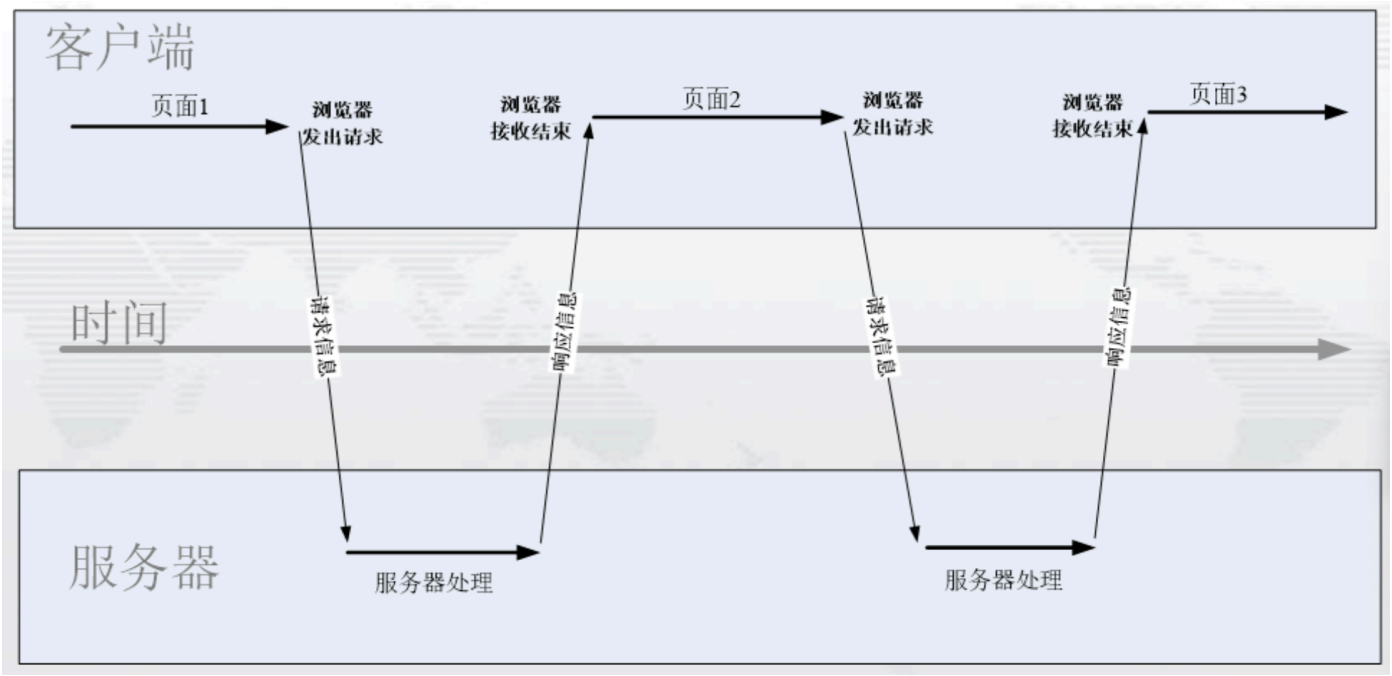
