Web API简介

Overview: Client-side Web API

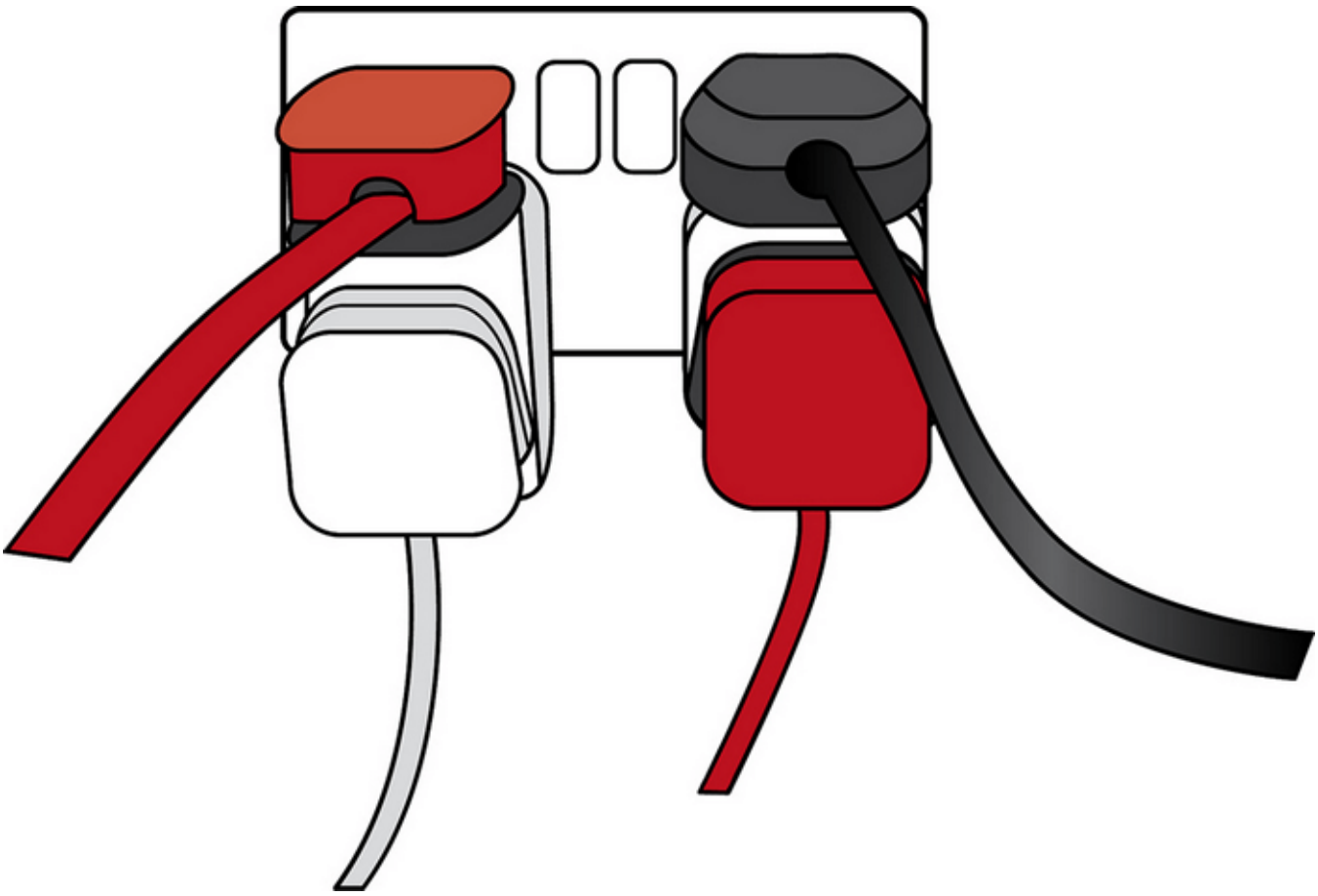
首先，我们将从一个高层次看看API - 它们是什么；他们如何工作；如何在代码中使用它们，以及它们是如何组织的。我们也将看看不同主要类别的API以及它们的用途。

预备知识	基本计算机知识，对于HTML和CSS的基本理解（见 JavaScript 第一步，创建JavaScript代码块，JavaScript 对象入门）。
目标	熟悉API，他们可以做什么，以及如何在代码中使用它们。

什么是API?

