25. OOPs : 'this' & Arrow functions

Agenda

- This Keyword
  - browser
  - node JS
  - arrow functions
-

OOPs

(Object Oriented Programming.)

this :

(Special Reserved keyword in JS)

Represents Current execution execution or current object

Context → ??

JS → function fun() {

```
    console.log('Hello')
}

func();
```

① Global Context

- Outside any function.

this → browser : window object

NodeJs : global

② Function Context

OOPS + JS

1. Encapsulation
2. Abstraction
3. Polymorphism
4. Inheritance
5. Overloading.

helps to organize & structure code well in large codebases.

'this'

Encapsulation :

Abstraction

↳ Hiding Complex information/implementation details of a system.

↳ Exposing necessary details to the

encapsulation (3)

methods (function) | variables

↑

Class (function)

Abstraction

- Hiding unwanted info.

Encapsulation

- Bundling data together.
- ~~Info to a method~~
- Hide method/data in a single entity

Function Context (this)

fun()

→ Different ways to invoke a function

(to be discussed in coming classes)

1. Regular function
2. Method invocation
3. Constructor invocation
4. call() & apply methods
5. Arrow functions

2nd (C+I)

this	Non strict	Strict
1) Regular invocation	global object	Undefined
2) Method	'this' refers to Object that the	

Method is called on.

3) Constructor invocation

'this' refers to instance being created by constructor function.

4) call & apply

Explicitly set 'this' value

5) Arrow

- Arrow fn don't have their own 'this' value
- Retain value of enclosing lexical context.

Scenario Code Output Explanation

Node Strict Mode:

this-node-env-strict-mode.js

aug-awesome-batch > fullstack-lld-august > 25 - oops 1 this > this-node-env-strict-mode.js > ...

```
// this behaviour in """""""""""nodeJS""""""""""" strict mode.

'use strict'

// Scenario 1
console.log('Scenario 1: ')
console.log(this); // Output : {}
// Explantaion -- in global context, this refers to empty object


// Scenario 2

console.log('Scenario 2:');
function fnGlobal() {
    console.log(this); // undefined
}
fnGlobal();
// exaplnation - in strict mode, when u call a function without specifying i
// it is set to undefined.
// since, change prevents function from unintentially modifying the global c

// Scenario 3

console.log('Scenario 3:')
var obj = {
    prop: 'Property',
    fn: function () {
        console.log(this); // object itself - { fn: [Function: fn] }
        console.log(this.prop); // Property
    }
}
obj.fn();
// explanation - inside an object method, this refers to the object itself.

console.log('Scenario 4 :')

var obj2 = {
    fn: function () {
        console.log(this); // object itself - { fn: [Function: fn] }
        var nestedFn = function () {
            console.log(this); // undefined
        }
        nestedFn()
    }
}
obj2.fn();
// output : nestedFn = global object
// explanation - inside a nested function, this reverts back to global object.
```

At Node non strictMode

aug-awesome-batch > fullstack-lld-august > 25 - oops 1 this > JS this-node-env.js > fnGlobal

```
1  // this behaviour in """""""""""nodeJS""""""""""".
2
3  // Scenario 1
4  console.log('Scenario 1: ')
5  console.log(this); // Output : {}
6  // Explantaion -- in global context, this refers to empty object
7
8  // Scenario 2
9
10 console.log('Scenario 2:');
11 function fnGlobal() {
12     console.log(this); // global object
13 }
14 fnGlobal();
15 // exaplnation - inside a regular function, this refers to global object
16
17 // Scenario 3
18
19 console.log('Scenario 3:')
20 var obj = {
21     fn: function () {
22         console.log(this); // object itself - { fn: [Function: fn] }
23     }
```

