CALCULATOR

Armando Vieira y Hugo Durán

Explain what lessons you’ve learned during this project

Uso de las clases en Javascript (class) con un método constructor para crear e inicializar el objeto.

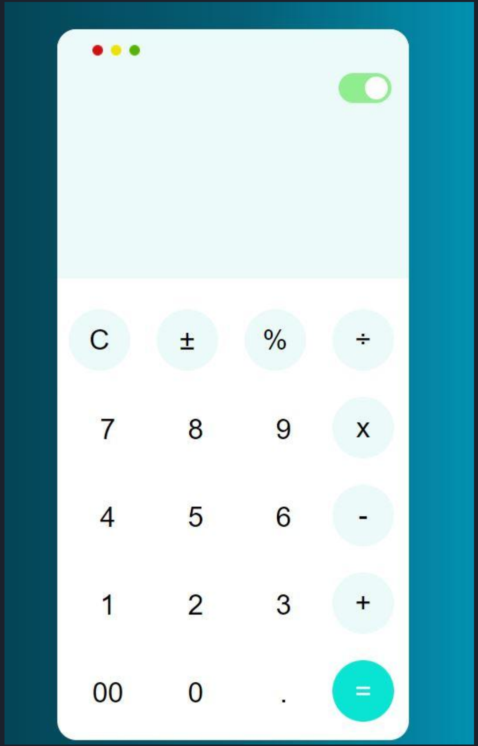
Uso de “this” para invocar las funciones dentro de la class creada anteriormente.

Uso de “toggle” para añadir o quitar clases de css haciendo referencia al dark mode.

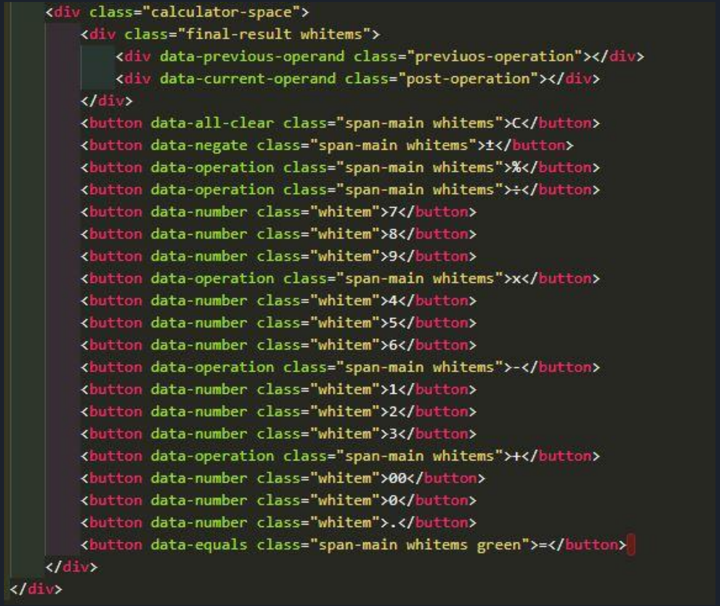
Uso de word-wrap y word-break para hacer un salto de línea y ampliar la caja para los dígitos de las operaciones de la calculadora

How have you decided to organize the three stages of the app that you had to develop?

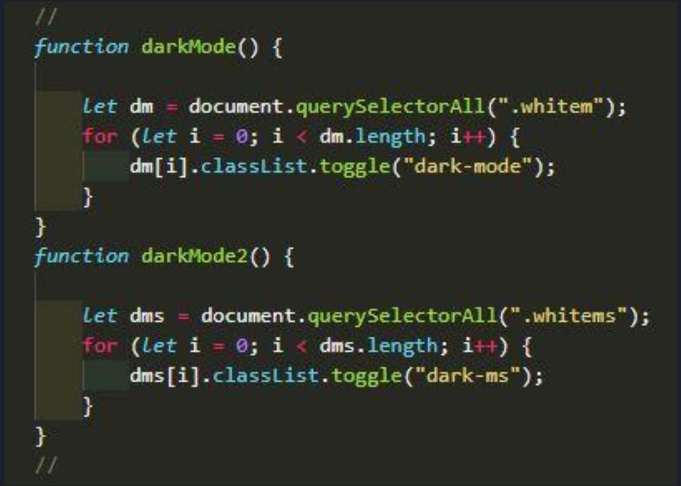
- Pensar el diseño de la calculadora
- Realizarlo (diseño light primero)



