ES5 / p.8

Whyte @ Webdev Club

OOP

with flavour of *prototypes*

ES5 / p.8

CLARKE
sign-specialists.co.uk

Unavený Petr
signspecialists.co.uk

```
object petr   {
        mluvit: function(){ … },
        cist: function(){ … },
        behat: function(){ … },
        simulovat: function(){ … },
        chytat: function(){ … },
        rychleReagovat: function(){ … }
}
```

ES5 / p.8

```
object petr   {
    mluvit: function(){ … },
    cist: function(){ … },
    behat: function(){ … },
    simulovat: function(){ … },
    chytat: function(){ … },
    rychleReagovat: function(){ … }
}
```

ES5 / p.8

```
object petr {
    mluvit: function(){ … },
    cist: function(){ … },
    behat: function(){ … },
    simulovat: function(){ … },
    chytat: function(){ … },
    rychleReagovat: function(){ … }
}
```

```
{
    mluvit: function(){ … },
    cist: function(){ … }
}
```

```
{
    behat: function(){ … },
    simulovat: function(){ … }
}
```

```
{
    chytat: function(){ … },
    rychleReagovat: function(){ … }
}
```

```
function Clovek(){ … }
```

```
{
    mluvit: function(){ … },
    cist: function(){ … }
}
```

```
object petr {
    mluvit: function(){ … },
    cist: function(){ … },
    behat: function(){ … },
    simulovat: function(){ … },
    chytat: function(){ … },
    rychleReagovat: function(){ … }
}
```

```
{
    behat: function(){ … },
    simulovat: function(){ … }
}
```

```
function Fotbalista(){ … }
```

```
function Brankar(){ … }
```

```
{
    chytat: function(){ … },
    rychleReagovat: function(){ … }
}
```

obj + obj + obj = petr

ES5 / p.8

```
var

clovek = new Clovek(),
fotbalista = new Fotbalista(),
brankar = new Brankar(),

petr = clovek + fotbalista + brankar;
```

ES5 / p.8

var **WRONG**

clovek = new Clovek(),
fotbalista = new Fotbalista(),
brankar = new Brankar(),

petr = clovek + fotbalista + brankar;

<u>Inheritance</u>

ES5 / p.8

Fotbalista inherit Clovek
Brankar inherit Fotbalista

petr = ***new Brankar();***

ES5 / p.8

CORRECT

Fotbalista inherit Clovek
Brankar inherit Fotbalista

petr = ***new Brankar();***

constructor function

or

Object.create()

ES5 / p.8

```
function Clovek(){
	this.mluvit = function(){ … };
	this.cist = function(){ … };
}

var martin = new Clovek();
```

ES5 / p.8

“function-constructor” or **“class-function”**

```
function Clovek(){
    this.mluvit = function(){ ... };
    this.cist = function(){ ... };
}


var martin = new Clovek();
```

“object” or **“instance”**

```
martin:

{
    mluvit: function(){ … },
    cist: function(){ … }
}
```

ES5 / p.8

```
function Fotbalista(){
    this.behat = function(){ … };
    this.simulovat = function(){ … };
}

var marek = new Fotbalista();
```

ES5 / p.8

marek:

```
{
	behat: function(){ … },
	simulovat: function(){ … }
}
```

```
function Brankar(){
    this.chytat = function(){ … };
    this.rychleReagovat = function(){ … };
}

var pavel = new Brankar();
```

ES5 / p.8

pavel:

```
{
    chytat: function(){ … },
    rychleReagovat: function(){ … }
}
```

ES5 / p.8

Clovek – 2 methods
Fotbalista – 2 methods
Brankar – 2 methods

ES5 / p.8

Clovek – 2 methods
Fotbalista – 4 methods (2 original)
Brankar – 6 methods (2 original)

<u>Prototype</u>

ES5 / p.8

Prototype is:

— property of *constructor function*
— *object*

ES5 / p.8

Fotbalista.*prototype* = { … };

ES5 / p.8

This is a BASE for <u>every</u> object, which Fotbalista-constructor will create.

```
Fotbalista.prototype = { … };
```

```
function Fotbalista(){
	/* empty */
}
```

```
function Fotbalista(){
	/* empty */
}

Fotbalista.prototype = {
	iHaveLegs: true
};
```

```
function Fotbalista(){
	/* empty */
}

Fotbalista.prototype = {
	iHaveLegs: true
};
```

```
var
f1 = new Fotbalista(),
f2 = new Fotbalista(),
f3 = new Fotbalista();

console.log(f1.iHaveLegs);
// true
console.log(f2.iHaveLegs);
// true
console.log(f3.iHaveLegs);
// true
```

Created object consists of ***2 layers**:*

***this**-defined properties*

+

***prototype** properties*

highest priority

Created object consists of ***2 layers**:*

***this**-defined properties*

+

***prototype** properties*

lowest priority

Prototype is object

ES5 / p.8

Any object

ES5 / p.8

Any object

ES5 / p.8

If we want to <u>inherit</u> properties of A to B…

ES5 / p.8

If we want to <u>inherit</u> properties of A to B…

…then we just have to <u>create instance</u> of A…

ES5 / p.8

If we want to <u>inherit</u> properties of A to B…

…then we just have to <u>create instance</u> of A…

…and <u>assign</u> this instance <u>as prototype</u> of B!

ES5 / p.8

B.prototype = new A();

ES5 / p.8

This object will be used as BASE of every object created by B

```
B.prototype = new A();
```

This object will be used as BASE of every object created by B

B.prototype = new A();

…and we can still use B constructor function to define new properties using “this” reference

```
function Clovek(){
        this.mluvit = function(){ … };
        this.cist = function(){ …};
};
```

```
function Clovek(){
        this.mluvit = function(){ … };
        this.cist = function(){ …};
};
```

```
function Fotbalista(){
        this.behat = function(){ … };
        this.simulovat = function(){ …};
};
```

```
function Clovek(){
        this.mluvit = function(){ … };
        this.cist = function(){ …};
};

function Fotbalista(){
        this.behat = function(){ … };
        this.simulovat = function(){ …};
};

Fotbalista.prototype = new Clovek();
```

ES5 / p.8

```
function Clovek(){
        this.mluvit = function(){ … };
        this.cist = function(){ …};
};
```

```
function Fotbalista(){
        this.behat = function(){ … };
        this.simulovat = function(){ …};
};
```

```
function Brankar(){
        this.chytat = function(){ … };
        this.rychleReagovat = function(){ …};
};
```

```
function Clovek(){
        this.mluvit = function(){ … };
        this.cist = function(){ …};
};

function Fotbalista(){
        this.behat = function(){ … };
        this.simulovat = function(){ …};
};

function Brankar(){
        this.chytat = function(){ … };
        this.rychleReagovat = function(){ …};
};

Fotbalista.prototype = new Clovek();
Brankar.prototype = new Fotbalista();
```

ES5 / p.8

```
function Clovek(){
        this.mluvit = ...;
        this.cist = ...;
};
function Fotbalista(){
        this.behat = ...;
        this.simulovat = ...;
};
function Brankar(){
        this.chytat = ...;
        this.rychleReagovat = ...;
};
```

```
Fotbalista.prototype = new Clovek();
Brankar.prototype = new Fotbalista();

var martin = new Clovek();
// available: mluvit, cist

var marek = new Fotbalista();
// available: mluvit, cist, behat, simulovat

var petr = new Brankar();
// available: mluvit, cist, behat,
// simulovat, chytat, rychleReagovat
```

