# Hoisting (variables and functions)

Let's explore **hoisting** in JavaScript in detail — it's a foundational concept that can help you avoid tricky bugs.

---

## 🚀 What Is Hoisting?

**Hoisting** is JavaScript's default behavior of **moving declarations to the top of their scope** before code execution.

It applies to:

- **Variables** (`var`, `let`, `const`)
- **Functions** (function declarations)

---

## 🧠 Key Principle

> JavaScript only hoists declarations, not initializations.

---

## ✅ Function Hoisting

```
greet();

function greet() {
  console.log("Hello!");
}
```

### Output:

```
Hello!
```

- ✅ Function declarations are fully hoisted — both **name and definition**.
- You can **call them before** they're declared.

---

## ❌ Function Expressions (Not Hoisted the Same Way)

```
greet(); // ❌ TypeError: greet is not a function

var greet = function () {
  console.log("Hi!");
};
```

- Only the **variable** `greet` **is hoisted** (as `undefined`), not the function.
- You get a **TypeError**, not a ReferenceError, because `greet` exists but isn't callable yet.

---

## ✅ Variable Hoisting

### Using `var`:

```
console.log(a); // undefined
var a = 10;
```

- The declaration `var a` is hoisted, but the assignment happens later.
- So `a` exists but is `undefined`.

### Behind the scenes:

```
var a; // hoisted
console.log(a); // undefined
a = 10;
```

### Using `let` and `const` (Temporal Dead Zone)

```
console.log(b); // ❌ ReferenceError
let b = 20;
```

- `let` and `const` are also hoisted **but not initialized**.
- Accessing them before the declaration results in a **ReferenceError**.
- The time between entering the scope and the declaration is called the **Temporal Dead Zone (TDZ)**.

---

## 🧪 Hoisting Order

- Function declarations are hoisted **before** variables.
- But if a variable has the **same name** as a function, the variable can **overwrite** the function reference.

```
console.log(foo()); // ❌ TypeError: foo is not a function

var foo = function () {
  return "bar";
};

function foo() {
  return "baz";
}
```

- Function `foo()` is hoisted first.
- Then `var foo` hoists the variable (as `undefined`), overwriting the function reference.

---

## ✅ Summary

| Declaration Type | Hoisted | Initialized | Access Before Declaration |
| --- | --- | --- | --- |
| `var` | ✅ Yes | ✅ Yes (`undefined`) | ✅ (value: `undefined`) |
| `let` / `const` | ✅ Yes | ❌ No | ❌ ReferenceError (TDZ) |
| `function` | ✅ Yes | ✅ Yes | ✅ |
| Function expression (`var`) | ✅ (variable only) | ❌ No | ❌ TypeError |

---

Would you like a visual diagram of the hoisting process or TDZ next?