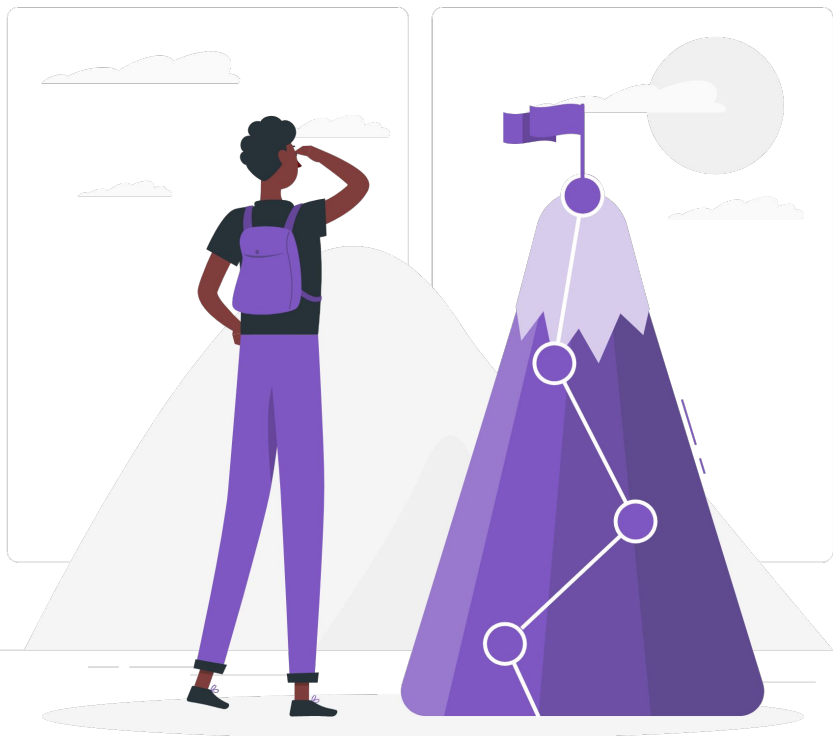




Sass & BEM

DEV.F
DESARROLLAMOS(PERSONAS);



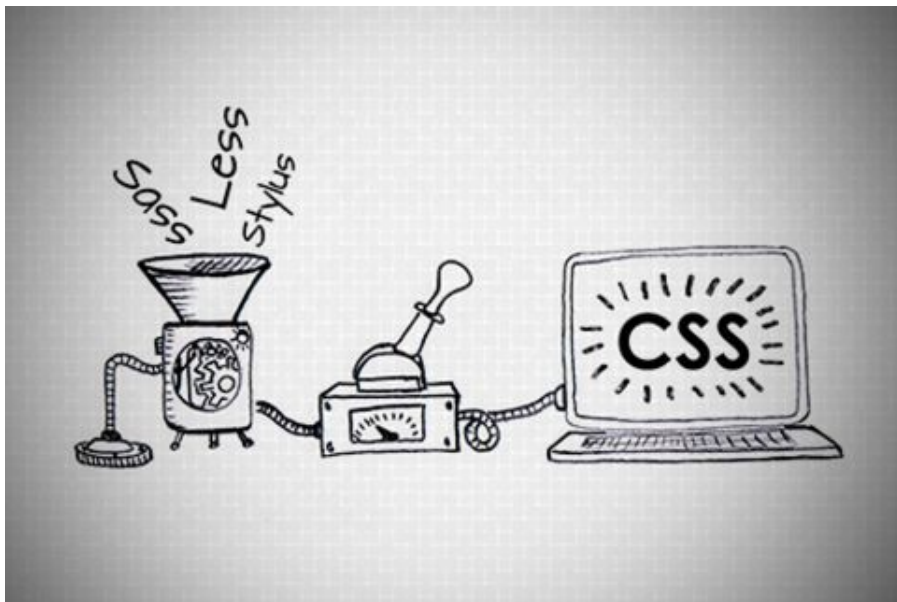
Objetivos

- Aprender que son los Preprocesadores
- Comprender que problemas resuelve SASS
- Diferenciar entre SASS y SCSS
- Aprender a utilizar usar SASS: Sintaxis, Variables, Mixins
- Aprender qué es la metodología BEM
- Aplicar BEM en SASS

Preprocesadores

DEV.F
DESARROLLAMOS(PERSONAS);

dev



Preprocesadores CSS

Un **preprocesador de CSS** se puede definir como **una herramienta que nos permite escribir pseudocódigo CSS que luego será compilado** para convertirlo en CSS tal y como lo conocemos de forma habitual.