In fact, we can define all the things with prototype only

(even without “this” reference)

ES5 / p.8

```
function Clovek(){ /* empty as desert */ }
Clovek.prototype = {
            mluvit: …,
            cist: …
};

function Fotbalista(){ /* no “this” */ }
Fotbalista.prototype = {
            behat: …,
            simulovat: …
};

function Brankar(){ /* nothing */ };
Brankar.prototype = {
            chytat: …,
            rychleReagovat: …
};
```

```
function Clovek(){ /* empty as desert */ }
Clovek.prototype = {
            mluvit: …,
            cist: …
};

function Fotbalista(){ /* no “this” */ }
Fotbalista.prototype = {
            behat: …,
            simulovat: …
};

function Brankar(){ /* nothing */ };
Brankar.prototype = {
            chytat: …,
            rychleReagovat: …
};
```

```
Fotbalista.prototype = new Clovek();
Brankar.prototype = new Fotbalista();

var martin = new Clovek();
// available: mluvit, cist

var marek = new Fotbalista();
// available: mluvit, cist, behat, simulovat

var petr = new Brankar();
// available: mluvit, cist, behat,
// simulovat, chytat, rychleReagovat
```

```
function Clovek(){ /* empty as desert */ }
Clovek.prototype = {
        mluvit: …,
        cist: …
};

function Fotbalista(){ /* no “this” */ }
Fotbalista.prototype = {
        behat: …,
        simulovat: …
};

function Brankar(){ /* nothing */ };
Brankar.prototype = {
        chytat: …,
        rychleReagovat: …
};
```

```
Fotbalista.prototype = new Clovek();
Brankar.prototype = new Fotbalista();

var martin = new Clovek();
// available: mluvit, cist

var marek = new Fotbalista();
// available: mluvit, cist, behat, simulovat

var petr = new Brankar();
// available: mluvit, cist, behat,
// simulovat, chytat, rychleReagovat
```

```
function Clovek(){ /* empty as desert */ }
Clovek.prototype = {
        mluvit: …,
        cist: …
};

function Fotbalista(){ /* no “this” */ }
Fotbalista.prototype = new Clovek();
Fotbalista.prototype.behat = …;
Fotbalista.prototype.simulovat = …;

function Brankar(){ /* nothing */ };
Brankar.prototype = new Fotbalista();
Brankar.prototype.chytat = …;
Brankar.prototype.rychleReagovat: …;
```

```
var martin = new Clovek();
// available: mluvit, cist

var marek = new Fotbalista();
// available: mluvit, cist, behat, simulovat

var petr = new Brankar();
// available: mluvit, cist, behat,
// simulovat, chytat, rychleReagovat
```

If I can define everything with only prototypes…

…why should I use constructors?

ES5 / p.8

Stored in prototype chain

Abstract vs. *Specific*

Built using constructor’s *this* reference

ES5 / p.8

```
function Clovek(jmeno, vek){
        this.jmeno = jmeno;
        this.vek = vek;
}

Clovek.prototype = {
        mluvit: function(){ … },
        cist: function(){ … }
};
```

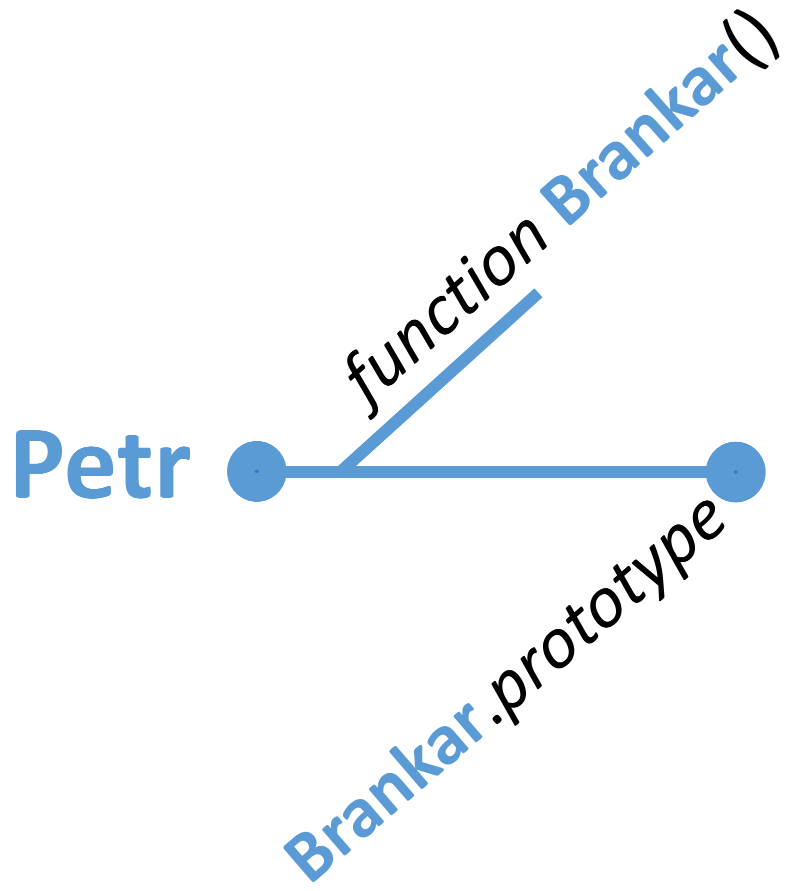
Specific ***part,***
always different
for each instance

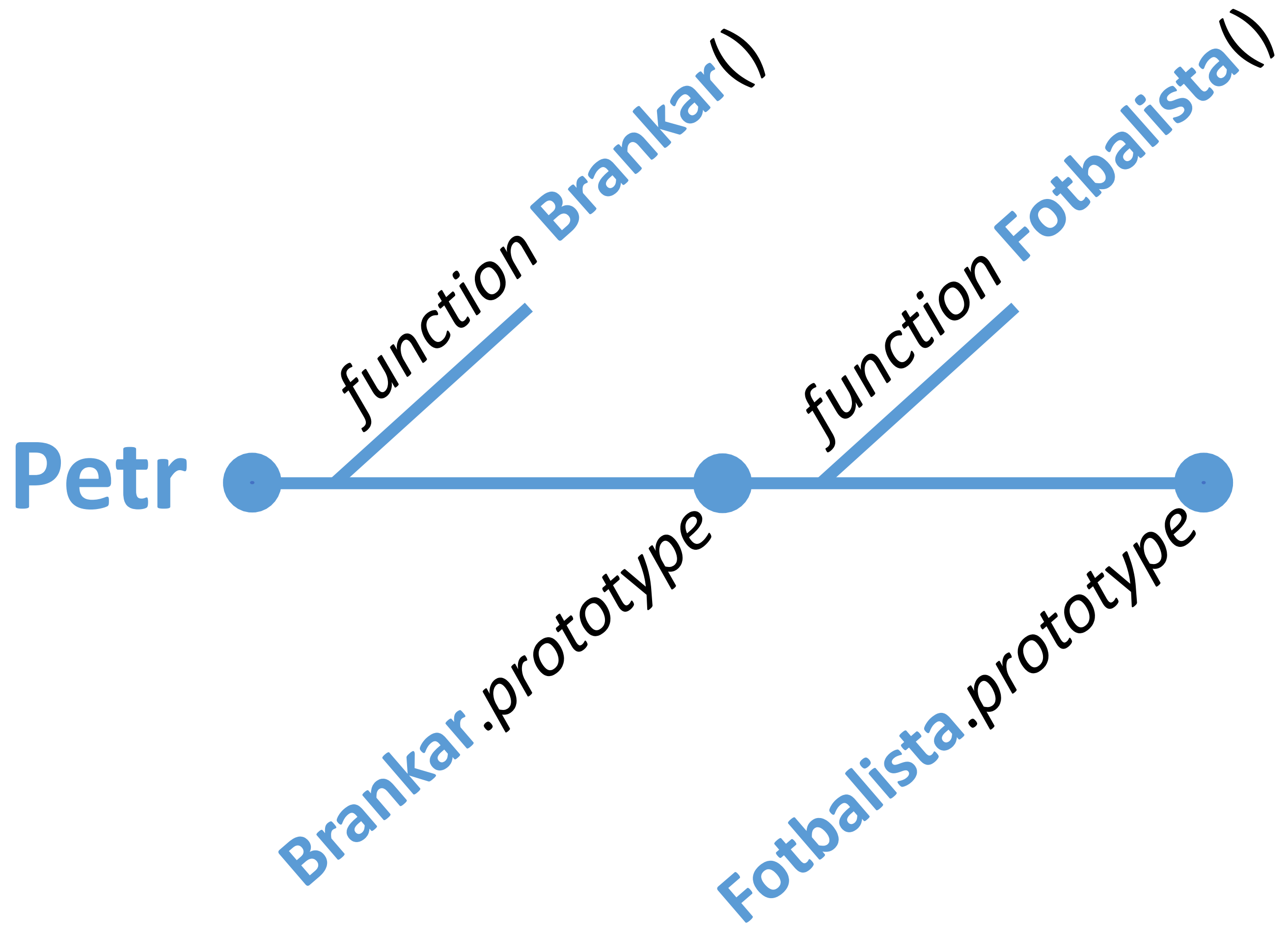
```
function Clovek(jmeno, vek){
        this.jmeno = jmeno;
        this.vek = vek;
}
```

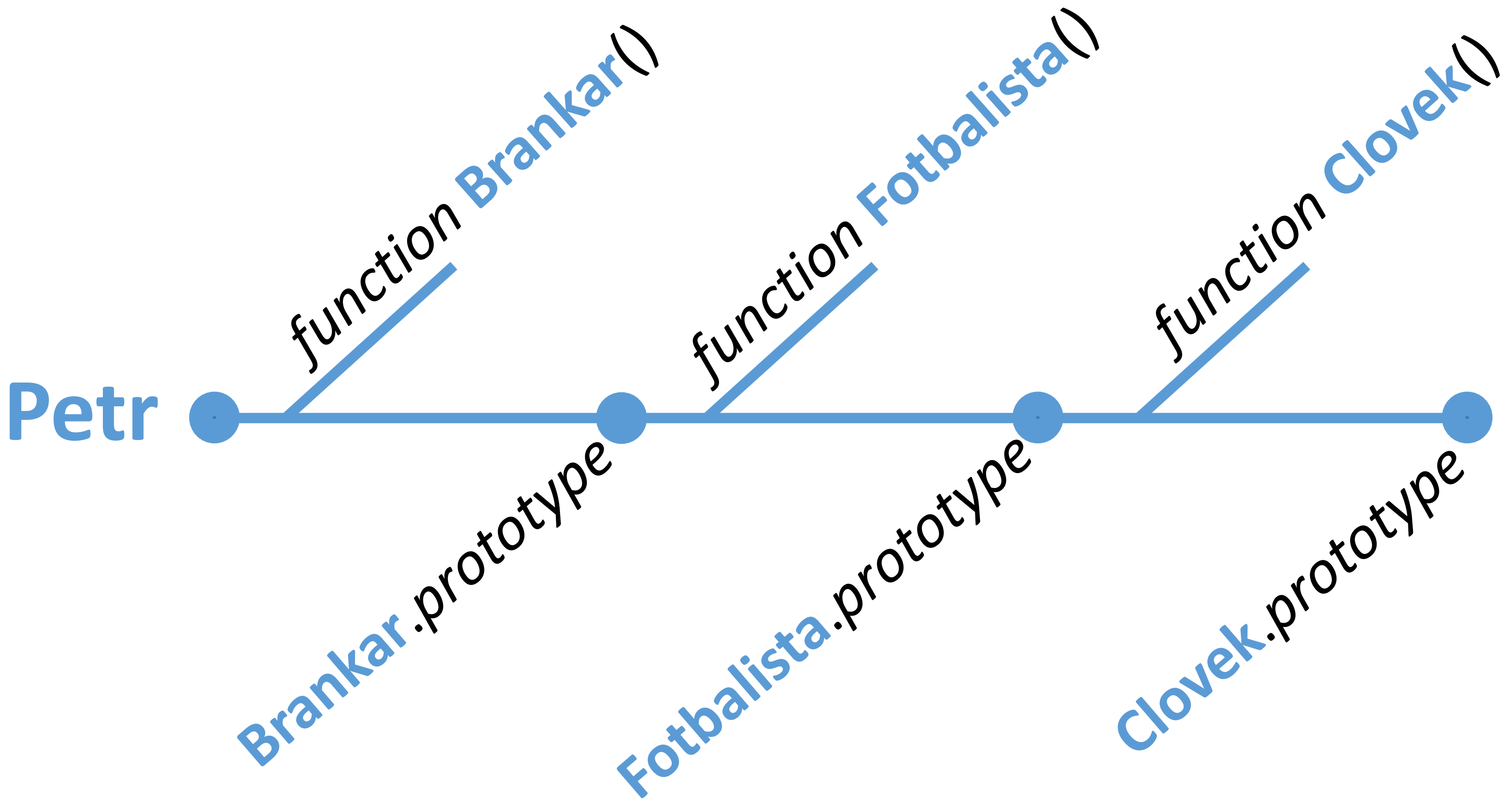
```
Clovek.prototype = {
        mluvit: function(){ … },
        cist: function(){ … }
};
```

