TEXT CLASS REVIEW

TEMAS A TRATAR EN LA CUE:

- Tipos de diseño: estático, fluido, adaptativo y responsivo.
- Patrones del diseño responsivo.
- Mobile First.
- Componentes del diseño responsivo.
- Transformación de medidas absolutas a relativas.

Como ya lo hemos revisado en CUEs anteriores, es muy probable que los usuarios visitantes de nuestras creaciones web visualizarán el contenido desde pantallas con resoluciones y tamaños distintos al nuestro, lo cual puede provocar que éste se cargue de una manera que no esperábamos, dificultando su consumo. Por lo tanto, es de gran importancia utilizar los principios del diseño responsivo, para así controlar cómo se visualizará el contenido en distintos tipos de dispositivos.

TIPOS DE DISEÑO

Hasta el momento hemos trabajado con dos: estático y fluido. La diferencia entre ambos radica en el tipo de medida utilizada para escalar las páginas web, siendo los píxeles en el diseño estático, y unidades relativas, como el porcentaje, en el diseño fluido.

- *Diseños estáticos*: éstos se mantendrán fijos en función de un ancho definido en el documento. Esto implica que, si inspeccionamos la página web, nos encontraremos con un ancho ya establecido, y lo mismo ocurrirá con el ancho de los bloques e imágenes.
- *Diseños fluidos*: también conocidos como líquidos, se modificarán según el tamaño del `viewport` utilizando medidas relativas. Se entiende por `viewport` del navegador, el área de la ventana donde el contenido será visible. Esto significa que el tamaño de cada elemento dependerá del ancho del `viewport`, y fluirá en función de este.

DISEÑOS ADAPTATIVOS

Se caracterizan por definir el diseño y disposición de los elementos en función de `media queries`, las cuales determinan dichos parámetros para distintos rangos de tamaños, establecidos por el desarrollador. Es decir, existirán tipos de diseño y disposición de los elementos predefinidos, en función de la resolución de la pantalla donde se visualice el contenido. Éstos utilizan medidas absolutas, lo que permite mayor control en el flujo de la información.

Su principal ventaja, es la posibilidad de establecer distintos diseños para diversos tipos de dispositivos y tamaños, mejorando así la experiencia de navegación del usuario. Por otro lado, tiene algunas desventajas, como el hecho de que no es tan flexible, es decir, solo se ajustará a las resoluciones que hayamos definido. En este sentido, es un poco tedioso en su implementación, ya que requiere mayor tiempo para lograr ajustar el contenido a cada `breakpoint` o punto de quiebre, siendo éstos especialmente delicados, ya que pueden quedar vacíos de resolución y por ende, poco agradables para el usuario.

DISEÑOS RESPONSIVOS

Caracterizados por existir un solo tipo de diseño, que se irá adecuando al tamaño o resolución del dispositivo desde el que navega el usuario. Éstos utilizan unidades relativas y se definen por tres elementos principales.

- *Grillas*: permiten organizar de manera mucho más sencilla la distribución de los elementos en un `layout`. El contenido se organiza en filas y columnas.
- *Imágenes flexibles*: como lo indica su nombre, son imágenes con la característica de fluir a través del diseño, sin perder consistencia.
- *Media Queries*: estas directrices de CSS permiten especificar los estilos que se aplicarán en la resolución determinada por nosotros, dando la posibilidad de agregar colores, formatos y tamaños distintos a las diferentes resoluciones.

El diseño responsivo tiene la ventaja de ser fácil de aplicar, mantener y actualizar, ya que se cuenta con uno que irá ajustando el contenido al sistema de grillas implementado. Los frameworks CSS, como Bootstrap, permiten implementarlos rápidamente, además de tener documentación que nos orientarán en su uso.

Otra de sus ventajas, es que permite al usuario visualizar la misma información, tanto en su computador como en su dispositivo móvil. Por último, los sitios web que utilizan este diseño, cuentan con una carga más rápida, ya que la probabilidad de que exista contenido duplicado es mínima.

Por otro lado, también presenta algunas desventajas: al cargar el mismo contenido, tanto en la versión de escritorio como en la móvil, el tiempo de carga en dispositivos móviles será mayor, y supone un problema al ofrecer soluciones personalizadas para cada tipo de dispositivo. Además, el uso de frameworks CSS, puede significar una complicación en caso de desear mayor personalización del estilo, conllevando aumento de los tiempos en el desarrollo.

