

Mini proteyecto UT3

Preporcesadores

Autor:Kostadin Kirilov

Asignatura:Diseño de interfaces web

Indice

1.- Uso de estilos (UT2).....	5
1.1.- Contadores.....	5
1.2.- Prioridades a la hora de aplicar reglas.....	8
1.3.- Flexbox.....	8
1.3.1.- Flex-direction:.....	10
1.3.1.1.- Column:.....	11
1.3.1.2.- Column-reverse.....	11
1.3.1.3.- Row:.....	12
1.3.1.4.- Row:-revesese.....	12
1.3.2.- Flex-wrap:.....	13
1.3.2.1.- Wrap:.....	14
1.3.2.2.- Nowrap:.....	14
1.3.2.3.- Wrap-revese:.....	15
1.3.3.- Flex-flow:.....	15
1.3.3.1.- Flex-flow: row wrap;.....	16
1.3.4.- Justify-content.....	17
1.3.4.1.- flex-start.....	17
1.3.4.2.- Flex-end.....	17
1.3.4.3.- Space-around.....	18
1.3.4.4.- Space-between.....	18
1.3.5.- Align-items:.....	19
1.3.5.1.- center.....	19
1.3.5.2.- flex-start.....	19
1.3.5.3.- flex-end.....	20
1.3.5.4.- stretch.....	20
1.3.5.5.- baseline.....	21
1.4.- align-self.....	22
1.5.- Elementos hijos:.....	23
1.5.1.- Order:.....	23
1.5.2.- Flex-grow:.....	24
1.5.3.- Flex-shrink:.....	24
2.- Media Queries.....	25
2.1.- Ejemplos de Media Queries:.....	25
2.1.1.- Ejemplo 1:.....	25
2.1.2.- Ejemplo 2:.....	26
2.2.- Atributos importantes dentro de las Media Queries:.....	28

2.2.1.- Ejemplo: menú responsive.....	29
2.3.- Atriburos:.....	30
3.- Preprocesadores CSS (UT3).....	31
3.1.- El concepto de preprocesador.....	31
3.2.- Preprocesadores más extendidos.....	31
3.3.- Instalación de Sass.....	31
3.3.1.- Mediante Node.js.....	33
3.4.- Primeros pasos en Sass.....	33
3.4.1.- archivo scss.....	33
3.5.- Variables en Sass.....	35
3.5.1.- variables en scss:.....	35
3.5.2.- resultado en css:.....	35
3.5.3.- html.....	36
3.6.- Anidaciones.....	37
3.6.1.- Selectores hijos:.....	37
3.6.2.- Selectores descendientes:.....	39
3.7.- Importaciones.....	42
3.8.- Operadores.....	45

1.- Uso de estilos (UT2)

1.1.- Contadores

Los contadores son variables en CSS que nos pueden ayudar a la hora de organizar los contenidos de la web y mostrar listas personalizadas con gran detalle a la hora de enumerar sus contenidos.

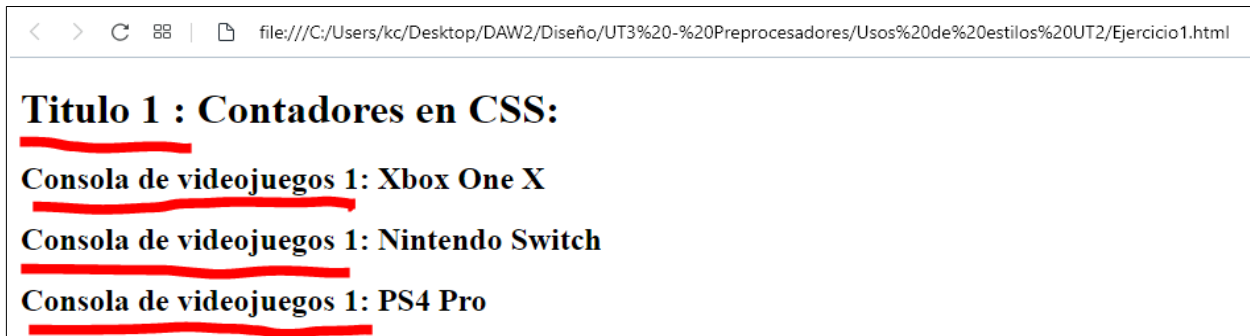
Tenemos varias propiedades para utilizar este tipo de variables:

- **counter-reset** – Crea o reinicia el valor de un contador.
- **counter-increment** – Incrementa el valor de un contador.
- **content** – Inserta un contenido personalizado por el desarrollador.
- **counter()** – Añade el valor de un contador a un elemento.

Ejemplo: en este caso vamos a crear una variable consola para la página principal en el body. Cada vez que aparezca un elemento h2 incrementaremos el contador y además mostraremos un contenido delante del elemento.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=2">
  <title>Web de pruebas</title>
  <style>
    body {
      counter-reset: consola;
      counter-reset: titulo;
    }
    h2::before {
      counter-increment: consola;
      content: "Consola de videojuegos " counter(consola) ": ";
    }
    h1::before {
      counter-increment: titulo;
      content: "Titulo " counter(titulo) " : ";
    }
  </style>
</head>
```

```
<body>
  <h1>Contadores en CSS:</h1>
  <h2>Xbox One X</h2>
  <h2>Nintendo Switch</h2>
  <h2>PS4 Pro</h2>
</body>
</html>
```



Otro ejemplo: creamos un contador para el ámbito de la página y llevar el control de las consolas, y además creamos otro contador para el ámbito de los h1 y llevar el control de los videojuegos asociados a esas consolas. En cada h1 el contador se pone a 0.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body {
      counter-reset: consola;
    }
    h1 {
      counter-reset: videojuegos;
    }
    h1::before {
      counter-increment: consola;
      content: "Consola "counter(consola) ". ";
    }
    h2::before {
      counter-increment: videojuegos;
      content: counter(consola) ". "counter(videojuegos) " ";
    }
  </style>
</head>
<body>
```

```
    }  
  </style>  
</head>  
<body>  
  
  <h1>Xbox One X</h1>  
  <h2>Halo Infinite</h2>  
  <h2>Gears 5</h2>  
  
  <h1>Nintendo Switch</h1>  
  <h2>The Legend Of Zelda: Breath Of The Wild</h2>  
  <h2>Super Mario Odyssey</h2>  
  
  <h1>PS4 Pro</h1>  
  <h2>God Of War</h2>  
  <h2>The Last Of Us 2</h2>  
  
</body>  
</html>
```

< > ↺ 🪟 | 📄 file:///C:/Users/kc/Desktop/DAW2/Diseño/UT3%20-%20Preprocesadores/Usos%20de%20estilos%20UT2/Ejercicio1.html

Consola 1. Xbox One X

1.1 Halo Infinite

1.2 Gears 5

Consola 2. Nintendo Switch

2.1 The Legend Of Zelda: Breath Of The Wild

2.2 Super Mario Odyssey

Consola 3. PS4 Pro

3.1 God Of War

3.2 The Last Of Us 2

1.2.- Prioridades a la hora de aplicar reglas

En ocasiones pueden existir conflictos entre diferentes reglas. Por ejemplo, podemos encontrarnos una clase que establezca un tamaño de fuente y además tener un tamaño de fuente específico para todo el texto del body. En estos casos CSS establece un orden de prioridades a la hora de resolver las reglas.

1,000	100	10	1
inline style	ID	class	element
		attribute	pseudo-element
		pseudo-class	

1.3.- Flexbox

Flexbox es una propiedad esencial a la hora de maquetar o crear layouts en las páginas web. Permite evitar el uso de position o de float, que puede ser muy duro de implementar para el desarrollador. Esta propiedad nace de la necesidad de facilitar el maquetado, sobre todo a nivel horizontal, de los elementos de una página web.

Existen diferentes maneras de aplicar flexbox:

- **Block**, para las secciones de una página web.
- **Inline**, para el texto.
- **Table**, para tablas de dos dimensiones (filas y columnas).
- **Positioned**, para especificar exactamente dónde queremos que aparezca un elemento.

