Scope, IFFE, Dynamic Scope, THIS

Samuil Gospodinov

CODICAMP

EDUCATION REINVENTED

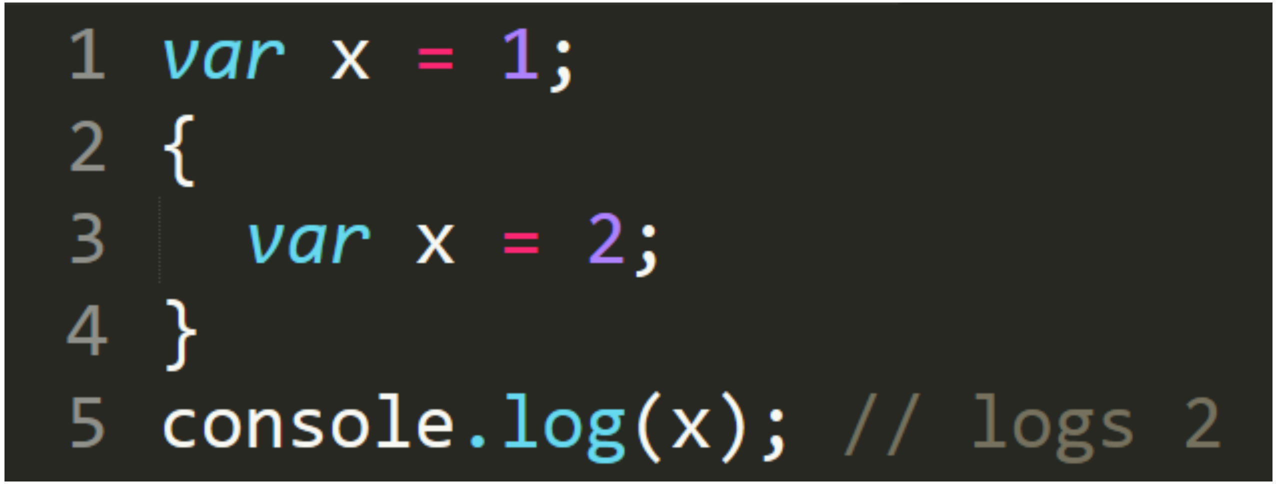
Today’s plan

- block scope review
- IIFE
- Dynamic Scope
- Hoisting

Block Operators

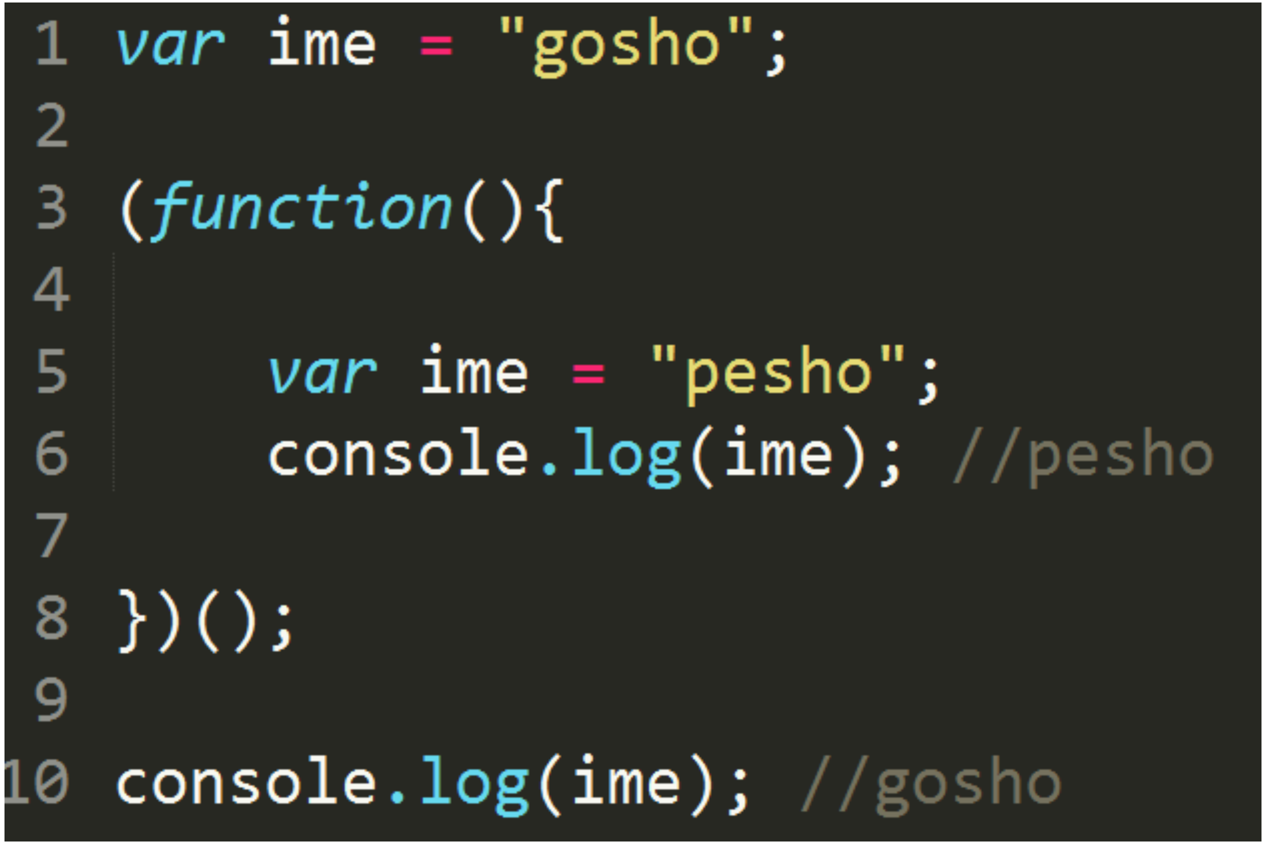
- if { } else { }
- for () { }
- switch () {}
- while () {}

