MÓDULO II

Lógica de Programação utilizando a Linguagem JavaScript

A Educação é o primeiro passo para um futuro melhor...

MÓDULO II

6 projetos de JavaScript para iniciantes!

1. Hello World

Para iniciarmos nossa jornada pelo mundo dos projetos em JavaScript, vamos começar por um bem simples: **exibir um "Olá, mundo!" na tela**.

Vamos à prática

Como se trata de um projeto simples, vamos precisar apenas de **um arquivo HTML**.

Neste arquivo, vamos criar o **código JavaScript diretamente na tag *script***, dentro do elemento ***body*** da página. A tag ***body*** indica para a página onde estão os elementos principais que devem ser exibidos, já a tag ***script*** é responsável por compreender o **JavaScript** adicionado ao **HTML**.

MÓDULO II

Optamos por essa estratégia por se tratar de um código simples e que não necessitaria de muitas linhas. Em projetos mais complexos, como os que vamos fazer adiante, é aconselhável separar o código **JavaScript** em um outro arquivo, para que fique mais organizado.

Seguindo com o projeto, **dentro da tag *script* vamos adicionar um *alert***. O ***alert*** é uma função **JavaScript** que dispara um modal com uma mensagem quando a página for carregada. **Dentro do *alert* vamos informar a mensagem que desejamos** que seja exibida, o "**Olá, mundo!**".

