ES 2015 Сегодня

Краткий обзор возможностей ECMAScript 2015
и введение в практическую разработку.

- JavaScript
- ECMAScript
- ES5
- Harmony
- ES6
- ECMAScript 2015
- ES7
- ES.Next

ES6 vs ES2015 vs Harmony

- Синтаксический сахар
- Новая семантика
- Новые типы данных
- Изменения в стандартной библиотеке
- Синтаксическая поддержка модулей

Новые возможности

```
function Person(firstName, lastName) {
    this._firstName = firstName;
    this._lastName = lastName;
}

Person.prototype.getFullName = function () {
    return firstName + ' ' + lastName;
};

function inherits(ctor, superCtor) {
    ctor.super_ = superCtor;
    ctor.prototype = Object.create(superCtor.prototype);
}

function Employee(firstName, lastName, salary) {
    Employee.super_.call(this, firstName, lastName);
    this._salary = salary;
}

inherits(Employee, Person);

Employee.prototype.increaseSalary = function (amount) {
    this._salary += amount;
};
```

Class syntax

```
class Person {
    constructor(firstName, lastName) {
        this._firstName = firstName;
        this._lastName = lastName;
    }

    getFullName() {
        return firstName + ' ' + lastName;
    }
}

class Employee extends Person {
    constructor(firstName, lastName, salary) {
        super(firstName, lastName);
        this._salary = salary;
    }

    increaseSalary(amount) {
        this._salary += amount;
    }
}
```

Class syntax

```
var squares = [1, 2, 3].map(function (x) {
    return x * x;
});

this.on('new-data', function (data) {
    this.update(data);
    this.render();
}.bind(this));
```

```
var squares = [1, 2, 3].map(x => x * x);

this.on('new-data', data => {
    this.update(data);
    this.render();
});
```

Arrow functions

Function arguments

- Default arguments

```
function init(options = {}) {
    // do something with options.
}
```

- Rest

```
function sum(start, ...nums) {
    return nums.reduce((res, x) => res + x, start);
}
```

- Spread

```
var nums = [5, 4, 3, 9, 5, 6];

Math.max(...nums);
```

```
var [a, b] = ['foo', 'bar'];
var {x, y} = { x: 1, y: 42 };

// swap vars
[a, b] = [b, a];

// use rest
var [one, two, ...tail] = [1, 2, 3, 4, 5];
```

Destructuring

```
var obj = {
    ['foo' + 'Bar']: 42,

    shorthandMethod() {
        return 'I am so cool';
    },

    'this is awkward'() {
        return 'O_o';
    },

    get prop() {
        console.log('Access denied');
    },

    set prop(value) {
        console.log('Ok, boss...');
    }
};
```

```
var x = 1, y = 42;
var obj = { x, y };
// obj === { x: 1, y: 42 }
```

Object literal

```
var firstName = 'John', lastName = 'Doe';
`${firstName[0]}. ${lastName}`; // => 'J. Doe'
```

```
function fahrenheit(strings, ...vals) {
    return strings.join(`${(vals[0] * 1.8 + 32).toFixed(2)} ℉`);
}

var temp = 36.6;
fahrenheit`The body temperature around ${temp} is considered normal.`
// => 'The body temperature around 97.88 ℉ is considered normal.'
```

Template strings

- Symbol
- Map
- Set
- Proxy
- Reflect
- Promise

New types

```
function showOverlay(overlayType, sender, showOverlayCallback) {
    for (var i = 0; i < _adapters.length; i++) { // TODO: DRY
        var settings = _adapters.settings[i];
        var playerAdapter = _adapters[i];

        if (settings && settings.overlays) {
            var sett;
            for (var k = 0; k < settings.overlays.length; k++)
                if (settings.overlays[k].type === overlayType)
                    sett = settings.overlays[k];
                    break;
                }
            }

            if (sett && !sett.disabled) {
                // todo: refactor this with moving the logic to appPlugin
                var elements = null;
                switch (overlayType) {
                    case Realeyesit.Models.OverlayType.MAIN_AREA:
                    case Realeyesit.Models.OverlayType.DISMISSED_DOORHANGER_NOTICE:
                    case Realeyesit.Models.OverlayType.WAITING_OVERLAY:

                        if (sender && sender.getDisplayArea && sender === playerAdapter) {
                            elements = [sender.getDisplayArea];
                        }

                        if (overlayType === Realeyesit.Models.OverlayType.DISMISSED_DOORHANGER_NOTICE) {
                            _currentAdapter = sender;
                            setTimeout(self.hideOverlay.bind(self, Realeyesit.Models.OverlayType.DISMISSED_DOORHANG
                        }

                        break;
                    case Realeyesit.Models.OverlayType.INACTIVE_AREA:
                        if (playerAdapter.getDisplayArea && sender !== playerAdapter) {
                            elements = [playerAdapter.getDisplayArea];
                        }
                        break;
                    case Realeyesit.Models.OverlayType.OPTOUT:
                        if (playerAdapter.getDisplayArea) {
                            elements = [playerAdapter.getDisplayArea];
```

