

Aide-mémoire

JavaScript & Typescript

HTML

Inclure un fichier JavaScript

```
<script defer src="./chemin/du/fichier.js"></script>
```

Lorsque le fichier est un module

```
<script defer type="module" src="./chemin/du/fichier.js"></script>
```

Inclure un fichier CSS

```
<link rel="stylesheet" href="./chemin/du/fichier.css" />
```

HTML

Inclure un script JavaScript dans le fichier HTML

Les scripts doivent être inclus dans des tags `<script></script>`

```
<script>  
  function mafunction() {}  
</script>
```

Inclure une stylesheet dans le fichier HTML

Les styles doivent être inclus dans des tag `<style></style>`

```
<style>  
  .center { margin: auto; width: 50%; }  
</style>
```

JavaScript

Dans un module Node

Installation de nodemon

```
npm i -D nodemon
```

Exporter une fonction, une constante ou une variable

```
module.exports = { maconstante, mavariable, mafunction };
```

Importations des éléments exportés (à partir de monmodule.js)

```
const { maconstante, mavariable, mafunction } = require('./monmodule');
```

Exportation et importation d'un seul élément

Exportation

```
module.exports = mafunction;
```

Importation

```
const mafunction = require('./monmodule');
```

JavaScript

Dans un module ES6 (Web)

Exporter une fonction

```
export function mafonction() {}
```

Exporter une constante ou une variable

```
export const maconstante = 3.14159;  
export let mavariable;
```

Importations des éléments dans un module JavaScript

Ne pas oublier `type="module"` dans le tag script

```
import { mafonction, maconstante, mavariable } from './monfichier.js';
```

Exporter une valeur par défaut

```
export default function mafonction() {}
```

Importer une valeur par défaut

```
import mafonction from './monfichier.js';
```

TypeScript

Dans un module Node

Installation de ts-node

```
npm i -D ts-node
```

Exporter une fonction, une constante ou une variable

```
module.exports = { maconstante, mavariable, mafunction };
```

Ou

```
export = { maconstante, mavariable, mafunction }
```

Exportation d'un seul élément

```
module.exports = mafunction;
```

Ou

```
export = mafunction;
```

On utilise la même syntax que JavaScript pour l'importation des fichiers.

TypeScript

Dans un navigateur Web

On utilise exactement la même syntaxe que pour le JavaScript.

NPM

NPM: Node Package Manager

Obtenir la version

`npm -v`

Générer le fichier d'initialisation `Package.json`

`npm init -y`

Attention le répertoire courant ne doit pas comporter de caractères accentués, sinon exécuter la commande

`npm init`

Puis se laisser guider.

NPM

Installation d'un package

Installation d'une version localement

```
npm i <nom du package>
```

Installation d'une version pour un environnement de développement

```
npm i -D <nom du package>
```

Installation d'une version globale (accessible de n'importe quel projet).

```
npm i -g <nom du package>
```

Desinstallation d'un package

```
npm uninstall <nom du package>
```

Voir la liste des packages installés localement

```
npm list
```

```
npm list --depth=0
```

NPM

Voir la liste des packages installés globalement

`npm list -g`

Installer toutes les dépendances de l'application

`npm install`

Exécuter une commande de script décrite dans Package.json

`npm run <nom du script>`

Coté serveur

CORS (Cross Origin Resource Sharing)

Installation de la librairie cors

```
npm i cors
```

Importation de la librairie cors

```
import cors from 'cors';
```

Les options pour cors

```
const corsOptions = {  
  origin: 'http://127.0.0.1:5500',  
  methods: ['GET', 'POST']  
};
```

Ajout du middleware cors

```
app.use(cors(corsOptions));
```

Coté serveur

Utilisation de Express

Installation

```
npm i express
```

Importation des librairies en TypeScript et JavaScript (si "type": "module") dans le package.json

```
import express from 'express';  
import cors from 'cors';
```

Importation des librairies en JavaScript (hors modules)

```
const express = require('express');  
const cors = require('cors');
```

Coté serveur

Ajout du middleware pour la gestion des messages JSON

```
app.use(express.json());
```

Gestion des requêtes par Express

```
app.post('/add', (req, res) => {  
  res.send(req.body);  
});
```

On peut également envoyer un statut d'erreur

```
res.status(404).send(`The course with the id ${req.params.id} was not found`);
```

Lancer le serveur sur un port particulier

```
app.listen(LISTENING_PORT, () => console.log(`Listening...`));
```

Coté serveur

Utilisation d'un routeur Express

Côté client

Utilisation de l'API fetch

Envoi et réception d'un message JSON

```
const request = JSON.stringify({ a: param1, b: param2 });
fetch(SERVER_URL, {
  method: 'POST',
  headers: {
    'Accept': 'application/json',
    'Content-Type': 'application/json'
  },
  body: request
})
.then(response => response.json())
.then(response => console.log(response));
```

Browserify

Browserify permet d'utiliser des modules nodes dans un navigateur.

Initialiser NPM

```
npm init -y
```

Installer browserify en plus de la librairie souhaitée

```
npm i browserify
```

Créer un fichier index.js qui exporte la librairie souhaitée. Ci-dessous le code pour exporter la librairie Joi.

```
const Joi = require('joi');  
module.exports=Joi
```

Puis générer le fichier .js de la librairie que l'on souhaite exporter à l'aide de la commande
node_modules/.bin/browserify index.js -s <name of module> > <name of output JavaScript file>

Exemple

```
node_modules/.bin/browserify index.js -s Joi > Joi.js
```


Unit test Jest

Installation de Jest

`npm i -D jest`

Pour TypeScript

`npm i -D ts-jest`

On peut créer un fichier de configuration pour Jest

`jest.config.js`

Les fichiers tests doivent être nommés avec l'extension suivante

`*.test.js` pour JavaScript

`*.test.ts` pour TypeScript

Unit test Jest

Fichier de configuration

Exemple de fichier de configuration pour TypeScript

```
/** @type {import('ts-jest/dist/types').InitialOptionsTsJest} */  
module.exports = {  
  preset: 'ts-jest',  
  testEnvironment: 'node',  
  testMatch: ["**/src/*.test.(js|ts|tsx)"],  
  moduleFileExtensions: ["ts", "js"]  
};
```

Note pour une Web app vous devez utiliser

`testEnvironment: 'jsdom'`

Unit test Jest

Exemples de test

```
test("Digit to Roman", () => {  
  expect(script.digitalToRoman(1)).toBe("I");  
});
```

Ou

```
describe("test getSuccessiveNumbers", () => {  
  it("Digit to Roman", () => {  
    expect(script.digitalToRoman(1)).toBe("I");  
  });  
});
```