OLOO style of coding

Samuil Gospodinov

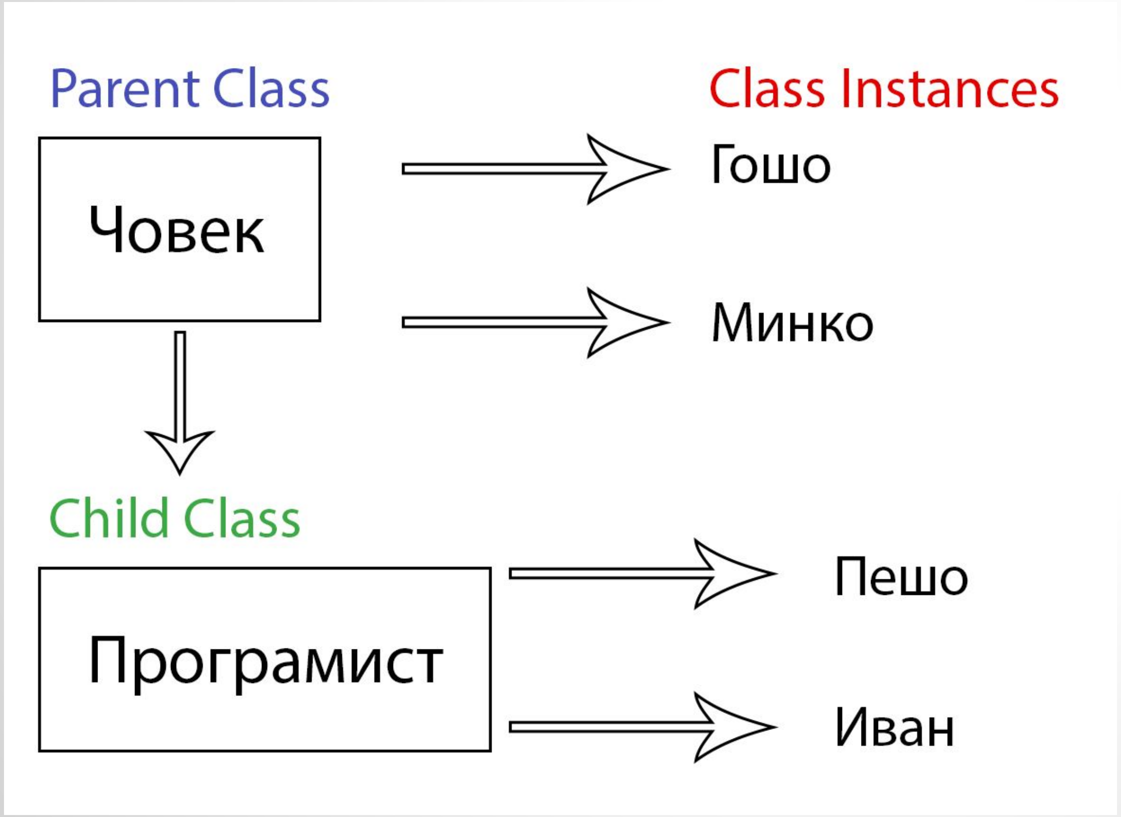
CODICAMP

EDUCATION REINVENTED

Today’s Plan

- Inheritance
- Prototypal Inheritance
- Behaviour Delegation Pattern
- OOLO
- Object Pooling Design Pattern
- Quiz

