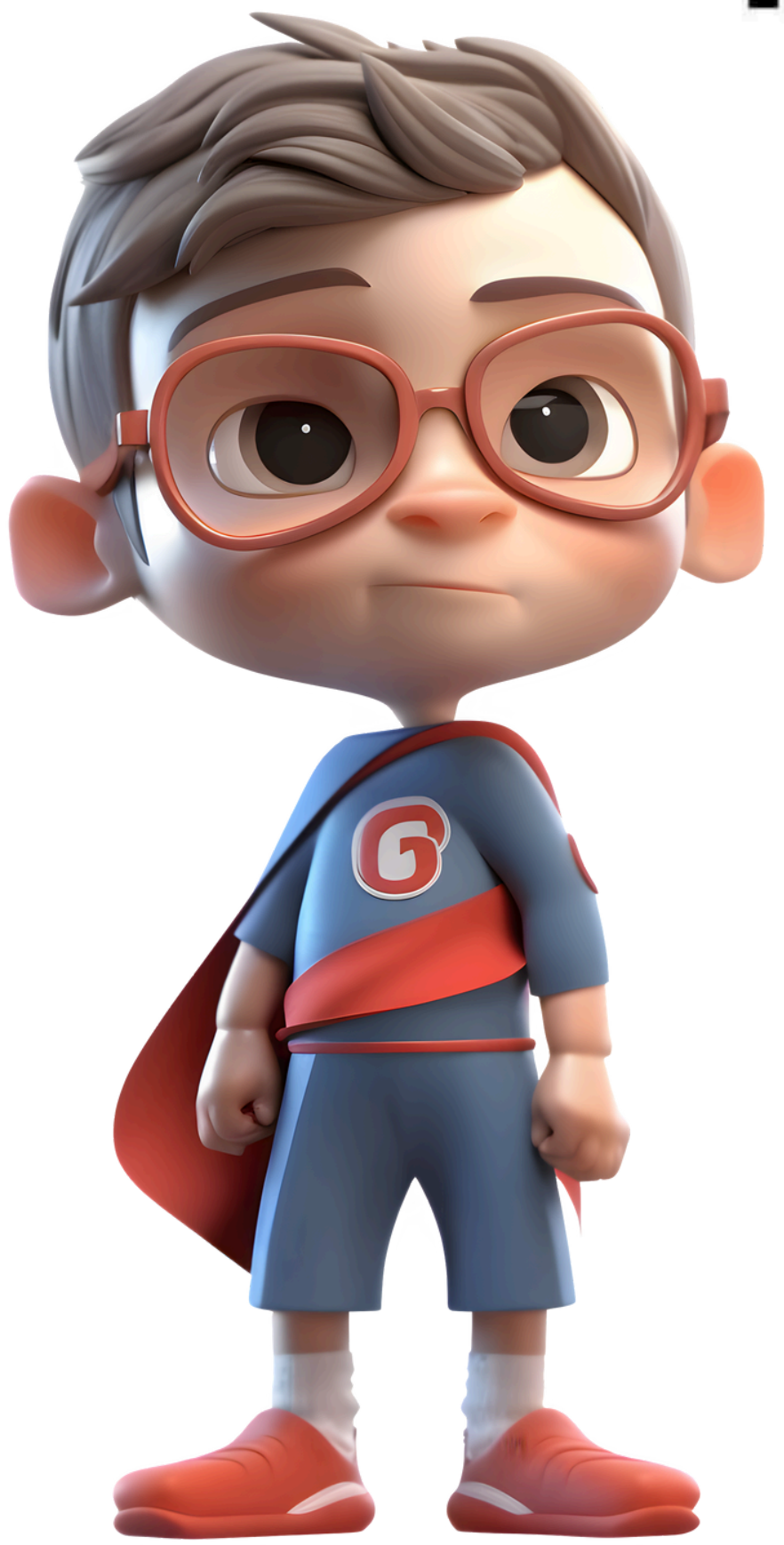
ARRAY

Javascript



In JavaScript, an array is a data structure that allows you to store multiple values in a single variable. Arrays can hold a collection of items of any type, including numbers, strings, objects, and even other arrays.