应用程序接口（API）是基于编程语言构建的结构，使开发人员更容易地创建复杂的功能。它们抽象了复杂的代码，并提供一些简单的接口规则直接使用。

来看一个现实中的例子：想想您的房子、公寓或其他住宅的供电方式，如果您想在您的房子里用电，只要把电器的插头插入插座就可以，而不是直接把它连接到电线上——这样做非常低效，而且对于不是电工的人会是困难和危险的。



图片来自：Overloaded plug socket 提供者： The Clear Communication People于Flickr。

同样，比如说，编程来显示一些3D图形，使用以更高级语言编写的API（例如JavaScript或Python）将会比直接编写直接控制计算机的GPU或其他图形功能的低级代码（比如C或C++）来执行操作要容易得多。

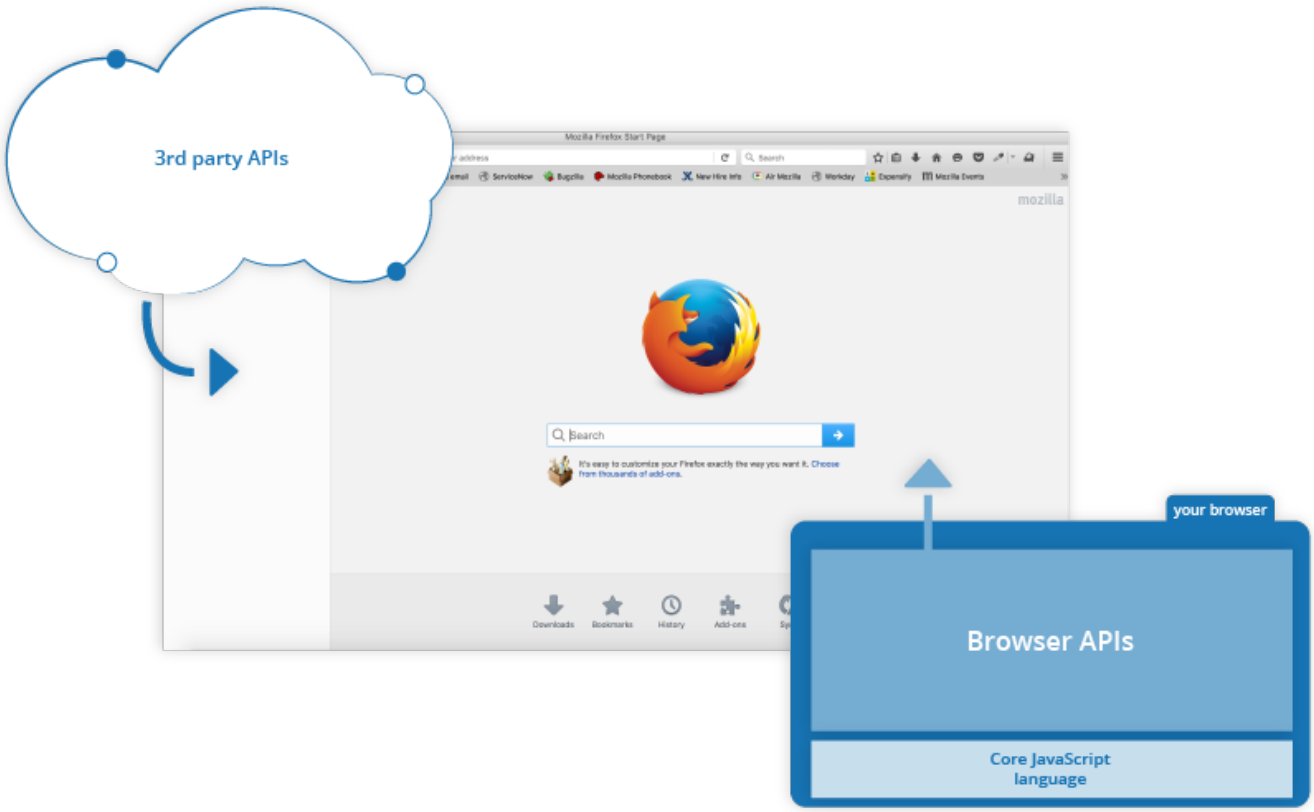
> 注：详细说明请见API - Glossary。

客户端JavaScript中的API

客户端JavaScript中有很多可用的API — 他们本身并不是JavaScript语言的一部分，却建立在JavaScript语言核心的顶部，为使用JavaScript代码提供额外的超强能力。他们通常分为两类：

- **浏览器API**内置于Web浏览器中，能从浏览器和电脑周边环境中提取数据，并用来做有用的复杂的事情 。例如Geolocation API提供了一些简单的JavaScript结构以获得位置数据，因此您可以在Google地图上标示您的位置。在后台，浏览器确实使用一些复杂的低级代码（例如C++）与设备的GPS硬件（或可以决定位置数据的任何设施）通信来获取位置数据并把这些数据返回给您的代码中使用浏览器环境；但是，这种复杂性通过API抽象出来，因而与您无关。