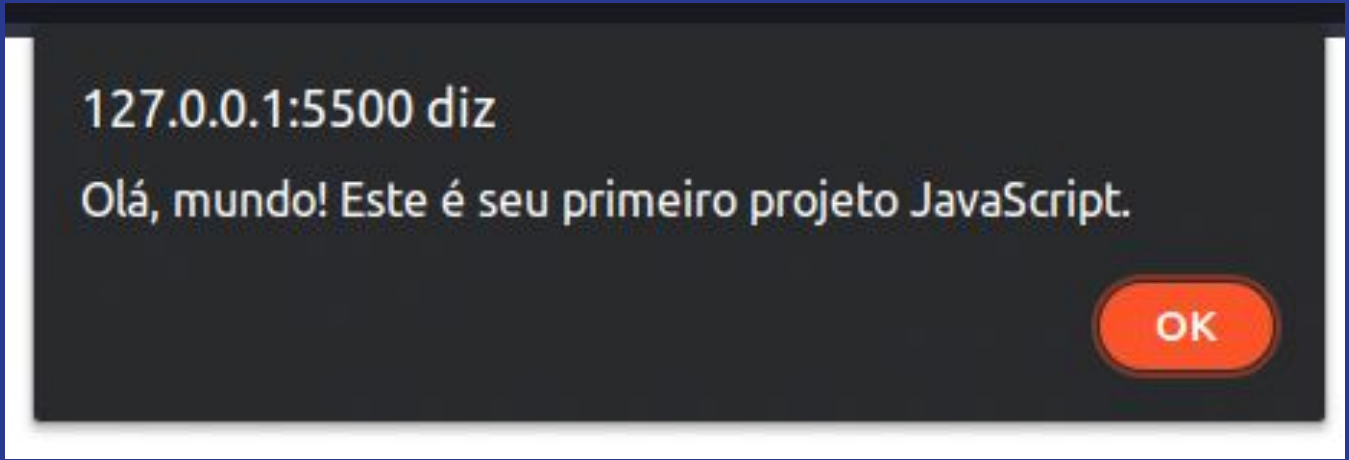
MÓDULO II

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hello World</title>
</head>
<body>
    <script>
        alert("Olá, mundo! Este é seu primeiro projeto JavaScript.")
    </script>
</body>
</html>
```

MÓDULO II

Resultado

Como você pode notar, ao final do projeto, quando a página for carregada, um alerta é disparado na tela, exibindo a mensagem informada.



MÓDULO II

2. Mudar a cor de fundo da página

Neste projeto, vamos aprender sobre como **alterar a cor de fundo de uma página utilizando um seletor de cores**.

Vamos à prática

Para organizarmos melhor o projeto, separamos o código em **três arquivos: um para o HTML, um para o CSS e outro para o JavaScript**.

No arquivo **HTML**, vamos **importar o link do arquivo CSS** dentro da tag **head**, utilizando o elemento **link**. Vamos também **adicionar o arquivo JavaScript na propriedade *source* da tag *script***, ao final do elemento **body** da página.

MÓDULO II

Em seguida, criamos a estrutura da página do projeto. Para isso, usamos a **tag *main***, que atua como uma ***div*** e **vai envolver o conteúdo do projeto**. Semanticamente, a tag ***main*** torna a estrutura da página mais organizada, pois indica qual é a ***div*** principal da estrutura.

À tag *main* atribuímos também a classe "container", que será utilizada para identificá-la. Dentro da tag, adicionamos uma ***div***. Dentro dessa ***div***, criamos uma legenda, orientando a pessoa usuária a selecionar uma cor. Além disso, também **adicionamos o campo input, do tipo *color*, para a seleção da cor.**

Além disso, passamos a **função *onChangeBg* para o campo**. Assim, quando uma cor for selecionada, a função será executada para que altere a cor de fundo da página.

MÓDULO II

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Background Color</title>
    <link rel="stylesheet" href="styles.css" />
</head>
<body>
    <main class="container">
        <div>
            <span>Selecione uma cor:</span>
            <input type="color" onchange="onChangeBg()" />
        </div>
    </main>
    <script src="script.js"></script>
</body>
</html>
```

MÓDULO II

Agora que já temos a estrutura **HTML**, vamos à estilização. Neste projeto, apenas centralizamos a legenda e o seletor de cores ao centro da página, usando a propriedade ***flex*** do **CSS**. Além disso, também adicionamos uma cor de funda a ***div*** que envolve estes elementos, para que eles fiquem em destaque.

MÓDULO II

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

main {
    width: 100%;
    height: 100vh;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
}

div {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    background-color: darkcyan;
    padding: 20px;
    border-radius: 5px;
}

span {
    color: #fff;
    font-size: 25px;
    margin-bottom: 15px;
}

input {
    border: none;
    height: 36px;
    width: 56px;
    cursor: pointer;
}
```

MÓDULO II

Por fim, é criada a lógica que vai alterar a cor de fundo da página. Para isso, no arquivo de **script**, iniciamos **capturando o conteúdo da tag identificada como "container"**. Isso é possível utilizando o comando **document.querySelector** do **JavaScript**, que recebe como parâmetro o identificador do elemento que deve buscar.

Por isso, passamos para ele o "**.container**", que indica que o elemento chamado de "**container**" deve ser buscado em todo o documento e ter suas propriedades capturadas. Além disso, usamos o mesmo comando para **capturar também as propriedades do *input*** seletor de cores da página.

Como este elemento não se trata nem de uma classe (referenciada pelo "." antes do nome) e nem de um id (referenciado pela "#"), apenas passamos o nome do elemento como parâmetro.

MÓDULO II

Em seguida, **definimos uma cor padrão** para a página. Se nenhuma cor for selecionada, o fundo se manterá branco. Por outro lado, quando uma cor for selecionada, a função ***onChangeBg* será executada e a cor de fundo da página será atualizada** para a escolhida.

```
const container = document.querySelector('.container');
const color = document.querySelector('input');

const defaultColor = '#fff';
container.style.backgroundColor = `${defaultColor}`;

function onChangeBg(){
  container.style.backgroundColor = `${color.value}`;
}
```

MÓDULO II

Resultado

Ao final, você terá uma página web com um seletor de cores. Inicialmente, a página terá a cor de fundo branca. Entretanto, ao selecionar uma cor, a página terá a cor de fundo alterada para a selecionada.

MÓDULO II

3. Relógio

Neste projeto em JavaScript vamos exibir um **relógio digital** na página.

Vamos à prática

Assim como no projeto anterior, vamos dividir o código em três arquivos.
Esta estrutura também será usada para os demais a seguir.
Iniciando com o arquivo **HTML**, adicionamos o **link** de estilo à tag ***head*** e o ***script*** ao final da tag ***body***.
Quanto ao conteúdo da página, adicionamos a **tag *main* ao *body* da página**.
Dentro de ***main*** **criamos os textos que vão representar as horas, minutos, segundos e a separação entre eles na tela**. Para isso usamos a tag **span**, que representa um conteúdo genérico.

MÓDULO II

Os *span* que contêm o valor das horas, minutos e segundos foram identificados com os respectivos **ids**: “**hour**”, “**min**” e “**sec**”. Estes identificadores serão utilizados pelo **JavaScript** para localizar os elementos que deverão ser alterados dinamicamente com os valores.

MÓDULO II

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Relógio</title>
    <link rel="stylesheet" href="styles.css" />
</head>
<body>
    <main>
        <span id="hour">00</span>
        <span>:</span>
        <span id="min">00</span>
        <span>:</span>
        <span id="sec">00</span>
    </main>
    <script src="script.js"></script>
</body>
</html>
```

