FINEST en Desarrollo Web

Tema 1 - HTML

Agenda de la clase

Agenda

- ¿Qué es programar?
- Front–End
- HTML
- Etiquetas HTML
- Visual Studio Code
- Imágenes
- Links
- Comentarios

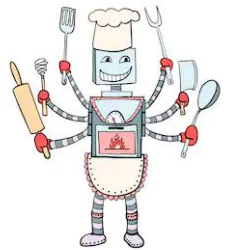
¿Qué es programar?

¿Qué es programar? (1/3)

Programar es *“crear un conjunto de instrucciones que serán ejecutadas por una computadora”.*

Por el momento, las computadoras son máquinas bastante “tontas”. Hay que decirles, con lujo de detalles (y sin ambigüedades), lo que queremos que hagan.

Analogía: Escribir un programa es como escribir una receta de cocina para una persona que nunca cocinó algo en su vida. Imagínense que en lugar de una persona es un robot. No alcanza con especificar los ingredientes y las cantidades. Es necesario explicar con precisión cada uno de los pasos que hay que realizar. Ejemplo: *“Girar la cuchara 90 veces en sentido horario a una velocidad de 300 rpm. Si quedan grumos, girar en sentido anti-horario a 120 rpm hasta que desaparezcan, etc”*. A la persona (o al robot) no le deben quedar dudas sobre cómo proceder.



¿Qué es programar? (2/3)

Así como los humanos disponen de distintos lenguajes (idiomas) para comunicarse entre sí, a la hora de darle instrucciones a una máquina (PC) también es necesario elegir un lenguaje de programación. Algunos ejemplos son: JavaScript, Java, PHP, Python, C++ y Ruby.

Tanto los lenguajes “humanos” (lenguajes naturales) como los lenguajes de programación disponen de sintaxis y semántica.

La ventaja de los lenguajes de programación es que permiten dar instrucciones sin ambigüedades. En cambio, los lenguajes naturales son ambiguos.

¿Qué es programar? (3/3)

Tipos de lenguajes de programación:

- **Código de máquina / Código binario**
  - Es el menor nivel de abstracción. Consiste en unos y ceros.

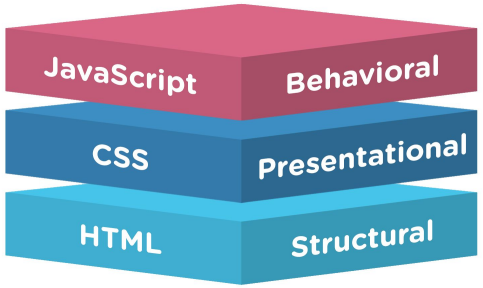
```
0110001100
1011010110
1111011110
```

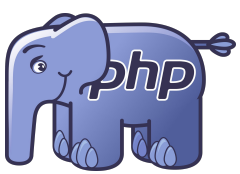
- **Lenguaje Assembly**
  - Representación simbólica de código de máquina.
- **Lenguajes compilados**
  - Lenguajes de alto nivel que necesitan ser compilados a código de máquina.
  - Ej: C, C++, Go, Fortran, Pascal, Java (que se compila a un lenguaje intermedio llamado *bytecode*).
- **Lenguajes interpretados**
  - Lenguajes de alto nivel que no necesitan ser compilados. Los programas residen en la memoria RAM.
  - Ej: JavaScript, Python, PHP, Ruby.

Más nivel de abstracción

Front-End


Frontend
Backend


JavaScript
Behavioral
CSS
Presentational
HTML
Structural


php


MariaDB


MySQL


mongoDB.


C#


Java


python


node
JS

Estructura de una página web

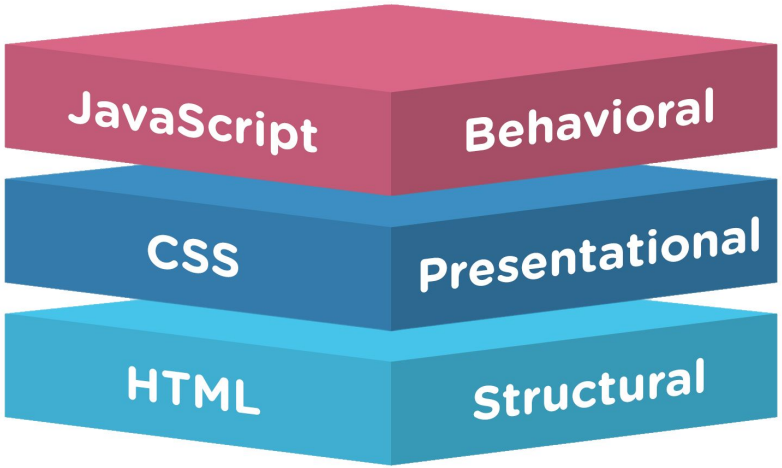
Contenido: Texto, imágenes, videos.

