Function Declaration vs Expression in JavaScript

Agenda

1. What are Function Declarations & Expressions?
2. Key Differences (Hoisting, Syntax, Use Cases)
3. Examples & Comparison Table
4. Interview Pro Tips
5. Best Practices
6. Q&A + Coding Challenge
7. Quiz Time

Introduction to Function Declaration vs Expression

Function Declaration

```
function calculateTotal() {
  // Logic here
}
```

✅ *Hoisted* (can be called before declaration)
✅ *Named by default* (better stack traces)
✅ *Best for*: Core logic, reusable functions

Function Expression

```
const calculateTotal = function() {
  // Logic here
};
```

⏳ *Not hoisted* (must be defined before use)
🔗 *Flexible* (can be anonymous, assigned dynamically)
💡 *Best for*: Callbacks, conditionally assigned functions

Central Divider:
⚖️ **Key Difference:**
"Declarations are like planting a flag, Expressions are like building a bridge"

🔍 **Why It Matters:**
• Impacts code organization & debugging
• Critical for interview technical questions

Function Declaration

Key Features:

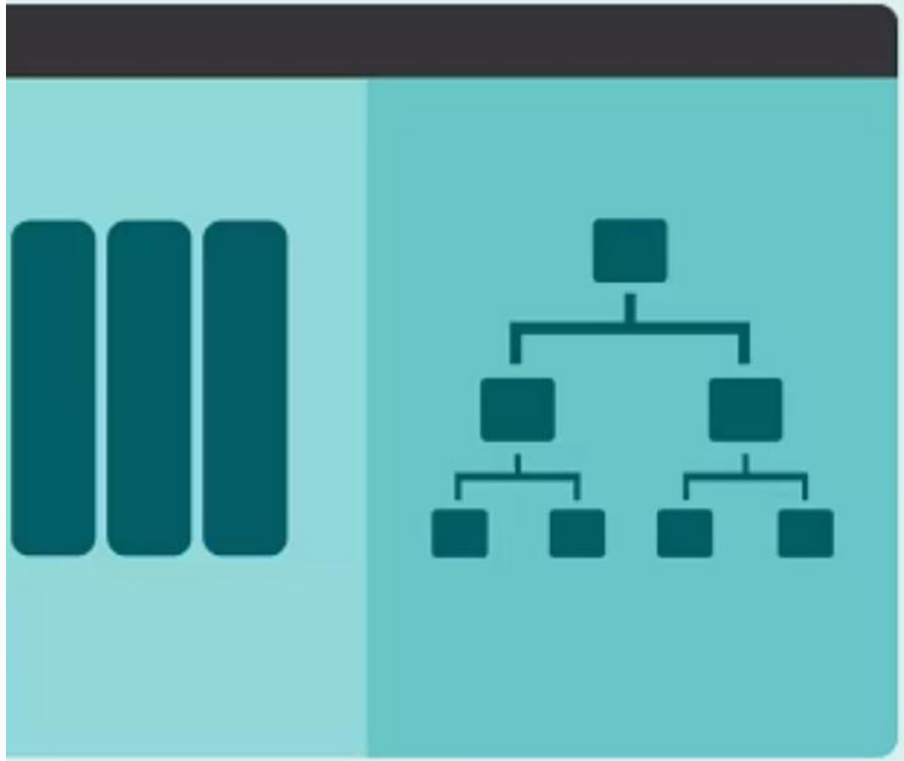
- Hoisted to the top of the scope
- Can be called before declaration
- Syntax: `function name() {...}`

Example:

```
console.log(square(5)); // Works (hoisted)
function square(n) { return n * n; }
```

Use Case:

- General-purpose functions
- When you need hoisting



Function Expression

Key Features:

- Not hoisted (must be defined before calling)
- Can be anonymous or named
- Syntax: `const fn = function() {...}`

Example:

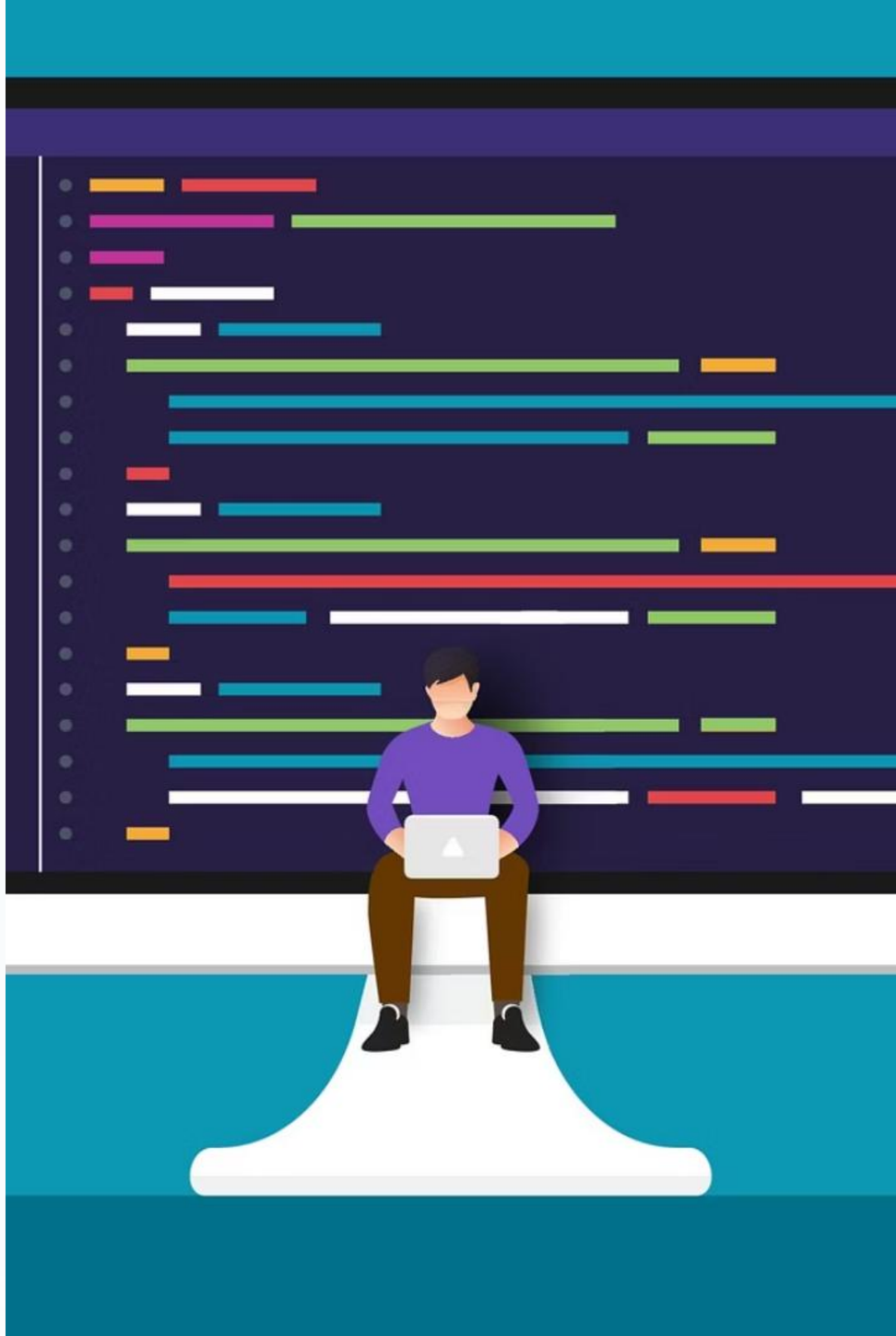
```
console.log(square(5)); // Error!
const square = function(n) { return n * n; };
```

Use Case:

- Callbacks
- Conditional function definitions

Key Differences Table

Feature	Declaration	Expression
Hoisting	✅ Yes	❌ No
Syntax	`function x() {...}`	`const x = function() {...}`
Use Cases	General logic	Callbacks, IIFEs
Anonymous?	❌ No	✅ Yes



Interview Pro Tips

✅ Must-Know Questions:

1. "Why does this function work before its declaration?" (Hoisting)
2. "When would you use an expression over a declaration?"
3. "What's the difference between `const fn = function() {}` and `function fn() {}`?"

❌ Avoid:

- Using expressions before declaration (ReferenceError)
- Overusing anonymous functions (hurts debugging)

Best Practices

✓ Use Declarations for:

- Main logic (better stack traces)
- When hoisting is needed

✓ Use Expressions for:

- Callbacks (e.g., `array.map(function(x) {...})`)
- Conditional assignments (e.g., `const apiCall = isProd ? prodFn : devFn`)

✓ Always Name Functions:

- Helps debugging (anonymous → "anonymous" in stack traces)

Q&A + Practice Challenge

Q&A:

- Can you reassign a function expression?
- Which one is better for performance?

Challenge:

```
// Convert this declaration to an expression:
function greet(name) { return `Hello ${name}!`; }

// Solution:
const greet = function(name) { return `Hello ${name}!`; };
```

Quiz Time!

1️⃣ What prints?

```
console.log(typeof foo);
function foo() {}
var foo = 1;
```

A) "function" ✅
B) "number"

2️⃣ Which supports hoisting?

A) Declaration ✅
B) Expression

3️⃣ Best for callbacks?

A) Declaration
B) Expression ✅

Resources

- Free eBook: "JavaScript Quick Reference"

W3Schools.com

https://www.geeksforgeeks.org/javascript/

- Follow for daily JS tips
- GitHub repo with code examples

Thank You!

Recap:

- Declarations = Hoisted, general use
- Expressions = Flexible, callback-friendly

CTA:

- Try the challenges on JSFiddle!
- Subscribe for more deep dives!

Try coding exercises on HackerRank or LeetCode!