

JavaScript básico





<title>

AGENDA DO </> </> </> MÓDULO

</title>

 Agenda do módulo

 Anotações



{Conheça o projeto}

{Dando vida ao formulário}

{Adicionando elementos na DOM}

{Criando uma tabela de
administração para o formulário}

{Adicionando funcionalidades à nossa tabela}

{Adicionando função de administrador na página}

JavaScript básico: Finalizando o projeto



<title>

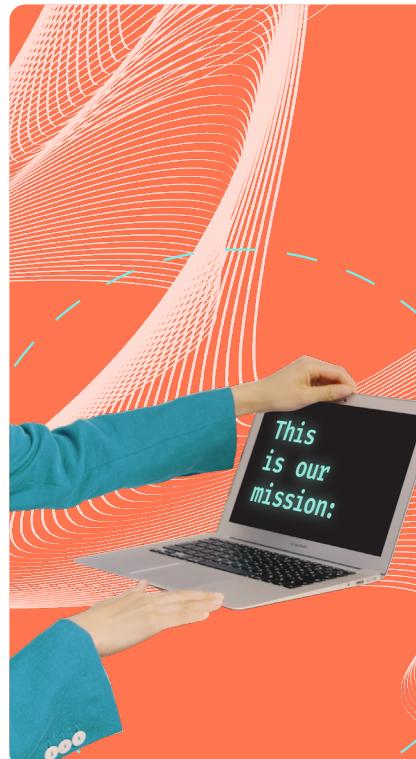
MISSÃO DO MÓDULO

</title>

Aplicar novas funcionalidades na landing page, usando **JavaScript**.

{01.}

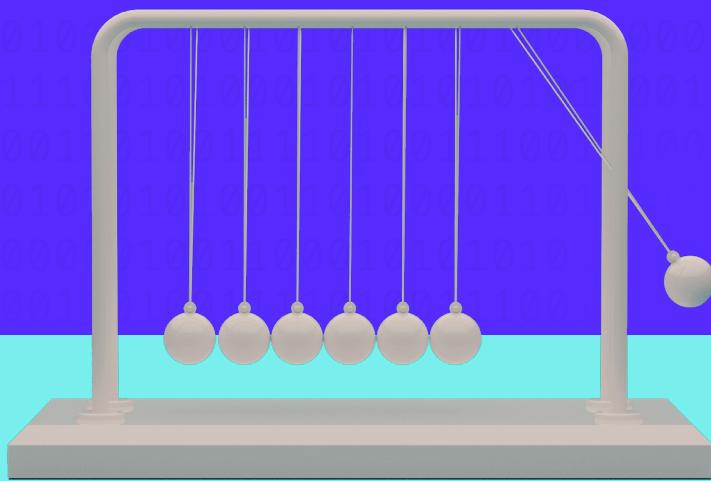
Conheça o projeto



JavaScript básico: Finalizando o projeto



Toda ação



tem uma função.

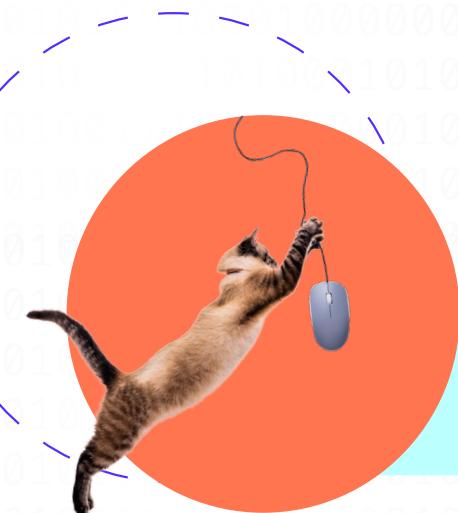


É hora de retomar aquele projeto e trazer mais interatividade e vida para o seu conteúdo usando os conhecimentos adquiridos em JavaScript. Assim como no primeiro projeto, o modelo está desenhado no Figma, que é uma ferramenta de prototipagem bastante usada nos projetos de design para web.

Caso não tenha feito o projeto, você pode baixar o código fonte no GitHub da Conquer, disponível no QRCode ao lado.

Vá até a pasta “projeto”. É esta a pasta em que será realizado o nosso projeto de JavaScript.

