Core JavaScript Mastery Guide

Perfect! Let's **deep dive into Anonymous Functions** — very important for callbacks, IIFEs, and many interview scenarios.

1️⃣ Anonymous Function — Definition

An **Anonymous Function** is a **function without a name**.

- Usually **assigned to a variable**, passed as an argument, or executed immediately.
- Cannot reference itself internally unless assigned to a variable.

```
js

const greet = function(name) {
  console.log("Hello " + name);
};
greet("Anoop"); // Hello Anoop
```

Here, the function has **no internal name**, it's stored in `greet`.

2️⃣ Syntax Variants

(a) Variable Assignment

```
js

const sum = function(a, b) {
  return a + b;
};
console.log(sum(2,3)); // 5
```

(b) As Callback

```
js

setTimeout(function() {
  console.log("Hello after 1 second");
```

```
}, 1000);
```

(c) Immediately Invoked Function Expression (IIFE)

```
js

(function() {
  console.log("IIFE executed!");
})();
```

3️⃣ Key Points

1. **No Name**

- Cannot reference itself internally.

```
js

const f = function() { console.log(typeof f); };
f(); // function (because variable stores function)
```

2. **Used as Callback**

- Perfect for functions that run **once** or are passed as arguments.

```
js

[1,2,3].map(function(n){ return n*2; }); // [2,4,6]
```

3. **Closures**

- Anonymous functions are commonly used in closures:

```
js

const counter = (function() {
  let count = 0;
  return function() { return ++count; };
})();
console.log(counter()); // 1
console.log(counter()); // 2
```

4. **Cannot be Constructed**

```
js
```

```
const Person = function(name) { this.name = name; };
const p = new Person("Anoop"); // ❌ works only if used like normal function expression
```

4️⃣ Differences vs Named Function Expression

Feature	Anonymous Function	Named Function Expression
Name	None	Exists internally
Recursion	❌ Cannot call itself internally	✅ Can call internal name
Stack Trace	“anonymous”	Shows internal name (better debugging)
Common Use	Callbacks, IIFEs, closures	Recursion, debugging

5️⃣ Tricky Examples (Interview Level)

Example 1: Callback

```
js

const nums = [1,2,3];
const doubles = nums.map(function(n) { return n*2; });
console.log(doubles); // [2,4,6]
```

Example 2: IIFE

```
js

(function() {
  let name = "Anoop";
  console.log(name);
})(); // "Anoop"
// name not accessible outside
```

Example 3: Closures

```js
const add = (function() {
  let total = 0;
  return function(n) { return total += n; };
})();
console.log(add(5)); // 5
console.log(add(3)); // 8
```

Example 4: Reassignment Safety

```js
let f = function(n){ return n*2; };
let g = f;
f = null;
console.log(g(5)); // 10 ✅ works because function stored in variable g
```

6️⃣ Key Interview Notes

1. Anonymous functions are **used everywhere in JS**: callbacks, closures, event handlers, IIFEs.
2. Cannot call themselves internally → recursion not possible unless assigned to a variable.
3. Often replaced by **arrow functions** in modern JS for brevity.
4. **IIFE pattern** is classic for module pattern / private variables.

✅ Summary:

- **Anonymous Function** = unnamed function.
- Used as **callback, closure, IIFE**.
- No internal name → cannot self-reference.
- Perfect for **inline, temporary functionality**.

If you want, next we can **compare all three: Function Expression, Named Function Expression, and Anonymous Function** with **tricky examples + arrow function conversion** — which is exactly how interviews test this topic.

Do you want me to do that next?