ECMAScript 6

Course 8

ECMA

歐洲電腦製造商協會

Outline

- Syntax
- New Objects
- Fetch
- 2016

Syntax

Syntax

- const
- let
- Template string
- Arrow function
- Default parameter
- Spread operator
- Destructing assign
- Object initializer
- Class
- Import/Export

const

```
const foo = 100;
foo; // 100

foo = 101;

foo; // 100
```

let

```
{
   let foo = 100;
   foo; // 100
}

foo; //undefined
```

let

```
for (let i = 0; i < len; i++) {
  // blah
}
```

Template String

```
var name = 'world';

var greeting = `hello ${name}`;

greeting; //hello world;
```

Tab
Q
W
E
A
S

`

- Grave accent
- Back tick
- Shell: execute command in between

Arrow Function

```
var square = (x) => {
  return x * x;
};

var square2 = x => x * x;
```

Arrow Function

```
// Empty function body
var foo = (x) => {}

// Single parameter
var foo = x => {}

// No parameter
var foo = () => {}

// More parameters
var foo = (x, y) => {}
```

Arrow Function

```
// Single expression
var foo = (x) => x*x

// Multiple expression
var foo = (x) => {
 let y = x * x;
 // need return
 return y * x;
}
```

Arrow Function

- Auto return result of single expression
- Lexical this , like CoffeeScript

Default Parameter

```
function foo(x = 5, y = 5) { }
```

Rest Parameters

```
function foo(x = 5, ...rest) {
  rest;
}

foo(1, 2, 3, 4, 5, 6);
// [2,3,4,5,6]
```

Spread Operator

```
function f(x, y, z) { }
var args = [0, 1, 2];

f.apply(null, args);

f(...args);
```

Spread Operator

```
var arg2 = [...args, 3, 4];
// [0,1,2,3,4]

var arg3 = arg.push(...arg2);
// [0,1,2,0,1,2,3,4]
```

Destructing Assign

```
var a, b;

[a, b] = [1, 2];
//a:1, b:2
```

Destructing Assign

```
[a, b] = [b, a];
//swap

[a, ,[b, c]] = [1, 0, [2, 3]];
//a:1, b:2, c:3

{lan: a, lon: b} = getPOS();
//object destructing
```

Destructing & Spread

```
[a, ...b] = [1, 2, 3, 4, 5];
//a:1, b:[2,3,4,5]
```

Object Initializer Shorthand

```
function getPoint() {
  var x = 1;
  var y = 10;

  return {x, y};
}
```

Object Initializer Shorthand

```
var a = "foo", b = 42, c = {};
var o = { a: a, b: b, c: c };

var a = "foo", b = 42, c = {};
var o = { a, b, c };
```

Computed Property

```
var prop = "foo";
var o = {
  [prop]: "hey",
  ["b" + "ar"]: "there",
};
```

Class

```
class Counter {
  constructor() {
    this.count = 0;
  }
  tick() {
    this.count++;
  }
  get count() {
    return this.count;
  }
}
```

Class Extend

```
class People extends Counter {
  constructor(people) {
    this.people = people;
    for (let p in people) {
      this.tick();
    }
  }
}

var p = new People([1,2,3,4,5]);
p.count; //5
```

Class

- No multiple inheritance
- Define property only in constructor

Import/Export

- The syntax to define module

Export

```
export default 42;
export default function foo() {};

export { encrypt };
export { decrypt as dec };
export { encrypt as en } from 'crypto';
export * from 'crypto';
```

Export

```
export default 42;
export default function foo() {};

export { encrypt };
export { decrypt as dec };
export { encrypt as en } from 'crypto';
export * from 'crypto';
```

Import

```
import 'jquery';
import $ from 'jquery';
import { $ } from 'jquery';
import { $ as jQuery } from 'jquery';
```

Map/Set

Map

- Like object, {...}
- Key, value pair data structure
- Use non-string data as key
- Native object's key will use toString

Map

```
m = new Map();
m.set(true, 'T');
m.set(false, 'F');

m.size; //2

m.get(true); //"T"
m.get(false); //"F"

m.get(1); // undefined
```

Map Methods

clear

delete

entries

forEach

get

has

keys

set

values

Set

- Like array, […]
- Can’t get value at specific index
- Use for…of

Set

```
s = new Set();
s.add('A');
s.add('B');
s.add('C');

for (v of s) {
  console.log(v);
}
// A, B ,C
```

Set Methods

add

clear

delete

entries

forEach

has

values

WeakMap/WeakSet

- Not prevent Garbage Collection

for…of

for…of

- New loop method
- Like CoffeeScript’s for…in
- Used to loop iterable object items

Iterable

Array

String

Map

Set

Iterator

Iterator

- A new interface in ES spec
- User can implement custom iterator
- An object with next method

Iterator.next

- Return an object with value and done
- value is next item's value
- done shows is this iterator finished
- Can't reuse

Iterator

```
var it = idMaker();

console.log(it.next().value);
console.log(it.next().value);
console.log(it.next().value);
```

Generator

Generator

- Like idMaker
- Generator is a function, generate iterator
- Different invoke will create different iterator, iterate the same list.

Generator

```
function idMaker() {
  var index = 0;
  return {
    next: function () {
      return {
        value: index++,
        done: false
      };
    }
  };
}
```

yield

- yield is generator helper
- Let you easy to create generator

yield

```
function* idMaker(){
  var index = 0;
  while(true)
    yield index++;
}
```

yield

```
function* idMaker(){
  var index = 0;
  while(index < 6)
    yield index++;
}
```

yield

- * is the indicator to tell runtime
- yield is return point

yield

This is a generator

```
function* idMaker(){
  var index = 0;
  while(index < 6)
    yield index++;
}
```

http://mdn.io/generator

yield

```
function* idMaker(){
  var index = 0;
  while(index < 6)
    yield index++;
}
```

starts here

↓

return

yield

```
   function* idMaker(){
     var index = 0;
     while(index < 6)
return yield index++;
   }
```

starts here

yield

- Function end will return done: true

Iterable

- Have generator function in the object
- Under @@iterator property

Iterable

```
ID = {};

ID['@@iterator'] = idMaker;
//or use Symbol
ID[Symbol.iterator] = idMaker;

for (id of ID) {
  id;
  //0,1,2,3,4,5
}
```

Comprehension

簡約式

```
var ns = [1, 2, 3, 4];

var dbls = [for (i of ns) i*2];

dbls; // [2, 4, 6, 8]
```

CoffeeScript Syntax

```
arr = [1, 2, 3, 4];

res = (x for x in arr);
```

2 Level Comprehension

```
var ms = [1, 2, 3, 4];
var ns = [2, 3, 4, 5];

[for (i of ms) for (j of ns) i*j];
// [2, 3, 4, 5, 4, 6, 8, 10, 6, 9, ...]
```

Conditional Comprehension

```
var ns = [1, 2, 3, 4];

[for (i of ns) if (i % 2) i];
//[1, 3]
```

Comprehension for Iterator

```
var ns = [1, 2, 3, 4];

(for (i of ns) if (i % 2) i);
//iterator with values [1, 3]
```

Symbol

Symbol

- New primitive type
- Used as property key
- Each symbol is unique

Symbol

```
let symbol1 = Symbol();
```

Symbol

```
const MY_KEY = Symbol();
let obj = {
    [MY_KEY]: 123
};
```

Symbol

```
const RED    = Symbol();

const BLUE   = Symbol();

const GREEN  = Symbol();

const YELLOW = Symbol();
```

Symbol.iterator

```
var myIterable = {}
myIterable[Symbol.iterator] = function* () {
    yield 1;
    yield 2;
    yield 3;
};

[...myIterable] // [1, 2, 3]
```

Promise

ES6 Promise

- Inspires by jQuery Deferred
- Based on CommonJS Promise/A & Promise/A+

Promise/A+

- https://promisesaplus.com/

Promise

```
new Promise(function (resolve, reject) {
  //...
});
```

Promise

```
function JQXHR(url) {
  var xhr = new XMLHttpRequest();
  var dfd = new $.Deferred();
  xhr.onreadystatechange = function () {
    if (xhr.readyState == 4) {
      dfd.resolve(xhr.response);
    }
  }
  xhr.open('GET', url);
  xhr.send();
  return dfd.promise();
}
```

Promise

```
function ESXHR(url) {
  var xhr = new XMLHttpRequest();
  var dfd = new Promise(function (resolve) {
    xhr.onreadystatechange = function () {
      if (xhr.readyState == 4) {
        resolve(xhr.response);
      }
    }
  });
  xhr.open('GET', url);
  xhr.send();
  return dfd;
}
```

Promise

```
dfd.then(null, function () {
  return $.Deferred().resolve();
});
```

Promise

```
dfd.then(function (val) {
  if (val === 0) {
    throw new Error('');
  }
});
```

Promise

```
dfd.then(null, function () {
  //rejected handler
});

dfd.catch(function () {
  //rejected handler
});
```

Promise

- Cache
- ImageBitmap
- ServiceWorker
- Notification
- RTC
- Web Audio
- TV
- Feature Detect
- Stream
- fetch

Proxy/Reflect

Proxy

- The Proxy object is used to define custom behavior for fundamental operations

- e.g. property lookup, assignment, enumeration, function invocation, etc

Proxy

```
let target = {};
let handler = {
    get(target, propKey, receiver) {
        console.log('get ' + propKey);
        return 123;
    }
};

let proxy = new Proxy(target, handler);
```

Traps

- getPrototypeOf
- setPrototypeOf
- isExtensible
- preventExtensions
- getOwnPropertyDescriptor
- defineProperty
- has
- get
- set
- deleteProperty
- enumerate
- ownKeys
- apply
- construct

Use Cases

- DOM
- Observer
- Negative array index
- Deal unknown properties

Reflect

- A “@reflect” module is a more natural place for many of the reflection methods previously defined on Object.
- For backwards-compatibility purposes, it is unlikely that the static methods on Object will disappear.

http://stackoverflow.com/questions/25421903/what-does-the-reflect-object-do-in-javascript

Reflect Define

```
try {
  Object.defineProperty(obj, name, desc);
  // success
} catch (e) {
  // failure
}
```

https://github.com/tvcutsem/harmony-reflect/wiki

Reflect Define

```
if (Reflect.defineProperty(obj, name, desc)) {
  // success
} else {
  // failure
}
```

