Новые возможности ES6 и ES7

- Новые возможности ES6
- Новые возможности ES7

Новые возможности ES6

- **Константы (const)**
- **Блочная видимость (let)**
- **Стрелочные функции, лексическое this**
- Классы
- Улучшения в литералах объектов
- Строки-шаблоны
- Деструктурирование
- Параметры функций
- Итераторы, for..of
- Генераторы
- Юникод

Новые возможности ES6 (продолжение)

- Модули
- Map, Set, WeakMap, WeakSet
- Proxy, Reflection
- Symbol
- Подклассы для встроенных классов
- **Промисы**
- Добавления Math, Number, String, Array, Object
- RegExp lastIndex
- Двоичные и восьмеричные литералы
- i18n

Классы (ES6)

Определение класса

```
class Shape {
    constructor (id, x, y) {
        this.id = id
        this.move(x, y)
    }
    move (x, y) {
        this.x = x
        this.y = y
    }
}
```

Классы (ES6)

Наследование

```
class Circle extends Shape {
    constructor (id, x, y, radius) {
        super(id, x, y)
        this.radius = radius
    }
}
```

Классы (ES6) - продолжение

```
var aggregation = (baseClass, ...mixins) => {
    let base = class _Combined extends baseClass {
        ...
    }
    ...
    return base
}

class Colored {...}
class ZCoord {...}
class Shape {...}

class Rectangle extends
   aggregation(Shape, Colored, ZCoord) {}

var rect = new Rectangle(7, 42)
rect.z     = 1000
rect.color = "red"
console.log(rect.x, rect.y, rect.z, rect.color)
```

Классы (ES6) - продолжение

JS

Доступ к базовому классу

```
class Shape {
    ...
    toString () {return `Shape(${this.id})`}
}
class Rectangle extends Shape {
    constructor (id, x, y, width, height) {
        super(id, x, y)
        ...
    }
    toString () {return "Rectangle > " + super.toString()}
}
class Circle extends Shape {
    constructor (id, x, y, radius) {
        super(id, x, y)
        …
    }
    toString () {return "Circle > " + super.toString()}
}
```

Классы (ES6) - продолжение

JS

Статические члены класса

```
class Rectangle extends Shape {
    …
    static defaultRectangle () {
        return new Rectangle("default", 0, 0, 100, 100)
    }
}
class Circle extends Shape {
    …
    static defaultCircle () {
        return new Circle("default", 0, 0, 100)
    }
}
var defRectangle = Rectangle.defaultRectangle()
var defCircle    = Circle.defaultCircle()
```

Классы (ES6) - продолжение

Геттеры и сеттеры

```
class Rectangle {
  constructor (width, height) {
    this._width  = width
    this._height = height
  }
  set width  (width)  { this._width = width }
  get width  ()       { return this._width  }
  set height (height) { this._height = height }
  get height ()       { return this._height }
  get area   ()       { return this._width*this._height }
}

var r = new Rectangle(50, 20)
r.area === 1000
```

Литералы объектов (ES6)

```
obj = { x, y }     // { x: x, y: y }
```

```
let obj = {
    foo: "bar",
    [ "baz" + quux() ]: 42
}
```

```
obj = {
    foo (a, b) {...},
    bar (x, y) {...},
    *quux (x, y) {...}
}
```

Строки-шаблоны (ES6)

```
var message = `Hello ${customer.name},
want to buy ${card.amount} ${card.product}
for
a total of ${card.amount * card.unitprice}
bucks?`
```

```
get`http://example.com/foo?bar=${bar +
baz}&quux=${quux}`
```

```
String.raw `foo\n${ 42 }bar` // "foo\\n42bar"
```

Деструктурирование (ES6)

```
var list = [ 1, 2, 3 ]
var [ a, , b ] = list
[ b, a ] = [ a, b ]
```

```
var { op, lhs, rhs }=getSNode();
var { op: a, lhs: { op: b }, rhs: c }=getNode();
```

```
var obj = { a: 1 }
var list = [ 1 ]
var { a, b = 2 } = obj
var [ x, y = 2 ] = list
```

