AJAX 概述

背景

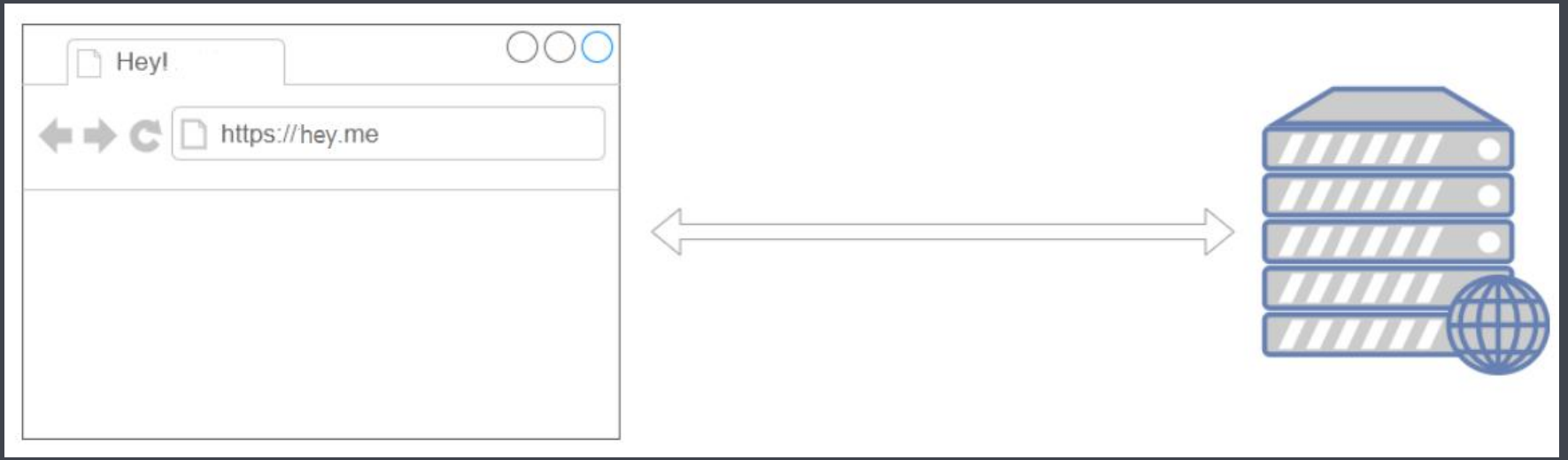
背景

- 在了解 AJAX 之前我们可以简单的认为「JavaScript 能力有限」，因为在此之前 Web 平台提供所有的 API 都只停留在「单机」的阶段。

- 这样就会造成一些无法实现的功能，例如：
  1. 无法在实现用户登录功能时，当用户输入邮箱地址显示用户对应的头像
  2. 无法在实现用户注册功能时，当用户输入邮箱或者用户名就提示是否存在
  3. 无法在实现留言板功能时，实时看到最新的用户留言