Lo primero que necesitamos es definir un contenedor de tipo flex – Utilizará la propiedad display:flex

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .flexcontenedor {
      display: flex;
      background-color: DodgerBlue;
    }
    .flexcontenedor > div { /*Selector hijo. Hijos inmediatos.*/
      background-color: #f1f1f1;
      margin: 10px;
      padding: 20px;
      font-size: 30px;
    }
  </style>
</head>
<body>
  <div class="flexcontenedor">
    <div>Elemento 1</div>
    <div>Elemento 2</div>
    <div>Elemento 3</div>
  </div>
</body>
</html>
```



1.3.1.- Flex-direction:

Indicará la dirección en la que se apilarán los elementos. Tenemos varias opciones:

```
<!DOCTYPE html>
<html>

<head>
  <style>
    .flexcontenedor {
      display: flex;
      background-color: DodgerBlue;
      flex-direction: column;
    }
    .flexcontenedor > div { /*Selector hijo. Hijos inmediatos.*/
      background-color: #f1f1f1;
      margin: 10px;
      padding: 20px;
      font-size: 30px;
    }
  </style>
</head>

<body>

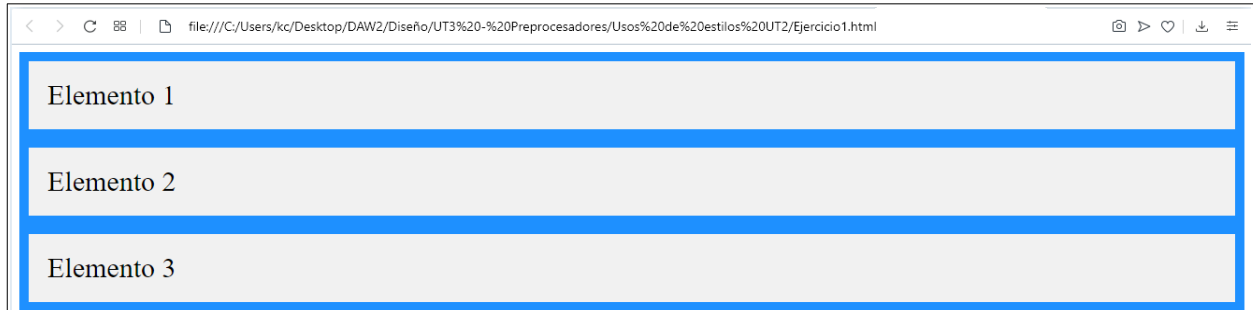
  <div class="flexcontenedor">
    <div>Elemento 1</div>
    <div>Elemento 2</div>
    <div>Elemento 3</div>
  </div>
</body>

</html>
```

1.3.1.1.- Column:

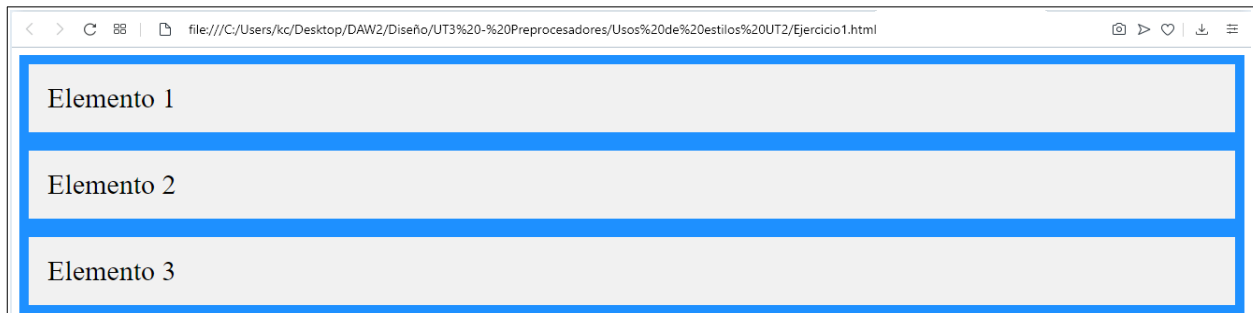
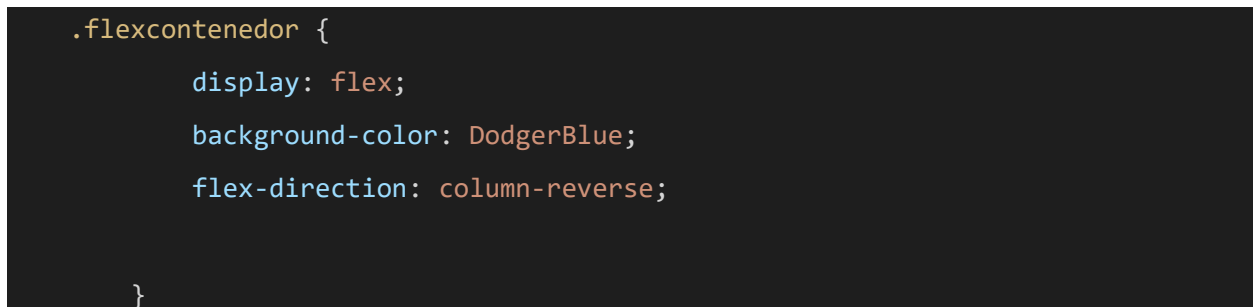
Apila los elementos verticalmente y en orden natural.

Ejemplo:



1.3.1.2.- Column-reverse

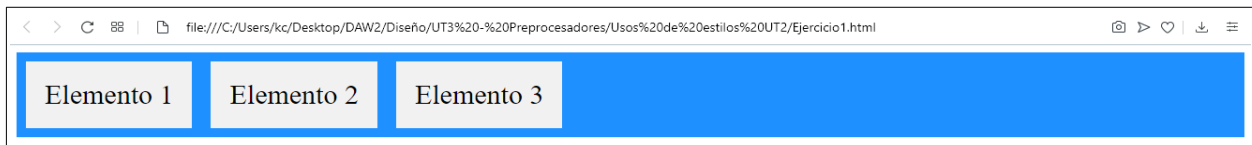
Apila los elementos verticalmente y en sentido inverso.



1.3.1.3.- Row:

Apila los elementos horizontalmente de izquierda a derecha.

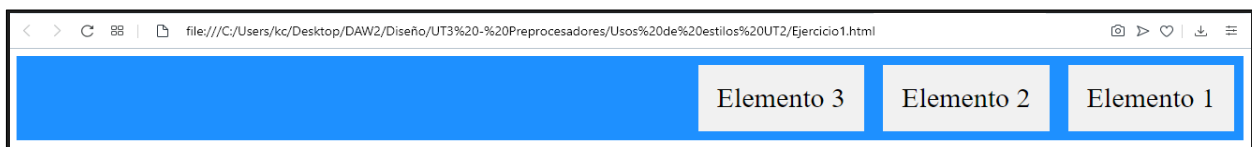
```
.flexcontenedor {  
    display: flex;  
    background-color: DodgerBlue;  
    flex-direction: row;  
}
```



1.3.1.4.- Row:-revesese

Apila los elementos horizontalmente de derecha a izquierda.

```
.flexcontenedor {  
    display: flex;  
    background-color: DodgerBlue;  
    flex-direction: row-reverse;  
}
```



1.3.2.- Flex-wrap:

Permite el ajuste automático cuando hay elementos que no quepan en el ancho del contenedor en el que se encuentren.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .contenedorflex {
      display: flex;
      flex-wrap: wrap;
      background-color: DodgerBlue;
    }
    .contenedorflex > div {
      background-color: #f1f1f1;
      width: 100px;
      margin: 10px;
      text-align: center;
      line-height: 75px;
      font-size: 30px;
    }
  </style>
</head>
<body>
  <div class="contenedorflex">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
    <div>6</div>
    <div>7</div>
    <div>8</div>
    <div>9</div>
```

```
<div>10</div>

<div>11</div>

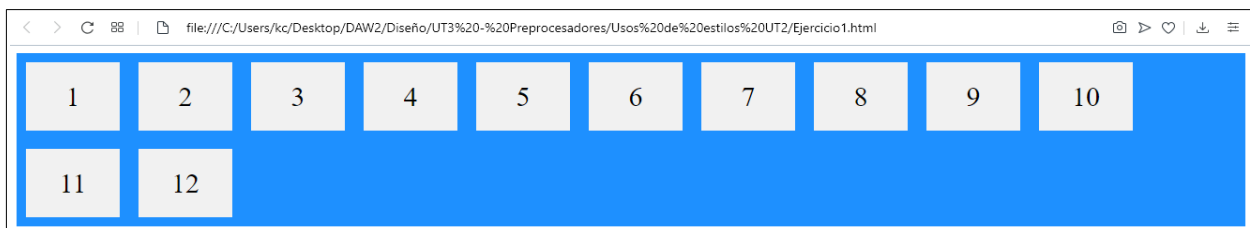
<div>12</div>

</div>
</body>
</html>
```

1.3.2.1.- Wrap:

Los elementos pasan de línea si no caben

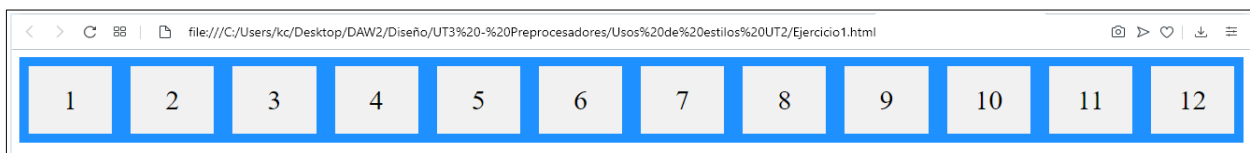
```
.contenedorflex {
    display: flex;
    flex-wrap: wrap;
    background-color: DodgerBlue;
}
```



1.3.2.2.- Nowrap:

Los elementos cambian de ancho automáticamente para adaptarse al ancho del contenedor. Es la opción por defecto.

