Scope of “this” in functions

let h=6;

let k=20;

let c=15;

function f1() {

this.h = 9;

this.k = 10;

this.c = 8;

return this;

}

alert(this.c); -> if we print the value of this.c means it will print 15

let h=6;

let k=20;

let c=15;

function f1() {

this.h = 9;

this.k = 10;

this.c = 8; // here i did not return any value

}

let a=f1()

alert(this.c); // here if i print the value of this.c it will print 8 because the value is overwritten by c in the function

let h=6;

let k=20;

let c=15;

function f1() {

let h = 9;

let k = 10;

let c = 8;

return this.h; // here i am returning the value as this.h

}

let a=f1()

alert(this.h); // even though i am returning the value of h it will print 6 only because i declared h in the function as let “it a blocked scope”.

let h=6;

let k=20;

let c=15;

function f1() {

let h = 9;

let k = 10;

let c = 8;

return k;

}

let a=f1()

alert(a.this) // as we know that if we print like this means it should print window but it gives an error because this is used to hold object type we are returning a value but we try print an object type. So it raises an erroe

let h=6;

let k=20;

let c=15;

function f1() {

let h = 9;

let k = 10;

let c = 8;

return this;

}

let a=f1()

alert(this.k) //here it takes the global value which the k was assigned because this by default it takes window in that we are globally assign a value of k=20 so it takes global value even though we are returning the value of this in funtion it takes global value only.

let h=6;

let k=20;

let c=15;

function f1() {

let h = 9;

let k = 10;

let c = 8;

alert(this.c) // here i am checking the value of this.c it is taking global value

return this;

}

let a=f1()

alert(this.k);

Scope of “this” in Object

let obj = {

first\_name: 'krishna',

last\_name: 'teja',

age:45,

full\_name: (): string => {

let val=45

alert("in->" + this.first\_name);

return this.first\_name + this.last\_name;

}

};

alert(obj.full\_name());

In above example i used arrow function here if i executes this program it is thowing the the error that it is undefined value.

If the same example is written in normal function it will return the value perfectly as below example:

function full\_name () {

let first\_name = "krishna";

let last\_name = "teja";

return first\_name + last\_name;

}

let a = full\_name();

alert("out of the functoin" + a);

if i do same that by taking a class, in the class it has arrow function if we use this it works perfectly.

check below example:

class obj {

first\_name= 'krishna';

last\_name='teja';

age= 45;

full\_name= (): string =>

{

return this.first\_name + this.last\_name;

}

}

let val = new obj();

alert(val.full\_name());

full\_name method is defined with an arrow function. But on invocation obj.fullname throws a Type Error, because this.full\_name is evaluated to undefined   
When invoking the method fullname() on the obj object, the context still remains window It happens because the arrow function binds the context lexically with the window object.

The solution is to use a function expression or [shorthand syntax](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Method_definitions) for method definition (available in ECMAScript 6). In such case this is determined by the invocation, but not by the enclosing context. Let's see the fixed version:

let obj = {

first\_name: 'krishna',

last\_name: 'teja',

age:45,

full\_name: function() {

return this.first\_name + this.last\_name;

}

};

let c = obj.full\_name;

let x = c.bind(obj);

alert(x())

Conclusion:

* Basically this is global context untill unless if u define it with this.something it will be global .i.e, window.
* It will change for object, functions as per its context

Bind

this.f\_name = "krishna";

this. age = 23;

let obj = {

f\_name: "teja",

age:24,

person: function ()

{

return this.f\_name+" "+this.age

}

}

var c = obj.person;

let x = c.bind(obj);

alert("out side function->" + this.f\_name+" "+ this.age);

alert("inside the object function->"+x())

class obj

{

name: string;

age: number;

last\_name: string;

add = () =>

{

let name = this.name;

let age = this.age;

}

r = (last\_name:string) =>

{

alert("My name is ->"+this.name+"age is ->"+this.age+"last name is ->"+last\_name)

}

}

let val = new obj();

val.name = "krishna";

val.age = 24;

val.add.bind(val.name,val.age);

val.r("teja")