# JavaScript + Angular

Part 1

Core JavaScript

Author: Andrey Kucherenko

```
<script></script>
```

ES6/2015

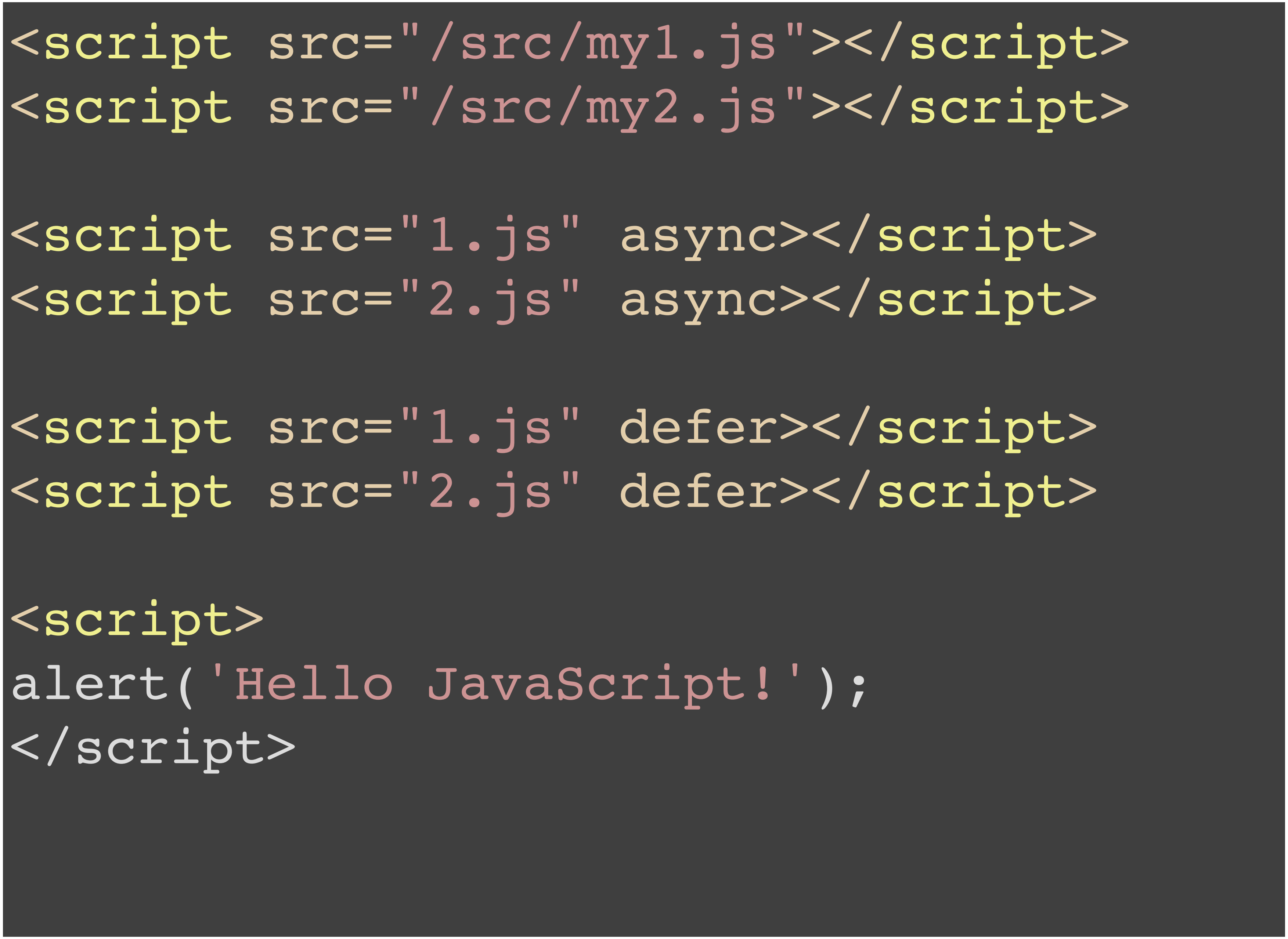
TypeScript

CoffeeScript

node
JS®

http://c9.io

```
alert('Hello
JavaScript!');
```



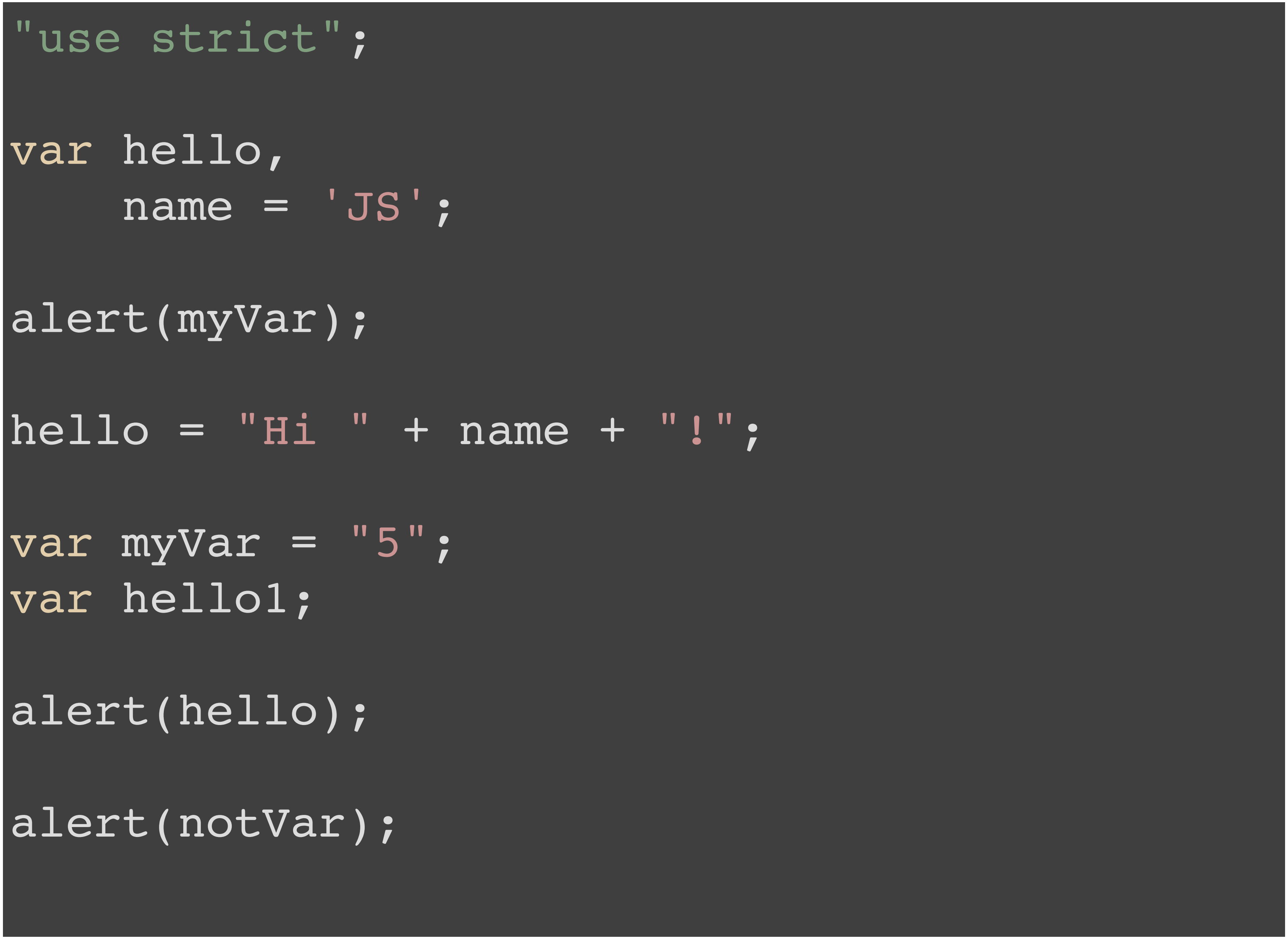
```
<script src="/src/my1.js"></script>
<script src="/src/my2.js"></script>

<script src="1.js" async></script>
<script src="2.js" async></script>

<script src="1.js" defer></script>
<script src="2.js" defer></script>

<script>
alert('Hello JavaScript!');
</script>
```

