educación IT
CODE YOUR FUTURE

Bootcamp Java Developer

Fase 2 - Java Web Developer

Módulo 14

educación IT
CODE YOUR FUTURE

Pseudo Clases básicas

Pseudo Clases

`:link`

Esta pseudo clase me permite indicar qué sucederá con aquellos **vínculos que no han sido visitados:**

```
a:link { color: purple; }
```

Estos elementos se visualizan de forma predeterminada en color azul y subrayados.

`:visited`

Permite indicar qué sucederá con los **vínculos que ya han sido visitados:**

```
a:visited { color: green; }
```

`:active`

Esta pseudo clase me permite indicar **qué sucederá con los vínculos cuando hacemos clic** sobre ellos:

```
a:active { color: green; }
```

Estos elementos no tienen características predeterminadas y **se activan con el clic,** a diferencia de **`:hover`**.

`:hover`

Esta pseudo clase permite indicar **qué sucederá con los vínculos al pasar el cursor del mouse sobre ellos:**

```
a:hover { color: green; }
```

Posibilidades

`:hover` y **`:active`** pueden implementarse en cualquier elemento del HTML (a diferencia de **`:link`** y **`:visited`**):

```
div { width: 300px; height: 400px; border-radius: 120px; overflow: hidden;}
div:hover { background-color: gray;}
```

De esta forma podemos trabajar con nuestros elementos de forma dinámica desde CSS.

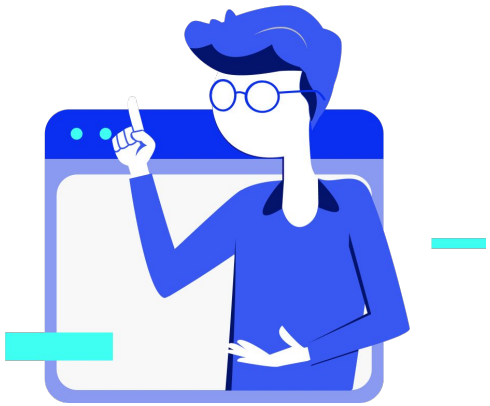
`:last-child`

Este selector permite indicar una propiedad o regla de estilo referenciada a aquel elemento, que sin importar su tipo, es **el último hijo de un elemento contenedor o padre:**

```
<style>

p:last-child { color: blue; }

</style>
```



¿Sabías que...?

:focus es una pseudo clase que nos **permite trabajar en elementos cuando hacemos foco con el cursor.**

El uso más común es en campos de texto o vínculos:

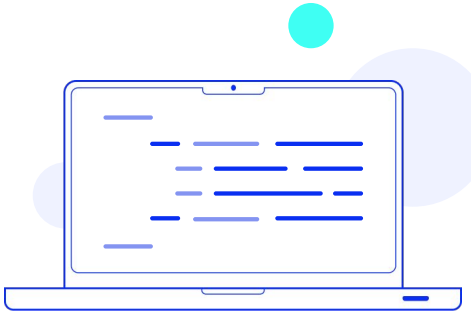
```
input:focus { background: gray;}
```

Pseudo Clases avanzadas

`:first-child`

Referencia al **primer hijo de un elemento padre - no importa el tipo de elemento -.**

Por ejemplo, si en CSS escribo **`:first-child`**, esto afectará a todos los elementos que sean primeros hijos de un padre (cada nuevo *parent* o padre, hay un nuevo hijo).



Ejemplo:

```
p:first-child { background: orange;}

</style>
</head>
<body>

<div>
<p> primer hijo y es párrafo </p>
<p> Segundo párrafo no es un primer hijo sino que es el segundo </p>
</div>

<div>
<p> primer hijo, es párrafo y es un primer hijo porque le div que lo contiene determina un nuevo padre </p>
<p> segundo hijo</p>

</div>
```

`:first-of-type`

Referencia al **primer hijo de un tipo específico,** por ejemplo:

```
p:first-of-type { font-style: bold; font-size: 2.5em;}
```

Generamos la siguiente estructura:

```
<h1>Mi conocimiento del mundo</h1>

<p> Lorem ipsum dolor sit amet consectetur adipisicing elit. Ipsa quis in explicabo, deserunt est itaque nemo
ex nihil quisquam molestias illo labore obcaecati quia dolorem illum laborum sit qui ab.</p>
```

El ejemplo anterior es interesante porque, si bien el primer **`child`** o hijo es el **`h1`**, el **`p`** que vemos ahí sería el primer **`p:first-of-type`**, ya que es el primero de su tipo.

Entonces el resultado será el siguiente:

Mi conocimiento del mundo

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vehicula sapien.

`:last-of-type`

El selector anterior es interesante y encuentra su contrapartida en **`:last-of-type`** en sentido inverso.

