

# What's new in ES2018?



[@bmeurer, @mathias].join(@v8js)

# Synchronous iteration

Recap

```
function sum(a) {  
  let sum = 0;  
  for (const x of a) {  
    sum += x;  
  }  
  return sum;  
}
```

[@bmeurer, @mathias].join(@v8js)

# Synchronous iteration interfaces

```
interface Iterable {  
  [Symbol.iterator]() : Iterator;  
}  
  
interface Iterator {  
  next() : IterResultObject;  
}  
  
interface IterResultObject {  
  value : any;  
  done : boolean;  
}
```

[@bmeurer, @mathias].join(@v8js)

```
function sum(iterable) {  
  let result = 0;  
  const iterator = iterable[Symbol.iterator]();  
  while (true) {  
    const object = iterator.next();  
    if (object.done) break;  
    result += object.value;  
  }  
  return result;  
}
```

[@bmeurer, @mathias].join(@v8js)

```
function sum(iterable) {  
  let result = 0;  
  const iterator = iterable[Symbol.iterator]();  
  while (true) {  
    const object = iterator.next();  
    if (object.done) break;  
    result += object.value;  
  }  
  return result;  
}
```

[@bmeurer, @mathias].join(@v8js)

# Asynchronous iteration

```
async function asyncSum(a) {  
  let sum = 0;  
  for await (const x of a) {  
    sum += x;  
  }  
  return sum;  
}
```

[@bmeurer, @mathias].join(@v8js)

# Asynchronous iteration interfaces

```
interface AsyncIterable {  
  [Symbol.asyncIterator]() : AsyncIterator;  
}  
  
interface AsyncIterator {  
  next() : Promise<IterResultObject>;  
}  
  
interface IterResultObject {  
  value : any;  
  done : boolean;  
}
```

[ @bmeurer , @mathias ] . join(@v8js)

```
async function asyncSum(iterable) {  
  let result = 0;  
  const iterator = iterable[Symbol.asyncIterator]();  
  while (true) {  
    const object = await iterator.next();  
    if (object.done) break;  
    result += object.value;  
  }  
  return result;  
}
```

[@bmeurer, @mathias].join(@v8js)

```
async function asyncSum(iterable) {  
  let result = 0;  
  const iterator = iterable[Symbol.asyncIterator]();  
  while (true) {  
    const object = await iterator.next();  
    if (object.done) break;  
    result += object.value;  
  }  
  return result;  
}
```

[@bmeurer, @mathias].join(@v8js)

```
// All iterables are implicitly async iterable
// via so-called Async-from-Sync Iterator Objects™.
asyncSum([1, 2, 3]).then(console.log);
// → prints 6
```

[@bmeurer, @mathias].join(@v8js)

```
// All iterables are implicitly async iterable
// via so-called Async-from-Sync Iterator Objects™.
asyncSum([1, 2, 3]).then(console.log);
// → prints 6

const set = new Set([1, 2, 3, 4, 5, 6]);
(async () => {
  const result = await asyncSum(set);
  console.log(result);
})();
// → prints 21
```

[@bmeurer, @mathias].join(@v8js)

```
// Node streams are async-iterable starting with Node 10.  
// (WARNING: the feature is considered experimental!)  
const fs = require('fs');  
  
(async () => {  
    const readStream = fs.createReadStream('file');  
    for await (const chunk of readStream) {  
        console.log(chunk);  
    }  
})();
```

[@bmeurer, @mathias].join(@v8js)

```
// For backwards compatibility, use a wrapper like
// github:basicdays/node-stream-to-async-iterator (mths.be/bwu)
require('core-js/es7/symbol');
const fs = require('fs');
const S2A = require('stream-to-async-iterator');

(async () => {
  const readStream = fs.createReadStream('file');
  for await (const chunk of new S2A(readStream)) {
    console.log(chunk);
  }
})();
```

[@bmeurer, @mathias].join(@v8js)

# Asynchronous generators

```
async function* asyncRange(start, end) {  
  for (let i = start; i <= end; ++i) {  
    yield i;  
  }  
}
```

[@bmeurer, @mathias].join(@v8js)

```
async function* asyncRange(start, end) {
  for (let i = start; i <= end; ++i) {
    yield i;
  }
}

asyncSum(asyncRange(1, 10))
  .then(console.log);
// → prints 55
```

[@bmeurer, @mathias].join(@v8js)

```
async function* asyncRandomNumbers() {  
    // This is a web service that returns a random number.  
    const url = 'https://www.random.org/decimal-fractions/' +  
        '?num=1&dec=10&col=1&format=plain&rnd=new';  
    while (true) {  
        const response = await fetch(url);  
        const text = await response.text();  
        yield Number(text);  
    }  
}
```

