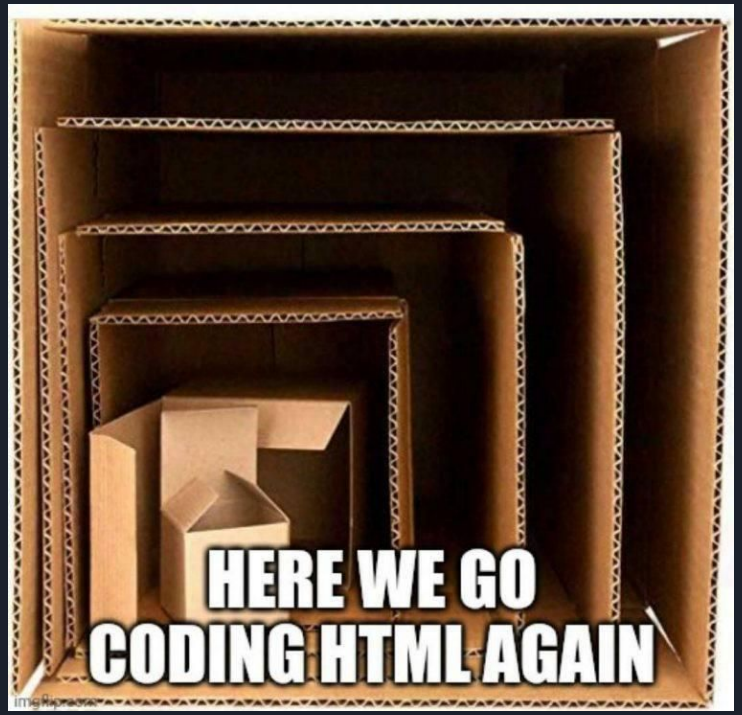
HTML & CSS

TB022 - Esteban - Riesgo - Kristal

HTML



HTML

HyperText Markup Language

HTML es un *lenguaje de maquetado* o markup language que sirve para definir la estructura del contenido que será mostrado en un ambiente dado. Es el standard para muchos ambientes, web incluído.

Fue desarrollado en 1993 por la W3C foundation justamente para establecer un standard para usar en la Internet.

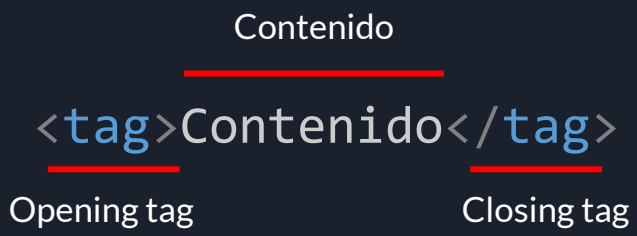
Consiste de una serie de elementos que engloban o *wrappean* diferentes partes del contenido para que se muestre de cierta manera.

No es un lenguaje de programación.

HTML Element

Cada elemento le dará un estilo puntual a una parte puntual del contenido que querramos mostrar.

Cómo se compone un elemento:



Cada elemento va a estar delimitado por un tag que define el nombre del elemento y el estilo que tendrá que *wrappea* el contenido al cual se le dará estilo.

Cada tag es especial y se usa para objetivos distintos.

HTML Element: p

Creemos un archivo ejemplo.html en nuestro editor de texto y pongamos el siguiente contenido:

```
<p>Mi gato es naranja.</p>
```

El tag p es de *paragraph*

Luego, abramos el archivo usando un navegador web.

Podemos anidar tags

```
<p>Mi gato es <b>naranja</b>.</p>
```

El tag b es de **bold**

Si refrescamos el navegador, veremos el nuevo resultado.

Además (y acá viene lo interesante) podemos asignarle atributos a nuestros tags para modificar cómo se muestran. Cada tag va a tener distintos atributos, aunque algunos los comparten todos.

HTML void elements

No todos los elementos tienen un opening y closing tags. Algunos, llamados *void elements*, tienen solamente un tag. Por ejemplo *img*.

```
<p>Seguro que este es Pikachu</p>
<img
     src="https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/37.png"
     alt="Pikachu?"
>
```

`src` y `alt` son atributos del tag `img`. Se imaginan que hace cada uno?

También podemos asignarle `width` y `height`, entre otros. Siempre el valor va a ir dentro de `""`.

HTML document

Los elementos que venimos viendo nos van a servir para definir nuestro document, el cual va a englobar todo el contenido que queremos mostrar. Todos los sitios siguen el siguiente formato:

```
<!doctype html>
<html lang="<lang>">
 <head>
   <meta charset="utf-8" />
   <title>My test page</title>
 </head>
 <body>
   <p>This is my page</p>
 </body>
</html>
```

`<!doctype html>`

Necesario por cuestiones que hoy en día no se usan más, pero hay que ponerlo igual

`<html>`

Root element. Todo estará dentro de éste.

`<head>`

Información adicional a la que se va a mostrar, como el título <title> del sitio entre otros.

`<body>`

Todos los elementos que se van a mostrar por pantalla. Nos vamos a enfocar en los elementos de esta parte.

HTML text fundamentals

Vamos a ver qué tipos de tags hay y para qué se usan

Headings

Los encabezados sirven para dar jerarquía al contenido. Existen desde h1 hasta h6 (más grande e importante a menos). Los *headings* le darán importancia al contenido mostrado en los *paragraphs*. Es importante usarlos correctamente para los navegadores de búsqueda.

Listas `<ul>` o `<ol>`

Sirven para listar cosas. Si queremos que la lista esté ordenada usaremos `<ol>` (*ordered list*). Cada elemento de la lista debe tener su propio tag de *list item* `<li>`. Podemos anidar listas tantas veces como querramos.

HTML text fundamentals

Vamos a ver qué tipos de tags hay y para qué se usan

Tablas `<table>`

Nos permiten organizar la información en forma de tabla o matriz. Generalmente útil cuando el contenido a mostrar es una lista de elementos donde cada elemento tiene muchos campos e información que queremos mostrar por separado, pero todos de la misma manera.

Cada `<table>` tendrá una fila `<tr>` (table row) y cada fila tendrá la misma cantidad de celdas `<td>`

HTML element: a

Para links.

Go search in Google

HTML element: a

Para links.

Que pueden ser externos (a cualquier otro sitio):

```
<a href="https://www.google.com">Go search in Google</a>
```

O que pueden ser internos (a otras partes o "páginas" del mismo sitio). En este caso, debemos aclarar el path al archivo html:

```
<a href="index.html">Homepage</a>
```

HTML: ejercicio

Hacer un sitio que tenga dos páginas:

- index.html que sea el homepage del sitio. El mismo debe mostrar un mensaje de bienvenida y permitir al usuario ir a la página de:
- pokemon.html que muestre en una tabla la información básica de 3 Pokemon (elegir cuales quiera) y muestre la imagen de cada uno.

Las páginas deben tener la estructura básica que toda página debe tener: título, y headings correspondientes.

Web browser inspector

Todos los navegadores modernos poseen una herramienta que permite visualizar el contenido de una página en su formato renderizado junto con el código HTML que la compone.

Esta herramienta es muy útil para entender qué es cada cosa pero además nos permite editar el contenido interactivamente. Esto es doblemente útil porque nos permite probar cambios menores sin necesidad de ir al código, editarlo, guardarlo y recargar la página cada vez. Eso sí, no hay que olvidarse de qué cambios hicimos porque si refrescamos, todo el contenido que hayamos modificado se pierde.

HTML: tags de estilo

Hay muchos tags más en existencia, algunos nos permite darle cierto estilo al contenido más allá de la forma en la que estén organizados (como veníamos viendo). Algunos de ellos son:

```
<b> - Bold text
<strong> - Important text
<i> - Italic text
<em> - Emphasized text
<mark> - Marked text
<small> - Smaller text
<del> - Deleted text
<ins> - Inserted text
<sub> - Subscript text
<sup> - Superscript text
```

La realidad sin embargo, es que éstos prácticamente no se usan.

Para dar estilo a nuestro contenido se utilizan principalmente CSS.

CSS

CSS: Cascading Style Sheets

Es la principal herramienta que vamos a usar para estilar nuestras páginas. Es la herramienta por defecto que se usa en páginas HTML pero también sirve para utilizarse en otros lenguajes de maquetado como XML o SVG.

HTML nos permite definir la estructura y semántica de nuestro contenido, CSS se usa para darle estilo y disponer de los elementos (dondé se colocarán con respecto al documento y entre sí).

CSS Inline

Todos los HTML elements pueden tienen el tag *style* que se puede usar para darle, justamente, estilo al elemento con CSS.

```
<p style="propiedad: valor; propiedad2: valor2;">Contenido</p>
```

Cada *propiedad* que queramos darle de estilo debe ir separada por un `;`. Cada propiedad tiene determinados valores posibles.

Recordar SIEMPRE que toda propiedad que involucre un tamaño, lleva un valor con UNIDAD. Por simplicidad recomendamos usar siempre `px` (pixeles).

Qué estilo podemos dar? Hay miles de opciones. Si se bajan el plugin de VSCode para HTML, les va a mostrar sugerencias y auto completado muy útiles.

El web browser inspector nos permite también cambiar estos estilos desde el navegador. Recordar siempre que estos cambios son temporales y no se guardan.

CSS Inline

Qué pasa si queremos darle el mismo estilo a más de un elemento?

Por ejemplo, si armamos una tabla querríamos que todas las celdas se vean iguales y que los encabezados tengan un estilo especial.

Deberíamos escribir a mano el mismo estilo para cada uno. Si queremos cambiar una cosa en particular, deberíamos cambiar todo. Es un embole.

Podemos aplicar CSS a través de hojas de estilo.

CSS Stylesheets

Son archivos .css que van a contener el diseño de nuestra página.

```css
p {
    color: royalblue;
    background-color: crimson;
    border: 10px solid black;
    border-radius: 25px;
    font-size: 20pt;
    padding-left: 20px;
    inline-size: 250px;
    font-family: monospace;
}
```

Estamos diciendo que todos los elementos de tipo <p> van a tener este estilo.
Podemos borrar el tag `style` del HTML.

Para que el HTML sepa donde buscar estos estilos, debemos agregar una sentencia en el header de nuestro documento:

```html
<link rel="stylesheet" href="path/to/file.css" />
```

Pero qué pasa si no queremos que TODOS los elementos de tipo <p> tengan este estilo?

CSS: class tag

Es un tag especial que tienen todos los elementos. Se usa para identificar grupos de elementos a los que se les quiere dar el mismo estilo.

```
.pikachu {
    ...
}
.not-pikachu {
    ...
}
```

```
<p class="pikachu">Este es Pikachu</p>

<p class="not-pikachu">Este no es Pikachu</p>
```

CSS: posicionamiento

Hasta ahora vimos formas de darle estilo a nuestros elementos. Nos falta ver como disponer de ellos en el espacio del documento.

`display`

Todos los elementos pueden tener el tag display para determinar cómo disponer del elemento. Por defecto la mayoría de los elementos tienen `"display: block"`. Los elementos se colocan uno debajo del otro en vertical. `"display: inline"` en cambio haría que los elementos se dispongan todos en la misma línea horizontalmente.

La propiedad de display debe ser establecida por el elemento padre, y será aplicado a todos sus elementos hijos.

Hay muchas opciones, pero el único que debería usarse hoy en día si no se quiere usar el default es `display: flex`.

Flexbox

Este tipo de display se introdujo recientemente (menos de 10 años). Originalmente todos los distintos tipos de display servían para mostrar el contenido de distinta manera.

Hoy en dia `display: flex` sirve para organizar el contenido de cualquier manera. Hay muchas otras propiedades que hay que setear para definirlo. Por ejemplo:

`flex-direction`: si es vertical u horizontal o cualquiera de ellos invertido.

Pero no es lo único. La mayoría de las propiedades nos permite definir cómo se comportarán los elementos hijos del padre.

`flex-wrap`: si el contenido se va de pantalla, lo pone en la siguiente línea o sigue de largo

`align-items`: cómo centrar los elementos en el sentido opuesto a flex-direction

`justify-content`: cómo centrar los elementos en el sentido de flex-direction

Si quieren saber más de Flexbox pueden ver esta guía o seguir este tutorial. No es necesario que lo usen para el TP.

CSS: jerarquía de elementos

Hasta ahora todos los elementos estaban "sueltos" en el body. Técnicamente todos serían hijos de este elemento.

Si aplicásemos una propiedad de display al elemento body, se aplicaría a todos los elementos.

Cuando queremos aplicar este tipo de propiedades a un grupo de elementos y no a todos, debemos entonces crear estos grupos y asignarle la propiedad que querramos a ellos.

Estos grupos también serán elementos HTML pero no serán de ningún tipo en particular de los que vimos, sino que serán `<div>`.

CSS:

Un elemento de tipo sirve para servir de padre para un grupo de elementos.

Podemos asignar las clases que queremos y cualquier otra propiedad disponible de los demás tipos.

En el ejemplo anterior, agregar un por cada par de elementos p, img. En cada uno agreguemos una clase (o sino inline en el HTML) el estilo `display: flex` y veamos qué pasa.

Este tipo de elemento también nos permite llenar el espacio. Podemos asignarle margin (espacio del elemento con otros elementos), padding (espacio interno del elemento), background color (igual que a cualquier elemento) pero afectará a todo el grupo.

Si un elemento hijo redefine una propiedad del elemento padre, se aplicará ese estilo, sobreescribiendo el valor que había puesto el padre originalmente.

CSS: tips

Todo el contenido que vemos en una página está ordenado de alguna manera.

Generalmente vamos a encontrar secciones que tienen un orden vertical (una debajo de otra) porque generalmente el sentido del contenido es de arriba hacia abajo. Pero, dentro de cada una de estas secciones, suele haber subsecciones que pueden también estar en vertical o en horizal, generando una nueva jerarquía, y así sucesivamente muchos niveles.

Miren estos ejemplos:

https://www.apple.com/

https://www.infobae.com/

https://pokemondb.net/

Se notan las jerarquías y el sentido de orden de cada grupo de elementos?

Bonus Tracks

CSS Grid

Además de Flexbox, existe un mecanismo para deteminar el layout de nuestra página. Es un recurso aún más nuevo y no está totalmente adoptado su uso, pero es recomendable usarlo.

Se usa en conjunto con Flexbox.

La idea es que CSS Grid determina justamente el layout general del documento y flexbox se usa para deteminar la disposición de cada grupo de elementos y sus hijos.

Para más información pueden ver esta guía o seguir este tutorial.

Además, CSS Grid se usa para hacer que el sitio sea *responsive*.

Responsiveness

Cada usuario al ver un sitio web, lo hará desde una pantalla distinta. Hay decenas, cientos de tamaños distintos de pantalla (pcs de escritorio, notebooks, tablets, celulares, etc.). Todos los usuarios deberían poder visualizar correctamente el sitio web no importa desde donde se conecten. Esto no es fácil de implementar, pero si se logra, se dice que el sitio es *responsive*.

CSS Grid permite determinar cómo disponer los elementos en nuestro documento para distintos tipos de pantalla.

Un mismo sitio que vemos en el celular, dispone los elementos de manera distinta que cuando lo vemos en una PC de Escritorio en un monitor 4k de 32".

Pueden leer un poco más de info acá y ver ejemplos con los que jugar acá.

SCSS, Sass, Less

Todas estas son extensiones, llamadas preprocesadores, al CSS básico que estuvimos viendo para hacerlo más potente.

Cada uno agrega sus beneficios puntuales, pero general agregan cosas como variables, funciones, ciclos, loops. Pero esto tiene un coste.

Estos super scripts se tienen que compilar.

Pueden leer más en éste artículo que los compara.

FIN