

CSS

Primeros Pasos

BIANCHI - ZULAICA

Tabla de Contenidos

1. Introducción	4
Definición	4
Sintaxis	4
Vinculación	5
Externa	5
Interna	5
En Linea	5
2. Selectores	7
Basicos	7
Etiqueta	7
Clase	7
Id	8
Universal	8
Combinados	8
Anidados	9
Multiples	9
Combinadores	10
Descendientes ()	10
Descendientes directos (>)	11
Hermano siguiente (+)	12
Hermanos siguientes (~)	13
Pseudo-Selectores	14
Pseudo Clases	14
Pseudo Elementos	16
3. Orden de Aplicación	17

4. Herramientas (DevTools)	19
Abrir Herramientas	19
Inspección y Edición	22
Contenido	22
Estilo	23
Guardar Cambios	25
Editores Interactivos	26
5. Colores	27
Atributos de Color	27
Opacidad	27
Formatos de Color	28
Colores Nombrados	28
Hexadecimal	28
RGB & RGBA	29
HSL & HSLA	29
Otros	30
6. Texto	31
Atributos de Texto	31
Color	31
Tamaño	31
Familia	32
Peso	33
Alineación	34
Otros	34
Fuentes	35
7. Modelo de Cajas	37
Componentes de Caja	38
Tamaño de Caja	38
Relleno	39
Borde	40
Margen	41

8. Posición y Presentación	41
Posición	41
Estatica	41
Relativa	41
Fija	41
Absoluta	41
Pegajosa	41
Presentación	42
En Línea	42
Bloque	42
Bloque en Línea	42
Ninguna	42
Flexible (Flexbox)	42
Grilla	42
9. Variables y Funciones	43
Variables	43
Funciones	45
10. Referencias	46
Unidades de medida	46
Referencias externas	46

1. Introducción

Definición

Cascading Style Sheets, CSS, es el lenguaje que define la apariencia de la web. Podemos traducir sus siglas a español como Hojas de Estilo en Cascada.

Hojas de Estilo refiere al aspecto que da al contenido html. Propiedades como el color, el texto, el posicionamiento y otros que pueden ser editadas.

Mientras que la Cascada hace referencia al orden en el que se aplican las reglas, donde una regla puede sobreescribir a otra si cumple ciertos criterios.

Sintaxis

Un archivo .css tendrá varias reglas, donde cada regla está compuesta por:

- Un **selector** que determinará los elementos que se verán afectados
- Un conjunto de **propiedades** que definirán su aspecto.

```
selector {  
    Propiedad 1: valor 1;  
    Propiedad 2: valor 2;  
    ...  
    Propiedad N: valor N;  
}
```

Vinculación

1. Externa

Debemos utilizar la etiqueta `<link>` dentro del html, preferiblemente dentro de head, para vincular un archivo de estilo externo.

El atributo **rel** definirá la relación del archivo, que en este caso será *stylesheet* para hojas de estilo; mientras que el atributo **href** será la ruta al archivo.

```
<link rel='stylesheet' href='estilo.css'>
```

HTML

Este tipo de **vinculación** es la **recomendada**, porque permite separar el contenido (código HTML), del estilo (reglas CSS); y así, reutilizar las reglas CSS escritas, relacionando el mismo archivo .css con las páginas HTML que sea conveniente relacionar. Esto posibilita dar un estilo homogéneo a nuestro sitio de modo más fácil y seguro; y ante futuros cambios, será más simple y eficiente, realizarlos.

2. Interna

Podemos utilizar la etiqueta `<style>` dentro del html, preferiblemente dentro de head, y definir las reglas directamente en su contenido.

```
<style>
  div { color: red; }
</style>
```

HTML



Se recomienda evitar, ya que es preferible separar los estilos del contenido.

3. En Línea

Podemos utilizar el **atributo style** para definir las propiedades de estilo que aplican a una etiqueta de manera directa

```
<div style='color: red;'>
  ...
</div>
```

HTML



No se recomienda este uso, ya que obliga a repetir estilos comunes a varios elementos.

Ejemplo

Utilizando vinculación externa

```
<h1>CSS</h1>
<p>
    El lenguaje que le da su <b>apariencia</b> a la web
</p>
```

HTML

```
* {
    font-family: sans-serif;
    background-color: #5076da;
    text-align: center;
}
h1 {
    color: #152b59;
    text-shadow: 0px 0px 6px #3369;
}
p { color: white; }
```

CSS



2. Selectores

Basicos

Etiqueta

Podemos seleccionar todas las etiquetas html de cierto tipo utilizando el selector de etiqueta, para esto simplemente escribimos su **nombre**.

```
<p>Hola Mundo</p>
```

HTML

```
p { color: red; }
```

CSS

Clase

El **atributo class** de una etiqueta html es usado para asignarle una clase ó tipo, luego, en css los elementos de una clase son referidos utilizando un **punto (.)** seguido por el nombre de la clase (sin espacios).

En general, varios elementos pueden tener la misma clase.

```
<p class='verde'>Hola Mundo</p>
<strong class='verde'>Hola Mundo</strong>
```

HTML

```
.verde { color: green; }
```

CSS

Id

El **atributo id** de una etiqueta html es usado para asignarle un identificador, luego, en css podemos referirnos al elemento utilizando un **numeral (#)** seguido por el nombre de la clase (sin espacios).

Los identificadores son únicos; es decir, no habrá dos elementos distintos con un mismo id en una misma página.

```
<p id='azul'>Hola Mundo</p>
```

HTML

```
#azul { color: blue; }
```

CSS

Universal

El selector universal, **asterisco (*)**, selecciona **todos** los objetos html.

```
* { font-family: Roboto, Arial, sans-serif; }
```

CSS

Combinados

Al separar selectores con un **espacio** podemos estilar elementos sólo si son descendientes de otros, es decir, si se encuentran dentro.

```
ul li { font-family: sans-serif; } ➡ asigna la familia de tipografía
```

sans-serif los elementos *li* dentro de *ul*

```
nav * { color: red; } ➡ le otorga a todos los elementos dentro de un nav
```

el color rojo

```
.envio button { background-color: green; } ➡ los botones dentro de un
```

elemento con clase “envio” serán de color verde

CSS

Anidados

Anidando selectores de clase ó id, es decir, colocándolos sucesivamente (sin espacios) podemos seleccionar elementos que cumplan varios criterios al mismo tiempo.

En líneas generales la sintaxis es: `etiqueta.clase1.clase2#id`

```
<p>Texto Plano</p>
<p class='estilado'>Rojo</p>
<span class='estilado importante'>Negrita</span>
```

HTML

```
p.estilado { color: red; }
.estilado.importante { font-weight: bold; }
```

CSS



Aquí podemos ver como un elemento puede tener múltiples clases.

Múltiples

Cuando tenemos un estilo que es común a múltiples elementos, podemos separar varios selectores utilizando la **coma (,)** para que aplique a todos ellos.

```
input, select, textarea {
    width: 100%;
    padding: 0.5rem 1rem;
}
```

CSS

Combinadores

Descendientes ()

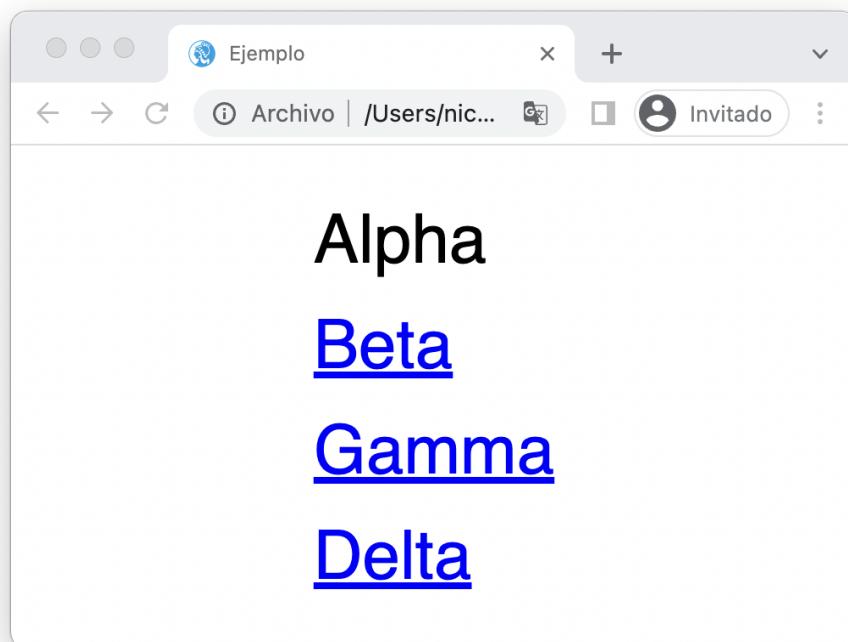
El selector de **descendientes**, usando el carácter “ ” (espacio), selecciona los elementos que estén dentro de otro en cualquier nivel.

```
<section>
  Alpha
  <div>
    <p>Beta</p>
  </div>
  <p>Gamma</p>
  <p>Delta</p>
</section>
```

HTML

```
section p { color: blue; text-decoration: underline; }
```

CSS



Descendientes directos (>)

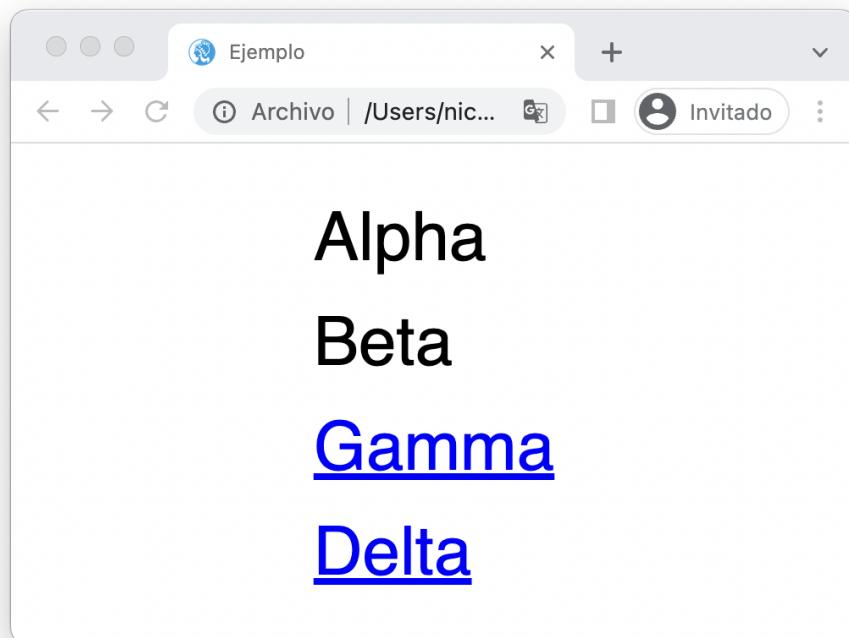
El selector de **descendientes directos**, usando el carácter “>” (mayor), selecciona los elementos que estén directamente dentro de otro.

```
<section>
  Alpha
  <div>
    <p>Beta</p>
  </div>
  <p>Gamma</p>
  <p>Delta</p>
</section>
```

HTML

```
section > p { color: blue; text-decoration: underline; }
```

CSS



Hermano siguiente (+)

