Daily_Learning(2019-12-2)

简单讲解一下http2的多路复用(网易)

在 HTTP/1 中，每次请求都会建立一次HTTP连接，也就是我们常说的3次握手4次挥手，这个过程在一次请求过程中占用了相当长的时间，即使开启了 Keep-Alive ，解决了多次连接的问题，但是依然有两个效率上的问题：

- 第一个：串行的文件传输。当请求a文件时，b文件只能等待，等待a连接到服务器、服务器处理文件、服务器返回文件，这三个步骤。我们假设这三步用时都是1秒，那么a文件用时为3秒，b文件传输完成用时为6秒，依此类推。（注：此项计算有一个前提条件，就是浏览器和服务器是单通道传输）
- 第二个：连接数过多。我们假设Apache设置了最大并发数为300，因为浏览器限制，浏览器发起的最大请求数为6，也就是服务器能承载的最高并发为50，当第51个人访问时，就需要等待前面某个请求处理完成。

HTTP/2的多路复用就是为了解决上述的两个性能问题。 在 HTTP/2 中，有两个非常重要的概念，分别是帧（frame）和流（stream）。 帧代表着最小的数据单位，每个帧会标识出该帧属于哪个流，流也就是多个帧组成的数据流。 多路复用，就是在一个 TCP 连接中可以存在多条流。换句话说，也就是可以发送多个请求，对端可以通过帧中的标识知道属于哪个请求。通过这个技术，可以避免 HTTP 旧版本中的队头阻塞问题，极大的提高传输性能。

输出以下代码运行结果

```
// example 1
var a={}, b='123', c=123;
a[b]='b';
a[c]='c';
console.log(a[b]);
```

```
----------------------
// example 2
var a={}, b=Symbol('123'), c=Symbol('123');
a[b]='b';
a[c]='c';
console.log(a[b]);

----------------------
// example 3
var a={}, b={key:'123'}, c={key:'456'};
a[b]='b';
a[c]='c';
console.log(a[b]);
```

- 对象的键名只能是字符串和 Symbol 类型。
- 其他类型的键名会被转换成字符串类型。
- 对象转字符串默认会调用 toString 方法。

```
// example 1
var a={}, b='123', c=123;
a[b]='b';

// c 的键名会被转换成字符串'123'，这里会把 b 覆盖掉。
a[c]='c';

// 输出 c
console.log(a[b]);
```


```
// example 2
var a={}, b=Symbol('123'), c=Symbol('123');

// b 是 Symbol 类型，不需要转换。
a[b]='b';


// c 是 Symbol 类型，不需要转换。任何一个 Symbol 类型的值都是不相等的，所以不会覆盖掉 b。
a[c]='c';


// 输出 b
console.log(a[b]);
```

// example 3
var a={}, b={key:'123'}, c={key:'456'};

// b 不是字符串也不是 Symbol 类型，需要转换成字符串。
// 对象类型会调用 toString 方法转换成字符串 [object Object]。
a[b]='b';

// c 不是字符串也不是 Symbol 类型，需要转换成字符串。
// 对象类型会调用 toString 方法转换成字符串 [object Object]。这里会把 b 覆盖掉。
a[c]='c';