Les Modules et Bundler

Sur un navigateur on ne peut pas utiliser `npm` ou `nodejs`, le `package.json` ne sert à rien ...

Heureusement, il éxiste aujourd'hui des `bundlers`. Ce sont des outils qui permettent d'utiliser `nodejs` et `npm` sur votre navigateur !

Ce sont des outils qui s'occupe de « compilé », « offusqué », « optimiser » touts vos fichiers css, html, json, images etc ...

Aujourd'hui incontournable ! Toutes les applications web les plus connus utilise un `bundler`.
C'est ce qui nous permet d'installer des libraries, d'utiliser et d'optimiser le css, ou même d'utiliser d'autre langages de programmation comme python, typescript, sass, stylus etc ...

Il en existe beaucoup :

- WebPack
- Bun : C'est aussi un compilateur javascript
- Parcel

Utiliser `Parcel`

Parcel est un `bundler` très puissant sans aucune configuration nescessaire, il comprend tout seul comme un grand toutes les technologies web moderne (typescript, coffee, sass) ...

Installer `parcel`

Dans un projet javascript (un dossier avec un `package.json`) installer parcel :

```
$ npm i parcel
```

> pour certains utilisateurs windows il vous faudra installer tout le dev kit :
>
> - Python : https://apps.microsoft.com/store/detail/python-310/9PJPW5LDXLZ5?hl=en-us&gl=us
> - Build Tools : https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=BuildTools
> - Visual Studio : https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community
> - lancer la commande dans un terminal : `npm config set msvs_version 2017`

> Si le problème persiste vous pouvez aussi désinstaller node et essayer avec un version plus ancienne de nodejs :
>
> - Version 16 : https://nodejs.org/dist/v16.16.0/node-v16.16.0-x64.msi
> - Version 14 : https://nodejs.org/dist/v14.19.0/node-v14.19.0-x64.msi

Utiliser parcel

Pour utiliser parcel il faut tout d'abord une page html :

```
<!-- index.html -->
<!DOCTYPE html>
<html>
  <head>
    <title>Mon Site</title>
  </head>
  <body>
    <h1>Coucou les amis</h1>
  </body>
</html>
```

Il faut pour utiliser parcel, lancer la commande suivante :

```
$ npx parcel index.html
```

> Très important, lorsqu'on lance la commande `npx parcel index.html` on lance un serveur ! C'est à dire que notre terminal n'est plus disponible. Je dois appuyer sur les touches `CTRL-C`.

Ajouter du javascript

Pour ajouter un fichier javascript rien de plus simple :

```
// src/index.js
console.log('Coucou javascript')
```

```
<!-- index.html -->
<!DOCTYPE html>
<html>
  <head>
    <title>Mon Site</title>
  </head>
  <body>
    <h1>Coucou les amis</h1>
    <script type="module" src="src/index.js">
  </body>
</html>
```

Séparer son code en module

Sur une véritable application web, nous n'avons pas qu'un seul fichier javascript. Il est possible de séparer son code en plusieurs fichier javascript. Chaque fichier javascript est ce que l'on appel un **module**.

Dans un module nous pouvons choisir d'exporter certains membres (de les rendre accessible aux autres fichiers). Toujours dans un module nous pouvons importer d'aures membres d'autre fichier.

On utilise pour cela les mot clefs `import` et `export`

Exemple :

```
// src/fichier1.js

// Nous exportons la fonction pour d'autre fichier javascript !
export function additionner(x, y) {
  return x + y
}
```

```
// src/index.js

// Je peux importer ma fonction additionner
import { additionner } from './monfichier1'

console.log(additionner(3, 4))
```

> De base tout ce que vous écrivez dans un fichier est **privée**, c'est à dire que c'est uniquement disponible pour le fichier

Les `export` marchent avec les fonctions mais aussi avec les constantes et les variables :

```
// src/fichier1.js

export const PI = 3.14

// Nous exportons la fonction pour d'autre fichier javascript !
export function additionner(x, y) {
  return x + y
}
```

```
// src/index.js

// Je peux importer ma fonction additionner
import { additionner, PI } from './monfichier1'

console.log(additionner(3, 4))
console.log(PI)
```

Les `import` et `export` par défaut

Il éxiste un autre type d'export et d'import, c'est celui par défaut.

Exemple :

```
// calcule.js
export default function calculer(x, y) {
  return x + y
}
```

```
// index.js
import calculer from 'calculer'

console.log(calculer(12, 5))
```

> Nous ne pouvons faire qu'un seul export par défaut. Il est déconseillé d'utilisé les export par défaut à moins que ce soit l'unique export d'un fichier.

Installer et importer des librairies

Lorsque l'on lance la commande `npm i react`, nous installons la librairie react. Cette librairies c'est elle aussi un module !

Nous pouvons donc utiliser `import` pour l'importer :

```
import { createRoot } from 'react-dom'

createRoot(document.querySelector('.container'))

// etc
```

Bonnes pratique

VSCode

Lorsqu'on developpe en javascript, html, css etc ... Lorsque l'on fais une application web il est conseillé d'avoir une bonne éditeur de code. Le plus utilisé aujourd'hui est VSCode

VSCode à la particularité de comprendre html, css et javascript nativement. C'est un éditeur très intelligent qui vous faira gagner beaucoup de temps.

> D'ailleurs VSCode lui même est developpé en javascript !

L'organisation

Dans un projet utilisant javascript, tout nos fichier `.js` doivent être ranger dans un dossier `src` (source), à l'exception des librairies qui elle se range dans `node_modules`.

Formatteur de code

Il est très important et essentiel de bien formatter son code. De bien répécter les espaces, d'être toujours cohérent, de bien indenter son code ...

Heureusement il éxiste des formatteur automatique, le plus célébre : prettier

Documenter votre code !

Un point essentiel pour trouver son premier emploie en tant que développeur c'est la documentation. On vous embauchera bien plus facilement si votre code contient des commentaires expliquant la démarche et le fonctionnement des différents membre.

Pas de `dead code`

ATTENTION : Ne jamais commenter du code ! Le code soit on le laisse comme tel, soit on le supprime.