We are going to learn **THIS** today

a. This keyword in browser (strict and non strict)
b. This keyword in Node.js (strict and non strict)
c. This keyword with Arrow Functions

This: always points to the **calling object** of the function.

```
console.log("Scenario 1:");
console.log(this);
```

```
console.log("Scenario 2:");
function fnGlobal() {
  console.log(this);
}
fnGlobal();
```

```
console.log("Scenario 3:");
var obj = {
  fn: function () {
    console.log(this);
  }
};
obj.fn();


console.log("Scenario 4:");
var obj2 = {
  fn: function () {
    console.log(this);
    var nestedFn = function () {
      console.log(this);
    };
    nestedFn();
  }
};
obj2.fn();
```

Ecmascript (ES6)

Let
Arrow function
Use Strict

Developers use to face issues with the var keyword

Arrow function is a way to declare a function where you don't need to use a function keyword and you declare a function using =>
And when you only want to return from an arrow function
var a = (d) => d;

```
const obj = {
  name: 'John',
  regularFunc: function() {
    console.log('Regular function:', this.name);
  }
};

obj.regularFunc();
```

```
const obj = {
  name: 'John',
  arrowFunc: () => {
    console.log('Arrow function:', this.name);
  }
};

obj.arrowFunc();
```

In an arrow function this doesn't point to the calling object

In an arrow function the this points to the surrounding scope or the parent scope, enclosing scope