Este pseudocódigo está formado por **variables, condicionales, bucles o funciones**, elementos habituales de cualquier lenguaje de programación.

Por este motivo, podríamos decir que tenemos **un lenguaje de programación cuya misión es la de generar el código CSS**



```
$linkcolour: #f60;  
  
a {  
  color: $linkcolour;  
}  
.headers {  
  color: $linkcolour;  
}
```



```
@linkcolour: #f60;  
  
a {  
  color: @linkcolour;  
}  
.headers {  
  color: @linkcolour;  
}
```

P
R
E
P
R
O
C
E
S
S
O
R

Output - CSS

```
a {  
  color: #f60;  
}  
.headers {  
  color: #f60;  
}
```

Beneficios de Preprocesadores CSS

Nuevas funcionalidades: Permiten realizar acciones que no son posibles en CSS tradicionales, como lógica condicional.

Reutilización: Favorecen reusar mucho del código repetitivo de CSS, ahorrando tiempo y esfuerzo.

Cambios rápidos: Al solo cambiar valor de variables o hacer uso de mixins.

Compatibilidad: Facilita la creación de código compatible con múltiples navegadores.



Sass

SASS

DEV.F

DESARROLLAMOS(PERSONAS);

dev



¿Que es Sass?

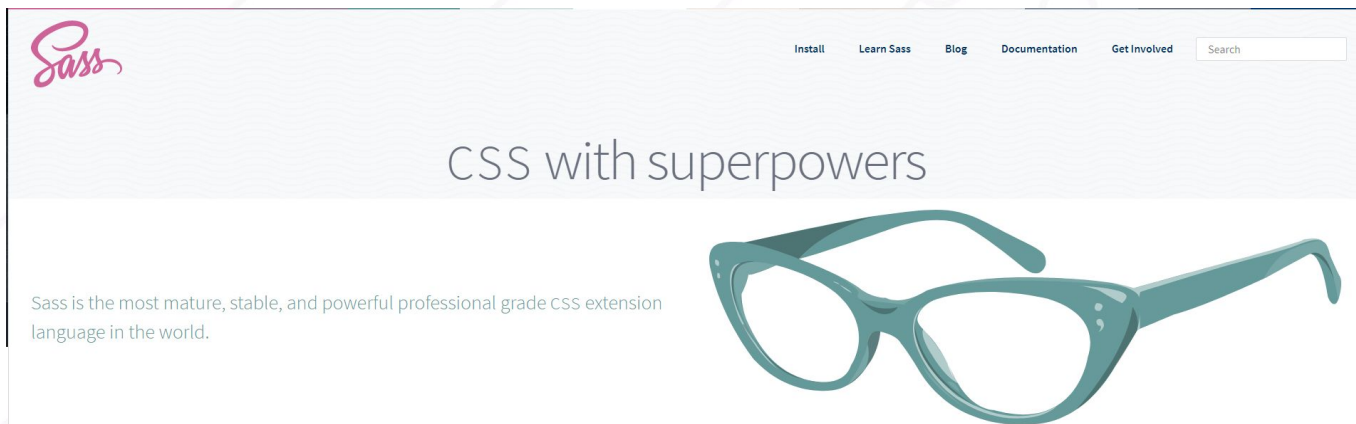
Sass (Syntactically Awesome Stylesheets)

es un lenguaje de hoja de estilos en cascada inicialmente diseñado por Hampton Catlin y desarrollado por Natalie Weizenbaum en 2006.

Sass es un procesador CSS, por lo que es una herramienta que nos permite generar, de manera automática, hojas de estilo, añadiéndoles características que no tiene CSS, y que son propias de los lenguajes de programación, como pueden ser variables, funciones, selectores anidados, herencia, etcétera.

