MATHEMA

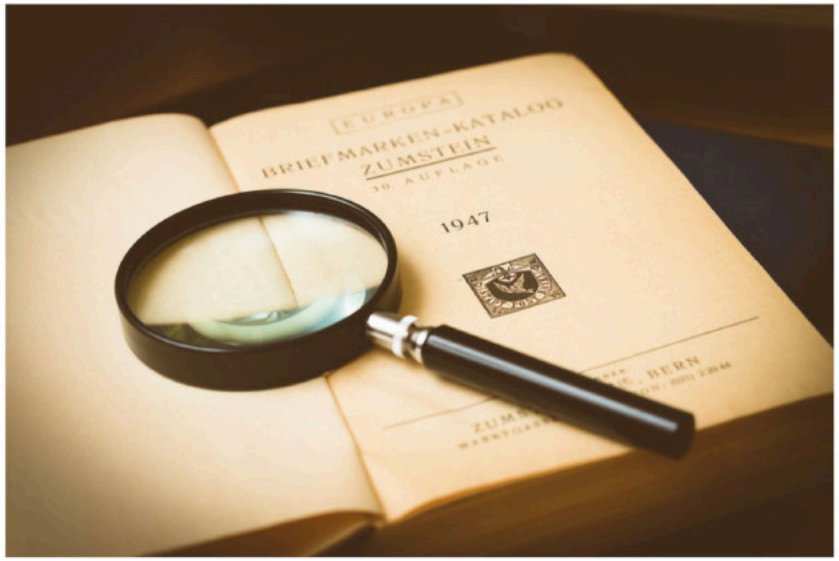
ECMAScript unter der Lupe

Lisa Messerli

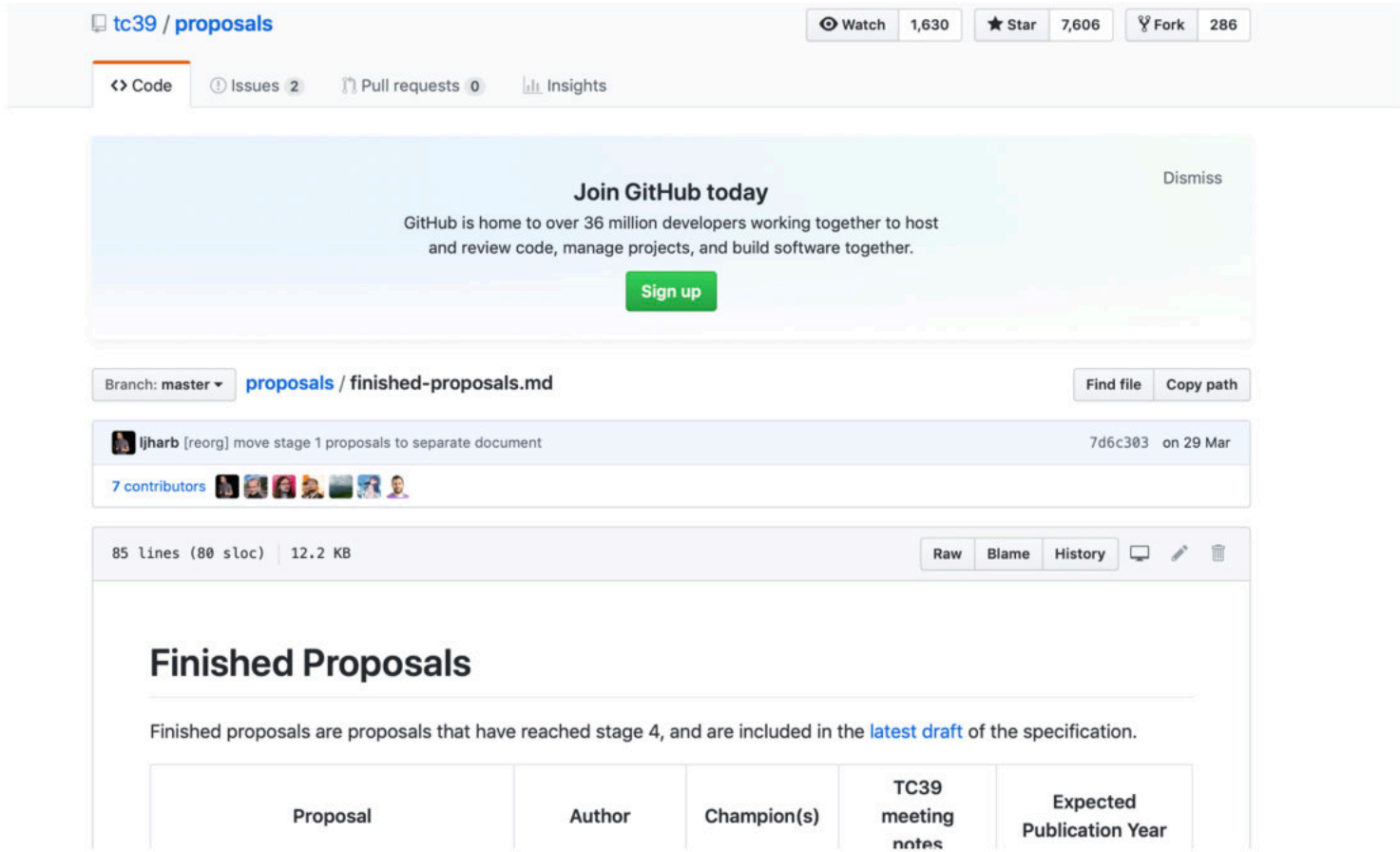
MATHEMA Software GmbH

ECMA

European Computer Manufacturers Association



TC39

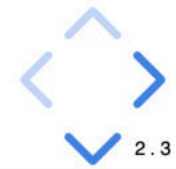


TC39

Finished Proposals

Finished proposals are proposals that have reached stage 4, and are included in the latest draft of the specification.

Proposal	Author	Champion(s)	TC39 meeting notes	Expected Publication Year
`Array.prototype.includes`	Domenic Denicola	Domenic Denicola Rick Waldron	November 2015	2016
Exponentiation operator	Rick Waldron	Rick Waldron	January 2016	2016
`Object.values` / `Object.entries`	Jordan Harband	Jordan Harband	March 2016	2017
String padding	Jordan Harband	Jordan Harband Rick Waldron	May 2016	2017
`Object.getOwnPropertyDescriptors`	Jordan Harband Andrea Giammarchi	Jordan Harband Andrea Giammarchi	May 2016	2017
Trailing commas in function parameter lists and calls	Jeff Morrison	Jeff Morrison	July 2016	2017
Async functions	Brian Terlson	Brian Terlson	July 2016	2017
Shared memory and atomics	Lars T Hansen	Lars T Hansen	January 2017	2017

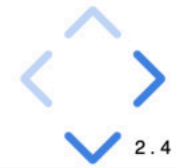


TypeScript

```
const name: string = "Hubertus";
```

```
export interface Adresse {
    ort: string;
    plz: string;
    strasse: string;
    hausnummer: string;
}
const adressbuch: Adresse[] = adressen;
```

```
function formatName(name: string): string => {
    return name.format();
};
```



