Week retrospective

What did you achieve last week?

Try to remember the main things (projects, tools, new methods)

What worked well? What did not?

- What should you start doing?
- Stop doing?
- Continue doing?

Reactjs

week 2

Margit Tennosaar

Last session

- Overview of ReactJS
- Setting up a React environment
- Exploring the file structure
- Creating a basic React app

This session

- Components
- Introduction to Props and State
- Handling events
- Basic Hooks usage (useState)
- List and keys

Creating a new react project

```
npm create vite@latest

or

npx create-vite your-project-name --template react
```

```
npm install
```

```
npm run dev
```

```
git init
```

NOTE: from now on we will use npm run dev script to run our app in localhost.
(Basically, it means that you must forget liveserver extension ☺)

https://vitejs.dev/guide/#scaffolding-your-first-vite-project

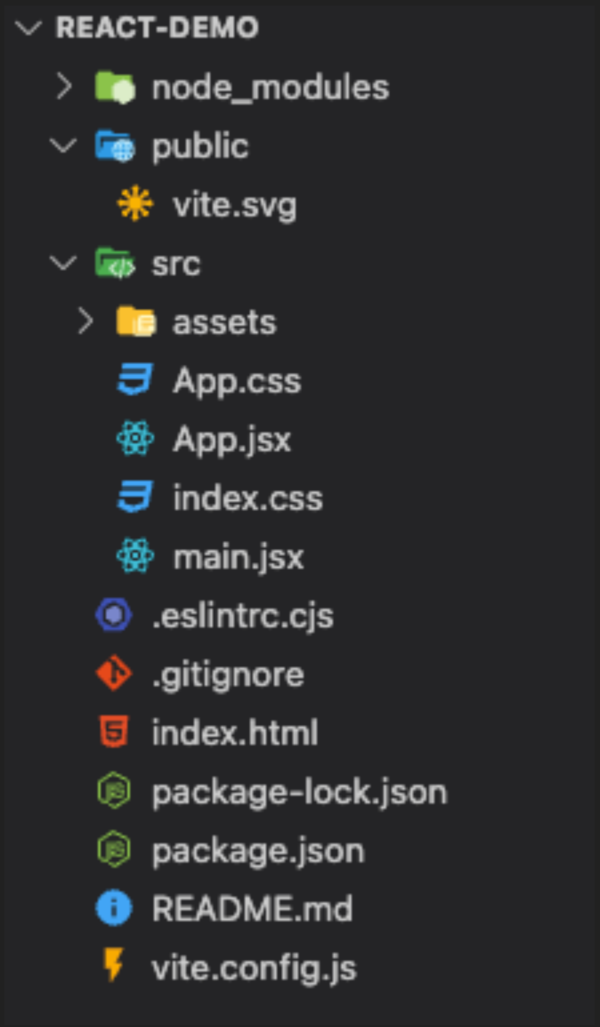
Created files

src - React components

main.js: renders the application typically with no need to touch the root component

App.js: root component

index.css, App.css: component specific CSS-files



```
REACT-DEMO
  > node_modules
  v public
      vite.svg
  v src
    > assets
      App.css
      App.jsx
      index.css
      main.jsx
    .eslintrc.cjs
    .gitignore
    index.html
    package-lock.json
    package.json
    README.md
    vite.config.js
```

sdasd

React basic

```
import React from "react";
import ReactDOM from "react-dom";

const name = "Margit Tennosaar";
const first_page = 1998;

const Demo = () => {
  return (
    <div>
      <h1>Hello, my name is {name}</h1>
      <p>I have {2023 - first_page} years of building webpages
       experience</p>
    </div>
  );
};

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode><Demo /></React.StrictMode>
);
```

