

# **Tema 2.10 - Generación de Interfaces I - CSS (BEM, funciones matemáticas)**

**Módulo: Desarrollo de Interfaces**

Curso 2º DAM



# ¿Qué es BEM (Block Element Modifier)?

BEM (Block-Element-Modifier) es una convención de nomenclatura ampliamente utilizada para nombres de clases CSS. Ayuda a escribir CSS más legible, comprensible y escalable.

Los nombres en BEM siguen esta estructura:



```
[Block]__[Element]--[Modifier]
```

- **Block:** Representa el componente principal, como un botón o una tarjeta.
- **Element:** Subpartes o elementos dentro del bloque, como un icono o un título.
- **Modifier:** Variaciones en el estilo del bloque o de un elemento, como un botón secundario o deshabilitado.

# Ejemplos simples BEM

## 1. Bloque sin elementos ni modificadores



```
<button class="btn"></button>

.btn {...}
```

## 2. Bloque con modificador



```
<!-- CORRECTO -->
<button class="btn btn--secondary"></button>

.btn {...}
.btn--secondary {...}

<!-- INCORRECTO -->
<button class="btn--secondary"></button>

.btn--secondary {...}
```

## 3. Bloque con elementos



```
<figure class="photo">
  
  <figcaption class="photo__caption">Look at me!</figcaption>
</figure>

.photo {...}
.photo__img {...}
.photo__caption {...}
```

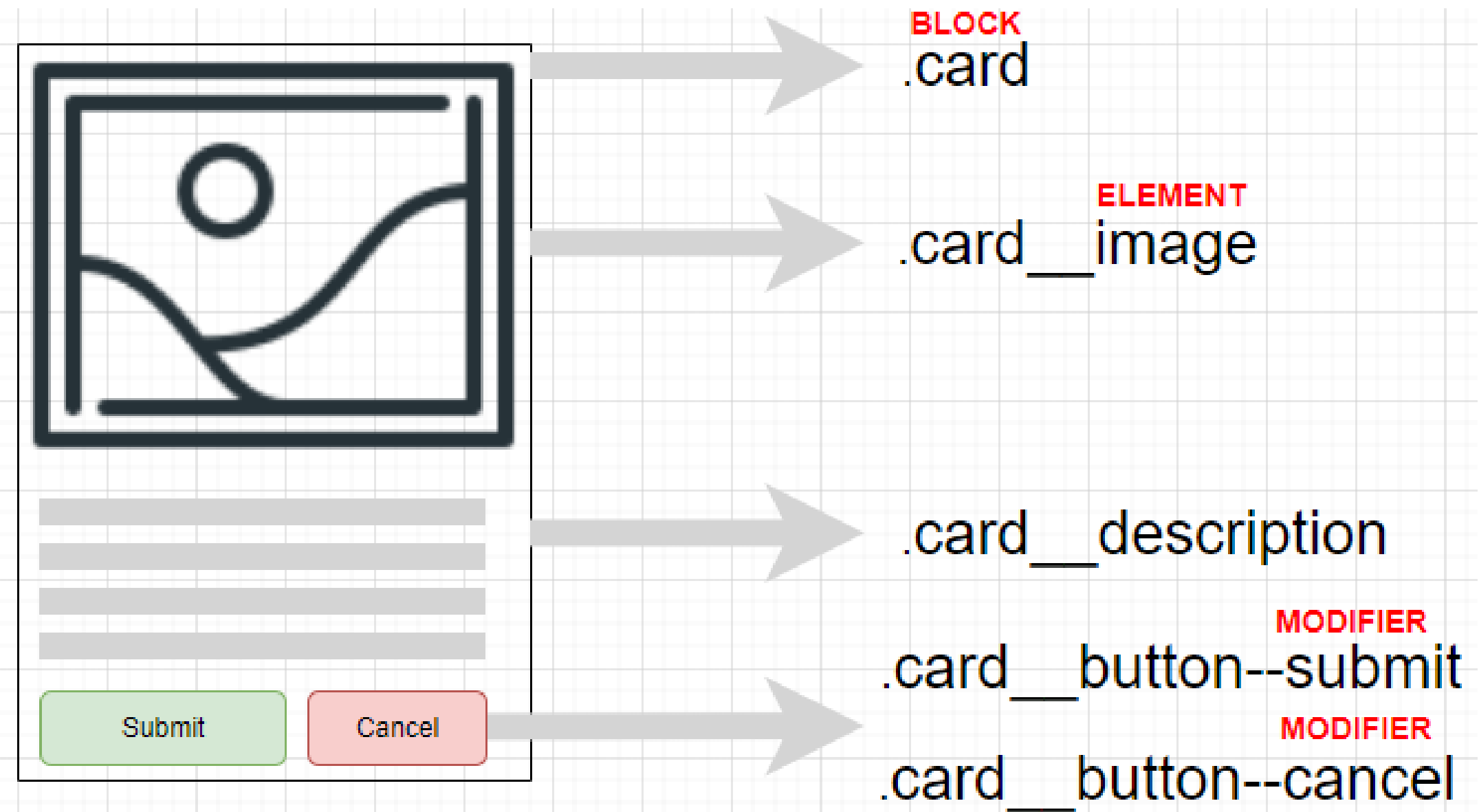
## 4. Bloque con elementos y modificadores



```
<figure class="photo">
  
  <figcaption class="photo__caption photo__caption--large">Look at me!</figcaption>
</figure>

.photo {...}
.photo__img {...}
.photo__caption {...}
.photo__img--framed {...}
.photo__caption--large {...}
```

