**React:** A JavaScript library for building user interfaces. It uses JavaScript in the browser to update the page and the user interface without reloading the page.

Using “Just JavaScript” Typically isn’t a great option because

* Writing complex JavaScript code quickly becomes cumbersome
* Complex JavaScript code quickly becomes error-prone
* Complex JavaScript code often is hard to maintain or edit
* React offers a simpler mental mode

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.When working with React, in the end, you are writing declarative code which means you define the target your user interface states. Not the necessary steps to get there. Instead, React will figure out and perform the necessary steps. On the other hand, when writing JavaScript code, you are writing imperative code, not declarative which means, you are not defining the goal, but instead the steps needed to get there.

You can practice creating a react proyect by typing “react.new” in the search bar without installing anything locally.

**Installing React**

If you want to use install react locally, you will need to install [Node.js](https://nodejs.org/es) because it comes with tools like “vite”. There is an alternative to vite that’s named Create React App. Both of these give you simple commands to create a proyect.

When using vite, you have to write

To check if you have them installed use:

node -v

npm -v

npm create vite@latest react-proyect

Whenever you first install a proyect, you have to run

npm install to download and install all packages

npm run dev to start a development preview server

**Why do you need a special proyect setup to run react?**

React code is JavaScript code that Typically uses JSX (html in JavaScript). This code needs to be transformed and optimized by a build tool like Vite. Then the browser can read this code without JSX.

<script src="assets/scripts/app.js" defer></script> you can use the “defer” option to execute the script only after the rest of the html document has been read and parsed.

It’s also quite common to have a type=”module”. This lets us use the import sintax which allows us to import from script A to script B.

## Javascript refresher

React projects use a build process: the code you write is not he code that gets executed in the browser. Instead, the code is transformed before it’s handed off to he browser. The library that does this is called react-scripts and is located in the package.json

React projects use a build process because:

1. Raw, unprocessed React code won’t execute in the browser because React uses a special JSX feature which is not a default JavaScript feature.
2. The code would not be optimized for production. (e.g., not minifield)

#### Importing

When exporting variables, if you choose to export default “xxx”, you can only have one exporting value.

You can also import everything using import \* as util from “./util.js”;

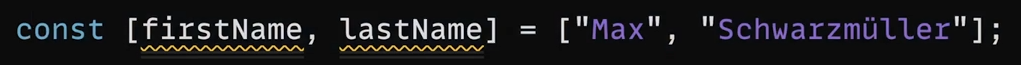
#### Destructuring

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Texto

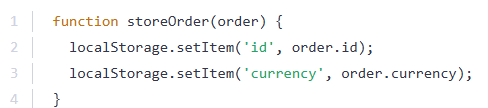
El contenido generado por IA puede ser incorrecto.



***Destructuring in Function Parameter Lists***

For example, if a function accepts a parameter that will **contain an object** it can be destructured to *"pull out"* the object properties and make them available as **locally scoped variables** (i.e., variables only available inside the function body).

Here's an example:



Instead of accessing the order properties via the *"dot notation"* inside the storeOrder function body, you could use destructuring like this:

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

The destructuring syntax is the same as taught in the previous lecture - just without creating a constant or variable manually.

Instead, id and currency are *"pulled out"* of the incoming object (i.e., the object passed as an argument to storeOrder).

It's very important to understand, that storeOrder **still only takes one parameter** in this example! It does **not** accept two parameters. Instead, it's one single parameter - an **object** which then just is destructured internally.

The function would still be called like this:

1. storeOrder({id: 5, currency: 'USD', amount: 15.99}); // one argument / value!

**Destructuring allows you to easily access the values of arrays or objects and assign them to variables.**

Here's an example for an array:

Texto, Carta

El contenido generado por IA puede ser incorrecto.

And here for an object:

Texto, Carta

El contenido generado por IA puede ser incorrecto.

Destructuring is very useful when working with function arguments. Consider this example:

Texto

El contenido generado por IA puede ser incorrecto.

Here, we only want to print the name in the function but we pass a complete person object to the function. Of course this is no issue but it forces us to call personObj.name inside of our function. We can condense this code with destructuring:

Texto

El contenido generado por IA puede ser incorrecto.

We get the same result as above but we save some code. By destructuring, we simply pull out the name  property and store it in a variable/ argument named name  which we then can use in the function body.

#### The spread operator

If we want to merge the hobbies list with another newHobbies list, I could use the special spread operator which is the three dots and then the name of the first array. These three dots will pull all the elements of the array and add them as separate values to this new list.

Texto

El contenido generado por IA puede ser incorrecto.

#### ES6 Arrow Functions

Read more: [Arrow function expressions - JavaScript | MDN](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions)

Arrow functions are a different way of creating functions in JavaScript. Besides a shorter syntax, they offer advantages when it comes to keeping the scope of the this  keyword (see [here](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions#No_binding_of_this)).