Reflect Apply

```
f.apply(obj, args)

Function.prototype.apply.call(f, obj, args)
```

Reflect Apply

```
Reflect.apply(f, obj, args)
```

Reflect Construct

```
var obj = new F(...args)
```

Reflect Construct

```
var obj = Reflect.construct(F, args)
```

fetch

XHR

- XMLHttpRequest
- by Microsoft, 1999
- Make HTTP request in web page

XHR Spec

- W3C Working Draft since 2006, Level 2 since 2008
- WHATWG Living Standard since 2011

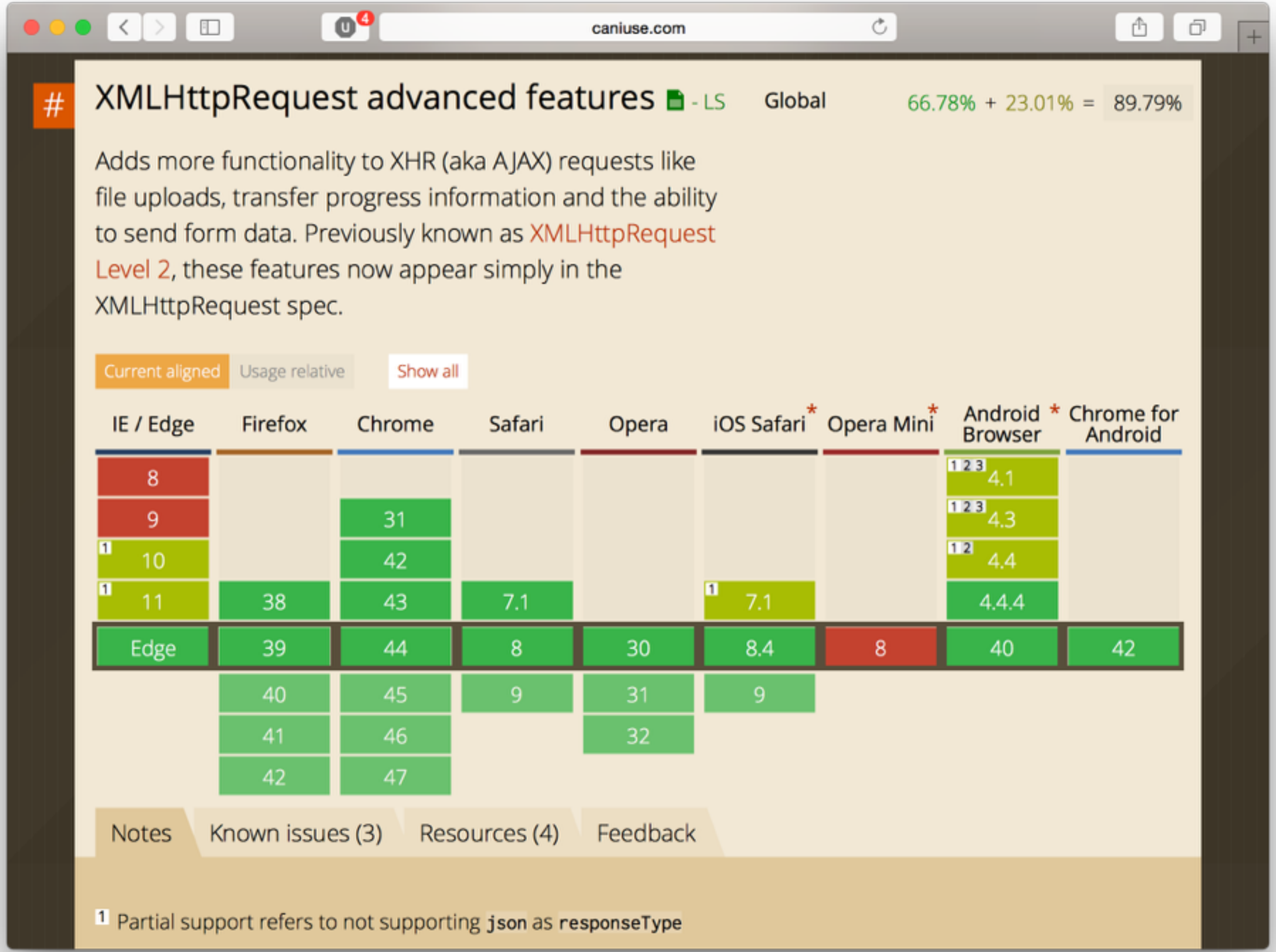
What’s in 2.0

CORS

FormData

ArrayBuffer

File/Blob



XMLHttpRequest advanced features - LS

Global 66.78% + 23.01% = 89.79%

Adds more functionality to XHR (aka AJAX) requests like file uploads, transfer progress information and the ability to send form data. Previously known as XMLHttpRequest Level 2, these features now appear simply in the XMLHttpRequest spec.

IE / Edge	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
8							[1,2,3] 4.1	
9		31					[1,2,3] 4.3	
[1] 10		42					[1,2] 4.4	
[1] 11	38	43	7.1		[1] 7.1		4.4.4	
Edge	39	44	8	30	8.4	8	40	42
	40	45	9	31	9			
	41	46		32				
	42	47						

[1] Partial support refers to not supporting `json` as `responseType`

XHR

- 15 years old

New Stuffs

JSON

FormData

File/Blob

Promise

ServiceWorker

ArrayBuffer

Not every new tech can apply to XHR without break it.

So here is the new XHR.

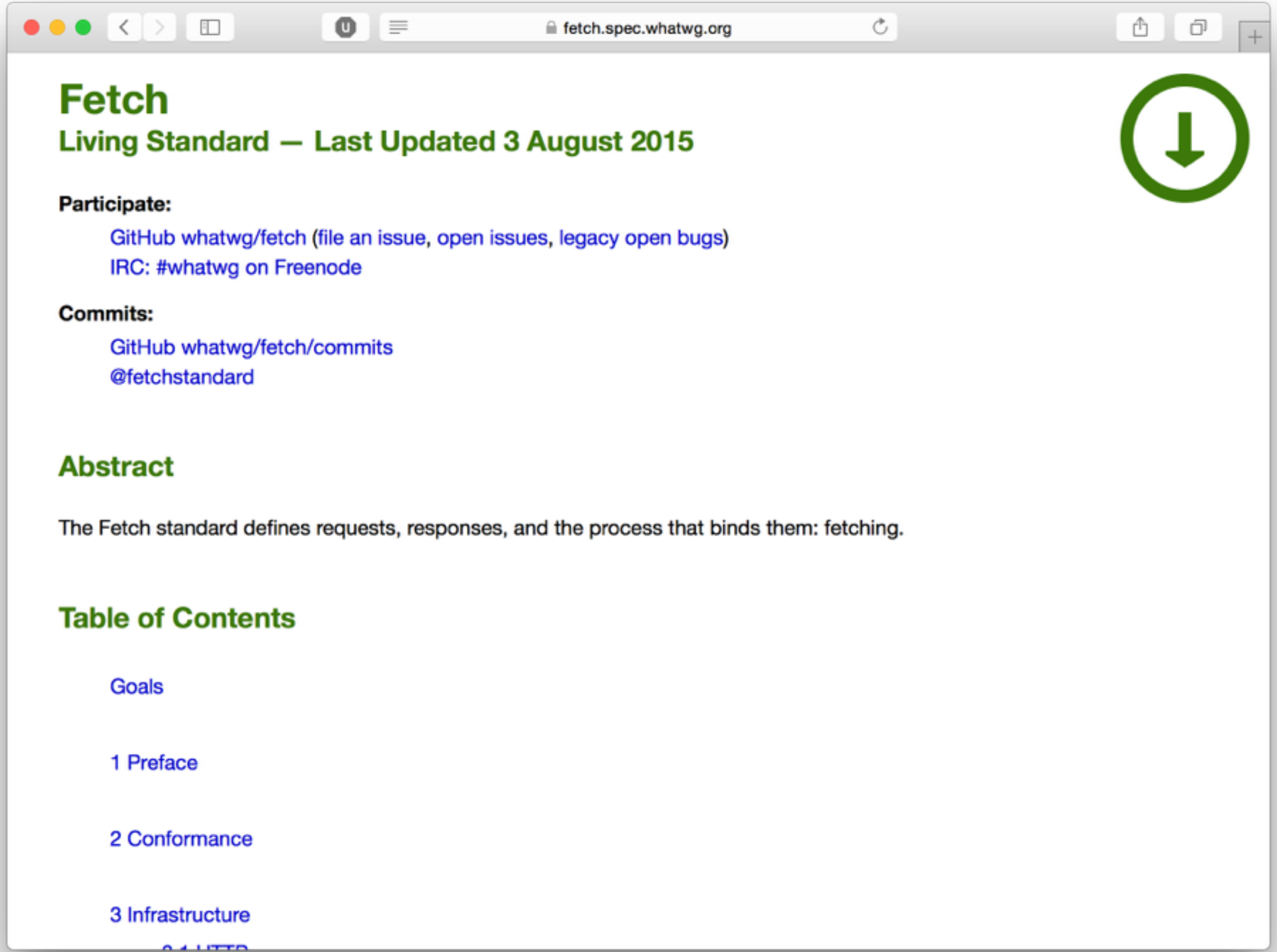
Fetch

Fetch

- The unified architecture of resource fetching
  - img, script, css, cursor image, …etc
- fetch() JavaScript API

fetch

- Promise based
- JSON support
- Simple naming, options object
- Designed as low level API



Fetch

Living Standard — Last Updated 3 August 2015

Participate:

GitHub whatwg/fetch (file an issue, open issues, legacy open bugs)
IRC: #whatwg on Freenode

