

Paper encuentro 5

CSS	1
Incorporar estilos a nuestro documento HTML	1
Anatomía	1
BEM (Block Element Modifier)	2
El Bloque	2
El Elemento	2
El Modificador	3
Posición	3
Display	3
Pseudo-clases y pseudo-elementos	4
Modelo de caja	4
Medidas	4
Herencia y especificidad	5
Selectores de combinación	6
Continuar con ejercicio	6
Git	6
Crear repositorio en github.com	7
Iniciar repositorio local	9
Subiendo mis cambios al repositorio remoto	9
Obteniendo los últimos cambios	9
Utilidades	10

CSS

Hojas de Estilo en Cascada (del inglés Cascading Style Sheets) o CSS es el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML. CSS describe cómo debe ser renderizado el elemento estructurado en la pantalla, en papel, en el habla o en otros medios.

Recordemos que es uno de los estándares HTML que el navegador comprende.

Incorporar estilos a nuestro documento HTML

Existen tres formas de incorporar estilos css a nuestro documento HTML

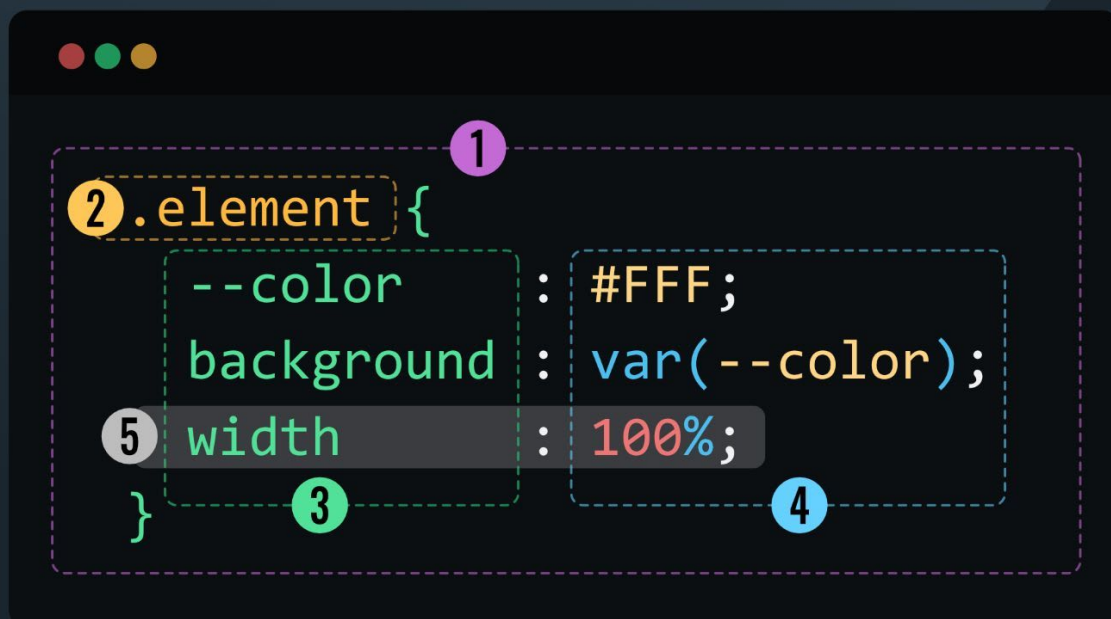
1. Enlazando un archivo externo mediante la etiqueta <link>.
2. Colocar los estilos en el mismo documento dentro de <style></style>.
3. Colocar el estilo a cada elemento en línea.

La forma 1 es la correcta y recomendada ya que las otras dos son poco prácticas en proyectos medianos y grandes.

Anatomía



¿CONOCES LA SINTAXIS DE CSS?



1 Regla CSS

2 Selector

3 Propiedades

4 Valores

5 Declaración

BEM (Block Element Modifier)

Es una metodología de trabajo que consiste en nombrar y clasificar selectores CSS de manera estricta, transparente e informativa. Este método se basa en nombrar las clases en un modo muy específico, ayudándonos a distinguir de manera simple de qué objeto hablamos y si tiene o no aplicado algún tipo de modificador en su estilo, ya sea por interacción del usuario, o por tipología del objeto

BEM distingue claramente 3 conceptos: **el Bloque, el Elemento y el Modificador**

El Bloque

Representa la entidad independiente, es decir, el objeto al que aplicar el estilo. Para ejemplificar pensaremos en la cabecera de una web: pondremos la clase de nuestro bloque como `.main-header`

El Elemento

Figura como una pieza concreta de un Bloque cualquiera, que cumple una función. Evidentemente, un bloque puede estar compuesto de varios elementos. Las clases con las

que identificamos cada elemento las escribiremos después del nombre del bloque, y las separaremos con dos guiones bajos: **bloque__element**

Para nuestro ejemplo, tendremos los siguientes elementos:

```
.main-header__brand  
.main-header__primary-nav  
.main-header__recursive-nav  
.main-header__lang-chooser
```

El Modificador

Son las entidades que usaremos para definir la apariencia o comportamiento de un Bloque o Elemento concreto. Su uso es opcional, pero nos será muy útil para separar claramente el objeto de su estilo gráfico.

