Webpack
que faz sentido!

um e-book ***produzido pela:***

CodePrestige

Agradecimentos

Seja muito bem-vindo!

O Webpack é uma das ferramentas mais famosas e utilizadas no mundo do front-end, no entanto, também é uma das mais confusas. Para os programadores de primeira viagem, a configuração da ferramenta não tem pé nem cabeça. Não preocupe que com este e-book, o Webpack finalmente fará sentido e você terá a chance de finalmente dizer: *“ahhh, agora entendi!”*.

Boa leitura e bons códigos!

Diego Martins de Pinho
Fundador da Code Prestige
@DiegoPinho

Sumário

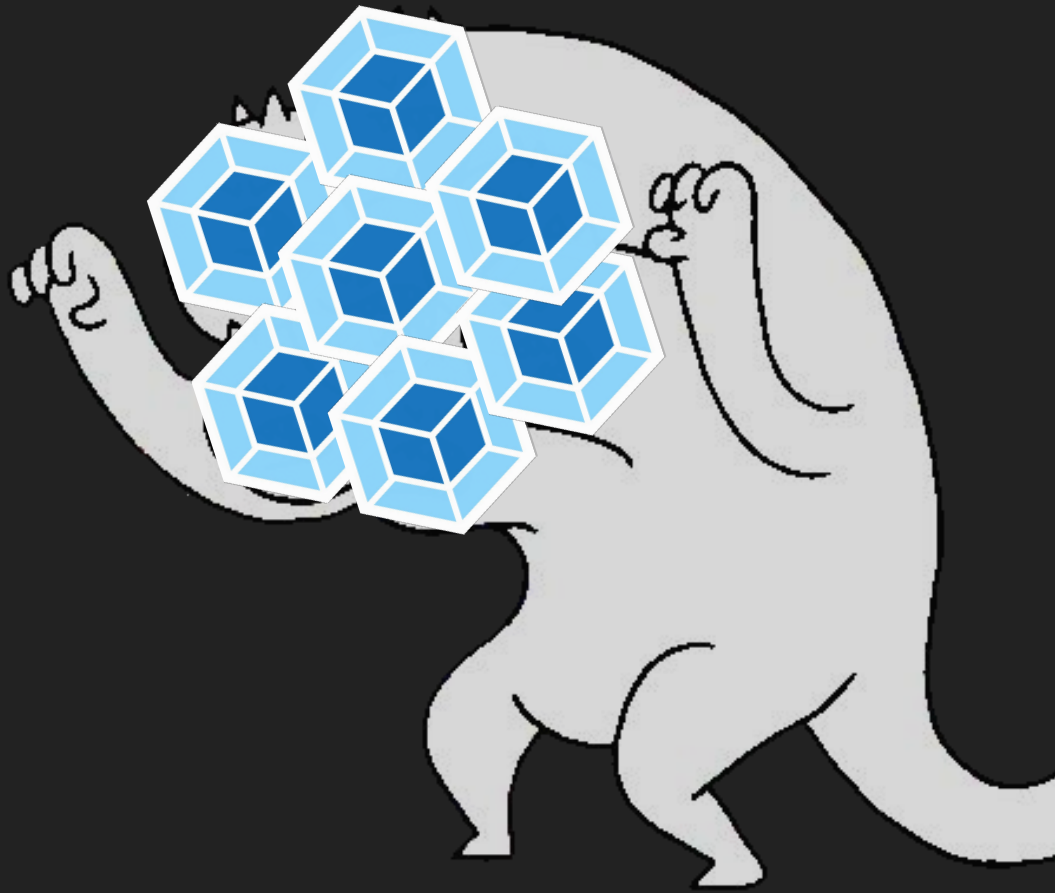
- O que raios é o Webpack?
- Por que utilizá-lo?
- Como instalar?
- Partindo do zero
- Como é no mundo real

O que raios é o Webpack?

O Webpack é o que chamamos de **Module Bundler.**

Em resumo, o que ele faz é pegar todos os módulos do seu projeto (sejam criados por você ou como dependência) e gera arquivos estáticos.

