

# Développement front

MMI 3 – TP#4 S5

Danielo **JEAN-LOUIS**

# Nodejs

- Outil permettant l'utilisation du javascript côté serveur
  - Utilisation des mêmes fonctions sauf celles manipulant une page
  - Accès au système : dossiers, fichiers...

# Nodejs

- Eco-système vaste ayant permis l'émergence d'outils variés et utiles pour les développeurs front
  - Création d'application natives
  - Système d'exploitation
  - Bundlers
  - ...

# Bundlers

- Outils nécessitant nodejs pour fonctionner
- Améliorent l'environnement de développement front-end
- Permettent de découper son code javascript pour le fusionner en un fichier

# Bundlers

- Optimisent les ressources
- Peuvent éliminer le code non utilisé
- Permettent d'utiliser du code non compatible pour le navigateur en temps normal

# Liste de bundlers (non exhaustive)

- Browserify (l'un des premiers)
- Grunt / Gulp (gestionnaires de tâches)
- Webpack
- Rollup
- Parcel
- ...
- **Vite**

# Vite

- Outil permettant d'améliorer l'environnement front-end
- Fonctionne avec Nodejs
- Se greffe à Rollup
- Créée par Evan You, créateur de VueJS

Source(s) :

- <https://vitejs.dev/>



# Vite

- Fonctionne clé en main
  - `npm create vite@latest`
- Dernière version majeure en date : v4
- Nécessite très peu de configuration par défaut

Source(s) :

- <https://vitejs.dev/>

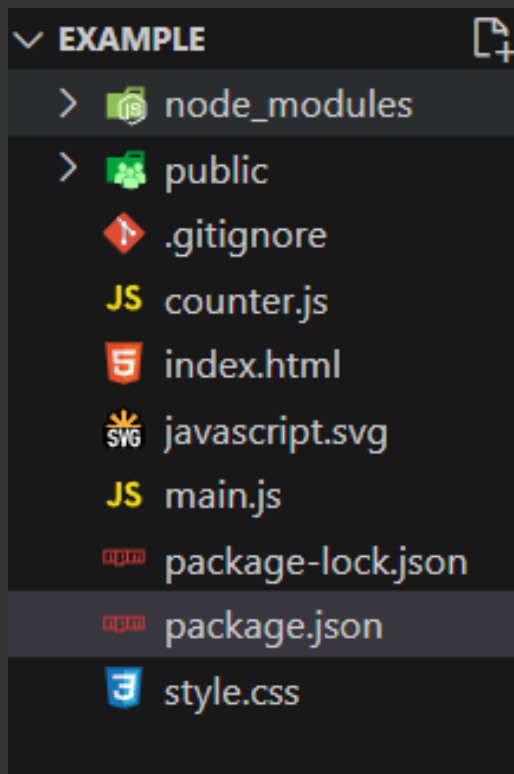
# Vite

- Gère plusieurs frameworks js : Angular, React...
- Conforme aux derniers standards javascript
- Gère les ressources **de tout type dans le javascript**

Source(s) :

- <https://vitejs.dev/>

# Vite

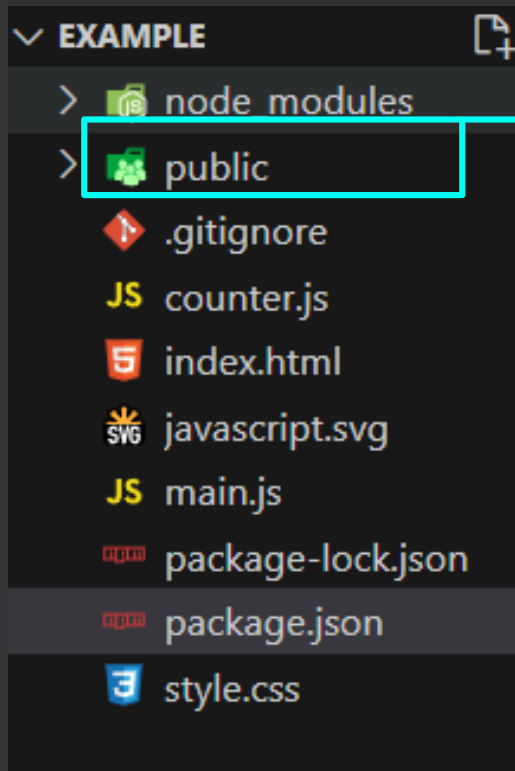


Après avoir installé les dépendances (npm install), nous sommes prêts à travailler

Source(s) :

- <https://vitejs.dev/>

# Vite



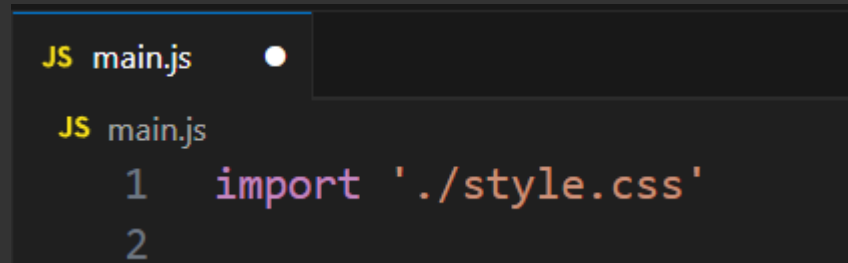
Contient les dépendances externes. Fichiers qui ne seront pas gérés par vite

