# General Information

## What is ReactJS?

ReactJS is a javascript framework that makes the website feel like its an application. For example take Netflix in the browser, it feels like using a native application instead of a traditional website

Traditionally in web apps you click a link and wait for a new page to load.

And this is where JS comes to the rescue because JS allows us to run logic in the browser. We can manipulate the DOM with JS and this allows us to modify what the user sees without fetching a new HTML page

ReactJS is a client side Javascript library its all about building modern and reactive UI.

## Why react instead of just Javascript?

Basically we can write less code that does more and with JS we have to write every single step that needs to be created.

Ex: In a todo app > User click delete buton > crate model > create 2 buttons in the modal > add event listeners to these buttons etc. **This is an imperative approach**

Also react allows us to write code in a declarative, component focused approach

## Building Single page applications

We often create something called single page applications. React can control parts of the HTML page like a sidebar, widget but we can control the entire page with react even switch pages

A good example is Netflix where the Server sends only one HTML page thereafter react takes over and controls the UI

## React.js Alternatives

React is a lean and focused component based UI Library. Certain features like routing are added via community packages

Angular is another popular frontend framework that is also component based and it comes with more features and also uses TypeScript

Vue js is a mix between react and angular which si also component based with less features like angular but more than react

# Javascript Refresher

## Let, const, var.

Let and const are local scoped and var is global scoped.

Use let for something that will change in the future and use const for “constant” variables.

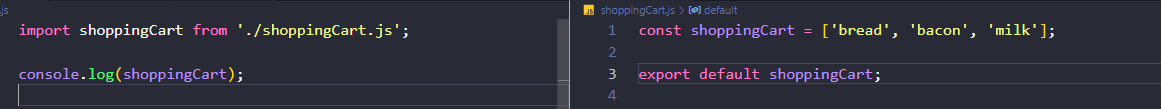
## Arrow functions

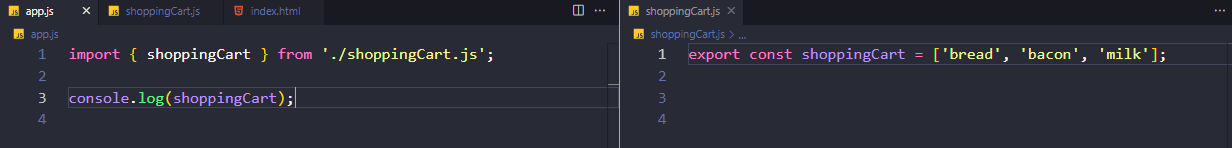
Function logMyName = (name) => console.log(name);

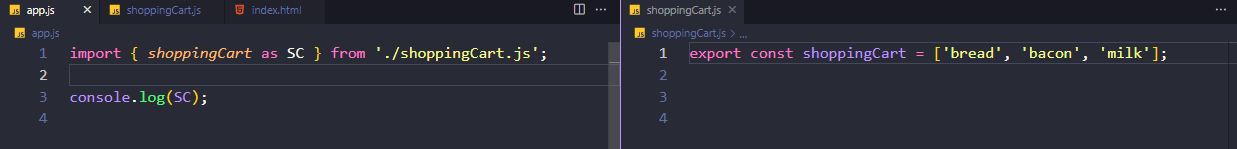
Arrow functions don’t have their own this keyword

## Export and imports

Below we can find default export (only one item will be exported from the exporting module)

Below are named exports



And another example of named exports  


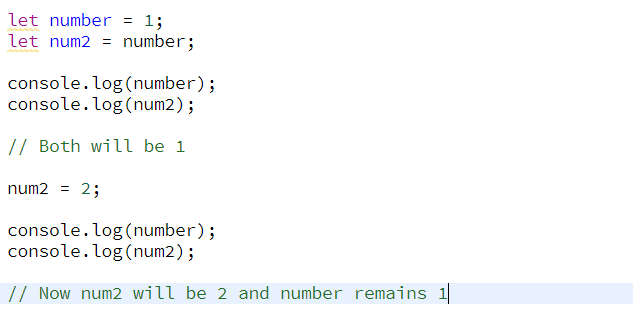
Don’t forget, in order for the modules to work we have to write type=’module’ in the script tag from the html file

## Classes Properties and Methods

Come back to this.

