

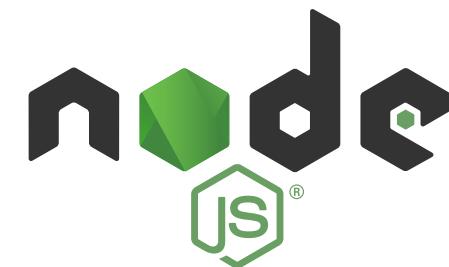
# A propos de moi

**Developpeur full stack**

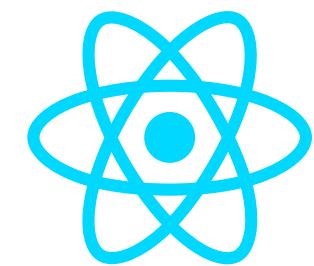
**@ Aventique (boite de Dev Web et Mobile)**

[linkedin.com/in/ayoub-bouguettaya/](https://www.linkedin.com/in/ayoub-bouguettaya/)  
[ayoub.bouguettaya2020@gmail.com](mailto:ayoub.bouguettaya2020@gmail.com)

# NodeJs et ReactJs

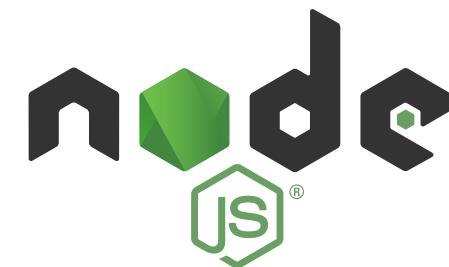


Node

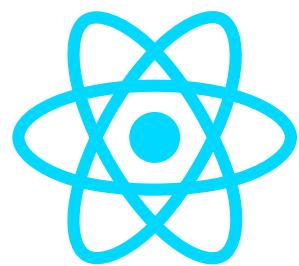


React

# Pourquoi NodeJs et ReactJs



Node



React

## Pourquoi NodeJs et ReactJs

technologie utilise pour developper  
des applications web moderne

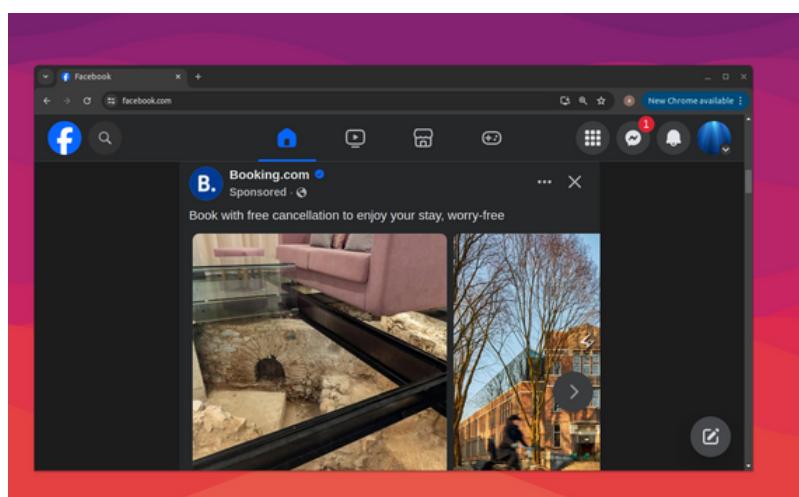


Rappelle

comment fonction le web ?



navigateur  
browser



Requete Exemple: facebook.com

Via Internet

Interface Utilisateur html,css + js

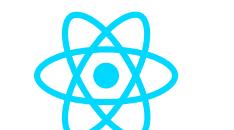
Requete donnee Exemple: facebook.com

donnees format Text / Json / blob ....



serveur  
de facebook

example technologie web  
serveur node / serveur react



## caractéristique Requête Http/https

## URL ,Méthode

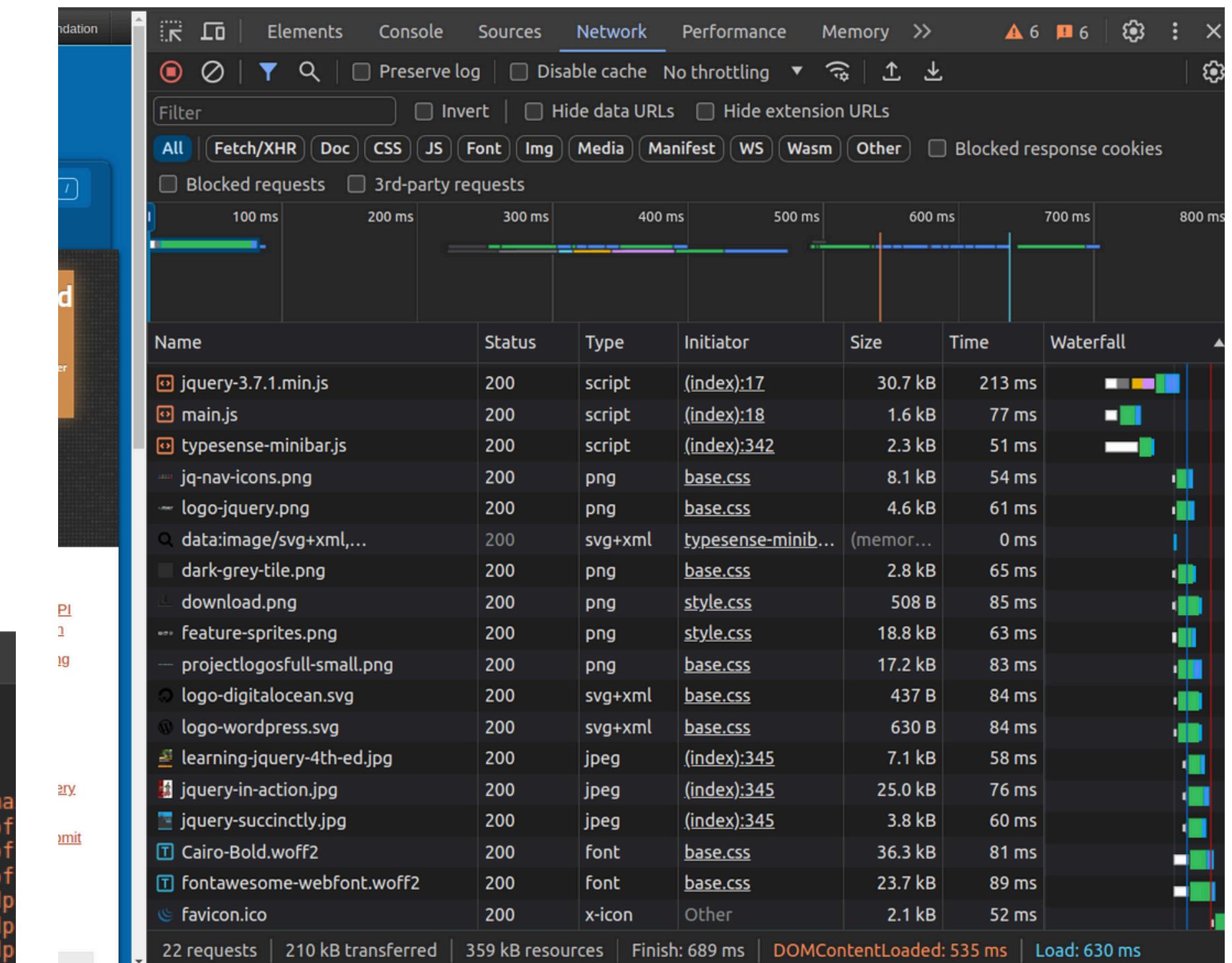
	Headers	Preview	Response	Initiator	Timing
General					
Request URL:	https://nodejs.org/en				
Request Method:	GET				
Status Code:	200 OK				