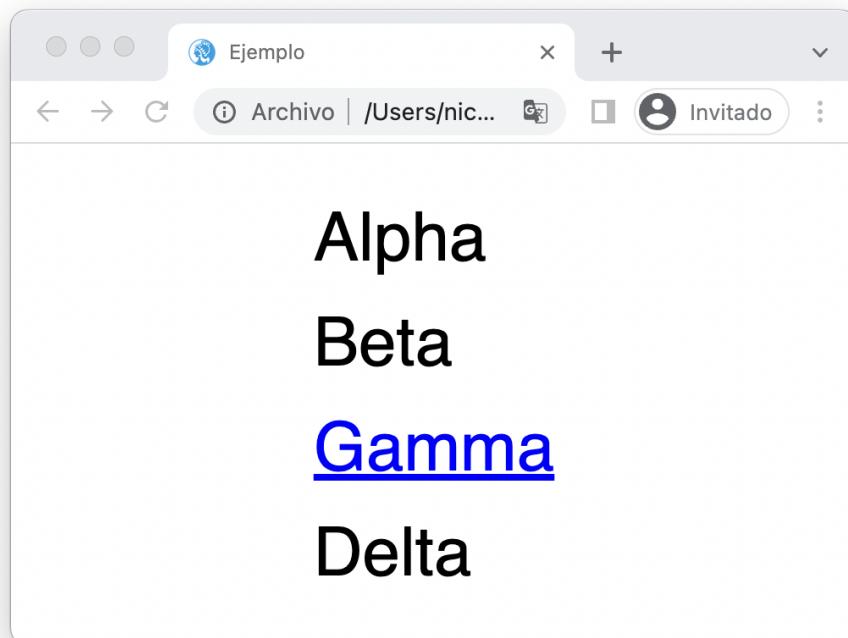
El selector de **hermano siguiente**, usando el carácter “+” (más), selecciona el primer elemento que cumpla el selector luego de otro.

```
<section>
  Alpha
  <div>
    <p>Beta</p>
  </div>
  <p>Gamma</p>
  <p>Delta</p>
</section>
```

HTML

```
div + p { color: blue; text-decoration: underline; }
```

CSS



Hermanos siguientes (~)

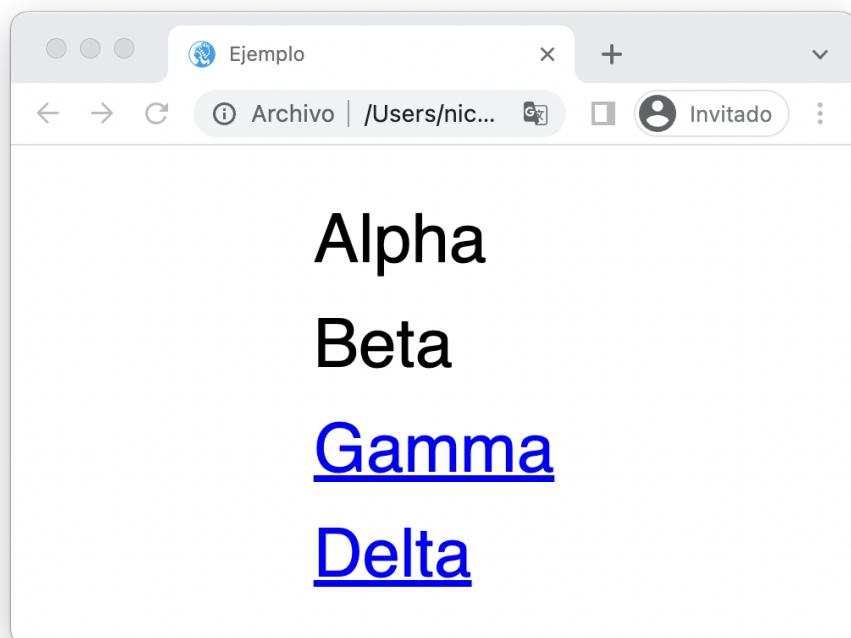
El selector de **descendientes**, usando el carácter “~” (virgulilla), selecciona los elementos que cumplan el selector luego de otro.

```
<section>
  Alpha
  <div>
    <p>Beta</p>
  </div>
  <p>Gamma</p>
  <p>Delta</p>
</section>
```

HTML

```
div ~ p { color: blue; text-decoration: underline; }
```

CSS



Pseudo-Selectores

Pseudo Clases

Utilizamos pseudo clases para seleccionar elementos cuando se encuentran en un **estado especial**. Se utilizan del modo **selector:pseudo-clase**.

Podemos estilar elementos al pasarles por arriba con el mouse utilizando **:hover**, también podríamos estilar un botón al presionarlo con **:active**.

```
button {  
    background-color: #55d;  
    box-shadow: 0 0 2px 0 #0004;  
}  
  
button:hover {  
    background-color: #44b;  
    box-shadow: 0 0 3px 0 #0008;  
}
```

CSS

Presionar

Presionar



Utilizamos **:disabled** para elementos deshabilitados, lo cual es muy común al trabajar con interacciones de usuario, ejemplo: formularios.

```
* :disabled {  
    cursor: not-allowed;  
}  
  
textarea:disabled {  
    background-color: #a331;  
    border: 1px solid #a33;  
}
```

CSS

Nombre

John Doe

País

Argentina



Mensaje

El pseudo selector de `:nth-child`, o enésimo hijo en español, es útil en tablas o cuando tenemos elementos repetitivos. Este nos permite editar un hijo según su número (1: primero, 2: segundo, etc) ó incluso grupos de elementos.

```
table tr:nth-child(1) {  
    background-color: #152b59;  
    color: white;  
}  
  
table tr:nth-child(2n) {  
    background-color: #f4f4f4;  
}  
  
table tr:nth-child(2n+3) {  
    color: #237;  
}
```

CSS

Nombre	Apellido
Emiliano	Martinez
Lionel	Messi
Enzo	Fernández
Ángel	Di María
Julián	Álvarez
Sergio	Aguero

⚠ El estilado de la cabecera utilizando `:nth-child` es a fin de ejemplo, en la práctica es mejor dar estilo a los th o seleccionar la tabla con `thead` y `tbody`.

Vemos en el ejemplo que podemos usar `:nth-child(2n)` para estilar las filas **pares** (también podríamos haber usado even).

Para estilar las filas **impares** podríamos usar `:nth-child(odd)` o `:nth-child(2n+1)` que es igual al de pares sumando un **desfase**. Lo que conseguimos con desfase +3 es ignorar la primera fila.

También es común al usar tablas estilar las filas al pasar con el mouse:

```
tbody tr:hover {  
    background-color: #dde;  
    border: 2px solid black;  
}
```

CSS

Lionel	Messi
Enzo	Fernández
Ángel	Di María
Julián	Álvarez

Pseudo Elementos

Utilizamos pseudo elementos para seleccionar **partes específicas** de un elemento. Se utilizan del modo **selector::pseudo-elemento**.

Los pseudo elementos más comunes son **::before** y **::after** que agregan una caja de texto antes y después del contenido de un elemento.

```
<p>Hola Mundo</p>
```

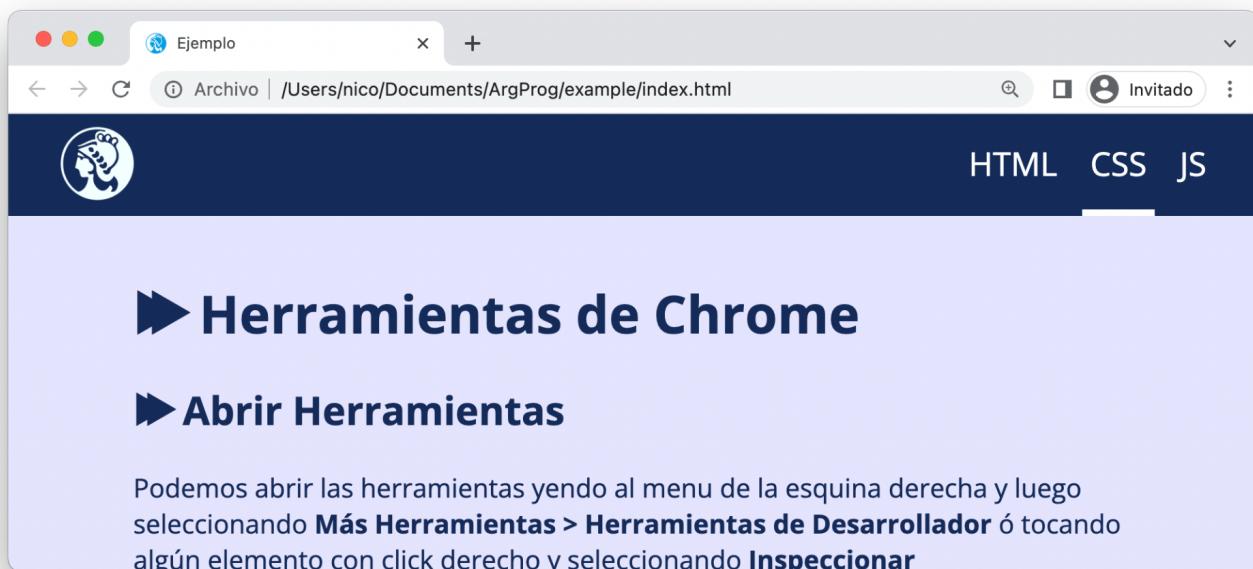
HTML

```
p::before { content: 'i'; }  
p::after { content: '!'; }
```

CSS

i Hola Mundo !

Estos elementos se suelen utilizar con fines decorativos, por ejemplo para marcar la página en una barra de navegación o agregar iconos a los títulos.



3. Orden de Aplicación

Orden de Aparición

Cuando varias reglas editan un mismo atributo de un elemento, aplicará la regla de mayor prioridad según la “cascada” de criterios.

El primer criterio, y menos importante, es el **orden de aparición** en el código; de modo que, si dos reglas tienen los mismos selectores, la última en aparecer tendrá precedencia.

```
p {  
    color: red; /* NO APLICA */  
    font-weight: bold; /* SI APLICA */  
}  
  
p {  
    color: blue; /* SI APLICA */  
}
```

css

⚠ En general, trataremos de no depender de este criterio, ya que no es claro

Especificidad

El siguiente criterio será la **especificidad**, esto refiere a que tan específico es un selector. En líneas generales, podemos decir que un selector es más específico mientras más descriptivo es del elemento seleccionado.

```
* { /* MENOS ESPECIFICO */ }  
main { ... }  
main p { ... }  
body main p { ... }  
main p.nota { /* MÁS ESPECIFICO */ }
```

css

ℹ Encontraremos el cálculo exacto de especificidad en la [documentación de MDN](#)

Origen

El tercer criterio es el **origen** de la regla; por ejemplo, hay estilos del navegador ó del usuario que tienen menos precedencia que los que definamos en css. Normalmente no podremos editarlos y pueden diferir entre usuarios, lo cual debemos tener en cuenta si queremos que una página se vea igual para todos.

Importancia

Por último, el criterio más prioritario es la **importancia**; para marcar un atributo como importante agregamos un “**!important**” luego del valor.

```
/* Quitar subrayado de los enlaces */  
a {  
    text-decoration: none!important;  
}
```

CSS

⚠ Usar lo menos posible, ya que pisa todos los estilos, dificultando la edición y causando inconvenientes a usuarios con estilos definidos.

Para resumir, tenemos 4 criterios de desempate para la aplicación del estilo; normalmente deberíamos utilizar la **especificidad** para que los estilos queden bien definidos y en algunos casos podríamos necesitar la **importancia**.