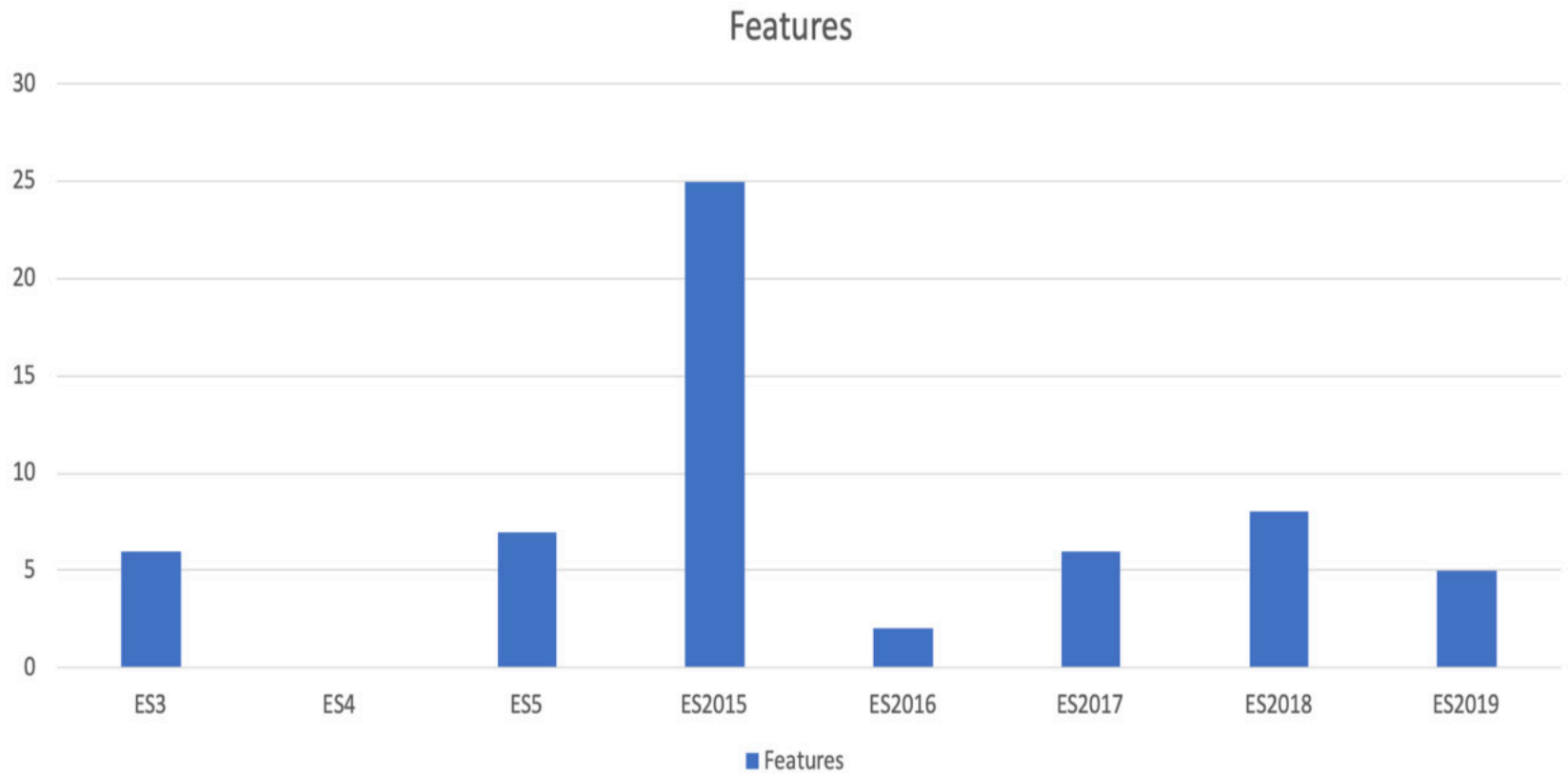
Wer Wie Was Wo?

Warum gibt es eigentlich keine Versionsbücher zu JavaScript?

Warum hört man hauptsächlich von ES6?

Warum hat man den Namen geändert?

Versionsübersicht



Was wir nicht anschauen

Versionen

- **ECMAScript 1** (1997)
- **ECMAScript 2** (1998)
- **ECMAScript 3** (1999)
- **ECMAScript 4** (2008)

Unter der Lupe

```
for(var i = 0; i < 4; i++) {
    console.log(i);
}

console.log(i);
```

Immediately Invoked Function Expression

```
(function() {

    for(var i = 0; i < 4; i++) {
        console.log(i);
    }

})();

console.log(i);
```

Hoisting

```
(function() {
    var i;

    for(i = 0; i < 4; i++) {
        console.log(i);
    }

})();

console.log(i);
```

Unter der Lupe

```
var funktionen = [];


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


funktionen.forEach(
    function(funk) {
        funk();
    }
);
```

Fix Hoist with IIFE

```
var funktionen = [];

for(var i = 0; i < 10; i++) {
    funktionen.push(
        (function(wertVonI) {
            return function() {
                console.log(wertVonI);
            }
        }(i)
    );
}

funktionen.forEach(
    function(funk) {
        funk();
    }
);
```

Unter der Lupe

```
(function() {

    for(i = 0; i < 4; i++) {
        console.log(i);
    }

})();

console.log(i);
```

ECMAScript 5

ECMAScript 5 (2009)

- Strict Mode
- String.trim( )
- JSON.parse( )
- JSON.stringify( )
- Date.now( )
- Object Property Methods
- Property Getter und Setter

Teil 2

Spaß mit Arrays

- Array.reduceRight( )
- Array.every( )
- Array.some( )
- Array.indexOf( )
- Array.lastIndexOf( )
- Array.isArray( )
- Array.forEach( )
- Array.map( )
- Array.filter( )
- Array.reduce( )
- Array.from( )

Aller guten Dinge sind 3

Syntax-Änderungen

- Access on strings
- Trailing commas
- Multiline string literals
- Reserved words as property names

"use strict"

```
"use strict";

var i = 42;

(function() {

    for(var i = 0; i < 4; i++) {
        console.log(i);
    }

})();

console.log(i);

// Ausgabe: 0, 1, 2, 3, 42
```

"use strict"

```
"use strict";

var i = 42;

(function() {

    for(i = 0; i < 4; i++) {
        console.log(i);
    }

})();

console.log(i);

// Ausgabe: 0, 1, 2, 3, 5
```

