



12 DE MAYO DE 2023

HITO INDIVIDUAL - LENGUAJE DE MARCAS

ALVARO BARRENA REVILLA



ÍNDICE

FASE 1 – Calculadora	2
Explicación de la web:	2
Para abordar la funcionalidad de la calculadora con JavaScript:	4
Imágenes de la calculadora integrada en la web:	5
FASE 2 – API REST JSON	6
Explicación de la web	6
Petición de datos de la API REST del tiempo mediante el método “ <i>fetch</i> ”:	6

FASE 1 – Calculadora

Para esta primera fase voy a explicar el HTML y JavaScript que he usado para llevar a cabo la calculadora.

He tenido algunos problemas pero los he acabado solucionando para que la calculadora funcione perfectamente.

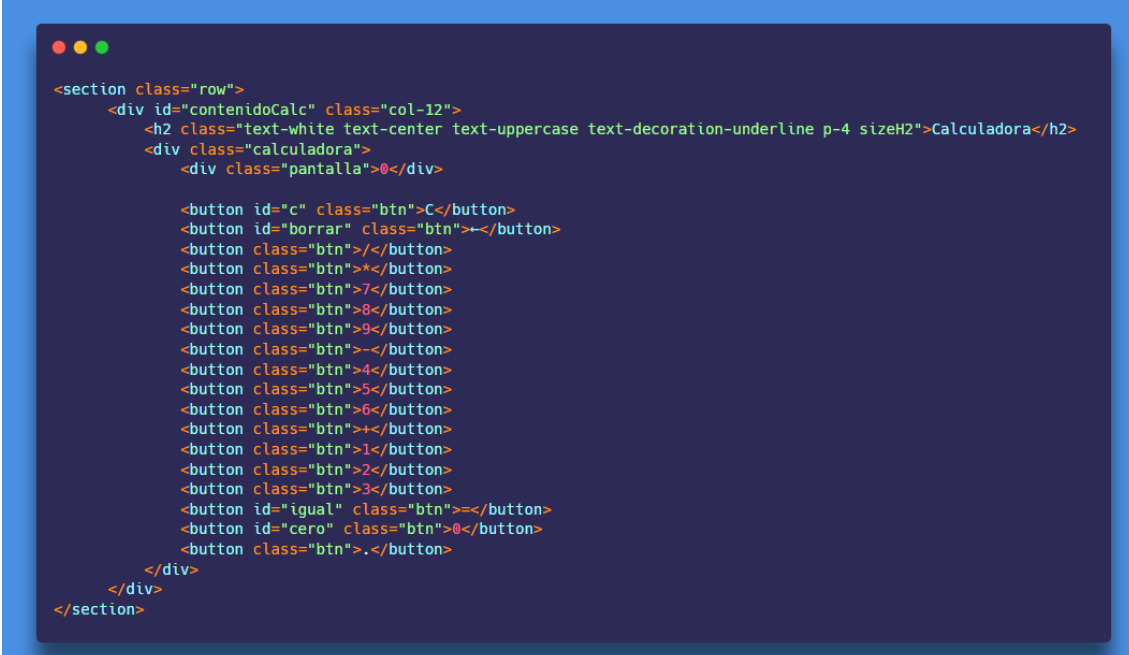
Explicación de la web:

Para empezar, he creado un index.html en el que he hecho la estructura que voy a usar en las dos páginas web también (calculadora y apiRest).

Las tecnologías utilizadas para la web de la calculadora en el HTML han sido:

- Bootstrap: he utilizado esta tecnología para un diseño más profesional, mayor facilidad de poner estilos y facilidad de que la web sea responsive.
- CSS: he utilizado esta tecnología para diseñar mejor la web, ya que había zonas en las que Bootstrap no podía cambiar el diseño y CSS sí.
- HTML: el lenguaje principal de marcado utilizado para estructurar y presentar el contenido en la web.
- JavaScript: utilizado para definir las acciones que se producen cuando se hace clic en los diferentes botones de la calculadora, como mostrar los números y las operaciones en la pantalla y realizar los cálculos correspondientes.

El código HTML para la calculadora lo he englobado en una etiqueta <section> y he ido poniendo todos los botones necesarios.