Orden de Aparición

Especificidad

Origen

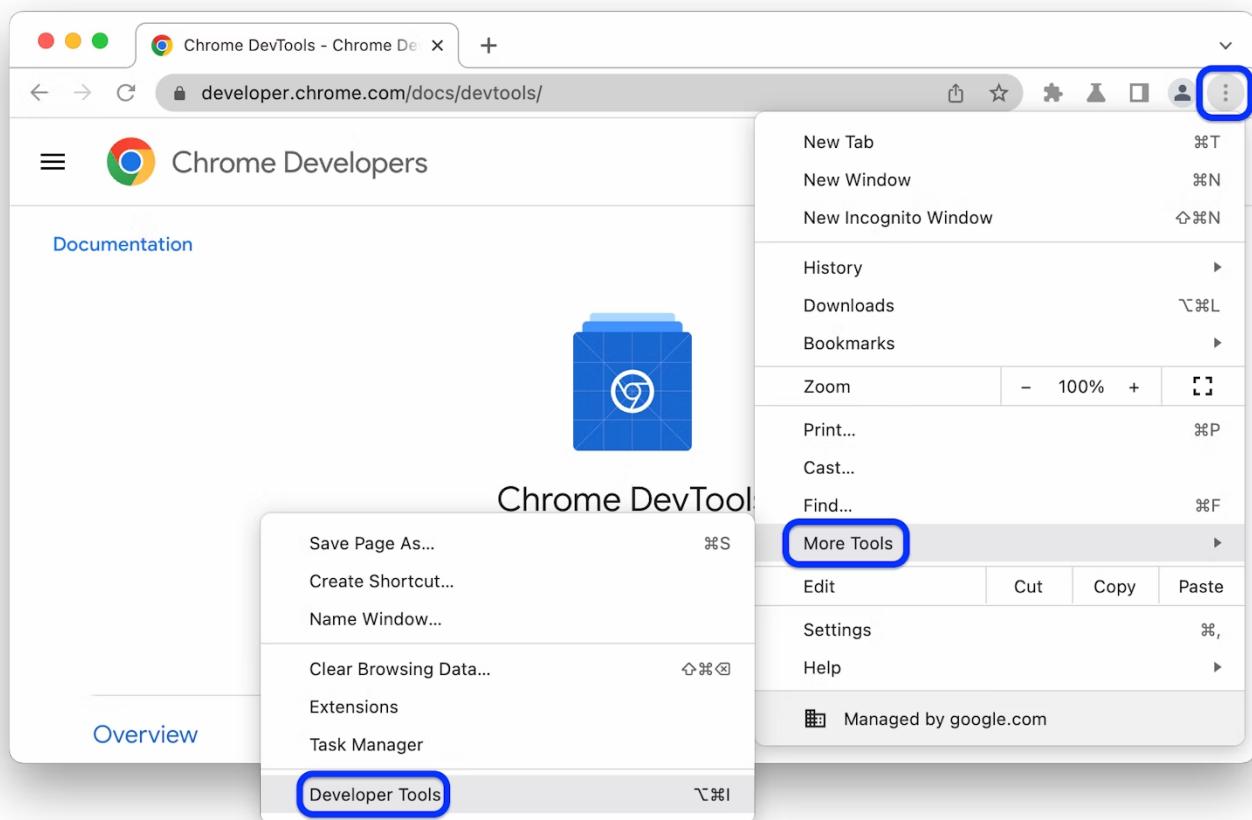
Importancia

4. Herramientas (DevTools)

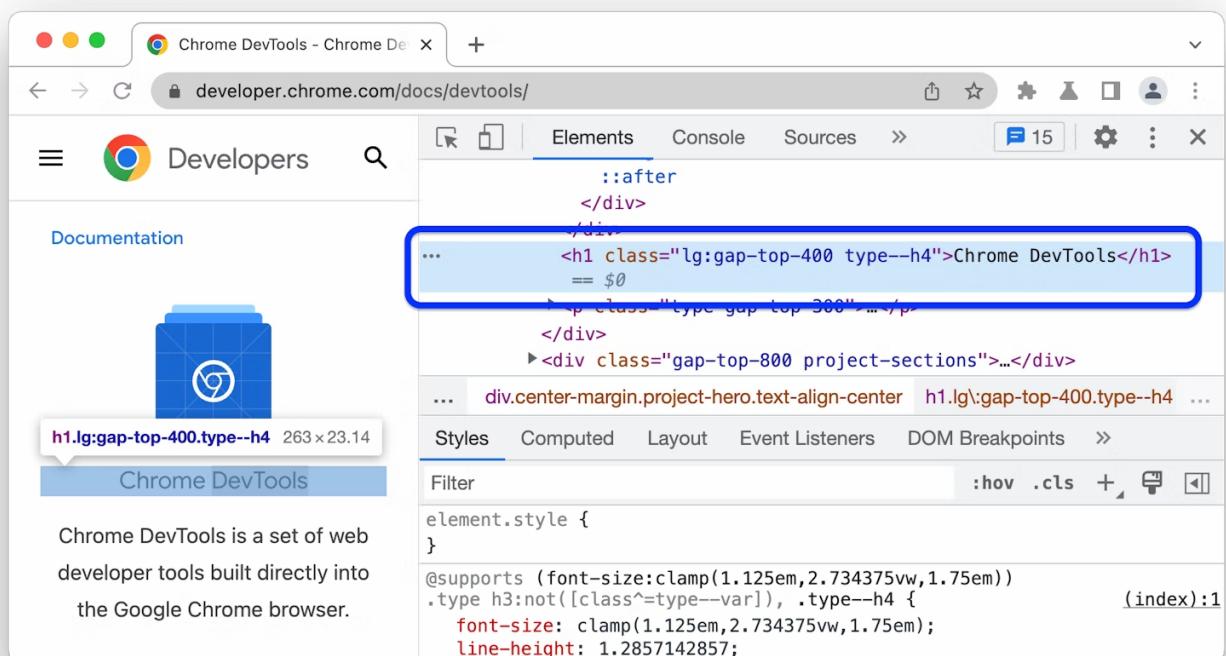
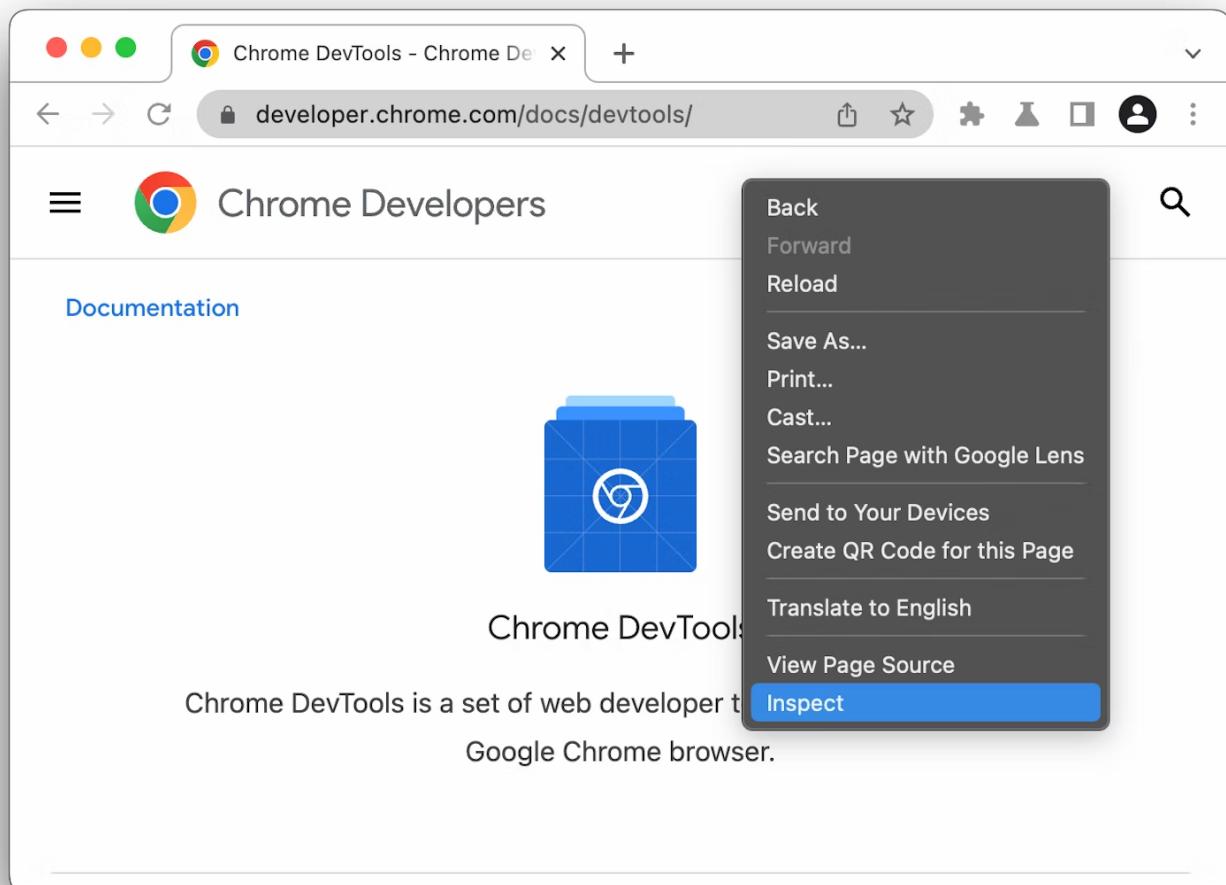
Los navegadores modernos nos proveen con herramientas que nos facilitan el desarrollo, veremos cómo utilizar las que proporciona **google chrome**.

Abrir Herramientas

Podemos abrir las herramientas yendo al menú (⋮) de la esquina superior derecha y luego seleccionando **Más Herramientas > Herramientas de Desarrollador**.



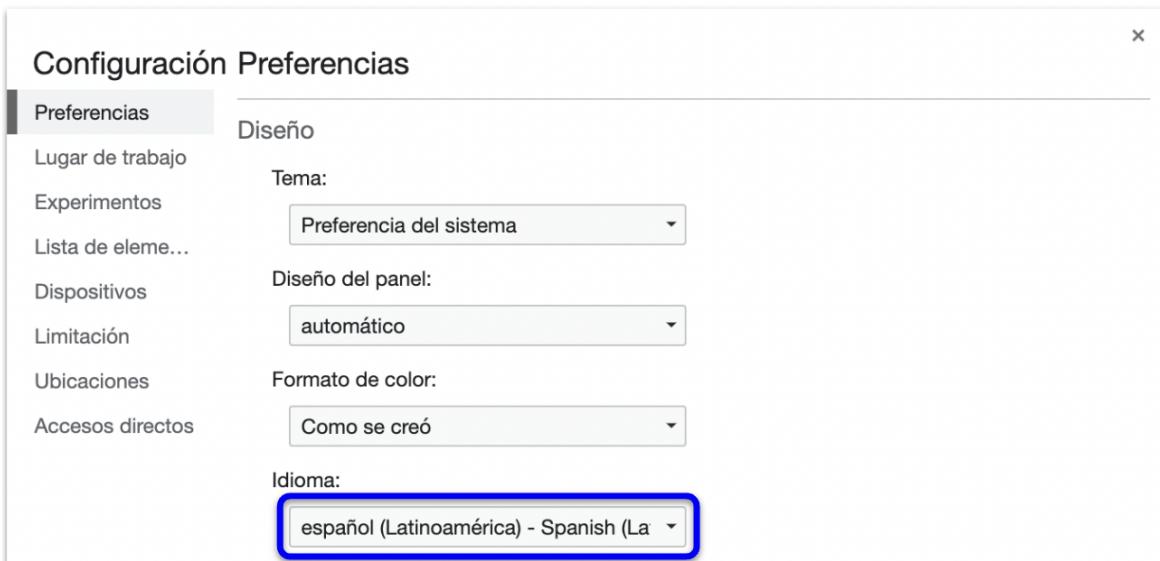
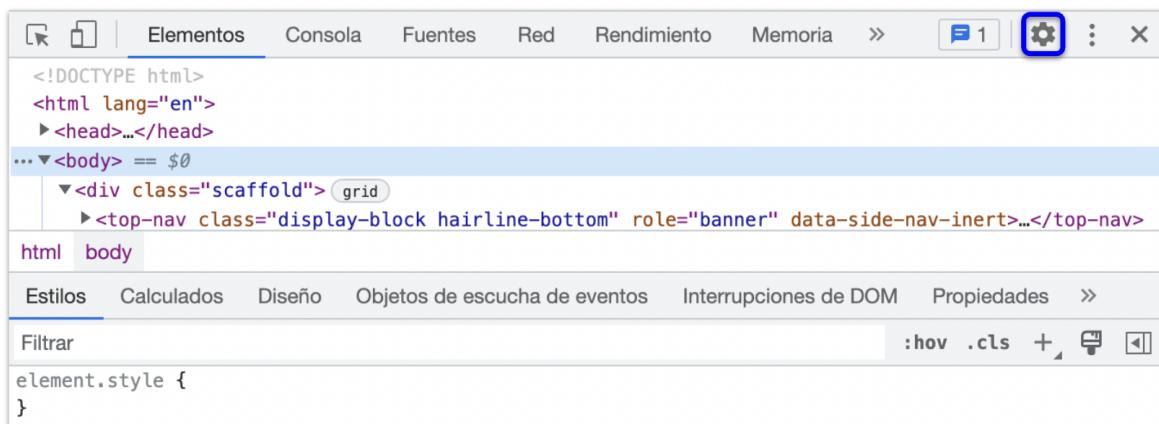
También, al tocar algún elemento con click derecho y luego la opción **Inspeccionar**, se abrirán las herramientas y se seleccionará el elemento.