- **第三方API**缺省情况下不会内置于浏览器中，通常必须在Web中的某个地方获取代码和信息。例如Twitter API 使您能做一些显示最新推文这样的事情，它提供一系列特殊的结构，

可以用来请求Twitter服务并返回特殊的信息。



JavaScript，API和其他JavaScript工具之间的关系

如上所述，我们讨论了什么是客户端JavaScript API，以及它们与JavaScript语言的关系。让我们回顾一下，使其更清晰，并提及其他JavaScript工具的适用位置：

- JavaScript — 一种内置于浏览器的高级脚本语言，您可以用来实现Web页面/应用中的功能。注意JavaScript也可用于其他象Node这样的的编程环境。但现在您不必考虑这些。
- 客户端API — 内置于浏览器的结构程序，位于JavaScript语言顶部，使您可以更容易的实现功能。
- 第三方API — 置于第三方普通的结构程序（例如Twitter，Facebook），使您可以在自己的Web页面中使用那些平台的某些功能（例如在您的Web页面显示最新的Tweets）。
- JavaScript库 — 通常是包含具有特定功能的一个或多个JavaScript文件，把这些文件关联到您的Web页以快速或授权编写常见的功能。例如包含jQuery和Mootools
- JavaScript框架 — 从库开始的下一步，JavaScript框架视图把HTML、CSS、JavaScript和其他安装的技术打包在一起，然后用来从头编写一个完整的Web应用。

