HOMEWORK

NEED TO BE DONE
BEFORE THE CLASS

Watch the video and answer questions

https://www.youtube.com/watch?v=-DTUdOJv8w8&list=PL0X6fGhFFNTe_vJIlAQQo0IEgPgk9er3g&index=2

Q1 - At min 3, the author is talking about model (state) and UI : What does he mean by **MODEL** and **UI** ?
Q2 – Open the sandbox : Change code so that the warning "10 already" is displayed when count >=10

Q3 - When we click on the button, we are doing 3 tasks

- TASK 1 Update the count value (+ 1)
- TASK 2 Update the label
- TASK 3 Show warning if count >=10

For each task : are they related to MODEL or UI update?

Q4 - At min 4, the author presents ReactJS as declarative: What is the difference between **DECLARATIVE** and **IMPERATIVE** code

Q5 - Open the sandbox : this is exactly the same project, but using React : what are the differences ? Can you understand the code ?

Q6 - Why don't we need to update the UI in a React project ?

FRONT END DEVELOPMENT

WEEK 4 – An overview of React

CADT IDT

What will you learn today?

- ✔ The difference between **MODEL** and **UI**
- ✔ The difference between **IMPERATIVE** and **DECLARATIVE**

- ✔ What is **REACT JS** library?
- ✔ What is a **SINGLE PAGE** application?
- ✔ What is a **COMPONENT-BASED** approach ?

- ✔ Run a first React project

DEBATE

10 MIN

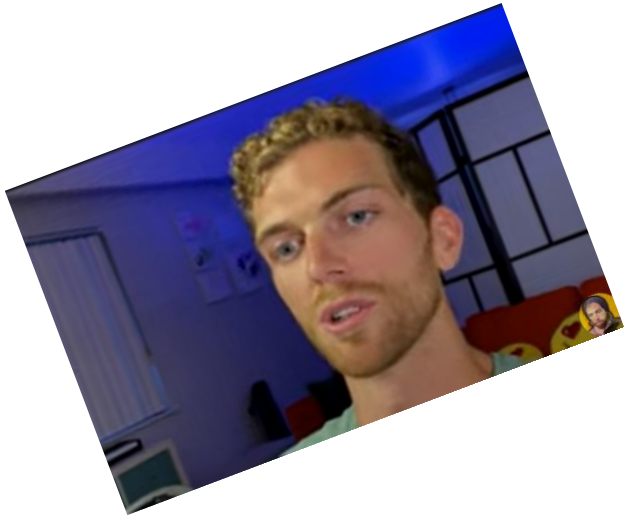
*Group discussion about the **homework***

Q1 – What is the difference between **MODEL** and **UI** updates ?

Discuss around the Vanilla JS sandbox

Q2 - What is the difference between **DECLARATIVE** and **IMPERATIVE** code?
Why don't we need to update the UI in React project ?

Discuss around the React JS sandbox



CORRECTION

Model vs UI

- ✔ The **Model** refers to the business logic of the application.
- ✔ The **UI** refers the graphical rendering (*HTML /CSS*) of the application

```
// Event functions
const handleCLick = () => {

  //  STEP 1 :
  count++;

  //  STEP 2 :
  label.textContent = count;

  //  STEP 3 :
  if (count >= 10) {
    setVisible(warning, true);
  }
});
```

STEP 1 updates the **model** of the application

STEP 2 updates the **UI** of the application *(here HTML change)*

STEP 3 updates the **UI** of the application *(here CSS change)*

Open the sandbox for correction

CORRECTION

Imperative vs Declarative

Vanilla JS (imperative)

***You** manage the UI update*

```
const handleCLick = () => {
  count = count + 1;

  label.textContent = count;

 });
```

React JS (declarative)

***React** manages the UI update*

```
const handleCLick = () => {
    setCount(count + 1);
};

return (
    <>
      <button onClick={handleCLick}>CLICK</button>
      <label class="tag">{count}</label>
    </>
  );
```


Managed by React

EXPLAIN

What is React JS ?

Explain

React is a JavaScript **library** for building **user interfaces** based on **components**

Explain

Explain

EXPLAIN

Why choosing React JS ?

