V8: ECMAScript 2015 and Beyond

Adam Klein <adamk@chromium.org>
BlinkOn 6
2016-06-17
Video recording of talk: https://www.youtube.com/watch?v=KrGOzEwqRDA

What's this “ECMAScript 2015”?

- Also known as “ES6”, it's a huge addition to the JavaScript language spec
- V8 added support incrementally, with a big push in the past year; Chromium M52 includes ES2015 + ES2016
- Well supported by SpiderMonkey (Firefox), Chakra (Edge) and JavaScriptCore (Safari)
- Already in broad use by developers through transpilers

A brief tour of ES2015

...but first, a brief history of ES5

So you want to define a class

```
function MyClass(val) {
  this.val = val;
}
```

So you want to define a class

```
function MyClass(val) {
  this.val = val;
}

MyClass.prototype.method = function() {
  return this.val;
};
```

So you want to define a (sub)class

```
function MySubClass(val, otherVal) {
  MyClass.call(this, val);
  this.otherVal = otherVal;
}
```

So you want to define a (sub)class

```
function MySubClass(val, otherVal) {
  MyClass.call(this, val);
  this.otherVal = otherVal;
}

MySubClass.prototype = Object.create(MyClass.prototype);
```

So you want to define a (sub)class

```
function MySubClass(val, otherVal) {
  MyClass.call(this, val);
  this.otherVal = otherVal;
}

MySubClass.prototype = Object.create(MyClass.prototype);
MySubClass.prototype.method = function() {
  console.log("overriding, otherVal = " + this.otherVal);
  return MyClass.prototype.method.call(this);
};
```

So you want to define a (sub)class

```
function MySubClass(val, otherVal) {
  MyClass.call(this, val);
  this.otherVal = otherVal;
}

MySubClass.prototype = Object.create(MyClass.prototype);
MySubClass.prototype.method = function() {
  console.log("overriding, otherVal = " + this.otherVal);
  return MyClass.prototype.method.call(this);
};
MySubClass.prototype.constructor = MySubClass;
```

ES2015 Classes to the rescue

Defining a class in ES2015

```
class MyClass {
  constructor(val) {
    this.val = val;
  }

  method() {
    return this.val;
  }
}
```

Defining a subclass in ES2015

```
class MySubClass extends MyClass {
  constructor(val, otherVal) {
    super(val);
    this.otherVal = otherVal;
  }

  method() {
    console.log("overriding, otherVal = " + this.otherVal);
    return super.method();
  }
}
```

Built-in classes can be subclassed, too!

```
class MyElement extends HTMLElement {
  ...
}
```

More capable standard library

Sets

```
let set = new Set([1, 2, 3, 4]);
set.contains(3);  // true
set.contains(5);  // false
set.add(5);
set.contains(5);  // true
```

Sets & Maps

```
let set = new Set([1, 2, 3, 4]);
set.contains(3);  // true
set.contains(5);  // false
set.add(5);
set.contains(5);  // true

let map = new Map();
map.set("answer", 42);
map.get("answer");  // 42
```

Iteration

```
for (let item of set) { ... }
```

Iteration

```
for (let item of set) { ... }

for (let [key, value] of map) { ... }
```

Iteration

```
for (let item of set) { ... }

for (let [key, value] of map) { ... }

for (let key of map.keys()) { ... }
```

Iteration

```
for (let item of set) { ... }

for (let [key, value] of map) { ... }

for (let key of map.keys()) { ... }

for (let str of ["a", "b", "c"]) { ... }
```

Iteration

```
for (let item of set) { ... }

for (let [key, value] of map) { ... }

for (let key of map.keys()) { ... }

for (let str of ["a", "b", "c"]) { ... }

for (let div of document.querySelectorAll("div")) { ... }
```

More syntactic goodness

Arrow functions

```
[1, 2, 3, 4].map(el => el * 2);  // [2, 4, 6, 8]
```

Arrow functions

```
[1, 2, 3, 4].map(el => el * 2);  // [2, 4, 6, 8]

class MyClass {
  constructor() {
    // |this| is bound as you might expect!
    this.callback = () => this.doStuff();
    ...
    setTimeout(this.callback, 0);
  }
  ...
}
```

...and much, much more

- Destructuring
- Default, rest, & spread parameters
- Template strings
- Let & const
- Literal enhancements
- Generators
- WeakMap & WeakSet
- Proxies & Reflect
- Unicode Regexps
- Symbols
- Promises

Check out Seth Thompson's I/O talk for more details

Beyond

...but first, a bit more history

Async API: Callbacks

```
navigator.geolocation.getCurrentPosition(
  function(pos) { /* do something with pos */ },
  function(err) { /* handle the error */ }
);
```

Async API: Promises

```
fetch("/some/url/for/state")
  .then(function(response) { return response.json(); })
  .then(function(json) { /* do something with json */ })
  .catch(function(e) { /* handle exception */ });
```

Async API: Promises & arrow functions

```
fetch("/some/url/for/state")
  .then(response => response.json())
  .then(json => { /* do something with json */ })
  .catch(e => { /* handle exception */ });
```

async/await

```
async function fetchJSON() {
  let response = await fetch("/some/url/for/state");
  let json = await response.json();
  return json;
}
```

async/await

```
async function fetchJSON() {
  try {
    let response = await fetch("/some/url/for/state");
    let json = await response.json();
    return json;
  } catch (e) {
    /* handle exception */
  }
}
```

async/await

```
async function fetchJSON() {
  let response = await fetch("/some/url/for/state");
  let json = await response.json();
  return json;
}

fetchJSON().then(json => { /* do something with json */ });
```

async/await

```
async function fetchJSON() {
  let response = await fetch("/some/url/for/state");
  let json = await response.json();
  return json;
}

async function updateState() {
  let json = await fetchJSON();
  ...
}
```

Status

- At Stage 3 in the TC39 spec process (ready for implementations)

Status

- At Stage 3 in the TC39 spec process (ready for implementations)
- Likely to be part of ES2017

Status

- At Stage 3 in the TC39 spec process (ready for implementations)
- Likely to be part of ES2017
- In development in V8, hoping to ship in Q3

& Beyond

What's next?

- Modules
    - Work in progress in V8 + Blink to support <script type="module">
    - Spec work ongoing on a comprehensive loader
    - Come to the unconference session up next to learn more
- Possible upcoming features
    - Class fields, public and private
    - Async iteration
- TC39 proposals: https://github.com/tc39/proposals
    - Later stages are closer to stable

Thanks!

Questions?