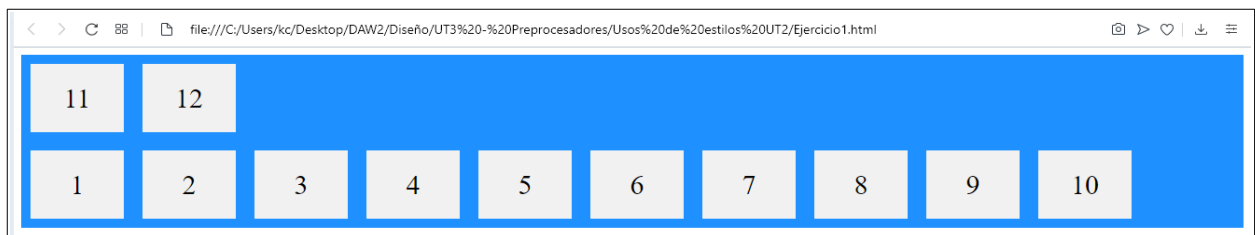
```
.contenedorflex {
    display: flex;
    flex-wrap: nowrap;
    background-color: DodgerBlue;
}
```



1.3.2.3.- Wrap-reverse:

Aplica exactamente lo mismo que wrap, pero en sentido inverso.

```
.contenedorflex {  
    display: flex;  
    flex-wrap: wrap-reverse;  
    background-color: DodgerBlue;  
}
```



1.3.3.- Flex-flow:

Es una forma de reducir código al utilizar las propiedades flex-direction y flex-wrap.

Ejemplo:

```
<!DOCTYPE html>  
<html>  
<head>  
    <style>  
        .contenedorflex {  
            display: flex;  
            flex-wrap: wrap;  
            justify-content: center;  
            background-color: DodgerBlue;  
        }  
        .contenedorflex > div {  
            background-color: #f1f1f1;  
            width: 100px;  
            margin: 10px;  
            text-align: center;  
            line-height: 75px;  
            font-size: 30px;  
        }  
    </style>  
</head>  
<body>  
    <div class="contenedorflex">  
        <div>11</div>  
        <div>12</div>  
        <div>1</div>  
        <div>2</div>  
        <div>3</div>  
        <div>4</div>  
        <div>5</div>  
        <div>6</div>  
        <div>7</div>  
        <div>8</div>  
        <div>9</div>  
        <div>10</div>  
    </div>  
</body>  
</html>
```

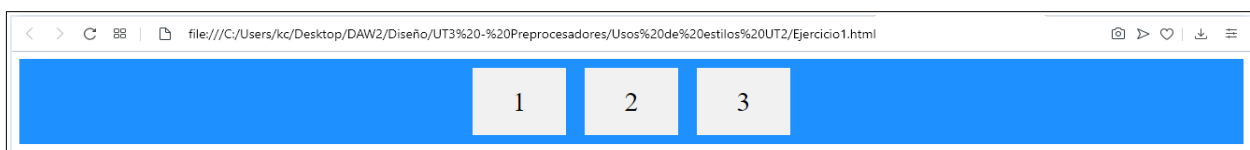
```
</style>
</head>
<body>
  <div class="contenedorflex">
    <div>1</div>
    <div>2</div>
    <div>3</div>
  </div>
</body>
</html>
```

1.3.3.1.- Flex-flow: row wrap;

- Justify-content: permite alinear horizontalmente los elementos que están dentro del contenedor.
- Center: alinea los elementos en el centro.

Ejemplo:

```
.contenedorflex {
  display: flex;
  flex-wrap: row wrap;
  justify-content: center;
  background-color: DodgerBlue;
}
```



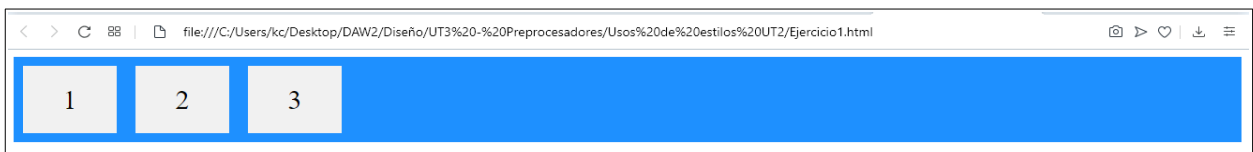
1.3.4.- Justify-content

Justify-content: permite alinear horizontalmente los elementos que están dentro del contenedor.

1.3.4.1.- flex-start

Posiciona los ítems al inicio del contenedor

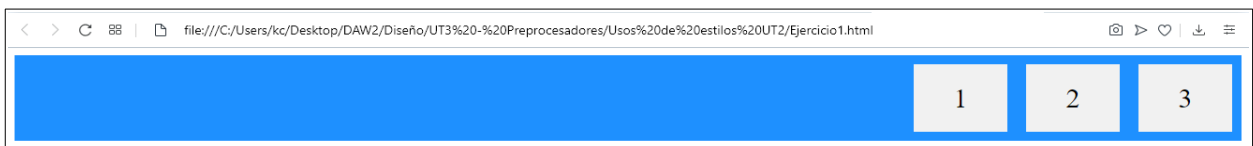
```
.contenedorflex {  
  display: flex;  
  justify-content: flex-start;  
  background-color: DodgerBlue;  
}
```



1.3.4.2.- Flex-end

Posiciona los ítems al final del contenedor

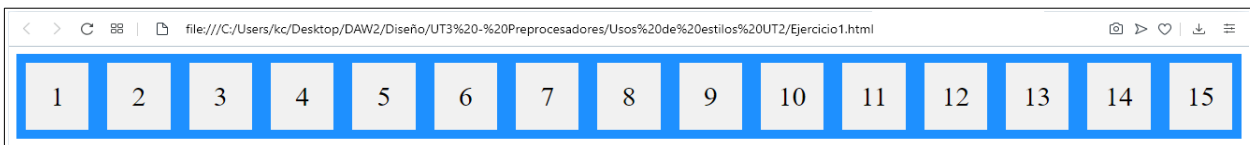
```
.contenedorflex {  
  display: flex;  
  justify-content: flex-end;  
  background-color: DodgerBlue;  
}
```



1.3.4.3.- Space-around

Deja espacio antes, después y entre medias de los ítems. Lo adapta al tamaño de línea (Probar ejemplo con muchos ítems).

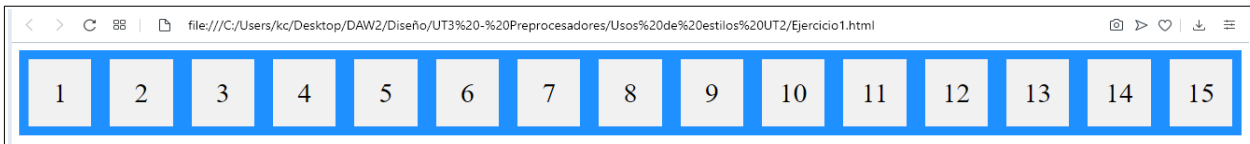
```
.contenedorflex {  
    display: flex;  
    justify-content: space-around;  
    background-color: DodgerBlue;  
}
```



1.3.4.4.- Space-between

Deja espacio entre medias de los ítems. Lo adapta al tamaño de línea (Probar ejemplo con muchos ítems).

```
.contenedorflex {  
    display: flex;  
    justify-content: space-between ;  
    background-color: DodgerBlue;  
}
```



1.3.5.- Align-items:

Es una propiedad que permite alinear verticalmente los elementos de un contenedor. Debemos poner una altura al contenedor.

