Calculadora

Primero haremos el html de nuestra calculadora, para este caso en especifico utilizaremos una etiqueta en nuestro script la cual será la etiqueta "**defer**" en HTML se utiliza en el elemento "script" para indicar que el script que se está cargando no debe detener el análisis del documento HTML mientras se carga.

Cuando se utiliza la etiqueta "defer", el navegador continuará analizando el resto del documento HTML mientras el script se carga en segundo plano. Luego, cuando el análisis del documento HTML esté completo, el script se ejecutará. Esto ayuda a mejorar el rendimiento del sitio web al permitir que el contenido visible se muestre más rápidamente al usuario.


```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Calculadora</title>
  <link href="css/styles.css" rel="stylesheet">
  <script src="js/script.js" defer></script>
</head>
<body>
  <div class="calculator-grid">
    <div class="output">
      <div data-previous-operand class="previous-operand"></div>
      <div data-current-operand class="current-operand"></div>
    </div>
    <button data-all-clear class="span-two">Limpiar</button>
    <button data-delete>Borrar</button>
    <button data-operation>÷</button>
    <button data-number>1</button>
    <button data-number>2</button>
    <button data-number>3</button>
    <button data-operation>*</button>
    <button data-number>4</button>
    <button data-number>5</button>
    <button data-number>6</button>
    <button data-operation>+</button>
    <button data-number>7</button>
    <button data-number>8</button>
    <button data-number>9</button>
    <button data-operation>-</button>
    <button data-number>.</button>
    <button data-number>0</button>
    <button data-equals class="span-two">=</button>
  </div>
</body>
</html>
```

A continuación, aplicaremos el CSS a nuestra calculadora:

```css
*, *::before, *::after {
  box-sizing: border-box;
  font-family: Gotham Rounded, sans-serif;
  font-weight: normal;
}

body {
  padding: 0;
  margin: 0;
  background: #d2e2eb;
}

.calculator-grid {
  display: grid;
  justify-content: center;
  align-content: center;
  min-height: 100vh;
  grid-template-columns: repeat(4, 100px);
  grid-template-rows: minmax(120px, auto) repeat(5, 100px);
}

.calculator-grid > button {
  cursor: pointer;
  font-size: 2rem;
  border: 1px solid white;
  outline: none;
  background-color: rgba(255, 255, 255, .75);
}
```

La primera parte del código es un selector universal ***** que selecciona todos los elementos de la página, así como los pseudo-elementos **::before** y **::after**. Las propiedades que se establecen para estos elementos son:

- **box-sizing: border-box;** que hace que el ancho y alto de los elementos incluyan el borde y el padding.
- **font-family: Gotham Rounded, sans-serif;** establece la fuente del texto del elemento.
- **font-weight: normal;** establece el grosor de la fuente a normal.

Luego, el selector **body** se utiliza para establecer algunas propiedades para el cuerpo del documento:

- **padding: 0;** elimina cualquier margen interno que pueda haber en el cuerpo.
- **margin: 0;** elimina cualquier margen externo que pueda haber en el cuerpo.
- **background: #d2e2eb;** establece un color de fondo para el cuerpo.

A continuación, se define una clase **.calculator-grid** que se utiliza para dar estilo a un contenedor que contiene una calculadora en la página:

- **display: grid;** establece el elemento como una cuadrícula.
- **justify-content: center;** centra los elementos horizontalmente en el contenedor.
- **align-content: center;** centra los elementos verticalmente en el contenedor.
- **min-height: 100vh;** establece una altura mínima del contenedor igual a la altura de la ventana gráfica del navegador.
- **grid-template-columns: repeat(4, 100px);** establece cuatro columnas en la cuadrícula, cada una con un ancho de 100 píxeles.
- **grid-template-rows: minmax(120px, auto) repeat(5, 100px);** establece una fila de altura mínima de 120 píxeles y las filas restantes con una altura fija de 100 píxeles.

A continuación, se definen algunas reglas para los botones dentro de la cuadrícula **.calculator-grid > button**:

- **cursor: pointer;** cambia el cursor del mouse a una mano cuando se pasa sobre los botones.
- **font-size: 2rem;** establece el tamaño de fuente de los botones en 2 rem (unidades relativas al tamaño de la fuente de la página).
- **border: 1px solid white;** establece un borde de 1 píxel de ancho sólido y blanco alrededor de los botones.
- **outline: none;** elimina el contorno predeterminado que aparece en los botones cuando se enfocan.
- **background-color: rgba(255, 255, 255, .75);** establece un color de fondo semi-transparente para los botones.