Деструктурирование (ES6)

```
function f([ name, val ]) {...}
f([ "bar", 42 ])

function g ({ name: n,val: v }) {...}  // n,v
g({ name: "foo", val:  7 })

function h ({ name, val }) {...}
h({ name: "bar", val: 42 })
```

```
var list = [ 7, 42 ]
var [ a = 1, b = 2, c = 3, d ] = list
// a === 7
// b === 42
// c === 3
// d === undefined
```

Параметры функций (ES6)

```
function f (x, y = 7, z = 42) {return x + y + z}
f(1) // 50
```

```
function f (x, y, ...a) {
    return (x + y) * a.length
}
f(1, 2, "hello", true, 7) // 9
```

```
var params = [ "hello", true, 7 ]
var other = [ 1, 2, ...params ]
  // [ 1, 2, "hello", true, 7 ]
f(1, 2, ...params) // 9

var str = "foo"
var chars = [ ...str ] // [ "f", "o", "o" ]
```

Итераторы, for..of (ES6)

```
let fibonacci = {
    [Symbol.iterator]() {
        let pre = 0, cur = 1
        return {
           next () {
               [ pre, cur ] = [ cur, pre + cur ]
               return { done: false, value: cur }
           }
        }
    }
}

for (let n of fibonacci) {
    if (n > 1000)
        break
    console.log(n)
}
```

Генераторы (ES6)

```
let fibonacci = {
    *[Symbol.iterator]() {
        let pre = 0, cur = 1
        for (;;) {
            [ pre, cur ] = [ cur, pre + cur ]
            yield cur
        }
    }
}

for (let n of fibonacci) {
    if (n > 1000)
        break
    console.log(n)
}
```

Генераторы (ES6) - продолжение

```
function* range (start, end, step) {
    while (start < end) {
        yield start
        start += step
    }
}

for (let i of range(0, 10, 2)) {
    console.log(i) // 0, 2, 4, 6, 8
}
```

Генераторы (ES6) - продолжение

```
class Clz {
    * bar () {
        …
    }
}
let Obj = {
    * foo () {
        …
    }
}
```

Юникод (ES6)

```
"吉".length === 2
"吉".match(/./u)[0].length === 2
"吉" === "\uD842\uDFB7"
"吉" === "\u{20BB7}"
"吉".codePointAt(0) == 0x20BB7

for (let codepoint of "吉") {
  console.log(codepoint);
}
```

Модули (ES6)

```
//  lib/math.js
export function sum (x, y) { return x + y }
export var pi = 3.141593

//  someApp.js
import * as math from "lib/math"
console.log("2π = " + math.sum(math.pi, math.pi))

//  otherApp.js
import { sum, pi } from "lib/math"
console.log("2π = " + sum(pi, pi))
```

Модули (ES6)

```
//  lib/mathplusplus.js
export * from "lib/math"
export var e = 2.71828182846
export default (x) => Math.exp(x)

//  someApp.js
import exp, { pi, e } from "lib/mathplusplus"
console.log("e^{п} = " + exp(pi))
```

Map, Set, WeakMap, WeakSet (ES6)

```
let s = new Set()
s.add("hello").add("goodbye").add("hello")
s.size === 2
s.has("hello") === true
for (let key of s.values()) console.log(key)
```

```
let m = new Map()
let s = Symbol()
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)
```

WeakMap, WeakSet – не хранят указатели

Proxy, Reflection (ES6)

```
let target = {foo: "Welcome, foo"}
let proxy = new Proxy(target, {
    get (receiver, name) {
        return name in receiver ?
           receiver[name] : `Hello, ${name}`;
    }
})
proxy.foo   === "Welcome, foo"
proxy.world === "Hello, world"
```

```
let obj = { a: 1 }
Object.defineProperty(obj, "b", { value: 2 })
obj[Symbol("c")] = 3
Reflect.ownKeys(obj) // [ "a", "b", Symbol(c) ]
```

