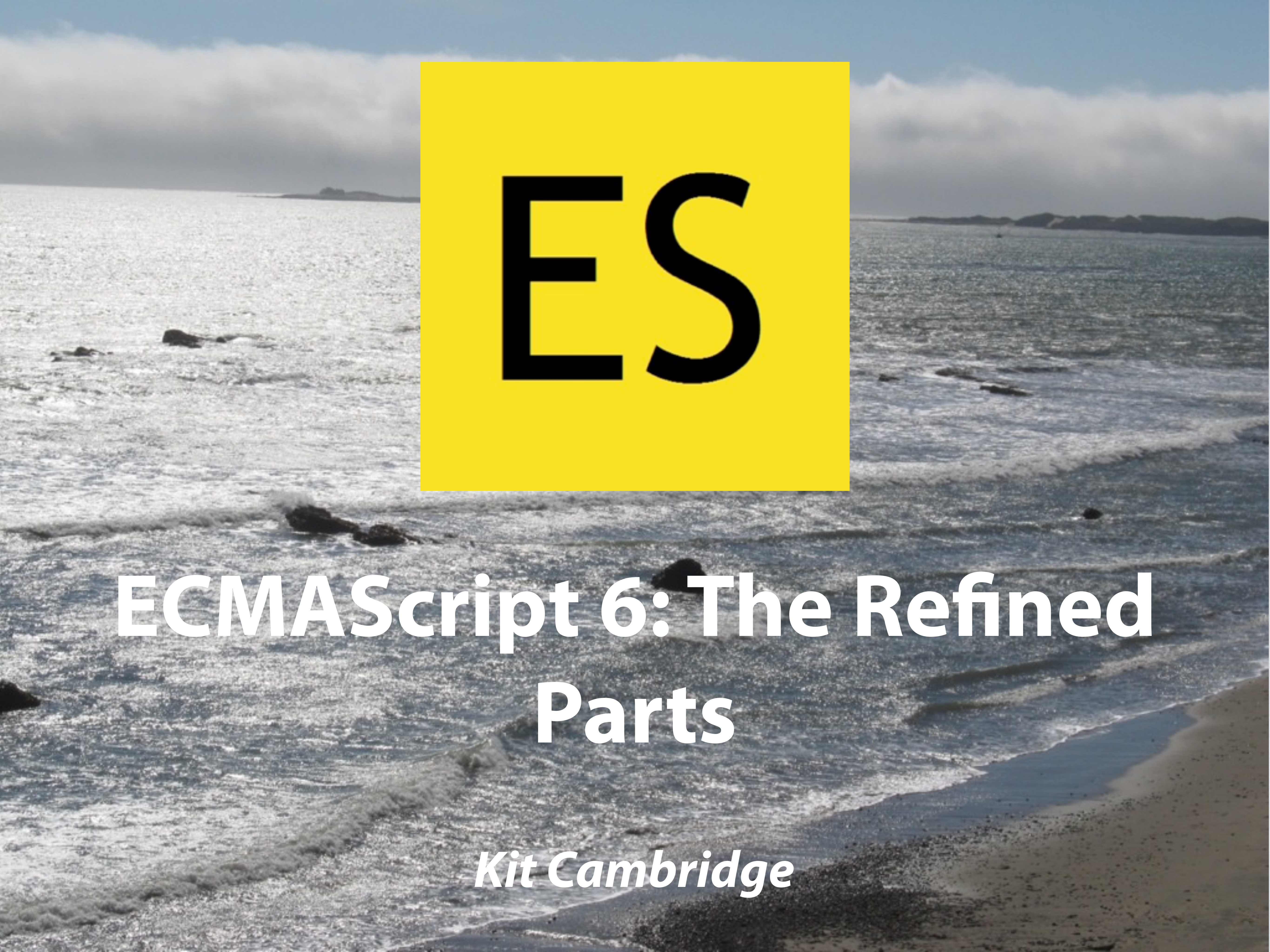

ES

ECMAScript 6: The Refined Parts

Kit Cambridge

Every technology has a story.