## Reference vs Primitive Types

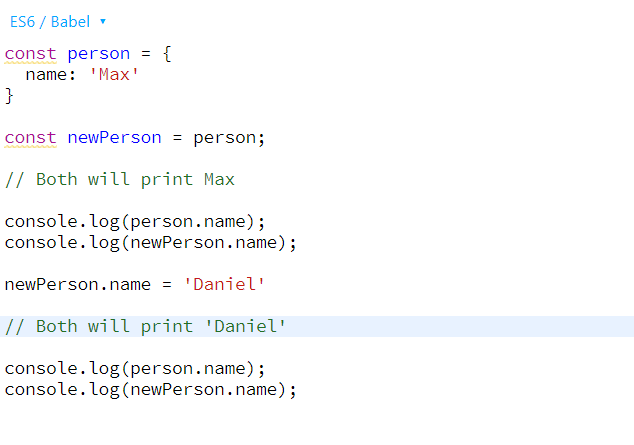
When we copy primitive types it will create a real copy of that primitive.



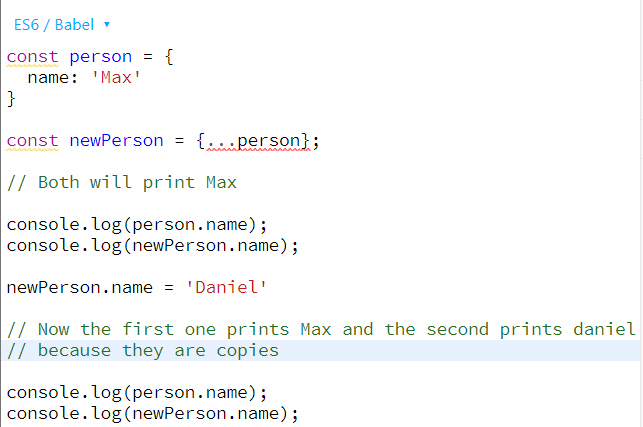
The two will be different because they are primitive types.

In the example below we use objects which are reference types. When we say newPerson = person we don’t actually make a copy we point that newPerson to the place in memory where person is. And now if we change one both will change because they POINT to the same place in memory.

This is very important in react because we might run into some bugs where we change something and that something changes everywhere so to circumvent this behaviour we want to make an actual copy of the object and we can do that using the spread operator



Example using spread operator.



# React Basics & Working with Components

## Module Introduction

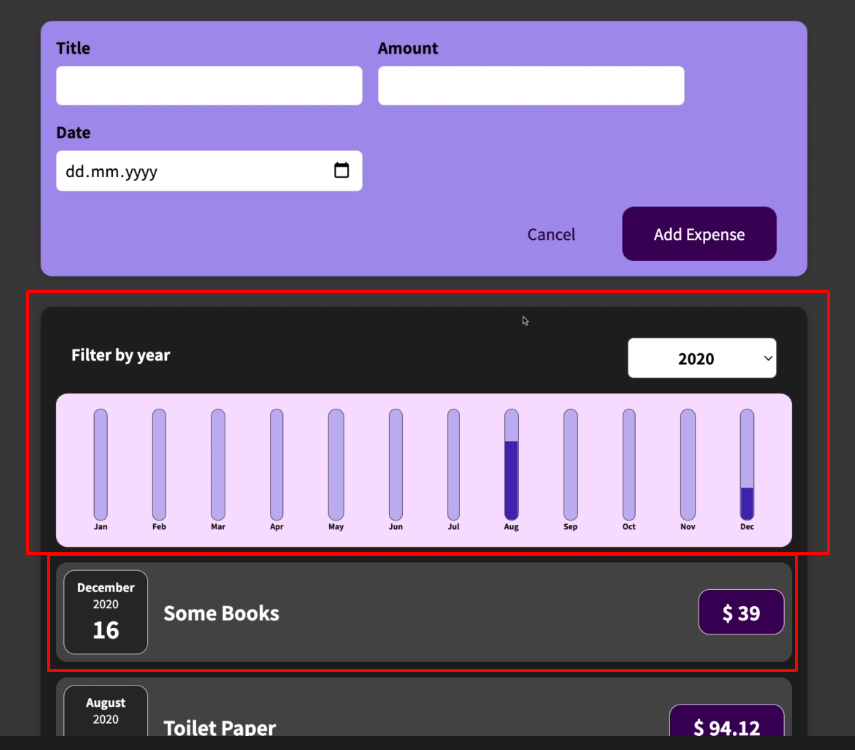
We will learn how about basics and most importantly how to build component driven user interfaces.