E se você fez a landing page no curso de HTML e CSS, vai reparar que o projeto é -quase- o mesmo, mas existem existem alguns elementos, estilos e propriedades a mais presentes no código HTML. Por isso recomendamos que você também use o template que disponibilizamos ao invés do seu projeto.

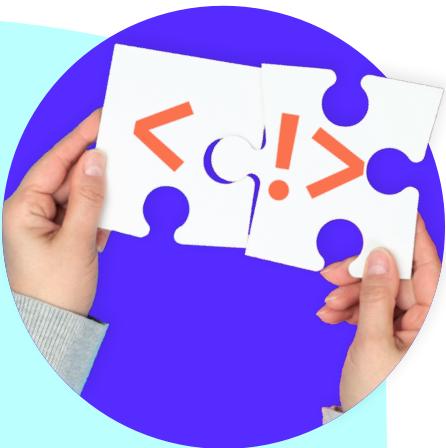


<title>

PULO DO GATO

</title>

- Aprenda com o **código** de outros programadores.



Após o usuário preencher e clicar no botão, faça com que sejam printados no console todos os campos preenchidos no formulário.

Importante: A página não deve recarregar ao ser dado submit no formulário.

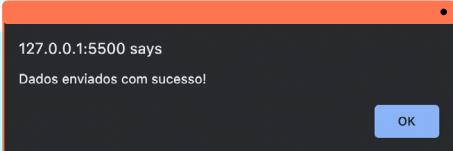
Deve aparecer um pop up mostrando que o submit deu certo!

```
<title>
```

MÃO + + NA MASSA

```
</title>
```

```
y (nome: 'John Doe', email: 'john.doe@email.com', telefone: '9999999999', experiencia: '2', vagas: '1', ...)
  apresentacao: "Uma breve descrição sobre mim!"
  linkedin: 'https://www.linkedin.com/in/johndoe'
  email: 'john.doe@gmail.com'
  experiencia: '2'
  linguagens: '>Any'
  github: 'https://github.com/a'
  perfilUrl: 'https://github.com/a'
  telefone: '9999999999'
  vagas: '1'
> [Object]
```





{02.}

Dando vida ao formulário

Vamos conferir se todos os requisitos pedidos na aula anterior foram atendidos.

E para começar o projeto, precisamos criar um arquivo “index.js”.

Feito isso, a próxima coisa que vamos fazer é coletar os dados do formulário que está na página. Para isso, precisamos buscar uma forma de conectar o formulário que está no arquivo index.html com o JavaScript.

Você já viu no módulo 6 como fazer isso. Existem várias formas, e a que vamos usar aqui é utilizando a propriedade “id” do HTML.





O próximo passo é criar um evento em que o JavaScript fica “escutando” o submit do form.

Para isso, existe uma função nativa do JavaScript chamada “addEventListener”.

Ela recebe dois parâmetros, sendo o primeiro uma string com o que é para ser “escutado”, como por exemplo: um clique ou um submit de form.

O segundo parâmetro é uma função, em que podemos executar qualquer coisa. Essa função será executada assim que o evento escolhido acontecer.

Se testarmos na página, notaremos que ela irá recarregar ao dar submit, e não irá printar o código no console. Isso acontece porque esse é um comportamento padrão do HTML.

Embora seja padrão, é possível quebrá-lo.

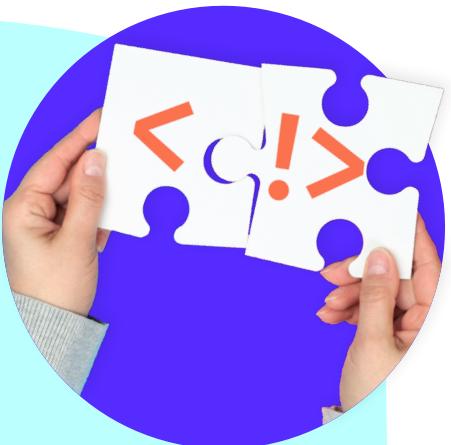


<title>

PULO DO GATO

</title>

- **preventDefault:** serve para prevenir o comportamento padrão de um evento.



<title>

MÃO + + NA MASSA

</title>

Adicione um botão na página que faça com que, ao clicar nele, a página volte ao topo.

