BOSCH
Invented for life

Aula 3 - Javascript Assíncrono

Docupedia Export

Author: Lima Queila (CtP/ETS)
Date: 23-Jun-2022 14:22

Table of Contents

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

1 Promises

É uma classe que permite a construção de funções de processamento assíncrono representando um valor que poderá estar disponível no futuro. Assim evitamos criar uma árvore de callbacks com uma função dentro outra dependendo de condicionais. Podemos executar funções assíncronas de forma que pareçam síncronas.

1.1 Sintaxe:

```
new Promise((resolve: Function, reject: Function) => void)
```

- **resolve**: função para retornar o resultado da promise.
- **reject**: função para retornar o erro da promise.

1.2 Propriedades

A lista a seguir descreve as propriedades do objeto.

- **constructor** - função construtora que recebe um callback criando uma função assíncrona.

1.3 Métodos

A lista a seguir descreve os métodos do objeto.

- **then** - permite definir o bloco executado mediante o cumprimento de uma promise.
- **catch** - permite definir o bloco executado mediante a rejeição de uma promise.

1.4 Exemplo

Abaixo estamos representando eventos assíncronos e independentes.

```
function rand(min,max){
    min*=1000;
    max*=1000;
     return Math.floor(Math.random()*(max-min)+min)
}
```

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

```
6
7   function wait(msg, time){
8     setTimeout(()=>{
9         console.log(msg);
10    }, time);
11  }
12
13  wait( 'Evento 1' , rand(1,3));
14  wait( 'Evento 2' , rand(1,3));
15  wait( 'Evento 3' , rand(1,3));
```

Cada evento ocorrerá no seu determinado tempo, porém nesse caso precisamos que ocorram na sequência certa, para isso utilizamos promises.

```
1   function rand(min,max){
2       min*=1000;
3       max*=1000;
4        return Math.floor(Math.random()*(max-min)+min)
5   }
6
7   function wait(msg, time){
8      return  new Promise((resolve,reject)=>{
9         setTimeout(()=>{
10            resolve(msg);
11        }, time);
12    });
13  }
```

Criamos uma função promise com os métodos resolve e reject que são os termos padrões para o caso de a execução funcionar ou não respectivamente. Utilizamos then e catch para definir a sequência de execuções.

```
1   function rand(min,max){
2       min*=1000;
3       max*=1000;
4        return Math.floor(Math.random()*(max-min)+min)
5   }
6
7   function wait(msg, time){
```

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

```
     return   new  Promise((resolve,reject)=>{
        setTimeout(()=>{
            resolve(msg);
        }, time);
    });
}

wait( 'Evento 1' , rand(1,3))
    .then(resposta=>{
        console.log(resposta);
         return  wait( 'Evento 2' , rand(1,3));
    })
    .then(resposta =>{
        console.log(resposta);
         return  wait( 'Evento 3' ,rand(1,3));
    })
    .then( resposta => {
        console.log(resposta);
    })
    . catch ();
```

Dessa forma esperamos o evento um retornar uma resposta, exibimos ele só então acontece a chamada do evento 2 e assim por diante. A função catch acontece quando o reject é retornado.

Vamos definir que se a resposta for diferente de uma string teremos uma rejeição, para exemplificar o erro.

```
function  rand(min,max){
    min*=1000;
    max*=1000;
     return  Math.floor(Math.random()*(max-min)+min)
}

  function  wait(msg, time){
     return   new  Promise((resolve,reject)=>{
         if ( typeof  msg!== 'string' ) reject( 'Bad value' );
        setTimeout(()=>{
            resolve(msg);
        }, time);
```

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

```
    });
}

wait( 'Evento 1' , rand(1,3))
    .then(resposta=>{
        console.log(resposta);
         return  wait(222, rand(1,3));
    })
    .then(resposta =>{
        console.log(resposta);
         return  wait( 'Evento 3' ,rand(1,3));
    })
    .then( resposta => {
        console.log(resposta);
    })
    . catch (e =>{
        console.log( 'ERRO:' , e);
    });
```

