**Hands-on: 9. ReactJS-HOL**

**Introduction**

ES6, also known as ECMAScript 2015, brought a major update to JavaScript by introducing a wide array of powerful features. These features enhanced readability, scalability, and performance in modern JavaScript applications. This document outlines the core features of ES6, with a focus on variable declarations, classes, arrow functions, and new data structures like Set and Map.

1. **Features of ES6**

ES6 introduced several new features, including:

* let and const for block-scoped variables.
* Arrow functions for concise function syntax.
* Classes and inheritance for object-oriented programming.
* Template literals for easier string formatting.
* Default and rest parameters for flexible functions.
* Destructuring assignment for cleaner code.
* Modules for code organization.
* Promises for asynchronous programming.
* New data structures like Map, Set, WeakMap, and WeakSet.

1. **JavaScript let**

let is a variable declaration keyword introduced in ES6. It allows you to declare block-scoped variables, unlike var, which is function-scoped.

let x = 10;

{

let x = 20;

console.log(x); // 20

}

console.log(x); // 10

This helps avoid accidental overwriting of variables and improves code clarity.

1. **Difference Between var and let**

|  |  |  |
| --- | --- | --- |
| Feature | var | let |
| Scope | Function-scoped | Block-scoped |
| Hoisting | Yes (initialized as undefined) | Yes (but not initialized) |
| Redeclaration | Allowed | Not allowed in the same block |
| Temporal Dead Zone | No | Yes |

* Example:

console.log(a); // undefined

var a = 5;

console.log(b); // ReferenceError

let b = 10;

1. **JavaScript const**

const is used to declare constants, meaning the variable's value cannot be reassigned.

const PI = 3.1416;

PI = 3.14; // Error

* Like let, const is block-scoped.
* const variables must be initialized at the time of declaration.
* For objects and arrays, the reference cannot change, but contents can be mutated.

1. **ES6 Class Fundamentals**

ES6 introduced the class syntax to simplify object-oriented programming:

class Person {

constructor(name) {

this.name = name;

}

greet() {

return `Hello, ${this.name}`;

}

}

const p = new Person("Alice");

console.log(p.greet()); // Hello, Alice

* Uses constructor() method for initialization.
* Can define methods directly inside the class.

1. **ES6 Class Inheritance**

Inheritance is achieved using the extends keyword:

class Animal {

speak() {

return "Animal sound";

}

}

class Dog extends Animal {

speak() {

return "Woof!";

}

}

let d = new Dog();

console.log(d.speak()); // Woof!

* super() is used to call the parent class constructor.
* Encourages reusable and hierarchical code structures.

1. **Arrow Functions**

Arrow functions provide a shorter syntax for writing functions:

const add = (a, b) => a + b;

* Key benefits:
* Shorter syntax
* Lexical this binding (does not bind its own this)
* Cannot be used as constructors

function normal() {

console.log(this); // Own 'this'

}

const arrow = () => {

console.log(this); // Inherits 'this' from enclosing scope

};

1. **Set() and Map()**

* Set

A Set is a collection of unique values.

let mySet = new Set([1, 2, 2, 3]);

console.log(mySet); // Set(3) {1, 2, 3}

* No duplicates allowed
* Useful for filtering unique elements
* Map

A Map is a collection of key-value pairs with keys of any type.

let myMap = new Map();

myMap.set('name', 'John');

myMap.set(1, 'one');

console.log(myMap.get(1)); // one

* Maintains the insertion order of entries
* Keys can be of any type (unlike objects, where keys are strings)

**Conclusion**

ES6 marked a pivotal evolution in JavaScript, making it more robust, readable, and efficient. Understanding the use of let, const, arrow functions, and ES6 classes empowers developers to write cleaner and more maintainable code. Additionally, powerful data structures like Set and Map offer better performance and flexibility in managing data collections. Mastering these ES6 features is essential for modern JavaScript development.