## response , code status

	Headers	Preview	Response	Initiator	Timing
1	<!DOCTYPE html>				
-	<html class="__variable_7eb81d __variable_9790f2" dir="ltr" lang="en-GB">				
-	<head>				
-	<meta charset="utf-8"/>				
-	<meta name="viewport" content="width=device-width, initial-scale=1, ma				
-	<link rel="preload" href="/_next/static/media/3478b6abef19b3b3-s.p.wof				
-	<link rel="preload" href="/_next/static/media/3d9ea938b6afa941-s.p.wof				
-	<link rel="preload" href="/_next/static/media/be2416cbb012c256-s.p.wof				
-	<link rel="stylesheet" href="/_next/static/css/f4db1b58a3b2c635.css?dp				
-	<link rel="stylesheet" href="/_next/static/css/b63106d781681399.css?dp				
-	<link rel="stylesheet" href="/_next/static/css/2458e5048894197c.css?dp				
-	<link rel="stylesheet" href="/_next/static/css/4350c1bd3e47245b.css?dp				

type de réponse :

html, js, css, image , text , font



ouvrir google chrome et inspecter les différentes requêtes https  
aller sur le tableau Network (Réseau)

## C'est Quoi Node Js

c'est un environnement d'executer javascript en dehors de navigateur  
developper a la base de moteur google chrome v8

## pourquoi Node Js

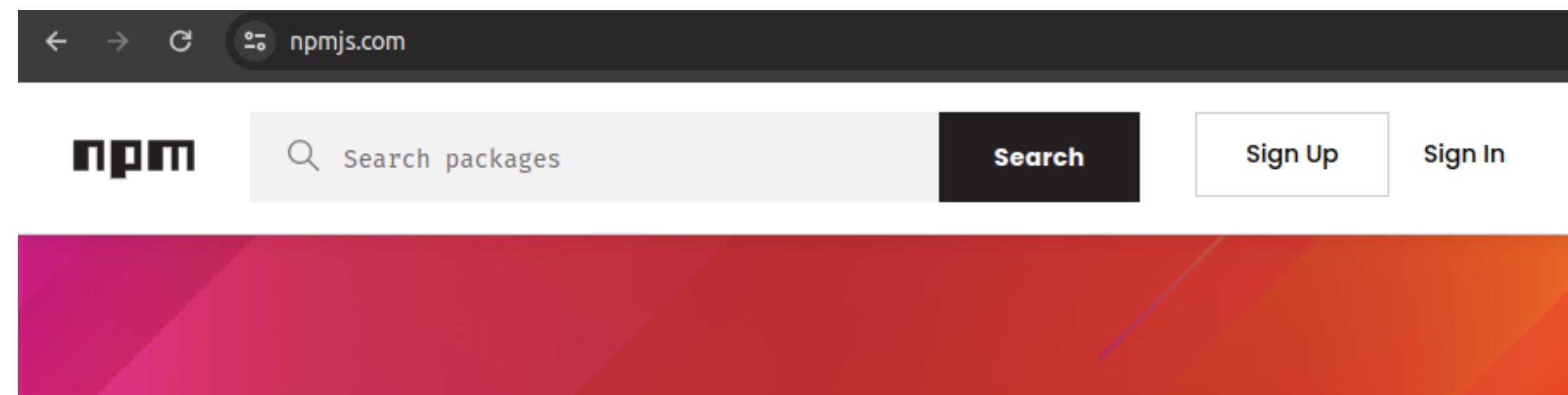
utilise le language javascript dans les serveurs web.

les serveurs web c'est la partie **invisible** dans notre interaction avec une page web ou application web/mobile . parmi ces roles:

- servir des pages htmls / css / javascript
- manipuler des donnees de l'application stocker en base de donnees, fichiers, disque

# NPM (node package manager)

- npm (Node Package Manager) est le gestionnaire de paquets par défaut de Node.js.
- Il permet d'installer, gérer et partager des bibliothèques et dépendances JavaScript.
- Il est préinstallé avec Node.js.
- Il dispose du plus grand écosystème de paquets open-source pour JavaScript.



