Difference Between Arrow and Normal Functions in JavaScript (with `this`)

Code Recap

```
const chai = () => {
    let username = "talka sing";
    console.log(this.username);
}
chai();

const chai2 = function() {
    let usertwo = "chaman";
    console.log(this.usertwo);
}
chai2();
```

Key Differences Between Arrow and Normal Functions

Feature	Arrow Function (chai)	Normal Function (chai2)
Function Type	Arrow function	Regular function
`this` refers to	Lexical (outer) scope	The caller
Has its own `this`?	No	Yes
Can access local vars via `this`?	No	No
Example Result	`this.username` -> undefined	`this.usertwo` -> undefined

Explanation

Arrow Function (chai):

- Does NOT have its own `this`
- Inherits `this` from outer (lexical) scope

Difference Between Arrow and Normal Functions in JavaScript (with `this`)

- `username` is local to the function, not accessible via `this`
- So, `this.username` is undefined

Normal Function (chai2):

- Has its own `this`
- But `usertwo` is a local variable, not a property of `this`
- So, `this.usertwo` is also undefined

To make the difference clearer, assign values to `this`:

```
const obj = {
    usertwo: "chaman",
    chai2: function() {
        console.log(this.usertwo); // Works: prints "chaman"
    }
};
obj.chai2();
```

Conclusion

- Use normal functions when you need dynamic `this` based on the caller.
- Use arrow functions when you want `this` to remain the same as outer context.
- Local variables declared with `let` inside a function are never accessible via `this` unless explicitly assigned.