WebSockets 简介：将套接字引入网络


By Malte Ubl and Eiji Kitamura

已发布: 十月 20th, 2010
Comments: 49

问题：低延迟的客户端-服务器和服务器-客户端连接

一直以来，网络在很大程度上都是围绕着所谓 HTTP 的请求/响应模式而构建的。客户端加载一个网页，然后直到用户点击下一页之前，什么都不会发生。在 2005 年左右，AJAX 开始让网络变得更加动态了。但所有 HTTP 通信仍然是由客户端控制的，这就需要用户互动或定期轮询，以便从服务器加载新数据。

长期以来存在着各种技术，可让服务器得知有新数据可用时，立即将数据发送到客户端。这些技术名目繁多，例如“推送”或 Comet。最普遍的一种黑客手段是对服务器发起的链接创建假象，这称为长轮询。利用长轮询，客户端可打开指向服务器的 HTTP 连接，而服务器会一直保持连接打开，直到发送响应。服务器只要实际拥有新数据，就会发送响应（其他技术包括 Flash、XHR multipart 请求和所谓的 htmlfiles）。长轮询和其他技术非常好用，您在 Gmail 聊天等应用中经常使用它们。

但是，这些解决方案都存在一个共同的问题：它们带有 HTTP 的开销，导致它们不适用于低延迟应用。可以想象一下浏览器中的多人第一人称射击游戏，或者其他任何带有即时要素的在线游戏。

WebSocket 简介：将套接字引入网络

WebSocket 规范定义了一种 API，可在网络浏览器和服务器之间建立“套接字”连接。简单地说：客户端和服务器之间存在持久的连接，而且双方都可以随时开始发送数据。

使用入门

只需调用 WebSocket 构造函数即可打开 WebSocket 连接：

```
var connection = new
```

```
WebSocket('ws://html5rocks.websocket.org/echo', ['soap',
'xmpp']);
```

请注意 `ws:`。这是 WebSocket 连接的新网址架构。对于安全 WebSocket 连接还有 `wss:`，就像 `https:` 用于安全 HTTP 连接一样。

立即向连接附加一些事件处理程序可让您知道连接打开、收到传入讯息或出现错误的时间。

第二个参数可接受可选子协议，它既可以是字符串，也可以是字符串数组。每个字符串都应代表一个子协议名称，而服务器只能接受数组中通过的一个子协议。访问 WebSocket 对象的 `protocol` 属性可确定接受的子协议。

子协议名称必须是 IANA 注册表中的某个注册子协议名称。截止 2012 年 2 月，只有一个注册子协议名称 (soap)。

```
// When the connection is open, send some data to the server
connection.onopen = function () {
  connection.send('Ping'); // Send the message 'Ping' to the
server
};

// Log errors
connection.onerror = function (error) {
  console.log('WebSocket Error ' + error);
};

// Log messages from the server
connection.onmessage = function (e) {
  console.log('Server: ' + e.data);
};
```

与服务器通信

与服务器建立连接后（启动 `open` 事件时），我们可以开始对连接对象使用 `send('your message')` 方法，向服务器发送数据。该方法以前只支持字符串，但根据最新的规范，现在也可以发送二进制讯息了。要发送二进制数据，您可以使用 `Blob` 或 `ArrayBuffer` 对象。

```
// Sending String
connection.send('your message');

// Sending canvas ImageData as ArrayBuffer
var img = canvas_context.getImageData(0, 0, 400, 320);
var binary = new Uint8Array(img.data.length);
```

```
for (var i = 0; i < img.data.length; i++) {
  binary[i] = img.data[i];
}
connection.send(binary.buffer);

// Sending file as Blob
var file =
document.querySelector('input[type="file"]').files[0];
connection.send(file);
```

