Coding time

¿Cuánto puedes aprender en 90 minutos?

Germán Álvarez
Lead Instructor Web Development @ Ironhack

La 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

Javascript

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* definen el *guión* o comportamiento de la interfaz.

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

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

#step1

Acceso a los datos a través de una API

Web APIS

Las APIS son servidores que ofrecen información en forma de **datos estructurados**.

Estos datos son consumidos por otros sistemas (webs o apps, generalmente) para ofrecerla al usuario final tras ser maquetada, tabulada o convertida, por ejemplo, en gráficos interactivos.

API de países

API de productos

API de montañas

#step2

Creación de un chart

Librerías

Las *librerías* son un script reutilizable orientado a cubrir una necesidad de desarrollo específica: videojuegos, realidad virtual, apps móviles, visualización de datos…

Las librerías proporcionan funcionalidades pre configuradas que evitan a los desarrolladores asumir la tarea de aprender y desarrollar los detalles profundos de implementación.

ChartJS

Chart.js es la librería de visualización de datos para el navegador más extendida en la industria. Gratuita y de código abierto, permite la creación de 8 tipos de gráficos.

Instalación - incorporar el script de la librería en el HTML

```
<script src="https://cdn.jsdelivr.net/npm/chart.js@3.7.1/dist/chart.min.js"></script>
```

Implementación - instanciar un new Chart() argumentando los dos datos necesarios:

```
new Chart('canvasID', { type: 'typeName', dataObject, optionsObject })
```

Argumento 1: canvasID

El primer argumento de la instancia es el ID de la etiqueta de HTML <canvas> sobre la que la librería debe renderizar el gráfico.

Si deseamos que el gráfico se renderice en esta etiqueta…

```
<canvas id="myCanvasTag"></canvas>
```

...indicaremos como primer argumento un string con el valor de su ID:

```
new Chart('myCanvasTag', { type: 'typeName', dataObject, optionsObject })
```

Argumento 2: detalles

El segundo argumento es un objeto con tres propiedades:

- **`type`**: string con el tipo de gráfico, pudiendo elegir entre 8 diferentes (bar, polarArea, doughnut, pie, line…).

- **`data`**: objeto con detalles de renderizado (etiquetas, valores y tratamiento estético).

- **`options`**: objeto con opciones de visualización (leyenda, ejes, tooltips…)

#step3

Menos bla bla y más picar

Integración

```
const data = {
   labels: ['Uno', 'Dos', 'Tres', 'Cuatro'],
   datasets: [{
       data: [100, 200, 300, 400],
       label: 'unidades'
   }]
}

const options = {
   plugins: {
       legend: {
           position: 'left'
       }
   }
}

new Chart('whatever', { type: 'bar', data, options })
```

El código es el lenguaje de la creatividad contemporánea.

Gracias por vuestra atención

Germán Álvarez
Lead Instructor Web Development @ Ironhack