

JavaScript

im Jahr 2016

Christian Kaltepoth / @chkal

Slides: <http://bit.ly/wjax16-js2016>

Christian Kaltepoth

Senior Developer @ ingenit

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

<http://blog.kaltepoth.de>

JavaScript

aka

ECMAScript

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 (very small update)
- ECMAScript 8: 2017 (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  
}
```


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)



Source: <https://twitter.com/andreysitnik/status/792697579712675840>

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 ]
```


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

Default Parameter

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

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

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

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
```

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" );
```

Promises

Callback Hell

```
asyncFunc1( function () {  
    asyncFunc2( function () {  
        asyncFunc3( function () {  
            asyncFunc4( function () {  
  
                // :-(  
  
            } );  
        } );  
    } );  
} );
```


Promise

```
const promise = asyncFunc();

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

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

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
  } );
```

Example: HTML5 Geolocation API

```
navigator.geolocation.getCurrentPosition(  
    function( position ) {  
        // handle success  
    },  
    function( error ) {  
        // handle error  
    }  
);
```

Example: HTML5 Geolocation API

```
function requestPosition() {
```

```
}
```

Example: HTML5 Geolocation API

[illegible]

Example: HTML5 Geolocation API

Example: HTML5 Geolocation API

```
function requestPosition() {  
    return new Promise( ( resolve, reject ) => {  
        navigator.geolocation.getCurrentPosition(  
            position => {  
                },  
            error => {  
                }  
        );  
    } );  
}
```

Example: HTML5 Geolocation API

```
function requestPosition() {  
    return new Promise( ( resolve, reject ) => {  
        navigator.geolocation.getCurrentPosition(  
            position => {  
                resolve( position.coords );  
            },  
            error => {  
                reject( error );  
            }  
        );  
    } );  
}
```


Example: HTML5 Geolocation API

```
requestPosition().then( coords => {  
    console.log( "Position: " + coords );  
} )  
.catch( error => {  
    console.log( "Failed!" );  
} );
```

**And what about
ES2016?**

New in ES2016

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

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

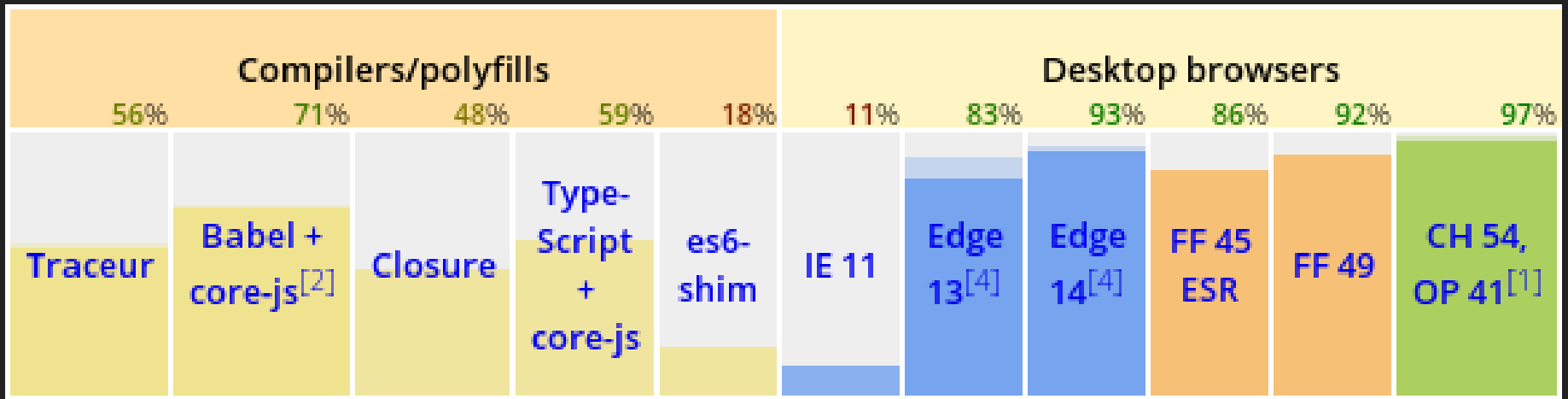
```
const numbers = [ 1, 2, 4, 8 ];  
  
numbers.includes( 2 );       // > true  
  
numbers.includes( 3 );       // > false
```

TC39 Process

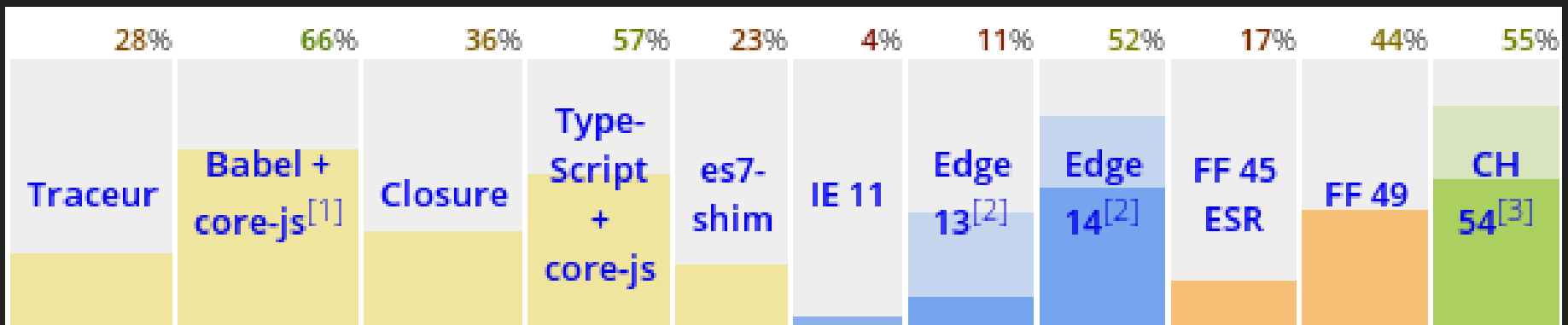
- Frequent releases (yearly)
- Feature stages:
 - Stage 0: Strawman
 - Stage 1: Proposal
 - Stage 2: Draft
 - Stage 3: Candidate
 - Stage 4: Finished

Can I use this stuff?

ES2015 Compatibility



ES2016 Compatibility



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

Babel REPL

BABEL



```
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"

Babel · v6.x · Distributed under MIT License

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

Java Integration

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

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

Thanks!

Questions?

<http://bit.ly/wjax16-js2016>

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

Christian Kaltepoth / @chkal