Символы (ES6)

Тип данных для property id

```
Symbol("foo") !== Symbol("foo")
const foo = Symbol()
const bar = Symbol()
typeof foo === "symbol"
typeof bar === "symbol"
let obj = {}
obj[foo] = "foo"
obj[bar] = "bar"
JSON.stringify(obj) // {}
Object.keys(obj) // []
Object.getOwnPropertyNames(obj) // []
Object.getOwnPropertySymbols(obj) // [ foo, bar ]
```

Символы (ES6) - продолжение

Глобальные символы

```
Symbol.for("app.foo") === Symbol.for("app.foo")
const foo = Symbol.for("app.foo")
const bar = Symbol.for("app.bar")
Symbol.keyFor(foo) === "app.foo"
Symbol.keyFor(bar) === "app.bar"
typeof foo === "symbol"
typeof bar === "symbol"
let obj = {}
obj[foo] = "foo"
obj[bar] = "bar"
JSON.stringify(obj) // {}
Object.keys(obj) // []
Object.getOwnPropertyNames(obj) // []
Object.getOwnPropertySymbols(obj) // [ foo, bar ]
```

Array, Object (ES6)

```
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
```

```
[ 1, 3, 4, 2 ].find(x => x > 3) // 4
[ 1, 3, 4, 2 ].findIndex(x => x > 3) // 2
```

String (ES6)

```
" ".repeat(4 * depth)
"foo".repeat(3)
```

```
"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, Math (ES6)

```
Number.isNaN(42) === false
Number.isNaN(NaN) === true

Number.isFinite(-Infinity) === false
Number.isFinite(NaN) === false
Number.isFinite(123) === true

Number.isSafeInteger(42) === true
Number.isSafeInteger(9007199254740992) === false

console.log(0.1 + 0.2 === 0.3) // false
console.log(Math.abs((0.1 + 0.2) - 0.3) < Number.EPSILON) //
true

console.log(Math.trunc(42.7)) // 42
console.log(Math.trunc(-0.1)) // -0

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
```

RegExp lastIndex (ES6)

```
var re = /(hi)?/g;

console.log(re.exec('hi'));  // ["hi","hi"]
console.log(re.lastIndex);   // 2

console.log(re.exec('hi'));  // ["",undefined]
console.log(re.lastIndex);   // 2

re.lastIndex = 0;
console.log(re.exec('hi'));  // ["hi","hi"]
```

Двоичные и восьмеричные литералы (ES6)

```
0b111110111 // 503
```

```
0o767 // 503
```

i18n (ES6)

```
var list = [ "ä", "a", "z" ]
var l10nDE = new Intl.Collator("de")
l10nDE.compare("ä", "z") === -1
console.log(list.sort(l10nDE.compare))
// [ "a", "ä", "z" ]
```

```
var l10nDE = new Intl.NumberFormat("de-DE")
l10nDE.format(1234567.89) === "1.234.567,89"
```

```
var l10nGBP = new Intl.NumberFormat(
   "en-GB", { style: "currency", currency: "GBP" });
l10nGBP.format(100200300.40) === "£100,200,300.40"
```

```
var l10nDE = new Intl.DateTimeFormat("de-DE")
l10nDE.format(new Date("2015-01-02")) === "2.1.2015"
```

Новые возможности ES7

JS

- Array.prototype.includes()

```
let countries = ['UK', 'USA', 'Egypt', 'France'];

countries.includes('UK', 1);          // => false
countries.includes('Ireland', 1); // => true
countries.includes('USA', 6);        // => false
```

- Возведение в степень (**)