Por último, podemos utilizar atajos de teclado para abrir diferentes pestañas:

Sistema Operativo	Elementos	Consola	Ultima Pestaña
Windows o Linux	Ctrl + Shift + C	Ctrl + Shift + J	F12 Ctrl + Shift + I
Mac	Cmd + Option + C	Cmd + Option + J	Fn + F12 Cmd + Option + I

Podemos cambiar el lenguaje de las herramientas a español yendo a la configuración de idioma.



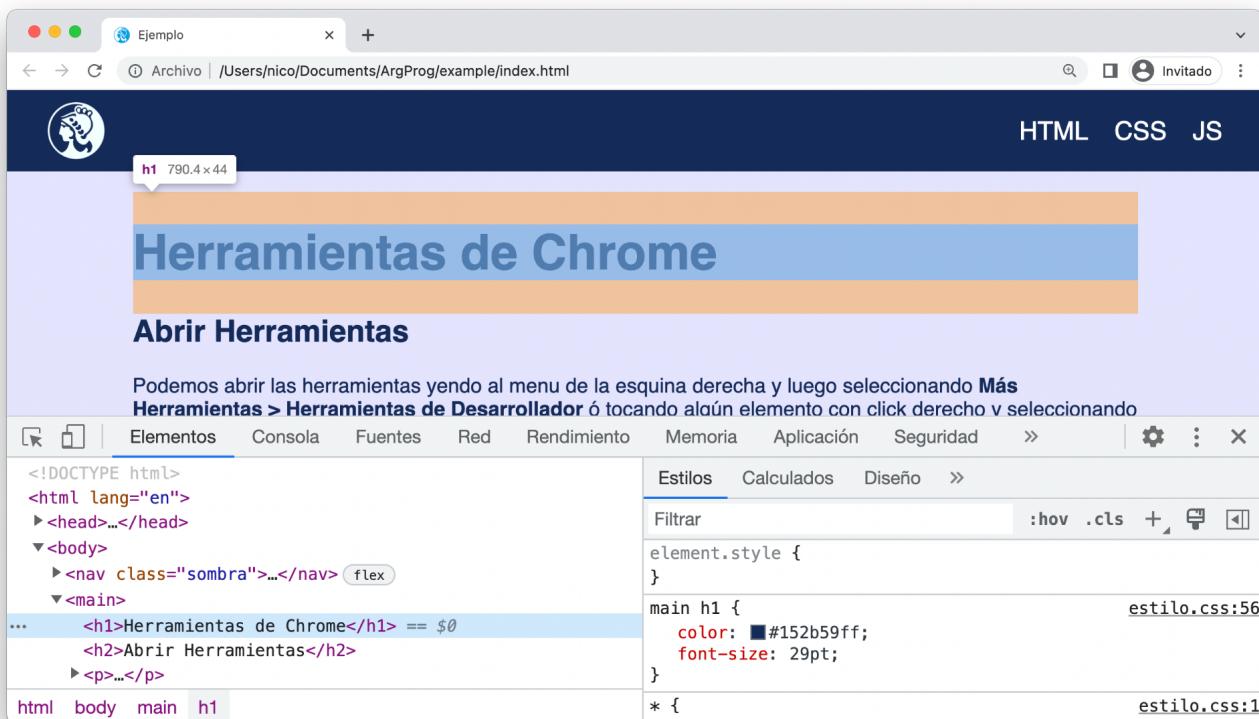
Inspección y Edición

Contenido

Para editar el **contenido** podemos dar *doble clic* tanto sobre los nombres de las etiquetas, sus atributos o el contenido de texto.

También, podemos hacer *clic derecho* sobre una etiqueta para ver más opciones, donde la opción de **Editar como HTML** nos da la mayor versatilidad.

Con esto podríamos editar el contenido existente, por ejemplo: agregar una clase nueva a un elemento, o incluso crear nuevas etiquetas.



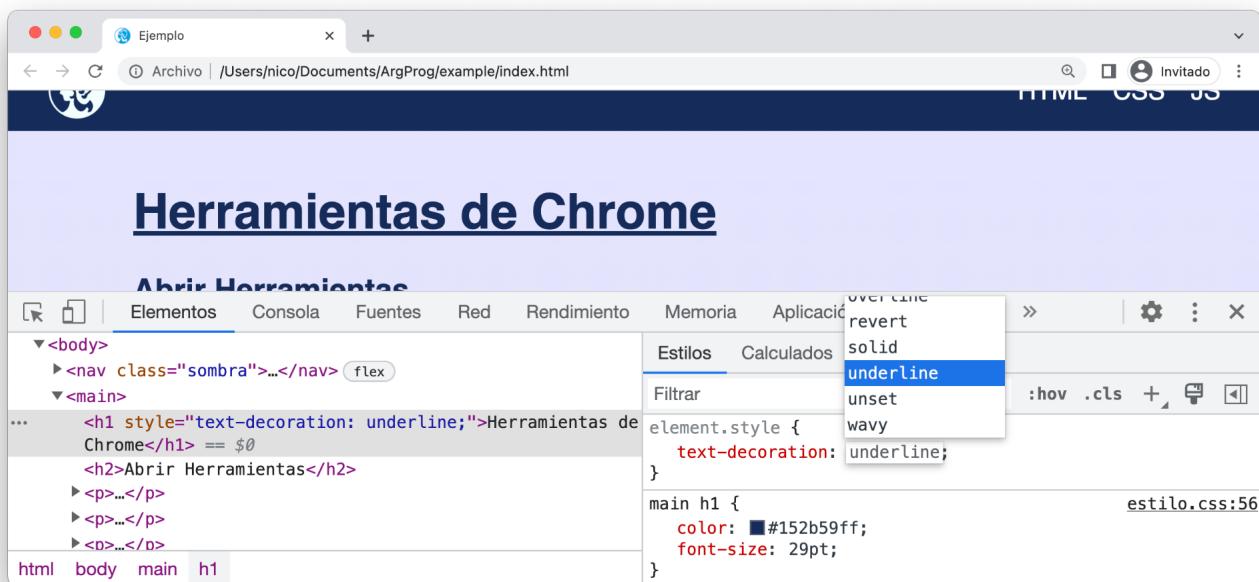
El botón de inspección (arriba a la izquierda de las herramientas) nos permite seleccionar contenido con el puntero directamente sobre la página.

Podemos utilizar **Ctrl +** en windows o **⌘ +** en macos para aumentar el zoom y **Ctrl -** en windows o **⌘ -** en macos para disminuir el zoom.

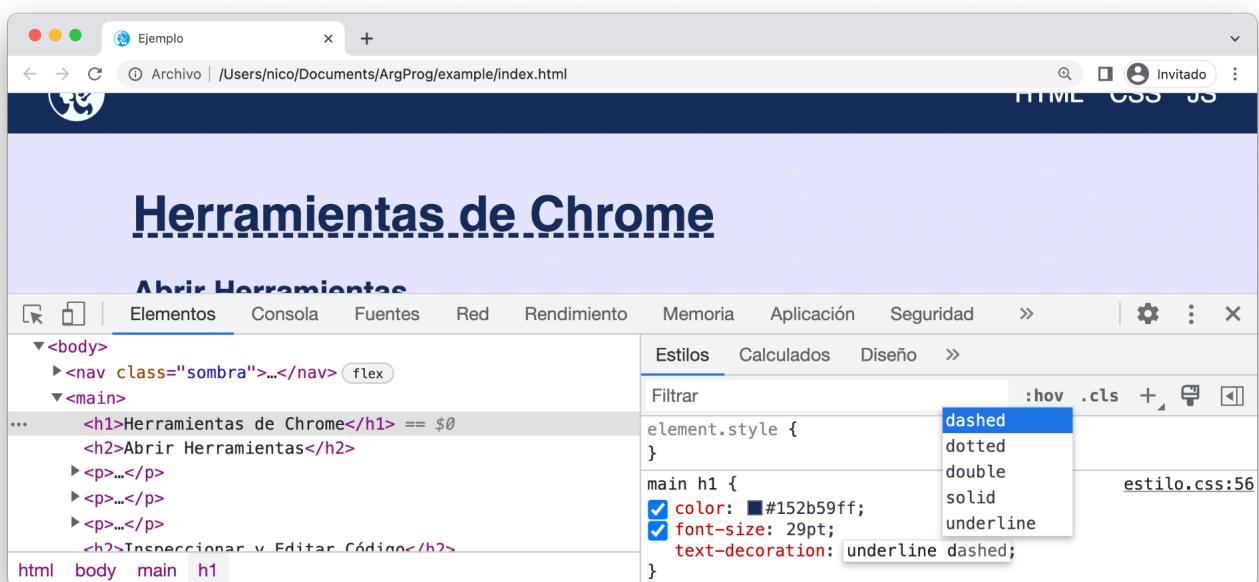
Estilo

Podemos editar el **estilo** sobre las reglas visibles en el panel a la derecha.

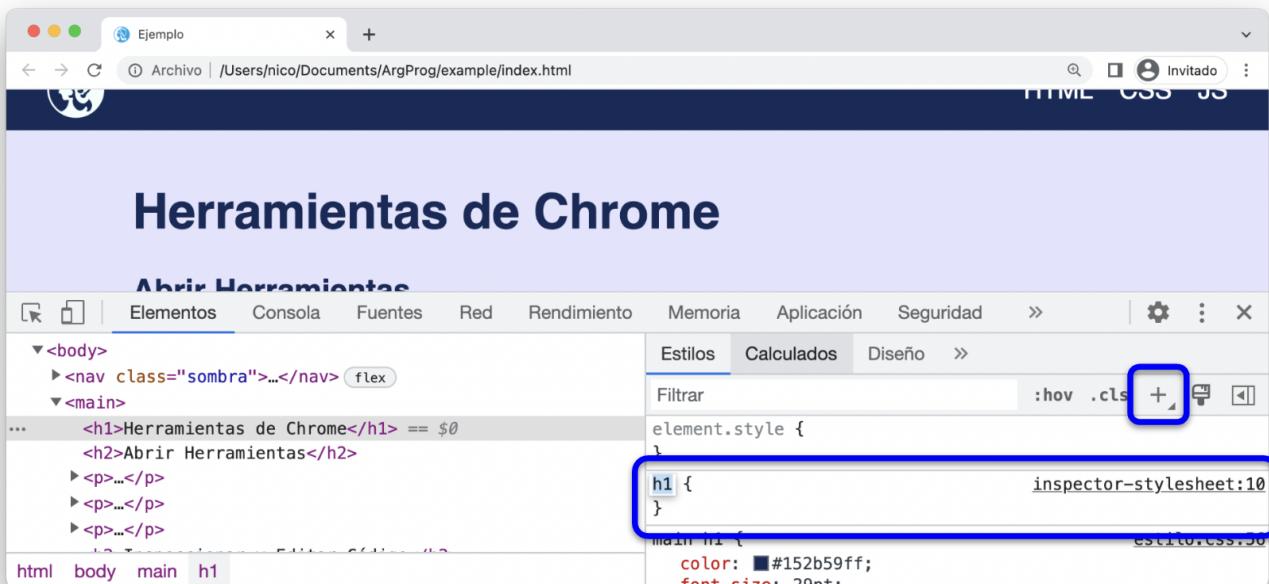
El selector `element.style` aplicará a los estilos en línea de la etiqueta, veremos como se refleja en el atributo `style` de la etiqueta.