DIFERENCIAS ENTRE DISEÑO RESPONSIVO Y ADAPTATIVO

Tal como se ha revisado, se evidencian claras distinciones entre ambos tipos de diseño, las cuales convierten a uno de éstos en el más recomendable.

Primero, el diseño responsivo es más económico que el responsivo, en cuanto a tiempo y personas necesarias para su implementación. Además, el uso de éste le ofrece al usuario una experiencia lo suficientemente buena para navegar sin problemas en un sitio o aplicación web multidispositivo.

En segundo lugar, el diseño responsivo es más popular en el mundo del desarrollo, y cuenta con una gran aceptación en el mundo de la tecnología.

Por lo anterior, se puede entender que el responsivo destaca sobre el adaptativo en implementación y popularidad.

PATRONES DE DISEÑO RESPONSIVO

Existen varios que ya están establecidos y funcionan correctamente en distintos dispositivos de escritorio y móviles. Los principales son: `mostly fluid`, `column drop`, `layout shifter`, `tiny tweaks` y `off canvas`. Cualquiera de estos patrones se puede combinar sin problemas en un mismo sitio web.

Para seleccionar correctamente el tipo o tipos de patrón a utilizar, es importante identificar primero el flujo que tendrá el contenido en el sitio web, al ir disminuyendo el tamaño del `viewport`.

MOSTLY FLUID

Como lo indica su nombre, este patrón fluye la mayor parte del tiempo. Actualmente, es el más utilizado.

En los dispositivos móviles tipo smartphone, se forma una sola columna con filas que generan bloques. Luego, al ir creciendo la pantalla, los bloques se agrupan utilizando todo el espacio disponible. En el caso de dispositivos más grandes, el diseño es el mismo, solo que el contenido se agrupa dentro de un contenedor centrado respecto a la pantalla con un ancho fijo.

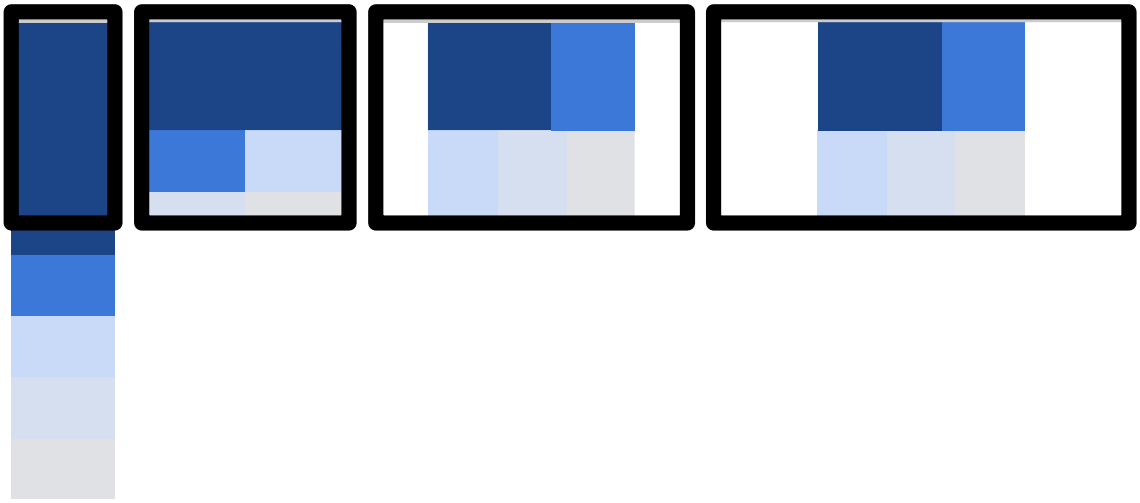

Imagen 1. Patrón mostly fluid.

COLUMN DROP

En éste, cada bloque de contenido se visualizará como una fila en los smartphones, mientras que al ir creciendo el tamaño de la pantalla, éstos se irán acomodando en columnas.