\+ **HTML**: Estructura + Semántica

\+ **CSS**: Presentación + Diseño

\+ **JS**: Interacción

= Página web



Nota: Esta es la estructura básica de una página desde el punto de vista de un desarrollador Front-End. Un desarrollador Back-End también consideraría, por ejemplo, el guardado (persistencia) de los datos.

Estructura de una página web

Analogía con la construcción de una casa:

HTML

Qué

CSS

Cómo

JavaScript

Interacciones

HTML → Qué

CSS → Cómo

HTML

HTML: “HyperText Markup Language”

Qué es:

- Es el lenguaje nativo de los browsers (creado en 1993).
- Se utiliza para describir la estructura y contenido de las páginas web.
- Está compuesto por etiquetas (*tags*).
- Se escribe en un archivo de texto con extensión `.htm` o `.html` (sobre todo la segunda opción).

Qué no es:

- No es un lenguaje de programación (es un lenguaje de marcado o etiquetas).

HTML – Documentación

- Documentación en Mozilla Developer Network (muy completa):
  https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5 (Inglés)
  https://developer.mozilla.org/es/docs/HTML/HTML5 (Español)

- Documentación en W3 Schools (más básica; más simple):
  http://www.w3schools.com/html/default.asp

Empiecen a acostumbrarse a consultar la **documentación** de cada tecnología.

En general, suelen ser mejores las versiones en **inglés**.

Etiquetas HTML

¿Qué es una etiqueta (*tag*) HTML?

Son palabras clave escritas entre < y >.

Se abren y cierran.

```
<etiqueta></etiqueta>
```

Opcionalmente, pueden tener contenido y atributos.

```
<etiqueta atributo="valor">contenido</etiqueta>
```

Ejemplos de etiquetas HTML (1/2)

La etiqueta <p> se utiliza para representar párrafos en una página.

```
<p>Este es un breve párrafo de mi página</p>
```

Si el párrafo tiene mucho texto, se suele escribir así:

```
<p>
    Este es un párrafo que tiene mucho texto y se lo suele
    escribir de esta manera. Es equivalente a lo anterior.
</p>
```

Ejemplos de etiquetas HTML (2/2)

La etiqueta `<a>` (*anchor*) se utiliza para representar *links* en una página.

```
<a href="https://ha.dev">Este es un enlace</a>
```

En esta etiqueta, es fundamental definir el atributo `href`. De lo contrario sería un *link* que no apunta a ningún lado.

Notar que en los ejemplos anteriores, en ningún momento se dijo cómo se desean mostrar los elementos en la página.
Recordar que con HTML sólo se establecerá qué elementos componen una página, pero no cómo se deberán mostrar.

