¡NO TE DOY DEBERES!

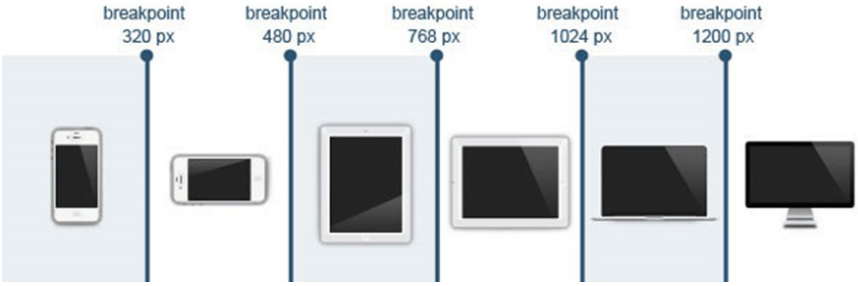
¡TE DOY PODERES!

CSS3 ya incorpora propiedades responsive

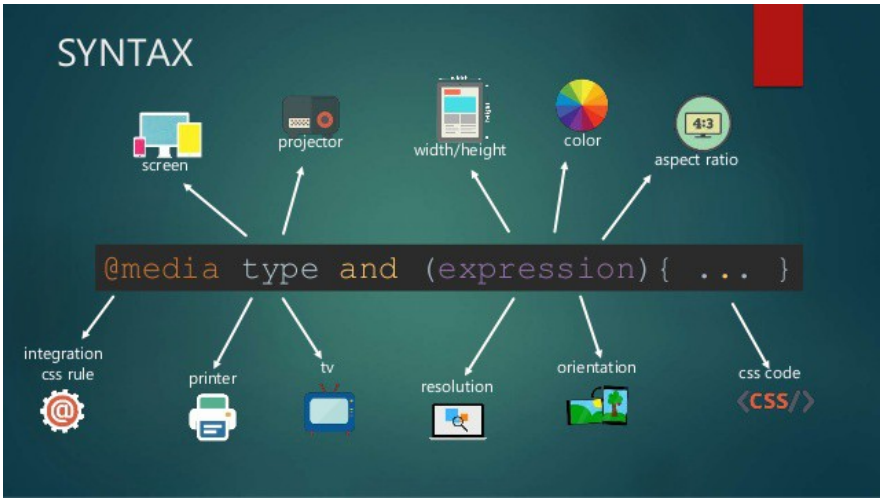
En una batería anterior dedicada a practicar las propiedades que CSS pone a disposición del desarrollador a la hora de confeccionar documentos web, nos quedamos adquiriendo experiencia sobre sobre posicionamiento de cajas. Pero es que el mundo de las CSS da para mucho más y para muestra constatar la evolución que ha sufrido la recomendación de la W3C, que ya va por la CSS3.

Pues es en esta versión en las que se introdujeron mejoras que han significado un salto cualitativo para el estilado de elementos HTML5: transiciones, animaciones y *media queries*.

Estas *media queries*, como ya viste en la presentación de clase, se valen de la detección automática del tamaño en píxeles del viewport en cada momento para permitir al desarrollador dar las pautas de rediseño de los elementos que le vengan bien si se cumple una condición determinada asociada normalmente al ancho máximo o mínimo de pantalla (breakpoints). Eso sí, todo a manita.



La sintaxis tipo para comprender una *media query* podría ser esta:

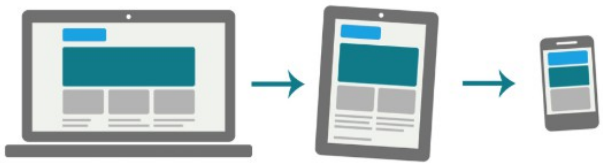


Diferencia de enfoques: Desktop First vs. Mobile First

Como ya nos quedó claro al trabajar con CSS, las propiedades de estilado se aplican a un documento HMTL5 a medida que se van leyendo. Además, cualquier modificación que se realiza sobre una propiedad sobreescribe su anterior valor.

