✅ JavaScript this Keyword & Arrow Function - Full Notes

🔥 1. What is this?

- this refers to the **current execution context**.
- Think of it like "**who is calling me right now?**"

Context	Value of this
Inside an object method	Refers to that **object** itself
In global scope (node)	Refers to an **empty object {}** in strict mode
In global scope (browser)	Refers to the **window** object
Inside a function	undefined in strict mode (in browser window)
Inside arrow function	this of **lexical parent** (No own this)

📝 2. Code Breakdown (your code + explanation)

👉 Object Method with this

```
js
CopyEdit
const user = {
  username: "Pranay",
  price: 999,
  welcomeMessage: function() {
    console.log(`${this.username}, welcome to website`);
  }
}
user.welcomeMessage();
user.username = "Pratik";
user.welcomeMessage();
```

✅ Explanation:

- Here, this.username refers to user.username.
- When you changed username to "Pratik", this.username reflected that change.
- Why? Because **method call through object => this refers to that object**.

👉 this in Normal Function

js

CopyEdit

```
function chai() {
  let username = "pranay";
  console.log(this.username); // undefined
}
chai();
```

✅ **Explanation:**

- Regular function call => this refers to **global object**.
- But username is a **local variable**, not a property of global.
- So, this.username is undefined.
- In **strict mode**, this is undefined.

👉 this in Function Expression

js

CopyEdit

```
const chai = function() {
  let username = "pranay";
  console.log(this.username); // undefined
}
chai();
```

- Same result. Still a **normal function call**.
- So this refers to global object → again undefined.

👉 this in Arrow Function

js

CopyEdit

```
const chai = () => {
  let username = "pranay";
```

```
  console.log(this); // {}
}
chai();
```

✅ **Explanation:**

- Arrow functions **don't have their own this**.
- They **lexically inherit this from parent scope**.
- Since here, parent is global scope → {} in Node.js.

🆚 **3. Difference between Normal Function & Arrow Function (w.r.t this)**

Feature	Normal Function	Arrow Function
Own this?	✅ Yes	❌ No (inherits from parent scope)
In Object Method	this refers to that object	Prefer normal function for methods
In Event Listeners	this refers to the DOM element	Arrow inherits from parent (won't refer to element)
Constructor Function	✅ Can use this	❌ Cannot be used as constructor

---

🚀 **4. Arrow Function Syntax (Quick Recap)**

Type	Syntax Example	Notes
Normal arrow function	const add = (a, b) => { return a + b; }	Curly braces need return
Implicit return	const add = (a, b) => a + b	No {}, auto-return
Single parameter shortcut	const square = n => n * n	Can skip parentheses
Returning object literal	const getObj = () => ({ name: 'Pranay' })	Wrap object in ()

🧠 **5. Easy Tricks to Remember**

🔹 **Trick 1: "Caller decides this"**

- Who **calls the function** → that decides this.
- Object calls → refers to object.
- Plain function call → global or undefined.

🔹 **Trick 2: Arrow function → "Borrow this from Parent"**

- Think of arrow functions as **"copycat"**.
- No own this, just borrows from where it was defined.

🔹 **Trick 3: Event Listeners Example**

```
js
CopyEdit
button.addEventListener('click', function() {
  console.log(this); // refers to button
});

button.addEventListener('click', () => {
  console.log(this); // refers to parent scope (probably window or undefined)
});
```

👉 Use **normal function for events** if you want this to refer to the element.

🔹 **Trick 4: Don't use arrow function in Object methods (if you need this)**

```
js
CopyEdit
const user = {
  name: "Pranay",
  greet: () => {
    console.log(this.name); // undefined
  }
}
user.greet();
```

👉 Use normal function for methods.

📝 **6. Interview Ready Points**

- **this in arrow function** → lexical scope, no own this.
- In normal function → dynamic this based on caller.
- Arrow function is great for callbacks, but **NOT for object methods or constructors**.
- Arrow functions can do **implicit return** (clean code).
- In Node.js global scope, this is {}.

✅ 7. Pro Tip: Visualization

Whenever confused, imagine this mental image:

🗣️ **"Who is calling me?"** → This is this.

For arrow functions:

🏠 **"I don't care who calls me, I'll just take my this from my home (parent scope)"**.

📄 Final Summary Table

Function Type	this value	Usage
Object method (normal fn)	Refers to object	✅ Preferred
Object method (arrow fn)	Refers to parent scope, not object	❌ Not preferred
Function in global scope	Global object / undefined	Careful with context
Arrow in global scope	Lexical parent scope (global)	Useful for callbacks
Event listener (normal fn)	Refers to element	✅ Preferred
Event listener (arrow fn)	Refers to parent scope	❌ Avoid if need element reference

---