No block scope



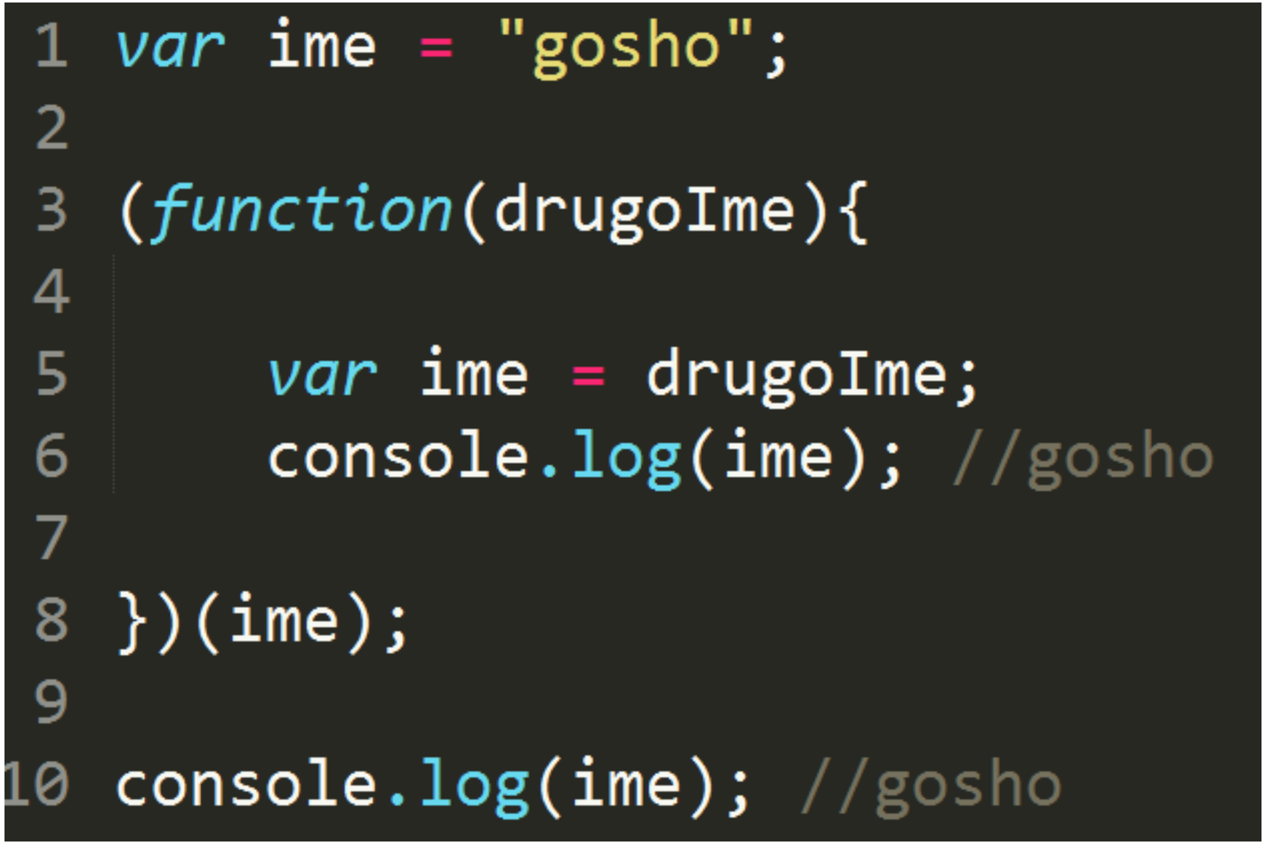
```
var x = 1;
{
  var x = 2;
}
console.log(x); // logs 2
```

IIFE Pattern



```
var ime = "gosho";

(function(){

    var ime = "pesho";
    console.log(ime); //pesho

})();

console.log(ime); //gosho
```

IIFE - variations

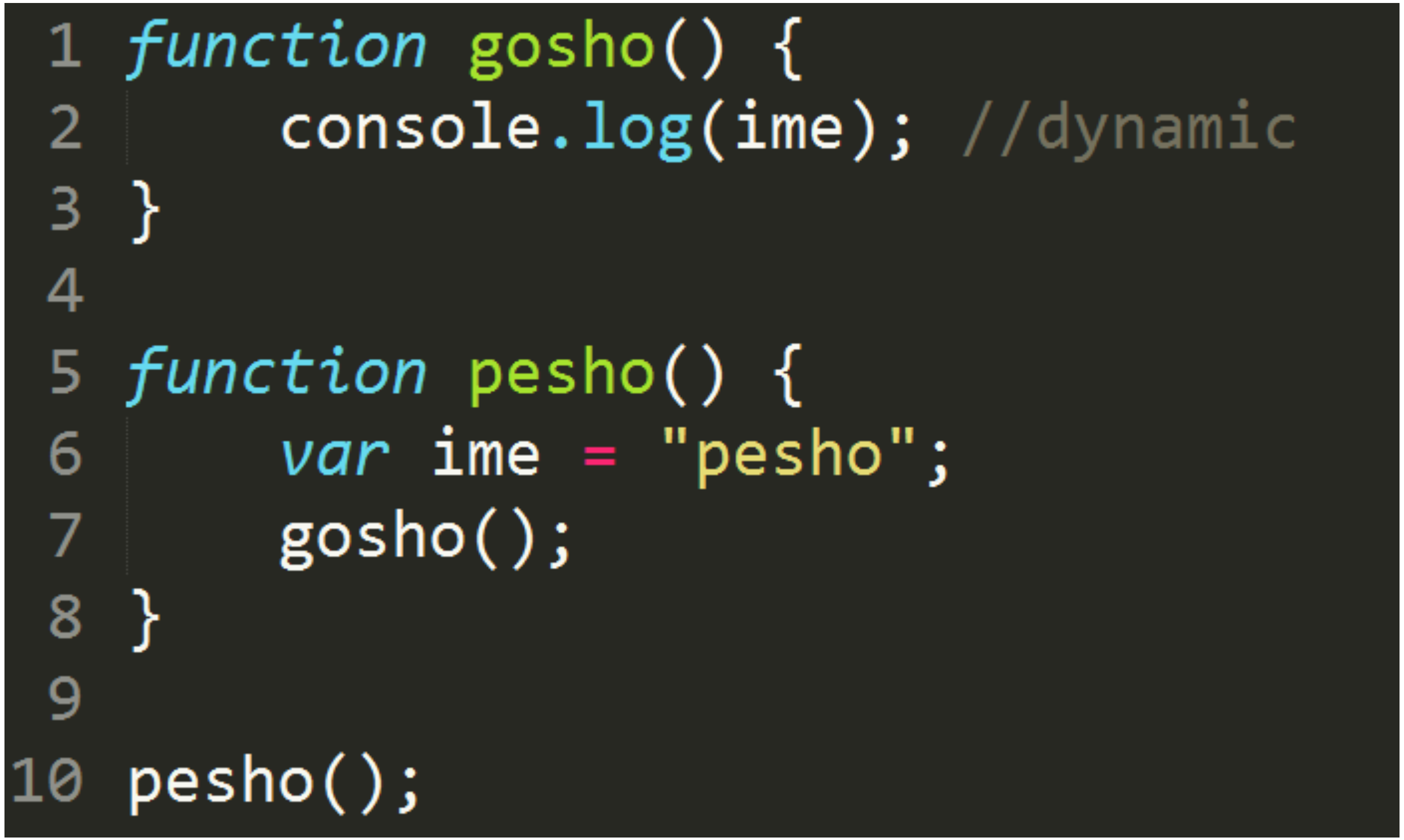


```
var ime = "gosho";

(function(drugoIme){

    var ime = drugoIme;
    console.log(ime); //gosho

})(ime);

console.log(ime); //gosho
```

