ES6

ECMA-262

# EcmaScript 6

Üstün Özgür
@ustunozgur
ustunozgur.com

sellercrowd

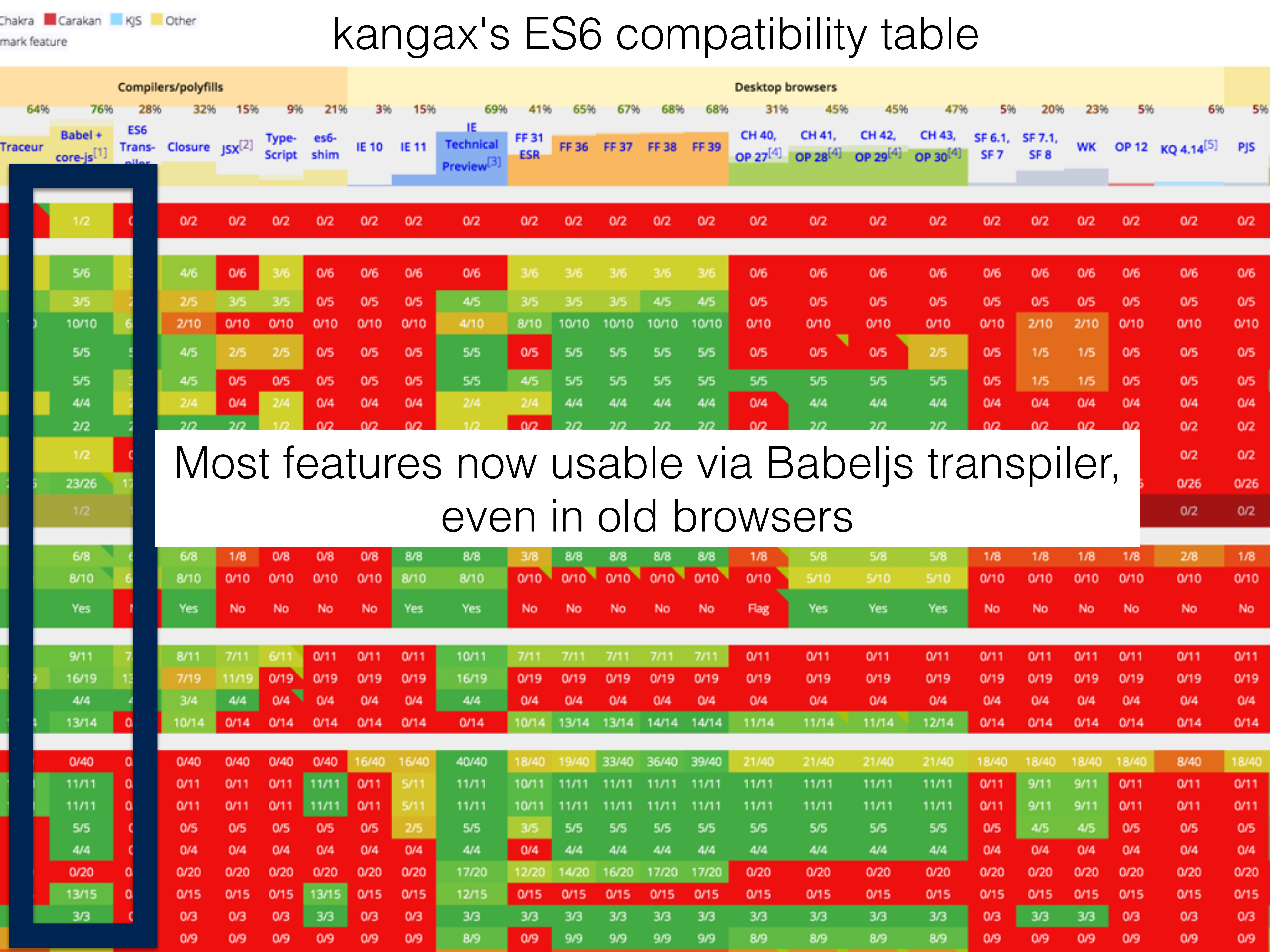
# History

- JavaScript: Implementation
- EcmaScript: Specification
- Netscape JavaScript ve Microsoft JScript, ActionScript
- ECMA standard organization project no. 262
- TC-39: Technical Committee 39