Cuando esta circunstancia entra en contacto con el mundo de las *media queries*, a poco que uno entienda que el paso de una *media query* a otra supone la sobreescritura de algunas de sus propiedades CSS, llega a la conclusión de que no es lo mismo empezar a codificar en el fichero .css por la *media query* de menos píxeles (tamaño pantalla *smartphone*) que por la de más píxeles (tamaño pantalla ordenador sobremesa/portátil).

De ahí se derivan los dos enfoques que actualmente coexisten: **Mobile First** y **Desktop First**. En el primero se propone un diseño de escritorio (Desktop), codificando por tanto la primera *media query* para ese entorno. A partir de ahí y echando mano de lo que se conoce como diseño receptivo, se van adaptando las siguientes *media queries* para dispositivos móviles. Procediéndose al revés con el enfoque **Mobile First**. De esta forma, se logra una mejor experiencia de usuario (UX) en según qué dispositivo.

Desktop First	***Mobile First***
	

Como nos podemos imaginar, la elección de un enfoque u otro para el desarrollador irá en función de la estimación porcentual del tipo de dispositivo de los usuarios a los que va dirigida la aplicación.

En la actualidad, se considera que el enfoque **Mobile First** es una buena práctica para el desarrollo web por el hecho de que los usuarios de dispositivos móviles son cada vez más numerosos. De ahí que la mayor parte de los desarrollos adopten este enfoque.

Sin embargo, dependiendo del proyecto, puede ser más adecuado utilizar otros enfoques siempre primando que diseño y desarrollo se adapte a las necesidades y expectativas de los usuarios, centrándose en crear una experiencia de usuario positiva y efectiva.

Pero nada como ponerlo en forma de código para rematar el aprendizaje:

URWD-0.1 Implementar un código CSS3 que permita, mediante una media query, indicar que el color de fondo del documento en pantalla será el rojo mientras la anchura de pantalla esté por debajo de 801 píxeles.	pantalla
	Hasta esta anchura
	`@media only screen and (max-width: 800px) {`
	`  body {`
	`  background-color: red;`
	CSS a aplicar
	`  }`
	`}`
	Hoy en día el **only** no es necesario

URWD-0.2 Implementar un código HTML5 que, mediante la utilización de media queries (CSS3), permita alterar el aspecto de un párrafo tanto en su fondo (rojo, azul, verde, naranja y rosa) como en los estilos de fuente (negrita, cursiva, etc..). El párrafo irá precedido por un encabezado de tipo 2. Atentos, porque a medida que se ensanche o estreche la ventana del navegador deben ir reflejándose dichos cambios.

```
<!DOCTYPE html>
<html>
   <head>
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <style>
         .cajatexto {
           padding: 20px;
           color: white;
           font-family: Arial;
           font-size: 16px;
         }

         /* Dispositivos muy pequeños (por debajo de 600px [móviles]) */
         @media screen and (max-width: 599px) {
           .cajatexto {background: red; font-style: normal;}
         }

         /* Dispositivos pequeños  (igual o por encima de 600px [móviles grandes y minitablets]) */
         @media screen and (min-width: 600px) {
           .cajatexto {background: green; font-style: italic;}
         }

         /* Dispositivos medianos (igual o por encima de 768px [tablets]) */
         @media screen and (min-width: 768px) {
           .cajatexto {background: blue; font-weight: bold;}
         }

         /* Dispositivos grandes (igual o por encima de 992px [portátiles/escritorio]) */
         @media screen and (min-width: 992px) {
           .cajatexto {background: orange; font-style: normal;}
         }

         /* Dispositivos muy grandes (igual o por encima de 1200px [portátiles/escritorio grandes]) */
         @media screen and (min-width: 1200px) {
           .cajatexto {background: pink; font-weight: normal;}
         }
      </style>
   </head>
   <body>
      <h2>Puntos de ruptura habituales para media queries</h2>
      <p class="cajatexto">A medida que vayas ensanchando/estrechando la anchura de la ventana del navegador observarás cómo el fondo de la caja que contiene el texto va cambiando de color y el estilo de la letra en función de esos tamaños.</p>
   </body>
</html>
```