Dynamic Scope

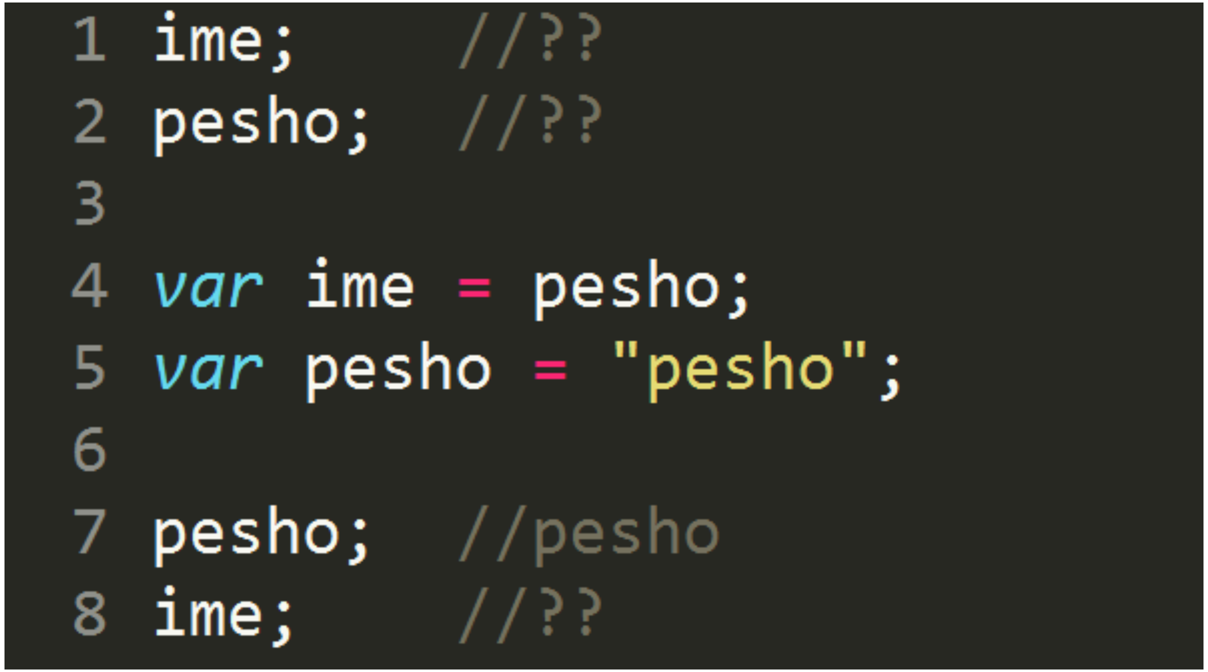
There is no such animal in JavaScript

Theoretical Example



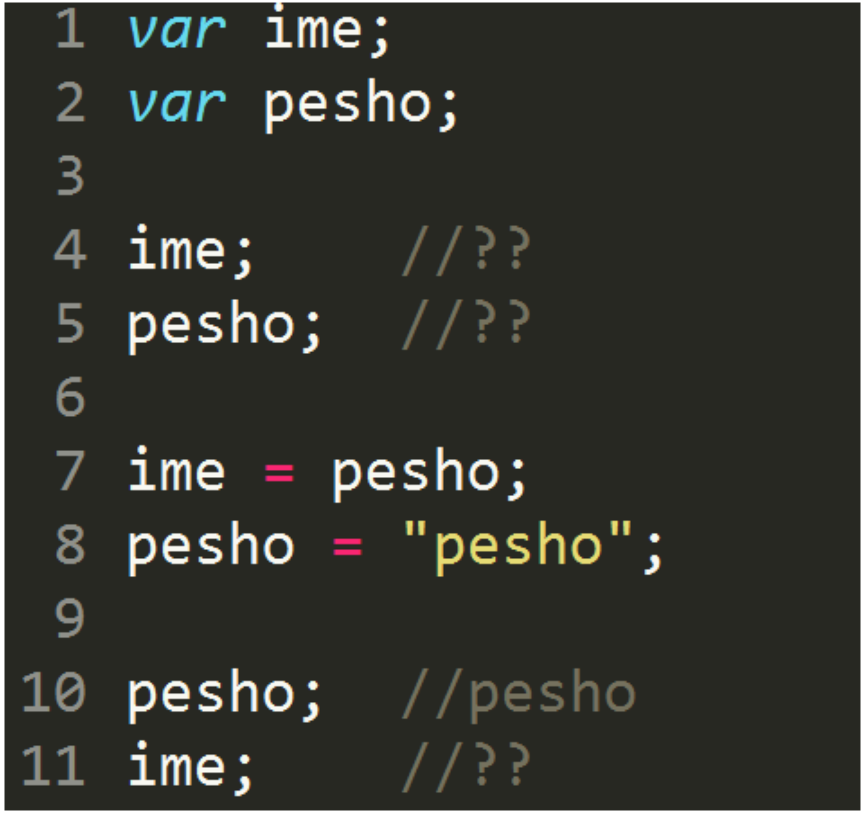
```javascript
function gosho() {
    console.log(ime); //dynamic
}

function pesho() {
    var ime = "pesho";
    gosho();
}

pesho();
```

Hoisting



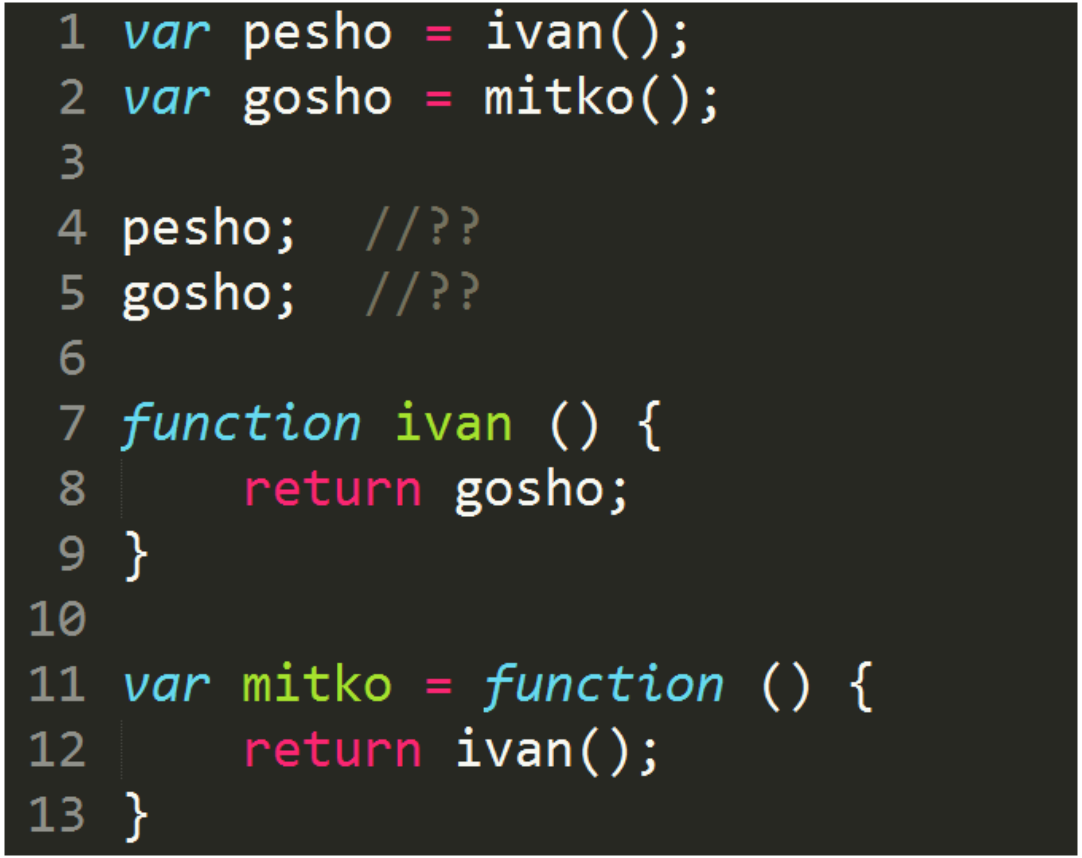
```
ime;     //??
pesho;   //??

var ime = pesho;
var pesho = "pesho";

pesho;   //pesho
ime;     //??
```