MÓDULO II

Para a estilização, novamente centralizamos o conteúdo na página.
Aplicamos também uma cor de fundo a ela e aos **span**, para que fiquem em destaque.

```css
body {
  background-color: darkcyan;
}

main {
  width: 100%;
  height: 100vh;
  justify-content: center;
  align-items: center;
  display: flex;
}

span {
  font-size: 50px;
  margin: 5px;
  padding: 10px;
  border-radius: 5px;
}

#hour, #min, #sec {
  background-color: #fff;
}
```

MÓDULO II

Por fim, quanto à lógica **JavaScript**, iniciamos **capturando o conteúdo dos *span* de hora, minuto e segundo** com o comando ***document.querySelector***. Após isso, usamos a propriedade ***setInterval*** do **JavaScript** para definirmos um intervalo de tempo em que determinado código deve ser executado.

Em nosso caso, **obtemos a hora atual de 1 em 1 segundo** usando o método ***Date()***. para obter a data e hora atual. Também usamos ***getHours()***, ***getMinutes()*** **e** ***getSeconds()*** para capturarem as horas, minutos e segundos atuais, de forma separada, a partir da data obtida.

Com isso, **modificamos o conteúdo exibido na tela usando *innerHTML*** e executamos a função ***formatTime***. Essa função realiza um **tratamento dos valores**, atribuindo um **zero** em valores menores que **10**, como em **9:40:22**, que passaria a ser **09:40:22**.

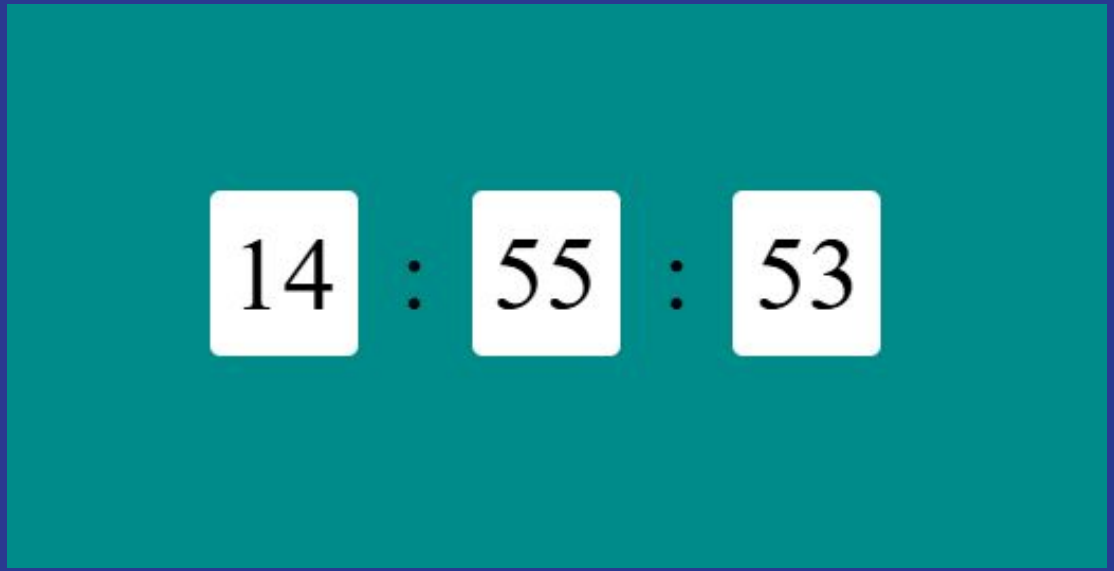
MÓDULO II

```
const hour = document.querySelector('#hour');
const min = document.querySelector('#min');
const sec = document.querySelector('#sec');

setInterval(() => {
  let date = new Date();
  let hours = date.getHours();
  let minutes = date.getMinutes();
  let seconds = date.getSeconds();

  hour.innerHTML = `${formatTime(hours)}`;
  min.innerHTML = `${formatTime(minutes)}`;
  sec.innerHTML = `${formatTime(seconds)}`;
}, 1000);

function formatTime(time){
  return time < 10 ? "0" + time : time;
}
```

MÓDULO II

Resultado

O resultado final será um relógio digital exibido na página web.