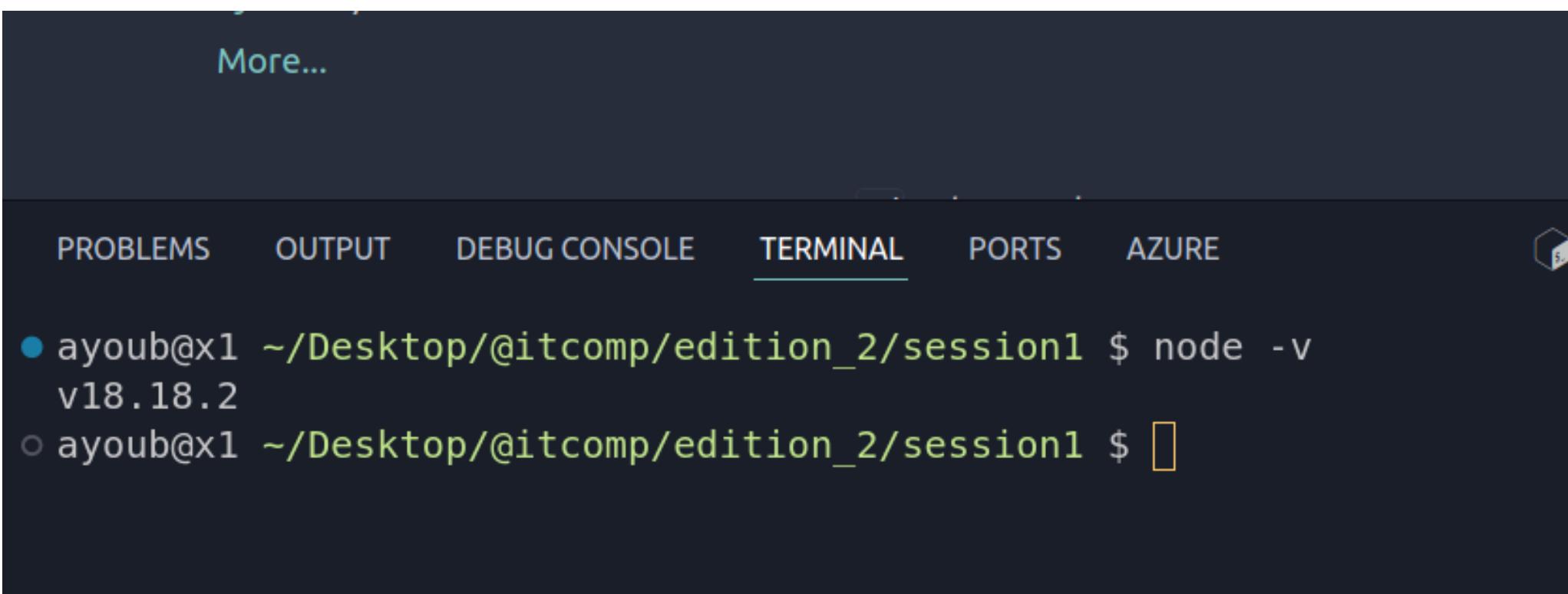
# NPM (node package manager)

- . Comment fonctionne npm ?
- npm télécharge et installe des paquets depuis le registre npm (<https://www.npmjs.com>).
- Il utilise un fichier `package.json` pour suivre les dépendances du projet.
- Il permet d'installer des paquets localement (dans un projet) ou globalement (sur tout le système).

Commandes de base npm	
Commande	Description
<code>npm init -y</code>	Crée un fichier <b>package.json</b> (initialisation du projet)
<code>npm install &lt;package&gt;</code>	Installe un paquet localement (ex: <code>npm install express</code> )
<code>npm install -g &lt;package&gt;</code>	Installe un paquet globalement (ex: <code>npm install -g nodemon</code> )

## Exemple: simple http server

verifier que node est bien installer dans notre machine



A screenshot of the Visual Studio Code (VS Code) interface, specifically focusing on the Terminal tab. The terminal window shows the following command-line interaction:

```
● ayoub@x1 ~/Desktop/@itcomp/edition_2/session1 $ node -v
v18.18.2
○ ayoub@x1 ~/Desktop/@itcomp/edition_2/session1 $ 
```

The terminal tab is highlighted with a blue underline. Above the terminal, there is a dark header bar with the VS Code logo and some icons. Below the header, there are several tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is active), PORTS, and AZURE. There is also a small icon in the top right corner.

# Exemple: simple http server

initialiser npm dans le projet

```
npm init -y
```

installer le package express

```
npm install express
```

Vérifier package.json

Créer un serveur Express simple

dans un fichier index.js

```
node server.js
```

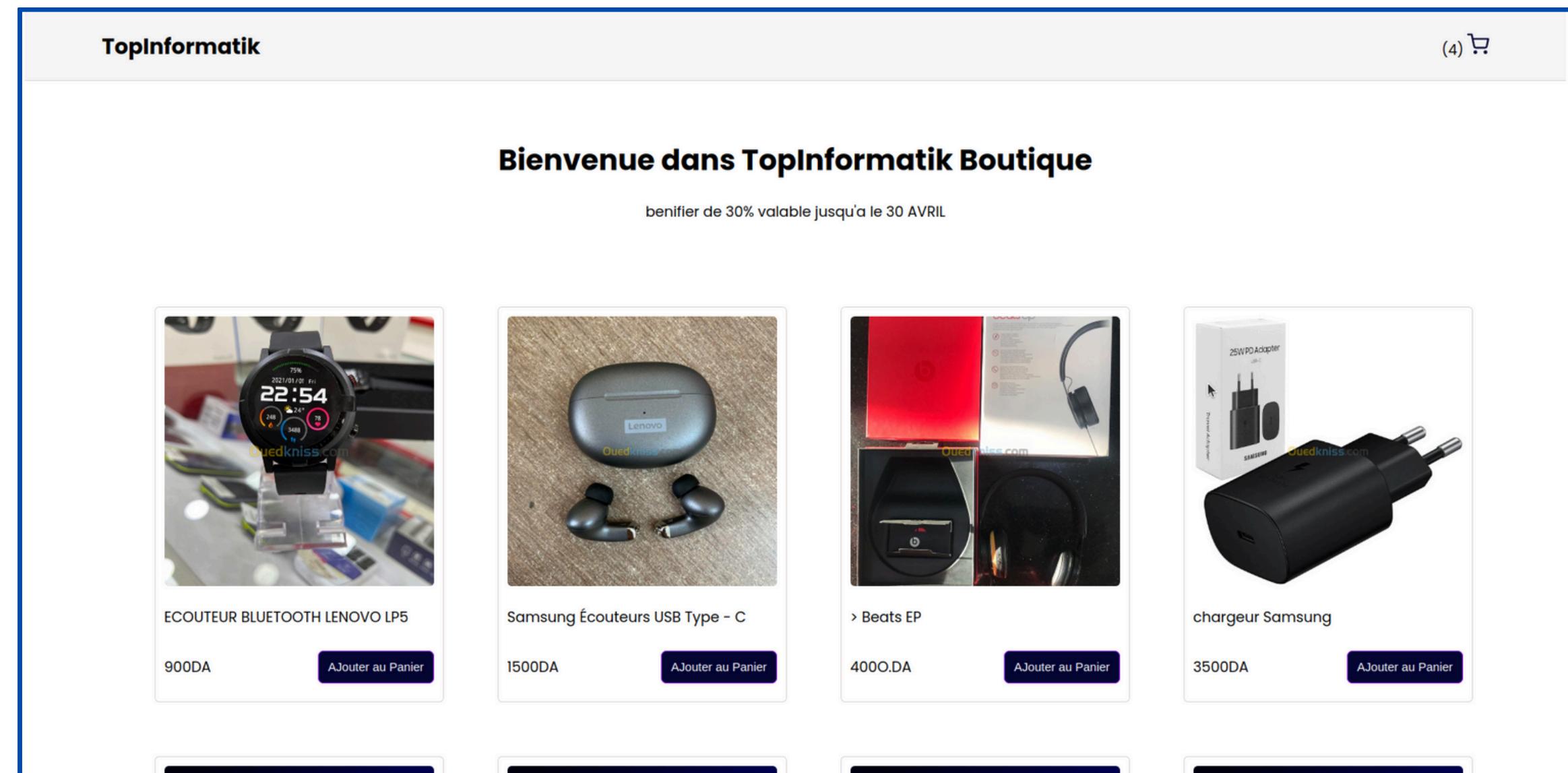
```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
  res.send('Hello, World!');
});

app.listen(3000, () => {
  console.log('Serveur démarré sur le port 3000');
});
```