"use strict"-Modus

Fehlerhafter Code

```
"use strict";

name = 'Hubertus';
```

Löschen ist nicht mehr erlaubt

```
"use strict";

var name: string = 'Hubertus';
delete name;
```


Object Property Methods

Object.defineProperty(objekt, eigenschaft, schlagwort)

Ändert oder ergänzt eine Objekt-Eigenschaft

Object.defineProperty(objekt, eigenschaften)

Ändert oder ergänzt mehrere Objekt-Eigenschaften

Object.getOwnPropertyDescriptor(objekt, eigenschaft)

Zugriff auf die Eigenschaft

Object.getOwnPropertyNames(objekt)

Alle Eigenschaften als Array

Object.keys(objekt)

Alle aufzählbaren Eigenschaften als Array

Unter der Lupe

Object.getPrototypeOf(objekt)

Zugriff auf den Prototyp

Object.preventExtension(objekt)

Verhindert Hinzufügen von Eigenschaften

Object.isExtensible(objekt)

Ist true, wenn Objekt-Eigenschaften hinzugefügt werden können

Object.seal(objekt)

Verhindert Änderungen an den Eigenschaften

Object.isSealed(objekt)

Ist true, wenn Objekt-Eigenschaften nicht geändert werden können

Object.freeze(objekt)

Verhindert alle Änderungen

Object.isFrozen(objekt)

Ist true, wenn keine Änderungen vorgenommen werden können

Getter

```
const telefonnummer = {
    vorwahl: '0911',
    rufnummer: '234567',

    get nummer(): string {
        return `${this.vorwahl}/${this.rufnummer}`;
    }
};


console.log(telefonnummer.nummer);

// Ausgabe: 0911/234567
```

Setter

```
const telefonnummer = {
    vorwahl: '0911',
    rufnummer: '234567',

    set nummer(neueNummer: string) {
        this.rufnummer = neueNummer;
    }
};


telefonnummer.nummer = '9999';


console.log(telefonnummer.rufnummer);

// Ausgabe: 9999
```

Array Additions

- isArray( )
- forEach( )
- map( )
- filter( )
- from( )
- reduce( )
- reduceRight( )
- every( )
- some( )
- indexOf( )
- lastIndexOf( )

Array Additions

Array.isArray( )

```
const telefonbuch: string[] = ['0911/123456',
                               '0911/9999',
                               '0911/98765'];


Array.isArray(telefonbuch);

// Ergebnis: true
```

Array Additions

Array.forEach( )

```
const telefonbuch: string[] = ['0911/123456',
                               '0911/9999',
                               '0911/98765'];


telefonbuch.forEach(nummer => console.log(nummer));

// Ausgabe: "0911/123456"
// Ausgabe: "0911/9999"
// Ausgabe: "0911/98765"
```

Array Additions

Array.map( )

```
const fibonacci: number[] = [1,1,2,3,5,8,13];

const malZwei = (zahl: number) => {
    return zahl * 2;
}


const kaputteFibonacci: number[] = fibonacci.map(malZwei);

console.log(kaputteFibonacci);

// Ausgabe: [ 2, 2, 4, 6, 10, 16, 26 ]
```

Array Additions

Array.filter( )

```
const fib: number[] = [1,1,2,3,5,8,13];

const highFive = (zahl: number) => {
    return zahl > 5;
}


const kaputteFibonacci: number[] = fib.filter(highFive);

console.log(kaputteFibonacci);

// Ausgabe: [ 8, 13 ]
```

Unter der Lupe

Bisher:

```
Array.prototype.slice.call(arrayLike);
```

Array Additions

Array.from( )

Bisher:

```
function formatiereZuArray(arrayLike) {
    return Array.prototype.slice.call(arrayLike);
}

const neuesArray = formatiereZuArray(liste);
```

Neu:

```
const neuesArray = Array.from(liste);
```

Array Additions

Array.reduce( )

```
const artikelPreise: number[] = [1, 19, 8, 3, 11];

const kosten: number = artikelPreise.reduce(
    berechneGesamtpreis);

function berechneGesamtpreis(vorherigerWert, naechsterWert) {
    return vorherigerWert + naechsterWert;
}


console.log(kosten);

// Ausgabe: 42
```

Array Additions

Array.reduceRight( )

```
const fibGroups = [[1,1],[2,3],[5,8]];

const umordnen = (neueSortierung, naechstesPaar) =>
    neueSortierung.concat(naechstesPaar);


const umsortiert = fibGroups.reduceRight(umordnen);


console.log(umsortiert);

// Ausgabe: [ 5, 8, 2, 3, 1, 1 ]
```