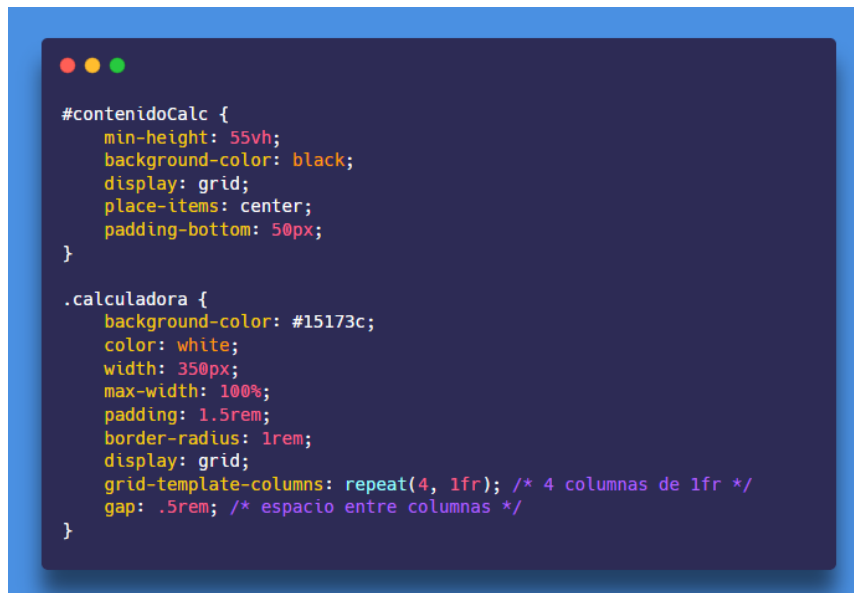
```
<section class="row">
  <div id="contenidoCalc" class="col-12">
    <h2 class="text-white text-center text-uppercase text-decoration-underline p-4 sizeH2">Calculadora</h2>
    <div class="calculadora">
      <div class="pantalla">0</div>

      <button id="c" class="btn">C</button>
      <button id="borrar" class="btn">←</button>
      <button class="btn">+</button>
      <button class="btn">*</button>
      <button class="btn">7</button>
      <button class="btn">8</button>
      <button class="btn">9</button>
      <button class="btn">-</button>
      <button class="btn">4</button>
      <button class="btn">5</button>
      <button class="btn">6</button>
      <button class="btn">+</button>
      <button class="btn">1</button>
      <button class="btn">2</button>
      <button class="btn">3</button>
      <button id="igual" class="btn">=</button>
      <button id="cero" class="btn">0</button>
      <button class="btn">.</button>
    </div>
  </div>
</section>
```

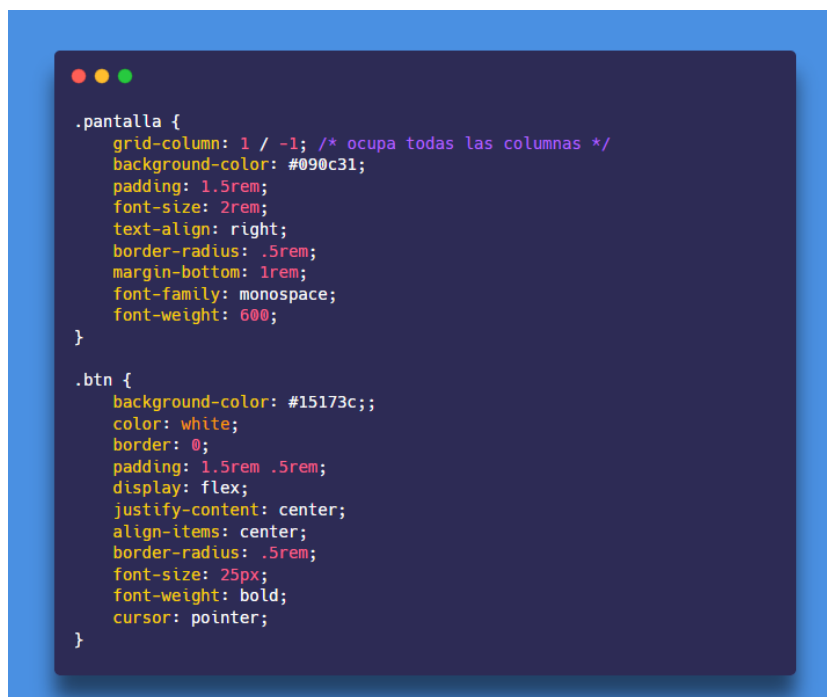
Después de hacer el código HTML, le he dado estilos a la calculadora con CSS para un diseño real de una calculadora.

