HITO 1 DEL 3° TRIMESTRE DE LENGUAJE DE MARCAS

Óscar Durán Sánchez

Fecha de entrega: 30/04/2024
CampusFP

Índice

Fase 1. JavaScript

1. **Borrar el contenido del campo de entrada:**
    - Al hacer clic en el botón "C", se llama a la función **clearDisplay()**.
    - Esta función selecciona el campo de entrada por su ID y establece su valor en una cadena vacía.

```
<button class="btn btn-danger" onclick="clearDisplay()">C</button>
```

```
function clearDisplay() {
  document.getElementById('display').value = '';
}
```

2. **Eliminar el último carácter del campo de entrada:**
    - Al hacer clic en el botón "←", se llama a la función **backspace()**.
    - Esta función selecciona el campo de entrada, obtiene su valor, y luego elimina el último carácter utilizando **slice()**.

```
<button class="btn btn-danger" onclick="backspace()">←</button>
```

```
function backspace() {
  var display = document.getElementById('display');
  display.value = display.value.slice(0, -1);
}
```

3. **Agregar caracteres al campo de entrada:**
    - Al hacer clic en cualquier botón numérico o de operación, se llama a la función **appendChar(char)**.
    - Esta función agrega el carácter correspondiente al final del campo de entrada.

```
<button class="btn btn-secondary" onclick="appendChar('7')">7</button>
```

```
<button class="btn btn-primary" onclick="appendChar('+')">+</button>
```

```
function appendChar(char) {
  document.getElementById('display').value += char;
}
```

4. **Calcular el resultado de la expresión matemática:**
    - Al hacer clic en el botón "=", se llama a la función **calculate()**.
    - Esta función evalúa la expresión matemática en el campo de entrada utilizando **eval()** y muestra el resultado.

```html
<button class="btn btn-success" onclick="calculate()">=</button>
```

```javascript
function calculate() {
  var display = document.getElementById('display');
  try {
    display.value = eval(display.value);
  } catch (error) {
    display.value = 'Error';
  }
}
```

Fase 1. Tecnologías Empleadas

1. **HTML (HyperText Markup Language):**
   Define la estructura de la página web, incluyendo la interfaz de la calculadora con un campo de visualización (**<input>**), botones numéricos y de operaciones matemáticas.
2. **CSS (Cascading Style Sheets):**
   Se utiliza para aplicar estilos visuales a los elementos HTML, como los botones y el campo de visualización. Se hace referencia a una hoja de estilo externa llamada "calculadora.css" para definir la apariencia de la calculadora.
3. **JavaScript:**
    - **clearDisplay()**: Limpia el campo de visualización.
    - **backspace()**: Elimina el último carácter del campo de visualización.
    - **appendChar(char)**: Añade caracteres al campo de visualización cuando se pulsan los botones numéricos y de operaciones.
    - **calculate()**: Evalúa la expresión matemática en el campo de visualización y muestra el resultado.
    - **calculateSquare()**: Calcula el cuadrado del número en el campo de visualización.
    - **calculateSquareRoot()**: Calcula la raíz cuadrada del número en el campo de visualización.

Fase 2. JSON

1. **fetch() para obtener datos de la API:**

```
fetch('https://www.el-tiempo.net/api/json/v2/home')
  .then(response => response.json())
  .then(data => {
```

```
.catch(error => console.error('Error fetching data:', error));
```

- Esta sección utiliza la función fetch() para realizar una solicitud GET a la API de El Tiempo. Una vez que la respuesta es recibida, la primera función .then() convierte la respuesta a formato JSON.

  Luego, otra función .then() toma esos datos JSON y procede a procesarlos. Si ocurre algún error durante el proceso de solicitud o conversión de datos, se captura y se maneja en la función .catch().

2. **Creación de elementos HTML dinámicamente:**

```
data.ciudades.forEach(city => {
```

- Aquí, se itera sobre el array de ciudades obtenido de los datos JSON. Para cada ciudad, se crean elementos HTML dinámicamente utilizando el método document.createElement(). Estos elementos se agregan al documento para mostrar la información del clima de cada ciudad.

3. **Selección de iconos de clima:**

```
img.src = getWeatherIcon(city.stateSky.description);
```

- Esta línea de código asigna la URL de la imagen del icono del clima correspondiente a la descripción del estado del cielo de la ciudad. La función getWeatherIcon() toma la descripción del estado del cielo como argumento y devuelve la URL del icono correspondiente. Esto permite mostrar un icono de clima adecuado para cada ciudad en función de su estado del cielo.

4. **Función getWeatherIcon():**

```
function getWeatherIcon(description) {
  switch (description) {
    case 'Despejado':
      return 'https://www.el-tiempo.net/assets/images/estados/despejado.png';
    case 'Cubierto con lluvia':
      return 'https://www.el-tiempo.net/assets/images/estados/cubierto_con_lluvia.png';
    default:
      return 'https://www.el-tiempo.net/assets/images/estados/na.png';
  }
}
```

   - Esta función toma la descripción del estado del cielo como argumento y devuelve la URL del icono del clima correspondiente. Actualmente, solo maneja dos casos ('Despejado' y 'Cubierto con lluvia'), pero puede ampliarse según sea necesario.

Fase 2. Tecnologías Empleadas

1. **HTML (HyperText Markup Language):**
   - HTML se utiliza para estructurar el contenido de la página web. En este caso, el código HTML define la estructura básica de la página, incluyendo las etiquetas **<html>**, **<head>**, **<body>**, y elementos como ** **, **<h1>**, **<img>**, **<p>**, etc.
2. **CSS (Cascading Style Sheets):**
   - El archivo **api.css** contiene estilos CSS que se aplican a los elementos HTML para controlar su presentación en la página web. En este caso, el CSS probablemente define estilos para los elementos de la clase **.container**, **.city**, **.weather**, y otros que se usan en el código HTML.
3. **JavaScript:**
   - JavaScript se utiliza para agregar interactividad a la página web. En este código, JavaScript se utiliza para:
     - Hacer una solicitud (**fetch**) a la API de El Tiempo para obtener datos meteorológicos.
     - Procesar los datos recibidos y actualizar dinámicamente la página con la información meteorológica.
     - Manipular el DOM (Document Object Model) para crear elementos HTML y mostrar la información obtenida de la API.
     - Definir la función **getWeatherIcon** que devuelve la URL de un icono meteorológico basado en la descripción del estado del tiempo.