

Módulos, npm e webpack

Programação para Internet I
IFB - Tecnologia em Sistemas para Internet

Marx Gomes van der Linden

Módulos

- Um módulo é um componente de código isolado com a maior parte do conteúdo privado
 - » Não acessível para o restante do código
- Apenas é visível o que o módulo decide **exportar**
- Permite melhor organização
- Reduz conflitos de nomes

Módulos – sintaxe antiga (IIFE)

```
let meumodulo = (function() {  
    function ola(){  
        console.log('Olá, Mundo!');  
    }  
  
    let numeroPrivado = 123;  
  
    return {  
        numero: numeroPrivado + 1,  
        saudacao: function(){  
            ola();  
        }  
    }  
})();  
  
meumodulo.saudacao();  
console.log( meumodulo.numero );
```

Módulo Node (CommonJS)

- Projetado inicialmente para o servidor
- Cada arquivo é um módulo
- Para disponibilizar dados para fora do módulo, use o objeto **exports**
- Para trazer dados de outro módulo, use a função **require(nome)**
 - » Extensão **.js** opcional

CommonJS – Múltiplas exportações

meumodulo.js

```
function ola(){  
    console.log('Olá, Mundo!');  
}  
  
let numeroPrivado = 123;  
  
exports.numero = numeroPrivado + 1  
exports.saudacao = function(){  
    ola();  
}
```

principal.js

```
let mm = require('./meumodulo');  
  
mm.saudacao();  
console.log(mm.numero);  
// console.log(mm.numeroPrivado); ← undefined
```

CommonJS – Exportação padrão

meumodulo.js

```
let numeroPrivado = 123;  
  
function ola(){  
    console.log('Olá, Mundo! ' + numeroPrivado);  
}  
  
module.exports = ola;
```

principal.js

```
let mm = require('./meumodulo');  
  
mm();
```

Módulo nativo (ECMAScript)

- Requer execução em um servidor
 - » URL iniciando com **http://** e não **file://**
- Use **type="module"** na tag script
- Exportar:
 - » Modificador **export** antes da declaração (**let**, **const**, **function**)
- Importar:
import { nome1, nome2...} **from** 'modulo';

Módulo nativo – Múltiplas exportações

meumodulo.js

```
function ola(){  
    console.log('Olá, Mundo!');  
}  
  
let numeroPrivado = 123;  
  
export let numero = numeroPrivado + 1;  
export function saudacao(){  
    ola();  
}
```

principal.js

```
import { numero, saudacao } from './meumodulo.js';  
  
saudacao();  
console.log(numero);  
//console.log(numeroPrivado); ← erro
```


npm

- Node Package Manager
 - » <https://www.npmjs.com/>
 - » Coletânea online de pacotes de código Javascript e ferramentas para utilizá-los
- Parte integrante do Node.js
- Ferramenta de linha de comando: **npm**

Instalando pacotes

– `npm install nomedopacote`


- » Instala o pacote localmente (para o projeto atual)
- » Pacotes vão para o diretório `node_modules`
- » Opção `--save` ou `--save-dev`
 - › Atualiza o arquivo `package.json`

– `npm install -g nomedopacote`

- » Instala globalmente (para todo o sistema)

Executando pacotes

- Se o pacote inclui uma ferramenta de linha de comando, você pode executá-la com o comando npx:
- **npx *nomedopacote***
 - » Ao invés disso, você pode também procurar o nome do executável instalado no sistema




```
C:\Users\visitante>npm install -g browserify
C:\Users\visitante\AppData\Roaming\npm\browserify -> C:\Users\visitante\
AppData\Roaming\npm\node_modules\browserify\bin\cmd.js
+ browserify@16.5.0
added 137 packages from 109 contributors in 16.712s
```

Arquivo package.json

- Contém todas as informações sobre um pacote, incluindo suas dependências

```
{
  "name": "teste",
  "version": "1.0.0",
  "description": "Um pacote de teste",
  "main": "teste.js",
  "dependencies": {
    "uglify-js": "^3.6.2",
    "underscore": "^1.9.1"
  },
  "devDependencies": {
    "webpack": "^5.12.1",
    "webpack-cli": "^4.3.1"
  },
  "author": "Fulano de tal",
  "license": "ISC"
}
```



--save

--save-dev

Criando o `package.json`

- Uma maneira de criar o `package.json` é executando:

- **`npm init`**

- » O arquivo será criado de acordo com os campos preenchidos

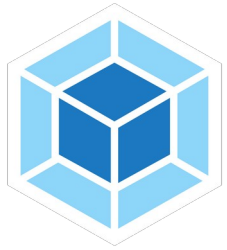
Instalando pacotes

- Para instalar todas as dependências especificadas no package.json, basta executar:
- **npm install**



webpack

- Solução completa para o empacotamento de módulos para uso no navegador
 - » JavaScript, CSS/SASS e imagens
 - » Todos os tipos de módulos JavaScript
- Gerenciamento automático de dependências
- Poderoso e configurável



webpack

- Para instalar localmente:

```
npm install --save-dev webpack webpack-cli
```

- Para ejecutar:

```
npx webpack
```


Estrutura padrão do projeto

✓ dist

JS main.js

Código gerado pelo webpack a partir do index.js + dependências

> node_modules

✓ src

JS index.js

Código escrito pelo desenvolvedor

{ } package-lock.json

{ } package.json

webpack.config.js

Configuração do Webpack (opcional)

No HTML:

```
<script src="dist/main.js" defer></script>
```

src/index.js

```
let prettyMilliseconds = require('pretty-ms');

module.exports = {
  minhaFuncao: function(){
    $('#resultado').text(
      prettyMilliseconds(
        parseInt($('#texto').val())
      )
    );
  },

  outraFuncao: function(){ /* ... */ }
}
```

No arquivo HTML:

```
<script src="dist/main.js" defer></script>
```

webpack.config.js

```
const path = require('path');  
  
module.exports = {  
  output: {  
    library: 'exemplo'  
  }  
};
```

No arquivo HTML:

```
<button onclick="exemplo.minhaFuncao()">
```

Mais opções

```
const path = require('path');
```

```
module.exports = {
```

```
  entry: {
```

```
    main: './src/meucodigo.js'
```

```
  },
```

```
  output: {
```

```
    filename: 'testeprod.js',
```

```
    library: 'exemplo'
```

```
  }
```

```
};
```

Arquivo de entrada
(ao invés de index.js)



Arquivo de saída
(ao invés de main.js)