JavaScript

im Jahr 2017

Christian Kaltepoth / @chkal

Slides: http://bit.ly/javaland-js2017

Christian Kaltepoth

Senior Developer @ ingenit

christian@kaltepoth.de / @chkal

http://blog.kaltepoth.de

JavaScript

aka

ECMAScript

History

- ECMAScript 1 (1997): first release
- ECMAScript 2 (1998): minor alignment
- ECMAScript 3 (1999): regex, exceptions, ...
- ECMAScript 4: killed in 2007
- ECMAScript 5 (2009): strict mode, JSON, ...
- ECMAScript 2015: major update!
- ECMAScript 2016: exp. operator, ...
- ECMAScript 2017: async functions, ...

Block Scope

ES5 Scoping

```
function someFunction() {

  for( var i = 0; i < 4; i++ ) {
    var j = i * i;
  }

  console.log( j );
  // > 9

}
```

Immediately-Invoked Function Expression (IIFE)

```
(function() {

  var secret = 42;

})();

console.log( secret );
// > ReferenceError: secret is not defined
```

ES2015 Block Scope

```
function someFunction() {

  for( let 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 ];

numbers.filter( function( n ) {
  return n % 2 !== 0;
} );
// > [ 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 ]
```

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

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 Parameter

```
function formatMoney( value, currency = "$" ) {
  return value.toFixed( 2 ) + currency;
}

formatMoney( 42.99, "€" );
// > 42.99€

formatMoney( 42.99 );
// > 42.99$
```

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;
```

```
// foobar.js
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;
```

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

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

Promises

Callback Hell

```
asyncFunc1( function () {
  asyncFunc2( function () {
    asyncFunc3( function () {
      asyncFunc4( function () {

        // OMG!

      } );
    } );
  } );
} );
```

Promise

```
const promise = asyncFunc();

promise.then( result => {
  // success (resolved)
} );

promise.catch( error => {
  // failure (rejected)
} );
```

Chaining Promises

```
asyncFunc1()                          // Step #1
  .then( result1 => {
    return asyncFunc2();              // Step #2
  } )
  .then( result2 => {
    return asyncFunc3();              // Step #3
  } )
  .then( result3 => {
    // handle final result
  } )
  .catch( error => {
    // handle all errors
  } );
```

Creating Promises

```
function asyncFunc() {









}
```

Creating Promises

```
function asyncFunc() {

  return new Promise( ( resolve, reject ) => {









  } );
}
```

Creating Promises

```
function asyncFunc() {
  return new Promise( ( resolve, reject ) => {
    setTimeout( () => {


    }, 5000);
  } );
}
```

Creating Promises

```
function asyncFunc() {
  return new Promise( ( resolve, reject ) => {
    setTimeout( () => {
      resolve( "Promise resolved!" );
    }, 5000);
  } );
}
```

And what about ES2016 / ES2017?

New in ES2016

```
Math.pow( 3, 2 );          // ES2015
// > 9

3 ** 2                     // ES2016
// > 9
```

```
const numbers = [ 1, 2, 4, 8 ];

numbers.includes( 2 );       // > true

numbers.includes( 3 );       // > false
```

TC39 Process

- Frequent releases (yearly)
- Feature stages:
  - Stage 0: Strawman
  - Stage 1: Proposal
  - Stage 2: Draft
  - Stage 3: Candidate
  - Stage 4: Finished

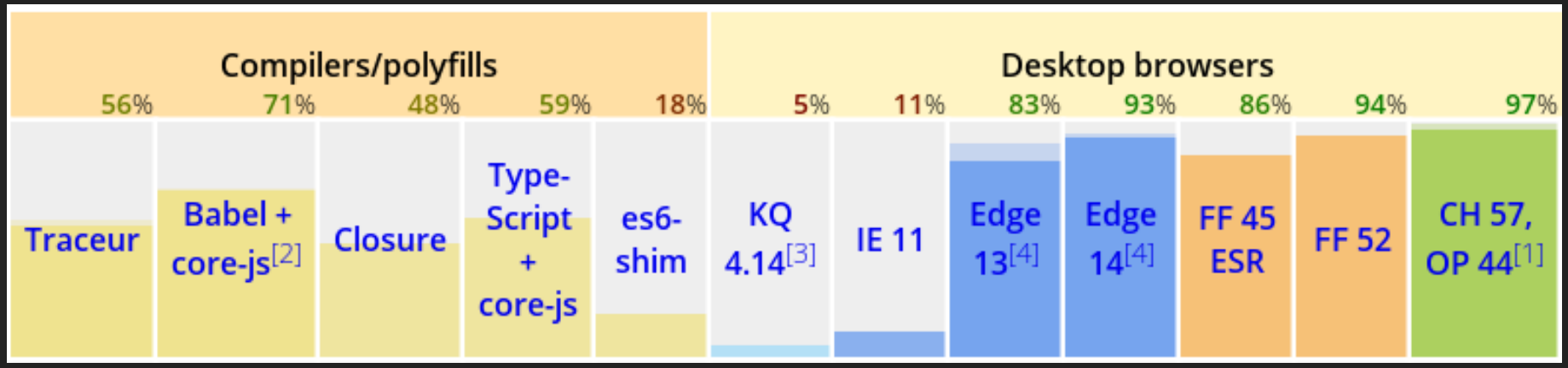
ES2017: Async Functions