Se utilizan diferentes selectores para la estructura general y el diseño de la calculadora, la pantalla y los botones de ésta.

Se define un fondo negro para el contenedor principal de la calculadora, se establece el tamaño y la forma de los botones, se especifica la posición de la pantalla de la calculadora y se definen las fuentes y los colores que se utilizan en la interfaz.



Además, se utilizan técnicas de diseño responsive para que la calculadora se adapte a diferentes tamaños de pantalla, como los de los dispositivos móviles. También se definen algunas animaciones y efectos visuales para mejorar la interactividad de la calculadora y proporcionar una mejor experiencia de usuario.



Hasta ahora he explicado el diseño y la creación de la calculadora, ahora explicaré su funcionalidad con JavaScript.

Para abordar la funcionalidad de la calculadora con JavaScript:

1. He utilizado el método ***“querySelector”*** para obtener el elemento que tiene la clase “pantalla”, que es donde se mostrarán los resultados de las operaciones. Después de utiliza el método ***“querySelectorAll”*** para obtener un array de todos los elementos que tiene la clase “btn”, es decir los botones de la calculadora.

```
const pantalla = document.querySelector(".pantalla"); // Obtiene el
elemento con la clase pantalla
const botones = document.querySelectorAll(".btn"); // Obtiene todos los
elementos con la clase btn
```

2. Posteriormente he utilizado el bucle ***“forEach”*** para recorrer todos los elementos del array de botones y le añado un evento “click” que se activará cuando el usuario haga clic en un botón. Aquí tuve problemas porque en el primer “if” al llamar a la clase “c”, la puse en minúsculas en el HTML y en el JS la puse en mayúsculas, por lo que cada vez que pulsaba el botón “C” me pintaba una C en vez de borrar el contenido.

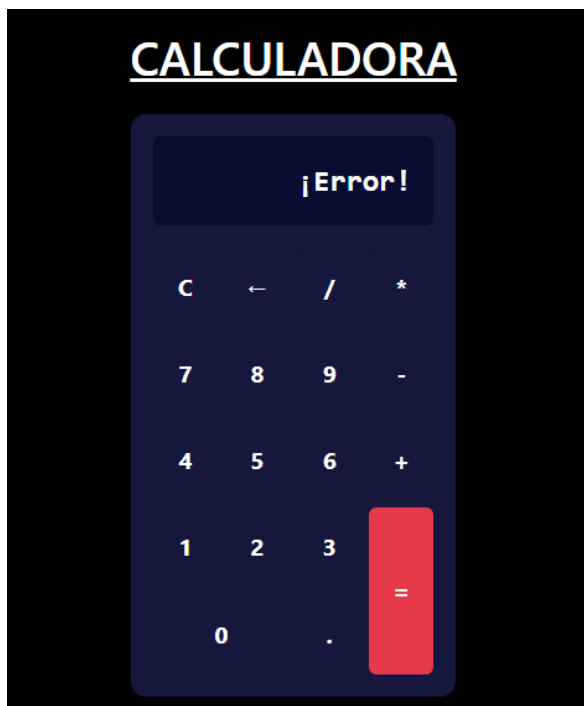
```
botones.forEach(boton => {
  boton.addEventListener("click", () => {
    const botonApretado = boton.textContent; // Obtiene el texto del elemento que se ha apretado

    if (boton.id === "c") {
      pantalla.textContent = "0"; // Si el elemento apretado tiene el id c, se pone la pantalla a 0
      return;
    }
  })
})
```

3. Cada vez que se hace clic en un botón, se ejecuta una función que hace lo siguiente:
 - Obtiene el texto del botón que se ha apretado.
 - Si el botón que se ha apretado tiene el id "c", se borra el contenido de la pantalla y se pone a cero.
 - Si el botón que se ha apretado tiene el id "borrar", se borra el último carácter de la pantalla, a menos que la pantalla tenga un solo carácter o muestre el texto "¡Error!", en cuyo caso se pone la pantalla a cero.
 - Si el botón que se ha apretado tiene el id "igual", se evalúa la expresión matemática que se encuentra en la pantalla y se muestra el resultado. Si ocurre algún error durante la evaluación, se muestra el mensaje "Error!" en la pantalla.