[@bmeurer, @mathias].join(@v8js)

```
// Generate a bunch of random numbers.  
(async () => {  
  for await (const number of asyncRandomNumbers()) {  
    console.log(number);  
    if (number > 0.95) break;  
  }  
})();
```

[@bmeurer, @mathias].join(@v8js)

# Iterating events

A use case study

```
const AsyncIteration = {  
  // Turns `type` events produced by a DOM `element`  
  // into an async iterable stream of events. Inspired  
  // by RxJS's `Observable.fromEvent`.  
  fromEvent: async function*( element, type ) {  
    let resolve;  
    const promises = [ new Promise(r => resolve = r) ];  
    element.addEventListener(type, event => {  
      resolve(event);  
      promises.push(new Promise(r => resolve = r));  
    });  
    while (true) yield await promises.shift();  
  }  
};
```

[@bmeurer, @mathias].join(@v8js)

```
<button>Clickety-click!</button>

<script type="module">
  import { AsyncIteration } from './async-iteration.mjs';
  (async () => {
    const button = document.querySelector('button');
    const iterator = AsyncIteration.fromEvent(button, 'click');
    for await (const event of iterator) {
      console.log(event);
    }
  })();
</script>
```

[@bmeurer, @mathias].join(@v8js)

# Taking it further...

Watch your back, Observables 😎

```
const AsyncIteration = {
  // Generates values that pass the provided condition.
  // Inspired by RxJS's `filter`.
  filter: async function*( iterable, select, thisArg ) {
    for await (const value of iterable) {
      if (select.call(thisArg, value)) yield value;
    }
  }
};
```

[@bmeurer, @mathias].join(@v8js)

```
const AsyncIteration = {
  // Merges N async iterables into a single async iterable.
  // Inspired by RxJS's `merge`.
  merge: async function* (...args) {
    let resolve;
    const promises = [new Promise(r => resolve = r)];
    async function pump(iterator) {
      while (true) {
        const result = await iterator.next();
        resolve(result);
        promises.push(new Promise(r => resolve = r));
      }
    }
    args.map(iterable => iterable[Symbol.asyncIterator]()).forEach(pump);
    while (true) {
      const result = await promises.shift();
      if (result.done) break;
      yield result.value;
    }
  }
};

[@bmeurer, @mathias].join(@v8js)
```

```
<button id="button-1">Hello</button>
<button id="button-2">World</button>
<script type="module">
  import { AsyncIteration } from './async-iteration.mjs';
  (async () => {
    const button1 = document.querySelector('#button-1');
    const button2 = document.querySelector('#button-2');
    const a = AsyncIteration.fromEvent(button1, 'click');
    const b = AsyncIteration.fromEvent(button2, 'click');
    for await (const event of AsyncIteration.merge(a, b)) {
      console.log(event.target);
    }
  })();
</script>
```

[@bmeurer, @mathias].join(@v8js)

Live demo



[@bmeurer, @mathias].join(@v8js)

# RegExp features

Improving readability and maintainability

# RegExp dotAll mode

a.k.a. the s flag

```
const input = `

Lorem ipsum dolor sit amet, consectetur adipiscing hello
world elit. Nam sit amet elit id risus aliquam porta.

`;
```

[@bmeurer, @mathias].join(@v8js)

```
const input = `

Lorem ipsum dolor sit amet, consectetur adipiscing hello

world elit. Nam sit amet elit id risus aliquam porta.

`;
```

[@bmeurer, @mathias].join(@v8js)

```
const input = `

Lorem ipsum dolor sit amet, consectetur adipiscing hello

world elit. Nam sit amet elit id risus aliquam porta.

`;

/hello.world/u.test(input);

// → ?
```

[@bmeurer, @mathias].join(@v8js)

```
const input = `

Lorem ipsum dolor sit amet, consectetur adipiscing hello

world elit. Nam sit amet elit id risus aliquam porta.

`;

/hello.world/u.test(input);

// → false 🤔
```

[@bmeurer, @mathias].join(@v8js)

```
const input = `  
Lorem ipsum dolor sit amet, consectetur adipiscing hello  
world elit. Nam sit amet elit id risus aliquam porta.  
`;  
/hello.world/u.test(input);  
// → false 🤦
```



[@bmeurer, @mathias].join(@v8js)

```
const input = `

Lorem ipsum dolor sit amet, consectetur adipiscing hello

world elit. Nam sit amet elit id risus aliquam porta.

`;

/hello[\s\S]world/u.test(input);

// → true 😞
```