1.3.5.1.- center

Para centrar los elementos a nivel vertical:

```
.contenedorflex {  
  display: flex;  
  align-items:center;  
  background-color: DodgerBlue;  
  height: 200px;  
}
```



1.3.5.2.- flex-start

Para alinear los elementos arriba del contenedor:

```
.contenedorflex {  
  display: flex;  
  height: 200px;  
  align-items: flex-start;  
}
```



1.3.5.3.- flex-end

Para alinear los elementos abajo del contenedor:

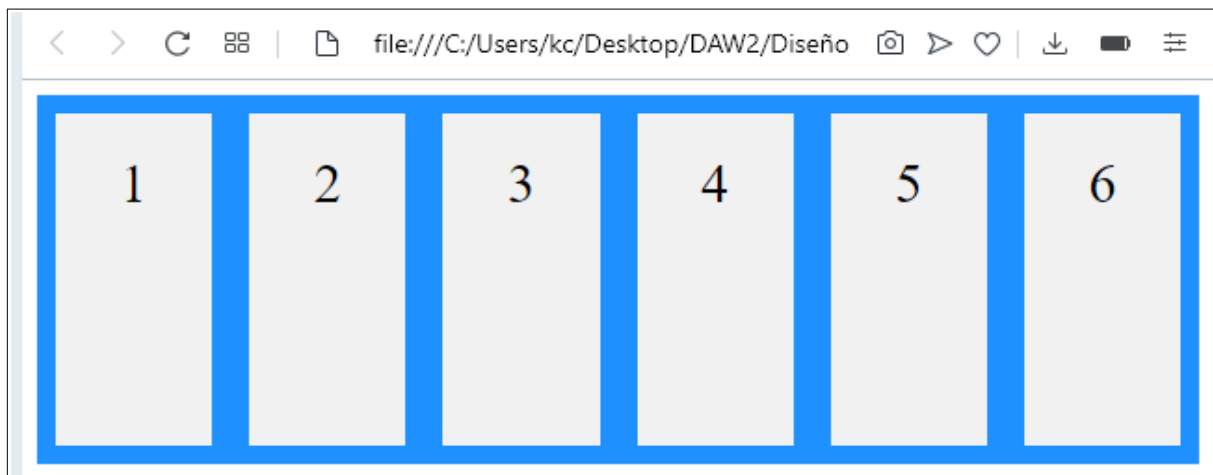
```
.contenedorflex {  
  display: flex;  
  height: 200px;  
  align-items: flex-end;  
}
```



1.3.5.4.- stretch

Para rellenar automáticamente y adaptarse al tamaño del contenedor:

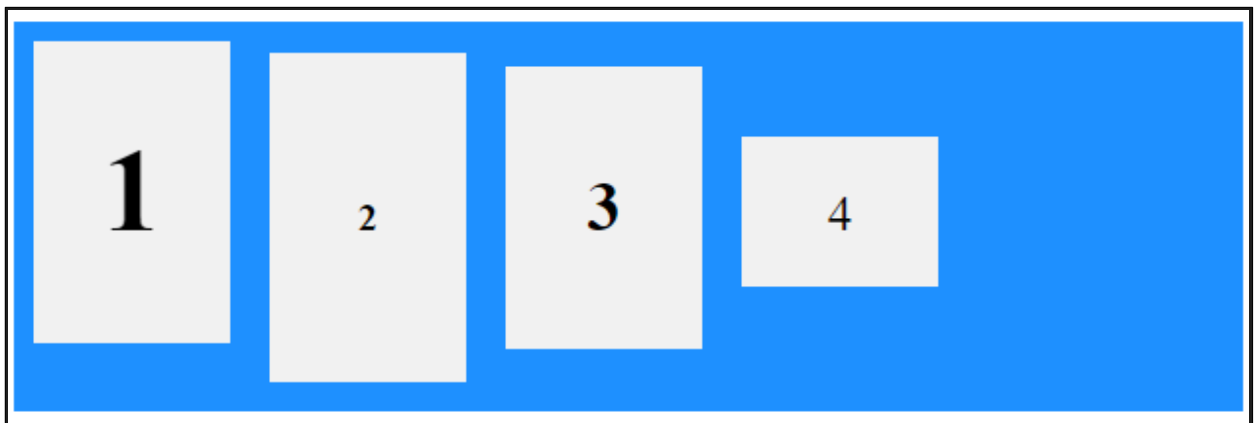
```
.contenedorflex {  
  display: flex;  
  align-items: stretch;  
  background-color: DodgerBlue;  
  height: 200px;  
}
```



1.3.5.5.- baseline

```
.contenedorflex {  
  display: flex;  
  align-items: baseline;  
  background-color: DodgerBlue;  
  height: 200px;  
}
```

```
<body>  
  <div class="contenedorflex">  
    <div><h1>1</h1></div>  
    <div><h6>2</h6></div>  
    <div><h3>3</h3></div>  
    <div><small>4</small></div>  
  </div>  
</body>
```



1.4.- align-self

Html

```
<h2>Ejemplo con align-self</h2>
<div class="contenedorFlex">
  <div class="elementoHijo">Elemento1</div>
  <div class="elementoHijo">Elemento1</div>
</div>
```

SCSS

```
.contenedorFlex{
  display: flex;
  width: 500px;
  background-color: DodgerBlue;
  padding: 10px;
  min-height: 300px;
  >.elementoHijo{
    background-color: #f1f1f1;
    padding: 10px;
    align-self: flex-end;
    margin: 10px;
  }
}
```

CSS

```
.contenedorFlex {
  display: flex;
  width: 500px;
  background-color: DodgerBlue;
  padding: 10px;
  min-height: 300px;
}

.contenedorFlex > .elementoHijo {
  background-color: #f1f1f1;
  padding: 10px;
  align-self: flex-end;
  margin: 10px;
}
```

1.5.- Elementos hijos:

Los hijos directos de un contenedor flex automáticamente se convierten en elementos flex.

Tienen estas propiedades:

1.5.1.- Order:

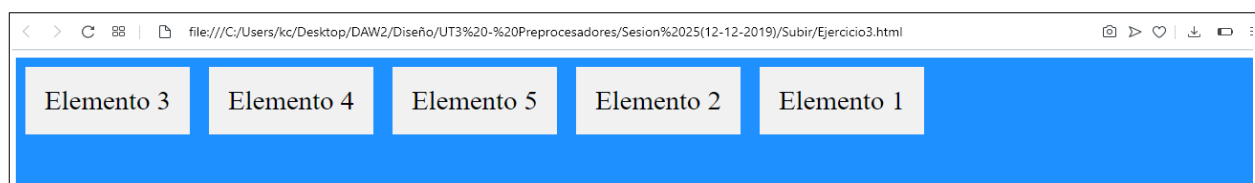
determina el orden de los elementos dentro de un contenedor flex.

html

```
<body>
  <div class="contenedorFlex">
    <div style="order: 5;">Elemento 1</div>
    <div style="order: 4;">Elemento 2</div>
    <div style="order: 1;">Elemento 3</div>
    <div style="order: 2;">Elemento 4</div>
    <div style="order: 3;">Elemento 5</div>
  </div>
</body>
```

SCSS:

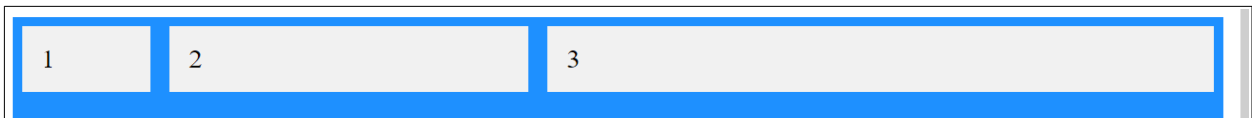
```
%contenedor{
  min-height: 800px;
  background-color: dodgerblue;
  display: flex;
}
%capaHija{
  padding: 20px;
  margin: 10px;
  font-size: 1.8rem;
  background-color: #f1f1f1;
}
.contenedorFlex{
  @extend %contenedor;
  align-items: flex-start;
  >div{
    @extend %capaHija;
  }
}
```



1.5.2.- Flex-grow:

especifica cuánto va a crecer un elemento flex con respecto al resto de elementos flex. Se identificará con un valor numérico.

```
<div class="contenedorFlex">
  <div style="flex-grow: 1;">1</div>
  <div style="flex-grow: 4;">2</div>
  <div style="flex-grow: 8;">3</div>
</div>
```



1.5.3.- Flex-shrink:

Es una propiedad que determina cuánto puede encogerse un elemento flex respecto al resto. Esta propiedad se identificará con un valor numérico.

Ejemplo: el tercer elemento no encogerá nada respecto al resto de elementos.