这些功能的开发最终都卡在一个相同的问题上

数据存放在服务端，无法通过已知的 API 获取


Hey!
https://hey.me

已知发送请求的方式

- 地址栏输入地址，回车，刷新
- 特定元素的 href 或 src 属性
- 表单提交

- 这些方案都是我们无法通过或者很难通过代码的方式进行编程操作的

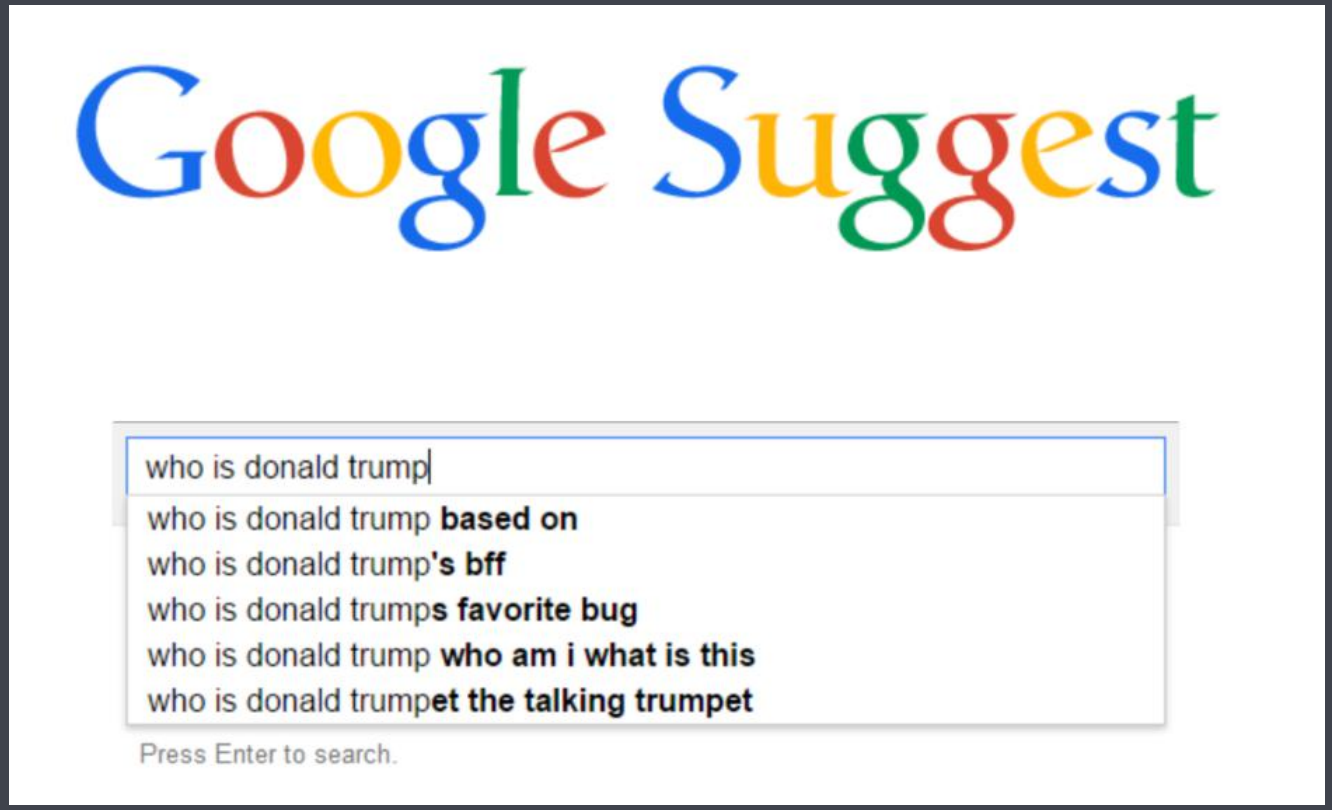
需求

- 对服务端发出请求并且接受服务端返回的响应
- 如果我们可以通过 JavaScript 直接发送网络请求，那么 Web 的可能就会更多，随之能够实现的功能也会更多，至少不再是只能开发「单机游戏」

Google Suggest

Google Suggest

- AJAX（Asynchronous JavaScript and XML），最早出现在 2005 年的 Google Suggest
- 它不是像 HTML、JavaScript 或 CSS 这样的一种“正式的”技术
- 它是在浏览器端进行网络编程（发送请求、接收响应）的技术方案
- 它使我们可以通过 JavaScript 直接获取服务端最新的内容而不必重新加载页面
- 让 Web 更能接近桌面应用的用户体验


Google Suggest
who is donald trump
who is donald trump based on
who is donald trump's bff
who is donald trumps favorite bug
who is donald trump who am i what is this
who is donald trumpet the talking trumpet
Press Enter to search.