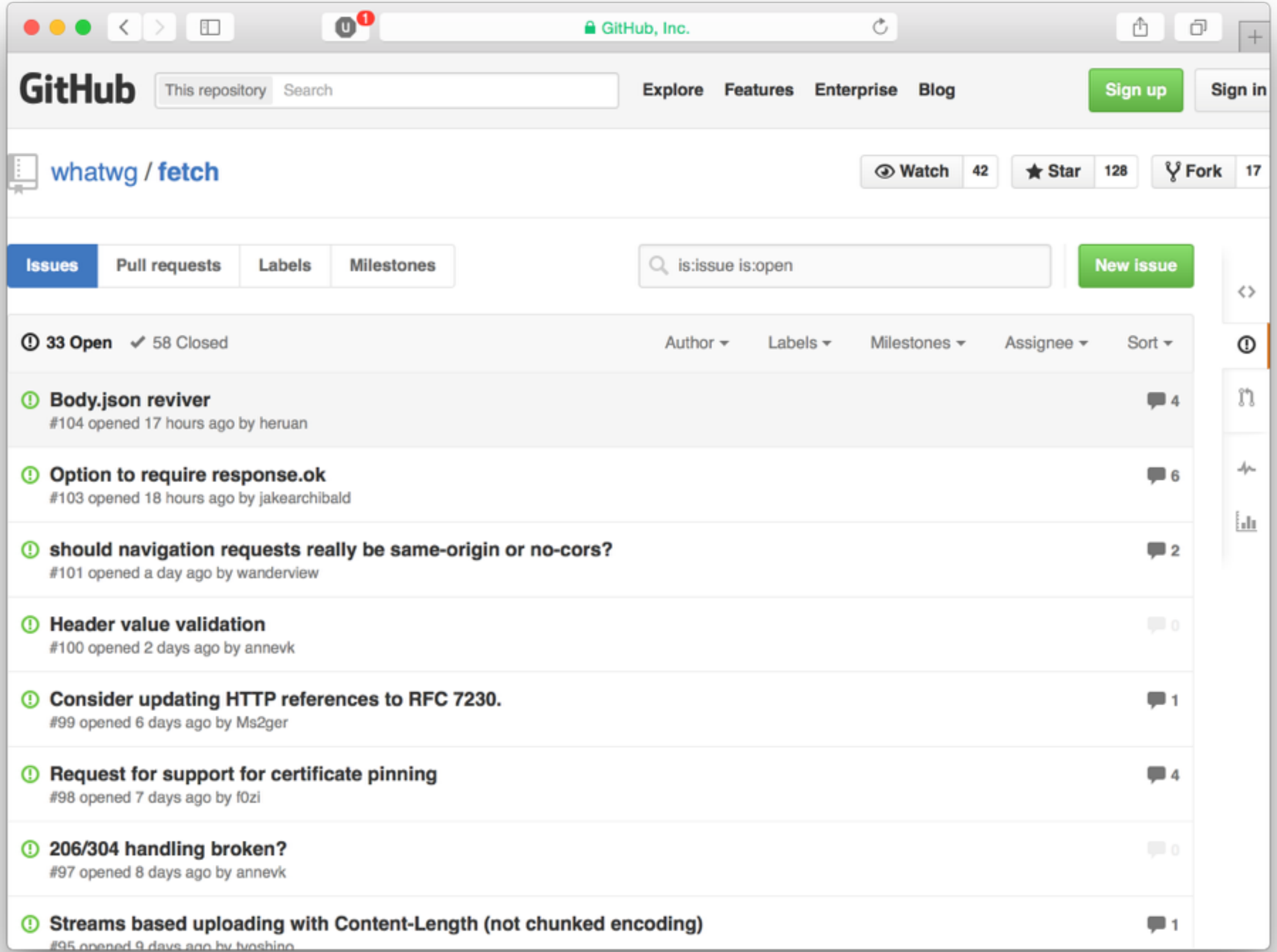
Commits:

GitHub whatwg/fetch/commits
@fetchstandard

Abstract

The Fetch standard defines requests, responses, and the process that binds them: fetching.

Table of Contents



33 Open ✓ 58 Closed

Body.json reviver 4
#104 opened 17 hours ago by heruan

Option to require response.ok 6
#103 opened 18 hours ago by jakearchibald

should navigation requests really be same-origin or no-cors? 2
#101 opened a day ago by wanderview

Header value validation 0
#100 opened 2 days ago by annevk

Consider updating HTTP references to RFC 7230. 1
#99 opened 6 days ago by Ms2ger

Request for support for certificate pinning 4
#98 opened 7 days ago by f0zi

206/304 handling broken? 0
#97 opened 8 days ago by annevk

Streams based uploading with Content-Length (not chunked encoding) 1
#95 opened 9 days ago by tyoshino

```
fetch('./api/some.json').then(function(response) {
  return response.json();
}).then(function(data) {
  console.log(data);
});
```

```
fetch('./api/some.json').then(function(response) {
  return response.json();
}).then(function(data) {
  console.log(data);
});
```

```
fetch('./api/some.json').then(function(response) {
  return response.json();
}).then(function(data) {
  console.log(data);
});
```

```
fetch('./api/some.json').then(function(response) {
  return response.json();
}).then(function(data) {
  console.log(data);
});
```

```
fetch('./api/some.txt').then(function(response) {
  return response.text();
}).then(function(data) {
  console.log(data);
});
```

```
fetch('./api/some.file').then(function(response) {
  return response.blob();
}).then(function(data) {
  // console.log(data);
});
```

```
fetch('./api/some.json')
  .then(res  => res.json())
  .then(data => console.log(data));
```

```
fetch('./api/entry', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(data)
}).then(function(response) {

  response.json().then(data => console.log(data));

  return response;

});
```

```
fetch('./api/entry', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/x-www-form-url-encoded'
  },
  body: $.param(data)
});
```

http://api.jquery.com/jquery.param/

```
fetch('./api/entry', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/x-www-form-url-encoded'
  },
  body: $.param(data)
});
```

http://api.jquery.com/jquery.param/

```
fetch('./api/entry', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/x-www-form-url-encoded'
  },
  body: $.param(data)
});
```

http://api.jquery.com/jquery.param/

```
var obj = { key1: 'value1', key2: [10, 20, 30] };

var str = $.param(obj);

// "key1=value1&key2%5B%5D=10&key2%5B%5D=20&key2%5B%5D=30"
```

https://www.npmjs.com/package/jquery-param

```
var obj = { key1: 'value1', key2: [10, 20, 30] };

var str = $.param(obj);

// "key1=value1&key2%5B%5D=10&key2%5B%5D=20&key2%5B%5D=30"
```

http://www.w3.org/TR/html5/forms.html#application/x-www-form-urlencoded-encoding-algorithm

```
var obj = { key1: 'value1', key2: [10, 20, 30] };

var str = $.param(obj);

// "key1=value1&key2%5B%5D=10&key2%5B%5D=20&key2%5B%5D=30"

//  key1=value1 &
//  key2[]=10 &
//  key2[]=20 &
//  key2[]=30
```

http://www.w3.org/TR/html5/forms.html#application/x-www-form-urlencoded-encoding-algorithm

Or you can post JSON, or other format

```
fetch('./api/entry', {
  method:'POST',
  headers: {
    'Content-Type': 'application/x-www-form-url-encoded'
  },
  body: $.param(data)
});
```

method

body

crendentials

headers

mode

cache

method

headers

body

mode

crendentials

cache

mode

- The one XHR can’t do
    - same-origin
    - cors
    - no-cors

no-cors

- Don't check CORS
- Get **opaque** response
- For ServiceWorker
- XHR can't do this

method

headers

body

mode

crendentials

cache

headers

- POJSO(Plain Old JavaScript Object)

```
{
    'Content-Type': 'text/plain',
    'X-Custom-Header', 'ProcessThisImmediately'
}
```

```
fetch('./api/some.json', {
  method: 'POST',
  mode: 'no-cors',
  headers: {
    'Content-Type': 'text/plain',
    'X-Custom-Header', 'ProcessThisImmediately'
  }
});
```

What if

What if

Header have multiple entry with the same name?

```
Set-Cookie: JSESSIONID=alphabeta120394049; HttpOnly
Set-Cookie: AWSELBID=baaadbeef6767676767690220; Path=/alpha
```

https://community.apigee.com/articles/2319/how-to-handle-multi-value-headers-in-javascript.html

RFC 2616

> Multiple message-header fields with the same field-name MAY be present in a message if and only if the entire field-value for that header field is defined as a comma-separated list [i.e., #(values)]

http://www.w3.org/Protocols/rfc2616/rfc2616-sec4.html#sec4.2

Header Class

- New global class
- For both request and response headers

https://developer.mozilla.org/en-US/docs/Web/API/Headers

```
var h = new Headers();

h.append(
  'Set-Cookie',
  'JSESSIONID=alphabeta120394049; HttpOnly'
);

h.append(
  'Set-Cookie',
  'AWSELBID=baaadbeef6767676767690220; Path=/alpha'
);

fetch(url, {
  headers: h
});
```

```
var h = new Headers();

h.append(
  'Set-Cookie',
  'JSESSIONID=alphabeta120394049; HttpOnly'
);

h.append(
  'Set-Cookie',
  'AWSELBID=baaadbeef6767676767690220; Path=/alpha'
);

fetch(url, {
  headers: h
});
```

```
var h = new Headers();

h.append(
  'Set-Cookie',
  'JSESSIONID=alphabeta120394049; HttpOnly'
);

h.append(
  'Set-Cookie',
  'AWSELBID=baaadbeef6767676767690220; Path=/alpha'
);

fetch(url, {
  headers: h
});
```

append()

delete()

get()

getAll()

has()

set()

Deal with Response

```
fetch('./api/some.json').then(function(response) {
  return response.json();
})
```

```
fetch('./api/some.json').then(function(response) {
  return response.json();
})
```

Response

- New global class
- Contains status, headers, body ...

ok

url

headers

status

statusText

type

ok

status

url

statusText

headers

type

ok

- “ok” is true when status is 2xx

Means

- Fetch promise will resolve even when server respond 404, 500 …
- Request is complete so Promise is resolved.

Reject?