Hoisting - variables

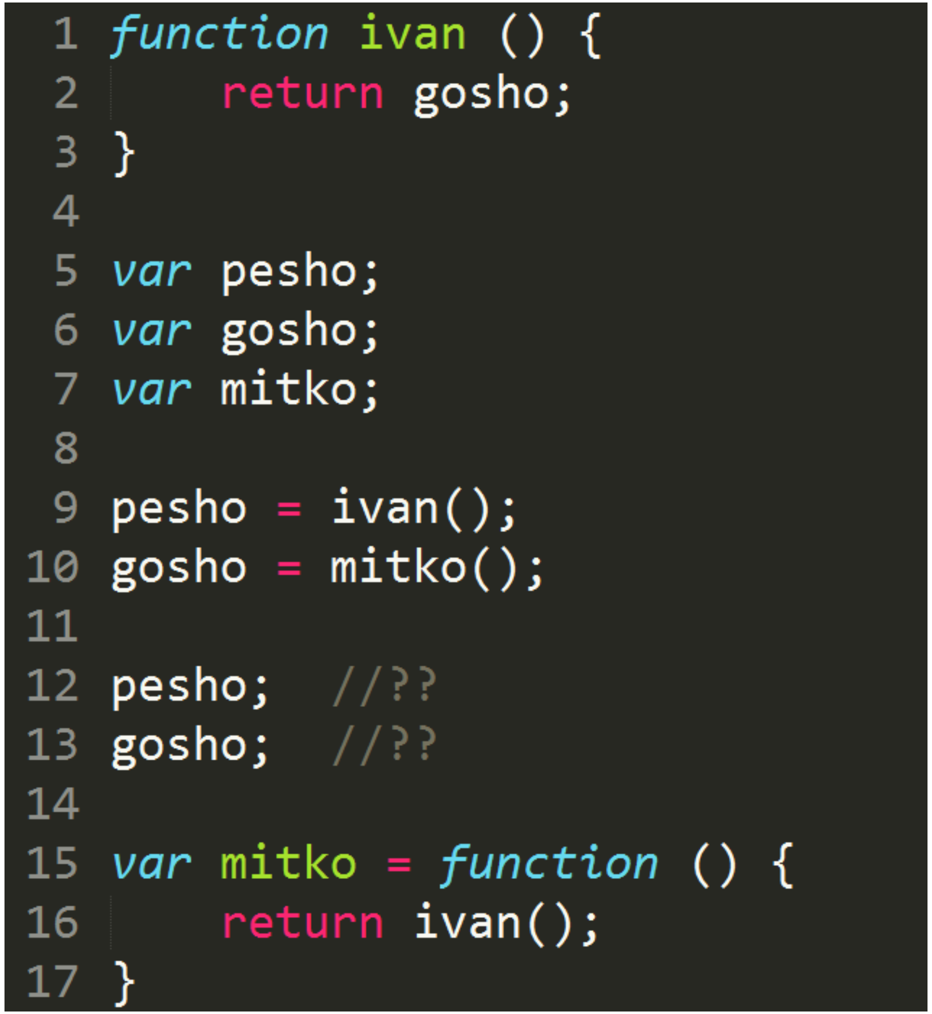


```
var ime;
var pesho;

ime;      //??
pesho;    //??

ime = pesho;
pesho = "pesho";

pesho;    //pesho
ime;      //??
```

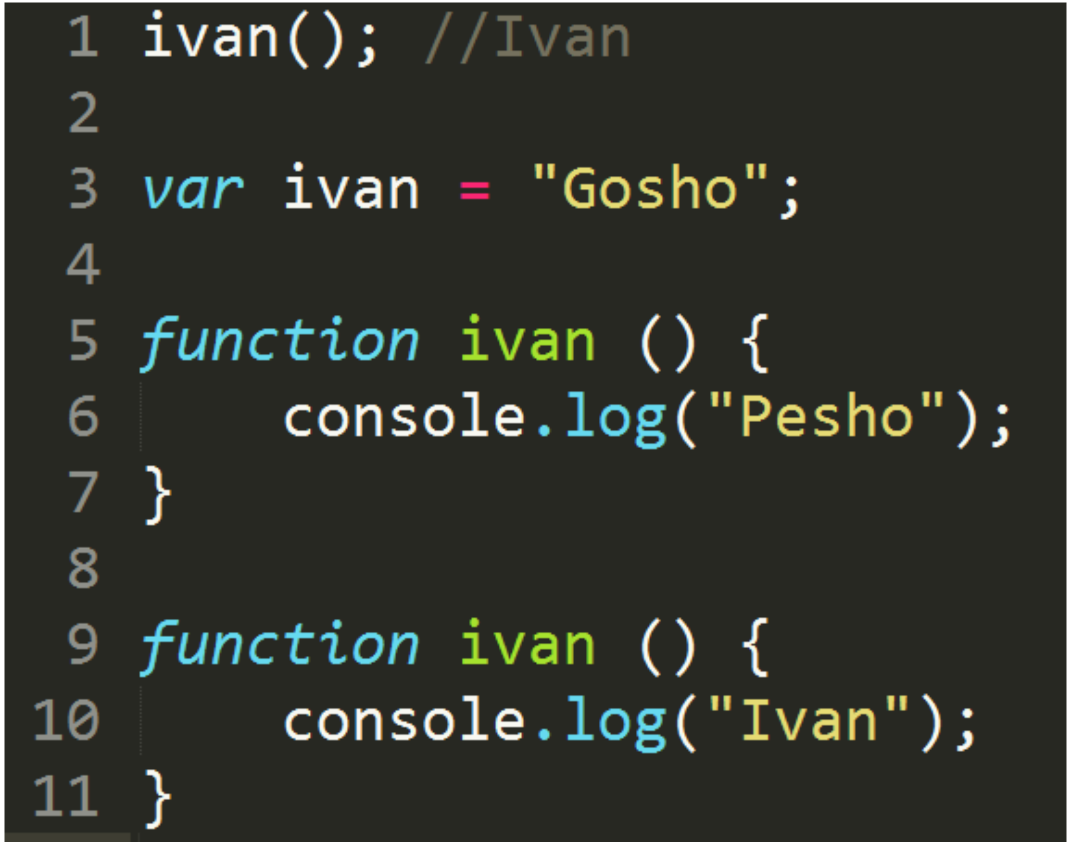
Hoisting functions



```
var pesho = ivan();
var gosho = mitko();

pesho;  //??
gosho;  //??

function ivan () {
    return gosho;
}

var mitko = function () {
    return ivan();
}
```