Abstract ***part,***
common for
each instance

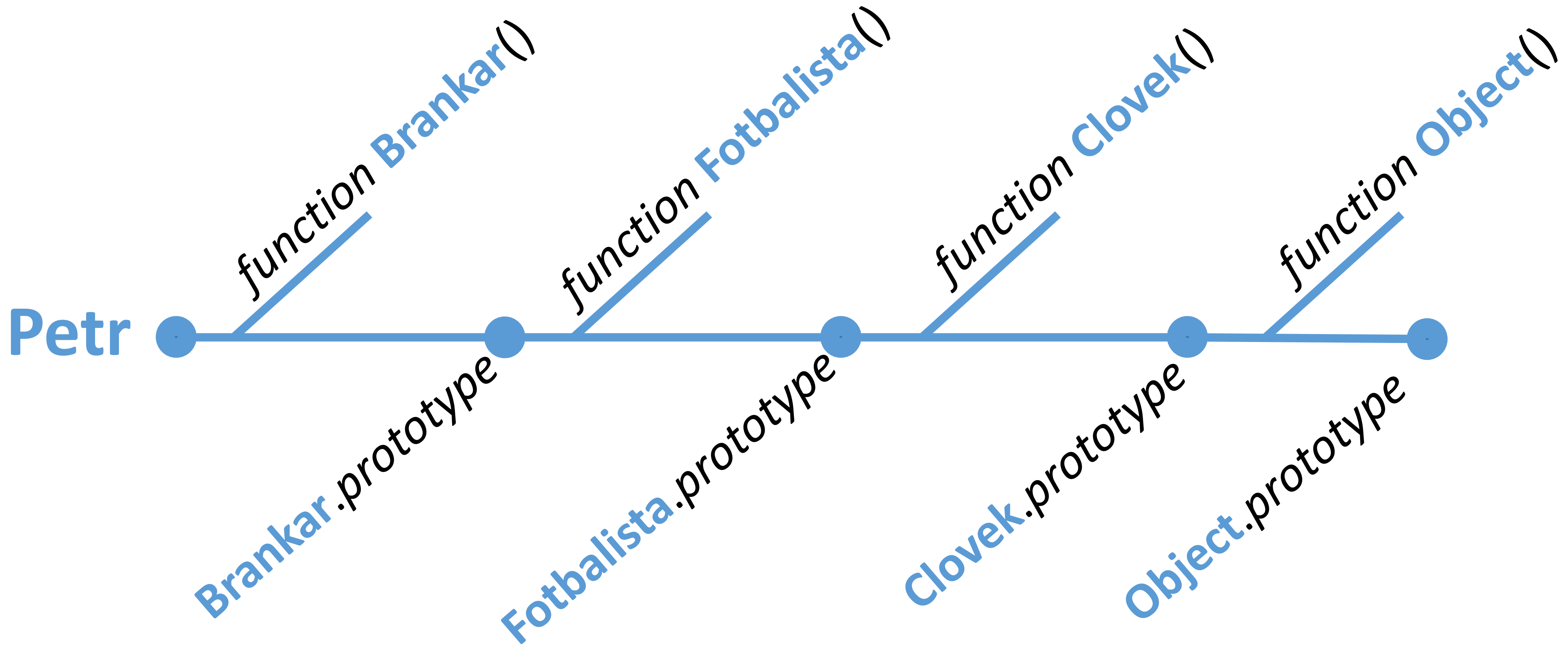
Petr ●


Petr
function Brankar()
Brankar.prototype

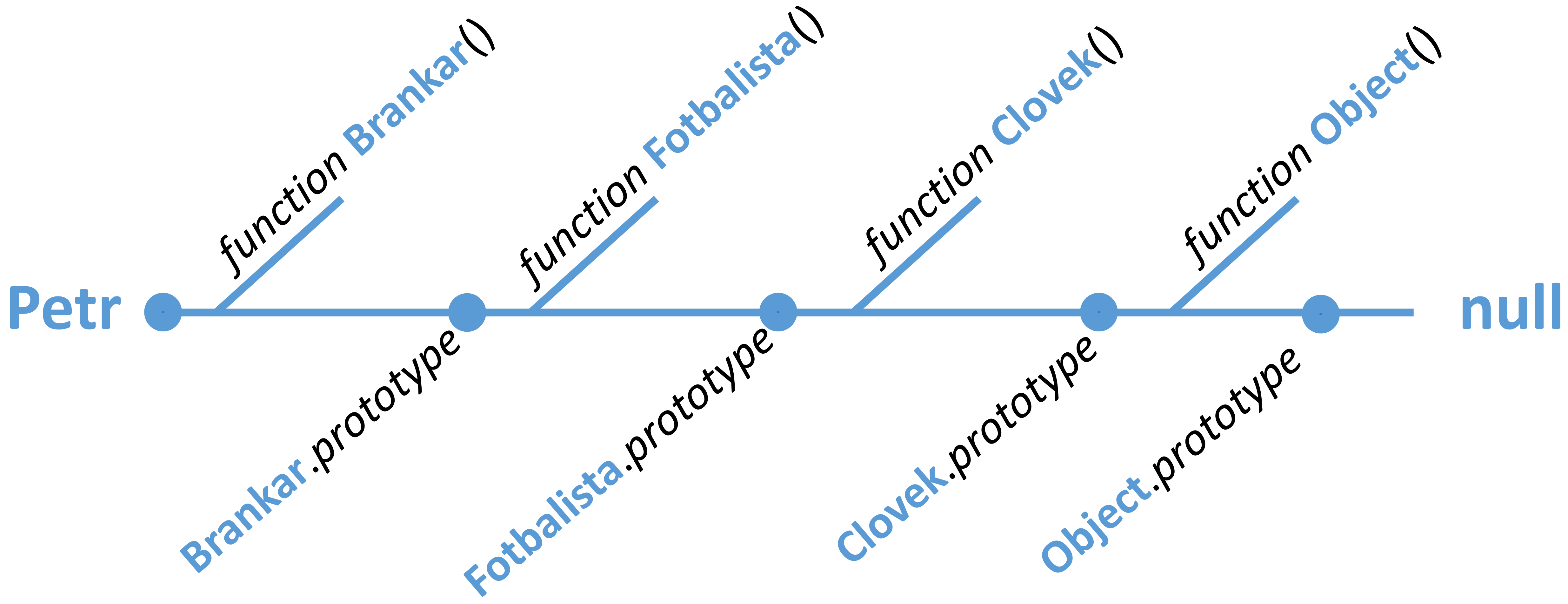

Petr
function Brankar()
Brankar.prototype
function Fotbalista()
Fotbalista.prototype