```
2**3      // 8
```

Новые возможности ES6 и ES7

- Новые возможности ES6
- Новые возможности ES7

Новые возможности ES6

- **Константы (const)**
- **Блочная видимость (let)**
- **Стрелочные функции, лексическое this**
- Классы
- Улучшения в литералах объектов
- Строки-шаблоны
- Деструктурирование
- Параметры функций
- Итераторы, for..of
- Генераторы
- Юникод

Новые возможности ES6 (продолжение)

- Модули
- Map, Set, WeakMap, WeakSet
- Proxy, Reflection
- Symbol
- Подклассы для встроенных классов
- **Промисы**
- Добавления Math, Number, String, Array, Object
- RegExp lastIndex
- Двоичные и восьмеричные литералы
- i18n

Классы (ES6)

JS

Определение класса

```
class Shape {
    constructor (id, x, y) {
        this.id = id
        this.move(x, y)
    }
    move (x, y) {
        this.x = x
        this.y = y
    }
}
```

Классы (ES6)

JS

Наследование

```
class Circle extends Shape {
    constructor (id, x, y, radius) {
        super(id, x, y)
        this.radius = radius
    }
}
```

Классы (ES6) - продолжение

```
var aggregation = (baseClass, ...mixins) => {
    let base = class _Combined extends baseClass {
        ...
    }
    ...
    return base
}

class Colored {...}
class ZCoord {...}
class Shape {...}

class Rectangle extends
   aggregation(Shape, Colored, ZCoord) {}

var rect = new Rectangle(7, 42)
rect.z     = 1000
rect.color = "red"
console.log(rect.x, rect.y, rect.z, rect.color)
```

Классы (ES6) - продолжение

JS

Доступ к базовому классу

```
class Shape {
    ...
    toString () {return `Shape(${this.id})`}
}
class Rectangle extends Shape {
    constructor (id, x, y, width, height) {
        super(id, x, y)
        ...
    }
    toString () {return "Rectangle > " + super.toString()}
}
class Circle extends Shape {
    constructor (id, x, y, radius) {
        super(id, x, y)
        …
    }
    toString () {return "Circle > " + super.toString()}
}
```

Классы (ES6) - продолжение

JS

Статические члены класса

```
class Rectangle extends Shape {
    …
    static defaultRectangle () {
        return new Rectangle("default", 0, 0, 100, 100)
    }
}
class Circle extends Shape {
    …
    static defaultCircle () {
        return new Circle("default", 0, 0, 100)
    }
}
var defRectangle = Rectangle.defaultRectangle()
var defCircle    = Circle.defaultCircle()
```

Классы (ES6) - продолжение

Геттеры и сеттеры

```
class Rectangle {
  constructor (width, height) {
    this._width  = width
    this._height = height
  }
  set width  (width)  { this._width = width }
  get width  ()       { return this._width  }
  set height (height) { this._height = height }
  get height ()       { return this._height }
  get area   ()       { return this._width*this._height }
}

var r = new Rectangle(50, 20)
r.area === 1000
```

Литералы объектов (ES6)

```
obj = { x, y }    // { x: x, y: y }
```

```
let obj = {
    foo: "bar",
    [ "baz" + quux() ]: 42
}
```

```
obj = {
    foo (a, b) {...},
    bar (x, y) {...},
    *quux (x, y) {...}
}
```

Строки-шаблоны (ES6)

JS

```
var message = `Hello ${customer.name},
want to buy ${card.amount} ${card.product}
for
a total of ${card.amount * card.unitprice}
bucks?`
```

```
get`http://example.com/foo?bar=${bar +
baz}&quux=${quux}`
```

```
String.raw `foo\n${ 42 }bar` // "foo\\n42bar"
```

Деструктурирование (ES6)

```
var list = [ 1, 2, 3 ]
var [ a, , b ] = list
[ b, a ] = [ a, b ]
```

```
var { op, lhs, rhs }=getSNode();
var { op: a, lhs: { op: b }, rhs: c }=getNode();
```

