We/Build

- Bienvenid@s al mundo tech -

Germán Álvarez
Lead Instructor Web Development @ Ironhack Madrid

HTML

La capa de contenido web

Navegar supone visualizar páginas web a través del navegador, diferenciando en estas tres capas independientes:

- **Contenido**, en formato HTML
- **Estilo**, en formato CSS
- **Lógica**, en formato JS

#BASICS

El HTML en una web abarca todos los **contenidos** tanto textuales (títulos, párrafos, listas…) como audiovisuales (imágenes, audios, vídeos, gráficos…)

Estos contenidos se distribuyen en archivos en formato `.html`, un lenguaje que otorga la naturaleza a sus diferentes contenidos según la etiqueta utilizada:

```
<h1>Soy un título de máxima relevancia</h1>
<p>Soy un párrafo</p>
<ul>
   <li>Soy un punto de lista.</li>
   <li>¡Yo otro!</li>
</ul>
```

#HTML

Todos los documentos HTML comparten la misma **estructura base**:

- **`doctype`**: indica el tipo de documento
- **`html`**: abarca la totalidad del código HTML
- **`head`**: abarca datos identificativos y archivos externos
- **`body`**: abarca el contenido visible para el usuario

```
<!DOCTYPE html>
<html>
   <head></head>
   <body></body>
</html>
```

#HTML

Las etiquetas HTML pueden disponer de **atributos**: piezas de información en formato `nombre="valor"`, pudiendo así seleccionarlas desde otros lenguajes como CSS o Javascript.

Los atributos más utilizados para esto son `class` e `id`:

```
<h1 class="red-text">Soy un título de máxima relevancia</h1>
<p id="main">Soy un párrafo</p>
```

#HTML

CSS

La capa de tratamiento estético web

CSS es un **lenguaje de estilizado** que asume el tratamiento estético de las etiquetas HTML aportando aspecto, emoción y carácter a una interfaz.

Se vale de *hojas de estilo* en formato `.css` orientadas a modificar tanto la **estética** como la **distribución espacial** de la web, enlazadas en la etiqueta `<head>` del HTML:

```
<head>
   <title>IronCart | ¿Qué necesitas comprar?</title>
   <link rel="stylesheet" href="css/style.css">
</head>
```

#CSS

Internamente, las hojas de CSS se componen de **reglas de CSS**, donde encontramos tres partes fundamentales:

```
selector {
   propiedad: valor;
   propiedad: valor;
   propiedad: valor;
}
```

El **selector** supone el patrón de alcance de la regla respecto al HTML, mientras que cada **propiedad** manipula un aspecto estético en relación a su **valor**.

#CSS

JS

La capa de lógica web

Javascript es el lenguaje de **programación** que los navegadores web pueden interpretar, dotando a cada página del comportamiento deseado.

Se vale de *scripts* en formato `.js` que a través de diferentes órdenes o *instrucciones* permiten la interacción persona - ordenador.

Estos *scripts* se enlazan en el HTML, generalmente en el `<head>` o al final de la etiqueta `<body>`:

```
<script src="js/script.js"></script>
```

#JAVASCRIPT

Internamente, un *script* está compuesto por un conjunto de elementos de programación entre los que podemos encontrar:

- Variables
- Arrays
- Funciones
- Bucles
- Eventos
- Selectores
- Comentarios

#JAVASCRIPT

CODING TIME

Menos bla bla, vamos a picar

Una **variable** es una unidad de información nominal, donde podemos almacenar un valor...

```
let number = 8;
let name = 'Ironhack';
let boring = false;
```

...para después hacer uso del mismo bajo el nombre con el que ha sido **declarado**:

```
let result = number + 10;
console.log(result);              // 18
```

#VARIABLES

Las variables en Javascript pueden almacenar cualquier tipo de valor:

```
let number = 8                                                  // numérico
let string = 'Ironhack'                                         // texto
let boolean = false                                             // boleano
let array = ['Coca-cola', 'Fanta', 'Agua', 'Cacaolat']          // array o colección
let object = {                                                  // objeto
   name: 'Germán',
   age: 33,
   celiac: false
}
```

#VARIABLES

Los *arrays* o **colecciones** son juegos de valores almacenados en una única variable, entre corchetes...

```
let drinks = ['Coca-cola', 'Fanta', 'Agua', 'Cacaolat']
```

...donde hacemos uso del método `.forEach()` para iterar sobre sus posiciones, obteniendo en el interior del método cada valor almacenado:

```
drinks.forEach(eachDrink => {
    console.log(eachDrink)
})
```

=>

```
Coca-cola
Fanta
Agua
Cacaolat
```

#ARRAYS

Los **objetos** son un conjunto de pares `clave:valor` almacenados en una variable, separados por coma y entre llaves...

```
let person = {
   name: 'Germán',
   age: 33,
   celiac: false
}
```

...donde hacemos uso del nombre de una **clave** tras la notación del punto para acceder a su valor:

```
console.log(person.name)
```

=>

```
Germán
```

#OBJETOS

Integrar dentro de un array una colección de objetos permite crear **datos estructurados**: información en formato computacional fácil de consultar y manipular.

```
let students = [
   { name: 'Marta', age: 33 },
   { name: 'Sergi', age: 28 },
   { name: 'Laura', age: 26 },
   { name: 'María', age: 23 }
]
```