- The **most popular front end library**

https://npmtrends.com/@angular-devkit/core-vs-react-is-vs-vue

- **Declarative** code
- **Component-**based
- **Cross-**platform

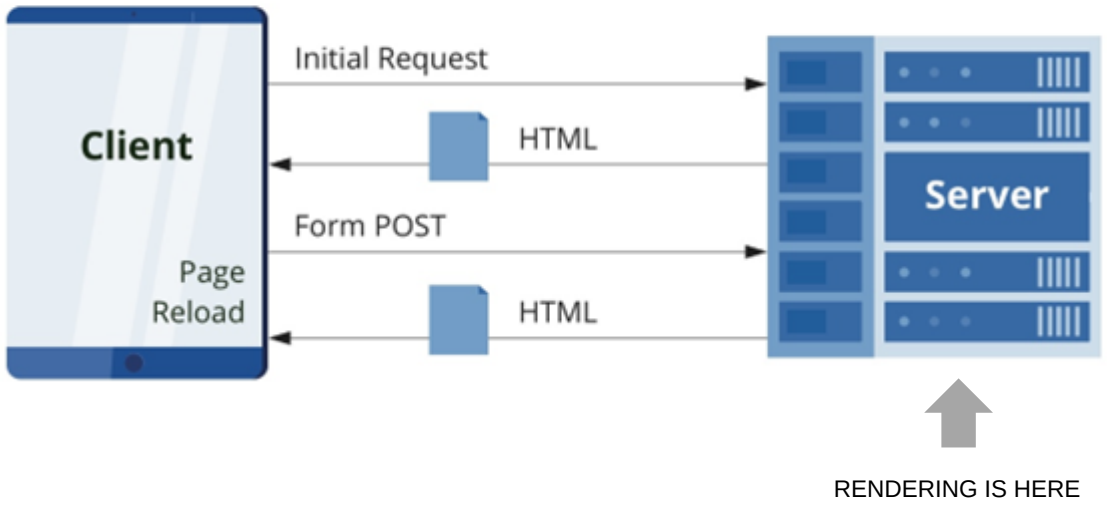
EXPLAIN

Other Front-end Libraries Or Frameworks

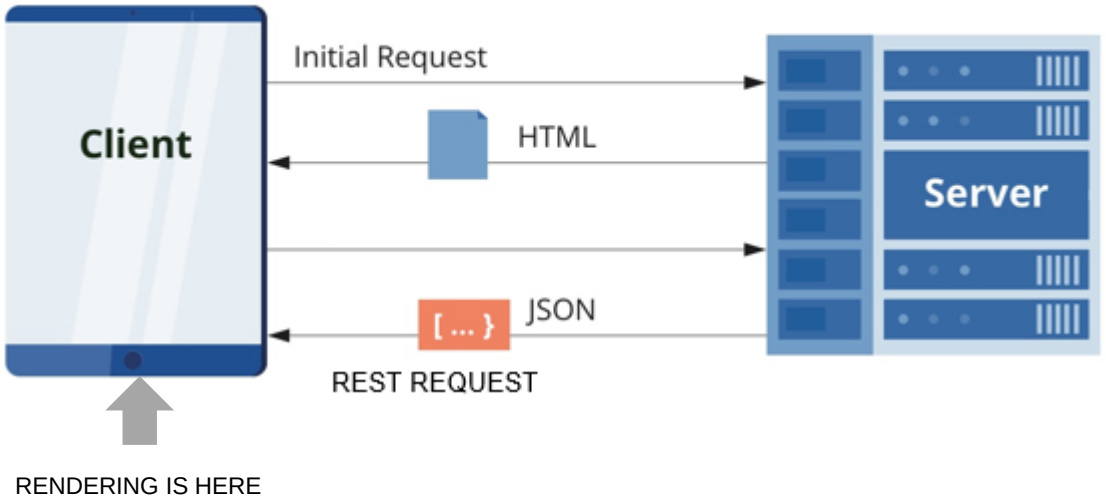
EXPLAIN

Server-Side Rendering VS Client-Side Rendering

Server-Side Rendering



Client-Side Rendering

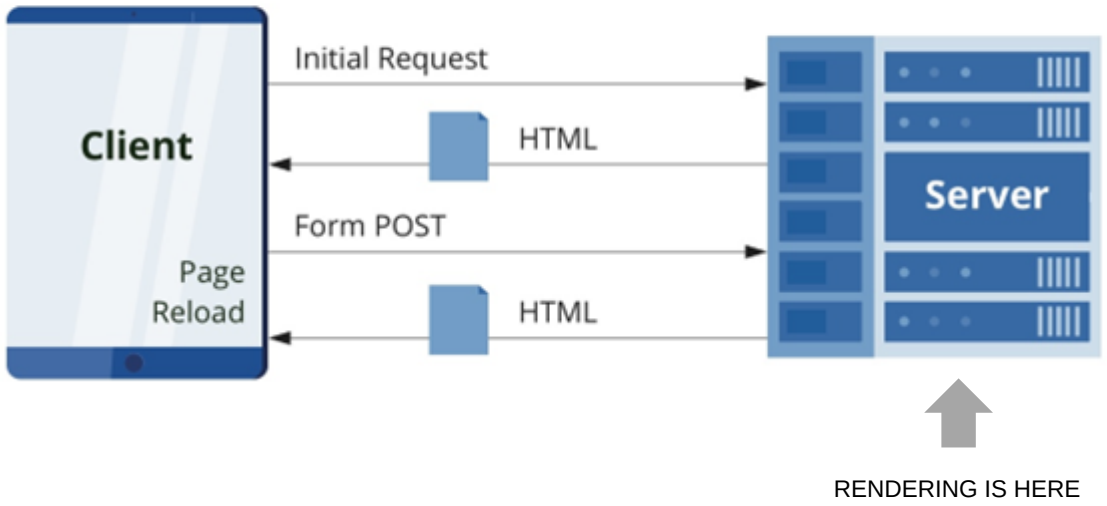


MORE INFO ? CLICK HERE !