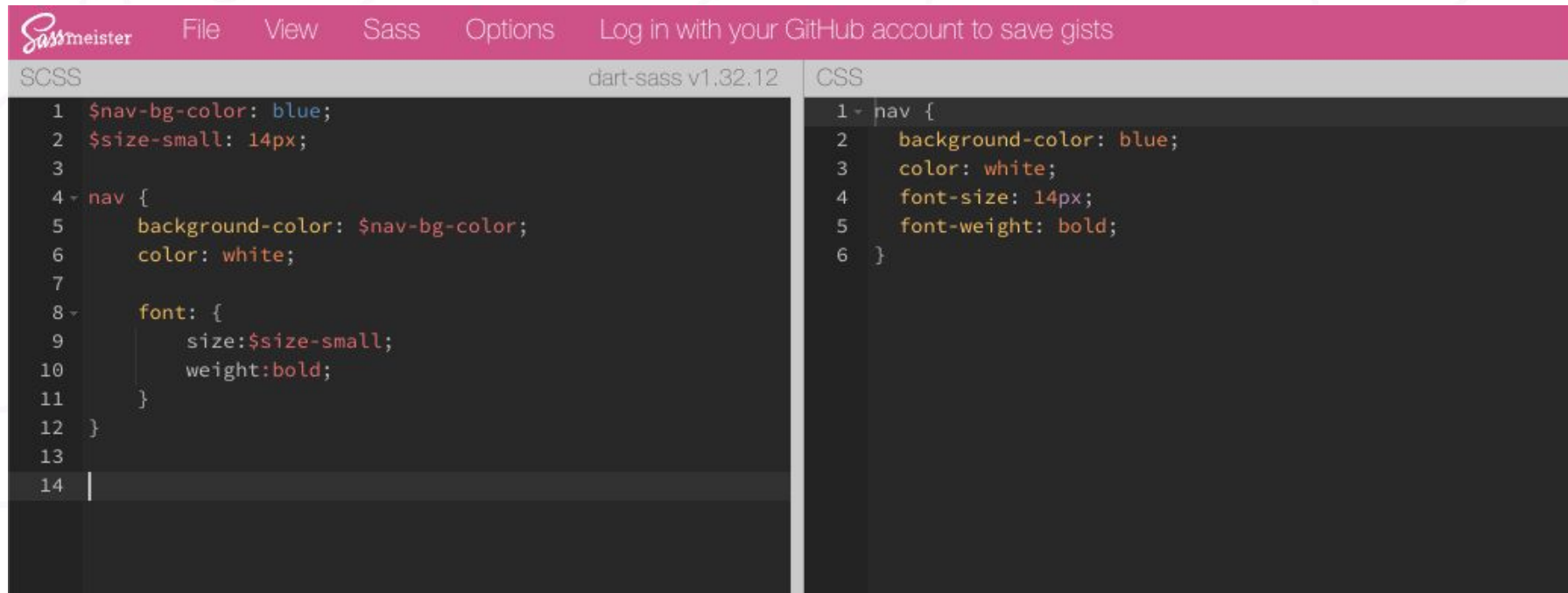
Sass

Sass es el lenguaje de extensión (preprocesador) de CSS de grado profesional más maduro, estable y potente del mundo.



<https://sass-lang.com/>

Playground: Sass meister



The screenshot shows the Sassmeister web application. The top navigation bar is pink and contains the 'Sassmeister' logo, a menu with 'File', 'View', 'Sass', and 'Options', and a link to 'Log in with your GitHub account to save gists'. Below the navigation bar, there are two panels: 'SCSS' on the left and 'CSS' on the right. The 'SCSS' panel shows the following code:

```
1 $nav-bg-color: blue;
2 $size-small: 14px;
3
4 nav {
5   background-color: $nav-bg-color;
6   color: white;
7
8   font: {
9     size:$size-small;
10    weight:bold;
11  }
12 }
13
14
```

The 'CSS' panel shows the compiled CSS output:

```
1 nav {
2   background-color: blue;
3   color: white;
4   font-size: 14px;
5   font-weight: bold;
6 }
```

<https://www.sassmeister.com/>

Variables en Sass

SCSS

dart-sass v1.32.12

```
1 // SCSS
2 $myFont: Helvetica, sans-serif;
3 $myFontColor: red;
4 $myFontSize: 18px;
5 $myWidth: 680px;
6
7 body {
8   font-family: $myFont;
9   font-size: $myFontSize;
10  color: $myFontColor;
11 }
12
13 #container {
14   width: $myWidth;
15 }
```

CSS

```
1 body {
2   font-family: Helvetica, sans-serif;
3   font-size: 18px;
4   color: red;
5 }
6
7 #container {
8   width: 680px;
9 }
10
```

Para crear variables usamos el \$ para nombrarlas, posteriormente definiremos su valor.

Para usar variables, solo escribiremos el \$ seguido del nombre de la variable.