Los Modificadores los representaremos con doble guión, por ejemplo:

```
bloque—modificador_bloque  
bloque__elemento—modificador_elemento
```

Si usamos el selector de idiomas del ejemplo, un modificador claro sería si aparece desplegado o no:

```
main-header__lang-chooser  
main-header__lang-chooser—isOpen
```

Posición

- **Static:** Posición por defecto de los elementos, conservan la posición y espacio de donde son colocados (estáticos). No se puede usar top, right, bottom y left en esta posición.
- **Absolute:** Permanecen en la posición de donde fueron colocados pero pierden su espacio físico (se sobreponen a los elementos que ocupan dicho espacio), se los puede posicionar mediante las propiedades top, right, bottom y left.
Importante: Al aplicar las propiedades top, right, bottom y left se tomará de referencia al contenedor más cercano con posición relativa.
- **Relative:** Conservan su posición original y espacio físico pero se los puede posicionar mediante las propiedades top, right, bottom y left sin perder dicho espacio físico.
- **Fixed:** Pierden su espacio físico y permanecen de forma fija (siguen el scroll, se colocan al lado izquierdo del viewport), se los puede posicionar mediante las propiedades de top, right, bottom y left.
- **Sticky:** Conservan su espacio físico pero cuando el scroll los alcanza lo siguen (sin perder dicho espacio físico), es muy usado para barras de navegación y se lo puede posicionar con las propiedades top, right, bottom y left.

Display

Display:inline

No puedo usar margin y padding arriba ni abajo, solo derecha e izquierda. Tampoco se puede aplicar width o height.

Display:block

El contenido del elemento toma el 100% del width, se puede usar margin y padding en cada lado.

Display:inline-block

Se puede usar margin y padding por todos lados, así como darle width y height, y el contenido es del mismo tamaño que el elemento.

Elementos como **p** y **div** vienen por default con **display:block**

Elementos como **span** viene por default con un **display:inline**

Pseudo-clases y pseudo-elementos

Una **pseudo-clase** CSS es una palabra clave que se añade a los selectores y que especifica un estado especial del elemento seleccionado. Por ejemplo, **:hover** aplicará un estilo cuando el usuario haga hover sobre el elemento especificado por el selector.

Sintaxis

```
selector:pseudo-clase { propiedad: valor; }
```

Los pseudo-elementos se añaden a los selectores, pero en cambio, no describen un estado especial sino que, permiten añadir estilos a una parte concreta del documento. Por ejemplo, el pseudo-elemento **::after** permite añadir contenido al final de elemento actual.

Sintaxis

```
selector::pseudo-elemento { propiedad: valor; }
```

¿La diferencia?

Básicamente los pseudo-elementos por lo general hacen referencia a partes específicas de un elemento, mientras que las pseudo-clases hacen referencia al estado del elemento.

Modelo de caja

<https://developer.mozilla.org/es/docs/Web/CSS/box-sizing>

El modelo de caja hace referencia al espacio y lugar que un elemento HTML ocupa en el documento.

Medidas

- **Medidas absolutas:** su valor se encuentra definido en términos concretos y de manera medible. Esto quiere decir que no depende de otro valor de referencia, ni del contexto, las medidas absolutas son:
 - mm = milímetros
 - cm = centímetros
 - in = pulgada

- pc = picas
- px = pixel
- **Medidas relativas:** las medidas relativas no son valores exactos sino que se calculan a partir de otro valor de referencia, las medidas relativas son :
 - %
 - em (hace referencia a element)
 - rem (hace referencia al root element)

EM: unidad relativa al tamaño de texto definido en un determinado contexto

```
body{
    font-size: 20px;
}
h1{
    font-size: 2em; /* 2 x 20 = 40px */
}
```

REM: funciona igual que el em, con la diferencia que es relativo al valor de la fuente del elemento html, y no tiene en cuenta el valor heredado o del elemento que lo contiene.

