原文地址：http://www.ruanyifeng.com/blog/2017/05/websocket.html

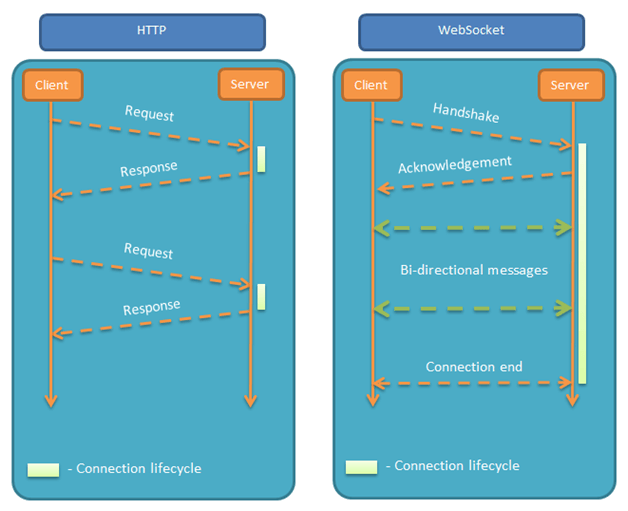
**Http**

HTTP 协议有一个缺陷：通信只能由客户端发起。

这种单向请求的特点，注定了如果服务器有连续的状态变化，客户端要获知就非常麻烦。我们只能使用"轮询"：每隔一段时候，就发出一个询问

**WebSocket**

WebSocket 的最大特点就是，服务器可以主动向客户端推送信息，客户端也可以主动向服务器发送信息，是真正的双向平等对话



**客户端的简单示例**

var ws = new WebSocket("wss://echo.websocket.org");

ws.onopen = function(evt) {

  console.log("Connection open ...");

  ws.send("Hello WebSockets!");

};

ws.onmessage = function(evt) {

  console.log( "Received Message: " + evt.data);

  ws.close();

};

ws.onclose = function(evt) {

  console.log("Connection closed.");

};

**客户端的 API**

**WebSocket 构造函数**

var ws = new WebSocket('ws://localhost:8080');

**webSocket.readyState**

readyState属性返回实例对象的当前状态，共有四种

CONNECTING：值为0，表示正在连接。

OPEN：值为1，表示连接成功，可以通信了。

CLOSING：值为2，表示连接正在关闭。

CLOSED：值为3，表示连接已经关闭，或者打开连接失败。

**webSocket.onopen**

onopen属性，用于指定连接成功后的回调函数

ws.onopen = function () {

    ws.send('Hello Server!');

}

如果要指定多个回调函数，可以使用addEventListener方法

ws.addEventListener('open', function (event) {

    ws.send('Hello Server!');

});

**webSocket.onclose**

onclose属性，用于指定连接关闭后的回调函数

ws.onclose = function (event) {

    var code = event.code;

    var reason = event.reason;

    var wasClean = event.wasClean;

    // handle close event

};

ws.addEventListener("close", function (event) {

    var code = event.code;

    var reason = event.reason;

    var wasClean = event.wasClean;

    // handle close event

});

**webSocket.onmessage**

onmessage属性，用于指定收到服务器数据后的回调函数

ws.onmessage = function (event) {

    var data = event.data;

    // 处理数据

};

ws.addEventListener("message", function (event) {

    var data = event.data;

    // 处理数据

});

服务器数据可能是文本，也可能是二进制数据（blob对象或Arraybuffer对象）

ws.onmessage = function (event) {

    if (typeof event.data === String) {

        console.log("Received data string");

    }

    if (event.data instanceof ArrayBuffer) {

        var buffer = event.data;

        console.log("Received arraybuffer");

    }

}

除了动态判断收到的数据类型，也可以使用binaryType属性，显式指定收到的二进制数据类型

// 收到的是 blob 数据

ws.binaryType = "blob";

ws.onmessage = function (e) {

    console.log(e.data.size);

};

// 收到的是 ArrayBuffer 数据

ws.binaryType = "arraybuffer";

ws.onmessage = function (e) {

    console.log(e.data.byteLength);

};

**webSocket.send()**

send()方法用于向服务器发送数据

// 发送文本

ws.send('your message');

// 发生文件 blob

var file = document

    .querySelector('input[type="file"]')

    .files[0];

ws.send(file);

// 发送文件 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];

}

ws.send(binary.buffer);

**webSocket.bufferedAmount**

实例对象的bufferedAmount属性，表示还有多少字节的二进制数据没有发送出去。它可以用来判断发送是否结束。

var data = new ArrayBuffer(10000000);

socket.send(data);

if (socket.bufferedAmount === 0) {

    // 发送完毕

} else {

    // 发送还没结束

}

**webSocket.onerror**

onerror属性，用于指定报错时的回调函数

socket.onerror = function (event) {

    // handle error event

};

socket.addEventListener("error", function (event) {

    // handle error event

});