- On exception, ex: Network Error

arrayBuffer()

blob()

formData()

json()

text()

bodyUsed

arrayBuffer()

blob()

formData()

json()

text()

bodyUsed

Response Body

- For formatted response body
- No auto format detection
- Body can consume only once
  (save memory)
- Use clone when need consume twice

```
fetch('...').then(function (res) {
  if (res.ok) {
    if (res.headers.get('Content-Type') === 'application/json'
      return res.json();
    }
  }
})
```

```
fetch('...').then(function (res) {
  if (res.ok) {
    if (res.headers.get('Content-Type') === 'application/json'
      return res.json();
    }
  }
})
```

```
fetch('...').then(function (res) {
  if (res.ok) {
    if (res.headers.get('Content-Type') === 'application/json'
      return res.json();
    }
  }
})
```

```
fetch('...').then(function (res) {
  if (res.ok) {
    if (res.headers.get('Content-Type') === 'application/json'
      return res.json();
    }
  }
})
```

New Classes

- Headers
- Response

New Classes

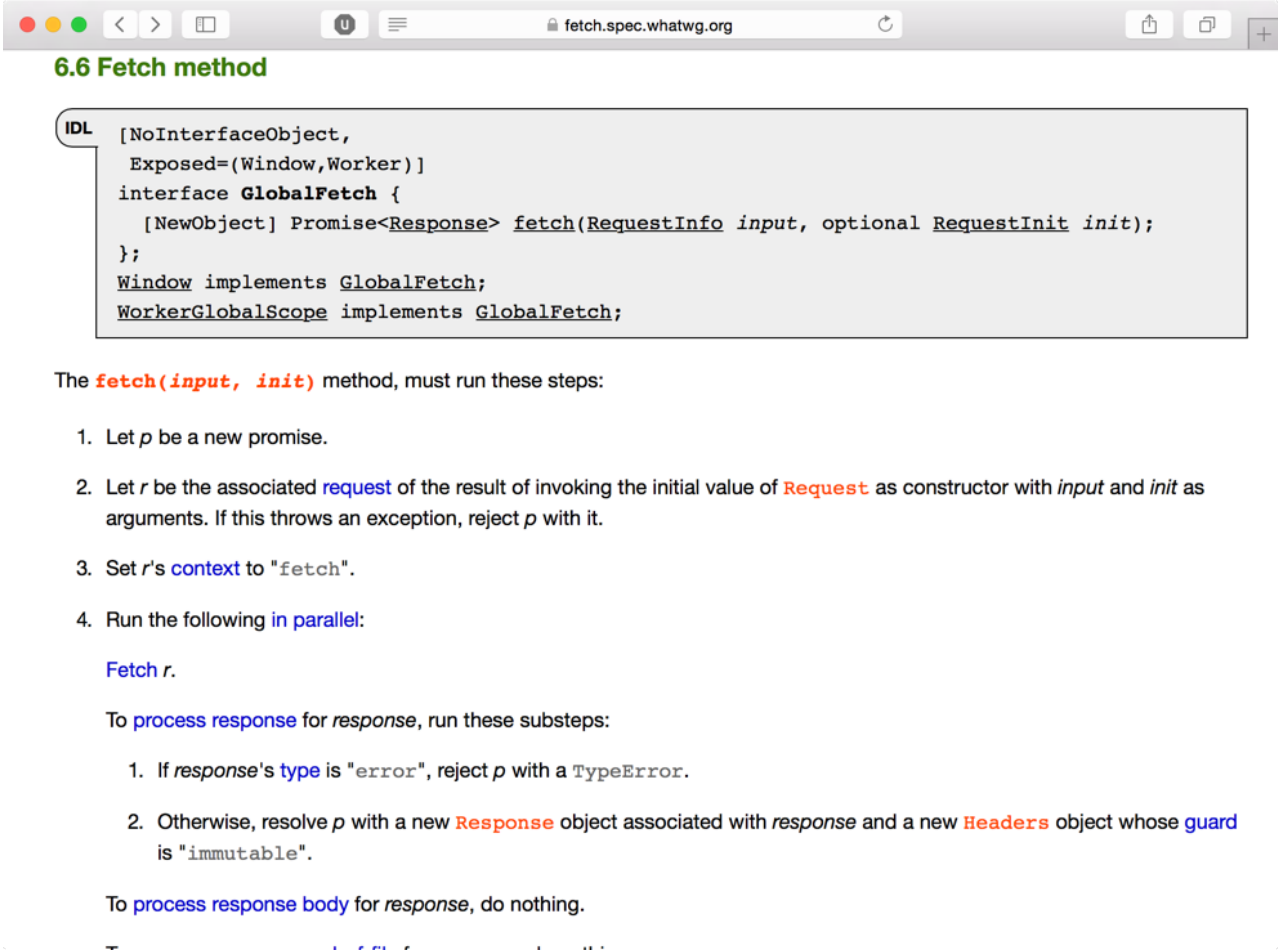
- Headers
- Response
- Request

Request

- Another new global class
- Contains method, headers, body …

Where to Use

- Inside fetch
- As fetch input



6.6 Fetch method

```
IDL
[NoInterfaceObject,
 Exposed=(Window,Worker)]
interface GlobalFetch {
  [NewObject] Promise<Response> fetch(RequestInfo input, optional RequestInit init);
};
Window implements GlobalFetch;
WorkerGlobalScope implements GlobalFetch;
```

The **`fetch(input, init)`** method, must run these steps:

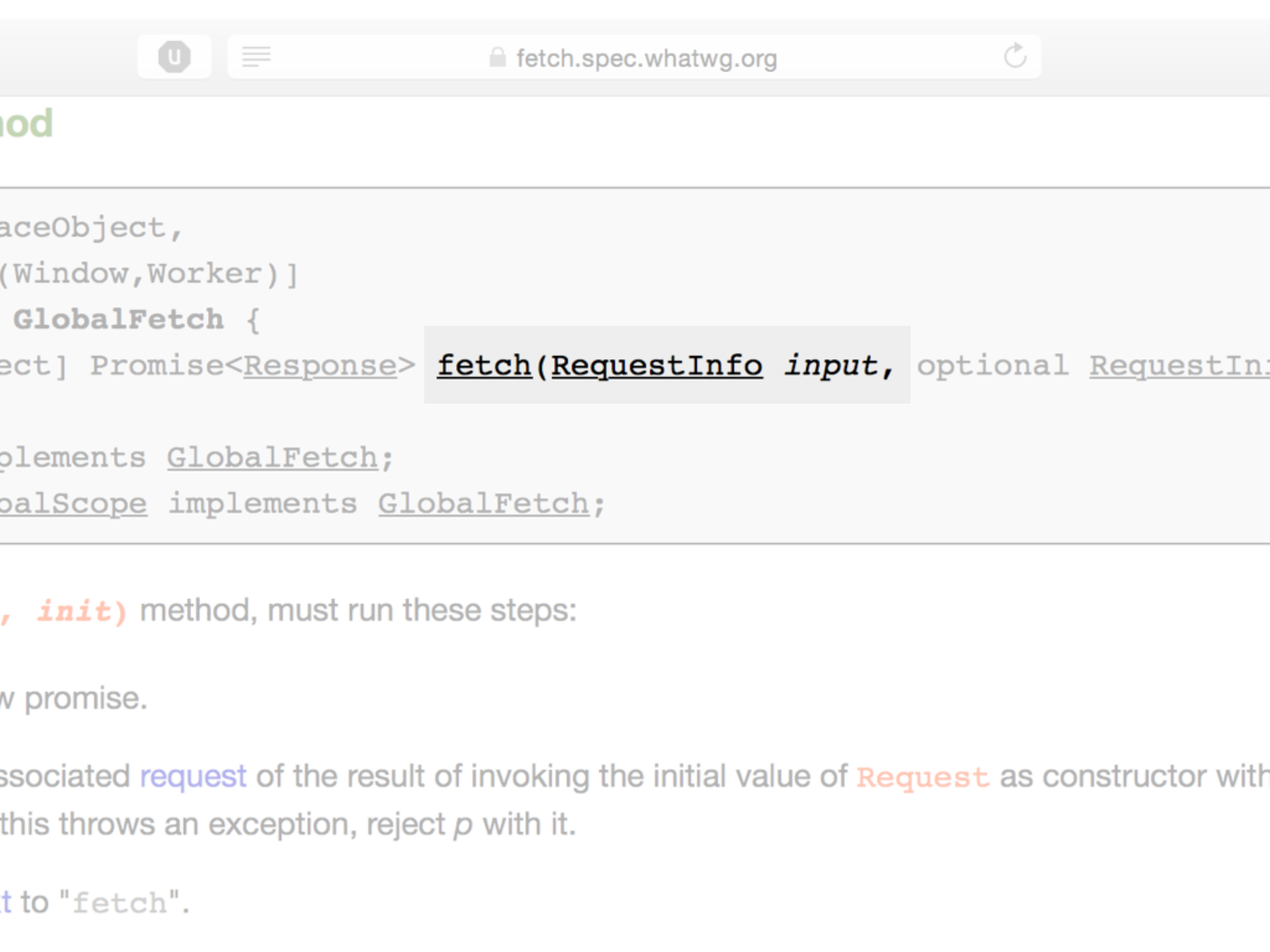
1. Let *p* be a new promise.
2. Let *r* be the associated request of the result of invoking the initial value of `Request` as constructor with *input* and *init* as arguments. If this throws an exception, reject *p* with it.
3. Set *r*'s context to "`fetch`".
4. Run the following in parallel:

   Fetch *r*.

   To process response for *response*, run these substeps:

   1. If *response*'s type is "`error`", reject *p* with a `TypeError`.
   2. Otherwise, resolve *p* with a new `Response` object associated with *response* and a new `Headers` object whose guard is "`immutable`".