Importante: O botão deve sempre aparecer caso a altura entre o começo da página e o topo da página seja de 300px.

Ir para o topo



{03.}

Adicionando elementos na DOM



Primeiro, precisamos criar um botão, dar a ele os estilos necessários: um id e sua função ao clicar.

Precisamos criar uma função que faça a página ir até o topo.

Para isso, criaremos uma função com o nome “scrollToTop”.

Nela, podemos usar o objeto window, que contém uma propriedade chamada “scrollTo”.

Essa é uma função que recebe um objeto que aceita receber algumas propriedades.
Iremos usar duas, sendo elas “top” e “behavior”.

Agora que temos o botão e sua funcionalidade, vamos fazer com que o botão só apareça depois de um certo ponto na página.





Vamos inserir uma função que, quando o usuário fizer scroll, verifica se qual a distância entre o começo da página e onde a tela está.

Se a distância for maior que 300px, o botão deve aparecer. Para isso, vamos inserir o botão na página, usando "appendChild".

```
<title>
```

MÃO + NA MASSA

```
</title>
```

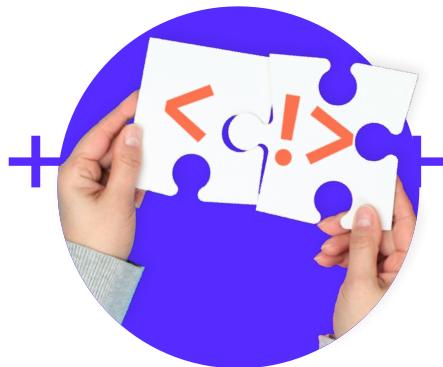
Crie uma tabela de administração para o formulário em tela.

Para cada submit no formulário, insira as informações dos inputs para a tabela.

A tabela deve ficar assim:

Usuários cadastrados			
Nome	E-mail	Telefone	Vaga
João Santos	joao@email	419999999	2

E quando for menor do que 300, deve-se remover o botão da página. E para isso, iremos utilizar a propriedade "removeChild".



OBS: De todas as informações do formulário, deve conter na tabela apenas:

- Nome
- E-mail
- Telefone
- Número da vaga

Dica: Deve ser criada uma função apenas para a inserção dos dados na tabela.



{04.}

Criando uma tabela de administração para o formulário

Agora, vamos começar a implementar o código JS para que a nossa tabela funcione junto ao formulário.

Agora precisamos construir de fato a função. Ela precisa:

- Inserir uma linha nova na tabela.
- Inserir um ID de usuário de forma automática.
- Inserir na tabela as informações que o usuário preencheu no form.





Temos que adicionar novas linhas dentro do `tbody`, sem alterar o `header`. E estamos usando o seletor de array (`[0]`), pois podem ser selecionados mais de um `tbody` (caso existam) dentro da `table`. (É mais uma garantia de que estamos pegando o elemento certo)!

Para inserirmos os valores vindos do formulário, vamos usar as propriedades que ‘setamos’ na nossa função.

```
<title>
```

MÃO + + + + + -

NA MASSA

```
</title>
```

- › Adicione um ID para cada novo usuário inserido na tabela.
- › Adicione um botão de excluir o usuário cadastrado.
- › O botão deve ter a label de “Excluir”.

Atenção: Ao clicar no botão de excluir, deve ser excluída apenas a linha em que o botão foi clicado.

Usuários cadastrados				
Nome	E-mail	Telefone	Vaga	Ação
João Santos	joao@email	419999999	2	<button>Excluir</button>

+ + + + + -



A persistência é o caminho do êxito.

Charles Chaplin



{05.}

Adicionando funcionalidades à nossa tabela

A missão agora é melhorar a tabela criada anteriormente.

Para inserir as adições propostas pelo exercício, a primeira coisa a se fazer é adicionar um identificador para cada usuário novo cadastrado na tabela (muito comumente chamado de “id”).

Vamos adicionar um contador de forma automática no nosso código.

O primeiro passo é inserir mais uma célula em nossa tabela.

O próximo passo é contar quantos elementos têm dentro de um array. Para isso, vamos usar a função “length”.

Basta você adicioná-la ao final de um array, como por exemplo: [“pato”, “cachorro”, “gato”].len





Já inserimos um botão através do JavaScript, e aqui não será diferente! Precisamos criar o elemento do botão, dar as características de classe, onclick e estilo para ele.



