

Modernes JavaScript mit ECMAScript 2015

Christian Kaltepoth / @chkal

Slides: <http://bit.ly/javaland16-es2015>

Christian Kaltepoth
Senior Developer @ ingenit

christian@kaltepoth.de / [@chkal](#)

<http://blog.kaltepoth.de>

ECMAScript

aka

JavaScript

History

- ECMAScript 1: 1997
- ECMAScript 2: 1998 (alignment)
- ECMAScript 3: 1999 (regex, exceptions, ...)
- ECMAScript 4: killed in 2007
- ECMAScript 5: 2009 (strict mode, JSON, ...)
- ECMAScript 6: 2015 (major update)
- ECMAScript 7: 2016? (WIP)

Show me code!

Block Scope

ES5 Scoping

```
function someFunction() {  
  
    for( var i = 0; i < 4; i++ ) {  
        var j = i * i;  
    }  
  
    console.log( j );  
    // > ?  
  
}
```

ES5 Hoisting

```
function someFunction() {  
    var j; // hoisting  
  
    for( var i = 0; i < 4; i++ ) {  
        j = i * i;  
    }  
  
    console.log( j );  
    // > 9  
}
```


ES2015 Block Scope

```
function someFunction() {  
  
    for( var i = 0; i < 4; i++ ) {  
        let j = i * i;  
    }  
  
    console.log( j );  
    // > ReferenceError: j is not defined  
  
}
```

ES2015 Constants

```
const users = [ "Christian" ];  
  
users.push( "Jim" );  
// > 2  
  
users = [ "Bob" ];  
// > SyntaxError: "users" is read-only
```

Recommendation

1. `const`

2. `let`

3. ~~`var`~~ (ignore)

Arrow Functions

ES5 Functions

```
var numbers = [ 1, 2, 3, 4, 5, 6, 7, 8, 9 ];  
  
var odd = numbers.filter( function(n) {  
    return n % 2 !== 0;  
} );  
  
console.log( odd );  
// > [ 1, 3, 5, 7, 9 ]
```

ES2015 Arrow Functions

```
numbers.filter( n => {  
  return n % 2 !== 0;  
} );  
// > [ 1, 3, 5, 7, 9 ]
```

```
numbers.filter( n => n % 2 !== 0 );  
// > [ 1, 3, 5, 7, 9 ]
```

```
numbers.filter( n => n % 2 );  
// > [ 1, 3, 5, 7, 9 ]
```

ES5 Callbacks

```
var ClickCounter = function() {  
    this.count = 0;  
  
    var _this = this;    // save 'this'  
    $( "#some-button" ).click( function() {  
        _this.count++;  
    } );  
  
};  
  
var obj = new ClickCounter();
```

ES2015 Callbacks

```
var ClickCounter = function() {  
  this.count = 0;  
  
  $( "#some-button" ).click( () => {  
    this.count++;  
  } );  
  
};  
  
var obj = new ClickCounter();
```


Template Strings

ES5 String Concatenation

```
var name = "Christian";  
var count = 213;  
  
var message = "Hello " + name + ", you have "  
    + count + " unread messages."  
  
console.log( message );
```

ES2015 Template Strings

```
const name = "Christian";  
const count = 213;  
  
const message =  
  `Hello ${name}, you have ${count} messages.`;
```

```
const html =  
  `

# Hello ${name}</h1> <p> You have ${count} unread messages </p>`;


```

ES2015 Template Strings

```
const name = "Christian";
const count = 213;
const total = 500;

const greeting =
  `Hello ${name.toUpperCase()}!`;

const message =
  `Unread ratio: ${100 * count / total}%`;
```

Collection Types

ES2015 Sets

```
const tags = new Set();

tags.add( "java" );
tags.add( "javascript" );
tags.add( "java" );

tags.size === 2;
// > true

tags.has( "java" );
// > true
```

ES2015 Maps

```
const map = new Map();  
  
map.set( "hello", 42 );  
  
map.size === 1;  
// > true  
  
map.get( "hello" );  
// > 42  
  
map.delete( "hello" );  
// > true
```

ES5 Iteration

```
var primes = [ 3, 5, 7, 11, 13 ];  
  
for( var i = 0; i < primes.length; i++ ) {  
    console.log( primes[i] );  
}  
  
// ES5  
primes.forEach( function(n) {  
    console.log( n );  
} );
```


ES2015 for..of

```
// arrays
const primes = [ 3, 5, 7, 11, 13 ];
for( let p of primes ) {
  console.log( p );
}
```

```
// collections
const set = new Set();
set.add( "foo" );
set.add( "bar" );
for( let s of set ) {
  console.log( s );
}
```

