**Spread Operator:** In Next.js, the spread operator (...) is used in various contexts, similar to its usage in regular JavaScript and React. The spread operator can be used to expand arrays, objects, and function arguments.

**Example:**

  let obj1 = {

    name: "hanzala",

    age: "25",

    favLang: "JavaScript"

  }

  let obj2 = {...obj1, favLang: "Html/Css"}

  console.log(obj2)

**Destructuring:** Destructuring in Next.js, like in regular JavaScript and React, is a convenient way to extract values from arrays or objects and assign them to variables. This feature can be particularly useful when dealing with props, state, and data fetching in Next.js applications.

**Example:**

  let obj1 = {

    name: "hanzala",

    age: "25",

    favLang: "JavaScript"

  }

let {name, age, favLang} = obj1

console.log(name, age, favLang)

**Async Await:** Async/Await simplifies asynchronous code by allowing the use of the await keyword to pause the execution until the Promise is resolved. It’s particularly useful when dealing with multiple asynchronous operations.

**Example:**

const fetchProducts = async () => {

let data = await fetch('https://dummyjson.com/products');

data = await data.json();

setProductList(data.products);

};

**Map:** Map() is a powerful function in JavaScript, commonly used in React JS for efficient array handling. This method efficiently transforms and iterates over array elements, making it a cornerstone for dynamic and streamlined UI development in React applications.

**Example:**

let a = [1,2,3,4]

let a2 = a.map((value)=>{

return value\*2

})

console.log(a2)

**Map Filter:** The filter() method takes each element in an array and it applies a conditional statement against it. If this conditional returns true, the element gets pushed to the output array. If the condition returns false, the element does not get pushed to the output array.

**Example:**

let a = [1,2,3,4, 10, 15, 5, 30]

let a2 = a.filter((value)=>{

return value>9

})

console.log(a2)

**Map Reduce:** The reduce() method reduces an array of values down to just one value. To get the output value, it runs a reducer function on each element of the array.

**Example:**

let a = [1,2,3,4]

let a2 = a.reduce((val1,val2)=>{

return val1+val2

})

console.log(a2)

**Double and Triple Equal:** The double equal operator only checks if two objects are references to the same object in memory, while the triple equal operator checks if two objects have the same properties and values.

**Example:**

if (1=="1"){

console.log("this is true")

}

else{

console.log("this is false");

}

**Output:** this is true

if (1==="1"){

console.log("this is true")

}

else{

console.log("this is false");

}

**Output:** this is false