Es decir, en la siguiente estructura, el último párrafo se vería afectado:

```
<p> Lorem, ipsum dolor sit amet consectetur adipisicing elit. Esse, repudiandae ducimus possimus iure aliquam
sit officia dolore explicabo aperiam, eaque deleniti illo error fugit autem ipsam sint aut deserunt. Velit!</p>


<h2>Enunciado final </h2>
```

Con el código anterior, el resultado de la siguiente regla:

```
p:last-of-type { color: red;}
```

Será el siguiente, ya que si bien el **h2** es el último **child**, de su tipo este párrafo es el último:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vehicula sapien

Enunciado final

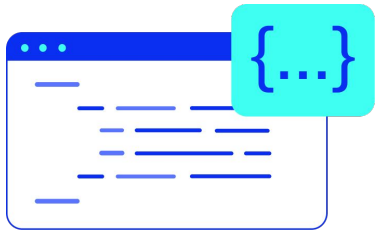
Con el código anterior, si sumamos un párrafo más en la estructura, el resultado será el siguiente:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vehicula sapien

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vehicula sapien

Enunciado final

Pues habiendo dos párrafos, sólo uno de ellos puede ser el último o el primero.



`:nth-child(n)`

A nuestros selectores ya vistos podemos sumarles algunas complejidades, por ejemplo:

```
p:nth-child(3) { font-style: italic; font-size: 1.3em; }
p:nth-last-child(2) { color: orange;}
```

Indicar la posición es una posibilidad bastante interesante al momento de trabajar.

También se puede especificar algo más genérico como afectar a pares e impares de forma conjunta.

:nth-child(odd)

Nos permite afectar a todos los elementos de posición número impar, por ejemplo en la siguiente estructura:

```
<p> Lorem ipsum dolor sit amet consectetur adipisicing elit. Ipsa quis in explicabo, deserunt est itaque nemo ex nihil quisquam molestias illo labore obcaecati quia dolorem illum laborum sit qui ab.</p>

<p> Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur, quaerat. Quibusdam voluptatum ratione ducimus iste adipisci atque, tenetur totam exercitationem maxime. Molestias eius ex quae voluptas fugiat tempore facilis totam. </p>

<p> Lorem ipsum dolor sit, amet consectetur adipisicing elit. Repellat expedita voluptate alias voluptas culpa corrupti officia nemo, reprehenderit hic dolor quaerat accusamus quia, voluptates veritatis, magnam necessitatibus maxime nisi aperiam? </p>


<p> Lorem, ipsum dolor sit amet consectetur adipisicing elit. Esse, repudiandae ducimus possimus iure aliquam sit officia dolore explicabo aperiam, eaque deleniti illo error fugit autem ipsam sint aut deserunt. Velit!</p>
```

Con la estructura anterior, y la siguiente regla:

```
p:nth-child(odd) {background-color: red; }
```

Obtendremos el siguiente resultado.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vehicula sapien varius mauris porttitor, vel semper

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vehicula sapien varius mauris porttitor.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vehicula sapien

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vehicula sapien

`:nth-child(even)`

Podemos sumar una versión par:

```
p:nth-child(odd) {background-color: red; }
p:nth-child(even) {background-color: green; }
```

El resultado será:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vehicula sapien varius mauris porttitor, vel semper

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vehicula sapien varius mauris porttitor.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vehicula sapien

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vehicula sapien

:nth(n)

Existen variantes de todos los tipos antes vistos, por ejemplo:

```
:nth-last-child(2)
:nth-last-of-type(6)
:nth-of-type(3)
```

De esta forma la variedad de trabajo y dinamismo representado por estos selectores es sumamente **útil en contenido que puede variar pero debe mantener sus características de diseño u origen.**

Pseudo clases asociadas con validación

Podemos contar con diversas pseudo clases asociadas con validación en formularios. Por ejemplo si deseamos afectar a todos aquellos elementos cuyo contenido es válido de color verde utilizaremos la siguiente regla:

```
:valid {background-color: green;}
```

Si, por el contrario, deseamos afectar a aquellos elementos inválidos, el trabajo será de la siguiente manera:

```
:invalid {background-color: red;}
```

Pseudo clases asociadas con validación

Una implementación muy interesante puede ser combinar **`::after`** o **`::before`** (*) con pseudo clases como **:required** (para elementos que tengan el atributo `required` aplicado) u **:optional** (por el contrario, elementos que no lo tengan) y así agregar de manera más rápida y fácil el famoso asterisco, que representa obligatoriedad en un formulario.

```
:required::after {content: '*';}
```

(*) Nota: veremos más sobre los selectores `::after` y `::before` en el siguiente manual.

Pseudo clases asociadas con validación

Las casillas de verificación y radios que suponen que el usuario las pueda chequear o deschequear poseen su propia pseudo clase asociada también a esa acción:

```
input:checked { box-shadow: 1px 1px  1px black}
```

Revisión

- Repasar los conceptos vistos de **pseudo clases.**
- Implementar estas nuevas propiedades.
- Ver todos los videos y materiales necesarios antes de continuar.

¡Sigamos trabajando!