# History

- ES 3: 1999
- ES 4: Abandoned
- ES 5 : 2009 and 5.1: 2011
  - 258 pages
- ES 6: Draft ready
  - 657 pages
  - ETA June 2015
  - Harmony
- ES 7: Already in progress

# kangax's ES6 compatibility table



Chakra 🟥 Carakan 🟦 KJS 🟨 Other
mark feature

| Compilers/polyfills | | | | | | | Desktop browsers | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 64% | 76% | 28% | 32% | 15% | 9% | 21% | 3% | 15% | 69% | 41% | 65% | 67% | 68% | 68% | 31% | 45% | 45% | 47% | 5% | 20% | 23% | 5% | 6% | 5% |
| Traceur | Babel + core-js[1] | ES6 Trans- | Closure | JSX[2] | Type-Script | es6-shim | IE 10 | IE 11 | IE Technical Preview[3] | FF 31 ESR | FF 36 | FF 37 | FF 38 | FF 39 | CH 40, OP 27[4] | CH 41, OP 28[4] | CH 42, OP 29[4] | CH 43, OP 30[4] | SF 6.1, SF 7 | SF 7.1, SF 8 | WK | OP 12 | KQ 4.14[5] | PJS |
| | 1/2 | | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 |
| | 5/6 | | 4/6 | 0/6 | 3/6 | 0/6 | 0/6 | 0/6 | 0/6 | 3/6 | 3/6 | 3/6 | 3/6 | 3/6 | 0/6 | 0/6 | 0/6 | 0/6 | 0/6 | 0/6 | 0/6 | 0/6 | 0/6 | 0/6 |
| | 3/5 | | 2/5 | 3/5 | 3/5 | 0/5 | 0/5 | 0/5 | 4/5 | 3/5 | 3/5 | 3/5 | 4/5 | 4/5 | 0/5 | 0/5 | 0/5 | 0/5 | 0/5 | 0/5 | 0/5 | 0/5 | 0/5 | 0/5 |
| | 10/10 | | 2/10 | 0/10 | 0/10 | 0/10 | 0/10 | 0/10 | 4/10 | 8/10 | 10/10 | 10/10 | 10/10 | 10/10 | 0/10 | 0/10 | 0/10 | 0/10 | 0/10 | 2/10 | 2/10 | 0/10 | 0/10 | 0/10 |
| | 5/5 | | 4/5 | 2/5 | 2/5 | 0/5 | 0/5 | 0/5 | 5/5 | 0/5 | 5/5 | 5/5 | 5/5 | 5/5 | 0/5 | 0/5 | 0/5 | 2/5 | 0/5 | 1/5 | 1/5 | 0/5 | 0/5 | 0/5 |
| | 5/5 | | 4/5 | 0/5 | 0/5 | 0/5 | 0/5 | 0/5 | 5/5 | 4/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 0/5 | 1/5 | 1/5 | 0/5 | 0/5 | 0/5 |
| | 4/4 | | 2/4 | 0/4 | 2/4 | 0/4 | 0/4 | 0/4 | 2/4 | 2/4 | 4/4 | 4/4 | 4/4 | 4/4 | 0/4 | 4/4 | 4/4 | 4/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 |
| | 2/2 | | 2/2 | 2/2 | 1/2 | 0/2 | 0/2 | 0/2 | 1/2 | 0/2 | 2/2 | 2/2 | 2/2 | 2/2 | 0/2 | 2/2 | 2/2 | 2/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 |
| | 1/2 | | | | | | | | | | | | | | | | | | | | | | 0/2 | 0/2 |
| | 23/26 | | | | | | | | | | | | | | | | | | | | | | 0/26 | 0/26 |
| | 1/2 | | | | | | | | | | | | | | | | | | | | | | 0/2 | 0/2 |
| | 6/8 | | 6/8 | 1/8 | 0/8 | 0/8 | 0/8 | 8/8 | 8/8 | 3/8 | 8/8 | 8/8 | 8/8 | 8/8 | 1/8 | 5/8 | 5/8 | 5/8 | 1/8 | 1/8 | 1/8 | 1/8 | 2/8 | 1/8 |
