Sebastiano Armeli
@sebarmeli

2014
CEE-SEC(R)
Software Engineering
Conference in Russia

http://html5hub.com/wp-content/uploads/2013/11/es6-hiway-sign.png

Sebastiano Armeli

@sebarmeli

Sebastiano Armeli-Battana

MVC Applies to JavaScript

developer press

2014 CEE-SEC(R) Software Engineering Conference in Russia

ES6

History

1995
(May) **B. Eich invented Mocha**
(Sep) Mocha renamed to LiveScript
(Dec) LiveScript renamed to **JavaScript**

1996
JSScript

1997
(June) **ECMA-262 Ed.1**
by **TC39** committee

1998
ECMA-262 Ed.2

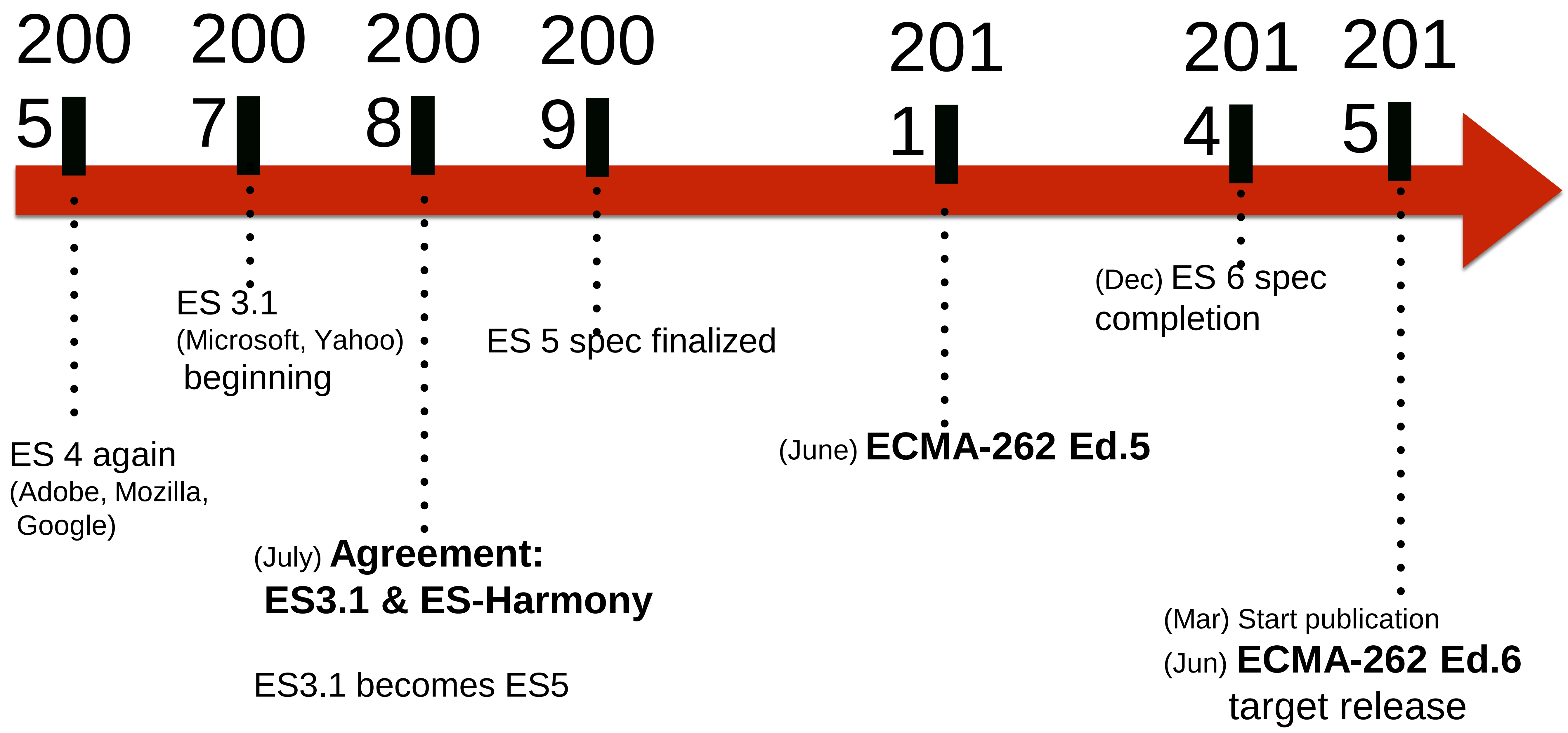
1999
ECMA-262 Ed.3

2000
ECMA-262 Ed.4 started

2003
ECMA-262 Ed.4 abandoned

2014 CEE-SEC(R)
Software Engineering
Conference in Russia

History


2005
ES 4 again (Adobe, Mozilla, Google)
2007
ES 3.1 (Microsoft, Yahoo) beginning
2008
(July) **Agreement: ES3.1 & ES-Harmony**
ES3.1 becomes ES5
2009
ES 5 spec finalized
2011
(June) **ECMA-262 Ed.5**
2014
(Dec) ES 6 spec completion
2015
(Mar) Start publication
(Jun) **ECMA-262 Ed.6** target release

ECMA

ES 4

ECMA-262

TC39

ES.Next

ES-Harmony

ES 6

ES 7

es-discuss

Summary

Arrow Functions

Scoping / Destructing / Parameters

Iteration & Generators

Collections

Modularity / Classes / Templates

API improvements

Proxies

Summary

Arrow Functions

Scoping / Destructing / Parameters

Iteration & Generators

Collections

Modularity / Classes / Templates

API improvements

Proxies

2014 CEE-SEC(R) Software Engineering Conference in Russia

(Fat) arrow function

ES6

```
var y = (x) => x + 1
```

ES5

```
var y = function(x) {
  return x + 1;
}
```

(Fat) arrow function

ES6

```
var y = (x) => x + 1
```

ES5

```
