🦉

React.2

▼ What does spread operator do?

it creates a new copy of an array with the same values

```
const numbers = [1, 2, 3];
const grades = [...numbers];
console.log(grades); // [1, 2, 3] (shallow copy)
```

▼ What is a shallow copy?

a copy of the items inside an array (but down to 1 level)

▼ How can you use spread syntax to concatenate arrays?

```
const winners = ["Jane", "Bob"];
const losers = ["Ronald", "Kevin"];

const players = [...winners, ...losers];
console.log(players); // ['Jane', 'Bob', 'Ronald', 'Kevin']
```

▼ How do we add an item to an array in an immutable way?

We don't use .push() because it will mutate original array.

We have to make a shallow copy and then insert the new item in a new array.

```
const numbers = [1, 2, 3];
const result = [...numbers, 4];
console.log(result); //[1, 2 ,3 ,4]
```

▼ How to update array items in an immutable way?

you can use the .map method to return a copy of the array while modifying one or more items

```
const grades = [10, 20, 18, 14];
// change 18 to 17
const updatedGrades = grades.map(grade => {
    if (grade === 18){
        return 17;
    }
    // in all other cases, keep it as it was
    return grade;
});
console.log(updatedGrades); //[10, 20, 17, 14]
```

▼ Why is slice() immutable and splice() is not?

splice() method returns the removed item(s) in an array

slice() method returns the selected element(s) in an array, as a new array object

▼ How to remove array items in an immutable way?

you can use the .slice method which returns the selected element(s) in an array, as a new array object

you can also use the .filter method which will return a subset of the original array based on a condition

```
const grades = [10, 8, 9, 4, 16];

// remove the first grade
// think of it as: get all grades except the first one
const subset1 = grades.slice(1); //start from position 1
console.log(subset1); // [8, 9, 4, 16]

// remove the last 2 grades
// think of it as: get all grades except the last 2
// so start from 0 and stop after 5 - 2 = 3 items
const subset2 = grades.slice(0, grades.length - 2);
console.log(subset2); // [10, 8, 9]
```

```
const grades = [10, 8, 9, 4, 16];

// return all grades >= 10
const subset1 = grades.filter(grade => grade >= 10);
console.log(subset1); // [10, 16]

// remove the 2nd grade
const subset2 = grades.filter(grade => grade !== 8);
console.log(subset2); // [10, 9, 4, 16]
```

▼ How can .map can be used **inside** JSX to loop through arrays?

Every time you have a map in JSX, you need to provide a key or else you will get a warning.

React needs to be able to know what item to update in a list without re-rendering the whole list for every update.

The key should be a unique representation of the single item inside the map.

```
import React from "react";

function Grades(){
    const grades = [8, 18, 10, 7, 14];

    return <ul>{
        grades.map((grade, index) => <li key={index}>{grade}</li>)
    }</ul>;
}
```

▼ Why do we need keys in React?

For example, given a list React needs to be able to know which <li></li> to update thus it requires a unique key so that it is able to only update that item without having to remove all the items and render them again.

▼ What does a key allow React to do efficiently?

update the DOM with the least amount of operations

▼ How to add a key/value to object immutably?

```
const data = {
    id: 1,
    name: "Sam"
}

// immutable
const newObj = {...data, age: 18}
console.log(newObj); // {id: 1, name: "Sam", age: 18}
```

▼ How to replace the value of an existing key immutably?

we need to create a new copy of that object with { ...data } and then merge it with the new same key but a different value

It's important to note that when you want to replace, the new values should be after the copy of the old object in order to override old value.

```
const data = {
    id: 1,
    age: 19
}

// immutable
const newObj = {...data, age: 20};
console.log(newObj); // {id: 1, age: 20}
console.log(data); // original object did not change {id: 1, age: 19}
```

▼ How to to create a new copy of an existing object?

{ ...obj }

▼ What does new Date() return?

an instance of the Date object that gives us the current date & time

e.g. "Tue Feb 18 2020 16:34:15 GMT..."