MÓDULO II

4. Contador

Agora, entre nossos projetos em **JavaScript**, vamos criar um **contador**. Será exibido um valor na tela e haverá dois botões, um para incrementar o valor e outro para reduzi-lo.

Vamos à prática

Novamente iniciamos pelo arquivo **HTML**, adicionando o arquivo de **style** e de ***script***. Criamos também nossa ***div*** principal usando a tag ***main***. **Dentro de main teremos um *span*, identificado pelo id "value"**, que vai exibir o valor do contador. Logo abaixo teremos nossos **botões**, criados com a tag *button*. Os **botões terão os textos "+" e "-"**, para identificarmos sua função na tela. O botão de **adição terá o atributo *onclick***, chamando a função ***onIncrement***, quando clicado. Já o **botão de subtração terá o *onclick*** chamando ***onDecrement***, quando clicado.

MÓDULO II

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contador</title>
    <link rel="stylesheet" href="styles.css"/>
</head>
<body>
    <main>
        <span id="value">0</span>
        <button onclick="onIncrement()">+</button>
        <button onclick="onDecrement()">-</button>
    </main>
    <script src="script.js"></script>
</body>
</html>
```

MÓDULO II

Na estilização, repetimos a ideia dos projetos em **JavaScript** anteriores. Centralizamos o conteúdo na página e atribuímos uma cor de fundo ao ***span*** e aos botões, para que fiquem destacados.

```css
body {
  background-color: darkcyan;
}

main {
  width: 100%;
  height: 100vh;
  justify-content: center;
  align-items: center;
  display: flex;
  flex-direction: column;
}

#value {
  background-color: #fff;
  color: black;
  width: 150px;
  height: 150px;
  border-radius: 5px;
  margin-bottom: 10px;
  align-items: center;
  justify-content: center;
  display: flex;
  font-size: 50px;
}

button {
  border: none;
  width: 150px;
  height: 56px;
  border-radius: 5px;
  margin-bottom: 10px;
  font-size: 30px;
  cursor: pointer;
  color: #fff;
  background-color: darkblue;
}
```

MÓDULO II

No ***script*** do projeto, teremos algo bem simples. **Capturamos o valor do *span***, através de ***document.querySelector***, passando "**#value**" como parâmetro. Repare que, por se tratar de um **id**, precisamos adicionar o símbolo **"#"** antes do nome atribuído.

Após isso, **criamos as funções** que vão aumentar e reduzir o valor atual: ***onIncrement*** e ***onDecrement***.

A função ***onIncrement*** atualiza o valor na página através do ***innerHTML***, que **recebe o valor atual acrescido de 1**.

Perceba que para a operação foi necessário recuperar o texto atual da página e usar o método ***parseInt*** do **JavaScript**. Isso é necessário pelo fato do valor vir em formato de texto e não ser possível realizar a soma numérica entre um texto e um número. O ***parseInt*** **converte o texto em número**.

O mesmo processo ocorre para a função ***onDecrement***.

Entretanto, **em vez de acrescentar o valor 1 ao valor atual, subtraímos**.

MÓDULO II

```
const value = document.querySelector('#value');

function onIncrement(){
  value.innerText = `${parseInt(value.innerText) + 1}`;
}

function onDecrement(){
  value.innerText = `${parseInt(value.innerText) - 1}`;
}
```

MÓDULO II

Resultado

Como você pode ver, criamos um contador, em que o valor é acrescido ou reduzido de acordo com a manipulação dos botões na página.

MÓDULO II

5. Slideshow

Neste projeto em **JavaScript** vamos construir um **slideshow**.
Desse modo, a página vai exibir uma sequência de imagens de modo autônomo.

Vamos à prática

Para este projeto, será necessário que você **tenha em sua máquina algumas imagens** para serem exibidas pelo slideshow. Por isso, na organização de seu projeto, crie uma **pasta chamada "img"** e coloque as imagens que desejar dentro. Recomendamos que use ao menos três.

Ademais, como nos projetos anteriores, adicionamos novamente os links do arquivo de estilo e o de ***script*** ao **HTML**. Em seguida, criamos nossa ***div*** principal usando a tag ***main***. Dentro da tag ***main*** **adicionamos um título**, utilizando a tag ***h1***, que indica o principal título da página e também o de maior fonte.

MÓDULO II

Logo abaixo, **adicionamos a tag *img***, que será responsável por exibir as imagens na página. Nessa tag, também adicionamos a **largura** e a **altura** que desejamos que o slideshow tenha. Além disso, utilizamos a **classe "slideshow"** para identificar essa tag.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Slideshow</title>
    <link rel="stylesheet" href="styles.css" />
</head>
<body>
    <main>
        <h1>Slideshow</h1>
        <img class="slideshow" width="600" height="400" />
    </main>
    <script src="script.js"></script>
</body>
</html>
```

