
JavaScript Assignment

---

1. let keyword

Explanation:
let is used to declare variables in JavaScript. It is **block-scoped**, meaning it works only inside { }. The value can be changed later.

Example:

let a = 10;

a = 20;

---

2. const keyword

Explanation:
const declares constants. Once assigned, the value **cannot be changed**. It is also block-scoped.

Example:

const PI = 3.14;

---

3. Primitive Data Types

Explanation:
Primitive data types store **single values**. These include number, string, boolean, null, and undefined.

Example:

let age = 25;

let name = "JavaScript";

---

4. Reference Data Types

Explanation:
Reference types store **memory references**. Objects, arrays, and functions are reference types.

Example:

const user = { name: "Rahul" };

---

5. Truthy Values

Explanation:
Values that are treated as true in conditions are called truthy values.

Example:

```
if ("hello") {
  console.log("Truthy");
}
```

---

6. Falsy Values

Explanation:
Falsy values are false, 0, "", null, undefined, and NaN.

Example:

```
if (0) {
} else {
  console.log("Falsy value");
}
```

---

7. Equality Operators (== vs ===)

Explanation:
== checks only value, while === checks both **value and type**.

Example:

```
5 == "5";   // true
5 === "5"; // false
```

---

8. Logical Operators

Explanation:
Logical operators && (AND) and || (OR) combine multiple conditions.

Example:

```
if (age > 18 && age < 60) {
  console.log("Eligible");
}
```

---

9. Ternary Operator

Explanation:
The ternary operator is a **short form of if-else**.

Example:

```
let result = age >= 18 ? "Adult" : "Minor";
```

---

10. if Statement

Explanation:
The if statement executes code when the condition is true.

Example:

```
if (marks > 40) {
  console.log("Pass");
}
```

---

11. Function Declaration

Explanation:
A function declaration defines a named function. It is **hoisted**.

Example:

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

---

12. Function Expression

Explanation:
A function expression stores a function in a variable. It is **not hoisted**.

Example:

```
const sub = function(a, b) {
  return a - b;
};
```

---

13. Arrow Function

Explanation:
Arrow functions provide **short syntax** and do not have their own this.

Example:

```
const square = x => x * x;
```

---

14. Default Parameters

Explanation:
Default parameters give a value if an argument is not passed.

Example:

```
function greet(name = "Guest") {
  return name;
}
```

---

15. Return Statement

Explanation:
return sends a value back from a function.

Example:

```
return total;
```

---

16. Object Creation

Explanation:
Objects store data in **key-value pairs**.

Example:

```
const student = { id: 1, name: "Amit" };
```

---

17. Object Access

Explanation:
Object properties can be accessed using dot or bracket notation.

Example:

```
student.name;
student["id"];
```

---

18. Array Basics

Explanation:
Arrays store ordered data and use **index numbers**.

Example:

```
const numbers = [10, 20, 30];
```

---

19. map()

Explanation:
map() creates a new array by modifying each element.

Example:

```
numbers.map(n => n * 2);
```

---

20. filter()

Explanation:
filter() returns elements that match a condition.

Example:

```
numbers.filter(n => n > 15);
```

---

21. find()

Explanation:
find() returns the **first matching element**.

Example:

```
numbers.find(n => n === 20);
```

---

22. reduce()

Explanation:
reduce() converts an array into a single value.

Example:

```
numbers.reduce((sum, n) => sum + n, 0);
```

---

23. some()

Explanation:
some() checks if **any** element satisfies the condition.

Example:

numbers.some(n => n > 25);

---

24. every()

Explanation:
every() checks if **all** elements satisfy the condition.

Example:

numbers.every(n => n > 5);

---

25. Destructuring

Explanation:
Destructuring extracts values from arrays or objects.

Example:

const { name } = student;

---

26. Spread Operator (...)

Explanation:
Spread operator copies or merges data.

Example:

const newArr = [...numbers, 40];

---

27. Rest Operator (...)

Explanation:
Rest operator collects multiple values into an array.

Example:

function sum(...nums) {

  return nums.length;

}

---

28. Template Literals

Explanation:
Template literals allow **string interpolation**.

Example:

`Hello ${name}`

---

29. import / export

Explanation:
Used to share code between files.

Example:

export const x = 10;

import { x } from "./file.js";

---

30. ES6 Classes

Explanation:
Classes are blueprints for creating objects.

Example:

class Person {}

---

31. Constructor

Explanation:
Constructor initializes object values.

Example:

constructor(name) {

  this.name = name;

}

---

32. this keyword

Explanation:
this refers to the current object.

Example:

this.name;

---

33. Promises

Explanation:
Promises handle asynchronous operations.

Example:

```
fetch(url).then().catch();
```

---

34. DOM Basics

Explanation:
DOM allows JavaScript to manipulate HTML elements.

Example:

```
document.getElementById("title");
```
