# CSS Introducción.

## Contenido



## 1. Linkar el archivo de estilos.

En la cabecera del documento (<head>) crear un link

<link rel=”stylesheet” href=”ruta archivo”

## 2. Sintaxis:

```
selector {
        propiedad:valor;
        propiedad2: valor2;
        ……
}
```

## 3. Prefijos: reglas destinadas a navegadores en concreto.

Actualmente los navegadores tienen implementadas muchas de las nuevas características de CSS3 utilizando sus propias versiones de cada propiedad mediante prefijos.

Esto se hace así para evitar los posibles errores ocasionados por las primeras implementaciones que aún no son estables. Por ello, los navegadores proporcionan valores utilizando sus prefijos propios y una declaración sin prefijo.

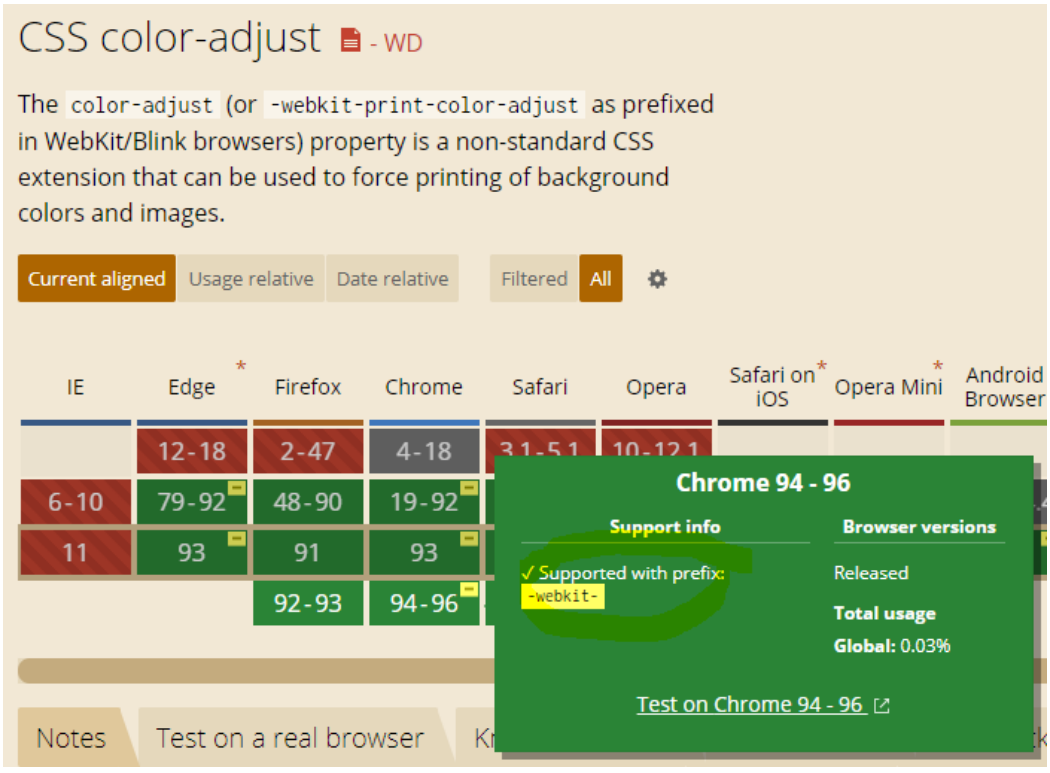
Después de un tiempo, cuando las especificaciones son estables, se eliminarán las propiedades con prefijo.

-webkit-valor

| Prefijo | Navegador |
|---|---|
| -moz- | Firefox |
| -webkit- | Safari y Chrome |
| -o- | Opera |
| -khtml- | Konqueror |
| -ms- | Internet Explorer |
| -chrome- | Google Chrome |

[Can I use](#)

Prefijos a las propiedades de CSS experimentales o no estándar y las API de JavaScript




Automáticamente:

1. Instalar Extensión Autoprefixer para Visual Studio Code
2. Escribir CSS
3. Ctrl+shift+p –> Escribir autoprefixer run

Ejemplo:


## 4. Selectores básicos:

**Etiqueta html=selector**

a) **Universal :**
*** {}**

b) **Básicos:**
Para combinar selectores afectados por las mismas reglas separarlos por comas antes de la apertura de llaves.
*Ejemplo:* p, h1, h2, span{estilos}

c) **Descendente:**
***Selector_padre*** **(espacio_en blanco) selector_hijo** {
Las reglas afectan a los selectores de tipo hijo que anteriormente tenga el selector previo (no tiene por qué ser hijo directo).
*Ejemplo: p span a*

d) **Clase:**
**. clase** {}
Esta clase tiene que estar definida en el html del selector con el atributo class=””
La misma clase puede ser utilizada por varias etiquetas y un elemento html puede tener más de una clase definida en el atributo class

e) **Restringir el selector clase:**
**selector.clase**
Las reglas afectarán a los selectores de ese tipo cuya clase sea clase
*Ejemplo: a.destacado {*
Las reglas afectarán a los selectores tipo a (enlaces) cuya clase sea destacado.

f) **Restringir afección regla selector clase múltiple**
**.clase1.clase2 {estilos}**
Las reglas afectarán a los selectores que tengan al menos ambas clases (podría tener más, pero no menos de esas 2)
*Ejemplo:*
*.error {color:verde}*