1995

Function expressions

Regular expressions

Array and object literals

`throw`

String, number, and date methods

1999

`try...catch`

Standardization and the future

`switch`

`in`

`===`

`instanceof`

`do...while`

Static type checking

Pragmas

Namespaces

Classes

Strict mode

Packages

Interfaces

Optional type annotations

Lexical binding

2005

Iterators

Diversions, Digressions, and Detours

Tail-call optimization

Block scope

`...rest` *parameters*

Overloading

Generators

Tighter regular expression, `eval`*,*
error, and global object semantics

New `Object`
reflection methods

Array extras

2009

Strict mode

JavaScript takes off

Native JSON support

Getters and setters

2013

JS

Full Circle

Iterators

Shorthand object literal syntax

New object methods

`const`

Arrow Functions

Maps

`let`

Generators

Default Parameters

`WeakMaps`

Template strings

Proxies

`...rest` *parameters*

Destructuring assignment

Sets

Modules

`...spread` *operator*

Tail-call optimization

Classes

Reflection methods

Symbols

Binary data

Arrow Functions

Pattern matching

Full Unicode support for strings

Block Scope

Lexical declarations with `let` and `const`

`let` introduces new semantics for blocks

Function declarations are now supported within blocks

```
if (false) {
  var value = 123;
  function getValue() {
    return value;
  }
}
getValue();
// => undefined
```

```
if (false) {
  let value = 123;
  function getValue() {
    return value;
  }
}
getValue();
// Throws a ReferenceError.
```

Closure Required

This is more common than you might think...

```
// The problem.
for (var length = 2; length--;) {
  var dimension = length ? 'Width' : 'Height';
  // Both methods will report the element's height.
  $['get' + dimension] = function getDimension(element) {
    return element['offset' + dimension];
  };
}
```

```
// The ES 6 solution.
for (var length = 2; length--;) {
  let dimension = length ? 'Width' : 'Height';
  $['get' + dimension] = function getDimension(element) {
    return element['offset' + dimension];
  };
}
```

Gotchas

Lexical declarations introduce new semantics.

```
function getMood({actions: {isSinging}}) {
  // Syntax error. Lexical declarations must be nested
  // within blocks.
  if (isSinging) let mood = 'greatlyImproved';

  // Syntax error. Constants cannot be re-declared.
  const isWritingCode = true;
  const isWritingCode = false;

  // Type error in strict mode. Constants are read-only.
  isWritingCode = false;

  // Syntax error. `var` declarations cannot shadow
  // `let` and `const` declarations.
  var isWritingCode = true;
}
```

Shorthand Object Literal Syntax

Initializers

```
var name = 'Kit';
var occupation = 'developer';
var results = {
  'name': name,
  'occupation': occupation
};
```

```
var name = 'Kit';
var occupation = 'developer';
var results = {name, occupation};
```

Method Definitions

```
function Speaker(name) {
  this.name = name;
  this.years = 0;
}

Speaker.prototype = {
  speak(message) {
    return `${this.name}: ${message}`;
  },
  get age() {
    return this.years;
  },
  set age(years) {
    if (Number.isFinite(years)) {
      this.years = years;
    }
};
```

Shorthand Method

Template Literal

Getter

New API Method

Destructuring Assignment

Extract values from arrays and objects

Swap variables

```
var {parse, stringify} = JSON;

var [, areaCode, local] = /^(\d{3})-(\d{3}-\d{4})$/.exec(phone);

[left, right] = [right, left];

({'request': {headers}}) => headers
```

Destructuring Nested Objects

Extract values from a complex structure in a single statement.

```
var poets = [{
  "name": "T.S. Eliot",
  "works": [{
    "title": "The Love Song of J. Alfred Prufrock",
    "date": 1915
  }, {
    "title": "Rhapsody on a Windy Night",
    "date": 1917
  }]
}, {
  "name": "Ezra Pound",
  "works": [{
    "title": "Ripostes",
    "date": 1912
  }]
}];

var [{'name': author, 'works': [, {title, date}]}] = poets;
`"${title}", by ${author}, was published in ${date}.`
// => '"Rhapsody on a Windy Night", by T.S. Eliot, was published in 1917.'
```