Assim na execução do evento 2 temos um erro que leva a execução da função catch e interrupção do restante do código.

```
function  rand(min,max){
    min*=1000;
    max*=1000;
     return  Math.floor(Math.random()*(max-min)+min)
}

  function  wait(msg, time){
     return   new  Promise((resolve,reject)=>{
         if ( typeof  msg!== 'string' ) reject( 'Bad value' );
        setTimeout(()=>{
            resolve(msg);
        }, time);
    });
}

wait( 'Evento 1' , rand(1,3))
    .then(resposta=>{
```

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

```
        console.log(resposta);
         return  wait( 'Evento 2' , rand(1,3));
    })
    .then(resposta =>{
        console.log(resposta);
         return  wait( 'Evento 3' ,rand(1,3));
    })
    .then( resposta => {
        console.log(resposta);
    })
    . catch (e =>{
        console.log( 'ERRO:' , e);
    });
console.log( 'Evento assíncrono' );
```

Qualquer evento fora do promise acontecerá de forma independente no seu próprio tempo.

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

2 Promise.all

Com essa função, podemos criar um conjunto de promises e retornar valor apenas após todas estarem resolvidas.

```
function  rand(min,max){
    min*=1000;
    max*=1000;
     return  Math.floor(Math.random()*(max-min)+min)
}

  function  wait(msg, time){
     return   new  Promise((resolve,reject)=>{
        setTimeout(()=>{
            resolve( -  'Resolvido' );
        }, time);
    });
}

const promises = [
     'Primeiro valor' ,
    wait( 'Promise 1' , 2000),
    wait( 'Promise 2' , 500),
    wait( 'Promise 3' , 4000),
     'Outro valor'
];

Promise.all(promises)
    .then( function (valor){
        console.log(valor);
    })
    . catch ( function (erro){
        console.log(erro);
    })
```

No exemplo temos uma lista com eventos a serem ocorridos numa determinada ordem, após todos acontecerem teremos uma resposta, os passados pela promise retornam como "resolvidos".

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

3 Promise.race

Esse método como o nome diz assemelha-se a uma corrida, onde um conjunto de eventos acontece e recebemos uma única resposta, o evento mais rápido.

```
function  rand(min,max){
    min*=1000;
    max*=1000;
     return  Math.floor(Math.random()*(max-min)+min)
}

  function  wait(msg, time){
     return   new  Promise((resolve,reject)=>{
        setTimeout(()=>{
            resolve( -  'Resolvido' );
        }, time);
    });
}

const promises = [
    wait( 'Promise 1' , 2000),
    wait( 'Promise 2' , 500),
    wait( 'Promise 3' , 4000),
];

Promise.race(promises)
    .then( function (valor){
        console.log(valor);
    })
    . catch ( function (erro){
        console.log(erro);
    })
```

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

4 Async & Await

Uma função async permite utilizar o método await para a mesmo objetivo das promises, fazer com que eventos assíncronos acontecam de forma que suas respostas parecem síncronas.

```
function  rand(min,max){
    min*=1000;
    max*=1000;
     return  Math.floor(Math.random()*(max-min)+min)
}

  function  wait(msg, time){
     return   new  Promise((resolve,reject)=>{
        setTimeout(()=>{
            resolve( -  'Resolvido' );
        }, time);
    });
}
async  function  executa(){
     try {
        const fase 1 = await wait( 'Fase 1' , rand(1,5));
        console.log(fase1);
        const fase 2 = await wait( 'Fase 2' , rand(1,5));
        console.log(fase1);
        const fase 3 = await wait( 'Fase 4' , rand(1,5));
        console.log(fase1);
    } catch (e){
        console.log(e)
    }
}
 executa();
```

