INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

JAVASCRIPT - Aula 11

Objetivos da aula:

- Chamadas HTTP com fetch

INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

Chamadas HTTP

Muitas vezes para a criação de um site é necessário enviar ou receber informações entre servidores. A internet está armazenada em um servidor. Um site está armazenado em um servidor. Para acessar seu banco de dados é necessário também um servidor backend. Esta troca de informações se dá através de **HTTP**.
O **Hypertext Transfer Protocol** é um protocolo de comunicação utilizado para sistemas de informação. Ele é a base de transferência de informações entre backend e frontend, que são feitas sempre por meio de **rotas** (como a rota para acessar o site google, www.google.com) e se baseia em alguns métodos:

- **GET** – ler/captar/receber informações;
- **POST** – Enviar/criar informações;
- **PUT** ou **PATCH** – Atualizar informações;
- **DELETE** – deletar informações

Juntas, elas formam o conceito de **CRUD: Create (POST), Read (GET), Update (PUT/PATCH) e Delete**

https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview

Chamadas HTTP

O Javascript dispõe nativamente de um método para chamadas HTTP chamado **fetch**. Pense que você deseja trazer do seu banco de dados todos os usuários cadastrados, ou seja, fazer uma requisição **GET**. O backend da aplicação deve estar preparado para receber esta requisição por meio de rotas já configuradas, e basta você saber a **url** desta rota para fazer a chamada**.**
O fetch recebe como parâmetro a url e retorna a resposta desta rota. Esta resposta pode ser captada então pelo método **then (então, em inglês)** , onde você pode dizer o que irá acontecer com esta resposta e captar algum erro caso aconteça com o **catch.**

```
fetch("/usuarios")
.then(resposta => faz algo)
.catch(error => console.log(error.message))
```

Toda chamada HTTP retorna uma promise. Saberemos detalhadamente na próxima aula.

https://developer.mozilla.org/pt-BR/docs/Web/API/Fetch_API/Using_Fetch

fetch

A maioria dos dados retornados ou enviados em uma chamada HTTP vêm em formato de JSON (aprendido na aula 08). Podemos captar esta resposta em JSON e transformar em código js, com o método **response.json(), um método de promises.** Depois da resposta transformada, podemos mostrar a resposta ao usuário. Podemos encadear outro **then** para organizar o código. O código captado no segundo **then** precisa ser o retorno dos dados do primeiro **then**.

```
fetch("/usuarios")
.then(resposta => resposta.json())
.then(data => alert(data))
.catch(error => console.log(error.message))
```

Digamos agora que você queira enviar dados para o cadastro de um usuário para o banco de dados. A requisição agora será um **POST**.

fetch

Você precisa indicar o método para o fetch e também enviar os dados caso a requisição não seja GET. O fetch dispõe de um segundo parâmetro opcional, **um objeto que irá enviar os dados da requisição**. É necessário passar como propriedades deste objeto o **method**, o **body** e o **headers**, entre outros.

- O **method** indica o método HTTP
- O **body** é onde ficam os dados do usuário
- **O headers** recebe outro objeto que indica metadados da requisição. Um metadado necessário neste caso é o tipo de dado que está sendo enviado no corpo, que será um dado JSON. Então o código fica assim:

```
const usuarioNovo = {//dados}

fetch("/usuarios", {
   method: “POST”,
   body: JSON.stringify(usuarioNovo),
   headers: {
      “Content-type”: “application/json”
   }
})
.then(res => res.json())
.then(data => alert(data))
.catch(error => {
   console.log(error.message)
})
```

Atividade prática

O endereço https://dog.ceo/dog-api/ apresenta uma API gratuita que traz fotos aleatórias de cachorros e informações sobre raças. Leia a documentação dela e faça os exercícios:

Exercício 1: Crie um botão que fará uma chamada para captar fotos aleatórias de cães.

Exercício 2: Crie uma chamada que retornará raças de cachorros e mostre ao usuário.

Exercício 3: Modifique a chamada do exercício 1 para retornar imagens de apenas uma raça de cachorros, de acordo com que o usuário informar.

Dica: Todas as requisições serão GET. As três chamadas já têm endpoints (rotas) específicos indicados na documentação do site.

Atividade Prática

Dica: Em index.js, o objeto musicians simula um banco de dados.

Exercício 1: Modifique os dados do objeto **musicians**, adicionando a banda de cada artista e algumas de suas músicas.

Exercício 2: Crie um endpoint para retornar todas as músicas do banco de dados.

Exercício 3: Crie um endpoint para retornar as músicas de um determinado cantor.

Exercício 4: Permita buscar uma canção por nome. Emita uma mensagem se a canção não for encontrada.

Obs: Para cada exercício, crie um botão de chamada e um campo de resposta específico na página.

Você concluiu a aula 11 do seu módulo de Javascript.
Continue praticando e até a próxima aula!

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER