New Types and Object Extensions

- Symbols
- Well-known Symbols
- Object Extensions
- String Extensions
- Number Extensions
- Math Extensions
- RegExp Extensions
- Function Extensions

"A **symbol** is a unique and immutable data type and may be used as an identifier for object properties."

Mozilla Developer Network

Symbols

```
let eventSymbol = Symbol('resize event');
console.log(typeof eventSymbol);
```

Q What shows in the console?

A symbol

```
let eventSymbol = Symbol('resize event');
console.log(eventSymbol.toString());
```

Q What shows in the console?

A Symbol(resize event)

```
const CALCULATE_EVENT_SYMBOL = Symbol('calculate event');
console.log(CALCULATE_EVENT_SYMBOL.toString());
```

Q What shows in the console?

A Symbol(calculate event)

```
let s = Symbol.for('event');
console.log(s.toString());
```

Q What shows in the console?

A Symbol(event)

```
let s = Symbol('event');
let s2 = Symbol('event');
console.log(s === s2);
```

Q What shows in the console?

A false

```
let s = Symbol.for('event');
let s2 = Symbol.for('event');
console.log(s === s2);
```

Q What shows in the console?

A true

```
let s = Symbol.for('event');
let s2 = Symbol.for('event2');
console.log(s === s2);
```

Q What shows in the console?

A false

```
let s = Symbol.for('event');
let description = Symbol.keyFor(s);
console.log(description);
```

Q What shows in the console?

A event

```
let article = {
    title: 'Whiteface Mountain',
    [Symbol.for('article')]: 'My Article'
};

let value = article[Symbol.for('article')];
console.log(value);
```

Q What shows in the console?

A My Article

```
let article = {
    title: 'Whiteface Mountain',
    [Symbol.for('article')]: 'My Article'
};

console.log( Object.getOwnPropertyNames(article) );
```

Q What shows in the console?

A ['title']

```
let article = {
    title: 'Whiteface Mountain',
    [Symbol.for('article')]: 'My Article'
};

console.log( Object.getOwnPropertySymbols(article) );
```

Q What shows in the console?

A [Symbol(article)]

Well-known Symbols

```
let Blog = function () {
};
let blog = new Blog();
console.log( blog.toString() );
```

Q What shows in the console?

A [object Object]

```
let Blog = function () {
};
Blog.prototype[Symbol.toStringTag] = 'Blog Class';
let blog = new Blog();
console.log( blog.toString() );
```

Q What shows in the console?

A [object Blog Class]

```
let values = [8, 12, 16];
console.log([].concat(values));
```

Q What shows in the console?

A [8, 12, 16]

```
let values = [8, 12, 16];
values[Symbol.isConcatSpreadable] = false;
console.log([].concat(values));
```

Q What shows in the console?

A [[8, 12, 16]]

```
let values = [8, 12, 16];
let sum = values + 100;
console.log(sum);
```

Q What shows in the console?

A 8,12,16100

```
let values = [8, 12, 16];
values[Symbol.toPrimitive] = function (hint) {
    console.log(hint);
    return 42;
};
let sum = values + 100;
console.log(sum);
```

Q What shows in the console?

A default

142

Object Extensions

```
let a = {
    x: 1
};
let b = {
    y: 2
};

Object.setPrototypeOf(a, b);
console.log(a.y);
```

Q What shows in the console?

A 2

```
let a = { a: 1 }, b = { b: 2 };

let target = {};
Object.assign(target, a, b);
console.log(target);
```

Q What shows in the console?

A {a: 1, b: 2}

```
let a = { a: 1 }, b = { a: 5, b: 2 };

let target = {};
Object.assign(target, a, b);
console.log(target);
```

Q What shows in the console?

A {a: 5, b: 2}

```
let a = { a: 1 }, b = { a: 5, b: 2 };

Object.defineProperty(b, 'c', {
    value: 10,
    enumerable: false
});

let target = {};
Object.assign(target, a, b);
console.log(target);
```

Q What shows in the console?