   To process response body for *response*, do nothing.



hod

```
aceObject,
(Window,Worker)]
 GlobalFetch {
ect] Promise<Response> fetch(RequestInfo input, optional RequestIn

plements GlobalFetch;
balScope implements GlobalFetch;
```

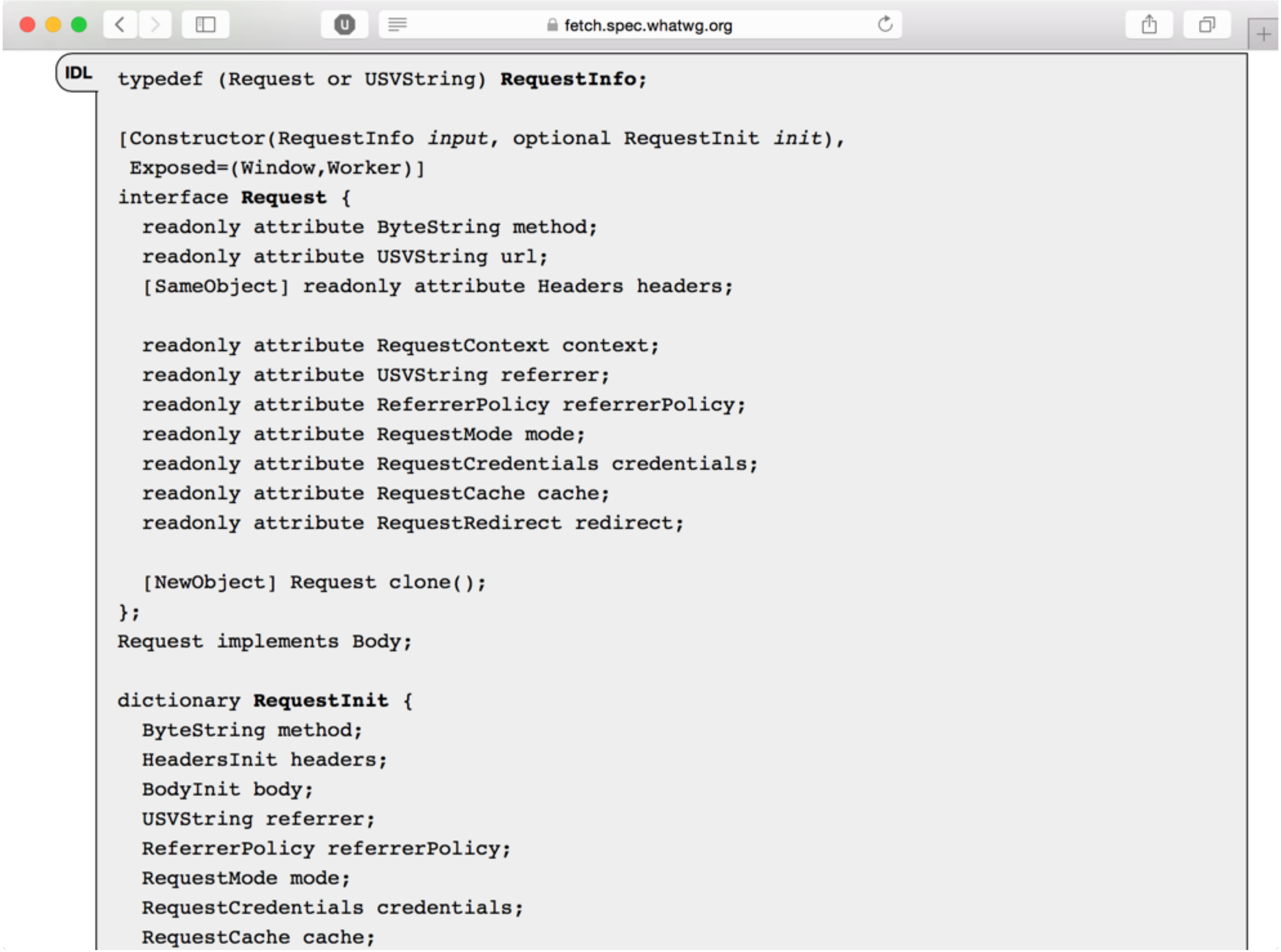
, *init*) method, must run these steps:

w promise.

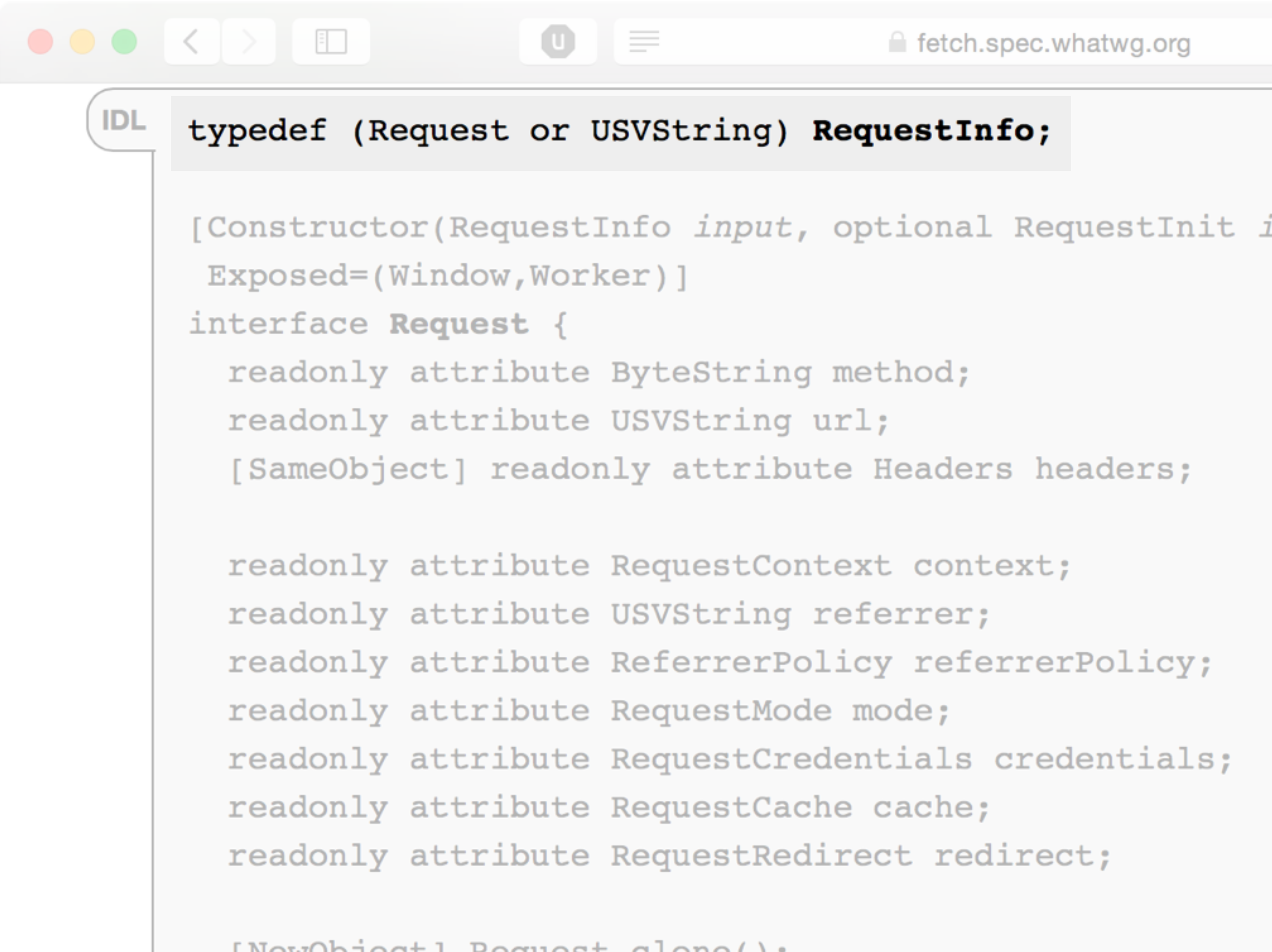
ssociated request of the result of invoking the initial value of `Request` as constructor with

this throws an exception, reject *p* with it.

t to "`fetch`".



```
typedef (Request or USVString) RequestInfo;

[Constructor(RequestInfo input, optional RequestInit init),
 Exposed=(Window,Worker)]
interface Request {
  readonly attribute ByteString method;
  readonly attribute USVString url;
  [SameObject] readonly attribute Headers headers;

  readonly attribute RequestContext context;
  readonly attribute USVString referrer;
  readonly attribute ReferrerPolicy referrerPolicy;
  readonly attribute RequestMode mode;
  readonly attribute RequestCredentials credentials;
  readonly attribute RequestCache cache;
  readonly attribute RequestRedirect redirect;

  [NewObject] Request clone();
};
Request implements Body;

dictionary RequestInit {
  ByteString method;
  HeadersInit headers;
  BodyInit body;
  USVString referrer;
  ReferrerPolicy referrerPolicy;
  RequestMode mode;
  RequestCredentials credentials;
  RequestCache cache;
```



```
typedef (Request or USVString) RequestInfo;

[Constructor(RequestInfo input, optional RequestInit i
 Exposed=(Window,Worker)]
interface Request {
  readonly attribute ByteString method;
  readonly attribute USVString url;
  [SameObject] readonly attribute Headers headers;

  readonly attribute RequestContext context;
  readonly attribute USVString referrer;
  readonly attribute ReferrerPolicy referrerPolicy;
  readonly attribute RequestMode mode;
  readonly attribute RequestCredentials credentials;
  readonly attribute RequestCache cache;
  readonly attribute RequestRedirect redirect;

  [NewObject] Request clone();
```

Ideally You Can

- Create a request object
- Reuse it

```
var req = new Request('http://blah', {
  method: 'POST',
  body: requestBody
});


fetch(req);

fetch(req);
```

But

- Body can only consumed once

```
var req = new Request('http://blah', {
  method: 'POST',
  body: requestBody
});


fetch(req);

fetch(req); // TypeError
```

Issue #61

- Fix the different behaviors
- Request will not able to reuse in the future

https://github.com/whatwg/fetch/issues/61

Polyfill

- by Github

  https://github.com/github/fetch

Node Polyfill

- node-fetch
  https://www.npmjs.com/package/node-fetch

- isomorphic-fetch
  https://www.npmjs.com/package/isomorphic-fetch

Fact

- Debug:
  Chrome, at ‘other tab’, request body not shows
  Firefox, depend on response type