SPA in React

Typically only one ReactDOM.render() call

```
function Card(props) {
  return (
    <h1> Hello, {props.name}, {props.text}</h1>
  );
}

function App() {
  return (
    <div>
      <Card name="Sara" text="It is nice to see you!" />
      <Card name="Cahal" text="It is nice to see you!" />
    </div>
  );
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

JSX

Similar to XML or HTML

Embedding expressions – <h1> { JS expression } </h1>

Self-closing tags if no content- <Something />

camelCase – className, onClick, fontSize, htmlFor

https://react.dev/learn/writing-markup-with-jsx#the-rules-of-jsx

Components

Note: Always start component names with a capital letter.

Function (stateless) component

The simplest way to define a component is to write a JavaScript function.

Also referred to as "presentational", "dumb", or "stateless" components

Functions take input and return an output. The inputs are passed with props object, and the output is a component instance in plain JSX.

```
function Welcome(props) {
   return <h1>Hello, {props.name}</h1>;
}
```

```
const Welcome = (props) => {
   return <h1>Hello, {props.name}</h1>
}
```

Class component

You can also use ES6 Class Components extending from the React component. The extend hooks all the lifecycle methods available in the React component API to the component. This gives you the option to use the render() class method. You can also store and manipulate states.

Also referred to as “containers”, “smart”, or "stateful" components

```
class Welcome extends React.Component {
 render() {
   return <h1>Hello,{this.props.name}</h1>;
 }
}
```

https://react.dev/reference/react/Component

Props

Props allow you to pass data

from a parent (wrapping) component

to a child (embedded) component.

```
const Welcome = (props) => {
  return
    <h1>Hello,{props.name}</h1>
}
```

```
class Welcome extends Component {
 render() {
   return
     <h1>Hello, {this.props.name}</h1>;
 }
}
```

```
<Welcome name=“Margit” />
```

https://react.dev/learn/passing-props-to-a-component

Props are read-only and help maintain the unidirectional flow of data in React applications, ensuring a clear and predictable data flow between components.

It is a plain JavaScript object containing data that you pass from a parent component to a child component. Props is creating a custom attribute to the component.

```
const Welcome = (props) => {
  return
    <h1>Hello,{props.name}</h1>
}
```

```
<Welcome name=“Margit” />
```

State

The state object is where you store property values that belong to the component.

The state is used to change the component.

Whenever the state changes, the component will re-render and reflect the new state.

Prior to React 16.8, state was managed in class components. Now, with Hooks like `useState`, functional components can handle state efficiently.

Hooks

Hooks rules

- The naming convention of hooks should start with the prefix use. So, we can have hooks named useState, useEffect, etc.
- Hooks must be called at the top level of a component, before the return statement. They can’t be called inside a conditional statement, loop, or nested functions.
- Hooks must be called from a React function (inside a React component or another hook). It shouldn’t be called from a Vanilla JS function.

useState

The most basic and useful React hook

```
import { useState } from 'react';

function Person() {
  const [person, setPerson] = useState({ id: 1, name: 'Margit' });
  const [age, setAge] = useState(29)

  return (
    <div>
      <h2>Person Details</h2>
      <p>ID: {person.id}</p>
      <p>Name: {person.name}</p>
      <p>Age: {age}</p>
    </div>
  );
}

export default Person;
```

https://react.dev/reference/react/useState

```
import { useState } from 'react';

function Person() {
  const [person, setPerson] = useState({ id: 1, name: 'Margit' });
  const [age, setAge] = useState(29)

  return (
    <div>
      <h2>Person Details</h2>
      <p>ID: {person.id}</p>
      <p>Name: {person.name}</p>
      <p>Age: {age}</p>
    </div>
  );
}

export default Person;
```

ESlint

Code Style

Code style is the rules and guidelines a specific team/person/company uses while developing a project.

You don't necessarily need to have a code style; however, having a code style can make your code less prone to bugs and prevent some bugs ahead of time, all while making your code more homogeneous.

Having homogeneous code in a big team is essential to improve the long-term maintenance of the project.

The most popular JavaScript tool for enforcing/recommending certain code styles is called ESLint.

ESLint is a tool for identifying and reporting patterns found in ECMAScript/JavaScript code to make code more consistent and avoid bugs.

ESLint is written using Node.js to provide a fast runtime environment and easy installation via npm.

Set up Eslint in vanilla JS projects

- Add extension ESlint (once)

- Install ESlint check in your project (once in the project)