EXPLAIN

Server-Side Rendering VS Client-Side Rendering

Server-Side Rendering



Lifecycle

- ✓ Pages are built + rendered **on the server side**
- ✓ Pages are delivered – fully formed – to the client

Impacts

- ✓ Pages loaded from server **for each** user interaction
- ✓ Fast execution on client

- ✓ **Lack of portability** (mobile…)

EXPLAIN

Server-Side Rendering VS Client-Side Rendering

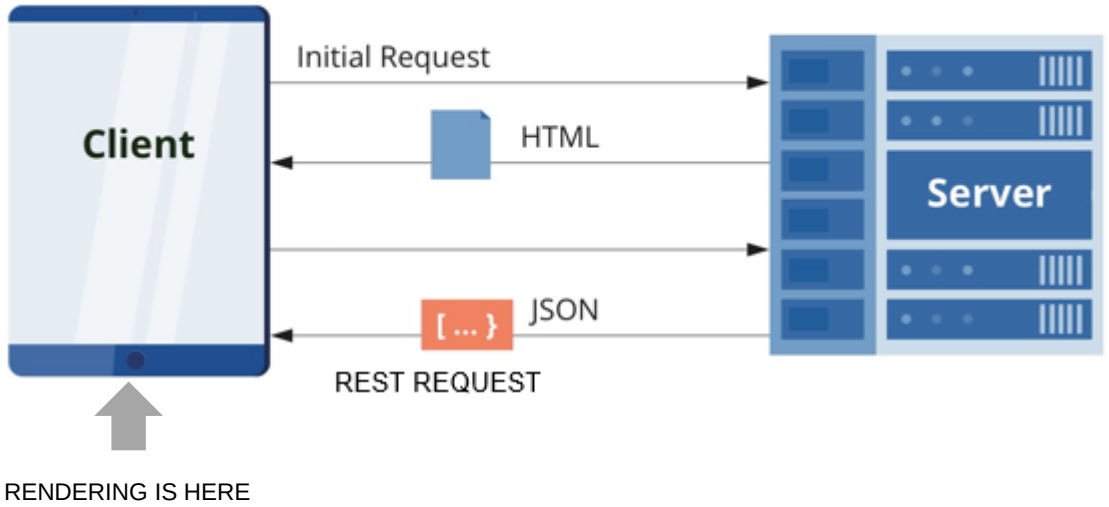
Lifecycle

- ✓ **Only data comes** from server
- ✓ DOM is **updated**, but not **reloaded**

Impacts

- ✓ The app dynamically update the current page **without reloading** the whole page
- ✓ A **smoother experience** to user
- ✓ The server acts as a **data API** provider

Client-Side Rendering



React JS comes with many features

At the end of this chapter you will know..