Podemos editar las reglas preexistentes en el css; al seleccionar un elemento veremos las reglas cuyos selectores abarquen a dicho elemento.

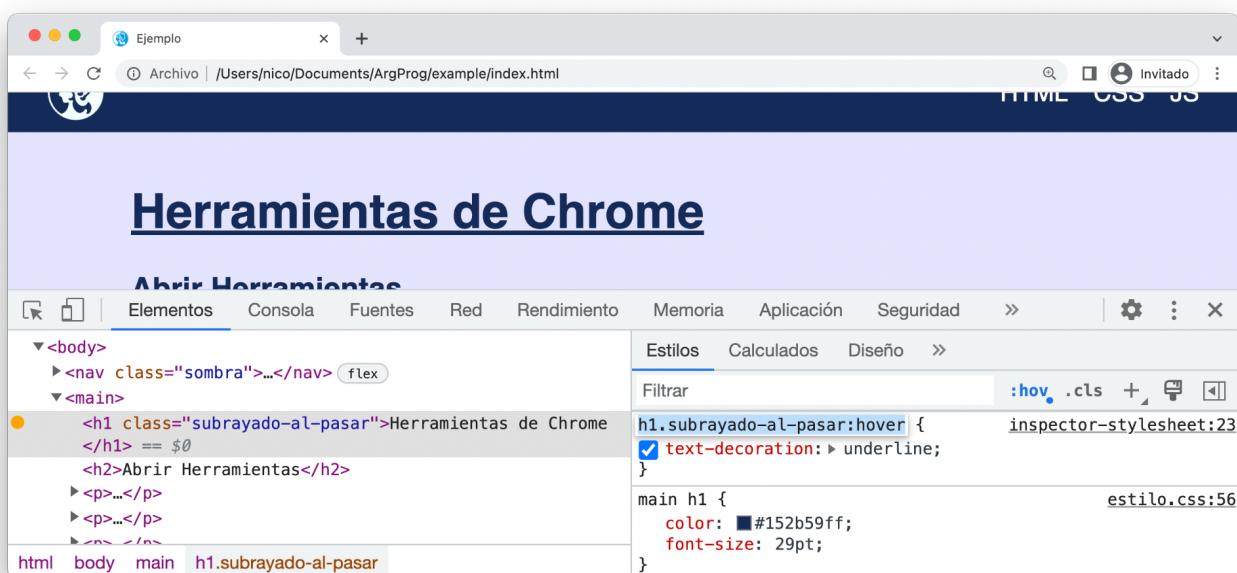


Si queremos agregar una nueva regla, podemos presionar el botón de más (+), que creará una nueva con un selector editable correspondiente al elemento.



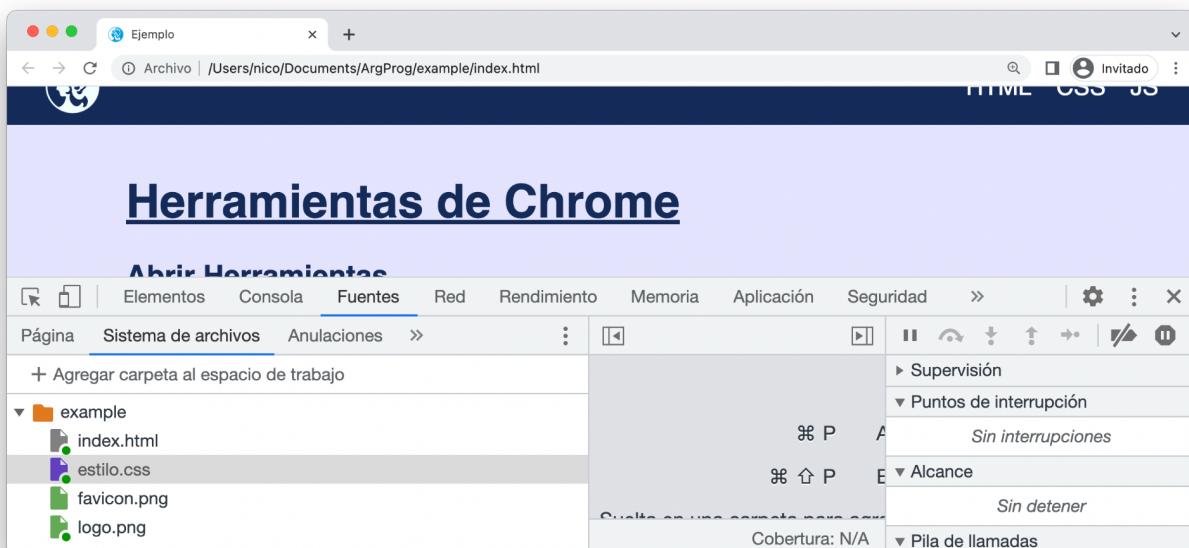
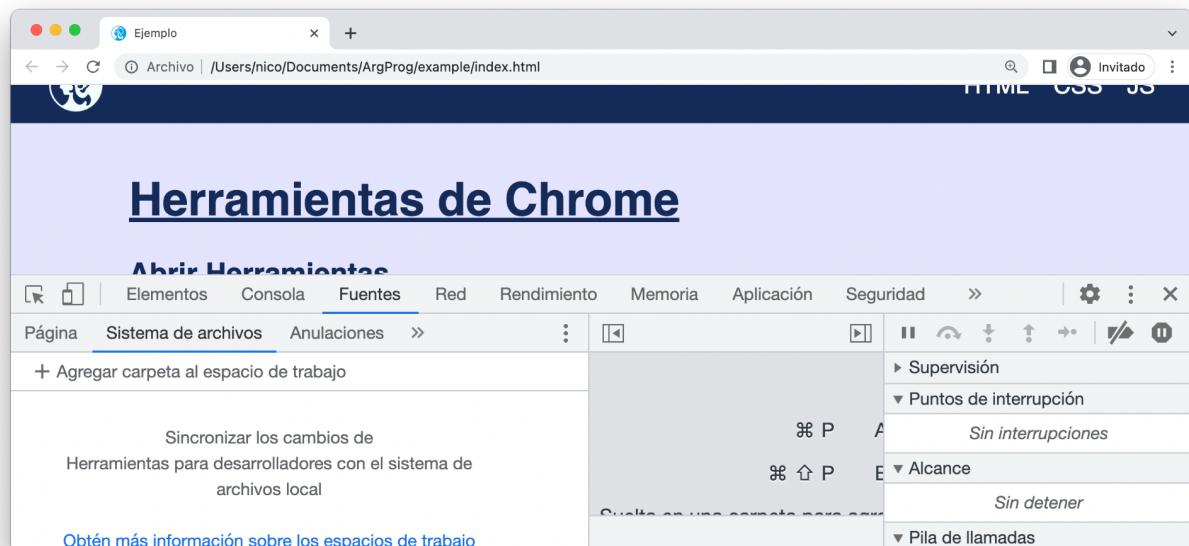
Ejemplo:

1. Se le agrega la clase subrayado-al-pasar al h1 utilizando el botón **.cls**.
2. Se fuerza el estado de hover en el menú de **.hov**.
3. Se define una regla de subrayado al hacer hover.



Guardar Cambios

Para que los cambios se guarden automáticamente en nuestro código iremos a '**fuentes**' > '**sistema de archivos**', tras seleccionar **Agregar carpeta al espacio de trabajo** buscaremos la carpeta del proyecto y permitimos el acceso.



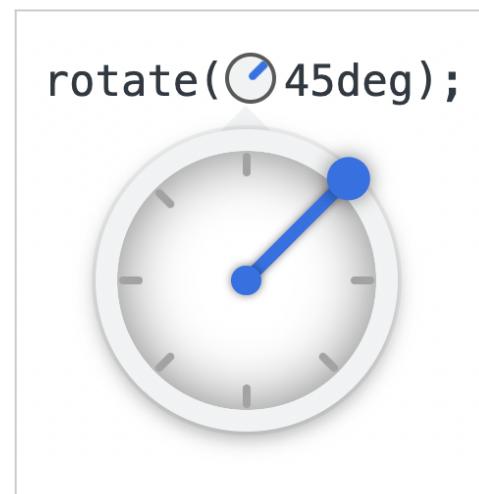
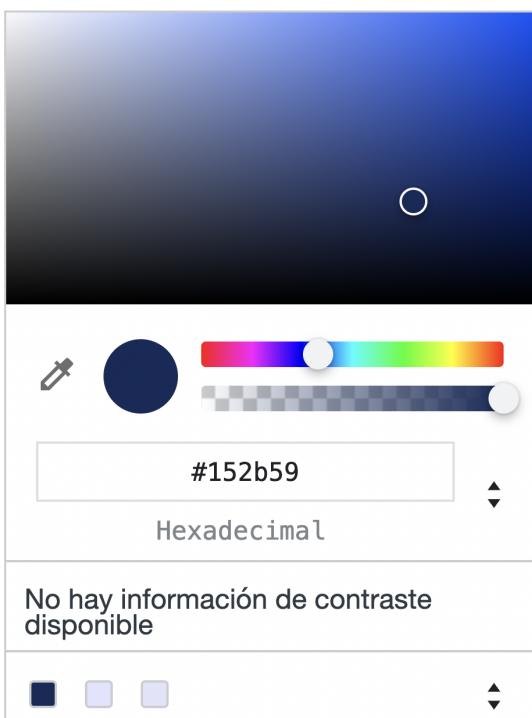
```
main h1 {          ↗ estilo.css:56
    font-size: 29pt;
    color: #152b59ff;
}
```

i Podremos ver el archivo de los estilos y un punto verde indicando la correcta vinculación a las herramientas del navegador.

Editores Interactivos

Al editar estilos, el navegador nos provee algunos editores interactivos, podemos identificarlos mediante los iconos junto a las propiedades.