`...spread` operator

Expand an array of arguments without altering `this`

Supports constructors

Convenient syntax for merging arrays and array-like objects

Convert any object with a `length` property into an array

`...rest` parameters

Supplants the `arguments` object

Always returns an array, even when parameters are omitted

```
function getCredentials({request: {headers: {authorization}}}) {
  let [scheme, ...components] = authorization.split(' ');
  if (scheme != 'Basic' || components.length > 1) {
    return null;
  }
  let [credentials] = components;
  credentials = atob(credentials);
  if (!credentials.contains(':')) {
    return null;
  }
  let [, name, password] = /^([^:]+):(\w+)$/.exec(credentials);
  return {name, password};
}

getCredentials(request);
// => { 'name': 'Kit', 'password': '...' }
```

Full Unicode String Support

Regular expressions, `charAt`, `charCodeAt`, `slice`, etc. operate on code units, not characters

Unicode characters outside the basic multilingual plane are represented as surrogate pairs comprising two code units

`mathiasbynens.be/notes/javascript-encoding`

```
'𝌆'.length;
// => 2
'𝌆'.charCodeAt(0);
// => 55348
String.fromCharCode(119558);
// => '팆'
'𝌆' == '\ud834\udf06'
// => true
```

```
'𝌆'.length;
// => 2
'𝌆'.codePointAt(0);
// => 119558
String.fromCodePoint(119558);
// => '𝌆'
'𝌆' == '\u{1d306}'
// => true
```

Maps, WeakMaps, and Sets

Maps support **sub-linear lookup** times

```
.get(), .set(), .has(), .delete(), .clear(), .forEach()
```

Entries are enumerated in insertion order

`WeakMaps` use weak references to allow **garbage collection**

Sets can find **unique** array elements in linear time

```
var unique = [...new Set([1, 2, 0, 2, 3, 'A', 'B', 0, 'C', 'C', 'D'])];
// => [1, 2, 0, 3, 'A', 'B', 'C', 'D']
```

WeakMaps all the way down...

```
// Leak-free element storage engine.
var storage = new WeakMap();
```

```
function store(element, name, value) {
  if (!storage.has(element)) {
    // Create the element data store if
    // it doesn't exist.
    storage.set(element, new WeakMap());
  }
  // Associate the name and value with
  // the element.
  storage.get(element).set(name, value);
  return element;
}

function retrieve(element, name) {
  if (!storage.has(element)) {
    return;
  }
  return storage.get(element).get(name);
}
```

```
function unstore(element, name) {
  if (!storage.has(element)) {
    return;
  }
  let data = storage.get(element);
  let value = data.get(name);
  data.delete(name);
  return value;
}
```

Tagged Template Literals

```javascript
function escape(values, ...substitutions) {
  let {raw, 'raw': {length}} = values, results = '';
  for (let index = 0; index < length; index++) {
    results += raw[index];
    if (index + 1 == length) {
      break;
    }
    results += String(substitutions[index]).replace(/[&<>"']/g,
      (match) => `&#x${match.charCodeAt(0).toString(16)};`)
  }
  return results;
}

let name = 'Kit<script>alert(1)</script>';
escape`<span class="name">${name}</span>`;
// => '<span class="name">Kit&#x3c;script&#x3e;alert(1)&#x3c;/
script&#x3e;</span>'
```

=> Functions

Based on expression closures

```
[1, 2, 3, 4, 5].filter(function (value) value % 2);
// => [1, 3, 5]
```

Parentheses not required for single-parameter functions

```
[1, 2, 3, 4, 5].filter(value => value % 2);
// => [1, 3, 5]
```

Blocks are not required for single-value expressions

```
let identify = (() => ({ 'toString': () => { return 'Kit'; } });
`${identify()} <3s JavaScript.`;
// => 'Kit <3s JavaScript.'
```

Semantics identical to **bound functions**

What else?

Proxies

Generators

Symbols

Iterators

Modules

Binary Data

Classes

Tail-call Optimization

Pattern Matching

Reflection Methods

There’s something for everyone.

Shorthand Syntax

Tooling

Data Structures

Modularity

Core Refinements

When can I use...?

Firefox has supported some features since 2.0

Toggle the “Enable Experimental JavaScript” option (`about:flags`) in Chrome and Chromium

`kangax.github.com/es5-compat-table/es6`

When can I use...?

Follow @**esdiscuss** for digestible summaries

Try **`benvie.github.com/continuum`**

Thank you!

@kitcambridge

`kitcambridge.be`