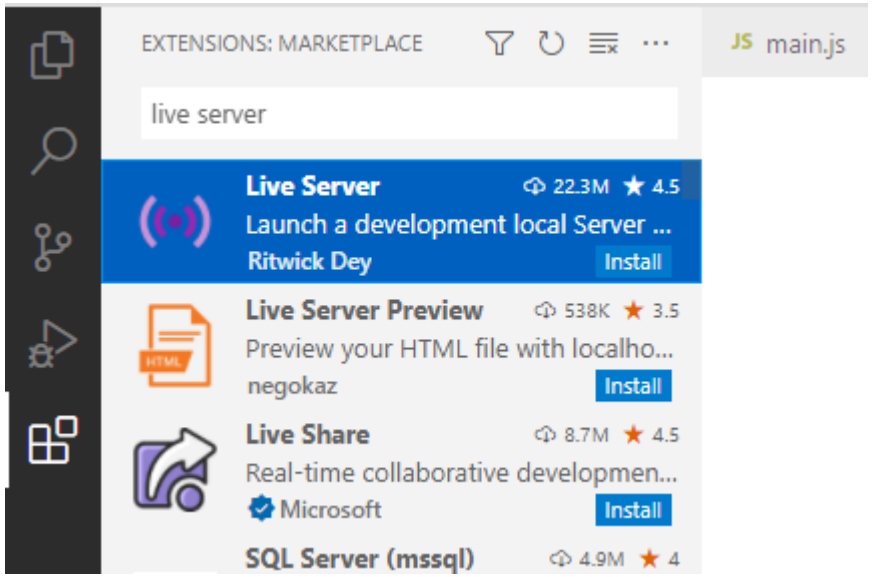
Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

5 Fetch API

Fornece uma interface JavaScript para acessar e manipular partes do pipeline HTTP, tais como os pedidos e respostas. Ela também fornece o método global **fetch**() que fornece uma maneira fácil e lógica para buscar recursos de forma assíncrona através da rede.

Para utilizarmos precisamos de um servidor local, vamos instalar a extensão live server.

Procure pelo extensão e clique em instalar.



Crie um arquivo index.html

```
<!DOCTYPE html>
<html lang= "pt-BR" >

<head>
  <meta charset= "UTF-8" >
  <meta name= "viewport"  content= "width=device-width, initial-scale=1.0" >
  <meta http-equiv= "X-UA-Compatible"  content= "ie=edge" >
  <title>Modelo</title>
```

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

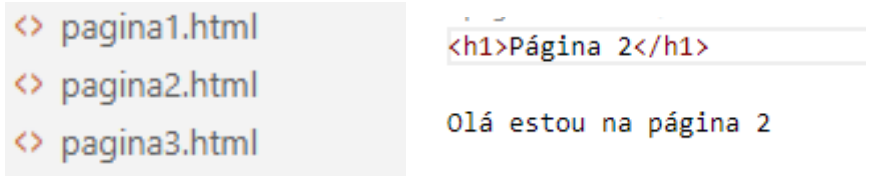
```
  <link rel= "stylesheet"  href= "style.css" >
</head>

<body>

  <section class= "container" >
    <h1>Olá mundo!</h1>
    <a href= "pagina1.html" >Página 1</a>
    <a href= "pagina2.html" >Página 2</a>
    <a href= "pagina3.html" >Página 3</a>


  <script src= "main.js" ></script>
</body>

</html>
```

Em seguida crie o main.js e o clique aqui para baixar o style.css.

Com o arquivo index.html aberto clique em 

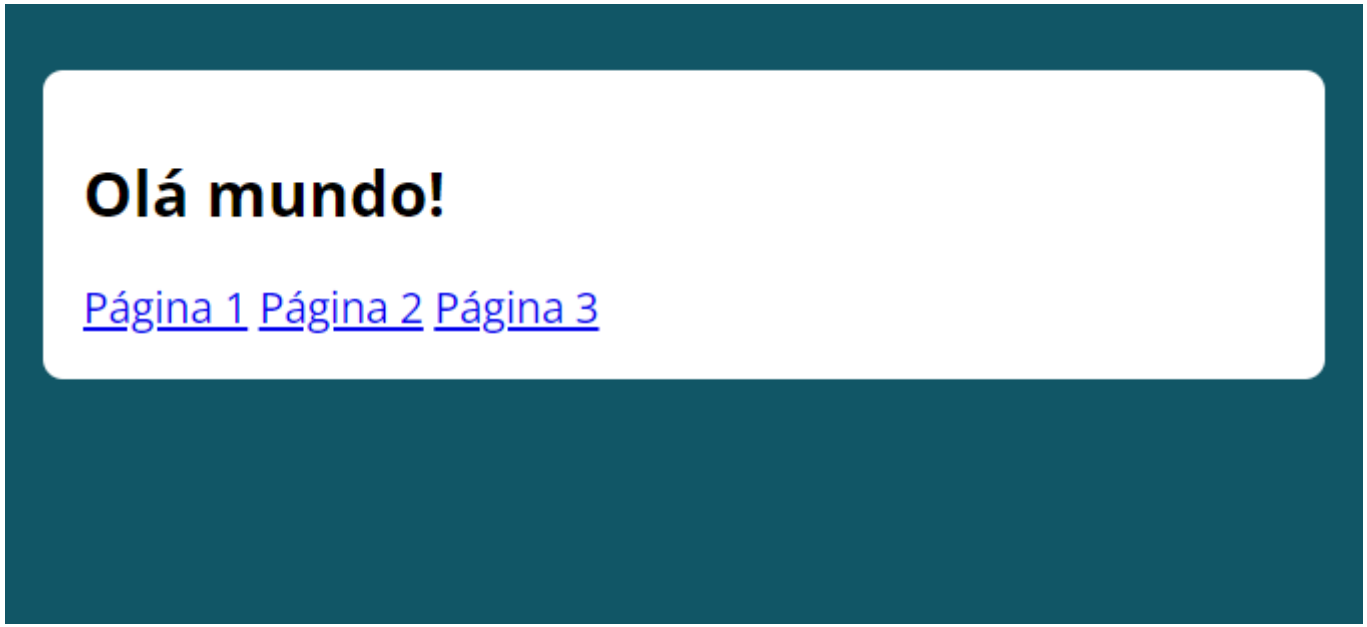
para abrir a página no live server.

