```
console.clear();

/*
=====================================================================

                Data Types Interview Questions 🔥

=====================================================================
*/

console.log("Data Types Interview Questions");

/*

1. What is the difference between null and undefined in JavaScript?

👉 `null` is an assigned value that represents the intentional absence of any object value.

👉 `undefined` means a variable is declared but not assigned any value.

*/

// Example:

let a = null;

let b;

console.log("a:", a, "Type:", typeof a); // object (JS bug)

console.log("b:", b, "Type:", typeof b); // undefined

/*

2. What is the purpose of the `typeof` operator in JavaScript?

👉 The `typeof` operator is used to find the data type of a variable.
```

```

*/

// Example:

console.log(typeof "JavaScript"); // string

console.log(typeof 42); // number

console.log(typeof false); // boolean

/*

3. What are the primitive data types in JavaScript?

👉 JavaScript has 7 primitive data types:

   ✅ Number

   ✅ String

   ✅ Boolean

   ✅ Undefined

   ✅ Null

   ✅ BigInt

   ✅ Symbol

*/

// Example:

let myFavoriteNumber = -7; // number

let myFavoriteString = "JavaScript"; // string
let myFavoriteBoolean = true; // boolean

let myFavoriteUndefined; // undefined

let myFavoriteNull = null; // null

let myFavoriteBigInt = 1234567890123456789012345678901234567890n; // bigint

let myFavoriteSymbol = Symbol("id"); // symbol

/*

4. What are the non-primitive data types in JavaScript?

👉 The non-primitive data types are Objects, Arrays, Functions, Maps, Sets, WeakMap, WeakSet etc.

*/

/*

5. How do you check if a variable is an array in JavaScript?

👉 Use `Array.isArray(variable)` method.

*/

console.log(Array.isArray([1, 2, 3])); // true

console.log(Array.isArray({ name: "John" })); // false

/*

6. What is the difference between implicit and explicit type coercion?

👉 Implicit Coercion happens automatically by JavaScript.

👉 Explicit Coercion happens when we manually convert a value.
```

```javascript

*/

// Implicit

console.log(10 + "5"); // "105" (number converted to string)

// Explicit

console.log(Number("5") + 10); // 15 (string converted to number)

/*

7. Convert a number to a string.

👉 Use `toString()` or concatenate with `""`.

*/

let num = 123;

console.log(num.toString(), typeof num.toString()); // "123" string

console.log("" + num, typeof "" + num); // "123" string

/*

8. Convert a string to a number.

👉 Use `Number()` or `+` operator.

*/

console.log(Number("50"), typeof Number("50")); // 50 number

console.log(+"50", typeof +"50"); // 50 number

/*
```

```

9. What is the difference between parseInt() and parseFloat()?

👉 `parseInt()` converts a string to an integer.

👉 `parseFloat()` converts a string to a decimal.

*/

console.log(parseInt("99.99")); // 99

console.log(parseFloat("99.99")); // 99.99

/*

10. What are truthy and falsy values?

👉 Truthy values: non-empty strings, numbers (except 0), objects, arrays.

👉 Falsy values: `false`, `0`, `""`, `null`, `undefined`, `NaN`.

*/

console.log(Boolean("hello")); // true

console.log(Boolean("")); // false

/*

11. How to check if a variable is `NaN`?

👉 Use `Number.isNaN(value)`

*/

console.log(Number.isNaN(NaN)); // true

console.log(Number.isNaN("hello")); // false
```

```js

/*

12. What is the output of `typeof NaN`?

*/

console.log(typeof NaN); // number ❗(weird JS behavior)

/*

13. How to check if a number is finite?

👉 Use `Number.isFinite(value)`

*/

console.log(Number.isFinite(100)); // true

console.log(Number.isFinite(Infinity)); // false

/*

14. What is Symbol in JavaScript?

👉 `Symbol` is a unique primitive value used for object properties.

*/

const sym1 = Symbol("id");

const sym2 = Symbol("id");

console.log(sym1 === sym2); // false (each symbol is unique)

/*

15. What is the difference between shallow copy and deep copy?
```

```
👉 Shallow copy only copies references, while deep copy clones all values.

*/

const obj1 = { name: "John" };

const obj2 = obj1;

console.log(obj1 === obj2); // true (shallow copy)

const obj3 = { ...obj1 };

console.log(obj1 === obj3); // false (deep copy)

/*

=====================================================================

                Output-Based JavaScript Interview Questions 🔥

=====================================================================
*/

console.log("Output-Based Interview Questions");

// 1️⃣ String + Number

console.log("10" + 20); // "1020"

// 2️⃣ Number - String

console.log(10 - "5"); // 5 (string converted to number)

// 3️⃣ Boolean + Number

console.log(true + 1); // 2 (true is 1)

```

```
// 4️⃣ Boolean - Boolean

console.log(false - true); // -1 (false = 0, true = 1)

// 5️⃣ Null + Number

console.log(null + 10); // 10 (null treated as 0)

// 6️⃣ Undefined + Number

console.log(undefined + 10); // NaN (undefined cannot be converted)

// 7️⃣ Empty String + Number

console.log("" + 10); // "10" (string concatenation)

// 8️⃣ Empty String - Number

console.log("" - 10); // -10 ("" is treated as 0)

// 9️⃣ Comparing null and 0

console.log(null == 0); // false

console.log(null >= 0); // true ❗ (unexpected behavior)

// 🔟 Comparing undefined and null

console.log(undefined == null); // true

console.log(undefined === null); // false

// 1️⃣1️⃣ Comparing Boolean values

console.log(true == "1"); // true (string "1" is converted to number)

// 1️⃣2️⃣ Logical OR (||) behavior

```

```
console.log(null || 5); // 5 (null is falsy)

console.log(undefined || 10); // 10 (undefined is falsy)

// 1️⃣3️⃣ Logical AND (&&) behavior

console.log(1 && "Hello"); // "Hello" (1 is truthy, so returns second value)

console.log(0 && "World"); // 0 (0 is falsy, so returns first value)

// 1️⃣4️⃣ Double NOT (!!) coercion

console.log(!!"Hello"); // true

console.log(!!0); // false

// 1️⃣5️⃣ Using `typeof` in strange cases

console.log(typeof null); // object ❗

console.log(typeof NaN); // number ❗

console.log(typeof function () { }); // function ❗

console.log(typeof []); // object ❗ (arrays are objects)

// Extra: Checking empty object

console.log(Object.keys({}).length === 0); // true (empty object)

console.log(Object.keys({ a: 1 }).length === 0); // false (non-empty object)
```