Anidamiento de Reglas en Sass (Nesting)

```
SCSS                                     dart-sass v1.32.12
1  // SCSS
2  nav {
3    ul{
4      margin: 0;
5      padding: 0;
6      list-style: none;
7    }
8    li {
9      display: inline-block;
10   }
11   a {
12     display: block;
13     padding: 6px 12px;
14     text-decoration: none;
15   }
16 }
17
```

```
CSS
1  nav ul {
2    margin: 0;
3    padding: 0;
4    list-style: none;
5  }
6  nav li {
7    display: inline-block;
8  }
9  nav a {
10   display: block;
11   padding: 6px 12px;
12   text-decoration: none;
13 }
14
```

Si colocamos elementos dentro de las `{ }` de otro elemento padre, estas serán tomadas en cuenta. Por lo que resulta cómodo para agrupar estilos que tengan qué ver con un elemento en concreto.

Anidamiento de Propiedades

```
SCSS                                dart-sass v1.32.12
1  // SCSS
2  body {
3    font:{
4      family: Helvetica, sans-serif;
5      size: 18px;
6      weight: bold;
7    }
8  }
9
10 .title {
11   text:{
12     align: center;
13     transform: lowercase;
14     overflow: hidden;
15   }
16 }
17
```

```
CSS
1 body {
2   font-family: Helvetica, sans-serif;
3   font-size: 18px;
4   font-weight: bold;
5 }
6
7 .title {
8   text-align: center;
9   text-transform: lowercase;
10  text-overflow: hidden;
11 }
```

Importar Archivos Parciales con Sass (_Partials)

SCSS dart-sass v1.32.12

```
1 // SCSS
2 // _colors.scss
3 $myRed: #b75038;
4 $myBlue: #1376b8;
5 $myGreen: #378f66;
6
```

SCSS dart-sass v1.32.12

```
1 // SCSS
2 // main.scss
3 x @use 'colors';
4
5 body {
6   font-family: Helvetica, sans-serif;
7   font-size: 18px;
8   color: $myBlue;
9 }
10
```

CSS

```
1 body {
2   font-family: Helvetica, sans-serif;
3   font-size: 18px;
4   color: #1376b8;
5 }
6
```

Puedes crear archivos Sass parciales que contengan pequeños fragmentos de CSS que puedes incluir en otros archivos Sass.

Un parcial es un archivo Sass nombrado con un guión bajo inicial (ejemplo. `_colors.scss`).

El guión bajo permite a Sass saber que el archivo es sólo un archivo parcial y que no debe ser generado en un archivo CSS.

Los parciales de Sass se utilizan con la regla `@use`.

Reutilización de estilos: @mixins en Sass

SCSS dart-sass v1.32.12

```
1 // SCSS
2 @mixin important-text {
3     color: white;
4     font-size: 32px;
5     font-weight: bold;
6 }
7
8 .danger {
9     @include important-text;
10    background-color: red;
11 }
12
13 .notice {
14     @include important-text;
15     background-color: blue;
16 }
17
```

CSS

```
1 .danger {
2     color: white;
3     font-size: 32px;
4     font-weight: bold;
5     background-color: red;
6 }
7
8 .notice {
9     color: white;
10    font-size: 32px;
11    font-weight: bold;
12    background-color: blue;
13 }
14
```

Los mixins nos ayudan a crear bloques de estilos reutilizables.

Se declaran con **@mixin**
nombre-del-mixin

Para usarlos, utilizamos **@include**
nombre-del-mixin

Los @mixins con parámetros y valores por defecto


SCSS

dart-sass v1.32.12

```
1 // SCSS
2 // Mixins con dos argumentos
3 @mixin bordered($color, $width) {
4     border: $width solid $color;
5 }
6
7 .myArticle {
8     @include bordered(blue, 1px)
9 }
10
11 .myNotes {
12     @include bordered(red, 2px)
13 }
14
15 // Mixins con valores por defecto
16 @mixin with-border($color: blue, $width: 1px) {
17     border: $width solid $color;
18 }
19
20 .myTips {
21     @include with-border($color: orange);
22 }
```

CSS

```
1 .myArticle {
2     border: 1px solid blue;
3 }
4
5 .myNotes {
6     border: 2px solid red;
7 }
8
9 .myTips {
10     border: 1px solid orange;
11 }
12
```



Los @mixins + if

SCSS Sass

```
@mixin avatar($size, $circle: false) {  
  width: $size;  
  height: $size;  
  
  @if $circle {  
    border-radius: $size / 2;  
  }  
}  
  
.square-av {  
  @include avatar(100px, $circle: false);  
}  
  
.circle-av {  
  @include avatar(100px, $circle: true);  
}
```