Por defecto el html viene con un tamaño de fuente de 16px así que siempre

1 REM = 16px

Si queremos aplicar rem de una forma mas sencilla para no tener que hacer tantos cálculos hacemos lo siguiente:

```
html {
    font-size: 62.5%;
}
```

esto lo que hará es darle a el html un valor de 10px ya que $16px - 62.5\% = 10px$

ahora si por ejemplo a una etiqueta le asignamos 2rem este hará referencia a 20px, o si por ejemplo le damos un valor de 1.5rem su valor será de 15px.

Herencia y especificidad

1. Mostrar estilos en hoja y estilo en línea para señalar el contenido de la siguiente tabla con las importancias.
2. Comentar sobre **!important** y evitar su uso.
3. Mostrar desde el inspector como trabaja este contenido de importancias

Selectores	Especificidad
!important	1,0,0,0,0
Inline styles	0,1,0,0,0
#id	0,0,1,0,0
.class	0,0,0,1,0

tag	0,0,0,0,1
-----	-----------

Selectores de combinación

https://developer.mozilla.org/es/docs/Learn/CSS/Building_blocks/Selectores_CSS/Combinadores

Se llaman así porque combinan otros selectores de manera que proporciona una relación útil entre ellos y la ubicación del contenido en el documento.

Hermano adyacente o cercano Adjacent sibling	hermano general General sibling
div + p { ... }	div ~p { ... }
Hijo Child	Descendiente Descendant
div > p { }	div p { }

Continuar con ejercicio

Armar estructura básica y contenidos

<https://xd.adobe.com/view/51795fcc-7085-48a6-bb10-26070be36c9d-042a/>

1. Preguntar si hay algún voluntario que quiera compartírnos pantalla
2. Revisar e ir ajustando al momento entre todos
3. Primero repasar elementos del header (title, charset y viewport, luego mostrar meta description).
4. Pasar a controlar y ajustar elementos del body, repaso por las etiquetas que vamos encontrando y sus propiedades principales.

Git

<https://medium.com/@sthefany/primeros-pasos-con-github-7d5e0769158c>

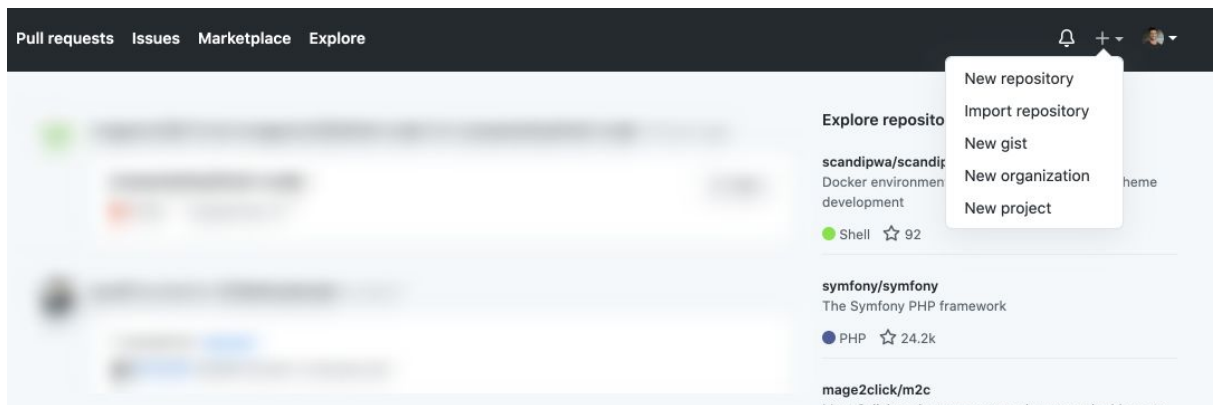
Git es un software de control de versiones. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

Github es una plataforma de desarrollo colaborativo donde se alojan los repositorios.

1. Crear repositorio público en github.com
2. Iniciar repo y configurar el origin al repo recién creado en github.com
3. revisar el estado de los archivos, stage y commit
4. Pull para traer cambios, push para enviarlos a nuestro origin

Crear repositorio en github.com

1. Ingresar a github.com
2. Iniciar sesión con su cuenta (en caso de no tener cuenta deben crearse una)
3. Una vez dentro de su cuenta de github desde el botón + (más) ubicado en la parte superior derecha encontrará la opción de crear un repositorio




4. Luego en la vista de crear nuevo repositorio es necesario ingresar un nombre (sin espacios ni caracteres especiales) y dependiendo del alcance puede ser público o

privado. El resto de opciones dejar en blanco.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner *

 joseamietta ▾

Repository name *

/ prueba ✓

Great repository names are short and memorable. Need inspiration? How about [verbose-octo-meme](#)?