Arrow function syntax may look strange but it's actually simple.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

which you could also write as:

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

becomes:

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

**Important:**

When having **no arguments**, you have to use empty parentheses in the function declaration:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

When having**exactly one argument**, you may omit the parentheses:

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

When **just returning a value**, you can use the following shortcut:



Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.That's equal to:

#### Methods of arrays

The following page gives a good overview over the various methods you can use on the array prototype - feel free to click through them and refresh your knowledge as required: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array>

Particularly important in this course are:

* map()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map>
* find()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/find>
* findIndex()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/findIndex>
* filter()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter>
* reduce()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce?v=b>
* concat()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/concat?v=b>
* slice()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/slice>
* splice()  => <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice>

## React Essentials

#### Components

Components are reusable building blocks which you can create and which you then combine to build the overall user interface. **React apps are built by combining components.**

Why components?

* Reusable building blocks: create small building blocks and compose the UI from them.
* Related code lives together: Related HTML, JS and CSS code is stored together.
* Separation of concerns: different components handle different data & logic. It vastly simplifies the process of working on complex apps.

#### Setting up the starting project

First you have to have node js. You can use this commands in cmd to check if you have them installed

node -v & npm -v

Then you have to use npm install before running the project.

Then you use

npm run dev

to start the development server. This development server will allow you to visit a preview of the website.

The files have the extension .jsx because it is a JavaScript file that uses non-standard JavaScript syntax. It uses a JavaScript syntax extension called JSX which stands for JavaScript Syntax eXtension. This extension allows developers to describe and create HTML elements by writing HTML markup code inside of JavaScript files. But it is a feature that’s not supported by browsers. Instead, the code is transformed by the development server before it reaches the browser.

#### Rules of components

* Name should start with uppercase character
* Returns “Renderable” content

Look how I write the header in a function and utilize it in the html.

Texto

El contenido generado por IA puede ser incorrecto.

#### Loading images

Using src/… is not the best way to add images because it is pointing to an image in the src folder and the image might get lost once we prepare this react project for deployment.

So, to load images, you have to use an import statement

#### Props

React allows you to pass data to components via a concept called **Props**.

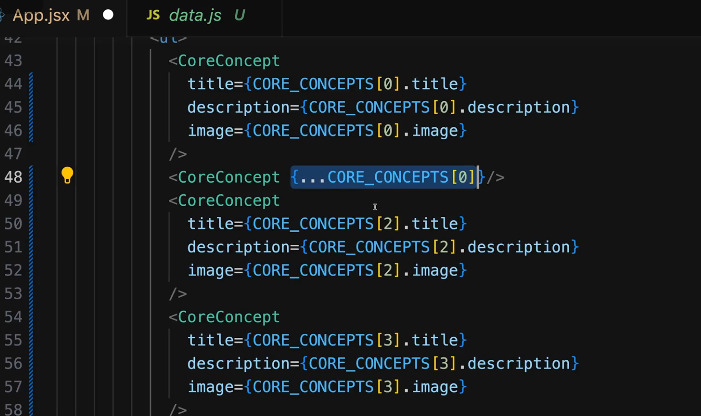
Imagen que contiene Diagrama

El contenido generado por IA puede ser incorrecto.

In normal JavaScript functions, you would simply add one or more parameters to accept input values and then use those input values. But when working with React component functions you can only accept one parameter which typically is named props. Since it’s react who will execute this function, its him who will pass a value for this props parameter to the function when it calls it. And the value that will be passed for this parameter to the function by react will be an object that has all the key value pairs that were assigned here.

Diagrama, Texto

El contenido generado por IA puede ser incorrecto.

If your prop names are similar to the properties of the object that you are using, you can use the spread operator to pull out all the key value pairs of an object

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

You can also use object destructuring by adding opening and closing curly braces as an input parameter

## Buttons

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.There is one prop which you will always get, the special, built-in children prop. This prop refers to the content between you component text.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Diagrama, Teams

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Este funciona como valor de salida

Interfaz de usuario gráfica, Aplicación, Teams

El contenido generado por IA puede ser incorrecto.

Diagrama

El contenido generado por IA puede ser incorrecto.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.There is a concept called **state** which is to register variables that are handled by react and that are updated with the help of a function that’s provided by react that will also tell react that data changed.

State function must be imported from the React library

All the functions that start with **use** are called *React hooks.*

Texto

El contenido generado por IA puede ser incorrecto.Also, you must call those hook functions directly inside of the component function. Not nested inside of a component function.

Diagrama

El contenido generado por IA puede ser incorrecto.Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

## Classes

If you want to add something like a class to a button, you need to use

className = “”

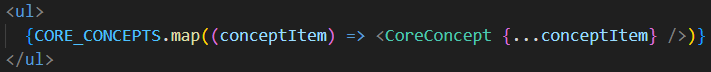
Texto

El contenido generado por IA puede ser incorrecto.