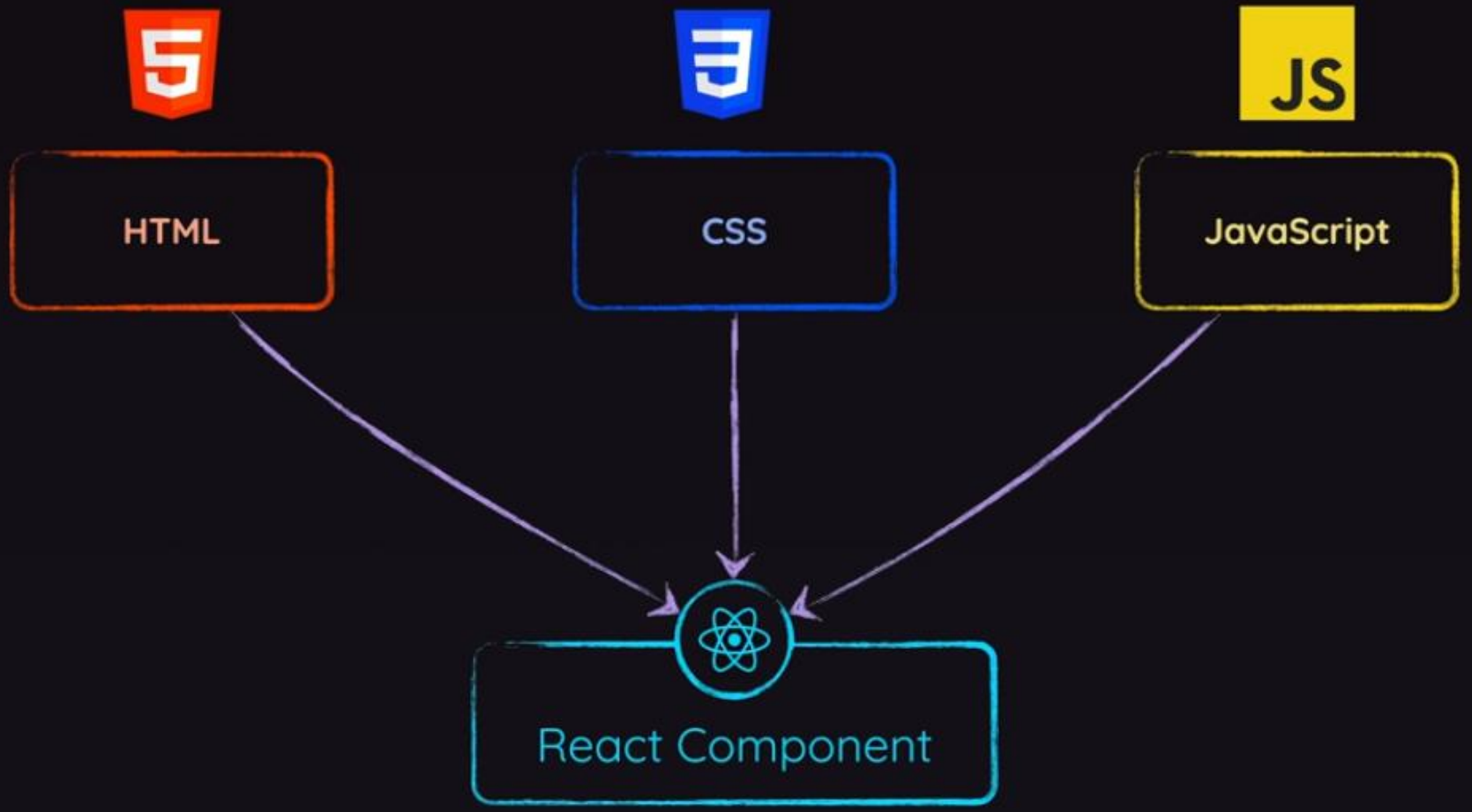
Four Fundamental React Concepts

What is a component?