| | 8/10 | | 8/10 | 0/10 | 0/10 | 0/10 | 0/10 | 8/10 | 8/10 | 0/10 | 0/10 | 0/10 | 0/10 | 0/10 | 0/10 | 5/10 | 5/10 | 5/10 | 0/10 | 0/10 | 0/10 | 0/10 | 0/10 | 0/10 |
| | Yes | | Yes | No | No | No | No | Yes | Yes | No | No | No | No | No | Flag | Yes | Yes | Yes | No | No | No | No | No | No |
| | 9/11 | | 8/11 | 7/11 | 6/11 | 0/11 | 0/11 | 0/11 | 10/11 | 7/11 | 7/11 | 7/11 | 7/11 | 7/11 | 0/11 | 0/11 | 0/11 | 0/11 | 0/11 | 0/11 | 0/11 | 0/11 | 0/11 | 0/11 |
| | 16/19 | | 7/19 | 11/19 | 0/19 | 0/19 | 0/19 | 0/19 | 16/19 | 0/19 | 0/19 | 0/19 | 0/19 | 0/19 | 0/19 | 0/19 | 0/19 | 0/19 | 0/19 | 0/19 | 0/19 | 0/19 | 0/19 | 0/19 |
| | 4/4 | | 3/4 | 4/4 | 0/4 | 0/4 | 0/4 | 0/4 | 4/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 |
| | 13/14 | | 10/14 | 0/14 | 0/14 | 0/14 | 0/14 | 0/14 | 0/14 | 10/14 | 13/14 | 13/14 | 14/14 | 14/14 | 11/14 | 11/14 | 11/14 | 12/14 | 0/14 | 0/14 | 0/14 | 0/14 | 0/14 | 0/14 |
| | 0/40 | | 0/40 | 0/40 | 0/40 | 0/40 | 16/40 | 16/40 | 40/40 | 18/40 | 19/40 | 33/40 | 36/40 | 39/40 | 21/40 | 21/40 | 21/40 | 21/40 | 18/40 | 18/40 | 18/40 | 18/40 | 8/40 | 18/40 |
| | 11/11 | | 0/11 | 0/11 | 0/11 | 11/11 | 0/11 | 5/11 | 11/11 | 10/11 | 11/11 | 11/11 | 11/11 | 11/11 | 11/11 | 11/11 | 11/11 | 11/11 | 0/11 | 9/11 | 9/11 | 0/11 | 0/11 | 0/11 |
| | 11/11 | | 0/11 | 0/11 | 0/11 | 11/11 | 0/11 | 5/11 | 11/11 | 10/11 | 11/11 | 11/11 | 11/11 | 11/11 | 11/11 | 11/11 | 11/11 | 11/11 | 0/11 | 9/11 | 9/11 | 0/11 | 0/11 | 0/11 |
| | 5/5 | | 0/5 | 0/5 | 0/5 | 0/5 | 0/5 | 2/5 | 5/5 | 3/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 0/5 | 4/5 | 4/5 | 0/5 | 0/5 | 0/5 |
| | 4/4 | | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 | 4/4 | 0/4 | 4/4 | 4/4 | 4/4 | 4/4 | 4/4 | 4/4 | 4/4 | 4/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 |
| | 0/20 | | 0/20 | 0/20 | 0/20 | 0/20 | 0/20 | 0/20 | 17/20 | 12/20 | 14/20 | 16/20 | 17/20 | 17/20 | 0/20 | 0/20 | 0/20 | 0/20 | 0/20 | 0/20 | 0/20 | 0/20 | 0/20 | 0/20 |
| | 13/15 | | 0/15 | 0/15 | 0/15 | 13/15 | 0/15 | 0/15 | 12/15 | 0/15 | 0/15 | 0/15 | 0/15 | 0/15 | 0/15 | 0/15 | 0/15 | 0/15 | 0/15 | 0/15 | 0/15 | 0/15 | 0/15 | 0/15 |
| | 3/3 | | 0/3 | 0/3 | 0/3 | 3/3 | 0/3 | 0/3 | 3/3 | 3/3 | 3/3 | 3/3 | 3/3 | 3/3 | 3/3 | 3/3 | 3/3 | 3/3 | 0/3 | 3/3 | 3/3 | 0/3 | 0/3 | 0/3 |
| | | | 0/9 | 0/9 | 0/9 | 0/9 | 0/9 | 0/9 | 8/9 | 0/9 | 9/9 | 9/9 | 9/9 | 9/9 | 8/9 | 8/9 | 8/9 | 8/9 | 0/9 | 0/9 | 0/9 | 0/9 | 0/9 | 0/9 |

