The library for web and native user interfaces

Basics of ES6 JavaScript:

* Arrays
* Object
* Arrow functions
* Array methods (Map, Filter, Reduce)
* Destructuring
* Spead operator
* Mutability & Immutability of Objects

1. **Arrays:** An array is a collection of elements. It can hold multiple values under a single name, and the values can be accessed via indices.

Example:

**let fruits = ['Apple', 'Banana', 'Cherry'];**

**console.log(fruits [0]); // Output: Apple**

**console.log(fruits.length); // Output: 3**

1. **Objects:** An object is a collection of key-value pairs. Objects are used to store various keyed collections and more complex entities.

Example:

**let person = {**

**firstName: 'John',**

**lastName: 'Doe',**

**age: 30**

**};**

**console.log(person.firstName); // Output: John**

**console.log(person['age']); // Output: 30**

1. **Arrow Functions:** Arrow functions provide a more concise syntax for writing function expressions.

**// Regular function**

**function add(a, b) {**

**return a + b;**

**}**

**// Arrow function**

**const add = (a, b) => {**

**return a + b;**

**};**

**// Even shorter arrow function with implicit return**

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

1. **Array methods:** Map, Filter, Reduce Methods These array methods are crucial for processing and transforming arrays in a functional programming style.
   1. **map:** creates a new array by applying a function to each element of the original array.

**let numbers = [1, 2, 3, 4];**

**let doubled = numbers.map(n => n \* 2);**

**console.log(doubled); // Output: [2, 4, 6, 8]**

* 1. **filter:** creates a new array with all elements that pass the test implemented by the provided function.

**let numbers = [1, 2, 3, 4];**

**let evens = numbers.filter(n => n % 2 === 0);**

**console.log(evens); // Output: [2, 4]**

* 1. **reduce:** executes a reducer function (that you provide) on each element of the array, resulting in a single output value.

**let numbers = [1, 2, 3, 4];**

**let sum = numbers.reduce((total, n) => total + n, 0);**

**console.log(sum); // Output: 10**

1. **Destructuring:** Destructuring is a convenient way of extracting multiple values from arrays or objects and assigning them to variables.

Array Destructuring

**let [a, b] = [1, 2];**

**console.log(a); // Output: 1**

**console.log(b); // Output: 2**

Object Destructuring

**let person = {**

**fistName: "Ekant",**

**lastName: "Chandrakar"**

**}**

**const  { firstName, lastName } = person;**

Another example:

**function calculate(a, b) {**

**const add = a + b;**

**const subtract = a - b;**

**const multiply = a \* b;**

**const divide = a / b;**

**return [add, subtract, multiply, divide];**

**}**

**const [add, subtract, multiply, divide] = calculate(4, 7);**

1. **Spread Operator:** The spread operator (...) allows an iterable such as an array to be expanded in places where zero or more arguments (for function calls) or elements (for array literals) are expected.

**let arr1 = [1, 2, 3];**

**let arr2 = [...arr1, 4, 5];**

**console.log(arr2); // Output: [1, 2, 3, 4, 5]**

**let obj1 = { a: 1, b: 2 };**

**let obj2 = { ...obj1, c: 3 };**

**console.log(obj2); // Output: { a: 1, b: 2, c: 3 }**

1. **Mutability and Immutability:**

**Mutable objects** can be changed after creation. Arrays and objects in JavaScript are mutable by default.

**let arr = [1, 2, 3];**

**arr.push(4); // modifies the original array**

**console.log(arr); // Output: [1, 2, 3, 4]**

**Immutable objects** cannot be changed after creation. In React, immutability is often favored because it helps in maintaining state consistency and optimizing rendering performance.

**let arr = [1, 2, 3];**

**let newArr = [...arr, 4]; // creates a new array**

**console.log(arr); // Output: [1, 2, 3]**

**console.log(newArr); // Output: [1, 2, 3, 4]**