



En rang 2 par 2

On va organiser tout ça, les modules c'est la vie



En rang 2 par 2

Bien bien bien... On a fait du chemin jusqu'ici, on sait manipuler le DOM, on sait créer des fonctions, des variables, ajouter des classes... C'est bien mais vous pouvez constater que le fichier index.js (ou le code que vous écrivez dans la balise `<script>`) commence à être un peu volumineux et qu'on peut s'y perdre à terme.

Au cours de cette mission, nous allons voir comment organiser notre code en **modules**. Toute la logique sera extraite du code principal pour être découpée en différents blocs, avec comme focus : 1 idée -> 1 module.



En rang 2 par 2

Préambule:

Avant de vous lancer à corps perdu dans le code, il va falloir installer un mini environnement de travail pour faire fonctionner nos modules.

En local:

Installer WAMP (sur Windows) ici : <https://www.wampserver.com/>

ou MAMP (sur Mac) ici : <https://www.mamp.info/en/downloads/>

En rang 2 par 2



MAMP (ou WAMP) un serveur local, il va simuler la présence de votre site Web en ligne pour faire simple, et donc autoriser un dialogue machine/serveur indispensable pour le fonctionnement de nos modules en JS.

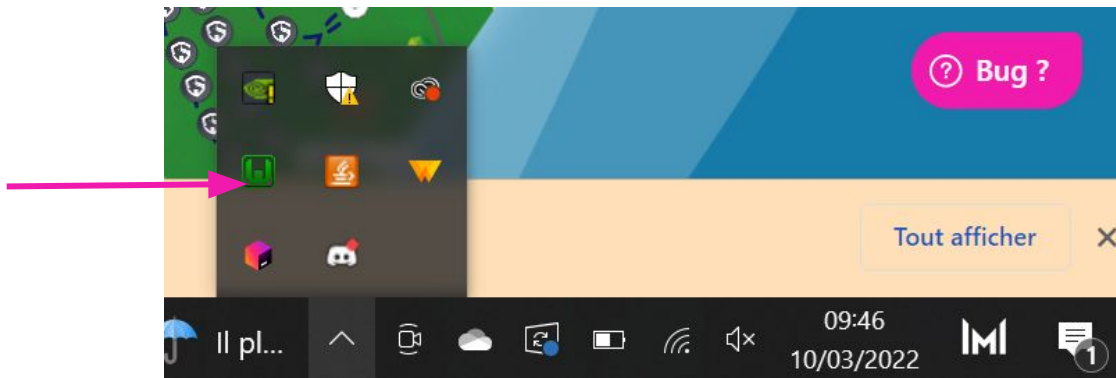
Une fois installé, il faudra lancer le serveur :

En rang 2 par 2



Windows

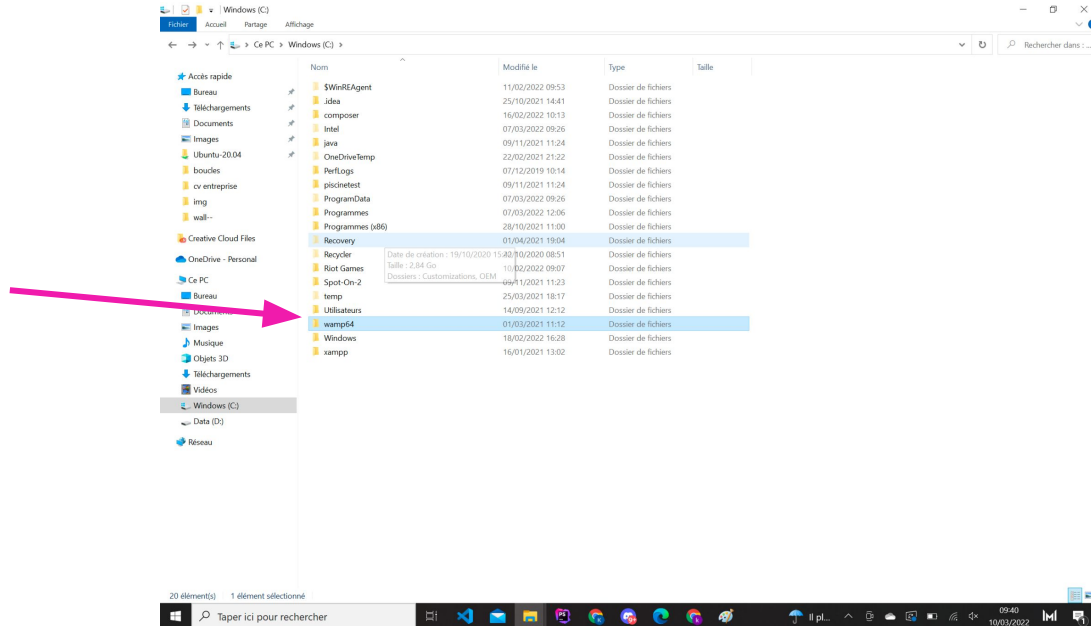
- Ouvrez le fichier WAMP, vérifier le lancement du serveur ici