## What are components and why is React all about them?

React is a JavaScript library for building UI and we use react because it simplifies using UI.

React makes building complex, interactive and reactive user interfaces simpler and for this react embraces a concept called “Components” because all UI are made up of components

What is a component?



Everything here is a component and we combine them together to build a UI.

Why components?

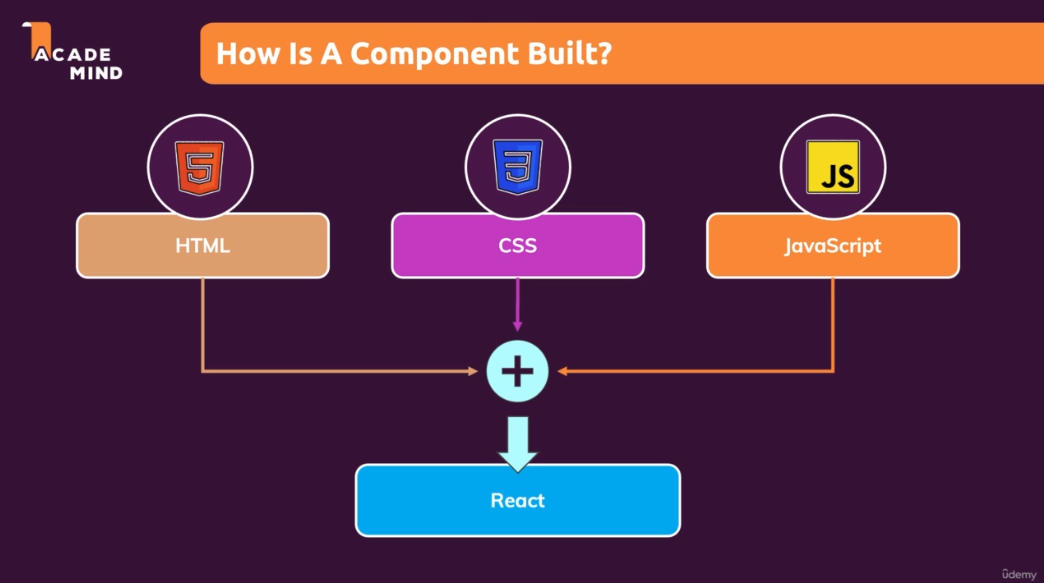
For reusability => for DRY Principle

For Separation of Concerns => Don’t do too many things in one and the same place (function)

## React code is written in a Declarative Way

How is a component built?

In general is important to keep in mind that UI is about HTML, CSS and JS. And these components are about combining HTML CSS and JS when we work with react we combine all of these to create components.



React allows you to create re-usable and reactive components consisting of HTML , CSS and Javascript.

React uses a declarative approach to build these components which means that you will not tell react that a certain HTML element should be created and inserted in a specific place of the UI as you would be doing it with vanilla JS instead you will always define the desired end state and its react job is to figure which element of the actual webpage might be added or removed or updated.

## Creating a new React App

The easiest way to start a new react project is to use a tool called ‘create-react-app’. These are preconfigured folders with some basic react code files and most importantly a bunch of config files that helps build react files for production use. This will also give us a dev environment.

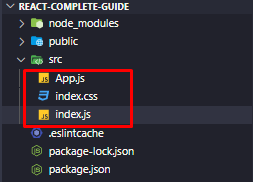


This is all the command line code we need to setup a new react app.

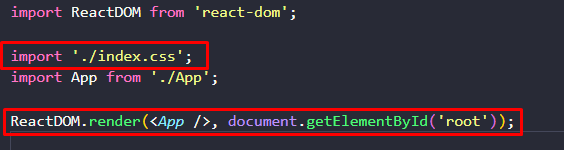
However for the project we used a simplified version of this which removes all the bloat.

## Analyzing a Standard React Project

Notice that we have 2 js and a css file.



Whenever we start the project the index.js file will the first file that will be executed.

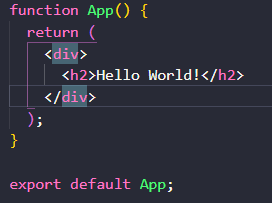


In this index.js file there is some weird code that in normal JS wouldn’t work for example importing a .css file into a JS file or the last line where we run some sort of HTML code in JS.



This weird syntax is called JSX.

Another strange thing is what is inside the App.js file



Notice this is a function that returns HTML Code that is also JSX invented by the React team that allows us to use it in our project.

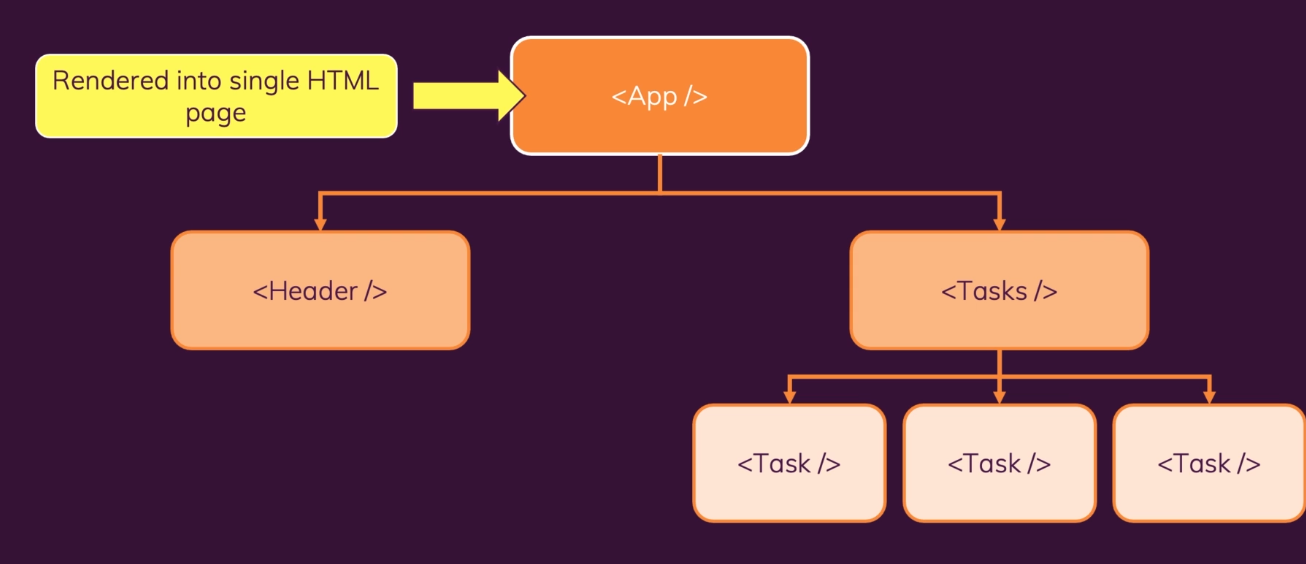
## What is JSX?

Stands for Javascript XML and is basically HTML code in Javascript. This only works because there are transformation steps behind the scenes.