Crie novas páginas e adicione algum conteúdo nelas, como no exemplo, para trabalharmos com requisições.



Teremos algo como:

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.



Agora vamos para as funções javascript. Ao clicarmos nos link somos encaminhados para as respectivas página, mas queremos interromper esse evento, para o conteúdo da página seja direcionado para o index através de requisições.

Adicione a div 'resultado' que exibirá o conteúdo.

```
<!DOCTYPE html>
<html lang= "pt-BR" >

<head>
  <meta charset= "UTF-8" >
  <meta name= "viewport"  content= "width=device-width, initial-scale=1.0" >
  <meta http-equiv= "X-UA-Compatible"  content= "ie=edge" >
  <title>Modelo</title>
  <link rel= "stylesheet"  href= "style.css" >
</head>

<body>
```

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

```
13
14    <section class= "container" >
15      <h1>Olá mundo!</h1>
16      <a href= "pagina1.html" >Página 1</a>
17      <a href= "pagina2.html" >Página 2</a>
18      <a href= "pagina3.html" >Página 3</a>
19
20      <div class= "resultado" ></div>
21    </section>
22
23    <script src= "main.js" ></script>
24  </body>
25
26  </html>
```

Primeiramente vamos prevenir a execução padrão dos eventos de click e substituí-lo pela execução de outra função.

```
1  document.addEventListener( 'click' , e => {
2    const el = e.target;
3    const tag = el.tagName.toLowerCase();
4
5     if  (tag ===  'a' ) {
6      e.preventDefault();
7      carregaPagina(el);
8    }
9  });
```

Agora vamos criar a função 'carregaPagina', utilizando async

```
1  async  function  carregaPagina(el) {
2    const href = el.getAttribute( 'href' );  // atribui o valor declarado no atributo href a uma variável
3    const response = await fetch(href);  // Aguarda a execução do evento, no caso o clicar do link, o fetch
   cria automaticamente uma promise
4    const html = await response.text();  // armazena a resposta da requisição em forma de texto (código html)
```

