🚀 Introducción a HTML

HTML (Hypertext Markup Language) es el lenguaje empleado para crear páginas web. Utilizando HTML, el contenido de una página web se puede estructurar lógica y semánticamente.

Comenzaremos tu aprendizaje con las etiquetas HTML fundamentales que se emplean para generar contenido en la web.

Aprender HTML es como aprender un nuevo idioma. Al igual que un idioma tiene su propia gramática, vocabulario y reglas de sintaxis, HTML tiene etiquetas, atributos y estructura para construir el significado y la forma de una página web. Así como al aprender un idioma empiezas con palabras básicas y frases simples, en HTML comienzas con etiquetas simples como `<p>` para párrafos o `<img>` para imágenes.

Gradualmente, al igual que en un idioma, vas construyendo oraciones más complejas y estructuras más sofisticadas, aprendiendo cómo combinar diferentes etiquetas y atributos para crear páginas web completas y funcionales. En ambos casos, la práctica constante y la experimentación son claves para alcanzar la fluidez, ya sea en comunicarte en un nuevo idioma o en diseñar y estructurar sitios web eficaces.

HTML: Lenguaje de Marcado

HTML no es un lenguaje de programación, es un lenguaje de marcado que define la estructura de tu contenido. Basa su sintaxis en un elemento base al que llamamos marca, tag o simplemente etiqueta. A través de las etiquetas vamos definiendo los elementos del documento, como enlaces, párrafos, imágenes, etc.

Así pues, un documento HTML estará constituido por un conjunto de etiquetas para definir la función que juega cada contenido dentro de la página. Todo eso le servirá al navegador para saber cómo se tendrá que presentar el texto y otros elementos en la página.

Existen etiquetas para crear:

- Cabecera `<header></header>`
- Bloque principal `<main></main>`
- Artículo `<article></article>`
- Párrafos `<p></p>`
- Imágenes `<img />`
- Listas `<ul></ul>`
- Enlaces `<a></a>`
- etc.

Así pues, aprender HTML es básicamente aprenderse una serie de etiquetas, sus funciones, sus usos y saber un poco sobre cómo debe de construirse un documento básico.

Es una tarea muy sencilla de afrontar, al alcance de cualquier persona, puesto que el lenguaje es muy entendible para los seres humanos.

Por ejemplo, toma la siguiente línea de texto:

```
Unset
Creando un futuro increíble en EGG
```

Si quieres especificar que se trata de un párrafo, podrías encerrar el texto con la etiqueta de párrafo `<p>`:

```
Unset
<p>Creando un futuro increíble en EGG</p>
```

Anatomía de un documento HTML

Hasta ahora has visto lo básico de elementos HTML individuales, pero estos no son muy útiles por sí solos. Ahora verás cómo los elementos individuales son combinados para formar una página HTML entera.

Los documentos HTML van a ser archivos de texto con la extensión .html y tienen la siguiente anatomía:

```
Unset
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
  <title>Título</title>
  <style></style>
 </head>
 <body>
```

```
  <div></div>
 </body>
</html>
```

Estructura básica de un documento HTML

```
Unset
<!DOCTYPE html>
```

Esta etiqueta define el tipo de documento y la versión de HTML que se está utilizando, en este caso, HTML5.

```
Unset
<html lang="es"></html>
```

La etiqueta `<html>` es el elemento raíz de un documento HTML, y el atributo lang se utiliza para especificar el idioma del contenido de la página, en este caso, español (es). Esta etiqueta encierra toda la página entera y, a veces, se la llama raíz (root element).

```
Unset
<head></head>
```

El elemento `<head>` contiene metadatos y enlaces a recursos externos que no se muestran en la página web, como el título, la codificación de caracteres, palabras clave (keywords), una descripción de la página que quieres que aparezca en resultados de búsquedas, código CSS para dar estilo al contenido, declaraciones del juego de caracteres, etc.

```
Unset
<body></body>
```

El elemento `<body>` contiene el contenido visible de la página web, como texto, imágenes, enlaces y otros elementos que se mostrarán en el navegador.

