BACK TO BASICS

ENVIRONNEMENT DE DÉVELOPPEMENT

BIEN CHOISIR SES OUTILS

NODE & NPM

```
node -v
```

```
npm -v
```

ANGULAR-CLI

```
npm install -g @angular/cli
```

```
ng -v
```

CHROME

Mais testez sur Firefox !

et IE...

VSCODE

Download

GIT INTEGRATION

RACCOURCIS UTILES

EXTENSIONS

- eslint
- tslint

PACKAGING

PACKAGING ?

TODO: le packaging est bon pour les développeurs mais pas pour les utilisateurs

"OLD SCHOOL" PACKAGING

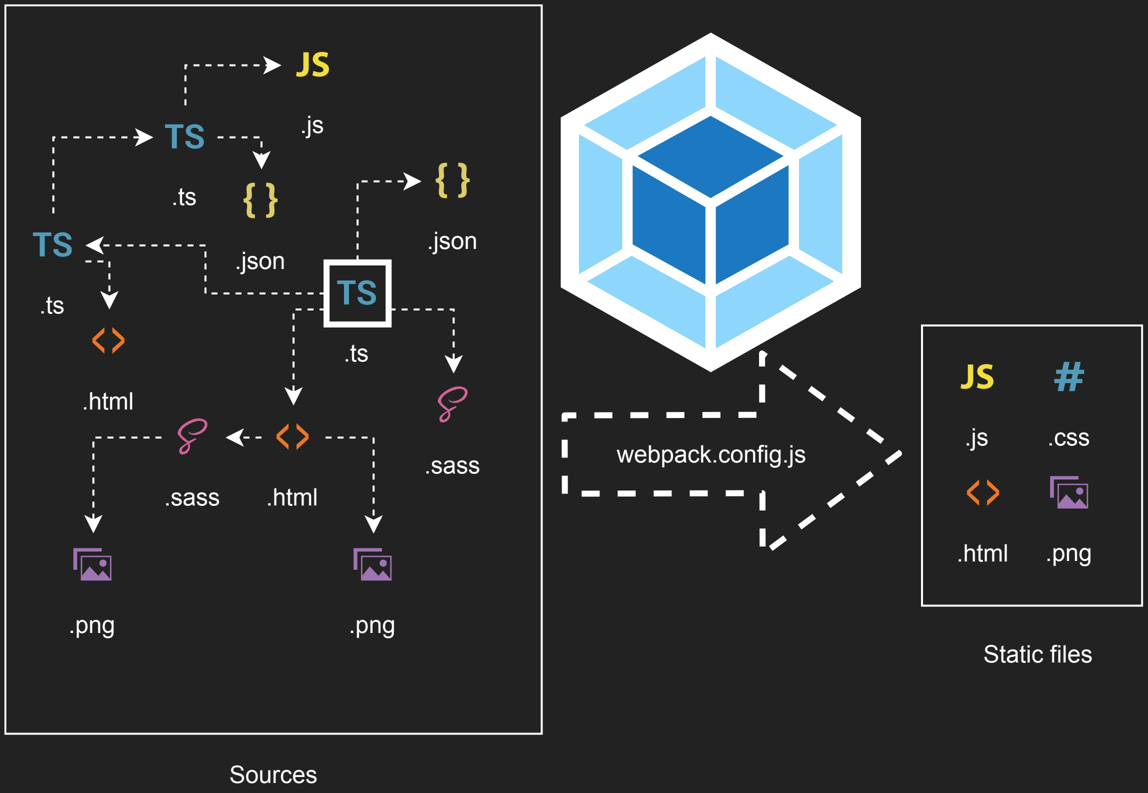
Taskrunners

"OLD SCHOOL" PACKAGING

WEBPACK

Welcome module bundlers!

WEBPACK


JS
.js
TS
.ts
{ }
.json
TS
.ts
{ }
.json
TS
.ts
.html
.sass
.html
.sass
.png
.png
Sources
webpack.config.js
JS
.js
#
.css
.html
.png
Static files

INSTALLATION DE WEBPACK

```
npm init -y
```

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

HELLO, WEBPACK!

```
// src/index.js
console.log('Hello, Webpack!');
```

NPM RUN BUILD

```
"scripts": {
  "build": "webpack"
  ...
}
```

WEBPACK.CONFIG.JS

```
module.exports = {
  mode: process.env.NODE_ENV ? process.env.NODE_ENV : 'develop
};
```

INSTALLATION DE WEBPACK-DEV-SERVER

```
npm install --save-dev webpack-dev-server
```

NPM START

```
"scripts": {
  "start": "webpack-dev-server",
  ...
}
```

INSTALLATION DE HTML-WEBPACK-PLUGIN

```
npm install --save-dev html-webpack-plugin@webpack-contrib/htm
```

WEBPACK.CONFIG.JS

```
const HtmlWebpackPlugin = require('html-webpack-plugin');
module.exports = {
  ...
  plugins: [new HtmlWebpackPlugin({
    template: 'src/index.html',
  })]
};
```