Estilo que se aplica sea cual sea el tamaño de ventana

Añade al estilo cajatexto un color de fondo rojo y un estilo de fuente normal

Añade por sustitución al estilo cajatexto un color de fondo verde y un estilo de fuente cursivo

Añade por sustitución al estilo cajatexto un color de fondo azul y un estilo de fuente negrita

Añade por sustitución al estilo cajatexto un color de fondo naranja y un estilo de fuente normal

Añade por sustitución al estilo cajatexto un color de fondo rosa y un estilo de fuente normal

De este ejemplo se puede extraer un primer conocimiento que te resultará vital a partir de ahora. La disposición de los media queries no ha sido casual. Primero se ha establecido el estilo de la caja independientemente del tamaño de ventana. Y en ese momento se ha tomado la decisión de ir secuenciando los *media queries* considerando que lo habitual es que se vea en un móvil aunque si se va ensanchando la ventana vaya adquiriendo por añadidura o sustitución nuevas propiedades. A este enfoque se le denomina *mobile first*.

No obstante, no es más que una decisión puesto que se puede considerar que el documento se verá habitualmente en monitor grande de PC y se le prepara para que reaccione al estrechar la ventana. Observa como los **max-width** y **min-width** han variado del ejemplo anterior al que se ahora se presenta con enfoque *desktop first*.

URWD-0.3 Implementar un código HTML5 que, mediante la utilización de media queries (CSS3), permita alterar el aspecto de un párrafo tanto en su fondo (rojo, azul, verde, naranja y rosa) como en los estilos de fuente (negrita, cursiva, etc..). El párrafo irá precedido por un encabezado de tipo 2. Atentos, porque a medida que se ensanche o estreche la ventana del navegador deben ir reflejándose dichos cambios.

```
<!DOCTYPE html>
<html>
   <head>
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <style>
         .cajatexto {
           padding: 20px;  color: white;  font-family: Arial;  font-size: 16px;
         }

         /* Dispositivos muy grandes (por encima de 1200px [portátiles/escritorio grandes]) */
         @media screen and (min-width: 1201px) {
           .cajatexto {background: pink; font-weight: normal;}
         }

         /* Dispositivos grandes (igual o por debajo de 1200px [portátiles/escritorio]) */
         @media screen and (max-width: 1200px) {
           .cajatexto {background: orange; font-style: normal;}
         }

         /* Dispositivos medianos (igual o por debajo de 992px [tablets]) */
         @media screen and (max-width: 992px) {
           .cajatexto {background: blue; font-weight: bold;}
         }

         /* Dispositivos pequeños  (igual o por debajo de 768px [móviles grandes y minitablets]) */
         @media screen and (max-width: 768px) {
           .cajatexto {background: green; font-style: italic;}
         }

         /* Dispositivos muy pequeños (igual o por debajo de 600px [móviles]) */
         @media screen and (max-width: 600px) {
           .cajatexto {background: red; font-style: normal;}
         }
      </style>
   </head>
   <body>
      <h2>Puntos de ruptura habituales para media queries</h2>
      <p class="cajatexto">A medida que vayas ensanchando/estrechando la anchura de la ventana del navegador observarás
cómo el fondo de la caja que contiene el texto va cambiando de color y el estilo de la letra en función de esos tamaños.</p>
   </body>
</html>
```

Estilo que se aplica sea cual sea el tamaño de ventana

Añade al estilo cajatexto un color de fondo rosa y un estilo de fuente normal

Añade por sustitución al estilo cajatexto un color de fondo naranja y un estilo de fuente normal