CSS

```
.square-av {  
  width: 100px;  
  height: 100px;  
}  
  
.circle-av {  
  width: 100px;  
  height: 100px;  
  border-radius: 50px;  
}
```

[Sass: @if and @else \(sass-lang.com\)](https://sass-lang.com)

Herencia @extend

SCSS

dart-sass v1.32.12


```
1 // SCSS
2 // Este CSS %message-shared se utiliza
3 // con extends y se mostrará en el código.
4 %message-shared {
5   border: 1px solid #ccc;
6   padding: 10px;
7   color: #333;
8 }
9
10 // Este CSS %equal-heights no se utiliza
11 // con extends, por lo tanto no aparecerá.
12 %equal-heights {
13   display: flex;
14   flex-wrap: wrap;
15 }
16
17 .message {
18   @extend %message-shared;
19 }
20
21 .success {
22   @extend %message-shared;
23   border-color: green;
24 }
25
26 .warning {
27   @extend %message-shared;
28   border-color: yellow;
29 }
30
```

CSS

```
1 .warning, .success, .message {
2   border: 1px solid #ccc;
3   padding: 10px;
4   color: #333;
5 }
6
7 .success {
8   border-color: green;
9 }
10
11 .warning {
12   border-color: yellow;
13 }
14
```









Extensión nueva para utilizar







Sass

v1.8.22

Syler |  1,470,171 |      (7)

Indented Sass syntax Highlighting, Autocomplete & Formatter

Disable  Uninstall   

This extension is enabled globally.

Details | Feature Contributions | Changelog | Runtime Status

build passing

codefactor A

version v1.8.22

downloads 6.2M

installs 1.5M

last commit july

issues 11 open

![[Tested With]]([https://img.shields.io/badge/Syntax Tested With-test--grammar-red?style=for-the-badge](https://img.shields.io/badge/Syntax%20Tested%20With-test--grammar-red?style=for-the-badge))

USING SASS-FORMATTER

Indented Sass syntax highlighting, autocomplete & Formatter for VSCode

Categories

Programming Languages

Formatters

Extension Resources

[Marketplace](#)

[Repository](#)

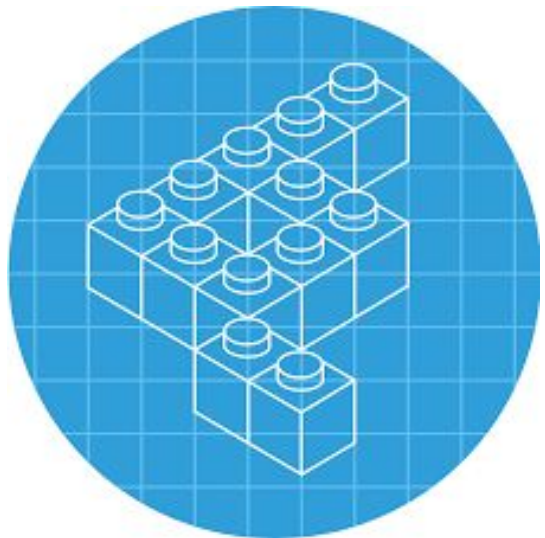
[License](#)



BEM

DEV.F.
DESARROLLAMOS(PERSONAS);

dev



BEM (Block Element Modifier)

BEM es una filosofía que nos indica cómo debemos dividir y estructurar lógicamente las piezas en qué se compone una página web.

Es decir, es una manera de nombrar las clases de los nodos de tu HTML para posteriormente seleccionarlos con CSS de una manera fácil, sencilla y clara.

Este método nos propone dividir la interfaz de usuario en bloques independientes para crear componentes, ayudándonos a distinguir de manera simple de qué componente y parte hablamos.



BEM CSS

Beneficios BEM

Modularidad: Los estilos de bloque nunca dependen de otros elementos de una página, por lo que nunca colisionaran con otros (problemas de cascada) y facilita transferir bloques de tus proyectos terminados a otros nuevos.

Reusabilidad: Componer bloques independientes de diferentes maneras, y reutilizarlos de forma inteligente, reduce la cantidad de código CSS que tendrás que mantener.