Estructura básica de una página HTML

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Título de la página</title>
    </head>
    <body>
        <h1>Mi primer título</h1>
        <p>Mi primer párrafo.</p>
    </body>
</html>
```

Componentes:

- **DOCTYPE**
  Indica la versión del HTML que se utilizará. En este caso se indica que se está usando HTML versión 5.
- **HTML**
- **HEAD**
  Información y atributos de la página.
- **BODY**
  Contenido “visible” de la página.

HTML – Semántico, Accesible y Organizado

HTML – Semántico, Accesible y Organizado

El HTML que escribe un desarrollador debería ser:

- Semántico: Debe tener significado. HTML se debe ocupar del *qué*, no del *cómo*.
- Accesible: Para que el sitio web pueda ser accedido por la mayor cantidad de personas. Particularmente hay que pensar en las personas no-videntes.
- Organizado: El código debe ser prolijo, organizado, fácil de leer y entender.

Ejemplo de semántica: con HTML se puede crear un párrafo `<p>` y luego con CSS cambiar el tipo y tamaño de letra para que *parezca* un título. Pero por más de que se parezca a un título, semánticamente hablando, no deja de ser un párrafo y, por lo tanto, no es correcto hacer esto. Este concepto es muy importante en HTML.

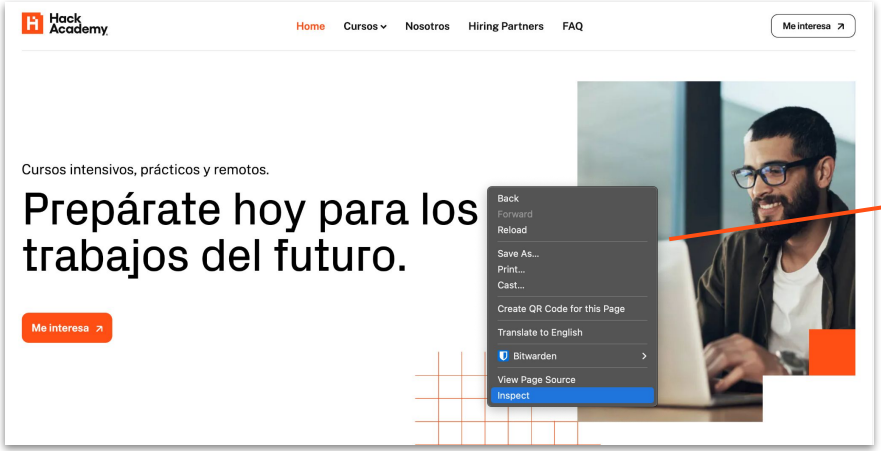
De hecho, sucede exactamente lo mismo en un editor de textos como Microsoft Word. La forma de crear un título en Word no es cambiándole el tamaño de letra a un párrafo sino creando un Encabezado (*Heading*).

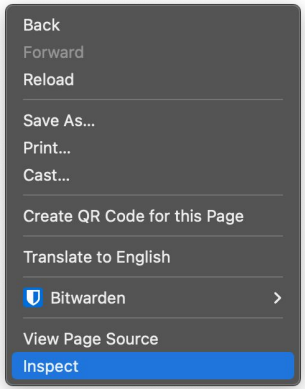