```
npm install eslint
```

- Set up ESlint (once in the project)

```
npm init @eslint/config
```

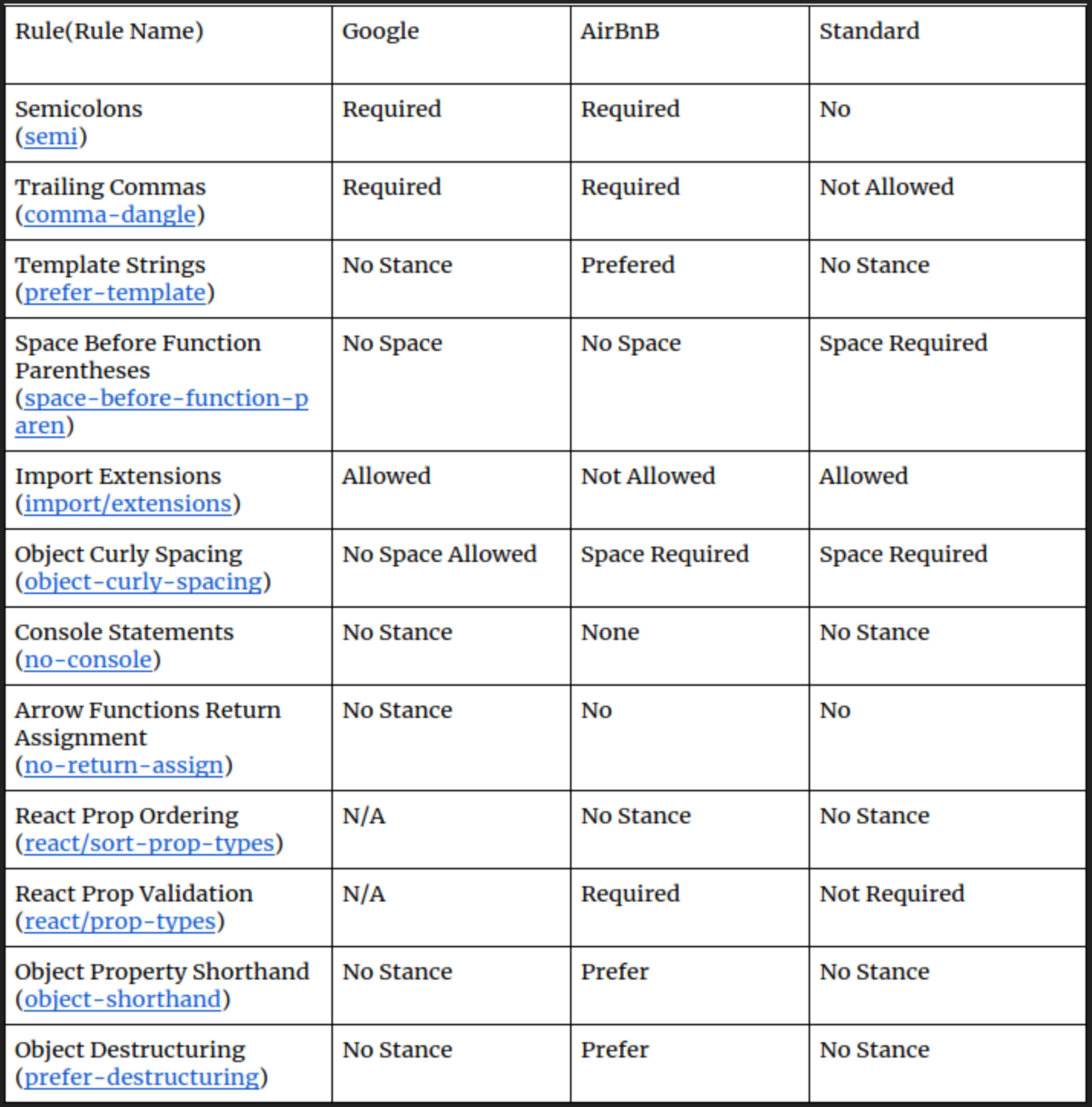
- Configure ESlint if needed