Unter der Lupe

```
const schueler: number[] = [20, 21, 22, 21, 15, 28, 23];

const nichtVolljaehrig = (pruefWert: number) => {
    return pruefWert < 18;
};


const ergebnis: boolean =
        schueler.some(nichtVolljaehrig);
```

Array Additions

Array.some( )

```
const schueler: number[] = [20, 21, 22, 21, 15, 28, 23];

const nichtVolljaehrig = (pruefWert: number) => {
    return pruefWert < 18;
};


const mindestensEinerUnter18: boolean =
        schueler.some(nichtVolljaehrig);


console.log(mindestensEinerUnter18);

// Ausgabe: true
```

Array Additions

Array.every( )

```
const schueler: number[] = [10, 11, 12, 11, 15, 8, 13];

const nichtVolljaehrig = (pruefWert: number) => {
    return pruefWert < 18;
};


const keinerIstVolljaehrig: boolean =
        schueler.every(nichtVolljaehrig);


console.log(keinerIstVolljaehrig);

// Ausgabe: true
```

Array Additions

Array.indexOf( )

```
const tiere: string[] = ['Hund', 'Katze', 'Maus',
                         'Esel', 'Katze'];

console.log(tiere.indexOf('Katze'));
// Ausgabe: 1

console.log(tiere.indexOf('Kuh'));
// Ausgabe: -1
```

Array.lastIndexOf( )

```
const tiere: string[] = ['Hund', 'Katze', 'Maus',
                         'Esel', 'Katze'];

console.log(tiere.lastIndexOf('Katze'));
// Ausgabe: 4
```

String.trim( )

```
const schlechtFormatiert: string =
      "         Mein Text!          ";

console.log(schlechtFormatiert.trim());

// Ausgabe: "Mein Text!"
```

Property Access on Strings

Bisher:

```
const telefonnumer: string = '0911/123456';

telefonnummer.charAt(1);
```

Neu:

```
const telefonnumer: string = '0911/123456';

telefonnummer[1];
```

Multiline String

```
const text: string = "Dieser Text \
geht über zwei Zeilen";
```

Date.now( )

```
const aktuelleZeitInMS = Date.now();

console.log(aktuelleZeitInMS);
// Ausgabe: 1557378008797
```

JSON

JSON.parse( )

```
const jsonString: string =
      '{"land":"Deutschland", "ort": "Nürnberg"}';


const location = JSON.parse(jsonString);


console.log(location);

// Ausgabe: { land: 'Deutschland', ort: 'Nürnberg' }
```

JSON.stringify( )

```
console.log(JSON.stringify(location));

// Ausgabe: {"land":"Deutschland","ort":"Nürnberg"}
```

ECMAScript 2015

ECMAScript 2015

- This scope
- Arrows
- Classes
- Enhanced object literals
- Template strings
- Destructuring
- Default, rest, spread
- Let, const
- Iterators
- Generators
- Unicode

ECMAScript 2015

ECMAScript 2015

- Modules
- Map,set, weakmap, weakest
- Proxies
- Symbols
- Subclassable built-ins
- Promises
- Math additions
- Binary and octal literals
- Reflect Api
- Tail calls
- Array.find( )
- Array.findIndex( )

Arrow Functions

Bisher:

```
const ladeDaten = function ladeDaten() {
    // ...
}
```

Neu:

```
const ladeDaten = () => {
    // ...
}
```

Arrow Functions

Bei nur einem Parameter:

```
const ladeDatenVon = serverName => {
    // ...
}
```

Mehrere Parameter:

```
const ladeDatenVon = (url: string, port: number) => {
    // ...
}
```


Classes

```
class Tier {
    constructor(laut) {
        this.laut = laut;
    }

    gibLaut(): string {
        return 'Das Tier macht ' + laut;
    }
}

class Hund extends Tier {
    gibLaut(): string {
        return super.gibLaut() + ' und ist ein Hund.'
    }
}

const bello: Hund = new Hund('wuff');
bello.gibLaut();

// Ausgabe: Das Tier macht wuff und ist ein Hund.
```

Default Parameter

```
wähleTelefonnummer(vorwahl: String = '0911',
                   rufnummer: number): Telefonnummer {
    return {
         console.log(`${newTelefonnummer.vorwahl}
            /${newTelefonnummer.rufnummer}`);
    }
}
```