Veremos que ele não é nenhum **bicho de sete cabeças.**



O Webpack lê o ponto de entrada (entry point) e avalia suas dependências, assim como as dependências das dependências.

cursos.js

```
var cursos = ['ES6', 'ES7', 'ES8'];
module.exports = cursos;
```

app.js

```
var cursos = require('./cursos.js');
console.log(cursos);
```

bundle.js

```
!function(r){function
t(o){if(n[o])return
n[o].exports; var
e=n[o]={i:o,
l:!1,exports:{}};return
r[o].call(e.exports,e,e.exp
orts,t),
e.l=!0,e.exports}([function
(r,t){var cursos=['ES6',
'ES7', 'ES8'];
r.exports=n}]),console.log(
cursos)])};
```

O Webpack então reúne todos estas dependências em um único arquivo.

Por que utilizá-lo?

Há uma série de *bundlers* no mercado (como o Browserify), mas o Webpack possui algumas vantagens bem legais.

Parsing Inteligente

O parseamento é muito inteligente, de forma que conseguimos alcançar os arquivos desejados sem se matar com RegEx.

Code Splitting

Nos permite dividir o código em vários pontos de forma que não seja necessário sempre baixar todo o conteúdo em todas as páginas.

Plugins

Além disso, é possível inserir mais funcionalidades através de plugins.

Loaders

O Webpack possui uma série de peças em que podemos encaixar para que ele interprete vários tipos de arquivos diferentes, por exemplo: JSX, SASS, PUG.

Como utilizá-lo?

o que é necessário?  

execute o comando no terminal:

```
npm install -g webpack webpack-cli webpack-serve
```

Isso fará com que o pacote seja instalado de forma global na sua máquina.

webpack e o seu utilitário para linha de comando (cli)

Este pacote é completamente opcional, mas altamente recomendável.

** se desejar, é possível deixar somente dependência do projeto com o --save-dev

Partindo do Zero

A melhor maneira de aprender a usar, é vendo como utilizá-la partindo do zero. Para isso, vamos considerar:

Projeto npm do zero

Para isso basta executar o comando **npm init.** Feito isso, haverá somente o arquivo **package.json**.

Arquivo HTML

Crie um arquivo index.html com uma estrutura básica de qualquer página HTML (head, body).

Arquivo JavaScript

Crie um arquivo index.js vazio.

Configurando o HTML

Como o Webpack compacta tudo em um único arquivo, precisamos importá-lo para dentro do HTML.

Vamos chamar este arquivo de **bundle.js**

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>WebPack</title>
  </head>
  <body>
    <script src="bundle.js"></script>
  </body>
</html>
```

Partindo para o JS

Para entender como utilizar as dependências, vamos baixar o jQuery e incluí-lo no **index.js** para manipular o body do nosso arquivo HTML.

```
let $ = require('jquery');

const cursos = ['ES6', 'ES7', 'ES8'];

cursos.forEach(curso => {
  $('body').append('<h1>'+ curso +'</h1>');
});
```

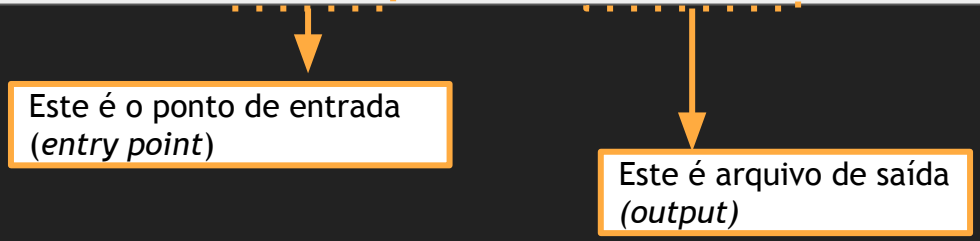
Não se esqueça de incluir o jQuery como dependência do seu projeto! Faça isso com:

```
npm install --save jquery
```

Webpack em ação

execute o comando no terminal (na raiz do projeto):

