Grace_Xu Homework JS1

Difference Between let, const, and var

- var:
    - Introduced in ES5.
    - Function-scoped: Accessible throughout the function where it's declared.
    - Redeclaration allowed: Can declare the same variable multiple times.
    - Hoisted: Moved to the top of its scope but initialized as undefined.
- let:
    - Introduced in ES6.
    - Block-scoped: Limited to the block where it's defined.
    - No redeclaration: Cannot redeclare the same variable in the same scope.
    - Hoisted but uninitialized: Accessible only after its declaration.
- const:
    - Introduced in ES6.
    - Block-scoped: Same as let.
    - Immutable: Cannot reassign, but objects/arrays can be mutated.
    - Hoisted but uninitialized: Same behavior as let.

What is Hoisting?

Hoisting is JavaScript's behavior of moving variable and function declarations to the top of their scope before code execution.

How Variables/Functions Hoist Differently

- var: Hoisted with undefined initialization.
- let/const: Hoisted but remain in a temporal dead zone until declared.
- Functions: Fully hoisted, meaning they can be called before their definition.

Reference Type vs Primitive Type

- Primitive Types: Stored as values.
    - Examples: string, number, boolean, null, undefined, symbol, bigint.
    - Immutable and compared by value.
- Reference Types: Stored as references (memory addresses).
    - Examples: object, array, function.
    - Mutable and compared by reference (i.e., two objects with the same data are not equal unless they reference the same memory).

What is Type Coercion?

Type coercion is the automatic or implicit conversion of values from one type to another.

Examples:

- Implicit:

  1 + "1"; // "11" (number to string)
- "5" - 2; // 3 (string to number)

- Explicit:

  Number("123"); // 123
- String(123); // "123"

Difference Between == and ===

- == (Loose Comparison):
  - Compares values with type coercion.
  - Example:

    "1" == 1; // true
    null == undefined; // true

- === (Strict Comparison):
  - Compares values without type coercion (checks type and value).
  - Example:

    "1" === 1; // false
    null === undefined; // false

Which to Use?

- Always use === to avoid unexpected behavior from type coercion.

Notes: Differences and Key Concepts in JavaScript

var vs let vs const

1. var:
    - Introduced in ES5.
    - Scope: Function-scoped.
    - Redeclaration: Allowed.
    - Hoisting: Hoisted and initialized as undefined.
2. let:
    - Introduced in ES6.
    - Scope: Block-scoped.
    - Redeclaration: Not allowed in the same scope.
    - Hoisting: Hoisted but uninitialized (temporal dead zone).
3. const:
    - Introduced in ES6.
    - Scope: Block-scoped (same as let).
    - Immutability: Cannot be reassigned, but objects/arrays can be mutated.
    - Hoisting: Same behavior as let.

Hoisting

- JavaScript's behavior of moving declarations to the top of their scope.
- Differences:
    - var: Hoisted with undefined initialization.
    - let/const: Hoisted but remain in a temporal dead zone until declared.
    - Functions: Fully hoisted (can be invoked before their declaration).

Reference Type vs Primitive Type

1. Primitive Types:
    - Stored as values.
    - Examples: string, number, boolean, null, undefined, symbol, bigint.
    - Immutable and compared by value.
2. Reference Types:
    - Stored as references (memory addresses).
    - Examples: object, array, function.
    - Mutable and compared by reference (two objects with identical data are not equal unless referencing the same memory).

Type Coercion

- Automatic conversion between types.

1. Implicit:
    - Example: 1 + "1" → "11" (number to string).
    - Example: "5" - 2 → 3 (string to number).
2. Explicit:
    - Example: Number("123") → 123.
    - Example: String(123) → "123".