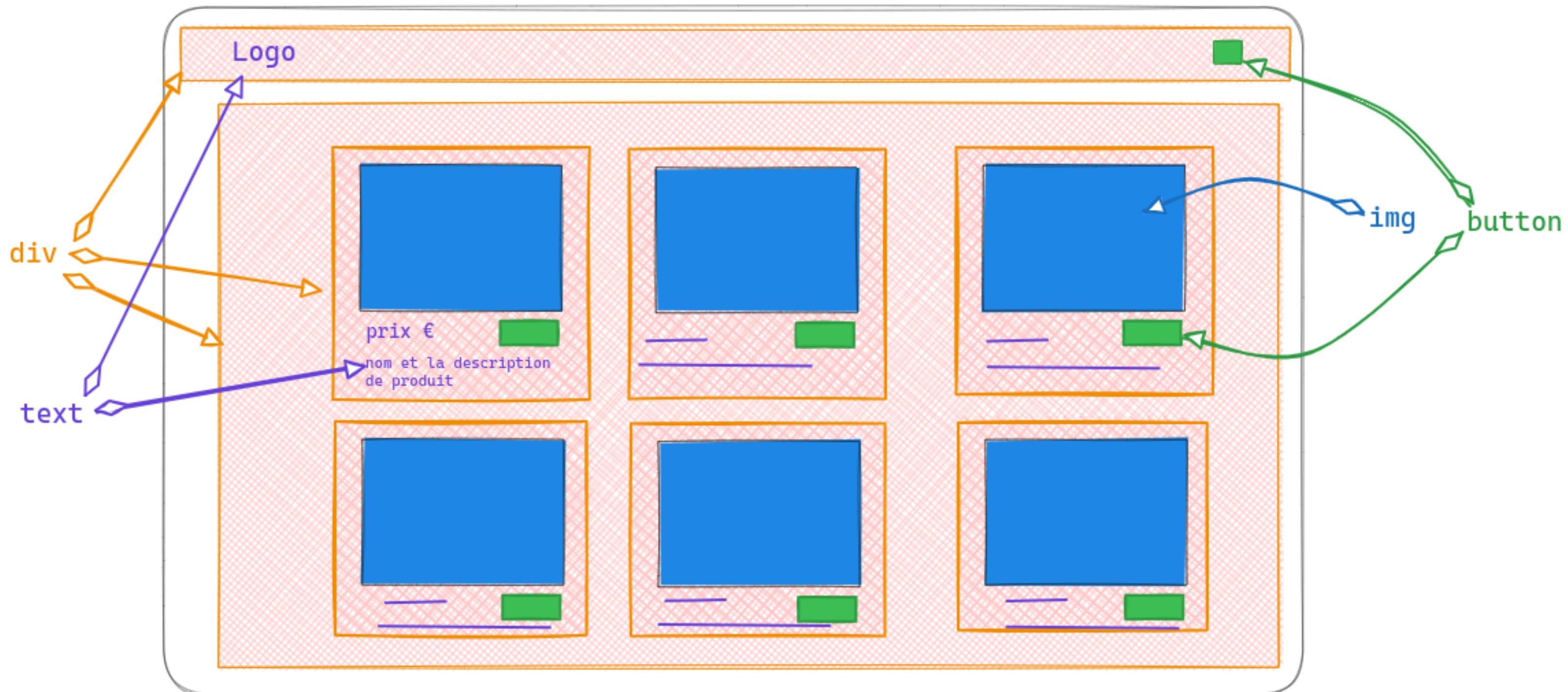
# Rappelle | Exercice

On utilisant: html ,css , javascript



les tags html utilisé: div ul li button img

css: display flex, align-items, justify-content , background, padding, margin



# Etapes Exercice

- ouvrir VScode + installer Live Server
- crée des fichiers vide ***index.js index.html styles.css***

```
<!DOCTYPE html>
<html lang="en">
  <head>
    ...
    <link rel="stylesheet" href=".//styles.css" />
    <script src="index.js"></script>
    ...
  </head>
  <body>
  </body>
</html>
```

## Rappelle

### function standard

```
const getRectArea = function (width, height) {  
    return width * height;  
};  
  
console.log(getRectArea(3, 4));  
// Expected output: 12
```

```
function name(param0) {  
    statements  
}  
  
function name(param0, param1) {  
    statements  
}
```



### Arrow function expressions

une manière différente de déclarer les fonctions

```
// Traditional anonymous function  
(function (a) {  
    return a + 100;  
});  
  
// 1. Remove the word "function" and place arrow between the argument and opening body  
brace  
(a) => {  
    return a + 100;  
};  
  
// 2. Remove the body braces and word "return" – the return is implied.  
(a) => a + 100;  
  
// 3. Remove the parameter parentheses  
a => a + 100;
```

## Rappelle

# L'opérateur conditionnel

L'opérateur (**ternaire**) conditionnel utilisé comme raccourci de l'expression **if ... else**

## syntaxe

```
condition ? exprSiVrai : exprSiFaux;
```

## example

```
"Le prix est : " + (estMembre ? "15 €" : "30 €");
```

## example2

```
function getFee(isMember) {  
    return isMember ? '$2.00' : '$10.00';  
}  
  
console.log(getFee(true));  
// Expected output: "$2.00"  
  
console.log(getFee(false));  
// Expected output: "$10.00"  
  
console.log(getFee(null));  
// Expected output: "$10.00"
```

## Rappelle Operation sur les tableaux (Arrays)

### Array.map()

La méthode map() crée un nouveau tableau avec les résultats de l'appel d'une fonction fournie sur chaque élément du tableau appelant.