En rang 2 par 2



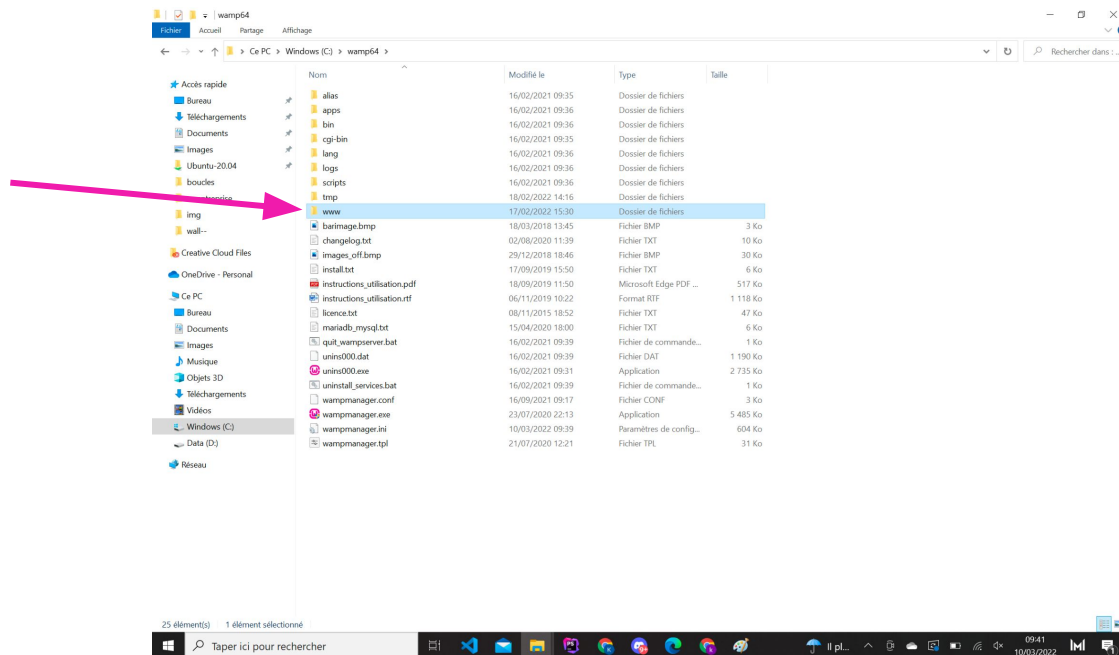
- Allez dans le dossier d'installation de WAMP



