1. var,let,const 有什么区别?

1. 核心概念

`var`, `let`, 和 `const` 是 JavaScript 中用于声明变量的关键字。它们的主要区别在于**作用域 (scope)**、**提升 (hoisting)** 行为以及**可变性 (mutability)**。

- `let` 和 `const` 引入了**块级作用域 (block scope)**，是 ES6 推荐的变量声明方式。
- `var` 使用的是**函数作用域 (function scope)**，存在一些容易导致错误的特性。

2. 为什么需要 `let` 和 `const`？

`let` 和 `const` 的出现解决了 `var` 带来的几个关键痛点：

1. **解决变量提升导致的混乱**: `var` 声明的变量会被提升到其作用域顶部，并用 `undefined` 初始化，可能在声明前被访问而不报错，违反直觉。`let` 和 `const` 存在**暂时性死区 (TDZ)**，在声明前访问会直接抛出 `ReferenceError`。
2. **提供块级作用域**: `var` 没有块级作用域，导致在 `for` 循环或 `if` 语句块中声明的变量会泄露到外部。`let` 和 `const` 仅在声明它们的代码块 (`{...}`) 内有效，提供了更强的封装性。
3. **引入常量能力**: `const` 允许声明一个值不能被重新赋的常量，有助于提升代码的可维护性和健壮性（防止意外修改）。

3. API 与用法 (API & Usage)

`var`

- **作用域**: 函数作用域或全局作用域。
- **提升**: 变量提升，初始化为 `undefined`。
- **重复声明**: 允许在同一作用域内重复声明。
- **可变性**: 可以重新赋值。

```
function varExample() {
  console.log(a); // 输出: undefined (提升)
  var a = 10;
  console.log(a); // 输出: 10

  if (true) {
    var a = 20; // 覆盖了函数作用域中的 a
  }
  console.log(a); // 输出: 20
}
```

`let`

- **作用域**: 块级作用域 (`{...}`)。
- **提升**: 不提升（存在暂时性死区）。
- **重复声明**: 不允许在同一作用域内重复声明。
- **可变性**: 可以重新赋值。

```
function letExample() {
  // console.log(b); // ReferenceError: Cannot access 'b' before
initialization (TDZ)
  let b = 10;
  console.log(b); // 输出: 10

  if (true) {
    let b = 20; // 这是一个新的、独立的变量，只在 if 块内有效
    console.log(b); // 输出: 20
  }
  console.log(b); // 输出: 10
}
```

`const`

- **作用域**: 块级作用域 (`{...}`)。
- **提升**: 不提升（存在暂时性死区）。
- **重复声明**: 不允许在同一作用域内重复声明。
- **可变性**: **不能**重新赋值。声明时必须初始化。

```
function constExample() {
  const c = 30;
  // c = 40; // TypeError: Assignment to constant variable.

  // 注意：对于对象或数组，const 保证的是变量绑定的引用不变，
  // 但对象或数组自身的内容是可变的。
  const obj = { name: 'React' };
  obj.name = 'Vue'; // 这是允许的
  console.log(obj.name); // 输出: 'Vue'

  // obj = {}; // TypeError: Assignment to constant variable.
}
```

4. 关键注意事项 (Key Considerations)

1. **暂时性死区 (TDZ)**: `let` 和 `const` 声明的变量，从其作用域开始到声明语句之间是TDZ。在此区域内访问变量会抛出 `ReferenceError` 。

2. **全局对象属性**: 在全局作用域下，使用 `var` 声明的变量会成为 `window` (浏览器环境) 或 `global` (Node.js 环境) 对象的属性。`let` 和 `const` 则不会。
3. **`const` 的不变性**: `const` 保证的是变量指向的内存地址（引用）不可变，而不是该地址中的数据不可变。对于引用类型（如对象、数组），其内部属性或元素是可以修改的。
4. **优先使用 `const`**: 默认应优先使用 `const` 来声明变量，只有当确定变量需要被重新赋值时，才使用 `let`。这能增强代码的可靠性，避免不必要的变量修改。

5. 参考资料 (References)

- MDN Web Docs: let
- MDN Web Docs: const
- MDN Web Docs: var
- ECMAScript® 2025 Language Specification: Let and Const Declarations