#### syntaxe

```
var nouveauTableau = arr.map( fonctionCallback )
```

#### exemple

dans cette exemple on a formater le tableau original  
pour un nouveau tableau qui contient un format d'element different

```
var tableauOrig = [
  { clé: 1, valeur: 10 },
  { clé: 2, valeur: 20 },
  { clé: 3, valeur: 30 },
];
var tableauFormaté = tableauOrig.map((obj) => {
  var rObj = {};
  rObj[obj.clé] = obj.valeur;
  return rObj;
});
// tableauFormaté vaut maintenant [{1:10}, {2:20}, {3:30}],
// tableauOrig vaut toujours
// [{clé:1, valeur:10},
//  {clé:2, valeur:20},
//  {clé:3, valeur: 30}
// ]
```

## Rappelle Operation sur les tableaux (Arrays)

### Array.filter()

La méthode filter() crée et retourne un nouveau tableau contenant tous les éléments du tableau d'origine qui remplissent une condition déterminée par la fonction callback.

#### syntaxe

```
var nouveauTableau = arr.filter( fonctionCallback )
```

#### example

dans cette example on a filter le tableau original pour contenir seulement les elements avec nombre de chaine de caractère supérieur a 6

#### JavaScript Demo: Array.filter()

```
1 const words = ['spray', 'elite', 'exuberant', 'destruction', 'present'];
2
3 const result = words.filter((word) => word.length > 6);
4
5 console.log(result);
6 // Expected output: Array ["exuberant", "destruction", "present"]
7
```

## Rappelle Operation sur js Objects

**example** manipulation des objects , l'ajoute ,mise a jour et récupération d'un attribut ,

### Définition de variable et fonction

```
> const ages = { alice: 18, bob: 27 };

function getAge(name) {
  return ages[name];
}

const setAge = (name,age) => {
  ages[name] = age;
}
```

### Sequence d'opération

```
> getAge("ahmed")
< undefined
> getAge("alice")
< 18
> setAge("ahmed",22)
< undefined
> getAge("ahmed")
< 22
> setAge("alice",19)
< undefined
> getAge("alice")
< 19
```

## Rappelle Operateur Spread sur les tableaux et object

**Syntaxe de décomposition** utilisation a fin de copier , concaténé object ou tableau

tableau avec  
plus d'elements

```
[...objetIterable, 4, 5, 6];
```

objet copie

```
let objclone = { ...obj };
```

### tableaux

JS

```
const parts = ["shoulders", "knees"];
const lyrics = ["head", ...parts, "and",
"toes"];
// ["head", "shoulders", "knees", "and",
"toes"]
```

### objets

```
const obj1 = { foo: "bar", x: 42 };
const obj2 = { bar: "baz", y: 13 };

const mergedobj = { ...obj1, ...obj2 };
// { foo: "bar", x: 42, bar: "baz", y: 13 }
```

```
const arr = [1, 2, 3];
const arr2 = [...arr]; // like arr.slice()

arr2.push(4);
// arr2 becomes [1, 2, 3, 4]
// arr remains unaffected
```

```
const obj1 = { foo: "bar", x: 42 };
const obj2 = { foo: "baz", y: 13 };

const mergedobj = { x: 41, ...obj1, ...obj2, y:
9 }; // { x: 42, foo: "baz", y: 9 }
```

# Rappelle Operateur destructuring sur les tableaux et object

**Syntaxe de Affecter par décomposition** permet d'extraire des données d'un tableau ou object

## examples: objets

### examples: tableaux

```
let a, b, rest;
[a, b] = [10, 20];

console.log(a);
// Expected output: 10

console.log(b);
// Expected output: 20

[a, b, ...rest] = [10, 20, 30, 40, 50];

console.log(rest);
// Expected output: Array [30, 40, 50]
```

```
let { a, b, ...reste } = { a: 10, b: 20, c: 30, d: 40 };

a; // 10
b; // 20
reste; // { c: 30, d: 40 }
```

```
const x = [1, 2, 3, 4, 5]; // On crée un "paquet" de données
const [y, z] = x; // On utilise l'affectation par décomposition
console.log(y); // 1
console.log(z); // 2
```

```
const person = {
  birthName: "ait amer",
  givenName: "ahmed",
  age: 24,
  orders: [
    {
      productId: "hvsjhdcks",
      quantity: 1
    }
  ]
}

const { orders } = person;

// ces 2 log sont équivalant
// Array [Object { productId: "hvsjhdcks", quantity: 1 }]

console.log(orders)
console.log(person.orders)
```

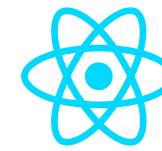
# les sites modern sont beaucoup plus complexe!

manipuler les etats des elements html DOM directement par javascript est trop complexe pour des site web interactives.

et c'est pour cela que les développeurs ont conçu des framework qui facilite cette tâche tel que :

- Reactjs (crée par facebook)
- Angular (par google)
- VueJs et plein d'autres.

# Reactjs



**Reactjs:** library javascript conçu pour faciliter la création de **complexe UI** en offrant la possibilité de découper UI en **petite composant** et la possibilité de **la re-utilisations** de ces composants .

## Concept de base

- Components
- JSX
- State
- Props
- Effect

# Reactjs

Composant React ou components c'est juste une fonction javascript qui se transforme en html DOM

la fonction doit

- retourner un object **JSX** qui a une syntaxe et un rôle similaire à html
- encapsule une logique concernant le composant **State & Effect**
- accepte des paramètres (**Props**)
- stocker et manipuler l'état de composant et manipuler le DOM indirectement

```
export const ProductCard = ( data ) => {  
    // add here your javascript logic  
  
    return (  
        <div className="card">  
            <img src={data.image} />  
            <p> {data.name} </p>  
            <div>  
                <p> {data.price}DA </p>  
                <button> ajouter au Panier </button>  
            </div>  
        </div>  
    )  
}
```