Aufruf:

```
wähleTelefonnummer(undefined, 98765);

// Ausgabe: 0911/98765
```

Rest Properties

```
const lesezeichen: string[] = ['Amazon','YouTube',
                    'KaffeeKlatsch','Heise.de'];

[onlineShopping, entertainment, ...onlineArtikel]  =
    lesezeichen;
```

Ausgabe

```
console.log(onlineShopping); // Amazon

console.log(entertainment); // YouTube

console.log(onlineArtikel); //KaffeeKlatsch, Heise.de
```

Spread Properties

```
const login = (nutzername: string,
               passwort: string,
               location: string,
               lastVisitedWebsite: string,
               nameDesErstenHaustiers: string,
               gebOrt: string): Benutzer => {

    // ... send data to NSA and login

    return benutzer;
};

const loginDaten: string[] = ['lisa', 'passwort123',
'Nürnberg','mathema.de','kreditCardInfo', 'DorfX'];
```

Spread Data

```
const eingeloggterNutzer: Benutzer = login(...loginDaten);
```

Modules

Exports:

```
export const dieAntwort: number = 42;
```

```
export function addiere(zahl1: number, zahl2: number): number {
    return zahl1 + zahl2;
}
```

```
export interface Telefonnummer {
    vorwahl: string;
    rufnummer: number;
}
```

Modules

Exports:

```
export class Telefonbuch {
    let buch: Telefonnummer[]= [];

    addEintrag(neueNummer: Telefonnummer) {
        buch.push(neueNummer);
    }
}
```

```
default export const dieAntwort: number = 42;
```

```
const dieAntwort: number = 42;
export {dieAntwort as magicNumber}
```


Unter der Lupe

```
import { Telefonnummer, Telefonbuch } from "./test.js";
```

```
import dieAntwort from "./test.js";
```

```
import dieAntwort, { Telefonbuch } from "./test.js";
```

Modules

Imports:

```
import { Telefonnummer, Telefonbuch } from "./test.js";

import * from "./test.js";
```

```
import dieAntwort from "./test.js";
```

```
import { Telefonnummer as Nummer } from "./test.js";
```

```
import "./test.js";
```

Shorthand Object Literals

Bisher:

```
wähleTelefonnummer(vorwahl: string, rufnummer: number) {

    return {
        vorwahl: vorwahl,
        rufnummer: rufnummer
     }
}
```

Neu:

```
wähleTelefonnummer(vorwahl: string, rufnummer: number) {

    return {
        vorwahl,
        rufnummer
     }
}
```

Template Literals

```
const vorwahl: string = '0911';
const rufnummer: number = 12345;

console.log(`Telefonnummer: ${vorwahl}/${rufnummer}`);
```

Object Literals

Dynamic properties

```
const schluessel = {
    ['verschl'+'_'+'uesselung']: 'passwort1234'
};

console.log(schluessel.verschl_uesselung);
// passwort1234
```

Unter der Lupe

```
var liste: number[] = [1, 2, 3];
var eins = liste[0],
    zwei = liste[1];
var drei = eins;

eins = zwei;
zwei = drei;
```

Destructing assignments

Neu:

```
const zahlen: number[] = [4, 2, 3];

let [vier, , drei] = zahlen;

[drei, vier] = [vier, drei];
```


Let and Const

ECMAScript 2016

ECMAScript 2016

- Array.prototype.includes( )
- Exponentiation operator

Unter der Lupe

```
const unterlagen: string[] = ['Reisepass',
                              'Personalausweis',
                              'Steuerunterlagen',
                              'Gehaltsnachweis'];


if (!unterlagen.indexOf('Kundenkarte') > -1) {
```

Array.prototype.includes()

Vorher:

```
const unterlagen: string[] = ['Reisepass','Personalausweis',
'Steuerunterlagen', 'Gehaltsnachweis'];

if (!unterlagen.indexOf('Kundenkarte') > -1) {
    console.log('Nicht gefunden.');
}
```


Array.prototype.includes()

Neu:

```
const unterlagen: string[] = ['Reisepass', 'Personalausweis',
'Steuerunterlagen', 'Gehaltsnachweis'];

if (!unterlagen.includes('Kundenkarte')) {
    console.log('Nicht gefunden.');
}
```

Exponentiation Operator

Keine Library mehr notwendig

Vorher:

```
console.log( Math.pow(7,3) );
```

Neu:

```
console.log( 7 ** 3 );
```

ECMAScript 2017

ECMAScript 2017

- String padding
- Object.values und Object.entries
- Object.getOwnPropertyDescriptors
- Trailing commas
- Async function
- Shared memories & atomics