En rang 2 par 2



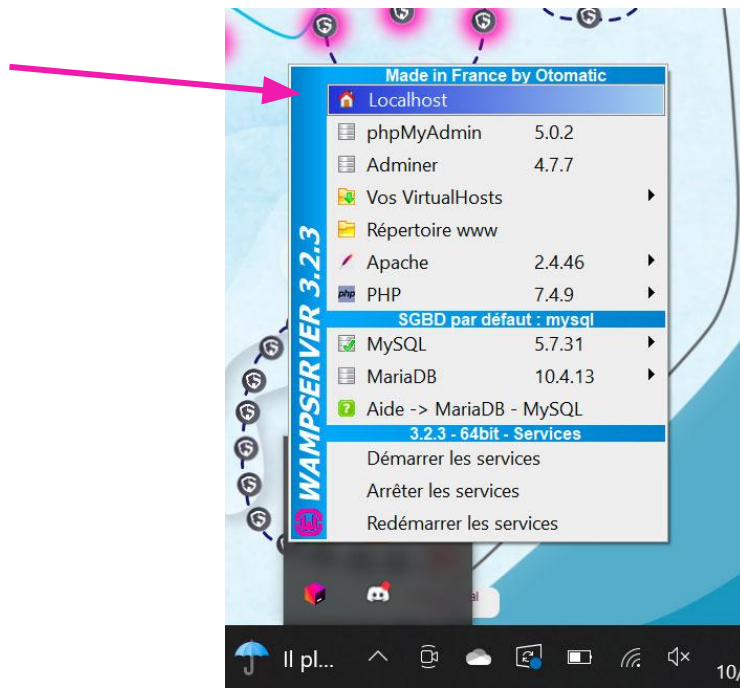
- Déposez votre projet entier (dossier racine) **DANS** le dossier www de wamp



En rang 2 par 2



- On revient sur notre **W**, puis on choisit **Localhost**

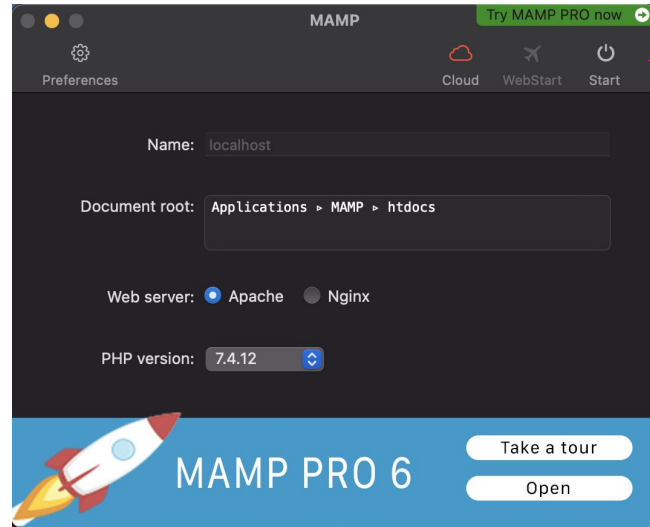


En rang 2 par 2



Votre site devrait se lancer, si vous avez plusieurs projets, il faudra cliquer sur le nom de votre projet et lancer la page d'accueil (votre fichier .html).

