JS → p.4

Nikita Mironov, 2019

ČVUT FIT BI-PJS

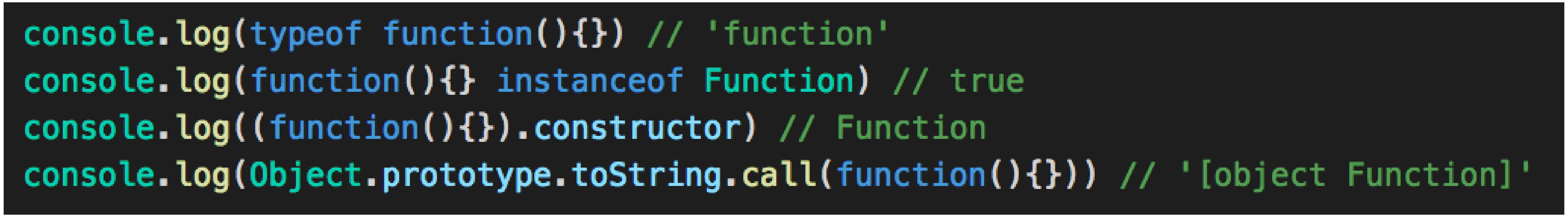
Function

typeof function(){} === ‘*function*’

Pozor, datový typ funkce
je ale object!

↓

typeof function(){} === ‘*function*’



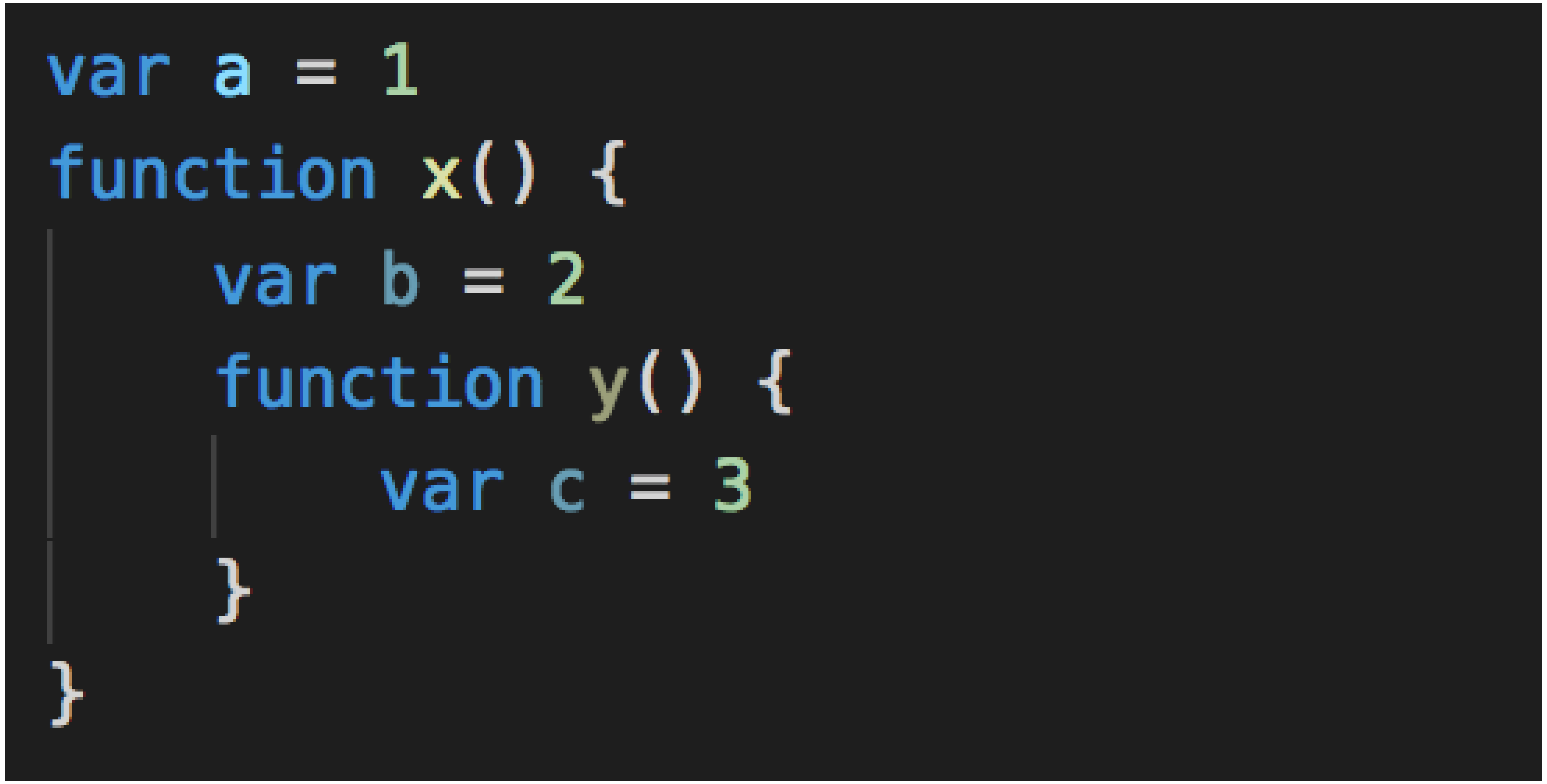
```
console.log(typeof function(){}) // 'function'
console.log(function(){} instanceof Function) // true
console.log((function(){}).constructor) // Function
console.log(Object.prototype.toString.call(function(){})) // '[object Function]'
```

funkce má všechny objektové vlastnosti

…má i některé navíc, například skryté vlastnosti [[Call]] *a* [[Scope]]

Scope

funkce je inkapsulační jednotka, vymezuje viditelnost proměnných



```
var a = 1
function x() {
    var b = 2
    function y() {
        var c = 3
    }
}
```

Globální vs. Lokální

window – *prohlížeč*
global – *Node.js*

↓

Globální vs. Lokální

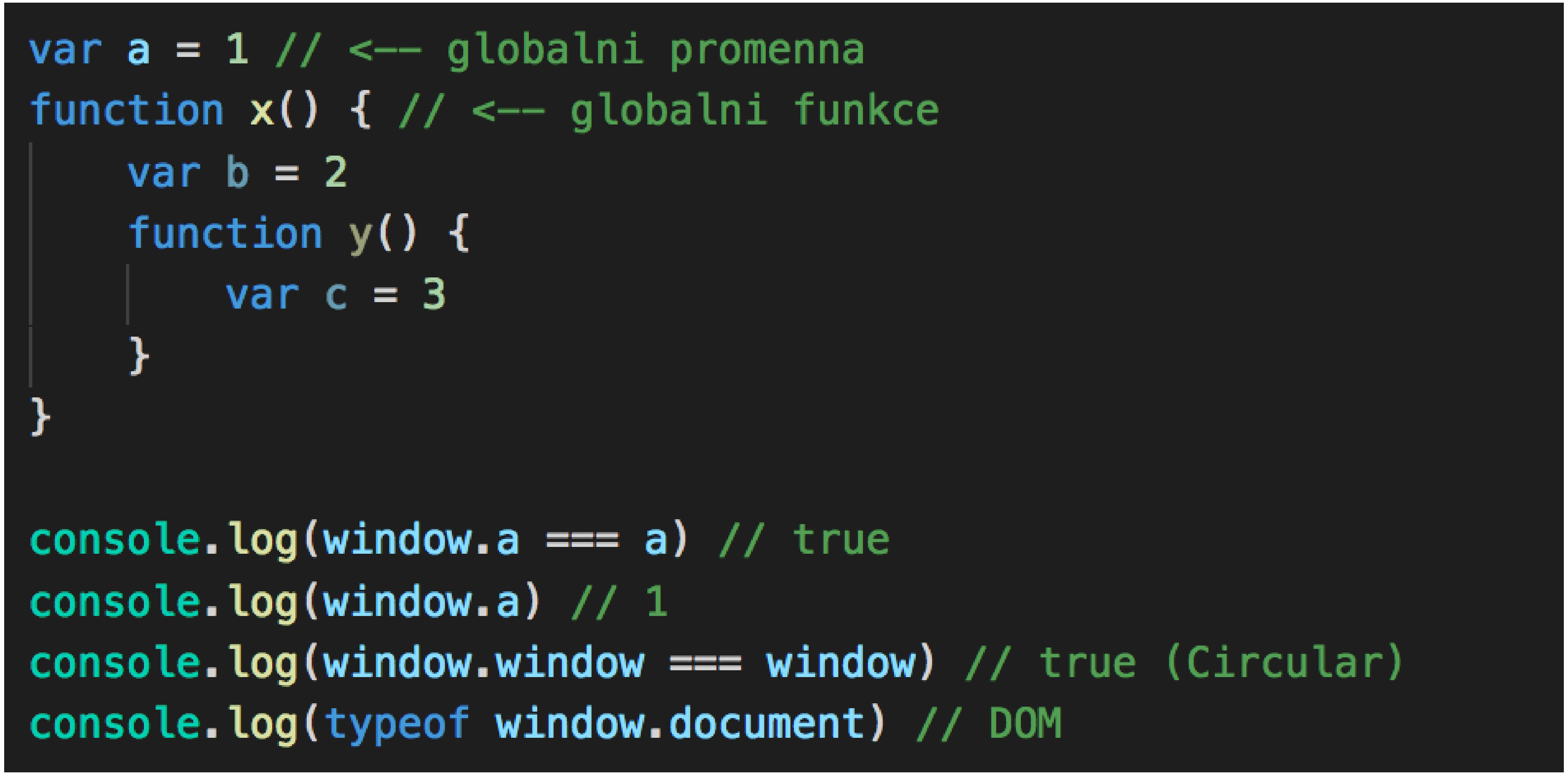
window – prohlížeč
global – Node.js

↓

Globální vs. Lokální

↑

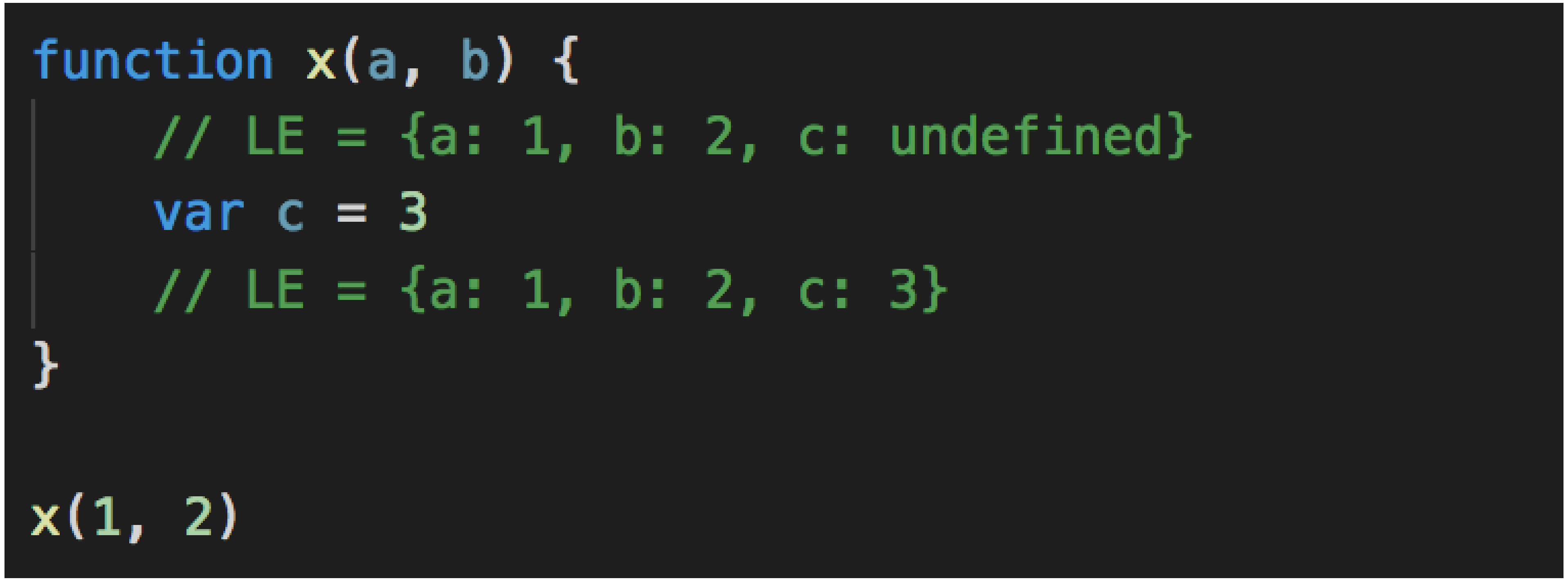
LexicalEnvironment



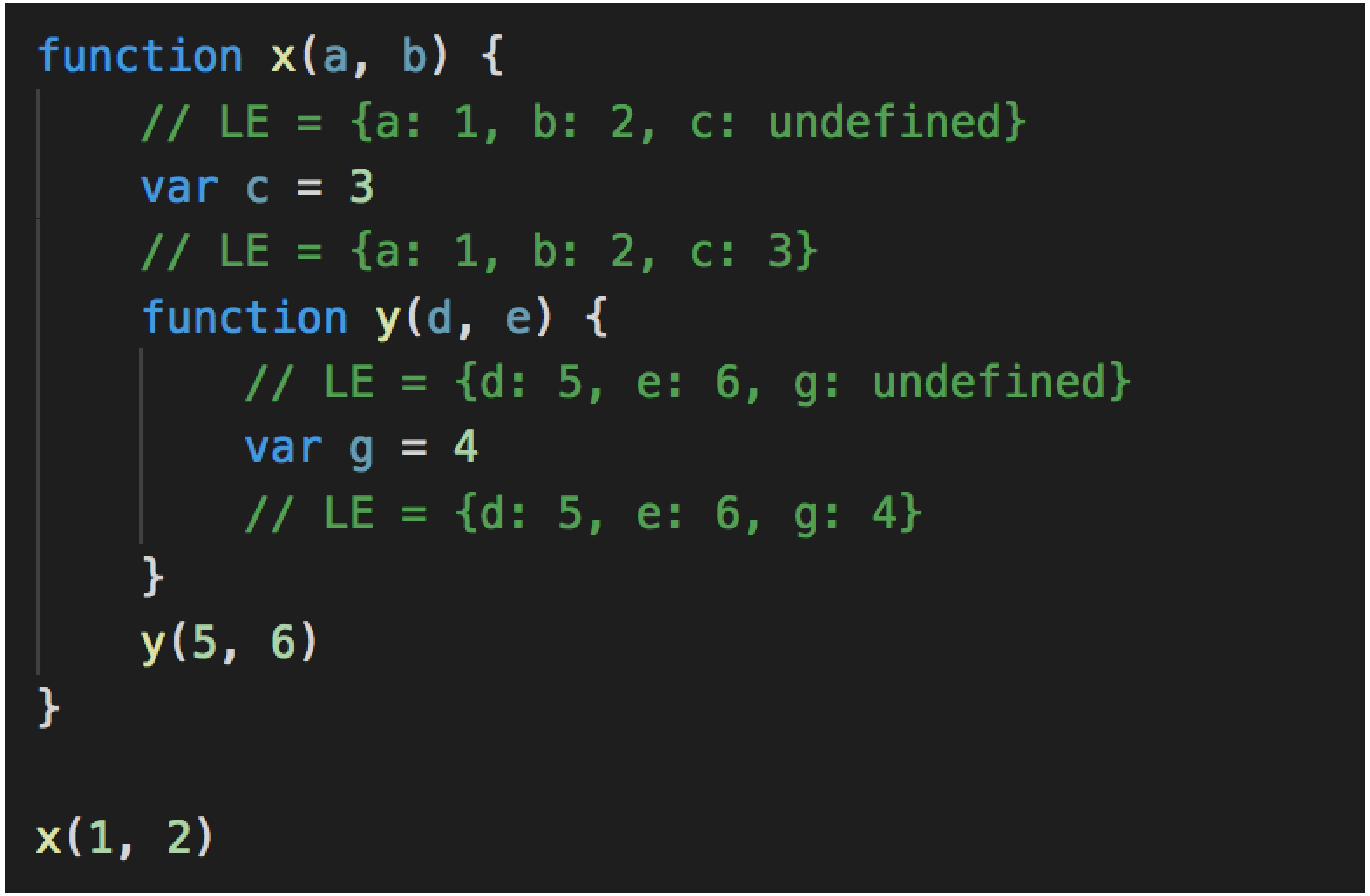
```
var a = 1 // <-- globalni promenna
function x() { // <-- globalni funkce
    var b = 2
    function y() {
        var c = 3
    }
}

console.log(window.a === a) // true
console.log(window.a) // 1
console.log(window.window === window) // true (Circular)
console.log(typeof window.document) // DOM
```

LexicalEnvironment

interní objekt (není pro nás přímo přístupný), obsahující lokální proměnné + argumenty

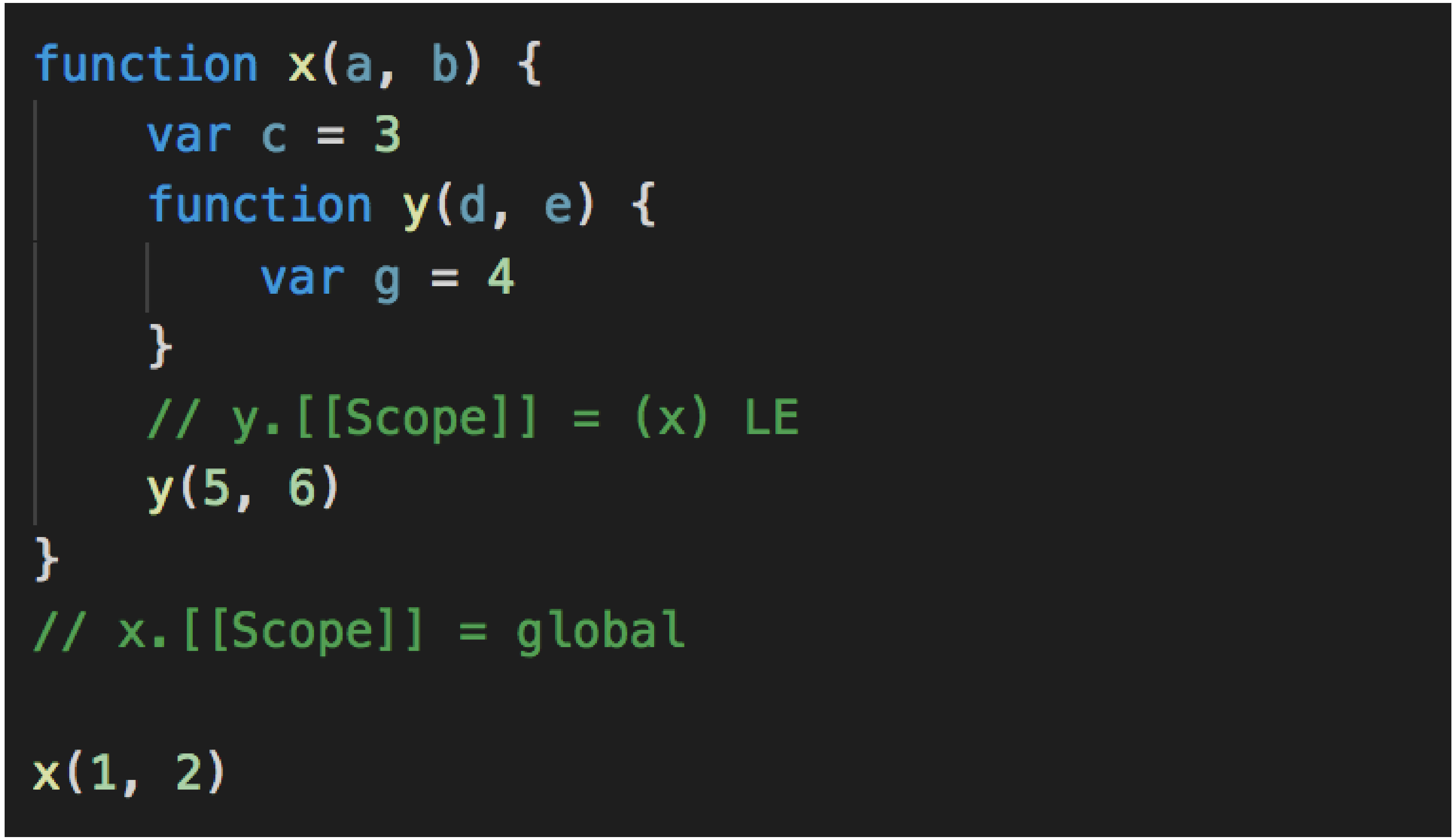


```
function x(a, b) {
    // LE = {a: 1, b: 2, c: undefined}
    var c = 3
    // LE = {a: 1, b: 2, c: 3}
}

x(1, 2)
```