### Outputting list data dynamically

Texto

El contenido generado por IA puede ser incorrecto.



You can transform arrays in JavaScript with the help of the built-in map method. Which produces a new array based on the existing array. For that, map takes a JavaScript function like an arrow function and that function will receive an array item as an input. This arrow function will now be executed by JavaScript for every item in that array. Inside of that arrow function we want to transform this item to the core concept component being used as JSX Also remember that when dynamically outputting a list of elements, every element should receive a unique "key" to help React tell the list items apart.

Texto, Escala de tiempo

El contenido generado por IA puede ser incorrecto.

Diagrama

El contenido generado por IA puede ser incorrecto.

## Introduction to fragments

There is a common error that is: JSX expressions must have one parent element. This problem is that in the return statement you are returning more than one sibling elements. You cannot do that. You have to return one parent element with n number of children. This is because **in JavaScript you cannot return two values by just wrapping them with parentheses and just splitting them in two lines.**

For this, React gives you an alternative: an special fragment component which you can use as a wrapper but you don’t want to render an actual element such as a div. You have to import **Fragment** from react and replace the outer div with Fragment. **Or** you can simply use <> … </> without importing fragments.

## When should you split components?

Having a single component that deals with a lot of different thing typically is a sign that you might want to split it up and break it up into multiple smaller sub Components.

If you are managing useState, whenever that state is updated using the set… function, **the entire component function to which this state belongs to is re-executed** so that react is able to find out if something changed in the jsx code and is able to update the screen. **Also, if there are other components imported in there, those will also re-execute.**

Being able to identify good places for extra Components and being able to split up your big components into smaller components and to separate responsibilities is an important skill for every react developer.

Usually you want to separate the component by features

### Props are not forwarded automatically

When you are setting attributes on a custom component, those props are not automatically forwarded to the JSX code used inside of that component.

To solve this, we can use a pattern called **forwarded props** or **proxy props**

When de-structuring the props of an export default function using parentheses and corchetes, we can also add some special built-in JavaScript syntax where we add three dots and then, any identifier of our choice (in this case we called it props but it can be whatever name).

These three dots tells JavaScript to collect all other props that might be received on this section component and merge them into a props object.

Imagen que contiene Gráfico

El contenido generado por IA puede ser incorrecto.

This will now ensure that all extra props that might be set on our custom section component will be forwarded to this built-in section element.

En React (JSX), la diferencia entre **class** y **className** viene de que JSX es en realidad **JavaScript**, no HTML puro.

**Por qué no usamos class en JSX**

* En JavaScript, class es una **palabra reservada** (sirve para declarar clases en ES6).
* Si React dejara usar class en JSX, chocaría con el uso normal de class en JavaScript, causando errores o ambigüedad.
* Por eso, React usa la propiedad className para referirse a la clase CSS.

**Cómo funciona**

| **HTML normal** | **JSX (React)** |
| --- | --- |
| <div class="mi-clase"></div> | <div className="mi-clase"></div> |

En el navegador, **React convierte className a class** en el HTML final que renderiza.

**Ejemplo en React**

// JSX

function App() {

return <div className="contenedor">Hola</div>;

}

// HTML que genera en el navegador

<div class="contenedor">Hola</div>

💡 **Dato:** En frameworks como Angular o Vue sí puedes usar class porque manejan plantillas HTML más directas, pero React trabaja con JSX que es más cercano a JavaScript.

## Closer Look: public/ vs assets/ for Image Storage

The public/ Folder

As shown in the previous lecture you can store images in the public/ folder and then **directly reference** them from inside your index.html or index.css files.

The reason for that is that images (or, in general: files) stored in public/ are made **publicly available** by the underlying project development server & build process. Just like index.html, those files can directly be visited from inside the browser and can therefore also be requested by other files.

If you try loading localhost:5173/some-image.jpg, you'll be able to see that image (if it exists in the public/ folder, of course).

The src/assets/ Folder

You can also store images in the src/assets/ folder (or, actually, anywhere in the src folder).

So what's the difference compared to public/?

Any files (of any format) stored in src (or subfolders like src/assets/) are **not made available to the public**. They can't be accessed by website visitors. If you try loading localhost:5173/src/assets/some-image.jpg, you'll get an error.

Instead, files stored in src/ (and subfolders) can be used in your code files. Images imported into code files are then picked up by the underlying build process, potentially optimized, and kind of *"injected"* into the public/ folder right before serving the website. Links to those images are automatically generated and used in the places where you referenced the imported images.

Which Folder Should You Use?

You should use the public/ folder for any images that should **not be handled by the build process** and that should be **generally available**. Good candidates are images used directly in the index.html file or favicons.

On the other hand, images that are used **inside of components** should typically be stored in the src/ folder (e.g., in src/assets/).

### Creating new components

When creating new components, you should create a folder in src