var y = function(x) {
  return x + 1;
}
```

Syntax sugar

2014 CEE-SEC(R) Software Engineering Conference in Russia

(Fat) arrow function

ES6

```
var y = (x) => x + 1
```

ES5

```
var y = function(x) {
  return x + 1;
}
```

Syntax sugar

Lexical `this` binding

(Fat) arrow function

ES6

```
var y = (x) => x + 1
```

ES5

```
var y = function(x) {
  return x + 1;
}
```

Syntax sugar

Lexical `this` binding

No constructor

ES6

```
var y = (x) =>
  {return x + 1}
```

ES5

```
var y = function(x) {
  return x + 1;
}
```

2014 CEE-SEC(R)
Software Engineering Conference in Russia

ES6

```
var y = (x) =>
  {return x + 1}
```

```
var z = (x, y) =>
  ({
     x: x,
     y: y
  })
```

ES5

```
var y = function(x) {
  return x + 1;
}
```

```
var z = function(x, y) {
  return {
    x: x,
    y: y
  };
}
```

ES3

```
var obj = {
  doIt: function(){},
  handle: function(){
    var that = this;
    document.addEventListener('click', function(e) {
      that.doIt();
    });
  }
}
```

ES3

```
var obj = {
  doIt: function(){},
  handle: function(){
    var that = this;
    document.addEventListener(‘click’, function(e) {
      that.doIt();
    });
  }
}
```

ES5

```
var obj = {
  doIt: function(){},
  handle: function(){
    document.addEventListener(‘click’, function(e) {
      this.doIt();
    }.bind(this));
  }
}
```

ES6

```
var obj = {
  doIt: function(){},
  handle: function(){
    document.addEventListener(‘click’,
      (e) => this.doIt());
  }
}
```

2014 CEE-SEC(R) Software Engineering Conference in Russia

```
Object.getPrototypeOf(() => {})
Object.getPrototypeOf(() => {})
```

```
Function.prototype
```

When to use ‘function’ ?