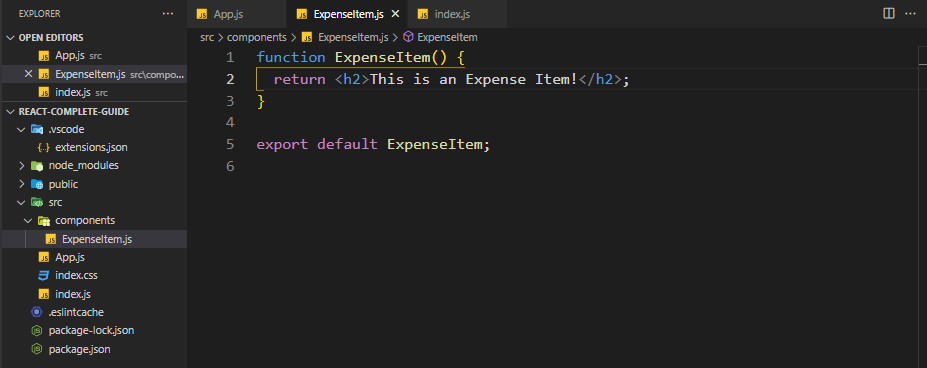
## Building our first custom component

To organize our code we will add a new folder in our src folder that will host all our components (might be hundreds in a big project).

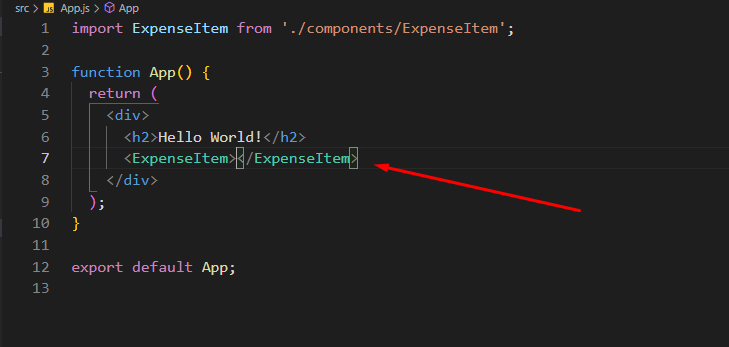
Ultimately with react we create a component tree



The convention of naming here is camelCase.



Now we are not going to import this in the index.js file because we only import our app.js file there but instead import it in the App.js file and insert it in the code there.



In React we are not allowed to have 2 root divs. Take the example below:



We cannot have another dive next to the main div but there are some workarounds in which we will dive in later.

## Adding Basic CSS Styling

For this we are going to use CSS which we will put next to our JS file. The css we will get from the course.

## 

After that we have to import the CSS file into our component and instead of adding class to the JSX we add className.

We can also use SASS but for now I’m going to use CSS

## Outputting Dynamic Data and Working with Expressions in JSX

## 

This being Javascript we can use normal variables and functions to output some sort of dynamic data for now. Keep in mind that for date we have to use toISOString() method because otherwise it will return an object and react cannot display that

## Passing Data via ‘props’

## In React we can make our components reusable by adding parameters and a concept called props. So in react our custom components can have attributes

## In every react component we will have a parameter called props which will be an object that data is stored in

## The data for our expenseItem components will be in the App.js file

## 

## Here we have some dummy data from the course and above is the way we write the parameters that our newly displayed component will use

## Ex. title={expense[0].title} etc.

## Now this is only the first part of the job the next is to make sure that our component accepts this data

## ’

## In the props objects we get key, value pairs.

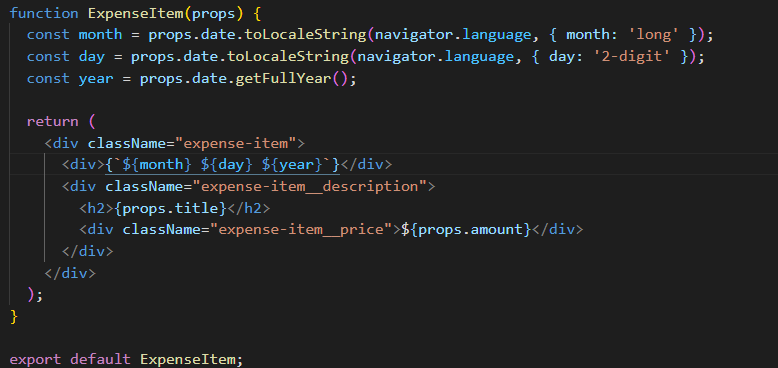
## The key which you access in on your prop object has to have the same name as you picked for your new component attribute.