String Padding

padStart

```
const telefonnummer: string = '0911/1234567';
const letzteStellen: string = telefonnummer.slice(-2);

const secretNr: string = letzteStellen
                         .padStart(telefonnummer.length,'*');



console.log(secretNr);

// Ausgabe: **********67
```

Object Enhancement

Object.values( )

```
const info = {
    land: 'Deutschland',
    ort: 'Nürnberg'
}

console.log(Object.values(info));

// Ausgabe: ["Deutschland", "Nürnberg"]
```

Object Enhancement

Object.entries()

```
const info = {
    land: 'Deutschland',
    ort: 'Nürnberg'
}

console.log(Object.entries(info)[1]);

// Ausgabe: ["ort", "Nürnberg"]
```

Object.getOwnPropertyDescriptors( )

Bisher:

```
const telefonnummer: string = {
    set nummer(nummer: string) {
        console.log(nummer);
    }
};

const zweiteNummer: string = {};

Object.assign(zweiteNummer, telefonnummer);
```

Problem:

```
telefonnummer.nummer = '0911/2345';
// Ausgabe: 0911/2345

zweiteNummer.nummer = '0911/11111'
```

Object.getOwnPropertyDescriptors( )

Neu:

```
const telefonnummer: string = {
    set nummer(nummer: string) {
        console.log(nummer);
    }
};

const zweiteNummer: string = {};

Object.defineProperties(zweiteNummer,
    Object.getOwnPropertyDescriptors(telefonnummer));
```

Ausgabe:

```
telefonnummer.nummer = '0911/2345';
// Ausgabe: 0911/2345

zweiteNummer.nummer = '0911/11111'
// Ausgabe: 0911/11111
```

Trailing Commas

Letzter Parameter darf ein Komma haben

```
wähleTelefonnummer(vorwahl: string, rufnummer: number) {

    return {
        vorwahl,
        rufnummer, // hier
     }
}
```

Asynchronous Functions

Funktion

```
async ladeDaten(url: string): Promise<string[]> {

    return fetch(url)
        .then(
            // Verarbeitung
        )
        .then( /* Weitere Schritte */ )
        .catch( error => {
            console.log("Fehler beim Laden der Daten.")
        });
}
```

Aufruf

```
const daten: string[] = await ladeDaten('google.de');
```

ECMAScript 2018

ECMAScript 2018

- Asynchronous Iteration
- Rest, spread for objects
- Promise finally
- RegExp
  - Named Capture Groups
  - Unicode Property Escapes
  - Lookbehind Assertions
  - S – Flag
  - Template Literal Revision

Asynchrous Iteration

Aufruf

```
const alleArtikel: Artikel[] = await ladeFachartikel();
```

Iterative for-Schleife

```
async ladeFachartikel(): Promise<Artikel[]> {

    for await (const fachartikel of
               holeAsyncDatenVon(url)) {
        // ...
    }
    return artikel;
}
```

Promise Finally

```
fetch (netzwerkDaten)
    .then (result => {
            //... Formatiere Daten zur Weiterverarbeitung
    })
    .catch (error => {
        console.log('Verbindungsabbruch bei Datenanfrage')
    })
    .finally (
        //... Schließe Netzwerkverbindung
    );
```

Regular Expression

- Named Capturing Groups
- Unicode
- sFlag
- Lookbehind
- Template Literal Revision

Named Capturing Groups

Bisheriger Aufruf:

```
const reg: RegExp = /([\w-]{2,10})\s([\w-]{2,20})/;
const ergebnis = reg.exec('Herr Mustermann');

console.log(ergebnis[0]); // Herr Mustermann
console.log(ergebnis[1]); // Herr
console.log(ergebnis[2]); // Mustermann
```

Named Capturing Groups

Anwendung:

```
const reg: RegExp =
/(?<anrede>[\w-]{2,10})\s(?<nachname>[\w-]{2,20})/;

const ergebnis = reg.exec('Herr Mustermann');
```

Abrufen der Daten:

```
console.log(ergebnis.groups.anrede); // Herr

console.log(ergebnis.groups.nachname); // Mustermann
```

Unter der Lupe

Anwendung:

```
/^\p{ASCII_Hex_Digit}+$/u.test('68 69');
```

Unicode Property Escapes

Stichwort	Ergebnis
ASCII	Alle ASCII-Zeichen “Hallo!”
ASCII_Hex_Digit	Valide Hexadezimalzahlen “0x68 0x69”
Uppercase	Großbuchstaben “A”
Lowercase	Kleinbuchstaben “a”
White_Space	Leerzeichen, Tabs, Neue Zeilen
Alphabetic	Alle Unicode-Zeichen, die als alphabetisch gelten A-Z, a-z
Script=”Greek”	Griechische Buchstaben „αθ“
Emoji	Valide Emojis “☕📖”