2.- Media Queries

Las Media Queries son reglas que se utilizan para comprobar parámetros del dispositivo en el que se está abriendo la página web y así poder adaptar la visualización para que sea óptima.

Se pueden utilizar diferentes tipos de Media Queries dependiendo del dispositivo al que vaya dirigida la página web:

- All -> Para todos los tipos de dispositivos.
- Print -> Para impresoras.
- Screen -> Para pantallas, tablets, teléfonos...
- Speech -> Orientado a lectores de pantalla.

Podemos tener una serie de condiciones enlazadas con los operadores **and, not, only y or.**

Ejemplo:

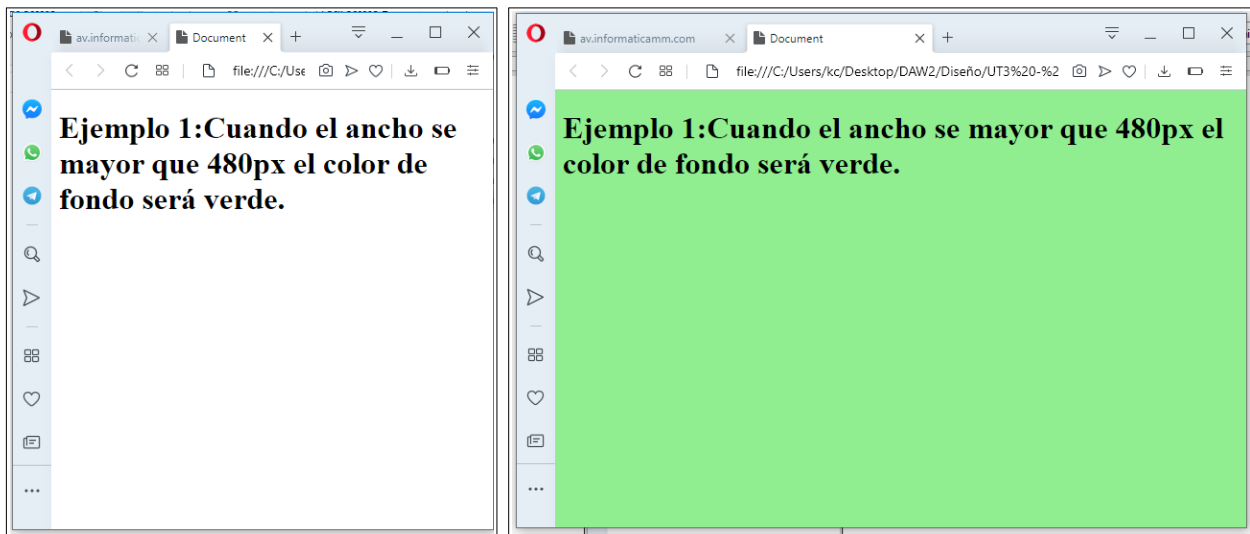
2.1.- Ejemplos de Media Queries:

2.1.1.- Ejemplo 1:

Cuando el ancho sea mayor que 480px el color de fondo será verde.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="HojaEstilos4.css">
    <title>Document</title>
  </head>
  <body>
    <h1>Ejemplo 1: Cuando el ancho sea mayor que 480px el color de fondo será
verde.</h1>
  </body>
</html>
```

```
@media screen and (min-width: 480px) {
  body {
    background-color: lightgreen;
  }
}
```

2.1.2.- Ejemplo 2:

Menú que se sitúa en la parte superior si la anchura es inferior a 480px.

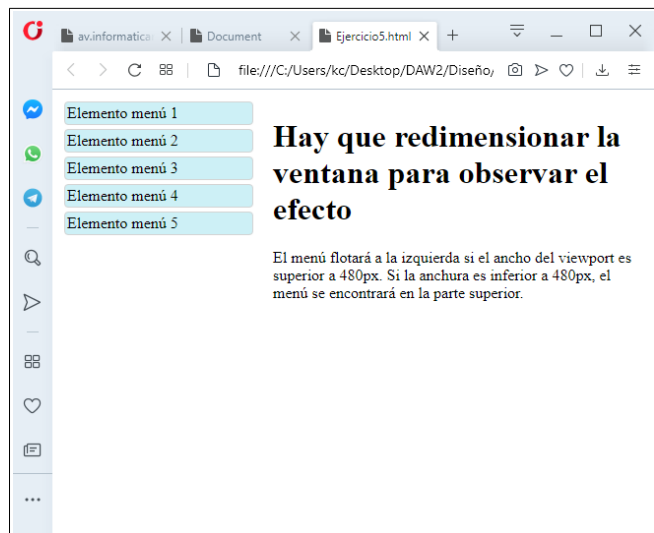
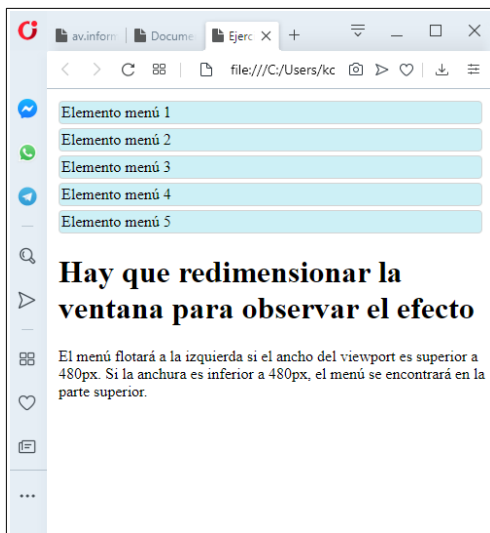
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    .contenedor {
      overflow: auto;
    }
    #principal {
      margin-left: 4px;
    }
    #barralateral {
      float: none;
      width: auto;
    }
    #listamenu {
      margin: 0;
      padding: 0;
    }
  </style>
</head>
<body>
  <div class="contenedor">
    <div id="principal">
      <div id="barralateral">
        <div id="listamenu">
          <ul>
            <li><a href="#">Inicio</a></li>
            <li><a href="#">Servicios</a></li>
            <li><a href="#">Contacto</a></li>
          </ul>
        </div>
      </div>
    </div>
  </div>
</body>
</html>
```

```
.elementomenu {
  background: #CDF0F6;
  border: 1px solid #d4d4d4;
  border-radius: 4px;
  list-style-type: none;
  margin: 4px;
  padding: 2px;
}

@media screen and (min-width: 480px) {
  #barralateral {
    width: 200px;
    float: left;
  }
  #principal {
    margin-left: 216px;
  }
}

</style>
</head>
<body>
  <div class="contenedor">
    <div id="barralateral">
      <ul id="listamenu">
        <li class="elementomenu">Elemento menú 1</li>
        <li class="elementomenu">Elemento menú 2</li>
        <li class="elementomenu">Elemento menú 3</li>
        <li class="elementomenu">Elemento menú 4</li>
        <li class="elementomenu">Elemento menú 5</li>
      </ul>
    </div>
    <div id="principal">
      <h1>Hay que redimensionar la ventana para observar el efecto</h1>
      <p>El menú flotará a la izquierda si el ancho del viewport es superior a 480px. Si la anchura es inferior a 480px,
```

```
    el menú se encontrará en la parte superior.  
  
    </div>  
  </div>  
</body>  
</html>
```



2.2.- Atributos importantes dentro de las Media Queries:

- **Max-width:** el ancho máximo de la zona que se está visualizando, normalmente la ventana del navegador.
- **Min-width:** el ancho mínimo de la zona que se está visualizando, normalmente la ventana del navegador.
- **Min-height:** el alto mínimo de la zona que se está visualizando, normalmente la ventana del navegador.
- **Max-height:** el alto máximo de la zona que se está visualizando, normalmente la ventana del navegador.
- **Orientation:** la orientación del navegador (landscape o portrait). Landscape se da cuando el ancho es mayor que el alto.

2.2.1.- Ejemplo: menú responsive.

2.3.- Atributos:

Existen varios atributos importantes dentro de las Media Queries:

- Max-width: el ancho máximo de la zona que se está visualizando, normalmente la ventana del navegador.
- Min-width: el ancho mínimo de la zona que se está visualizando, normalmente la ventana del navegador.
- Min-height: el alto mínimo de la zona que se está visualizando, normalmente la ventana del navegador.
- Max-height: el alto máximo de la zona que se está visualizando, normalmente la ventana del navegador.
- Orientation: la orientación del navegador (landscape o portrait). Landscape se da cuando el ancho es mayor que el alto.

Más ejemplos con Media Queries:

Ejemplo: menú responsive.