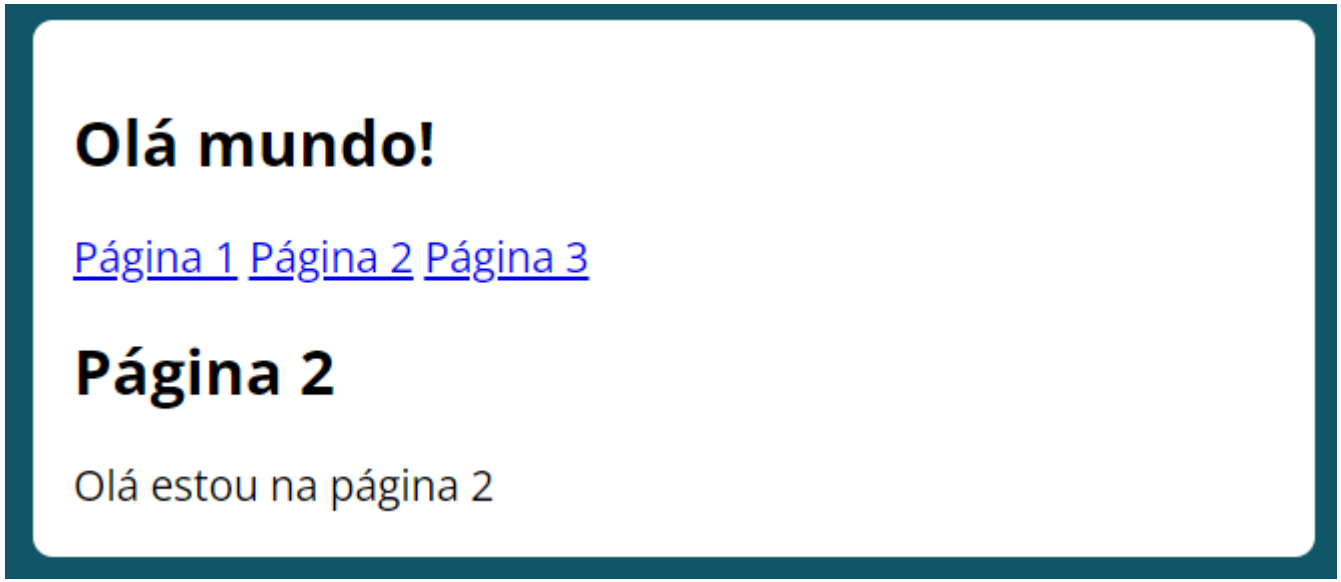
Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

```
  carregaResultado(html);  // Chama a função que exibe o resultado

}
```

Para concluir incluir o conteúdo na div resultado.

```
function  carregaResultado(response) {
  const resultado = document.querySelector( '.resultado' );  // seleciona a div resultado.
  resultado.innerHTML = response;  // muda o conteúdo html da div para a resposta da requisição.
}
```

Assim temos:



Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

6 FetchAPI e Axios (JSON)

Vamos trabalhar com um arquivo JSON contendo dados simbolicamente requisitados de algum banco. Inclua os aquivos pessoas.json e index.html na sua pasta.

Vamos trabalhar um os dados utilizando fetchAPI para exibi-los na página.

```
fetch( 'pessoas.json' )  // esta chamando o dados do arquivo
  .then(resposta => resposta.json())  // esperando a resposta e configurando ela para um formato json
  .then(json => carregaElementosNaPagina(json));  // uma nova promise que chama uma função
```

Assim temos a requisição dos dados e sua resposta agora vamos criar a fução para exibi-los.

```
function  carregaElementosNaPagina(json) {
  const table = document.createElement( 'table' );  // define a criação de uma tabela no html

   for (let pessoa of json) {  // para cada elemento do json "pessoa", vai repetir a execução
    const tr = document.createElement( 'tr' );

    let td1 = document.createElement( 'td' );
    td1.innerHTML = pessoa.nome;
    tr.appendChild(td1);

    let td2 = document.createElement( 'td' );
    td2.innerHTML = pessoa.idade;
    tr.appendChild(td2);

    table.appendChild(tr);
  }

  const resultado = document.querySelector( '.resultado' );
  resultado.appendChild(table);  // adiciona a tabela definida acima na div resultado
}
```

Uma forma mais fácil de fazer o mesmo é utilizando o axios.