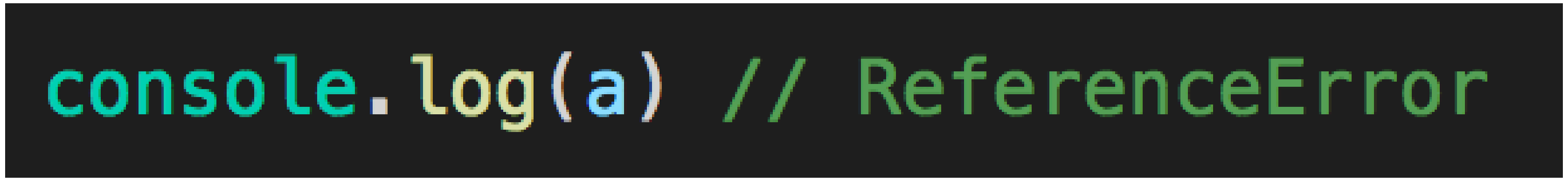
```
function x(a, b) {
    // LE = {a: 1, b: 2, c: undefined}
    var c = 3
    // LE = {a: 1, b: 2, c: 3}
    function y(d, e) {
        // LE = {d: 5, e: 6, g: undefined}
        var g = 4
        // LE = {d: 5, e: 6, g: 4}
    }
    y(5, 6)
}

x(1, 2)
```

fn.[[Scope]] = LE “rodičovské” funkce

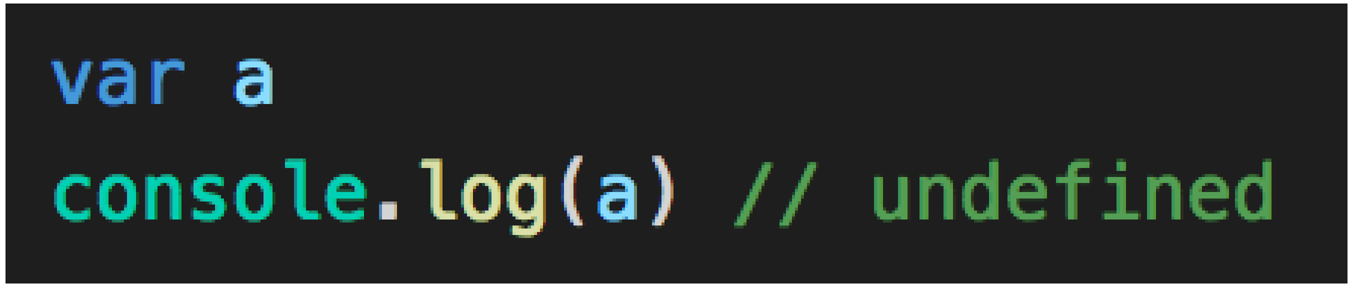


```
function x(a, b) {
    var c = 3
    function y(d, e) {
        var g = 4
    }
    // y.[[Scope]] = (x) LE
    y(5, 6)
}
// x.[[Scope]] = global

x(1, 2)
```

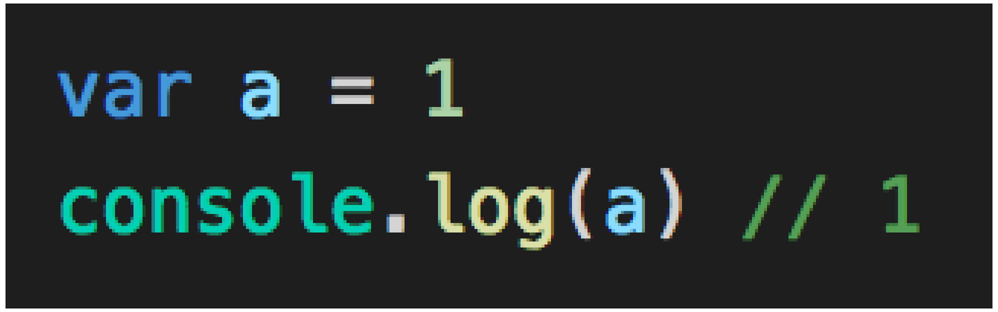
Hoisting → *var, function*



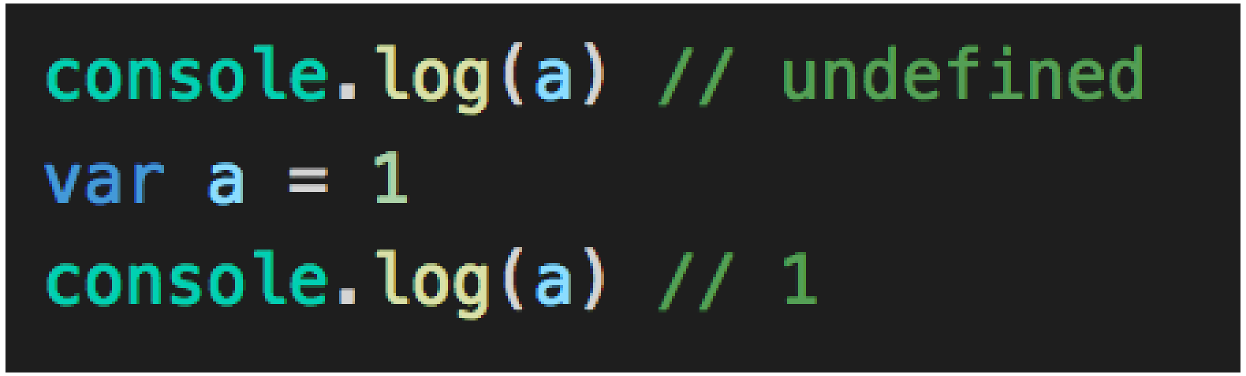
```
console.log(a) // ReferenceError
```



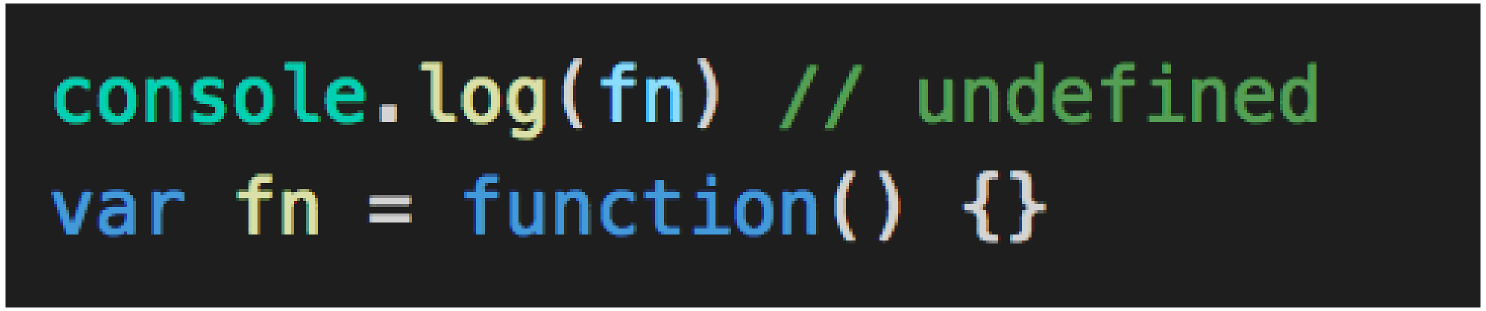
```
var a
console.log(a) // undefined
```

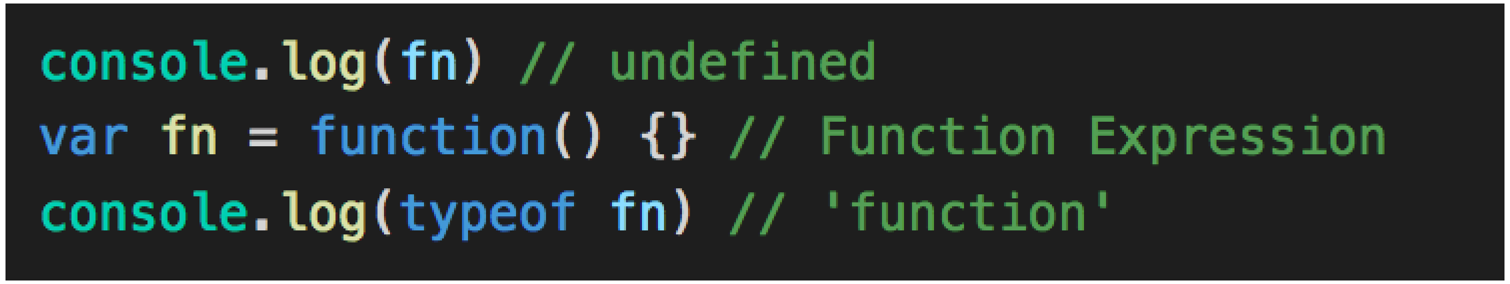


```
var a = 1
console.log(a) // 1
```

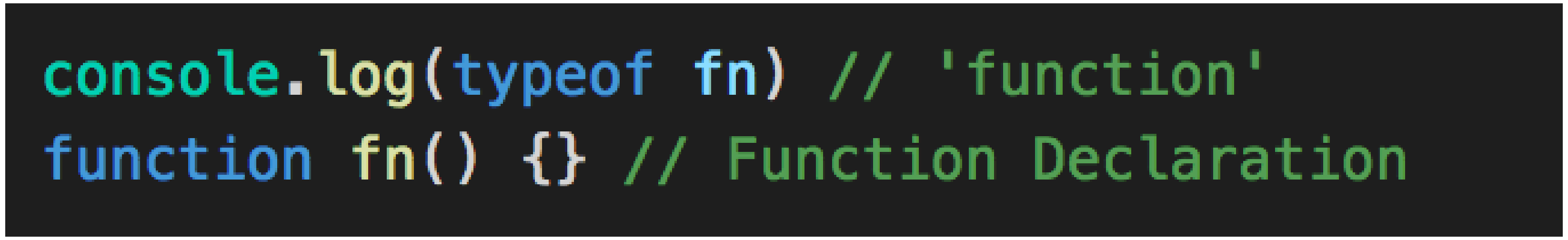


```
console.log(a) // undefined
var a = 1
console.log(a) // 1
```

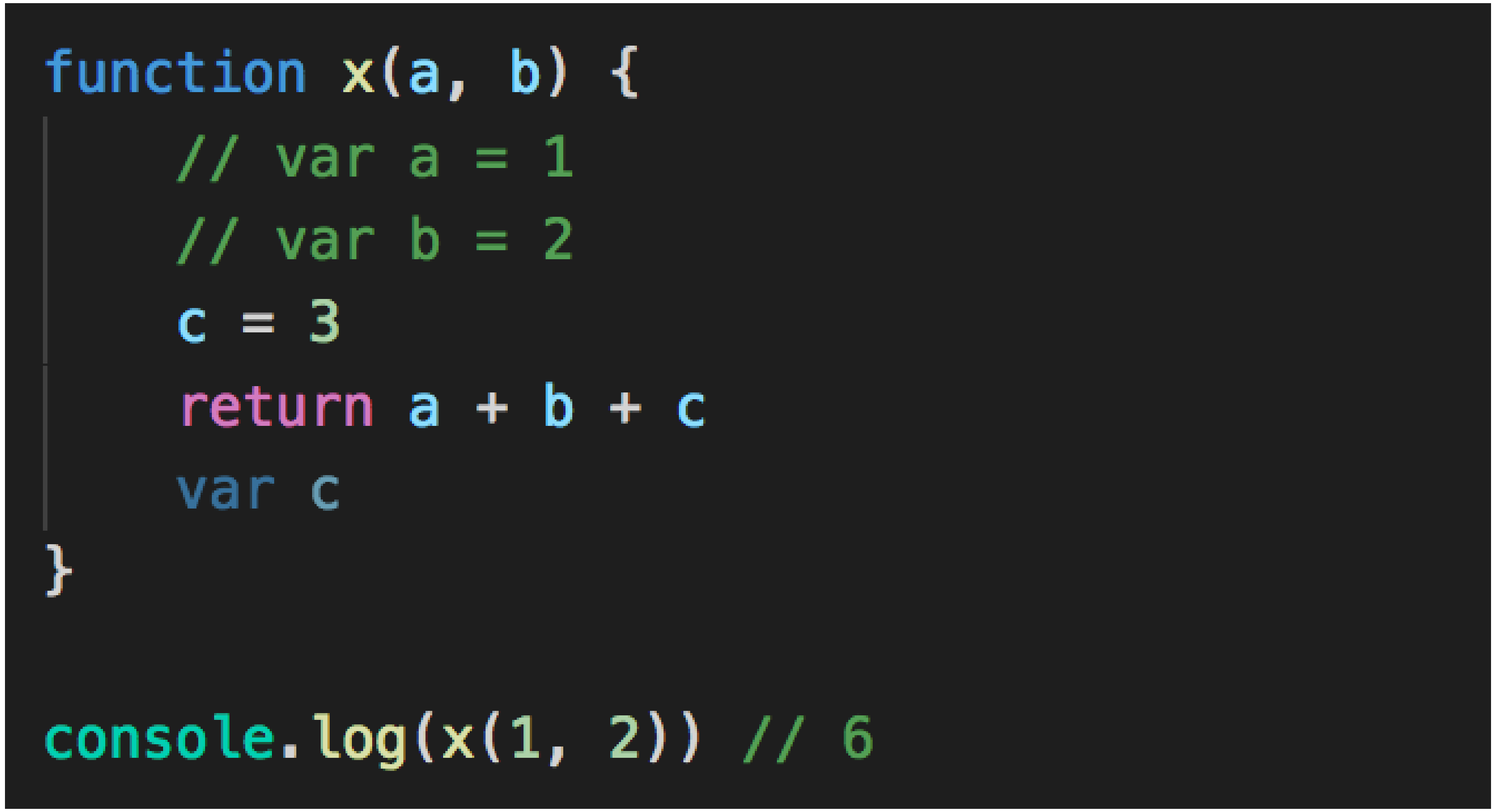

```
console.log(fn) // undefined
var fn = function() {}
```



```
console.log(fn) // undefined
var fn = function() {} // Function Expression
console.log(typeof fn) // 'function'
```

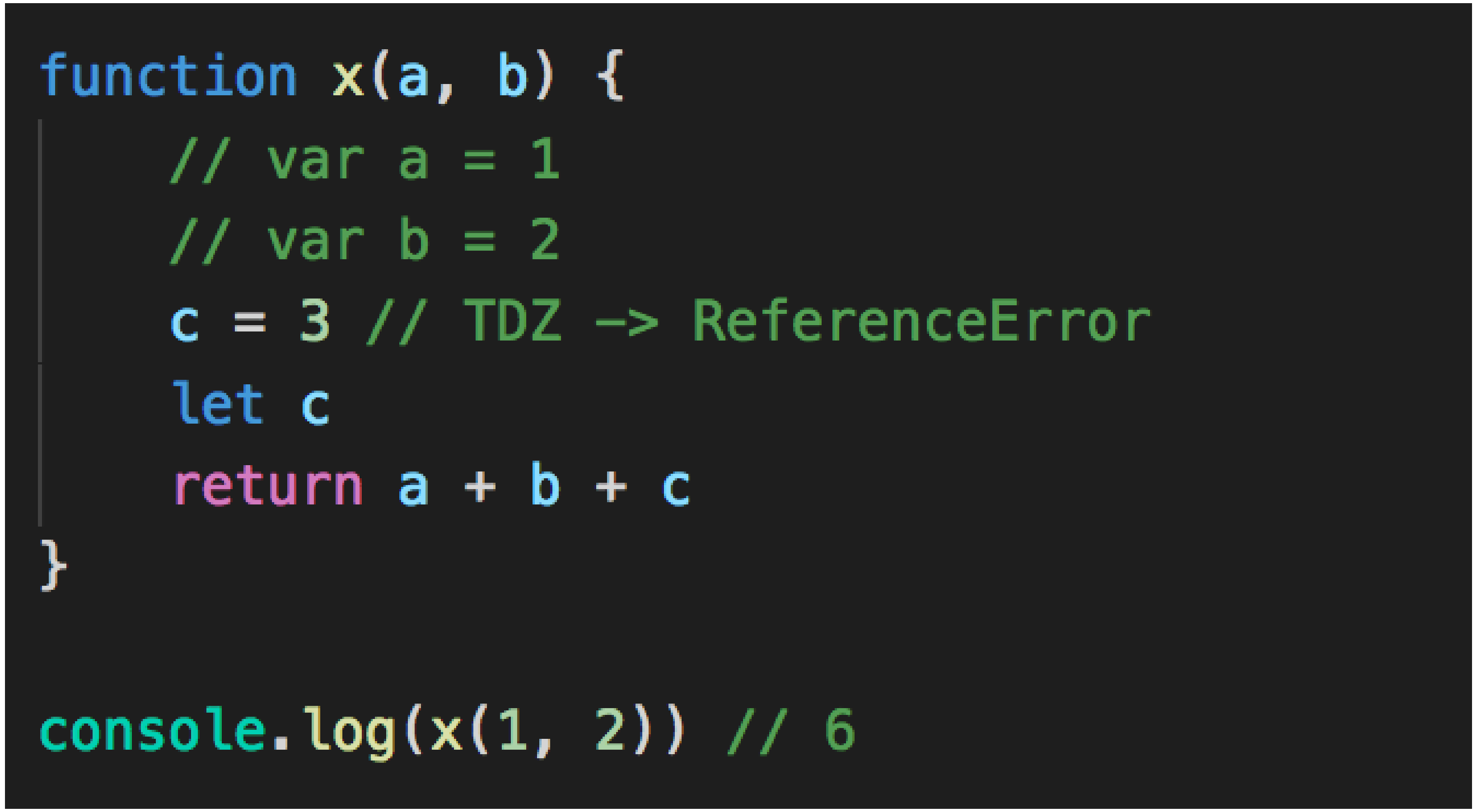


```
console.log(typeof fn) // 'function'
function fn() {} // Function Declaration
```



```
function x(a, b) {
    // var a = 1
    // var b = 2
    c = 3
    return a + b + c
    var c
}

console.log(x(1, 2)) // 6
```

Temporal Dead Zone → *let, const*

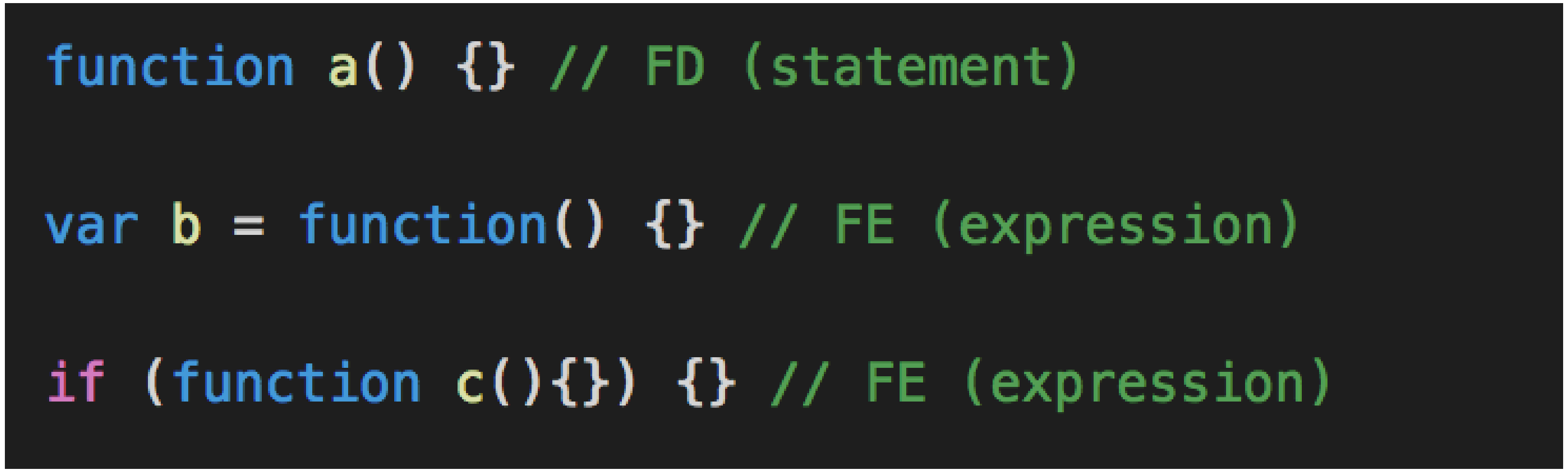


```
function x(a, b) {
    // var a = 1
    // var b = 2
    c = 3 // TDZ -> ReferenceError
    let c
    return a + b + c
}

console.log(x(1, 2)) // 6
```

Function

$$\begin{tikzcd}\textit{Function} \arrow[r, bend left] & \textit{Declaration}\end{tikzcd}$$

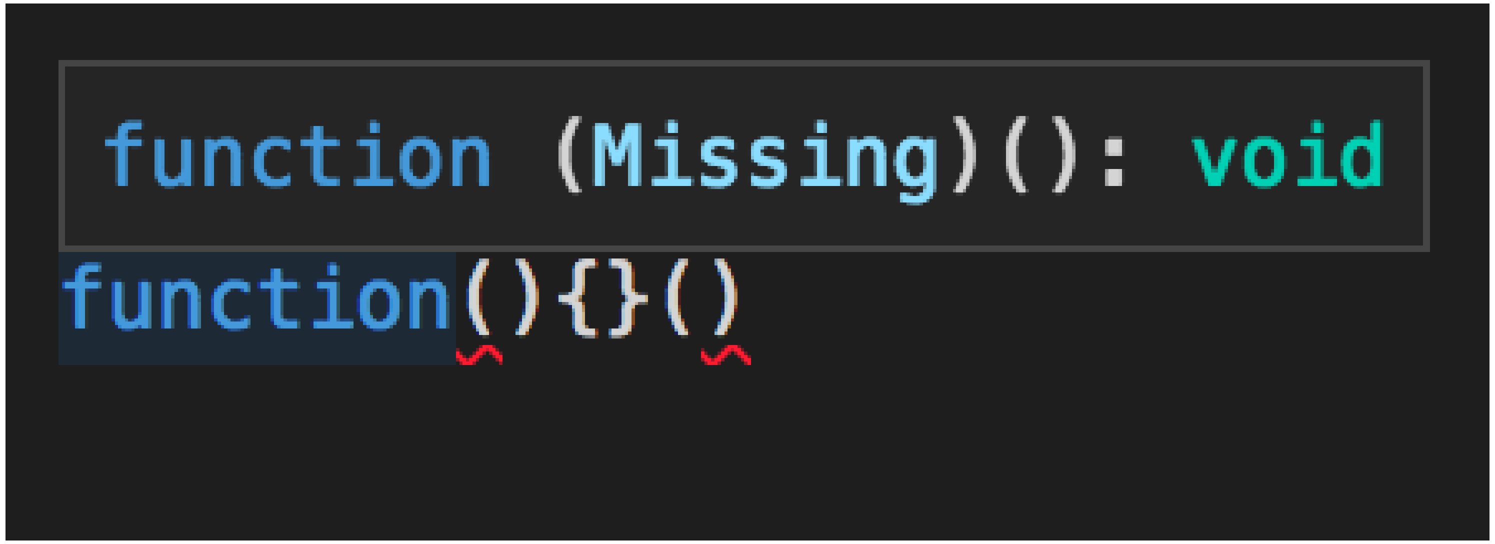
$$\begin{tikzcd} & \textit{Declaration} \\ \textit{Function} \arrow[ur] \arrow[dr] & \\ & \textit{Expression} \end{tikzcd}$$



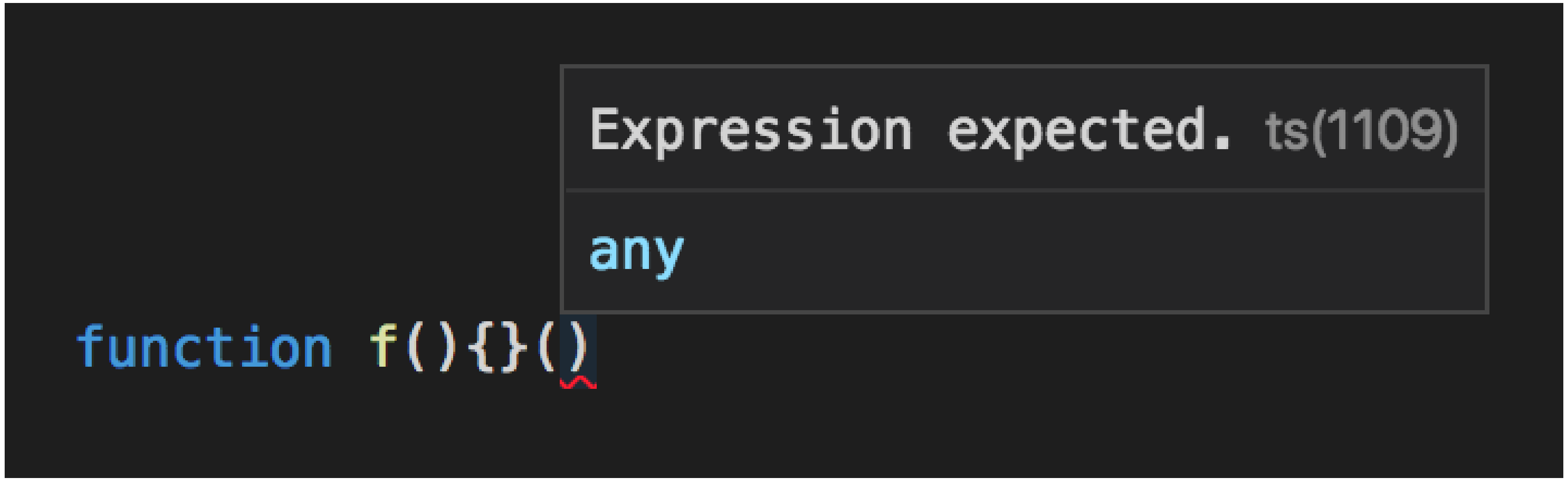
```
function a() {} // FD (statement)

var b = function() {} // FE (expression)

if (function c(){}) {} // FE (expression)
```

$$\begin{tikzcd}
 & \text{Declaration} \\
\text{Function} \arrow[ur] \arrow[dr] & \\
 & \text{Expression}
\end{tikzcd}$$

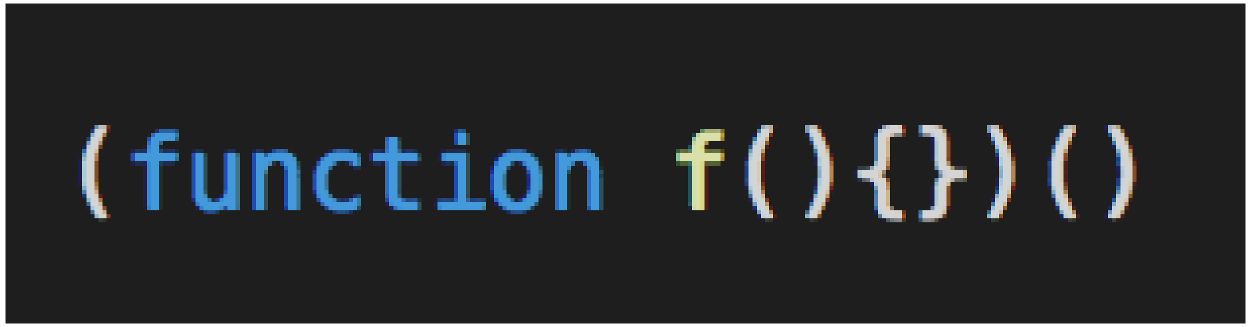
$$\begin{tikzcd}
 & & \text{Immediately Invoked} \\
 & \text{Declaration} & \\
\text{Function} \arrow[ur] \arrow[r] & \text{Expression} \arrow[uur] &
\end{tikzcd}$$



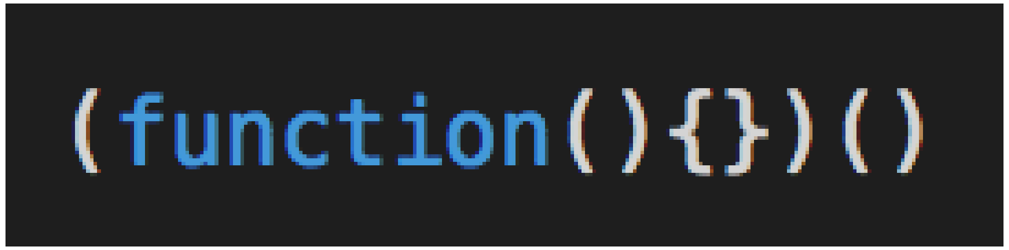
```
function (Missing)(): void
function(){}()
```



```
Expression expected. ts(1109)
any
function f(){}()
```



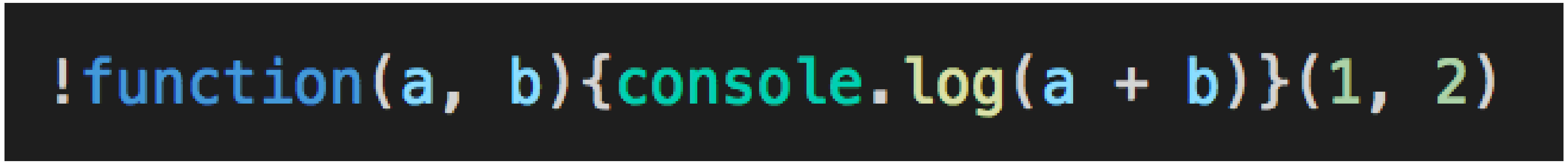
```
(function f(){})()
```



```
(function(){})()
```

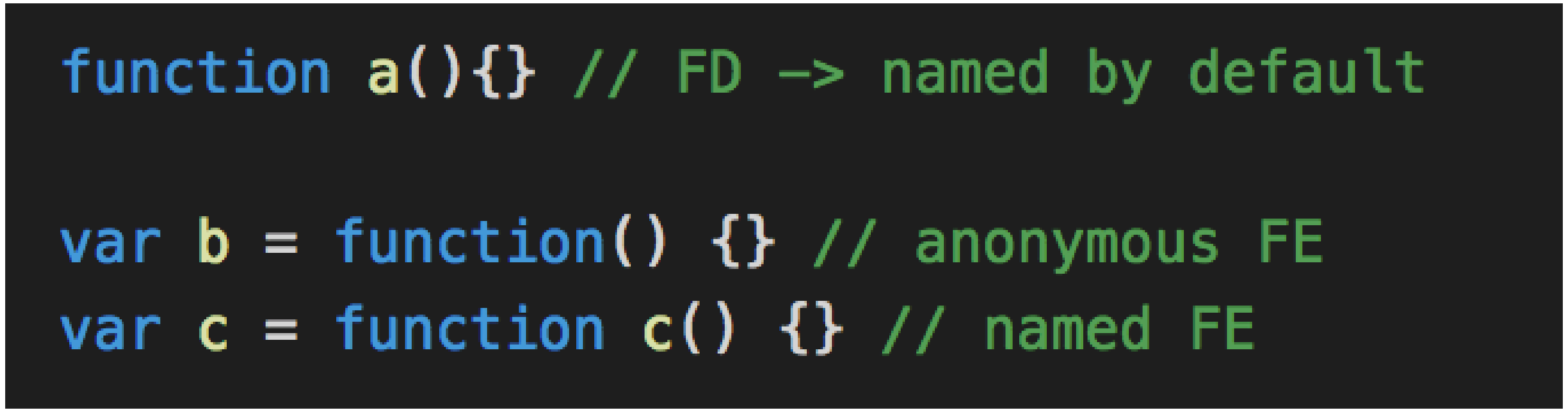
```
(function(a, b){console.log(a + b)})(1, 2)
```

```javascript
const three = (function(a, b){return a + b})(1, 2) // 3
```

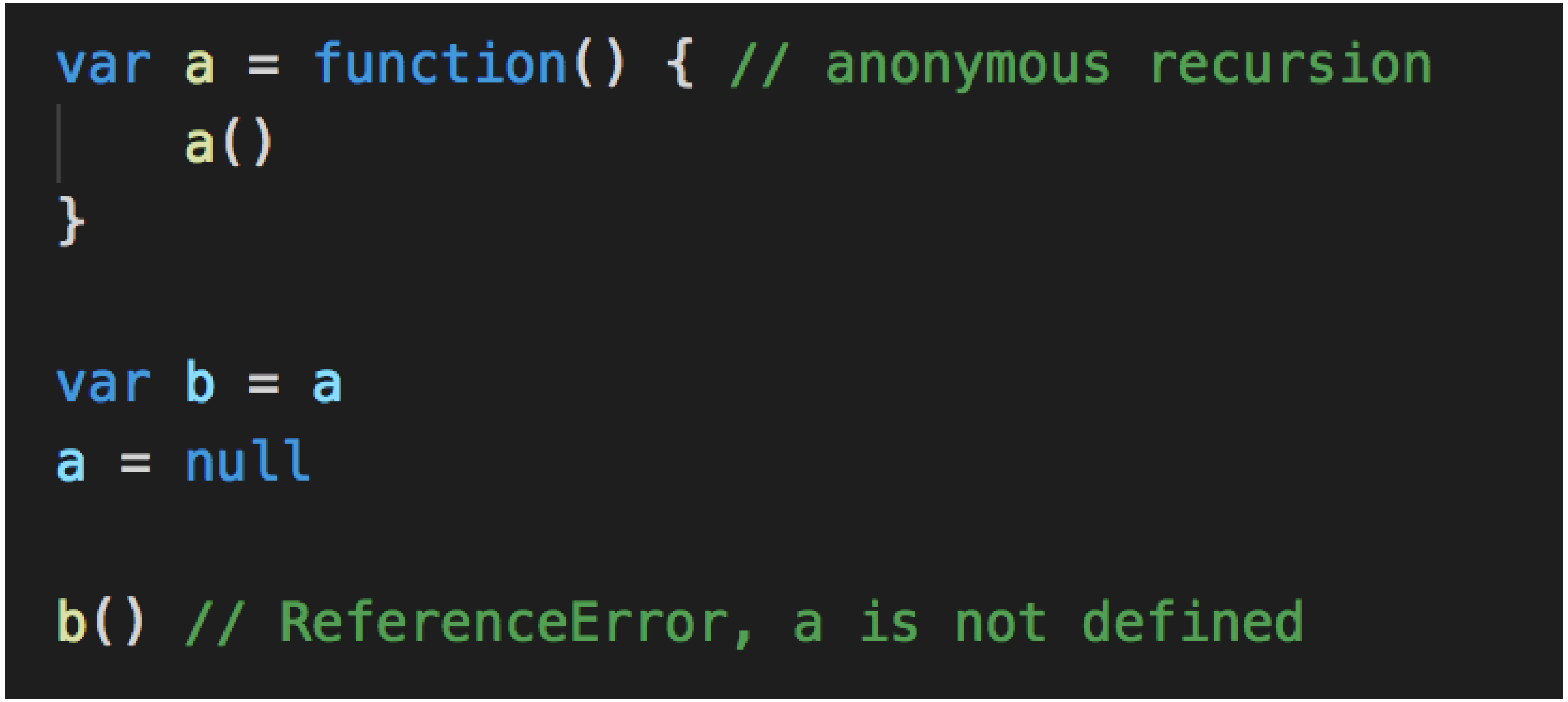


```
!function(a, b){console.log(a + b)}(1, 2)
```

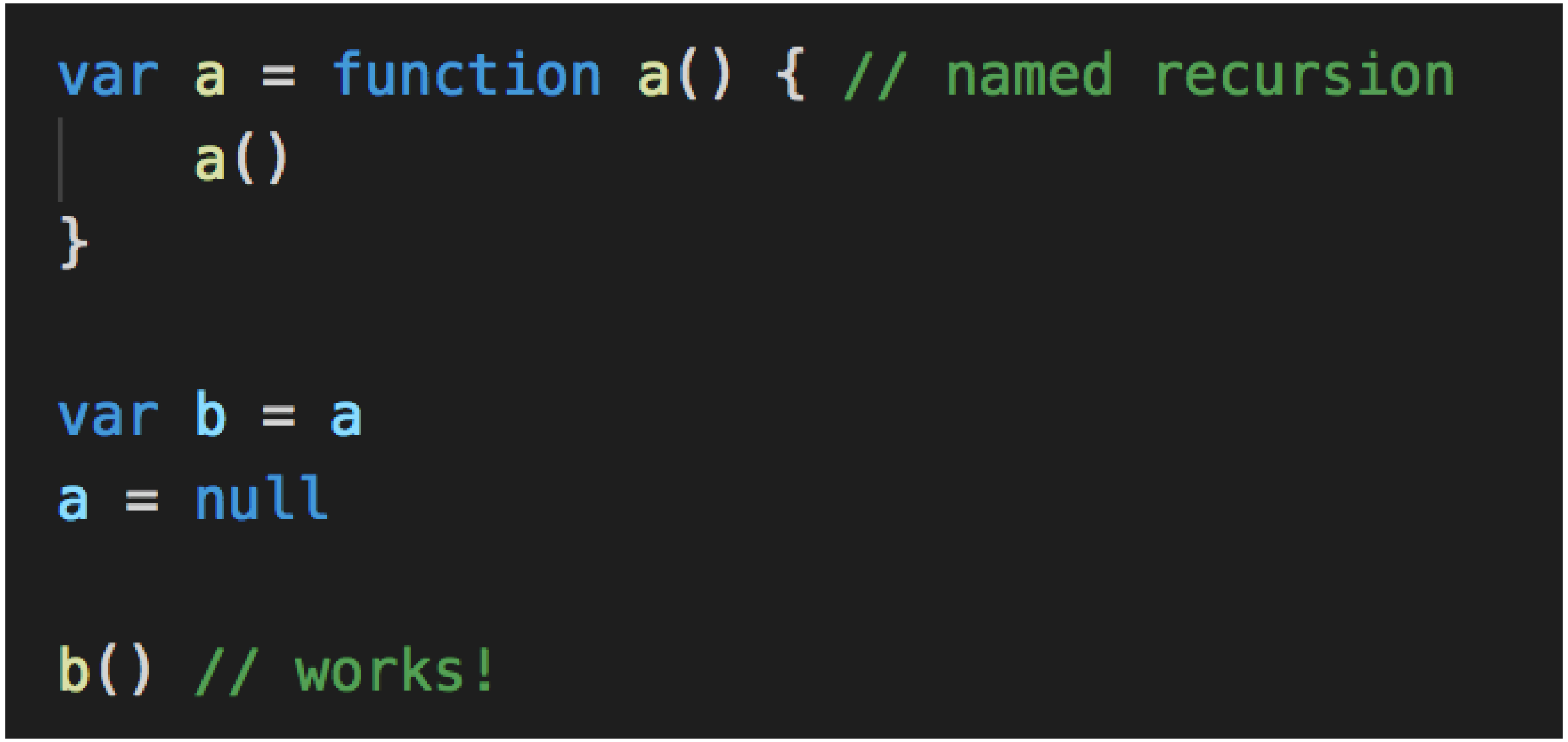
$$\begin{tikzcd}
 & & \textit{Immediately Invoked} \\
 & \textit{Declaration} & \textit{Named} \\
\textit{Function} \arrow[ur] \arrow[r] & \textit{Expression} \arrow[uur] \arrow[ur] &
\end{tikzcd}$$



```
function a(){} // FD -> named by default

var b = function() {} // anonymous FE
var c = function c() {} // named FE
```