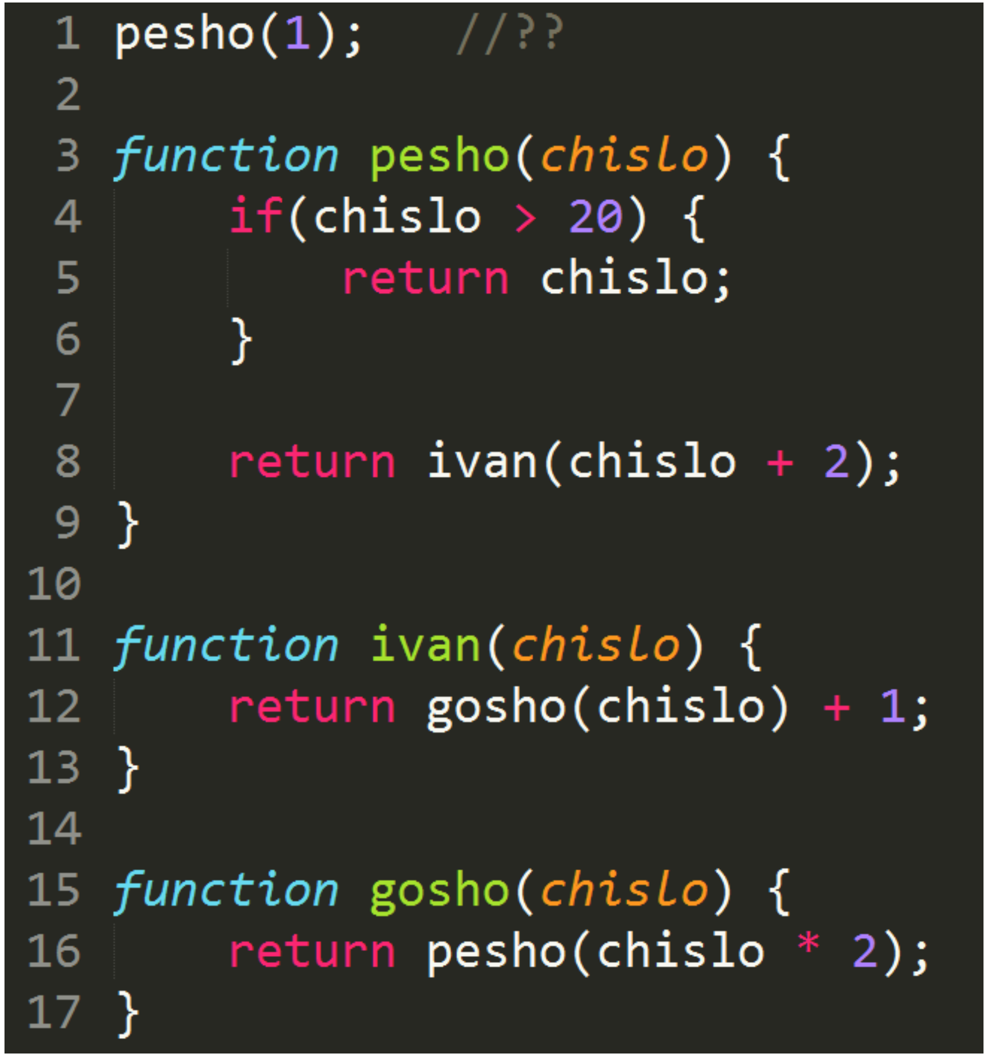

```
function ivan () {
    return gosho;
}

var pesho;
var gosho;
var mitko;

pesho = ivan();
gosho = mitko();

pesho;  //??
gosho;  //??

var mitko = function () {
    return ivan();
}
```

Hoisting - Functions comes first



```
ivan(); //Ivan

var ivan = "Gosho";

function ivan () {
    console.log("Pesho");
}

function ivan () {
    console.log("Ivan");
}
```

Hoisting



```
pesho(1);   //??

function pesho(chislo) {
    if(chislo > 20) {
        return chislo;
    }

    return ivan(chislo + 2);
}

function ivan(chislo) {
    return gosho(chislo) + 1;
}

function gosho(chislo) {
    return pesho(chislo * 2);
}
```