Fact

- Safety flag for CORS not affect fetch

  Affect Electron app

  Use XHR + polyfill instead

Fact

- Not able to cancel a fetch
- Issue #27
- No solution now

https://github.com/whatwg/fetch/issues/27

Fact

- Not support progress now
- fetch-with-streams
- Future spec

https://github.com/yutakahirano/fetch-with-streams

Streams

- WHATWG Spec
- Not Node Stream
- Base on Promise

```
fetch(url).then(response => {
  var reader = response.body.getReader();
  var decoder = new TextDecoder();
  function drain(valueSoFar) {
    return reader.read().then(function(result) {
      valueSoFar += decoder.decode(result.value || new Uint8Ar
      if (result.done) return valueSoFar;
      return drain(valueSoFar);
    });
  }
  return drain();
}).then(function(fullText) {
  console.log(fullText);
});
```

https://github.com/whatwg/fetch/issues/28#issuecomment-87209944

```
fetch(url).then(response => {
  var reader = response.body.getReader();
  var decoder = new TextDecoder();
  function drain(valueSoFar) {
    return reader.read().then(function(result) {
      valueSoFar += decoder.decode(result.value || new Uint8Ar
      if (result.done) return valueSoFar;
      return drain(valueSoFar);
    });
  }
  return drain();
}).then(function(fullText) {
  console.log(fullText);
});
```

```
fetch(url).then(response => {
  var reader = response.body.getReader();
  var decoder = new TextDecoder();
  function drain(valueSoFar) {
    return reader.read().then(function(result) {
      valueSoFar += decoder.decode(result.value || new Uint8Ar
      if (result.done) return valueSoFar;
      return drain(valueSoFar);
    });
  }
  return drain();
}).then(function(fullText) {
  console.log(fullText);
});
```

```
fetch(url).then(response => {
  var reader = response.body.getReader();
  var decoder = new TextDecoder();
  function drain(valueSoFar) {
    return reader.read().then(function(result) {
      valueSoFar += decoder.decode(result.value || new Uint8Ar
      if (result.done) return valueSoFar;
      return drain(valueSoFar);
    });
  }
  return drain();
}).then(function(fullText) {
  console.log(fullText);
});
```

```
fetch(url).then(response => {
  var reader = response.body.getReader();
  var decoder = new TextDecoder();
  function drain(valueSoFar) {
    return reader.read().then(function(result) {
      valueSoFar += decoder.decode(result.value || new Uint8Ar
      if (result.done) return valueSoFar;
      return drain(valueSoFar);
    });
  }
  return drain();
}).then(function(fullText) {
  console.log(fullText);
});
```

```
fetch(url).then(response => {
  var reader = response.body.getReader();
  var decoder = new TextDecoder();
  function drain(valueSoFar) {
    return reader.read().then(function(result) {
      valueSoFar += decoder.decode(result.value || new Uint8Ar
      if (result.done) return valueSoFar;
      return drain(valueSoFar);
    });
  }
  return drain();
}).then(function(fullText) {
  console.log(fullText);
});
```

Current Status

- Lots of new stuff is on the way
  - Streams, cancel
  - Browser integration
  - Cache, HTTP/2, CSP, subresource integrity, GET with body, busy indicator

ES7

Async/Await

Async

- Function is asynchronous
- Can stop somewhere
- Like generator function

Await

- Stop and wait for promise result

Async/Await

```
async function init() {
  var users = await fetch('/api/users');
  var items = await fetch('/api/items');
};
```

```
fetch(url).then(response => {
  var reader = response.body.getReader();
  var decoder = new TextDecoder();
  function drain(valueSoFar) {
    return reader.read().then(function(result) {
      valueSoFar += decoder.decode(result.value || new Uint8Ar
      if (result.done) return valueSoFar;
      return drain(valueSoFar);
    });
  }
  return drain();
}).then(function(fullText) {
  console.log(fullText);
});
```

https://github.com/whatwg/fetch/issues/28#issuecomment-87209944

```
fetch(url).then(async response => {
  var reader = response.body.getReader();
  var decoder = new TextDecoder();
  var fullText = '';
  var result;

  do {
    result = await reader.read();
    fullText += decoder.decode(result.value || new Uint8Array,
  } while (!result.done);

  console.log(fullText);
});
```

https://github.com/whatwg/fetch/issues/28#issuecomment-87209944

```
fetch(url).then(async response => {
  var reader = response.body.getReader();
  var decoder = new TextDecoder();
  var fullText = '';
  var result;

  do {
    result = await reader.read();
    fullText += decoder.decode(result.value || new Uint8Array,
  } while (!result.done);

  console.log(fullText);
});
```

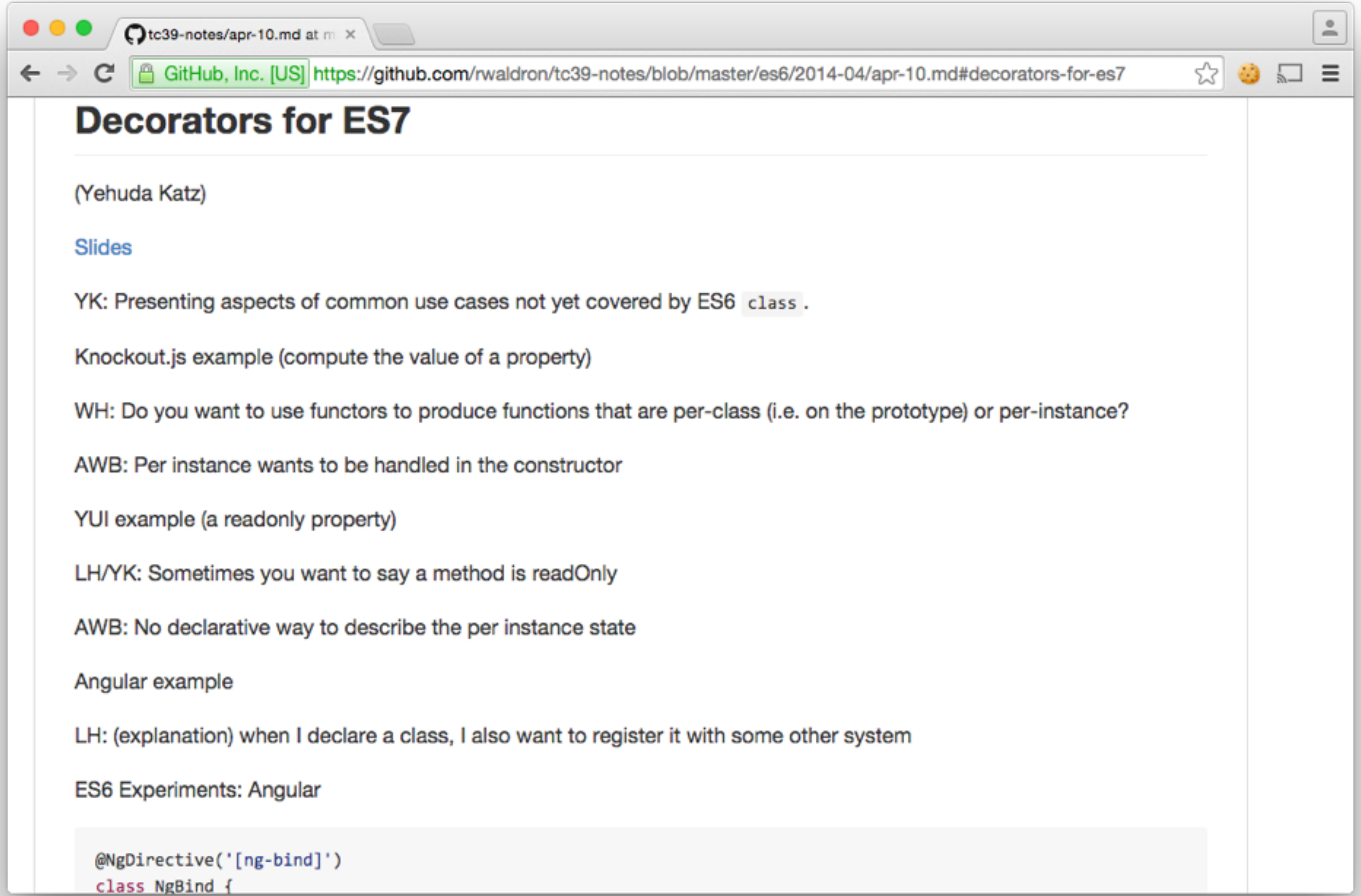
```
fetch(url).then(async response => {
  var reader = response.body.getReader();
  var decoder = new TextDecoder();
  var fullText = '';
  var result;

  do {
    result = await reader.read();
    fullText += decoder.decode(result.value || new Uint8Array,
  } while (!result.done);

  console.log(fullText);
});
```

```
fetch(url).then(async response => {
  var reader = response.body.getReader();
  var decoder = new TextDecoder();
  var fullText = '';
  var result;

  do {
    result = await reader.read();
    fullText += decoder.decode(result.value || new Uint8Array,
  } while (!result.done);

  console.log(fullText);
});
```

Decorator

Decorator

- Syntax sugar
- Looks like annotation
- Like Python decorator
- by Yehuda Katz



tc39-notes/apr-10.md at m
GitHub, Inc. [US]
https://github.com/rwaldron/tc39-notes/blob/master/es6/2014-04/apr-10.md#decorators-for-es7

Decorators for ES7

(Yehuda Katz)

Slides

YK: Presenting aspects of common use cases not yet covered by ES6 `class`.

Knockout.js example (compute the value of a property)

WH: Do you want to use functors to produce functions that are per-class (i.e. on the prototype) or per-instance?

AWB: Per instance wants to be handled in the constructor

YUI example (a readonly property)