# Tasks

- Create html file
- Create js file
- Add <script> section
- Add <script src="..."></script>
- Change order of scripts

# variables

```
"use strict";
```

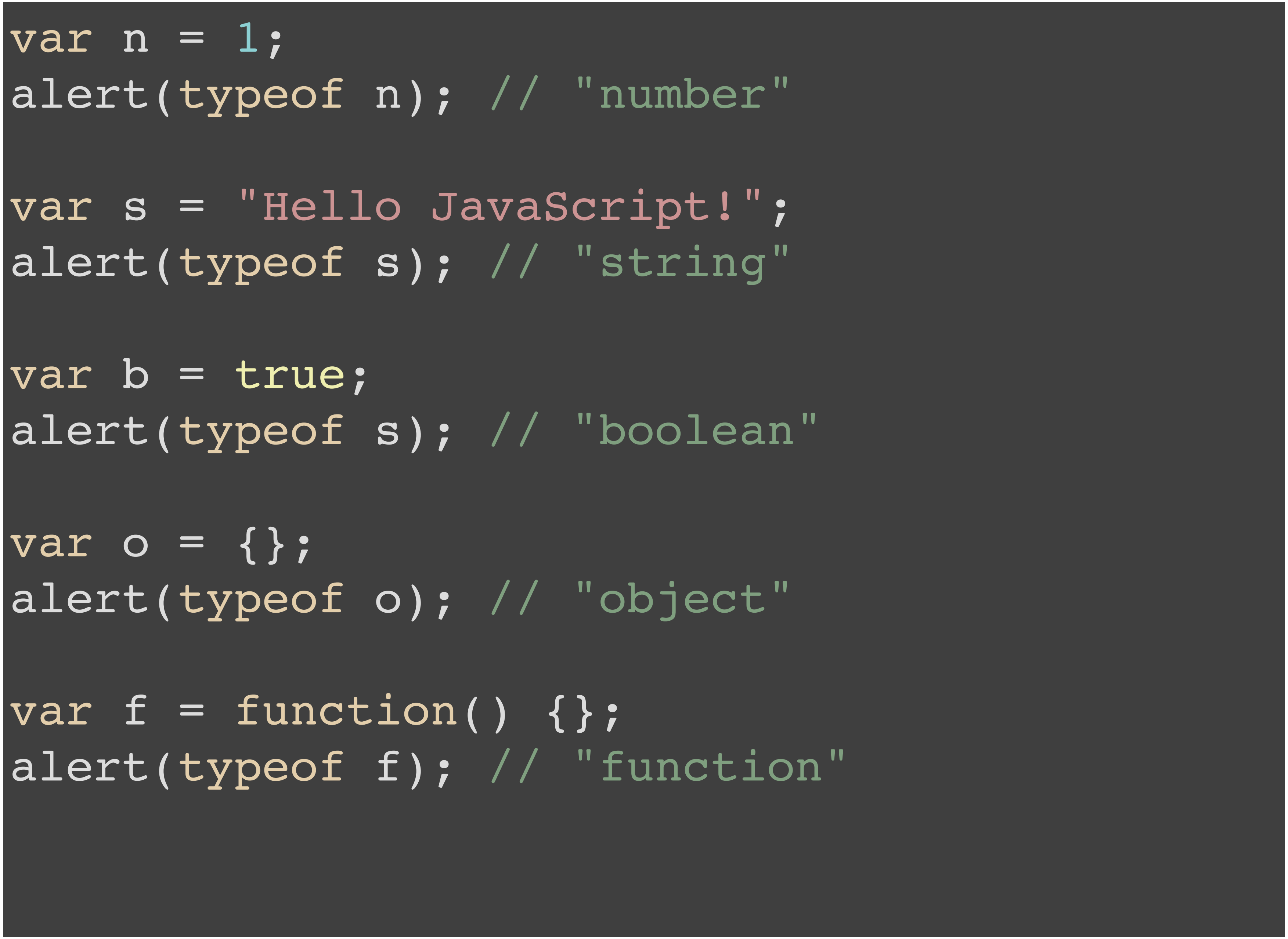


```
"use strict";

var hello,
    name = 'JS';

alert(myVar);

hello = "Hi " + name + "!";

var myVar = "5";
var hello1;

alert(hello);

alert(notVar);
```

# Tasks

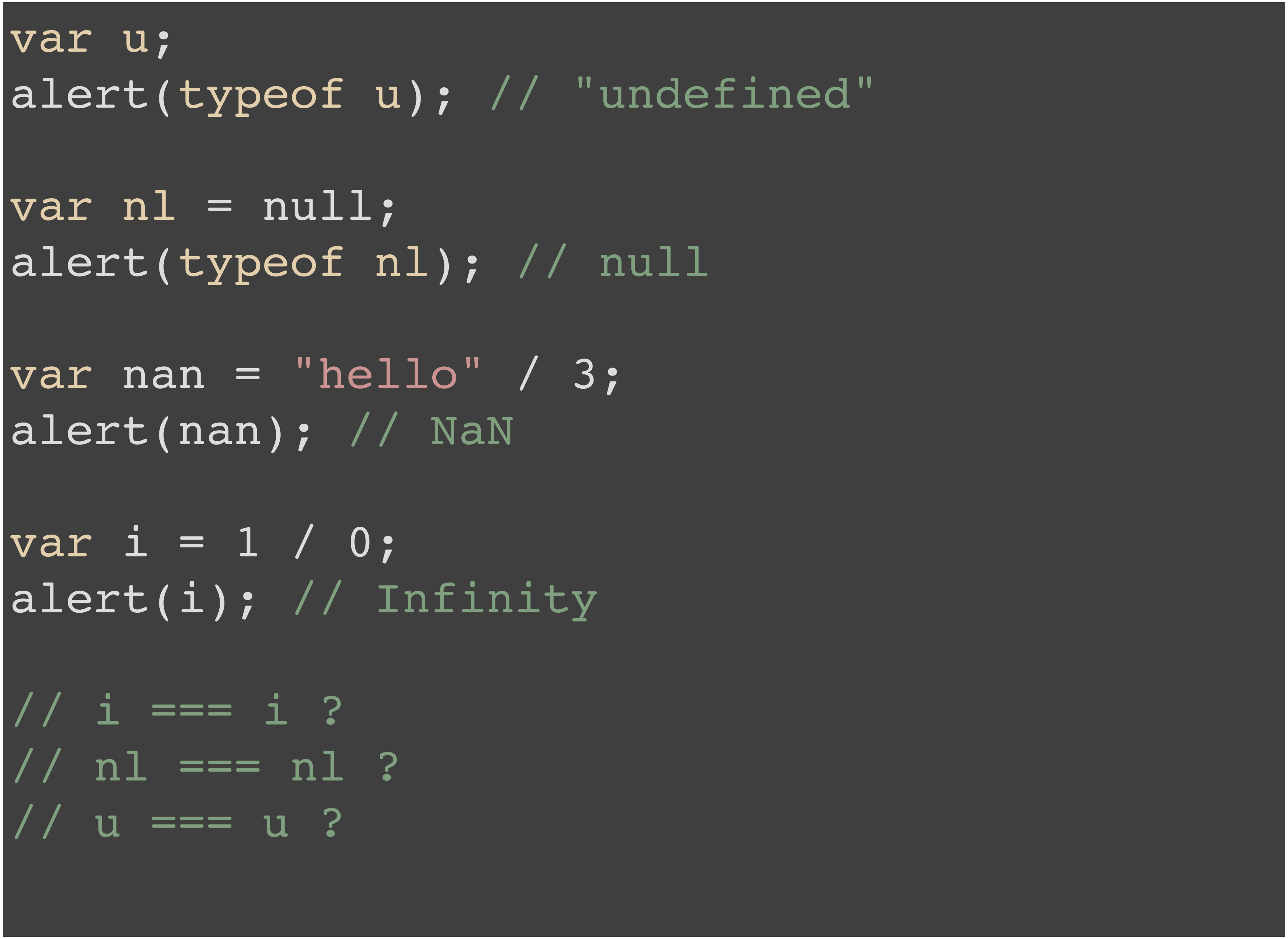
- Use created js file
- Add couple of new variables
- Try to change data in variables