A UI building block

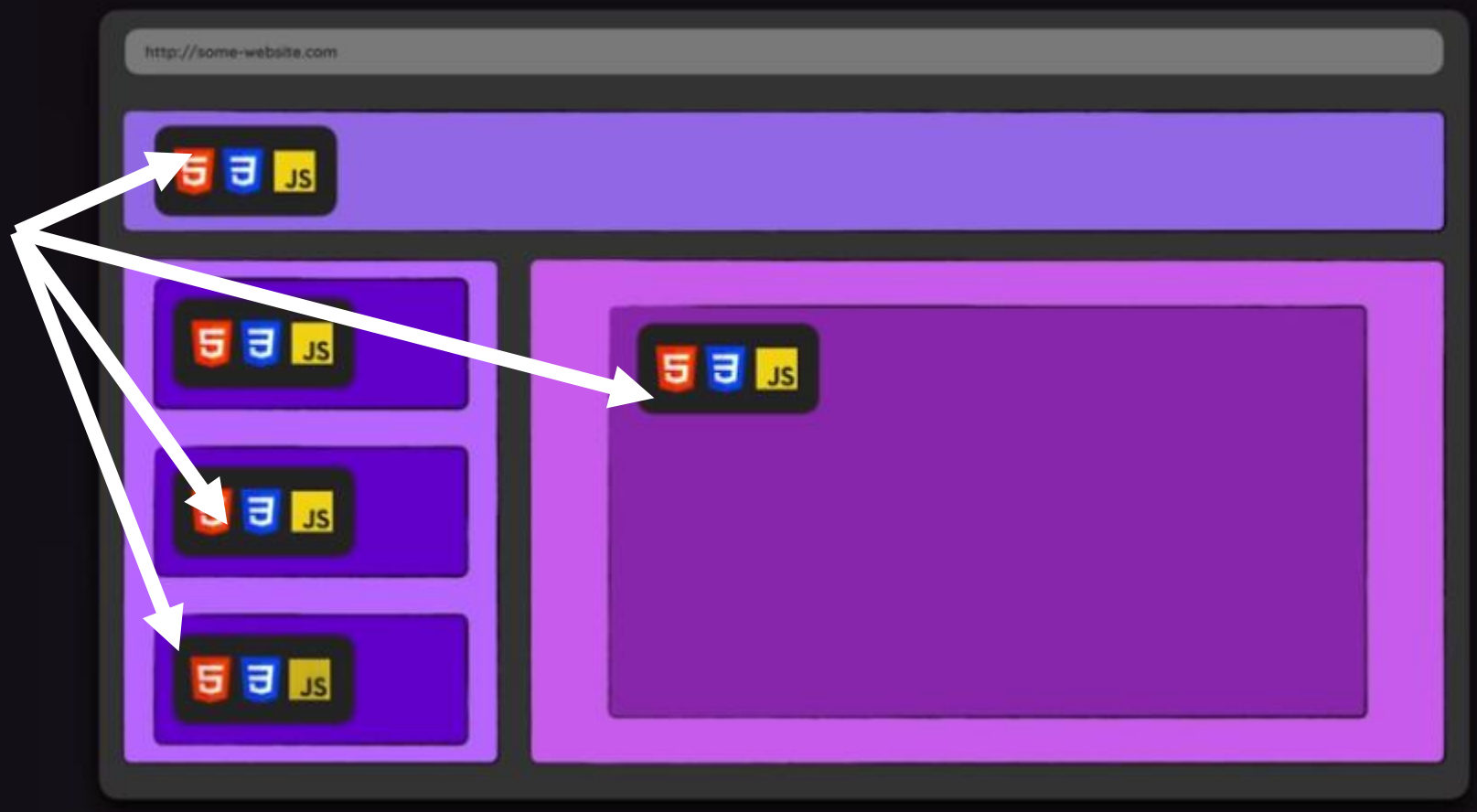
EXPLAIN

Components bring **HTML**, **CSS** and **JS** in a same place



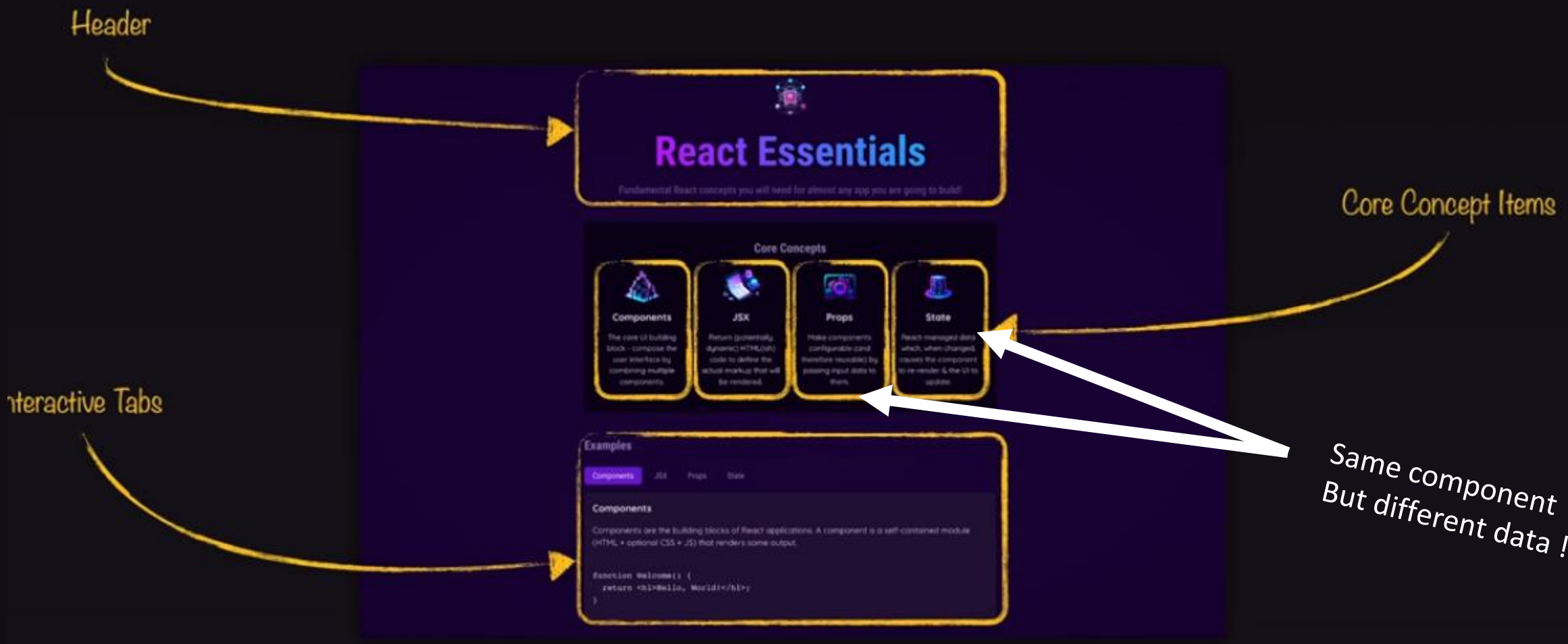
EXPLAIN

A same component can **be re-used** in many places of the application



EXPLAIN

A same component can **be re-used** with **different data**



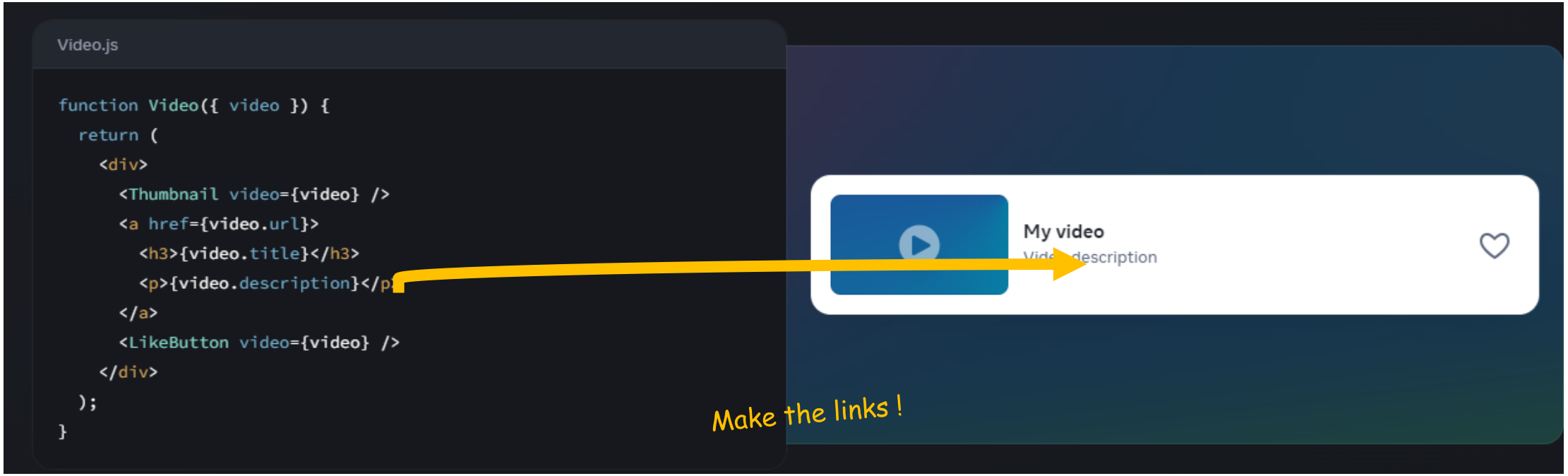