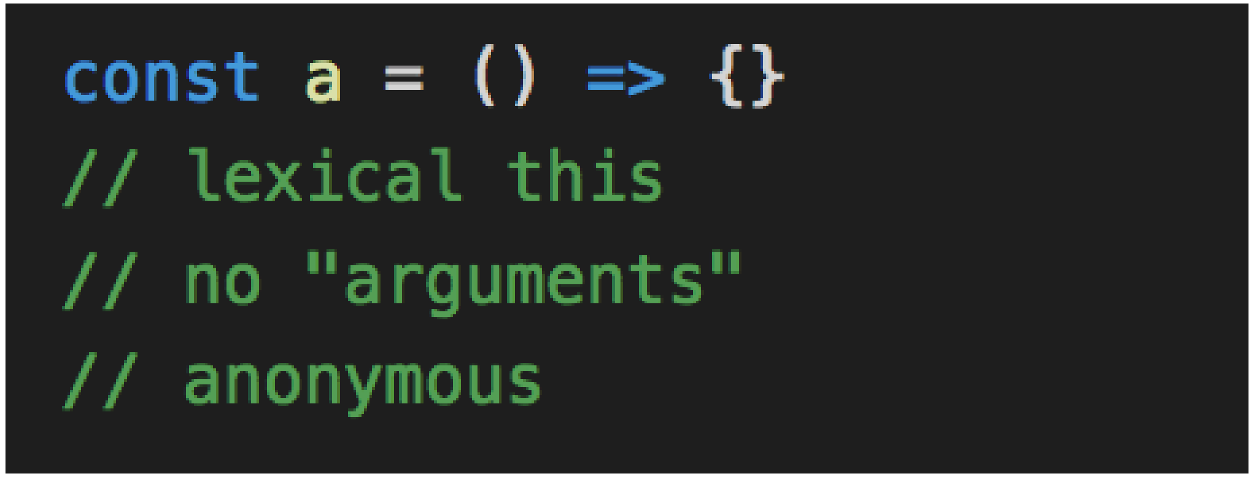
```
var a = function() { // anonymous recursion
    a()
}

var b = a
a = null

b() // ReferenceError, a is not defined
```



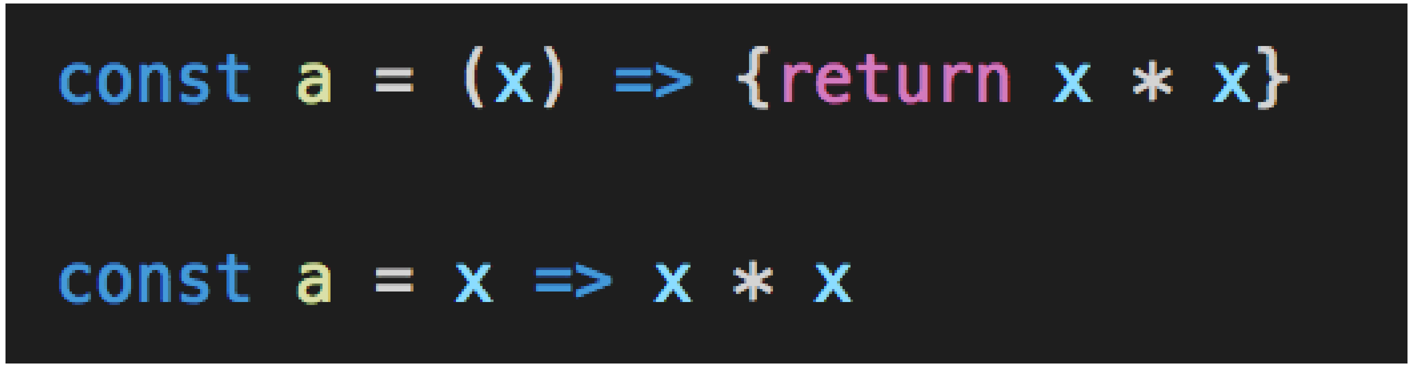
```
var a = function a() { // named recursion
    a()
}

var b = a
a = null

b() // works!
```

$$\begin{tikzcd}
 & & \textit{Immediately Invoked} \\
 & \textit{Declaration} & \textit{Named} \\
\textit{Function} \arrow[ur] \arrow[dr] & & \\
 & \textit{Expression} \arrow[uuur] \arrow[uur] &
\end{tikzcd}$$

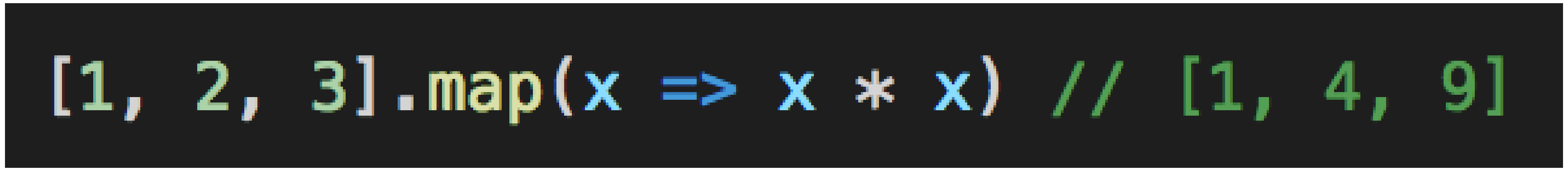
$$\begin{tikzcd}
 & & \textit{Immediately Invoked} \\
 & \textit{Declaration} & \textit{Named} \\
\textit{Function} \arrow[ur] \arrow[r] & \textit{Expression} \arrow[uur] \arrow[ur] \arrow[r] & \textit{Arrow}
\end{tikzcd}$$



```
const a = () => {}
// lexical this
// no "arguments"
// anonymous
```



```
const a = (x) => {return x * x}

const a = x => x * x
```

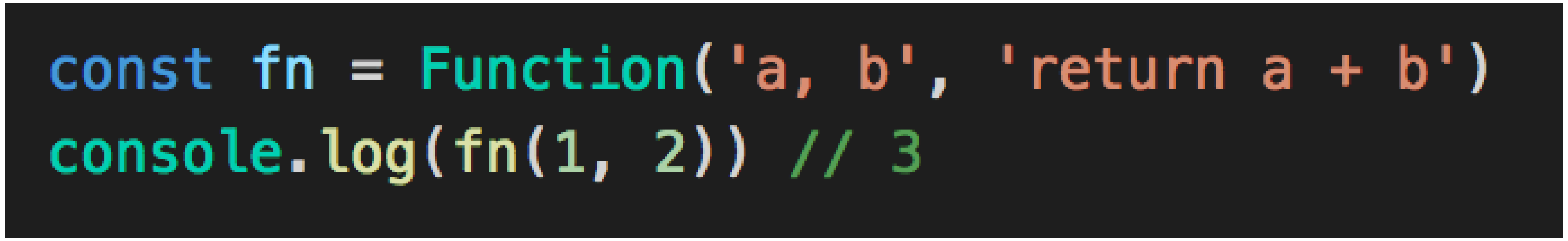


```
[1, 2, 3].map(x => x * x) // [1, 4, 9]
```

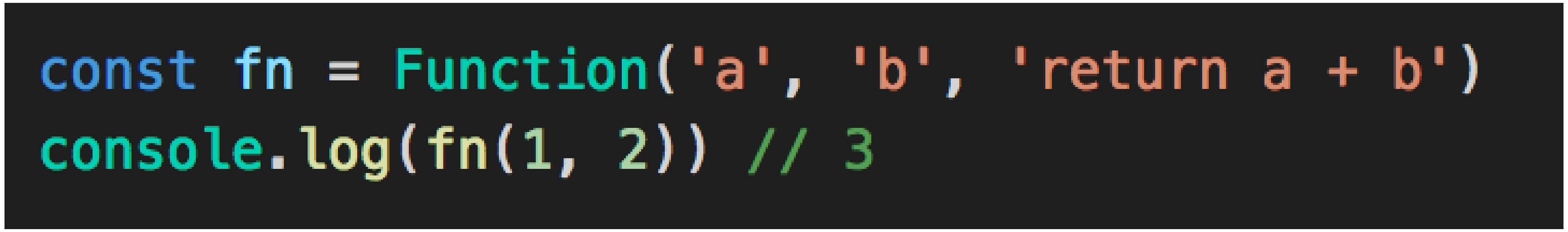
$$\begin{tikzcd}
 & & \text{Immediately Invoked} \\
 & \text{Declaration} & \text{Named} \\
\text{Function} \arrow[ur] \arrow[r] & \text{Expression} \arrow[uur] \arrow[ur] \arrow[r] & \text{Arrow}
\end{tikzcd}$$

$$\begin{tikzcd}
 & & \text{Immediately Invoked} \\
 & \text{Declaration} & \text{Named} \\
\text{Function} \arrow[ur] \arrow[dr] & & \\
 & \text{Expression} \arrow[uuur] \arrow[uur] \arrow[r] \arrow[dr] & \text{Arrow} \\
 & & \text{Anonymous}
\end{tikzcd}$$

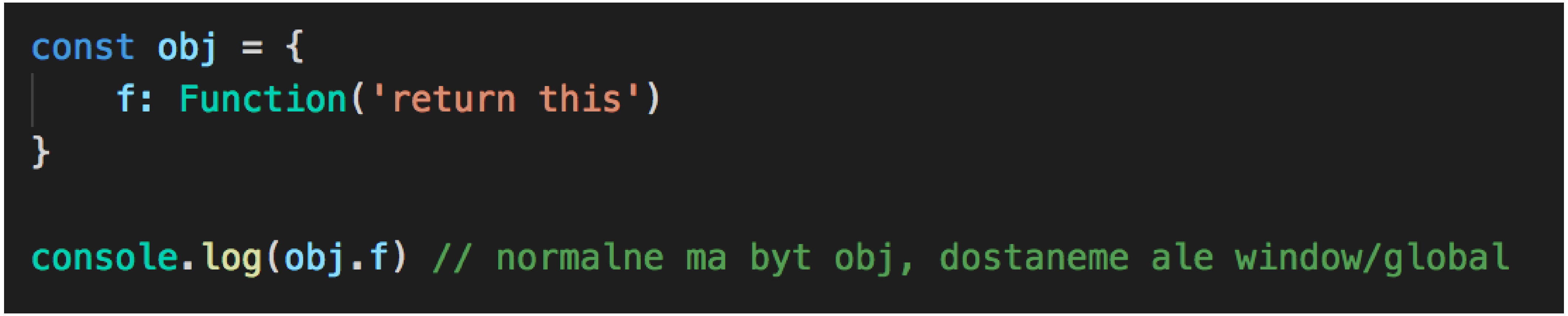
Function constructor



```
const fn = Function('a, b', 'return a + b')
console.log(fn(1, 2)) // 3
```



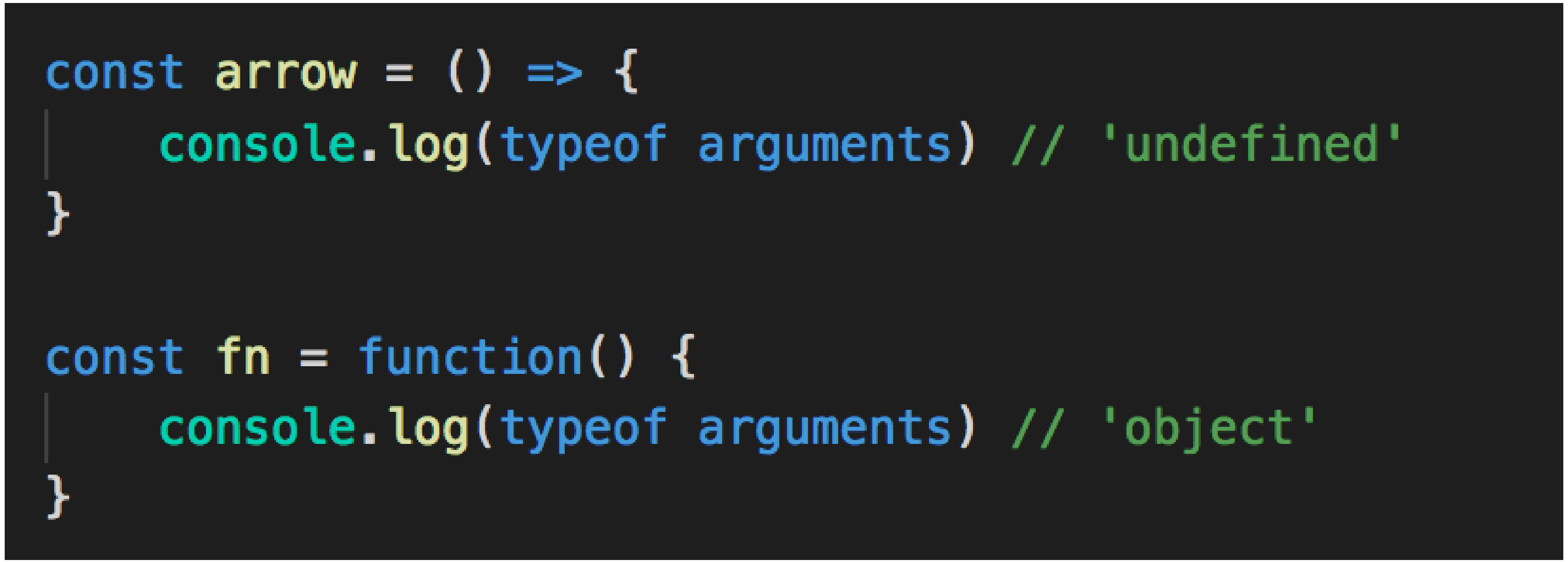
```javascript
const fn = Function('a', 'b', 'return a + b')
console.log(fn(1, 2)) // 3
```



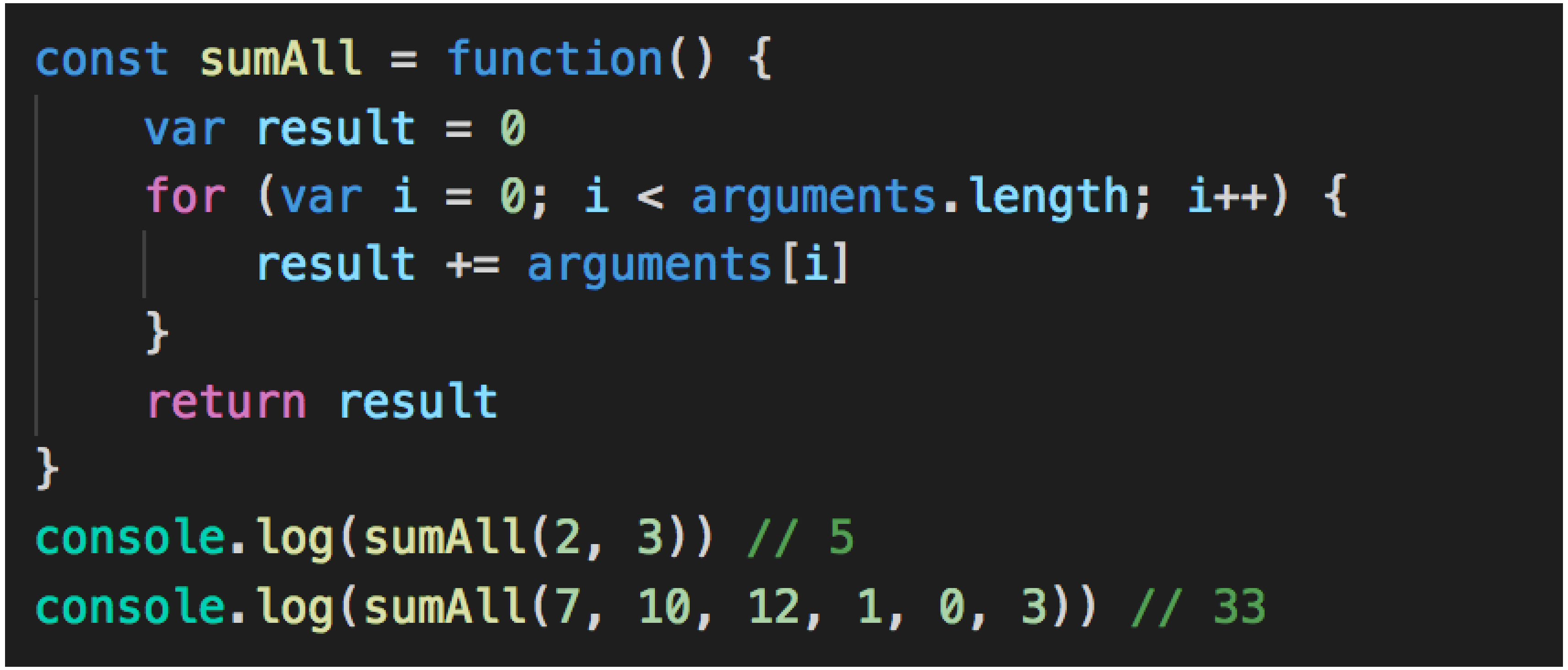
```
const obj = {
    f: Function('return this')
}

console.log(obj.f) // normalne ma byt obj, dostaneme ale window/global
```

argumenty

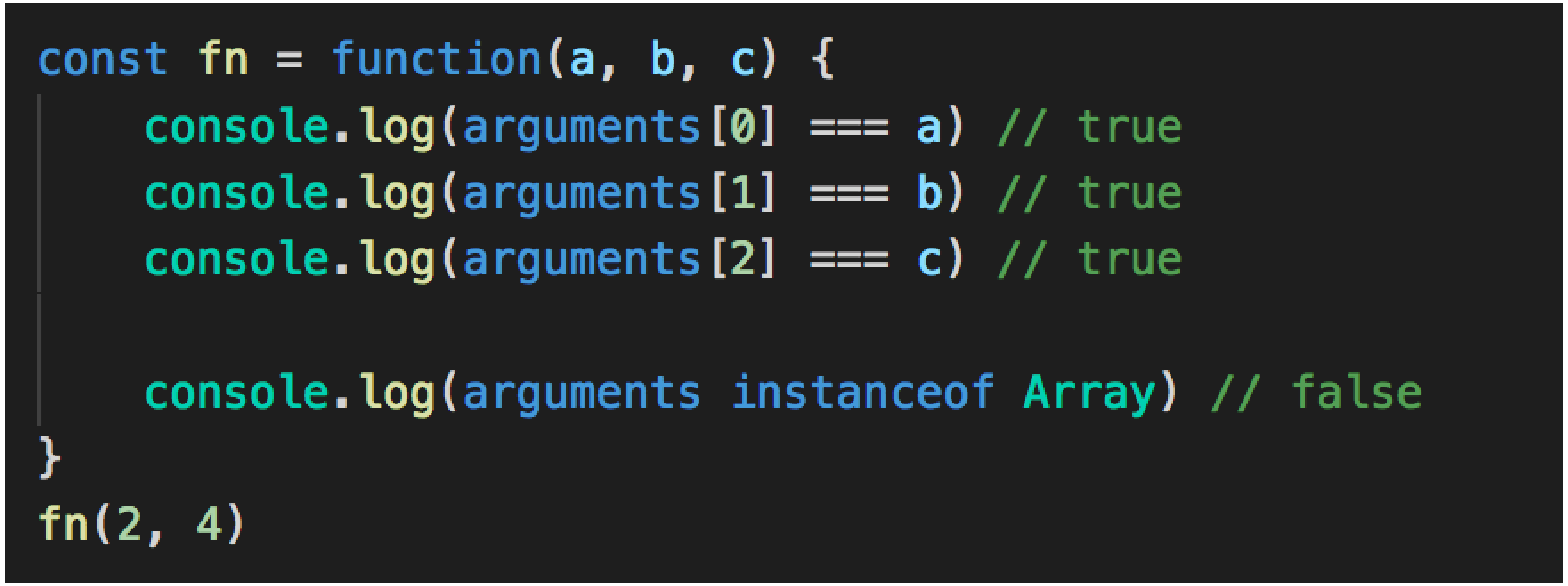
Array-like object

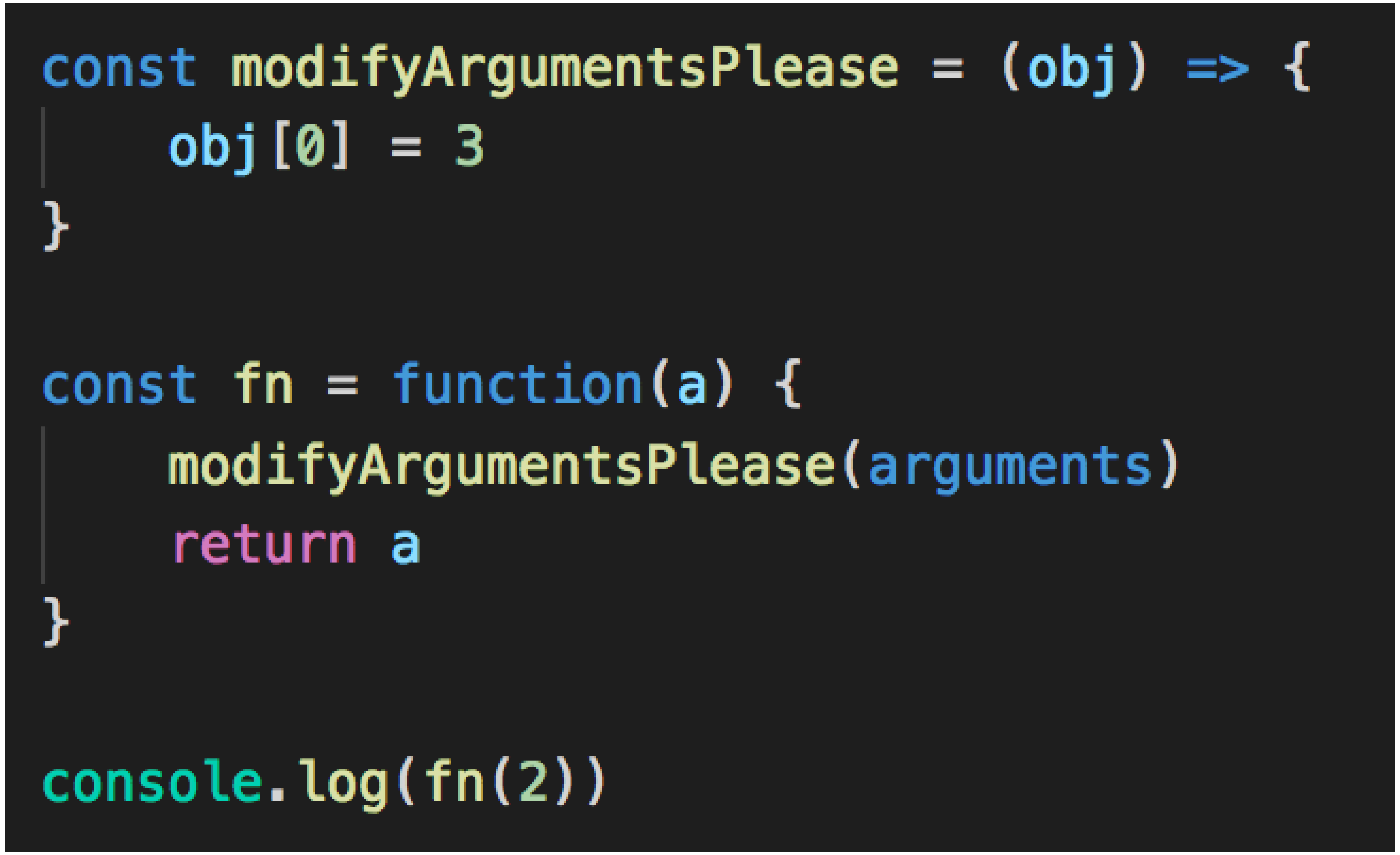


```
const arrow = () => {
    console.log(typeof arguments) // 'undefined'
}

const fn = function() {
    console.log(typeof arguments) // 'object'
}
```