Uso de etiquetas específicas

```
Unset
<meta charset="UTF-8" />
```

Esta etiqueta establece el juego de caracteres que tu documento usará en utf-8, que incluye casi todos los caracteres de todos los idiomas humanos. Básicamente, puede manejar cualquier contenido de texto que puedas incluir. No hay razón para no incluirlo, y puede evitar problemas en el futuro.

```
Unset
<meta name="viewport" content="width=device-width,
initial-scale=1.0" />
```

La etiqueta `<meta>` se utiliza para proporcionar información adicional sobre la página. En este caso, se define una metaetiqueta que controla la escala inicial y el ancho del viewport en dispositivos móviles.

```
Unset
<title></title>
```

La etiqueta `<title>` establece el título de tu página, que es el título que aparece en la pestaña o en la barra de título del navegador cuando la página es cargada o en los resultados de búsqueda, y se emplea para describir la página cuando es añadida a los marcadores o como favorita.

```
Unset
<style></style>
```

Dentro de la etiqueta `<style>`, se especifican reglas de estilo CSS que se aplicarán a los elementos en la página web. Esto permite la personalización y el diseño de la apariencia de la página de una manera más interna y específica.

```
Unset
<div></div>
```

Un contenedor genérico para contenido de flujo, sin significado semántico propio. Es útil para agrupar elementos para aplicar estilos CSS o realizar tareas con JavaScript.

Etiquetas Semánticas de HTML

```
Unset
<header></header>
```

Un elemento que representa un grupo de ayudas introductorias o de navegación. Puede contener elementos como logos, menús de navegación, y formas de búsqueda.

```
Unset
<section></section>
```

Define una sección en un documento, utilizada para agrupar temas relacionados, como capítulos, temas específicos o secciones de contenido.

```
Unset
<main></main>
```

Especifica el contenido principal del documento. Es importante para la accesibilidad, ya que ayuda a los lectores de pantalla y otras tecnologías de asistencia a identificar el contenido principal de la página.

```
Unset
<footer></footer>
```

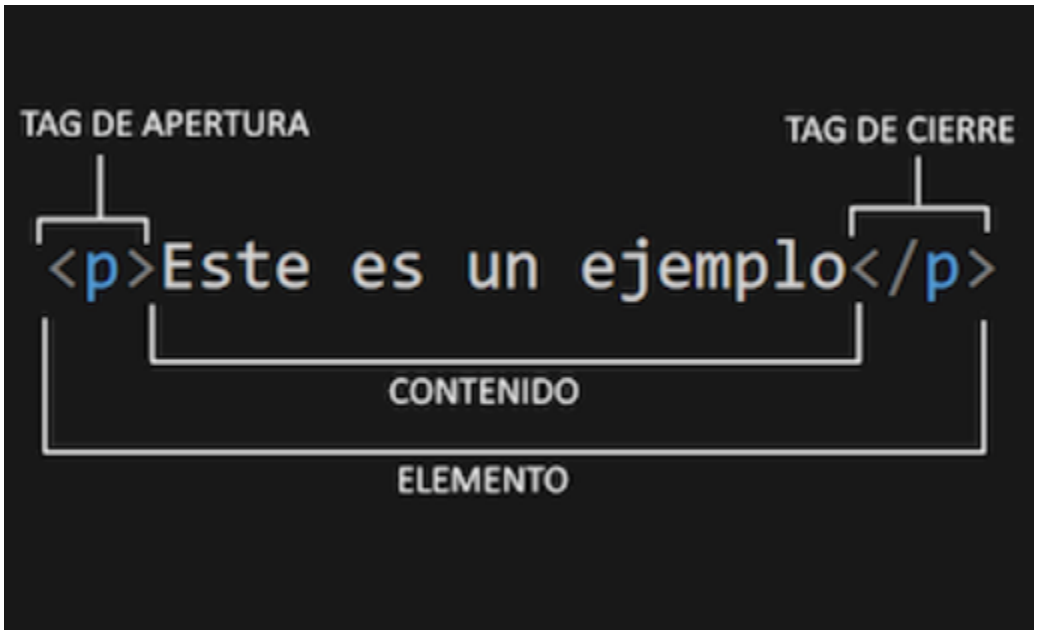
Representa un pie de página para el contenido más cercano o para el elemento raíz (body). Generalmente contiene información como derechos de autor, enlaces a términos de uso, información de contacto, entre otros.