Estructura: La metodología BEM proporciona a su código CSS una estructura sólida que sigue siendo sencilla y fácil de entender.

```
.bloque__elemento--modificador{  
  
}
```

```
.bloque  
.bloque__elemento  
.bloque--modificador  
.elemento--modificador  
.bloque__elemento--modificador
```

*Si un nombre tiene requiere “un espacio”
usamos un guión simple como separador,
otras alternativas es usar notación
camelCase o un guión bajo.*

<https://en.bem.info/methodology/naming-convention/>

BEM (Block Element Modifier)

BEM son tres siglas:

- B de bloque.
- E de elemento.
- M de modificador.

BEM nos indica qué debemos usar clases para nuestros selectores con una estructura definida (bloque, elemento, modificador).

BEM tiene unas convenciones acerca de cómo debemos nombrar a nuestras clases de CSS.

.bloque__elemento--modificador

HTML (bloque)

```
<header class="header">  
  <!-- aquí van mis elementos -->  
</header>
```



Bloques BEM

Un bloque es una sección independiente que tiene significado propio por sí solo.

Contiene todos los nodos HTML de una estructura a la que te estés refiriendo.

Para aclarar esto las siguientes etiquetas HTML son consideradas bloques a simple vista: **header**, **sidebar**, **aside**, **main**, **footer**, **article**, **section**, **ul**, **ol**, **div**.

`.bloque__elemento--modificador`



Elementos BEM

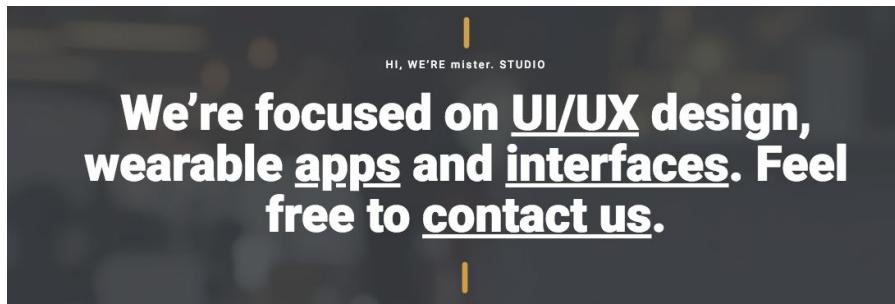
Un elemento **son porciones más pequeñas internas a un bloque.**

El elemento es una pieza de un bloque.
El bloque es el todo y los elementos son las piezas.

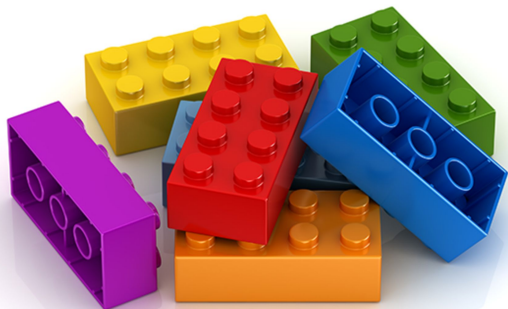
El método BEM nos indica que cada elemento nace a partir de la clase del bloque que conforma seguida de dos guiones bajos y el nombre clave del elemento.