```
function asyncFunc() {
  return new Promise( ( resolve, reject ) => {
    setTimeout( () => {
      resolve( "Foobar" );
    }, 2000);
  } );
}
```

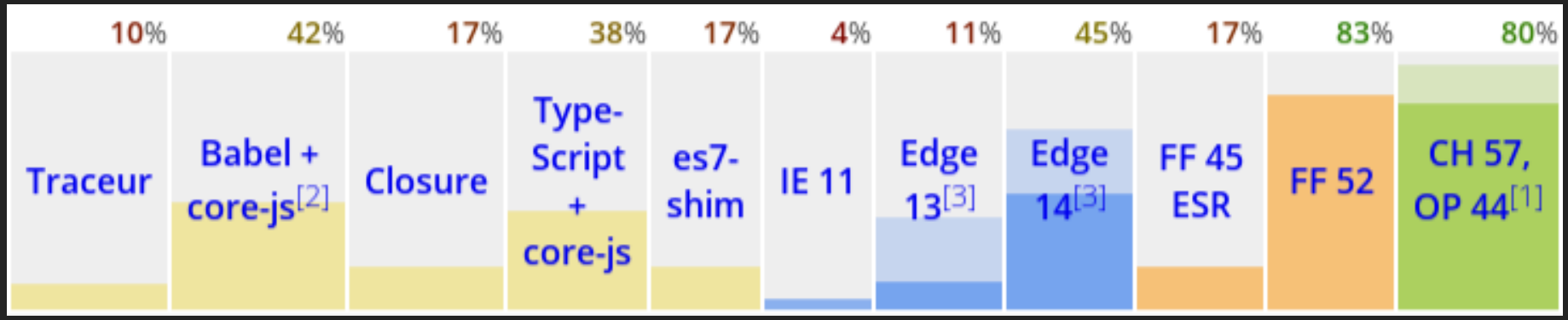
```
async function someFunction() {
  let result = await asyncFunc();
  console.log( result );    // > "Foobar"
}
```

Can I use this stuff?

ES2015 Compatibility

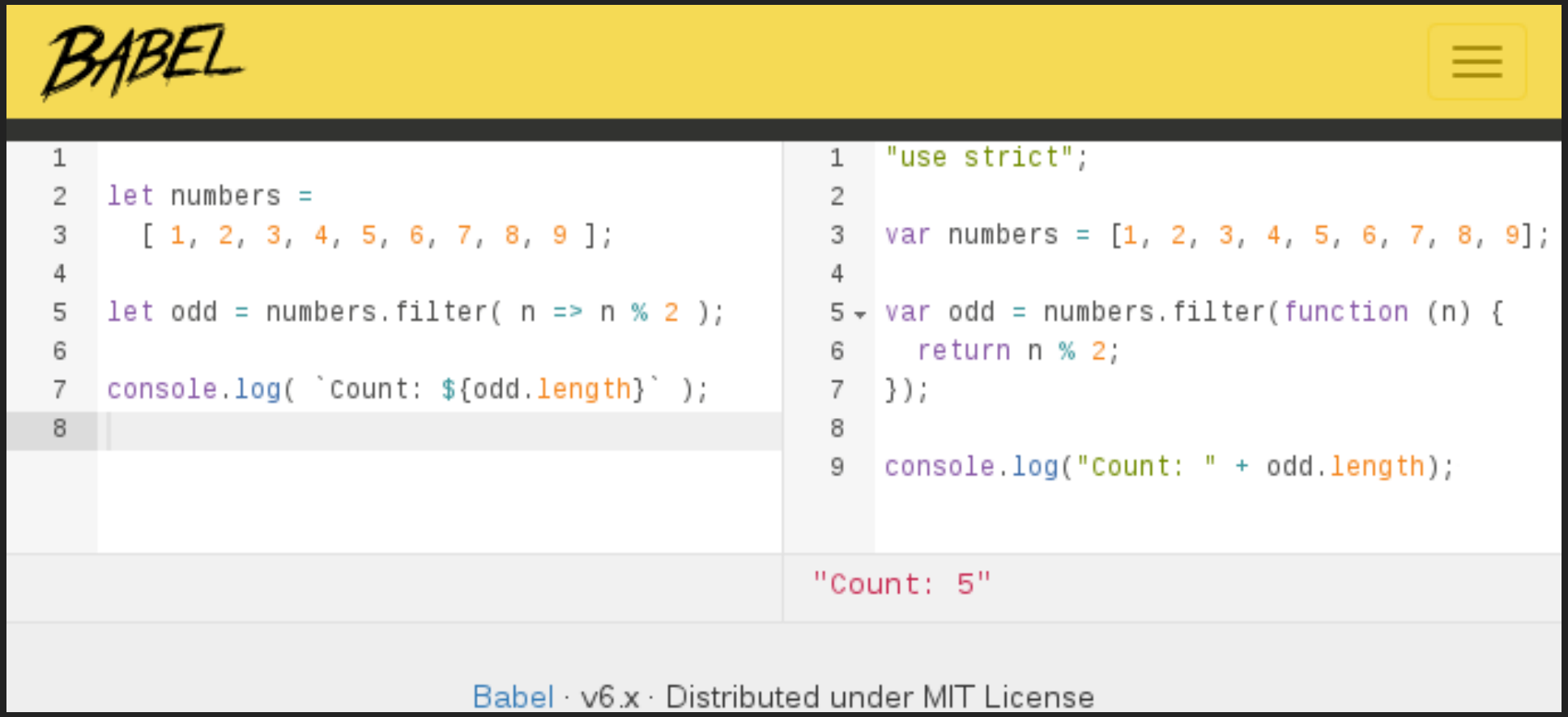


ES2016 + ES2017 Compatibility



Source: https://kangax.github.io/compat-table/

Babel REPL



https://babeljs.io/repl/

Java Boilerplate

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

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

Thanks!

Questions?

http://bit.ly/javaland-js2017

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

Christian Kaltepoth / @chkal