Selectores descendientes

Edición Web I

Hasta ahora creamos **una clase por cada elemento** que queremos dar propiedades:



** ** → .content
- **<img>** → .foto
- **<h1>** → .title
- **<p>** → .text

Css

```css
.content{
    background-color: aqua;
    padding:16px;
}

.foto{width:80px;}

.title{
    color: rgb(20, 137, 221);
    font-size:24px;
}

.text{
    color: rgb(20, 137, 221);
    font-size:24px;
}
```

Html

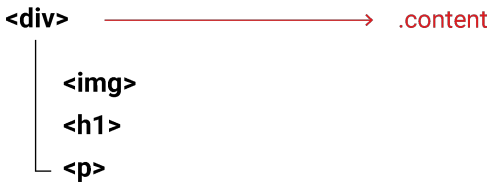
```html
<div class="content">
    <img class="foto" src="imagen.png">
    <h1 class="title">SEARCH</h1>
    <p class="text">Busca un término...</p>
</div>
```

Selectores descendientes

Podemos declarar una única clase css que afecta a diversos objetos **descendentemente** (hijos).

dejando un espacio entre la clase y el elemento sobre el que queremos actuar.





Css

```css
.content{
    background-color: aqua;
    padding:16px;
}

.content img{width:80px;}

.content h1{
    color: rgb(20, 137, 221);
    font-size:24px;
}

.content p{
    color: rgb(20, 137, 221);
    font-size:24px;
}
```

Html

```html
<div class="content">
    <img src="imagen.png">
    <h1>SEARCH</h1>
    <p>Busca un término...</p>
</div>
```

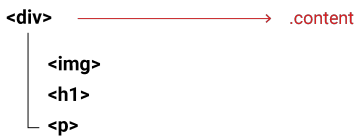
Edición Web I
Jonathan Faus

Html
Nos permite escribir menos clases y poder ver un html más legible.

Css
creamos la misma clase por cada elemento queramos actuar a diferencia que el nombre de la clase es el mismo, solo cambia el elemento que sigue a la clase.

!!! Esto solo sirve para elementos desdendentes que están dentro del contenedor principal que es quien tiene la clase.





Css

```css
.content{
    background-color: aqua;
    padding:16px;
}

.content img{width:80px;}

.content h1{
    color: rgb(20, 137, 221);
    font-size:24px;
}

.content p{
    color: rgb(20, 137, 221);
    font-size:24px;
}
```

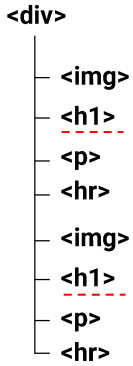
Html

```html
<div class="content">
    <img src="imagen.png">
    <h1>SEARCH</h1>
    <p>Busca un término...</p>
</div>
```

Afectando a todas las etiquetas descendientes

En este caso cuando declaramos **.content h1** va a afectar a todas las etiquetas <h1> que tengamos dentro del contendor, en este caso 2 etiquetas.





Css

```css
.content{
    background-color: aqua;
    padding:16px;
}

.content img{width:80px;}

.content h1{
    color: rgb(20, 137, 221);
    font-size:24px;
}

.content p{
    color: rgb(20, 137, 221);
    font-size:24px;
}
```

Html

```html
<div class="content">
    <img src="imagen.png">
    <h1>SEARCH</h1>
    <p>Busca un término...</p>
    <hr>
    <h1>SEARCH</h1>
    <p>Busca un término...</p>
    <hr>
</div>
```

nth-child( )

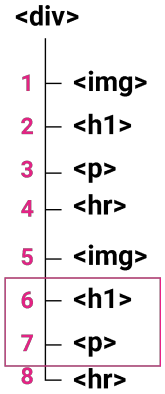
Especificar a qué elemento hijo quiero afectar

Si por ejemplo tenemos varios <h1> y queremos indicar que el segundo tengo un color diferente se lo tendremos que indicar mediante la pseudoclase
:nth-child(número de elemento a seleccionar);

En el ejemplo vamos a asignar un color diferente a los elementos 6 y 7 que tenemos dentro del contenedor:

Lo traduciremos como el hijo n°6 y n°7





Css

```css
.content :nth-child(6), :nth-child(7){
    color: red;
}
```

Separamos por comas para aplicar los mismos valores a varios hijos.

Html

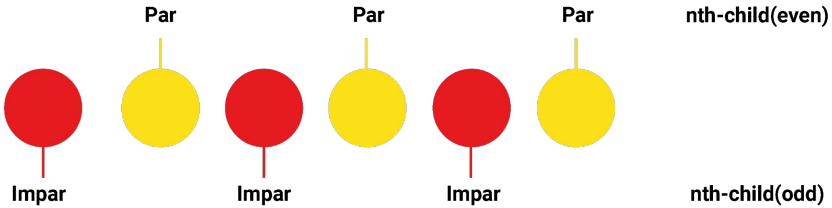
```html
<div class="content">
    <img src="imagen.svg">
    <h1>Seach</h1>
    <p>Busca un termino...</p>
    <hr>
    <img src="imagen.svg">
    <h1>Seach</h1>
    <p>Busca un termino...</p>
    <hr>
</div>
```

nth-child( )

seleccionar pares e impares

Pares
nth-child(2n); o podemos utilizar también
nth-child(even);

Pares
nth-child(2n+1); o podemos utilizar también
nth-child(odd);



Css

```css
.content :nth-child(odd){
    background-color: red;
}

.content :nth-child(even){
    background-color: yellow;
}
```

Html

```html
<div class="content">
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
</div>
```