Por último, hay un selector de clase **.span-two** que se utiliza para hacer que un elemento ocupe dos columnas de la cuadrícula. Esto se hace mediante la propiedad **grid-column: span 2;** que establece el elemento para ocupar dos columnas.

A continuación el resto del código:

```css
.calculator-grid > button:hover {
  background-color: rgba(255, 255, 255, .9);
}

.span-two {
  grid-column: span 2;
}

.output {
  grid-column: 1 / -1;
  background-color: rgba(0, 0, 0, .75);
  display: flex;
  align-items: flex-end;
  justify-content: space-around;
  flex-direction: column;
  padding: 10px;
  word-wrap: break-word;
  word-break: break-all;
}

.output .previous-operand {
  color: rgba(255, 255, 255, .75);
  font-size: 1.5rem;
}

.output .current-operand {
  color: white;
  font-size: 2.5rem;
}
```

La regla **.calculator-grid > button:hover** se aplica a los botones hijos directos de la clase **.calculator-grid** y cambia el color de fondo del botón cuando se coloca el cursor sobre él.

La clase **.span-two** cambia el número de columnas que ocupa un elemento en la cuadrícula de la calculadora.

La clase **.output** se aplica al contenedor de la pantalla de la calculadora y establece el color de fondo, la dirección del contenido y la forma en que se envuelve y rompe el texto.

Las clases **.output .previous-operand** y **.output .current-operand** definen el estilo para los elementos que muestran el operando anterior y el operando actual en la pantalla de la calculadora. Establecen el tamaño de fuente y el color del texto para cada elemento.

Ahora haremos el JavaScript:

```
class Calculator {
    constructor(previousOperandTextElement, currentOperandTextElement) {
      this.previousOperandTextElement = previousOperandTextElement
      this.currentOperandTextElement = currentOperandTextElement
      this.clear()
    }

    clear() {
      this.currentOperand = ''
      this.previousOperand = ''
      this.operation = undefined
    }

    delete() {
      this.currentOperand = this.currentOperand.toString().slice(0, -1)
    }

    appendNumber(number) {
      if (number === '.' && this.currentOperand.includes('.')) return
      this.currentOperand = this.currentOperand.toString() + number.toString()
    }

    chooseOperation(operation) {
      if (this.currentOperand === '') return
      if (this.previousOperand !== '') {
        this.compute()
      }
      this.operation = operation
      this.previousOperand = this.currentOperand
      this.currentOperand = ''
    }
```

```
compute() {
  let computation
  const prev = parseFloat(this.previousOperand)
  const current = parseFloat(this.currentOperand)
  if (isNaN(prev) || isNaN(current)) return
  switch (this.operation) {
    case '+':
      computation = prev + current
      break
    case '-':
      computation = prev - current
      break
    case '*':
      computation = prev * current
      break
    case '÷':
      computation = prev / current
      break
    default:
      return
  }
  this.currentOperand = computation
  this.operation = undefined
  this.previousOperand = ''
}
```

```
getDisplayNumber(number) {
  const stringNumber = number.toString()
  const integerDigits = parseFloat(stringNumber.split('.')[0])
  const decimalDigits = stringNumber.split('.')[1]
  let integerDisplay
  if (isNaN(integerDigits)) {
    integerDisplay = ''
  } else {
    integerDisplay = integerDigits.toLocaleString('en', { maximumFractionDigits: 0 })
  }
  if (decimalDigits != null) {
    return `${integerDisplay}.${decimalDigits}`
  } else {
    return integerDisplay
  }
}
```

@hdtoledo

```
  updateDisplay() {
    this.currentOperandTextElement.innerText =
      this.getDisplayNumber(this.currentOperand)
    if (this.operation != null) {
      this.previousOperandTextElement.innerText =
        `${this.getDisplayNumber(this.previousOperand)} ${this.operation}`
    } else {
      this.previousOperandTextElement.innerText = ''
    }
  }
}
```

Este código define la clase **Calculator**, que tiene varios métodos para realizar operaciones matemáticas en una calculadora. El constructor de la clase toma dos argumentos: **previousOperandTextElement** y **currentOperandTextElement**, que son los elementos HTML donde se mostrarán los operandos y el resultado de la operación actual.

