LEMONCODE

HTML

Especificidad en CSS

La especificidad es una parte clave de la cascada y una de las cuatro etapas distintas que ya hemos comentado en vídeos anteriores. Para entender por qué tus estilos CSS no se aplican a un elemento, es necesario comprender la especificidad en CSS. Usar el valor `!important` puede hacer que tu CSS salga de control, por lo que es importante entender cómo se aplican los estilos. Al saber exactamente qué selector se aplica a un elemento específico, es mucho más fácil cambiar rápidamente un estilo y se puede reducir la cantidad de código CSS que se escribe.

¿Qué es la especificidad en CSS?

La especificidad en CSS se refiere a la jerarquía de reglas que determina qué estilo se aplica a un elemento. Los navegadores utilizan la especificidad para determinar qué estilo definido se aplicará a un elemento. La especificidad se puede evaluar en dos partes: los estilos aplicados en línea en el HTML o los estilos aplicados utilizando un selector.

Cálculo de la especificidad

La especificidad es una puntuación que determina qué declaraciones de estilos se aplican a un elemento. Se utiliza un sistema que otorga puntos a cada tipo de selector utilizado y se suman para obtener la especificidad total. Hay cuatro categorías que definen el nivel de especificidad de un selector dentro de esta jerarquía. A continuación, se muestra una lista ordenada de las cuatro categorías de especificidad:

1. Estilos en línea
2. ID
3. Clases y pseudo-clases
4. Elementos y pseudo-elementos

Calculadora de especificidad

Para medir la especificidad de un selector, se utilizan cuatro valores que representan millares, centenares, decenas y unidades.

Puedes probar una calculadora de especificidad en línea para ver cómo funciona y obtener los valores de especificidad para tus propios selectores.

Ejemplos de selectores y su especificidad

A continuación, se presentan algunos ejemplos de selectores y su correspondiente especificidad:

1. Selector universal: `*` (Especificidad: `0,0,0`)
2. Selector de tipo o etiqueta: `p`, `div`, `span` (Especificidad: `0,0,1`)
3. Pseudo-elementos: `::after`, `::before` (Especificidad: `0,0,1`)

4. Selector de clase: `.title` (Especificidad: `0,1,0`)
5. Selector de ID: `#pageTitle` (Especificidad: `1,0,0`)

Ejemplos de especificidad

Vamos a ver un ejemplo con el selector universal `*`, este selector en CSS tiene un valor de 0.0.0, quiere decir que cualquier otro selector, por mínimo que sea, tendrá mayor peso que él y lo reescribirá.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta chartset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />
        <title>Especificidad en CSS</title>
    </head>
    <body>
        <p>Soy un párrafo</p>
    </body>
</html>
```

```
/* style.css */

* {
    box-sizing: border-box;
    font-size: 1rem;
}

p {
    font-size: 4rem;
}
```

Como veis, el selector universal tiene un peso menor que el selector con un elemento específico como en este caso es la P.

Vamos a ver cómo se comportan las clases con los atributos y las pseudo-clases, ya que el selector de clase es más importante que los selectores de tipo o elementos HTML y el selector universal.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta chartset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />
        <title>Especificidad en CSS</title>
    </head>
```

```
    <body>
        <header>
            <h1>
                <span class="title">Carmen Ansio</span>
            </h1>
        </header>
    </body>
</html>
```

Si yo utilizo la clase `.title` y le digo que quiero que tenga un color con valor pink, mi `<span>` se va a pintar de color rosa

```
/* style.css */

* {
    box-sizing: border-box;
    font-size: 1rem;
    color: indigo;
}

.title {
    color: pink;
}
```

Si ponemos `span {...}` y que su color sea red, vais a ver que no se modifica:

```
/* style.css */

* {
    box-sizing: border-box;
    font-size: 1rem;
    color: indigo;
}

.title {
    color: pink;
}

span {
    color: red;
}
```

Si pusiéramos un important aquí sí que se pintaría de rojo, esto es lo que no deberíamos hacer:

```
/* style.css */

* {
```

```
    box-sizing: border-box;
    font-size: 1rem;
    color: indigo;
}

.title {
    color: pink;
}

span {
    color: red !important;
}
```

Si queremos que un elemento haga caso a la regla de estilo que estamos poniendo, lo que tenemos que hacer es que sea más específico. Para poder hacer que el `span` sea más específico que una clase lo que vamos a hacer es incluirle un ID:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta chartset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />
        <title>Especificidad en CSS</title>
    </head>
    <body>
        <header>
            <h1>
                <span class="title" id="page-title">Carmen Ansio</span>
            </h1>
        </header>
    </body>
</html>
```

```
/* style.css */

* {
    box-sizing: border-box;
    font-size: 1rem;
    color: indigo;
}

.title {
    color: pink;
}

span {
    color: red;
```

```
}

#page-title {
    color: blue;
}
```

El selector de ID es el más específico de todos los selectores de CSS.

Dejando de lado el ejemplo con el ID, intentamos hacer que nuestro selector `span` tenga más especificidad que la clase `.title`, parra ello debemos incluir un selector más complejo.

Vamos a fijarnos en cómo está construido el DOM y en la anidación de nuestros elementos, en este caso tenemos un `header` con un `h1` y el `span`. Si utilizamos esta anidación y además añadimos la clase `.title` de la siguiente manera esto crea un selector mucho más específico que la clase `.title` y se pintará de color rojo el texto.

```
/* style.css */

* {
    box-sizing: border-box;
    font-size: 1rem;
    color: indigo;
}

.title {
    color: pink;
}

header h1 span.title {
    color: red;
}
```

Igualmente no hace falta añadir todos los elementos, es decir, nosotros diciéndole al `span` que tiene una clase `.title` ya funcionaría.

```
/* style.css */

* {
    box-sizing: border-box;
    font-size: 1rem;
    color: indigo;
}

.title {
    color: pink;
}

span.title {
```

```
    color: red;
}
```

Recuerda que los estilos en línea tienen mayor especificidad y siempre sobrescriben cualquier otro estilo.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta chartset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />
        <title>Especificidad en CSS</title>
    </head>
    <body>
        <header>
            <h1>
                <span style="color: green" class="title" id="page-title"
                    >Carmen Ansio</span
                >
            </h1>
        </header>
    </body>
</html>
```

Es recomendable mantener los estilos en una hoja de estilos externa y evitar el uso de `!important` a menos que sea necesario en casos especiales.

Métodos para controlar la especificidad

Existen diversas metodologías y arquitecturas CSS, como la metodología BEM, que ayudan a controlar la especificidad y a mantener los estilos organizados. Estas metodologías son útiles para evitar conflictos y anulaciones de estilos entre sí.

Recuerda que los navegadores utilizan la especificidad para decidir qué reglas CSS aplicar a los elementos. Comprender y aplicar adecuadamente la especificidad en CSS es fundamental para lograr estilos coherentes y consistentes en tus proyectos.