Beabá do JavaScript

Photo by Matthias Heyde on Unsplash

Módulo 3 - Aula 3 - HARD

RESILIA | Senac

Destructuring

Destructuring (ou desestruturação) é uma expressão JavaScript que permite a extração de dados de um array ou objeto.
Podemos transformar os valores constantes ou variáveis e manipulá-los no código separadamente.

Destructuring

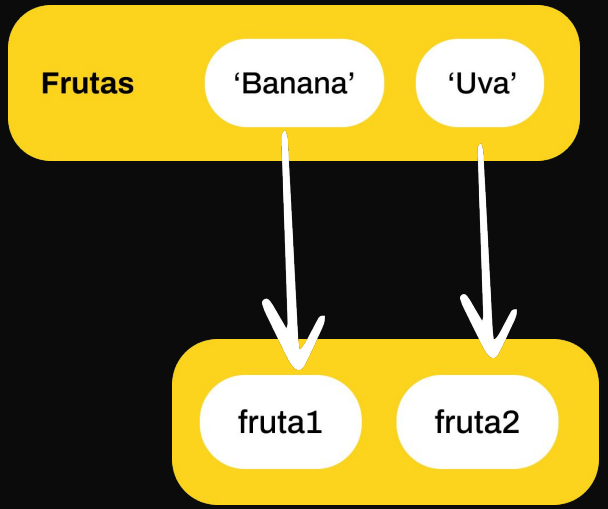
Isso pode nos facilitar para usar propriedades e métodos.

Muito usual quando estamos programando com React

Destructuring - Array

```
const frutas = ['Banana', 'Uva']

const [fruta1, fruta2] = frutas

// fruta1 = 'Banana'

// fruta2 = 'Uva'
```

Destructuring - Array



Destructuring - Objeto

```
const { nome, sobrenome } = { nome: "Gregório", sobrenome:
"Silva" }


// nome = 'Gregório'

// sobrenome = 'Silva'
```

Destructuring - Objeto

Destructuring - Objeto & Array

```
const objComplicado = {
  arrayProp: [
    "Resilia",
    { ramo: "Educação" }
  ]
}

const { arrayProp: [primeiro,  segundo ] } = objComplicado
```

Destructuring - Objeto + Array

Módulos (1)

Anos atrás o JavaScript era utilizado apenas para execução de pequenos scripts em páginas web, com o passar dos anos tornou-se necessária a separação de “pedaços” do código com intuito de melhorar a manutenção e separar responsabilidades, funcionalidades, etc.
Esses pedaços são o que chamamos de módulos.

Módulos (2)

Antes dos módulos, era necessário declarar variáveis globais nos arquivos para serem reutilizadas, agora podemos passar de um para outro importando e exportando.

Módulos são carregados de maneira síncrona, então módulos que dependem de outros módulos devem ser lidos mais abaixo do código.

CommonJS

Anteriormente chamado de ServerJS, o CommonJS permite o encapsulamento de código, isso se dá a partir do uso de módulos, sem variáveis globais que entrem em conflito com outras.

O CommonJS ajuda no gerenciamento de injeção de dependência.

CommonJS - Import & Export

```
// media.js
exports.calculaMedia = (notas) => {

  const total = notas.reduce((acc, atual) => {

     return acc + atual

  })

  return total / notas.length

}
```

CommonJS - Import & Export

```
// em outro arquivo

const media = require('./media.js')

const notasPorBimestre = [7, 9, 4, 9]

media.calculaMedia(notasPorBimestre) //
```

ES6 Modules

Além da sintaxe do CommonJS, temos o EcmaScript 6 Modules, que traz uma abordagem mais semântica.

ES6 Modules - Import & Export

```
// media.js
export function calculaMedia(notas) {
  const total = notas.reduce((acc, atual) => {
    return acc + atual
  })
  return total / notas.length
}
```

ES6 Modules - Import & Export

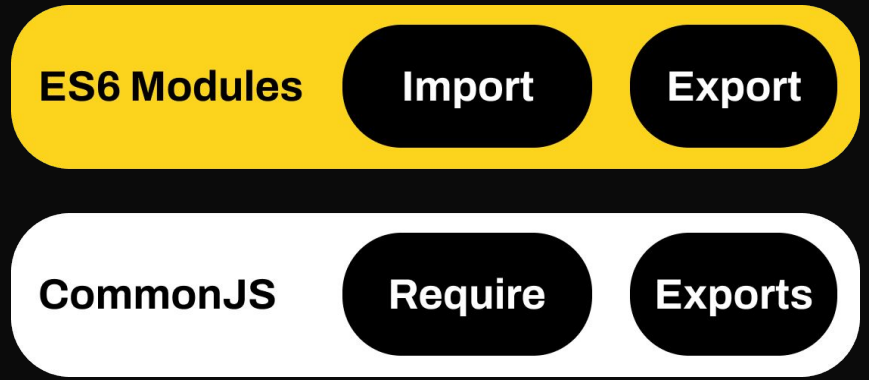
```
// em outro arquivo
import { calculaMedia } from './media.js'

const notasPorBimestre = [7, 9, 4, 9]

calculaMedia(notasPorBimestre)
```

CommonJS vs ES6 Modules

Saber essa diferença é necessária, pois o React utiliza a sintaxe dos ES6 Modules. Isso servirá para importarmos diversas dependências, outros módulos criados por nós, etc.