Classical Inheritance - COPY DOWN

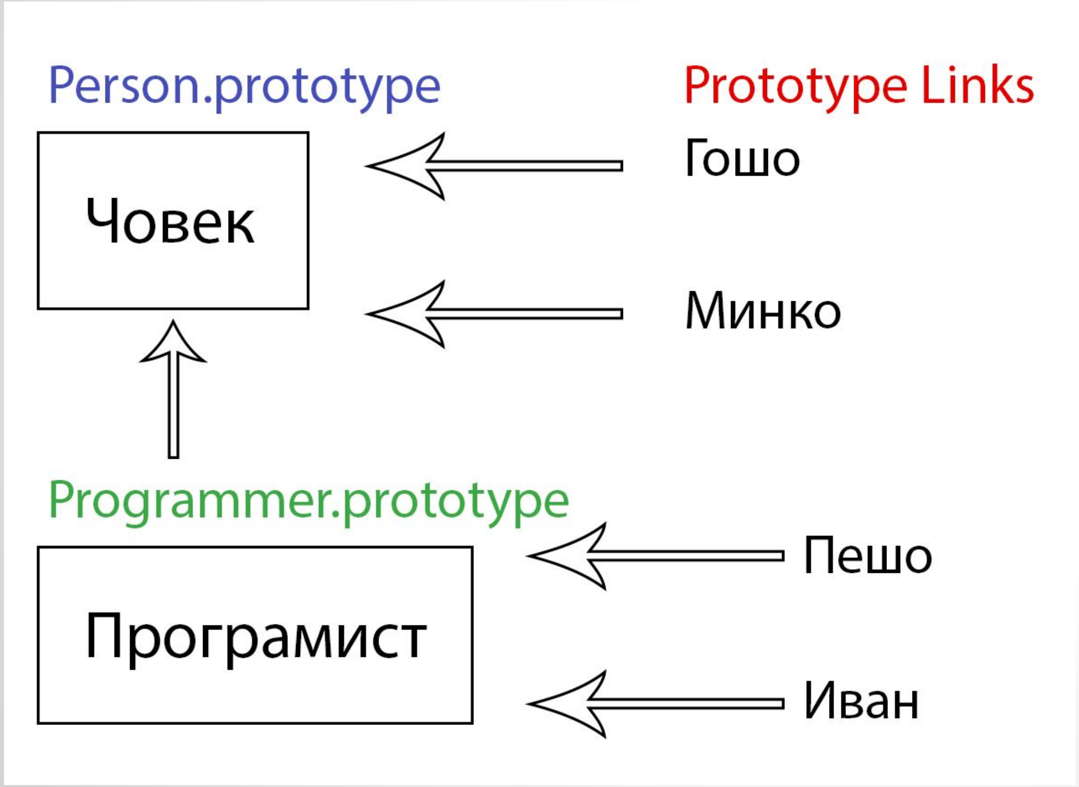


Prototypal Inheritance

It’s not an orange, it’s an orange apple…

Inheritance means COPY

Prototypal ~~Inheritance~~ delegation



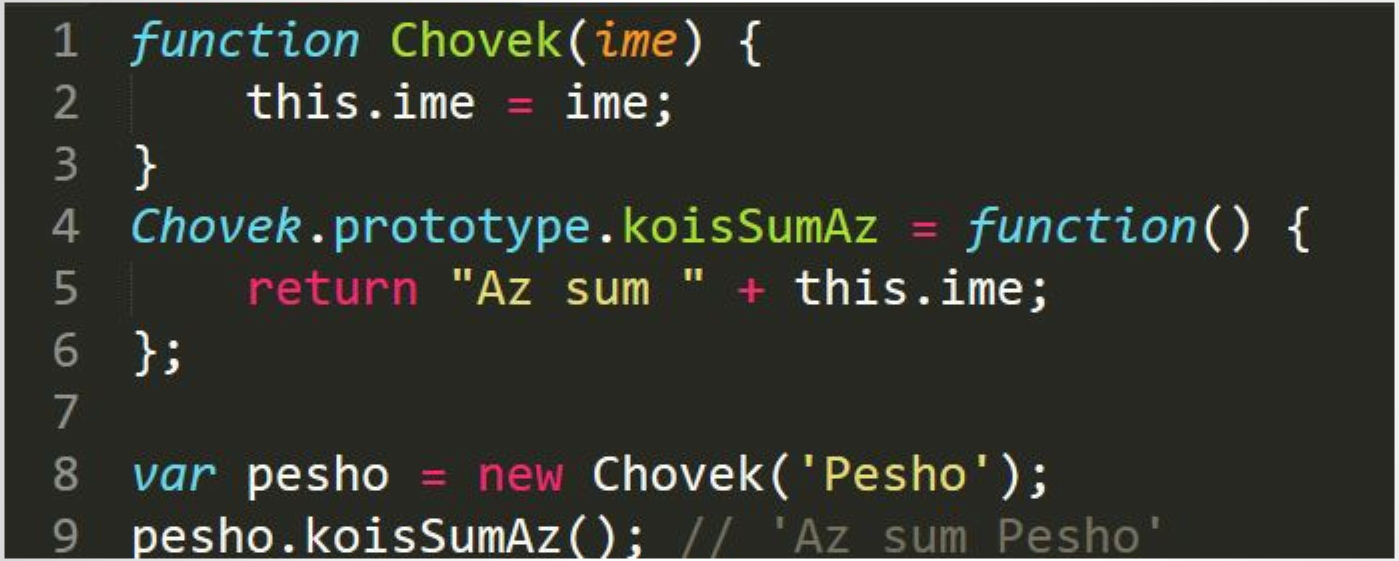
Prototypal ~~Inheritance~~ delegation

JS has ~~inheritance~~

nope, JS has Behaviour delegation

- It’s a design pattern!

Prototypal delegation



```
function Chovek(ime) {
    this.ime = ime;
}
Chovek.prototype.koisSumAz = function() {
    return "Az sum " + this.ime;
};

var pesho = new Chovek('Pesho');
pesho.koisSumAz(); // 'Az sum Pesho'
```

Prototypal delegation

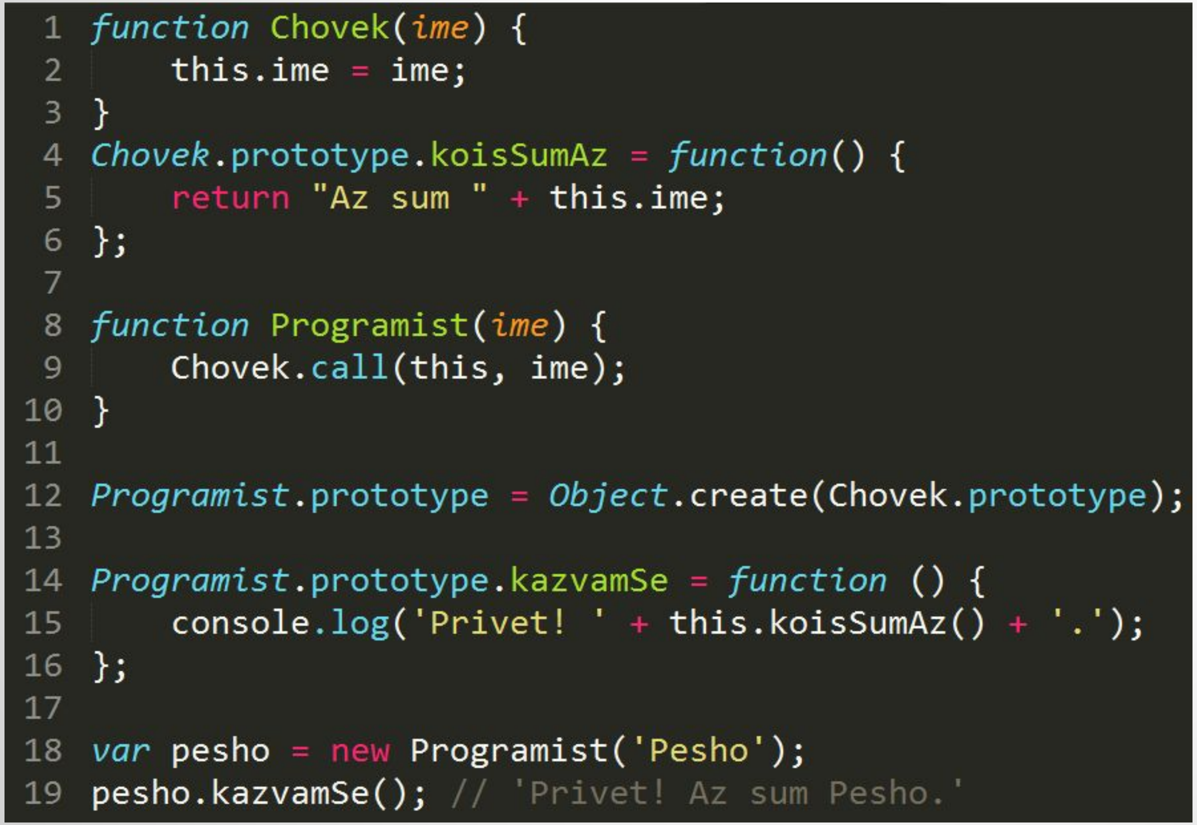
Keep it simple stupid!