La clase tiene los siguientes métodos:

- **clear()**: establece el valor actual y anterior en una cadena vacía y la operación actual en **undefined**.
- **delete()**: elimina el último carácter del valor actual.
- **appendNumber(number)**: agrega un número a la cadena del valor actual. Si el número ya contiene un punto decimal, no se agrega otro.
- **chooseOperation(operation)**: establece la operación actual, si se ha ingresado un valor en el campo actual. Si ya hay un valor en el campo anterior, la función **compute()** se llama para calcular el resultado de la operación anterior.
- **compute()**: realiza la operación anterior en el campo anterior y el campo actual y establece el campo actual en el resultado de la operación. La operación actual se establece en **undefined** y el campo anterior se establece en una cadena vacía.
- **getDisplayNumber(number)**: convierte el número en una cadena formateada para su visualización en la calculadora.
- **updateDisplay()**: actualiza los elementos HTML de los operandos y el resultado de la operación actual.

En resumen, la clase **Calculator** define una calculadora que puede realizar operaciones básicas de suma, resta, multiplicación y división y mostrar el resultado en un formato fácil de leer.

```
const numberButtons = document.querySelectorAll('[data-number]')
const operationButtons = document.querySelectorAll('[data-operation]')
const equalsButton = document.querySelector('[data-equals]')
const deleteButton = document.querySelector('[data-delete]')
const allClearButton = document.querySelector('[data-all-clear]')
const previousOperandTextElement = document.querySelector('[data-previous-operand]')
const currentOperandTextElement = document.querySelector('[data-current-operand]')

const calculator = new Calculator(previousOperandTextElement, currentOperandTextElement)

numberButtons.forEach(button => {
  button.addEventListener('click', () => {
    calculator.appendNumber(button.innerText)
    calculator.updateDisplay()
  })
})

operationButtons.forEach(button => {
  button.addEventListener('click', () => {
    calculator.chooseOperation(button.innerText)
    calculator.updateDisplay()
  })
})

equalsButton.addEventListener('click', button => {
  calculator.compute()
  calculator.updateDisplay()
})

allClearButton.addEventListener('click', button => {
  calculator.clear()
  calculator.updateDisplay()
})

deleteButton.addEventListener('click', button => {
  calculator.delete()
  calculator.updateDisplay()
})
```

Este código se encarga de conectar la calculadora visual con la lógica que hemos implementado en la clase **Calculator**.

Primero, el código crea variables para acceder a los elementos de la calculadora visual. Cada botón numérico y de operación es seleccionado con el método **querySelectorAll** y se almacenan en variables para poder asociarles un evento.

Luego, se crea una instancia de la clase **Calculator** con los elementos de texto para el operando actual y previo como parámetros.

A continuación, se agregan escuchadores de evento a cada botón numérico y de operación. Cuando se hace clic en un botón numérico, se llama al método **appendNumber** de la instancia de

la calculadora, que agrega el número al operando actual y luego se llama al método **updateDisplay** para actualizar la pantalla.

Cuando se hace clic en un botón de operación, se llama al método **chooseOperation** de la instancia de la calculadora, que establece la operación seleccionada, guarda el operando actual en el operando previo y luego establece el operando actual en una cadena vacía. Luego se llama al método **updateDisplay** para actualizar la pantalla.

Cuando se hace clic en el botón de igual, se llama al método **compute** de la instancia de la calculadora, que realiza la operación seleccionada en los dos operandos y establece el resultado como el operando actual. Luego se llama al método **updateDisplay** para actualizar la pantalla.

Cuando se hace clic en el botón de todo borrar, se llama al método **clear** de la instancia de la calculadora, que establece los operandos en cadenas vacías y la operación en **undefined**. Luego se llama al método **updateDisplay** para actualizar la pantalla.

Finalmente, cuando se hace clic en el botón de borrar, se llama al método **delete** de la instancia de la calculadora, que borra el último carácter del operando actual. Luego se llama al método **updateDisplay** para actualizar la pantalla.