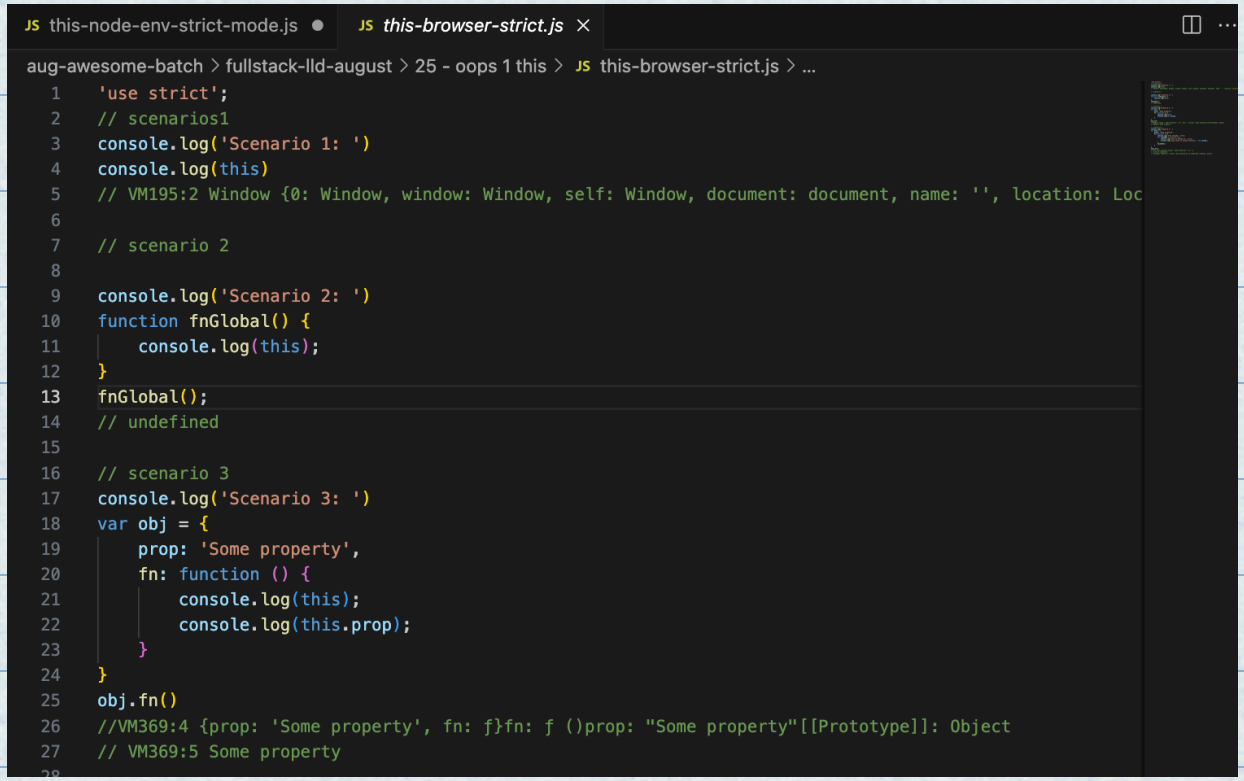
main* ⊗ 0 ⚠ 0 0 Ln 12, Col 23 Spaces: 4 UTF-8 LF {} JavaScript Go Live

aug-awesome-batch > fullstack-lld-august > 25 - oops 1 this > JS this-node-env.js > fnGlobal

```
19 console.log('Scenario 3:')
20 var obj = {
21     fn: function () {
22         console.log(this); // object itself - { fn: [Function: fn] }
23     }
24 }
25 obj.fn();
26 // explanation - inside an object method, this refers to the object itself.
27
28 console.log('Scenario 4 :')
29
30 var obj2 = {
31     fn: function () {
32         console.log(this); // object itself - { fn: [Function: fn] }
33         var nestedFn = function () {
34             console.log(this); // global object
35         }
36         nestedFn()
37     }
38 }
39 obj2.fn();
40 // output : nestedFn = global object
41 // explanation - inside a nested function, this reverts back to global object.
```