KISS

Prototypal delegation - OLOO

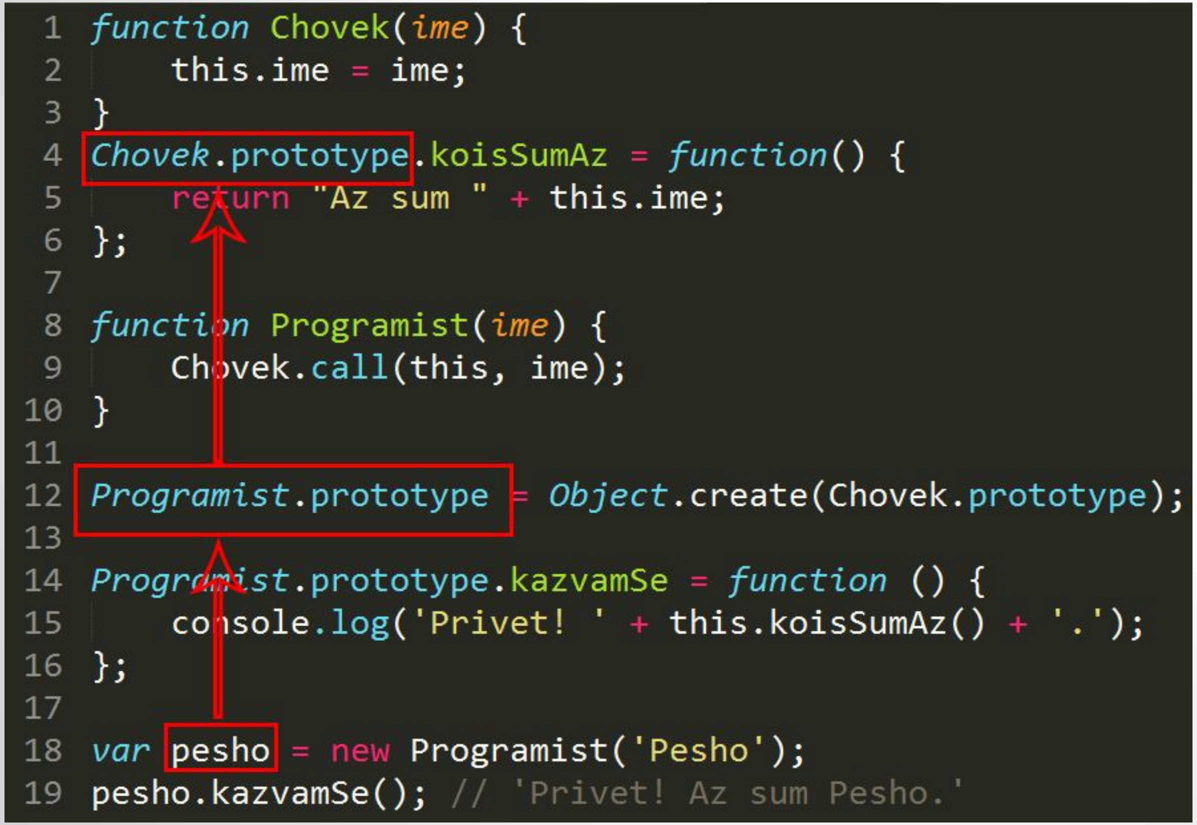
Objects Linked to Other Object

Prototypal delegation - OLOO



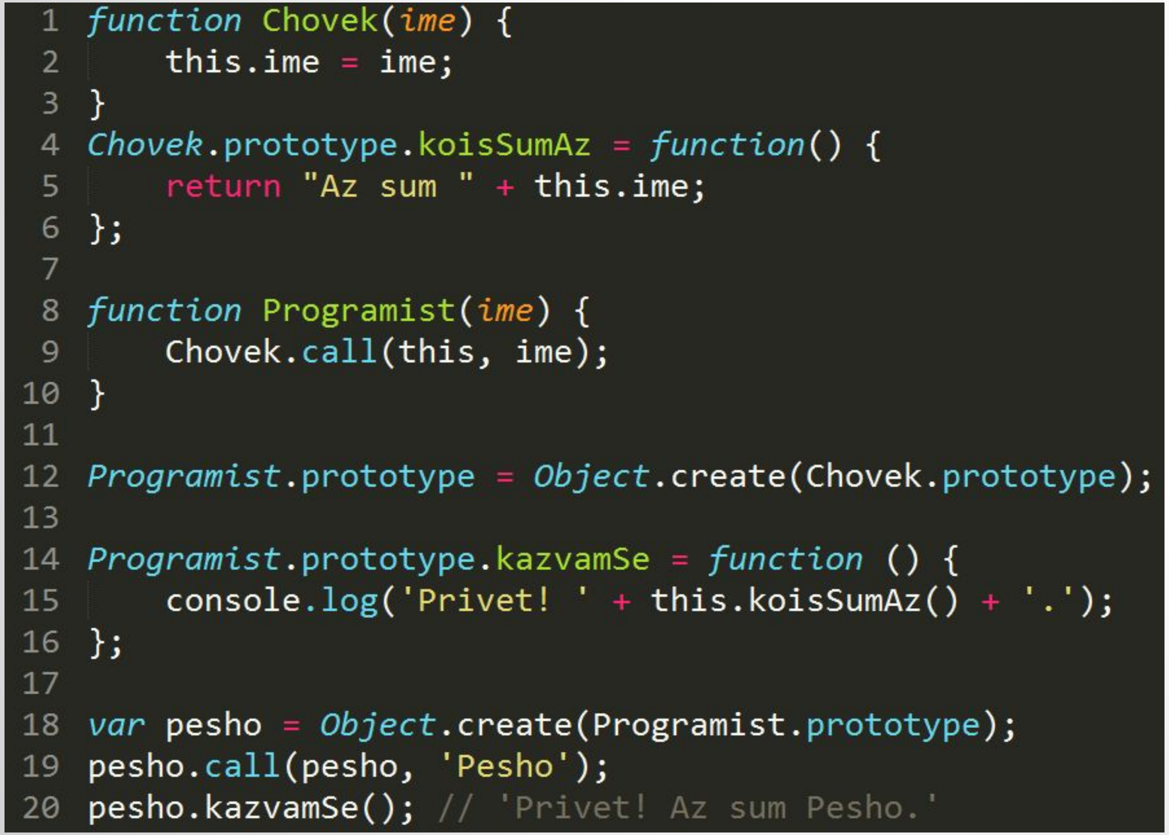
```javascript
function Chovek(ime) {
    this.ime = ime;
}
Chovek.prototype.koisSumAz = function() {
    return "Az sum " + this.ime;
};

function Programist(ime) {
    Chovek.call(this, ime);
}

Programist.prototype = Object.create(Chovek.prototype);

Programist.prototype.kazvamSe = function () {
    console.log('Privet! ' + this.koisSumAz() + '.');
};

var pesho = new Programist('Pesho');
pesho.kazvamSe(); // 'Privet! Az sum Pesho.'
```