# Data Types

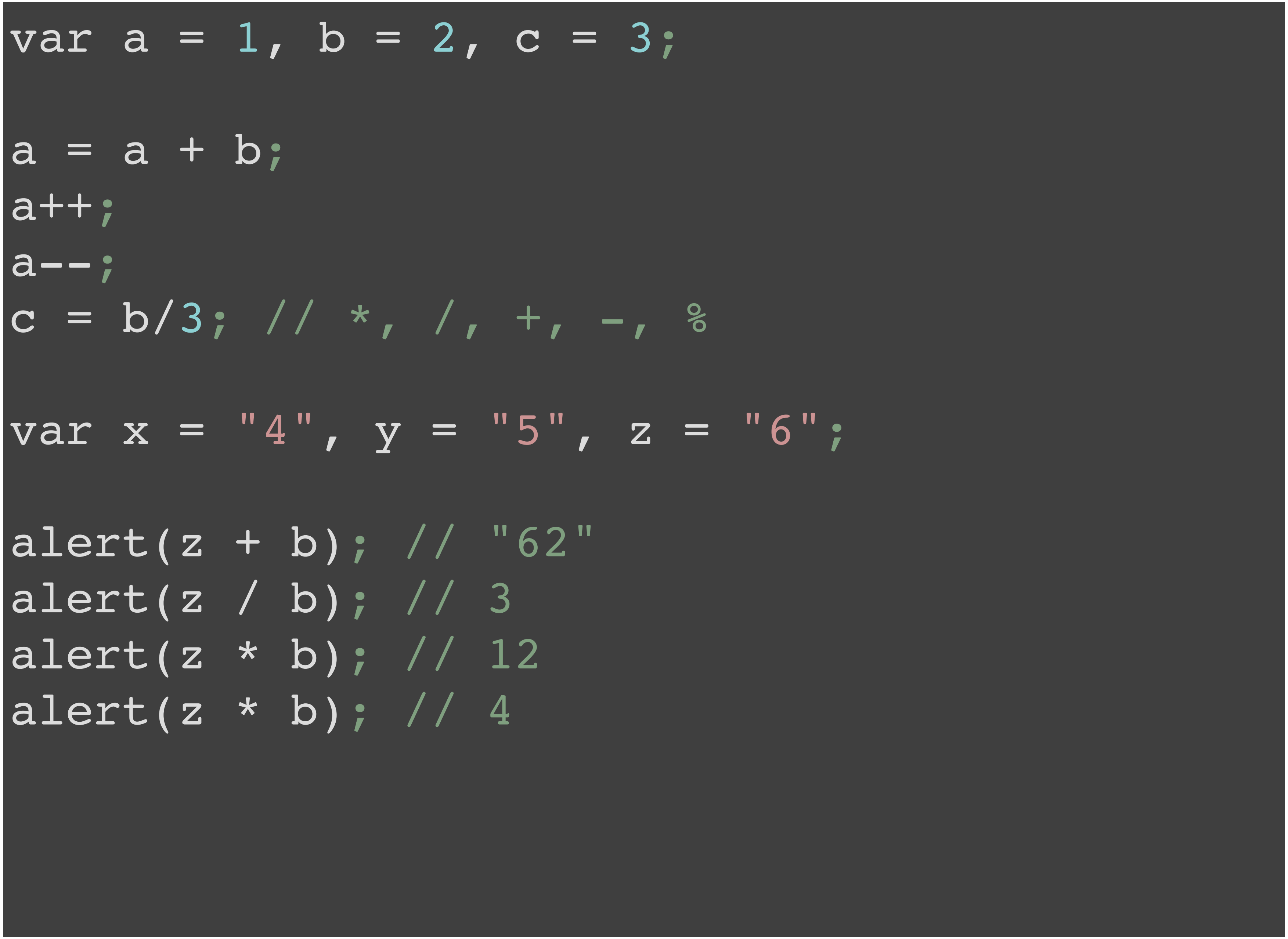


```
var n = 1;
alert(typeof n); // "number"

var s = "Hello JavaScript!";
alert(typeof s); // "string"

var b = true;
alert(typeof s); // "boolean"

var o = {};
alert(typeof o); // "object"

var f = function() {};
alert(typeof f); // "function"
```

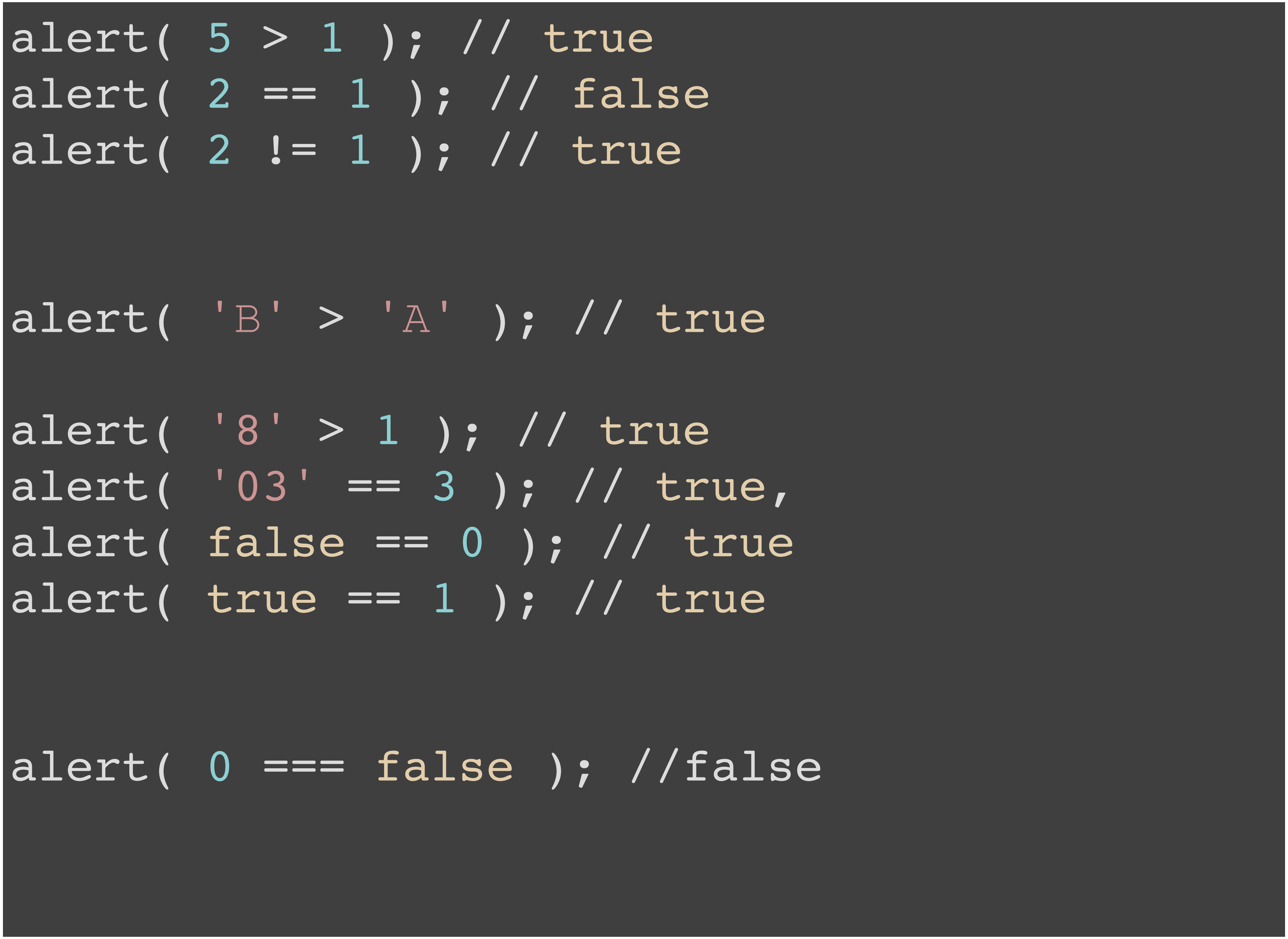
```
var a;

a = [1, 2];
a = new Array(1, 2);


var o;

o = {"test": 1, "zz": [1,2]};
o = new Object();
```



```
var u;
alert(typeof u); // "undefined"

var nl = null;
alert(typeof nl); // null

var nan = "hello" / 3;
alert(nan); // NaN

var i = 1 / 0;
alert(i); // Infinity

// i === i ?
// nl === nl ?
// u === u ?
```