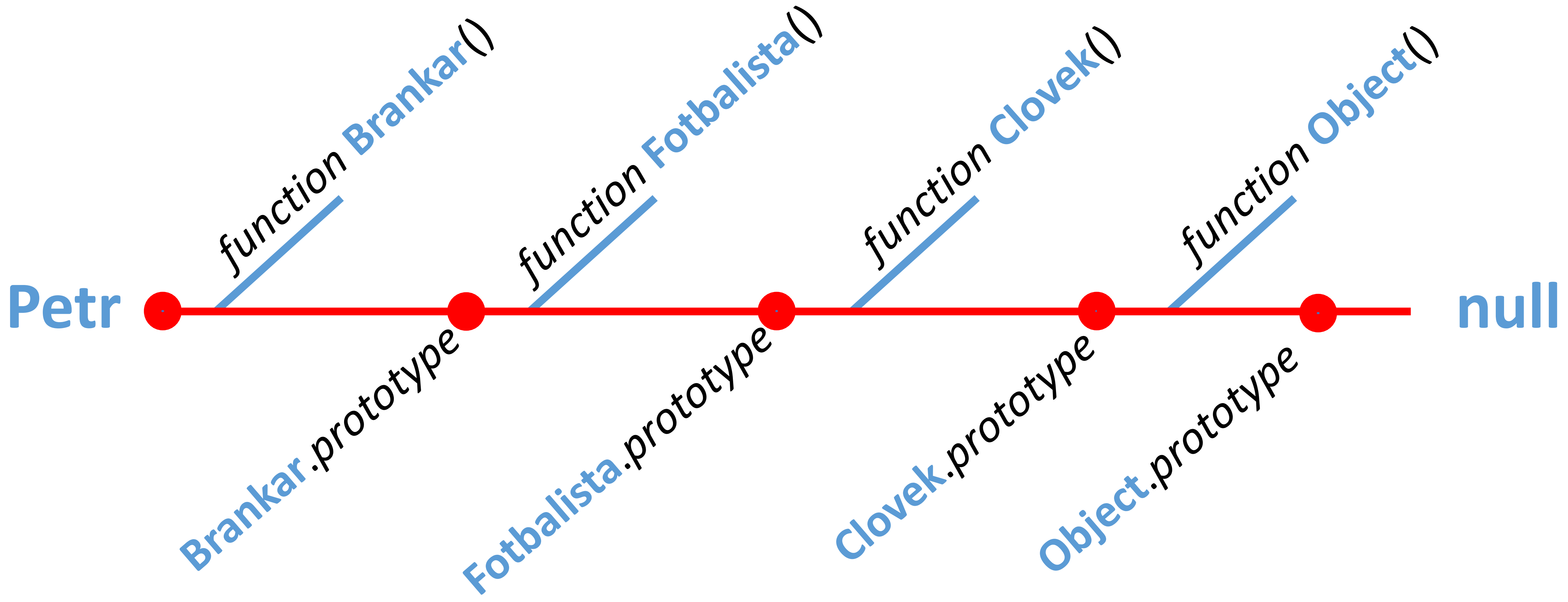

Petr
function Brankar()
Brankar.prototype
function Fotbalista()
Fotbalista.prototype
function Clovek()
Clovek.prototype

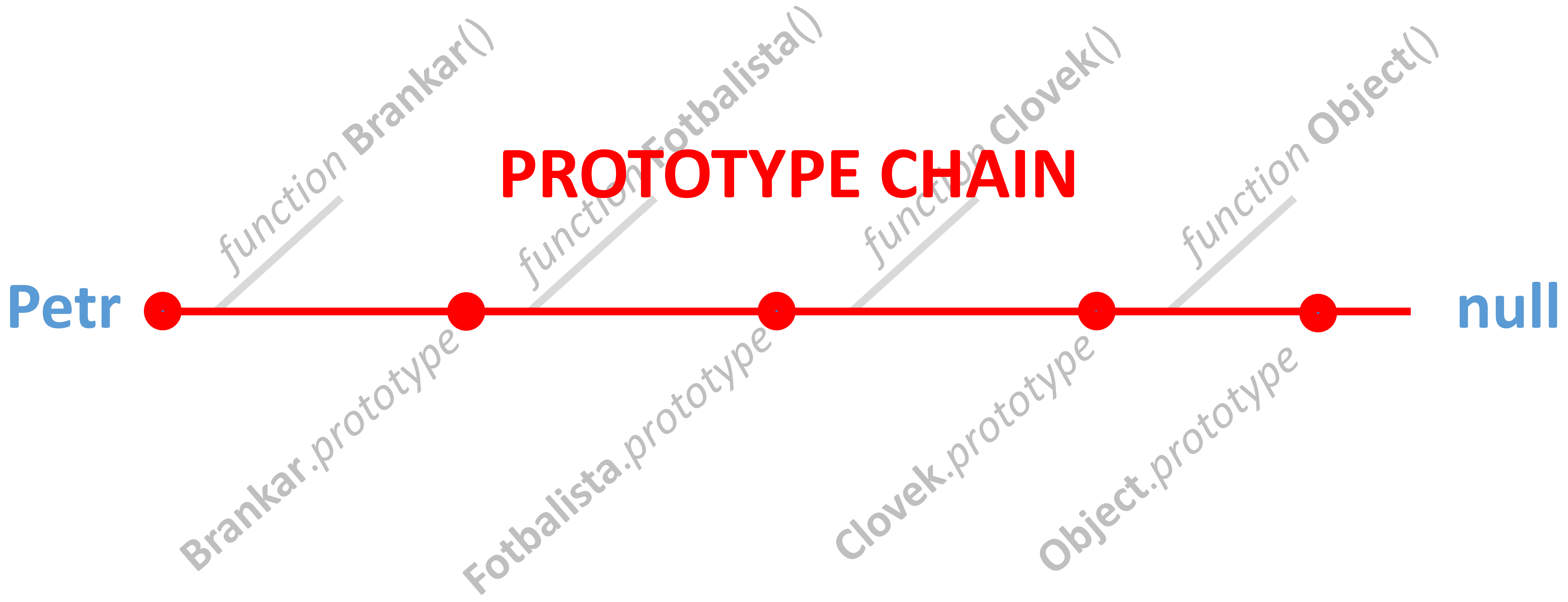
ES5 / p.8


Petr
function Brankar()
Brankar.prototype
function Fotbalista()
Fotbalista.prototype
function Clovek()
Clovek.prototype
function Object()
Object.prototype

