

Author

Roman Semko - JS experts @ SemkoDev
<https://semkodev.com>

web development on steroids!

Constants

```
const PI = 3.141593
PI > 3.0
```

Scoping

```
// Block-Scoped Variables
for (let i = 0; i < a.length; i++) {
    let x = a[i]
    ...
}
for (let i = 0; i < b.length; i++) {
    let y = b[i]
    ...
}
let callbacks = []
for (let i = 0; i <= 2; i++) {
    callba cks[i] =
function () { return i * 2 }
}
callba cks [0]() === 0
callba cks [1]() === 2
callba cks [2]() === 4
// Block- Scoped Functions
{
    fun ction foo () {
return 1 }
    foo() === 1
    {
        fun ction foo ()
{ return 2 }
        foo() === 2
    }
    foo() === 1
}
```

Arrow functions

```
odds = evens.map(v => v + 1)
pairs = evens.m ap(v => ({ even:
v, odd: v + 1 }))
nums = evens.m ap((v, i) => v +
i)
// Statement bodies
nums.f orE ach(v => {
    if (v % 5 === 0)
        fiv es.p ush(v)
})
// Lexical this - More intuitive
handling of current object
context.
this.n ums.fo rEa ch((v) => {
    if (v % 5 === 0) this.f -
ive s.p ush(v)
})
```

Extended parameter handling

```
// Default parameters
function f (x, y = 7, z = 42) {
    return x + y + z
}
f(1) === 50
// Rest parameters
function f (x, y, ...a) {
    return (x + y) *
a.length
}
f(1, 2, " hel lo", true, 7) ===
9
// Spread operator
var params = [ " hel lo", true,
7 ]
var other = [ 1, 2, ...params ]
// [ 1, 2, " hel lo", true, 7 ]
f(1, 2, ...params) === 9
var str = " foo "
var chars = [ ...str ] // [ " -
f", " o", " o" ]
```

Template Litarals

```
var customer = { name: "Foo" }
var card = { amount: 7, product:
" Bar ", unitprice: 42 }
message = `Hello ${cust ome -
r.n ame},
want to buy ${card.am ount}
${card.pr oduct} for
a total of ${card.amount *
card.u nit price} bucks?`
```

Extended Literals

```
0b111110111 === 503
0o767 === 503
```

Enhanced Object Properties

```
// Shorthand
obj = { x, y } // => obj = { x:
x, y: y };
// Computed properties
let obj = {
    foo: " bar ",
    [ " baz " + quux() ]: 42
}
// Method properties
obj = {
    foo (a, b) {...},
    bar (x, y) { ...}
}
```

Destructuring Assignment

```
// Array matching
var list = [ 1, 2, 3 ]
var [ a, , b ] = list
[ b, a ] = [ a, b ]
// Object matching, including
deep matching
var { op: a, lhs: { op: b },
rhs: c } = getAST Node()
// Parameter context matching
function f ([ name, val ]) {
    con sol e.l og( name,
val)
```



By Roman Semko
(romansemko)

Published 10th March, 2016.
Last updated 13th May, 2016.
Page 1 of 3.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>

Destructuring Assignment (cont)

```
> }
function g ({ name: n, val: v }) {
  console.log(n, v)
}
function h ({ name, val }) {
  console.log(name, val)
}
f([ "bar", 42 ])
g({ name: "foo", val: 7 })
h({ name: "bar", val: 42 })
```

Classes

```
class Rectangle extends Shape {
  constructor (id, x, y,
    width, height) {
    // Super call
    super(id, x, y)
    this.width = width
    this.height = height
  }
  // Getter and setter
  set width (width) { this._width = width }
  get width () { return this._width }
}
class Circle extends Shape {
  constructor (id, x, y,
    radius) {
    super(id, x, y)
    this.radius = radius
  }
  do_something(x) {
    let a = 12;
    // call parent method
    super.do_something(x + a);
  }
}
```

Classes (cont)

```
> static do_whatever() {
  // static access
}
Circle.do_whatever()
```

Maps & Sets

```
// Set
let s = new Set()
s.add( "hello").add("goodbye ")
s.size === 2
s.has( "hello") === true
for (let key of s.values()) // insertion order
  console.log(key)
// Map
let m = new Map()
m.set( "hello", 42)
m.set(s, 34)
m.get(s) === 34
m.size === 2
for (let [ key, val ] of m.entries())
  console.log(key + " = " + val)
```

New Builtin methods

```
// Object.assign
var dst = { quux: 0 }
var src1 = { foo: 1, bar: 2 }
var src2 = { foo: 3, baz: 4 }
Object.assign(dst, src1, src2)
dst.quux === 0
dst.foo === 3
dst.bar === 2
dst.baz === 4
// Array.find
[ 1, 3, 4, 2 ].find(x => x > 3)
// 4
```

New Builtin methods (cont)

```
> // String repeat
"".repeat(4 * depth)
"foo".repeat(3)
// String search
"hello".startsWith("ello", 1) // true
"hello".endsWith("hell", 4) // true
"hello".includes("ell") // true
"hello".includes("ell", 1) // true
"hello".includes("ell", 2) // false
// Number type checking
Number.isNaN(42) === false
Number.isNaN(NaN) === true
Number.isFinite(Infinity) === false
Number.isFinite(-Infinity) === false
Number.isFinite(NaN) === false
Number.isFinite(123) === true
// Number safety checking
Number.isSafeInteger(42) === true
Number.isSafeInteger(9007199254740992) === false
// Number truncating
console.log(Math.trunc(42.7)) // 42
console.log(Math.trunc(0.1)) // 0
console.log(Math.trunc(-0.1)) // -0
// Number sign determination
console.log(Math.sign(7)) // 1
console.log(Math.sign(0)) // 0
console.log(Math.sign(-0)) // -0
console.log(Math.sign(-7)) // -1
console.log(Math.sign(NaN)) // NaN
```



By Roman Semko
(romansemko)

cheatography.com/romansemko/
semkocodev.com

Published 10th March, 2016.
Last updated 13th May, 2016.
Page 2 of 3.

Sponsored by [Readable.com](https://readable.com)
Measure your website readability!
<https://readable.com>

Promises

```
function msgAfterTimeout (msg, who, timeout) {
    return new Promise(( resolve, reject)
=> {
        setTimeout(() => resolve(
    } Hello ${who}!), timeout)
    })
}

msgAfterTimeout("", " Foo ", 100).then (
(msg) =>
    msgAfterTimeout (msg, " Bar ", 200)
).then ((msg) => {
    console.log('after 300ms: ${msg}')
})

// Combining promises
function fetchAsync (url, timeout, onData,
onError) {
    ...
}

let fetchPromised = (url, timeout) => {
    return new Promise(( resolve, reject)
=> {
        fetchAsync(url, timeout,
resolve, reject)
    })
}

Promise.all([
    fetchPromised("http://backend-
d/footxt", 500),
    fetchPromised("http://backend-
d/bar.txt", 500),
    fetchPromised("http://backend-
d/baz.txt", 500)
]).then(( data) => {
    let [ foo, bar, baz ] = data
    console.log('success: foo=${foo} bar=${
bar} baz=${baz}')
}, (err) => {
    console.error(`${err}`)
})
}
```



By **Roman Semko**
(romansemko)

Published 10th March, 2016.
Last updated 13th May, 2016.
Page 3 of 3.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>