*.destacado {color:red}*
*.especial {color: blue}*
*. error.destacado {color:violet}*

*Si una etiqueta html tuviera en class todas las clases, se aplicarían las reglas correspondientes a .error.destacado*

**g) id:**

**#id{estilos}**

El id estará asignado en alguna etiqueta html (que debe ser único en todo el documento)

## 5. Selectores avanzados:

**h) Hijo directo:**

**selectorpadre>selector_hijo**

Depende del selector padre (no aplica a un hijo del selector_hijo)

*Ejemplo:*

*p a → Cualquier enlace que esté dentro de un p (independiente de su situación/relación con el p.*

*p>a → Solo los enlaces que sean hijos directos del p*

**i) Adyacente**

**selector1+selector2{}**

Selector1 y selector2 deben ser hermanos (mismo padre)

Selector2 debe aparecer inmediatamente después de selector1 (hermanos mellizos)

Las reglas afectan al selector2

**j) Hermanos**

**selector1~selector2{}**

Todos los elementos de tipo selector2 que sean hermanos de selector1

**k) Selector de atributo**

1) **[atributo]=valor→ Todos los elementos que tengan el atributo = valor**

   *Ejemplo*:

   [color=”red”] {}

2) **selector [atributo]=valor→Todos los elementos de tipo selector que tengan el atributo=valor**

   *Ejemplo*:

   a[color=”red”] {}

3) **[atributo~=valor]→ elementos que tienen en el atributo al menos el valor.**

   *Ejemplo*:

   [class ~=”externo”]{}

   En el html puede haber varios elementos así que serían afectados:

<p class=”externo clásico verde”>XXXX</p>

También es válido: selector[atributo]~=valor

4) **[atributo|=”valor”]→ elementos que tienen el atributo = a una <u>serie de palabras</u> cuyo valor puede ser exactamente valor o puede comenzar con valor seguido inmediatamente por un guion.**
   *Ejemplo:*
   [class|=”ext”] → Todos los elementos cuya clase empiece por ext o ext-* (ext, ext-erior, ext-erno,ext-erior,…..)
   También es válido: selector[atributo]|=”valor”

5) **[atributo^=”valor”]→ Elementos cuyo atributo comienza exactamente por la subcadena valor.**
   *Ejemplo:*
   [class^=”ext”]→ Todos los elementos cuya clase empiece por ext (ext, exterior, externo,…)

6) **[atributo$=”valor”]→ Elementos con el atributo que termina con la subcadena valor**
   *Ejemplo:*
   [class$=”ado”]→Elementos cuya clase acaba en ado (ado, abollado, acabado, empezado,....)

7) **[atributo*=”valor”] → Elementos con el atributo que contiene la subcadena valor**
   *Ejemplo:*
   [class*=”ado”]→ Elementos cuya clase tiene ado en cualquier posición (ado, prados,alado ...)

l) **Pseudoclases: estados de los elementos elemento:estado{} (excepto :not)**
   1) **:first-child: el primer hijo de un elemento**
   2) **:last-child: el ultimo hijo de un elemento**
   3) **: focus: se activa cuando el elemento tiene el foco**
   4) **: hover: se activa al pasar el ratón por encima**
   5) **:active: cuando se pulsa el ratón (imperceptible)**
   6) **:visited: cuando ha sido visitado (enlaces)**

*Ejemplo1:*

Elemento a:

: link → estado normal

: visited→ visitado

: hover → cursor encima

: active→ click encima

a: link{color:blue}

a: visited{color:red}

*Ejemplo2:*
p em:first-child{}→ el primer elemento de tipo em (cursiva) que esté dentro de cualquier p

7) **:not(condición)→selecciona los que no cumplen la condición**
   :not(#especial)→ selecciona el elemento que no tienen como id=especial
   :not(p)→ Selecciona los elementos que no sean párrafos
   :not(.externo)→ Selecciona los elementos que no tengan la clase externo

8) **:lang→ Seleccionar elementos según el idioma**
   :not(*:lang(es))→ Seleccionar elementos que no estén en castellano

9) **:first-child→ selecciona el elemento con la condición de que sea el primer hijo (de quien sea de sus descendientes, pero primer hijo)**

10) **:last-child→ selecciona el elemento con la condición de que se el ultimo hijo**

11) **:nth-child(n)→ selecciona en el caso de que sea el hijo enésimo del padre**
    - **ood: hijos impares, even: hijos pares**
    - **n=expresión regular (2n, 2n+4, n-2,…)**

12) **:nth-last-child(n)→ selecciona en el caso de que sea el hijo enésimo del padre contando desde el final**

13) **:empty→ selecciona en el caso de que no tenga ningún hijo**

14) **selector:nth-of-type(n)→ selecciona el elemento en el caso de que sea el enésimo elemento de este tipo**

15) **selector:nth-last-of-type(n) → selecciona el elemento en el caso de que sea el enésimo elemento de este tipo contando desde el final**

m) **Pseudoelementos:**
   **selector::pseudoelemento**
   **1)** ::first-line → Selecciona primera línea del elemento
   **2)** ::first-letter→ Selecciona primera letra del elemento
   **3)** ::before {content: “texto”}→ Añade el texto antes del selector
   **4)** ::after{content: “texto”}→ Añade el texto después del selector
   **5)** ::selection → Selecciona el texto seleccionado por el usuario