code minimal

### Example : d'un composant React



ECOUTEUR BLUETOOTH LENOVO LP5

900DA

AJouter au Panier

# Reactjs

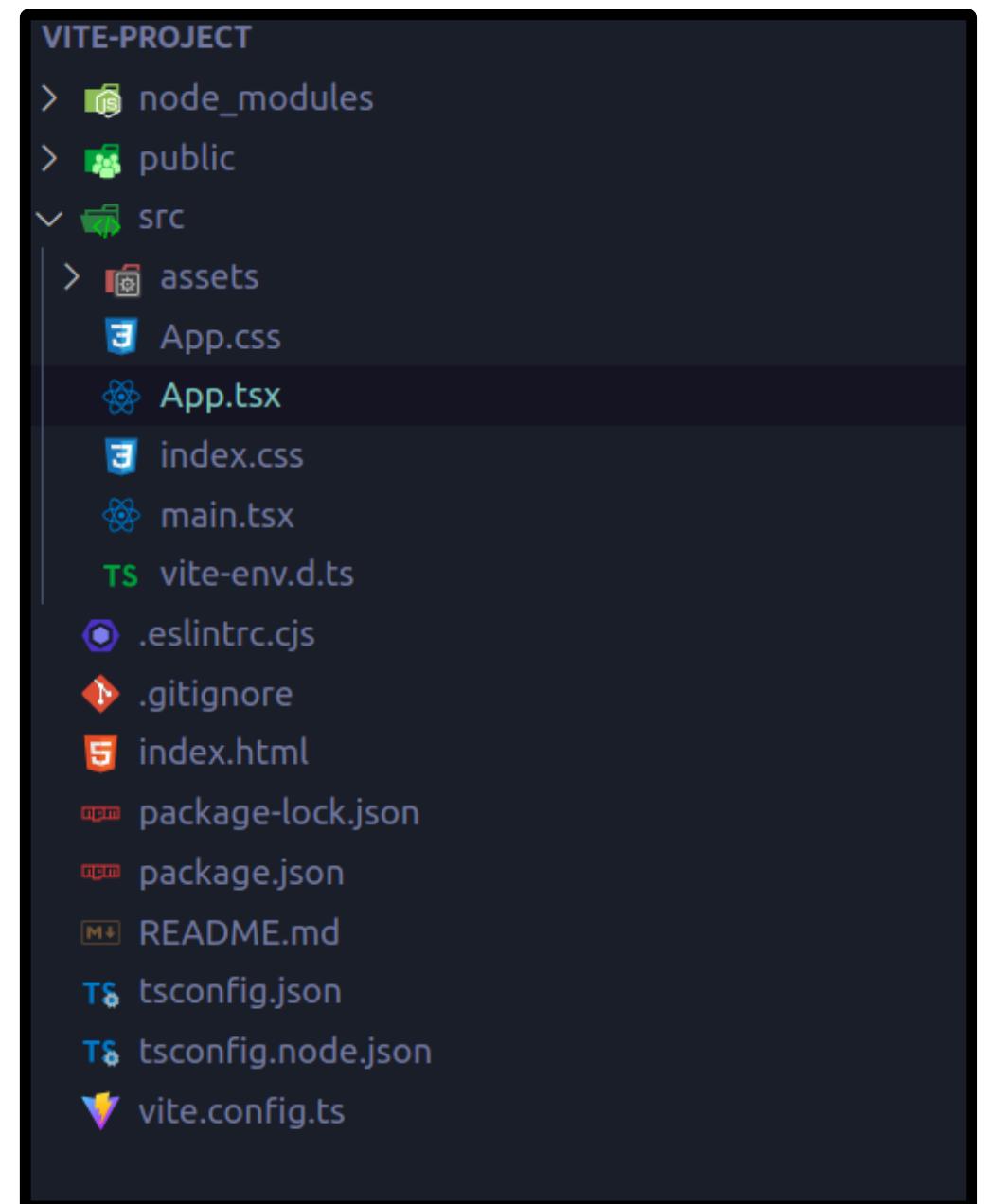
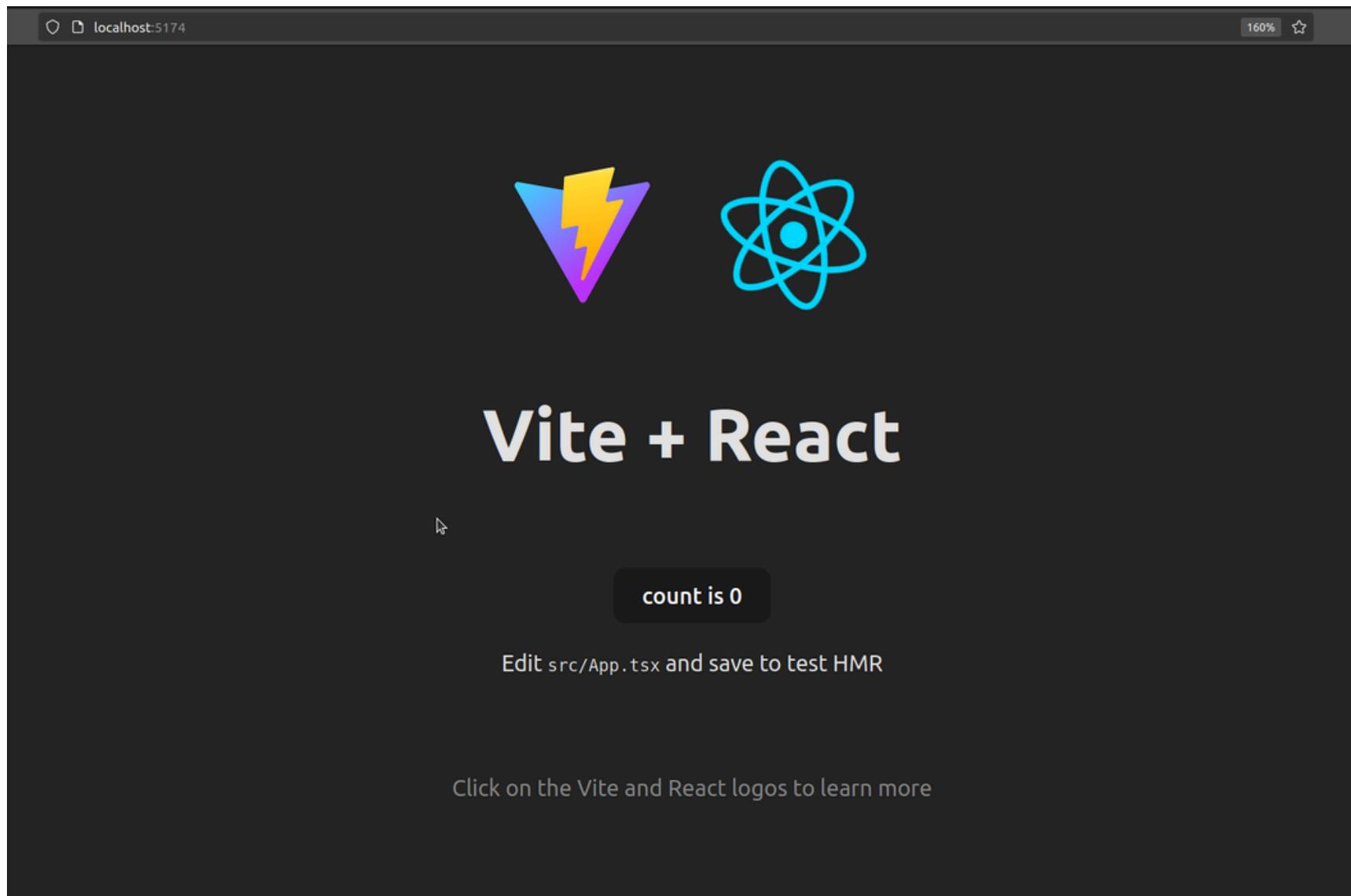
## Play around and discover Reactjs