Unicode Property Escapes

Anwendung:

```
const hello: string = 'hi';

/^\p{ASCII_Hex_Digit}+$/u.test('68 69'); // true
```

S-Flag

Bei möglichem Zeilenumbruch:

```
/aus.druck/s.test('Aus\ndruck'); // true
```


Template Literal Revision

Bisherige Fehlermeldung:

```
                              Hexadecimal digit expected.
const pfad: string = "c:\user\katzenfotos";
return pfad;
```

Korrekte Ausgabe:

```
String.raw`c:\user\katzenfotos`;
// Ausgabe: c:\user\katzenfotos
```

ECMAScript 2019

ECMAScript 2019

- Array.prototype.{flat,flatMap}
- String.prototype.{trimStart,trimEnd}
- JSON.stringify
- Object.fromEntries
- Function.prototype.toString revision

Unter der Lupe

```
const tiere: string[] = ['Hund',
                         ['Katze',
                          ['Maus','Ratte']]];


console.log(tiere.flat(2));
```

Array.flat()

```
const tiere: string[] = ['Hund', 'Katze', ['Maus',
                                           'Ratte']];

console.log(tiere.flat());

// Ausgabe [ 'Hund', 'Katze', 'Maus', 'Ratte' ]
```

Array.flat(2)

```
Schachtelung: [ [ [ x ] ] ] wird zu [ x ]
```

flat(Infinity)

```
Schachtelung: [ [ [ [ x ] ] ] ] wird zu [ x ]
```

Array.flatMap()

```
const fibonacci = [1, 1, 2, 3, [5, 8, 13]];

console.log(fibonacci.flatMap(zahl => [zahl *2]);

// Aktuell: [ 2, 2, 4, 6, NaN ]

// Bald: [ 2, 2, 4, 6, 10, 16, 26 ]
```


Optional Catch Binding

Bisher:

```
try {
   // probiere
} catch (error) {
    // Fehlerhandling
}
```

Neu:

```
try {
    // probiere
} catch {
    // Fehlerhandling
}
```


Object.fromEntries( )

```
const telefonnummer = {
    vorwahl: '0911',
    rufnummer: '123456'
};


const eintraege = Object.entries(telefonnummer);

// [ [ 'vorwahl', '0911' ], [ 'rufnummer', '123456' ] ]



const neueNummer = Object.fromEntries(eintraege);

// telefonnummer !== neueNummer -> true
```

String.trimStart( )

```
'   MeinText'.trimStart(); // 'MeinText'
'MeinText    '.trimStart(); // 'MeinText    '
```

String.trimEnd( )

```
'   MeinText'.trimEnd(); // '   MeinText'
'MeinText    '.trimEnd(); // 'MeinText'
```


JSON improvements

Fehler wurde behoben, der Zeilenumbrüche und Paragraphen nicht richtig bearbeitet werden konnten.

Bisher problematisch:

```
eval('"\u2028"');
// SyntaxError: Invalid or unexpected token
```

```
JSON.stringify('\uD800');
// "\ufffd"
```

Function.toString( )

Bisher:

```
function /*nützlicher Kommentar*/ tolleFunktion() {}

tolleFunktion.toString();
// function tolleFunktion() {}
```

Neu:

```
function /* nützlicher Kommentar */ tolleFunktion() {}

tolleFunktion.toString();
// function /* nützlicher Kommentar */ tolleFunktion() {}
```

ES.Next (2020)

- String.prototype.matchAll

Active proposals

Proposals follow this process document. This list contains only stage 1 proposals and higher that have not yet been withdrawn/rejected, or become finished.

Stage 3

Proposal	Author	Champion
`Intl.ListFormat`	Zibi Braniecki	Zibi Braniecki
`Intl.Locale`	Zibi Braniecki, Daniel Ehrenberg	Zibi Braniecki, Daniel Ehrenberg
`Intl.RelativeTimeFormat`	Zibi Braniecki, Daniel Ehrenberg	Zibi Braniecki, Daniel Ehrenberg
Intl.NumberFormat Unified API Proposal	Shane Carr	Shane Carr
`DateTimeFormat` `dateStyle` & `timeStyle`	Zibi Braniecki	Zibi Braniecki
`Intl.DateFormat.prototype.formatRange`	Felipe Balbontín	Sathya Gunasekaran

Viel Spaß beim Ausprobieren!