## And this is how you share date between react components.

## You will use props all the time, it allows you to make your components reusable and it allows to pass date from another component to this component

## Adding “normal” Javascript logic to our components

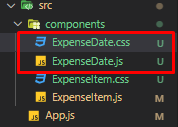
It is a good practice to separate the logic from the return function so we can format the date inside our function and in the return statement just enter what we formatted. I’ve used toLocaleString instead of internationalization API here because that’s what the teacher used in the course.



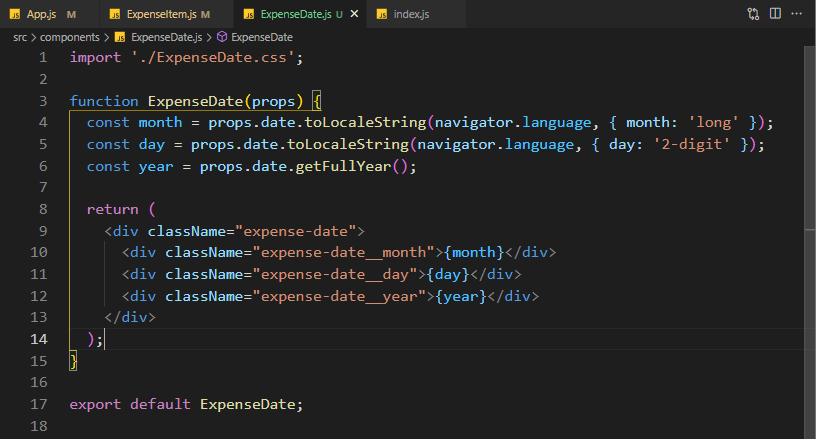
## Splitting Components into Multiple components

You will notice that in any project, your components will become bigger and bigger that is why react has this component concept. There is not a hard rule when to add a new component but we can argue in our expense component that it is getting a bit too big so the calendar item might be considered as a separate component.

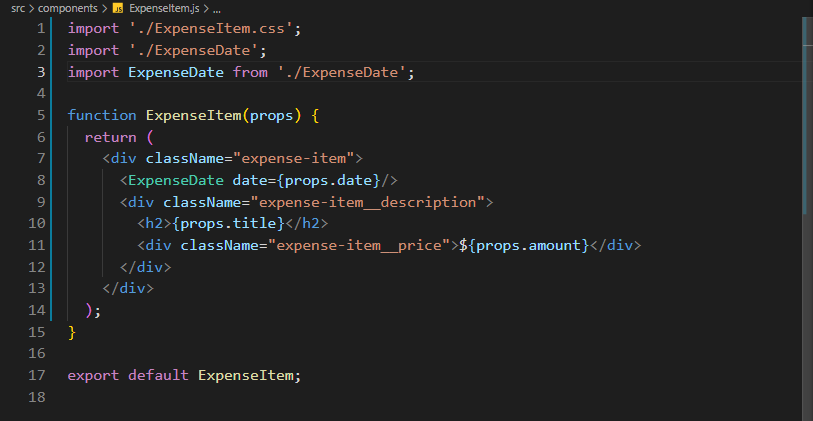
So like we did before we create a new JS and CSS File



The CSS is provided by teacher and the js file we will make ourselves



And now in the ExpenseItem where the date is needed we just write the component with a simple closing tag (we can do this if there is nothing between the tags)



This can get a bit confusing because we are funneling data trough multiple components.

First we get the data from App.js which we pass to ExpenseItem.js which we pass to ExpenseDate.js

It is totally fine to pass data between multiple components using props that’s how props work