- Si la pantalla muestra el número cero o el mensaje "¡Error!", se reemplaza ese contenido con el texto del botón que se ha apretado. De lo contrario, se añade el texto del botón a la pantalla.

Imágenes de la calculadora integrada en la web:



FASE 2 – API REST JSON

Para esta primera fase voy a explicar el HTML y JavaScript que he usado para llevar a cabo la petición de “*fetch*” de datos a la API Rest de la [web del tiempo](#).

Explicación de la web

El código HTML de esta web también lo he englobado todo en una etiqueta <section> y dentro de ella he creado varios <div>, entre ellos está el contenedor <div> que va a contener todos los datos de la API.

```
<section id="contenidoAPI" class="row">
  <div class="col-12">
    <div class="col-12">
      <h2 class="text-white text-center text-uppercase text-decoration-underline p-4 sizeH2">Api rest JSON</h2>
    </div>
    <div class="col-12">
      <div id="datos-api"></div>
    </div>
  </div>
</section>
```

Petición de datos de la API REST del tiempo mediante el método “*fetch*”:

Para empezar, al cargar la web se llama a la función “*cargarDatos*” utilizando el evento “*load*” del objeto “*window*”.

```
window.addEventListener('load', cargarDatos); // Cuando se cargue la página, llama a la función cargarDatos
```

Cuando se dispara el evento, la función “*cargarDatos*” hace una petición a una API REST mediante el método “*fetch()*”, que retorna una promesa.

Si la promesa se **resuelve correctamente**, la función “*then()*” transforma los datos en formato JSON y los pasa como argumento a la función “*mostrarDatos()*”, que se encarga de mostrar los datos en la web.

Si la promesa es **rechazada**, se muestra un mensaje de error en la consola.

```
function cargarDatos() {  
  fetch('https://www.el-tiempo.net/api/json/v2/home')  
    .then(response => response.json())  
    .then(data => {  
      // Verifica que los datos se hayan recibido correctamente  
      console.log(data);  
  
      // Maneja los datos obtenidos  
      mostrarDatos(data);  
    })  
    .catch(error => console.error(error));  
}
```

Finalmente, la función **“mostrarDatos()”** obtiene el elemento HTML donde se van a mostrar los datos utilizando el método **“getElementById()”**.

Después se crea un elemento **“ul”** y mediante un bucle **“forEach()”**, se crea un elemento **“li”** para cada ciudad que se recupera de la API.

En cada elemento **“li”**, se muestra el nombre de la ciudad y una descripción del estado del cielo.

Por último, se agrega el elemento **“ul”** al contenido de la página mediante el método **“appendChild()”**.

```
function mostrarDatos(data) {  
  // Obtén el elemento donde deseas mostrar los datos  
  const contenido = document.getElementById('datos-api');  
  
  // Crea un elemento ul para mostrar los datos  
  const ul = document.createElement('ul');  
  
  // Recorre los datos y crea un elemento li para cada uno  
  data.ciudades.forEach(ciudad => {  
    const li = document.createElement('li');  
    li.textContent = `${ciudad.name}:  
    ${ciudad.stateSky.description}`;  
    ul.appendChild(li);  
  });  
  
  // Agrega el elemento ul al contenido de la página  
  contenido.appendChild(ul);  
}
```

NOTA: Al cargar la página web de la API, hay que esperar unos segundos para que carguen los datos.

HITO INDIVIDUAL - LENGUAJE DE MARCAS

[Inicio](#) [Calculadora](#) [JSON](#)

API REST JSON

- Barcelona: Intervalos nubosos con tormenta
- Madrid: Poco nuboso
- Sevilla: Despejado
- València: Nuboso
- Bilbao: Nuboso con lluvia escasa
- Coruña, A: Poco nuboso
- Oviedo: Muy nuboso
- Puerto de la Cruz: Poco nuboso
- Eivissa: Intervalos nubosos
- Cáceres: Poco nuboso
- Almería: Intervalos nubosos con lluvia escasa
- Cazorla: Despejado

Desarrollado por Álvaro Barrena Revilla © 2023