Chrome Dev Tools

Chrome Dev Tools (1/2)

Las Chrome Dev Tools son herramientas de desarrollo del navegador que ya están instaladas en Chrome. Entre otros beneficios, sirven para inspeccionar el código de cualquier sitio web.

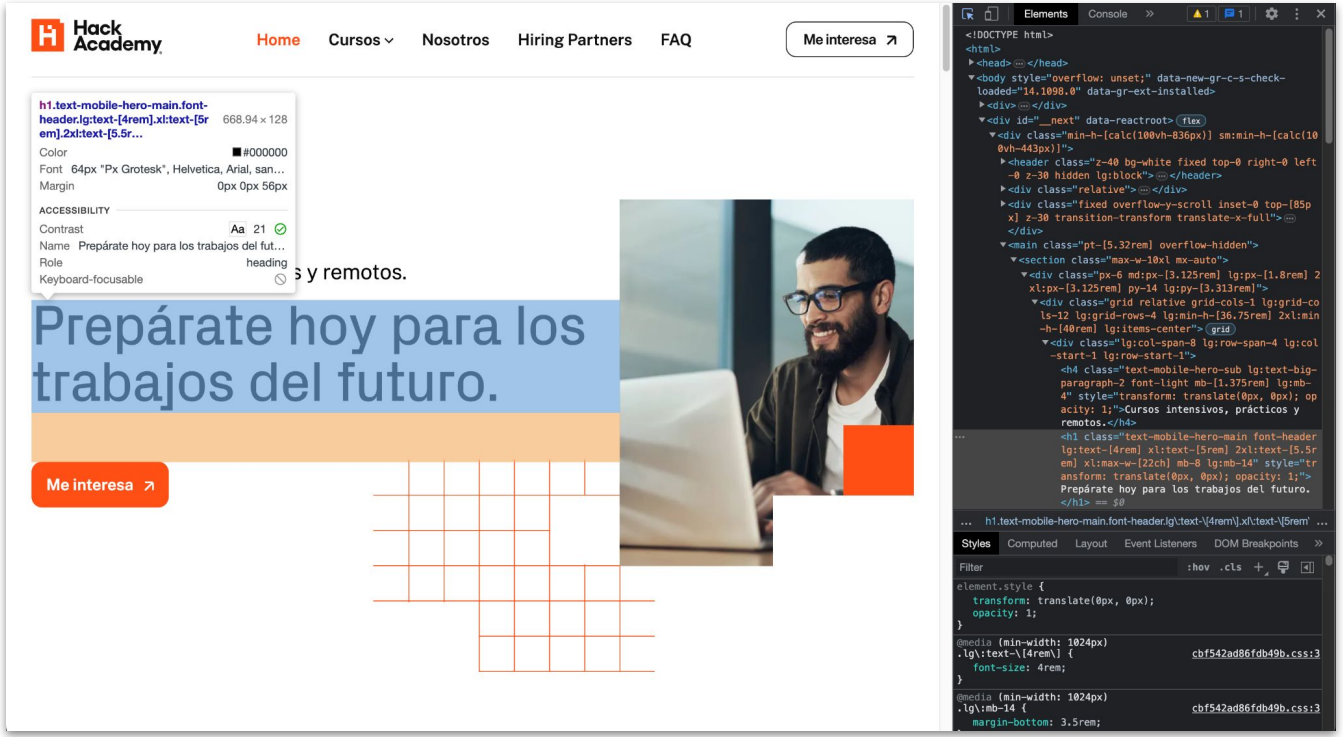
1. Hacer click derecho en cualquier lugar de la página.





2. En el menú contextual, elegir la opción **“Inspect”**o **“Inspeccionar”.**

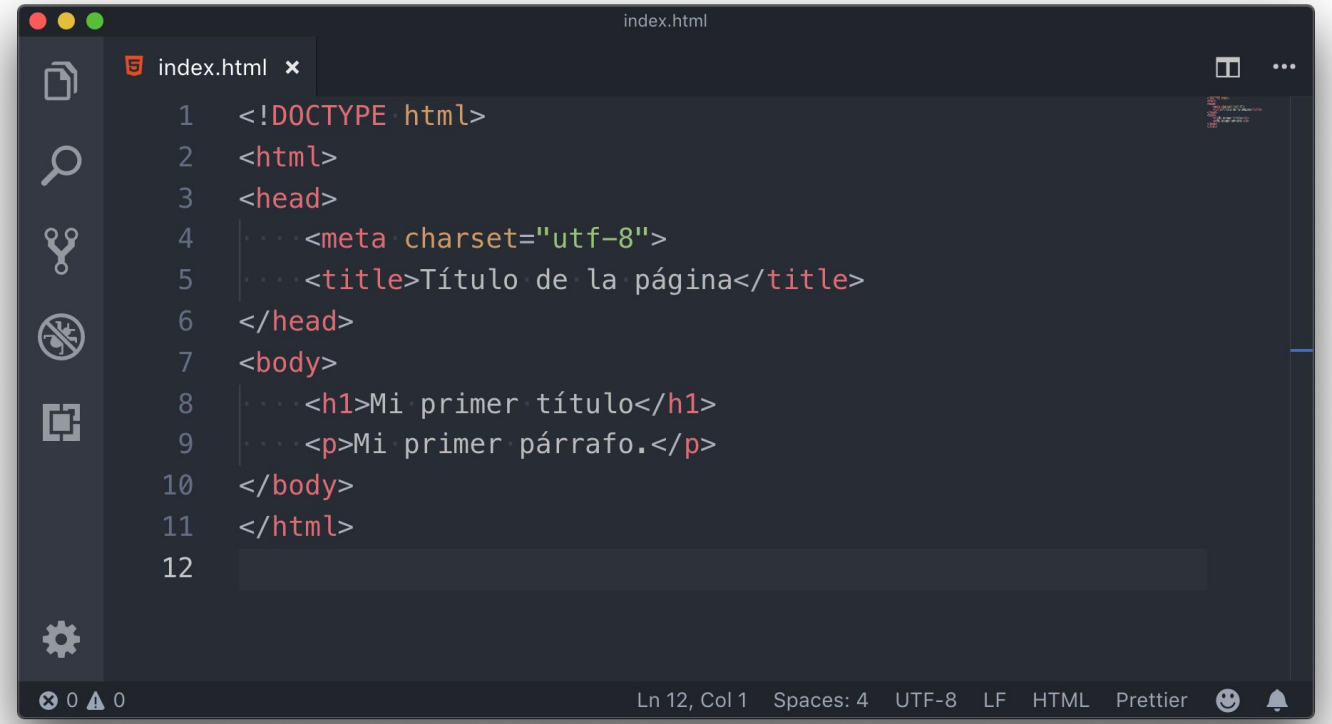
Chrome Dev Tools (2/2)



Se habilitará un panel lateral con el contenido **HTML** y las reglas **CSS**.

Visual Studio Code

Ahora, abrir el archivo anterior en Visual Studio Code 🎉



¿Qué ventajas tiene VSC con respecto al Bloc de Notas?

Visual Studio Code (VSC)

Para las próximas diapositivas, sugerimos que los estudiantes “sigan” al docente mientras que éste explica los conceptos teóricos en su PC.

Para ello, solicitamos que todos se creen una carpeta llamada `Clase01` y la abran en VSC con “File > Open Folder” en Windows (u “Open” en Mac/Linux). La carpeta también se puede crear directamente desde VSC (desde el cuadro de diálogo de “Open Folder”).

A esto le llamamos “abrir un proyecto” en VSC y lo haremos para cada ejercicio.

Una vez que tengan el proyecto abierto, crear un archivo `index.html` dentro de la carpeta. Luego, crear la estructura base de una página web con el atajo “!”.

Consejo para crear una etiqueta HTML en Visual Studio Code

Crear una etiqueta HTML en VSC (1/2)

Ej: si quieren crear un párrafo, **no** empiecen escribiendo el caracter “<”.