main* ⊗ 0 ⚠ 0 0 Ln 12, Col 23 Spaces: 4 UTF-8 LF {} JavaScript Go Live

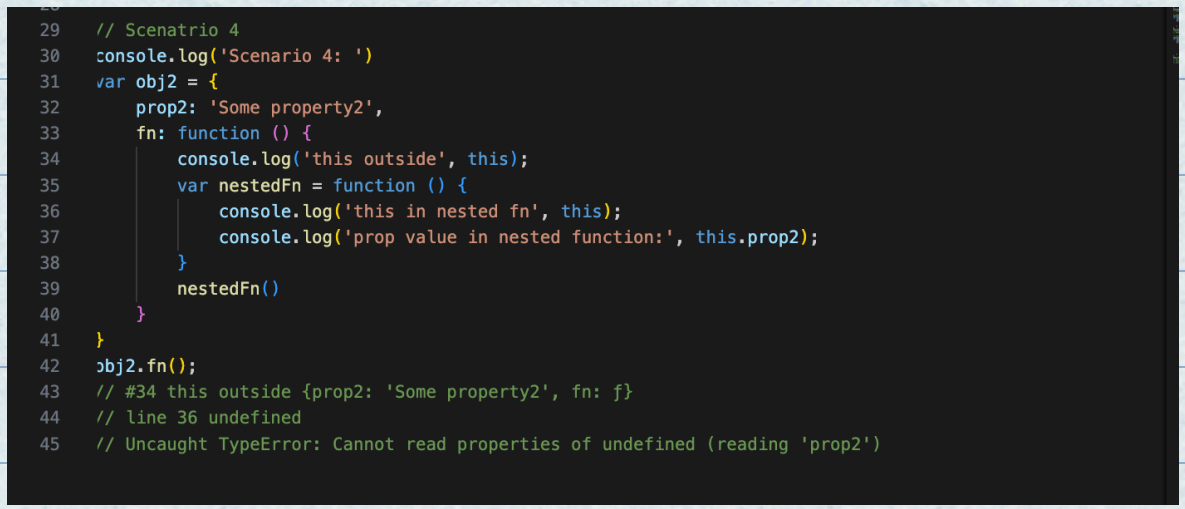
* Browser Strict Mode →



JS this-node-env-strict-mode.js ● JS this-browser-strict.js ×

aug-awesome-batch > fullstack-lld-august > 25 - oops 1 this > JS this-browser-strict.js > ...

```
1   'use strict';
2   // scenarios1
3   console.log('Scenario 1: ')
4   console.log(this)
5   // VM195:2 Window {0: Window, window: Window, self: Window, document: document, name: '', location: Loc
6
7   // scenario 2
8
9   console.log('Scenario 2: ')
10  function fnGlobal() {
11      console.log(this);
12  }
13  fnGlobal();
14  // undefined
15
16  // scenario 3
17  console.log('Scenario 3: ')
18  var obj = {
19      prop: 'Some property',
20      fn: function () {
21          console.log(this);
22          console.log(this.prop);
23      }
24  }
25  obj.fn()
26  //VM369:4 {prop: 'Some property', fn: ƒ}fn: ƒ ()prop: "Some property"[[Prototype]]: Object
27  // VM369:5 Some property
```



```
29  // Scenatrio 4
30  console.log('Scenario 4: ')
31  var obj2 = {
32      prop2: 'Some property2',
33      fn: function () {
34          console.log('this outside', this);
35          var nestedFn = function () {
36              console.log('this in nested fn', this);
37              console.log('prop value in nested function:', this.prop2);
38          }
39          nestedFn()
40      }
41  }
42  obj2.fn();
43  // #34 this outside {prop2: 'Some property2', fn: ƒ}
44  // line 36 undefined
45  // Uncaught TypeError: Cannot read properties of undefined (reading 'prop2')
```

* Browser Non Strict Mode →

```
> // scenarios1
  console.log(this)

                                                                    VM593:2
  ▶ Window {0: global, 1: Window, 2: global, window: Window, self: Win
    dow, document: document, name: '', location: Location, …}

<· undefined

> // scenario 2
  function fnGlobal() {
      console.log(this);
  }
  fnGlobal();

                                                                    VM633:3
  ▶ Window {0: global, 1: Window, 2: global, window: Window, self: Win
    dow, document: document, name: '', location: Location, …}
```

```
> // scenario 3
  var obj = {
      prop: 'Some property',
      fn: function () {
          console.log(this);
          console.log(this.prop);
      }
  }
  obj.fn()

  ▶ {prop: 'Some property', fn: f}                               VM663:5

  Some property                                                  VM663:6
```

```
> // scenario 4
  var obj2 = {
      prop2: 'Some property2',
      fn: function () {
          console.log('this outside', this);
          var nestedFn = function () {
              console.log('this in nested fn', this);
              console.log('prop value in nested function:',
  this.prop2);
          }
          nestedFn()
      }
  }
  obj2.fn();

  this outside ▶ {prop2: 'Some property2', fn: f}                VM688:5

  this in nested fn                                              VM688:7
  ▶ Window {0: global, 1: Window, 2: global, window: Window, self: Win
    dow, document: document, name: '', location: Location, …}

  prop value in nested function: undefined                       VM688:8
```