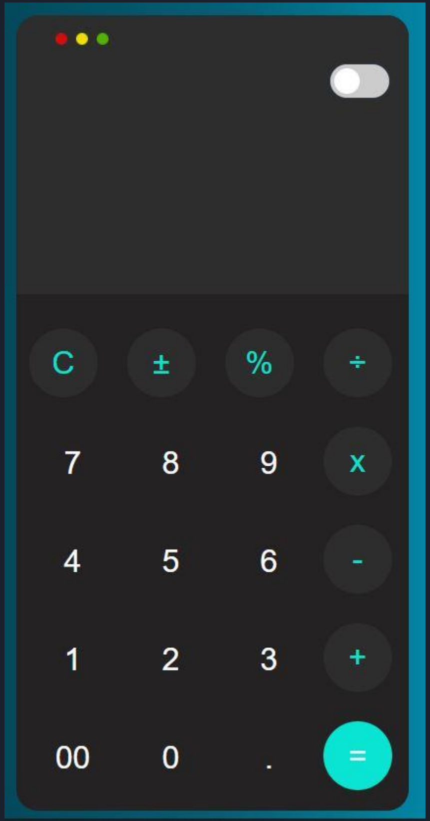
- Añadir las clases para el dark
- Dark mode en js



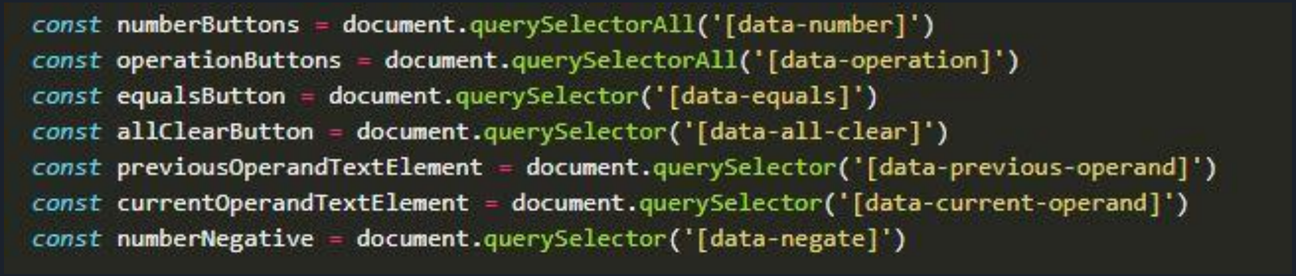
```
    <div class="calculator-space">
        <div class="final-result whitems">
            <div data-previous-operand class="previuos-operation"></div>
            <div data-current-operand class="post-operation"></div>
        </div>
        <button data-all-clear class="span-main whitems">C</button>
        <button data-negate class="span-main whitems">±</button>
        <button data-operation class="span-main whitems">%</button>
        <button data-operation class="span-main whitems">÷</button>
        <button data-number class="whitem">7</button>
        <button data-number class="whitem">8</button>
        <button data-number class="whitem">9</button>
        <button data-operation class="span-main whitems">x</button>
        <button data-number class="whitem">4</button>
        <button data-number class="whitem">5</button>
        <button data-number class="whitem">6</button>
        <button data-operation class="span-main whitems">-</button>
        <button data-number class="whitem">1</button>
        <button data-number class="whitem">2</button>
        <button data-number class="whitem">3</button>
        <button data-operation class="span-main whitems">+</button>
        <button data-number class="whitem">00</button>
        <button data-number class="whitem">0</button>
        <button data-number class="whitem">.</button>
        <button data-equals class="span-main whitems green">=</button>
    </div>
</div>
```



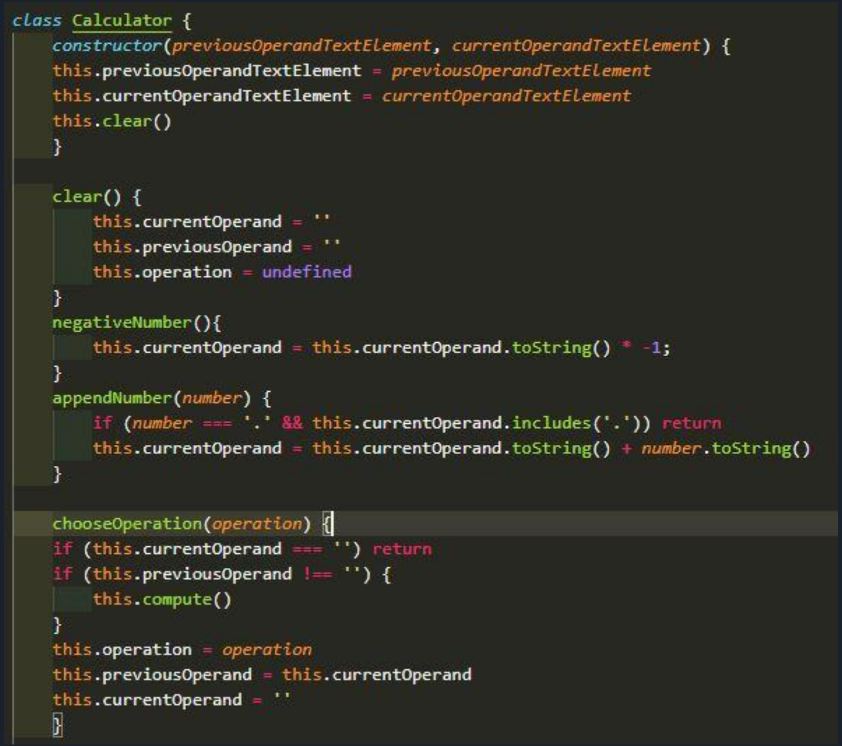
```
//
function darkMode() {

    let dm = document.querySelectorAll(".whitem");
    for (let i = 0; i < dm.length; i++) {
        dm[i].classList.toggle("dark-mode");
    }
}
function darkMode2() {

    let dms = document.querySelectorAll(".whitems");
    for (let i = 0; i < dms.length; i++) {
        dms[i].classList.toggle("dark-ms");
    }
}
//
```