Constructors

Generators

(Methods in object literals)

Summary

Arrow Functions

Scoping / Destructing / Parameters

Iteration & Generators

Collections

Modularity / Classes / Templates

API improvements

Proxies

2014 CEE-SEC(R)
Software Engineering
Conference in Russia

Block Scoping

Each **BLOCK** has got its lexical environment

let/**const** bind variables to the lexical environment

Variables declared with let/const are **NOT** hoisted

var vs let

```
(function() {
  console.log(y) // “undefined”
  if (true) {
    var y = “value”;
  }
  console.log(y) // “value”
}());
```

var vs let

```
(function() {
  console.log(y) // “undefined”
  if (true) {
    var y = “value”;
  }
  console.log(y) // “value”
}());
```

```
(function() {
  if (true) {
    let y = “value”;
  }
  console.log(y) // ERROR!!
}());
```

const

```
(function() {
  const X;
  X = "foo"; // ERROR: x unitialized
}());



(function() {
  const X = "foo";
  X = "foo2"; // ERROR: x is read-only
}());
```

Destructing array

```
var [x,y] = [‘a’, ‘b’];

console.log(x); // ‘a’

console.log(y); // ‘b’


var [x,y] = [y, x];

console.log(x); // ‘b’
```

2014 CEE-SEC(R)
Software Engineering
Conference in Russia

Destructing object

```
var obj = {width: 50, height: 100};


var {width: w, height: h} = obj;
var {width, height} = obj;


console.log(width); // 50
console.log(w); // 50
console.log(height); // 100
console.log(h); // 100
```

Parameter default values

```
function(foo) {
 foo = foo || “a”;
}
```

Parameter default values

```
function(foo) {
 foo = foo || “a”;
}
```

```
function(foo = “a”) {}
```

Rest parameters

```
function fn(…args) {
  console.log(args); //[“a”, “b”, “c”]
  args.forEach(function(arg) {
    console.log(arg);
  });
}

fn(“a”, “b”, “c”);

// a
// b
// c
```

Rest parameters

```
function fn(a, …args) {
  console.log(args); //[“b”, “c”]
  args.forEach(function(arg) {
    console.log(arg);
  });
}

fn(“a”, “b”, “c”);

// b
// c
```

Summary

Arrow Functions

Scoping / Destructing / Parameters

Iteration & Generators

Collections

Modularity / Classes / Templates

API improvements

Proxies

2014 CEE-SEC(R) Software Engineering Conference in Russia

for-of

`for-in` limitations

`for-of` loop on ‘iterables’ and iterators

Arrays/Sets/Maps are ’iterables’

2014 CEE-SEC(R) Software Engineering Conference in Russia

for-of

```
var array = [“a”, “b”, “c”];

for (let el of array) {
  console.log(el);
}

// “a”
// “b”
// “c”
```

Iterable

```
{ @@iterator: function() -> iterator }
```

Iterators

```
{ next: function() -> any }
```

2014 CEE-SEC(R) Software Engineering Conference in Russia

Iterator

`Iterator` from Array, Map, Set

```
var array = [“a”, “b”, “c”];

array.entries() // Array Iterator
array.keys() // Array Iterator
```

Generator

```
function* g() {
 yield “a”;
 yield “b”;
}
```

generator ‘constructor’

```
var generator = g();
generator.next(); //{ value: “a”, done: false}
generator.next(); //{ value: “b”, done:
false}generator.next(); //{ value: undefined,
                                   done: true}
```

```
function* g() {
 yield "a";
 var retVal = yield "b";
 return retVal;
}

var generator = g();

generator.next().value; //"a"
generator.next().value; //"b"
generator.next("c").value; //"c"
```

Promise

```
function* asyncFn() {
     var data = yield getUser();
     doSomethingElse(data);
}

function run(genFunction) {
    var generator = genFunction();
    generator.next().value.then(function(val){
        generator.next(val);
     }, function(err) {
         generator.throw(err);
     });
}

run(asyncFn);
```

