

TIW 8

Technologies Web synchrones et multi-dispositifs

CM1 - Rappels Stack Javascript

<https://aurelient.github.io/tiw8/2025/>

Plan

- ▶ Introduction au cours
- ▶ L'informatique Ubiquitaire
- ▶ **Rappels Stack Javascript du cours**

Node.js



Plateforme d'exécution basée sur le moteur Chrome

Permet de :

- ▶ mettre en place de “grosses” applications en **JS** et **modulaires**
- ▶ programmer dans le **même langage** côté serveur et côté client

Npm



Gestionnaire de paquets pour node

S'utilise “à la” apt-get

- ▶ localement : `npm install <package>`
- ▶ globalement : `npm install -g <package>`

S'initialise avec `npm init` qui créé un `package.json`, fichier décrivant l'application

Description de l'application dans un fichier `package.json`



Gestionnaire de paquets pour node

- ▶ `yarn add <package>`

Déterministe dans l'installation des paquets

Plus rapide (parallélisation)

Mise en cache

Compatibilité avec npm (Yarn2 peut la casser avec plugnplay)

Express

Framework Web pour node.js peu contraignant.

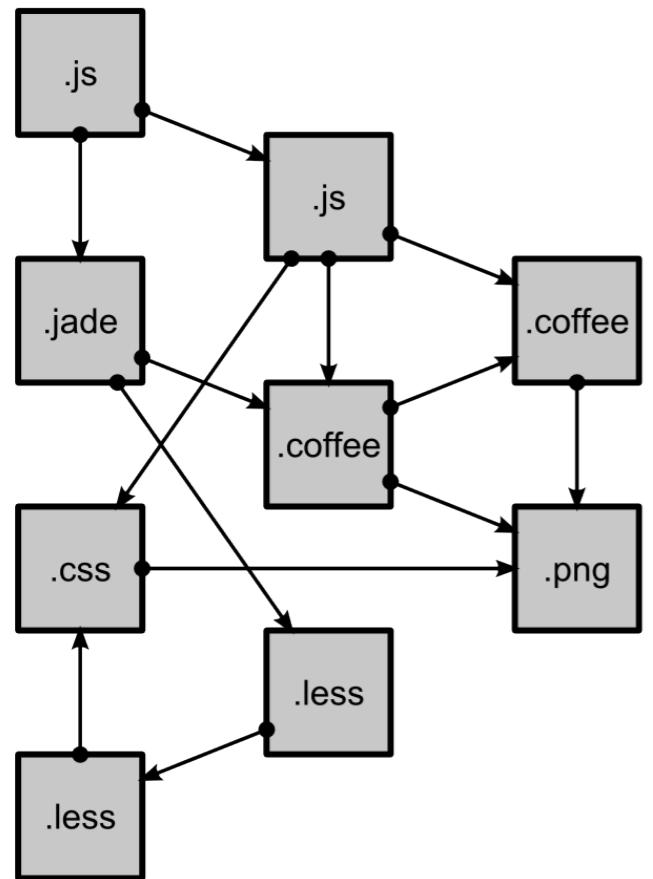
- ▶ Routage
- ▶ Rapide
- ▶ Négociation de contenu (html, pdf, json, ...)
- ▶ HTTP helpers (redirection, cache, etc)
- ▶ Compatible avec de nombreuses bibliothèques de templating