Default & Rest Params

Spread Operator

Default Parameter

```
function add( a, b = 10 ) {  
    return a + b;  
}
```

```
console.log( add( 3, 5 ) );  
// > 8
```

```
console.log( add( 3 ) );  
// > 13
```

Rest Parameter

```
function format( message, ...params ) {  
  for( let p of params ) {  
    message = message.replace( /\?/, p );  
  }  
  return message;  
}
```

```
format( "Die Summe von ? und ? ist ?", 3, 7, 10 );  
// > Die Summe von 3 und 7 ist 10
```

Spread Operator

```
console.log( Math.max( 1, 5, 2, 3 ) );  
// > 5
```

```
const numbers = [ 1, 5, 2, 3 ];  
  
console.log( Math.max(...numbers) );  
// > 5
```

Classes

ES5: Constructor Functions

```
var Person = function( name ) {  
    this.name = name;  
}  
  
Person.prototype.greet = function() {  
    return "Hello " + this.name;  
}  
  
var christian = new Person( "Christian" );  
  
christian.greet();    // > Hello Christian
```

ES2015 Classes

```
class Person {  
    constructor( name ) {  
        this.name = name;  
    }  
  
    greet() {  
        return "Hello " + this.name;  
    }  
}  
  
const christian = new Person("Christian");  
christian.greet();    // > Hello Christian
```


ES2015 Inheritance

```
class Developer extends Person {  
  
  constructor( name, languages ) {  
    super( name );  
    this.languages = languages;  
  }  
  
  getLanguages() {  
    return this.languages.join( ", " );  
  }  
}  
  
const christian = new Developer( "Christian",  
  [ "Java", "JavaScript" ] );
```

Modules

Export / Import

```
// math.js
export function max(a, b) {
  return a > b ? a : b;
}

export const PI = 3.14156;
```

```
import { max, PI } from "./math.js";

max(9, 13) === 13; // > true
PI === 3.14156; // > true
```

Export / Import

```
// math.js
export function max(a, b) {
  return a > b ? a : b;
}

export const PI = 3.14156;
```

```
import * as math from "./math.js";

math.max(9, 13) === 13    // > true
math.PI === 3.14156     // > true
```

Default Exports

```
// person.js
export default class Person {

  constructor(name) {
    this.name = name;
  }

}
```

```
import Person from "./person.js";

const christian = new Person("Christian");
```

Generators

Generators

```
function* sequence( max ) {  
  let i = 1;  
  while( i <= max ) {  
    yield i;  
    i++;  
  }  
}
```

```
let gen = sequence( 3 );
```

```
gen.next(); // > { value: 1, done: false }  
gen.next(); // > { value: 2, done: false }  
gen.next(); // > { value: 3, done: false }  
gen.next(); // > { value: undefined, done: true }
```

Generators

```
function* sequence( max ) {  
  let i = 1;  
  while( i <= max ) {  
    yield i;  
    i++;  
  }  
}
```

```
for( let i of sequence(10) ) {  
  console.log(i);  
}
```

