('https://xxxxxxxx.com/fonts/FuturaT-Bold.otf') format('otf'),
url('https://xxxxxxxx.com/fonts/FuturaT-Bold.ttf') format('ttf'),
url('https://xxxxxxxx.com/fonts/FuturaT-Bold.eot') format('eot'),
url('https://xxxxxxxx.com/fonts/FuturaT-Bold.woff') format('woff'),
url('https://xxxxxxxx.com/fonts/FuturaT-Bold.woff2') format('woff2');
font-style:normal;font-display:swap}
@font-face{font-family:'Futura LT Light';src:
url('https://xxxxxxxx.com/fonts/FuturaLTLight.ttf') format('ttf'),
url('https://xxxxxxxx.com/fonts/FuturaLTLight.otf') format('otf'),
url('https://xxxxxxxx.com/fonts/FuturaLTLight.woff') format('woff'),
url('https://xxxxxxxx.com/fonts/FuturaLTLight.eot') format('eot'),
url('https://xxxxxxxx.com/fonts/FuturaLTLight.woff2') format('woff2');
font-weight:100;font-style:normal;font-display:swap}
```

De esta manera tendríamos la llamada. desde nuestra hoja de estilos, a la tipografía deseada. Ahora ya solo quedaría el aplicarla al estilo concreto que quisiéramos.

Esto se haría de la siguiente manera:

```
p {font-family:'Futura T';}
```

Una vez hecho todo esto, ya estaremos usando la tipografía que hayamos escogido.

Para que esta tipografía sea adaptable y funcione correctamente en todos los navegadores y resoluciones, lo único que tenemos que tener en cuenta es:

- Asignar un tamaño de tipografía base al body o al html
- Trabajar el tamaño de los diferentes elementos de texto con medidas relativas, preferentemente con la unidad “rem”.




Diseños Fluidos

Para trabajar correctamente en diseños responsive hay que tener en cuenta que debemos trabajar con unidades relativas e intentar evitar las unidades fijas o estáticas

Una forma interesante de trabajar esa respuesta de los diseños responsive es utilizar propiedades como min-width o max-width, donde definimos tamaños mínimos o máximos, para que los elementos de nuestra página puedan modificar su anchura según sea necesario dependiendo de la pantalla del dispositivo utilizado.

Con estas propiedades podemos crear diseños que aprovechen al máximo toda la pantalla de dispositivos pequeños (*como móviles o tablets*), mientras que establecemos unos máximos en pantallas de dispositivos grandes, para crear unos espacios visuales que hacen que el diseño sea más agradable.



Un tema a tener en cuenta en este tipo de diseños es el de mantener el flujo de los elementos cuando cambian de tamaño y evitar que estos se solapen unos con otros. Estos elementos, además de ir redimensionándose, irán también recolocándose en pantalla.

Para todo esto utilizaremos ciertos «puntos de control».

Esta forma de trabajar nos proporciona múltiples ventajas:


- Es mucho más sencillo mostrar la misma información desde diseños de pantalla grande.
- Ayuda a evitar la mala práctica de ocultar bloques de información en dispositivos móviles.
- Incentiva a diseñar siguiendo buenas prácticas para facilitar la creación responsive.



Diseño en porcentajes

El primer paso para crear un diseño que se adapte correctamente, es comenzar a familiarizarse con un tipo de unidades relativas: los porcentajes. Los porcentajes son relativos al contenedor padre, por lo que si especificamos un porcentaje a un elemento, el navegador va a tomar dicho porcentaje del contenedor.

Al utilizar porcentajes no garantizamos un diseño adaptativo de calidad. El primer problema con el que nos solemos encontrar es que el navegador no solo nos suma los porcentajes que asignamos a los elementos que hay dentro de un contenedor, sino que también se nos suman los bordes, los margins y los paddings. Esto hace que al final los elementos se descuadren y no se adapten como habíamos pensado de primeras. Esto es algo muy habitual en CSS. Y frustrante al principio.



Hay varias formas de solucionar esto:

- Eliminar los bordes y reducir los porcentajes hasta que quepan en el 100% del padre.
- Usar box-sizing: border-box para cambiar el modo en el que se gestionan los tamaños.
- Utilizar un sistema moderno como Flexbox o Grid (*recomendado*).

Tamaños máximos y mínimos

Si buscamos un cierto grado de control aún mayor, podríamos recurrir a las propiedades max-width y min-width , con las que podemos indicar el ancho de un elemento como máximo y el ancho de un elemento como mínimo respectivamente, consiguiendo así garantizar cierto control del diseño.

De esta manera nuestro elementos irá variando en un rango desde el valor mínimo y el valor máximo que hayamos establecido, ajustándose al ancho de la ventana del navegador.


Viewport

También se puede trabajar en base al viewport del navegador. Esa palabra hace referencia a la región visible del navegador, o sea, la parte de la página que está visualizándose actualmente en el navegador. Los usuarios podemos redimensionar la ventana del navegador para reducir el tamaño del viewport y simular que se trata de una pantalla y dispositivo más pequeño.

Si queremos editar ciertos comportamientos del viewport del navegador, podemos editar el documento HTML para especificar el siguiente campo meta, antes de la parte del `</head>`:

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

Con esta etiqueta `<meta>`, estamos estableciendo unos parámetros de comportamiento para el viewport del navegador. Veamos que significan y cuales más existen:



Propiedad	Valor	Significado
width	device-width	Indica un ancho para el viewport.
height	device-height	Indica un alto para el viewport.
initial-scale	1	Escala inicial con la que se visualiza la página web.
minimal-scale	0.1	Escala mínima a la que se puede reducir al hacer zoom.
maximun-scale	10	Escala máxima a la que se puede aumentar al hacer zoom.
user-scalable	no/fixed yes/zoom	Posibilidad de hacer zoom en la página web.

Las propiedades **initial-scale** , **minimum-scale** y **maximum-scale** permiten valores desde el **0.1** al **10** , aunque ciertos valores se traducen automáticamente a ciertos números determinados:

Las propiedades initial-scale , minimum-scale y maximum-scale permiten valores desde el 0.1 al 10 , aunque ciertos valores se traducen automáticamente a ciertos números determinados:

- yes = 1
- no = 0.1
- device-width = 10
- device-height = 10

Por otra parte, user-scalable permite definir si es posible que el usuario pueda «pellizcar» la pantalla para ampliar o reducir el zoom.

Consulta de medios (Media Queries)

Las reglas media queries (*también denominadas MQ a veces*) son un tipo de reglas de CSS que permiten crear un bloque de código que sólo se procesará en los dispositivos que cumplan los criterios especificados como condición:

```
@media screen and (*condición*) {
```

```
/* reglas CSS */
```

```
/* reglas CSS */
```

```
}
```

```
@media screen and not (*condición*) {
```

```
/* reglas CSS */
```

```
/* reglas CSS */
```

```
}
```

Con este método, especificamos que queremos aplicar los estilos CSS para tipos de medios concretos (*screen: sólo en pantallas, en este caso*) que cumplan las condiciones especificadas entre paréntesis. De esta forma, una estrategia aconsejable es crear reglas CSS generales aplicadas a todo el documento: colores, tipo de fuente, etc. y luego, las particularidades que se aplicarían sólo en el dispositivo en cuestión.

Aunque suele ser menos habitual, también se pueden indicar reglas @media negadas mediante la palabra clave not, que aplicará CSS siempre y cuando no se cumpla una determinada condición.

Existen los siguientes tipos de medios:

Tipo de medio	Significado
screen	Monitores o pantallas de ordenador. Es el más común.
print	Documentos de medios impresos o pantallas de previsualización de impresión.
speech	Lectores de texto para invidentes (Antes aural, el cuál ya está obsoleto).
all	Todos los dispositivos o medios. El que se utiliza por defecto.

Recordemos que con el siguiente fragmento de código HTML estamos indicando que el nuevo ancho de la pantalla es el ancho del dispositivo, por lo que el aspecto del viewport se va a adaptar consecuentemente:

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

Con esto conseguiremos preparar nuestra web para dispositivos móviles y prepararnos para la introducción de reglas media query en el documento CSS.

Veamos un ejemplo de media queries en el que definimos diferentes estilos dependiendo del dispositivo que estamos utilizando. En el código existen 3 bloques @media donde se definen estilos CSS para cada uno de esos tipos de dispositivos.

El código sería el siguiente:

```
@media screen and (max-width: 640px) {  
  .menu {  
    background: blue;  
  }  
}  
  
@media screen and (min-width: 640px) and (max-width: 1280px) {  
  .menu {  
    background: red;  
  }  
}  
  
@media screen and (min-width: 1280px) {  
  .menu {  
    background: green;  
  }  
}
```

El ejemplo anterior muestra un elemento (*con clase menu*) con un color de fondo concreto, dependiendo del tipo de medio con el que se visualice la página:

- Azul para resoluciones menores a 640 píxeles de ancho (*móviles*).
- Rojo para resoluciones entre 640 píxeles y 1280 píxeles de ancho (*tablets*).
- Verde para resoluciones mayores a 1280 píxeles (*desktop*).

El número de bloques de reglas @media que se utilicen depende del desarrollador web, ya que no es obligatorio utilizar un número concreto. Se pueden utilizar desde un sólo media query, hasta múltiples de ellos a lo largo de todo el documento CSS.

Hay que tener en cuenta que los media queries también es posible indicarlos desde HTML, utilizando la etiqueta <link>:

```
<link rel="stylesheet" href="mobile.css" media="screen and (max-width: 640px)">
```

```
<link rel="stylesheet" href="tablet.css" media="screen and (min-width: 640px) and (max-width: 1280px)">
```

```
<link rel="stylesheet" href="desktop.css" media="screen and (min-width: 1280px)">
```


Tipos de características

Tenemos una lista de tipos de características que podemos utilizar:

Tipo de característica	Valores	¿Cuándo se aplica?
width	size	Si el dispositivo tiene el tamaño indicado exactamente.
min-width	size	Si el dispositivo tiene un tamaño de ancho mayor al indicado.
max-width	size	Si el dispositivo tiene un tamaño de ancho menor al indicado.
aspect-radio	aspect-radio	Si el dispositivo encaja con la proporción de aspecto indicada.
orientation	landscape portrait	Si el dispositivo está colocado en modo vertical o apaisado.



Enlaces de interés

Los siguientes enlaces que se facilitan son de sitios webs de confianza en los que podremos encontrar recursos e información más que interesante que nos ayudarán tanto a buscar información y ampliar lo aprendido en el curso, como a poder confirmar que estamos realizando bien el trabajo y podremos compartir los proyectos que vayamos haciendo.



Documentación sobre HTML

<https://www.w3schools.com/html/default.asp>

<https://lenguajehtml.com/html/>

<https://developer.mozilla.org/es/docs/Web/HTML>

Documentación sobre CSS

<https://www.w3schools.com/css/default.asp>

<https://lenguajecss.com/css/>

<https://developer.mozilla.org/es/docs/Web/CSS>



Documentación sobre Flexbox

https://www.w3schools.com/css/css3_flexbox.asp

<https://lenguajecss.com/css/maquetacion-y-colocacion/flex/>

[https://developer.mozilla.org/es/docs/Web/CSS/CSS Flexible Box Layout/Basic Concepts of Flexbox](https://developer.mozilla.org/es/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox)

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

<https://web.dev/i18n/es/learn/css/flexbox/>

<http://flexboxgrid.com/>

<https://flexbox.webflow.com/>



Juego para practicar Flexbox

<https://flexboxfroggy.com/#es>

<https://codingfantasy.com/games>

<http://www.flexboxdefense.com/>

Juego CSS Avanzado

<https://css-speedrun.netlify.app/>

<https://www.guess-css.app/>

<https://flukeout.github.io/>

Para limpiar CSS innecesarios

<https://purgecss.com/>



Lorem Ipsum, pero para fotos

<https://picsum.photos/>

Banco de videos, fotos y demás recursos gratuitos

<https://www.pexels.com/es-es/videos/>

<https://pixabay.com/es/videos/>

<https://es.videezy.com/>

Generador de Favicon

<https://favicon.io/favicon-converter/>

<https://www.favicon-generator.org/>



Conversor de tipografías

<https://anyconv.com/es/>