ES5 / p.8

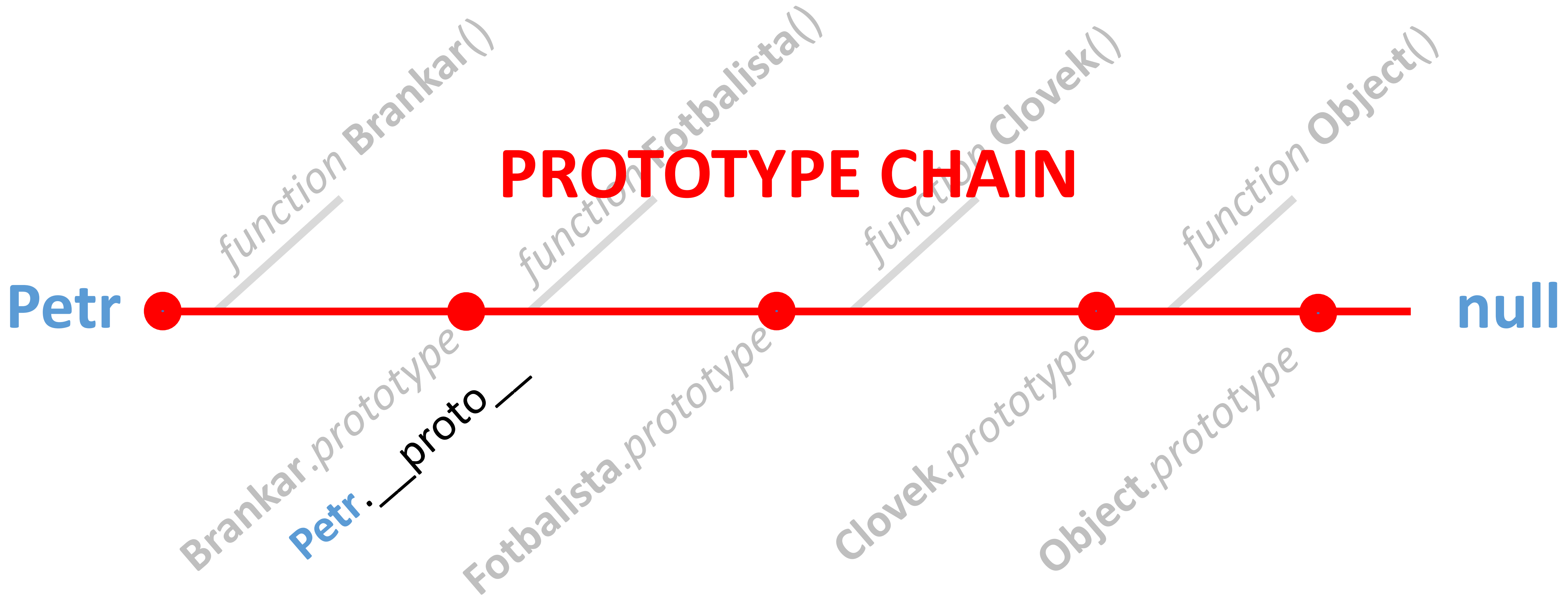

Petr
function Brankar()
Brankar.prototype
function Fotbalista()
Fotbalista.prototype
function Clovek()
Clovek.prototype
function Object()
Object.prototype
null


Petr
function Brankar()
Brankar.prototype
function Fotbalista()
Fotbalista.prototype
function Clovek()
Clovek.prototype
function Object()
Object.prototype
null

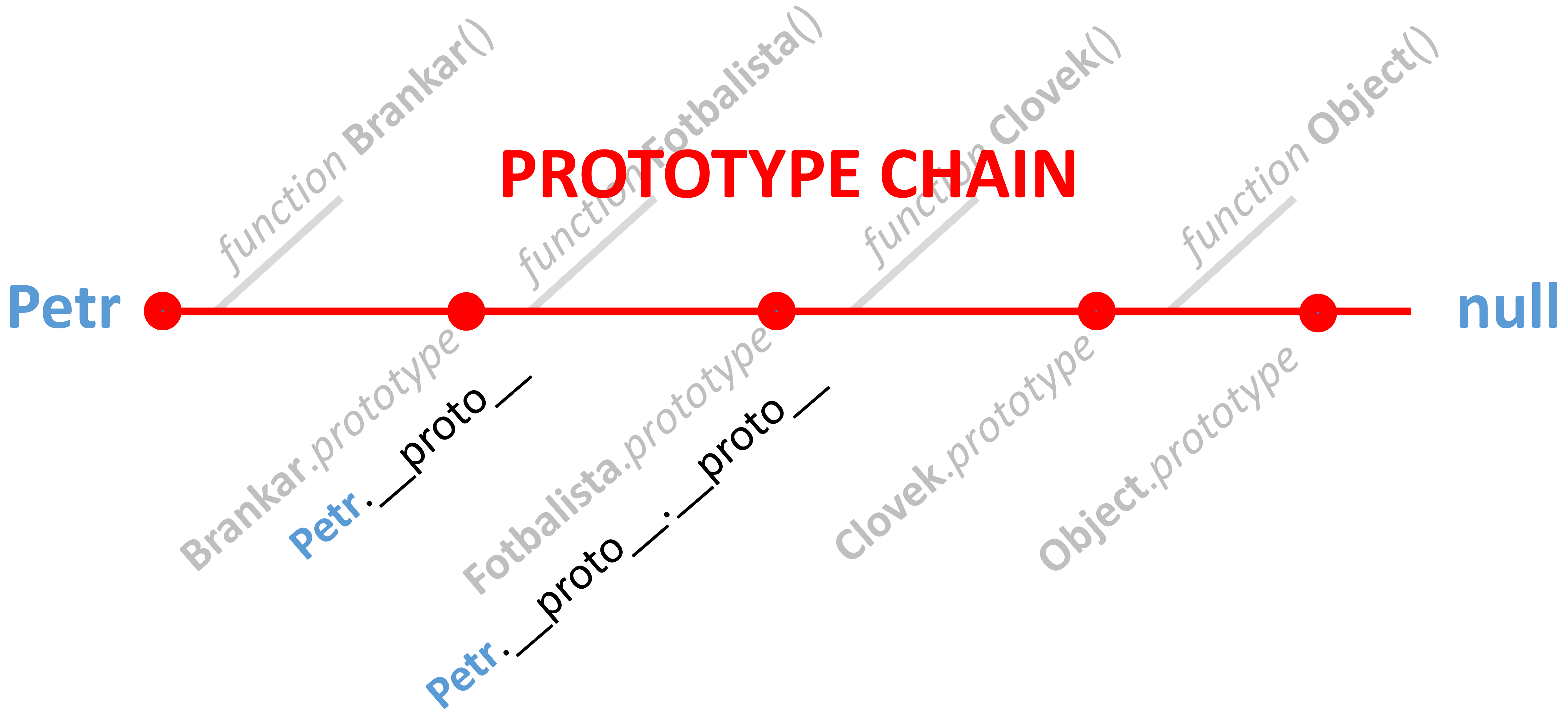

PROTOTYPE CHAIN
Petr
function Brankar()
Brankar.prototype
function Fotbalista()
Fotbalista.prototype
function Clovek()
Clovek.prototype
function Object()
Object.prototype
null