▼ How to immutably remove a key/value pair from an object?

The reason why this works is because const {year, ...rest} = obj is destructuring the value of the key year from obj.

So we end up with rest an immutable copy of obj excluding the year!

```
const obj = {
    id: 1,
    title: "Harry potter",
    year: 2017,
    rating: 4.5
}

// immutable
const {year, ...rest} = obj;
console.log(rest); // { id: 1, title: "Harry potter", rating: 4.5}
```

▼ How to loop through an object in JSX?

The Object.entries() method returns an **array** of a given object's own enumerable string-keyed property **[key, value]** pairs

```
import React from "react";

function App() {
    const settings = {
        title: "Blog",
        theme: "dark"
    }

    return <ul>{
        Object.entries(settings).map(item => {
            return <li key={item[0]}>{item[0]} with value {item[1]}</li>
        })
    }</ul>;
}

/*
<ul>
    <li key="title">title with value Blog</li>
    <li key="theme">theme with value dark</li>
</ul>
*/
```

▼ How to add a default value on a input in JSX?

```
<input type="text" name="address" defaultValue="Amsterdam" />
```

▼ How to add a **read only value** on a input in JSX?

```
<input type="text" name="address" value="Amsterdam" />
```

▼ Event handler for an input JSX?

onChange attribute

event.target refers to the element (in this example the <input />)

because it's an input, you read what's written inside of it by accessing the .value property

```
import React from "react";

function handleAddressChange(event) {
    console.log(event.target.value);
}

<input type="text" name="address" onChange={handleAddressChange} />
```

▼ What is a controlled component?

when you keep track of an input's value as state and update it whenever it changes

▼ How to create a controlled component?

1. We start by creating a **state** variable to store the value
2. This state will have a default value of an empty string `""` or another default value
3. We set the value of the input to that **state** variable
4. We update the state every time it changes

```
import React, {useState} from "react";

function App() {
    const [address, setAddress] = useState("");

    return <input type="text" value={address} onChange={event => setAddress(event.target.value)} />;
}
```

▼ Controlled select component (example)

```
import React, {useState} from "react";

function App() {
    const [country, setCountry] = useState("");

    return <select value={country} onChange={e => setCountry(e.target.value)}>
        <option>Country</option>
        <option value="netherlands">Netherlands</option>
        <option value="belgium">Belgium</option>
        <option value="france">France</option>
    </select>
}
```

▼ Controlled textarea component (example)

```
import React, {useState} from "react";

function App() {
    const [comment, setComment] = useState("");

    return <textarea value={comment} onChange={e => setComment(e.target.value)} />
}
```

▼ React submit a form (example)

event.preventDefault() : called on the event when submitting the form to prevent a browser reload/refresh

```
import React from "react";

function App(){

    function handleFormSubmit(event) {
        event.preventDefault();
    }

    return <form onSubmit={handleFormSubmit}>
        <input type="text" name="name" />
        <input type="submit" value="Add" />
    </form>;
}
```

▼ Why event.preventDefault() when submit form?

it's called on the event when submitting the form to prevent a browser reload/refresh

▼ What is accessibility?

the design and creation of web applications that can be used by everyone it's a practice that promotes inclusion because everyone (people with all abilities) will be able to use your website

▼ What is a11y?

You may often see accessibility shortened as a11y, which means it's a word that starts with a, ends with y, and has 11 characters in between (ccessibilit).

▼ Why is it important to **add a label element to every input** (except buttons), textarea and select in your form?

Accessibility

For: 1) Mouse users & users with motor impairment 2) Visually impaired users use screen-readers

▼ A <label /> needs an htmlFor attribute (React) for what?

to point to the ID of the element

```
<form>
    <label htmlFor="login-email">Email: </label>
    <input type="email" id="login-email" placeholder="alex@email.com" />

    <label htmlFor="login-password">Password: </label>
    <input type="password" id="login-password" placeholder="Password" />

    <input type="submit" />
</form>
```

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼

▼