Mac



Cliquer ici pour lancer le serveur

En rang 2 par 2

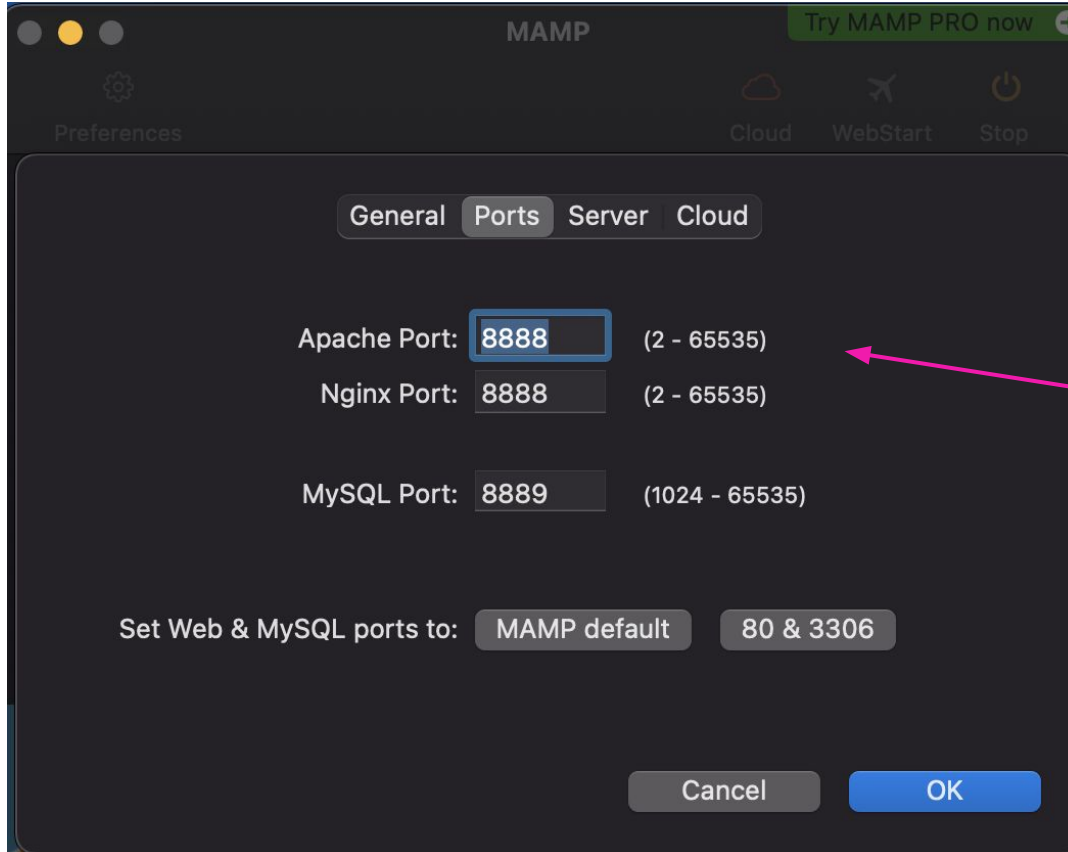


MAMP	
Nom	Date de modification
> bin	9 mars 2022 à 09:15
> cgi-bin	30 nov. 2020 à 09:06
> conf	1 déc. 2020 à 09:13
> db	1 déc. 2020 à 09:14
> fcgi-bin	30 nov. 2020 à 09:06
> htdocs	9 mars 2022 à 09:16
LEAME.rtf	30 nov. 2020 à 09:06
> Library	1 déc. 2020 à 09:14
> licences	1 déc. 2020 à 09:14
LIESMICH.rtf	30 nov. 2020 à 09:06
LISEZ-MOI.rtf	30 nov. 2020 à 09:06
> logs	18 févr. 2022 à 17:05
MAMP	23 janv. 2021 à 11:10
README.rtf	30 nov. 2020 à 09:06
> tmp	10 mars 2022 à 09:59
прочти.rtf	30 nov. 2020 à 09:06
お読みください.rtf	30 nov. 2020 à 09:06

- Aller dans Applications -> MAMP
- Mettre son dossier racine DANS le dossier htdocs



En rang 2 par 2



- Aller dans Préférences -> Ports
- Vérifier le port utilisé
- Dans Chrome, taper localhost:NuméroDuPort
- Ici: localhost:8888

En rang 2 par 2



Naviguer dans les dossiers (si plusieurs dossier présents dans votre HTDOCS) et lancer votre site (fichier HTML).



En rang 2 par 2

BIEN! Une bonne chose de faite non? On va pouvoir maintenant se concentrer sur notre découpage de logique en différents fichiers Javascript.

1ère question: Pourquoi?

Déjà, parce qu'il sera bien plus facile de maintenir du code léger plutôt qu'une énorme page de JS. Ensuite parce que nous pourrons exporter ce code dans d'autres projets, et ça c'est fort, très fort même. Vous remarquerez au cour de votre carrière que la logique se répète souvent, donc autant pouvoir exporter facilement cette dernière!