ES5 / p.8

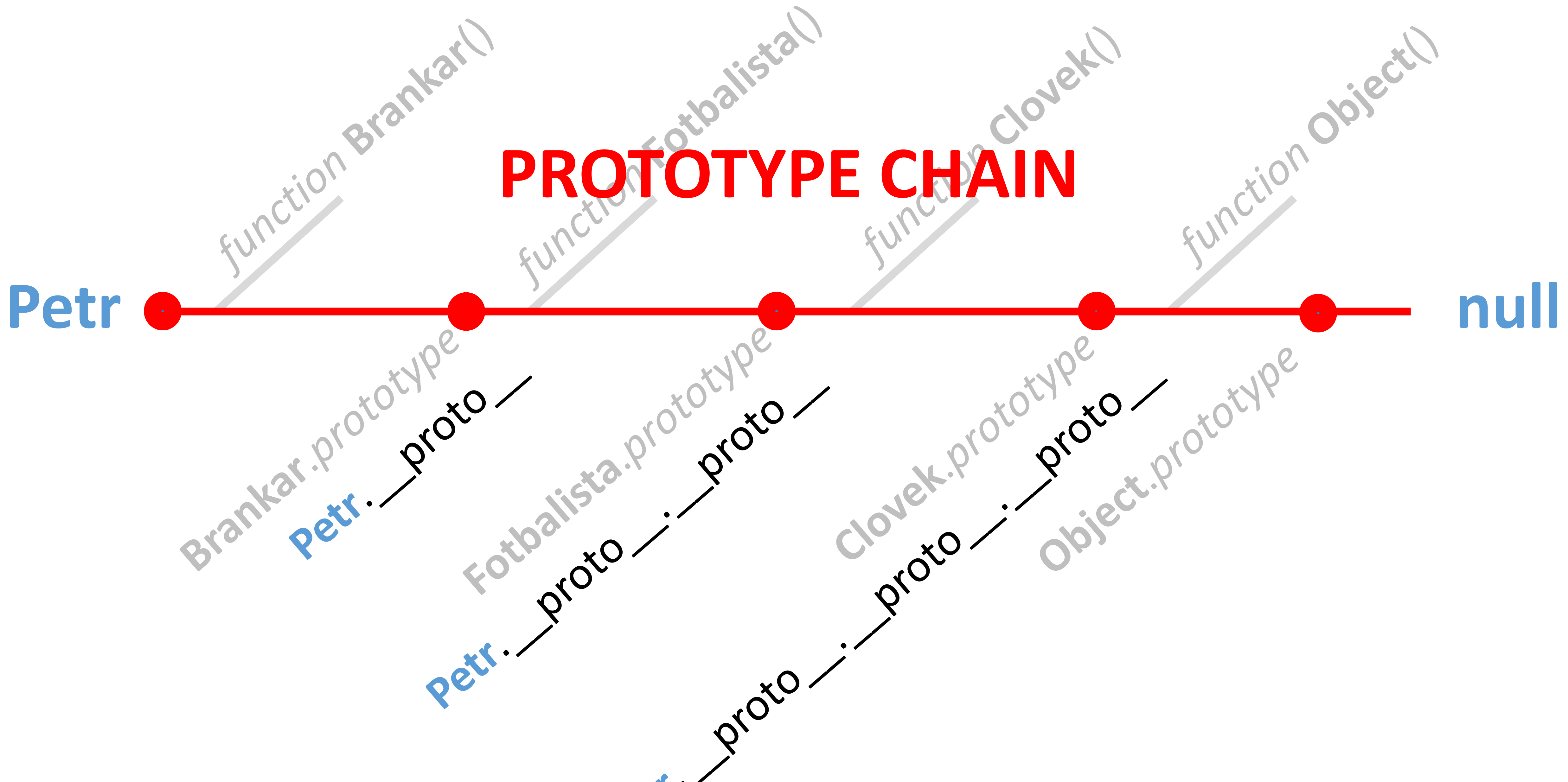
PROTOTYPE CHAIN



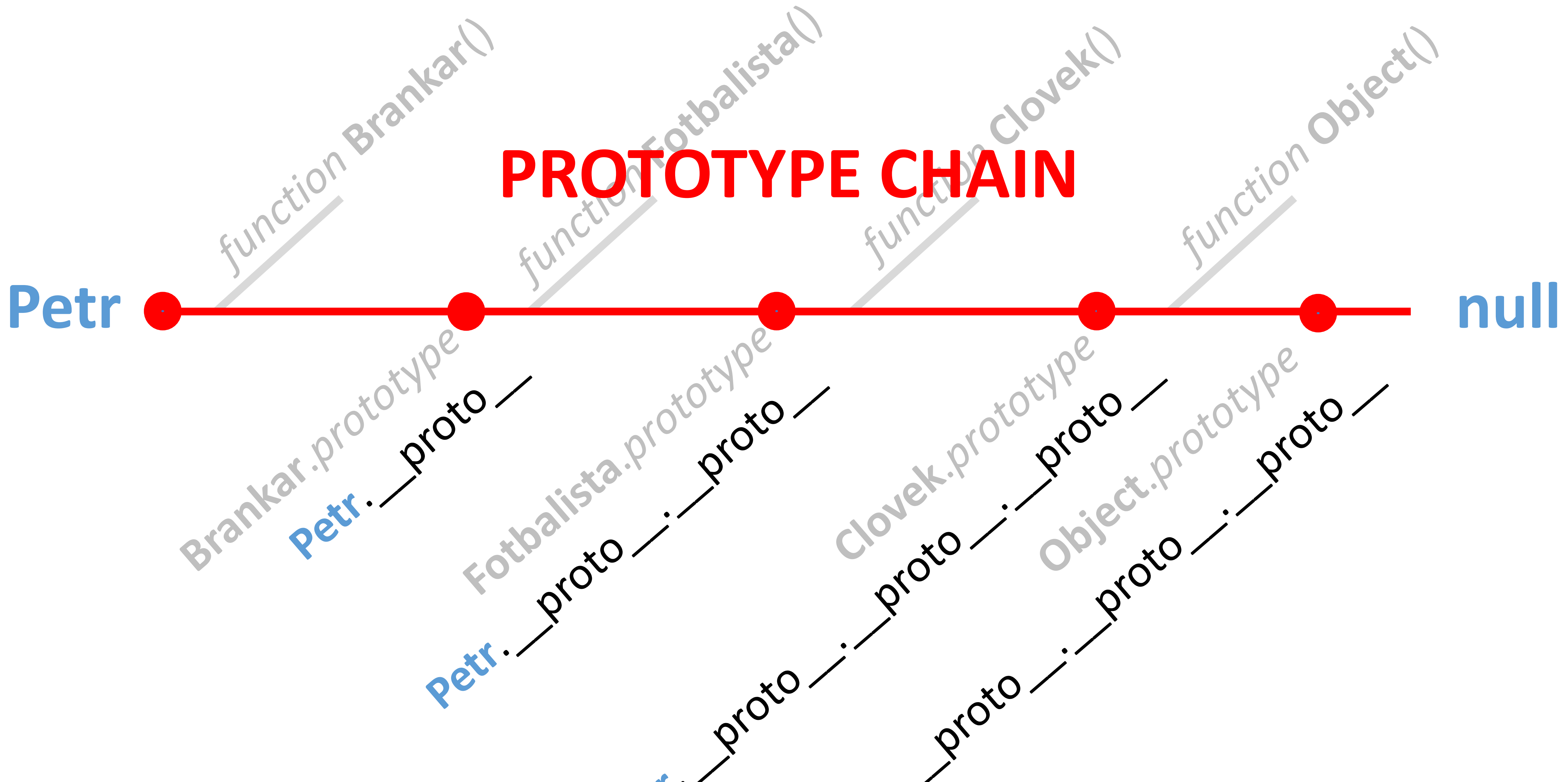
PROTOTYPE CHAIN



PROTOTYPE CHAIN



PROTOTYPE CHAIN



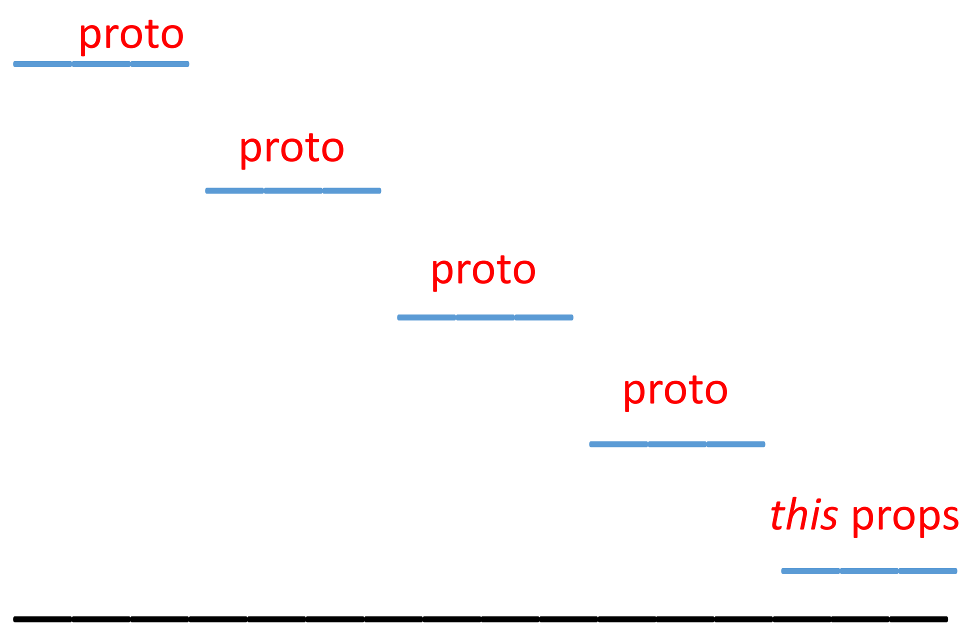
__proto__

a single layer of what object consists of

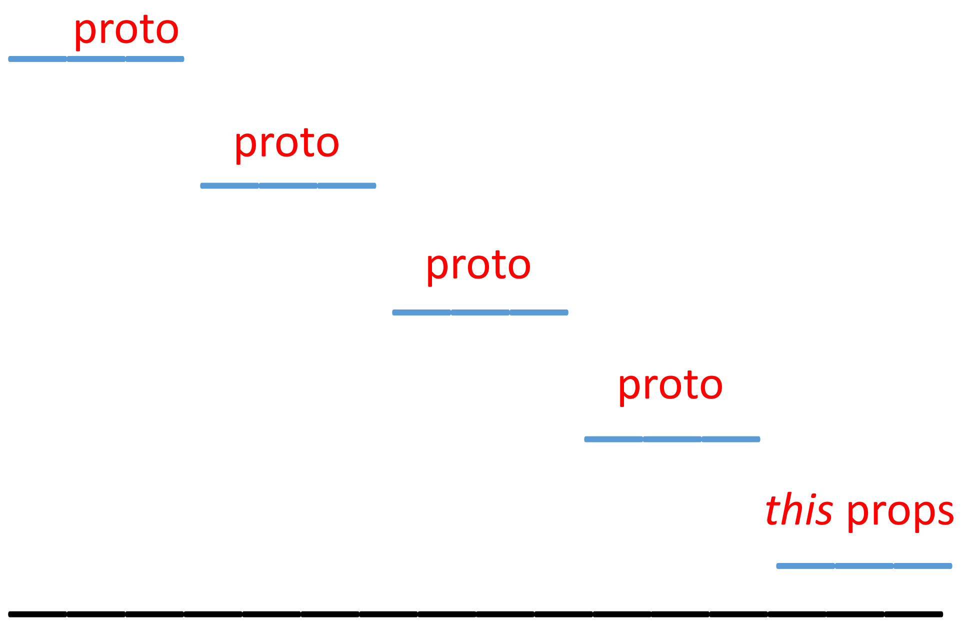