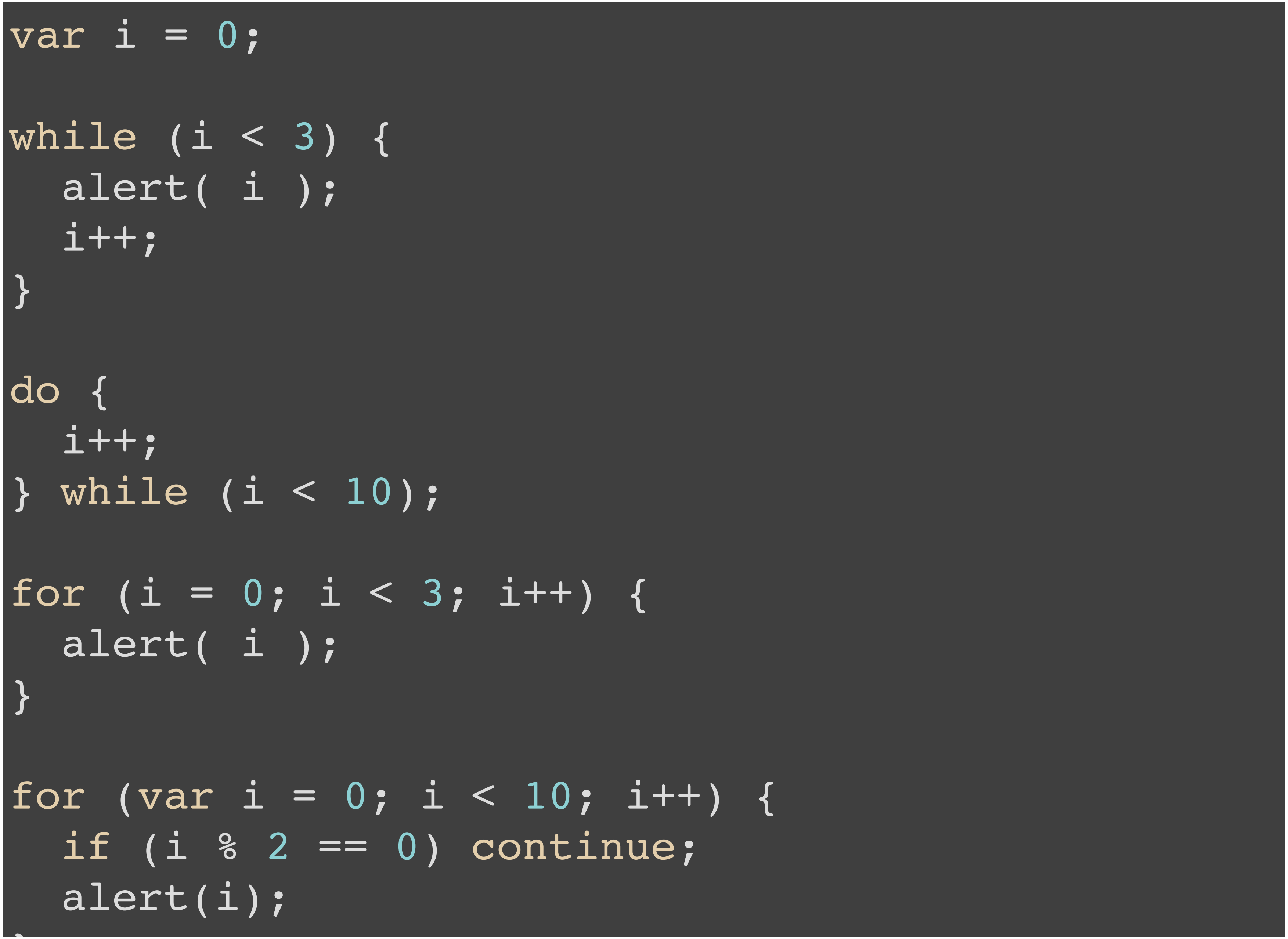
```
var a = 1, b = 2, c = 3;

a = a + b;
a++;
a--;
c = b/3; // *, /, +, -, %

var x = "4", y = "5", z = "6";

alert(z + b); // "62"
alert(z / b); // 3
alert(z * b); // 12
alert(z * b); // 4
```