Prototypal delegation - OLOO



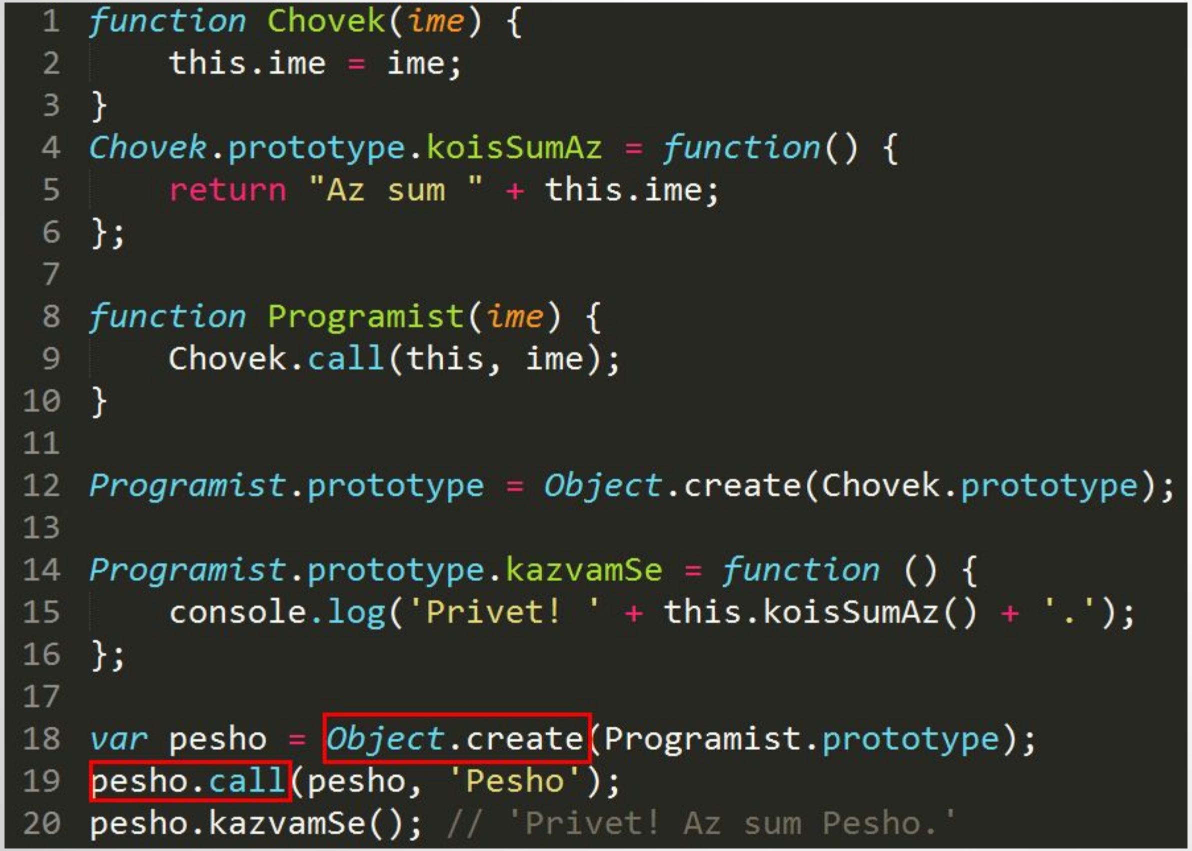
```
function Chovek(ime) {
    this.ime = ime;
}
Chovek.prototype.koisSumAz = function() {
    return "Az sum " + this.ime;
};

function Programist(ime) {
    Chovek.call(this, ime);
}

Programist.prototype = Object.create(Chovek.prototype);

Programist.prototype.kazvamSe = function () {
    console.log('Privet! ' + this.koisSumAz() + '.');
};

var pesho = new Programist('Pesho');
pesho.kazvamSe(); // 'Privet! Az sum Pesho.'
```