EXPLORE

Activity 1.1

A single React component…

Connect to https://react.dev/
Look at the component Video :

Q1 – Which **HTML elements** this component is composed of?
Q2 – Which **data** is displayed by this component?



EXPLORE

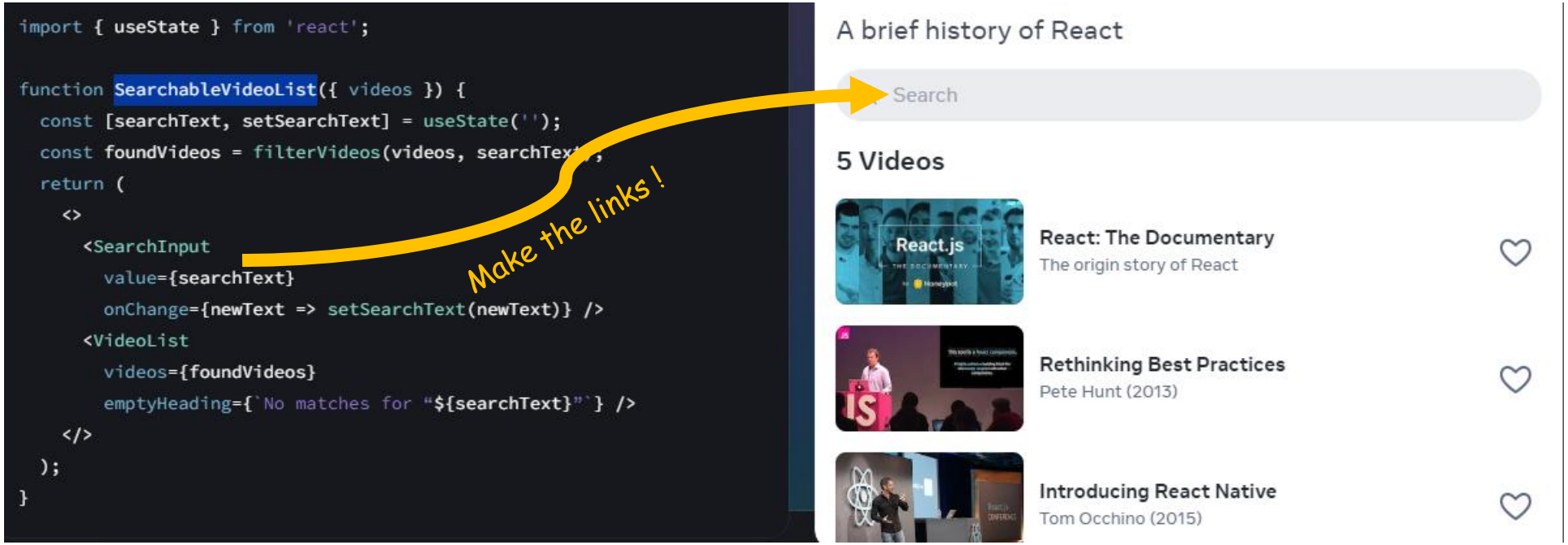
Activity 1.2

Nested React components

Look at the component SearchableVideoList :