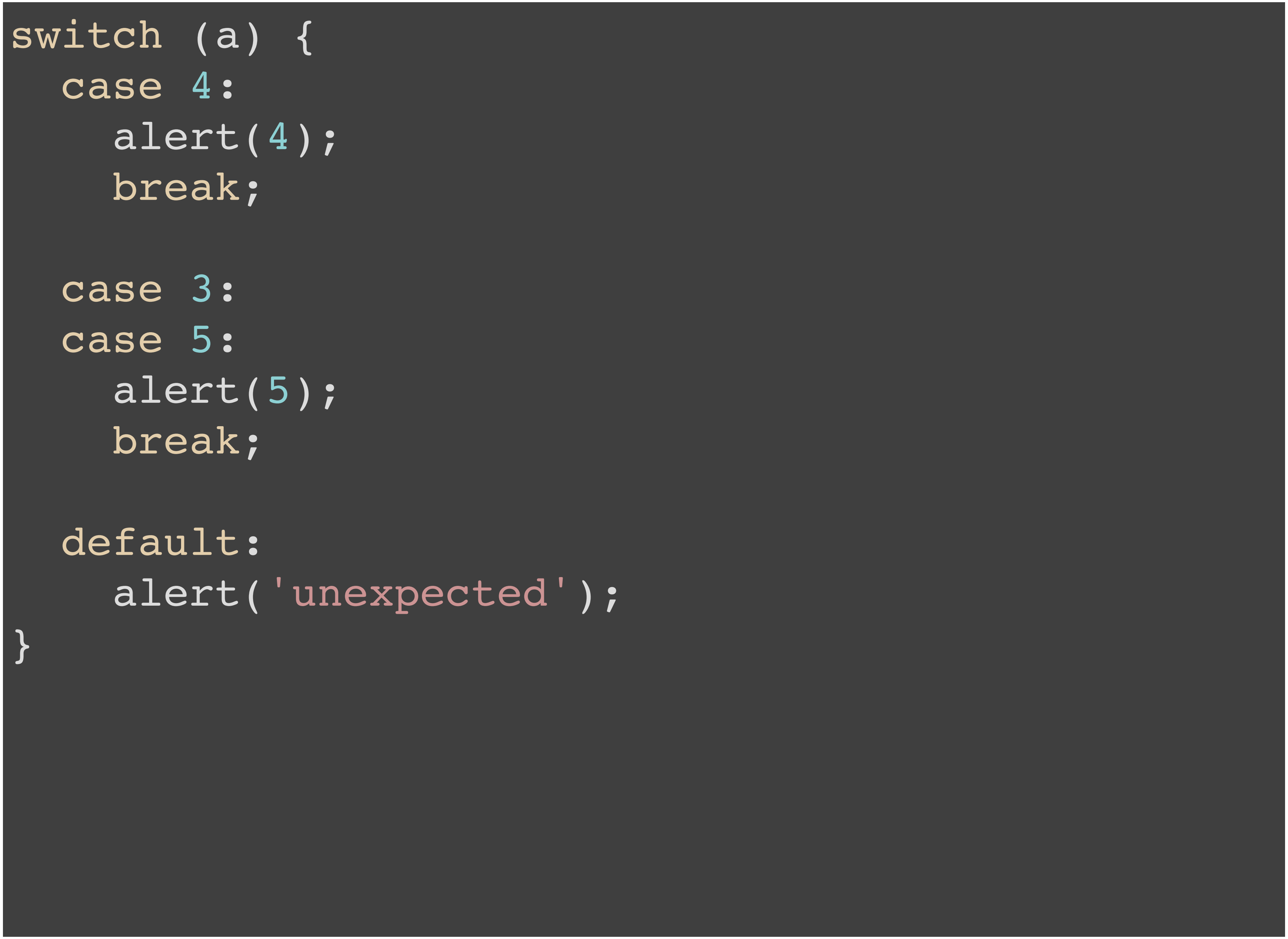
```
alert( 5 > 1 ); // true
alert( 2 == 1 ); // false
alert( 2 != 1 ); // true


alert( 'B' > 'A' ); // true

alert( '8' > 1 ); // true
alert( '03' == 3 ); // true,
alert( false == 0 ); // true
alert( true == 1 ); // true


alert( 0 === false ); //false
```

```
var a = 42, good;
if (a === 42) {
  good = true;
} else {
  good = false;
}

good = (a === 42);

good = (a > 42) ? true : false;
```

```
var result = zz || yy;

var result = x && y;

var result = !value;
```



```
var i = 0;

while (i < 3) {
  alert( i );
  i++;
}

do {
  i++;
} while (i < 10);

for (i = 0; i < 3; i++) {
  alert( i );
}

for (var i = 0; i < 10; i++) {
  if (i % 2 == 0) continue;
  alert(i);
}
```

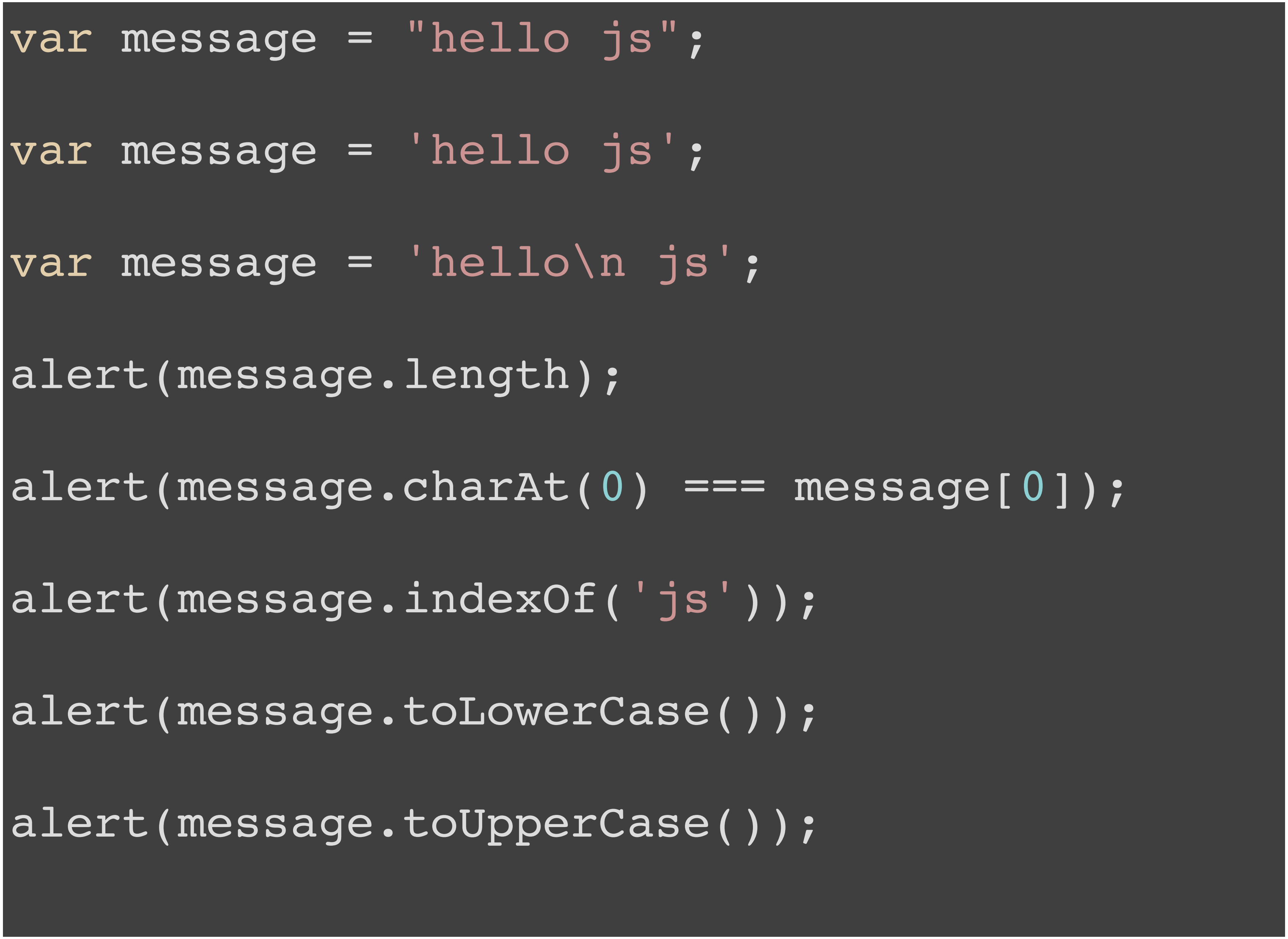


```
switch (a) {
  case 4:
    alert(4);
    break;

  case 3:
  case 5:
    alert(5);
    break;

  default:
    alert('unexpected');
}
```