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

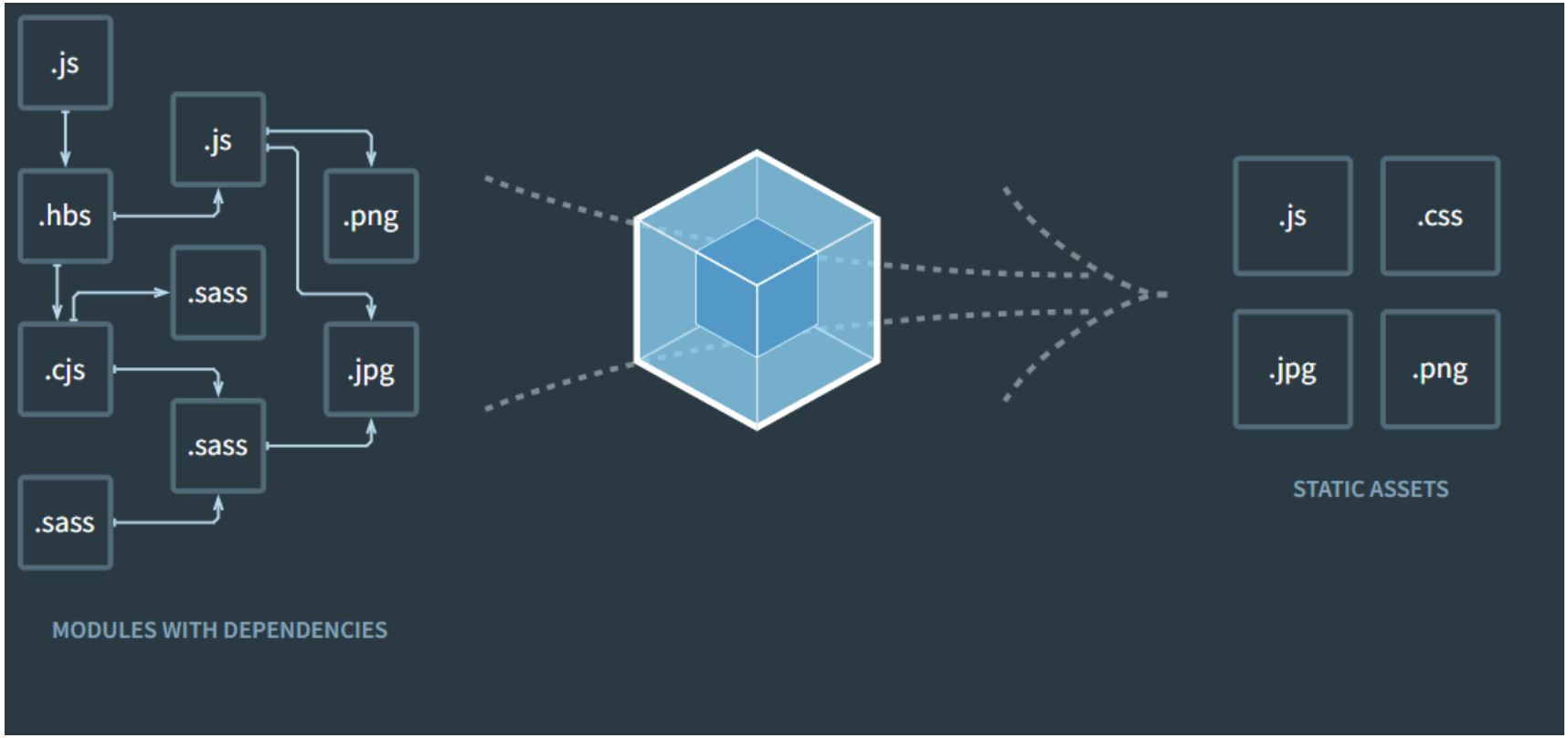
Axios é um cliente HTTP baseado em Promises para fazer requisições. Pode ser utilizado tanto no navegador quanto no Node. js ou qualquer serviço de API. Automaticamente transforma json para data.

Assim temos o mesmo resultado com:

```
axios( 'pessoas.json' )
.then(resposta => carregaElementosNaPagina(resposta.data));

function  carregaElementosNaPagina(json) {
  const table = document.createElement( 'table' );  // define a criação de uma tabela no html

   for (let pessoa of json) {  // para cada elemento do json "pessoa", vai repetir a execução
    const tr = document.createElement( 'tr' );

    let td1 = document.createElement( 'td' );
    td1.innerHTML = pessoa.nome;
    tr.appendChild(td1);

    let td2 = document.createElement( 'td' );
    td2.innerHTML = pessoa.idade;
    tr.appendChild(td2);

    table.appendChild(tr);
  }

  const resultado = document.querySelector( '.resultado' );
  resultado.appendChild(table);  // adiciona a tabela definida acima na div resultado
}
```

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

7 Webpack



O Webpack é um empacotador de módulos gratuito e de código aberto para JavaScript. Ele é feito principalmente para JavaScript, mas pode transformar ativos de front-end, como HTML, CSS e imagens, se os carregadores correspondentes forem incluídos.