```
// .eslintrc file
{
  "rules": {
       ”semi": ”always"
    }
}
```

Style Guide

Choose a style guide based on your project needs.

For example, Google does not support react



Rule(Rule Name)	Google	AirBnB	Standard
Semicolons (semi)	Required	Required	No
Trailing Commas (comma-dangle)	Required	Required	Not Allowed
Template Strings (prefer-template)	No Stance	Prefered	No Stance
Space Before Function Parentheses (space-before-function-paren)	No Space	No Space	Space Required
Import Extensions (import/extensions)	Allowed	Not Allowed	Allowed
Object Curly Spacing (object-curly-spacing)	No Space Allowed	Space Required	Space Required
Console Statements (no-console)	No Stance	None	No Stance
Arrow Functions Return Assignment (no-return-assign)	No Stance	No	No
React Prop Ordering (react/sort-prop-types)	N/A	No Stance	No Stance
React Prop Validation (react/prop-types)	N/A	Required	Not Required
Object Property Shorthand (object-shorthand)	No Stance	Prefer	No Stance
Object Destructuring (prefer-destructuring)	No Stance	Prefer	No Stance

https://github.com/gajus/eslint-config-canonical#table-of-comparison

ES lint in Vite projects

proptype errors – to ignore it:

```
"rules": {
    'react/prop-types’: 0
 }
```

0=off, 1=warn, 2=error. Defaults to 0

https://react.dev/learn/editor-setup#linting

Lists and keys

By using map() function you can create separated items based arrays you have.

```
const numbers = [1, 2, 3, 4, 5];
const doubled = numbers.map((number) => number * 2);
console.log(doubled);
```

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
  <li>{number}</li>
);
```

```
import { useState } from 'react';

function PersonList() {
  const [persons, setPersons] = useState([
    { id: 1, name: 'Margit' },
    { id: 2, name: ‘Maria' }
  ]);

  return (
    <div>
      <h2>Person List</h2>
      <ul>
        {persons.map(person => (
          <li key={person.id}>{person.name}</li>
        ))}
      </ul>
    </div>
  );
}

export default PersonList;
```

Keys help React identify which items have changed, are added, or are removed. Keys used within arrays should be unique among their siblings. However, they don’t need to be globally unique.

```
{persons.map(person => (
  <li key={person.id}>{person.name}</li>
))}
```

https://react.dev/learn/rendering-lists

Handling Events

event handler

With some syntax differences, handling events in React is very similar to handling DOM elements:

HTML

```
<button onclick=”clickHandler()"> Like </button>
```

React

```
<button onClick={clickHandler}> Like </button>
```

Note: this will execute function on load

```
<button onClick={clickHandler()}> Like </button>
```

```
const clickHandler = () =>
{
  console.log("test");
};
```

https://react.dev/learn/responding-to-events#adding-event-handlers

Modifing states

```
 const [name, setName] = useState(‘Margit');
```

```
function handleClick() {
  setName(‘Maria');
}
```

Calling the set function does not change the current state of the executing code. It only affects what useState will return starting from the next render.

https://react.dev/reference/react/Component#setstate

Practice for easter time

https://github.com/margittennosaar/react_basics/blob/main/exercise_1.md

https://github.com/margittennosaar/react_basics/blob/main/exercise_2.md

https://github.com/margittennosaar/react_basics/blob/main/exercise_3.md

https://github.com/margittennosaar/react_basics/blob/main/exercise_4.md

https://github.com/margittennosaar/react_basics/blob/main/exercise_5.md

https://github.com/margittennosaar/react_basics/blob/main/exercise_6.md