- Préparer l'environnement et installer Nodejs
- NPM node package manager

# create un project React

- ouvrir le terminal dans vscode dans un dossier vide et taper **npm create vite@latest**
- dans le terminal taper aussi **npm install** apres **npm run dev**
- ouvrir le navigateur et taper **http://localhost:5173/**

code générer



# structure de projet

The screenshot shows the file structure of a Vite.js project named "VITE-PROJECT". The structure includes:

- node\_modules
- public
- src
  - assets
    - App.css
    - App.tsx
    - index.css
    - main.tsx
  - vite-env.d.ts
- .eslintrc.cjs
- .gitignore
- index.html
- package-lock.json
- package.json
- README.md
- tsconfig.json
- tsconfig.node.json
- vite.config.ts

**Public** dossier contien des fichiers publics telque les images

**src/main.tsx** le composant d'entré (ou bien composant Pere)

**package.json** specification des modules / library javascript utilisé

# structure de projet

src/main.tsx le composant d'entrée (ou bien composant Pere)

la Fonction CreateRoot afficher les composants à partir de l'élément avec id égale à "root"

```
page.tsx      main.tsx ×  
src > main.tsx  
1 import React from 'react'  
2 import ReactDOM from 'react-dom/client'  
3 import { Page } from './page'  
4  
5 ReactDOM.createRoot(document.getElementById('root')!).render(  
6   <React.StrictMode>  
7     <Page />  
8   </React.StrictMode>,  
9 )
```

index.html

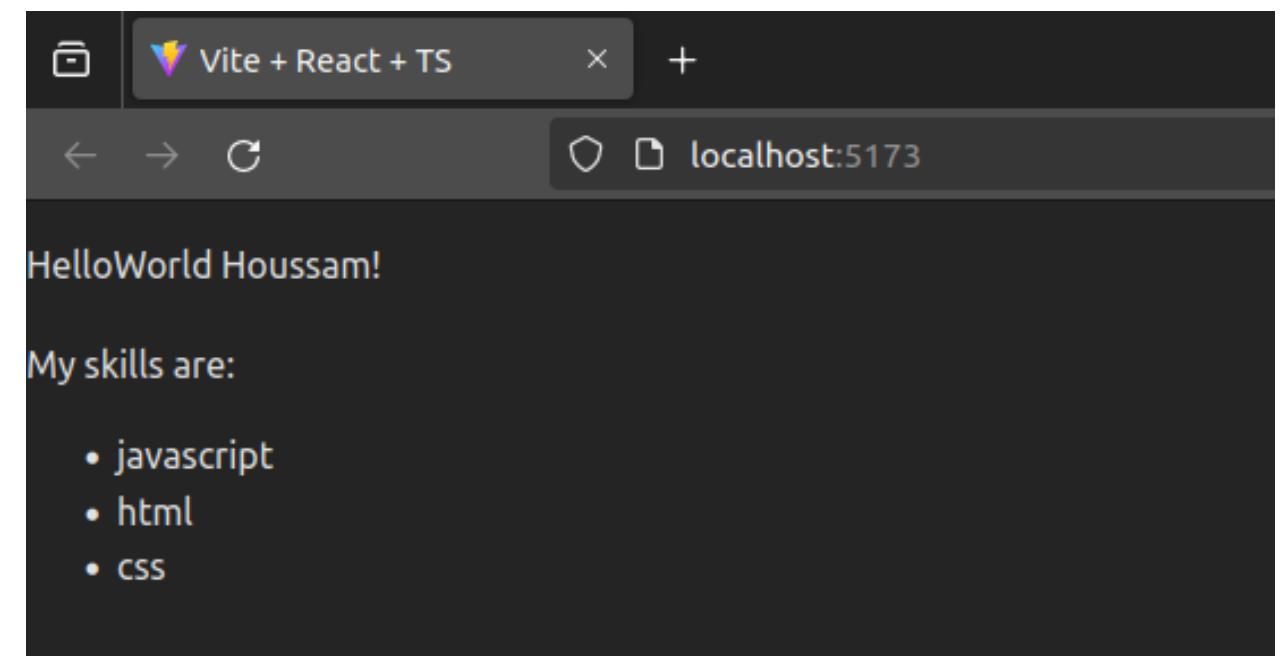
```
<!doctype html>  
<html lang="en">  
  <head>  
    <meta charset="UTF-8" />  
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />  
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
    <title>Vite + React + TS</title>  
  </head>  
  <body>  
    <div id="root"></div>  
    <script type="module" src="/src/main.tsx"></script>  
  </body>  
</html>
```

# React composant et Jsx

- jsx element syntax similar to html : <a> <div> <li> <img> <ul>
- inline css styling and className
- expression javascript intégrer au JSX
- re-utilisation des composants React

```
const HelloWorld = () => {
  const myName = "Houssam";
  const mySkills = ["javascript", "html", "css"];
  return (
    <div>
      <p> HelloWorld {myName}! </p>
      <p style={{marginTop: "20px"}}>My skills are: </p>
      <ul>
        {mySkills.map((item) => {
          return <li> {item} </li>;
        })}
      </ul>
    </div>
  );
};

export default HelloWorld;
```