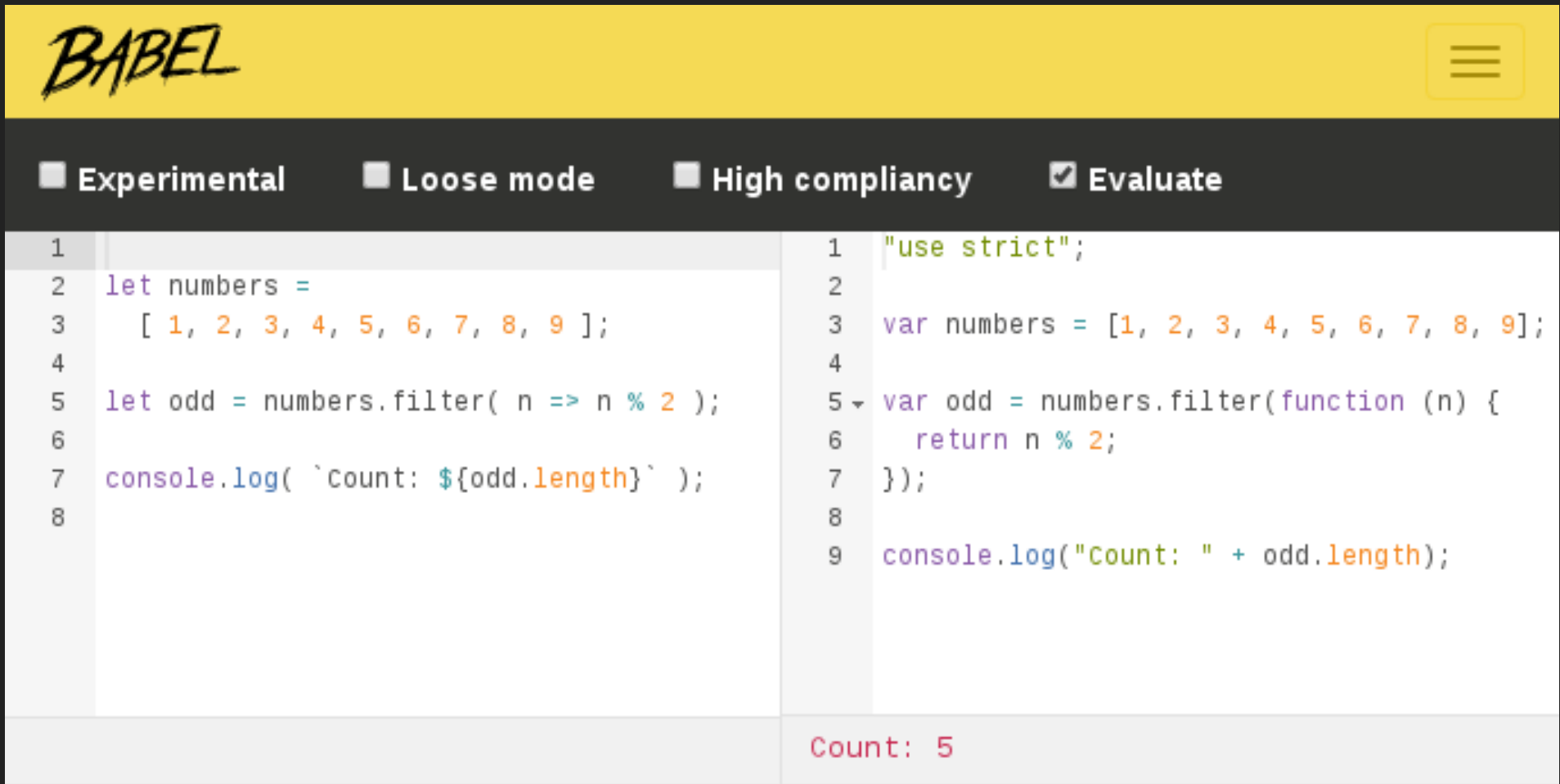

Can I use this stuff?

ES2015 Compatibility

Compilers/polyfills					Desktop browser					
60%	76%	35%	56%	17%	16%	62%	83%	67%	74%	64%
Traceur	Babel + core-js ^[1]	Closure	TypeScript + core-js	es6-shim	IE 11	Edge 12 ^[3]	Edge 13 ^[3]	FF 38 ESR	FF 44	CH 48, OP 35 ^[0]
0/2	1/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
4/7	6/7	4/7	4/7	0/7	0/7	0/7	0/7	3/7	4/7	0/7
4/5	4/5	2/5	3/5	0/5	0/5	5/5	5/5	4/5	5/5	5/5
15/15	13/15	12/15	4/15	0/15	0/15	12/15	15/15	15/15	15/15	15/15
6/6	6/6	4/6	6/6	0/6	0/6	6/6	6/6	6/6	6/6	6/6

<https://kangax.github.io/compat-table/es6/>

Babel REPL



The screenshot shows the Babel REPL interface. At the top left is the 'BABEL' logo in a stylized font. On the right is a hamburger menu icon. Below the header are four toggle buttons: 'Experimental' (unchecked), 'Loose mode' (unchecked), 'High compliancy' (unchecked), and 'Evaluate' (checked). The main area is split into two columns. The left column contains the input code, and the right column contains the output code. At the bottom right, the output of the evaluation is shown.

```
1  
2 let numbers =  
3   [ 1, 2, 3, 4, 5, 6, 7, 8, 9 ];  
4  
5 let odd = numbers.filter( n => n % 2 );  
6  
7 console.log( `Count: ${odd.length}` );  
8
```

```
1 "use strict";  
2  
3 var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9];  
4  
5 var odd = numbers.filter(function (n) {  
6   return n % 2;  
7 });  
8  
9 console.log("Count: " + odd.length);
```

Count: 5

<https://babeljs.io/repl/>

Java Integration

<https://github.com/chkal/frontend-boilerplate>

- Apache Maven
- node.js / npm
- Webpack / Babel / TypeScript
- Karma / Jasmine

Danke!

Fragen?

<http://bit.ly/javaland16-es2015>

<https://github.com/chkal/frontend-boilerplate>

Christian Kaltepoth / @chkal