Default Argument:

In JavaScript, a default argument is a value that is automatically assigned to a function parameter if no value is passed to that parameter when the function is called. This can be useful in cases where a function is expected to be called with certain parameters, but the caller may not always provide them.

Here is an example of a function with a default argument:

```javascript
function greet(name = "John") {
  console.log(`Hello, ${name}!`);
}

greet(); // Output: Hello, John!
greet("Mary"); // Output: Hello, Mary!
```

In the above example, the function greet has a single parameter name, which is assigned a default value of "John". If the function is called without any arguments, it will use the default value. However, if an argument is passed, it will use the value passed by the caller.

You can also set defaults for multiple arguments, and use a logical operator || to assign the default value

```javascript
function sum(a = 0, b = 0) {
  return a + b;
}
console.log(sum(1)) // 1
console.log(sum()) // 0
```

Default arguments are a feature of ES6 (ECMAScript 6), and can be used in all types of functions, including arrow functions, function expressions, and methods.

Array Destructuring:

Array destructuring is a feature of JavaScript that allows you to extract values from an array and assign them to individual variables. It is a concise and expressive way to work with arrays, and can make your code more readable and maintainable.

Here is an example of array destructuring:

```JavaScript
let colors = ["red", "green", "blue"];
let [firstColor, secondColor, thirdColor] = colors;
console.log(firstColor); // Output: "red"
console.log(secondColor); // Output: "green"
console.log(thirdColor); // Output: "blue"
```

In the above example, the values of the array colors are assigned to the variables firstColor, secondColor, and thirdColor in that order.

You can also use the rest operator ... in the destructuring to get all the remaining items in an array.

```JavaScript
let colors = ["red", "green", "blue"];
let [firstColor, ...remainingColors] = colors;
console.log(firstColor); // Output: "red"
console.log(remainingColors); // Output: ["green", "blue"]
```

You can also use destructuring in function arguments, it's useful when you want to pass an array as argument and unpack it's values to different variables

```JavaScript
function printColors([firstColor, secondColor, thirdColor]) {
  console.log(firstColor);
  console.log(secondColor);
  console.log(thirdColor);
}
printColors(colors);
```

Array destructuring is a powerful feature that can make your code more concise and expressive, and is especially useful when working with arrays of data.

Object Destructuring:

Object destructuring is a feature of JavaScript that allows you to extract values from an object and assign them to individual variables. It is a concise and expressive way to work with objects, and can make your code more readable and maintainable.

Here is an example of object destructuring:

```javascript
let person = {
  name: "John",
  age: 30,
  location: "New York"
};
let { name, age, location } = person;
console.log(name); // Output: "John"
console.log(age); // Output: 30
console.log(location); // Output: "New York"
```

In the above example, the properties of the object person are assigned to the variables name, age, and location in that order. You can also use different variable names and use the : operator to assign the value.

```javascript
let { name: personName, age: personAge, location: personLocation
} = person;
console.log(personName); // Output: "John"
console.log(personAge); // Output: 30
console.log(personLocation); // Output: "New York"
```

You can also use destructuring in function arguments; it's useful when you want to pass an object as an argument and unpack its properties to different variables.

JavaScript

```
function printPerson({ name, age, location }) {
  console.log(name);
  console.log(age);
  console.log(location);
}
printPerson(person);
```

You can also use the rest operator ... in destructuring, it will assign the remaining properties of an object to a variable.

JavaScript

```
let { name, age, ...others } = person;
console.log(others); // {location: "New York"}
```