Prototypal delegation - OLOO



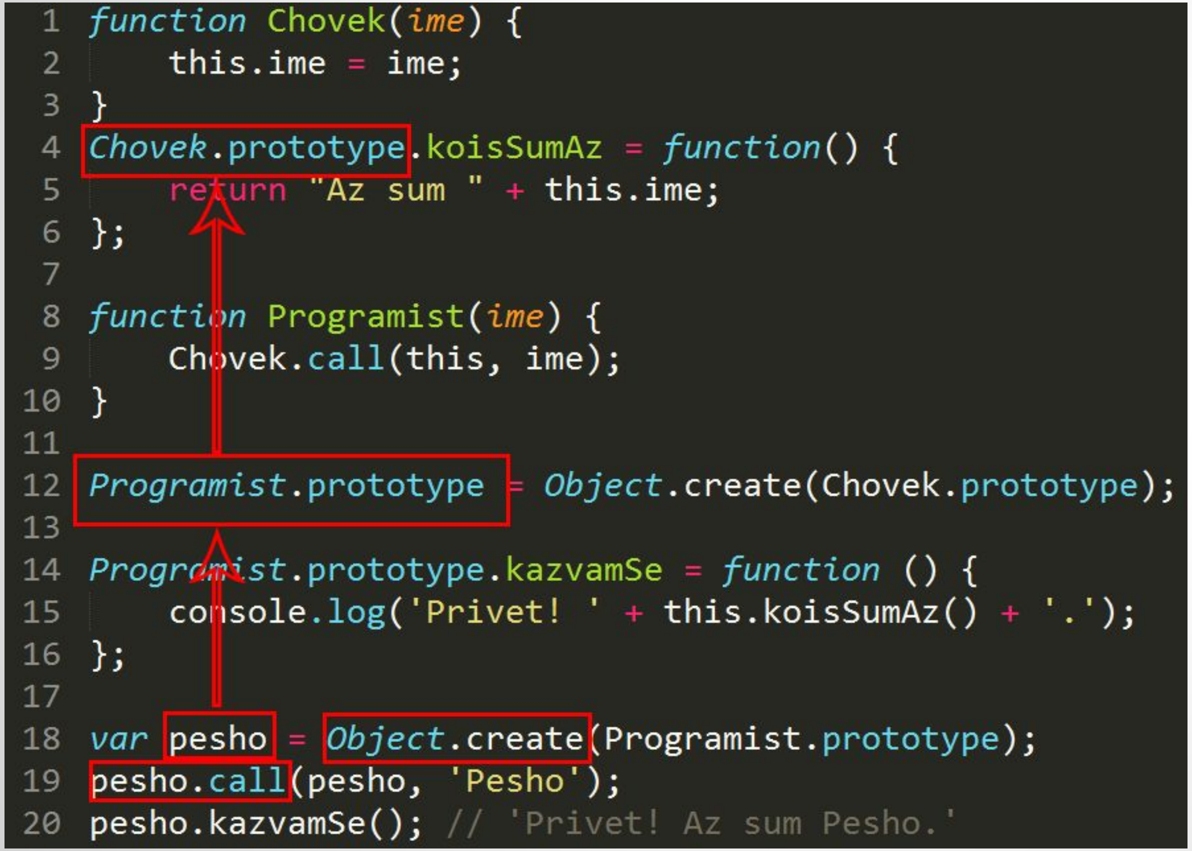
```javascript
function Chovek(ime) {
    this.ime = ime;
}
Chovek.prototype.koisSumAz = function() {
    return "Az sum " + this.ime;
};

function Programist(ime) {
    Chovek.call(this, ime);
}

Programist.prototype = Object.create(Chovek.prototype);

Programist.prototype.kazvamSe = function () {
    console.log('Privet! ' + this.koisSumAz() + '.');
};

var pesho = Object.create(Programist.prototype);
pesho.call(pesho, 'Pesho');
pesho.kazvamSe(); // 'Privet! Az sum Pesho.'
```

Prototypal delegation - OLOO



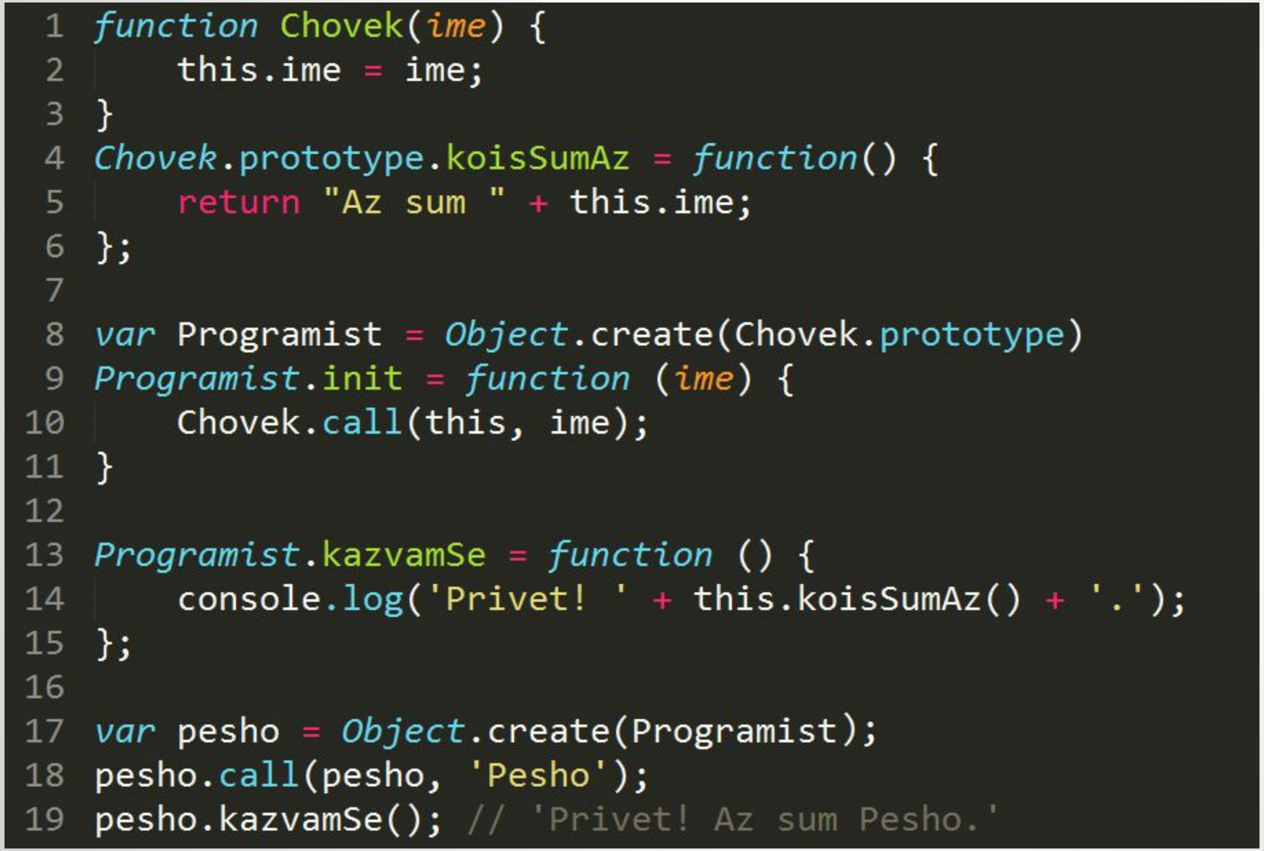
```
function Chovek(ime) {
    this.ime = ime;
}
Chovek.prototype.koisSumAz = function() {
    return "Az sum " + this.ime;
};

function Programist(ime) {
    Chovek.call(this, ime);
}

Programist.prototype = Object.create(Chovek.prototype);

Programist.prototype.kazvamSe = function () {
    console.log('Privet! ' + this.koisSumAz() + '.');
};

var pesho = Object.create(Programist.prototype);
pesho.call(pesho, 'Pesho');
pesho.kazvamSe(); // 'Privet! Az sum Pesho.'
```

