Desenvolvimento de Aplicações para Web

Profa. Marla Cristina Sopeña

Instituto Federal Sul-rio-grandense

1. Definindo Fetch

A API Fetch fornece uma interface JavaScript para acessar e manipular pedidos e respostas HTTP (`Request`/`Response`).

Esta API possui o método fetch(), que fornece uma maneira para buscar recursos de forma assíncrona através da rede. Este tipo de funcionalidade era obtida anteriormente utilizando XMLHttpRequest.

A API fetch provê ao navegador uma interface para a execução de requisições HTTP através de Promises.

O método fetch () tem um argumento obrigatório, o caminho para o recurso que se deseja obter. Ele retorna uma promessa (promise).

Promises definem uma ação que vai ser executada no futuro, ou seja, ela pode ser resolvida ou rejeitada.

2. Sintaxe básica:

```
fetch(url)
.then(function() {

})
.catch(function() {

});
```

método `fetch()`	Retorna uma promessa.
método `then()`	Executa uma função se a promessa retornada for `resolve`. Essa função contém o código para lidar com os dados recebidos do recurso.
método `catch ()`	Usado para lidar com `reject`. O código dentro de `catch()` será executado se um erro ocorrer ao chamar o recurso.

Ou seja, utilizamos o then quando queremos resolver a Promise, e o catch quando queremos tratar os erros de uma Promise rejeitada.

Tanto then quanto catch retornam outra Promise, e isso permite que possamos fazer o encadeamento de mais de um bloco then ou catch, por exemplo.

```
fetch(url)
.then(function(response) {
      return response.json()

})
.then(function(dados) {
        console.log(dados)
})
```

```
.catch(function(e) {
        Console.log("erro"+e)
})
```

2. Sintaxe com options:

```
Options= {
  method: 'POST',
  mode: 'cors', // pode ser cors ou basic(default)
  redirect: 'follow',
  headers: new Headers({
   Content-Type': 'text/plain'
  })
fetch(URL_TO_FETCH, options
})
.then(function(response) {
  // tratar a response
});
```

3. O que é CORS?

Cross-Origin Resource Sharing é a sigla atribuída ao mecanismo dos navegadores que gerencia o compartilhamento de recursos entre diferentes origens.

Por padrão, quando fazemos uma requisição através do XMLHttpRequest ou utilizando a API Fetch do JavaScript, os navegadores utilizam a política same-origin, que só autoriza a troca de recursos entre as mesmas origens (domínios).

Se for necessária essa troca entre diferentes origens, é necessário configurar as chamadas para que contenham os cabeçalhos CORS corretos.

A utilização do CORS é importante, porque além de poder restringir quem pode acessar os recursos de um servidor, também pode especificar como esses recursos devem ser acessados.

Como funciona o CORS?

O mecanismo funciona adicionando cabeçalhos HTTP nas respostas dos servidores para que estabeleçam um compartilhamento de recursos seguros entre diferentes origens.

Com ele é possível configurar as origens permitidas para acessar o recurso e quais métodos (GET, POST, PUT, DELETE, etc) elas podem utilizar.

- `"cors"`- o padrão, solicitações de origem cruzada são permitidas
- `"same-origin"` - solicitações de origem cruzada são proibidas,
- `"no-cors"` - somente solicitações seguras de origem cruzada são permitidas.

Cada método HTTP é usado com um proposito.

Método HTTP	Uso
GET	Obter algum ou alguns recursos, nesse caso para consulta
POST	Inserir uma nova informação
PUT	Atualizar uma informação.
DELETE	Remover uma informação.