Q3 – This component **uses other components** : explain the relation between those 4 components

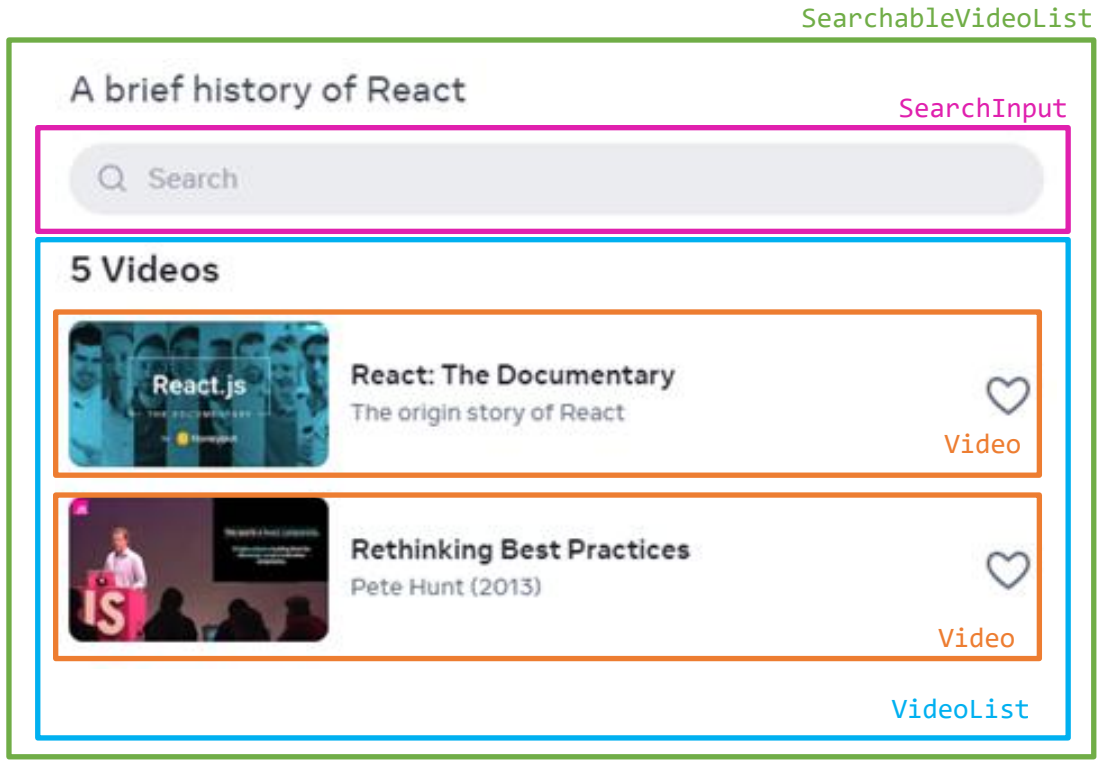
- `SearchableVideoList`
- `SearchInput`
- `VideoList`
- `Video`

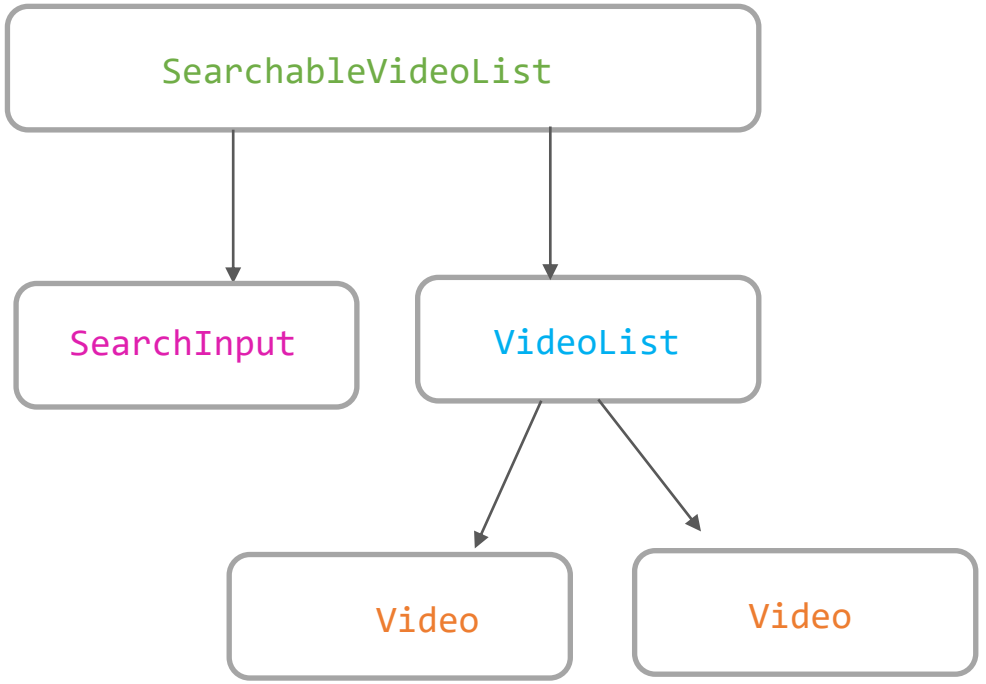


EXPLAIN

Component diagram

*A React app is composed of a **hierarchy** of components*





EXPLAIN

Benefits of a **COMPONENT** based approach

Reusable Building Blocks

Create **small building blocks** & **compose** the UI from them

If needed: **Reuse** a building block in different parts of the UI (e.g., a reusable button)

Related Code Lives Together

Related HTML & JS (and possibly CSS) code is **stored together**

Since JS influences the output, storing JS + HTML together makes sense

Separation of Concerns

Different components handle different data & logic

Vastly **simplifies** the process of working on complex apps

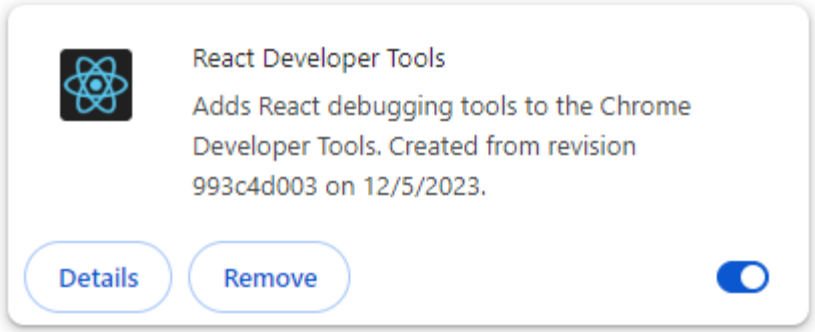
ACTIVITY

10 MIN

Activity 2

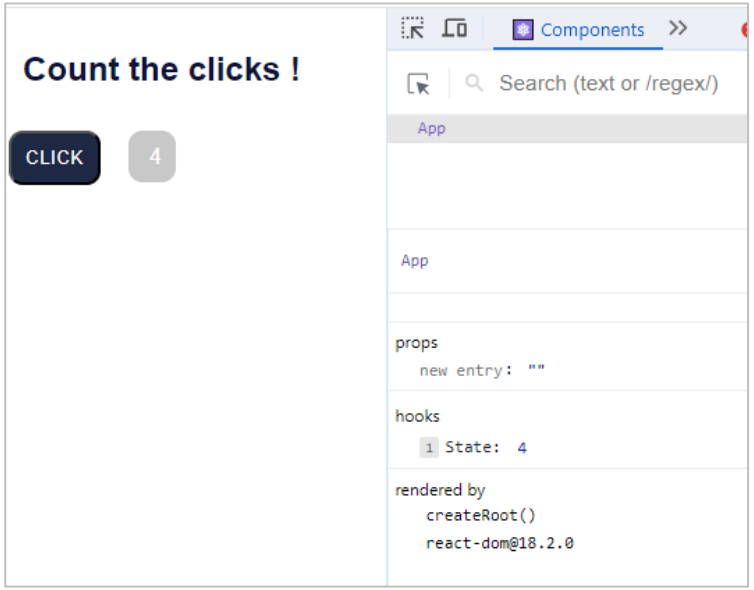
Chrome extension for React

1. Install the **React Developers Tools** extension on Chrome browser



2. Open a first React project : https://fnczpv.csb.app/

3. Open the component view and check the information you can get on each component.



ACTIVITY

10 MIN

Activity 3

Create a React app with Vite

✔ We follow the user guide: https://vitejs.dev/guide

1. Run NPM vite command to create a project

```
npm create vite@latest
```

2. Following the prompts !!

```
Project name: ... Choose a name
Select a framework: » React
Select a variant: » JavaScript
```

3. Run the NPM command to install all dependencies

```
npm install
```

4. Run the NPM command to run the project (developer mode)

```
npm run dev
```

Go further after the class...

- ✔ Look at the **W3C School** documentation !
- ✔ Watch this **new video** !
- ✔ Create your own React project following this guide!

- ✔ **Review** these slides
- ✔ **Review** correction code
- ✔ **Update your notes** on this course