# React - The Complete Guide (Incl Hooks, React Router, Redux)

-Academind by Maximilian Schwarzmüller, Maximilian Schwarzmüller

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## SECTION 2: JAVASCRIPT REFRESHER

### 11. Module Introduction (1:35m)

### 12. Understanding "let" and "const" (3:5m)

let: variable values(can re-assign)
const: constant values(cannot re-assign it)

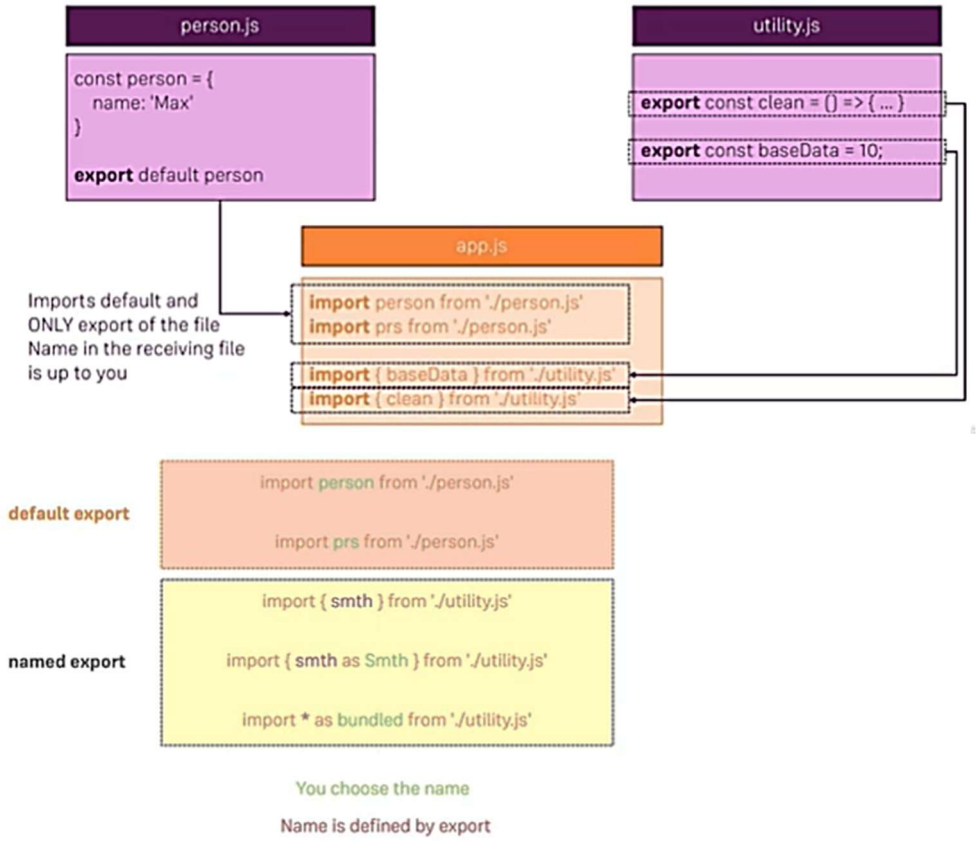
### 13. Arrow Functions (5:27m)

Advantage: keeps scope of *this* keyword, doesn't change its reference.

```
const returnSq = x => x * x;
```

### 14. Exports and Imports (4:43m)

In projects, you split your code across multiple JavaScript files - so-called modules. To access functionality in another file, you need export (to make it available) and import (to get access) statements.




A file can only contain one default and an unlimited amount of named exports.

### 15. Understanding Classes (4:37m)

Classes are a feature which basically replace constructor functions and prototypes. You can define blueprints for JavaScript objects with them.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Format

```
class Person {
  prop1 = 'value';
  method1 = () => {}
}
```

Usage (like constructor)

```
const myPerson = new Person();
myPerson.method1;
console.log(myPerson.prop1)
```

Inheritance (like prototype)

```
class Person extends Master
```

Note: Class Person now has access to all properties & methods of class Master

16. Classes, Properties and Methods (3:3m)

ES6:

```
constructor() {
  this.prop = 'value';
}
method() {}
```

ES7:

```
prop = 'value';
method = () => {}
```

17. The Spread & Rest Operator (6:30m)

Spread(…) used to split up array elements or object properties. With the spread operator you can create a shallow clone of the object or array. Ex:

```
const arr2 = [...arr1, 1, 2];
const obj2 = {...obj1, prop: 'value'};
```

Rest(…) used to merge a list of function arguments into array. Ex:

```
const sortArgs = (...argsArr) => console.log(argsArr);
```

18. Destructuring (3:13m)

Extract array elements or object properties and store them in variables. Ex:

```
const [a, , c] = [1, 2, 3];
const {fname} = {fname: 'first', lname: 'last'};
```

19. Reference and Primitive Types Refresher (4:26m)

**Primitive types:** numbers, strings, booleans… On assignment, it creates copy of value.

**Reference types**: objects, arrays… On assignment, it copies the pointer to the reference in the memory and not the value.

20. Refreshing Array Functions (2:45m) + 23. JavaScript Array Functions

filter(), map(), reduce(), find(), findIndex(), concat(), slice(), splice()

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

21. Wrap Up (52s)

22. Next-Gen JavaScript – Summary + Resources

next-gen-js-summary.pdf