```
webpack index.js --output bundle.js
```



A mágica acontece!

Por debaixo dos panos, o Webpack pegou o index.js e o jQuery e colocou tudo dentro do **bundle.js**.

O navegador então carregou esse arquivo, já que tínhamos especificado no **index.html**.

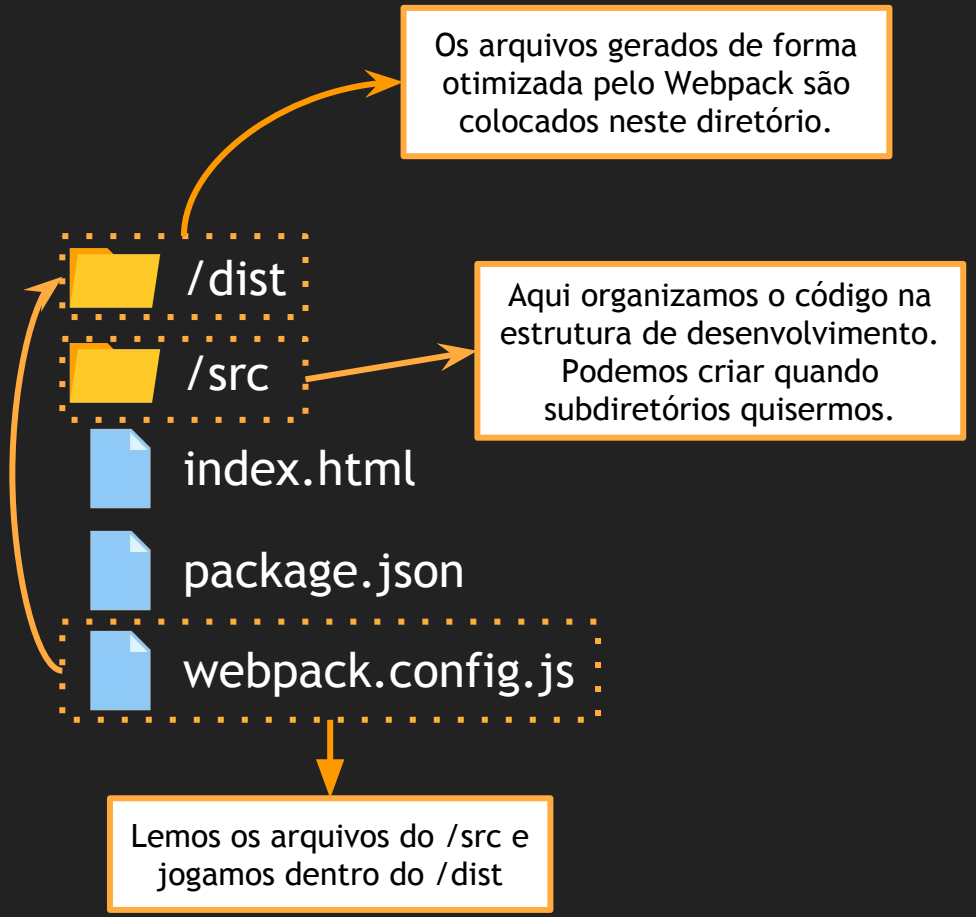
Como é no mundo real

Na vida real é sempre diferente...

Em projetos no mundo real, nós dividimos os arquivos em diversos subdiretórios.

Além disso, não utilizamos o webpack diretamente no terminal, configuramos tudo em um arquivo chamado **webpack.config.js**.

Geralmente temos uma estrutura muito semelhante a esta ao lado.



webpack.config.js

Vamos entender a estrutura básica

entry: arquivo de entrada do projeto
output: possui duas propriedades: ***path*** e ***filename***. na primeira dizemos o diretório onde ele deve salvar e no segundo o nome do arquivo.

modules: para cada loader que usamos, precisamos especificar regras (**rules**). neste exemplo, buscamos por arquivos js para usar o babel-loader e no outro arquivos css para usar o css-loader.

** estão aqui só como exemplo*

```
const path = require('path');
const config = {
  entry: './index.js',
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist')
  }
  module: {
    rules: [
      {
        test: /\.js$/,
        exclude: /node_modules/,
        use: [ 'babel-loader' ]
      },
      {
        test: /\.css$/,
        use: [ 'style-loader', 'css-loader' ]
      }
    ]
  }
};
module.exports = config;
```

webpack.config.js