AJAX

Asynchronous Javascript And XML

- AJAX 就是浏览器提供的一套 API，可以通过 JavaScript 调用，从而实现通过代码控制请求与响应。实现通过 JavaScript 进行网络编程。
- XML：最早在客户端与服务端之间传递数据时所采用的数据格式

应用场景

- 按需获取数据
- 对用户数据校验
- 自动更新页面内容
- 提升用户体验，无刷新的体验

体验 AJAX

体验

- 使用 jQuery 中封装的 Ajax，快速体验带来的效果
- 免费接口：https://jsonplaceholder.typicode.com/

原生 AJAX

工作中经常使用各种方便的封装的 AJAX 的库

但是，我们必须知道它的原理

发送 ajax 请求步骤

1、创建 XMLHttpRequest 类型的对象

2、准备发送，打开与一个网址之间的连接

3、执行发送动作

4、指定 xhr 状态变化事件处理函数

原生 AJAX 详解

发送 ajax 请求步骤

1、创建 XMLHttpRequest 类型的对象

2、准备发送，打开与一个网址之间的连接

3、执行发送动作

4、指定 xhr 状态变化事件处理函数

XMLHttpRequest 类型对象

AJAX API 中核心提供的是一个 XMLHttpRequest 类型，所有的 AJAX 操作都需要使用到这个类型。

```
var xhr = new XMLHttpRequest();
```

IE6 兼容

```
xhr = new ActiveXObject("Microsoft.XMLHTTP");
```

open() 方法开启请求

- 本质上 XMLHttpRequest 就是 JavaScript 在 Web 平台中发送 HTTP 请求的手段，所以我们发送出去的请求仍然是 HTTP 请求，同样符合 HTTP 约定的格式。
- 语法：xhr.open(method, url)
- method：要使用的HTTP方法，比如「GET」、「POST」、「PUT」、「DELETE」、等。
- url：要向其发送请求的 URL 地址，字符串格式。

setRequestHeader() 方法设置请求头

- 此方法必须在 open() 方法和 send() 之间调用。
- 语法：xhr.setRequestHeader(header, value);
- header：一般设置 “Content-Type” ，传输数据类型，即服务器需要我们传送的数据类型
- value：具体的数据类型，常用 "application/x-www-form-urlencoded" 和 "application/json"。

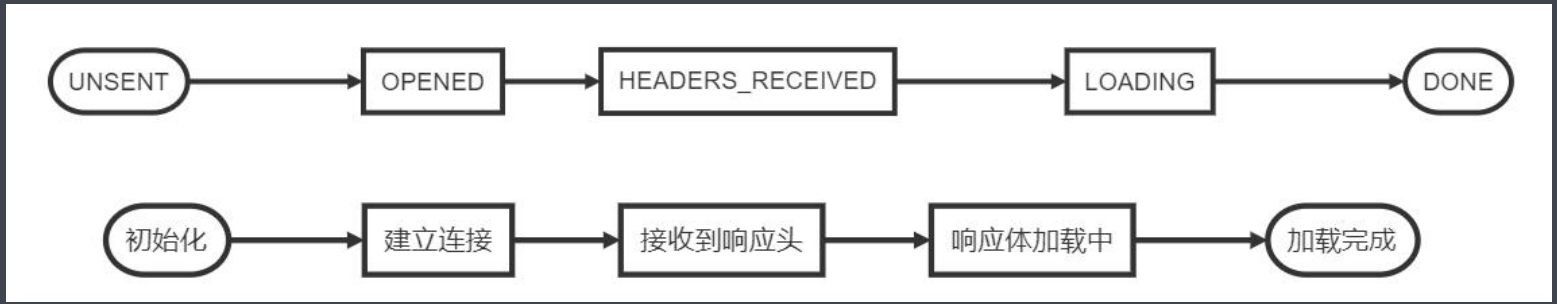
send() 方法发送请求

- 用于发送 HTTP 请求
- 语法：xhr.send(body)
- body：在XHR请求中要发送的数据体，根据请求头中的类型进行传参。
- 如果是 GET 方法，无需设置数据体，可以传 null 或者不传参。

readyState 属性

- readyState 属性返回一个 XMLHttpRequest 代理当前所处的状态，由于 readystatechange 事件是在 xhr 对象状态变化时触发（不单是在得到响应时），也就意味着这个事件会被触发多次，所以我们有必要了解每一个状态值代表的含义：

readyState	状态描述	说明
0	UNSENT	代理 XHR 被创建，但尚未调用 open() 方法
1	OPENED	open() 方法已经被调用，建立了连接。
2	HEADERS_RECEIVED	send() 方法已经被调用，并且已经可以获取状态行和响应头
3	LOADING	响应体下载中， responseText 属性可能已经包含部分数据
4	DONE	响应体下载完成，可以直接使用 responseText