`.bloque__elemento--modificador`

`.hero__text--small`



`.hero__text--big`



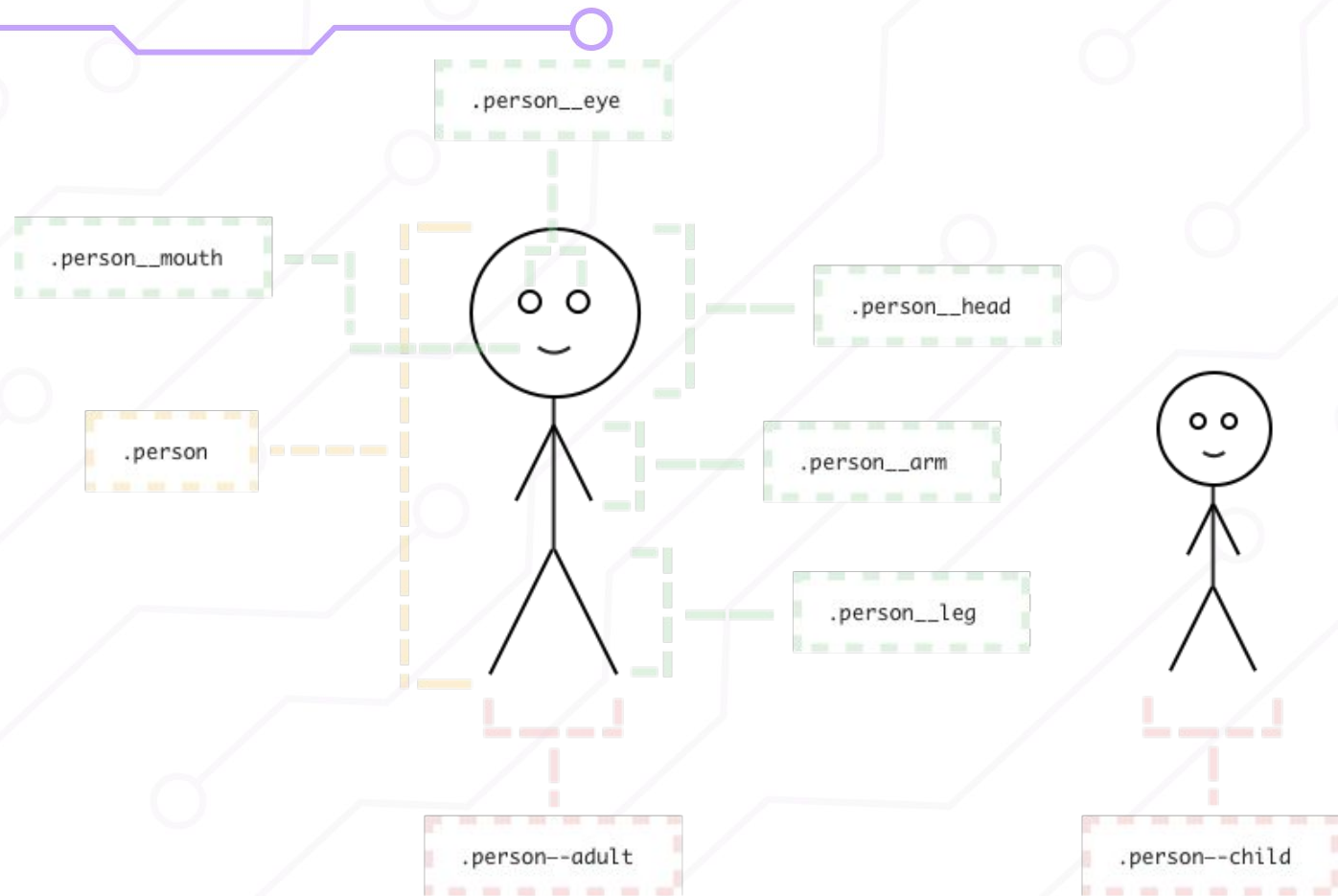
Modificadores BEM

Para modificar el estilo de un elemento específico, existen los modificadores.

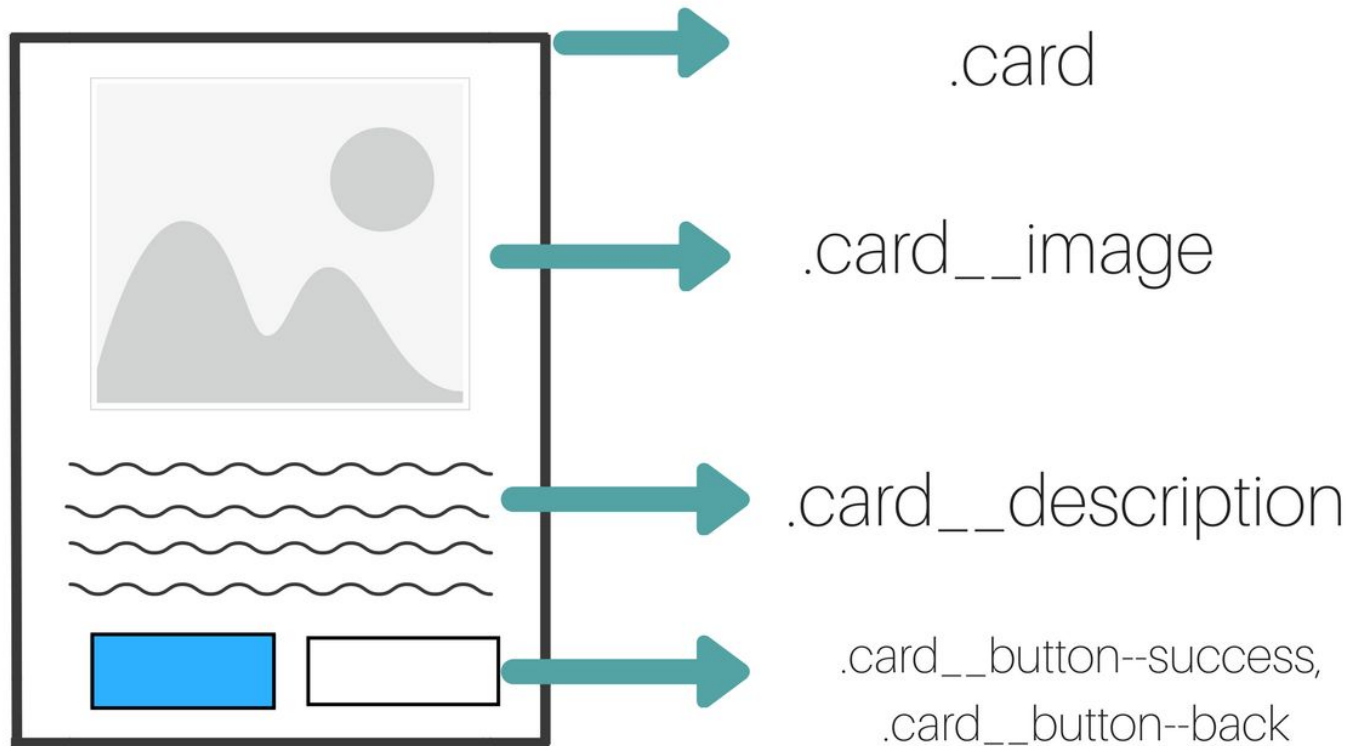
Estos se usan agregando un doble guión justo después del elemento o bloque que se quiere modificar.