API可以做什么？

在主流浏览器中有大量的可用API，您可以在代码中做许多的事情，对此可以查看MDN API index page。

常见浏览器API

特别地，您将使用的最常见的浏览器API类别（以及我们将更详细地介绍的）是：

- **操作文档的API**内置于浏览器中。最明显的例子是DOM（文档对象模型）API，它允许您操作HTML和CSS — 创建、移除以及修改HTML，动态地将新样式应用到您的页面，等等。每当您看到一个弹出窗口出现在一个页面上，或者显示一些新的内容时，这都是DOM的行为。 您可以在在Manipulating documents中找到关于这些类型的API的更多信息。
- **从服务器获取数据的API** 用于更新网页的一小部分是相当好用的。这个看似很小的细节能对网站的性能和行为产生巨大的影响 — 如果您只是更新一个股票列表或者一些可用的新故事而不需要从服务器重新加载整个页面将使网站或应用程序感觉更加敏感和“活泼”。使这成为可能的API包括`XMLHttpRequest`和Fetch API。您也可能会遇到描述这种技术的术语**Ajax**。您可以在Fetching data from the server找到关于类似的API的更多信息。
- **用于绘制和操作图形的API**目前已被浏览器广泛支持 — 最流行的是允许您以编程方式更新包含在HTML `<canvas>` 元素中的像素数据以创建2D和3D场景的Canvas和WebGL。例如，您可以绘制矩形或圆形等形状，将图像导入到画布上，然后使用Canvas API对其应用滤镜（如棕褐色滤镜或灰度滤镜），或使用WebGL创建具有光照和纹理的复杂3D场景。这些API经常与用于创建动画循环的API（例如`window.requestAnimationFrame()`）和其他API一起不断更新诸如动画和游戏之类的场景。
- **音频和视频API**例如`HTMLMediaElement`，Web Audio API和WebRTC允许您使用多媒体来做一些非常有趣的事情，比如创建用于播放音频和视频的自定义UI控件，显示字幕字幕和您的视频，从网络摄像机抓取视频，通过画布操纵（见上），或在网络会议中显示在别人的电脑上，或者添加效果到音轨（如增益，失真，平移等） 。
- **设备API**基本上是以对网络应用程序有用的方式操作和检索现代设备硬件中的数据的API。我们已经讨论过访问设备位置数据的地理定位API，因此您可以在地图上标注您的位置。其他示例还包括通过系统通知（参见Notifications API）或振动硬件（参见Vibration API）告诉用户Web应用程序有用的更新可用。
- **客户端存储API**在Web浏览器中的使用变得越来越普遍 - 如果您想创建一个应用程序来保存页面加载之间的状态，甚至让设备在处于脱机状态时可用，那么在客户端存储数据将会是非常有用的。例如使用Web Storage API的简单的键 - 值存储以及使用IndexedDB API的更复杂的表格数据存储。

常见第三方API

第三方API种类繁多; 下列是一些比较流行的你可能迟早会用到的第三方API:

- The Twitter API, 允许您在您的网站上展示您最近的推文等。
- The Google Maps API 允许你在网页上对地图进行很多操作（这很有趣，它也是Google地图的驱动器）。现在它是一整套完整的，能够胜任广泛任务的API。其能力已经被Google Maps API Picker见证。
- The Facebook suite of API 允许你将很多Facebook生态系统中的功能应用到你的app，使之受益，比如说它提供了通过Facebook账户登录、接受应用内支付、推送有针对性的广告活动等功能。
- The YouTube API, 允许你将Youtube上的视频嵌入到网站中去，同时提供搜索Youtube，创建播放列表等众多功能。
- The Twilio API, 其为您的app提供了针对语音通话和视频聊天的框架，以及从您的app发送短信息或多媒体信息等诸多功能。

> 注: 你可以在 Programmable Web API directory.上发现更多关于第三方API的信息。

API如何工作？

不同的JavaScript API以稍微不同的方式工作，但通常它们具有共同的特征和相似的主题。

它们是基于对象的

API使用一个或多个 JavaScript objects 在您的代码中进行交互，这些对象用作API使用的数据（包含在对象属性中）的容器以及API提供的功能（包含在对象方法中）。

> 注意：如果您不熟悉对象如何工作，则应继续学习 JavaScript objects 模块。

让我们回到Geolocation API的例子 - 这是一个非常简单的API，由几个简单的对象组成：

- `Geolocation`, 其中包含三种控制地理数据检索的方法
- `Position`, 表示在给定的时间的相关设备的位置。 — 它包含一个当前位置的 `Coordinates` 对象。还包含了一个时间戳,这个时间戳表示获取到位置的时间。
- `Coordinates`, 其中包含有关设备位置的大量有用数据，包括经纬度，高度，运动速度和运动方向等。

```
navigator.geolocation.getCurrentPosition(function(position) {
  var latlng = new google.maps.LatLng(position.coords.latitude,positio
  var myOptions = {
    zoom: 8,
    center: latlng,
    mapTypeId: google.maps.MapTypeId.TERRAIN,
    disableDefaultUI: true
  }
  var map = new google.maps.Map(document.querySelector("#map_canvas"),
});
```

