

Type checks

```
1 console.log(typeof ""); // "string"
2 console.log(typeof "hello"); // "string"
3 console.log(typeof String("hello")); // "string"
4 console.log(typeof new String("hello")); // "object"
5
6 console.log(typeof 0); // "number"
7 console.log(typeof -0); // "number"
8 console.log(typeof 0xff); // "number"
9 console.log(typeof -3.142); // "number"
10 console.log(typeof Infinity); // "number"
11 console.log(typeof -Infinity); // "number"
12 console.log(typeof NaN); // "number"
13 console.log(typeof Number(53)); // "number"
14 console.log(typeof new Number(53)); // "object"
15
16 console.log(typeof true); // "boolean"
17 console.log(typeof false); // "boolean"
18 console.log(typeof new Boolean(true)); // "object"
19
20 console.log(typeof undefined); // "undefined"
21
22 console.log(typeof null); // "object"
23
24 console.log(typeof Symbol()); // "symbol"
25
26 console.log(typeof []); // "object"
27 console.log(typeof Array(5)); // "object"
28
29 console.log(typeof function() {}); // "function"
30 console.log(typeof new Function); // "function"
31
32 console.log(typeof new Date); // "object"
33
34 console.log(typeof /^(.+)$/); // "object"
35 console.log(typeof new RegExp("^(.+)$")); // "object"
36
37 console.log(typeof {}); // "object"
38 console.log(typeof new Object); // "object"
```

Checking for null

```
1 functionisNull(value) {  
2   return value === null;  
3 }  
4 console.log(undefined == null); // true  
5 console.log(undefined === null); // falsev
```

Checking for NaN

```
1 console.log(isNaN(NaN)); // true  
2 console.log(isNaN(null)); // false  
3 console.log(isNaN(undefined)); // true  
4 console.log(isNaN(Infinity)); // false  
5  
6 console.log(Number.isNaN(NaN)); // true  
7 console.log(Number.isNaN(null)); // false  
8 console.log(Number.isNaN(undefined)); // false  
9 console.log(Number.isNaN(Infinity)); // false  
10  
11 var x = NaN;  
12  
13 console.log(x == NaN); // false  
14 console.log(x === NaN); // false  
15  
16 function isNaN(value) {  
17   return value !== value;  
18 }  
19  
20 Number.isNaN = Number.isNaN || (function(value) {  
21   return value !== value;  
22 })  
23  
24 function isNaN(value) {  
25   return Object.is(value, Number.NaN);  
26 }
```

Checking for arrays

```
1 // METHOD 1: constructor property
2 // Not reliable
3 function isArray(value) {
4   return typeof value == 'object' && value.constructor
5   === Array;
6 }
7
8 // METHOD 2: instanceof
9 // Not reliable since an object's prototype can be
10 changed
11 // Unexpected results within frames
12 function isArray(value) {
13   return value instanceof Array;
14 }
15
16 // METHOD 3: Object.prototype.toString()
17 // Better option and very similar to ES6 Array.isArray()
18 function isArray(value) {
19   return Object.prototype.toString.call(value) ===
20 '[object Array]';
21 }
22
23 // METHOD 4: ES6 Array.isArray()
24 function isArray(value) {
25   return Array.isArray(value);
26 }
```

Generic type checking

```
1 console.log(type('')); // "string"
2 console.log(type('hello')); // "string"
3 console.log(type(String('hello'))); // "string"
4 console.log(type(new String('hello'))); // "string"
5
6 console.log(type(0)); // "number"
7 console.log(type(-0)); // "number"
8 console.log(type(0xff)); // "number"
9 console.log(type(-3.142)); // "number"
10 console.log(type(Infinity)); // "number"
11 console.log(type(-Infinity)); // "number"
12 console.log(type(NaN)); // "number"
13 console.log(type(Number(53))); // "number"
14 console.log(type(new Number(53))); // "number"
15
16 console.log(type(true)); // "boolean"
17 console.log(type(false)); // "boolean"
18 console.log(type(new Boolean(true))); // "boolean"
19
20 console.log(type(undefined)); // "undefined"
21
22 console.log(type(null)); // "null"
23
24 console.log(type(Symbol())); // "symbol"
25 console.log(type(Symbol.species)); // "symbol"
26
27 console.log(type([])); // "array"
28 console.log(type(Array(5))); // "array"
29
30 console.log((function() { return type(arguments) })());
31 // "arguments"
32 console.log(type(function() {})); // "function"
33 console.log(type(new Function)); // "function"
34
35 console.log(type(class {})); // "function"
36
37 console.log(type({})); // "object"
38 console.log(type(new Object)); // "object"
```