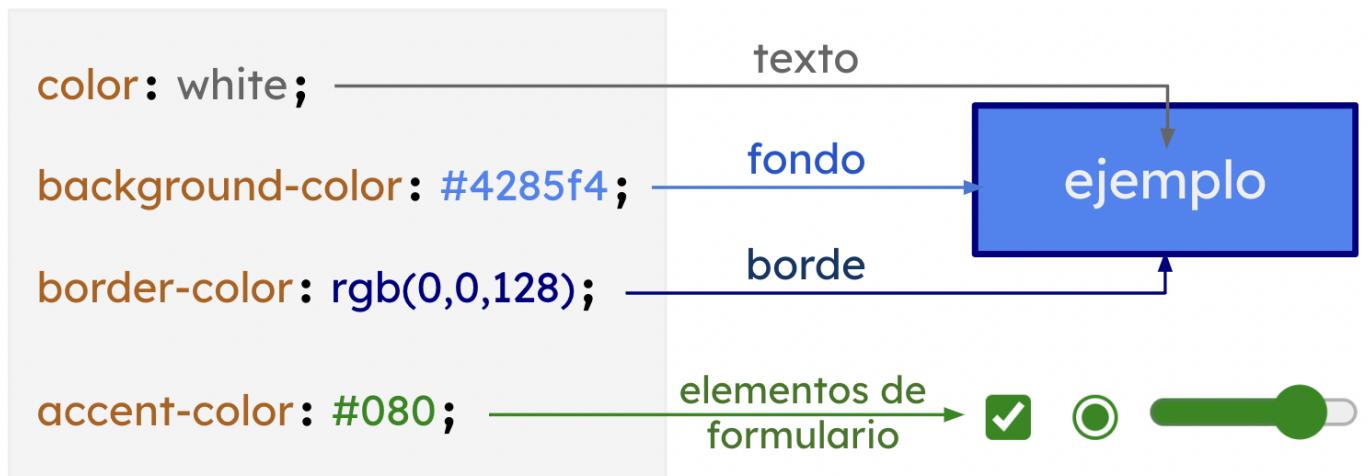
```
color: #152b59ff;  
display: flex;   
box-shadow: 0px 0px 3px 0px #0008;  
transform: rotate(45deg);
```



5. Colores

Atributos de Color

Varios atributos describen colores, podemos ver algunos a continuación:



Opacidad

La opacidad es tanto característica que describe la carencia de transparencia, es decir un elemento será más transparente mientras menos opacidad tenga.

El **atributo opacity** modifica la opacidad de un elemento así como la de sus descendientes; el valor que toma será un número entre 0 y 1 o un porcentaje.

 Si un elemento y su padre tienen opacidad 50%, entonces la opacidad efectiva del objeto será 25%.
Podemos evitar esto utilizando la transparencia del color en sí.



`opacity: 1;`



`opacity: 0.75;`



`opacity: 50%;`



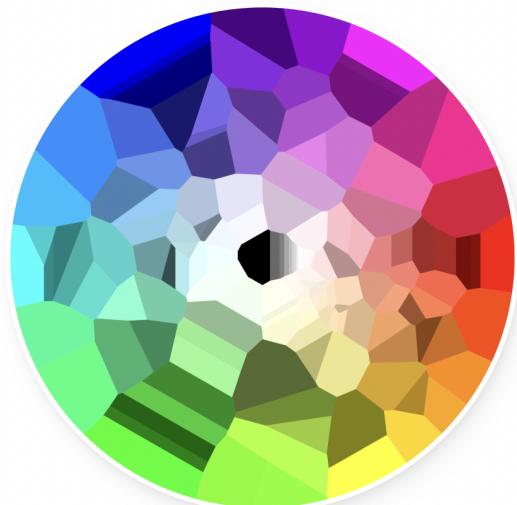
`opacity: 0.25;`

Formatos de Color

Colores Nombrados

Hasta el momento hay **140** colores nombrados que podemos utilizar simplemente mediante su nombre. Por ejemplo *red*, *green* y *blue*.

Aunque son limitados, son simples de usar y pueden ser útiles para prototipar elementos inicialmente o al realizar pruebas de concepto.



`black` `dimgray` `gray` `lightgray` `white` [Lieuallen](#)

Hexadecimal

Hexadecimal es un sistema numérico con **16 dígitos**. Usamos los números del 0 al 9 (del sistema decimal) más las primeras 6 letras del abecedario (A a F).

0 1 2 3 4 5 6 7 8 9 A B C D E F ^{10 11 12 13 14 15}

Utilizamos estos números para asignar un valor a los **canales** de color rojo, verde y azul; ademas, opcionalmente, al canal **alfa** de opacidad.

Podemos utilizar **1 o 2 dígitos** por canal, como en los siguientes ejemplos:



RGB & RGBA

Similar a hex, asignamos valores a los canales de color rojo (Red), verde (Green) y azul (Blue); ademas, opcionalmente, al canal alfa (Alpha).

Lo utilizamos llamando a la **función rgb**, o **rgba** si queremos transparencia, donde los parámetros serán valores de 0 a 255 para cada canal en orden r,g,b.

rgb(21, 43, 89)

rgb(68, 119, 238)

rgba(68, 119, 238, 0.33)

rgba(21, 43, 89, 0.87)

rgba(68, 119, 238, 0.67)

rgba(0, 0, 0, 0.07)

HSL & HSLA

Este modo de color se basa en el **tono o matiz** (Hue), **pureza o saturación** (Saturation) y la **luminosidad** (Lightness) del color.

Lo utilizamos llamando a la **función hsl**, eg: hsl(221deg 62% 22%); o **hsla** si necesitamos transparencia, eg hsla(221deg 100% 22% / 50%)



i A pesar de que los modos anteriores son los más comunes, este tiene la virtud de utilizar atributos que nos son intuitivos, volviéndola de las más útiles al diseñar.

i Los valores de “ángulo”, como el tono, pueden estar por debajo de los 0deg o arriba de los 360deg; darán la vuelta, de modo que: $\text{angulo} = \text{angulo} \pm 360 * n$.

Otros

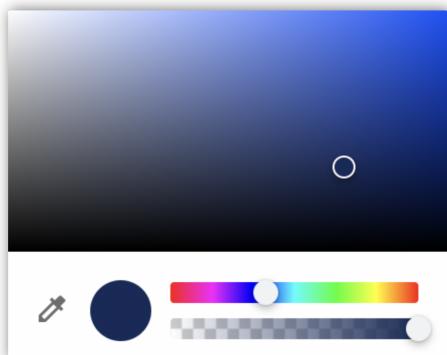
Otro modo de color interesante es el **valor currentcolor** que toma el mismo color que el texto, es decir, el atributo color.

También, podemos hacer gradientes utilizando la función **linear-gradient** o **radial-gradient** como valor del atributo **background** (background-image).

```
border-color: currentcolor;  
background: linear-gradient(90deg, #000, #236, #47E);
```

css

Además, existen los formatos **cmyk** (como el de las impresoras) y **hwb** (que utiliza el tono, blancura y negrura).



Los editores comúnmente usan **hwb**; que es similar a hsl pero usa **brillo** en vez de luminosidad, que es incluso más intuitivo. Esto se suma a las virtudes de utilizar las [herramientas del navegador](#).



⚠ hwb no es un modo de color nativo de css.

6. Texto

Atributos de Texto

Color

El atributo color define el color del texto.

```
color: red;
```

¡Hola Mundo!

```
color: green;
```

¡Hola Mundo!

```
color: blue;
```

¡Hola Mundo!

Tamaño

El atributo **font-size** define el tamaño de la letra.

```
font-size: 20pt;
```

¡Hola Mundo!

```
font-size: 30pt;
```

¡Hola Mundo!

```
font-size: 40pt;
```

¡Hola Mundo!



Podemos utilizar cualquier unidad de medida; la unidad pt (punto tipográfico) es la que suelen utilizar los editores de texto.



El tamaño por defecto de la letra es 16px.

Familia

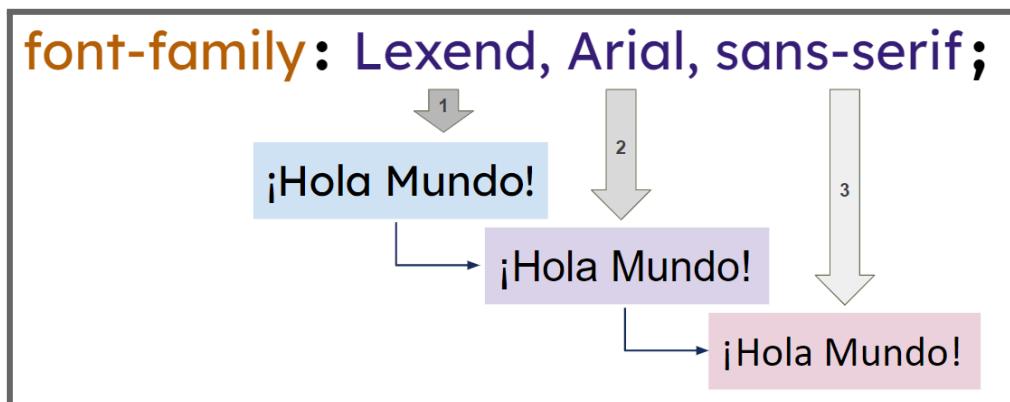
El atributo **font-family** define la familia de letra. Eg: Calibri o Times New Roman.

font-family: Arial; → ¡Hola Mundo!

font-family: Georgia; → ¡Hola Mundo!

font-family: Roboto; → ¡Hola Mundo!

Además, podemos agregar más fuentes como respaldo, por si una no está disponible, separándolas con coma.

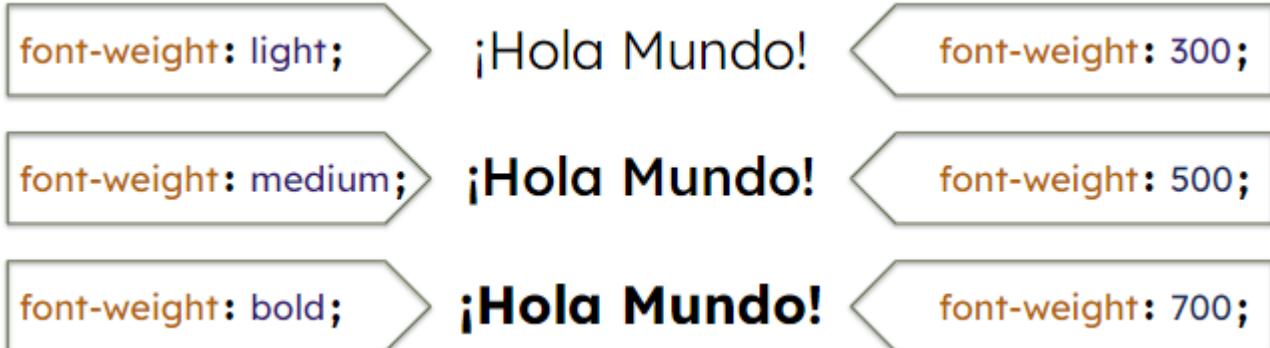


También hay [fuentes seguras](#) que siempre estarán disponibles y [fuentes genéricas](#) que utilizaran la primera de ese tipo disponible en el dispositivo.

Familia Generica	Ejemplos
Serif	Times New Roman Georgia
Sans-serif	Arial Verdana
Monospace	Courier New Lucida Console
Cursive	<i>Brush Script MT</i> <i>Lucida Handwriting</i>
Fantasy	COPPERPLATE Papyrus

Peso

El atributo **font-weight** define el peso o grosor de la letra. Eg: **bold** (negrita). Además de las palabras clave de grosor, podemos utilizar números.



Podemos ver aquí la relación entre los valores numéricos y los nombres:

Valor	Nombre común
100	Thin (Hairline)
200	Extra Light (Ultra Light)
300	Light
400	Normal (Regular)
500	Medium
600	Semi Bold (Demi Bold)
700	Bold
800	Extra Bold (Ultra Bold)
900	Black (Heavy)
950	Extra Black (Ultra Black)

⚠️ No todas las fuentes soportan todos los pesos. Además, los pesos pueden no ser consistentes entre diferentes familias.

Alineación

El atributo **text-align** define la alineación del texto. Al igual que en los editores de texto; el texto por defecto se acomoda a la izquierda, pero se puede ajustar a la derecha, centrar o justificar.

text-align: center;

La alineación del texto se indica mediante el uso de la propiedad **text-align**.



text-align: left;

La alineación del texto se indica mediante el uso de la propiedad **text-align**.



text-align: right;

La alineación del texto se indica mediante el uso de la propiedad **text-align**.



text-align: justify;

La alineación del texto se indica mediante el uso de la propiedad **text-align**.



Otros

→ Estilo: se utiliza **font-style** para hacer *ítálicas*.

font-style: italic;

¡Hola Mundo!

→ Decoración: se utiliza **text-decoration** para dar subrayados.

text-decoration: underline;

¡Hola Mundo!

→ Transformación: se utiliza para **text-transformation** para transformar las letras a mayúsculas, minúsculas, capitalizado, etc.

text-transform: uppercase;

¡HOLA MUNDO!

→ Sombreado: se utiliza **text-shadow** para agregar sombra a las letras.

→ Espaciado: se utiliza **text-spacing** para definir el espacio entre caracteres.

text-shadow

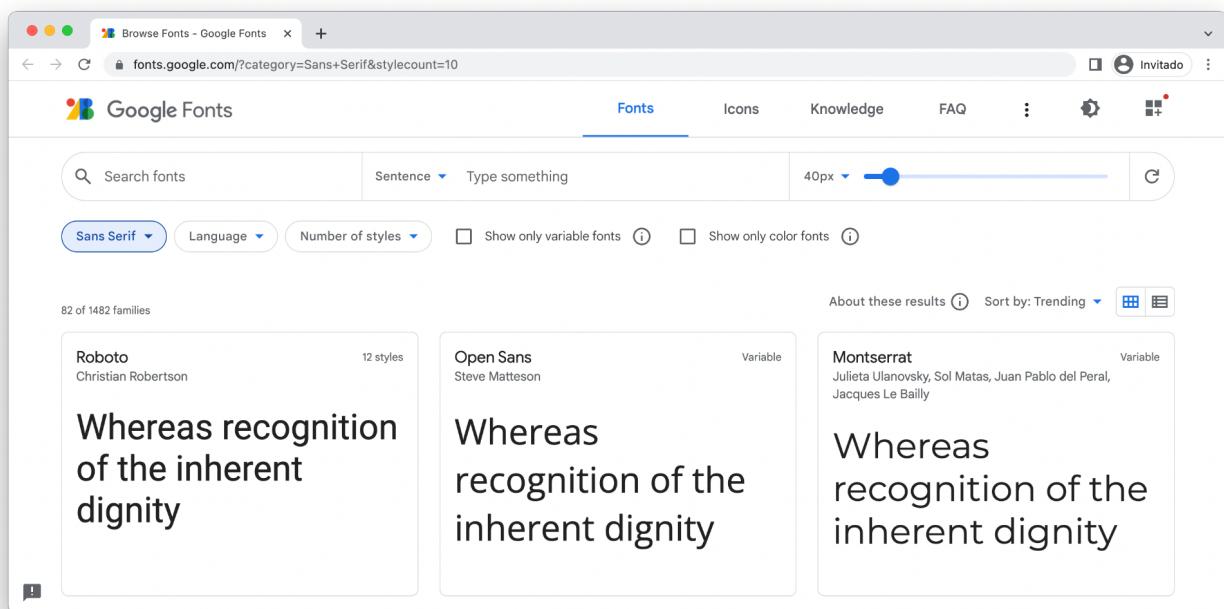
text-spacing



Existe el atributo abreviado font con el cual podemos definir múltiples valores en un solo campo. Eg: "font: bold 20px Arial, sans-serif;"

Fuentes

Podemos conseguir fuentes gratuitas en [Google Fonts](#).



En la pestaña de 'specimen' podemos agregar varios grosores de letra.

The screenshot shows the Google Fonts Specimen page for the 'Open Sans' font family. At the top, there are tabs for 'Fonts', 'Icons', 'Knowledge', 'FAQ', and a settings gear icon. The 'Specimen' tab is currently active. Below the tabs, there are three main sections for different font styles:

- SemiBold 600 Italic:** Displays the text "Whereas recognition of the inherent dignity".
- Bold 700:** Displays the text "Whereas recognition of the inherent dignity".
- ExtraBold 800:** Displays the text "Whereas recognition of the inherent dignity".

On the right side of the page, there is a sidebar titled "Selected family" which lists the available styles for "Open Sans": "Regular 400", "Bold 700", and "Add more styles". Below the sidebar, there is a section titled "Use on the web" with instructions for embedding the font in HTML. The code provided is:

```
<style>
@import url('https://fonts.googleapis.com/css2?family=Open+Sans:wght@400;700&display=swap');
</style>
```

Use on the web

To embed a font, copy the code into the <head> of your html

<link> @import

```
<style>
@import url('https://fonts.googleapis.com/css2?family=Open+Sans:ital,wght@0,300;0,400;0,700;1,500&display=swap');
</style>
```

CSS rules to specify families

```
font-family: 'Open Sans', sans-serif;
```

[Read our FAQs](#)

En vez de descargar las tipografías, podemos directamente importarlas desde `googleapis` en nuestro código.

Al desplegar el menú de fuentes seleccionadas  veremos una sección de uso en web.

Aquí podemos copiar el código de la sección `@import` directo en nuestro css para importar la fuente en los grosores seleccionados.

Luego podemos utilizarla con el nombre que aparece en la sección de abajo (reglas de css para especificar familias).

Ejemplo:

```
@import url('https://fonts.googleapis.com/css2?...');

* {
  box-sizing: border-box;
  font-family: 'Open Sans', sans-serif;
}
```

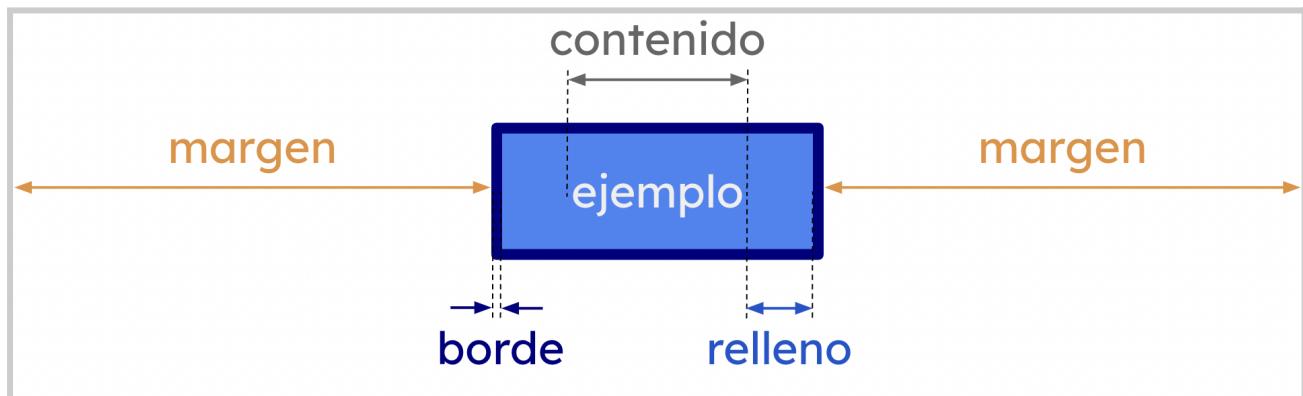
CSS

 Sólo es necesario importar los grosores que utilizaremos; si alguno no está disponible o no existe, se utilizará el grosor más cercano al especificado posible.

 Fuentes con mayor cantidad de grosores *generalmente* están mejor diseñadas.

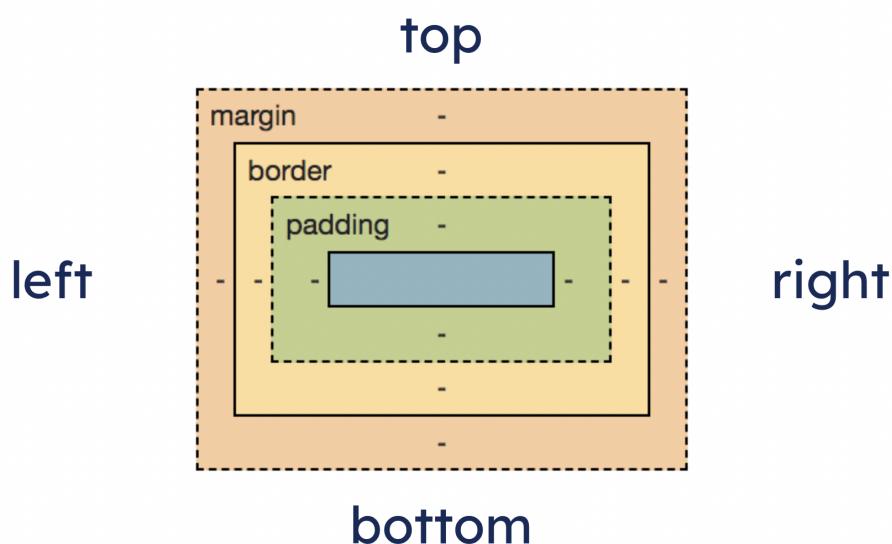
7. Modelo de Cajas

Modelamos los elementos de html como **cajas**, compuestas de 4 elementos.



- **Contenido (content)**: será texto o elementos descendientes.
- **Relleno (padding)**: agrega espacio entre contenido y borde.
- **Borde (border)**: recuadra al elemento.
- **Margen (margin)**: es el espacio entre elementos.

Estos componentes (salvo el contenido) se pueden editar de forma independiente en las 4 direcciones: arriba, derecha, abajo e izquierda.



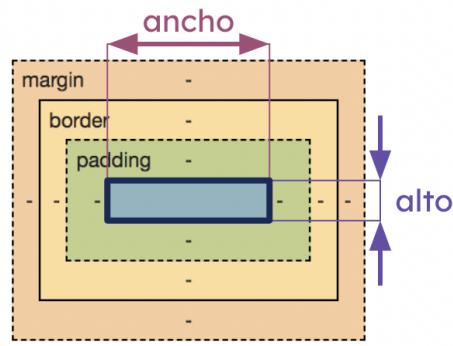
Utilizamos unidades de medida para las dimensiones de cada componente.

Componentes de Caja

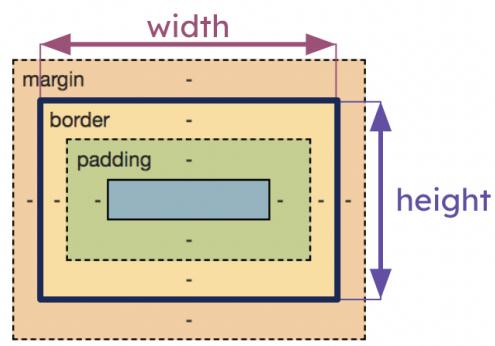
Tamaño de Caja

Utilizando las propiedades **width** (ancho) y **height** (alto) podemos dimensionar los elementos.

Por defecto el **box-sizing** (tamaño de caja) será de **content-box**, de modo que las dimensiones aplican al contenido de la caja; pero podemos darle el valor **border-box** para que las dimensiones incluyan el relleno y el borde.



box-sizing: content-box;



box-sizing: border-box;

Relleno

El atributo abreviado **padding** determina el relleno de un elemento:

padding: 1px 2px 3px 4px;

CSS

Podríamos indicar el relleno en cada dirección con las propiedades no abreviadas:

**padding-top: 1px;
padding-right: 2px;
padding-bottom: 3px;
padding-left: 4px;**

CSS

También podríamos utilizar menos de 4 valores si estos se repitieran:

padding: 2px; /* Todas direcciones */

CSS

padding: 3px 10%; /* Vertical y horizontal */

CSS

padding: 3px 10% 6px; /* Arriba, horizontal y abajo */

CSS

Borde

El atributo abreviado **border** determina el relleno de un elemento; similar al relleno, esta propiedad resume las cuatro direcciones, pero cada dirección también resume varios componentes del borde (grosor, estilo, color).

```
border: 1.5pt solid #555;
```

```
border-left: 1.5pt solid #555;  
border-right: 1.5pt solid #555;
```

```
border-width: 1.5pt;  
border-style: solid;  
border-color: #555;
```

```
border: 1.5pt solid #555;  
border-top-style: dotted;  
border-bottom-style: dashed;
```

Podemos utilizar la propiedad **border-radius** para redondear los bordes.

```
border-radius: 5px;
```

```
border-radius: 20px;
```

 Esta propiedad también puede tomar varios valores para editar las puntas de manera separada, empezando por arriba a la izquierda.