> **Note**: 当您第一次加载上述实例，应当出现一个对话框询问您是否乐意对此应用共享位置信息（参见 They have additional security mechanisms where appropriate 这一稍后将会提到的部分）。 您需要同意这项询问以将您的位置于地图上绘制。如果您始终无法看见地图，您可能需要手动修改许可项。修改许可项的方法取决于您使用何种浏览器，对于Firefox浏览器来说，在页面信息 > 权限 中修改位置权限，在Chrome浏览器中则进入 设置 > 隐私 > 显示高级设置 > 内容设置，其后修改位置设定。

我们首先要使用 `Geolocation.getCurrentPosition()` 方法返回设备的当前位置。浏览器的 `Geolocation` 对象通过调用 `Navigator.geolocation` 属性来访问.

```
navigator.geolocation.getCurrentPosition(function(position) { ... });
```

这相当于做同样的事情

```
var myGeo = navigator.geolocation;
myGeo.getCurrentPosition(function(position) { ... });
```

但是我们可以使用 "点运算符" 将我们的属性和方法的访问链接在一起，减少了我们必须写的行数。

`Geolocation.getCurrentPosition()` 方法只有一个必须的参数，这个参数是一个匿名函数，当设备的当前位置被成功取到时，这个函数会运行。 这个函数本身有一个参数，它包含一个表示当前位置数据的 `Position` 对象。

> 注意：由另一个函数作为参数的函数称为 (callback function "回调函数").

仅在操作完成时调用函数的模式在JavaScript API中非常常见 - 确保一个操作已经完成，然后在另一个操作中尝试使用该操作返回的数据。这些被称为 **asynchronous** “异步”操作。由于获取设备的当前位置依赖于外部组件（设备的GPS或其他地理定位硬件）， 我们不能保证会立即使用返回的数据。 因此，这样子是行不通的：

```
var position = navigator.geolocation.getCurrentPosition();
var myLatitude = position.coords.latitude;
```

如果第一行还没有返回结果，则第二行将会出现错误，因为位置数据还不可用。 出于这个原因，涉及同步操作的API被设计为使用 callback functions “回调函数”，或更现代的 Promises 系统，这些系统在ECMAScript 6中可用，并被广泛用于较新的API。

我们将Geolocation API与第三方API（Google Maps API）相结合， — 我们正在使用它来绘制Google地图上由 `getCurrentPosition()` 返回的位置。 我们通过链接到页面上使这个API可用。 — 你会在HTML中找到这一行：

```
<script type="text/javascript" src="https://maps.google.com/maps/API/j
```

要使用该API, 我们首先使用`google.maps.LatLng()`构造函数创建一个`LatLng`对象实例， 该构造函数需要我们的地理定位 `Coordinates.latitude` 和 `Coordinates.longitude`值作为参数：

```
var latlng = new google.maps.LatLng(position.coords.latitude,position.
```

该对象实例被设置为 `myOptions`对象的`center`属性的值。然后我们通过调用 `google.maps.Map()`构造函数创建一个对象实例来表示我们的地图， 并传递它两个参数 — 一个参数是我们要渲染地图的 `<div>` 元素的引用 (ID为 `map_canvas`), 以及另一个参数是我们在上面定义的`myOptions`对象

```
var myOptions = {
  zoom: 8,
  center: latlng,
  mapTypeId: google.maps.MapTypeId.TERRAIN,
  disableDefaultUI: true
}

var map = new google.maps.Map(document.querySelector("#map_canvas"), m
```

这样做一来，我们的地图呈现了。

最后一块代码突出显示了您将在许多API中看到的两种常见模式。 首先，API对象通常包含构造函数，可以调用这些构造函数来创建用于编写程序的对象的实例。 其次，API对象通常有几个可用的options(如上面的`myOptions`对象)，可以调整以获得您的程序所需的确切环境(根据不同的环境,编写不同的`Options`对象)。 API构造函数通常接受options对象作为参数，这是您设置这些options的地方。