同样，服务器也可能随时向我们发送讯息。只要发生这种情况，就会启动 onmessage 回调。该回调接收的是事件对象，而实际的讯息可通过 data 属性进行访问。

在最新规范中，WebSocket 也可以接收二进制讯息。接收的二进制帧可以是 Blob 或 ArrayBuffer 格式。要指定收到的二进制数据的格式，可将 WebSocket 对象的 binaryType 属性设为“blob”或“arraybuffer”。默认格式为“blob”（您不必在发送时校正 binaryType 参数）。

```
// Setting binaryType to accept received binary as either
'blob' or 'arraybuffer'
connection.binaryType = 'arraybuffer';
connection.onmessage = function(e) {
  console.log(e.data.byteLength); // ArrayBuffer object if
binary
};
```

WebSocket 的另一个新增功能是扩展。利用扩展，可以发送压缩帧、多路复用帧等。您可以检查 open 事件后的 WebSocket 对象的 extensions 属性，查找服务器所接受的扩展。截止 2012 年 2 月，还没有官方发布的扩展规范。

```
// Determining accepted extensions
console.log(connection.extensions);
```

跨源通信

作为现代协议，跨源通信已内置在 WebSocket 中。虽然您仍然应该确保只与自己信任的客户端和服务器通信，但 WebSocket 可实现任何域上多方之间的通信。服务器将决定是向所有客户端，还是只向驻留在一组指定域上的客户端提供服务。

代理服务器

每一种新技术在出现时，都会伴随着一系列问题。WebSocket 也不例外，它与大多数公司网络中用于调解 HTTP 连接的代理服务器不兼容。WebSocket 协议使用 HTTP 升级系统（通常用于 HTTP/SSL）将 HTTP 连接“升级”为 WebSocket 连接。某些代理服务器不支持这种升级，并会断开连接。因此，即使指定的客户端使用了 WebSocket 协议，可能也无法建立连接。这就使得下一部分的内容更加重要了。

立即使用 WebSocket

WebSocket 仍是一项新兴技术，并未在所有浏览器中全面实施。而在无法使用 WebSocket 的情况下，只要通过一些使用了上述某个回调的库，就可以立即使用 WebSocket 了。在这一方面使用非常普遍的库是 socket.io，其中自带了协议的客户端和服务器实施，并包含回调（截止 2012 年 2 月，socket.io 还不支持二进制讯息传输）。还有一些商业解决方案，例如 PusherApp，通过提供向客户端发送 WebSocket 讯息的 HTTP API，可轻松地集成到任何网络环境中。由于额外的 HTTP 请求，这些解决方案与纯 WebSocket 相比总是会有额外的开销。

服务器端

使用 WebSocket 为服务器端应用带来了全新的用法。虽然 LAMP 等传统服务器堆栈是围绕 HTTP 请求/响应循环而设计的，但是通常无法很好地处理大量打开的 WebSocket 连接。要同时维持大量连接处于打开状态，就需要能以低性能开销接收高并发数据的架构。此类架构通常是围绕线程或所谓的非阻塞 IO 而设计的。

服务器端实施

- Node.js
    - Socket.IO
    - WebSocket-Node
    - ws
- Java
    - Jetty
- Ruby
    - EventMachine
- Python
    - pywebsocket
    - Tornado

- Erlang
    - Shirasu
- C++
    - libwebsockets
- .NET
    - SuperWebSocket

协议版本

现在，WebSocket 的单线协议（客户端与服务器之间的握手和数据传输）是 RFC6455。最新版的 Chrome 浏览器和 Android 版 Chrome 浏览器与 RFC6455 完全兼容（包括二进制讯息传输）。另外，Firefox 11 和 Internet Explorer 10 也会实现兼容。您仍可以使用旧版协议，但由于它们已知存在漏洞，我们不建议使用。如果您有旧版 WebSocket 协议的服务器实施，我们建议您将其更新到最新版本。

用例

如果您需要在客户端与服务器之间建立极低延迟、近乎即时的连接，则可使用 WebSocket。请记住，这可能需要您重新考虑构建服务器端应用的方式，将新的关注点放在事件队列等技术上。以下是一些用例：

- 多人在线游戏
- 聊天应用
- 体育赛况直播
- 即时更新社交信息流

演示

- Plink
- Paint With Me
- Pixelatr
- Dashed
- 大型多人在线填字游戏
- Ping 服务器（在以上示例中使用）
- HTML5demos 示例

参考