Summary

Arrow Functions

Scoping / Destructing / Parameters

Iteration & Generators

Collections

Modularity / Classes / Templates

API improvements

Proxies

2014 CEE-SEC(R) Software Engineering Conference in Russia

Set

NO duplicates values

Set

NO duplicates values

Different types in a set

Set

NO duplicates values

Different types in a set

```
add(key)/ has(key) / delete(key)
```

2014 CEE-SEC(R) Software Engineering Conference in Russia

Set

NO duplicates values

Different types in a set

`add(key)/ has(key) / delete(key)`

`values() -> Iterator`

2014 CEE-SEC(R) Software Engineering Conference in Russia

```
let countries = new Set();
countries.add(“US”);
countries.add(“Italy”);
countries.add(“US”);

countries // Set [“US”, “Italy”]
```

Map

```
{"foo" : "bar"
}value
```

Map

```
{"foo" : "bar"
}value
```

Keys can be objects

2014 CEE-SEC(R) Software Engineering Conference in Russia

Map

```
{“foo” : “bar”
}value
```

Keys can be objects

```
get(key); has(key); set(key,val)
```

2014 CEE-SEC(R)
Software Engineering
Conference in Russia

Map

```
{"foo" : "bar"
}value
```

Keys can be objects

```
get(key); has(key); set(key,val)
```

```
delete(key); clear(); forEach();
```

```
let dict = new Map();
dict.set(“A”, 1); dict.set(“B”, 2);

dict                        // Map {“A”: 1, “B”: 2}
```

```
let dict = new Map();
dict.set("A", 1); dict.set("B", 2);

dict                    // Map {"A": 1, "B": 2}

dict.get("A");          // "1"
dict.delete("B");
```

WeakMap

Avoid memory leaks

WeakMap

Avoid memory leaks

Reference to the key obj held weakly

WeakMap

Avoid memory leaks

Reference to the key obj held weakly

Keys must be an objects

WeakMap

Avoid memory leaks

Reference to the key obj held weakly

Keys must be an objects

No iterators methods

2014 CEE-SEC(R)
Software Engineering Conference in Russia

Object properties

with

Map / WeakMap

Summary

Arrow Functions

Scoping / Destructing / Parameters

Iteration & Generators

Collections

Modularity / Classes / Templates

API improvements

Proxies

2014 CEE-SEC(R) Software Engineering Conference in Russia

Object Literal

```
let obj = {
   __proto__: parentObj,
   meth1(a,b) {
   }
};
```

Module

lib/ads.js

```
export function register(ad) {
  return ad;
}
```

app.js

```
import {register} from “ads”;
var app = {
 doIt: function() {
   register({});
  }
}
export app;
```

Class

```
class Animal {
  constructor(name) {
    this.name = name;
  }
  toString() {
    return “This is: ” + this.name;
  }
}
```

Subclass - super

```
class Cat extends Animal {
  constructor(name, ownerName) {
    super(name);
    this.ownerName = ownerName;
  }

  toString() {
    return super() + " owned by " +  this.ownerName;
  }
}
```

```
class Animal {
  constructor(name) {
    this.name = name;
  }
  toString() {
    return “This is: ” + this.name;
  }
}

class Cat extends Animal {
  constructor(name, ownerName) {
    super.constructor(name);
    this.ownerName = ownerName;
  }

  toString() {
    return super.toString() + “ owned by ” +
this.ownerName;
  }
}
```

```
function Animal(name) {
  this.name = name;
}

Animal.prototype.toString = function() {
  return "This is: " + this.name;
};

function Cat(name, ownerName) {
  Animal.call(this, name);
  this.ownerName = ownerName;
}

Cat.prototype = Object.create(Animal.prototype);
Cat.prototype.constructor = Cat;
Cat.prototype.parent = Animal;

Cat.prototype.toString = function() {
  var super = Animal.prototype.toString.call(this);
  return super + " owned by " +   this.ownerName;
};
```