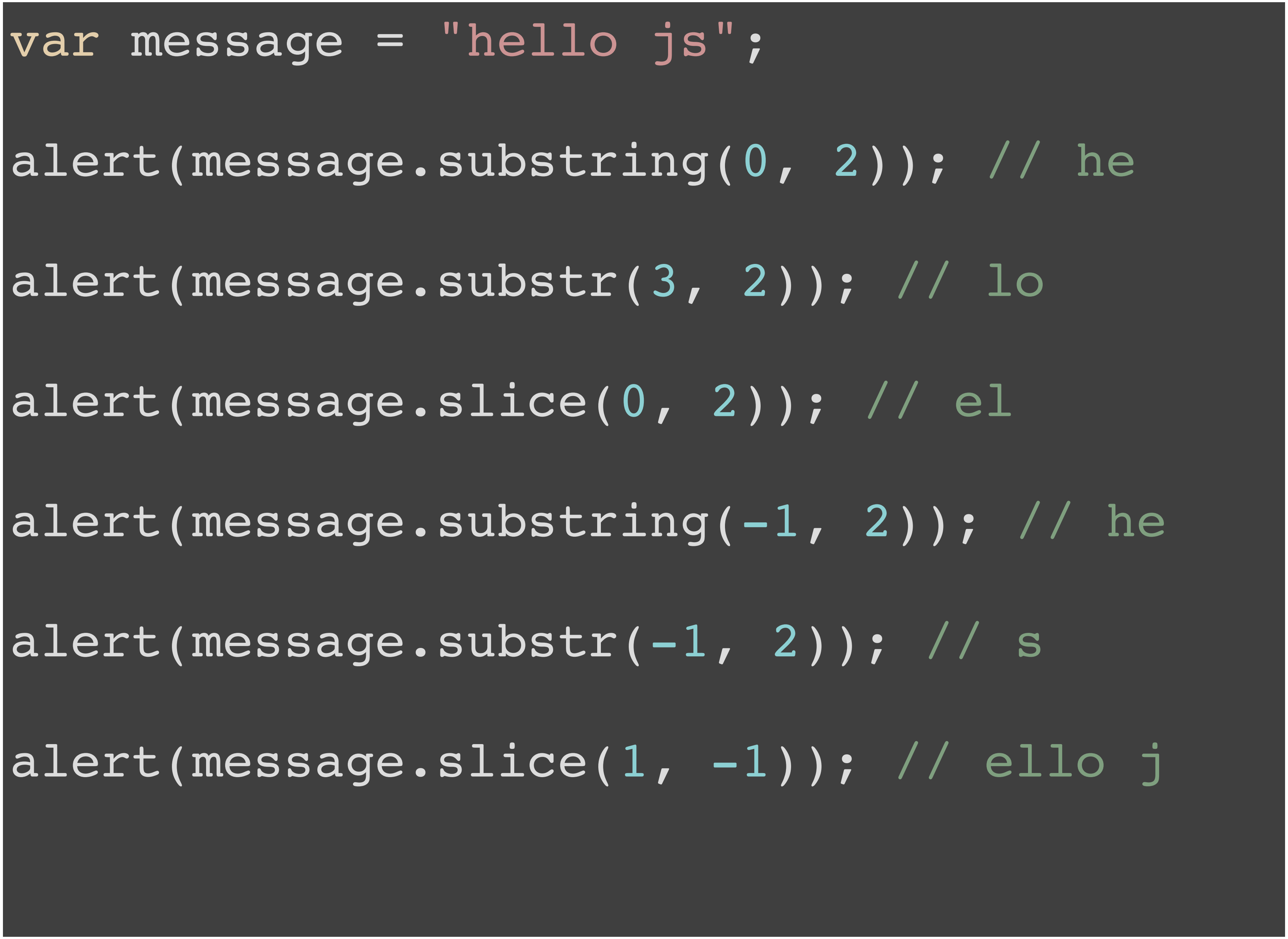
# Task

- Create array of fruits
- For each fruit add order number at start of name
- Calculate length of array
- Make object with keys named like fruits in array and value is order number

# str



```
var message = "hello js";

var message = 'hello js';

var message = 'hello\n js';

alert(message.length);

alert(message.charAt(0) === message[0]);

alert(message.indexOf('js'));

alert(message.toLowerCase());

alert(message.toUpperCase());
```



```
var message = "hello js";

alert(message.substring(0, 2)); // he

alert(message.substr(3, 2)); // lo

alert(message.slice(0, 2)); // el

alert(message.substring(-1, 2)); // he

alert(message.substr(-1, 2)); // s

alert(message.slice(1, -1)); // ello j
```

# Tasks

- Make function for reverse string
- Capitalize string
- Split string by separator

obj

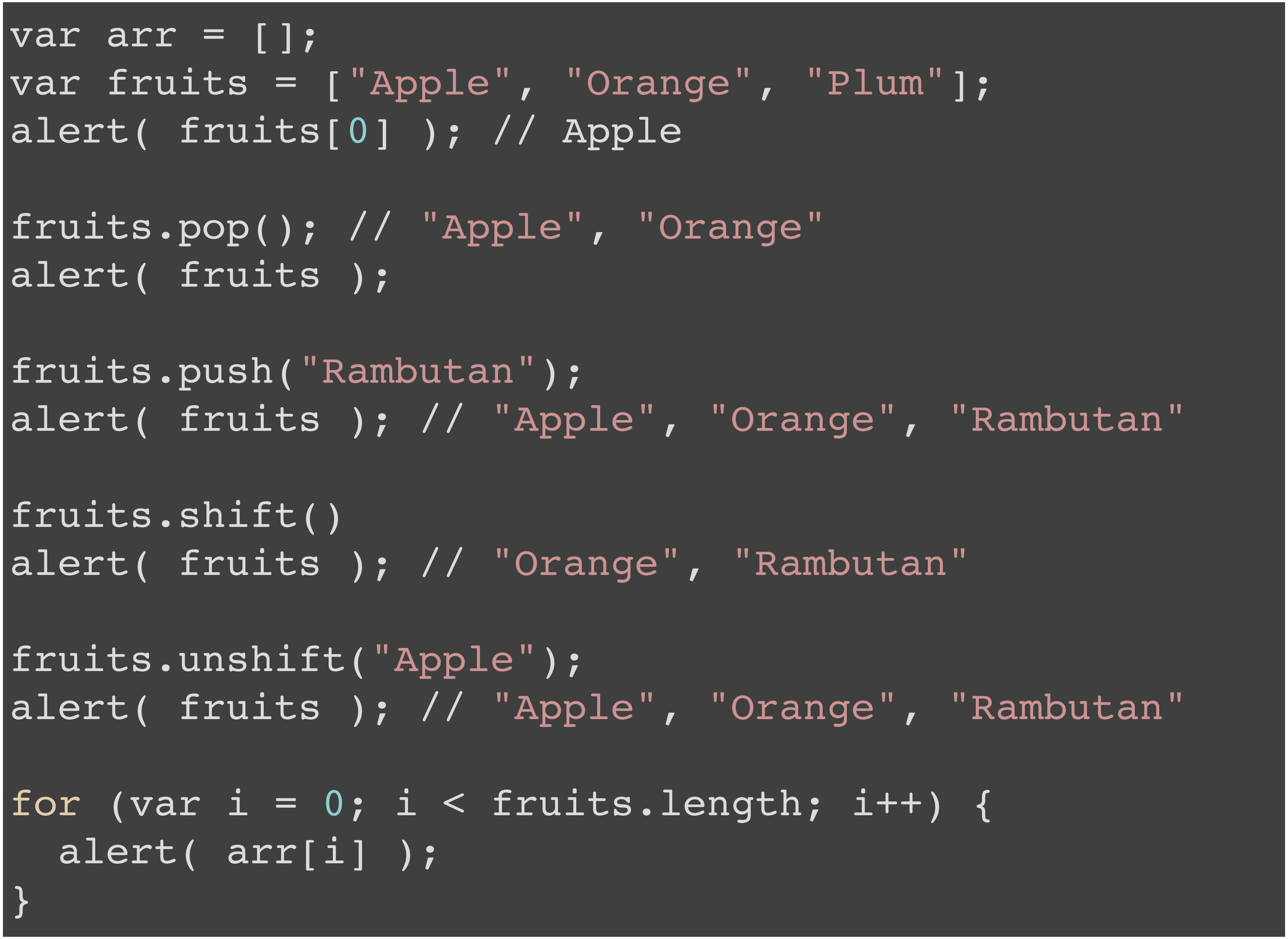
```
o = {};
o = new Object();


o.hello = 'js';

a['test'] = 'test';

for (key in o) {
  alert(key + ' = ' + o[key]);
}
```