LH/YK: Sometimes you want to say a method is readOnly

AWB: No declarative way to describe the per instance state

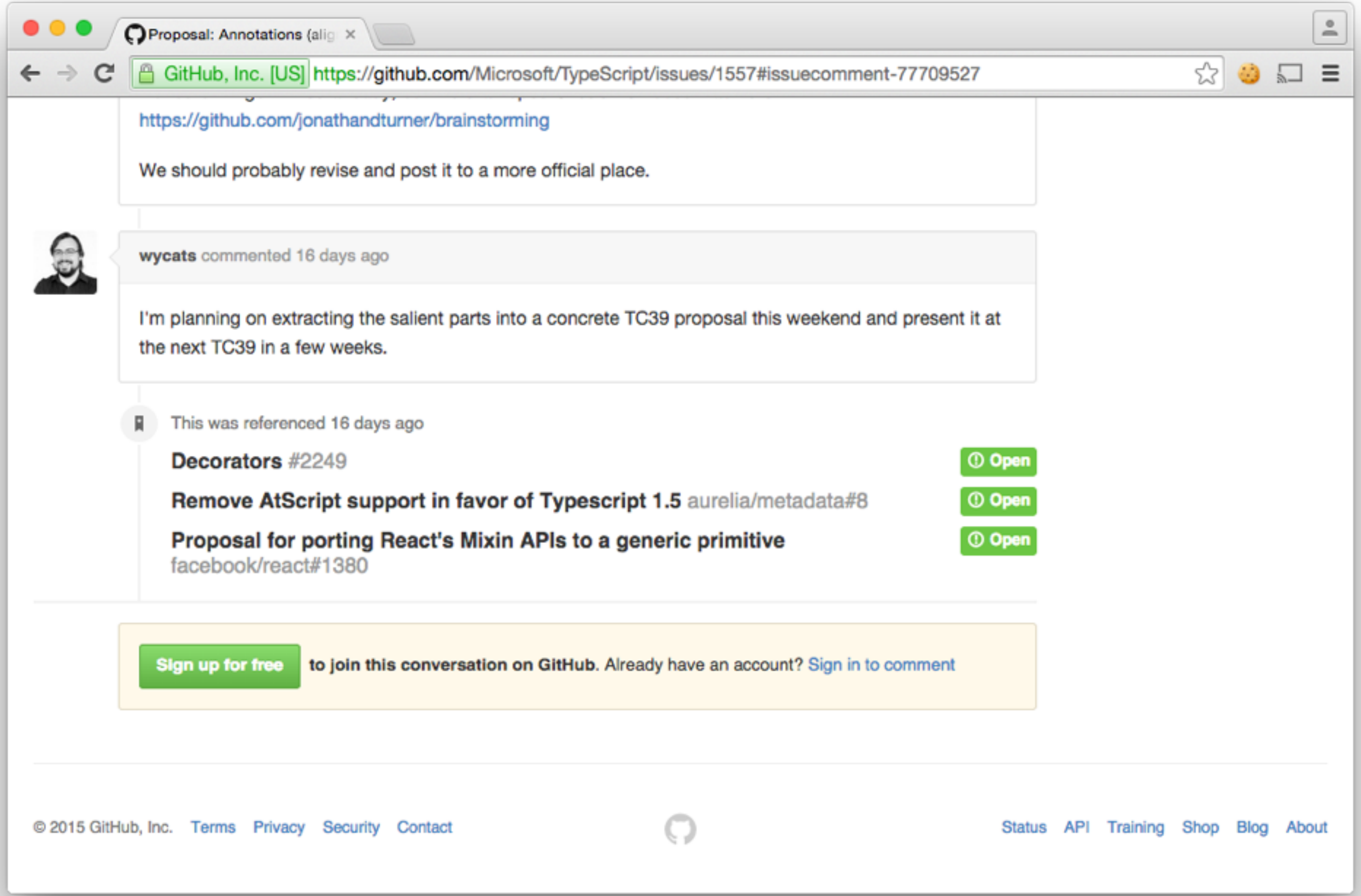
Angular example

LH: (explanation) when I declare a class, I also want to register it with some other system

ES6 Experiments: Angular

```
@NgDirective('[ng-bind]')
class NgBind {
```

https://github.com/rwaldron/tc39-notes/blob/master/es6/2014-04/apr-10.md#decorators-for-es7


Proposal: Annotations (alig ×
GitHub, Inc. [US] https://github.com/Microsoft/TypeScript/issues/1557#issuecomment-77709527

https://github.com/jonathandturner/brainstorming

We should probably revise and post it to a more official place.

wycats commented 16 days ago

I'm planning on extracting the salient parts into a concrete TC39 proposal this weekend and present it at the next TC39 in a few weeks.

This was referenced 16 days ago

Decorators #2249 — ⓘ Open

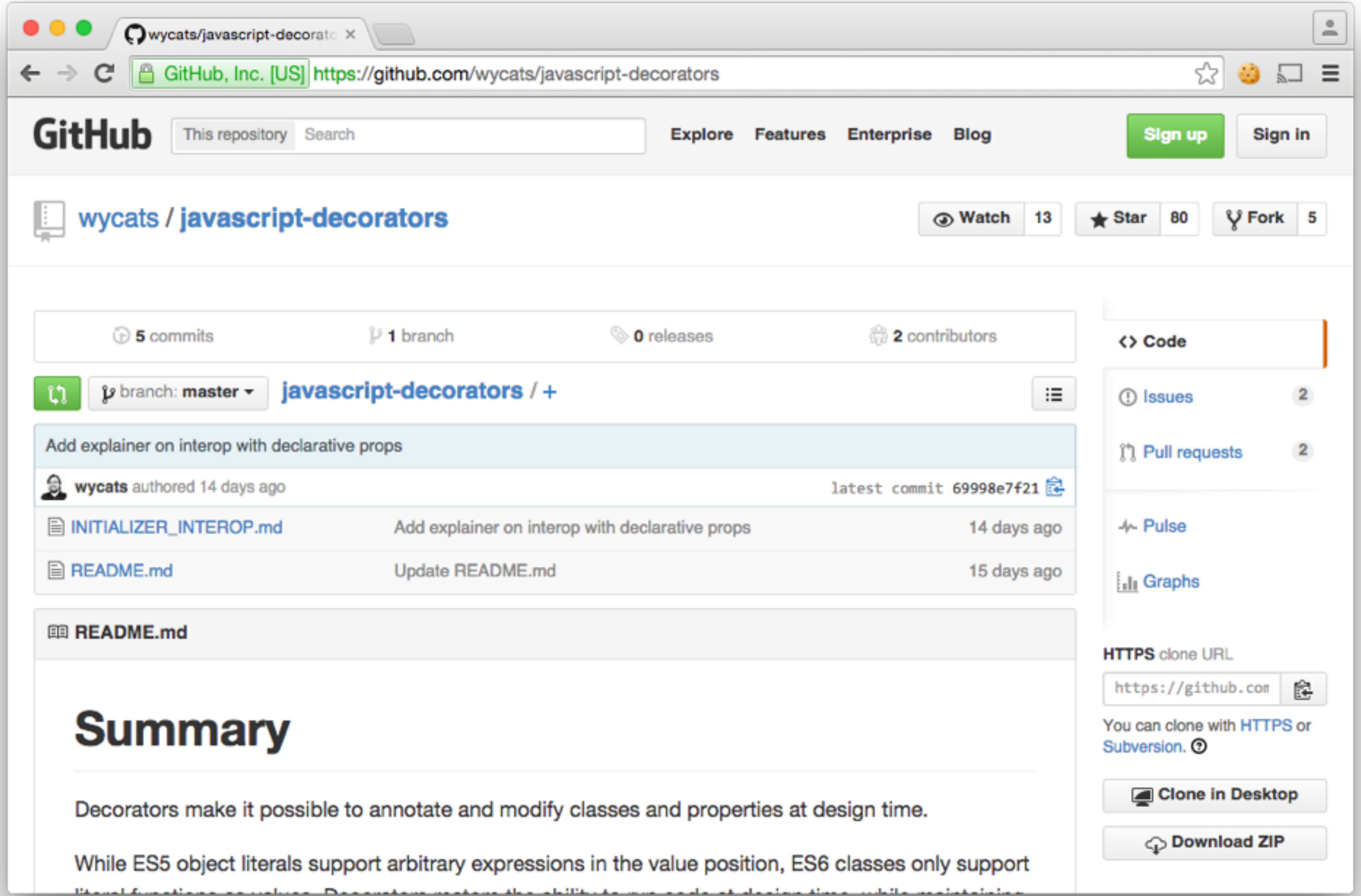
Remove AtScript support in favor of Typescript 1.5 aurelia/metadata#8 — ⓘ Open

Proposal for porting React's Mixin APIs to a generic primitive facebook/react#1380 — ⓘ Open

Sign up for free **to join this conversation on GitHub**. Already have an account? Sign in to comment

© 2015 GitHub, Inc. Terms Privacy Security Contact
Status API Training Shop Blog About

https://github.com/Microsoft/TypeScript/issues/1557#issuecomment-77709527



wycats / javascript-decorators

javascript-decorators / +

Add explainer on interop with declarative props

wycats authored 14 days ago latest commit 69998e7f21

INITIALIZER_INTEROP.md	Add explainer on interop with declarative props	14 days ago
README.md	Update README.md	15 days ago

You can clone with HTTPS or Subversion.

README.md

Summary

Decorators make it possible to annotate and modify classes and properties at design time.

While ES5 object literals support arbitrary expressions in the value position, ES6 classes only support literal functions as values. Decorators restore the ability to run code at design time, while maintaining

https://github.com/wycats/javascript-decorators

```
class M {
  @memorize
  fib(n) {
    if (n < 2) { return n }
    return this.fib(n - 1)
         + this.fib(n - 2)
  }
}
```