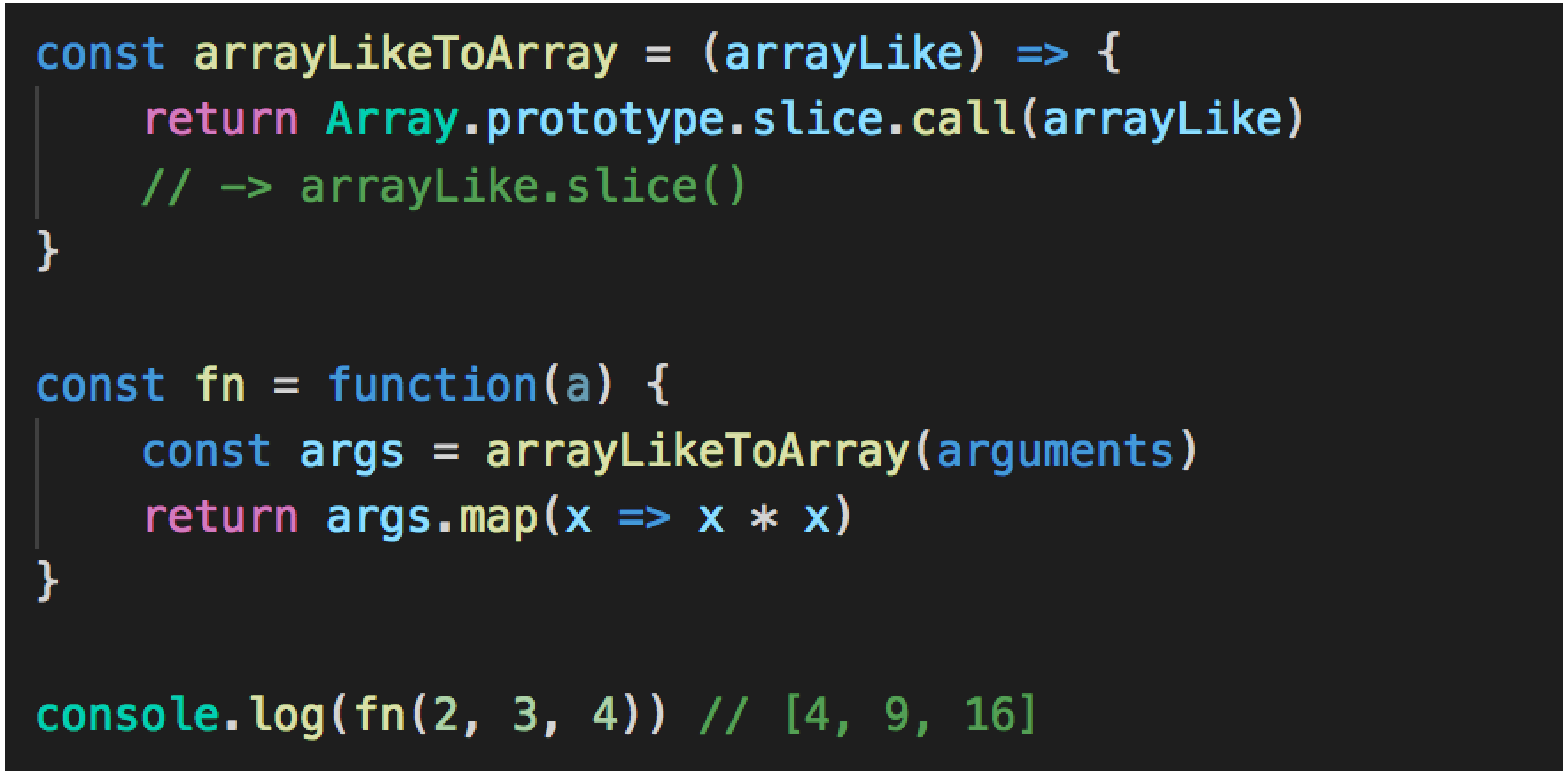


```
const sumAll = function() {
    var result = 0
    for (var i = 0; i < arguments.length; i++) {
        result += arguments[i]
    }
    return result
}
console.log(sumAll(2, 3)) // 5
console.log(sumAll(7, 10, 12, 1, 0, 3)) // 33
```

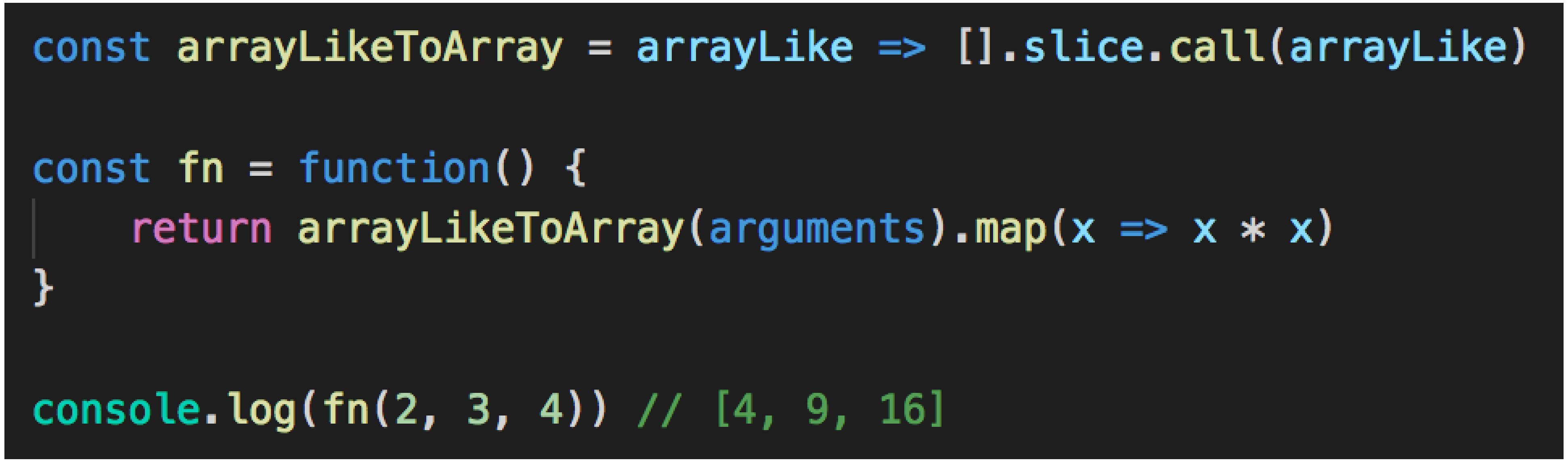

```
const fn = function(a, b, c) {
    console.log(arguments[0] === a) // true
    console.log(arguments[1] === b) // true
    console.log(arguments[2] === c) // true

    console.log(arguments instanceof Array) // false
}
fn(2, 4)
```



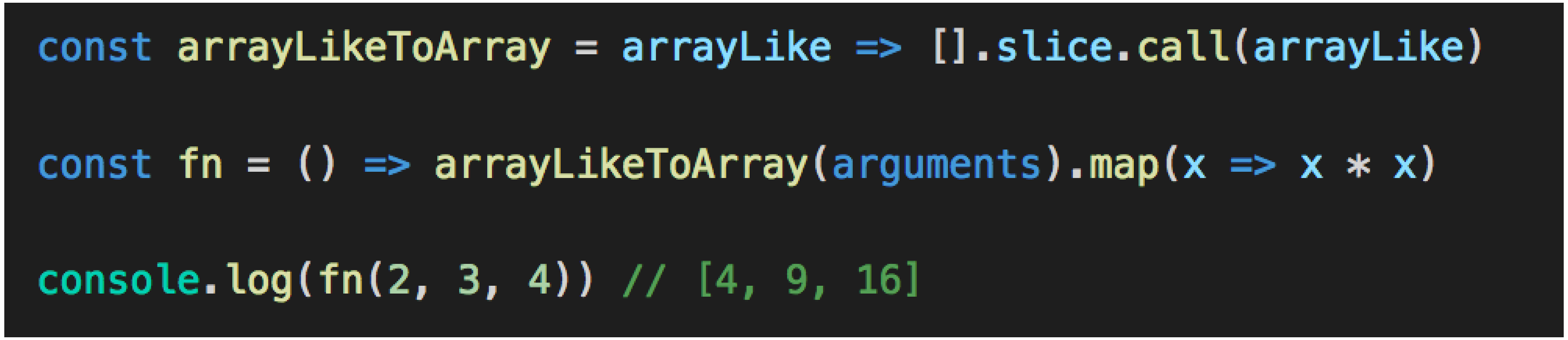
```
const modifyArgumentsPlease = (obj) => {
    obj[0] = 3
}

const fn = function(a) {
    modifyArgumentsPlease(arguments)
    return a
}

console.log(fn(2))
```



```
const arrayLikeToArray = (arrayLike) => {
    return Array.prototype.slice.call(arrayLike)
    // -> arrayLike.slice()
}

const fn = function(a) {
    const args = arrayLikeToArray(arguments)
    return args.map(x => x * x)
}

console.log(fn(2, 3, 4)) // [4, 9, 16]
```



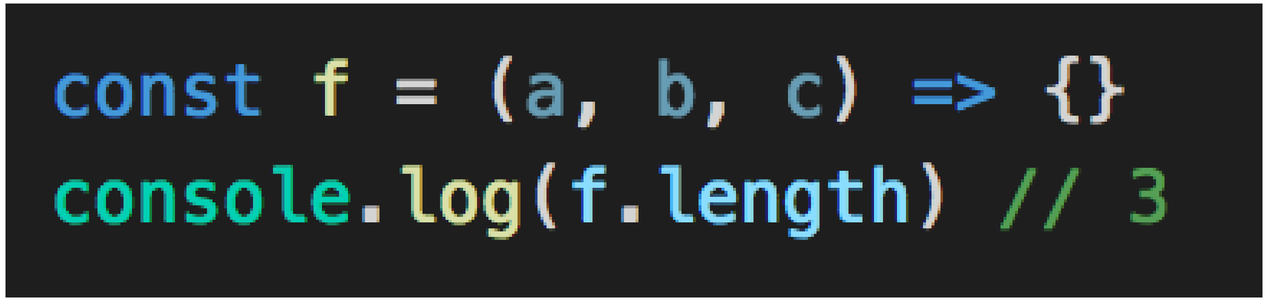
```javascript
const arrayLikeToArray = arrayLike => [].slice.call(arrayLike)

const fn = function() {
    return arrayLikeToArray(arguments).map(x => x * x)
}

console.log(fn(2, 3, 4)) // [4, 9, 16]
```



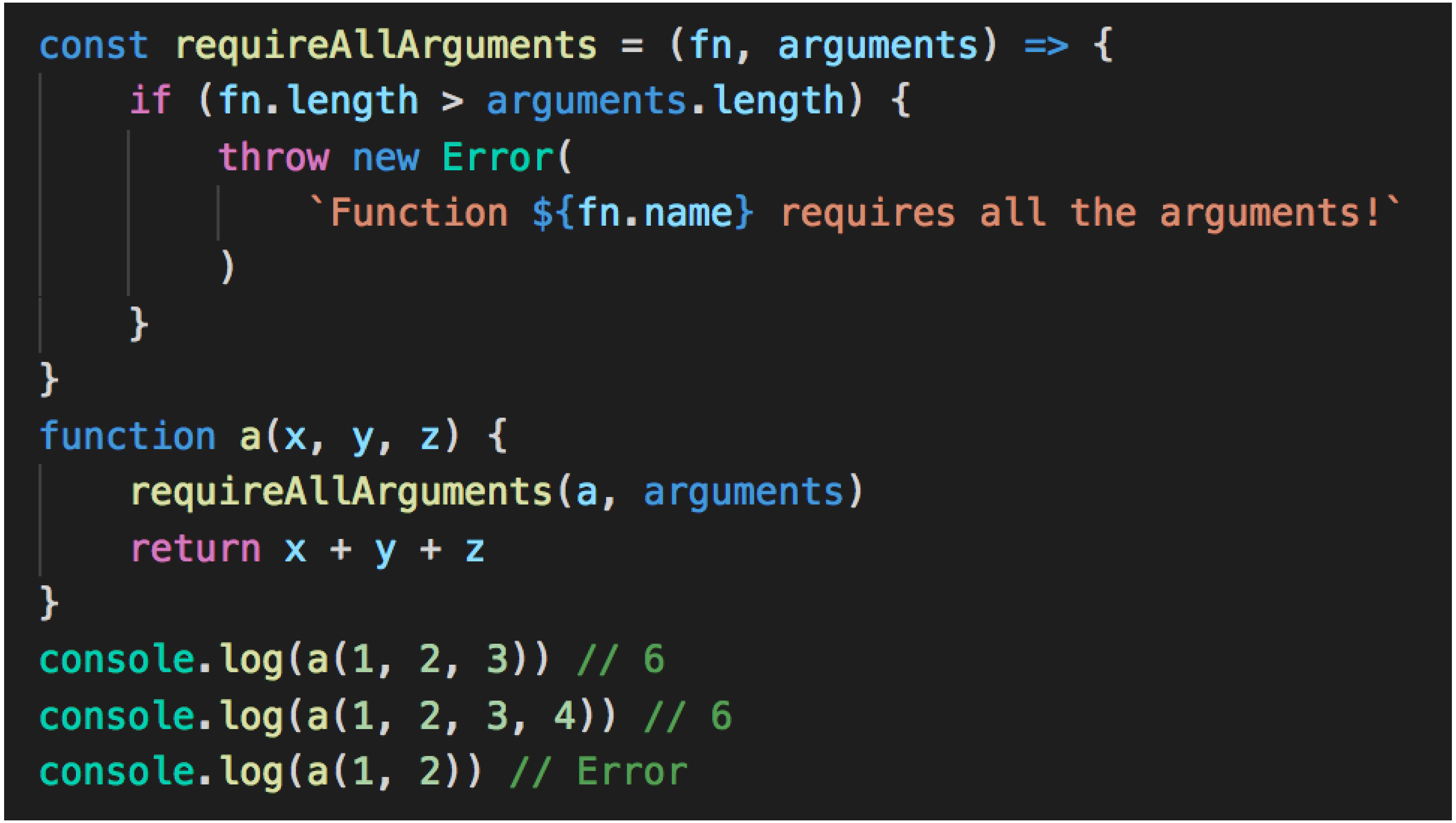
```
const arrayLikeToArray = arrayLike => [].slice.call(arrayLike)

const fn = () => arrayLikeToArray(arguments).map(x => x * x)

console.log(fn(2, 3, 4)) // [4, 9, 16]
```

length

počet pojmenovaných argumentů



```js
const f = (a, b, c) => {}
console.log(f.length) // 3
```



```
const requireAllArguments = (fn, arguments) => {
    if (fn.length > arguments.length) {
        throw new Error(
            `Function ${fn.name} requires all the arguments!`
        )
    }
}
function a(x, y, z) {
    requireAllArguments(a, arguments)
    return x + y + z
}
console.log(a(1, 2, 3)) // 6
console.log(a(1, 2, 3, 4)) // 6
console.log(a(1, 2)) // Error
```

name

anonymní funkce = funkce, která nemá název

```
const f = function() {}
```

je to pořád anonymní funkce

$\downarrow$

```
const f = function() {}
```

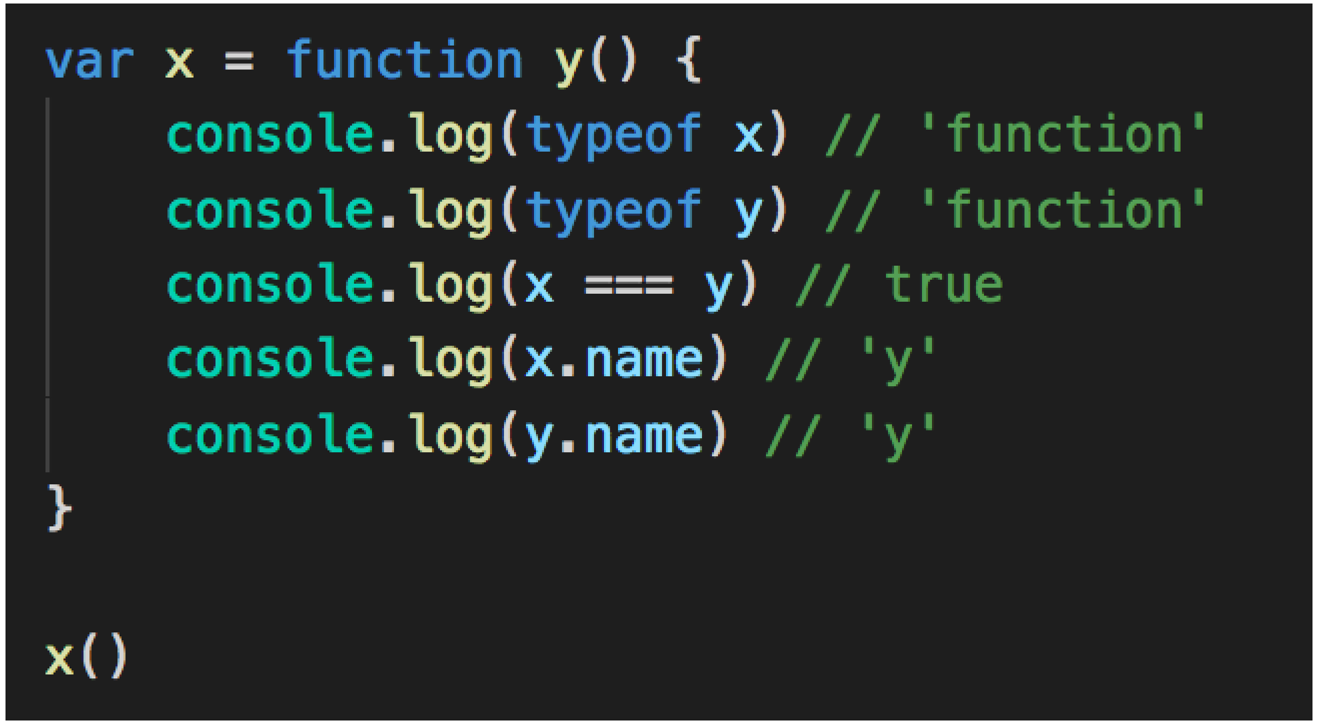
$\uparrow$

přiřazení neznamená pojmenování

pojmenovaná funkce

$\downarrow$

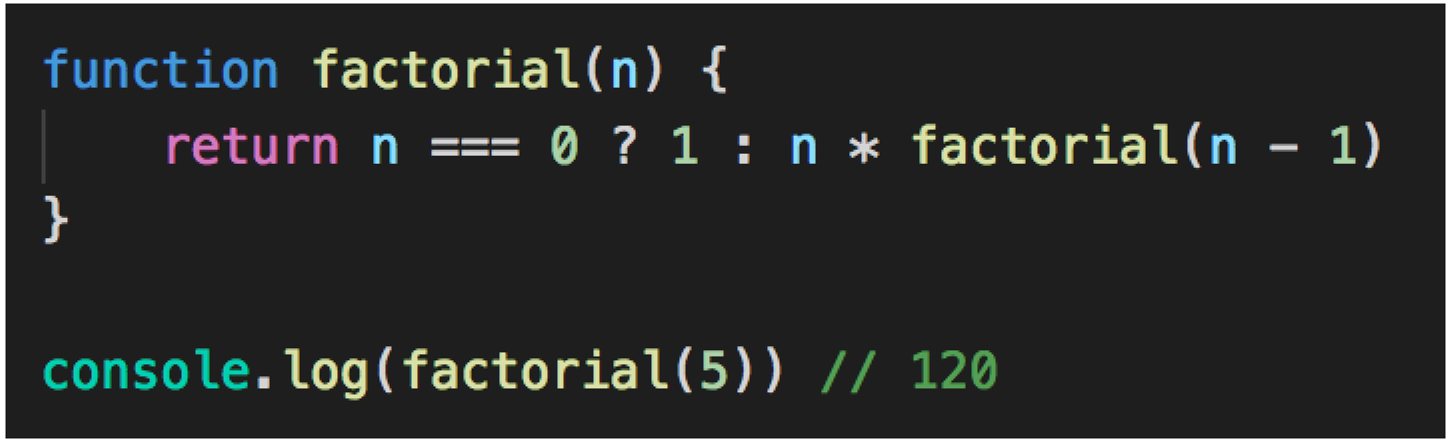
const f = function f() {}



```
var x = function y() {
    console.log(typeof x) // 'function'
    console.log(typeof y) // 'function'
    console.log(x === y) // true
    console.log(x.name) // 'y'
    console.log(y.name) // 'y'
}

x()
```