Elementos y etiquetas HTML

```
Unset
<p>Este es un ejemplo</p>
```

Esta línea es un elemento HTML que incluye la etiqueta `</p>`. Veremos a continuación la anatomía de un elemento HTML y las diferentes etiquetas que existen.

Anatomía de un elemento HTML



Las partes principales de un elemento conformado por una son:

1. **La etiqueta o tag de apertura:** consiste en el nombre de la etiqueta (en este caso, p), encerrado por paréntesis angulares (< >) de apertura y cierre. Establece dónde comienza o empieza a tener efecto la etiqueta, en este caso, dónde es el comienzo del párrafo.
2. **La etiqueta o tag de cierre:** es igual que la etiqueta de apertura, excepto que incluye una barra de cierre (/) antes del nombre de la etiqueta. Establece dónde termina la etiqueta, en este caso dónde termina el párrafo.
3. **El contenido:** este es el contenido de la etiqueta, que en este caso es solo texto.
4. **El elemento:** la etiqueta de apertura, más la etiqueta de cierre, más el contenido equivale al elemento.

Atributos de las etiquetas

Las etiquetas son la estructura básica del HTML. Estas etiquetas se componen y contienen otras propiedades, como son los atributos y el contenido.

Una etiqueta por sí sola a veces no contiene la suficiente información para estar completamente definida. Para ello contamos con los atributos.

Estos están compuestos de un par nombre-valor que se encuentran separados por "=" y escritos en la etiqueta inicial de un elemento después del nombre del elemento. El valor puede estar

encerrado entre "comillas dobles" o 'simples'. Existen, también, algunos atributos que afectan al elemento por su presencia en la etiqueta de inicio.

Esta sería la estructura general de una línea de código en lenguaje HTML:

```
Unset
<etiqueta atributo1="valor1"
atributo2="valor2">contenido</etiqueta>
```

Ejemplo 1:

```
Unset
<a href="http://www.enlace.com" target="_blank">Ejemplo de
enlace</a>
```

Donde:

- `<a>` es la etiqueta inicial y `</a>` la etiqueta de cierre.
- `href` y `target` son los atributos.
- `<http://www.enlace.com>` y `_blank` son las variables o valores de los atributos.
- `Ejemplo de enlace` es el contenido.

Ejemplo 2:

```
Unset
<img src="./assets/img1.jpg" alt="Imagen 1" />
```

Donde:

- `<img />` es la etiqueta, como podes observar no lleva etiqueta de cierre, se cierra en la misma etiqueta. Existen ciertas etiquetas que tienen esta estructura, las veremos más adelante.
- `src` y `alt` son los atributos.
- `./assets/img1.jpg` e `Imagen 1` son las variables o valores de los atributos.
- No tiene contenido.

Los atributos definen el comportamiento, los vínculos y la funcionalidad de los elementos. Algunos atributos son globales, lo que significa que pueden aparecer dentro de la etiqueta de apertura de cualquier elemento. Otros atributos se aplican a varios elementos, pero no a todos, mientras que otros atributos son específicos del elemento, relevantes solo para un elemento. En HTML, todos los atributos, excepto los booleanos y, hasta cierto punto, los atributos enumerados, requieren un valor.

Tipos de atributos

Aunque cada una de las etiquetas HTML define sus propios atributos, encontramos algunos comunes a muchas o casi todas las etiquetas, que se dividen en grupos según su funcionalidad:

- Atributos básicos
- Atributos booleanos

Atributos básicos Los atributos básicos se utilizan en la mayoría de las etiquetas HTML y XHTML, aunque adquieren mayor sentido cuando se utilizan hojas de estilo en cascada (CSS):

Atributo	Descripción
id="texto"	Establece un indicador único a cada elemento
class="texto"	Establece la clase CSS que se aplica a los estilos del elemento
style="texto"	Aplica de forma directa los estilos CSS de un elemento
title="texto"	Establece el título del elemento (Mejora la accesibilidad)

💡 Nota: Los atributos de id, class y style los veremos en mayor profundidad en la parte de CSS.

