

Scope, IFFE, Dynamic Scope, THIS

Samuil Gospodinov



Today's plan

- block scope review
- IIFE
- Dynamic Scope
- Hoisting

Block Operators

- if { } else { }
- for () { }
- switch () { }
- while () { }

No block scope

```
1 var x = 1;
2 {
3     var x = 2;
4 }
5 console.log(x); // logs 2
```

IIFE Pattern

```
1 var ime = "gosho";  
2  
3 (function(){  
4  
5     var ime = "pesho";  
6     console.log(ime); //pesho  
7  
8 })();  
9  
10 console.log(ime); //gosho
```

IIFE - variations

```
1 var ime = "gosho";  
2  
3 (function(drugoIme){  
4  
5     var ime = drugoIme;  
6     console.log(ime); //gosho  
7  
8 })(ime);  
9  
10 console.log(ime); //gosho
```

Dynamic Scope

There is no such animal in JavaScript

Theoretical Example

```
1 function gosho() {  
2     console.log(ime); //dynamic  
3 }  
4  
5 function pesho() {  
6     var ime = "pesho";  
7     gosho();  
8 }  
9  
10 pesho();
```


Hoisting

```
1 ime;    //??  
2 pesho;  //??  
3  
4 var ime = pesho;  
5 var pesho = "pesho";  
6  
7 pesho;   //pesho  
8 ime;     //??
```

Hoisting - variables

```
1 var ime;  
2 var pesho;  
3  
4 ime;    //??  
5 pesho;   //??  
6  
7 ime = pesho;  
8 pesho = "pesho";  
9  
10 pesho;   //pesho  
11 ime;     //??
```

Hoisting functions

```
1 var pesho = ivan();
2 var gosho = mitko();
3
4 pesho;    //??
5 gosho;    //??
6
7 function ivan () {
8     return gosho;
9 }
10
11 var mitko = function () {
12     return ivan();
13 }
```

```
1 function ivan () {  
2     return gosho;  
3 }  
4  
5 var pesho;  
6 var gosho;  
7 var mitko;  
8  
9 pesho = ivan();  
10 gosho = mitko();  
11  
12 pesho;    //??  
13 gosho;    //??  
14  
15 var mitko = function () {  
16     return ivan();  
17 }
```

Hoisting - Functions comes first

```
1 ivan(); //Ivan
2
3 var ivan = "Gosho";
4
5 function ivan () {
6     console.log("Pesho");
7 }
8
9 function ivan () {
10     console.log("Ivan");
11 }
```

Hoisting

```
1 pesho(1);    //??
2
3 function pesho(chislo) {
4     if(chislo > 20) {
5         return chislo;
6     }
7
8     return ivan(chislo + 2);
9 }
10
11 function ivan(chislo) {
12     return gosho(chislo) + 1;
13 }
14
15 function gosho(chislo) {
16     return pesho(chislo * 2);
17 }
```

THIS Keyword

Every function, **while executing** has a reference to its current execution context, called **this**.

THIS - execution context

- where a function is called
- when a function is called
- how a function is called

THIS - implicit & default binding

```
1 function kazvamSe () {  
2     console.log(this.ime);  
3 }  
4  
5 var ime = "pesho";  
6 var ivan = {  
7     ime: "ivan",  
8     kazvamSe: kazvamSe  
9 };  
10 var gosho = {  
11     ime: "gosho",  
12     kazvamSe: kazvamSe  
13 };  
14  
15 kazvamSe();           //pesho  
16 ivan.kazvamSe();      //ivan  
17 gosho.kazvamSe();     //gosho
```

THIS - implicit & default binding

```
1 var ivan = {
2     ime: "ivan",
3     kazvamSe: function () {
4         console.log(this.ime);
5     }
6 };
7
8 var ime = "pesho";
9 var gosho = {
10     ime: "gosho",
11     kazvamSe: iva.kazvamSe
12 };
13
14 var kazvamSe = ivan.kazvamSe;
15
16 kazvamSe();           //pesho
17 ivan.kazvamSe();      //ivan
18 gosho.kazvamSe();     //gosho
```

Binding confusion

```
1 function pesho () {  
2     var ime = "pesho";  
3     ivan();  
4 }  
5  
6 function ivan () {  
7     console.log(this.ime);  
8 }  
9  
10 var ime = "ivan";  
11 pesho();
```

Binding confusion

```
1 function pesho () {  
2     var ime = "pesho";  
3     this.ivan = ivan;  
4     this.ivan();  
5 }  
6  
7 function ivan () {  
8     console.log(this.ime);  
9 }  
10  
11 var ime = "ivan";  
12 pesho();
```

Explicit binding

```
1 function gosho () {  
2     console.log(this.ime);  
3 }  
4  
5 var ime = "gosho";  
6 var pesho = {ime: "pesho"};  
7  
8 gosho();           //gosho  
9 gosho.call(pesho); //pesho
```

Explicit binding

```
1 function koiSumAz () {  
2     console.log(this.ime);  
3 }  
4  
5 var ime = "gosho";  
6 var ivan = {ime: "ivan"};  
7 var pesho = {ime: "pesho"};  
8  
9 var originalKoiSumAz = koiSumAz;  
10 koiSumAz = function () {  
11     originalKoiSumAz.call(ivan);  
12 }  
13  
14 koiSumAz();           //ivan  
15 koiSumAz.call(ivan);  //ivan  
16 koiSumAz.call(pesho); //ivan
```

The Bind Method

```
1 function printiraiNesto (ivan, pesho) {  
2     console.log(this.ime + ' ' + ivan + ' ' + pesho);  
3 };  
4  
5 var gosho = {ime: 'gosho'};  
6 printiraiNesto = printiraiNesto.bind(gosho, 'ivan'); //ES5  
7  
8 printiraiNesto('pesho');
```

The New Keyword

```
1 function gosho () {  
2     this.ime ="gosho";  
3     console.log(this.godini + ' ' + ime);  
4 }  
5  
6 var godini = 19;  
7 var chovek = new gosho();  
8 console.log(chovek.ime);
```


THIS - 4 rules

1. Is the **NEW** keyword used in the function call?
2. Was it call with “**.call**” or “**.apply**” specifying an explicit **THIS**
3. Was it called from a containing/owning object (**Implicit**)
4. Default: global object (except strict mode)

The End