Template strings

```
var a = “hello”;
var b = “world”;

`${a} ${b}!`
```

Template strings

```
var a = “hello”;
var b = “world”;

`${a} ${b}!`



var multiline = `Hello
                 world
                 !!!`;
```

Software Engineering Conference in Russia

Summary

Arrow Functions

Scoping / Destructing / Parameters

Iteration & Generators

Collections

Modularity / Classes / Templates

API improvements

Proxies

String methods

```
String.prototype.startsWith(str)
=> boolean

String.prototype.endsWith(str)
=> boolean

String.prototype.contains(str)
=> boolean

String.prototype.repeat(num)
=> string
```

Number methods

```
Number.isInteger(num) => boolean

Number.isNaN(num) => boolean

Number.isFinite(num) => boolean

...
```

2014 CEE-SEC(R)
Software Engineering Conference in Russia

Array methods

```
Array.from(obj) => Array
Array.of(…args) => Array


Array.prototype.entries => Iterator

Array.prototype.keys => Iterator

Array.prototype.values => Iterator
```

```
var divs = document.querySelectorAll("div");

Array.from(divs);

// [<div></div>, </div></div>]


Array.of(10, 11);

// [10, 11]
```

```
var array = ["a", "b", "c"];

for (let [index, el] of array.entries()) {
  console.log(index, el); // 0 "a"
                           // 1 "b"
                           // 2 "c"
}

for (let index of array.keys()) {
  console.log(index);
}

for (let el of array.values()) {
  console.log(el);
}
```

Object methods

```
Object.setPrototypeOf(obj, proto)

Object.assign(obj, mixin)

Object.is(value1, value2)
```

Math methods

```
Math.log2(num)    => num

Math.log10(num)   => num

Math.sinh(num)    => num

Math.cosh(num)    => num
```

…

Summary

Arrow Functions

Scoping / Destructing / Parameters

Iteration & Generators

Collections

Modularity / Classes / Templates

API improvements

Proxies

2014 CEE-SEC(R) Software Engineering Conference in Russia

Proxies

```
Proxy(targetObject, interceptors)
```

Meta-programming

Different use cases (logging, mocking)

Proxies

```
var obj = {num: 1};

obj = new Proxy(obj, {
  set: function (target, property, value) {
    target[property] = value + 1;
  }
});

obj.num = 2    // [[Set]]
console.log(obj.num); // 3
```

2014 CEE-SEC(R) Software Engineering Conference in Russia

Proxies

```
function createDefensiveObject(target) {

    return new Proxy(target, {
        get: function(target, property) {
            if (property in target) {
                return target[property];
            } else {
                throw new ReferenceError();
            }
        }
    });
}

var obj = createDefensiveObject({name: “Seb”});
console.log(obj.lastname); //ReferenceError
```

http://www.nczonline.net/blog/2014/04/22/creating-defensive-objects-with-es6-proxies/

Recap

Arrow Functions

Scoping / Destructing / Parameters

Iteration & Generators

Collections

Modularity / Classes / Templates

API improvements

Proxies

Other Features..

Promises

Better Unicode support

Optimized tail calls

Symbols

ES6 today

Traceur compiler (Google)

es6-transpiler

es6-module-transpiler (Square)

defs.js

6to5

2014 CEE-SEC(R) Software Engineering Conference in Russia

http://wiki.ecmascript.org

https://people.mozilla.org/~jorendorff/es6-draft.html

http://kangax.github.io/compat-table/es6/

http://esdiscuss.org/

2014
CEE-SEC(R)
Software Engineering
Conference in Russia

Sebastiano Armeli
@sebarmeli

2014
CEE-SEC(R)
Software Engineering
Conference in Russia

http://goo.gl/4OOD73

Sebastiano Armeli
@sebarmeli