02

OPEN ORIENTED

凹 凸 实 验 室

了解 Fetch API

技术背景

Fetch API 是近年来被提及将要取代 XMLHttpRequest(XHR) 的技术新标准，是一个 HTML5 的 API。

Fetch 并不是 XHR 的升级版本，而是从一个全新的角度来思考的一种设计。Fetch 是基于 Promise 语法结构，而且它的设计足够低阶，这表示它可以在实际需求中进行更多的弹性设计。对于 XHR 所提供的能力来说，Fetch 已经足够取代 XHR ，并且提供了更多拓展的可能性。

技术对比

XMLHttpRequest 是一个设计粗糙的 API，不符合关注分离（Separation of Concerns）的原则。XHR 是基于事件的异步模型，在设计上将配置、输入、输出和事件监听混杂在一个对象里，必须通过创建实例的方式来发请求，而且基于事件的异步模型写起来没有现代的 Promise，generator/yield，async/await 优雅。

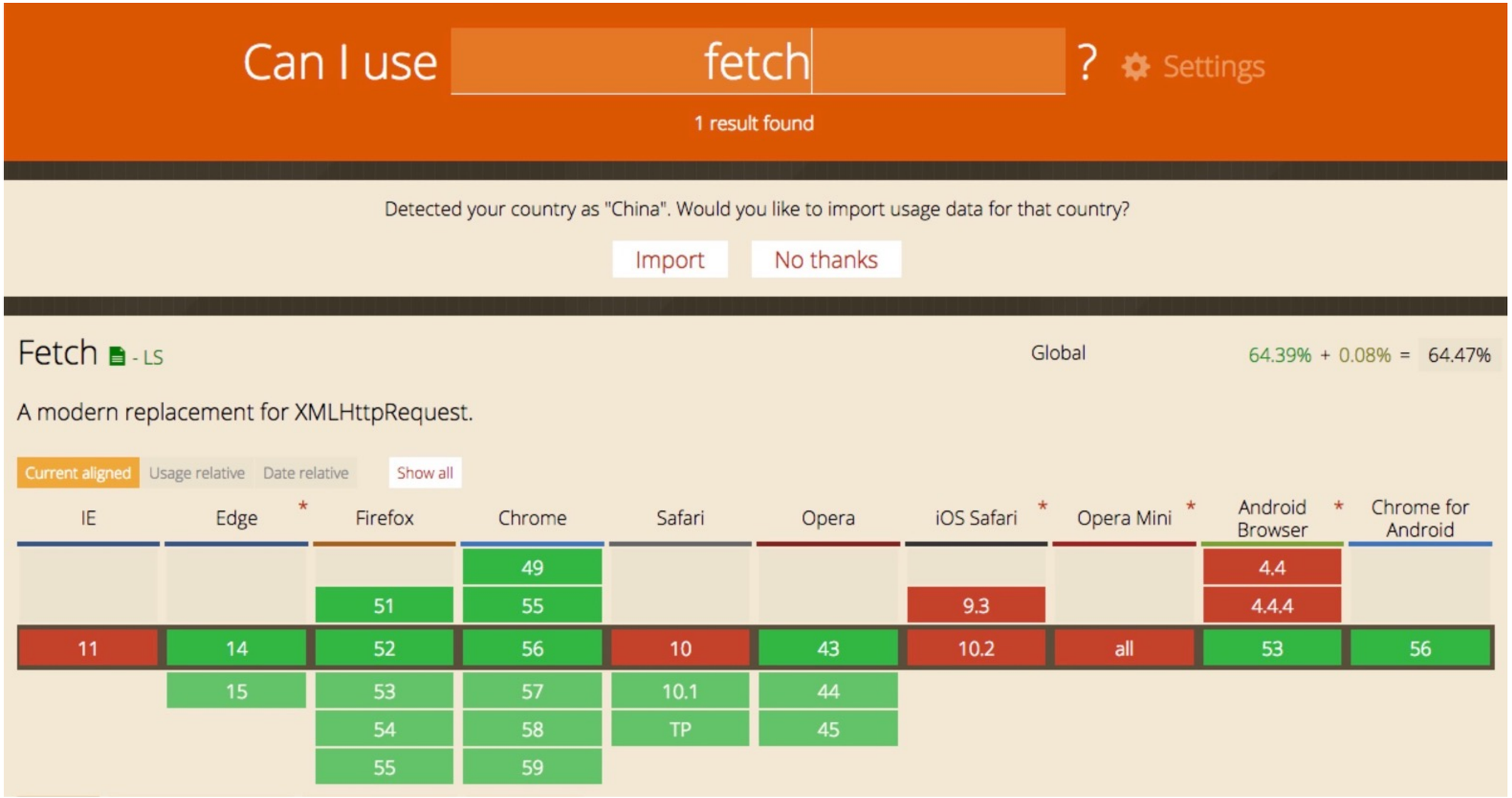
而 Fetch API 设计低阶简单，会返回一个 Promise 对象。

技术对比


```
// 使用 XHR
var xhr = new XMLHttpRequest();
xhr.open('GET', url);
xhr.responseType = 'json';
xhr.onload = function() {
  console.log(xhr.response);
};
xhr.onerror = function() {
  console.log('some thing error');
};
xhr.send();

// 使用 Fetch API
fetch(url).then(function(response) {
  return response.json();
}).then(function(data) {
  console.log(data);
}).catch(function() {
  console.log('some thing error');
});
```

兼容性


Can I use fetch ? Settings
1 result found
Detected your country as "China". Would you like to import usage data for that country?
Import
No thanks
Fetch - LS
Global
64.39% + 0.08% = 64.47%
A modern replacement for XMLHttpRequest.
Current aligned
Usage relative
Date relative
Show all
IE
Edge *
Firefox
Chrome
Safari
Opera
iOS Safari *
Opera Mini *
Android Browser *
Chrome for Android
49
51
55
9.3
4.4
4.4.4
11
14
52
56
10
43
10.2
all
53
56
15
53
57
10.1
44
54
58
TP
45
55
59

兼容性

可以看到现阶段的兼容性情况并不理想，如果我们想在项目中运用 Fetch API，那么我们需要引入一些 polyfill。

可以使用 https://github.com/github/fetch 提供的这个 polyfill 。原理是通过功能检测，Fetch API 的支持情况，可以通过检测 Headers、Request、Response 或 fetch() 是否在 Window 或 Worker 域中，如果不存在，则通过 XHR 进行模拟。

