Prerequisites for React Js - Topics from Js

Flow of topics

- Spread Operator
- Destructuring
- this keyword
- Arrow Functions
- Higher Order Functions (map, filter, reduce)

Spread operator

- The spread operator is used to expand the elements of an array into a list of arguments.
- It is used in function calls and in array literals.

Example

```
var a = [1, 2, 3];
var b = [4, 5, 6];
b = a; // shallow copy
a.push(4);
console.log(b); // [1, 2, 3, 4] since b was passed by reference
// using spread operator
console.log(...a); // 1 2 3 4 -> string form of array
b = [..a];
a.push(5);
console.log(b); // [1, 2, 3, 4] does not get affected as it is a shallow
copy and it got a separate memory in the heap
```

Using spread operator with objects

```
var a = {
  name: "John",
  age: 30,
};
var b = {
  name: "Mary",
  age: 25,
};
b = a; // shallow copy
a.name = "Bob";
console.log(b); // {name: 'John', age: 30} since b was passed by reference
// using spread operator
console.log(...a); // {name: 'Bob', age: 30} -> string form of object
b = { ...a };
a.name = "Alice";
```

```
console.log(b); // {name: 'Bob', age: 30} does not get affected as it is a
shallow copy and it got a separate memory in the heap
```

- The problem with shallow copy using spread is that 1 level only gets copied and the rest of the object is not copied.
- Object of the object is not copied.
- Inorder to copy the object of the object, we need to use the spread operator inside object parameters.

Example for nested object copy

```
var a = {
  name: "John",
  age: 30,
  address: {
    city: "New York",
    state: "NY",
  },
};
var b = { ...a, address: { ...a.address } }; // deep copy
a.address.city = "Boston";
console.log(a); // {name: 'John', age: 30, address: {city: 'Boston',
state: 'NY'}}
console.log(b); // {name: 'John', age: 30, address: {city: 'New York',
state: 'NY'}} deep copy
```

Example for nested array copy

```
var a = [1, 2, 3];
var b = [4, 5, 6];
var c = [...a, ...b]; // this is the spread operator working as
destructuring
console.log(c); // [1, 2, 3, 4, 5, 6]
```

- Much more elegant way is to use JSON.parse(JSON.stringify(obj)) to deep copy the object.

```
var a = {
  name: "John",
  age: 30,
  address: {
    city: "New York",
    state: "NY",
  },
};
var b = JSON.parse(JSON.stringify(a)); // deep copy
```

- Here JSON.stringify() converts the object into a string and JSON.parse() converts the string back to object.
- Here JSON.parse() helps to convert the stringified object into an object, which gets its own memory pool in the heap and hence it is a deep copy.

Destructuring

- Destructuring is a JavaScript feature that allows you to extract data from arrays, or properties from objects, into distinct variables.

Array Destructuring

- In array destructuring, the elements of an array are extracted into distinct variables.

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

- Nested array destructuring

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

Object Destructuring

- In object destructuring, the properties of an object are extracted into distinct variables which are keys of the object.

```
var a = {
  name: "John",
  age: 30,
};
var { name, age } = a;
console.log(name); // John
console.log(age); // 30
```

Nested Object Destructuring

```
var a = {
  name: "John",
  age: 30,
  address: {
    city: "New York",
    state: "NY",
  },
};
var {
  address: { city, state },
} = a;
console.log(city); // New York
console.log(state); // NY
```

this keyword

- The this keyword is used to refer to the current object.
- We need to understand two environment variables:
  - global object
  - local object
- Js can be used in two environments:
  - Browser
  - Node
- In browser, the global object is window object.
- In node, the global object is global object.

Arrow Functions

- Arrow functions are a new syntactic construct in JavaScript.
- Arrow functions are a concise way to write function expressions.

> (Will be discussed in the next section)