ES5 / p.8

Well, not just a layer.
A <u>cumulative</u> layer.

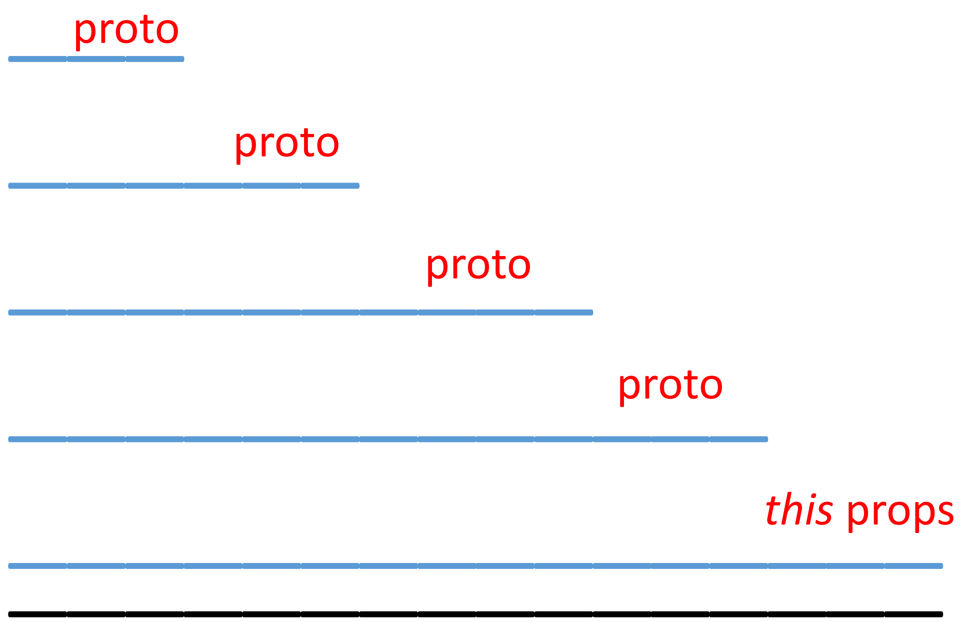
ES5 / p.8


proto
proto
proto
proto
this props

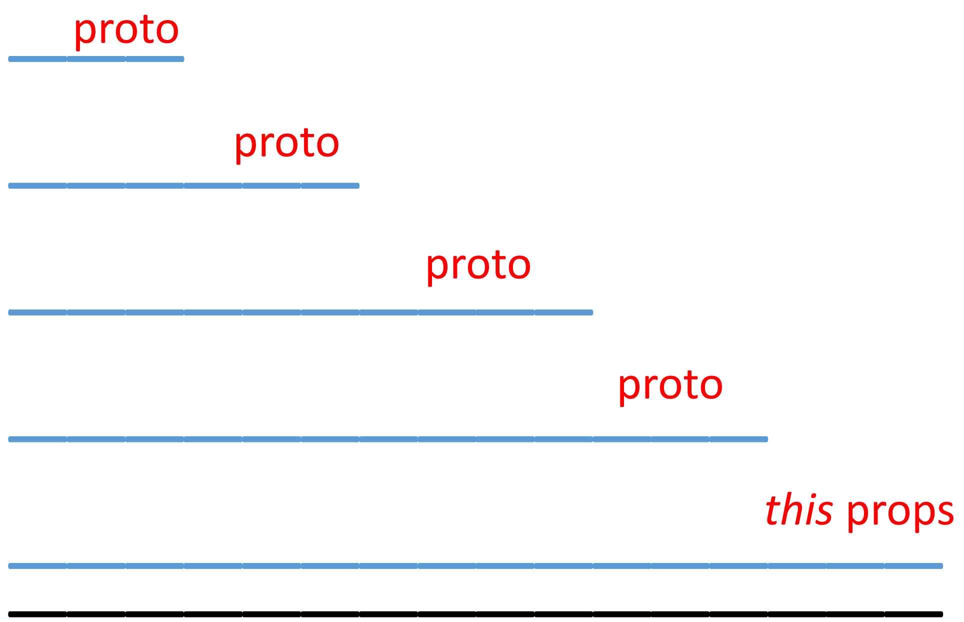
WRONG



ES5 / p.8


proto
proto
proto
proto
this props

CORRECT



// end

ES5 / p.8