MÓDULO II

Para a estilização, mais uma vez centralizamos os itens na página e adicionamos uma cor ao fundo, para que o conteúdo se destaque.

```css
body {
  background-color: darkcyan;
}

main {
  width: 100%;
  height: 100vh;
  align-items: center;
  justify-content: center;
  display: flex;
  flex-direction: column;
}

h1 {
  text-align: center;
  color: #fff;
}

img {
  border-radius: 10px;
}
```

MÓDULO II

Quanto ao **script**, iniciamos **capturando as propriedades do elemento identificado com a classe "slideshow"**, através do comando ***document.querySelector***. Definimos também uma variável que conterá um ***array*** **(lista), com o caminho das imagens** no projeto. Em nosso caso, como definimos a pasta "**img**", devemos acessar a pasta e indicar o nome dos arquivos.

Há ainda uma **variável que recebe o tempo que desejamos que o slide demore** para passar as imagens. Desse modo, atribuímos a variável o valor **2000**, que representa **2** segundos em milisegundos. Por fim, iniciamos uma **outra variável**, que terá a função de **contagem** no código, com o **valor zero**. Após isso, criamos a função ***moveSlideShow***. Na função, **a variável slide**, que contém as propriedades do elemento ***img***, **recebe o primeiro item da lista de imagens**. Perceba que o ***array*** de imagens recebe o valor inicial do contador como posição. Aqui é válido ressaltar que, na programação, a contagem dos itens de um ***array*** se inicia com valor zero, e não com o valor um. Por isso, o contador inicia com o valor zerado.

MÓDULO II

Em seguida, **a função realiza uma verificação**. Nessa análise, **é observado se o valor do contador é menor que o total de imagens da lista**. Em caso positivo, isso significa que ainda há imagens para exibir, por isso o contador tem seu valor acrescido de um. Caso contrário, o valor do contador é zerado e o slide é reiniciado, exibindo novamente a primeira imagem.

Ao final da função temos um *setTimeout*, um método **JavaScript** responsável por executar um determinado trecho de código a cada período de tempo. Para o método, **passamos como parâmetro a função e o tempo que definimos** anteriormente. Assim, a função será executada repetidas vezes, a cada **2** segundos, realizando a alteração das imagens.

Por último, **executamos a função utilizando *window.onload***, que é uma propriedade do **JavaScript** executada quando a página carrega pela primeira vez.

MÓDULO II

```
var slide = document.querySelector('.slideshow');
var images = [
  './img/1.jpg',
  './img/2.jpg',
  './img/3.jpg'
];

var time = 2000;
var count = 0;

function moveSlideShow(){
  slide.src = images[count];

  if(count < images.length - 1){
    count++;
  } else {
    count = 0;
  }

  setTimeout("moveSlideShow()", time);
}

window.onload = moveSlideShow;
```

MÓDULO II

Resultado

Nesse sentido, obtemos uma página com um quadro central, que vai exibir as imagens definidas, alternando-as a cada **2** segundos.

MÓDULO II

Modal

Para nosso último projeto em JavaScript de nível iniciante, vamos criar um **modal**. Um modal é uma janela flutuante, de tamanho reduzido, que é exibido ao clicar em um botão ou realizar uma ação na página. Geralmente, o modal apresenta algum conteúdo textual e um botão para que ele seja fechado e deixe de ser exibido.

MÓDULO II

Vamos à prática

Para este projeto em **JavaScript**, adicionamos mais uma vez o link de **style** e o ***script*** ao **HTML**.
Em seguida, **a tag *main* foi criada e identificada com a classe "container"**.
Essa tag envolve o conteúdo do **modal**.
Para o modal, criamos uma *div* identificada pela classe "modal". Dentro dela, adicionamos um título, utilizando a tag ***h2***, um texto genérico, utilizando a tag ***span***, e um botão, com a tag ***button***.
O botão foi identificado com a classe "btn-close" e recebeu em seu ***onclick* a função *onClose***.
Essa função fecha o **modal** quando o botão for clicado.

Fora da tag *main* e logo abaixo dela, temos o **botão de abertura** do **modal**. O botão foi identificado pela **classe "btn-open"** e recebeu no ***onclick* a função *onOpen***, que abre o **modal**.

MÓDULO II

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Modal</title>
    <link rel="stylesheet" href="styles.css" />
</head>
<body>
    <main class="container">
        <div class="modal">
            <h2>Titulo</h2>
            <span>
                Lorem ipsum dolor sit amet consectetur adipisicing elit. Quisquam
            </span>
            <button class="btn-close" onclick="onClose()">Fechar</button>
        </div>
    </main>
    <button class="btn-open" onclick="onOpen()">Abrir modal</button>
    <script src="script.js"></script>
</body>
</html>
```

MÓDULO II

Neste projeto, a estilização incorporou mais detalhes que os anteriores, pois para um **modal** o **CSS** também é parte fundamental. Desse modo, iniciamos centralizando o conteúdo na página e aplicando uma cor de fundo, para que as informações se destaquem.

Em seguida, damos início a estilização do **modal**. Para isso, aplicamos uma cor branca ao fundo da janela flutuante, uma borda e uma largura. A propriedade ***flex*** do **CSS** também foi utilizada para indicar que o conteúdo do modal deve permanecer em formato de coluna, isto é, um embaixo do outro.

Além disso, também justificamos o texto e aplicamos uma cor de fundo azul ao botão de fechar.

O botão de abrir também ficou com uma estilização semelhante. Em seguida, utilizando a **classe "container"**, que se refere a tag ***main***, definimos que **o modal deveria estar posicionado de modo fixo ao topo**.

MÓDULO II

Outro ponto importante é que **utilizamos *z-index* de 999**, para que o **modal se sobreponha a todos os demais elementos** da página quando exibido. Aplicamos também um ***display: none*** ao container, assim **o modal não será exibido quando a página carregar**.

Logo a seguir, definimos as **classes “.active” e “.active .modal”**, que contém a propriedade ***display: flex***, responsável por **retornar a exibição do modal na tela**, quando solicitado pelo **JavaScript**, e uma animação, que vai dar um efeito a exibição da janela.

MÓDULO II

```css
body {
  background-color: darkcyan;
  width: 100%;
  height: 100vh;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  display: flex;
}

.modal {
  background-color: #fff;
  display: flex;
  flex-direction: column;
  padding: 30px;
  border-radius: 10px;
  width: 30%;
}

span {
  margin: 10px auto;
  text-align: justify;
}

.btn-close {
  margin-top: 10px;
  color: #fff;
  cursor: pointer;
  font-size: 16px;
  padding: 10px;
  border: none;
  border-radius: 5px;
  background-color: darkblue;
}

.btn-open {
  background-color: darkblue;
  height: 56px;
}
```

MÓDULO II

```css
.btn-open {
  width: 100px;
  color: #fff;
  border: none;
  border-radius: 10px;
  font-size: 16px;
  cursor: pointer;
}

.container {
  width: 100vw;
  position: fixed;
  top: 30px;
  z-index: 999;
  display: none;
  justify-content: center;
  align-items: center;
}