# Tasks

- Create array of keys
- Concat two objects
- Show difference between objects

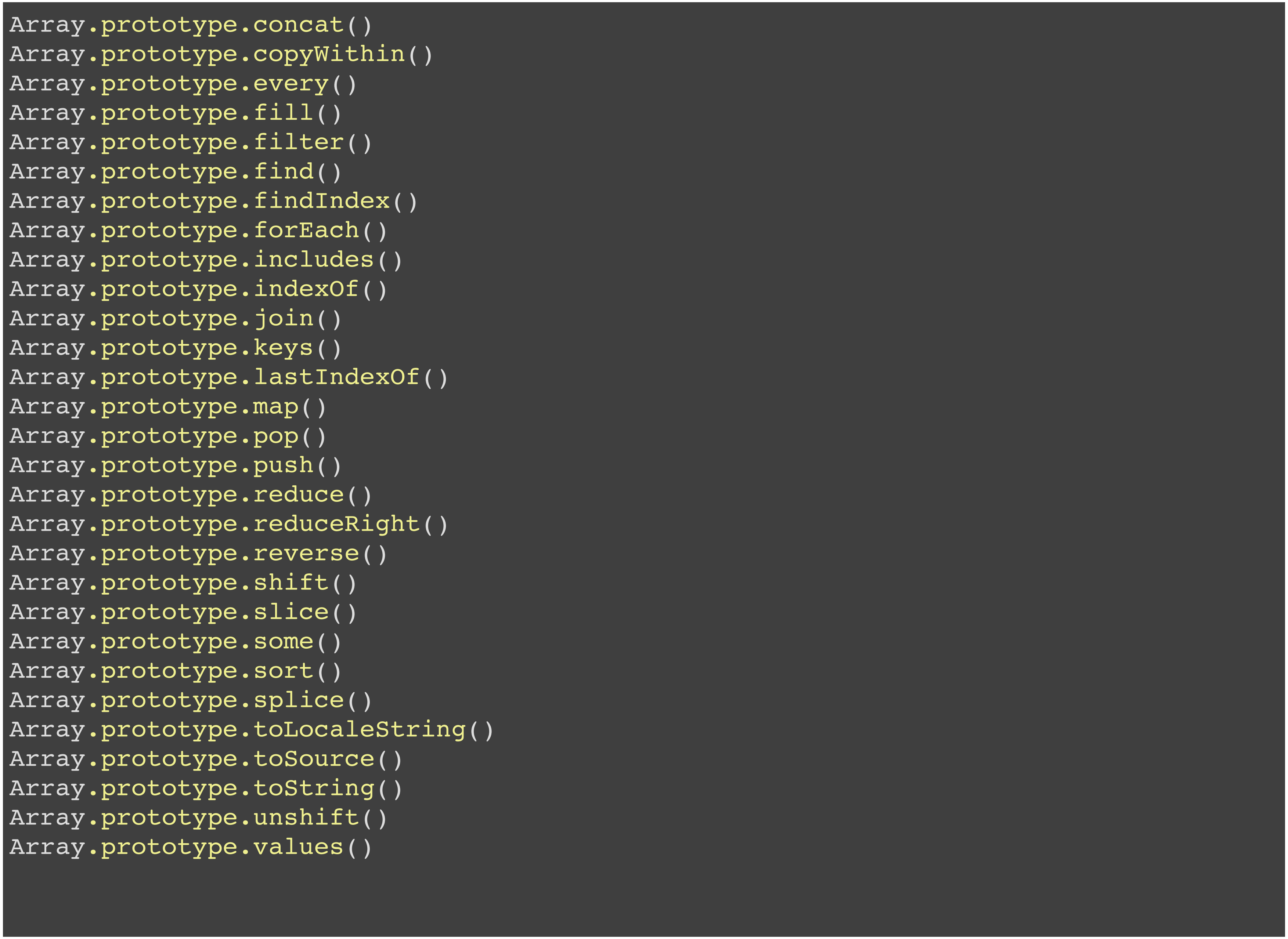
```
Object.assign()
Object.create()
Object.defineProperties()
Object.defineProperty()
Object.entries()
Object.freeze()
Object.getOwnPropertySymbols()
Object.getPrototypeOf()
Object.is()
Object.isExtensible()
Object.isFrozen()
Object.isSealed()
Object.keys()
Object.observe()
Object.preventExtensions()
Object.prototype.eval()
Object.prototype.hasOwnProperty()
Object.prototype.isPrototypeOf()
Object.prototype.propertyIsEnumerable()
Object.prototype.toLocaleString()
Object.prototype.toSource()
Object.prototype.toString()
Object.prototype.unwatch()
Object.prototype.valueOf()
Object.prototype.watch()
Object.seal()
Object.setPrototypeOf()
```

array



```
var arr = [];
var fruits = ["Apple", "Orange", "Plum"];
alert( fruits[0] ); // Apple

fruits.pop(); // "Apple", "Orange"
alert( fruits );

fruits.push("Rambutan");
alert( fruits ); // "Apple", "Orange", "Rambutan"

fruits.shift()
alert( fruits ); // "Orange", "Rambutan"

fruits.unshift("Apple");
alert( fruits ); // "Apple", "Orange", "Rambutan"

for (var i = 0; i < fruits.length; i++) {
  alert( arr[i] );
}
```

```
var matrix = [
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
];

alert( matrix[1][1] );


var array = "hello js".split(' ');
```