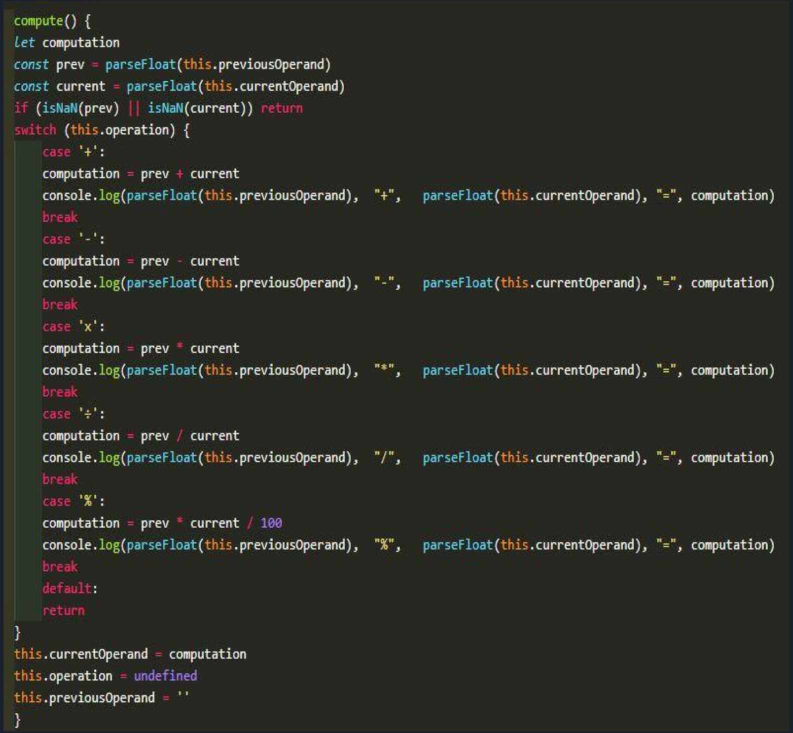
- Declaración de las constantes con queryselector


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

- Funciones de las “operaciones “(clear, choose, append number, compute)
- Operaciones matemáticas

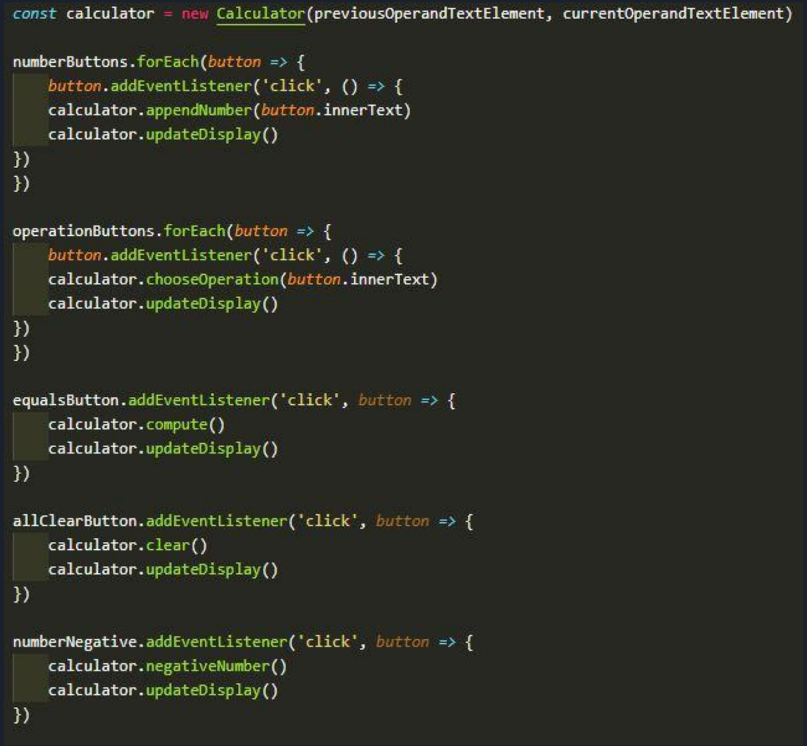


```
class Calculator {
    constructor(previousOperandTextElement, currentOperandTextElement) {
    this.previousOperandTextElement = previousOperandTextElement
    this.currentOperandTextElement = currentOperandTextElement
    this.clear()
    }

    clear() {
        this.currentOperand = ''
        this.previousOperand = ''
        this.operation = undefined
    }
    negativeNumber(){
        this.currentOperand = this.currentOperand.toString() * -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 = ''
    }
```