<title>

PULO DO GATO

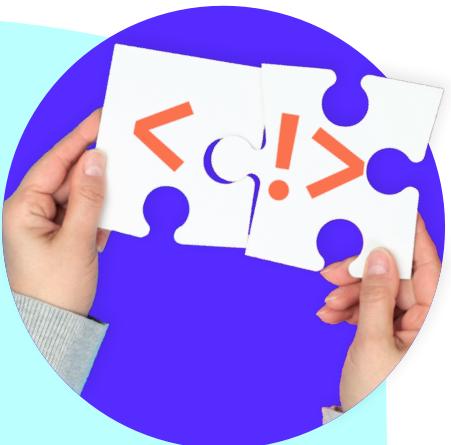
</title>

- Antes de criar o botão em si, **crie a função que será executada** ao clicar no botão.

Dito isso, como todos os nomes de função criados até agora, vamos dar um nome condizente ao objetivo da função.

O nome dela será “handleExcludeUser”, e ela basicamente precisa excluir a linha toda do botão clicado.

Essa função receberá um parâmetro, que é o evento do clique do botão.



<title>

MÃO + + NA MASSA

</title>

Adicione a funcionalidade de mostrar ou não a tabela.

No template, logo após o botão de “Enviar” do formulário, há um input com a label “Administrador”. Faça com que, caso ele seja marcado, a tabela apareça. Caso seja desmarcado, a tabela desapareça da tela.



{06.}

Adicionando função de administrador na página

Agora, iremos adicionar uma validação para saber se a pessoa que está acessando a página é um admin ou não.

Se ela for um admin, a tabela deve aparecer na página. Se não, ela não aparecerá.

Repare que no HTML já existe um input do tipo checkbox. E caso ele esteja marcado, a tabela deve aparecer. Para isso, usaremos um recurso já estudado, o “addEventListerner”.





Ele escuta eventos, e o evento a ser escutado aqui é do tipo “change”. Sempre que houver mudança no input, deve haver uma validação para: Se ele estiver marcado, mostrar a tabela. Se não, esconder a tabela.

Perceba que estamos validando a propriedade `.checked` do elemento checkbox, para ver se o input está marcado ou não.

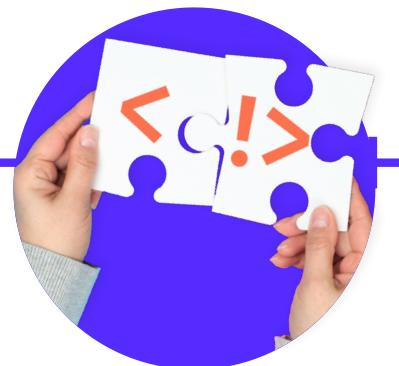
```
<title>
```

MÃO + + + NA MASSA

```
</title>
```

Crie um formulário com dois campos de checkbox, sendo um campo com um Label “Sim” e outro com um label “Não”.

Crie um botão do tipo submit, considerando a seguinte regra: o botão “Enviar” só poderá ser habilitado se o checkbox “Sim” estiver selecionado, ou seja, caso nenhum campo esteja selecionado ou o campo “Não” esteja selecionado — o botão “Enviar” deverá permanecer desabilitado. Exemplo:





<title>

DESAFIO CONQUER

</title>



```
if (challenge)  
// start
```

<body>

Utilizando a Landing Page que você criou no Desafio Conquer do módulo 08 de HTML/CSS:

Acrescente interações e funcionalidades que você aprendeu neste módulo, como:

- Customização de estilos.
- Botões, como o de retornar ao topo.
- E formulários.

Observação: Se você não realizou o curso de HTML e CSS, use o template deste módulo e customize sua própria landing page.

</body>



<title>

QUERO MAIS

</title>



A história do JS
pela visão do
próprio criador
Coding Tech

**Netflix JavaScript
Talks**
Explicação do time
de engenharia da
Netflix sobre como
foi fazer o episódio
interativo do Black
Mirror e os desafios de
engenharia envolvidos
no processo.

**Podcast
Conversa Ágil**



```
if (interested)  
// gimmemore
```



Anotações

-
-
-
-
-
-