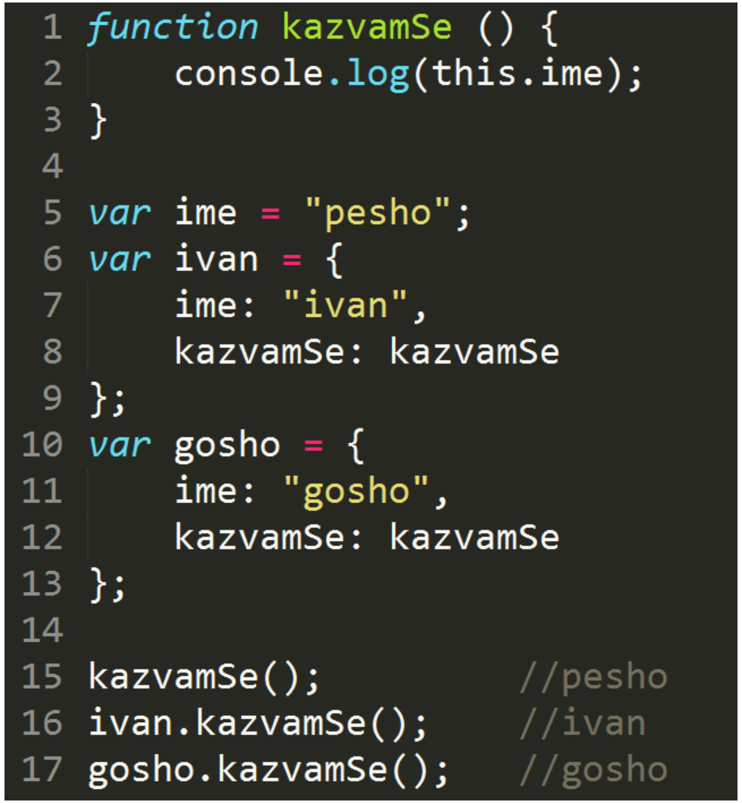
THIS Keyword

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

THIS - execution context

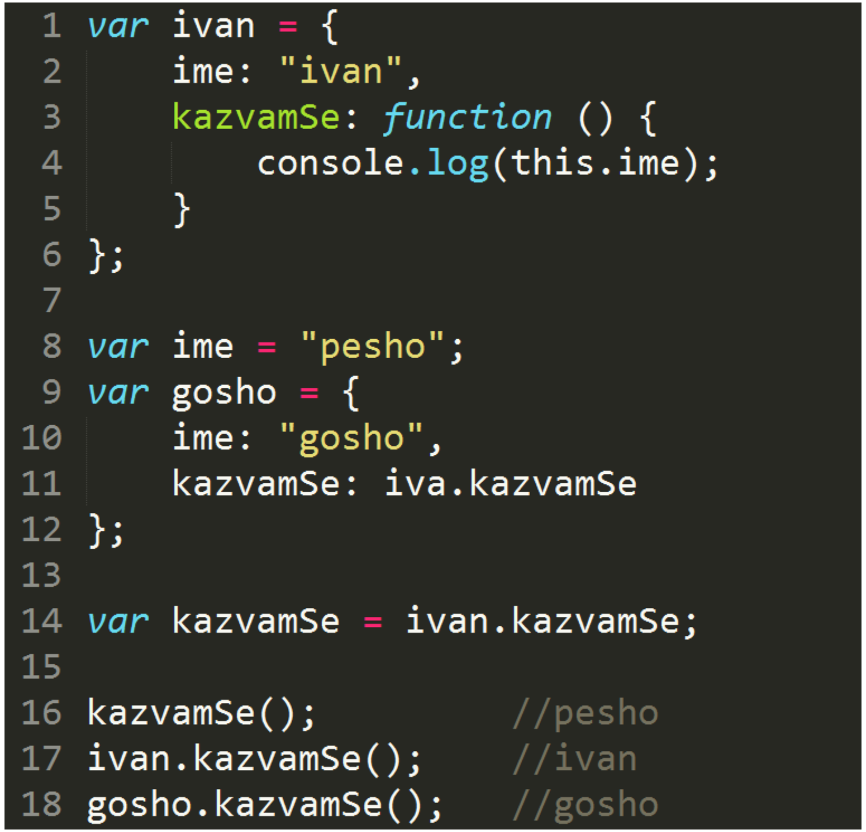
- where a function is called
- when a function is called
- how a function is called

THIS - implicit & default binding



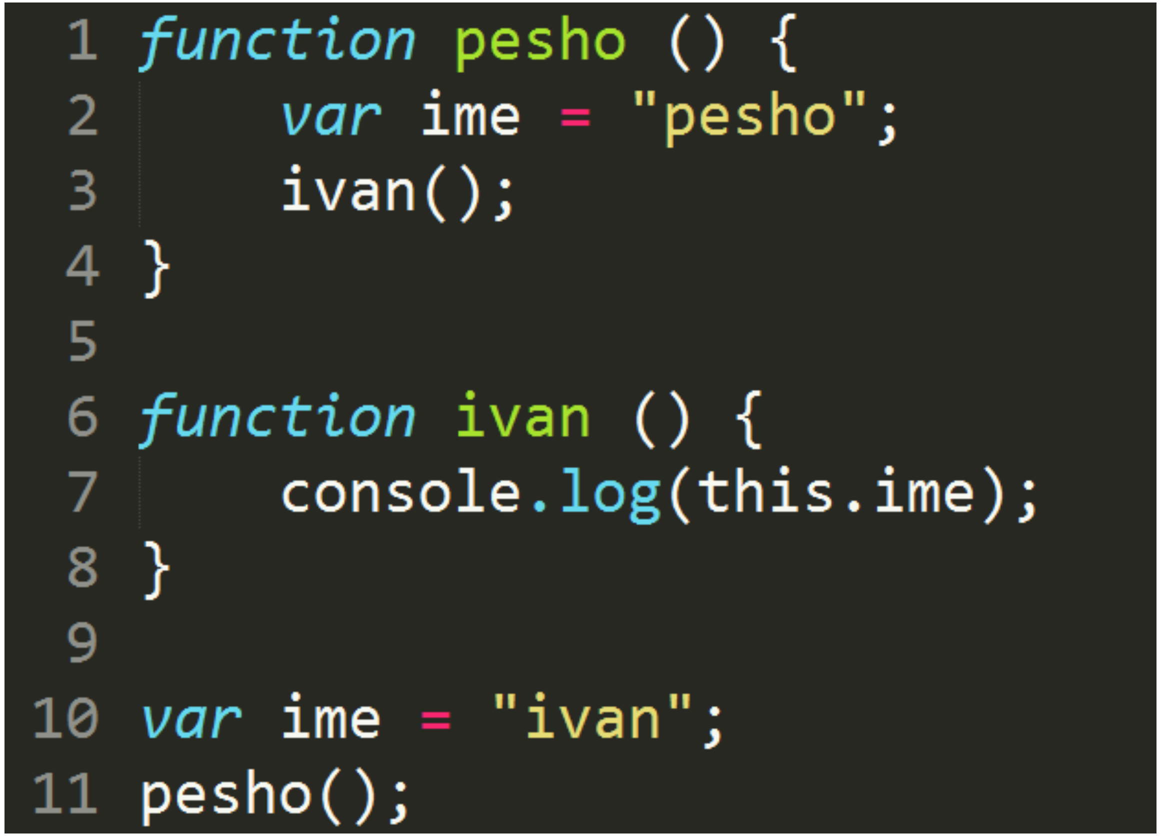
```
function kazvamSe () {
    console.log(this.ime);
}

var ime = "pesho";
var ivan = {
    ime: "ivan",
    kazvamSe: kazvamSe
};
var gosho = {
    ime: "gosho",
    kazvamSe: kazvamSe
};

kazvamSe();             //pesho
ivan.kazvamSe();        //ivan
gosho.kazvamSe();       //gosho
```

THIS - implicit & default binding



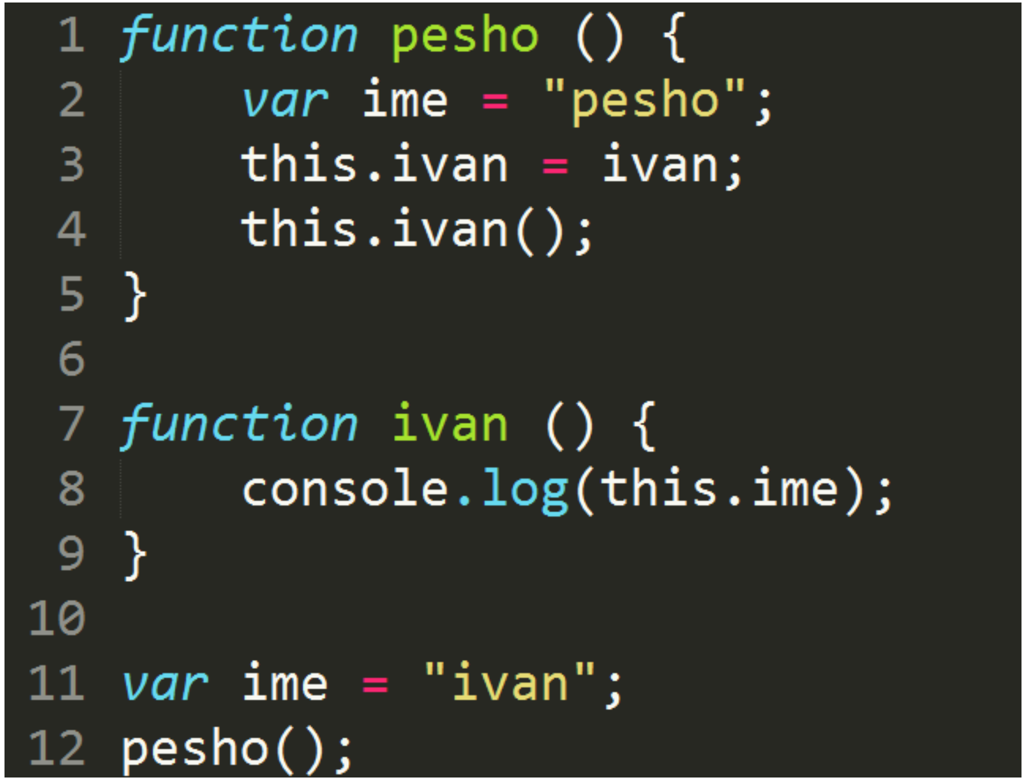
```javascript
var ivan = {
    ime: "ivan",
    kazvamSe: function () {
        console.log(this.ime);
    }
};

var ime = "pesho";
var gosho = {
    ime: "gosho",
    kazvamSe: iva.kazvamSe
};

var kazvamSe = ivan.kazvamSe;

kazvamSe();         //pesho
ivan.kazvamSe();    //ivan
gosho.kazvamSe();   //gosho
```