# Ejemplo estructura BEM



## Algunos errores comunes

<https://dev.to/visuellverstehen/common-mistakes-when-writing-css-with-bem-4921>



```
<div class="card">
  <div class="card__image"></div>
  <p class="card__description"></p>
  <button class="card__button--submit"></button>
  <button class="card__button--cancel"></button>
</div>
```



```
.card__image {
  display: block;
}
.card__description {
  display: block;
}
.card__button--submit {
  display: inline-block;
}
.card__button--cancel {
  display: inline-block;
}
```

# Funciones matemáticas útiles en CSS

- **calc()**: Realiza cálculos matemáticos.
- **min()** / **max()**: Devuelve el valor mínimo o máximo entre varias opciones (normalmente 2).
- **clamp()**: Define un valor dinámico dentro de un rango mínimo y máximo. 3 valores (mínimo, preferido, maximo).



```
width: calc(100% - 50px); /* Ancho calculado dinámicamente */
```



```
width: min(50%, 400px); /* Usa el menor valor */  
width: max(200px, 10%); /* Usa el mayor valor */
```



Comunmente utilizada para definir max-width, width o min-height, height.



```
font-size: clamp(1rem, 2.5vw, 3rem); /* Escala entre 1rem y 3rem */
```

Comunmente utilizada realizar tamaños fluidos en el texto.





320px

1440px

rango de px a  
tener en cuenta

## Fluid Type Scale Calculator

Generate font size variables for a fluid type scale with CSS clamp. Grab the output CSS and drop it into any design system.

### Minimum (Mobile)

At this minimum viewport width, all font sizes in your type scale are computed as the base font size times a power of your chosen ratio.

Base font size (px)

16

Screen width (px)

320

Type scale ratio

1.25

### Maximum (Desktop)

At this maximum viewport width, all font sizes in your type scale are computed as the base font size times a power of your chosen ratio.

Base font size (px)

19

Screen width (px)

1440

Type scale ratio

1.333

```
--font-size-sm: clamp(0.8rem, 0.13vi + 0.77rem, 0.89rem);  
--font-size-base: clamp(1rem, 0.27vi + 0.95rem, 1.19rem);  
--font-size-md: clamp(1.25rem, 0.48vi + 1.15rem, 1.58rem);  
--font-size-lg: clamp(1.56rem, 0.78vi + 1.41rem, 2.11rem);  
--font-size-xl: clamp(1.95rem, 1.23vi + 1.71rem, 2.81rem);  
--font-size-xxl: clamp(2.44rem, 1.87vi + 2.07rem, 3.75rem);  
--font-size-xxxl: clamp(3.05rem, 2.78vi + 2.5rem, 5rem);
```

Copy to clipboard

<https://www.fluid-type-scale.com/>

Utilizaremos *clamp()* para  
definir de manera fluida el  
font-size de nuestra  
interfaz