# The Jovascript this Keyword In Js, this keyword allows us to \* Reuse functions in different execution contexts, It means, a function once defined can be invoked for different objects using this keyword. \* Identifying the object in the current execution context when we invoke a method. The this keyword is very closely associated with Javascript functions When it comes to this, the fundamental thing to understand where a function is invoked. Becoz we don't know this keyword until the function is invoked.

	Date:
Types of Binding In Ja	vascrint
71	10 Pills
Default Binding Implicit Binding Explicit Binding Constructor Call Binding	
Implicit Binding	
e Explicit Binding	And with the
· Constructor Call Binding	
	Manage Commission
Default Binding In Java.	Script
V V V V V V V V V V V V V V V V V V V	
One of the first rule to rer	nember is that
the function housing a this	
One of the first rule to remember is that  If the function housing a this  reference is standalone functions  then that function is bound to the	
Then that function is bou	and to the
global object.	
Function about () &  Console. log. (this. name + 'is calling');  3	
console log (this name + (is calling!)	
3	is realing of
_ cons name = 'deepa':	
cons name = 'deepa'; alert (); Il deepa is calling	
- Standalone Function	
The second secon	Under March Albert

<

As your can see, name () is a standalone, unattached function, so it is bound to the global scope. As a result this, name reference resolves
to the global variable

cons name = (deepa) This oule doesn't hold if name() were to be defined in stoict mode. It will output Il Type Error: 'this' is undifined Implicit Binding In Java Script Acc. to binding rule in JavaScript, a function can use an object as its context only if that object is bound to it at call site. toreg > Function alert () {

Console log (this age + (years)); y (ons myObj = }

y age: 22

myObj. alut: alut;

myObj. alurt() 1122 years

when you call a function using dot motation, this is implicitly bound to the object the function is being called from. In this eg Since alert is being called from myObj , the this keyword is bound to myObj. So when about is called with myObj about this. age is 22, which is age property of myObj Another Eg > Function alert () {

console. log (this. age + (years)), const myObj = q alert : alert 8 nested Obj: S 3 alert: alert. myObj. alust (); 1/22 years my Obj. nested Obj. alut(); 1126 years

Explicit binding In Java Script If we want to force a function to use an object as its context without putting a property function sufference to object. we have 2 utility methods · call () · apply() Along with other set of utility functions, These 2 utilities are available to all functions in Jovascript via II Prototype JJ mechanism, To explicitly bind a function call to

context, you simply invoke a call ()

on that function and pass in context

object as parameter. For eg > Function alext() {

Console log (this age + (years)); Const my obj = 5 g age: 22;

alest. call (myobj) ; 1/ 22 years Even If you were to pass around the function multiple times to new variables (currying), every invocation will use same context because It has been locked (explicitly bound) to that object. This is called hard binding Eg-Function alext() {
Console.log(this.age); Const myObj' = Sage : 22;

3; const bar = function () {
alest call (my. Obj.); J) bar (); 1/22 Set Timeout (bars 100); 1/22 Il a hard bound boy' can no longer have it's this' context overnoden box · call (window); 1/22

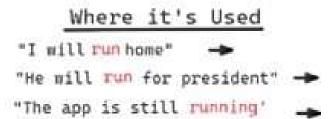
Hard binding is perfect way to lock a context into a function call and truly make that function into a method. Constructor Call binding In Java Script When a function is invoked with new keyword in front of it, known as Constructor call, following things occur A A brand new object is created A The newly constructed object is [[Prototype]] - linked to the function that Constructed it. A The newly constructed object is set as the this binding for that function call. function give. Age (age) {

this.age = age; const bor = new give Age (22); console log (bor age); 1/22

By calling, give Age (...) with new in front, we've constructed a new object as the this for Call. So new is final way that you can bind a function call's this

## JavaScript "this" keyword - Explained

"Run" is a polysemic word. It's meaning depends on the context in which it is used.



#### Meaning

move quickly on foot vying for election Software application is open and active

The this Keyword is very similar to "run", it points to a different object/scope depending on where it is used.

#### Where it's Used

#### 1. In a method

```
var me = {
    firstName: "Wingsloy",
    leatName: "Ubah",
    fullName: function() {
        return this firstName + * * + this.lastName;
    }
);
```

Refers to the owner object (me)

2. Alone and independent

```
var x = this.name;
```

Refers to the global object (Window)

3. In a function

```
function myFunction() {
  return this.name;
}
```

- Also refers to the global object

## JavaScript "this" keyword - Explained

### Where it's Used

4. In a function (strict mode)

```
"use strict";
function myFunction() {
  return this name;
}
```

Returns "undefined"

5. In Event Handlers

```
<button onclick="this style display='none">
   Click to Remove Me!
</button>
```

Refers to the HTML Element that received that event (button) 4. call() and apply() - Explicit binding

```
var person1 = {
  fullName () {
    return this firstName + " " * this lastName;
}

var person2 = {
  firstName "Kingsley",
  lastName "Ubah"
}

person1 fullName call(person2); // Kingsley Ubah
```

Forces the function - fullName() - to refer to the object - person2.