Nombres de variantes de escritura (y selectores)

- UPPERCASE, lowercase, Title Case
- camel case: miClase
- pascal case: MiClase
- kebeb case: mi-clase
- snake case: mi_clase

Especificidad

Es la manera mediante la cual los nav deciden que valores de una propiedad CSS son más relevantes para un elemento, y por lo tanto, serán aplicados. (que regla es + esppecífica que otra). Mientras más específica + peso tendrá, pero tb será + dificil de sobreescribir.

- La 1a colunma representa los ID
- La 2a columna es para clases y sus atributos (lo que está entre corchetes), y sus pseudo clases (que se escriben con :hover :root :required :nth-child(odd), :nth-of-type (3n)).
- La 3a columna es para etiquetas (p, input, strong, main...) y sus pseudo elementos (::before, ::after)

1 2 X

```
/* (1 0 0) */
#soyCaja {
    background-color: blue;
}

/* (0 0 2) */
p > br {

}

/* (0 2 0) */
.box:hover {
    background-color: green;
}

/* (0 1 0) */
.caja {
    background-color: red;
}


/* (1 1 0) */
#header .caja {
    background-color: red;
}


.boton {
```

```
}
```

```
<div id="soyCaja" class="caja box"
style="background:orange !important">
</div>
```

Metodologías BEM y SUIT

Sirven para organizar y escribir CSS de manera más estructurada, reutilizable y fácil de mantener

BEM

Organiza el código en bloques reutilizables.

Estructura de BEM

BEM se basa en tres conceptos principales:

- Bloque: Es el componente principal, ese nombre va todo en minuscula

  ...

- Elementos: son las partes que componen el bloque (los hijos) <block__elem>
- Modificadores: son variaciones del bloque/elemento (es el nombre del bloque y del elemento--nombre del modificador <block__elem--mod>

Titulo

soy un parrafo

Nomenclatura BEM (Block-Element-Modifier)

- Sus nombres son en kebab-case
- bloques:".boton", ".fomurlario", ".navegacion"
- elementos: son hijos de bloques ej: ".formualrio__checkbox", ".fomulario__boton", ".formulario__titulo"
- modificadores: Son clases que modifican tanto elementos como bloquesej: ".fomulario__boton--disabled"

.[bloque]__[elemento]--[modificador] {

}

```
 <div class="header">
    <ul class="header__list">
        <li class="header__li">Mis recetas</li>
        <li class="header__li">Recetas guardadas</li>
        <li class="header__li">Publicar nueva receta</li>
    </ul>

    <!-- bloque formulario -->
    <form action="buscar" class="formu" method="get">
        <input type="search" class="formu__input" placeholder="Buscar más
recetas">
        <button class="formu__buscar" type="submit">
            <i class="fa fa-search">
            </i>
        </button>
    </form>
  </div>
```

```
    .header {}
    .header__list {}
    .header__li {}

    .nav {}
    .nav__item {}

    .formu {}
    .formu__buscar {}
```

Nomenclatura SUIT

- Las clases de los componentes tienen un nombre único y claro.
- Los modificadores se aplican utilizando un prefijo con guion (-).

La convención para los nombres de clase en SUIT es más simple que la de BEM, pero aún sigue siendo modular.

Nombres de clases en SUIT: - Componentes: Component - Modificadores: Component--modifier

Ejemplo:

```
<button class="Button">Normal Button</button>
<button class="Button Button--primary">Primary Button</button>
```

```
/* Componente */
.Button {
```

```
  padding: 10px;
  font-size: 14px;
}

/* Modificador */
.Button--primary {
  background-color: blue;
  color: white;
}
```

Diferencias entre BEM y SUIT

1. BEM
    - tiene tres niveles: bloque, elemento y modificador.
    - hace una clara distinción entre bloques (componentes independientes) y elementos (partes de un bloque).
    - usa __ para separar elementos y -- para modificadores.
2. SUIT
    - usa solo dos niveles: componente y modificador. Esto hace que sea un poco más simple y directo.
    - no distingue entre bloques y elementos. Todo es un componente.
    - usa -- para modificadores, y no tiene un separador específico entre componentes y elementos.

¿Cuándo usar cada uno?

`BEM` es útil si prefieres una estructura más detallada y clara que distinga entre bloques y elementos. Es excelente para proyectos grandes con muchos componentes reutilizables.

`SUIT` es adecuado si prefieres una metodología más simple y directa para organizar los estilos. Es útil en proyectos donde no es necesario hacer una distinción tan estricta entre bloques y elementos.

Convenciones

- CAMEL CASE: se empieza con la 1a letra minúscula y la 1a letra de cada nueva palabra subyacente en mayúscula. (JavaScript, Java o C#) Ej. cosasParaHacer Ej. edadDelAmigo Ej. valorFinal
- PASCAL CASE: tambien se comoce como upper camel case/ capital case, combina palabras poniéndolas todas con la 1a letra en mayúscula. (C#, Java o Python.) Ej. CosasParaHacer Ej. EdadDelAmigo Ej. ValorFinal
- KEBAB CASE: se usa el guión paracombinar las palabras. Cuando el kebab case está en mayúsculas, se llama "screaming kebab case". Ideal para URLs o nombres de archivos (por ejemplo, user-profile.html). Ej. cosas-para-hacer Ej. edad-del-amigo Ej. valor-final
- SNAKE CASE: se usa guión bajpp en lugar de espacio para separar las palabras. Cuando skane case está en mayúsculas, se le conoce como "screaming snake case". (Python) Ej. cosas_para_hacer Ej.

edad_del_amigo Ej. valor_final