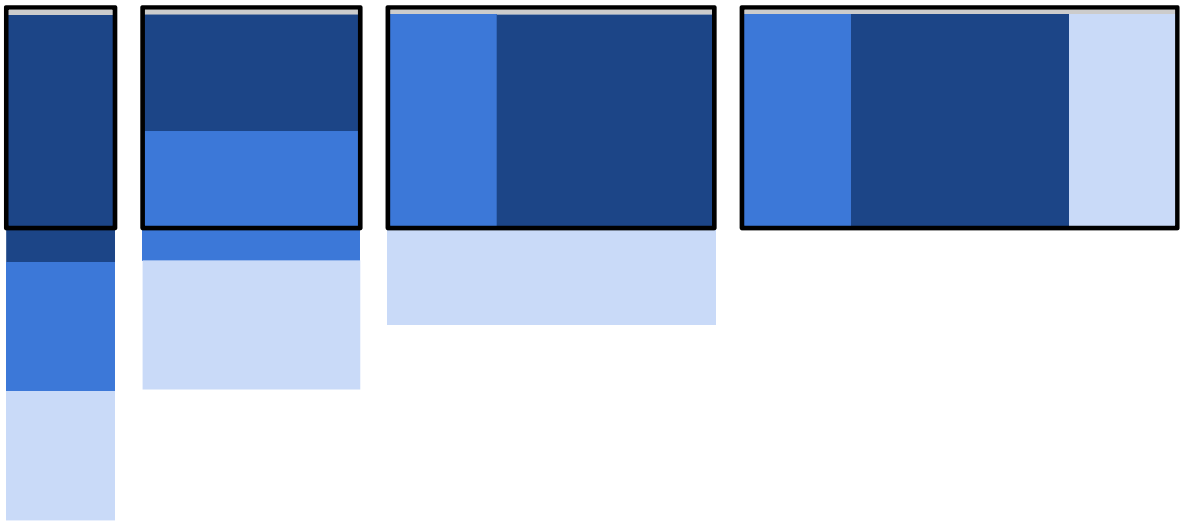

Imagen 2. Patrón column drop.

LAYOUT SHIFTER

Requiere la especificación de distintos puntos de quiebre, para los que se definirá una posición específica del diseño. A pesar de su gran parecido con los diseños adaptativos, sobretodo en la utilización de `breakpoints` en diferentes tamaños, y la colocación de elementos, este tipo de patrón utiliza unidades relativas, siendo el flujo de sus elementos fluido.

En tamaños de pantalla grande, se establece un margen, el cual evita que se desborde el diseño. En el caso de tamaños medianos y pequeños, los elementos se van apilando de manera vertical uno sobre el otro.

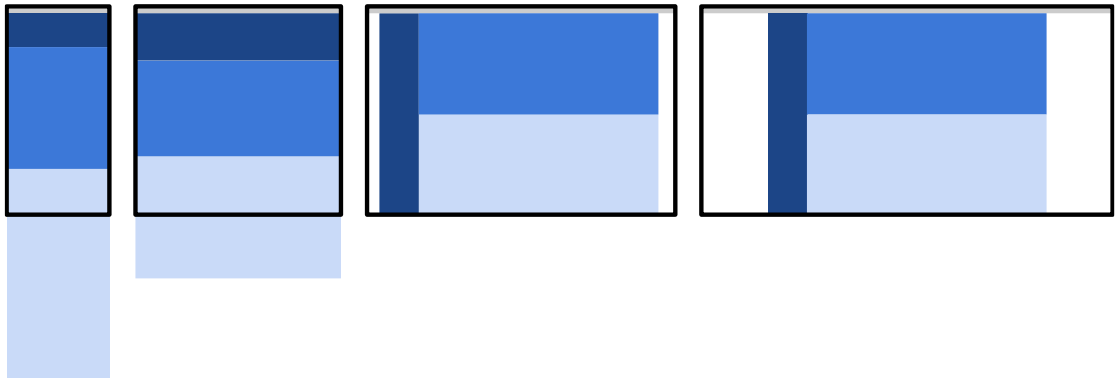

Imagen 3. Patrón layout shifter.

TINY TWEAKS

Se caracteriza por estar basado en una sola columna para el contenido, en cualquier tamaño de pantalla. Éste permite realizar pequeños cambios en el diseño, como: ajustar el tamaño de la fuente, el tamaño de imágenes, o desplazar el contenido de manera poco significativa.

Imagen 4. Patrón tiny tweaks.