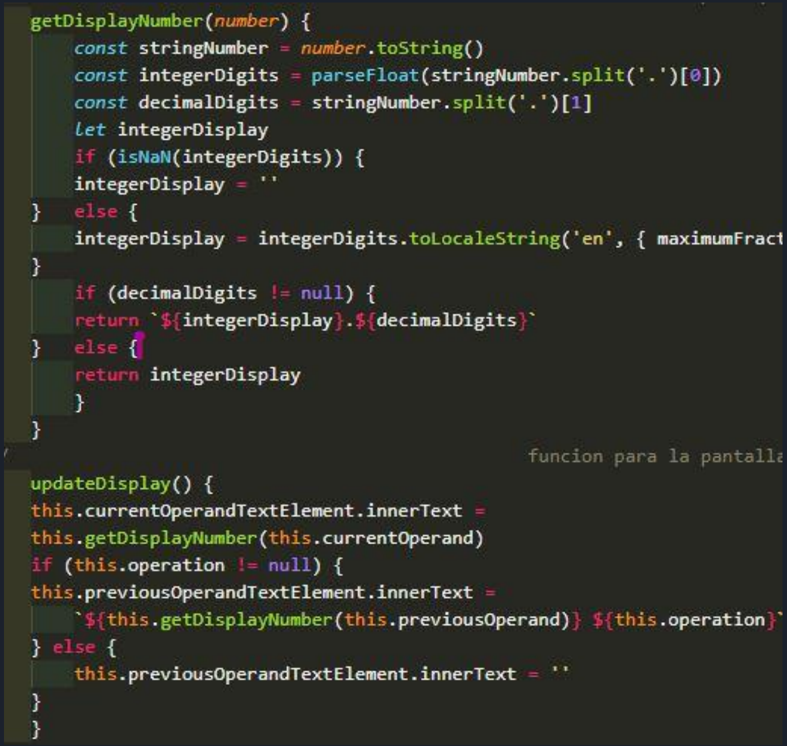


```
compute() {
let computation
const prev = parseFloat(this.previousOperand)
const current = parseFloat(this.currentOperand)
if (isNaN(prev) || isNaN(current)) return
switch (this.operation) {
    case '+':
    computation = prev + current
    console.log(parseFloat(this.previousOperand),  "+",   parseFloat(this.currentOperand), "=", computation)
    break
    case '-':
    computation = prev - current
    console.log(parseFloat(this.previousOperand),  "-",   parseFloat(this.currentOperand), "=", computation)
    break
    case 'x':
    computation = prev * current
    console.log(parseFloat(this.previousOperand),  "*",   parseFloat(this.currentOperand), "=", computation)
    break
    case '÷':
    computation = prev / current
    console.log(parseFloat(this.previousOperand),  "/",   parseFloat(this.currentOperand), "=", computation)
    break
    case '%':
    computation = prev * current / 100
    console.log(parseFloat(this.previousOperand),  "%",   parseFloat(this.currentOperand), "=", computation)
    break
    default:
    return
}
this.currentOperand = computation
this.operation = undefined
this.previousOperand = ''
}
```

- Funciones para los botones
- Función para que nos muestre el resultado final



```
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()
})

numberNegative.addEventListener('click', button => {
    calculator.negativeNumber()
    calculator.updateDisplay()
})
```



```
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', { maximumFract
}
    if (decimalDigits != null) {
    return `${integerDisplay}.${decimalDigits}`
}   else {
    return integerDisplay
    }
}
                                    funcion para la pantalla
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 = ''
}
}
```