FFFFFFF
FFFFFFF
FFFFFF
FFFUU
UUUU
UUUU
UUUU
UUUU
UUUU-

WeakMap, WeakSet

```
var playerRegistry = new WeakMap();

function addPlayer(adapter, settings) {
    playerRegistry.set(adapter, settings);
}

function startCollection(sender) {
    senderSettings = playerRegistry.get(sender);
    sender.pause();
    senderSettings.overlay.show()
        .then(sender.play.bind(sender));
}
```

WeakMap

```
for (var i = 0; i < 10; i++) {
    setTimeout(function () {
        console.log(i);
    }, 0);
}

for (let i = 0; i < 10; i++) {
    setTimeout(function () {
        console.log(i);
    }, 0);
}
```

const + let

```
let iterable = {
    [Symbol.iterator]() {
        let step = 0;
        return {
            next() {
                if (step < 10) {
                    return { value: `step: ${++step}`, done: false };
                }

                return { value: undefined, done: true };
            }
        };
    }
};

for (let val of iterable) {
    console.log(val);
}
```

Iterators and for-of

```
function* nums() {
    for (let i = 0;; i++) {
        yield i;
    }
}

function* take(n, iterable) {
    for (let x of iterable) {
        if (n <= 0) return;
        n--;
        yield x;
    }
}

let firstFour = [...take(4, nums())];
```

Generators

```
function co(genFunc) {
    let genObj = genFunc();

    function run(promiseResult = undefined) {
        let item = genObj.next(promiseResult);
        if (!item.done) {
            item.value
                .then(result => run(result))
                .catch(error => {
                    genObj.throw(error);
                });
        }
    }

    run();
}

co(function* () {
    try {
        let result = yield fetch('http://example.com/api')
            .then(res => res.text())

        console.log('Result: ', result);
    } catch (e) {
        console.log('Fetch error: ' + e);
    }
});
```

Generators

```
(async function () {
    try {
        let result = await fetch('https://example.com/api')
            .then(res => res.text())

        console.log('Result: ', result);
    } catch (e) {
        console.log('Fetch error: ' + e);
    }
}())
```

async - await

- Object
- String
- RegExp
- Array
- Number
- Math

StdLib extensions

```
// lib.js
export default {
    foo() {
        //
    }
};

// main.js
import coollib from 'lib';

coollib.foo();
```

Modules

```
// method extraction
Promise.resolve(123).then(console.log.bind(console));
Promise.resolve(123).then(::console.log);

// virtual methods
_.takeWhile(getPlayers()
    .map(x => x.character())
    , x => x.strength > 100)
    .forEach(x => console.log(x));

getPlayers()
    .map(x => x.character())
    ::takeWhile(x => x.strength > 100)
    .forEach(x => console.log(x));
```

Оператор bind (::)

Котик

```
"use strict";

var nums = [5, 4, 3, 9, 5, 6];
Math.max.apply(Math, nums);
```

```
$traceurRuntime.ModuleStore.getAnonymousModule(function() {
  "use strict";
  var $__1;
  var nums = [5, 4, 3, 9, 5, 6];
  ($__1 = Math).max.apply($__1, $traceurRuntime.spread(nums));
  return {};
});
```

Babel vs traceur

```
  .getOwnPropertyNames(object) -> array
  .seal(object) -> object, cap for ie8-
  .freeze(object) -> object, cap for ie8-
  .preventExtensions(object) -> object, cap for ie8-
  .isSealed(object) -> bool, cap for ie8-
  .isFrozen(object) -> bool, cap for ie8-
  .isExtensible(object) -> bool, cap for ie8-
  .keys(object) -> array
Array
  .isArray(var) -> bool
  #slice(start?, end?) -> array, fix for ie7-
  #join(string = ',') -> string, fix for ie7-
  #indexOf(var, from?) -> int
  #lastIndexOf(var, from?) -> int
  #every(fn(val, index, @), that) -> bool
  #some(fn(val, index, @), that) -> bool
  #forEach(fn(val, index, @), that) -> void
  #map(fn(val, index, @), that) -> array
```

Babel + core.js

- ES6 React Boilerplate
- WebPack Boilerplate
- github.com/este/este
- Тысячи их…

Boilerplates

- Для поддержки синтаксиса ES6 используется компилятор Babel.
- Проверка качества кода обеспечивается двумя утилитами, ESLint и JSCS.
- За сборку проекта отвечает browserify и babelify.
- Минификацию кода делает UglifyJS2.
- Unit тесты используют mocha, chai и sinon.
- Отчет о покрытии кода тестами генерируется с помощью istanbul и isparta.
- Integration тесты запускаются в karma.
- В качестве таск раннера используется Gulp.

ES6 browser boilerplate

- Google Closure Compiler
- Flow
- TypeScript
- And others

Аннотации vs typescript

- Давид Классен
- https://github.com/DavidKlassen

Спасибо за внимание!