```
var obj = { a: 1 }
var list = [ 1 ]
var { a, b = 2 } = obj
var [ x, y = 2 ] = list
```

Деструктурирование (ES6)

```
function f([ name, val ]) {...}
f([ "bar", 42 ])

function g ({ name: n,val: v }) {...}  // n,v
g({ name: "foo", val:  7 })

function h ({ name, val }) {...}
h({ name: "bar", val: 42 })
```

```
var list = [ 7, 42 ]
var [ a = 1, b = 2, c = 3, d ] = list
// a === 7
// b === 42
// c === 3
// d === undefined
```

Параметры функций (ES6)

```
function f (x, y = 7, z = 42) {return x + y + z}
f(1) // 50
```

```
function f (x, y, ...a) {
    return (x + y) * a.length
}
f(1, 2, "hello", true, 7) // 9
```

```
var params = [ "hello", true, 7 ]
var other = [ 1, 2, ...params ]
  // [ 1, 2, "hello", true, 7 ]
f(1, 2, ...params) // 9

var str = "foo"
var chars = [ ...str ] // [ "f", "o", "o" ]
```

Итераторы, for..of (ES6)

```
let fibonacci = {
    [Symbol.iterator]() {
        let pre = 0, cur = 1
        return {
           next () {
               [ pre, cur ] = [ cur, pre + cur ]
               return { done: false, value: cur }
           }
        }
    }
}

for (let n of fibonacci) {
    if (n > 1000)
        break
    console.log(n)
}
```

Генераторы (ES6)

```
let fibonacci = {
    *[Symbol.iterator]() {
        let pre = 0, cur = 1
        for (;;) {
            [ pre, cur ] = [ cur, pre + cur ]
            yield cur
        }
    }
}

for (let n of fibonacci) {
    if (n > 1000)
        break
    console.log(n)
}
```

Генераторы (ES6) - продолжение

```
function* range (start, end, step) {
    while (start < end) {
        yield start
        start += step
    }
}

for (let i of range(0, 10, 2)) {
    console.log(i) // 0, 2, 4, 6, 8
}
```

Генераторы (ES6) - продолжение

```
class Clz {
     * bar () {
         …
     }
}
let Obj = {
     * foo () {
         …
     }
}
```

Юникод (ES6)

JS

```
"𠮷".length === 2
"𠮷".match(/./u)[0].length === 2
"𠮷" === "\uD842\uDFB7"
"𠮷" === "\u{20BB7}"
"𠮷".codePointAt(0) == 0x20BB7

for (let codepoint of "𠮷" {
   console.log(codepoint);
}
```

Модули (ES6)

```
//  lib/math.js
export function sum (x, y) { return x + y }
export var pi = 3.141593

//  someApp.js
import * as math from "lib/math"
console.log("2π = " + math.sum(math.pi, math.pi))

//  otherApp.js
import { sum, pi } from "lib/math"
console.log("2π = " + sum(pi, pi))
```

Модули (ES6)

```
//  lib/mathplusplus.js
export * from "lib/math"
export var e = 2.71828182846
export default (x) => Math.exp(x)

//  someApp.js
import exp, { pi, e } from "lib/mathplusplus"
console.log("e^{π} = " + exp(pi))
```

Map, Set, WeakMap, WeakSet (ES6)

```
let s = new Set()
s.add("hello").add("goodbye").add("hello")
s.size === 2
s.has("hello") === true
for (let key of s.values()) console.log(key)
```

```
let m = new Map()
let s = Symbol()
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)
```

WeakMap, WeakSet – не хранят указатели

Proxy, Reflection (ES6)

```
let target = {foo: "Welcome, foo"}
let proxy = new Proxy(target, {
    get (receiver, name) {
        return name in receiver ?
           receiver[name] : `Hello, ${name}`;
    }
})
proxy.foo   === "Welcome, foo"
proxy.world === "Hello, world"
```