Most features now usable via Babeljs transpiler, even in old browsers

# Overview

- Let ve Const keywords vs Var
- Changes in functions
- Changes in objects and destructuring
- Classes
- Template strings
- Promises

# Let and Const

- JS, looks like C or Java due to curly brace blocks
  - but no block scope
- Vars are function scoped
- Let ve Const block scoped
- Causes confusion for beginners

# Var and Let

Hoisting

```
function foo () {
  var i;
  console.log(i); undefined

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

  console.log(i); 10
}
```

# Var ve Let

```
function foo () {

  console.log(i); // ERROR

  for (let i = 0; i < 10; i++) {
      console.log(i); // i defined only in this block
  }

  console.log(i); // ERROR
}
```

## Const

```
const PI = 3.14;

PI = 3; // ERROR
```

Better to have values than variables for less bugs

*DEMO*
*let_const.js*

# Changes in Functions

- Default parameters **function foo(name="Ustun")**
- Rest parameters **function foo(name, ...rest)**
- Destructured parameters **function foo({name, surname})**

# Default Params

```
function hello(name="Ustun", greeting="Hello") {
    console.log(greeting + " " + name);
  }


  hello();
  hello("Ahmet");
  hello("Mehmet", "Hola");
```

Not keyword params like in Python!

~~hello(greeting="Hello", name="Ozgur")~~

*DEMO*

*functions_default_params.js*

# Rest params

```
function sum(firstValue, ...rest) {

var total = firstValue;

        for (var i = 0; i < rest.length; i++) {

          total += rest[i];

      }

       return total;

}
```

```
sum(15, 1, 2, 3);

rest = [1, 2, 3]
```

*DEMO*

*functions_rest_params.js*

## Spread operator

- Math.max(1, 2, 3); 3
- Math.max([1, 2, 3]); // NaN
- var a = [1, 2, 3]; Math.max(a); // NaN
- Math.max.apply(Math, a); // 3
- Math.max(...a); // 3

## Parameter Destructuring

```
function hello(name, options) {
    var greeting;
    var lang = options.lang;
    if (lang == "en") greeting = "Hello";
    if (lang == "es") greeting = "Hola";
    return greeting + " " + name;
}
```

```
function hello(name, {lang}) {
var greeting;
    if (lang == "en") greeting = "Hello";
    if (lang == "es") greeting = "Hola";
    return greeting + " " + name;
}
```

*DEMO*
*functions_param_destructuring.js*

```
function setCookie(name, value, options) {

    options = options || {};

    var secure = options.secure,
        path = options.path,
        domain = options.domain,
        expires = options.expires;

    // ...
}

setCookie("type", "js", {
    secure: true,
    expires: 60000
});
```

```
function setCookie(name, value, { secure, path, domain, expires }) {

    // ...
}
```

# Arrow functions

```
var bar = (a, b) => a + b;
```

```
var foo = function (a,b) {

 return a + b; }

 foo(1,2) ; // 3

 bar(1,2); // 3
```

- var nums = [1, 2, 3, 4];
- nums.filter(x => x % 2 === 1)
- nums.reduce((a,b) => a * b)

*DEMO*
*arrow_functions.js*

# Arrow functions and this keyword

```
var x = {
    name: "Ustun",
    hello: function () {
        var that = this;
        var helper = function () {
            console.log("Name ", ~~this~~.name);
                                 that
        };
        helper();
    }
};
x.hello()
```

```
var x = {
    name: "Ustun",
    hello: function () {

        var helper = function () {
            console.log("Name ", this.name);
        }.bind(this);
        helper();
    }
};
x.hello()
```