> 注意：如果您不能立即理解这个例子的细节，请不要担心。 我们将在未来的文章中详细介绍第三方**API**。

它们有可识别的入口点

使用API时，应确保知道API入口点的位置。 在Geolocation API中，这非常简单 - 它是`Navigator.geolocation`属性, 它返回浏览器的`Geolocation`对象，所有有用的地理定位方法都可用。

文档对象模型 (DOM) API有一个更简单的入口点 —它的功能往往被发现挂在`Document`对象, 或任何你想影响的HTML元素的实例，例如：

```
var em = document.createElement('em'); // create a new em element
var para = document.querySelector('p'); // reference an existing p ele
em.textContent = 'Hello there!'; // give em some text content
para.appendChild(em); // embed em inside para
```

其他API具有稍微复杂的入口点，通常涉及为要编写的API代码创建特定的上下文。例如，Canvas API的上下文对象是通过获取要绘制的`<canvas>`元素的引用来创建的，然后调用它的`HTMLCanvasElement.getContext()`方法：

```
var canvas = document.querySelector('canvas');
var ctx = canvas.getContext('2d');
```

然后，我们想通过调用内容对象 (它是`CanvasRenderingContext2D`的一个实例)的属性和方法来实现我们想要对画布进行的任何操作, 例如：

```
Ball.prototype.draw = function() {
  ctx.beginPath();
  ctx.fillStyle = this.color;
  ctx.arc(this.x, this.y, this.size, 0, 2 * Math.PI);
  ctx.fill();
};
```

注意：您可以在我们的弹跳球演示中看到此代码的实际 运行情况 （ 也可以参阅它 现场运行）。

它们使用事件来处理状态的变化

我们之前已经在课程中讨论了事件，在我们的 事件介绍文章中 - 详细介绍了客户端Web事件是什么以及它们在代码中的用法。如果您还不熟悉客户端Web API事件的工作方式，则应继续阅读。

一些Web API不包含事件，但有些包含一些事件。当事件触发时，允许我们运行函数的处理程序属性通常在单独的 “Event handlers”(事件处理程序) 部分的参考资料中列出。作为一个简单的例子，`XMLHttpRequest` 对象的实例 （每一个实例都代表一个到服务器的HTTP请求,来取得某种新的资源）都有很多事件可用，例如 `onload` 事件在成功返回时就触发包含请求的资源，并且现在就可用。

下面的代码提供了一个简单的例子来说明如何使用它：

```
var requestURL = 'https://mdn.github.io/learning-area/javascript/oojs/
var request = new XMLHttpRequest();
request.open('GET', requestURL);
request.responseType = 'json';
request.send();

request.onload = function() {
  var superHeroes = request.response;
  populateHeader(superHeroes);
  showHeroes(superHeroes);
}
```

注意：您可以在我们的ajax.html示例中看到此代码 (或者 在线运行版本 see it live also).

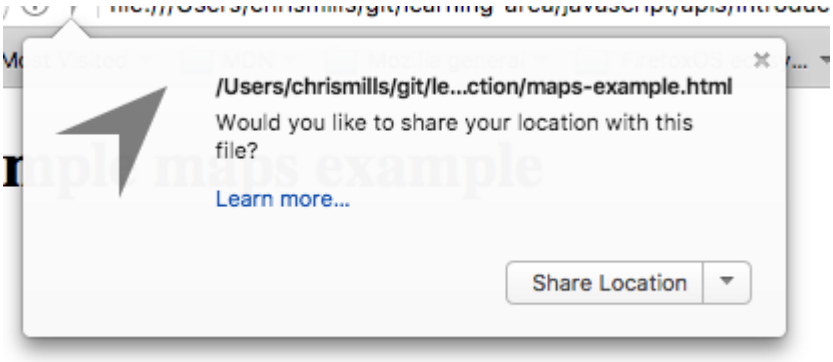
前五行指定了我们要获取的资源的位置，使用`XMLHttpRequest()`构造函数创建请求对象的新实例 ，打开HTTP 的 `GET` 请求以取得指定资源，指定响应以JSON格式发送，然后发送请求。

