DISEÑO WEB RESPONSIVO

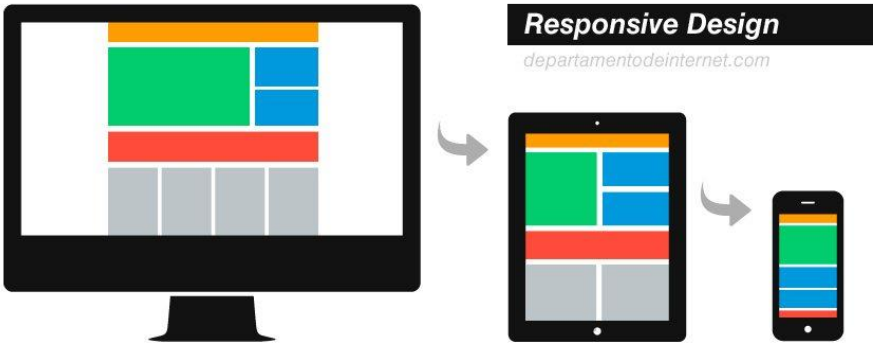
Contenidos

¿QUE ES EL DISEÑO RESPONSIVO? 1
ELEMENTOS RESPONSIVOS 2
VIEWPORT 2
LAYOUT 2
BREAKPOINTS 2
MEDIA QUERIES 3
PATRONES RESPONSIVOS 4
IMÁGENES RESPONSIVAS 5
TABLAS RESPONSIVAS 5
TEXTO RESPONSIVO 7

¿QUE ES EL DISEÑO RESPONSIVO?

Característica del diseño web que permite que una página se muestre correctamente en una variedad de dispositivos y de tamaños, adaptando el layout al entorno de visualización.

A grandes rasgos, esto se consigue mediante lenguaje CSS + ETIQUETAS HTML + MEDIA QUERIES. Y su funcionamiento se basa en realizar consultas a la pantalla sobre sus características con el ancho o el alto, y dependiendo de sus valores se aplicará una regla CSS u otra.



Curso completo fundamentos del diseño responsivo:
https://james-priest.github.io/udacity-nanodegree-mws/course-notes/responsive-web-design-fundamentals.html

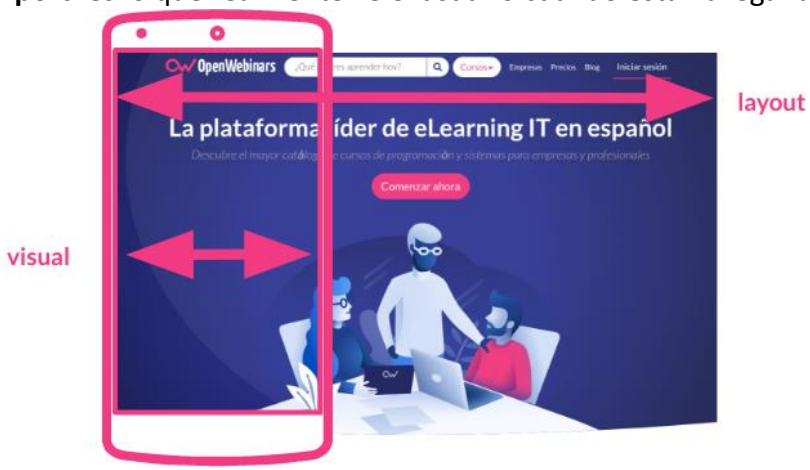
ELEMENTOS RESPONSIVOS

VIEWPORT

Área de la pantalla en la que el navegador puede renderizar contenido, es decir, el espacio disponible para mostrar mi página web.

En ordenadores de escritorio y portátiles el viewport coincide con la pantalla de nuestro navegador, pero en teléfonos y tablets no ocurre lo mismo. Es por ello, que fue necesaria la diferenciación de dos tipos de viewports:

- **Layout-viewport**: es lo que se tiene en cuenta para la aplicación de estilos CSS.
- **Visual-viewport**: es lo que realmente ve el usuario cuando está navegando.