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more](#).

☐ **Add .gitignore**

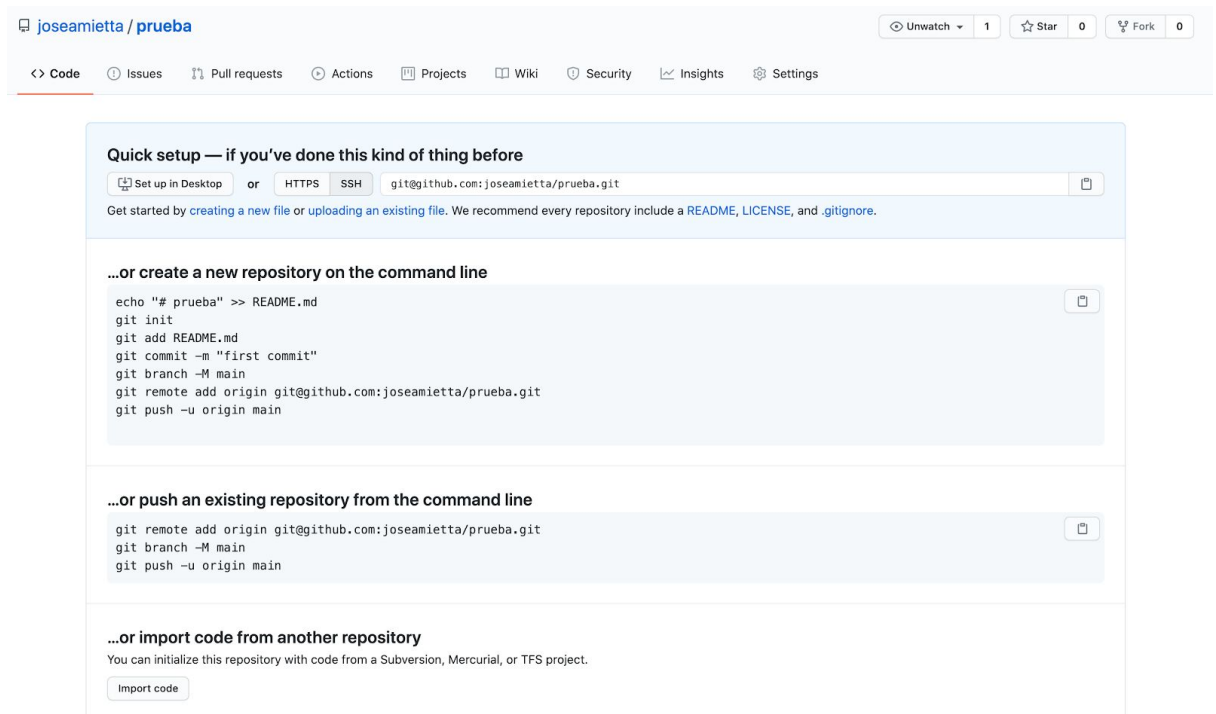
Choose which files not to track from a list of templates. [Learn more](#).

☐ **Choose a license**

A license tells others what they can and can't do with your code. [Learn more](#).

Create repository

5. Luego de creado los dirige a la pantalla del repositorio con algunas indicaciones para subir contenido



Iniciar repositorio local

Tenemos dos manera para iniciar un repositorio en local:

- git init
 - inicia un repositorio vacío, es la opción correcta para enlazar un repositorio de cero.
- git clone <url-repositorio> <directorio-de-instalación>
 - Recupera el contenido de un repositorio ya existente en la carpeta local indicada.

Subiendo mis cambios al repositorio remoto

Antes de comenzar es importante saber que existen distintos estados en git, archivos modificados, archivos preparados (stage) y archivos listos para subir (commit).

- git add <archivo>: prepara los archivos
- git commit: agregar una entrada en el historial git
- git push: sube los cambios realizado en el commit

Obteniendo los últimos cambios

- git pull: obtiene los cambios y actualiza nuestro local
- git fetch: obtiene los cambios pero nos los aplica

git log --oneline --graph --decorate --name-only

Utilidades

- <https://gitexplorer.com/>
- <https://autoprefixer.github.io/>
- <https://developer.mozilla.org/es/docs/Web/CSS/Pseudo-classes>
- <https://developer.mozilla.org/es/docs/Web/CSS/Pseudoelementos>
- <http://getbem.com/introduction/>
- https://seesparkbox.com/foundry/bem_by_example
- https://www.w3schools.com/css/css_combinators.asp