```
module:{
    rules: [
      {
        test: /\.js$/,
        exclude: /node_modules/,
        use: [ 'babel-loader' ]
      },
      {
        test: /\.css$/,
        use: [ 'style-loader', 'css-loader'
]
      }
    ]
  }
}
```

O que significa?

Entenda o que diz as expressões

1. Busque por todos os arquivos terminados em .js
2. Não mexa na pasta /node_modules
3. Aplique o loader chamado “babel-loader”

1. Busque por todos os arquivos terminados em .css
2. Aplique os loaders chamados “style-loader” e “css-loader”

package.json

Vamos entender a estrutura básica

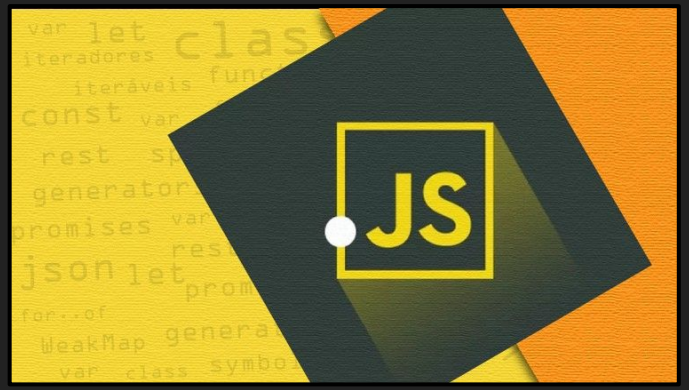
Neste arquivo costumamos definir no script **start** (ou **start:dev**) um servidor de desenvolvimento. Pra isso, usamos o **webpack-serve** (por padrão, ele já busca pelo webpack.config.js e sobe na porta 8000).

Para produção, executamos o webpack no script build.

O restante é bem semelhante a qualquer projeto com npm.

```
{
  "main": "app.js",
  "scripts": {
    "start:dev": "webpack-serve",
    "build": "webpack"
  },
  "author": "Code Prestige",
  "license": "ISC",
  "dependencies": {
    "jquery": "^3.2.1"
  },
  "devDependencies": {
    "babel-core": "^6.24.1",
    "babel-loader": "^6.4.1",
    "babel-preset-es2015": "^6.24.1",
    "css-loader": "^0.28.0",
    "style-loader": "^0.16.1"
  }
}
```

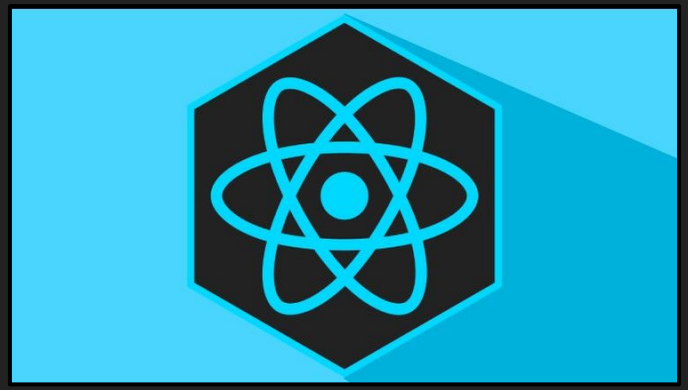
Conheça os nossos cursos online!



Entendendo o ECMAScript 6

Aprenda tudo sobre o ES6 no curso mais completo do tema baseado no livro ECMAScript 6 - Entre de cabeça no futuro do JavaScript publicado pela Casa do Código.

+900 alunos | +6 horas | Exercícios | Resumos



React 16 Definitivo

A última versão do React é uma reescritura do core da biblioteca com uma série de novas funcionalidades, mais performance e melhorias. Saiba como aproveitá-la ao máximo.

+5500 alunos | Exercícios | Resumos | Certificado

Gostou do material?

Se você gostou deste material gratuito, pedimos para que compartilhe com os seus colegas de trabalho e ajude a disseminar o conhecimento!

Dúvidas? Sugestões? Críticas?
Fale conosco através do e-mail:

contato@codeprestige.com.br

Nos acompanhe nas redes sociais para mais conteúdo!

CodePrestige
Ensino de programação à distância

Considere apoiar nosso projeto
https://apoia.se/codeprestige

Confira outros e-books, vídeos e cursos nas nossas redes sociais!