OFF CANVAS

Este tipo de diseño se caracteriza por el uso de columnas verticales, que se irán escondiendo al disminuir el tamaño de la pantalla según la importancia del contenido. Es decir, posiciona el contenido menos usado fuera de la pantalla, y solo lo muestra cuando el tamaño de ésta es lo suficientemente grande. En pantallas más pequeñas, dicho contenido es solo accesible con un clic.

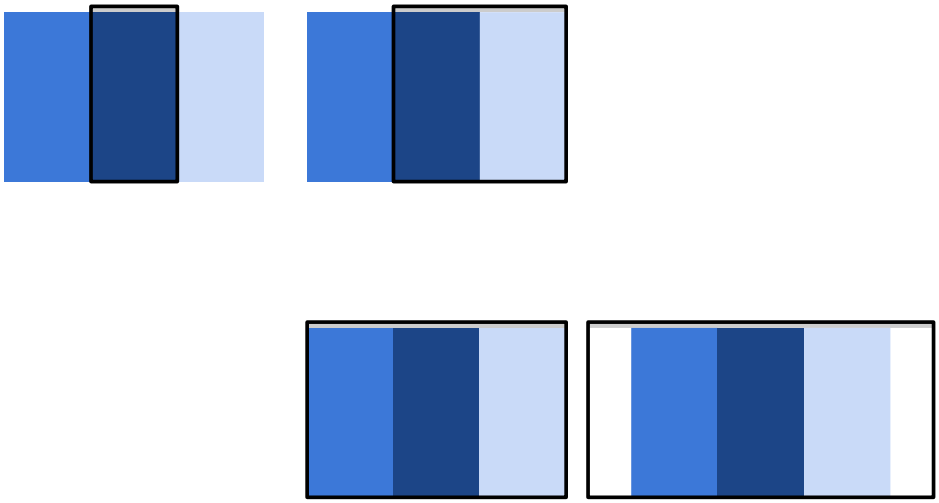

Imagen 5. Patrón off canvas.

MOBILE FIRST

Actualmente, todos contamos con uno (o más) teléfonos inteligentes en nuestro poder y, en la mayoría de los casos, éstos son nuestra primera opción al momento de consumir contenido web. La migración desde los computadores a los dispositivos móviles ha llevado a una transformación en los diseños utilizados a favor de éstos.

Mobile First, corresponde a una filosofía que aprovecha las capacidades de los dispositivos móviles, como: los gestos, el GPS, los sensores, entre otros, para mejorar la experiencia de usuario. Ésta sigue algunas directrices que nos ayudarán a construir mejores maquetas:

- Debe adecuarse a los cambios: el mundo móvil está en constante actualización, tanto en los tamaños de pantallas, sus resoluciones y las capacidades de los dispositivos. Es por esto,

que debemos estar atentos a dicha evolución, y tomar acciones rápidamente para utilizar las nuevas funcionalidades.

- Es importante usar la tecnología a nuestro favor: la implementación de etiquetas HTML, como `meta viewport`, nos permitirá adaptar nuestros sitios a los dispositivos móviles. Es importante además, utilizar las densidades de las pantallas en beneficio de esto, ayudándonos a mejorar la resolución de nuestras páginas.
- El uso del diseño responsivo considerando primero los dispositivos móviles: dentro de esta filosofía, es de gran importancia desarrollar en primer lugar el diseño de nuestro sitio para dispositivos móviles, y posteriormente, ir escalando a resoluciones más grandes.
- Como contraparte de esta filosofía, existe el Desktop First: éste prioriza la experiencia en dispositivos de escritorio. Dependiendo del proyecto en el que estés trabajando, y las características de éste, debes decidir cuál utilizar.
- Se debe homogeneizar la experiencia en dispositivos: primero debemos conocer las características y limitaciones de los dispositivos y, en base a eso, definir la mejor solución, para así poder ofrecer el mismo servicio, pero de distinta manera tanto en dispositivos móviles como de escritorio.
- Hay que priorizar el contenido y funcionalidades que se mostrarán en los dispositivos móviles.

COMPONENTES DE DISEÑO RESPONSIVO

- **Viewport**

  Al referirse a ello, se debe tener en consideración dos elementos: sus tipos de disposición y la etiqueta `<meta name=”viewport”>`.