Para resolver en cierto modo los problemas de usabilidad que se presentaban estos dispositivos móviles, se creó una solución genérica y sencilla que permitiera que cualquier página web se mostrase de una forma decente en cualquier tipo de dispositivo y la cual será la base de todos nuestros diseños responsivos:

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

LAYOUT

Se refiere a la disposición y el comportamiento que tendrán los elementos de una página web, según la unidad de medida que usemos para definir su tamaño y posición. Como ya vimos en la parte de Maquetación CSS tradicional, existen **3 tipos principales** de layout:

- ↳ ⚠️ **FIXED (pixeles):** control total sobre los elementos, pero suele generar scroll horizontal si no escalo.
- ↳ ⚠️ **ELASTIC (em):** control al usar zoom y se mantienen las proporciones. Pero al final no deja de ser un fixed, conlleva cálculos y no se adapta bien al cambiar de pantalla.
- ↳ ✅ **FLUID (%):** al estar expresado en porcentajes se adapta al viewport que tengamos en cada caso. Por ello es el que se usa para diseño RESPONSIVO.

BREAKPOINTS

Son las diferentes anchuras en las que se producirá un cambio de layout, en respuesta a las condiciones definidas con los **Media Queries.**

La elección de los breakpoints más adecuados es una tarea compleja, pero por suerte las grandes empresas ya han hecho estudios al respecto y podemos conocer cuales usan. Por ejemplo, unos recomendados son los usados por Bootstrap:

- *Menor de 576px (pantallas pequeñas)*
- *Entre 576px y 768px (móviles apaisados)*
- *Entre 768px y 992px (tablets)*
- *Entre 992px y 120ppx (escritorio, portátiles)*
- *Mayor de 1200px (pantallas grandes)*

MEDIA QUERIES

Característica de CSS3 que nos permite adaptar la representación de una página web a las características del dispositivo. Son **expresiones** en las que indicamos el **tipo de medio** donde voy a hacer este cambio y una **consulta** en relación a las características del dispositivo como alto, ancho e incluso color

```
@media mediatype [condiciones] {
/* Estilos a aplicar cuando se cumplan esas condiciones */
}
```

Los distintos ***mediatype*** que puedo usar son los siguientes, teniendo en cuenta que, si no lo indicamos, por defecto tomará el valor "`all`":

- `all`: aplicable a todos los dispositivos.
- `print`: destinado a material paginado y documentos visibles en una pantalla en modo de vista de impresión.
- `screen`: destinado principalmente a pantallas de computadora.

En cuanto a las condiciones que puedo consultar hay muchas y podemos hacer combinaciones de ellas, usando los operadores **AND**, **NOT**, **ALL** y **ONLY**. Las más comunes son:

- `width | min-width | max-width`
- `height | min-height | max-height`
- `orientation (landscape / portrait)`
- `aspect-ratio | min-aspect-ratio | max-aspect-ratio`
- `color | min-color | max-color`

Algunos ejemplos:

```
/* Estilos para todo tipo de pantallas con una anchura máxima de 576px*/
@media all and (max-width: 576px) {
  .....;
}
/* Estilos para pantallas con al menos 992px de anchura y que estén apaisadas*/
@media screen and (min-width: 992px) and (orientation: landscape) {
  ......;
}
/* Estilos sólo para pantallas que tengan al menos 768px de anchura*/
@media only screen and (min-width: 768px) {
  ......;
}
```

Si así lo preferimos, podemos usar diferentes hojas para las diferentes media queries que tengamos:

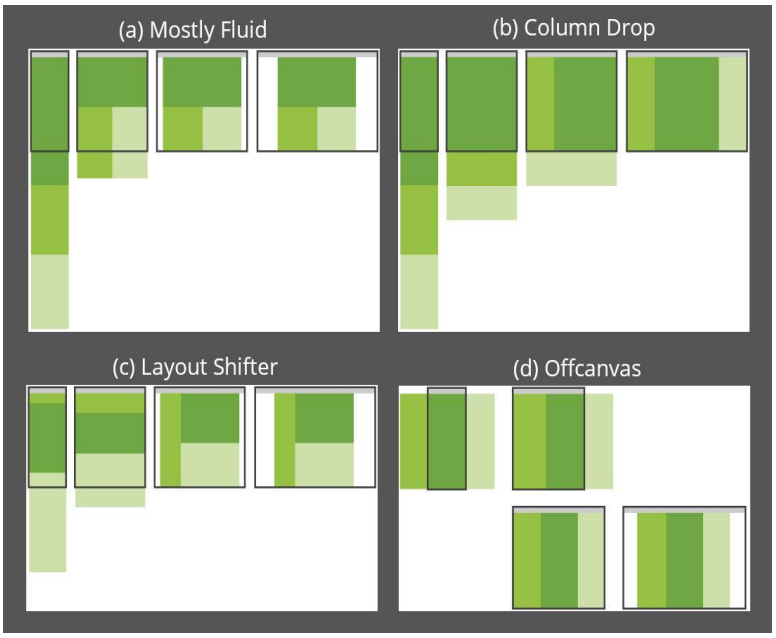
```
<!--Para pantallas de hasta 576px de ancho -->
<link rel="stylesheet" media="(max-width: 576px)" href="small.css" />

<!--Para pantallas entre 576px y 768px de ancho -->
<link rel="stylesheet" media="(max-width: 768px)" href="medium.css" />

<!--Para pantallas entre 768px y 992pxde ancho -->
<link rel="stylesheet" media="(max-width: 992px)" href="large.css" />
```

PATRONES RESPONSIVOS

Existen una serie de soluciones que se han dado por buenas para resolver diferentes cuestiones del diseño responsivo y podemos usarlas como base para nuestros proyectos:

- ✓ **START SMALL:** nuestra página web debe empezar haciendo que todo quede correctamente en pantallas pequeñas. Así priorizamos los contenidos adecuadamente y me pregunto en cada tamaño mayor, si es necesario un nuevo diseño.

- ✓ **COLUMN DROP**: Es el patrón más básico y consiste en que en cada breakpoint se va a apilando un elemento.
- ✓ **MOSTLY FLUID**: parecido a Column Drop. Es una cuadrícula fluida y en cada breakpoint hay redimensionamiento de varias columnas.
- ✓ **LAYOUT SHIFTER**: es el patrón más responsivo, en cada breakpoints cambia el diseño del layout, no únicamente el flujo y la anchura de los elementos.



Ejemplos prácticos de estos patrones en:
https://carlosazaustre.es/los-5-patrones-del-responsive-design

- ✓ **OFF CANVAS**: en vez de apilar contenidos éstos se colocan fuera de la pantalla cuando el tamaño de pantalla no es lo suficientemente grande.

IMÁGENES RESPONSIVAS

A la hora de optimizar el uso de imágenes para un diseño responsivo, se presentan dos grandes retos:

- ✓ Consumir el menor ancho de banda posible.
- ✓ Elegir la versión de una misma imagen más adecuada para cada resolución.

