React

Recap

- Frontend is what users see and interact with
- HTML, CSS (and Bootstrap), Javascript

What's next?

- Make websites interactive

How do we do that?

- Plain JavaScript (Assignment 1)
- jQuery (outdated)
- Javascript frameworks (React, Angular, Vue)

Why learn React?

- Build a website or app
- Most popular framework
- Lots of jobs look for React experience

ES6 Features

- 2015 update to Javascript
- Key concepts:
  - const and let
  - arrow functions
  - classes
  - destructuring
  - import and expor t

Variables

- const and let

```
const a = 1;
let b = 1;

a = 2; // Not okay
b = 2; // Okay
```

Arrow functions

```
const sum = (a, b) => a + b;

function sum(a, b) {
  return a + b;
}
```

map, filter, reduce

- Returns new array

```
arr = [1, 2, 3];

arr.map(x => x * 2); // [2, 4, 6]
arr.filter(x => x > 1); // [2, 3]
arr.reduce((sum, x) => sum + x, 0); // 6
```

Exercise

```
['ab', 'bc', 'ad']
```

- Return an array of elements that start with 'a'
- Return all the strings concatenated together
- Return the array ['abcd', bccd', 'adcd']

Classes

```
class Circle {
  constructor(radius) {
    this.radius = radius;
  }
}

const c = new Circle(4);
```

Array Destructuring

```
const [a, b, c] = [1, 2, 3]
console.log(a) // 1

const [a, ...rest]
console.log(rest) // [2, 3]
```

Object Destructuring

```
const o = { a: 1, b: 2, c: 3 };
const { a, b } = o;
console.log(a); // 1

const { a, ...rest } = o;
console.log(rest); // {b: 2, c: 3}
```

Nested Destructuring

```
const user = { id: 1, name: { first: "Jo", last: "Jo" } };

const {
  id,
  name: { first }
} = o;
console.log(first); // Jo
```

Function Parameter

```
const user = { id: 1, name: { first: "Jo", last: "Jo" } };

function user({ id, name: { first } }) {
  return id;
}
```

Object.entries()

```
const o = { a: 1, b: 2 };

Object.entries(o); // [['a', 1], ['b', 2]]
```

Modules

- Named imports and exports
- Several per module

```
// utils.js
export const sum = (a, b) => a + b;
export const sqrt = Math.sqrt;

// App.js
import { sum, sqrt } from './utils"
```

Modules

- Default imports and exports
- One per module

```
// utils.js
const sum = (a, b) => a + b;
export default sum;

// App.js
import importedSum from './utils"
```

What is React?

- Javascript library
- Makes it easy to build user interfaces
- Key features:
  - Declarative (with Virtual DOM)
  - Component-based

Key Concepts

- JSX
- Components
- Props
- State

JSX

- Similar to HTML
- "Transpiled" to JavaScript

```
const element = <h1 className="hello">Hello world!</h1>;

const element = React.createElement(
  "h1",
  { className: "hello" },
  "Hello World!"
);
```

Embedding JavaScript

- Embed any expression inside curly braces

```
const class = 'hello'
const getName = () => 'Jo Jo'
const element = <h1 className={class}>Hello {getName()}</h1>
```

Exercise

https://codesandbox.io/

Display this:

Menu item

Add

Components

- Building blocks of React
- Pieces of UI and logic
- Similar to functions
- Takes inputs (props) and outputs JSX
- Two types of components: function and class

Class Components

```
class Hello extends React.Component {
  constructor(props) {
    super(props);
  }

  render() {
    return (
      <div>
        <h1>Hello World!</h1>
      </div>
    );
  }
}
```

Function Components

- Recommended

```
const Hello = props => {
  return (
    <div>
      <h1>Hello World!</h1>
    </div>
  );
};
```

Rendering Components

- Components are elements too
- We can use them the same way is tags

```
const element = <Hello />;
```

Exercise

Create menu item component and display this:

Menu item

Add

Menu item

Add

Menu item

Add

Props

- A useful component is a reusable one
- Props make components configurable
- Pass props as attributes

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

const element = <Hello name="Jo Jo" >
```

Exercise

Make the component configurable

Spinach

Add

Ham

Add

Mayo

Add

Props

- Props are passed from parent to child
- Props are immutable

State

User interfaces are stateful

- Logged in users see a different screen
- Shopping cart stores user orders

State

- Stores variables that change over time
- Often changes in response to some event
- When those variables change React re-renders the component

Declaring new state variable called "state"

```
const Hello = props => {
  const [state, setState] = useState("Hello!");

  return (
    <div>
      <h1>{state}</h1>
    </div>
  );
};
```

Hello

Updating state

```
const Hello = props => {
  const [state, setState] = useState("Hello!");
  setState("World!");

  return (
    <div>
      <h1>{state}</h1>
    </div>
  );
};
```

World

Core Concept Review

- component
  - building blocks of React
  - combines logic (JS) and presentation (JSX)
- props
  - data passed to a component
  - immutable
- state
  - internal data specific to component
  - data that changes over time

Handling Events

- Similar to HTML
- Don't use document.addEventListener()
- Pass function as event handler

```
// HTML
<button onClick="handleClick()">

// React
<button onClick={handleClick}>
```

Event Handler

- handleClick is passed event

```
const Hello = props => {
  const handleClick = e => {
    console.log("Clicked!");
  };

  return (
    <div>
      <button onClick={handleClick}>Click me</button>
    </div>
  );
};
```

State and Events

- Often changes in response to some event

```
const Hello = props => {
  const [count, setCount] = useState(0);

  const handleClick = e => {
    setCount(count + 1);
  };

  return (
    <div>
      <button onClick={handleClick}>Click me</button>
    </div>
  );
};
```

Exercise

Click me

Counter: 0