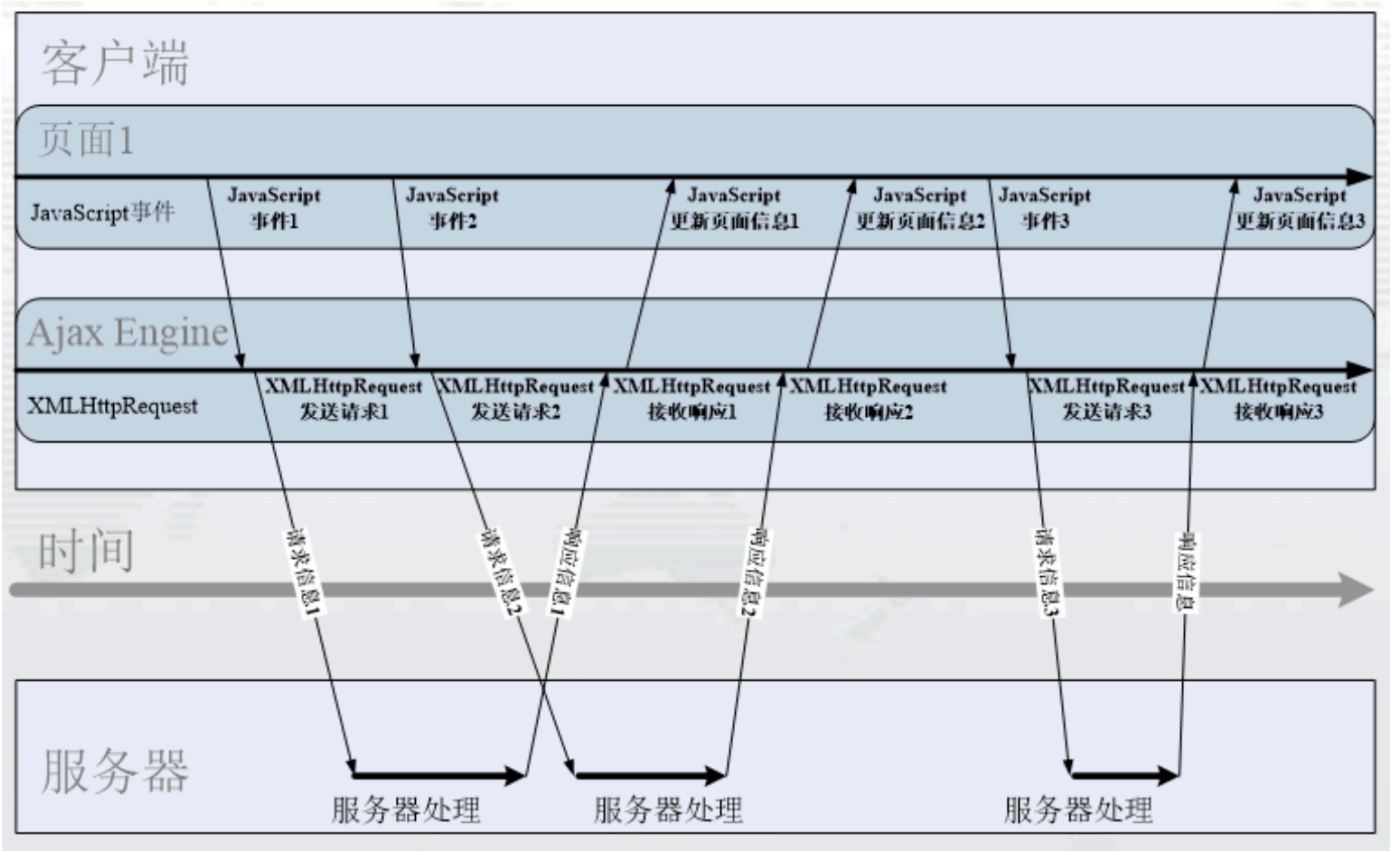
1. Ajax