Binding confusion



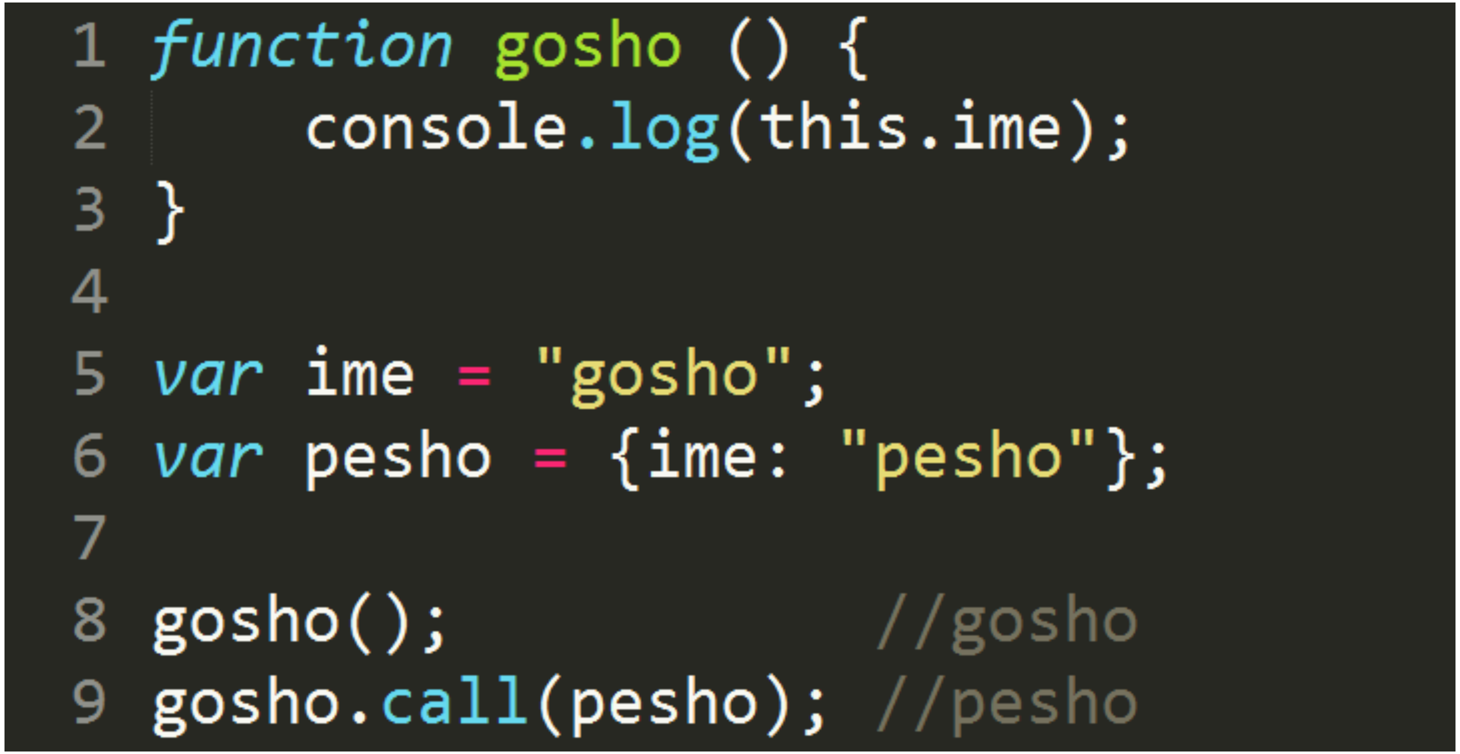
```
function pesho () {
    var ime = "pesho";
    ivan();
}

function ivan () {
    console.log(this.ime);
}

var ime = "ivan";
pesho();
```

Binding confusion



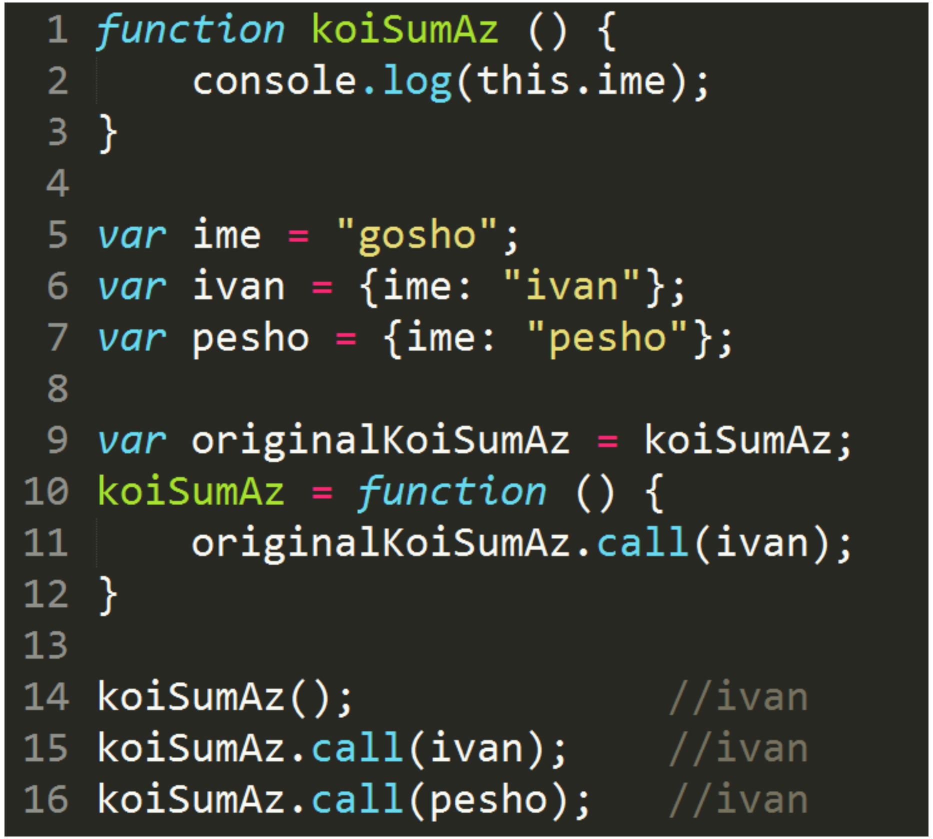
```
function pesho () {
    var ime = "pesho";
    this.ivan = ivan;
    this.ivan();
}

function ivan () {
    console.log(this.ime);
}

var ime = "ivan";
pesho();
```