Prototypal delegation - OLOO



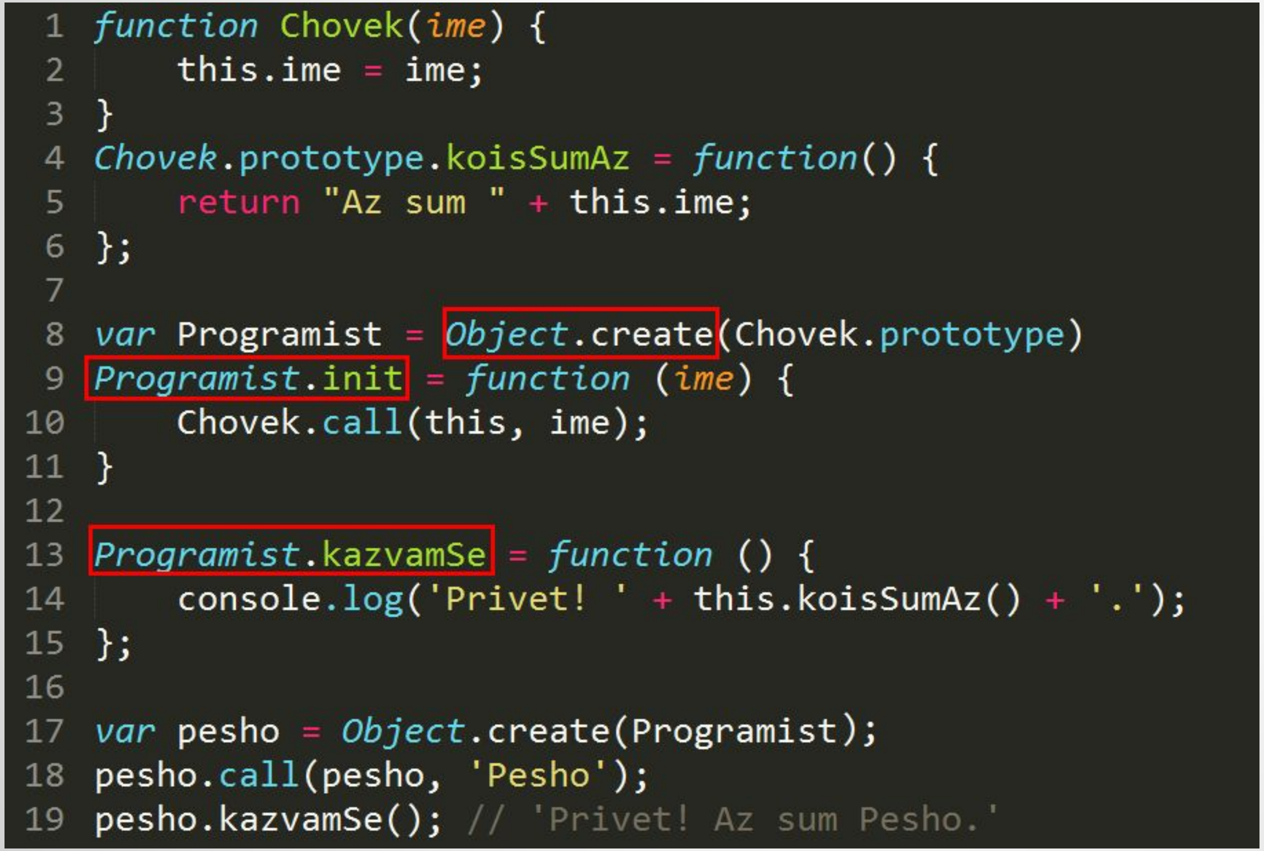
```
function Chovek(ime) {
    this.ime = ime;
}
Chovek.prototype.koisSumAz = function() {
    return "Az sum " + this.ime;
};

function Programist(ime) {
    Chovek.call(this, ime);
}

Programist.prototype = Object.create(Chovek.prototype);

Programist.prototype.kazvamSe = function () {
    console.log('Privet! ' + this.koisSumAz() + '.');
};

var pesho = Object.create(Programist.prototype);
pesho.call(pesho, 'Pesho');
pesho.kazvamSe(); // 'Privet! Az sum Pesho.'
```