Modifying Array Elements

You can modify an array element by accessing it through its inde
and assigning a new value.

```
let fruits = ["apple", "banana", "cherry"];
fruits[1] = "blueberry";
console.log(fruits); // Output: ["apple", "blueberry", "cherry"]
```

Array Properties and Methods

Properties:

- **length:** Returns the number of elements in an array.

```
let fruits = ["apple", "banana", "cherry"];
console.log(fruits.length); // Output: 3
```

Methods:

- **push():** Adds one or more elements to the end of an array.

```
fruits.push("date");
console.log(fruits); // Output: ["apple", "banana", "cherry", "date"]
```

Creating Arrays

You can create an array using two primary methods:

Array Literal Notation:

```
let fruits = ["apple", "banana", "cherry"];
```

Array Constructor:

```
let fruits = new Array("apple", "banana", "cherry");
```

Accessing Array Elements

Array elements are accessed using their index, which starts from 0.

```
let fruits = ["apple", "banana", "cherry"];
console.log(fruits[0]); // Output: apple
console.log(fruits[1]); // Output: banana
console.log(fruits[2]); // Output: cherry
```

- **pop():** Removes the last element from an array and returns that element.

```
let lastFruit = fruits.pop();
console.log(lastFruit); // Output: "date"
console.log(fruits); // Output: ["apple", "banana", "cherry"]
```

- **shift():** Removes the first element from an array and returns that element.

```
let firstFruit = fruits.shift();
console.log(firstFruit); // Output: "apple"
console.log(fruits); // Output: ["banana", "cherry"]
```

- **unshift():** Adds one or more elements to the beginning of an array.

```
fruits.unshift("avocado");
console.log(fruits); // Output: ["avocado", "banana", "cherry"]
```

- **splice():** Adds/Removes elements from an array.

```
// Add elements
fruits.splice(1, 0, "blueberry", "blackberry");
console.log(fruits); // Output: ["avocado", "blueberry", "blackberry", "banana", "cherry"]

// Remove elements
fruits.splice(1, 2);
console.log(fruits); // Output: ["avocado", "banana", "cherry"]
```

- **slice():** Returns a new array containing a portion of the original array.

```
let newFruits = fruits.slice(1, 3);
console.log(newFruits); // Output: ["banana", "cherry"]
```

- **concat():** Merges two or more arrays.

```
let moreFruits = ["date", "elderberry"];
let allFruits = fruits.concat(moreFruits);
console.log(allFruits); // Output: ["avocado", "banana", "cherry", "date", "elderberry"]
```

- **join():** Joins all elements of an array into a string.

```
let fruitString = fruits.join(", ");
console.log(fruitString); // Output: "avocado, banana, cherry"
```

- **reverse():** Reverses the order of the elements in an array.

```
fruits.reverse();
console.log(fruits); // Output: ["cherry", "banana", "avocado"]
```

- **sort():** Sorts the elements of an array.

```
fruits.sort();
console.log(fruits); // Output: ["avocado", "banana", "cherry"]
```

- **map():** Creates a new array by applying a function to each element of the original array.

```
let fruitLengths = fruits.map(fruit => fruit.length);
console.log(fruitLengths); // Output: [7, 6, 6]
```

- **filter():** Creates a new array with all elements that pass the test implemented by the provided function.

```
let longFruits = fruits.filter(fruit => fruit.length > 5);
console.log(longFruits); // Output: ["avocado", "banana"]
```

- **forEach():** Executes a provided function once for each array element.

```
fruits.forEach(fruit => console.log(fruit));
// Output:
// avocado
// banana
// cherry
```

- **reduce():** Executes a reducer function on each element of the array, resulting in a single output value.

```
let totalLength = fruits.reduce((sum, fruit) => sum + fruit.length, 0);
console.log(totalLength); // Output: 19
```

Multidimensional Arrays

Arrays can also contain other arrays, creating a multidimensional array.

```
let matrix = [
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
];
console.log(matrix[1][2]); // Output: 6
```

Checking if a Variable is an Array

To check if a variable is an array, you can use Array.isArray().

```
let fruits = ["apple", "banana", "cherry"];
console.log(Array.isArray(fruits)); // Output: true

let notAnArray = "apple";
console.log(Array.isArray(notAnArray)); // Output: false
```

Spread Operator

The spread operator (...) allows you to expand an array into its elements.

```
let fruits = ["apple", "banana", "cherry"];
let moreFruits = [...fruits, "date", "elderberry"];
console.log(moreFruits); // Output: ["apple", "banana", "cherry", "date", "elderberry"]
```

Rest Parameter

In function definitions, the rest parameter syntax (...) allows you to represent an indefinite number of arguments as an array.

```
function sum(...numbers) {
  return numbers.reduce((total, number) => total + number, 0);
}
console.log(sum(1, 2, 3, 4)); // Output: 10
```