3.- Preprocesadores CSS (UT3)

3.1.- El concepto de preprocesador

Un preprocesador de CSS es una herramienta que nos permite escribir pseudo-código CSS que luego será convertido a CSS real. Ese pseudo-código se conforma de variables, condiciones, bucles o funciones. Podríamos decir que tenemos un lenguaje de programación que genera CSS. El objetivo de estos preprocesadores es tener un código más sencillo de mantener y editar.

3.2.- Preprocesadores más extendidos

Sass: <http://sass-lang.com/>

Less: <http://lesscss.org/>

Stylus: <http://stylus-lang.com/>

PostCSS: <http://postcss.org/>

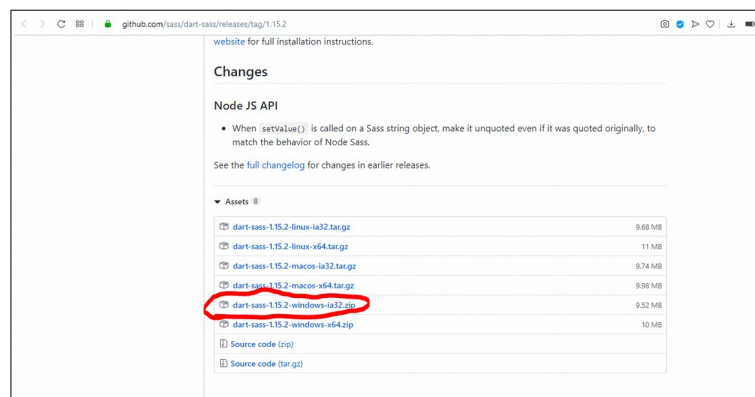
3.3.- Instalación de Sass

Existen varias formas de instalar Sass que se detallan aquí: <http://sass-lang.com/install>

Nos vamos a centrar en Dart Sass, que es la implementación principal de Sass y la primera en recibir todas las características. Tiene un repositorio oficial en Github: <https://github.com/sass/dart-sass>

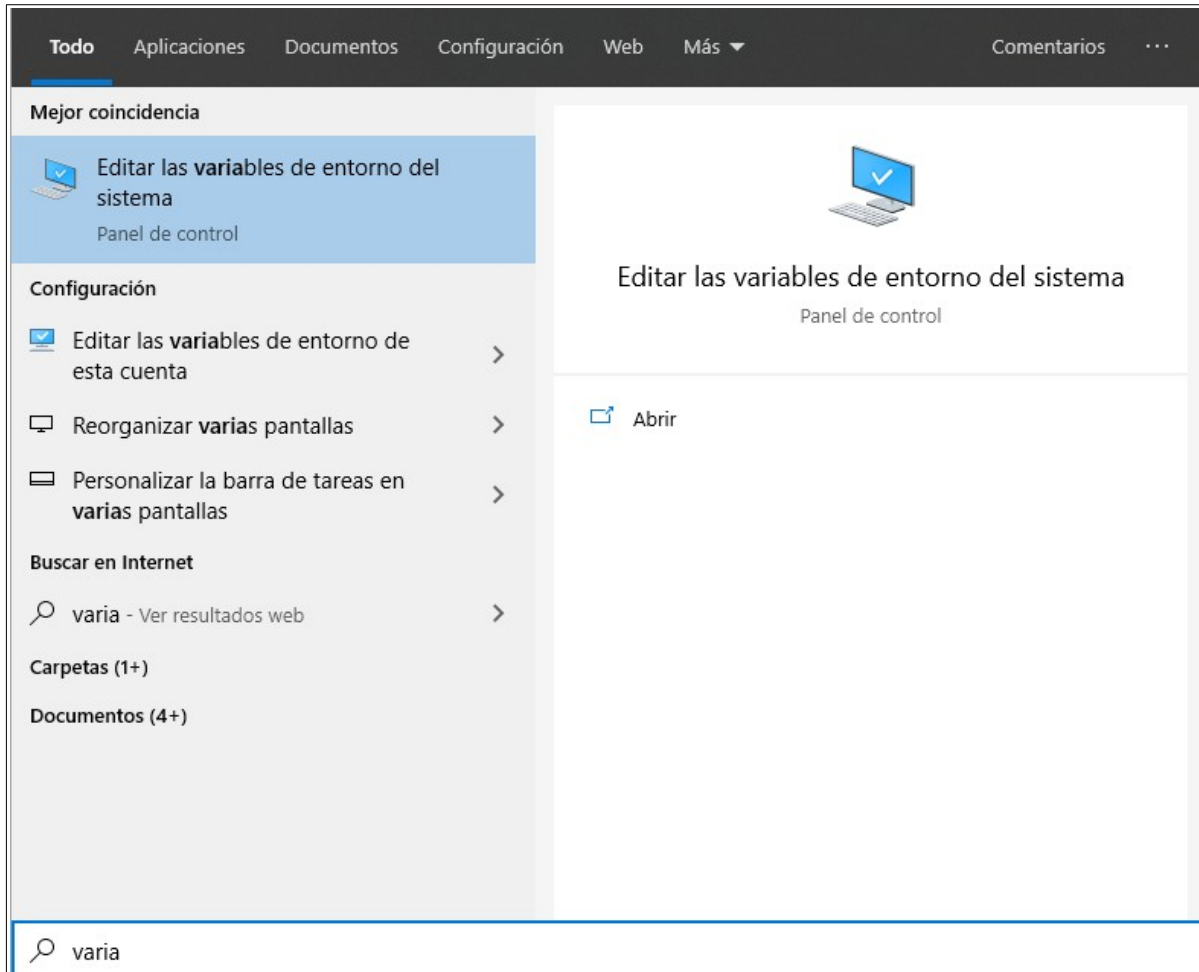
Para realizar la instalación tenemos varias opciones:

1. Instalar Dart Sass desde el repositorio de GitHub: <https://github.com/sass/dart-sass/releases/tag/1.15.2>
- Descargamos la versión que deseemos (es multiplataforma) y descomprimos el paquete en el directorio que queramos.

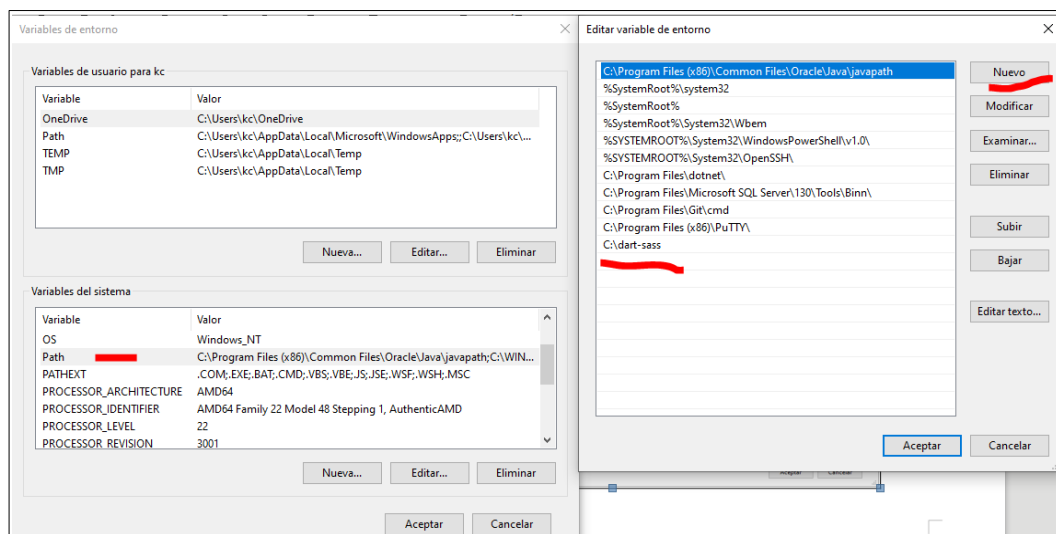


Cambiar por el título del Trabajo.

- Añadimos al PATH el directorio del paquete que acabamos de descargar. Cómo añadir al PATH el directorio: <https://katiek2.github.io/path-doc/>



2. Instalar Dart Sass mediante el instalador (versión Windows)
<http://www.gekorm.com/dart-windows/>



3.3.1.- Mediante Node.js

Instalación en node y posterior instalación de Sass en su versión de pura de JavaScript (es más lenta, pero la interfaz es igual que en el resto):

1. Instalar node: <https://nodejs.org/en/>
2. Abrimos una terminal en Visual Studio Code.
3. Ejecutamos este comando: `npm install -g sass`

Npm es el gestor de paquetes de paquetes de Node.js

Una vez que lo tengamos instalado, podemos lanzar desde una terminal (PowerShell o dentro del propio Visual Studio Code) el siguiente comando para verificar la versión actual de Sass instalada en nuestro sistema: `sass --version`

Nota importante: Ruby Sass se encuentra en proceso de desaparición.