Prototypal delegation - OLOO



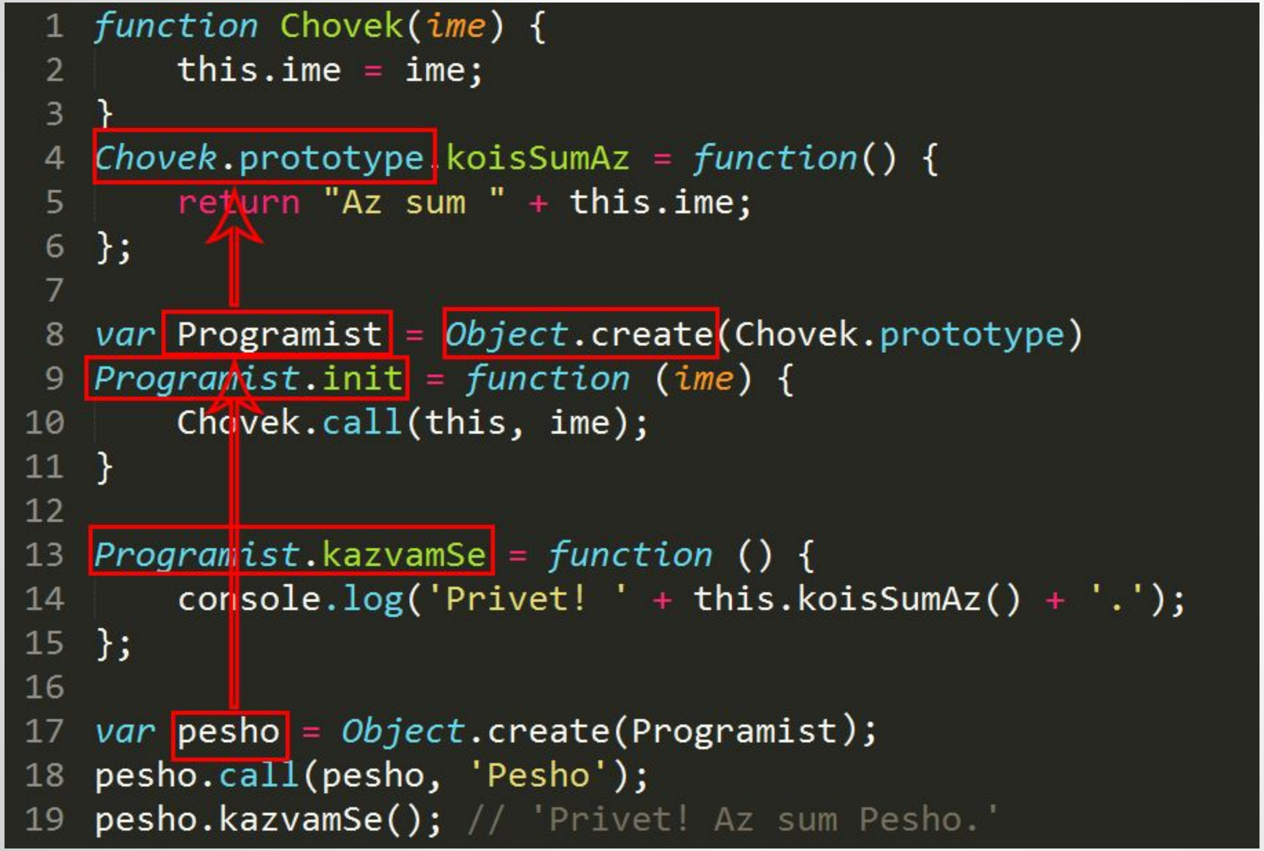
```
function Chovek(ime) {
    this.ime = ime;
}
Chovek.prototype.koisSumAz = function() {
    return "Az sum " + this.ime;
};

var Programist = Object.create(Chovek.prototype)
Programist.init = function (ime) {
    Chovek.call(this, ime);
}

Programist.kazvamSe = function () {
    console.log('Privet! ' + this.koisSumAz() + '.');
};

var pesho = Object.create(Programist);
pesho.call(pesho, 'Pesho');
pesho.kazvamSe(); // 'Privet! Az sum Pesho.'
```

Prototypal delegation - OLOO



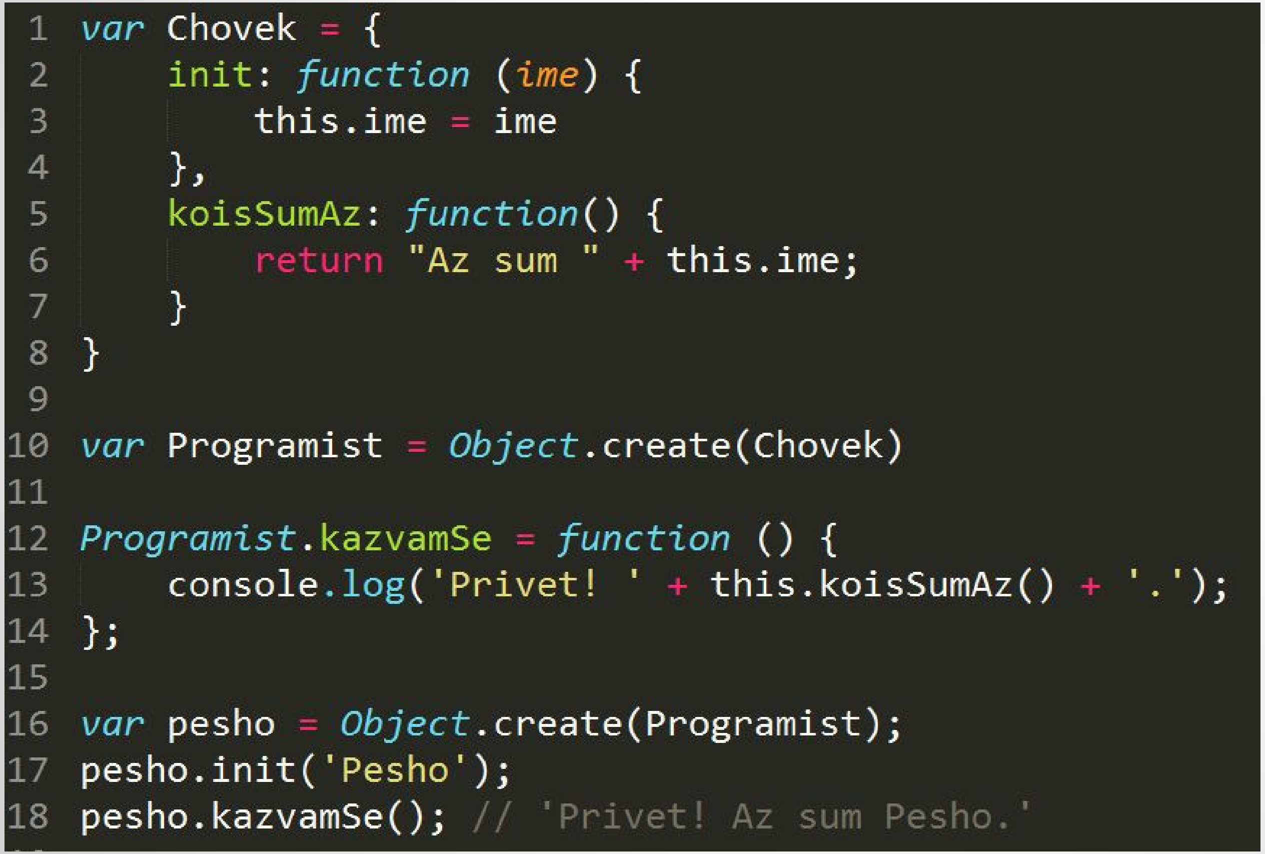
```
function Chovek(ime) {
    this.ime = ime;
}
Chovek.prototype.koisSumAz = function() {
    return "Az sum " + this.ime;
};

var Programist = Object.create(Chovek.prototype)
Programist.init = function (ime) {
    Chovek.call(this, ime);
}

Programist.kazvamSe = function () {
    console.log('Privet! ' + this.koisSumAz() + '.');
};

var pesho = Object.create(Programist);
pesho.call(pesho, 'Pesho');
pesho.kazvamSe(); // 'Privet! Az sum Pesho.'
```