## this refers to the value where function is defined, not called (lexical scope vs dynamic scope)

```
var x = {
    name: "Ustun",
    hello: function () {
        var helper = () => {
            console.log("Name ", this.name);
        };
        helper();
    }
};
x.hello()
```

*DEMO*
*this.js*

## Changes in Objects

# Destructuring

```
var ustun = {name: "Ustun", lastname: "Ozgur"}

var name = ustun.name;

var lastname = ustun.lastname;

var {name, lastname} = {name: "Ustun", lastname: "Ozgur"}



var {name: nombre} = ustun;
```

## Destructuring

- Array destructuring
- var [a,b] = [1,2];
- var [a,b] = [b,a]; // Swap
- Deep destructuring possible

# Shorthand for Object Creation

```
age = 30; name = "Ustun"; location = "Turkey";

ustun = {name: name, age: age, location: location};



age = 45; name = "Jose"; location = "Barcelona";

ahmet = {name, age, location};
```

***DEMO***
***objects.js***

## Shorthand for Object Creation

```
var ustun = {
  name: "Ustun",
  sayName: function () {
    console.log("I'm " + this.name);
  }
}
```

```
var ustun = {
  name: "Ustun",
  sayName() {
    console.log("I'm " + this.name);
  }
}
```

## Template Strings

```
name = "Ustun", age = 30;

console.log("I'm " + name + ". Yasim " + age);

console.log(`I'm ${name}. My age ${age}`);

console.log(`This spans

multiple lines`);
```

# Tagged Template Strings

- functionName`Hello ${name}`;
- safe`Hello ${name}`;
- uppercase`Hello ${name}`;
- var safe = function (literals, ...variables) { ...}
- var uppercase = function (literals, ...variables) {...}

# Class Keyword

- class and extends
- constructor
- transpiled to prototypes

# Classes

```
class Human {

    constructor(name, age) {

        this.name = name;

        this.age = age;

        this.party = null;

    }
```

```
class Ogrenci extends Human {

    constructor(name, age, school) {

        super(name, age);

        this.school = school;

    }
```

## Other features

- Modules
- Promises
- Generators

# Babel.js

- Transpiler
- babel source.js > destination.js
- babel --experimental source.js
- require('babel/polyfill')
- require("babel/register")
  - in node Modules
  - all require'd modules will be processed by babel automatically

# More Info

- Wiki: http://wiki.ecmascript.org/
- Understanding ECMAScript 6
    - https://leanpub.com/understandinges6/
- Taslaklar: http://wiki.ecmascript.org/doku.php?id=harmony:specification_drafts
- babeljs.io
- https://github.com/lukehoban/es6features
- http://kangax.github.io/compat-table/es6/

# Thank you!

- @ustunozgur
- https://github.com/ustun/ecmascript6-presentation
- ustun@ustunozgur.com
- **NEXT PUBLIC APPEARANCE:**
- **React.js Workshop at AtTheFrontend Conference in Copenhagen, May 26**
- **http://www.atthefrontend.dk/sessions/react-workshop-2/**

## Extras

## Modules

- import myfunc from mylib;
- export myfunc;

# Example

```
var links = document.getElementsByTagName('a')

for (var i = 0, len = links.length; i < len; i++){

   links[i].namedEventListener('click', function(e){

        alert('You clicked on link ' + i)

    }, false)

}
```

*DEMO*

*var_problem.html*

## Solution

Enclose the var in a function

```
var links = document.getElementsByTagName('a')
for (var i = 0, len = links.length; i < len; i++){
  (function (j) {
    links[j].namedEventListener('click', function(e){
        alert('You clicked on link ' + j)
     }, false)
  })(i);
}
```

*DEMO*

*var_cozum.html*

# Solution: Let

```
var links = document.getElementsByTagName('a')

for (let i = 0, len = links.length; i < len; i++){

   links[i].namedEventListener('click', function(e){

        alert('You clicked on link ' + i)

    }, false)

}
```