Añade por sustitución al estilo cajatexto un color de fondo azul y un estilo de fuente negrita

Añade por sustitución al estilo cajatexto un color de fondo verde y un estilo de fuente cursivo

Añade por sustitución al estilo cajatexto un color de fondo rojo y un estilo de fuente normal

¡Ojo!, porque también es posible aplicar un cambio de estilo que al desarrollador le pueda venir bien cuando el dispositivo, como en el caso de un smartphone o tablet, cambie de orientación en las manos del usuario. Pues también se puede conformar el *media query* para dicha circunstancia. Para ello hay que indicarle si la orientación es a lo alto (portrait) o a lo ancho (landscape).

URWD-0.4 Implementar un código CSS3 que permita, mediante una media query, indicar que el color de fondo del documento en pantalla será el azul luminoso cuando la orientación de la pantalla se encuentre en apaisado (a lo ancho).

```
@media screen and (orientation: landscape) {
  body {
    background-color: lightblue;
  }
}
```

pantalla

Apaisada

CSS a aplicar

Ahora me toca a mi. URWD-1

Una vez descargado el fichero comprimido asociado a la batería que encontrarás en el aula virtual, comprueba cómo se comportan los ejemplos en distintos dispositivos pero ahora en vez de estrechar/ensanchar la ventana házlo con la herramienta de comprobación que te suministra tu navegador Firefox y/o Chrome.

a) Coger el fichero **DesktopFirst.html** y pasarlo por Firefox -> Desarrollador Web -> Icono RWD
a.1) Probar con el primer ancho por defecto (Adaptable) 320 x 480 (vertical) y pulsar sobre icono orientación
a.2) Escoger IPhone X/XS IOS 12 ancho 375 x 812 (vertical) y pulsar sobre icono orientación
b) Tras pulsar en la opción Editar lista... y buscar la sección Tablets
b.1) Escoger IPad ancho 768 x 1024 (vertical) y pulsar sobre icono orientación
b.2) Escoger Kindle Fire HDX Linux ancho 800 x 1280 (vertical) y pulsar sobre icono orientación
c) Tras pulsar en la opción Editar lista... y buscar la sección Laptops
c.1) Escoger Laptop with MDPI screen ancho 1280 x 800 (horizontal) y pulsar sobre icono orientación

Ahora me toca a mi. URWD-2

Aprovechando que dispones de los ficheros DesktopFirst.html y MobileFirst.html estaría bien que añadieras y quitaras (a modo de prueba) la etiqueta meta que afecta al viewport para observar qué pasa. Con dicha etiqueta presente puedes probar a darle distintos valores para comprobar las diferencias que se producen. Comprueba especialmente cómo afecta la propiedad user-scalable.

a) Para **DesktopFirst.html**
a.1) Modificando el valor de initial-scale:
a.1.1) initial-scale=1 --> Resultado normal
a.1.2) initial-scale=2 --> Resultado igual que el anterior
a.1.3) initial-scale=0.5 --> Resultado igual que el anterior
a.2) Modificando el valor de user-scalable:
a.2.1) con user-scalable=yes / a.2.2 con user-scalable=no
a.2.3) con user-scalable=yes / a.2.4 con user-scalable=no
a.2.5) con user-scalable=yes / a.2.6 con user-scalable=no

b) Para **MobileFirst.html**
b.1) Modificando el valor de initial-scale:
b.1.1) initial-scale=1
b.1.2) initial-scale=2
b.1.3) initial-scale=0.5
b.2) Modificando el valor de user-scalable:
b.2.1) con user-scalable=yes / b.2.2 con user-scalable=no
b.2.3) con user-scalable=yes / b.2.4 con user-scalable=no
b.2.5) con user-scalable=yes / b.2.6 con user-scalable=no

Extra: se añadido al comprimido el fichero **MetaConfig.html** con la línea de metaconfiguración comentada de forma que la vayas modificando los parámetros y compruebes los efectos que produce en el resultado final.