- **Tipos de disposición**

  Existen dos: la primera está basada en el `viewport` que podemos ver, y la segunda, en el `viewport` por el que hemos diseñado. El visual varía en tamaño, en cambio el de diseño no; lo que implica que el de diseño es más amplio en esencia que el visual. A continuación, un ejemplo:

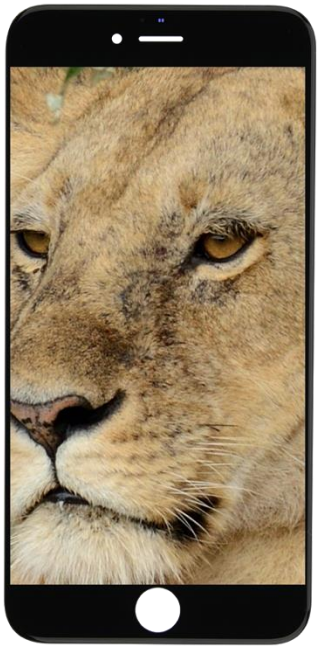

Imagen 6. Ejemplo viewport.

Se pudo observar un ejemplo, donde el tamaño del `viewport` de diseño es mucho mayor que la pantalla del dispositivo en el que se está desplegando, por lo que el navegador reconoce el diseño como más grande, y pareciera que se le ha aplicado zoom a la imagen, observándose solo parte de la cara del león.

Para evitar que esto ocurra, y no se recorte el contenido que queremos mostrar en nuestra página web, debemos utilizar la etiqueta: `<meta name="viewport">`, la cual es imprescindible para la construcción de diseños responsivos.

Ésta permite el correcto control y escalamiento del diseño del `viewport` en distintos tamaños de pantalla. Al ser una etiqueta meta, debemos ubicarla dentro del `<head>`.

- **DIRECTIVAS**

La etiqueta meta permite el uso de distintas directivas adicionales al atributo `name`, las cuales nos permitirán controlar nuestro `viewport`.

```
<head>
   <meta name="viewport"
content="directiva=valor,directive=valor" />
</head>
```

- `width:` permite comunicarle al navegador el ancho que queremos utilizar en el sitio web. Éste puede corresponder a un ancho específico en unidades relativas o absolutas, o podemos indicar que se defina de manera automática, ayudándonos del navegador utilizando la directiva: `width=device-width`.

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

- `height:` permite definir el alto de nuestro sitio web. Al igual que la directiva `width`, se puede establecer un alto específico, o podemos indicar que se haga de manera automática utilizando: `height=device-height`.

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

- `user-scalable:` permite comunicarle al navegador si se le dará o no al usuario, la posibilidad de hacer zoom sobre nuestra página web. Puede tomar los valores: `yes` y `no`.

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

- `initial-scale:` se utiliza para definir el zoom inicial con el que se desplegará el contenido. Sus valores pueden ir desde `0.1(10%)` hasta `10.0(1000%)`, y para declarar un zoom de `100%`, debemos utilizar `initial-scale=1`.

```
<meta name="viewport" content="initial-scale=1" />
```

- `maximum-scale:` permite definir el máximo zoom que el usuario podrá realizar sobre nuestra página web. Al igual que para `initial-scale`, los valores pueden ir desde `0.1` hasta `10`.

```
<meta name="viewport" content="initial-scale=1, maximum-
scale=10"/>
```

- `minimum-scale:` permite establecer el mínimo zoom que tendrá nuestro sitio web. Al igual que con las directivas anteriores, sus valores van de `0.1` a `10`.

```
<meta name="viewport" content="initial-scale=1, minimum-
scale=1" />
```

Como recomendación, es importante construir la etiqueta meta `viewport` de la forma en que se verá a continuación, así podrás definir correctamente, el comportamiento de ésta. Con las siguientes directivas podrás controlar el ancho y el zoom inicial del sitio web:

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

TRANSFORMACIÓN DE MEDIDAS ABSOLUTAS A RELATIVAS

Como se ha revisado en CUEs anteriores, las unidades absolutas corresponden a aquellas que no se escalarán en función del tamaño de fuente del documento, por ejemplo: los pixeles. Por otro lado, las unidades relativas son las que cambian su tamaño en función del tamaño de fuente, como el caso de `rem` y `em`, o en función del tamaño del `viewport`, como `%` y `vw`.