```
<p
```

Simplemente escriban “p” y luego utilicen el *autocomplete*:

```
p
```

Esto automáticamente les creará las etiquetas y , y les dejará el cursor en el medio de las dos etiquetas, pronto para que escriban el contenido del párrafo.

Crear una etiqueta HTML en VSC (2/2)

Al usar Visual Studio Code, no será necesario escribir los caracteres “<” ni “>” para crear etiquetas. Dejen que el *autocomplete* lo haga por ustedes.

Más elementos HTML...

¿Qué puede ir dentro del Body?

- Textos (párrafos, títulos, listas).
- Elementos estructurales (`<div>`, `<span>`, etc. – Ya los veremos).
- Links.
- Imágenes.
- Videos.
- Formularios (campos de texto, botones, etc).
- Comentarios en el código.

Body – Textos

```
<h1>Heading más importante</h1>
<h2>Heading de menor importancia</h2>
<h3>Heading aún menos importante</h3>

<p>Párrafos</p>

Salto de línea <br />


<p>
  <em>Énfasis</em> y <strong>Importante</strong>
</p>
```

Sólo crear un `h1` por página.

Hay hasta heading `h6`.

Algunos tags no necesitan cerrarse. En general, tratamos de no utilizar los
 (y nunca utilizar dos
 consecutivos).

Es posible "anidar" tags.

Body – Elementos estructurales (1/4)

Listas sin ordenar (sin numeración) `<ul>`:

```
<ul>
    <li>item de la lista usa "bullets" por defecto</li>
    <li>cada item va dentro de un tag</li>
    <li>no hay límites de items</li>
</ul>
```

Notar que la etiqueta `<li>` es *hija* de la etiqueta `<ul>`.

Listas ordenadas (con numeración) `<ol>`:

```
<ol>
    <li>números en lugar de bullets</li>
    <li>siempre dentro de estos tags</li>
</ol>
```

Notar que la etiqueta `<li>` es *hija* de la etiqueta `<ol>`.

Body – Elementos estructurales (2/4)

Elemento `<div>`:

```
<div>
    ...
    ...
</div>
```

Son elementos muy importantes en HTML que permiten crear “divisiones” (**secciones**) en el documento. Son muy útiles para agrupar otros elementos.

El div es un elemento genérico, no representa nada en particular, **no** tiene **semántica**. Esto se diferencia de, por ejemplo, un `<p>`, que *representa* un párrafo o un `<h1>`, que *representa* un título.

Body – Elementos estructurales (3/4)

Ejemplo de `<div>`:

Imagínense que necesitan declarar, en algún lugar de una página web, una sección donde aparecerán noticias del día.

Al día de hoy, HTML no trae integrado ninguna etiqueta llamada `<news>` ni similar. Por lo tanto, será necesario hacer uso de una etiqueta genérica (sin semántica) que represente la sección de noticias. Para ello está el `<div>`.

El significado de dicho `<div>` lo dará el contexto. Algo muy útil en estos casos, es identificar el `<div>` con un atributo llamado `id` que nos permitirá ponerle el nombre que queramos.

Ver código en la siguiente diapositiva.

Body – Elementos estructurales (4/4)

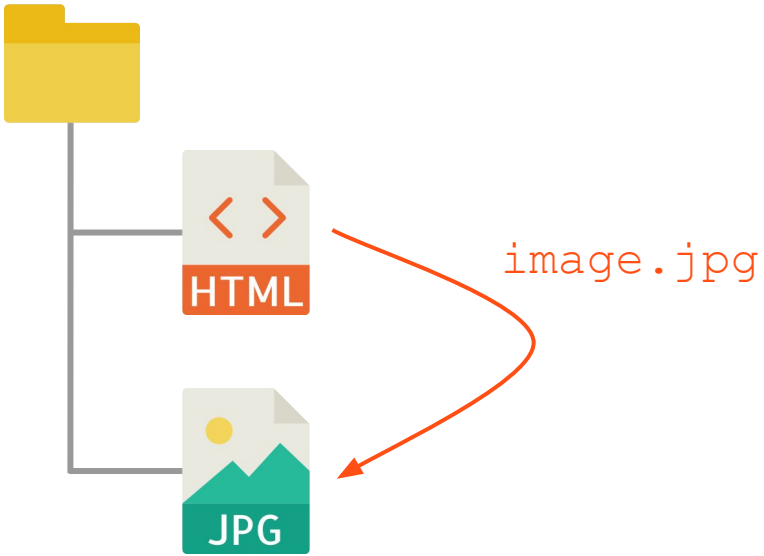
Ejemplo de `<div>`:

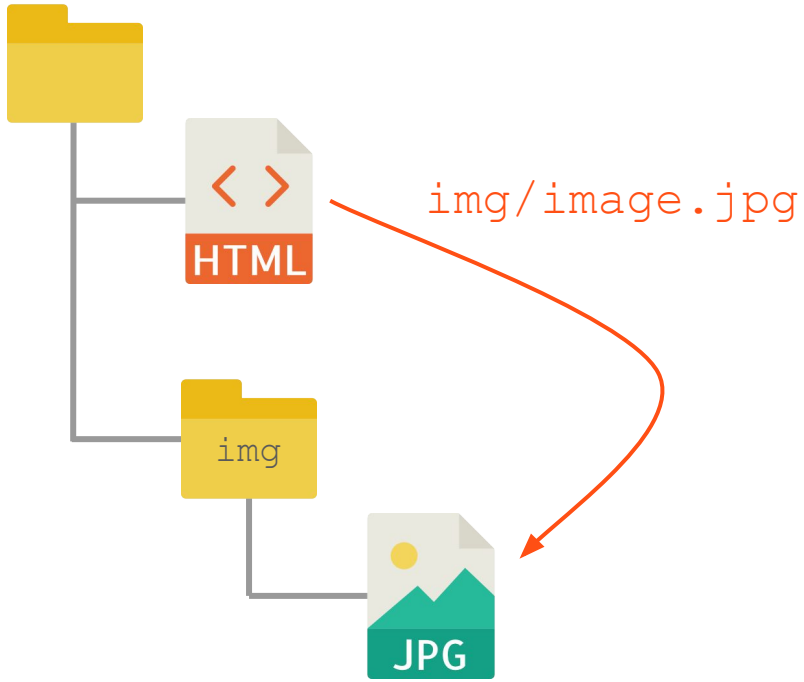
```
<div id="noticias">

   <h4>El software se está comiendo al mundo</h4>
   <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit...</p>

   <h4>Aprender a programar, te enseña a pensar</h4>
   <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit...</p>

</div>
```

Imágenes

Rutas relativas

Las rutas relativas indican la ubicación de un archivo con respecto al archivo HTML que lo "llama".





Body – Imágenes

Para insertar imágenes en un sitio web se utiliza la etiqueta `<img>`.

```
<img src="image.jpg" />
```

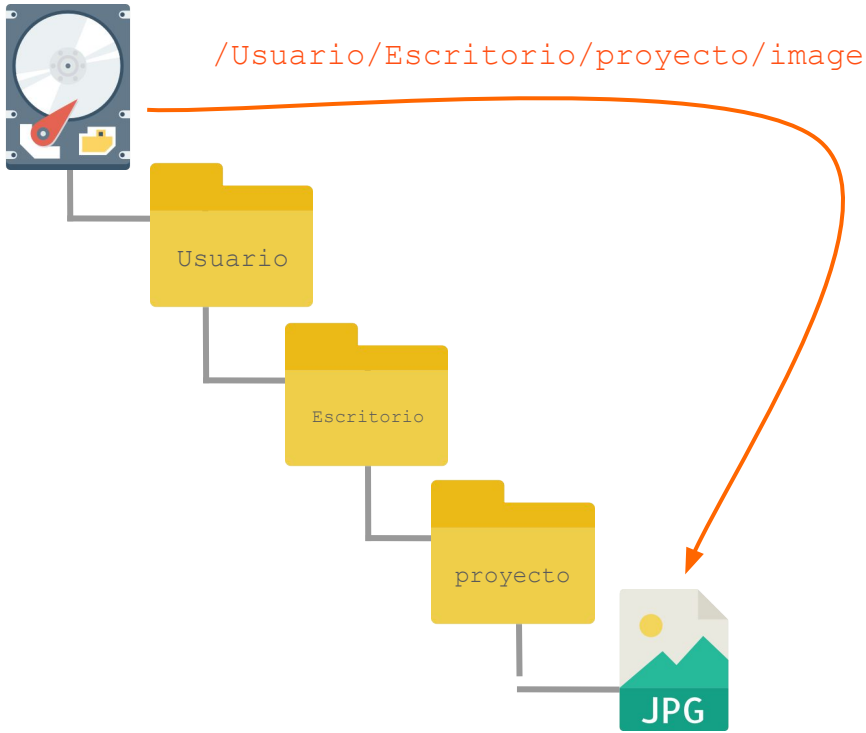
Notar que la etiqueta `<img>` es un tipo de etiqueta que no hace falta cerrar (sólo se abre).

Tengan cuidado cuando escriben el atributo `src`. Es común que lo escriban como `scr` y es un error difícil de encontrar.

La ruta de la imágen se puede escribir de forma **relativa** o **absoluta**. Ver siguiente diapositiva.

Rutas absolutas

En las rutas absolutas se indica la ruta completa al archivo desde la “raíz”.



La raíz puede ser el disco duro, el cual se indica con una barra / al comienzo de la ruta o la raíz puede ser un dominio, y en tal caso la ruta absoluta quedaría algo así: `https://ha.dev/img/image.jpg`.

Eviten usar rutas absolutas que hacen referencia a su propio disco duro, *ver siguiente slide*.

Body – Imágenes

Ejemplos de rutas relativas:

```
<img src="image.jpg" />
<img src="img/image.jpg" />
```

¿Qué es mejor? ¿Rutas relativas o absolutas? ¿Qué ventajas tiene cada una?

Ejemplos de rutas absolutas:

```
<img src="C:\Mis Documentos\Proyecto\image.jpg" />
<img src="/Usuario/Escritorio/Proyecto/image.jpg" />
<img src="https://ha.dev/img/image.jpg" />
```

Links

Body – Links (1/2)

Al igual a como sucede con las imágenes, los links pueden ser relativos o absolutos.

Ejemplo de un link relativo. Desde el archivo `index.html` se *linkea* al archivo `contacto.html`.

```
<a href="contacto.html">Página de Contacto</a>
```

Ejemplo de un link absoluto. Se suelen usar para *linkear* a páginas externas:

```
<a href="https://ha.dev">Link a Hack Academy</a>
```

Body – Links (2/2)

También es posible crear *links* a secciones dentro de una misma página. Para ello es necesario identificar a la sección usando el atributo `id`.