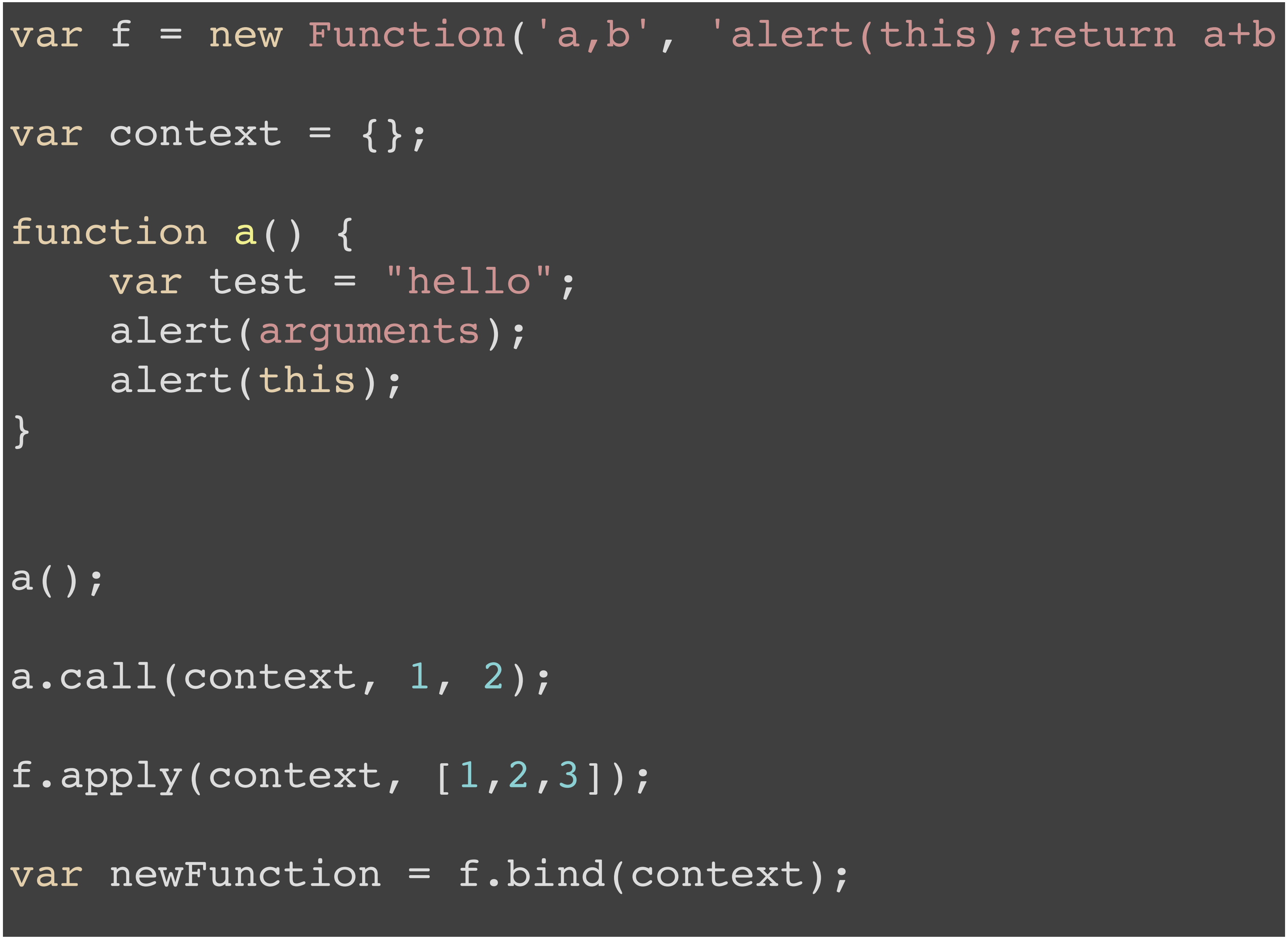


```
Array.prototype.concat()
Array.prototype.copyWithin()
Array.prototype.every()
Array.prototype.fill()
Array.prototype.filter()
Array.prototype.find()
Array.prototype.findIndex()
Array.prototype.forEach()
Array.prototype.includes()
Array.prototype.indexOf()
Array.prototype.join()
Array.prototype.keys()
Array.prototype.lastIndexOf()
Array.prototype.map()
Array.prototype.pop()
Array.prototype.push()
Array.prototype.reduce()
Array.prototype.reduceRight()
Array.prototype.reverse()
Array.prototype.shift()
Array.prototype.slice()
Array.prototype.some()
Array.prototype.sort()
Array.prototype.splice()
Array.prototype.toLocaleString()
Array.prototype.toSource()
Array.prototype.toString()
Array.prototype.unshift()
Array.prototype.values()
```

# Tasks

- Reverse array
- Find maximum and minimum value
- String calculator

# functions



```
var f = new Function('a,b', 'alert(this);return a+b

var context = {};

function a() {
    var test = "hello";
    alert(arguments);
    alert(this);
}



a();

a.call(context, 1, 2);

f.apply(context, [1,2,3]);

var newFunction = f.bind(context);
```

# Tasks

- Write function for summarize properties z and x
- Call this function with different context
- Create function with Function inside function and call it (check current context)

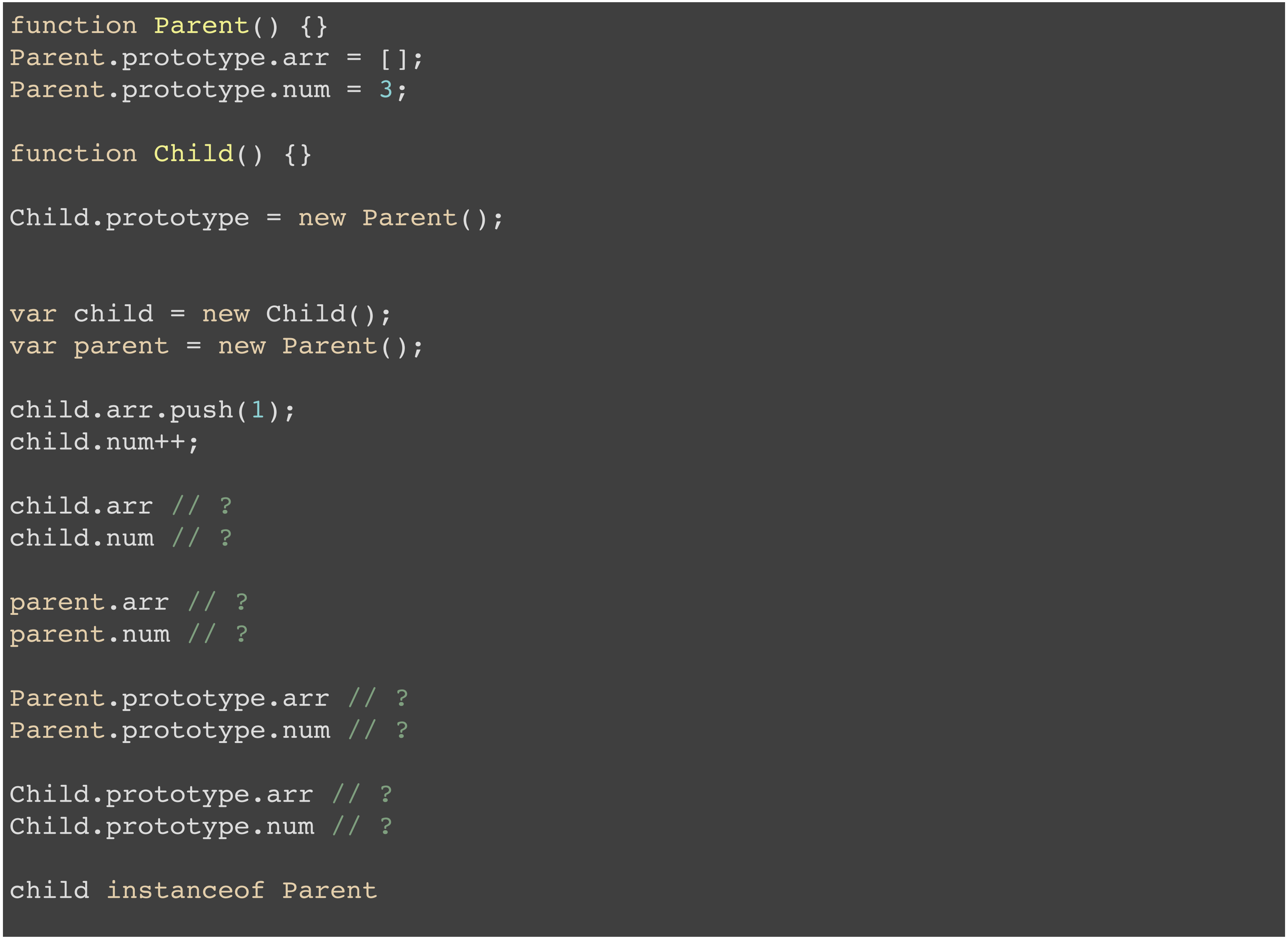
# closure

```
function makeCounter() {
  var currentCount = 1;

  return function() { // (**)
    return currentCount++;
  };
}
```

# Tasks

- Make function call inside loop and return current call index

# oop



```
function Parent() {}
Parent.prototype.arr = [];
Parent.prototype.num = 3;

function Child() {}

Child.prototype = new Parent();


var child = new Child();
var parent = new Parent();

child.arr.push(1);
child.num++;

child.arr // ?
child.num // ?

parent.arr // ?
parent.num // ?

Parent.prototype.arr // ?
Parent.prototype.num // ?

Child.prototype.arr // ?
Child.prototype.num // ?

child instanceof Parent
```

```
function MyParentClass(){
    this.parentMethod = function (){}

}


function MyChildClass() {
    MyParentClass.apply(this, arguments);
    this.myMethod = function () {
        alert('hello');
    }

}
```

# Tasks

- Create parent of parent
- Check instanceof Function, Object, etc.

# Project