CommonJS vs ES6 Modules

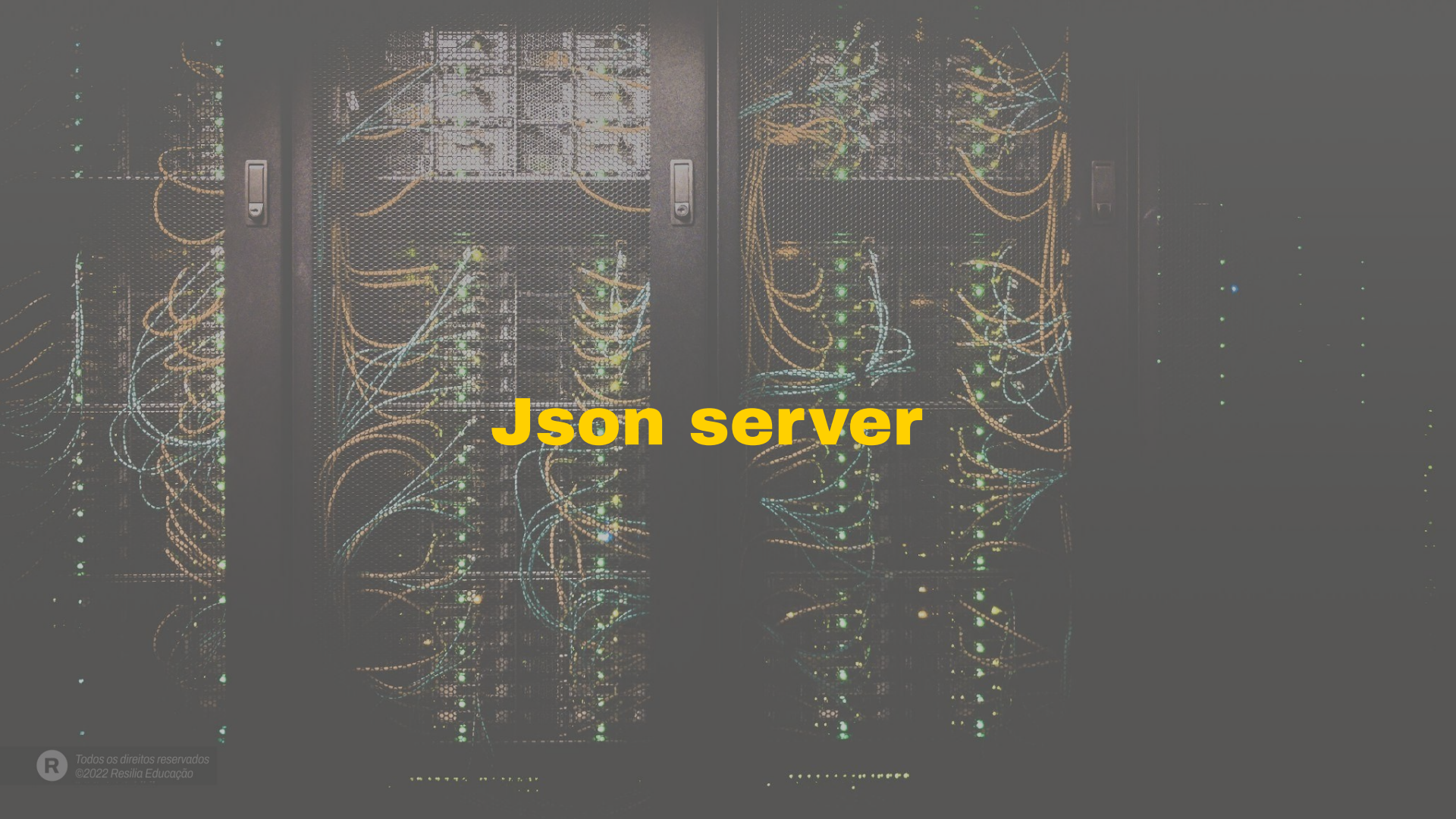
Usado pelo React



Mão na Massa: Desestruturando tudo

- Acessar o arquivo de exercícios no roteiro de aula
- Desestruturar todos os itens do array alunos
- Desestruturar todos os itens de array disciplinas, exceto o terceiro
- Acessar todos os itens dos dois objetos disponíveis

DICA: se atente caso haja necessidade de renomear uma propriedade’



Json server

Json server nos permite criar uma simulação de uma API REST (servidor que trabalha com HTTP) que se comunica em JSON.

Vamos ver Json-server na prática?

Json server - Instalação

Instale de forma global a biblioteca json-server usando o npm.

```
npm i json-server
```

Json server - Dados

Crie um arquivo chamado db.json, onde irá conter os dados do nosso servidor.

```
{
  "notas": [
    {
      "id": 1,
      "titulo": "Estudar JS",
      "status": true
    }
  ]
}
```

Json server - Servidor

Crie um arquivo chamado server.js, este arquivo ficará responsável por iniciar nosso servidor.

```
import jsonServer from 'json-server';
const server = jsonServer.create();
const router = jsonServer.router('db.json');
const middlewares = jsonServer.defaults();
const port = process.env.PORT || 3200;

server.use(middlewares);
server.use(router);

server.listen(port);
```

Json server - Run

Por fim, para iniciarmos nosso servidor devemos rodar o comando:

```
node server.js
```

Como colocamos nosso servidor na internet?

Render.com

Render.com

Render.com é um serviço de hospedagem onde podemos deixar nossos projetos rodando na internet.

Render.com - Hospedando

Para hospedar nosso servidor teremos que seguir os seguintes passos:

- Criar conta no render.com;
- Criar uma nova hospedagem de um “Web Service”;
- Conectar os repositórios com o gitHub;
- Selecionar o repositório do projeto que contém nosso json-server;
  - Aplicar um nome ao projeto;
  - Mudar o “Start Command” para “node server.js”.

Depois do site construir/preparar o projeto ele ficará online no link disponibilizado abaixo do nome do projeto.

Atividade em pares