Back to Previous Page

Quiz Result For "Advanced Javascript Quiz" Is Here -

Total Question - 24

Total Attempts - 24

Total Correct - 23

Total Incorrect - 1

Duration - 24Minutes

Passing Percentage - 70%

Your Percentage - 95.83%

Q1. Which of the following is a self-invoking function in JavaScript?

1)

```
function myFunction() {}
```

2)

```
(function() {})();
```

3)

```
const myFunction = function() {};
```

4)

```
function() {}
```

Your Answer: 2) ✔

```
(function() {})();
```

Correct Answer: 2)

```
(function() {})();
```

Q2. How does the <code>forEach</code> function differ from the map function in JavaScript?

1) forEach returns a new array, while map mutates the existing array

2) map is used for iterating over objects, while forEach is used for arrays

3) forEach mutates the existing array, while map returns a new array

4) map is a modern alternative to forEach

Your Answer: 3) ✔

forEach mutates the existing array, while map returns a new array

Correct Answer: 3)

forEach mutates the existing array, while map returns a new array

Q3. What is another name for a self-invoking function?

1) Auto-invoking function

2) Immediately invoked function expression (IIFE)

3) Recursive function

4) Anonymous function

Your Answer: 2) ✔

Immediately invoked function expression (IIFE)

Correct Answer: 2)

Immediately invoked function expression (IIFE)

Q4. Consider the arrow function below. What will be the output when called with multiply(5)?

1) 10

2) 5

3) 15

4) undefined

Your Answer: 1) ✔

10

Correct Answer: 1)

10

Q5. Analyze the code snippet below. What will be the value of <code>counter</code> after executing increment() three times?

1) 3

2) 1

3) 0

4) undefined

Your Answer: 1) ✔

3

Correct Answer: 1)

3

Q6. What is the key difference between function declarations and function expressions in terms of hoisting?

1) Both are hoisted

2) Function expressions are hoisted, but declarations are not

3) Function declarations are hoisted, but expressions are not

4) Neither is hoisted

Your Answer: 3) ✔

Function declarations are hoisted, but expressions are not

Correct Answer: 3)

Function declarations are hoisted, but expressions are not

Q7. (Function Declaration) What will be the output of the following code?

1) "Hello, I am hoisted!"

2) ReferenceError

3) undefined

4) SyntaxError

Your Answer: 1) ✔

"Hello, I am hoisted!"

Correct Answer: 1)

"Hello, I am hoisted!"

Q8. (Function Expression) Analyze the code snippet. What is the value of <code>result</code>?

1) 8

2) 15

3) "5 + 3"

4) undefined

Your Answer: 1) ✔

8

Correct Answer: 1)

8

Q9. (Pass by value and Pass by reference) Analyze the code snippet. What will be the value of <code>arr</code> after the function call?

1) [1, 2, 3]

2) [1, 2, 3, 4]

3) [4]

4) Error

Your Answer: 2) ✔

[1, 2, 3, 4]

Correct Answer: 2)

[1, 2, 3, 4]

Q10. Convert the following function expression into an arrow function:

1)

```
const greet = name => "Hello, " + name + "!";
```

2)

```
const greet = (name) => { return "Hello, " + name + "!"; };
```

3)

```
const greet = name => { "Hello, " + name + "!"; };
```

4)

```
const greet = name => "Hello, " + name;
```

Your Answer: 2) ❌

```
const greet = (name) => { return "Hello, " + name + "!"; };
```

Correct Answer: 1)

```
const greet = name => "Hello, " + name + "!";
```

Q11. In JavaScript, where are primitive data types stored?

1) Heap

2) Stack

3) Both Heap and Stack

4) d) Neither Heap nor Stack

Your Answer: 2) ✔

Stack

Correct Answer: 2)

Stack

Q12. What is the purpose of the localStorage object in the context of web development?

1) To store data permanently on the server

2) To store data temporarily on the client side

3) To define local variables within a function

4) To manage the heap memory in JavaScript

Your Answer: 2) ✔

To store data temporarily on the client side

Correct Answer: 2)

To store data temporarily on the client side

Q13. Convert the following function into a curried function:

1)

```
const curriedMultiply = x => y => z => x * y * z;
```

2)

```
const curriedMultiply = (x, y, z) => x * y * z;
```

3)

```
const curriedMultiply = (x, y) => z => x * y * z;
```

4)

```
const curriedMultiply = (x, y, z) => { return x * y * z; };
```

Your Answer: 1) ✔

```
const curriedMultiply = x => y => z => x * y * z;
```

Correct Answer: 1)

```
const curriedMultiply = x => y => z => x * y * z;
```

Q14. What will be the output of the following code?