有了 Fetch API 的 polyfill 支持，我们就能愉快地使用 Fetch API 了。

如何使用

上面我们在谈到 Fetch API 与 XHR 的区别时，简单的提到了 Fetch API 的使用。

```
// 使用 Fetch API
fetch(url)
  .then(function(response) {
    return response.json();
  })
  .then(function(data) {
    console.log(data);
  })
  .catch(function() {
    console.log('some thing error');
  });
```

如何使用


```
// 使用 Fetch API (es6 写法)
fetch(url).then(res => res.json())
  .then(data => console.log(data))
  .catch(err => console.log('some thing error'))

// 使用 Fetch API (es6 async/await)
(async() => {
  try {
    const response = await fetch(url);
    const data = await response.json();
    console.log(data);
  } catch (err) {
    console.log('some thing error');
  }
})();
```

如何使用


```
fetch(input, init)
  .then(function(response) { ... })
  .catch(function(err) { ... });
```

fetch 执行后返回一个 Promise 对象。

input：要获取的资源，可由一个url字符串和一个Request对象组成

init: （可选）请求资源的配置项，可配置 请求方式，请求头，请求Body，请求模式，请求的credentials，请求的缓存模式等。

如何使用


```
fetch('/submit.php', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/x-www-form-urlencoded'
  },
  body: 'firstName=lee&favColor=blue&password=123'
}).then(function(res) {
  if (res.ok) {
    alert('验证通过');
  } else if (res.status == 401) {
    alert('验证失败，您没有权限');
  }
}, function(e) {
  alert('some thing error');
});
```

小提示

- 默认不带 cookie，需要配置，在 fetch 的 option 中设置 credentials 属性值为 "same-origin" 或者 "include"，前者表示在同域的请求中都会带上 cookie，后者表示在所有的请求中，都会带上cookie。

- 服务器返回 200 ～ 999 并不会被 catch 到，需要自己手动去判断请求是否是成功。只有网络错误导致请求失败才会被 catch，执行 reject。

未来

现阶段的 Fetch API 想要取代 XHR仍有需要改进之处，为了支持流（Stream），Fetch API 需要提供可中断资源读取的能力，和提供可读取进度的 API。这些是 XHR 已具备的能力。

THANKS

FOR YOUR WATCHING

02

OPEN ORIENTED

凹 凸 实 验 室