然后 `onload` 处理函数指定我们如何处理响应。 我们知道请求会成功返回，并在需要加载事件（如`onload` 事件）之后可用（除非发生错误），所以我们将包含返回的JSON的响应保存在`superHeroes`变量中，然后将其传递给两个不同的函数以供进一步处理。

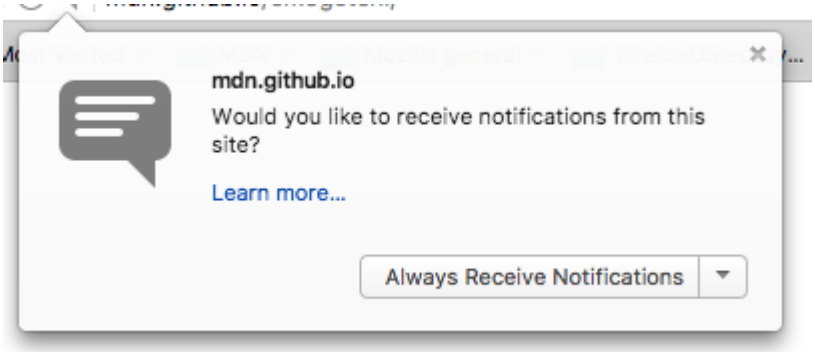
它们在适当的地方有额外的安全机制

WebAPI功能受到与JavaScript和其他Web技术（例如同源政策）相同的安全考虑 但是他们有时会有额外的安全机制。例如，一些更现代的WebAPI将只能在通过HTTPS提供的页面上工作，因为它们正在传输潜在的敏感数据（例如 服务工作者 和 推送）。

另外，一旦调用WebAPI请求，用户就可以在您的代码中启用一些WebAPI请求权限。作为一个例子，在加载我们之前的Geolocation 示例时，您可能注意到了类似下面的对话框 ：



该 通知API 请求以类似的方式许可：



这些许可提示会被提供给用户以确保安全 - 如果这些提示不在适当位置，那么网站可能会在您不知情的情况下开始秘密跟踪您的位置，或者通过大量恼人的通知向您发送垃圾邮件。

概要

在这一点上，您应该对API是什么，它们是如何工作的以及在JavaScript代码中可以对它们做什么有一个很好的了解。你可能很兴奋开始用特定的API来做一些有趣的事情，so let's go！接下来，我们将看到使用文档对象模型（DOM）处理文档。

Overview: Client-side Web API

最后修改： 2020年7月11日, 由 Mozilla 贡献者编辑

相关主题

新手请从这开始!

- Web 入门

HTML — 构建 Web

- HTML 介绍
- 多媒体与嵌入
- HTML 表格

CSS — 设计 Web

- CSS first steps
- CSS building blocks
- 样式化文字
- CSS 排版概述

JavaScript — 用户端动态脚本

- JavaScript 第一步

- ▶ JavaScript 基础要件
- ▶ JavaScript 对象介绍
- ▶ Asynchronous JavaScript
- ▼ 客户端网页 API

  客户端网页 API

  网页 API 介绍

  操纵文档

  从服务器获取数据

  第三方 API

  画图

  视频与音频 API

  客户端存储

Web forms — Working with user data

- ▶ Core forms learning pathway
- ▶ Advanced forms articles

可访问性 — 使每个人都能使用 Web

- ▶ 可访问性指南
- ▶ 可访问性测评

工具与测试

- ▶ Client-side web development tools
- ▶ Introduction to client-side frameworks
- ▶ React
- ▶ Ember
- ▶ Vue
- ▶ Git and GitHub

- 跨浏览器测试

服务端网页编程

- 第一步
- Django 网站框架 (Python)
- Express 网页框架 (node.js/JavaScript)

更多资源

- 常见问题

如何贡献