```
let obj = { a: 1 }
Object.defineProperty(obj, "b", { value: 2 })
obj[Symbol("c")] = 3
Reflect.ownKeys(obj) // [ "a", "b", Symbol(c) ]
```

Символы (ES6)

Тип данных для property id

```
Symbol("foo") !== Symbol("foo")
const foo = Symbol()
const bar = Symbol()
typeof foo === "symbol"
typeof bar === "symbol"
let obj = {}
obj[foo] = "foo"
obj[bar] = "bar"
JSON.stringify(obj) // {}
Object.keys(obj) // []
Object.getOwnPropertyNames(obj) // []
Object.getOwnPropertySymbols(obj) // [ foo, bar ]
```

Символы (ES6) - продолжение

Глобальные символы

```
Symbol.for("app.foo") === Symbol.for("app.foo")
const foo = Symbol.for("app.foo")
const bar = Symbol.for("app.bar")
Symbol.keyFor(foo) === "app.foo"
Symbol.keyFor(bar) === "app.bar"
typeof foo === "symbol"
typeof bar === "symbol"
let obj = {}
obj[foo] = "foo"
obj[bar] = "bar"
JSON.stringify(obj) // {}
Object.keys(obj) // []
Object.getOwnPropertyNames(obj) // []
Object.getOwnPropertySymbols(obj) // [ foo, bar ]
```

Array, Object (ES6)

```
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
```

```
[ 1, 3, 4, 2 ].find(x => x > 3) // 4
[ 1, 3, 4, 2 ].findIndex(x => x > 3) // 2
```

String (ES6)

```
" ".repeat(4 * depth)
"foo".repeat(3)
```

```
"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, Math (ES6)

```
Number.isNaN(42) === false
Number.isNaN(NaN) === true

Number.isFinite(-Infinity) === false
Number.isFinite(NaN) === false
Number.isFinite(123) === true

Number.isSafeInteger(42) === true
Number.isSafeInteger(9007199254740992) === false

console.log(0.1 + 0.2 === 0.3) // false
console.log(Math.abs((0.1 + 0.2) - 0.3) < Number.EPSILON) //
true

console.log(Math.trunc(42.7)) // 42
console.log(Math.trunc(-0.1)) // -0

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
```

RegExp lastIndex (ES6)

```
var re = /(hi)?/g;

console.log(re.exec('hi'));  // ["hi","hi"]
console.log(re.lastIndex);   // 2

console.log(re.exec('hi'));  // ["",undefined]
console.log(re.lastIndex);   // 2

re.lastIndex = 0;
console.log(re.exec('hi'));  // ["hi","hi"]
```

Двоичные и восьмеричные литералы (ES6)

```
0b111110111 // 503
```

```
0o767 // 503
```

i18n (ES6)

```
var list = [ "ä", "a", "z" ]
var l10nDE = new Intl.Collator("de")
l10nDE.compare("ä", "z") === -1
console.log(list.sort(l10nDE.compare))
// [ "a", "ä", "z" ]
```

```
var l10nDE = new Intl.NumberFormat("de-DE")
l10nDE.format(1234567.89) === "1.234.567,89"
```

```
var l10nGBP = new Intl.NumberFormat(
   "en-GB", { style: "currency", currency: "GBP" });
l10nGBP.format(100200300.40) === "£100,200,300.40"
```

```
var l10nDE = new Intl.DateTimeFormat("de-DE")
l10nDE.format(new Date("2015-01-02")) === "2.1.2015"
```

Новые возможности ES7

- Array.prototype.includes()

```
let countries = ['UK', 'USA', 'Egypt', 'France'];

countries.includes('UK', 1);      // => false
countries.includes('Ireland', 1); // => true
countries.includes('USA', 6);     // => false
```

- Возведение в степень (**)

```
2**3     // 8
```