En rang 2 par 2



2ème question: Comment?

Rien de bien compliqué ici. Il vous faudra un fichier qui centralise les imports de fichiers et donc les fameux fichiers de logique.

```
<script type="module" src="js/main.js"></script>
```

Le type MODULE est OBLIGATOIRE! Sinon pas d'import et d'export.

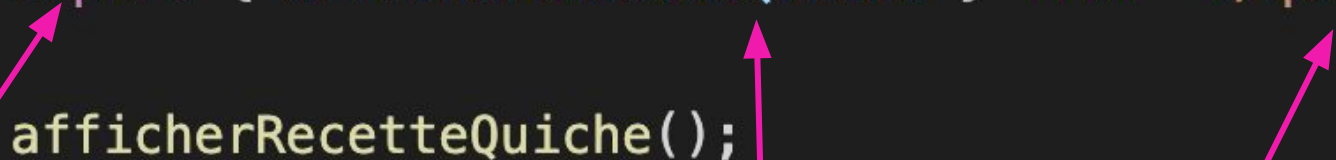
Ceci sera le seul lien vers un fichier JS fait dans votre HTML, juste avant la fermeture de la balise <body>



En rang 2 par 2

main.js

```
import { afficherRecetteQuiche } from './quiche.js'  
afficherRecetteQuiche();
```



On **import** un élément qui s'appelle **afficherRecetteQuiche** dans le fichier **quiche.js** (./ obligatoire ici)

Il ne reste qu'à appeler la logique importer



En rang 2 par 2

quiche.js

On **export** la
fonction (pour
pouvoir
l'**import** dans
main.js)

```
const titre = document.getElementById('recette');
export function afficherRecetteQuiche() {
  let recette = document.createElement('ul');
  let ingredients = ["Oeufs", "Lait", "Lardon", "Muscade", "Sel", "Poivre"];

  for (let i = 0; i < ingredients.length; i++) {
    let liIngredients = document.createElement('li');
    liIngredients.innerHTML = ingredients[i];
    recette.append(liIngredients);
    titre.after(recette)
  }
}
```

Toute la logique de création de la liste est ici. Même si c'est bref ici, c'est comme ça qu'il faudra penser vos projets par la suite.

En rang 2 par 2



Avec ces import/export on égratigne ici la surface de la POO (Programmation Orientée Objet), qui sera la BASE de votre vie de développeur.

On verra dans la mission d'après la notion de Class, d'héritage, de constructeur...
Bref, d'OBJET quoi!

En rang 2 par 2



Le but de cette mission va être de développer le **loto**, comme sur TF1 avec Jean-Pierre Foucault !

Il faudra donc créer un programme qui permet à l'utilisateur de rentrer ses valeurs pour le loto, puis qui va lancer un tirage aléatoire, afficher ce tirage et comparer le résultat avec les données entrées par l'utilisateur pour vérifier si il a gagné ou non!

On affichera aussi un message personnalisé en fonction du nombre de bons numéros et on donnera la possibilité à l'utilisateur de relancer un tirage complet!

Démo ici : <https://youtu.be/n3MvaPT9vag>



En rang 2 par 2

Rendu attendu:

- Un fichier permettant de tirer le Loto
- Un fichier permettant d'afficher le Loto
- Un fichier permettant de reset le Loto (ne rechargez pas juste la page, je vous vois venir)
- Un fichier permettant de récupérer les valeurs rentrées par le joueur
- Un fichier permettant de comparer le tirage et les valeurs du joueur
- Un fichier Permettant d'afficher un message perso en fonction du nombre de bons numéros (avec un switch, message pour 0 numéro, 1, 6 et le message par défaut)
- Un fichier regroupant la logique main.js, qui à les différents EventListener

En rang 2 par 2



Droppez votre projet dans une archive en .zip sous le nom
`prenomNom_loto.zip`

Bonne chance au tirage!!!