Si compruebas que todo se ve igual cambies lo que cambies, tranquilo. Seguro que lo estás viendo en un ordenador PC o portátil, donde el viewport es completamente inocuo. Donde se empiezan a operar cambios es cuando las pruebas se realizan sobre las emulaciones de móviles que ofrecen como herramienta los navegadores (Firefox y Chrome). Aun así, nunca quedará igual que probarlo sobre un móvil de verdad.

Esto tiene más tela de la que parece

Como bien se puede uno imaginar la complejidad que conlleva el desarrollo de algunos sitios web (por los requerimientos del cliente, claro) incide directamente en la complejidad que irá adquiriendo para el desarrollador el aspecto *responsive* de su confección.

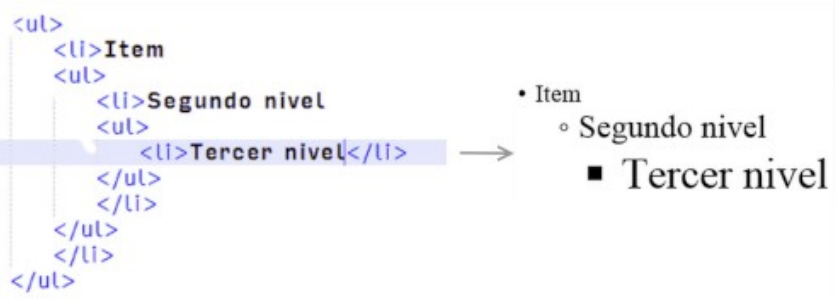
La cosa empieza a ponerse seria cuando hay que tener en cuenta tipografías adaptables, imágenes adaptables, multitud de elementos a posicionar en pantalla, etc... Ahí es donde el desarrollador experimenta la sensación del marinero que por mucho que avanza en su barco siempre se encuentra a la misma distancia del horizonte.

De ahí que haya que darle una vuelta de tuerca más al aprendizaje sobre esto del *responsive* incorporando conceptos como el de cuadrícula fluida o flexible. Para ello tienes la **MicroBatería Cuad_Fluida_4cols.pdf** con sus códigos de ejemplo asociados para practicar mientras la sigues.

Tipografías adaptables (o flexibles)

Esto no es nuevo para nosotros. En la batería de introducción a CSS ya hicimos mención a las unidades relativas (em y rem) que ya dijimos que dependen del tamaño que adquiera el contenedor y elemento raíz respectivamente a medida que se estreche o ensanche el mismo.

Sabemos también que cada navegador establece un tamaño de letra base, por lo que hay que conocer este detalle para evitar sorpresas. Asimismo, la utilización de ***em*** implica conocer que si un párrafo está dentro de dos etiquetas **div**, heredará en casacada los aumentos de cada **div** produciendo (si no se conoce esto) efectos indeseados.



De ahí que cada vez sea más habitual en los desarrolladores optar por la utilización de la unidad ***rem*** para que la referencia siempre sea el elemento raíz del documento.

No obstante, la utilización de tamaño de tipografía asociado a viewport (vw, vh, vmin, vmax) también es una alternativa para evitar problemas.

Como puedes comprobar no existe una receta perfecta. Todo depende de que el desarrollador conozca todas estas herramientas a su disposición y de la pericia que tenga al utilizarlas para conseguir un resultado lo más óptimo.

Se ha incluido en el aula virtual una carpeta con documentos de ejemplo donde puedes comprobar desde la evolución del paso de configurar una tipografía clásica a hacerla adaptable hasta el ajuste de la misma al viewport de turno.