+ Live examples

# Integration Javascript avec jsx

```
const HelloWorld = () => {
  const myName = "Houssam";
  const mySkills = ["javascript", "html", "css"];
  return (
    <div>
      <p> HelloWorld {myName}! </p>
      <p style={{marginTop: "20px"}}>My skills are: </p>
      <ul>
        {mySkills.map(item => {
          return <li> {item} </li>;
        })}
      </ul>
    </div>
  );
};

export default HelloWorld;
```

on utilise les accolades {} pour distinguer les variables javascript des simple chaine de caracteres

```
function App() {
  const products = [
    {
      name: "nom de produituis",
      image: "image-example.jpg",
      prix: 1200,
    },
  ];
  const userName = "riyad mahrez"

  return (
    <div>
      <Navbar userName={userName} />
      <div>
        <h1>hello {userName} !</h1>
        <ul>
          {products.map((element) => (
            <li>
              <p> {element.name}</p>
            </li>
          )));
        </ul>
      </div>
    </div>
  );
}

export default App;
```

un component react peu contenir un composant react

Jsx permet d'ajouter des variables js directement au elements .

la variable inserer au JSX doit être de type **chaine de caractere, number ou boolean**

en peut passer des variables de n'importe quelle type au composant fils

# State

**c'est quoi le state d'un composant =>** c'est les données qui détermine le composant UI

le state peut être : string, number ou object complexe.

**pourquoi on utilise un state =>** mettre à jour l'interface utilisateur dynamiquement pour réagir aux interactions utilisateur comme click sur button . ou bien changement de données correspondants

**comment l'utiliser**

```
const [stateVariable, setStateFunction] = useState(initialValue);
```

**stateVariable:** la variable qui contient la donnée.

**setStateFunction:** la fonction qui met à jour la donnée.

**initialValue:** la valeur initiale de la donnée.

# State

## Attention

si on declare une variable dans notre composant pour gerer un changement d'etat ala place d'utilise useState

***let stateVariable = initialeVariable***

*// quelque part dans la partie logic de composant*

```
const handleChangeState = () => {  
  stateVariable = nouvelleValeur  
}
```

*// incorrect*

le composant react n a aucune trace sur les changements de ca valeur donc il va afficher toujour le meme resultat

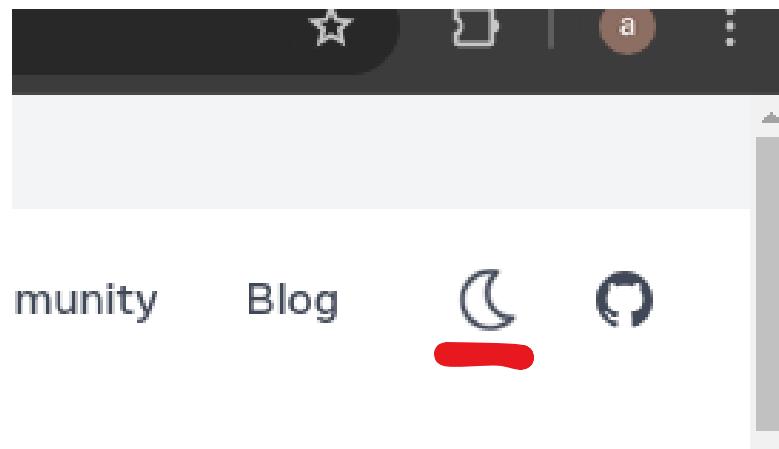
```
const [stateVariable, setStateFunction] = useState(initialValue);
```

***onst handleChangeState = () => {***

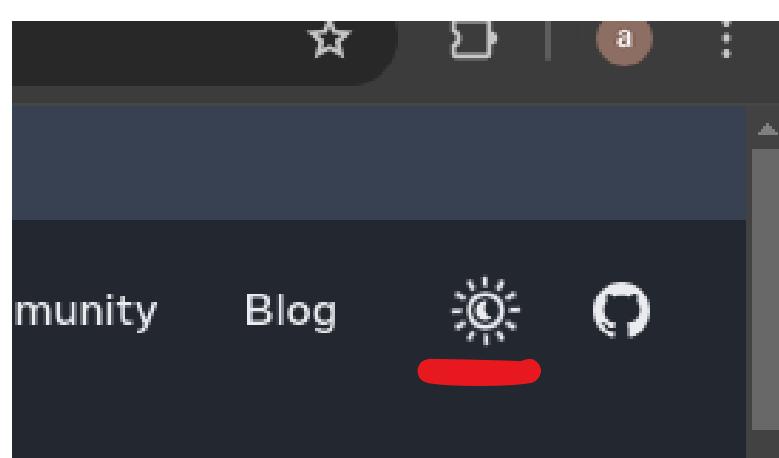
```
  setState(nouvelleValeur)  
}
```

*// correct*

# State



example switch de theme  
sombre / claire



```
● layout.tsx - 2-first-react-app - Visual Studio Code
File Edit Selection View Go Run Terminal Help
hello-world.tsx 1 App.tsx 4 layout.tsx 1 ●
vite-project > src > components > layout.tsx > ...
1 import { useState } from "react";
2
3 const Page = () => {
4     const [theme, setTheme] = useState("dark");
5
6     const toggleTheme = () => {
7         setTheme(prevState => prevState === "dark" ? "light" : "dark")
8     };
9     return (
10         <div>
11             <button onClick={toggleTheme}>
12                 {theme === "dark" ? "switch to Light" : "switch to dark"}
13             </button>
14             <div style={{ background: theme === "dark" ? "#333" : "#aaa" }}></div>
15         </div>
16     );
17 }
18
19 export default Page;
```

# Props

## Passage des données entre composant

se fait de composant Pere a composant fils

les données sont passé comme attribut jsx

peut etre de type string , number ,function ,object complexe ou un composant

```
type Props = {  
    userName: string;  
    isLoggedIn: boolean;  
};  
  
const NavbarA = (props: Props) => {  
    return (  
        <div>  
            {props.isLoggedIn ?  
                <p>{props.userName}</p> :  
                <button>Please Login</button>}  
        </div>  
    );  
};  
  
export default NavbarA;
```

```
function App() {  
  
    const userName = "riyad mahrez"  
    const isLoggedIn = true;  
      
    return (  
        <div>  
            <Navbar userName={userName} isLoggedIn={isLoggedIn} />  
            <div>  
                <h1>hello {userName} !</h1>  
            </div>  
        </div>  
    );  
}
```