Os módulos apresentados pelo programa são estáticos e se voltam para JavaScript moderno. Assim que todas as aplicações são processadas, será possível gerar um gráfico capaz de mapear cada um dos módulos, bem como suas dependências.

Além disso, o webpack pode ser utilizado para facilitar a utilização e troca de informações de um arquivo grande .js. Se todo o código do seu projeto está dentro desse arquivo, será difícil transmitir as suas informações sem enfrentar problemas de legibilidade, tamanho, escopo e facilidade de manutenção.

Para aquelas pessoas que desejam facilitar o andamento de um projeto, a utilização do programa é extremamente viável. Ele permite a combinação de arquivos com completa segurança, além de fazer com que o programador ou programadora não precise se preocupar com a colisão de escopo ou qualquer outro erro de legibilidade.

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

Por meio do webpack, alguns problemas são deixados completamente de lado. Geralmente, para alterar um arquivo, é necessário reconstruir todo o projeto do zero. A partir da concatenação, a reutilização de *scripts* de arquivos fica ainda mais simples. O programa pode ser executado no Node.js em qualquer computador ou servidor fora de um ambiente de navegador.

Junto com babel o webpack é responsavél por garantir que uma o código javascript seja interpretado por todos os compiladores mesmo que esses não reconheçam o ES6 que é a versão mais nova do JS, assim eles convertem o código para versões antigas, para utilizar os módulos do ES6. Para isso faça download dowebpack-boilerplate.zip webpack.

Para utilizarmos as funções do webpack precisamos executar no terminal o comando:

```
PS U:\JS\webpack> npm run dev
```

Assim todo o código JS será convertido para versões anteriores nos arquivos bundle.js de forma que todos os navegadores seriam capaz de interpretá-los.

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

8 ES6 Módulos

ECMAScript 2015 é a especificação mais recente usada para implementar a linguagem JavaScript. Vamos aprender as funções de importação e exportação de elementos js. Veremos os módulos de importação e exportação do ES6.

Dentro da pasta src crie os arquivos index.js e modulo1.js para trabalharmos em cima deles.

Cada módulo possui seus próprios elementos como variáveis, constantes, classes e funções de forma privada, para podermos ter acesso aos mesmos em outros módulos utilizamos as funções *export* e *import*.

modulo1.js - export

```
const nome =  'Maria' ;
const sobrenome =  'João'
const idade = 23

function  soma(x,y){
     return  x+y;
}

export{nome,sobrenome, idade, soma};
```

index.js - import

```
import {nome,sobrenome, idade, soma} from  './modulo1' ;

console.log(soma(2,5));
```

Dessa forma estamos exportando as variáveis e a função de um arquivo e aplicando a função em outro arquivo através da importação.

Podemos também renomear esses dados de forma que não exista conflito. Tanto na importação como na exportação.

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

```
const nome =  'Maria' ;
const sobrenome =  'João'
const idade = 23

function  soma(x,y){
     return  x+y;
}

export{nome as nome2, soma};
```

```
import {nome2,soma as somarf} from  './
modulo1' ;

var  soma = 1+2;
nome =  'Eduardo'

console.log(somaf(2,5));
```

Podemos também declarar a exportação diretamente na criação do elemento.

```
const nome =  'Maria' ;
const sobrenome =  'João'
const idade = 23

export class Pessoa(){
    constructor(nome, sobrenome){
         this .nome=nome;
         this .sobrenome=sobrenome;
    }
}
```

```
import {Pessoa as serHumano} from  './modulo1' ;

p1 =  new  serHumano( 'Maria' , 'Antonieta' );
console.log(p1)
```

Podemos também importar em único elemento todos os dados exportados de determinado módulo.

```
export const nome =  'Maria' ;
export const sobrenome =  'João'
const idade = 23

export class Pessoa(){
    constructor(nome, sobrenome){
         this .nome=nome;
         this .sobrenome=sobrenome;
```

```
import * as modulo from  './modulo1' ;

console.log(modulo)
```

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

```
    }
}
```

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

9 Exercício - Gerador de Senha

Com os conhecimentos adquiridos até o momento crie um gerador de senhas que utilize os tipos e quantidade de caractéres selecionados pelos usuários como no modelo abaixo.

Copyright Robert Bosch GmbH. All rights reserved, also regarding any disposal, exploration, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.