[@bmeurer, @mathias].join(@v8js)

```
const input = `

Lorem ipsum dolor sit amet, consectetur adipiscing hello

world elit. Nam sit amet elit id risus aliquam porta.

`;

/hello[\s\S]world/u.test(input);

// → true 😞

/hello[^]world/u.test(input);

// → true 🎉♂
```

[@bmeurer, @mathias].join(@v8js)

```
const input = `

Lorem ipsum dolor sit amet, consectetur adipiscing hello
world elit. Nam sit amet elit id risus aliquam porta.

`;  
// →  world  
// →  hello  
// → true  ♂
```

[@bmeurer, @mathias].join(@v8js)

```
const input = `

Lorem ipsum dolor sit amet, consectetur adipiscing hello

world elit. Nam sit amet elit id risus aliquam porta.

`;

/hello.world/su.test(input);

// → true 🎉
```



[@bmeurer, @mathias].join(@v8js)

```
const input = `

Lorem ipsum dolor sit amet, consectetur adipiscing hello

world elit. Nam sit amet elit id risus aliquam porta.

`;

/hello.world/us.test(input);

// → true 🇺🇸
```

[@bmeurer, @mathias].join(@v8js)

# RegExp lookbehind assertions



```
// Positive lookahead:  
const pattern = /\d+(?= dollars)/u;  
const result = pattern.exec('42 dollars');  
// → result[0] === '42'
```

[@bmeurer, @mathias].join(@v8js)

```
// Positive lookahead:  
  
const pattern = /\d+(?= dollars)/u;  
  
const result = pattern.exec('42 dollars');  
// → result[0] === '42'
```

```
// Negative lookahead:  
  
const pattern = /\d+(?! dollars)/u;  
  
const result = pattern.exec('42 pesos');  
// → result[0] === '42'
```

[@bmeurer, @mathias].join(@v8js)

```
// Positive lookbehind:  
const pattern = /( ?<=\$)\d+/u;  
const result = pattern.exec(' $42' );  
// → result[0] === '42'
```

[@bmeurer, @mathias].join(@v8js)

```
// Positive lookbehind:  
  
const pattern = /(?<=\$)\d+/u;  
const result = pattern.exec('$42');  
// → result[0] === '42'
```

```
// Negative lookbehind:  
  
const pattern = /(?<!\$)\d+/u;  
const result = pattern.exec('€42');  
// → result[0] === '42'
```

[@bmeurer, @mathias].join(@v8js)

```
// Positive lookbehind:  
const pattern = /(?<=\$)\d+/u;  
'Price: $42'.replace(pattern, '9001');  
// → 'Price: $9001'
```

[@bmeurer, @mathias].join(@v8js)

```
// Positive lookbehind:  
const pattern = /(?<=\$)\d+/u;  
'Price: $42'.replace(pattern, '9001');  
// → 'Price: $9001'
```

```
// Negative lookbehind:  
const pattern = /(?<! \$\d*)\d+/gu;  
'My 17 children have $42 and €3 each.'.replace(pattern, '9001');  
// → 'My 9001 children have $42 and €9001 each.'
```

[@bmeurer, @mathias].join(@v8js)

# RegExp named capture groups

maintainability++

```
const pattern = /(\d{4})-(\d{2})-(\d{2})/u;  
const result = pattern.exec('2018-03-06');  
// → result[0] === '2018-03-06'
```

[@bmeurer, @mathias].join(@v8js)

```
const pattern = /(\d{4})-(\d{2})-(\d{2})/u;  
const result = pattern.exec('2018-03-06');  
// → result[0] === '2018-03-06'
```

[@bmeurer, @mathias].join(@v8js)

```
const pattern = /(\d{4})-(\d{2})-(\d{2})/u;  
const result = pattern.exec('2018-03-06');  
// → result[0] === '2018-03-06'  
// → result[1] === '2018'
```

[@bmeurer, @mathias].join(@v8js)

```
const pattern = /(\d{4})-(\d{2})-(\d{2})/u;  
const result = pattern.exec('2018-03-06');  
// → result[0] === '2018-03-06'  
// → result[1] === '2018'  
// → result[2] === '03'
```

[@bmeurer, @mathias].join(@v8js)

```
const pattern = /(\d{4})-(\d{2})-(\d{2})/u;  
const result = pattern.exec('2018-03-06');  
// → result[0] === '2018-03-06'  
// → result[1] === '2018'  
// → result[2] === '03'  
// → result[3] === '06'
```

[@bmeurer, @mathias].join(@v8js)

```
const pattern = /(?<year>\d{4})-(?<month>\d{2})-(?<day>\d{2})/u;  
const result = pattern.exec('2018-03-06');
```

[@bmeurer, @mathias].join(@v8js)

```
const pattern = /(?<year>\d{4})-(?<month>\d{2})-(?<day>\d{2})/u;  
const result = pattern.exec('2018-03-06');  
// → result.groups.year === '2018'
```

[@bmeurer, @mathias].join(@v8js)

```
const pattern = /(?<year>\d{4})-(?<month>\d{2})-(?<day>\d{2})/u;  
const result = pattern.exec('2018-03-06');  
// → result.groups.year === '2018'  
// → result.groups.month === '03'
```

[@bmeurer, @mathias].join(@v8js)

```
const pattern = /(?<year>\d{4})-(?<month>\d{2})-(?<day>\d{2})/u;  
const result = pattern.exec('2018-03-06');  
// → result.groups.year === '2018'  
// → result.groups.month === '03'  
// → result.groups.day === '06'
```

[@bmeurer, @mathias].join(@v8js)

RegExp Unicode property escapes  
Iñternâtiônàlizætiøn

```
const regexGreek = /\p{Script_Extensions=Greek}/u;  
regexGreek.test('π');  
// → true
```

[@bmeurer, @mathias].join(@v8js)

```
const regexGreek = /\p{Script_Extensions=Greek}/u;  
regexGreek.test('π');  
// → true  
  
const regexAscii = /\p{ASCII}/u;  
regexAscii.test('_');  
// → true
```

[@bmeurer, @mathias].join(@v8js)

```
const regexGreek = /\p{Script_Extensions=Greek}/u;
regexGreek.test('π');
// → true

const regexAscii = /\p{ASCII}/u;
regexAscii.test('_');
// → true

const regexMath = /\p{Math}/u;
regexMath.test('≠');
// → true
```

[@bmeurer, @mathias].join(@v8js)

```
const pattern = /[ \p{Letter}\p{White_Space}]+/u;  
pattern.test('Γειά σου κόσμε');  
// → true
```

[@bmeurer, @mathias].join(@v8js)

```
<style>
  :valid { background: lightgreen; }
  :invalid { background: pink; }
</style>
<input pattern="[\p{Letter}\p{White_Space}]+"
       value="Γειά σου κόσμε">
<!-- Demo: https://mths.be/bwo -->
```

[@bmeurer, @mathias].join(@v8js)

# Object rest and spread properties

```
// Rest elements for array destructuring assignment:  
const primes = [2, 3, 5, 7, 11];  
const [first, second, ...rest] = primes;  
console.log(first); // 2  
console.log(second); // 3  
console.log(rest); // [5, 7, 11]
```

[@bmeurer, @mathias].join(@v8js)

```
// Rest elements for array destructuring assignment:  
const primes = [2, 3, 5, 7, 11];  
const [first, second, ...rest] = primes;  
console.log(first); // 2  
console.log(second); // 3  
console.log(rest); // [5, 7, 11]
```

```
// Spread elements for array literals:  
const primesCopy = [first, second, ...rest];  
console.log(primesCopy); // [2, 3, 5, 7, 11]
```

[@bmeurer, @mathias].join(@v8js)

```
// Rest properties for object destructuring assignment:  
const person = {  
  firstName: 'Sebastian',  
  lastName: 'Markbåge',  
  country: 'USA',  
  state: 'CA',  
};  
const { firstName, lastName, ...rest } = person;  
console.log(firstName); // Sebastian  
console.log(lastName); // Markbåge  
console.log(rest); // { country: 'USA', state: 'CA' }
```

[@bmeurer, @mathias].join(@v8js)

```
// Spread properties for object literals:  
const personCopy = { firstName, lastName, ...rest };  
console.log(personCopy);  
// { firstName: 'Sebastian',  
//   lastName: 'Markbåge',  
//   country: 'USA',  
//   state: 'CA' }
```

[@bmeurer, @mathias].join(@v8js)

```
// Shallow-clone an object:  
const data = { x: 42, y: 27, label: 'Treasure' };  
// The old way:  
const clone1 = Object.assign({}, data);
```

[@bmeurer, @mathias].join(@v8js)

```
// Shallow-clone an object:  
const data = { x: 42, y: 27, label: 'Treasure' };  
// The old way:  
const clone1 = Object.assign({}, data);  
// The new way:  
const clone2 = { ...data };  
// Either results in:  
// { x: 42, y: 27, label: 'Treasure' }
```