UNSENT
OPENED
HEADERS_RECEIVED
LOADING
DONE
初始化
建立连接
接收到响应头
响应体加载中
加载完成

事件处理函数

- 一般我们都是在 readyState 值为 4 时，执行响应的后续逻辑。

```
xhr.onreadystatechange = function() {
  if (this.readyState === 4) {
    // 后续逻辑.....
  }
};
```

同步与异步

现实场景理解

- 同步：一个人在同一个时刻只能做一件事情，在执行一些耗时的操作（不需要看管）不去做别的事，只是等待
- 异步：在执行一些耗时的操作（不需要看管）去做别的事，而不是等待

Ajax 中的实现

- xhr.open() 方法第三个参数要求传入的是一个 boolean 值，其作用就是设置此次请求是否采用异步方式执行
- 默认为 true 异步，如果需要同步执行可以通过传递 false 实现
- 如果采用同步方式执行，则代码会卡死在 xhr.send() 这一步

建议

- 为了让这个事件可以更加可靠（一定触发），在发送请求 send() 之前，一定是先注册 readystatechange
- 不论是同步或异步都能触发成功
- 了解同步模式即可，切记不要使用同步模式

响应数据格式

提问

- 如果希望服务端返回一个复杂数据，该如何处理？
- 关心的问题就是服务端发出何种格式的数据，这种格式如何在客户端用 JavaScript 解析。

XML

XML

- 一种数据描述手段
- 老掉牙的东西，简单演示一下，不在这里浪费时间，基本现在的项目不用了
- 淘汰的原因：数据冗余太多

JSON

JSON

- JavaScript Object Notation,JavaScript 对象表示法
- 也是一种数据描述手段，类似于 JavaScript 字面量方式
- 服务端采用 JSON 格式返回数据，客户端按照 JSON 格式解析数据

注意

- 不管是 JSON 也好，还是 XML，只是在 AJAX 请求过程中用到，并不代表它们与 AJAX 之间有必然的联系，它们只是数据协议罢了。
- 不管服务端是采用 XML 还是采用 JSON 本质上都是将数据返回给客户端。
- 服务端应该根据响应内容的格式设置一个合理的 Content-Type。

JSON Server

平时我们也会自己写一些数据，通过 Ajax 获取

所以，需要在本地搭建一个临时服务器

json-server

- json-server 是一个 Node 模块，运行 Express 服务器，你可以指定一个 json 文件作为 api 的数据源。
- 也就是说，我们可以使用它快速的搭建一个 web 服务器。
- 网址：https://github.com/typicode/json-server

原生 AJAX 具体用法

GET 请求

- 通常在一次 GET 请求过程中，参数传递都是通过 URL 地址中的 `?` 参数传递。
- 一般在 GET 请求中，无需设置请求头
- 无需设置响应体，可以传 null 或者干脆不传

POST 请求

- POST 请求过程中，都是采用请求体承载需要提交的数据。
- 需要设置请求头中的 Content-Type，以便于服务端接收数据
- 需要提交到服务端的数据可以通过 send 方法的参数传递

处理响应数据渲染

客户端中拿到请求的数据过后最常见的就是把这些数据呈现到界面上。

如果数据结构简单，可以直接通过字符串操作（拼接）的方式处理

但是如果数据过于复杂，字符串拼接维护成本太大，就不推荐了

可以使用模版引擎或者 ES6 提供的模板字符串

封装 AJAX 库

自己封装一个 AJAX 函数

- 这里主要是为了了解封装的过程，一般情况在开发中都是使用第三方提供的 AJAX 库，因为它们可能更加严谨。
- 为了在后续的开发过程中可以更方便的使用这套 API，一般的做法都是将其封装到一个函数中以便调用。

拉勾教育

—互联网人实战大学—

下载「拉勾教育App」
获取更多内容

扫码联系老师

技能评估、福利资料、课程优惠

Made with ♥ by LagouFed