Explicit binding



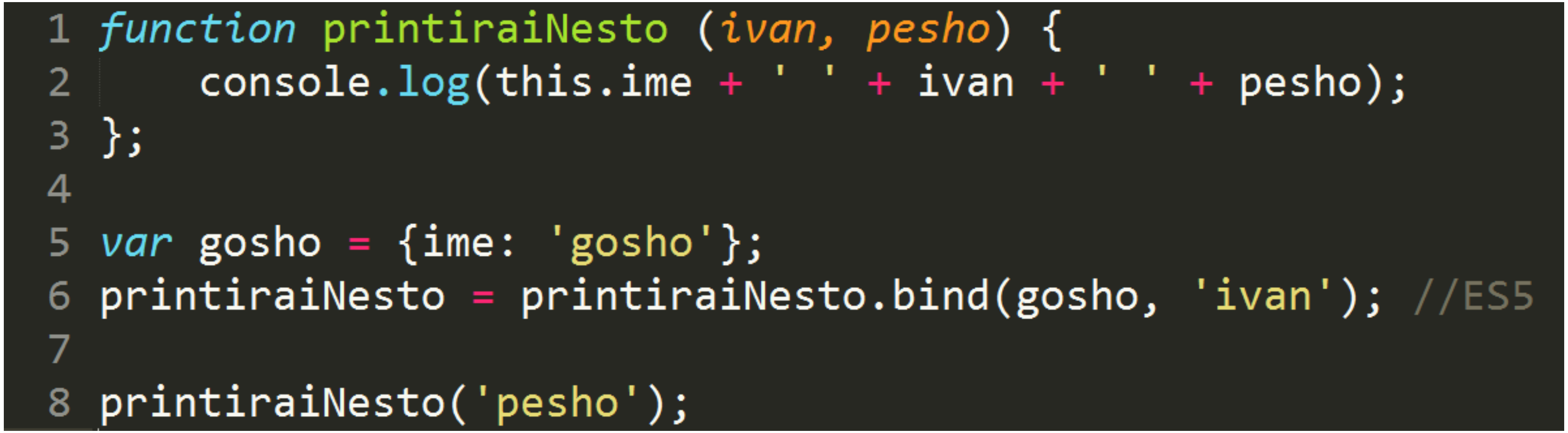
```
function gosho () {
    console.log(this.ime);
}

var ime = "gosho";
var pesho = {ime: "pesho"};

gosho();              //gosho
gosho.call(pesho); //pesho
```

Explicit binding



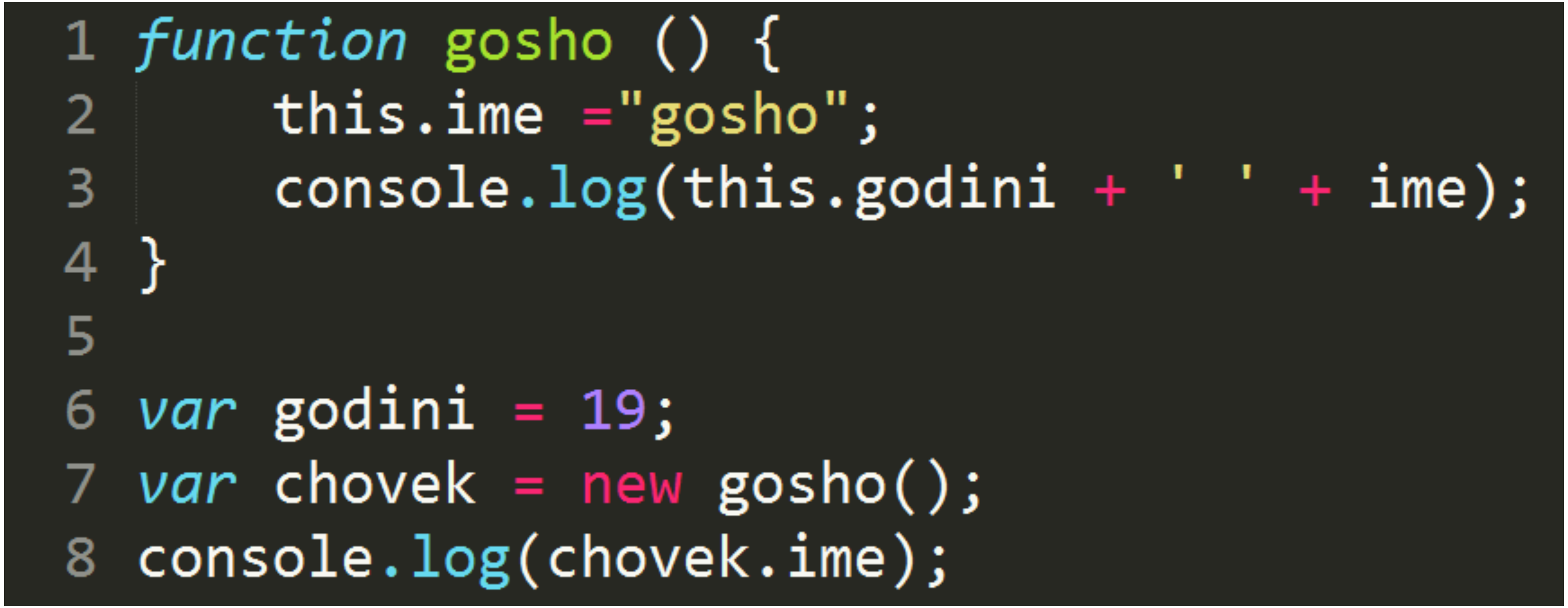
```
function koiSumAz () {
    console.log(this.ime);
}

var ime = "gosho";
var ivan = {ime: "ivan"};
var pesho = {ime: "pesho"};

var originalKoiSumAz = koiSumAz;
koiSumAz = function () {
    originalKoiSumAz.call(ivan);
}

koiSumAz();               //ivan
koiSumAz.call(ivan);      //ivan
koiSumAz.call(pesho);     //ivan
```

The Bind Method



```
function printiraiNesto (ivan, pesho) {
    console.log(this.ime + ' ' + ivan + ' ' + pesho);
};

var gosho = {ime: 'gosho'};
printiraiNesto = printiraiNesto.bind(gosho, 'ivan'); //ES5

printiraiNesto('pesho');
```

The New Keyword



```
function gosho () {
        this.ime ="gosho";
        console.log(this.godini + ' ' + ime);
}

var godini = 19;
var chovek = new gosho();
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