

86
 APRIL 2020
 this → not → variable
 this → Keyword → refer to object
 cannot change value
 In object Method w/ this → object
 Week 14 | 092-274

01

Wednesday

- 1) Alone → global object
- 2) In function
- 3) In function → in strict mode this → undefined
- 4) In event → refer to element that received the event (triggered)
- 5) Methods → calls, applies, binds } refer to this to any object

eg 1)

```

const person = {
  firstName: "John",
  lastName: "Doe",
  id: 5556,
  full: function() {
    return this.firstName + " " +
      this.lastName;
  }
};
person.full();
// o/p: "John Doe"
  
```

replace with return this;

eg 2)

```

let obj = {
  this: this,
  function myFunc() {
    return this;
  }
};
  
```

Object window

received event

target of object object

refer to HTML element

eg 3)

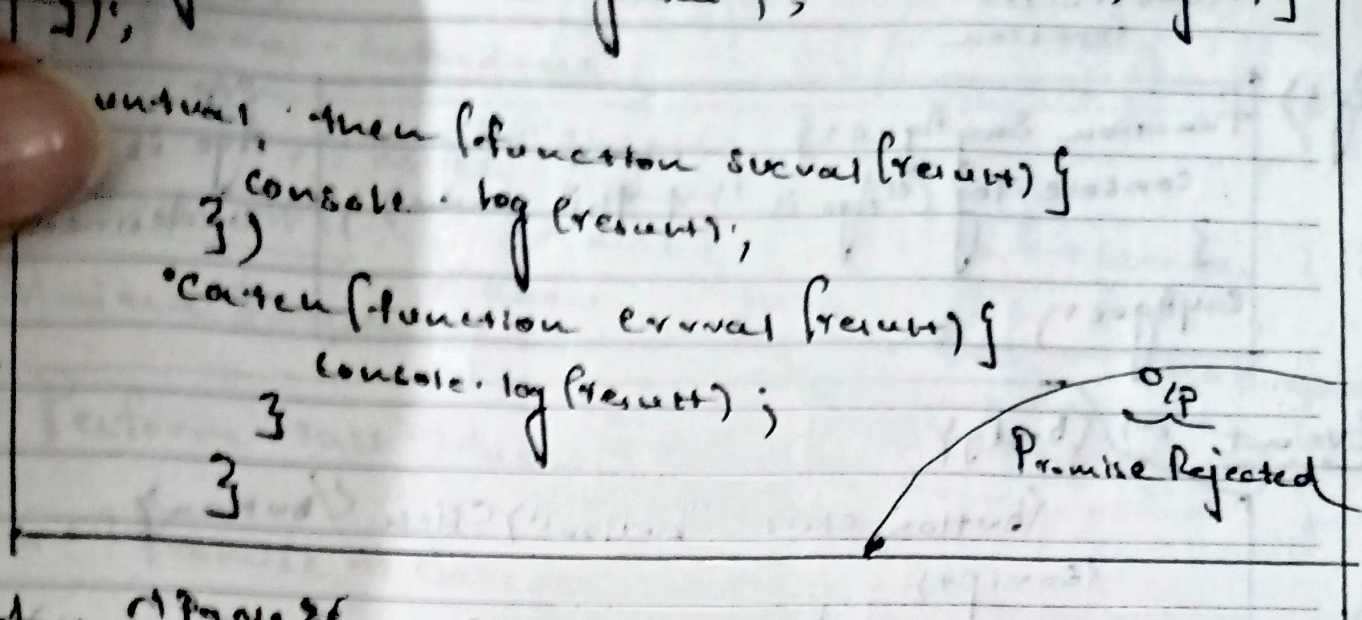
```

<button onclick = "this.style.display = 'none'">Click
</button>
  
```

1 2 3 4
 1) this precedence
 2) binds, applies
 3) object in global
 4) Method scope
 JS Arrow Function

MARCH 2020

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				



→ This fig no. 84

→ Global Context : alert ~~console.log (this)~~ ; [Object window]

→ Object Method context : This is used inside method of object
it refers to object itself.

JULY 2020						
Sa	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

→ Function Context

01

214-152 | Week 31

Saturday * If function is not in global object called directly (window in browser)

* If function called as `o1` this refers to object itself.
Method of object

* If function called using `new` this refers to newly created object.