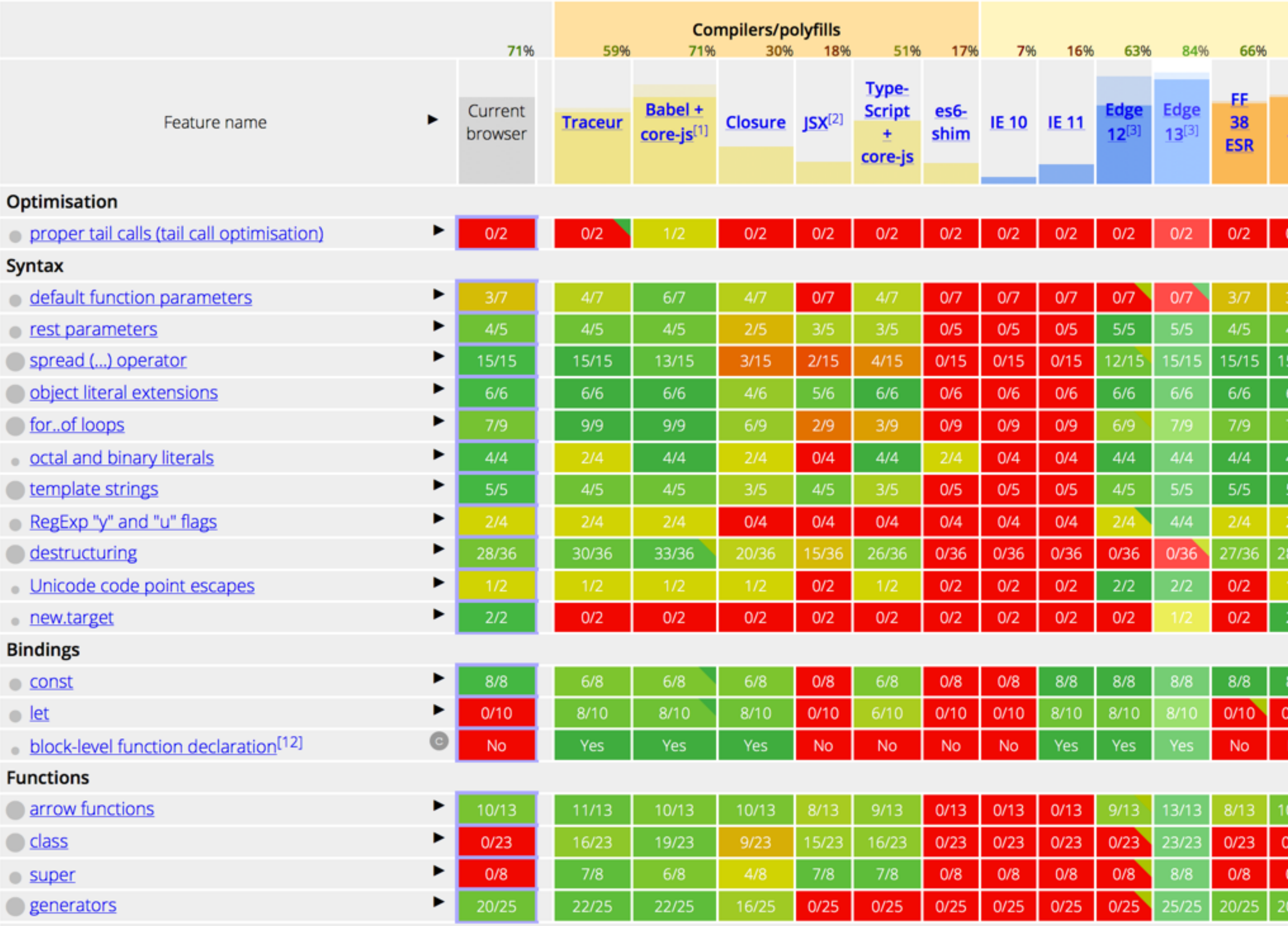
```
class M {
  @memorize
  fib(n) {
    if (n < 2) { return n }
    return this.fib(n - 1)
         + this.fib(n - 2)
  }
}
```

```
var M = (function () {
  class M {
    fib(n) {
      if (n < 2) { return n }
      return this.fib(n - 1) + this.fib(n - 2)
    }
  }

  var _temp
  _temp = memorize(Foo.prototype, "fib") || _temp
  if (_temp) Object.defineProperty(M.prototype, "fib", _te
  return M
})()
```

```
function memorize(target, name, descriptor) {
  let getter = descriptor.get, setter = descriptor.set;

  descriptor.get = function() {
    let table = memorizationFor(this);
    if (name in table) { return table[name]; }
    return table[name] = getter.call(this);
  }

  descriptor.set = function(val) {
    let table = memorizationFor(this);
    setter.call(this, val);
    table[name] = val;
  }

  return descriptor;
}
```

https://github.com/wycats/javascript-decorators

Use ES6 today



		Compilers/polyfills										
	71%	59%	71%	30%	18%	51%	17%	7%	16%	63%	84%	66%
Feature name	Current browser	Traceur	Babel + core-js[1]	Closure	JSX[2]	Type-Script + core-js	es6-shim	IE 10	IE 11	Edge 12[3]	Edge 13[3]	FF 38 ESR
Optimisation												
proper tail calls (tail call optimisation)	0/2	0/2	1/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
Syntax												
default function parameters	3/7	4/7	6/7	4/7	0/7	4/7	0/7	0/7	0/7	0/7	0/7	3/7
rest parameters	4/5	4/5	4/5	2/5	3/5	3/5	0/5	0/5	0/5	5/5	5/5	4/5
spread (...) operator	15/15	15/15	13/15	3/15	2/15	4/15	0/15	0/15	0/15	12/15	15/15	15/15
object literal extensions	6/6	6/6	6/6	4/6	5/6	6/6	0/6	0/6	0/6	6/6	6/6	6/6
for..of loops	7/9	9/9	9/9	6/9	2/9	3/9	0/9	0/9	0/9	6/9	7/9	7/9
octal and binary literals	4/4	2/4	4/4	2/4	0/4	4/4	2/4	0/4	0/4	4/4	4/4	4/4
template strings	5/5	4/5	4/5	3/5	4/5	3/5	0/5	0/5	0/5	4/5	5/5	5/5
RegExp "y" and "u" flags	2/4	2/4	2/4	0/4	0/4	0/4	0/4	0/4	0/4	2/4	4/4	2/4
destructuring	28/36	30/36	33/36	20/36	15/36	26/36	0/36	0/36	0/36	0/36	0/36	27/36
Unicode code point escapes	1/2	1/2	1/2	1/2	0/2	1/2	0/2	0/2	0/2	2/2	2/2	0/2
new.target	2/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	1/2	0/2
Bindings												
const	8/8	6/8	6/8	6/8	0/8	6/8	0/8	0/8	8/8	8/8	8/8	8/8
let	0/10	8/10	8/10	8/10	0/10	6/10	0/10	0/10	8/10	8/10	8/10	0/10
block-level function declaration[12]	No	Yes	Yes	Yes	No	No	No	No	Yes	Yes	Yes	No
Functions												
arrow functions	10/13	11/13	10/13	10/13	8/13	9/13	0/13	0/13	0/13	9/13	13/13	8/13
class	0/23	16/23	19/23	9/23	15/23	16/23	0/23	0/23	0/23	0/23	23/23	0/23
super	0/8	7/8	6/8	4/8	7/8	7/8	0/8	0/8	0/8	0/8	8/8	0/8
generators	20/25	22/25	22/25	16/25	0/25	0/25	0/25	0/25	0/25	0/25	25/25	20/25

Current Implements

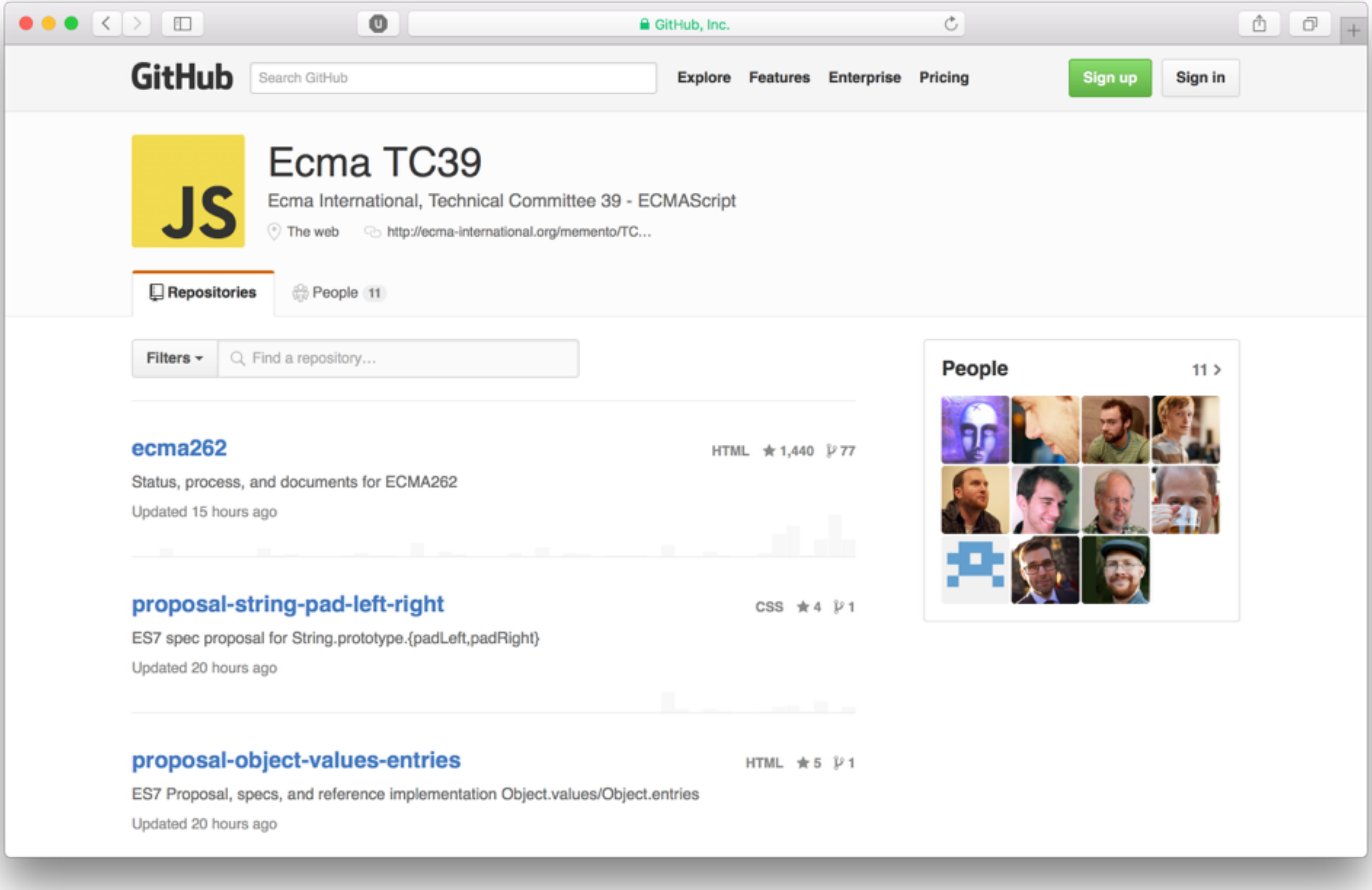
- https://kangax.github.io/compat-table/es6/
- Only partial support on major browsers
- No module supports
- Bad performance

Babel + Webpack

TC39

TC39

- Tech Committee of ECMAScript



Ecma TC39

Ecma International, Technical Committee 39 - ECMAScript

The web http://ecma-international.org/memento/TC...

ecma262 HTML ★ 1,440 77

Status, process, and documents for ECMA262

Updated 15 hours ago

proposal-string-pad-left-right CSS ★ 4 1

ES7 spec proposal for String.prototype.{padLeft,padRight}

Updated 20 hours ago

proposal-object-values-entries HTML ★ 5 1

ES7 Proposal, specs, and reference implementation Object.values/Object.entries

Updated 20 hours ago