Atributos booleanos Si un atributo booleano está presente, siempre es verdadero. Los atributos booleanos incluyen:

- checked
- disabled
- required
- default
- selected

Estas tres etiquetas son equivalentes:

✅ `<input required>` 🚩 `<input required="">` 🚩 `<input required="required">`

Si el valor del atributo es falso, omita el atributo. Si el atributo es verdadero, incluya el atributo, pero no proporcione un valor. Por ejemplo, required="required"no es un valor válido en HTML; pero como required es booleano, los valores inválidos se resuelven como verdaderos.

Anidamiento de etiquetas

El anidamiento de etiquetas en HTML es un concepto fundamental para la creación de estructuras de página web. Se refiere a la práctica de colocar una etiqueta HTML dentro de otra, lo cual es crucial para organizar el contenido y aplicar estilos y funcionalidades.

Reglas Básicas

- **Orden Correcto:** Las etiquetas deben cerrarse en el orden inverso al que se abren. Por ejemplo, si una etiqueta `<div>` contiene una etiqueta `<p>`, primero debes cerrar `<p>` antes de cerrar `<div>`.

```
Unset

<div>
 <p>Creando un futuro increíble en EGG</p>
</div>
```

- **Anidamiento Lógico:** El anidamiento debe tener sentido semántico y seguir las normas de la estructura del documento. Por ejemplo, no es semánticamente correcto poner un elemento `<body>` dentro de un `<p>`.

Ejemplo

```
Unset
<body>
 <header>...</header>
 <main>
  <section>...</section>
  <section>...</section>
 </main>
 <footer>...</footer>
</body>
```

> 💡 Existen herramientas en línea, como el Validador HTML de W3C, que pueden ayudarte a verificar si tu código HTML está bien estructurado y libre de errores de anidamiento.

Formato de párrafo en HTML

Previamente en nuestra guía habíamos visto la etiqueta `<strong>` que nos permitía darle formato a nuestro texto, más concreto ponerlo en negrita. Ahora veremos con más detalle las más ampliamente utilizadas y ejemplificaremos algunas de ellas posteriormente.

Formatear un texto pasa por tareas tan evidentes como definir los párrafos, justificarlos, introducir viñetas, numeraciones o bien poner en negrita, itálica, etc.

Hemos visto que para definir los párrafos nos servimos de la etiqueta `<p>` que introduce un salto y deja una línea en blanco antes de continuar con el resto del documento.

Podemos también usar la etiqueta `<br>`, de la cual no existe su cierre correspondiente, para realizar un simple salto de línea con lo que no dejamos una línea en blanco, sino que solo cambiamos de línea. Cabe destacar que la etiqueta `<br>`, no es la única etiqueta sin cierre.

Puedes comprobar que cambiar de línea en nuestro documento HTML sin introducir alguna de estas u otras etiquetas no implica en absoluto un cambio de línea en la página visualizada. En realidad, el navegador introducirá el texto y no cambiará de línea a no ser que esta llegue a su fin o bien lo especifiquemos con la etiqueta correspondiente.

¿Qué son las etiquetas de texto?

Las etiquetas de texto nos permiten dar formato a los textos, algunas poseen relevancia semántica y otras solo se usan para cambiar el estilo. Veamos algunas:

Etiqueta	Código	Resultado	Semántica	Uso
****	Bold	**Bold**	No	Le da estilo negrita al texto.
****	Strong	**Strong**	Si	Nos permite marcar con especial énfasis las partes más importantes de un texto
****	Cursiva	*Cursiva*	No	Le da estilo cursiva al texto.
****	Énfasis	*Énfasis*	Si	Enfatiza el texto respecto a la frase que lo contiene.
<small>	small	small	No	Permite reducir el tamaño de esa porción de texto.

Encabezados en HTML

Existen otras etiquetas para definir párrafos especiales, que funcionaran como títulos de nuestra página. Son los encabezados o **headings** en inglés.

Son etiquetas que forman el texto como un título, pero el hecho de que cambien el formato no es lo que nos tiene que preocupar, sino el significado en sí de la etiqueta. Es cierto que los navegadores asignan un mayor tamaño de letra y colocan el texto en negrita, pero lo esencial es que sirven para definir la estructura del contenido de un documento HTML.

Asimismo, los motores de búsqueda podrán analizar mejor el contenido de una página en función de los títulos y subtítulos.

Existen diversos tipos de encabezados, que se diferencian visualmente en el tamaño de la letra utilizada. En el ámbito de la etiqueta se encuentra la H1, para los encabezados más grandes, H2 para los de segundo nivel y, de esta forma, hasta el H6, el encabezado más pequeño.

Los encabezados se verán de esta manera en la página:

< h1> Encabezado 1 < /h1>

< h2> Encabezado 2 < /h2>

...

< h6> Encabezado 6 < /h6>

Los encabezados implican también una separación en párrafos, así que todo lo que escribamos dentro de H1 y su cierre (o cualquier otro encabezado) se colocará en un párrafo independiente.

🚀 Introducción a CSS

CSS es el acrónimo de Cascading Style Sheets, o lo que sería en español Hojas de Estilo en Cascada. Es un lenguaje que sirve para especificar el estilo o aspecto de las páginas web. CSS se define basándose en un estándar publicado por una organización llamada W3C, que también se encarga de estandarizar el propio lenguaje HTML.

> 💡 En W3CSchools encontrarás toda la documentación necesaria respecto a este tema.

Inclusión de estilos CSS

Hay distintas maneras de hacerlo y todas tienen sus momentos y lugares apropiados.

Comenzaremos desde el nivel más específico hasta el más general, es decir, aumentaremos la dificultad y la importancia progresivamente.

1. **Estilo Inline**
   Para definir estilos en partes específicas de una página, utilizamos el atributo **style** en la etiqueta correspondiente. La sintaxis CSS para este atributo determina las características de los estilos. Veamos un ejemplo en el que ciertas palabras de un párrafo se muestran en color amarillo:

```
Unset

<p>
 Yo estudié programación en
 <span style="color: #ffcd00"> Egg </span>
</p>
```

Yo estudié programación en Egg

💡 Recuerda: Aunque este método es útil para casos puntuales, no es recomendable para estilos a gran escala porque dificulta el mantenimiento del código.

2. **Estilo a nivel de elemento**
   Este método permite dar un estilo específico a una etiqueta completa. Por ejemplo, podemos hacer que un párrafo completo sea de color rojo. Para esto, utilizamos el atributo style en la etiqueta en cuestión.

```
Unset
<p style="color: red">Esto es un párrafo de color rojo.</p>
```

Esto es un párrafo de color rojo.

3. **Estilo a nivel de bloque**
   Utilizando la etiqueta `<div>`, podemos definir secciones de una página y aplicarle estilos con el atributo style. Así, podemos definir estilos para todo un bloque de la página de una sola vez.

```
Unset

<div style="color: green">
 <p>Estas etiquetas van en <strong>verde y negrita</strong></p>
 <p>Seguimos dentro del DIV, por lo tanto permanecen los estilos</p>
</div>
```

Estas etiquetas van en **verde y negrita**

Seguimos dentro del DIV, por lo tanto permanecen los estilos

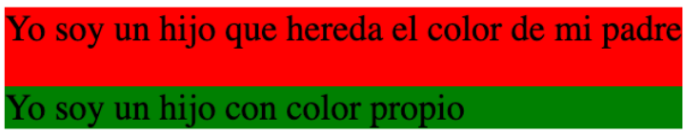
4. **Estilo a nivel de página**
   Una opción más potente y cómoda es definir estilos a nivel de página completa en la cabecera del documento. Esto nos permite evitar "ensuciar" las etiquetas HTML

colocando el atributo style. Además, podemos reutilizar estos estilos en diferentes etiquetas a lo largo del documento, simplificando el código y facilitando los cambios de estilo a gran escala.
A grandes rasgos, entre `<style>` y `</style>`, se coloca el nombre de la etiqueta (o selector) para la que queremos definir los estilos y entre llaves -{ }- colocamos en sintaxis CSS las características de estilos. El concepto de selectores lo veremos más adelante.

Unset

```
<head>
 ...
 <style>
  header {
   background-color: red;
  }
  .bg-color {
   background-color: green;
  }
 </style>
</head>

<body>
 <header>
  <div>
   <p>Yo soy un hijo que hereda el color de mi padre</p>
  </div>
  <div class="bg-color">Yo soy un hijo con color propio</div>
 </header>
</body>
```



💡 Los estilos definidos para un elemento padre son heredados por los elementos hijos, a menos que estos últimos tengan estilos propios.

5. **Estilo a nivel de sitio web**
   El método más poderoso y recomendado para el desarrollo de hojas de estilo es el uso

de una hoja de estilo externa (archivo .css). Este archivo solo contiene las declaraciones de estilo de la página y se enlaza a todas las páginas del sitio. Esto permite a todas las páginas compartir una declaración de estilos, reutilizando el código CSS de una manera más eficiente y manteniendo una coherencia en el diseño.

```
Unset

<head>
 <link rel="stylesheet" type="text/css" href="styles.css" />
</head>
```

Por ahora trabajaremos con la forma del punto 4, estilos a nivel de página.

Sintaxis CSS

La meta básica del lenguaje Cascading Stylesheet (CSS) es permitir al motor del navegador pintar elementos de la página con características específicas, como colores, posición o decoración. La sintaxis CSS refleja estas metas y estos son los bloques básicos de construcción:

- **La propiedad:** Es un identificador, un nombre legible por humanos, que define qué característica es considerada.
- **El valor:** Describe como las características deben ser manejadas. Cada propiedad tiene un conjunto de valores determinados, definido por una gramática formal, así como un significado semántico, implementados por el motor del navegador.

Este par (propiedad y valor) es llamado una declaración.

Ambas propiedades y valores son sensibles a mayúsculas y minúsculas en CSS. El par se separa por dos puntos, “ : ”, y espacios en blanco antes, entre ellos y después.

Bloques de declaraciones

Las declaraciones pueden ser agrupadas en bloques, que es una estructura delimitada por una llave de apertura, '{', y una de cierre, '}'. Los bloques en ocasiones pueden anidarse, por lo que las llaves de apertura y cierre deben de coincidir.

selector { Aquí escribiremos las declaraciones CSS }

- 💡 Las llaves delimitan el inicio y el final del bloque.

Esos bloques son naturalmente llamados bloques de declaraciones y las declaraciones dentro de ellos están separadas por un punto y coma, " ; ".

En cuanto a la última declaración de un bloque, esta no necesita terminar en un punto y coma, aunque es usualmente considerado una buena práctica porque previene el olvidar agregarlo cuando se extienda el bloque con otra declaración.

¿Qué partes conforman a una declaración en CSS?

Los diferentes términos se definen a continuación:

- **Regla:** cada uno de los estilos que componen una hoja de estilos CSS.
- **Selector:** indica el o los elementos HTML a los que se aplicará la regla CSS.
- **Declaración:** especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades CSS.
- **Propiedad:** permite modificar el aspecto de una característica del elemento.
- **Valor:** indica el nuevo valor de la característica modificada en el elemento.

Selectores CSS

Vamos a ver los selectores básicos que necesitarás para avanzar en el próximo paso, después veremos en profundidad el resto.

Los selectores CSS son patrones utilizados para seleccionar los elementos HTML a los que se les aplicará un conjunto de estilos.

1. **Selector de Tipo o Etiqueta:** Selecciona todos los elementos de un tipo específico. Por ejemplo, p selecciona todas las etiquetas `<p>`.

```
Unset

p {
 color: blue;
}
```

   Con esta regla todos los p dentro del html van a ser azules.

2. **Selector de Clase:** Selecciona todos los elementos del body que tienen una clase específica. Se denota con un punto seguido del nombre de la clase. Por ejemplo, .destacado selecciona todos los elementos con la clase destacado.

Ejemplo:

HTML:

```
Unset

<body>
 <p class="destacado">Párrafo 1</p>
```

```
 <p class="error">Párrafo 2</p>
 <p>Párrafo 3</p>
</body>
```

style CSS:

```
Unset

<style>
 .destacado {
  font-size: 18px;
 }
 .error {
  color: red;
 }

</style>
```

Resultado:

Entonces para especificar una clase debemos poner dentro de las etiquetas `<style></style>` del head un punto (‘.’) y el nombre de la clase que queremos que coincida con valor que pongamos en nuestro atributo class en el html.

Entonces, en el ejemplo podemos ver como el primer párrafo tiene el valor `destacado` y el segundo párrafo el valor `error` para el atributo class y en la etiqueta style CSS, hemos definido un estilo para esas clases.

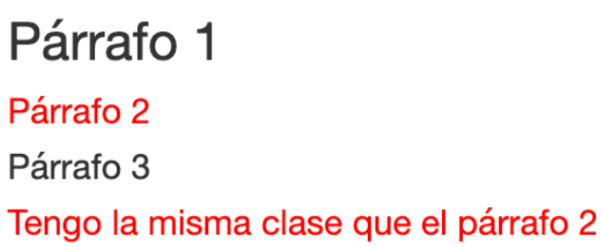
El beneficio del atributo class, además de dejarnos asignar estilos a un solo elemento, es que después podemos reutilizar esa class para asignarle ese estilo a otros párrafos concretos o a otras etiquetas, solo deberemos ponerle el valor de un estilo que ya existe en el atributo class.

Por ejemplo:
HTML:

```
Unset

<body>
 <p class="destacado">Párrafo 1</p>
 <p class="error">Párrafo 2</p>
 <p>Párrafo 3</p>
 <h3 class="error">Tengo la misma clase que el párrafo 2</h3>
</body>
```

Resultado:



3. Selector de ID En un documento HTML, los selectores de ID de CSS buscan un elemento basado en el contenido del atributo id. El atributo ID del elemento seleccionado debe coincidir exactamente con el valor dado en el selector.
   Este tipo de selectores sólo seleccionan un elemento de la página porque el valor del atributo id no se puede repetir en dos elementos diferentes de una misma página. Esa es la diferencia básica entre selector de clase y selector de ID.
   Ejemplo: HTML:

```
Unset

<body>
```

```
<div id="identificador">¡Este div tiene un ID especial!</div>
<div>Este solo es un div regular.</div>
```

4. `</body>`

style CSS:

```
Unset

<style>
 #identificador {
  background-color: blue;
 }
</style>
```

Resultado:

Este solo es un div regular.

En nuestro archivo CSS para especificar un ID, vamos a poner el numeral (‘#’) y el nombre del ID que queremos que coincida con el valor que pongamos en nuestro atributo ID en el html. Como podemos ver el ID, es muy parecido al atributo class, pero la diferencia es que el ID se puede usar para identificar un solo elemento, mientras que una clase se puede usar para agrupar más de uno.

Propiedad de Display

Cada elemento HTML tiene un valor de visualización predeterminado según el tipo de elemento que sea. El valor de visualización predeterminado para la mayoría de los elementos es **block** o **inline**.

- Los elementos de **bloque (block)** ocupan todo el ancho de la pantalla.
- Los elementos en **línea (inline)** sólo ocupan el espacio necesario.

Otras propiedades de display que existen son:

- **none:** Oculta completamente el elemento, es decir, no se muestra en la página web. Es muy útil para hacer que un elemento desaparezca temporalmente o para ocultarlo en diferentes condiciones de visualización (esto se logra manipulando el CSS con javascript).
- **inline-block:** Combina las características de los elementos inline y los elementos block. Los elementos inline-block se muestran como elementos en línea, pero permiten establecer ancho y altura como si fueran elementos de bloque.
- **inherit:** Hace que el elemento herede el valor de la propiedad display de su elemento padre.
- **flex:** Convierte un elemento en un contenedor flexible, lo que permite controlar el tamaño y la posición de sus elementos hijos.
- **grid:** Convierte un elemento en un contenedor de cuadrícula, lo que permite colocar elementos hijos en filas y columnas.

  💡 Tip: Puedes utilizar el centro de práctica de W3School para afianzar los conceptos y ejercitar la propiedad display

Flexbox en CSS

Concepto de Flexbox

Flexbox, o el Flexible Box Layout, es un modelo de diseño en CSS que permite a los desarrolladores crear interfaces de usuario complejas con diseños flexibles y responsivos. Es especialmente útil para distribuir el espacio entre elementos de un contenedor y alinearlos de manera eficaz.

Componentes Principales de Flexbox

1. **Contenedor Flex (Flex Container)**: El elemento padre que envuelve los elementos flexibles. Se define aplicando `display: flex` o `display: inline-flex`.
2. **Elementos Flex (Flex Items)**: Los elementos hijos directos del contenedor flex. Estos son los elementos que se organizan usando las propiedades de Flexbox.

Propiedades del Contenedor Flex

- **`display: flex`**: Define un bloque flex container.
- **`flex-direction`**: Establece la dirección principal de los elementos flex (horizontal o vertical). Valores comunes incluyen `row` (predeterminado, horizontal) y `column` (vertical).
- **`justify-content`**: Alinea los elementos flexibles a lo largo del eje principal (horizontal en `row`, vertical en `column`). Valores comunes son `flex-start`, `flex-end`, `center`, `space-between`, `space-around`.

- **`align-items`**: Alinea los elementos flexibles en el eje transversal (contrario al eje principal). Valores comunes son `stretch` (predeterminado), `flex-start`, `flex-end`, `center`, `baseline`.
- **`flex-wrap`**: Permite que los elementos flex se envuelvan o no en múltiples líneas. Valores comunes son `nowrap` (predeterminado), `wrap`, `wrap-reverse`.

Propiedades de los Elementos Flex

- **`flex-grow`**: Define la capacidad de un elemento flex para crecer si es necesario. Un valor de `0` significa que el elemento no crecerá.
- **`flex-shrink`**: Define la capacidad de un elemento flex para encogerse si es necesario. Un valor de `1` significa que el elemento puede encogerse.
- **`flex-basis`**: Establece el tamaño inicial del elemento antes de la distribución del espacio restante.
- **`flex`**: Es un atajo para `flex-grow`, `flex-shrink`, y `flex-basis`.

Ejemplo Básico

HTML

```
Unset
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
  <title>Document</title>
  <style>
   .container {
    width: 600px;
    margin: 60px auto;
   }
   .item-1 {
    background-color: salmon;
   }
   .item-2 {
    background-color: lightblue;
```

```
   }
   .item-3 {
    background-color: lightgreen;
   }
  </style>
 </head>
 <body>
  <div class="container">
   <div class="item item-1">Item 1</div>
   <div class="item item-2">Item 2</div>
   <div class="item item-3">Item 3</div>
  </div>
 </body>
</html>
```

Resultado inicial

Al aplicar los siguientes estilos de FlexBox...

HTML

Unset

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
  <title>Document</title>
  <style>
   .container {
```

```
    width: 600px;
    margin: 60px auto;

    /* Estilos de FlexBox */
    display: flex;
    justify-content: center;
    align-items: center;
   }
   .item {
    flex-grow: 1;
   }
   .item-1 {
    background-color: salmon;
   }
   .item-2 {
    background-color: lightblue;
   }
   .item-3 {
    background-color: lightgreen;
   }
  </style>
 </head>
 <body>
  <div class="container">
   <div class="item item-1">Item 1</div>
   <div class="item item-2">Item 2</div>
   <div class="item item-3">Item 3</div>
  </div>
 </body>
</html>
```

Obtenemos el siguiente resultado:

Uso de Flexbox

Flexbox es ideal para:

- Centrar elementos en un contenedor (tanto horizontal como verticalmente).
- Crear barras de navegación y pie de página.
- Diseñar layouts responsivos que se adaptan a diferentes tamaños de pantalla.
- Ordenar elementos sin necesidad de alterar el HTML.

Buenas Prácticas

- Utiliza Flexbox cuando necesites un diseño unidimensional (en una fila o columna).
- Para diseños más complejos y bidimensionales, considera usar CSS Grid.
- Combina Flexbox con unidades de medida relativas (como %, vw, vh) para mayor responsividad.

Flexbox representa una herramienta esencial en la caja de herramientas de un desarrollador front-end moderno, permitiendo crear interfaces de usuario eficientes, flexibles y accesibles.

> 💡 Tip: Puedes utilizar el centro de práctica de FlexBox Froggy para afianzar los conceptos y ejercitar la propiedad display flex

> 💡 Repasa los tipos de rutas del blog de Nube Colectiva, donde detallan las diferencias entre rutas absolutas y rutas relativas y como acceder a los archivos dentro de las carpetas.