Rekurze

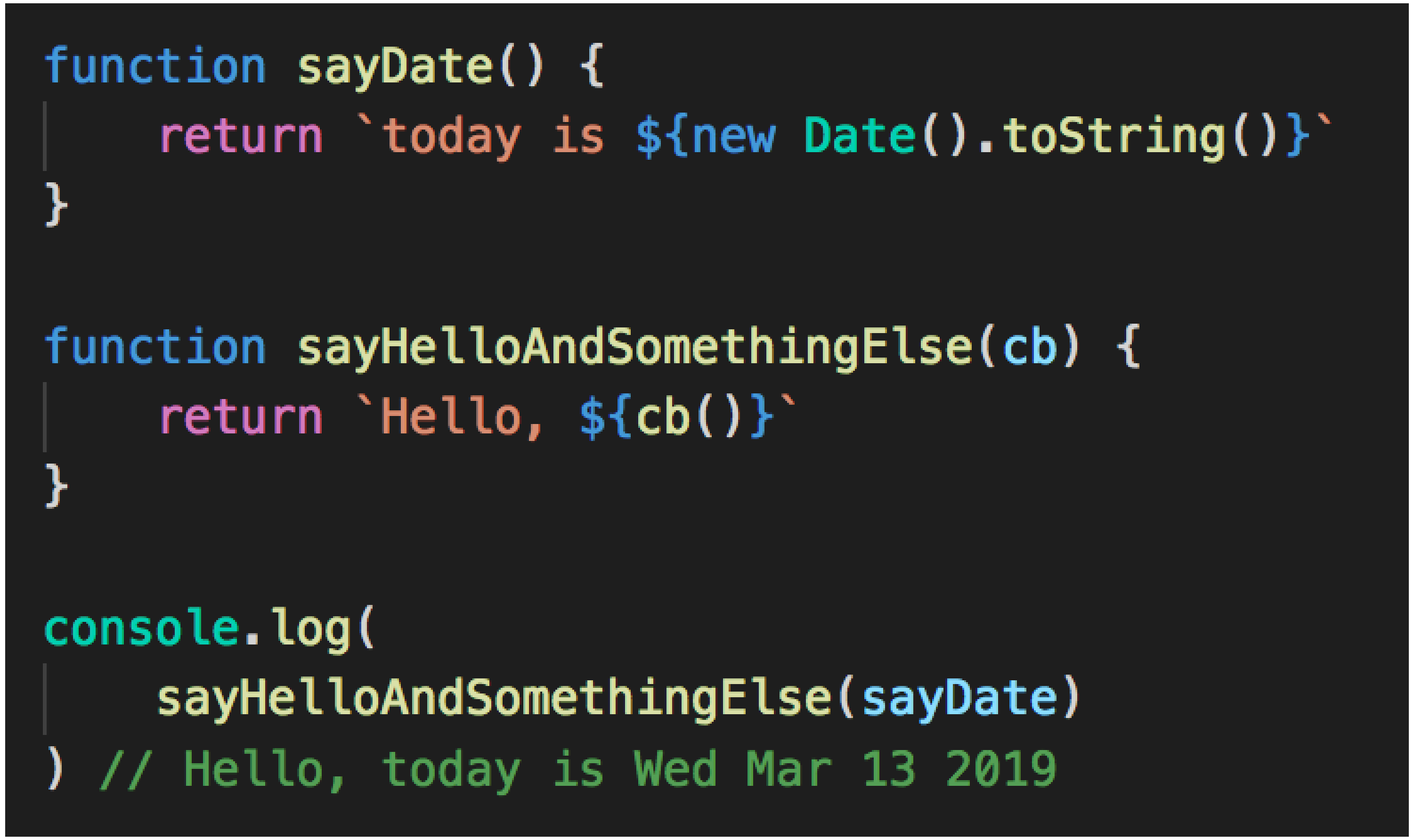
funkce znovu volána dříve, než je dokončeno její předchozí volání



```
function factorial(n) {
    return n === 0 ? 1 : n * factorial(n - 1)
}

console.log(factorial(5)) // 120
```

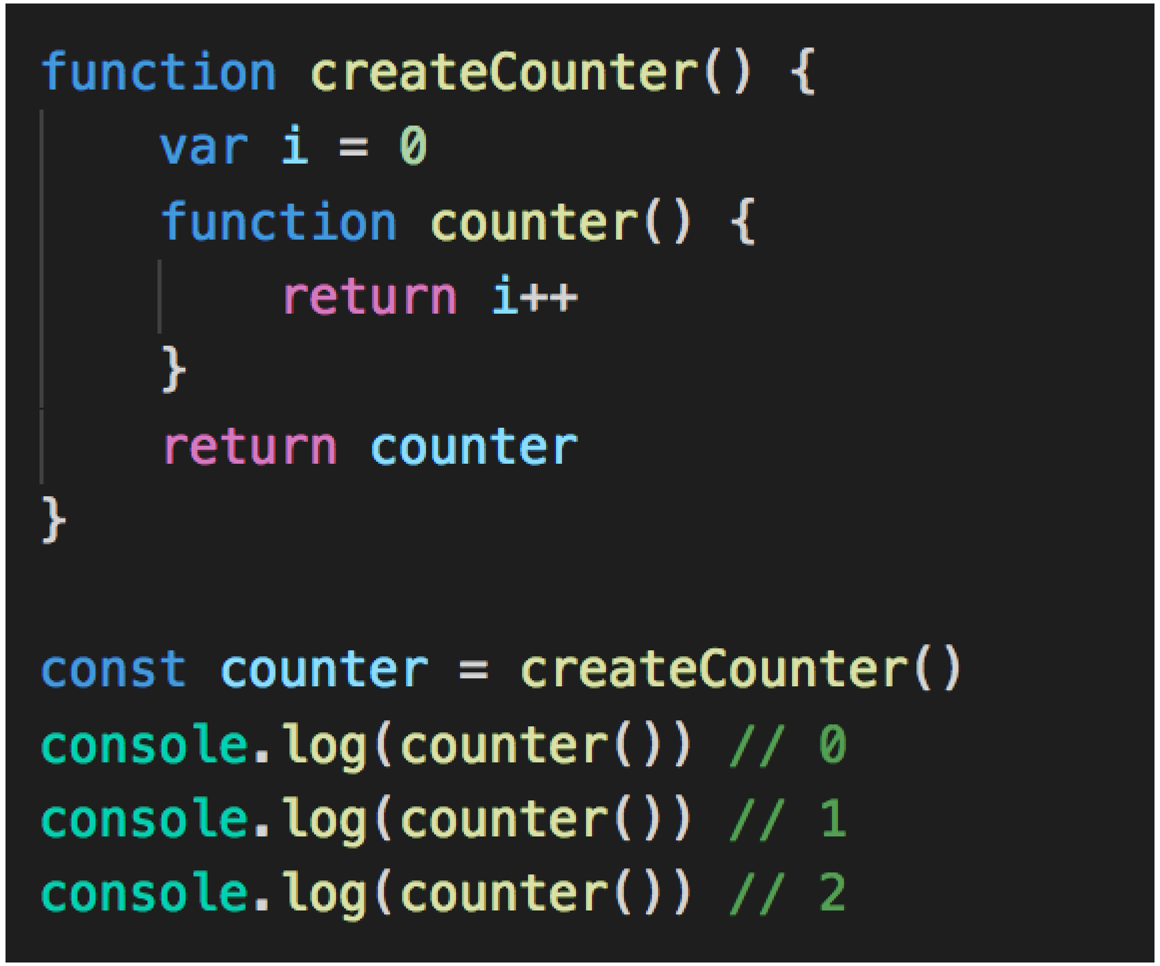
Callback

funkce jako hodnota, která je volána později

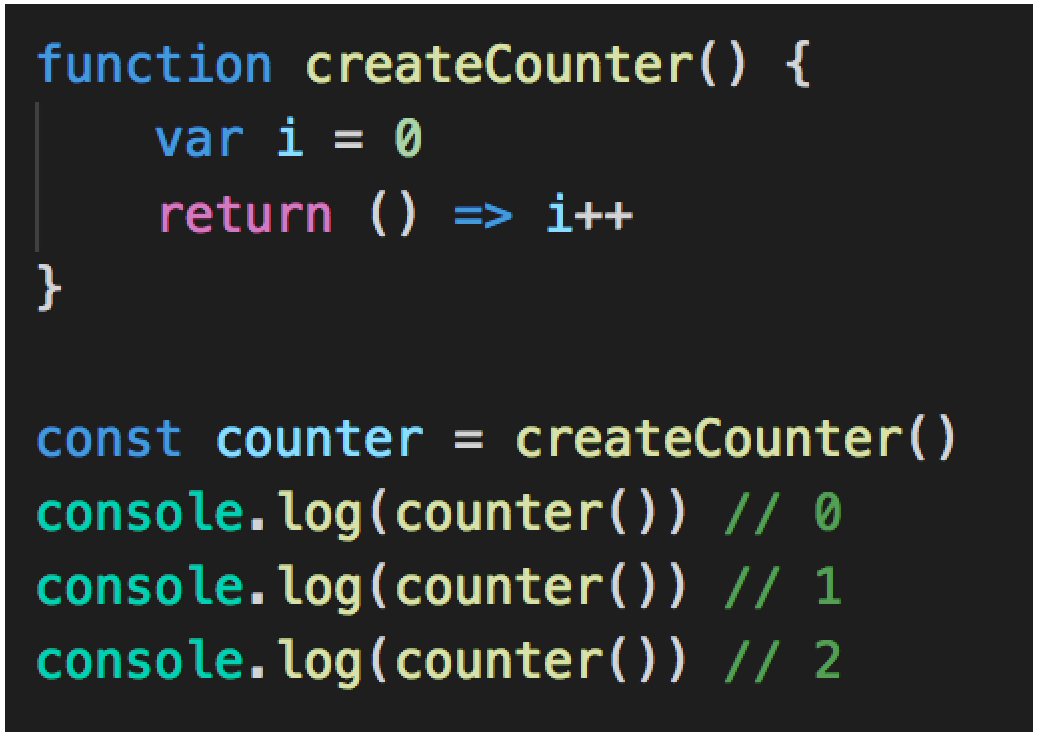


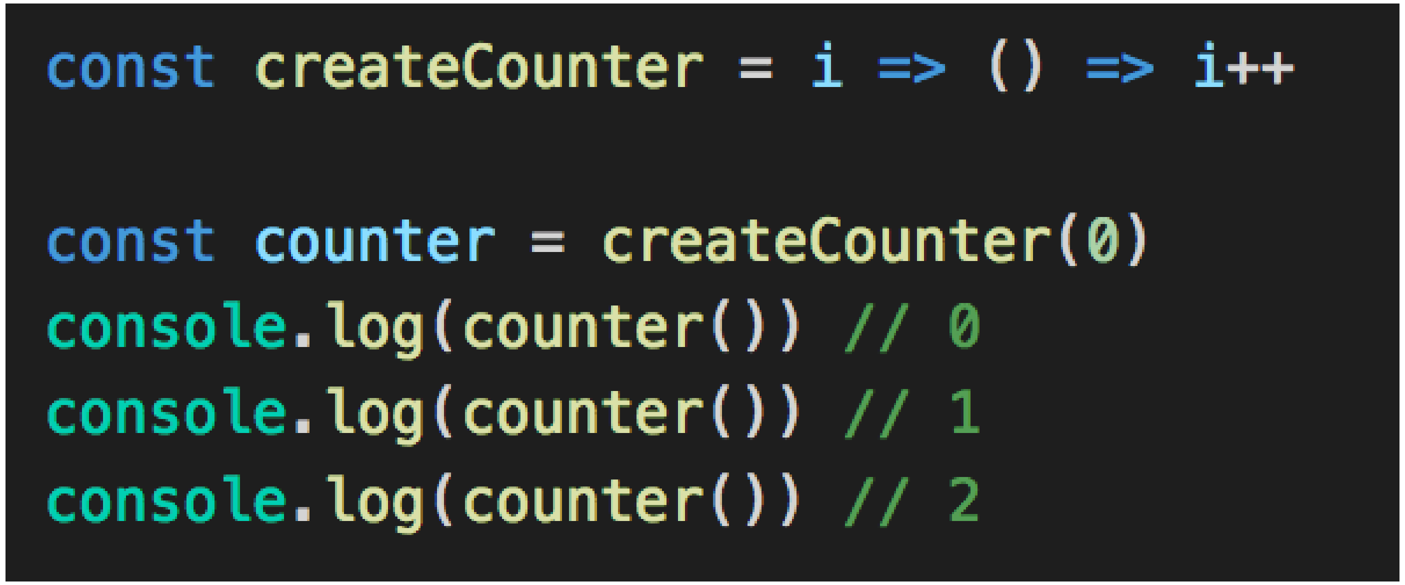
```
function sayDate() {
    return `today is ${new Date().toString()}`
}

function sayHelloAndSomethingElse(cb) {
    return `Hello, ${cb()}`
}

console.log(
    sayHelloAndSomethingElse(sayDate)
) // Hello, today is Wed Mar 13 2019
```

Closure



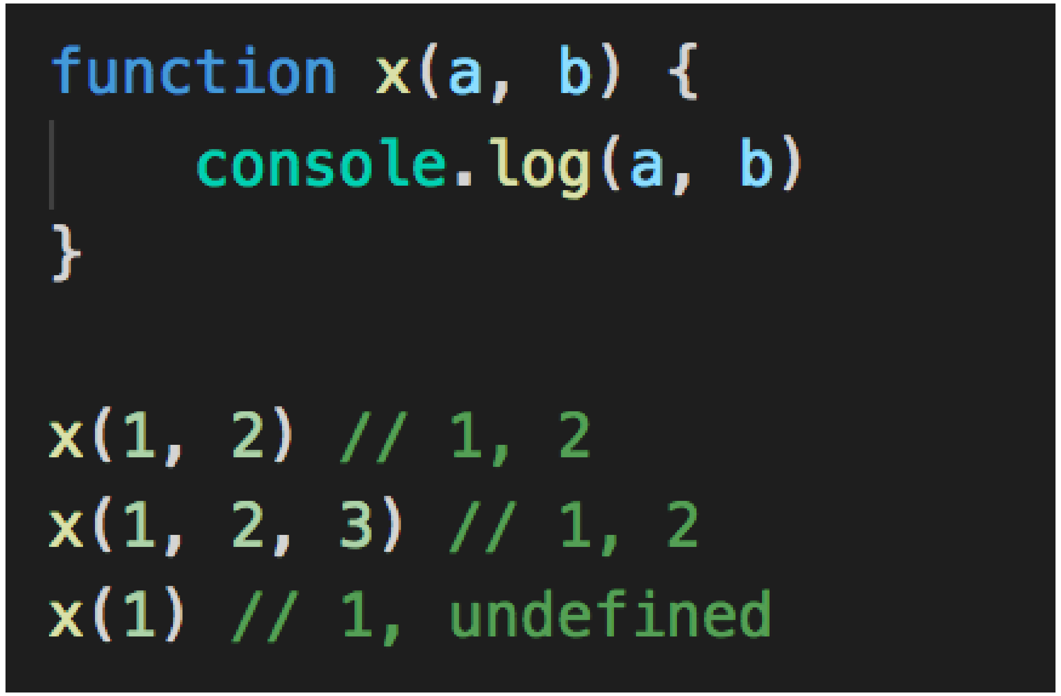
```
function createCounter() {
    var i = 0
    function counter() {
        return i++
    }
    return counter
}

const counter = createCounter()
console.log(counter()) // 0
console.log(counter()) // 1
console.log(counter()) // 2
```


```
function createCounter() {
    var i = 0
    return () => i++
}

const counter = createCounter()
console.log(counter()) // 0
console.log(counter()) // 1
console.log(counter()) // 2
```

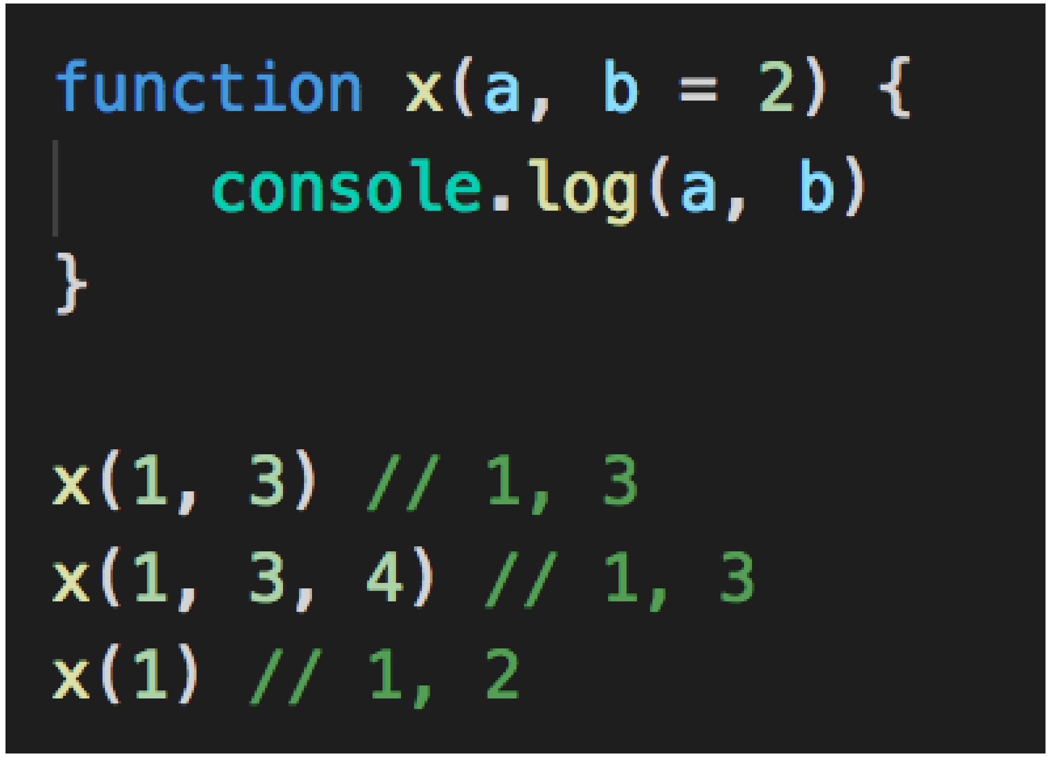


```javascript
const createCounter = i => () => i++

const counter = createCounter(0)
console.log(counter()) // 0
console.log(counter()) // 1
console.log(counter()) // 2
```