3.4.- *Primeros pasos en Sass*

Los ficheros en Sass pueden tener sintaxis scss, que es la más accesible para comenzar a trabajar con preprocesadores ya que es similar a CSS y supone un menor reto para introducirse en esta tecnología.

Para comenzar, crearemos un fichero “primerficherosass.scss”.

Escribiremos una serie de reglas respetando la sintaxis scss y compilaremos el fichero:

3.4.1.- archivo scss

```
div {  
  color: #00ff00;  
  width: 97%;  
  height:auto;  
  .cajaroja{  
    background-color: #ff0000;  
    color: #000000;  
    width:50%;  
    height:400px;  
    p{  
      color:yellow;  
    }  
  }  
}
```

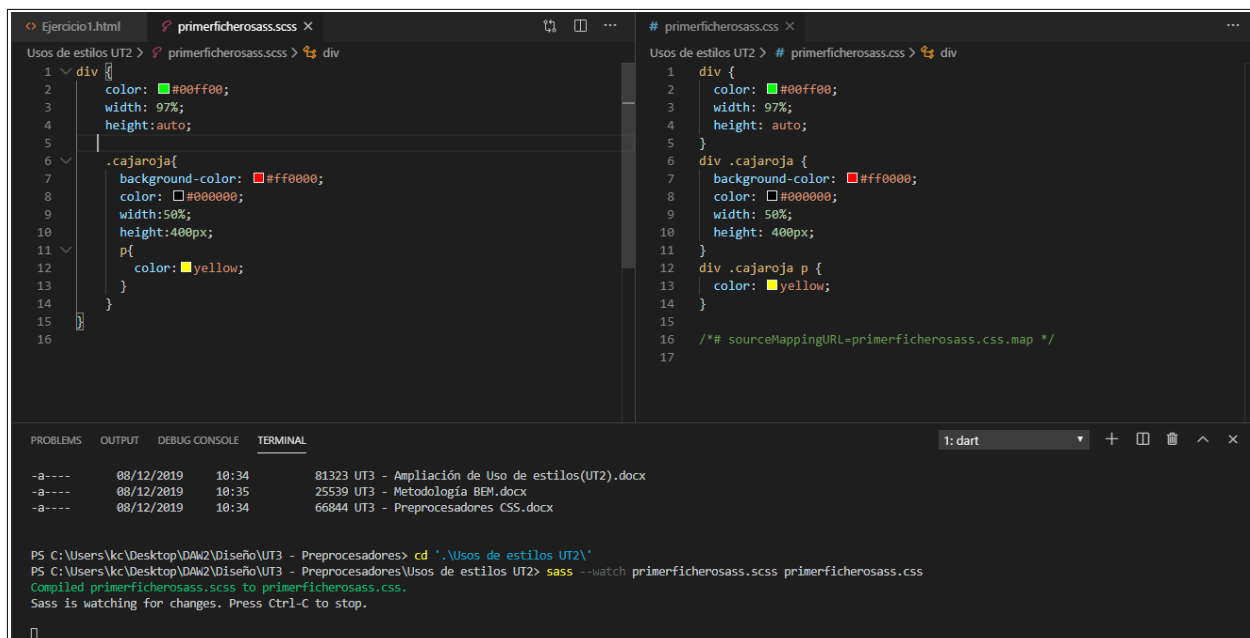
CSS generado


```
div {
  color: #00ff00;
  width: 97%;
  height: auto;
}

div .cajaroja {
  background-color: #ff0000;
  color: #000000;
  width: 50%;
  height: 400px;
}

div .cajaroja p {
  color: yellow;
}

/*# sourceMappingURL=primerficherosass.css.map */
```



¡Cuidado! Sass no admite rutas UNC (Universales) del tipo \\<nombre equipo>\<carpeta compartida>

Compilaremos el fichero scss para obtener el fichero de salida css:

- `sass primerficherosass.scss primerficherosass.css`

También podemos decidir que el fichero se autocompile cada que le demos a guardar:

- `sass --watch primerficherosass.scss primerficherosass.css`

Compilación de una carpeta completa

- `sass -w directorioorigen:directoriodestino`

Ejemplo:

- `sass -w ./carpetaSCSS:./carpetaSCSS`

Importante: los ficheros que comiencen por _ no generarán ficheros de salida al ser compilados.

3.5.- Variables en Sass

Mediante estas variables podemos almacenar colores, fuentes... cualquier valor en CSS que queramos reutilizar. Las variables se declaran con el símbolo \$ delante del identificador que deseemos.

Ejemplos:

3.5.1.- variables en scss:

```
$fuente1: Helvetica, sans-serif;
$colorprincipal: rgb(187, 57, 57);
/*En este ejemplo vamos a aprender una nueva propiedad: font. Sirve para establecer una familia de fuente y un tamaño de texto*/
body{
font: 1em $fuente1;
color: $colorprincipal;
}
```

3.5.2.- resultado en css:

```
@charset "UTF-8";
/*En este ejemplo vamos a aprender una nueva propiedad: font. Sirve para establecer una familia de fuente y un tamaño de texto*/
body {
font: 1em Helvetica, sans-serif;
color: #bb3939;
}
```

3.5.3.- html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <link rel="stylesheet" href="primerficherosass.css">
  </head>
  <body>
    <div class="contenedorflex">
      <h1>Preprocesadores CSS - Variables en scss</h1>
      <p>prueba de variables</p>
    </div>
  </body>
</html>
```



3.6.- Anidaciones

Mediante las anidaciones de reglas en Sass se puede conseguir un código mucho más claro que con CSS3 puro.

Ejemplos de anidaciones de reglas:

3.6.1.- Selectores hijos:

1. html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <link rel="stylesheet" href="HojaEstilos6.css">
  </head>
  <body>
    <h1>Secetrones hijos en sass </h1>
    <div class="flexcontenedor">
      <div>Elemento 1</div>
      <div>Elemento 2</div>
      <div>Elemento 3</div>
      <div>Elemento 4</div>
    </div>
  </body>
</html>
```

2. sass

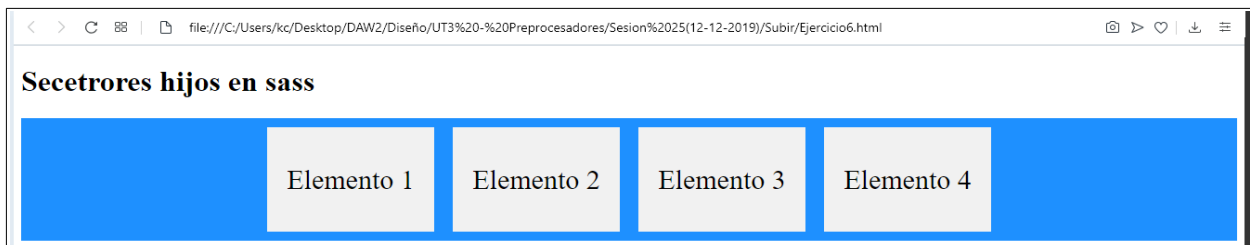
```
.flexcontenedor{
  display:flex;
  background-color:dodgerblue;
  flex-wrap:wrap;
  justify-content: center;
  > div{
    background-color: #f1f1f1;
    width: 140px;
    margin: 10px;
    padding: 20px;
    font-size: 30px;
    line-height: 75px;
```

```
    text-align: center;
  }
}
```

3. css

```
.flexcontenedor {
  display: flex;
  background-color: dodgerblue;
  flex-wrap: wrap;
  justify-content: center;
}
.flexcontenedor > div {
  background-color: #f1f1f1;
  width: 140px;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
  line-height: 75px;
  text-align: center;
}

/*# sourceMappingURL=hojaestilos6.css.map */
```



3.6.2.- Selectores descendientes:

1. Html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <link rel="stylesheet" href="HojaEstilos6.css">
  </head>
  <body>
    <h1>Selectores descendientes:</h1>
    <div class="contenedorflex">
      <div>Elemento 1</div>
      <div>Elemento 2</div>
      <div>Elemento 3</div>
      <div>Elemento 4</div>
    </div>
  </body>
</html>
```

2. sass

```
.contenedorflex{
  display: flex;
  background-color: dodgerblue;
  width: 100%;
  div{
    background-color: #f1f1f1;
    margin: 10px;
    padding: 20px;
    font-size: 30px;
  }
}
```

3. css