#DATOS_ESTRUCTURADOS

El acceso conjunto a estos datos es posible al combinar la técnica de iteración sobre arrays con la técnica de acceso a las propiedades de un objeto, pudiendo acceder a una de sus propiedades…

```
students.forEach(eachStudent => console.log(eachStudent.name))
```

...o a varias de manera simultánea:

```
students.forEach(eachStudent => {
   console.log(eachStudent.name)
   console.log(eachStudent.age)
})
```

#DATOS_ESTRUCTURADOS

Ya sabes que podemos hacer uso de cualquier *string* (texto), mencionando la variable en la que se encuentra...

```
let presentation = 'El nombre de la Ironhacker es Lucía'
console.log(presentation)   // El nombre de la Ironhacker es Lucía
```

...pero, además, podemos **interpolar** variables dentro de un string si hacemos uso de los *backticks*:

```
let name = 'Lucía'
let age = 33
let presentation = `El nombre de la Ironhacker es ${name}. y tiene ${age} años`
console.log(presentation)  // El nombre de la Ironhacker es Lucía, y tiene 33 años
```

#INTERPOLACIÓN

IronCart

//step1

Transferir los datos estructurados desde el array **`avaibaleFoodsArray`** al panel “Catálogo” de la interfaz:

- Vaciar el HTML demostrativo presente en el panel “Catálogo”, guardando como referencia la estructura de uno de los alimentos.
- Iterar el array **`avaibaleFoodsArray`**
- Replicar en el interior del bucle la estructura HTML de cada alimento, interpolando en el string el valor de cada propiedad del objeto.
- Inyectar en el HTML cada alimento.

La **programación orientada a eventos** permite al usuario interactuar con la aplicación: Javascript puede detectar situaciones provocadas por el usuario (**eventos**) adoptando un comportamientos determinado en respuesta a las mismas.

Para ello necesitamos un **selector**, el **evento** a detectar y la respuesta de la aplicación en forma de **función**:

```
document.querySelector('button').onclick = () => {
    console.log("Evento detectado :3")
}
```

#EVENTOS

En ocasiones necesitamos guardar información en el HTML para accederla desde JS. Para ello disponemos de los atributos **`data-`**

```
<button data-food="0009AL">Añadir</button>
```

Cuya información podemos rescatar desde Javascript mediante la propiedad **`.dataset`** del elemento.

```
let value = element.dataset.food
console.log(value)                    // 0009AL
```

#DATA-ATTRIBUTES

IronCart

//step2

Detectar un click sobre cualquiera de los botones de “Añadir” y obtener el ID del alimento clickado:

- Crear un selector de Javascript que alcance a todos los botones.
- Iterarlo para asociar a cada uno un evento **`.onclick`**, así como una función que permita interactuar con ellos.
- Hacer uso de la propiedad de Javascript **`.dataset`** para obtener el valor del atributo **`data-food`** del botón.

Extraer de un array un elemento en concreto es una situación común en programación que resolvemos a través del método **`.find()`**

```
let student = students.find(eachStudent => eachStudent.name == 'Lucía')
```

#ARRAY-FILTER

IronCart

//step3

Obtener del array **`avaibaleFoodsArray`** el alimento que coincida con el ID almacenado, y transferirlo al panel “Lista de compra”:

- Vaciar el HTML demostrativo presente en el panel “Lista de compra”, guardando como referencia la estructura de uno de los alimentos.
- Almacenar en una variable el alimento seleccionado buscándolo mediante **`.find()`** en el array **`availableFoodsArray`**
- Replicar en el interior del bucle la estructura HTML del alimento, interpolando en el string el valor de cada propiedad del objeto.
- Inyectar en el HTML el alimento.

Combinar operadores aritméticos permite realizar computaciones avanzadas reduciendo el volúmen de código...

```
let age = 33
age++                              // 34

let cost = 120
cost += 40                         // 160
```

...mientras que métodos como `.parseInt()`, `.toFixed()` o `.ceil()` permiten redondear valores numéricos:

```
let price = 33.73638
price.toFixed(2)                // 33.74
```

#OPERATORS

IronCart

//step4

Actualizar el texto informativo de resultados del panel con el total de items y el sumatorio de sus precios:

- Inicializar los valores demostrativos del texto inferior de resultados en el panel, guardando como referencia la estructura del texto.
- Calcular en dos variables el total de alimentos seleccionados, así como el sumatorio del coste de los mismos, respectivamente.
- Replicar el texto de resultados, interpolando en el string el valor resultante almacenado en las variables.
- Inyectar en el HTML el texto.

El método `.setTimeout()` permite ejecutar la función pasada como primer argumento cuando los milisegundos pasados como segundo argumento han transcurrido:

```
setTimeout(function () {
   console.log("Han pasado tres segundos y medio")
}, 3500)
```

#TIMINGS

IronCart

//step5

Actualizar la cantidad de items seleccionados del contador superior del panel:

- Inicializar el valor demostrativo del contador superior de alimentos totales, en el HTML.
- Transferir al contador el total de alimentos previamente acumulado en la variable.
- Incluir en cada cambio una ligera animación con el fin de desencadenar un proceso atencional en el usuario que confirme el éxito de la operación.

El código es el lenguaje de la creatividad contemporánea.

Gracias por vuestra atención

Germán Álvarez
Lead Instructor Web Development @ Ironhack Madrid