Para conseguirlos, podemos usar las siguientes etiquetas:

↳ **SRCSET / SIZES:**

Por un lado, podemos usar imágenes **SVG** que se escalan y encogen sin perder resolución. Pero no siempre es posible esta opción, para ello haremos uso de los atributos **`scrset`** y/o **`sizes`** de la imagen que queremos optimizar:

```
<!-- Considerando resolución: -->
<div class="container">
  <img             src="https://cdn.openwebinars.net/media/img/small.jpg"
srcset="img/big.jpg 2x, img/small.jpg 1x" />
</div>

<!-- Considerando dimensiones: -->
<div class="container">
  <img
    src="https://cdn.openwebinars.net/media/img/small.jpg"
    srcset="img/big.jpg 2000w, img/small.jpg 1000w"
    sizes="(min-width: 960px) 960px,100vw" />
</div>
```

↳ **ART DIRECTOR**

Opción del diseño que consiste en elegir una u otra imagen utilizando la etiqueta **`source`** dentro la etiqueta **`picture`** y sus atributos **`srcset`** y **`media`**, que funciona de manera similar a una media query:

```
<div class="art">
  <picture>
    <source media="(min-width: 576px)" srcset="img/big-art.jpg" />
    <source media="(max-width: 575px)" srcset="img/small-art.jpg" />
    <!-- En caso de no soportar picture: -->
    <img src="https://cdn.openwebinars.net/media/img/small-art.jpg" />
  </picture>
</div>
```

TABLAS RESPONSIVAS

Las tablas son un elemento problemático a la hora de realizar **Diseño Responsivo** ya que en cuanto tienen un número de columnas considerable pueden provocar la aparición del tan temido *scroll horizontal*.

Para afrontar este tipo de problemas hay tres soluciones principales:

- ✓ Esconder columnas.
- ✓ Convertir las filas en listas.
- ✓ Crear un scroll horizontal que solo se aplique a la tabla.

↳ **ESCONDER COLUMNAS:**

Esta técnica consiste básicamente en esconder ciertas columnas, que debe de ser las menos importantes, cuando el tamaño de la pantalla es menor que un breakpoint establecido:

```
@media screen and (max-width: 576px) {
  .hidden td.primero,
  .hidden th.primero {
    display: none;
  }
}

@media screen and (max-width: 768px) {
  .hidden td.segundo,
  .hidden th.segundo {
    display: none;
  }
}
```

↳ **CONVERTIR FILAS EN LISTAS:**

Esta técnica consiste en hacer desaparecer las cabeceras de la tabla cuando la pantalla es menor que una determinada cantidad y hacer que todas las celdas se conviertan en elementos de bloque para que se muestren una debajo de otra y no al lado.

```
@media screen and (max-width: 700px) {
  table.listas,
  table.listas thead,
  table.listas tbody,
  table.listas tr,
  table.listas th,
  table.listas td {
    display: block;
  }

  table.listas thead {
    display: none;
  }
```

↳ **SCROLL CONTROLADO:**

Consiste en acotar el scroll horizontal para que si ha de aparecer solo afecte a la tabla y no a la página entera. Para conseguirlo debemos “envolver” la tabla en un contenedor y darle las siguientes propiedades:

```
//HTML:
<div class="localscroll">
  <table>
    .....
  </table>
</div>

//CSS:
div.localscroll {
  overflow-x: auto;
  width: 100%;
}
```

TEXTO RESPONSIVO

Cuando hablamos de diseño responsivo solemos centrarnos principalmente en el layout, pero el texto mostrado es igual de importante para conseguir un buen diseño.

Debemos intentar evitar principalmente:

- ↳ Líneas demasiado cortas.
- ↳ Líneas demasiado largas.
- ↳ Tamaños muy pequeños o ilegibles.

Una forma recomendada de solucionar estos errores con garantías es indicar el tamaño del texto en unidades relativas al viewport. Como son:

- **vw:** en relación a la anchura del viewport.
- **vh:** en relación a la altura del viewport.
- **vmin:** el valor menor en relación a la dimensión pequeña del viewport (anchura o altura).
- **vmax:** el valor mayor en relación a la dimensión más grande del viewport (anchura o altura).

Teniendo en cuenta que, por ejemplo, **1vw sería el 1% de la anchura del viewport.** Una pauta por la que nos podemos guiar es aquella que dice que la longitud ideal de una línea de texto está entre **60 y 80 caracteres**.