```
.contenedorflex {  
  display: flex;  
  background-color: dodgerblue;  
  width: 100%;  
}  
.contenedorflex div {  
  background-color: #f1f1f1;  
  margin: 10px;  
  padding: 20px;  
  font-size: 30px;  
}  
  
/*# sourceMappingURL=hojaestilos6.css.map */
```

Selectores descendentes:

Elemento 1

Elemento 2

Elemento 3

Elemento 4

Otro ejemplo de selector descendiente:

1. html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <link rel="stylesheet" href="HojaEstilos6.css">
  </head>
  <body>
    <h1>Selectores descendentes:</h1>
    <nav>
      <ul>
        <li><a href="https://www.google.es">Google</a></li>
        <li><a href="https://www.kaspersky.es">Kaspersky</a></li>
        <li><a href="https://www.amazon.es">Amazon</a></li>
      </ul>
    </nav>
  </body>
</html>
```

2. sass

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```


3. css

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
  background-color: cornflowerblue;
}
nav li {
  display: inline-block;
  background-color: darkgoldenrod;
}
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
  border: 1px solid black;
}

/*# sourceMappingURL=hojaestilos6.css.map */
```

3.7.- Importaciones

Mediante Sass es posible importar otros ficheros scss más pequeños y más fáciles de mantener. Aquí entra en juego el concepto de “parciales” que pueden ser importados en otros ficheros Sass y que permiten modularizar nuestro proyecto.

Importante:

Los ficheros parciales que comiencen por _ no generarán salida al ser compilados por un comando que compile automáticamente todos los ficheros de una carpeta.

Ejemplo:

Creamos un fichero llamado ‘sassimportar.scss’ con el siguiente contenido:

1. html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <!-- <link rel="stylesheet" href="HojaEstilos6.css"> -->
    <link rel="stylesheet" href="proyectosass.css">
  </head>
  <body>
    <h1>Selectores descendentes:</h1>
    <nav>
```

```
        <ul>
          <li><a href="https://www.google.es">Google</a></li>
          <li><a href="https://www.kaspersky.es">Kaspersky</a></li>
          <li><a href="https://www.amazon.es">Amazon</a></li>
        </ul>
      </nav>
    </body>
  </html>
```

2. scss

sassimportar.scss

```
html,
body,
ul,
ol {
  margin: 0;
  padding: 0;
  letter-spacing: 1em;
}
```

proyectosass.scss

```
@import 'sassimportar.scss';
$fuente1: Helvetica, sans-serif;
$colorprincipal: rgb(187, 57, 57);

/*En este ejemplo vamos a aprender una nueva propiedad: font*/
body{
  font: 1em $fuente1;
  color: $colorprincipal;
}
```

3. css

sassimportar.css

```
html,
body,
ul,
ol {
  margin: 0;
  padding: 0;
  letter-spacing: 1em;
}
```

proyectosass

```
@import 'sassaimportar.scss';
$fuente1: Helvetica, sans-serif;
$colorprincipal: rgb(187, 57, 57);

/*En este ejemplo vamos a aprender una nueva propiedad: font*/
body{
  font: 1em $fuente1;
  color: $colorprincipal;
}
```



3.8.- Operadores

Realizar operaciones matemáticas en CSS puede resultar muy útil.

Operadores aritméticos:

+, -, *, / y %

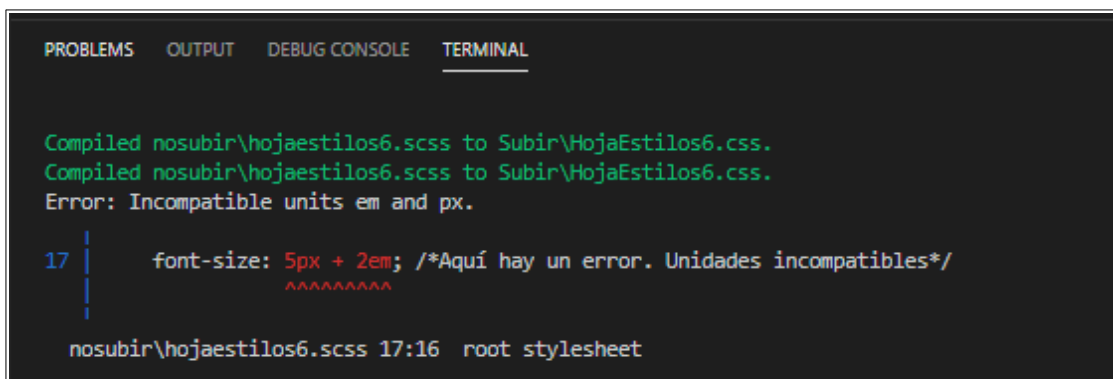
Hay que tener cuidado porque estos operadores aritméticos sólo funcionan con unidades compatibles.

La mejor forma de comprobar todo esto que vamos a hacer es echar un vistazo al css generado mediante la compilación del fichero scss.

Ejemplos:

1. Ejemplo1:

```
h2{
  font-size: 5px + 2em; /*Aquí hay un error. Unidades incompatibles*/
  font-size: 5px + 2; /*El resultado sería 7px*/
}
```



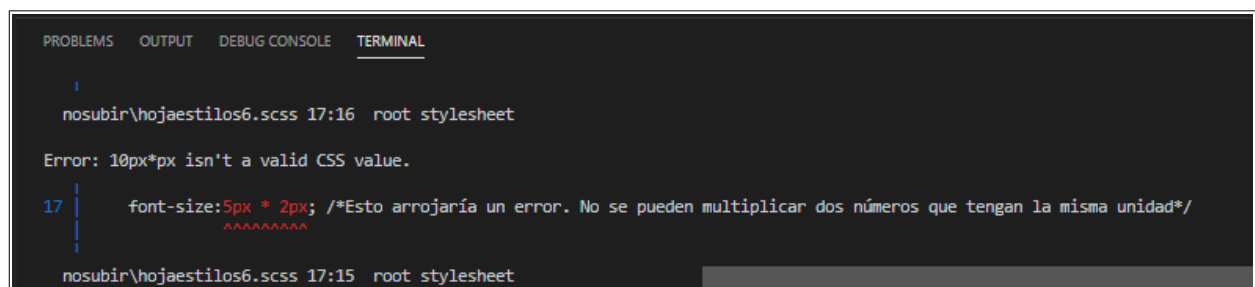
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Compiled nosubir\hojaestilos6.scss to Subir\HojaEstilos6.css.
Compiled nosubir\hojaestilos6.scss to Subir\HojaEstilos6.css.
Error: Incompatible units em and px.

17 |      font-size: 5px + 2em; /*Aquí hay un error. Unidades incompatibles*/
    |                  ^^^^^^^^^
nosubir\hojaestilos6.scss 17:16  root stylesheet
```

2. Ejemplo2:

```
h3{
  font-size:5px * 2px; /*Esto arrojaría un error. No se pueden multiplicar dos
números que tengan la misma unidad*/
  font-size: 5px * 2;
}
```



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

nosubir\hojaestilos6.scss 17:16  root stylesheet

Error: 10px*px isn't a valid CSS value.

17 |      font-size:5px * 2px; /*Esto arrojaría un error. No se pueden multiplicar dos números que tengan la misma unidad*/
    |                  ^^^^^^^^^
nosubir\hojaestilos6.scss 17:15  root stylesheet
```

```
16 | h3 {
17 |   font-size: 5px * 2; /*Esto arrojaría un error. No se pueden multiplicar dos números que tengan la misma unidad*/
18 |   font-size: 5px * 2;
19 | }
20 |
21 |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
nosubir\hojaestilos6.scss 17:16 root stylesheet
Error: 10px*px isn't a valid CSS value.
17 |   font-size: 5px * 2px; /*Esto arrojaría un error. No se pueden multiplicar dos números que tengan la misma unidad*/
   |   ~~~~~
nosubir\hojaestilos6.scss 17:15 root stylesheet
Compiled nosubir\hojaestilos6.scss to Subir\HojaEstilos6.css
```

Otro ejemplo:

Con operadores aritméticos y roles. En este caso vamos a realizar una rejilla fluida con dos elementos dentro de un contenedor y tomando como base 960px de ancho.

Más información sobre los roles en el fichero Roles-ARIA que se encuentra en esta misma carpeta.

Para ampliar esta información, podemos visitar en los siguientes <https://www.w3.org/TR/html-aria/> y http://nosolousabilidad.com/articulos/wai_aria.htm.

En el fichero scss: