

**SheCodes Basics**

## WELCOME

* Slack: medio de comunicación, para consultar dudas
* Instructions = software
  + Visual Studio Code
  + CodePen 🡪 para hacer algunos ejercicios
  + Slack 🡪 descargar
* Presentarse

En cada tema hay:

* Lección
* Challenge 🡪 enviar la solución
  + Cuando se envía aparecen sugerencias
  + Si te atascas, se puede buscar en Google con el lenguaje usado, por ejemplo “html lists”. O ir al enlace de ayuda para preguntar en Slack (usar :question: shecodes-week-Duda
  + Si encontramos errores, poner en Slack :bug:
* Homework 🡪 proyecto semanal con una fecha límite

Proyecto final:

* Dura una semana completa.
* Al aprobarlo, se genera un certificado del curso

## INSTRUCTIONS

* Lecciones semanales con vídeos, retos y una tarea final.
  + Retos: no tienen feedback, pero se da una propuesta de solución.
  + Tarea final: con feedback
* Programas que se utilizan:
  + Visual Studio Code
  + Chrome
  + CodePen
  + Slack

## GUIDELINES

* Be polite and nice
* Take a break if you feel frustrated
* You can request for an extension to send the tasks
* Ask questions through Slack, in the weekly channel
* Be patient
* Read other people’s questions before asking
* Have fun!

## **WEEK 1**

### LESSON 1

* HTML y CSS son lenguajes de marcado, sencillos de utilizar
  + HTML = estructura: párrafos, títulos…
  + CSS = estilo
* Challenges: Hay de dos tipos:
  + CodePen:
    - Página para principiantes
    - Habrá que compartir lo que hagamos aquí: Save > Copiar URL > Pegarla en el recuadro de envío en SheCodes
  + Visual Studio Code:
    - Más profesional
    - Abrir Chome en paralelo para comprobar lo que hacemos.
    - Intentar usar poco el ratón
    - Crear un archivo \*.html (en minúsculas)
    - Instalar la extensión Prettier para comprobar el código generado. Tiene que estar siempre con ✔. Si hay errores sale ❌
    - Guardar con frecuencia para ver los resultados en Chrome

### LESSON 2

Profesor: Matt

Full stack developer

### LESSON 3

#### Elementos HTML

HTML permite crear páginas básicas. Es fácil de aprender.

Los elementos HTML permiten dar estructura y formado.

* <h1> Encabezado 1
* <h6> Último nivel de encabezados
  + Si ponemos <h7> no pasa nada. No se pueden usar elementos que no estén creados en la sintaxis.
* <p> Párrafo
* <strong> Negrita
* <em> Cursiva

Los elementos se pueden anidar

Importante cerrar siempre los elementos: </h1>

* <br> Introduce un cambio de línea. Es un elemento vacío, por lo que la etiqueta es tanto de apertura como de cierre. Hay que poner <br />
* <hr> Introduce una línea horizontal. También se puede escribir <hr />

Listas:

* <li> Para cada elemento de la lista
* <ul> Engloba los <li> en una lista desordenada. Con viñetas
* <ol> Engloba los <li> en una lista ordenada numéricamente.
  + Dentro de <ul> y <ol> puede haber <li>, pero dentro de <li> podemos tener <p>, <strong>…

Challenge 🡪

Recrear la pantalla mostrada con HTML.

* Encabezado <h1>
* Lista desordenada <ul> <li>
* Línea <hr />
* “More info” como <p> <strong>
  + Abierto a otras opciones, como <h2>, <h3>…

### LESSON 4

#### Atributos HTML

Dan información extra al navegador.

<a> para crear enlaces:

Entre las etiquetas se pone el texto a mostrar  
 Con el atributo se pone la URL

<a href = ”URL”>  
<a href = “URL” target = \_blank”> 🡪 así el enlace se abre en otra pestaña nueva

<img /> es un elemento vacío, sin texto

<img src =”URL de la imgane” /> Indicando la fuente de donde se obtiene la imagen  
 <img src =”URL” width 0 “píxeles” /> para indicar el tamaño en píxeles

Challenge 🡪

Crear enlaces con el CodePen creado anteriormente.

* Enlaces: Dentro de <li> incluimos <a> con el atributo href con la URL y target para abrir en otra pestaña.
* Imagen: con <img> ponemos la URL de la imagen con el atributo src, y el tamaño con width, y una descripción textual de la imagen con el atributo alt.

### LESSON 5

#### Herramientas de desarrollador

Acceso al código fuente de cualquier página con Chrome 🡪

Botón derecho > Ver código fuente de la página

Con el botón derecho > Inspeccionar, podemos ver los elementos, modificarlos temporalmente para ver cómo quedarían los cambios. Esos cambios no se guardan.

Se pueden usar comandos:

* Ver código fuente: Ctrl + U
* Ver Inspeccionar: Ctrl + Shift + J

### LESSON 6

#### Editor de código

Vamos a usar Visual Studio Code.

* Abrir archivo nuevo
* Guardarlo como index.html
* Abrir con Chrome el archivo html creado para ver los cambios que hagamos en directo
* Guardar el archivo de VSC con Ctrl + S
* Refrescar Chrome con Ctrl + R

Estructura de archivos HTML:

<html>  
 <head> 🡪 metadatos  
 <body> 🡪 lo que se ve en pantalla

Debemos poner información previa con DOCTYPE. Para ello, escribimos ! y lo seleccionamos del listado.

En <title> ponemos el nombre que aparecerá en la pestaña del navegador

Extensiones de VSC interesantes:

* Prettier: En File > Preferences > Settings buscar format y marcar “Format on paste” y “Format on save”. Prettier deja el documento ordenado y comprueba la existencia de errores.

Challenge 🡪

Convertir nuestro archivo de CodePen a VSC y enviar el archivo \*.html generado.

### LESSON 7

#### Prettier

En File > Preferences > Settings buscar “default formatter” y seleccionar esbenp.prettier-vscode

Organiza el archivo.

Si está correcto lo marca con ✔

Si hay error, pone una advertencia ⚠ 🡪 se puede abrir para ver dónde está el error, aunque no siempre es muy ajustado diciendo dónde está o la explicación no es muy clara, pero suele indicar la línea en la que está el error.

### LESSON 8

#### Slack

Nos sirve para pedir ayuda y resolver dudas.

Hay muchas señales para saber si hay error:

* Prettier
* Colores en el código

Lo mejor ante un error es buscarlo en Google.

Si queremos consultar en Slack, pinchamos en el botón que hay en cada lección y nos redirige al chat correspondiente.

Antes de preguntar es mejor ver si alguien lo ha consultado antes, se puede buscar con Ctrl + F.

Para publicar una pregunta usamos:

* :question: aparece el icono ❓
* Poner en qué lección estamos
* Explicar qué sucede
* Poner el código en una línea nueva: para pegarlo hay que seleccionar “code block”
* Cuando el equipo revisa las dudas, ponen un emoji de ojos para indicar que están en ello, y el check verde cuando se ha resuelto.

### LESSON 9

#### Selectores de CSS

CSS es un nuevo lenguaje de marcado.

Sirve para dar estilo a las páginas.

Tiene su propia sintaxis.

En la parte de <head> tenemos que incluir la etiqueta <style> para especificar estilos de cada parte. La estructura es:

<style>  
 h1 {  
 color: blue;  
 font-size:42px;  
 text-align:center;  
 }  
 p {  
 color:green;  
 text-align:right;  
 text-transformation:uppercase;  
 }

VSC predice lo que vamos a poner y no hace falta saber todas las propiedades.

Hay cientos de propiedades que se pueden aplicar.

Challenge 🡪

Poner el título en rojo, subrayado, con un tamaño concreto… Hay que buscar en Google algunas propiedades que no se han explicado en el vídeo.

Todo lo que está dentro del elemento <style> está en CSS y no en HTML.

* Color 🡪 color:red;
* Tamaño 🡪 font-size:48px;
* Subrayado 🡪 text-decoration:underlined;
* Grosor de la fuente 🡪 font-weight:lighter/100; 🡪 Se puede poner el valor en texto o en valor numérico.

No importa el orden en que se pongan los atributos.

### LESSON 10

#### Clases en HTML

Sirven para dar un estilo concreto. Por ejemplo:

* Dar distintos estilos a los elementos de una lista.
* <li class=”free”>
  + En CSS ponemos:
    - .free{}

Las etiquetas del atributo class las elegimos nosotros. Tienen que ser descriptivas, pero no ser colores (por ejemplo, no poner un nombre de clase “red” si queremos que eso vaya en rojo, porque podría provocar errores).

Se pueden poner varias etiquetas, separadas por un espacio:  
 <li class = “free story”>

Si la etiqueta tiene varias palabras, unirlas con un guion:  
 <li class = “free women-workshop”>

Si queremos dar distintos estilos a la lista, como por ejemplo quitarle las viñetas, usamos:  
 li {  
 list-style:none;  
 }

<div> Sirve para añadir una clase sobre algún contenido y darle estilo con CSS.

<span> Sirve para añadir una clase sobre algún contenido, especialmente imágenes y texto, y darle estilo con CSS.

* Normalmente <div> se usa para un bloque de elementos y <span> para una parte del contenido, por ejemplo, una palabra dentro de una oración.

Challenge 🡪

Cambiar cada uno de los colores de los elementos de una lista.

¡Cuidado! Como dentro de <li> está <a>, la clase la tenemos que poner en <a>, no en <li>.

* En <a> cambia el formato del texto.
* En <li> cambia el formato de la viñeta.

### LESSON 11

#### Trucos de VSC

No hace falta personalizarlo en exceso.

* Personalizar los colores de la sintaxis:
  + Extensión: Dracula (la que usa Matt)
* Temas: se pueden buscar en Google:
  + VSCodethemes.com
* Fuentes: buscar Source Code Pro
* Spaces: suele estar en 2, ayuda a organizar el código
* Tamaño de la fuente: se cambia en Settings

### LESSON 12

#### Propiedades en CSS

Para dar estilo al texto.

* Color
* Font-size: en píxeles
* Text-align: alineación. No se recomienda justificarlo.

VSC es predictivo, va completando lo que vamos escribiendo. También da explicaciones,

* Background: fondo del apartado elegido
* Text-transformation: permite poner todo en mayúsculas, etc.

Si no sabemos cómo hacer algo en concreto, como poner en cursiva, lo mejor es buscar en Google. Podemos poner: CSS italics text. Es mejor consultar los resultados de MDN. La página pertenece a Mozilla y lo actualizan bastante.

Challenge 🡪

Practicar las propiedades de CSS:

* Centrar y subrayar el texto.
* Añadir texto con formato.

Se recomienda en <style> poner primero todos los elementos y luego las clases.

### LESSON 13

#### Colores en CSS

Hay 3 formas de referenciarlos:

* Poner el nombre: green, blue, red… (colores nativos).
  + No se usa mucho este sistema, es poco preciso.
* Sistema hexadecimal: #FFFFFF
  + Los suelen facilitar el diseñador gráfico. Con la extensión de Chrome ColorZilla podemos captar los colores usados en pantalla y obtener su valor hexadecimal. Obtenemos el valor exacto del color: #875DF1
* Sistema rgb (red-green-blue): (163, 162, 243)
  + Nos da 3 valores que combinados dan un color. Nos sirve en CSS para poner transparencias:
    - Color:rgb (135, 93, 241) 🡪 color sólido
    - Color: rgba(135, 93, 241, 0.7) 🡪 transparencia. El valor 0.7 es un valor entre 0 y 1 que muestra la opacidad. 0 es total transparencia y 1 opaco.

Suele ser el diseñador gráfico quien nos da estos valores.

Herramientas para encontrar paletas de color:

* Colorhunt.co 🡪 ofrece paletas
* Materialpalette.com 🡪 permite crear paletas

Challenge 🡪

Cambiar colores usando valores hexadecimales.

* Las letras de los valores HEX pueden ir tanto en mayúsculas como en minúsculas.
* Para cambiar el color de fondo de la página, se cambia el color de <body>, que es el elemento que engloba todos los demás elementos.

body {  
background: #f5f0fe;  
}

* + Normalmente en <body> no se aplica ningún otro tipo de estilo.

### LESSON 14

#### Paletas de colores

Herramientas para trabajar los colores:

* ColorZilla: extensión de Chrome para captar los colores de la pantalla. Da el número HEX y RGB.
  + Pero para obtener el color exacto es mejor acceder a Inspeccionar en la web.
* Shecodes.io/palettes: Web con paletas de 4 colores. Da los códigos de cada color e incluso el código CSS con los cuatro colores utilizados.
* Shecodes.io/gradients: Web con degradados de color. Incluye el código preciso para cada uno. Así no es necesario entender cómo se ha creado, sino que se puede copiar directamente.

No es aconsejable cargar la web con colores, porque acaba siendo desagradable a la vista.

### LESSON 15

#### Posicionamiento en CSS

Para alinear texto a izquierda, derecha o centro, usamos text-align.

Pero esto no funciona con imágenes. Para ello, usamos dos propiedades en CSS:

img {  
width: 200px; 🡪 tamaño de la imagen, puede ir en la parte de HTML también  
display: block;  
margin: 0 auto;  
}

En CSS y HTML todos los elementos son block o inline.

* Los elementos block crean una línea antes y después de ellos.
* Los inline van seguigos, como un texto continuo.
  + Para centrarlos hay que convertirlos en elementos block. Se hace con display:block;
  + Después, se alinean con la propiedad margin. Si le damos los valores 0 auto se centra 🡪 margin: 0 auto;

Challenge 🡪

Centrar elementos de la web:

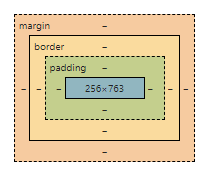
* “More info” 🡪 text-align 🡪 <p> es un elemento block
* Foto 🡪 <img> es un elemento inline. Lo tenemos que convertir en block y luego centrarlo.
  + También podemos poner la información sobre tamaño en CSS en lugar de en la parte de HTML utilizando la propiedad width con un valor en píxeles.

### LESSON 16

#### Espaciado en CSS

Con CSS podemos controlar todo el espaciado. Por ejemplo:

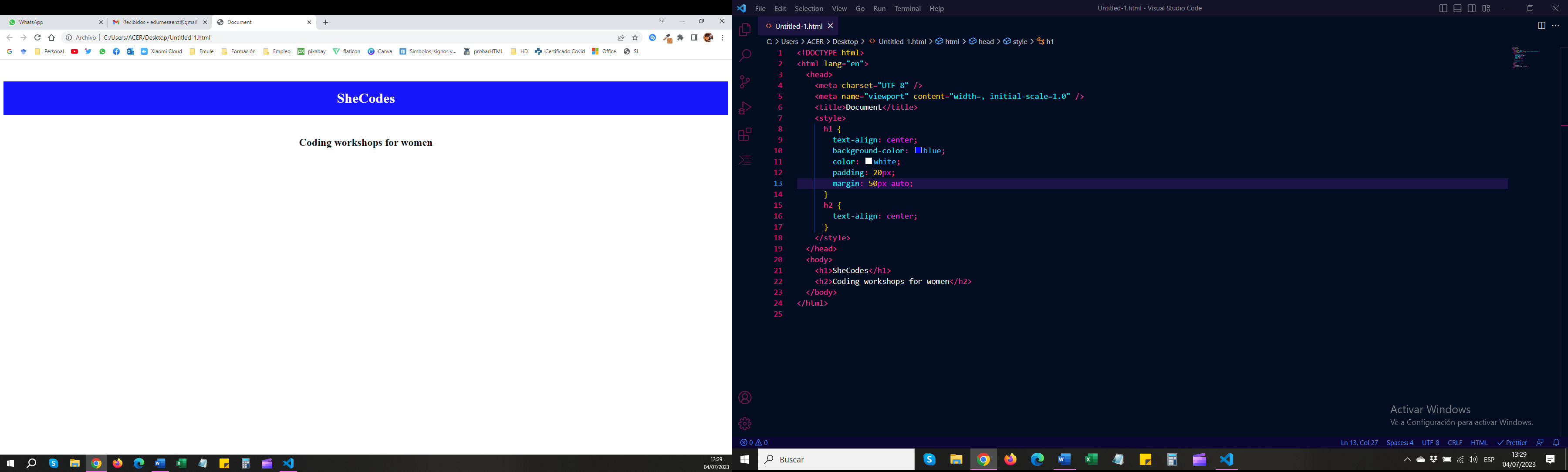
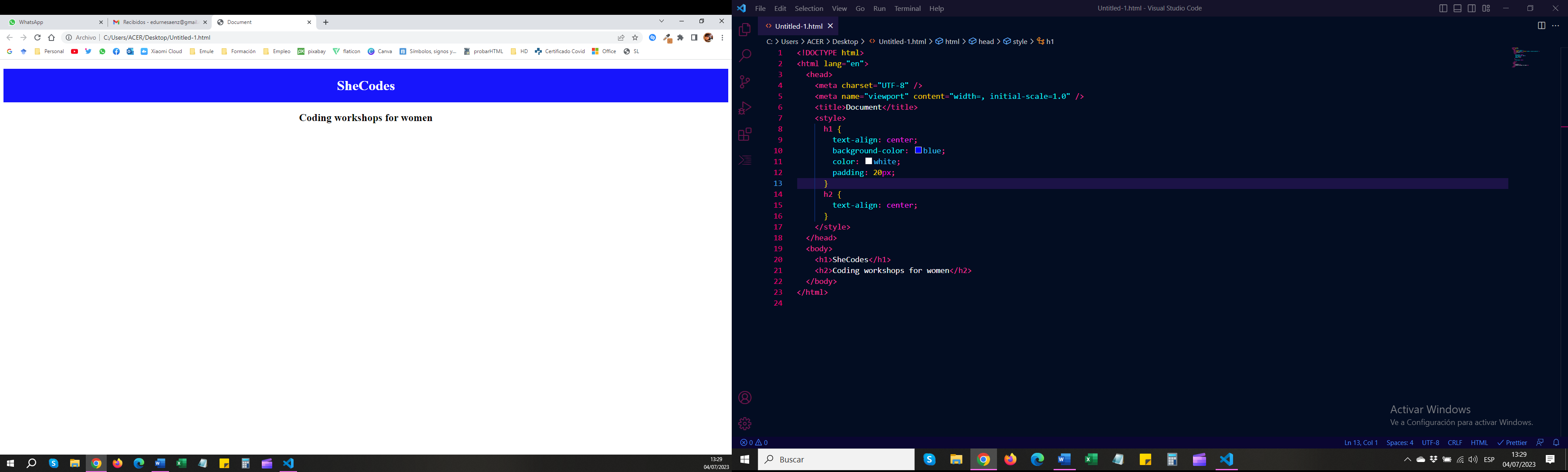
h1{  
text-align: center; 🡪 texto centrado  
background-color: blue; 🡪 color de fondo del texto  
color: white; 🡪 color de la letra  
padding: 20px; 🡪 espaciado dentro del elemento  
}



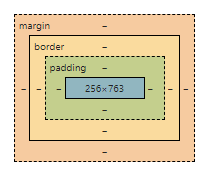
|  |  |
| --- | --- |
|  | sin padding |
|  | con padding |

*Si ponemos 4 valores en el padding, ponemos un padding diferente para cada lado, en el orden de las agujas del reloj:  
 padding: 40px 50px 100px 20px; 🡪 arriba, derecha, abajo, izquierda*

Margin: espaciado alrededor del elemento. Se le pueden dar 4 valores también. Si queremos que esté completamente centrado, ponemos un valor y auto: margin: 50px auto;



Si accedemos a Inspeccionar nos muestra los valores de cada elemento en todas las propiedades en este dibujo:



Challenge 🡪

Crear un botón con texto dentro. Aplicar espaciado en él.

* Hay un elemento de HTML llamado <button> para insertar botones.
* Damos formato con:

button {

display: block;

margin: 0 auto 30px; 🡪 da espaciado debajo

padding: 15px 30 px;

font-size: 18px;

color: #885df1;

background: transparent;

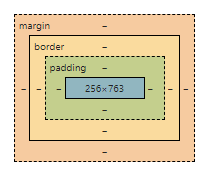
}

Lo importante en este caso es no poner un enlace en el botón, ya que son dos elementos diferentes. El link nos lleva a otra página y el botón hace un comportamiento distinto.

### LESSON 17

#### Bordes en CSS

La tercera capa que se puede añadir a un elemento. Por ejemplo, con una imagen tendríamos:



**FOTO**

En la foto (<img> en la parte de HTML) podemos incluir src, la fuente de la que extraemos la imagen, y alt, un texto que describa la imagen.

* Padding: espacio que forma parte del elemento.
* Margin: espacio externo al elemento.
* Border: contorno entre padding y margin
  + Tiene tres valores: tamaño (en píxeles), estructura (sólido, punteado, círculos…) y color.
  + border: 10px solid red;
  + La estructura normalmente es sólida y en color gris.
* Se pueden redondear las esquinas de la imagen con la siguiente propiedad:
  + boder-radius: en píxeles. Normalmente de usan 4px.
  + Para poner una imagen totalmente redonda se usa 50%, como en las fotos de perfil de Instagram.
  + Se pueden poner valores diferentes en cada esquina para hacer formas. Los valores van también desde arriba en el orden de las agujas del reloj. Si arriba-abajo y derecha-izquierda son iguales, se pone solo dos valores.

Challenge 🡪

Cambiar el aspecto del botón: redondear y dar sombra.

* Border: 1px solid #885df1;
* Border-radius: 30px;
* Box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.3); 🡪 usa un degradado

### LESSON 18

#### Pseudo-selectores de CSS

Para añadir efectos cuando pasamos el ratón sobre un elemento. Por ejemplo, encima de un botón.

button: hover {

background: white; 🡪 estilo diferente al que tiene previamente el botón

color: blueviolet;

cursor: pointer; 🡪 el curso se convierte en una mano

}

Para hacer que el cambio de estilo no sea inmediato, sino una transición, se añade esto en button, no en button:hover:

button {  
transition: all 200ms ease-in-out;  
}

* All = todos los elementos
* 200ms = tiempo en milisegundos
* Ease-in-out = tipo de transición. Esta es más natural, pero otro tipo es linear

Se pueden aplicar estas transiciones a cualquier elemento (h1, p, img…). Por ejemplo, para cambiar el color de un link en el que se ha clicado anteriormente. Pondríamos:

a:visited {}

Challenge 🡪

Cambiar el color del botón al pasar sobre él y la forma del cursor.

* Añadimos el selector hover para cambiar el color del fondo y el texto.
* Añadimos la transición en button para que el cambio no sea tan brusco.
* Cambiamos el cursor por una mano: pointer. Existen otras formas de cursor, pero esta es la más habitual.

### LESSON 19

#### Estilos CSS por defecto

Todos los elementos HTML tienen un estilo CSS por defecto.

* <h1> tiene un tamaño y características concretas, como estar en negrita, con espaciado alrededor… Lo podemos ver en la opción Inspeccionar.
* <h2> también lo tiene
* <ul> suele traer padding a la izquierda. Tiene un estilo diferente a <li>.
  + <li> es un elemento list-item.
  + <ul> es un elemento block con list-style-type:disc. Esto hace referencia al tipo de viñeta (el punto). Si lo queremos quitar y no usar una viñeta, ponemos: list-style:none;
* Los enlaces <a> son un elemento con un color por defecto y decoración (text-decoration: underlined) y un cambio en el cursor.

Challenge 🡪

Cambiar la negrita de <h2>.

Hacer que la lista de links no tenga viñetas, estén todas en una línea y se subrayen al pasar el cursor por encima.

* Quitamos la viñeta con:

ul {

list-style: none;

padding: 0;

}

* Ponemos toda la lista en una línea:
  + Hacemos que <li> sea un elemento inline.
  + Quitamos la línea del enlace:

a {

text-decoration: none;

}

a:hover {

text-decoration: underline;

}

* + Centramos los elementos de la lista:

ul {

text-align: center;

}

* + Separamos las palabras del listado:

li {

margin: 0 15px;

}

### LESSON 20

#### Elementos contenedor de HTML

<div> y <span>

Los demás elementos tienen un estilo predefinido, mientras que <div> y <span> no lo tienen. Estos elementos nos sirven para darle clase.

* <div> es un elemento block
* <span> es un elemento inline

<div class = “message”></div>

Con esto damos estilo únicamente a la clase message con CSS

Se pueden insertar elementos <div> dentro de otros elementos <div>. Al ser un elemento block, cada <div> irá en una línea diferente. Este es el elemento más popular de HTML.

<span> es un elemento neutro inline. Si ponemos uno dentro de otro, en este caso irá en la misma línea todos.

|  |  |  |
| --- | --- | --- |
| <div>  <span> </span>  <br />  </div> | = | <div>  <div>  </div>  </div> |

Estos dos elementos son muy útiles para dar estilo.

Challenge 🡪

Usar <div> y <span> para añadir información.

* Texto centrado en dos colores.
  + Primero escribimos el texto
  + Usamos un <div>
  + Le damos la clase “contact-details”
  + Damos estilo: centramos, cambiamos el tamaño de fuente
  + Cambiamos el color a algunas palabras. Como queremos que todo esté en una línea, usamos <span> con la clase “contact-platform”.
  + Damos estilo a .contact-platform: color.

### LESSON 21

#### Resumen semana 1

Cuestiones importantes:

* Cerrar las etiquetas
* Aprender de nuestros errores

## **WEEK 2**

### LESSON 1

#### JavaScript

* Lenguaje de programación
* Crea interacción entre HTML y el usuario
  + Tiene acceso a información: horario local, modelo de PC del usuario, navegador utilizado, cámara, ubicación…
* Importante abrir la consola de Inspeccionar: nos muestra dónde hay errores, algo que no aparece en VSC: Prettier no indica errores de JavaScript. Aunque no se entienda el tipo de error, nos indica en qué línea se ha cometido.
* Acciones que hace, por ejemplo: al dar like actualiza el contador de likes en Instagram

La versión de JS que vamos a utilizar es ES5.

### LESSON 2

#### JavaScript en el navegador

Podemos acceder a la consola de JavaScript desde cualquier navegador clicando en un espacio en blanco > Inspeccionar > Consola.

Nos permite darle instrucciones, aunque el código lo creamos únicamente con VSC.

### LESSON 3

#### Variables en JavaScript

Las variables actúan como cajas.

Para crear una variable desde la consola:

let course = “SheCodes”;

* Let = función
* Course = variable
* “SheCodes” = string de texto, lo que contiene la variable

La próxima vez que escribamos course en la consola, dará como resultado SheCodes.

Podemos juntar variables, pero si son dos de texto tenemos que añadir un espacio:

* Let course = “SheCodes”
* Let tool = “VSCode”
* \* Course + tool 🡪 SheCodesVSCode
* Course + “ “ + tool 🡪 SheCodes VSCode

No podemos volver a crear una variable con un nombre que ya hayamos utilizado. Para cambiar el valor de una variable hacemos así:

* Course = “SheCodes+”
  + Así podemos actualizar el valor de una variable en cualquier momento.

En la consola no se puede reescribir lo que hemos puesto antes, pero para borrar todo podemos darle a actualizar la página.

Challenge 🡪

En la consola, poner una frase con nuestra ciudad y país, pero con variables city y country. Ejemplo: I am from Toulouse, in France.

* Mi propuesta:
  + Let city = “Logroño”;
  + Let country = “Spain”;
  + “I am from “ + city+ “, in “ + country
* Respuesta: crear además una variable sentence que incluya las dos variables anteriores.
* ¡CUIDADO! Si creamos esa variable sentence y luego modificamos los valores de city o country, estos no se actualizan en sentence.

### LESSON 4

#### JavaScript en HTML

Cómo incluir JavaScript en un HTML:

* Lo ponemos en <body>, justo al final, delante de </body>
* Dentro de la etiqueta <script>.

Crear variables:

* Let VARIABLE = “contenido”;

Función alert:

* Sale un popup al cargar la página.

alert(“SheCodes”);

* + Podemos incluir una variable entre paréntesis:

let course = “SheCodes”;

alert(course);

Nombrar variables con varias palabras:

* No usar espacios
* No usar guiones
* Usar mayúsculas para identificar dónde empieza la nueva palabra: courseName
* Importante poner nombres claros, fáciles de entender

Funcionamiento del código:

* El ordenador lee línea a línea para ejecutarlo.

Errores:

* No los detecta VSC
* Debemos comprobar en la consola del navegador. Si hay error aparece esto ⊗

Función promp:

* Aparece un popup con un cuadro de escritura donde el usuario responde a una pregunta.

prompt(“What language do you speak?);

* Podemos crear una variable que contenga la respuesta del usuario y con eso crear una alerta:

let language = prompt(“What language do you speak?”);

alert(“You speak “ + language);

Challenge 🡪

Crear prompts para preguntar la ciudad de nacimiento y el nombre completo.

Mi propuesta:

* let hometown = prompt(“What is your hometown?”);
* alert(hometown);
* let firstName = prompt(“What is your first name?”);
* let lastName = prompt(“What is your last name?”);
* alert(firstName + “ “ + lastName);

En el segundo alert, en lugar de eso ponerlo todo en una variable:

* let fullName = firstName + “ “ + lastName;
* alert(fullName);

### LESSON 5

#### Estamentos IF en JavaScript

Añadimos lógica básica.

Podemos crear una variable a partir de un prompt con texto introducido por el usuario. Luego podemos crear una alerta específica en función del texto introducido: para esto utilizamos la función if.

let languageSpoken = prompt(“Which is your language?”);

if (languageSpoken === “French”) {

alert(“Bonjour”);

}

Si el texto introducido es “French”, saltará un popup que dice “Bonjour”. Si el texto es diferente, se indica con !==

if (languageSpoken !== “French”) {

alert(“👋🏼👋🏼👋🏼”);

}

Si utilizamos cifras en lugar de texto, hay más símbolos para hacer la comparativa: Por ejemplo, podemos preguntar la edad:

let age = prompt(“How old are you?”);

if (age >= 16) {

alert(“You can drive”);

}

* >= mayor o igual a

if (age < 16) {

alert(“You can’t drive 😔”);

}

* También podríamos poner <= 15 (menor o igual a)

Challenge 🡪

Crear un prompt en el que preguntar la edad. Dependiendo de si es mayor o igual, o menor a 18 sale un mensaje u otro.

Mi propuesta:

let age = prompt(“How old are you?”);

if (age >= 18) {

alert(“You can apply to SheCodes”);

}

if (age < 18) {

alert(“You cannot apply”);

}

* Cuidado: poner todos los símbolos y no añadir otros innecesarios.
* Cuidado: no poner las cifras entre comillas, porque entonces se convierten en texto y no se pueden hacer comparativas matemáticas con ellas.

### LESSON 6

#### Estamentos IF ELSE en JavaScript

Podemos simplificar el código anterior, que tenía dos IF, con un IF ELSE.

* if (languageSpoken === “French”)
* if (languageSpoken !== “French”)
  + Utilizar estos dos es redundante, ya que si uno es falso, el otro es verdadero.

let languageSpoken = prompt(“Which language do you speak?”);

if (languageSpoken === “French”) {

alert(“Bonjour);

} else {

alert(“👋🏼👋🏼👋🏼”);

}

* Si es cierto que languageSpoken es igual a “French” hace una cosa. Si no es igual, hace otra.

Además, podemos introducir más código dentro del IF:

if (languageSpoken === “French) {

let name = prompt(“What is your name?”);

alert(“Bonjour, “ + name);

} else {

alert(“👋🏼👋🏼👋🏼”);

}

Challenge 🡪

Preguntar la hora al usuario (de 0 a 23). En función de la hora, responder con buenos días , buenas noches…

let currentHour = prompt(“What is the current hour?”);

if (currentHour < 12) {

alert(“Good morning);

} else {

if (currentHour > 18) {

alert(“Good evening);

} else {

alert(“Good afternoon);

}

}

* Hemos animadado un IF ELSE en un ELSE.
* Hay muchas formas de escribir este código. Por ejemplo:

let currentHour = prompt(“What is the current hour?”);

if (currentHour < 12) {

alert(“Good morning);

} else {

if (currentHour < 18) {

alert(“Good afternoon);

} else {

alert(“Good evening);

}

}

* Cuidado con poner el segundo IF fuera del ELSE, porque si lo ponemos como en el segundo caso, al poner una hora menor de 12 y de 18 nos saldrán dos mensajes.

### LESSON 7

#### Operadores lógicos en JavaScript

Vamos a crear operaciones más sofisticadas.

let language = prompt(“Which language do you speak?”);

if (language === ”French”) {

alert(“Bonjour”);

}

A esto le podemos añadir otra condición:

let language = prompt(“Which language do you speak?”);

let continent = prompt(“Which continent are you from?”);

* + Para hacer que ambas sean verdad y así mostrar un mensaje concreto, utilizamos el operador AND 🡪 &&

if (language === ”French” && continent === “Europe”) {

alert(“Bonjour”);

}

También tenemos el operador OR 🡪 ||

let age = prompt(“How old are you?”);

if (age < 16 || > 120) {

alert(“You cannot drive a car”);

}

* Aparecerá el mensaje si una de las dos premisas es verdadera, es decir, que sea menor de 16 años o mayor de 120.

Importante: antes y después de && o de || tenemos que poner la variable a la que estamos haciendo referencia.

En el caso anterior de la edad, podemos poner un ELSE donde muestre otro mensaje en caso de que ambas premisas sean falsas (que tenga entre 16 y 120 años, así podría mostrar un mensaje indicando que sí puede conducir un coche).

Challenge 🡪

Decidir si alguien habla o no portugués:

* Preguntar el país de origen.
* Si no es de Brasil o Portugal, animarle a aprender portugués.
* Hacer que funcione también si el país se escribe mal.

Mi propuesta:

let country = prompt(“What country are you from?”);

if (

country.toLowerCase() === “brazil” ||

country.toLowerCase() === “brasil” ||

country.toLowerCase() === “portugal”

) {

alert(“You speak Portuguese”);

} else {

alert(“Go learn Portuguese!”);

}

Si queremos que el mensaje aparezca con independencia de si el usuario ha utilizado mayúsculas o minúsculas, tenemos que limpiar los datos.

Una opción es convertir el texto en minúsculas. Existe una función que lo hace 🡪 toLowerCase()

Si queremos convertir la variable country en esa variable en minúsculas, tenemos que usar

country = country.toLowerCase();

Así no tenemos que repetirlo cada vez como lo he puesto yo.

Si además queremos quitar los espacios que pueda haber al principio o al final de la palabra, usamos:

country = country.trim();

Otra forma de aceptar Brasil como Brazil es reemplazando una forma por la otra:

country = country.replace(“brasil”, “brazil”);

### LESSON 8

#### Funciones en JavaScript

Existen funciones propias de JavaScript, como prompt, pero también podemos crear nuestras propias funciones.

Las funciones son acciones. Contienen partes de código y sirven para ordenar el código. Son como “botones” que hacen una acción concreta, similar a los botones de una máquina de café. Según el botón, prepara un café u otro.

Podemos crear una función para saludar al usuario:

function greetUser() {

let language = prompt(“Which language do you speak?”);

if (language === “French”) {

alert(“Bonjour”);

} else {

alert(“👋🏼👋🏼👋🏼”);

}

}

Con function damos nombre a la función que vamos a crear. Se suelen utilizar verbos. Si hay varias palabras, se juntan con letra mayúscula al inicio de cada palabra.

Pero el código anterior solo declara la función, la define, pero no la estamos ejecutando. Para ellos tenemos que llamar a la función:

greetUser();

Podemos llamarla las veces que queramos si nuestra intención es que se ejecute varias veces seguidas.

Crear funciones permite ejecutarlas cuando queramos y no directamente al cargar la página.

Además, permite corregir errores o modificar datos solo donde hemos declarado la función.

Las funciones tenemos que definirlas en la parte inicial de <script>.

Challenge 🡪

Preguntar el país de origen, como antes, con Brasil y Portugal, pero dentro de una función.

* Primero creamos la función. En la realidad nosotros tenemos que darle nombre.
  + Por ejemplo, cuando damos like en Instagram estamos ejecutando la función like.
  + Lo bueno de crear funciones es que podemos eliminarlas más fácilmente.
* Tener cuidado de no olvidar cerrar todos los corchetes.

Las variables que hay dentro de una función funcionan cuando llamamos a dicha función.

### LESSON 9

#### Selectores en JavaScript

Muy útiles para crear JavaScript que interactúe con HTML. Por ejemplo:

* En una lista podemos elegir un elemento concreto, incluso mediante la clase que le hayamos dado:

<li class=”free”>

Si abrimos la consola, veremos que hay una variable llamada document que crea automáticamente JavaScript en todas las páginas web y que contiene el documento completo. Así, JavaScript tiene acceso al documento completo.

Si escribimos document.write(“Hello”); toda la página web se sustituye por Hello.

Si queremos seleccionar una parte del documento, como el encabezado, ponemos una función para seleccionarlo:

document.querySelector(“h1”);

* Nos devuelve el encabezado.

Así se puede seleccionar cualquier elemento creado en el HTML, incluso una clase, poniendo un punto delante, como hacemos en CSS:

document.querySelector(“.free”);

* Al utilizar esta función solo nos devuelve el primer elemento que encuentre con esta clase. Si queremos que nos devuelva todos los elementos que haya de ese tipo, tenemos que utilizar otra función:

document.querySelectorAll(“.free”);

Es muy útil porque no permite crear interacciones.

Challenge 🡪

1. Hay que seleccionar el <h2> de la web de SheCodes
2. Decir cuántas estudiantes hay en la lista de la web de estudiantes.

document.querySelector(“h2”);

document.querySelectorAll(“.coder”);

* Para saber qué tenemos que buscar, vamos a Inspeccionar y vemos qué tienen en común todos los perfiles. En este caso, vemos que todos tienen la categoría coder.

### LESSON 10

#### Eventos en JavaScript

Permiten conectar todo lo aprendido hasta ahora. Por ejemplo:

* Creamos un botón que, al clicarlo, nos pida el nombre y el email, y dé una respuesta.

1. Tenemos que seleccionar el botón en el documento.

let contactButton = document.querySelector(“button”);

1. Queremos que al clicar haga algo. Definimos qué tiene que hacer.

let name = prompt(“What is your name?”);

let email = prompt(“What is your email address?”);

alert(“Thank you “ + name + “, we’ll be in touch!”);

1. Como no queremos que esto pase al cargar la página, sino al clicar el botón, metemos las instrucciones en una función:

function contactUs() {

let name = prompt(“What is your name?”);

let email = prompt(“What is your email address?”);

alert(“Thank you “ + name + “, we’ll be in touch!”);

}

1. Tenemos que hacer que contactButton, la variable que contiene el botón (como hemos definido en el primer punto), haga una función al clicar.

contactButton.addEventListener(“click”, contactUs);

* + Variable = contactButton
  + Función = addEventListener
  + Evento = clicar (click)
  + Función que activar al clicar = contactUs

Así, la función contactUs, que activa los popups con las preguntas que tiene que responder el usuario y la alerta final con el mensaje, solo se ejecuta al hacer clic en el botón, no al cargar la página.

También podemos cambiar el contenido del HTML desde JavaScript.

Por ejemplo, en lugar de poner una alerta cuando tenemos la información introducida por el usuario, podemos poner esta información de los popups en el encabezado, en el <h1>, que editamos con nuestro mensaje personalizado.

* Seleccionamos <h1>, creando una variable que lo contenga:

let h1 = document.querySelector(“h1”);

* Le ponemos una propiedad (innerHTML) a esa variable h1 para que dentro del HTML ponga el mensaje que indicamos a continuación:

h1.innerHTML = “Thank you “ + name + “, we’ll be in touch!”);

Con esto podemos editar el HTML desde JavaScript.

El evento que activa el listener puede ser un clic, doble clic, pasar el ratón por encima, mover la barra lateral…

Dentro de JavaScript podemos dar cierto formato, como cursiva, poniendo:

“<em>Thank you</em>”

Challenge 🡪

Crear interacción al clicar el botón y cambiar el <h1> con un mensaje distinto según la edad introducida por el usuario.

* Seleccionar el botón
* Añadir al botón un eventListener
* Definir la función:
  + Crear el prompt para el nombre
  + Crear el prompt para la edad
  + Crear la condición if según la edad con el mensaje correspondiente y el else
* Modificar el <h1> en lugar de crear una alerta
  + Seleccionar el <h1>

### LESSON 11

#### Refactorizar en JavaScript

Hacer el código más legible.

Sirve para evitar errores, hacer que cada función solo tenga una acción. Hacer funciones muy específicas.

Por ejemplo, si partimos de esta función, podemos subdividirla en otras muchas que funcionarán de manera independiente, pero estarán conectadas:

|  |  |
| --- | --- |
|  | function apply() { |
|  | let firstName = prompt(“What is your first name?”); |
| 4 | let age = prompt(“How old are you?”); |
| 3 | let heading = document.querySelector(“h1”); |
| 5 | if (age >= 18) { |
| 1 | heading.innerHTML = “Hi “ + firstName + “, welcome!”); |
|  | } else { |
| 2 | heading.innerHTML = “Sorry, “ + firstName + “, you cannot apply”); |
|  | } |
|  | } |

1 y 2 🡪 heading.innerHTML = “Hi “ + firstName + “, welcome!”); /   
 heading.innerHTML = “Sorry, “ + firstName + “, you cannot apply”);

* Este código con innerHTML se repite. Además podemos concatenar el texto de otra manera más sencilla:

`Hi ${firstName}, welcome!`;

`Sorry, ${firstName}, you cannot apply`;

3 🡪 let heading = document.querySelector(“h1”);

* Creamos una nueva función para actualizar el <h1>

function updateHeading(newHeading) {

let heading = document.querySelector(“h1”);

heading.innerHTML = newHeading;

}

* En los puntos 1 y 2 lo modificamos con la función:

updateHeading(`Hi ${firstName}, welcome!`);

updateHeading(`Sorry, ${firstName}, you cannot apply`);

4 🡪 let age = prompt(“How old are you?”);

* Con la edad decidimos si la persona se puede matricular o no. Creamos una nueva función:

function canApply() {

let age = prompt(“How old are you?”);

if (age >= 18) {

return true;

} else {

return false;

}

}

* O se podría reducir más poniendo:

function canApply() {

let age = prompt(“How old are you?”);

return age >= 18;

}

5 🡪 if (age >= 18) {

* Entonces modificamos el if con la función anterior:

if (canApply()) {

updateHeading(`Hi ${firstName}, welcome!`);

} else {

updateHeading(`Sorry, ${firstName}, you cannot apply`);

}

Así, si falla algo sabemos en qué función concreta está sucediendo y es más fácil resolverlo.

### LESSON 12

#### Resumen semana 2

Hay muchas webs que utilizan JavaScript, como Google.

## **WEEK 3**

## Project overview

Durante la tercera semana se elaborará un proyecto donde se pone en práctica todo lo aprendido en las dos semanas anteriores.

Habrá que crear una web con:

* Encabezados
* Imágenes
* Enlaces
* Párrafos
* Botones

Habrá que añadir estilo con CSS:

* Fondo
* Alineación de encabezados
* Colores
* Bordes
* Padding
* Tamaño de fuente
* Tamaño de imagen
* Interlineado del párrafo
* Dar estilo a las clases
* Estilo de imagen (redondear esquinas)

Y añadir algún comportamiento al botón con JavaScript:

* Añadir un eventListener al botón
* Crear funciones que se activarán al clicar el botón y no al cargar la página

Una vez hecho todo, podemos seguir mejorando la web:

* Mejorar los títulos
* Poner la imagen en un servidor propio para evitar que deje de funcionar si la eliminan
  + Para ello SheCodes nos ofrece un servidor propio: [shecodes.io/uploads](https://www.shecodes.io/uploads)
  + Guardamos la imagen que queremos y la ponemos ahí
  + Copiamos la URL que general el servidor y la utilizamos en nuestra web
* Revisamos colores, formatos de texto
* Usamos paletas de colores o degradados, podemos fijarnos en otras webs que nos gusten…