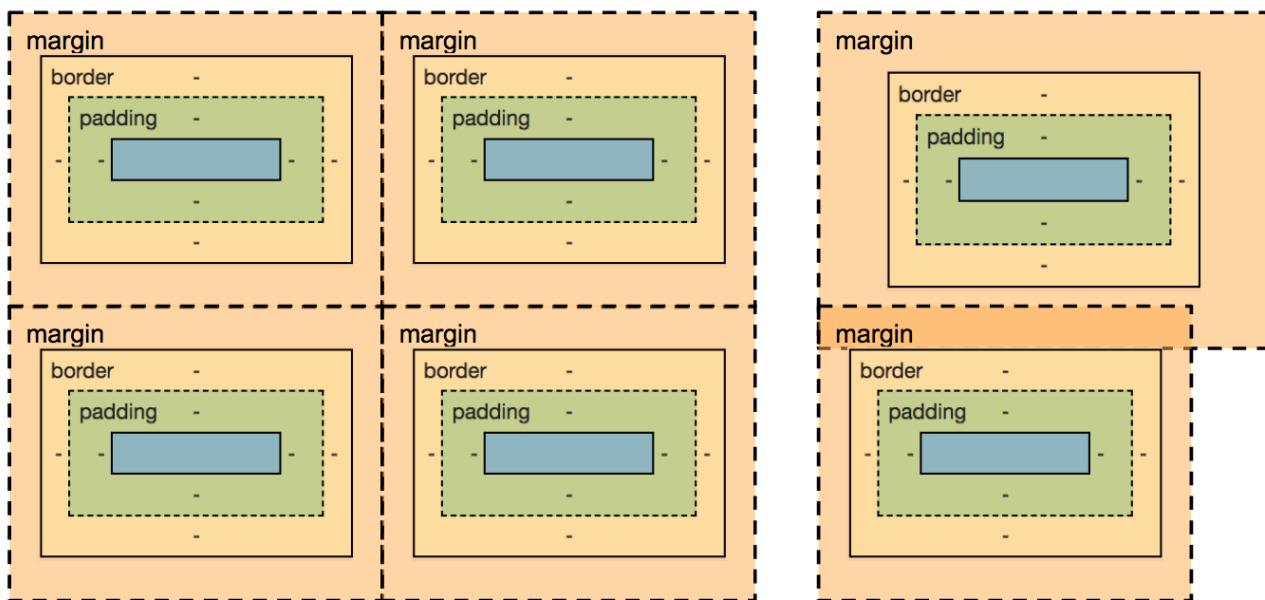
Margen

El atributo abreviado **margin** determina el margen de un elemento, el funcionamiento de esta abreviación es igual al del relleno

```
margin: 10px 25%;
```

```
margin-top: 10px; margin-bottom: 10px;  
margin-right: 25%; margin-left: 25%;
```

Normalmente los márgenes se “empujan”, pero en algunas situaciones los márgenes verticales se pueden “solapar”; eg: elementos con display en block.



Podemos profundizar sobre el colapso de los márgenes en el artículo [Entendiendo el Colapso de Márgenes de Lupita Code](#) en dev.to



Los márgenes pueden ser negativos, generando que los elementos se superpongan entre sí. Eg: `margin-top: -10px`

8. Posición y Presentación

Posición

El atributo **position** define el modo de posicionamiento de un elemento. Utilizamos los atributos **top**, **right**, **bottom** y **left** para determinar la distancia desde los respectivos bordes (esta puede ser negativa, eg: `left: -3px`).



Para ver estos modos puestos en práctica, podemos probar en este [pen](#) o visitar el [artículo de position en CSS-Tricks](#).

● Estática

La posición por defecto de un elemento es **static** (estática), en este modo de posicionamiento no podremos desplazar el elemento.

● Relativa

El modo **relative** posiciona un elemento relativo a su posición original. El resto de los objetos interactúan con este como si estuviera en su posición original.

● Fija

El modo **fixed** deja un elemento fijo en una posición de la página.

● Absoluta

El modo **absolute** posiciona un elemento relativo a otro, este otro será el primer antecesor con posición relativa o absoluta.

● Pegajosa

El modo **sticky** se comporta inicialmente de manera normal, pero se mantiene fija al 'chocar' con la pantalla según los atributos de desplazamiento.

Presentación

El atributo **display** define el modo en el que se presenta el elemento.

 Para ver estos modos puestos en práctica, podemos probar en este [pen](#) o visitar el [artículo de display en CSS-Tricks](#).

● En Línea

El modo **inline** se utiliza por defecto en las etiquetas de texto (span, em, b), donde los elementos van uno al lado del otro.

 En general no deberíamos asignarlo, ya que este modo ignora el ancho y márgenes horizontales que definamos; sus casos de uso están por defecto.

● Bloque

El modo **block** hará que el elemento ocupe su propia línea.

● Bloque en Línea

El modo **inline-block** hará que los elementos puedan compartir una línea, respetando las dimensiones y márgenes que definamos.

● Ninguna

El modo **none** esconderá un elemento, no será visible ni empujará al resto.

● Flexible (Flexbox)

Utilizamos el modo **flex** para organizar el contenido de un elemento de modo ‘flexible’; nos habilita atributos para centrar, espaciar y manejar el contenido.

Podemos profundizar con [la guía de flexbox de CSS-Tricks](#)

● Grilla

Utilizamos el modo grid para organizar el contenido en forma de grilla.

Podemos profundizar con [la guía de grid de CSS-Tricks](#)

9. Variables y Funciones

Variables

Las variables en CSS son valores que podemos reutilizar.

Las definimos como propiedades con 2 signos menos delante: **--variable**.

Accedemos a ellas mediante la función **var**.

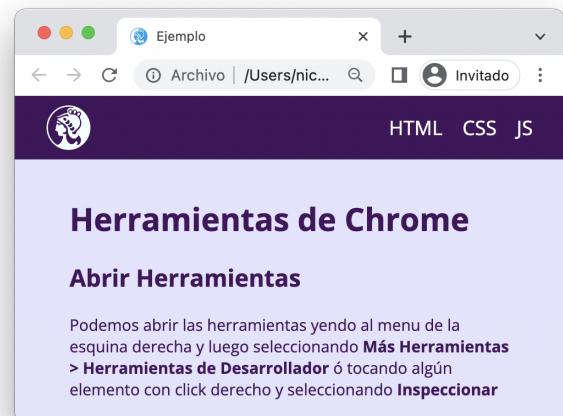
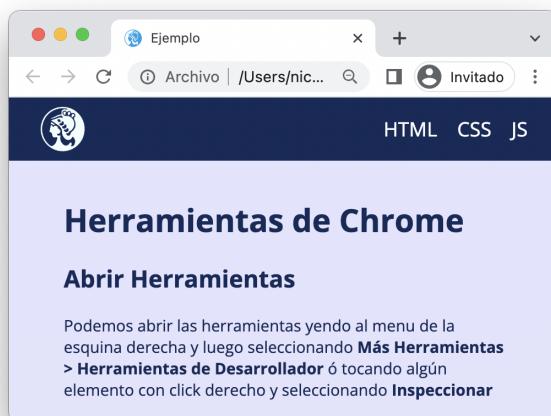
De este modo si hay un color, tamaño u otro que repetiremos varias veces en el código, podemos asignarlo y editarlos en un único lugar.

```
* { --color1: hsl(221, 62%, 22%); }  
nav { background: var(--color1); }  
main { color: var(--color1); }
```

CSS

```
* { --color1: hsl(279, 63%, 22%); }  
nav { background: var(--color1); }  
main { color: var(--color1); }
```

CSS



Es común definir variables globalmente (con el selector universal) para utilizar en cualquier elemento.

También, podemos redefinir una variable en un contexto más específico, de modo que reutilizamos cierta lógica.

Veamos el siguiente extracto de código como ejemplo:

```
<div class='caja chica'>Caja Chica</p>
<div class='caja '>Caja Normal</p>
<div class='caja grande'>Caja Grande</p>
```

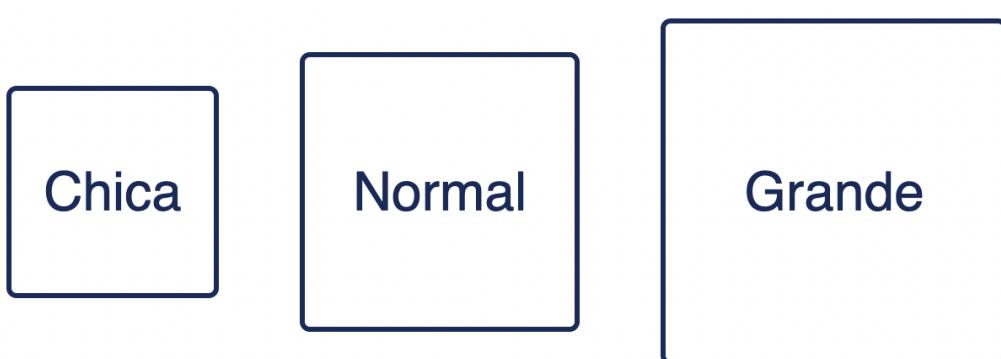
HTML

```
div.caja {
    --dimension: 20px;
    width: var(--dimension);
    height: var(--dimension);
    ...
}

div.caja.chica {
    --dimension: 10px;
}

div.caja.grande {
    --dimension: 30px;
}
```

CSS



Podemos editar el [código de este ejemplo en codepen](#), así como [otro ejemplo](#) más complejo que adapta las posiciones según los tamaños.

Funciones

Utilizamos las funciones para transformar valores según cierta lógica. Su sintaxis consiste del nombre de la función y una lista de parámetros entre paréntesis:
función(parametro1, parametro2, ...)

Ya hemos visto algunas funciones: `rgb()`, `hsl()` y `var()` por ejemplo. Veamos ahora algunas otras funciones útiles.

Podemos utilizar la función **calc** para realizar cálculos de suma, resta, multiplicación y división; además, esta nos permite mezclar unidades de cualquier tipo, variables y números.

```
width: calc(100% - 3rem);  
height: calc(var(--grande) * 3 / 4);
```

CSS

Podemos utilizar la función **min** y **max** para obtener el mínimo o máximo entre 2 valores (o más, si utilizamos otra función de estas como parámetro). Mientras que la función **clamp** recibe un valor mínimo, el valor preferido y el valor máximo.

```
width: min(80%, 1500px);  
height: max(30%, max(10vw, 100px));  
font-size: clamp( 1rem, 2.5vw, 2rem);
```

CSS

Otras funciones remarcables son las que se usan para los atributos de `transform` o `filter`. También, al usar grillas podemos utilizar la función `repeat` para que sus columnas se adapten al ancho disponible.

```
transform: rotate( 45deg );  
filter: drop-shadow(0px 0px 6px #000) hue-rotate(180deg);  
grid-template-columns: repeat( auto-fit, minmax(110px, 1fr) );
```

CSS

10. Referencias

Unidades de medida

Absolutas:

- **px**: pixel
- **pt**: punto tipografico
- **mm**: milimetro

Relativas:

- **em**: relativo al tamaño de letra (font-size) del elemento actual
- **rem**: relativo al tamaño de letra (font-size) del elemento raiz (:root)
- **%**: relativo a elemento padre
- **vw**: relativo al ancho de la pestaña (viewport width)
- **vh**: relativo al alto de la pestaña (viewport height)

Referencias externas

- [W3Schools - CSS](#)
- [CSS Tricks](#)
- [Joshw Comeau - CSS](#)
- [MDN - CSS](#)