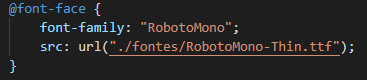
* Calculadora de base utilizada foi a do Mac
* **Características de Calculator.css**:
  + Height (altura) – 320px
  + Width (largura) – 235px
  + Border-radius (borda da calculadora) – 5px
  + Overflow hidden (tudo que estiver fora do componente, como um texto sobressalente, ele vai esconder)
  + Display: grid (colocar o visual como uma grid, inicialmente ele vai ficar um embaixo do outro, para modificarmos isso, utilizamos outras opções junto, como a seguir)
  + Grid-template-colums: repeat(4,25%) (dividir a grid em 4 colunas e cada uma dividindo 25% do espaço)
  + Grid-template-rows: 1f 48px 48px 48px 48px 48px (definindo o tamanho para cada uma das linhas da grid. O 1f é quando você quer utilizar todo o espaço restante)
* **Características de index.css**:
  + Definir a fonte padrão do script:



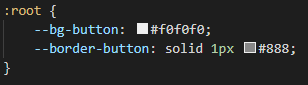


Em @font-face importamos de um arquivo interno a fonte, e depois em \*{font-family...} falamos para usar a RobotoMono, e caso não for encontrado do arquivo, use a monospace

* Display: flex
* Height(altura) – 100vh (ocupar toda a tela)
* Justify-content – center ( centralizar na horizontal da página os componentes)
* Align-items – center (centralizar na vertical da páginas os componente)
* Text-align: center (Quando é inserido um texto, o texto estará centralizado também)
* Background:

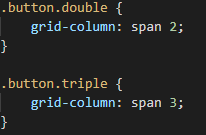


* **Características Button.css:**
  + Iremos criar variáveis de estilo no css:

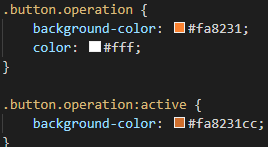


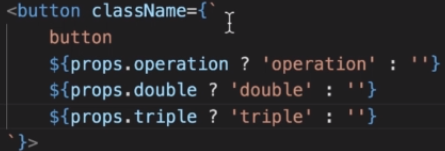
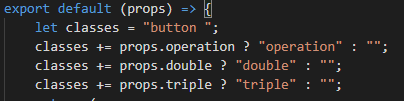
Onde o –bg-button é o nome da variáveis, e em seguida, o que contém

* + Font-size - 1.4em
  + Background-color - var(--bg-button) (vai chamar a variável criada anteriormente)
  + Border – none
  + Border-right: variável border-button
  + Border-bottom: variável border-button
  + Outline: none (Para sumir aquela marcação azul quando clicamos com o mouse)
  + Foi criado um .button:active{background-color: #ccc} para definir o estilo do botão depois que clicamos, no caso, o botão ficará mais escuro
  + Iremos criar o estilo para a classe double e triple para definir o span dos botões (quantas colunas o botão ocupará)



* + Iremos criar um estilo para a classe .operations, que são os botões de operação de cor alaranjada, e iremos definir a mudança de cor assim que clicamos no botão



* **Característica Display.css:**
  + Grid-colum - span 4 (você define o número de colunas que o componente vai requerir)
  + Background-color - #0004
  + Display – flex
  + Justify-context – flex-end (alinhar o conteúdo dentro de display para a direita)
  + Align-items – center (centralizar o conteúdo verticalmente)
  + Padding – 20px (distanciar o conteúdo das bordas)
  + Font-size: 2.2 (definir o tamanho da fonte)
  + Overflow: hidden (só para reforçar o hidden do Calculator.css)
* Quanto menor o número de states em uma aplicação, melhor, se for possível criar um export default props => para exportar os componente.
* No componente Button.jsx iremos aplicar o conceito de denominação de classe por condição, tem dois jeitos de fazermos isso:
  + 
  + 
* O evento de clique em Button.jsx irá retornar o valor presente no componente:
  + 

O clique vai disparar uma arrow function, que receberá o evento (como um parâmetro, por exemplo).

O && irá garantir que ele só irá realizar a função caso tenha sido passado a função em props.

Arrow function irá chamar a função click() passada por props, e passará como parâmetro o conteúdo do componente (props.label)

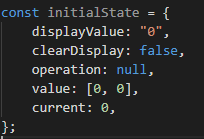
* Para adicionarmos o dígito, iremos criar uma função addDigit fora de render() e instanciá-la dentro de render()



Estamos criando uma arrow function chamada addDigit, passando o parâmetro (n) e referindo ela a função criada exteriormente a render()

Para garantirmos que o this passado para o componente será o mesmo this de dentro da classe, fazemos este novo const

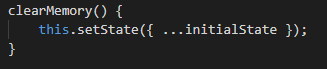
* Iremos definir as opções iniciais da calculadora fora da classe:



E iremos copiá-la dentro do state da classe:



* + - displayValue – valor inicial do display
    - clearDisplay – Se ele irá limpar o display
    - operation – Operação que está sendo utilizada
    - value – os dois valores a serem utilizados
    - current – Qual o índice que estou trabalho em value (0 ou 1)
* **Função clearMemory**:
  + - Caso seja apertado AC, irá restaurar o state para o estado inicial:



* **Função addDigit:**
  + - Caso seja apertado “.” e já existir ponto no valor, ele não irá retornar nada



* + - Se o número no display for “0”, e você digitar um número diferente de 0, ele vai apagar o “0” e digitar o novo número. A mesma coisa se a variável clearDisplay em state estiver em true



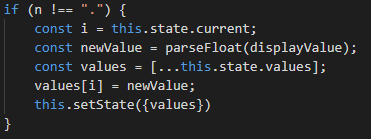
* + - Para pegarmos o valor já presente no display, iremos utilizar a variável de state clearDisplay, se ela for true, iremos pegar um valor nulo, caso não, iremos pegar pegar o valor em displayValue



* + - Uma vez que inserimos o valor no display, podemos passar para false a flag clearDisplay



* + - Qualquer valor diferente de ponto será armazenado na variável values de state (no índice current)



* **Função setOperation:**
  + - Caso estejamos passando o primeiro valor da conta, o state irá receber a operação clicada e passará current para 1 (falando que o número a ser inserido, na próxima operação é o segundo número. E iremos setar clearDisplay para 1.

this.setState({ operation, current: 1, clearDisplay: true });

* + - Caso apertamos alguma operação e já está no segundo índice, ele irá verificar se a operação é a de “igual”

const equals = operation === "="

* + - Ao selecionarmos a operação, o resultado será armazenado no primeiro índice de values (em state), e iremos zerar o segundo índice:

values[0] = eval(`${values[0]} ${currentOperation} ${values[1]}`);

            values[1] = 0;

O eval é uma função para executarmos um código presente em uma string

* + - Ao final, iremos salvar o novo estado. Iremos colocar no display o valor após a operação (e armazenado no primeiro índice de values em state). Iremos guardar a operação caso ela seja diferente de equals. Caso tenhamos apertado o equals, ele irá “zerar” a calculadora e atribuir que você não vai continuar a operação com o número antigo.Você irá setar a opção clearDisplay e passar o novo values