Un modificador nos permite **tener variantes de estilo sobre un bloque o elemento.**

Ejemplo de BEM aplicado a una persona



Ejemplo de nomenclatura BEM aplicada



Ejemplo de BEM aplicado en HTML

SOME ENGAGING TITLE

There are many variations of passages of Lorem Ipsum available.

The majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable.

READ MORE

```
<article class="card">
```

```
<h1 class="card__title">Some engaging title</h1>
```

```
<p class="card__text"> There are many variations....</p>
```

```
<p class="card__text card__text--secondary"> The majority....</p>
```

```
<a class="card__button button" href="#">Read more</a>
```

```
</article>
```

Resumen

Blocks, Elements and Modifiers

You will not be surprised to hear that BEM is an abbreviation of the key elements of the methodology — Block, Element and Modifier. BEM's strict naming rules can be found [here](#).

Block

Standalone entity that is meaningful on its own.

Examples

`header` , `container` , `menu` , `checkbox` , `input`

Element

A part of a block that has no standalone meaning and is semantically tied to its block.

Examples

`menu item` , `list item` , `checkbox caption` , `header title`

Modifier

A flag on a block or element. Use them to change appearance or behavior.

Examples

`disabled` , `highlighted` , `checked` , `fixed` , `size big` , `color yellow`



Sass conoce BEM

DEV.F.
DESARROLLAMOS(PERSONAS);

BEM + SASS

BEM conoce SASS

Hasta ahora, los módulos BEM los hemos escrito en CSS. Pero recordar que podemos condensar código CSS usando SASS.

Un módulo BEM escrito en SASS se vería así:

```
.bloque{
  margin:0 auto;

  &__boton{
    border: 1px solid black;

    &--rojo{
      background:red;
    }
  }
}
```

Aquí usamos:

- Anidación
- Selectores padre (&)



Usando Sass en Vite

DEV.FX
DESARROLLAMOS(PERSONAS);

dev

Vite ya incluye soporte para Sass

That said, Vite does provide built-in support for `.scss`, `.sass`, `.less`, `.styl` and `.stylus` files. There is no need to install Vite-specific plugins for them, but the corresponding pre-processor itself must be installed:

A dark blue terminal window with three colored window control buttons (red, yellow, green) at the top left. The text 'npm add sass -D' is displayed in a monospaced font, with 'npm' in orange, 'add' in white, 'sass' in white, and '-D' in pink.

```
npm add sass -D
```

Pero es necesario instalar el pre-procesador de Sass como dependencia de desarrollo (-D).

Referencia: <https://vitejs.dev/guide/features.html>

Para más información, puede consultarse:

- <https://sass-lang.com/guide>
- <https://platzi.com/blog/bem/>
- <https://9elements.com/bem-cheat-sheet/>