```
Ir a la <a href="#seccion2">sección 2</a> de la página.
...
...
...
...
...
<h2 id="seccion2">Título sección 2</h2>
<p>Contenido de la sección 2</p>
```

Comentarios

Body – Comentarios

```
<!-- ESTE ES UN COMENTARIO. -->
<!-- ESTE ES OTRO COMENTARIO. -->

<!--
    Este es un comentario,
    en varias líneas...
-->
```

Nota: Los comentarios no son visibles en la web, pero cualquiera puede verlos inspeccionando. el código fuente. En Visual Studio Code, pueden comentar una línea con el atajo **`CTRL + K + C`**.

Para finalizar...

HTML Tips

- Cerrar todos los *tags* (excepto `<br>`, `<img>`, `<meta>` y algunos otros).
- Usar indentación correctamente (En VSC: clic derecho y dar formato).
- Usar las minúsculas en los *tags*.
- Evitar espacios, tildes y eñes en los nombres de los archivos.

```
<DIV><ul><li>Contenido 1
</li><LI>Contenido 2
</LI></UL></div>
```

PÉSIMA PRÁCTICA

```
<div>
    <ul>
        <li>Contenido 1</li>
        <li>Contenido 2</li>
    </ul>
</div>
```

BUENA PRÁCTICA

Un código prolijo habla muy bien de un programador.

Por las dudas repetimos...

Cuiden la prolijidad de sus códigos.

🙌 Recomendación: Prettier (1/3)

Hay una extensión (*plugin*) para Visual Studio Code, llamada Prettier, que es muy útil (y muy popular) para mantener un código prolijo y consistente (*formateado*).

Para instalarlo deberán abrir el panel de Extensiones de VSC y buscar "Prettier". Luego elegir la extensión llamada *"Prettier - Code formatter"* desarrollada por Esben Petersen y darle click en el botón *"Install"*.

✅ También recomendamos que configuren VSC con la opción *"Format on save"* que sirve para que el código se ordene de forma automática, cada vez que le dan "guardar" a un archivo, lo cual es muy cómodo. Si no saben cómo realizar esta última configuración, vean este tutorial.

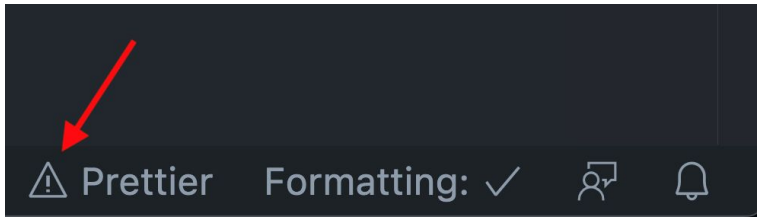
❌ Para evitar conflictos, no recomendamos instalar más de un *formateador* de código.

🙌 Recomendación: Prettier (2/3)

Prettier es un *formateador* de código “opinionado” por lo que no precisa ninguna configuración extra. Por defecto, él ya toma un montón de decisiones sobre cómo *formatear* su código. Si lo desean, lo pueden configurar de forma “personalizada”, pero no precisamos que lo hagan (menos en este curso). Lo instalan y listo.

✓ Cuando estén usando Prettier, siempre estén atentos a la esquina inferior derecha de VSC. Si aparece un *“tick”* o un *“doble tick”* quiere decir que la sintaxis de su código está OK.

⚠️ En cambio, si en algún momento aparece un símbolo de exclamación, quiere decir que hay un problema y que lo deben solucionar. Cuanto antes se den cuenta de que apareció el símbolo de exclamación, mejor, así les será más fácil solucionar el error.



🙌 Recomendación: Prettier (3/3)

Formateador por defecto:

A veces, podría llegar a ser necesario que se le explicite a VSC que se desea usar a Prettier como el *formateador* por defecto para HTML, CSS y JavaScript.

Esto es particularmente importante si es que ya habían instalado otros *formateadores* de código. De lo contrario, si hay varios *formateadores* de código, VSC podría no saber cuál usar. Esto se soluciona haciendo *click* derecho arriba de un archivo (por ejemplo: HTML) y luego clic en la opción *"Format document with..."*.

Luego seleccionar la opción *"Configure default formatter..."* y finalmente elegir a Prettier del listado.