Výchozí parametry



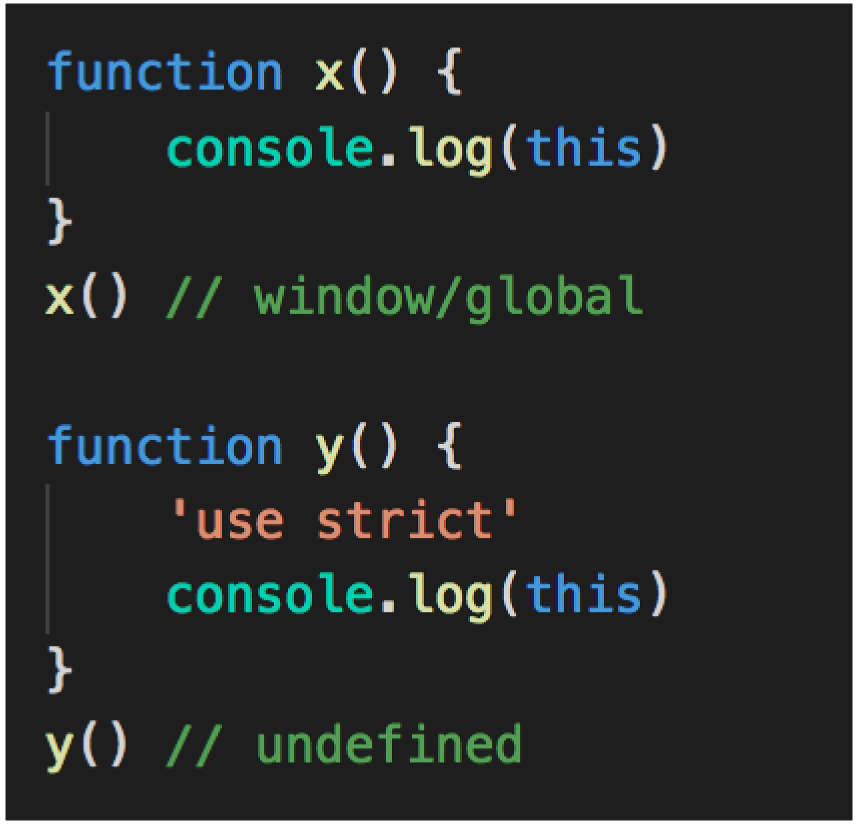
```
function x(a, b) {
    console.log(a, b)
}

x(1, 2) // 1, 2
x(1, 2, 3) // 1, 2
x(1) // 1, undefined
```



```
function x(a, b = 2) {
    console.log(a, b)
}

x(1, 3) // 1, 3
x(1, 3, 4) // 1, 3
x(1) // 1, 2
```

Context (this)

$$\begin{tikzcd}
\text{Context (this)} \arrow[r, bend right] & \text{strict \& non-strict}
\end{tikzcd}$$



```
function x() {
    console.log(this)
}
x() // window/global

function y() {
    'use strict'
    console.log(this)
}
y() // undefined
```

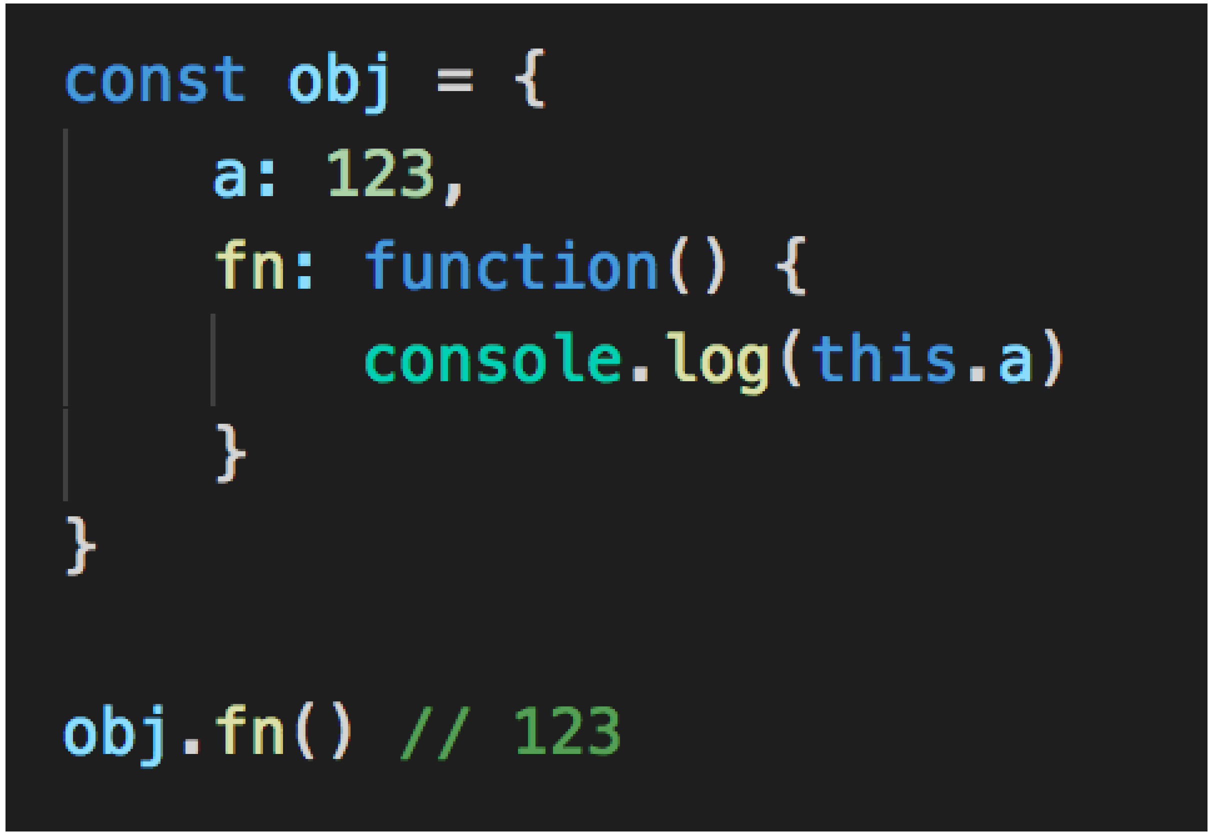
strict & non-strict

Context (this)

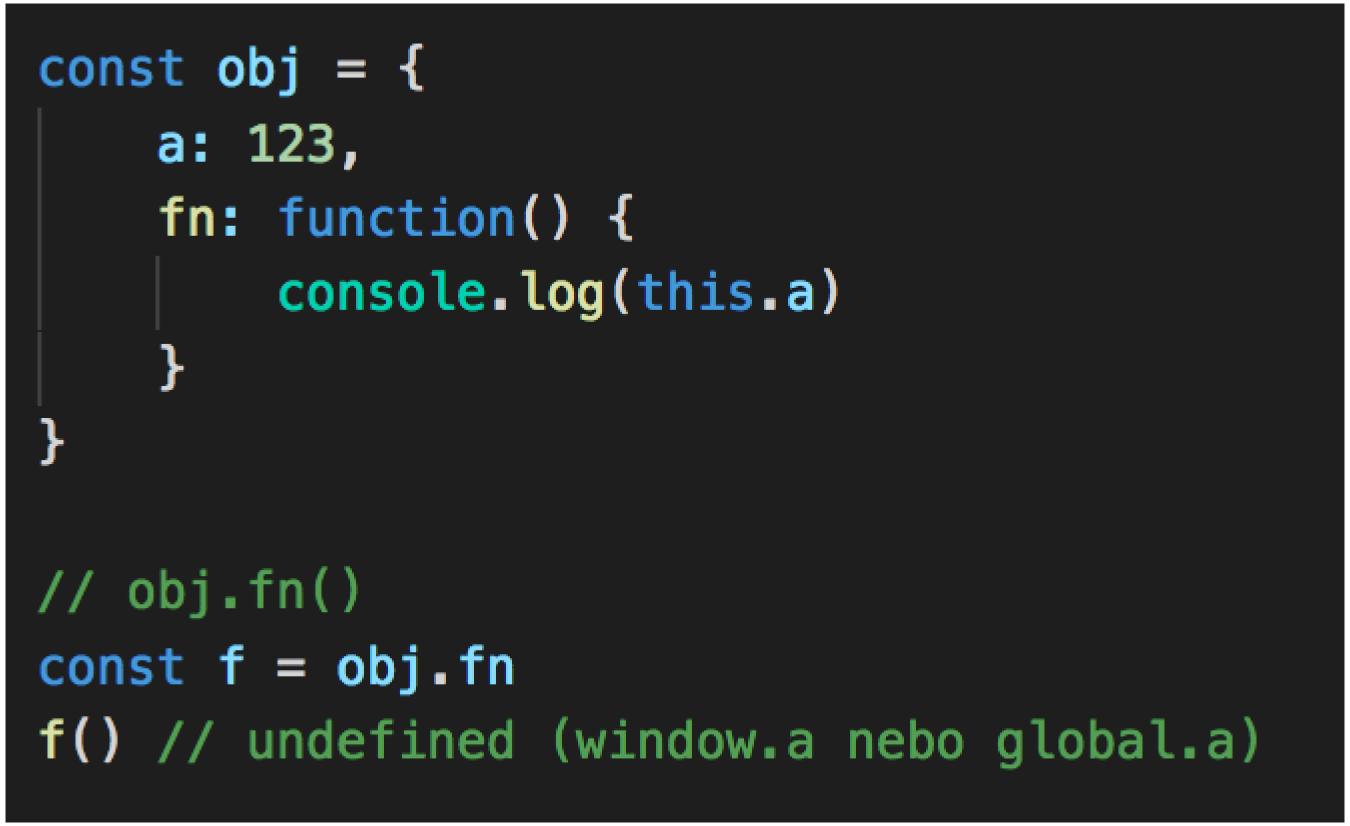
Context (this)

strict & non-strict

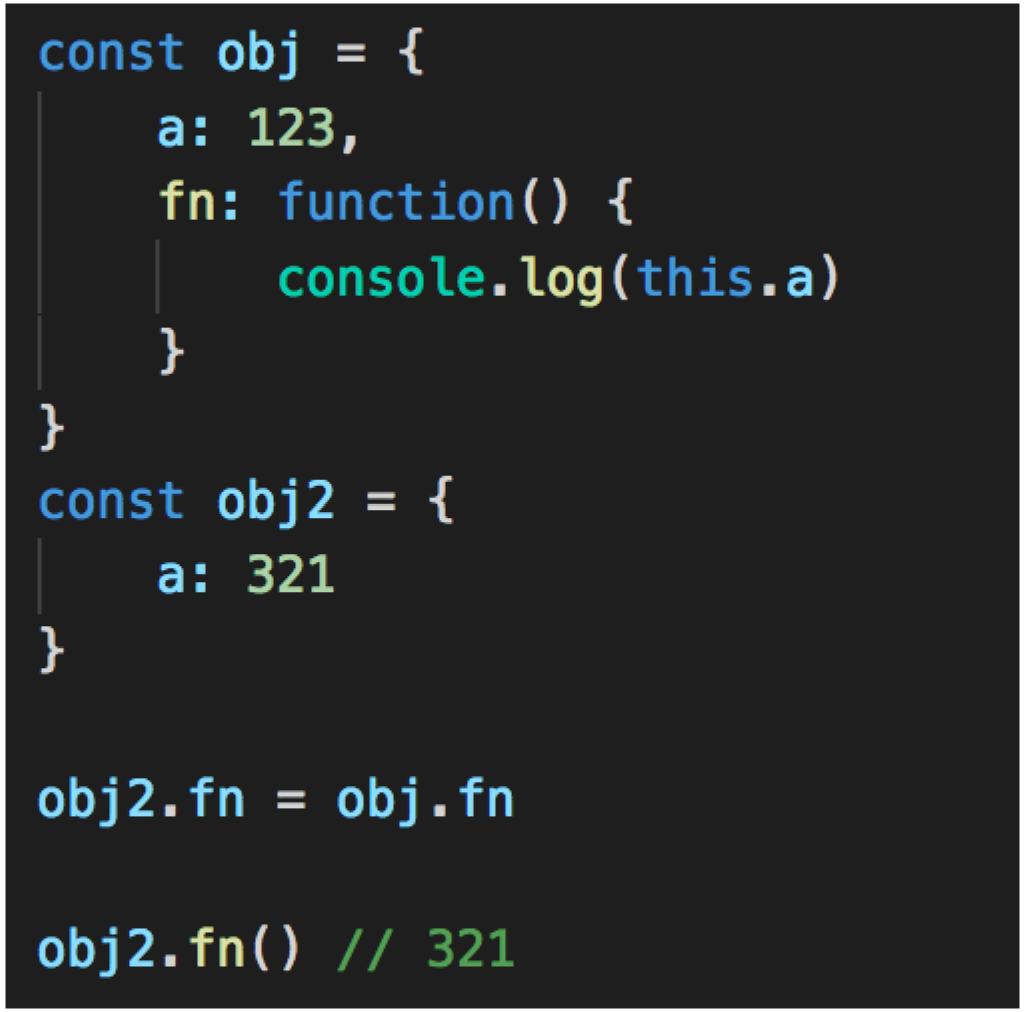
volání metody



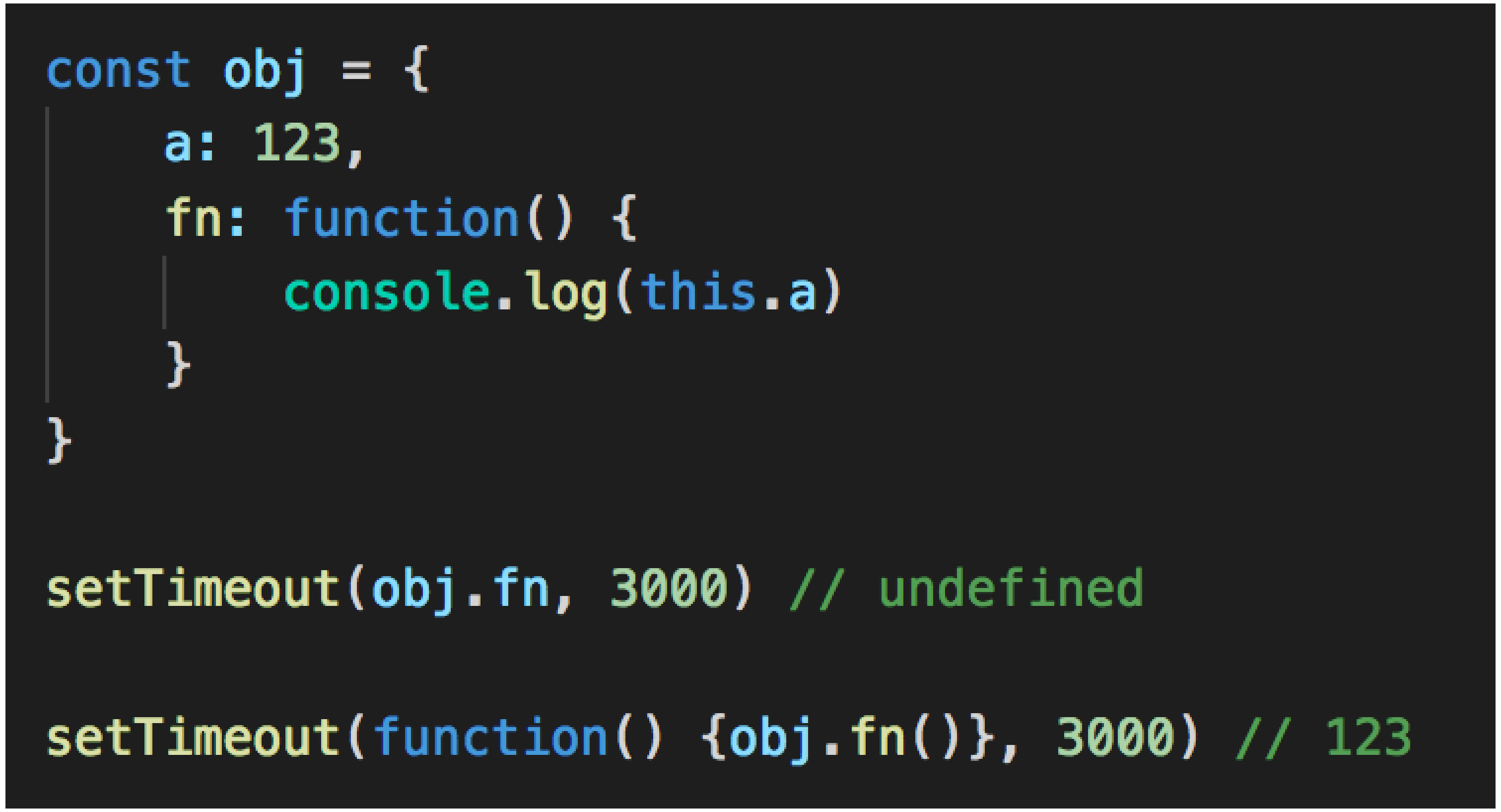
```
const obj = {
    a: 123,
    fn: function() {
        console.log(this.a)
    }
}

obj.fn() // 123
```



```
const obj = {
    a: 123,
    fn: function() {
        console.log(this.a)
    }
}

// obj.fn()
const f = obj.fn
f() // undefined (window.a nebo global.a)
```