*DEMO*

*let_cozum.html*

## Computed Properties

```
var fieldName = "firstName";

var a = {

 fieldName: "ustun";

 }
```

```
a.fieldName == "ustun"

a.firstName ? undefined

a[fieldName] = "ustun";

a.firstName == "ustun"
```

```
var a = {

 [fieldName]: "ustun";

}
```

## List Comprehensions

- `var x = [for (i of [0, 1, 2, 3]) i * i];`
- `[ 0, 1, 4, 9 ]`

- `var y = [for (i of [0, 1, 2, 3]) if (i % 2 === 0) i * i * i];`
- `[ 0, 8 ]`

## Promises

```
student = findStudent(123)

className = findClass(student)

school = findSchool(className)
```

## Callbacks & Pyramid of Doom

```
student = findStudent(123, function (student) {

  findClass(student, function (className) {

    findSchool(className, function (school) {

        console.log(school);

   }}}
```

## Pyramid of Doom Sol'n

```
findStudent(123)

  .then(findClass)

  .then(findSchool)

  .then(function (school) {console.log(school);}

  .catch(function () { console.log("error")})
```

# Promise

- new Promise(resolve, reject)
- Promise.all([promise1, promise2]).then
- Promise.race([promise1, promise2]).then

*DEMO*

# Generators

- Functions that **yield** instead of return
- Different run-to-completion that normal functions
- yield keyword
- function * syntax
- Bidirectional communication with caller
  - can yield
  - can be yielded (i.e. get value from caller via next())

## Generators

```
var foo = function* () {
 yield 2;
 yield 3;
 yield 4;
}
var iterator = foo();
iterator.next(); // {value: 2, done: false}
iterator.next(); // {value: 3, done: false}
iterator.next(); // {value: 4, done: false}
```

# Generators are Iterators

```
var foo = function* () {
 yield 2;
 yield 3;
 yield 4;
}
 for (let x of foo()) {
       console.log(x);
  }
 // 2, 3, 4
```

# Sync Example

```
function main() {

    var result1 = requestSync( "http://some.url.1" );

    var data = JSON.parse( result1 );



    var result2 = requestSync( "http://some.url.2?id=" +
data.id );

    var resp = JSON.parse( result2 );

    console.log( "The value you asked for: " + resp.value );

}
```

## ASync Example

```
function *main() {
    var result1 = yield request( "http://some.url.1" );
    var data = JSON.parse( result1 );

    var result2 = yield request( "http://some.url.2?id=" + data.id );
    var resp = JSON.parse( result2 );
    console.log( "The value you asked for: " + resp.value );
}
var it = main();
it.next();
```

# ASync Example

```
function request(url) {

    // this is where we're hiding the asynchronicity,

    // away from the main code of our generator

    // `it.next(..)` is the generator's iterator-resume

    // call

    makeAjaxCall( url, function(response){

        it.next( response );

    } );

    // Note: nothing returned here!

}
```

* from http://davidwalsh.name/async-generators:

Very good series on Generators

## Generator Version of Student Example:
## Generators + Promises
## Async Code Almost the Same as Sync Code

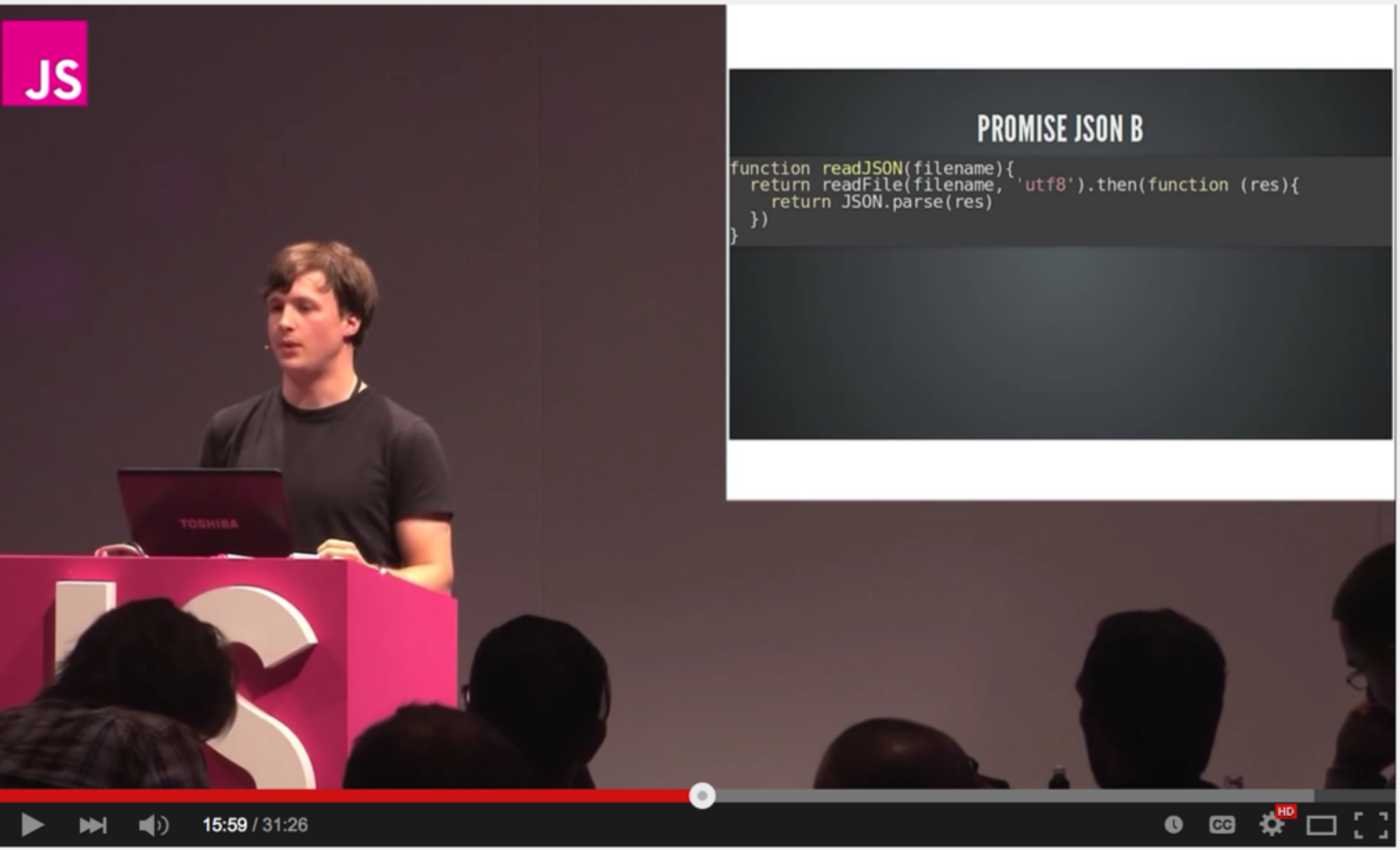
```
spawn(function *() {

    student = yield findStudent(123)

    className = yield findClass(student)

    school = yield findSchool(className)

});
```

*Using task.js or with ES7 await instead of yield keyword*

*http://taskjs.org/*

```
spawn(function*() {
    var data = yield $.ajax(url);
    $('#result').html(data);
    var status = $('#status').html('Download complete.');
    yield status.fadeIn().promise();
    yield sleep(2000);
    status.fadeOut();
});
```

```
require("babel/polyfill");

var makeAjaxCall = function (url, cb) {

    setTimeout(function () {

        cb("Result for " + url);

    }, 100);};

function request(url) {

    makeAjaxCall( url, function(response){

        it.next( response );

    });}

var main = function *() {

    var result1 = yield request( "http://some.url.1" );

    var data = encodeURIComponent(result1.toUpperCase());

    var result2 = yield request( "http://some.url.2?id=" + data );

    var resp = result2;

    console.log( "The value you asked for: " + resp );

}; var it = main(); it.next();
```



PROMISE JSON B

```
function readJSON(filename){
  return readFile(filename, 'utf8').then(function (res){
    return JSON.parse(res)
  })
}
```

Forbes Lindesay: Promises and Generators: control flow utopia -- JSConf EU 2013

JSConf

13,236

## See Also:

- https://www.youtube.com/watch?v=qbKWsbJ76-s
- http://davidwalsh.name/async-generators