Transformar `px` a unidades relativas al tamaño de fuente

Para hacer la transformación, solo debemos saber el tamaño original del elemento a transformar, y el contexto de la unidad. Por ejemplo: el tamaño de fuente del documento.

$$\frac{objetivo}{contexto} = resultado$$

Transformemos primero un título con tamaño `40px` a `rem`. Para lograrlo, solo debemos dividir los `40px` por el de la fuente base del documento, que en este caso valdrá `16px`.

$$\frac{40px}{16px} = 2.5rem$$

Así obtendremos que la nueva medida en unidades relativas del título será `2.5rem`.

Si también quisiéramos realizar la misma transformación a `em`, debemos conocer el tamaño de fuente definido por el elemento padre.

Transformar `px` *a unidades relativas al* `viewport`

Al igual que en el caso anterior, es posible transformar pixeles a unidades basadas en el ancho del `viewport`. Para lograr esto, necesitamos utilizar la siguiente ecuación:

$$100 * \frac{objetivo}{contexto} = resultado$$

Revisemos este ejemplo: debemos transformar el ancho de un contenedor, definido en pixeles e igual a `500px`, a porcentaje `%`. El ancho máximo del diseño fue establecido en `1180px`.

$$100 * \frac{500}{1180} = 42.373\%$$

Así, obtenemos que el ancho equivalente del contenedor será de aproximadamente `42,373%`.

Es importante destacar, que en el caso de querer transformar a `vw`, se debe seguir el mismo procedimiento, ya que `1vw` es igual a `1%` del ancho del `viewport`.

MEDIA QUERIES

Como lo mencionamos anteriormente, éstas permiten manejar los cambios de nuestro sitio web en diferentes tamaños de dispositivos. Corresponden a una directiva de CSS, que permite informarle al navegar cómo se deben comportar nuestras clases bajo determinadas condiciones, ejemplo: el ancho del dispositivo en el caso del diseño responsivo. Es decir, permiten agregar estilos específicos a ciertos tamaños de pantalla, solo si la condición es cierta. Como muestra: si el navegador detecta que el tamaño de la pantalla del dispositivo es de `300px`, y existe una media query que cubre dicho tamaño, éste cargará el bloque de estilos correspondiente.

DIRECTIVA @MEDIA

Forma parte de la estructura básica de un `media query`, y que además contiene el tamaño desde/hasta el que se aplicará y el bloque de estilos asignado. A continuación, se muestran los `breakpoints` de dispositivos típicos según W3Schools.

```
/* Extra small devices (phones, 600px and down) */

@media only screen and (max-width: 600px) {
        ...
}


/* Small devices (portrait tablets and large phones, 600px and up) */

@media only screen and (min-width: 600px) {
        ...
}


/* Medium devices (landscape tablets, 768px and up) */

@media only screen and (min-width: 768px) {
        ...
}


/* Large devices (laptops/desktops, 992px and up) */

@media only screen and (min-width: 992px) {
```

```
        ...
}


/* Extra large devices (large laptops and desktops, 1200px and up) */

@media only screen and (min-width: 1200px) {
        ...
}
```

Además de la estructura básica presentada, se pueden agregar algunos parámetros para hacerlo más específico.

- **Tipos de medio:** permite especificar el tipo de medio al que afectarán nuestros estilos. Por ejemplo, si queremos que nuestro media `query` afecte solo a pantallas, debemos utilizar la palabra `screen`. En caso contrario, si queremos que solo afecte a las impresiones, se emplea la palabra `print`.
  Los más comunes son: `all`, `screen` y `print`.

```
@media screen {
        p {
                font-size: 1.5rem
        }
}

@media screen {
        p {
                font-size: 3rem
        }
}
```

- **Expresiones:** nos permiten utilizar alguna característica del dispositivo para activar o desactivar un `media query`. Por ejemplo, si queremos que en dispositivos con pantallas pequeñas (ancho bajo `350px`), el color de un texto cambie.

```
@media screen (max-width: 350px) {
        p {
                color: red
        }
}
```

- **Reglas lógicas:** corresponden a otras palabras claves que nos permitirán agregar más características. Por ejemplo:
  La palabra `and` permite agregar más de dos tipos de medio o expresiones.

```
@media screen and (min-width: 200px) and (max-width: 500px)
{
        h1 {
                color: green
        }
}
```

La palabra `only` permite prevenir que navegadores antiguos no soporten nuestra `media queries`.

```
@media only screen and (min-width: 200px) and (max-width:
500px) {
        h1 {
                color: green
        }
}
```