Prototypal delegation - OLOO



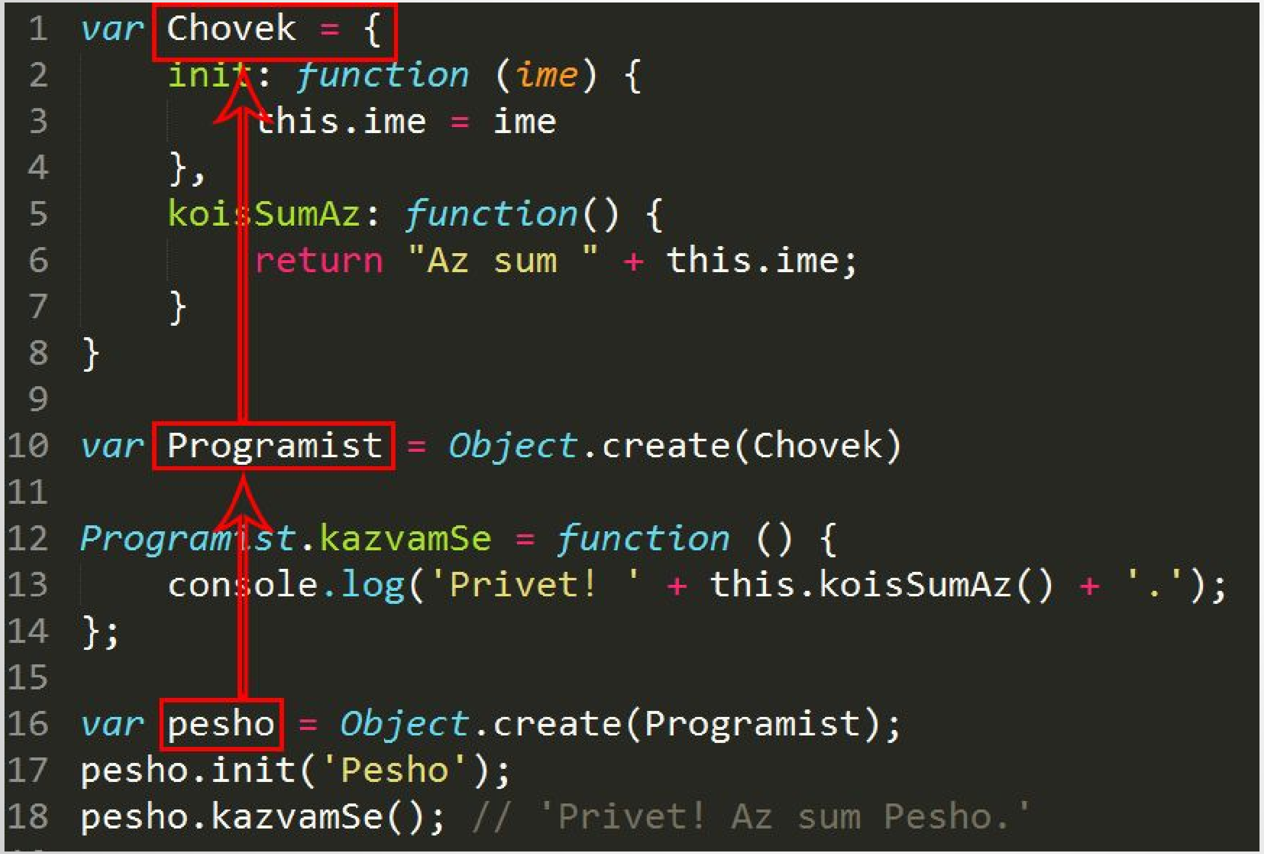
```
function Chovek(ime) {
    this.ime = ime;
}
Chovek.prototype.koisSumAz = function() {
    return "Az sum " + this.ime;
};

var Programist = Object.create(Chovek.prototype)
Programist.init = function (ime) {
    Chovek.call(this, ime);
}

Programist.kazvamSe = function () {
    console.log('Privet! ' + this.koisSumAz() + '.');
};

var pesho = Object.create(Programist);
pesho.call(pesho, 'Pesho');
pesho.kazvamSe(); // 'Privet! Az sum Pesho.'
```

Prototypal delegation - OLOO



```javascript
var Chovek = {
    init: function (ime) {
        this.ime = ime
    },
    koisSumAz: function() {
        return "Az sum " + this.ime;
    }
}

var Programist = Object.create(Chovek)

Programist.kazvamSe = function () {
    console.log('Privet! ' + this.koisSumAz() + '.');
};

var pesho = Object.create(Programist);
pesho.init('Pesho');
pesho.kazvamSe(); // 'Privet! Az sum Pesho.'
```

Prototypal delegation - OLOO



```
var Chovek = {
    init: function (ime) {
        this.ime = ime
    },
    koisSumAz: function() {
        return "Az sum " + this.ime;
    }
}

var Programist = Object.create(Chovek)

Programist.kazvamSe = function () {
    console.log('Privet! ' + this.koisSumAz() + '.');
};

var pesho = Object.create(Programist);
pesho.init('Pesho');
pesho.kazvamSe(); // 'Privet! Az sum Pesho.'
```

Quiz

1. What’s the difference between JS [[Prototype]] links and traditional classical inheritance?
2. What is behaviour delegation?
3. How is behaviour delegation helpful?
4. What are the tradeoffs of behaviour delegation?

Object Pooling

Object Pooling design pattern