[@bmeurer, @mathias].join(@v8js)

```
// Merge two objects:  
const defaultSettings = { logWarnings: false, logErrors: false };  
const userSettings = { logErrors: true };  
// The old way:  
const settings1 = Object.assign({}, defaultSettings, userSettings);
```

[@bmeurer, @mathias].join(@v8js)

```
// Merge two objects:  
const defaultSettings = { logWarnings: false, logErrors: false };  
const userSettings = { logErrors: true };  
// The old way:  
const settings1 = Object.assign({}, defaultSettings, userSettings);  
// The new way:  
const settings2 = { ...defaultSettings, ...userSettings };  
// Either results in:  
// { logWarnings: false, logErrors: true }
```

[@bmeurer, @mathias].join(@v8js)

Promise.prototype.finally

```
fetchAndDisplay( {  
  url: someUrl,  
  element: document.querySelector('#output')  
} );
```

[@bmeurer, @mathias].join(@v8js)

```
const fetchAndDisplay = ({ url, element }) => {
  showLoadingSpinner();
  fetch(url)
    .then((response) => response.text())
    .then((text) => {
      element.textContent = text;
      hideLoadingSpinner();
    })
    .catch((error) => {
      element.textContent = error.message;
      hideLoadingSpinner();
    });
};
```

[@bmeurer, @mathias].join(@v8js)

```
const fetchAndDisplay = ({ url, element }) => {
  showLoadingSpinner();
  fetch(url)
    .then((response) => response.text())
    .then((text) => {
      element.textContent = text;
      hideLoadingSpinner();
    })
    .catch((error) => {
      element.textContent = error.message;
      hideLoadingSpinner();
    });
};
```

[@bmeurer, @mathias].join(@v8js)

```
const fetchAndDisplay = ({ url, element }) => {
  showLoadingSpinner();
  fetch(url)
    .then((response) => response.text())
    .then((text) => {
      element.textContent = text;
      hideLoadingSpinner();
    })
    .catch((error) => {
      element.textContent = error.message;
      hideLoadingSpinner();
    });
};
```

[@bmeurer, @mathias].join(@v8js)

```
const fetchAndDisplay = ({ url, element }) => {
  showLoadingSpinner();
  fetch(url)
    .then((response) => response.text())
    .then((text) => {
      element.textContent = text;
      hideLoadingSpinner();
    })
    .catch((error) => {
      element.textContent = error.message;
      hideLoadingSpinner();
    });
};
```

[@bmeurer, @mathias].join(@v8js)

```
const fetchAndDisplay = ({ url, element }) => {
  showLoadingSpinner();
  fetch(url)
    .then((response) => response.text())
    .then((text) => {
      element.textContent = text;
      hideLoadingSpinner();
    })
    .catch((error) => {
      element.textContent = error.message;
      hideLoadingSpinner();
    });
};
```

[@bmeurer, @mathias].join(@v8js)

```
const fetchAndDisplay = ({ url, element }) => {
  showLoadingSpinner();
  fetch(url)
    .then((response) => response.text())
    .then((text) => {
      element.textContent = text;
      hideLoadingSpinner();
    })
    .catch((error) => {
      element.textContent = error.message;
      hideLoadingSpinner();
    });
};
```

[@bmeurer, @mathias].join(@v8js)

```
const fetchAndDisplay = ({ url, element }) => {
  showLoadingSpinner();
  fetch(url)
    .then((response) => response.text())
    .then((text) => {
      element.textContent = text;
    })
    .catch((error) => {
      element.textContent = error.message;
    })
    .finally(() => {
      hideLoadingSpinner();
    });
};
```

[@bmeurer, @mathias].join(@v8js)

```
const fetchAndDisplay = async ({ url, element }) => {
  showLoadingSpinner();
  try {
    const response = await fetch(url);
    const text = await response.text();
    element.textContent = text;
  } catch (error) {
    element.textContent = error.message;
  } finally {
    hideLoadingSpinner();
  }
};
```

[@bmeurer, @mathias].join(@v8js)

# More info

- Async iterators and generators: [mths.be/bwt](https://mths.be/bwt)
- Promise.prototype.finally: [mths.be/bwq](https://mths.be/bwq)
- Object rest and spread properties: [mths.be/bwr](https://mths.be/bwr)
- RegExp features: [mths.be/bws](https://mths.be/bws)

[[@bmeurer](#), [@mathias](#)].join(@v8js)

# PWA Roadshow @ Google Munich

March 8th, i.e. this Thursday

**Register now: [mths.be/bwp](http://mths.be/bwp)**

# Thank you!



[**@bmeurer**, **@mathias**].join(**@v8js**)