Source(s) :

- <https://vitejs.dev/>

# Vite – Différences avec l'existant

- Gestion du CSS dans le javascript
  - **On importe le CSS dans nos fichiers javascript**

A screenshot of a code editor with a dark theme. The top tab is labeled 'JS main.js' with a white dot indicating it is the active file. The editor shows two lines of code: line 1 is 'import './style.css'' and line 2 is empty. The text is color-coded: 'import' is purple, the string './style.css' is orange, and the line numbers are light blue.

```
JS main.js
JS main.js
1  import './style.css'
2
```

Source(s) :

- <https://vitejs.dev/>

# Pratiquons ! - Vite (Partie 1)

Pré-requis :

- Avoir la ressource [ressources/vite](#)

A télécharger ici :

# Vite

- Extensible via un système de plugins
  - Rajoute de nouvelles fonctionnalités et nouveaux types d'imports dans les fichiers javascript comme les **préprocesseurs CSS**

Source(s) :

- <https://github.com/vitejs/awesome-vite#plugins>

# Préprocesseurs CSS

- Méta-langages CSS
- Ne sont pas lus par les navigateurs
- **Doivent être compilés en CSS**
- Permettent de simplifier l'écriture du CSS
- SCSS est le plus utilisé

## Source(s) :

- <https://www.alsacreations.com/article/lire/1717-les-preprocesseurs-css-c-est-sensass.html>
- <https://grafikart.fr/tutoriels/differences-sass-scss-329>
- <https://la-cascade.io/se-lancer-dans-sass/>
- <https://sass-lang.com/>



# Préprocesseurs CSS

- Utilisent une syntaxe proche du CSS
- Apportent de nouvelles fonctionnalités
  - Imbrication de sélecteurs
    - Limite la répétition de code
  - Conditions / boucles
  - **Variables compilées** – Elles ne sont pas modifiables après coup
  - ...

# Préprocesseurs CSS - Exemple

```
✓ .conteneur-boite {  
  background-color: ■ red;  
  padding: 0.8rem;  
  
  .titre {  
    font-size: 1.5rem;  
  }  
  // ...  
}
```

Code SCSS



```
.conteneur-boite {  
  background-color: ■ red;  
  padding: 0.8rem;  
}  
  
.conteneur-boite .titre {  
  font-size: 1.5rem;  
}
```

Code CSS  
(Une fois compilé)

# Préprocesseurs CSS

- SCSS fonctionne avec Vite dès l'installation de SCSS dans le projet

# Pratiquons ! - Vite (Partie 2)

Pré-requis :

- Avoir la ressource [ressources/vite](#)

A télécharger ici :

# Moteur de templating

- Langage permettant de produire des documents à partir de données
  - Exemple : une page web
- Permet de respecter le V du modèle MVC
  - Limite le code spaghetti

# Moteur de templating

- Propose d'importer d'autres templates ou d'en hériter
  - Gestion de l'orienté objet
- Souvent affecté à un framework
  - Symfony (php) → **twig**
  - Django (python) → **jinja**
  - ...

# twig

- Moteur de templating associé à symfony
- Utilise la même syntaxe que jinja ou encore nunjucks
  - Connaître un fait apprendre le trois
- Utilisable avec vite via un plugin

Source(s) :

- <https://twig.symfony.com/>

# twig

- Propose une syntaxe claire et facile à apprendre
- Extension de fichier en .twig

Source(s) :

- <https://twig.symfony.com/>



# twig



```
<?php
foreach ($items as $value) {
    if ($value.active) {
?>
        
<?php
    }
}
?>
```

← Code PHP

Code twig →



```
{% for value in items if value.active %}
    
{% endfor %}
```

Source(s) :

- <https://twig.symfony.com/>


# Twig - Syntaxe

- Trois syntaxes :
  - `{% __mot_clé__ %}` : fait quelque chose
    - Boucle, condition...
  - `{{ __mot_clé__ }}` : affiche quelque chose
  - `{# __mot_clé__ #}` : commentaire

Source(s) :

- <https://twig.symfony.com/>

# Twig - Syntaxe



```
<ul>
    {% for user in users %}
        <li>{{ user.username }}</li>
    {% endfor %}
</ul>
```

Ici on parcourt un tableau “users” contenant des objets dont on accède à la clé “username” et on affiche le contenu dans la balise <li>

Source(s) :

- <https://twig.symfony.com/>

# Variables d'environnement (env vars)

- Permettent d'injecter du contenu statique dans nos fichiers
  - Par exemple : serveur d'API

**Questions ?**