.active {
  display: flex;
}

.active .modal {
  animation: modal .4s;
}

@keyframes modal {
  from {
    opacity: 0;
    transform: translate3d(0, -60px, 0);
  }
  to {
    opacity: 1;
    transform: translate3d(0, 0, 0);
  }
}
```

MÓDULO II

No ***script***, **capturamos as propriedades do elemento identificado pela classe "container"**. Após isso, definimos as funções ***onOpen*** e ***onClose***. Quando ***onOpen*** for chamada, ela **vai adicionar a classe "active" ao container**, modificando a propriedade inicial ***display: none*** **para** ***display: flex***, exibindo assim a janela na página.

Por outro lado, quando ***onClose*** for chamada, **a classe será removida** e o modal deixará de ser exibido. Essas ações são possíveis em virtude da propriedade ***classList***, que permite adicionar e remover classes do conteúdo da página **HTML**.

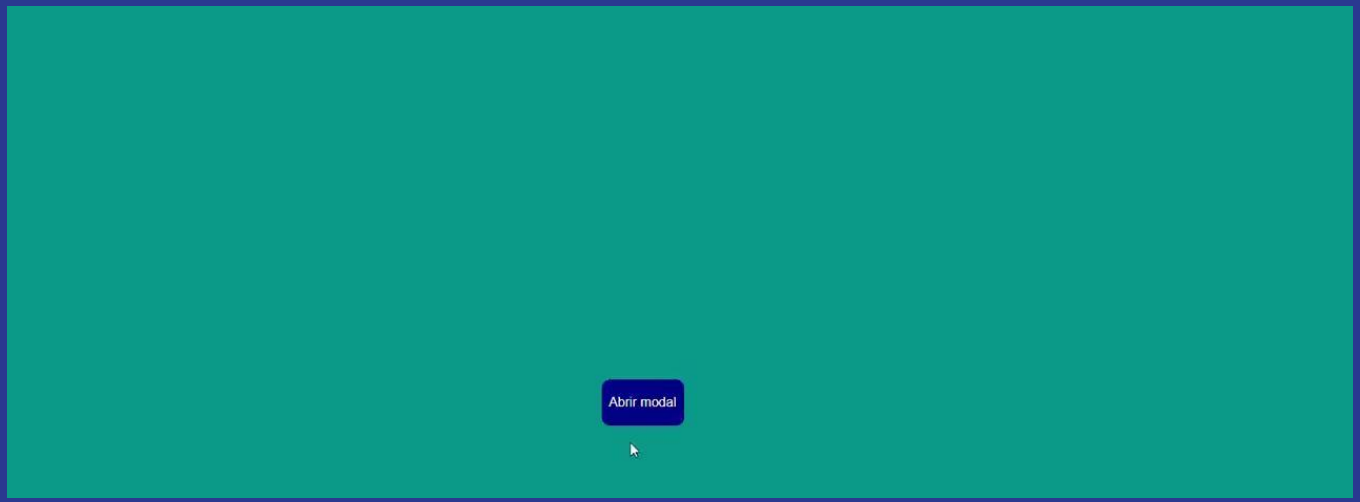
MÓDULO II

```
const modal = document.querySelector('.container');

function onOpen(){
  modal.classList.add('active');
}

function onClose(){
  modal.classList.remove('active');
}
```

MÓDULO II

Resultado

Ao final, você terá uma página com um botão principal para abrir o modal.
Quando clicado, o modal será exibido, mostrando seu conteúdo e um botão para que seja fechado.



MÓDULO II

5 ferramentas legais para aprender e treinar Javascript!

1. Guia para pessoas desenvolvedoras do Mozilla

Trata-se do guia JavaScript mais completo da internet. Mantido pelo Mozilla, empresa responsável pelo navegador Firefox, o guia possui uma **documentação completa sobre a linguagem**.
Lá, você pode encontrar **explicações sobre a sintaxe do JS, conceitos e boas práticas.**

2. CoderByte

CoderByte é uma **plataforma de desafios de programação**. Nela você pode encontrar uma variedade de **exercícios para praticar, com diferentes conceitos e níveis de dificuldade**. O ponto negativo é que apenas os 20 primeiros exercícios são grátis, após isso é necessário pagar uma assinatura.

MÓDULO II

3. Paqmid

Paqmid também é uma **plataforma de exercícios de codificação**. Novo no mercado, o site traz algumas inovações, como **respostas completas** para as atividades, exercícios criados por pessoas especialistas e tarefas em diferentes formatos, como testes e atividades padrões de resolução de problemas.

4. Fóruns

Fóruns, como o Stack Overflow, apesar de não oferecerem exercícios, podem ser uma grande chance de **aprimorar as habilidades tentando responder a dúvidas** de pessoas que estão fazendo perguntas por lá.

5. Hacker Earth

Hacker Earth, além de ser uma **plataforma de desafios de programação online**, também oferece a opção de **competições ao vivo**. Dessa forma, você pode aprimorar suas habilidades e também concorrer a prêmios nas competições.

MÓDULO II

EXERCÍCIOS...

MÓDULO II

FIM...

A Educação é o primeiro passo para um futuro melhor...