A {a: 5, b: 2}

```
let a = { a: 1 }, b = { a: 5, b: 2 }, c = { c: 20 };

Object.setPrototypeOf(b, c);

let target = {};
Object.assign(target, a, b);
console.log(target);
```

Q What shows in the console?

A {a: 5, b: 2}

```
let amount = NaN;
console.log(amount === amount);
```

Q What shows in the console?

A false

```
let amount = NaN;
console.log(Object.is(amount, amount));
```

Q What shows in the console?

A true

```
let amount = 0, total = -0;
console.log(amount === total);
```

Q What shows in the console?

A true

```
let amount = 0, total = -0;
console.log(Object.is(amount, total));
```

Q What shows in the console?

A false

```
let article = {
    title: 'Whiteface Mountain',
    [Symbol.for('article')]: 'My Article'
};

console.log( Object.getOwnPropertySymbols(article) );
```

Q What shows in the console?

A [Symbol(article)]

String Extensions

```
let title = 'Santa Barbara Surf Riders';
console.log(title.startsWith('Santa'));
```

Q What shows in the console?

A true

```
let title = 'Santa Barbara Surf Riders';
console.log(title.endsWith('Rider'));
```

Q What shows in the console?

A false

```
let title = 'Santa Barbara Surf Riders';
console.log(title.includes('ba'));
```

Q What shows in the console?

A true

```
var title = "Surfer's \u{1f3c4} Blog";

console.log(title);
```

Q What shows in the console?

A Surfer's 🏄 Blog

```
var surfer = "\u{1f3c4}";

console.log(surfer.length);
```

Q What shows in the console?

A 2

```
var surfer = "\u{1f30a}\u{1f3c4}\u{1f40b}";

console.log(Array.from(surfer).length);

console.log(surfer);
```

Q What shows in the console?

A 3

```
var title = "Mazatla\u0301n";

console.log(title + ' ' + title.length);
```

Q What shows in the console?

A Mazatlán 9

```
var title = "Mazatla\u0301n";

console.log(title + ' ' + title.normalize().length);
```

Q What shows in the console?

A Mazatlán 8

```
var title = "Mazatla\u0301n";

console.log(title.normalize().codePointAt(7).toString(16));
```

Q What shows in the console?

A 6e

```
console.log(String.fromCodePoint(0x1f3c4));
```

Q What shows in the console?

A 

```
let title = 'Surfer';
let output = String.raw`${title} \u{1f3c4}\n`;
console.log(output);
```

Q What shows in the console?

A Surfer \u{1f3c4}\n

```
let wave = '\u{1f30a}';
console.log(wave.repeat(10));
```

Q What shows in the console?

A

Number Extensions

```
console.log(Number.parseInt === parseInt);
```

Q What shows in the console?

A true

```
console.log(Number.parseFloat === parseFloat);
```

Q What shows in the console?

A true

```
let s = 'NaN';
console.log(isNaN(s));
console.log(Number.isNaN(s));
```

Q What shows in the console?

A
true
false

```
let s = '8000';
console.log(isFinite(s));
console.log(Number.isFinite(s));
```

Q What shows in the console?

A
true
false

```
let sum = 408.2;
console.log(Number.isInteger(sum));
```

Q What shows in the console?

A false

```
console.log(Number.isInteger(NaN));
console.log(Number.isInteger(Infinity));
console.log(Number.isInteger(undefined));
console.log(Number.isInteger(3));
```

Q What shows in the console?

A false false false
true

```
let a = Math.pow(2, 53) - 1;
console.log(Number.isSafeInteger(a));
a = Math.pow(2, 53);
console.log(Number.isSafeInteger(a));
```

Q What shows in the console?

A
true
false

```
console.log(Number.EPSILON);
console.log(Number.MAX_SAFE_INTEGER);
console.log(Number.MIN_SAFE_INTEGER);
```

Q What shows in the console?