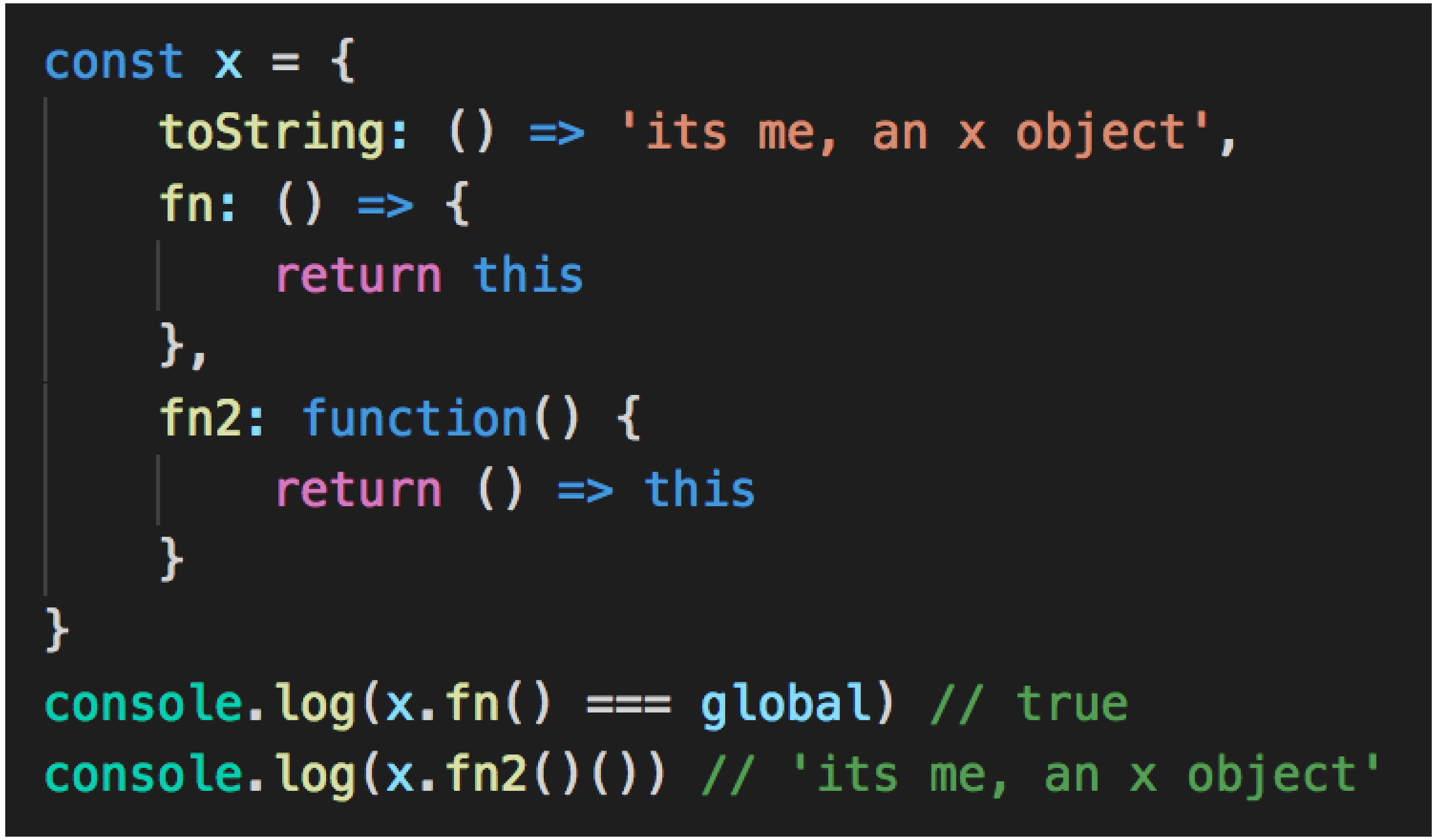


```
const obj = {
    a: 123,
    fn: function() {
        console.log(this.a)
    }
}
const obj2 = {
    a: 321
}

obj2.fn = obj.fn

obj2.fn() // 321
```


```javascript
const obj = {
    a: 123,
    fn: function() {
        console.log(this.a)
    }
}

setTimeout(obj.fn, 3000) // undefined

setTimeout(function() {obj.fn()}, 3000) // 123
```

$$\begin{tikzcd}
 & \textit{strict \& non-strict} \\
\textit{Context (this)} \arrow[ur] \arrow[r] & \textit{volání metody}
\end{tikzcd}$$

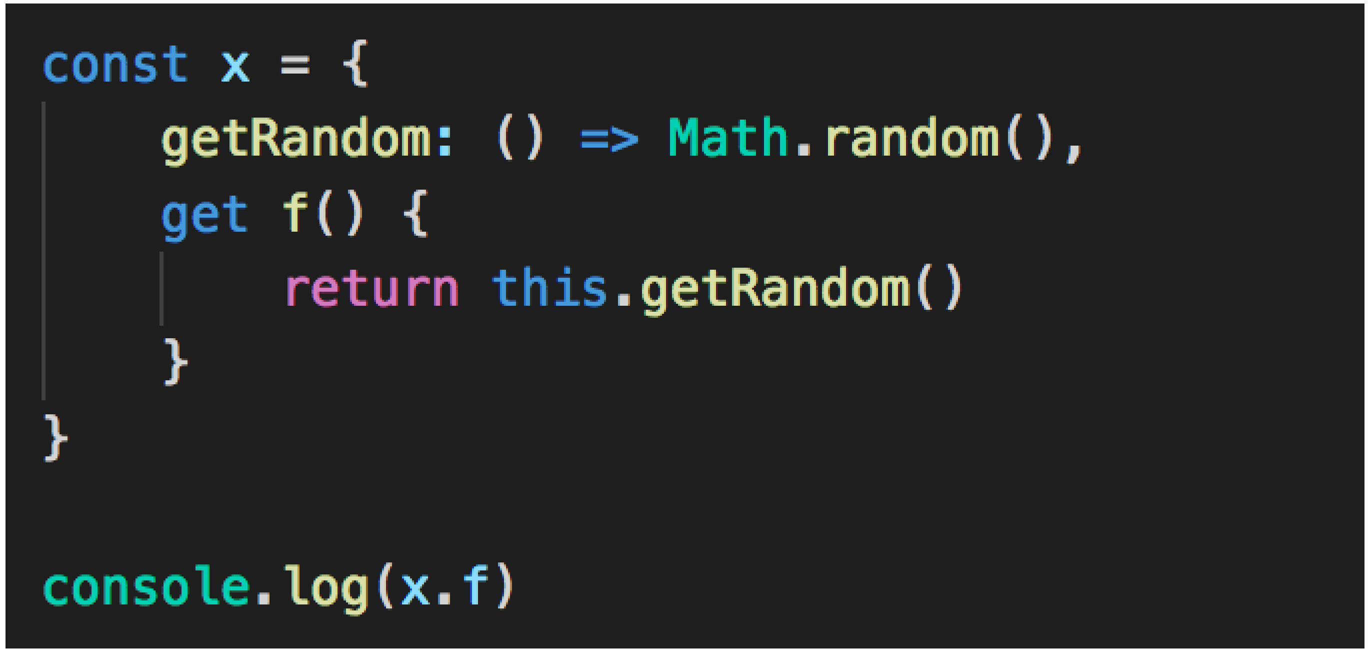
$$\begin{tikzcd}
 & \textit{strict \& non-strict} \\
\textit{Context (this)} \arrow[ur] \arrow[r] \arrow[dr] & \textit{volání metody} \\
 & \textit{arrow function expression}
\end{tikzcd}$$



```
const x = {
    toString: () => 'its me, an x object',
    fn: () => {
        return this
    },
    fn2: function() {
        return () => this
    }
}
console.log(x.fn() === global) // true
console.log(x.fn2()()) // 'its me, an x object'
```

$$\begin{tikzcd}
 & \textit{strict \& non-strict} \\
\text{Context (this)} \arrow[ur] \arrow[r] \arrow[dr] & \textit{volání metody} \\
 & \textit{arrow function expression}
\end{tikzcd}$$

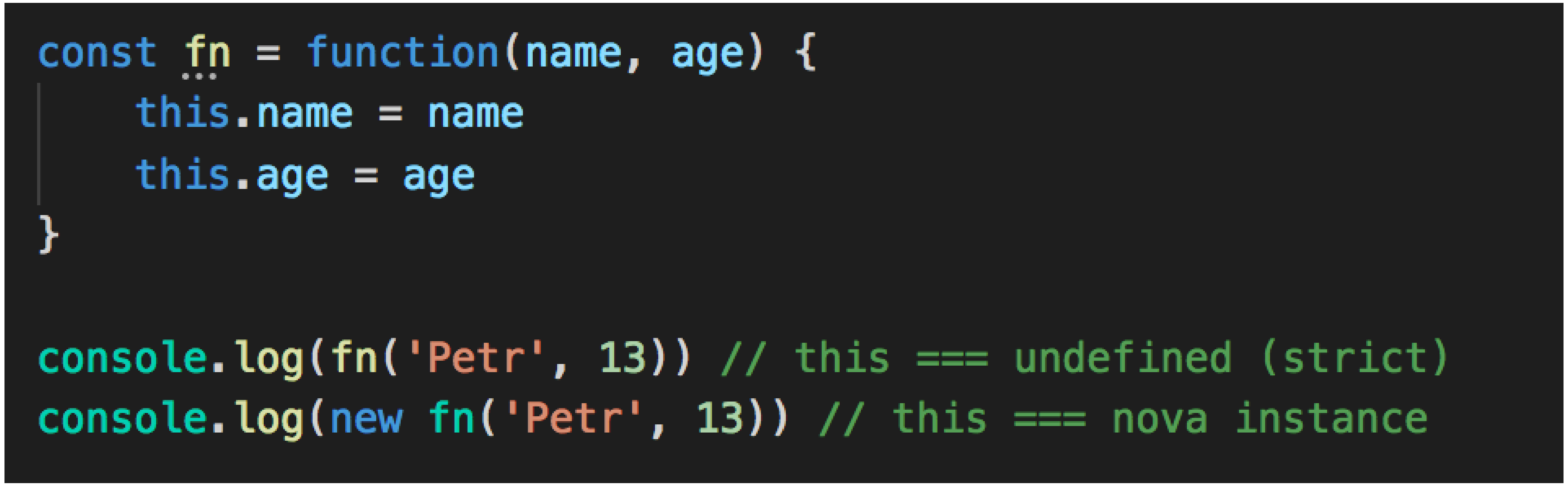
$$\begin{tikzcd}
 & \textit{strict \& non-strict} \\
 & \textit{volání metody} \\
\textit{Context (this)} \arrow[ur] \arrow[uur] \arrow[r] \arrow[dr] & \textit{arrow function expression} \\
 & \textit{getter \& setter}
\end{tikzcd}$$



```
const x = {
    getRandom: () => Math.random(),
    get f() {
        return this.getRandom()
    }
}

console.log(x.f)
```

$$\begin{tikzcd}
 & \text{strict \& non-strict} \\
 & \text{volání metody} \\
\text{Context (this)} \arrow[ur] \arrow[uur] \arrow[r] \arrow[dr] & \text{arrow function expression} \\
 & \text{getter \& setter}
\end{tikzcd}$$

$$\begin{tikzcd}[column sep=large]
 & \textit{strict \& non-strict} \\
 & \textit{volání metody} \\
\textit{Context (this)} \arrow[ur] \arrow[uur] \arrow[r] \arrow[dr] \arrow[ddr] & \textit{arrow function expression} \\
 & \textit{getter \& setter} \\
 & \textit{operator new}
\end{tikzcd}$$

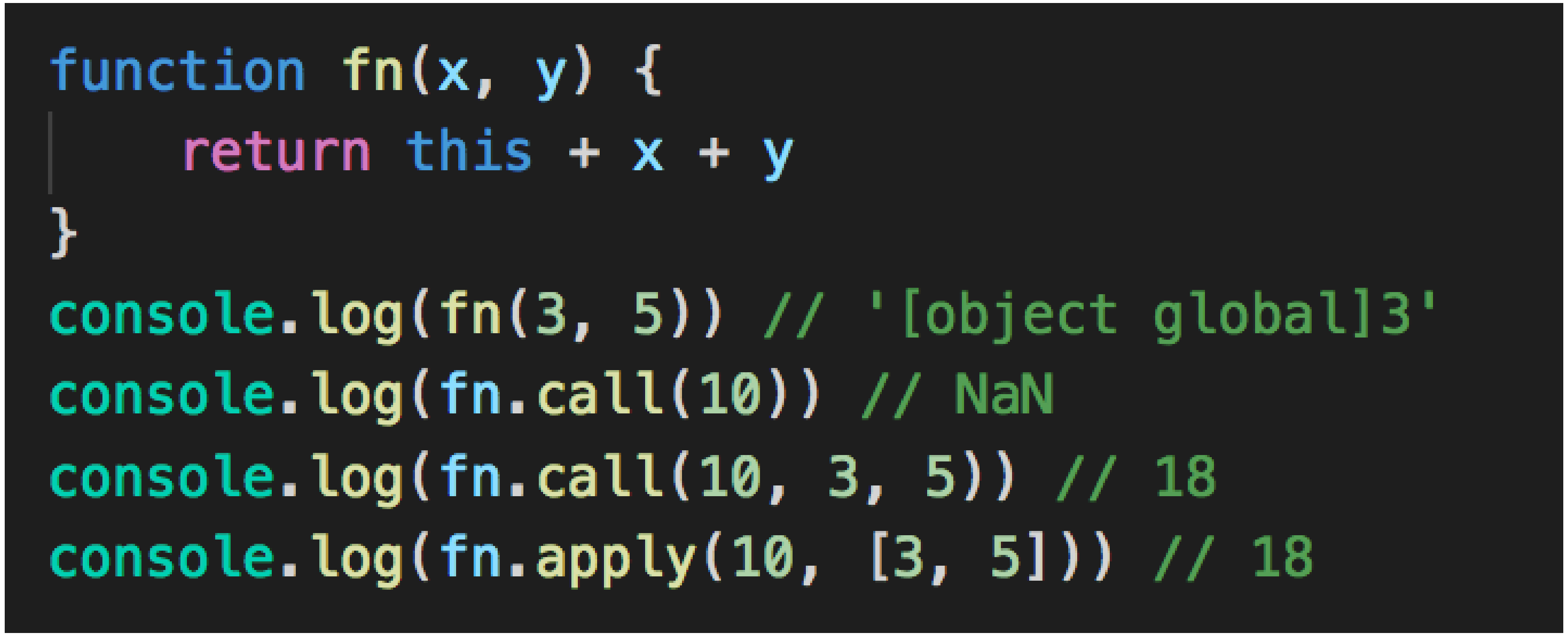

```
const fn = function(name, age) {
    this.name = name
    this.age = age
}

console.log(fn('Petr', 13)) // this === undefined (strict)
console.log(new fn('Petr', 13)) // this === nova instance
```

Context (this)

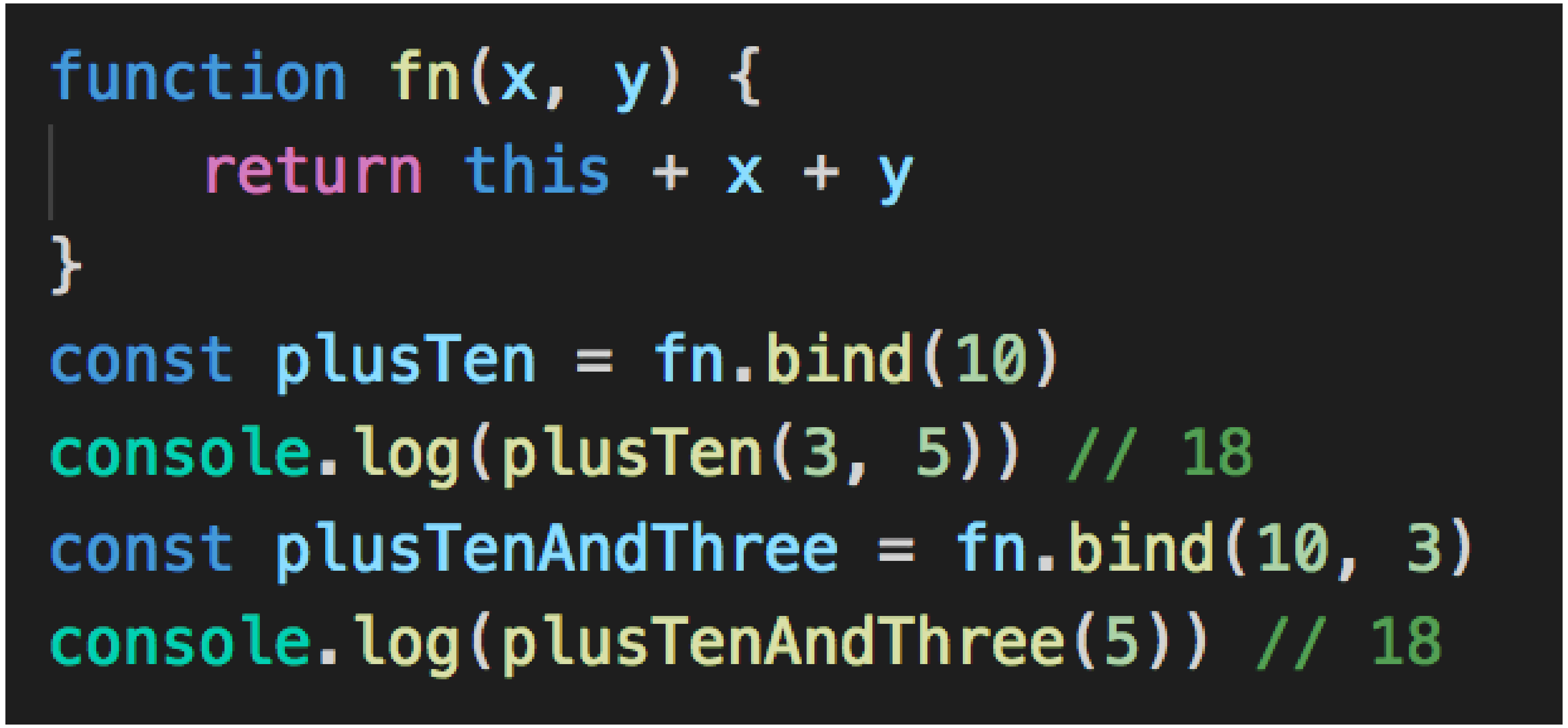
- *strict & non-strict*
- *volání metody*
- *arrow function expression*
- *getter & setter*
- *operator new*

Context (this)

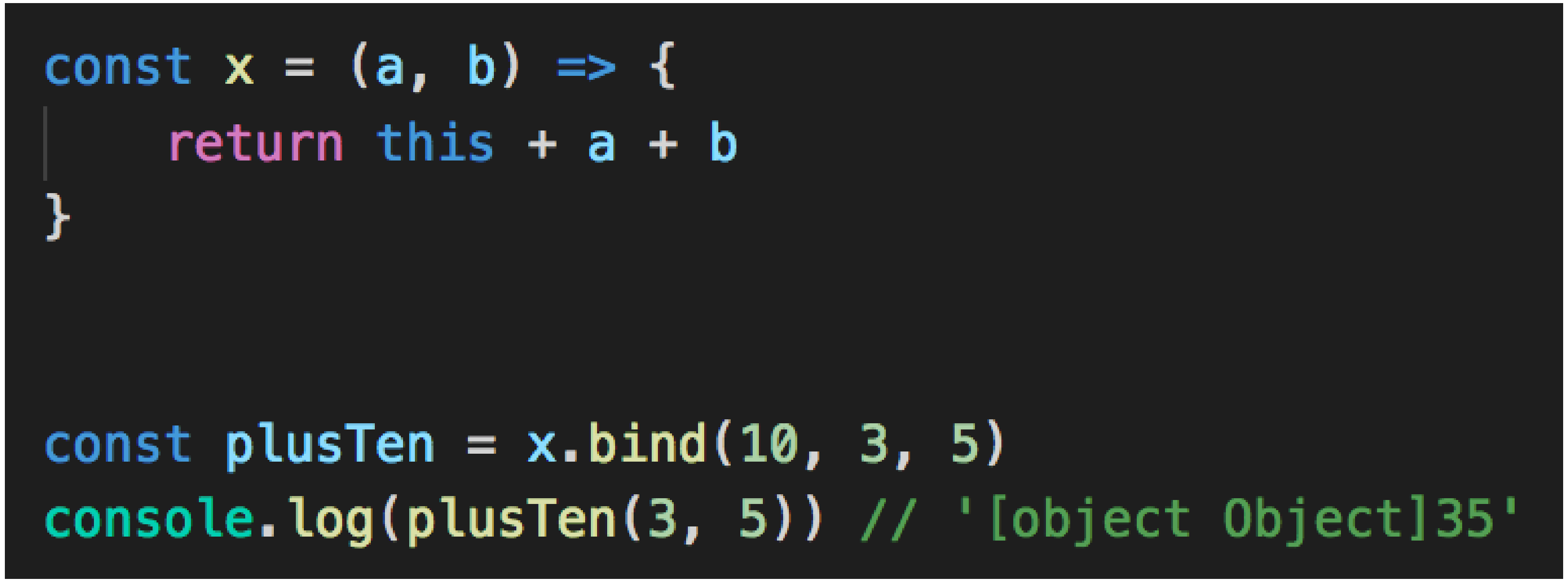
- *strict & non-strict*
- *volání metody*
- *arrow function expression*
- *getter & setter*
- *operator new*
- *.call() & .apply() & .bind()*



```
function fn(x, y) {
    return this + x + y
}
console.log(fn(3, 5)) // '[object global]3'
console.log(fn.call(10)) // NaN
console.log(fn.call(10, 3, 5)) // 18
console.log(fn.apply(10, [3, 5])) // 18
```



```
function fn(x, y) {
    return this + x + y
}
const plusTen = fn.bind(10)
console.log(plusTen(3, 5)) // 18
const plusTenAndThree = fn.bind(10, 3)
console.log(plusTenAndThree(5)) // 18
```



```
const x = (a, b) => {
    return this + a + b
}


const plusTen = x.bind(10, 3, 5)
console.log(plusTen(3, 5)) // '[object Object]35'
```

Úkoly → bit.ly/2Fbxw28

// end