MODULES - DEFAULT EXPORT

```
// src/hello.js
export default 'Hello, Webpack!';
```

```
// src/index.js
import hello from './hello';
console.log(hello);
```

MODULES - NAMED EXPORT

```
// src/mylib.js
export function add(a, b) {
  return a + b;
}
```

```
// src/index.js
import { add } from './mylib';
console.log(add(19, 23));
```

```
// src/index.js
import { add as addNumbers } from './mylib';
console.log(addNumbers(19, 23));
```

AJOUT D'UN TYPE DE FICHIER

Le JavaScript est géré par défaut par webpack

```
import './style.css';
```

Il faut ajouter des *Loaders* pour gérer les autres types de fichiers

AJOUT DE RÈGLES DE CHARGEMENT

```
npm install --save-dev style-loader css-loader
```

```
module.exports = {
  ...
  module: {
    rules: [
      { test: /\.css$/, use: ['style-loader', 'css-loader'] },
    ]
  },
  ...
};
```

UN PEU D'EXERCICE !

Ajouter `file-loader` pour charger des images

```
module.exports = {
  ...
  { test: /\.(gif)$/i, use: 'file-loader' },
  ...
};
```

DEBUGGING

```
module.exports = {
  devtool: 'source-map',
  ...
};
```

2017
2016
2015
ES

ECMASCRIPT

ECMASCRIPT

ECMAScript est la norme du JavaScript

Évolution rapide depuis quelques années :

- classes
- modules
- Array methods
- Arrow functions
- Template literals
- Destructuring
- ...

LET'S TRY ECMASCRIPT

```
npm install --save-dev eslint
```

```
node_modules/.bin/eslint --init
```

```
"scripts": {
  "lint": "eslint ./src"
}
```

```
import books from './books';
```

CLASSES

```
export class Book {
  constructor(args) {
    this.id = args.id;
    this.name = args.name;
    this.authors = args.authors;
    this.releaseDate = new Date(args.releaseDate);
  }
}
```

ARRAY METHODS

```
books.map(function (book) {
  return new Book(book);
});
```

```
books.filter(function (book) {
  return book.name.startsWith('The');
});
```

```
books.sort(function (a, b) {
  return a.localeCompare(b);
});
```

ARROW FUNCTIONS

```
books.map(book => new Book(book));
```

```
books.filter(book => book.name.startsWith('The'));
```

```
books.sort((a, b) => a.localeCompare(b));
```

TEMPLATES

```
'Book ' + book.name + ' by ' + book.authors.join()
```

```
`Book ${book.name} by ${book.authors.join()}`
```

DESTRUCTURING

```
const { name, releaseDate } = book;
```

```
const { name = 'Untitled', releaseDate } = book;
```

```
const { name = 'Untitled', releaseDate, ...rest } = book;
```

PROMISE

Timeout

ASYNC/AWAIT

INDEXEDDB

? / localforage

Pas entièrement supporté par tous les navigateurs :(

BABEL

Compilateur JavaScript pour utiliser tous les standards ECMAScript!

INSTALLATION DE BABEL

```
npm install --save-dev babel-loader babel-core babel-preset-en
```

```
npm install babel-polyfill
```

```
{
  "presets": ["env"]
}
```

```
module.exports = {
  entry: ['babel-polyfill', './src/index.js'],
  ...
  { test: /\.js$/, exclude: /node_modules/, use: 'babel-loader
  ...
};
```

SÉLECTION DES NAVIGATEURS CIBLÉS

```
{
  "presets": [
    ["env", {
      "targets": ["last 2 versions", "not ie <= 10"]
    }]
  ]
}
```

TYPESCRIPT

TYPESCRIPT

- Annotations de types
- Proche du JavaScript
- Compatible avec les bibliothèques JavaScript

TYPES

```
const name: string = 'Hello';
```

```
const book: Book = new Book(...);
```

```
function read(books: Book[]): void { ... }
```

```
const foo: any = { bar: 'baz' };
```

LET'S TRY TYPESCRIPT!

```
npm install --save-dev tslint
```

```
node_modules/.bin/tslint --init
```

```
"scripts": {
  "lint": "tslint ./src"
}
```

```
npm install --save-dev typescript ts-loader
```

```
module.exports = {
  ...
  entry: ['babel-polyfill', './src/index.ts'],
  module: {
    rules: [
      { test: /\.ts$/, use: ['babel-loader', 'ts-loader'] },
      ...
    ]
  },
  resolve: {
    extensions: ['.ts', '.js']
  },
  ...
};
```

```json
{
  "compilerOptions": {
    "outDir": "./dist/",
    "sourceMap": true,
    "target": "es6"
  }
}
```

```
export class Book {
  id: string;
  name: string;
  authors: string[];
  releaseDate: Date;

  constructor(args) {
    this.id = args.id;
    this.name = args.name;
    this.authors = args.authors;
    this.releaseDate = new Date(args.releaseDate);
  }
}
```

```
declare module "*.json" {
  const value: any;
  export default value;
}
```

ajouter les décorateurs

configurer le linter

types non gérés par typescript

interfaces

mixins

Web components (if we have time)