A
2.220446049250313e-16
9007199254740991
-9007199254740991

Math Extensions

Hyperbolic Functions

cosh()

acosh()

sinh()

asinh()

tanh()

atanh()

hypot()

Arithmetic Functions

cbrt()	cube root
clz32()	count leading zeros (32 bit integers)
expm1()	equal to exp(x) - 1
log2()	log base 2
log10()	log base 10
log1p()	equal to log(x + 1)
imul()	32 bit integer multiplication

Miscellaneous Functions

sign()	the number's sign: 1, -1, 0, -0, NaN
trunc()	the integer part of a number
fround()	round to nearest 32 bit floating-point value

```
console.log(Math.sign(0));
console.log(Math.sign(-0));
console.log(Math.sign(-20));
console.log(Math.sign(20));
console.log(Math.sign(NaN));
```

Question

What shows in the console?

Answer

0
-0 (0 in Edge)
-1
1
NaN

```
console.log(Math.cbrt(27));
```

Question

What shows in the console?

Answer

3

```
console.log(Math.trunc(27.1));
console.log(Math.trunc(-27.9));
```

Question

What shows in the console?

Answer

27
-27

RegExp Extensions

```
let pattern = /\u{1f3c4}/;
console.log(pattern.test('🏄'));
```

Question

What shows in the console?

Answer

false

```
let pattern = /\u{1f3c4}/u;
console.log(pattern.test('🏄'));
```

Question

What shows in the console?

Answer

true

```
let pattern = /^.Surfer/;
console.log(pattern.test('🏄Surfer'));
```

Question

What shows in the console?

Answer

false

```
let pattern = /^.Surfer/u;
console.log(pattern.test('🏄Surfer'));
```

Question

What shows in the console?

Answer

true

```
let pattern = /900/y;
console.log(pattern.lastIndex);
console.log(pattern.test('800900'));
```

Question

What shows in the console?

Answer

0
false

```
let pattern = /900/y;
pattern.lastIndex = 3;
console.log(pattern.test('800900'));
```

Question

What shows in the console?

Answer

true

```
let pattern = /900/yg;
console.log(pattern.flags);
```

Question

What shows in the console?

Answer

gy

(Order will be "gimuy")

Function Extensions

```
let fn = function calc() {
    return 0;
};
console.log(fn.name);
```

Question

What shows in the console?

Answer

calc

```
let fn = function() {
    return 0;
};
console.log(fn.name);
```

Question

What shows in the console?

Answer

fn

```
let fn = function() {
    return 0;
};
let newFn = fn;
console.log(newFn.name);
```

Question

What shows in the console?

Answer

fn

```
class Calculator {
    constructor() {
    }
    add() {
    }
}
let c = new Calculator();
console.log(Calculator.name);
console.log(c.add.name);
```

Question

What shows in the console?

Answer

Calculator
add

Function.name isn't writable

but

it's configurable with
Object.defineProperty()

Summary

Symbols

```
let eventSymbol = Symbol('resize event');
console.log(typeof eventSymbol);
```

Summary

Well-known Symbols

```
let values = [8, 12, 16];
values[Symbol.isConcatSpreadable] = false;
console.log([].concat(values));
```

Summary

Object Extensions

```
let a = {
    x: 1
};
let b = {
    y: 2
};

Object.setPrototypeOf(a, b);
console.log(a.y);
```

Summary

String Extensions

```
var title = "Surfer's \u{1f3c4} Blog";

console.log(title);
```

Summary

Number Extensions

```
let s = 'NaN';
console.log(isNaN(s));
console.log(Number.isNaN(s));
```

Summary

Math Extensions

```
console.log(Math.sign(0));
console.log(Math.sign(-0));
console.log(Math.sign(-20));
console.log(Math.sign(20));
console.log(Math.sign(NaN));
```

Summary

RegExp Extensions

```
let pattern = /\u{1f3c4}/u;
console.log(pattern.test('🏄'));
```

Summary

Function Extensions

```
let fn = function() {
    return 0;
};
console.log(fn.name);
```