1 `Inst` ← `Create` ← `new` Keyword
↓
2 `Constructor function`

<p>Ex 1</p> <pre> 3 function SayAge() { 4 console.log("Age is " + this.name); 5 } 6 SayAge(); </pre>	<p>O/P</p> <p>Age is undefined</p>
--	------------------------------------

Event

```

6 <body>
7   <button class="button" Click /button>
8   <script>
9     const but = document.querySelector("button");
10    but.addEventListener("click", function() {
11      console.log("Element Clicked");
12      console.log(this.toString());
13    });
14  </script>
15 </body>

```

O/P

Element Clicked
Object HTMLButtonElement

Arrow Functions

UST
020

192
AUGUST
2020

1) No own 'this' context; In head

1) From \rightarrow Surrounding code

03

Week 32 | 216-150

Monday
Parent
scope.

eg..

```
const person = {
```

```
  Design: "D9001",
```

```
  age: 21,
```

```
  id: 210,
```

```
  introduce: function() {
```

```
    const greet = ()  $\Rightarrow$  {
```

```
      console.log('this Design');
```

```
    };
```

```
  } greet;
```

```
};
```

```
person.introduce();
```

O/P
D9001

simple

Is this

- 1) Imagine \rightarrow var playground \rightarrow JS
- 2) When you are outside of any function/object \rightarrow this keyword refers global object
- 3) The global object in Browser environment \rightarrow window

eg 1)

```
<script>
let one = document.getElementById("one");
one.innerHTML = this;
</script>
```

- 4) When this is used inside \rightarrow method of an object refers to object itself

Think of it as object speaking about itself.

```
<script>
let two = doc.getElementById("two");
let three = doc.getElementById("three");
// object
const personalDetails = {
  qualified : 120000,
  age : 25,
  letterOfId : function() {
    two.innerHTML = this;
    three.innerHTML = "My current id is " + this.qualified;
  }
};
personalDetails.letterOfId();
</script>
```

If	description
Function called directly	this \rightarrow refers \rightarrow global object
Function called as method of an object	this \rightarrow refers \rightarrow object itself
Function called using new keyword or create instance of const function	this \rightarrow refers \rightarrow Newly created object

(g) >

</script>

```
let one = document.getElementById('one');
let two = document.getElementById('two');
let three = document.getElementById('three');

function sayHello1() {
  console.log('this');
  one.innerHTML = `Hello ${this.name}`;
}

sayHello1;
```

```
// -----

function sayHello2() {
  console.log('this');
  two.innerHTML = `Hello ${this.name}`;
}
```

```
const person = {
  name: 'Imtiaz',
  greet2: sayHello2
};
```

```
person.greet2();
```

```
// -----
```

```
function Person(name) {
  console.log('this');
  this.name = name;
}
```

```
const person3 = new Person('Imran');
```

```
three.innerHTML = person3.name;
```

</script>

Hello

Hello Imtiaz

Imran

console

Window { }

{ name: 'Imtiaz', greet2: f }

Person { }

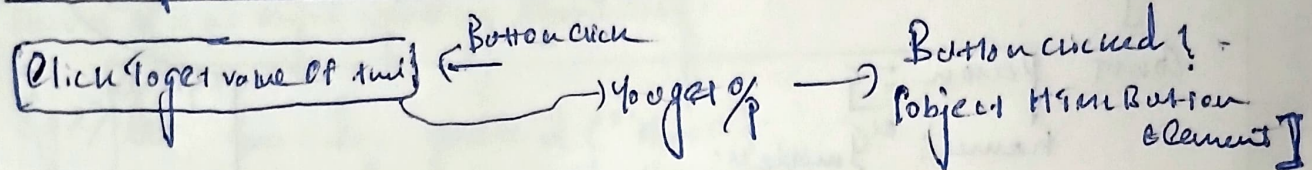
Event Handler

When event is triggered → this → refers → DOM element that triggered event.

```

<script>
  let one = doc.getElementById("one");
  let two = doc.getElementById("two");
  let btn = doc.querySelector("button");
  btn.addEventListener("click", function(e) {
    one.innerHTML = "Button clicked";
    two.innerHTML = this;
  });
</script>

```



Arrow Functions

They → not have → their own → this context. Instead inherits Value of this in parent scope. ← means surrounding code.

```

<script>
  let one = doc.getElementById("one");
  let two = doc.getElementById("two");
  const summer = {
    name: "Heat",
    destiny: function() {
      const khajali = 1;
      two.innerHTML = `this value: {this}`;
      console.log(this);
      one.innerHTML = `Hello & {this.name}`;
    }
  };
  console.log(summer);
  khajali();
</script>

```