Imágenes adaptables (o flexibles)

El primer concepto que debemos tener en cuenta al entrar en este apartado es que las imágenes utilizadas en sitios web han sido generadas **en una única resolución**, seleccionada como un mínimo común denominador adaptable a las distintas resoluciones y tamaños de pantalla. Pero

ya somos muy conscientes de que las imágenes dejadas al libre albedrío de la anchura de una pantalla/viewport de cada dispositivo pueden degradar la visualización de dichas imágenes.

De ahí que haya que trabajarse el paño para conseguir eludir el mayor número posible de problemas que se nos puedan presentar (pérdida de foco, pesos y formatos), intentando que la experiencia de usuario sea la mejor posible confeccionando en todo momento imágenes fluidas. Si pero la pregunta es, ¿cómo se consiguen imágenes fluidas si éstas nos vienen ya con un nº fijo de píxeles tanto de ancho como de alto? Bueno, sí que es verdad que ya sabíamos cambiarle por la cara la dimensión a una imagen en el momento de incluirla dentro de la etiqueta <img>. Si no se le indica nada la muestra con su dimensión original, pero hay unos atributos que sí nos permitían cambiarla a voluntad para su remuestreo:

```
<img src="miimagen.png" width="tuanchura" height="tualtura">
```

Está muy bien lo anterior pero si coges un globo, le escribes una palabra con bolígrafo y empiezas a hincharlo verás como la palabra se lee cada vez peor. Las imágenes forzadas a un cambio de dimensión acaban, más tarde o más temprano, dando problemas de pixelación. De ahí que se aconseje la utilización de las ya conocidas unidades relativas de medida como **%**, **em**, **rem** y **vw** como forma de encaminarnos hacia soluciones fluidas en esta materia.

El uso de dimensiones relativas para las imágenes de un sitio web habitualmente es más eficaz cuando se combina con una distribución fluida (vistos y por ver, Pure, BS4, Flexbox y Grid), lo cual permite aprovechar de la mejor forma el espacio disponible.

Abundando en qué más se debe tener en cuenta para conseguir hacer más responsive una imagen, existe también la posibilidad de detectar la densidad de píxeles en pantalla de un dispositivo y utilizar dicha información para que un desarrollador pueda suministrar imágenes de distinta resolución de manera transparente al usuario.

```
<!-- Imagen para 1x, para 2x y en el resto de casos-->
    <img srcset="miimagen_1x.png 1x,
                 miimagen_2x.png 2x"
         src="miimagen.png" alt="la mia" />
```

Existe también la posibilidad de detectar el tamaño del viewport de un dispositivo y utilizar dicha información para que un desarrollador pueda suministrar imágenes distintas de manera transparente al usuario.

```
<!-- Imagen para viewports de 512px, para 768px, para 1024px  y en el resto de casos-->
  <img srcset="miimagen_512.png 512w,
               miimagen_768.png 768w,
               miimagen_1024.png 1024w"
       src="miimagen.png" alt="la mia" />
```

Pero dado que la experiencia de usuario no sólo consiste en que no se note el cambio de imagen sino en que el usuario del sitio no se tire dos horas esperando a que se cargue la que toca, hay que afinar también en los formatos de imágenes que se utilizan a la hora de implementar un sitio web. Por conocimientos vistos anteriormente, sabemos que de un tiempo a esta parte han surgido nuevos formatos como AVIF y Webp que evitan tener que bajarse los distintos PNG o JPG según tamaños, densidades y demás. Estos sólo descargan el que se necesita en cada situación, minimizando los tiempos de carga de la página en cuestión.

```
<h1>Imágenes en formatos alternativos según las posibilidades del navegador</h1>
        <picture>
           <source type="image/avif"  srcset="miimagen .avif" />
           <source type="image/webp"  srcset="miimagen .webp" />
            <img src="miimagen .png" />
        </picture>
```

Se ha incluido en el aula virtual una carpeta con un ejemplo de imagen de fondo fluida y unos ejemplos de imágenes *responsive* atendiendo a resoluciones, tamaños y formatos.