# HTML - SASS @media

Responsive Web Design con Preprocesadores

# Intro

Hemos visto previamente el uso de SASS. Vamos a demostrar como se puede usar SASS también para hacer RWD de una manera muy sencilla.

La técnica estaría basada en tres conceptos clave:

- Uso de la tipografía y espaciado de manera proporcional. Para ello nos centramos en el tamaño básico de la fuente.
- Uso de CSS-Grid para poder trabajar con elementos generales
- Uso de flex-direction para controlar el posicionamiento de columnas.

# Caso simple

La tendencia del trabajo en CSS ha sido la de hacer todos los estilos, y una vez hechos estos estilos ya fuera para Desktop (Desktop-first approach) o ya para dispositivos móviles (Mobile-first approach), se creaban los estilos necesarios para adaptar el resto de pantallas.

Además de esto se tendía a agrupar todos los estilos a modificar en una media query.

Ahora ha cambiado un poco

```
h1 {
}
.project {
}
.sidebar {
}
// etc...

@media screen and (max-width: 600px) {
    h1 {
    }
    .project {
    }
    .sidebar {
    }
    // etc...
}
```

# Caso simple

Siempre conviene hacer CSS ya pensando en cómo va a ser en cualquier dispositivos, con tamaños y medidas relativas y haciendo layouts lo más fluidos posible.

Y luego cuando tengamos que sobreescribir estilos hacerlo con la etiqueta CSS @media.

En SASS se hace anidando la etiqueta al estilo. Veamos un ejemplo.

```
html {
  font-size: 16px;

  @media screen and (max-width: 700px) {
    font-size: 17px;
  }
}
```

# Componentizar lo máximo posible

Esto implica que las adaptaciones a diferentes dispositivos las vamos a ir haciendo componente por componente, o grupo por grupo de estilos. Lo que obliga a agrupar estos estilos de manera inteligente.

El caso anterior nos permite centrarnos en modificar la tipografía básica de toda la página.

Pero por ejemplo si tenemos un layout general creado con CSS-grid, conviene separarlo y crear las adaptaciones necesarias.

```
.grid_container {
  display: grid;
  grid-gap: 1rem;
  grid-template-columns: repeat(12, 1fr);

  @media screen and (max-width: 700px) {
    grid-gap: 1rem;
    grid-template-columns: repeat(9, 1fr);
  }
  @media screen and (max-width: 600px) {
    grid-gap: 0.75rem;
    grid-template-columns: repeat(6, 1fr);
  }
  @media screen and (max-width: 420px) {
    grid-gap: 0.5rem;
    grid-template-columns: repeat(4, 1fr);
  }
}

.block_in_grid {
  grid-area: 1 / 2 / span 1 / span 3;
  // ... demás estilos

  @media screen and (max-width: 700px) {
    grid-area: 1 / 3 / span 1 / span 4;
  }
  @media screen and (max-width: 600px) {
    grid-area: 1 / 2 / span 1 / -1;
  }
  @media screen and (max-width: 420px) {
    grid-area: 1 / 4 / span 1 / -1;
  }
}
```

# Se pueden usar variables y mixins

Ya que estamos usando SASS, podemos hacer uso de sus características como el uso de variables y mixins. Las variables ya las conocemos y nos ayudan a almacenar los breakpoints por ejemplo.

Los mixins son algo nuevo, pero básicamente un mixin es un conjunto de propiedades de CSS o de reglas que se pueden agrupar –sería como una función para SASS-, veamos un ejemplo de mixin.

El uso de mixins viene explicado en este link: https://scotch.io/tutorials/how-to-use-sass-mixins

```
@mixin important_text($color) {
    font-weight: bold;
    color: $color;
    text-decoration: underline;
}

span.important {
    @include important_text(#444);
}
```

```
span.important {
    font-weight: bold;
    color: #444;
    text-decoration: underline;
}
```

# Se pueden usar variables y mixins

Un mixin que podemos usar para hacer media queries sería así:

```scss
@mixin media($bp) {
    @media screen and (max-width: #{$bp}) {
        @content;
    }
}

html {
    font-size: 16px;

    @include media(600px) {
        font-size: 17px;
    }
    @include media(440px) {
        font-size: 18px;
    }
}
```

```scss
html {
    font-size: 16px;
}
@media screen and (max-width: 600px) {
    html {
        font-size: 17px;
    }
}
@media screen and (max-width: 440px) {
    html {
        font-size: 18px;
    }
}
```