React

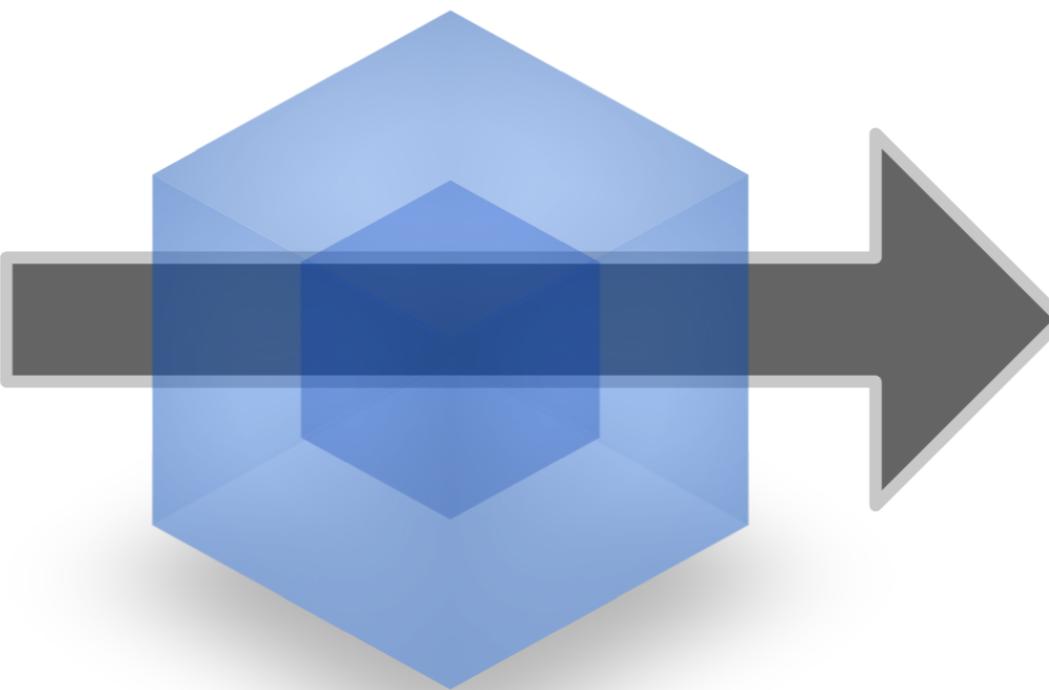
Framework JS

- ▶ DOM Virtuel
- ▶ Basés sur des composants
- ▶ Gestion des Vues + routeur + états
- ▶ ReactNative pour le mobile
- ▶ Redux pour la gestion avancée des états

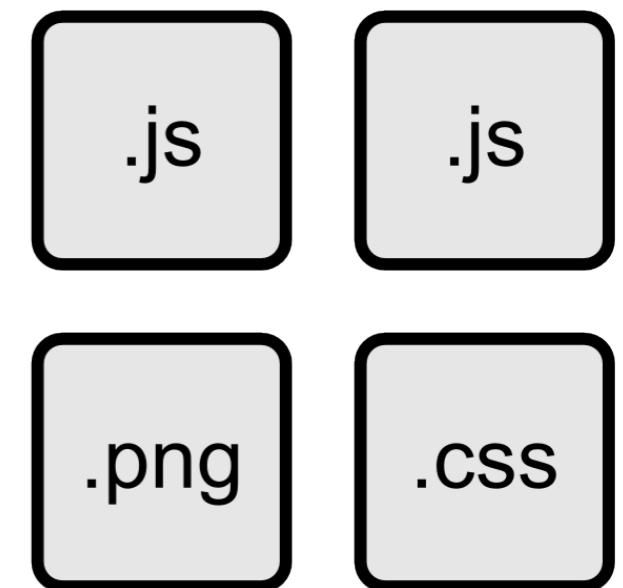
Webpack



modules
with dependencies



webpack
MODULE BUNDLER



static
assets

Webpack

- ▶ Permet la gestion de module (require / import)
- ▶ Agnostique au type de modules : AMD, UMD, CommonJS etc.
- ▶ Traite les assets statiques comme des modules (CSS, images)
- ▶ Découpe le code pour optimiser le chargement
- ▶ Injection de dépendance et multi-compilation
- ▶ S'intègre aux autres outils de build (Grunt/Gulp/etc)

Webpack

Write your code

app.js

```
import bar from './bar';

bar();
```

bar.js

```
export default function bar() {
  //
}
```

Bundle with webpack

webpack.config.js

```
module.exports = {
  entry: './app.js',
  output: {
    filename: 'bundle.js'
  }
};
```

page.html

```
<html>
  <head>
    ...
  </head>
  <body>
    ...
    <script src="bundle.js"></script>
  </body>
</html>
```

Then run `webpack` on the command-line to create `bundle.js`.

Webpack et debug

Problème : tout est optimisé / minifié

Solution : création d'une source-map liant code bundlé au code "réel"

Des outils de dev :

<https://react.dev/learn/react-developer-tools>

Vite

Bundler plus léger que Webpack

- ▶ Plusieurs modules basés sur les routes de l'application
- ▶ Tree shaking par défaut
- ▶ Build plus rapide
- ▶ Hot reload par défaut
- ▶ Plus facile à prendre en main

Webpack reste utile sur de gros projets

Autres outils JS

Tests : Jest

Vérification de code : ESLint

Compilation

- ▶ ES6 → ES5 : Babel
- ▶ TypeScript → ES5 : tsc

Gestion des styles : Tailwind

Bibliothèques de composants : shadcn/ui, MUI...

Déploiement : VM