JavaScript:

What is the difference between null and undefined in JavaScript?

- **Answer:** undefined means a variable has been declared but not assigned a value, while null is an assignment value that represents no value or no object. undefined is a type itself, while null is an object.

What is the difference between == and === in JavaScript?

- **Answer:** == is the equality operator that compares two values for equality after converting them to a common type (type coercion), while === is the strict equality operator that compares both value and type without type coercion.

```
console.log(5 == '5'); // true
console.log(5 === '5'); // false
console.log(null == undefined); // true
console.log(null === undefined); // false
```

What is event bubbling and event capturing in JavaScript?

- **Answer:** Event bubbling and capturing are two phases of event propagation. In event capturing, the event propagates from the root to the target element (capturing phase). In event bubbling, the event propagates from the target element up to the root (bubbling phase). By default, event handlers are called in the bubbling phase.

```

document.getElementById('parent').addEventListener('click', function() {
   console.log('Parent clicked');
}, true); // Capturing phase

document.getElementById('child').addEventListener('click', function() {
   console.log('Child clicked');
});

// Clicking on the child element will log:
// Parent clicked
// Child clicked
```

Advanced JavaScript:

Explain the concept of closures in JavaScript.

- **Answer:** A closure is a function that has access to its own scope, the scope of the outer function, and the global scope. Closures allow a function to access variables from an enclosing scope even after it has returned.

```
function outerFunction(outerVariable) {
   return function innerFunction(innerVariable) {
      console.log('Outer variable:', outerVariable);
      console.log('Inner variable:', innerVariable);
   };
}
const newFunction = outerFunction('outside');
newFunction('inside');
// Output:
// Outer variable: outside
// Inner variable: inside
```

What is the difference between let, const, and var?

- **Answer:**
  - var is function-scoped or globally scoped and can be redeclared and updated.
  - let is block-scoped and can be updated but not redeclared within the same scope.
  - const is block-scoped and cannot be updated or redeclared; it must be initialized at the time of declaration.

What are the differences between call(), apply(), and bind()?

- **Answer:**
  - call() invokes a function with a given this context and arguments provided individually.
  - apply() is similar to call(), but arguments are provided as an array.
  - bind() returns a new function with a given this context and, optionally, initial arguments.

```
function greet(greeting, punctuation) {
   console.log(`${greeting}, ${this.name}${punctuation}`);
}

const person = { name: 'Alice' };

```

```
greet.call(person, 'Hello', '!'); // Hello, Alice!
greet.apply(person, ['Hi', '?']); // Hi, Alice?
const boundGreet = greet.bind(person, 'Hey');
boundGreet('!!'); // Hey, Alice!!
```

What are arrow functions and how do they differ from regular functions?

- **Answer:** Arrow functions provide a concise syntax for writing functions and have lexical `this` binding, meaning they inherit `this` from the parent scope. They cannot be used as constructors, and they do not have their own `arguments` object.

Explain the difference between synchronous and asynchronous code.

- **Answer:** Synchronous code is executed sequentially, blocking subsequent code until the current task is completed. Asynchronous code allows other tasks to run while waiting for the current task to complete, preventing blocking and improving performance, especially in I/O operations.

What is the difference between deep copy and shallow copy in JavaScript?

- **Answer:** A shallow copy copies only the references of nested objects, not the objects themselves, while a deep copy duplicates everything, including all nested objects.

```
const obj = { a: 1, b: { c: 2 } };
const shallowCopy = Object.assign({}, obj);
const deepCopy = JSON.parse(JSON.stringify(obj));

obj.b.c = 3;
console.log(shallowCopy.b.c); // 3
console.log(deepCopy.b.c); // 2
```

Task 1: Employee Salary Analysis

Objective: You are given an array of employee objects, each containing information about the employee's name, department, and salary. Your task is to:

1. Increase the salary of each employee by 10%.
2. Filter out employees whose salary (after the increase) is below $50,000.
3. Calculate the total salary of the remaining employees.

```
const employees = [
  { name: 'Ravi', department: 'Engineering', salary: 45000 },
  { name: 'Suresh', department: 'Marketing', salary: 55000 },
  { name: 'Rajesh', department: 'Engineering', salary: 60000 },
  { name: 'Anil', department: 'HR', salary: 40000 },
  { name: 'Praveen', department: 'Marketing', salary: 48000 },
];
```

Step 1: Map - Increase Salary by 10%

```
const increasedSalaries = employees.map(employee => ({
  ...employee,
  salary: employee.salary * 1.1,
}));

console.log(increasedSalaries);
```

output:

```

[
  { name: 'Ravi', department: 'Engineering', salary: 49500 },
  { name: 'Suresh', department: 'Marketing', salary: 60500 },
  { name: 'Rajesh', department: 'Engineering', salary: 66000 },
  { name: 'Anil', department: 'HR', salary: 44000 },
  { name: 'Praveen', department: 'Marketing', salary: 52800 },
]
```

Step 2: Filter - Remove Employees with Salary Below $50,000

```
const filteredEmployees = increasedSalaries.filter(employee => employee.salary >= 50000);

console.log(filteredEmployees);
```

Output:

```
[
   { name: 'Suresh', department: 'Marketing', salary: 60500 },
   { name: 'Rajesh', department: 'Engineering', salary: 66000 },
   { name: 'Praveen', department: 'Marketing', salary: 52800 },
]
```

Step 3: Reduce - Calculate Total Salary of Remaining Employees

```
const totalSalary = filteredEmployees.reduce((sum, employee) => sum + employee.salary, 0);

console.log('Total salary of remaining employees:', totalSalary);
```

Output:

```
Total salary of remaining employees: 179300
```