1)

```
6
```

2)

```
"123"
```

3)

```
[1, 2, 3]
```

4)

```
undefined
```

Your Answer: 1) ✔

```
6
```

Correct Answer: 1)

```
6
```

Q15. Using the map function, create code that doubles each element in the <code>array [1, 2, 3]</code>. What is the resulting array?

1)

```
const result = array.map(item => item * 2);
```

2)

```
const result = array.map(item => item + item);
```

3)

```
const result = array.map(item => item * item);
```

4)

```
const result = array.map(item => item / 2);
```

Your Answer: 1) ✔

```
const result = array.map(item => item * 2);
```

Correct Answer: 1)

```
const result = array.map(item => item * 2);
```

Q16. What will be the output of the following code?

1)

```
"AliceBob"
```

2)

```
"Bob"
```

3)

```
"AliceCharlie"
```

4)

```
"Aliceundefined"
```

Your Answer: 3) ✔

```
"AliceCharlie"
```

Correct Answer: 3)

```
"AliceCharlie"
```

Q17. What is the value of <code>matrix[2][1]</code> in the following nested array?

1) 2

2) 8

3) 6

4)

Your Answer: 2) ✔

8

Correct Answer: 2)

8

Q18. What does the <code>Object.keys()</code> method return?

1) The values of an object

2) The keys of an object

3) The prototype of an object

4) The length of an object

Your Answer: 2) ✔

The keys of an object

Correct Answer: 2)

The keys of an object

Q19. Analyze the result of the code below:

1) 5

2) 10

3) 15

4) undefined

Your Answer: 3) ✔

15

Correct Answer: 3)

15

Q20. Extract the name and age properties from the <code>person</code> object using object destructuring.

1)

```
const { name, age } = person;
```

2)

```
const [name, age] = person;
```

3)

```
const { name, age } = { person };
```

4)

```
const [name, age] = { person };
```

Your Answer: 1) ✔

```
const { name, age } = person;
```

Correct Answer: 1)

```
const { name, age } = person;
```

Q21. What is the purpose of the <code>rest</code> parameter in a function parameter list?

1) To spread an array into individual elements

2) To collect multiple function arguments into an array

3) To concatenate two arrays

4) To destructure an object

Your Answer: 2) ✔

To collect multiple function arguments into an array

Correct Answer: 2)

To collect multiple function arguments into an array

Q22. Perform object destructuring on the <code>person</code> object and alias the age property as years.

1)

```
const { name, age: years } = person;
```

2)

```
const { name, years: age } = person;
```

3)

```
const { name, age } = person.years;
```

4)

```
const { name, years } = person.age;
```

Your Answer: 1) ✔

```
const { name, age: years } = person;
```

Correct Answer: 1)

```
const { name, age: years } = person;
```

Q23. Use the spread operator to create a new object <code>newPerson</code> with the same properties as person, but change the name to <code>"Bob"</code>.

1)

```
const newPerson = { ...person, name: "Bob" };
```

2)

```
const newPerson = { person, name: "Bob" };
```

3)

```
const newPerson = { ...person, person: "Bob" };
```

4)

```
const newPerson = { person, ...name: "Bob" };
```

Your Answer: 1) ✔

```
const newPerson = { ...person, name: "Bob" };
```

Correct Answer: 1)

```
const newPerson = { ...person, name: "Bob" };
```

Q24. What is the right way to create a class <code>SUV</code> that extends the <code>Car</code> class and has a new property <code>offRoadCapability</code>.

1)

```
class SUV extends Car {
  constructor(make, model, offRoadCapability) {
    super(make, model);
    this.offRoadCapability = offRoadCapability;
  }
}
```

2)

```
class SUV {
  constructor(make, model, offRoadCapability) {
    this.make = make;
    this.model = model;
    this.offRoadCapability = offRoadCapability;
  }
}
```

3)

```
class SUV extends Car {
  constructor(offRoadCapability) {
    super();
    this.offRoadCapability = offRoadCapability;
  }
}
```

4)

```
None of the above
```

Your Answer: 1) ✔

```
class SUV extends Car {
  constructor(make, model, offRoadCapability) {
    super(make, model);
    this.offRoadCapability = offRoadCapability;
  }
}
```

Correct Answer: 1)

```
class SUV extends Car {
  constructor(make, model, offRoadCapability) {
    super(make, model);
    this.offRoadCapability = offRoadCapability;
  }
}
```