1. What is an anonymous function in JavaScript?
A. A function declared without a name.
B. A function that cannot return a value.
C. A function that is automatically bound to an object.
D. A function that must be declared globally.
Ans:- A

---

2. Which of the following correctly defines an anonymous function and assigns it to a variable?

A.

```
var add = function(a, b) {
  return a + b;
};
```

B.

```
function add(a, b) {
  return a + b;
}
```

C.

```
var add = add(a, b) {
  return a + b;
};
```

D.

```
var add = function add(a, b) {
  return a + b;
};
```

Note: Although option D uses a name in the function expression, it is still considered an anonymous function for external purposes (the internal name is not accessible in the outer scope).
Ans :- A

---

3. What is a potential drawback of using anonymous functions compared to named functions?
A. They are slower to execute.
B. They make debugging more difficult because stack traces may lack meaningful names.
C. They cannot accept parameters.
D. They are hoisted more than named functions.
Ans :- B

---

4. When an anonymous function is assigned to a variable using the `var` keyword, which part is hoisted?
A. The entire function definition.
B. Only the variable declaration (the assignment remains in place).
C. Both the variable declaration and the function body.
D. Neither the variable declaration nor the function body.
Ans :- A

---

5. Which of the following is the correct syntax to define an anonymous function that multiplies two numbers?

A.

```
var multiply = function(a, b) {
  return a * b;
};
```

B.

```
var multiply = function multiply(a, b) {
  return a * b;
};
```

C.

```
function(a, b) {
  return a * b;
}
```

D.

```
function multiply(a, b) {
  return a * b;
}
```

Ans :- A

---

6. Can anonymous functions be recursive?

A. Yes, by referring to themselves via a variable or using `arguments.callee` (in non-strict mode).
B. No, anonymous functions cannot call themselves.
C. Only if they are assigned as object properties.
D. Yes, but only when used in a callback context.

Ans :- A

---

7. Which of the following statements is true regarding naming anonymous functions?

A. They have no name in the outer scope but can be assigned to variables for reference.
B. They always require a name for recursion.
C. They are automatically given a name based on the variable they are assigned to.
D. They cannot be stored in object properties.

Ans :- A

---

8. How does using an anonymous function affect code readability and debugging?

A. It always improves readability because the code is shorter.
B. It may reduce readability and make debugging harder since stack traces show "anonymous."
C. It has no effect on readability or debugging.
D. It makes the code self-documenting by default.

Ans :- A

---

9. In which scenario might you choose to use an anonymous function?

A. When you need a one-time helper function that won't be reused elsewhere.
B. When the function needs to be called recursively from multiple places.
C. When the function must be hoisted for use in the entire file.
D. When you need to define a method that will be referenced in multiple stack traces.

Ans :- A

10. What is the primary difference between a function declaration and a function expression using an anonymous function?

A. Function declarations are not hoisted, whereas function expressions are fully hoisted.

B. Function declarations are hoisted entirely; in function expressions, only the variable is hoisted, not the function definition.

C. There is no difference; both are hoisted equally.

D. Function expressions always result in syntax errors if not named.

Ans :- B

---

11. How can you assign an anonymous function to an object’s property?

A.

```
var obj = {
  greet: function() {
    return "Hello";
  }
};
```

B.

```
var obj = {};
obj.greet = function() {
  return "Hello";
};
```

C. Both A and B

D. Neither A nor B

Ans :- C

---

12. What does the following code output?

```
var person = {
  sayHi: function() {
    return "Hi there!";
  }
};
console.log(person.sayHi());
```

A. `undefined`

B. `"sayHi"`

C. `"Hi there!"`

D. A runtime error

Ans :- C

---

13. Which statement best describes how anonymous functions can create closures?

A. Anonymous functions cannot create closures.

B. They capture variables from their enclosing scope just like named functions.

C. They capture only global variables, not local ones.

D. They require an explicit syntax to capture variables from outer scopes.

Ans :- C

---

14. How do anonymous functions help in avoiding global namespace pollution?

A. They cannot be stored in global variables.

B. When used properly (e.g., as function expressions), they limit the exposure of function names to the outer scope.

C. They force all variables declared inside them to be global.

D. They automatically bind to the window object.

Ans :- B

---

15. Which of the following is a limitation when using anonymous functions in JavaScript?

A. They cannot be assigned to variables.

B. They do not appear with a name in debugging stack traces, making error tracing harder.

C. They cannot accept parameters.

D. They always run before named functions.

Ans :- B

---

16. How do you access the arguments passed to an anonymous function?

A. Through the `params` object.

B. Through the function’s name.

C. Using the `arguments` object provided by JavaScript.

D. They are automatically assigned to global variables.

Ans :- A

---

17. Which scenario is generally not ideal for using an anonymous function?

A. When defining a short, one-time utility.

B. When creating a method that is reused throughout an application, where a name would aid debugging.

C. When encapsulating variables to create a private scope.

D. When the function is intended to be self-contained.

Ans :- B

---

18. When assigning an anonymous function to a variable declared with `var`, which statement is true regarding hoisting?

A. The function definition is hoisted and can be used before its declaration.

B. Only the variable declaration is hoisted; the function assignment is not available until execution reaches that line.

C. Neither the variable declaration nor the function assignment is hoisted.

D. Both the variable declaration and the function assignment are hoisted.

Ans :- B

---

19. Is it possible to define an anonymous function as a method within an object literal?

A. Yes, by assigning the anonymous function as the value for the method’s key.

B. No, methods within an object must always be named.

C. Only if the object is declared using the `new` keyword.

D. Only when the function is defined externally.

Ans :- A

---

20. Which statement is true regarding the usage of anonymous functions in JavaScript?

A. They can be used anywhere a function is expected and help keep the global scope uncluttered.

B. They are only useful for one-time events.

C. They always lead to slower execution times than named functions.

D. They are automatically hoisted with full definitions.

Ans :- A