- **Ajax** 异步请求: 异步 `JavaScript` 和 `XML` ，使不刷新页面向服务器发起请求成为可能
- 通过异步请求实现了网页的局部刷新
- 非阻塞的

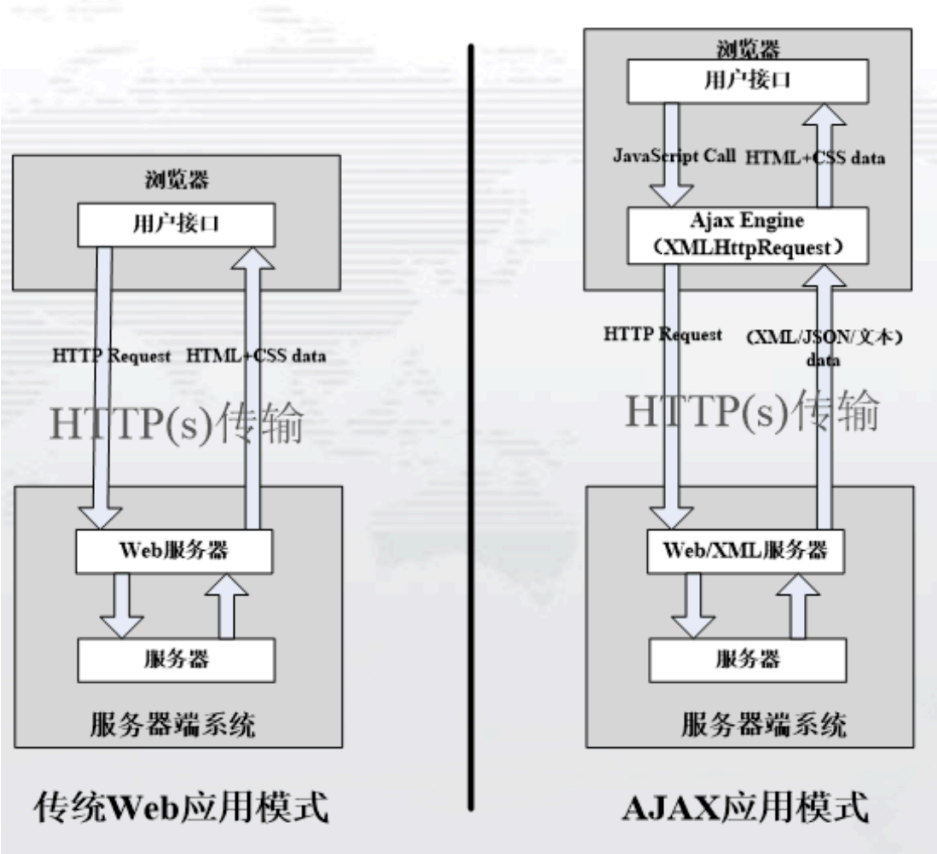
- 传统 `Web` 请求，同步请求
- 发送请求 ⟹ 等待 ⟹ 发送请求 ⟹ 等待



- Ajax 请求
- 媒介 `Ajax Engine` 发送异步请求，客户端在相应到达前可以进行其它操作，`Ajax Engine` 继续监听服务器的响应状态，服务器完成响应后，获取响应结果更新当前页面内容。



- 两种请求过程比较



- Ajax 关键技术
- 使用 `XHTML(HTML)` 和 `CSS` 构建标准化的展示层;

- 使用 `DOM` 进行动态显示和交互;
- 使用 `XML` 或 `JSON` 等进行数据交换和操纵 ;
- 使用 `XMLHttpRequest` 来和服务器进行异步通信
- 使用 `JavaScript` 将所有元素绑定在一 起。
- 其中 `XMLHttpRequest` 是Ajax技术得以实现的一 个重要的JavaScript对象。

2. XMLHttpRequest

`XMLHttpRequest` 实质上是一个 JavaScript对象，是 Ajax 的核心，使用这个对象,可以在客户端向服务器发起 HTTP 请求，并且可以访问和处理服务器返回的应答数据。

- 声明 `XMLHttpRequest`

```
var xhr = false;

try {
  xhr = new XMLHttpRequest();
} catch(ex) {
  try {
    xhr = new ActiveXObject("Microsoft.XMLHTTP");
  } catch {
    xhr = false;
  }
}

if(!xhr) {
  alert("false");
}
else alert("success");
```

2.1. 属性

属性	描述
readyState	状态表示
onreadystatechange	状态改变触发的事件处理程序
responseText	服务器进程返回的数据的字符串形式
responseXML	服务器进程返回的XML文档
status	状态码 404 未找到，200 就绪
statusText	伴随状态码的字符串信息

- `readyState` 取值
    - 0：“未初始化” 状态; 此时，已创建一个 `XMLHttpRequest` 对象，还没有初始化
    - 1：发送状态，此时，代码已调用 `XMLHttpRequest.open()` 方法并且 `XMLHttpRequest` 已经准

备好把一个请求发送到服务器
    - 2：“发送”状态，此时通过 `send()` 方法把请求发送到服务器端，但还没有收到一个响应
    - 3：“正在接收”状态，已接收到 HTTP 响应头部信息，但是消息体部分还没有完全接收结束
    - 4：“已加载”状态，此时，响应已经被完全接收
- `status` 属性
    - 200： 请求成功
    - 202： 请求被接受但未被处理
    - 400： 错误请求
    - 404： 资源未找到
    - 500： 内部服务器错误
- `onreadystatechange` 属性
    - `onreadystatechange` 属性用于存储函数（或函数名）,每当readyState属性改变时，就会调用该函数，因此该函数正常情况下会被调用4次。在该函数中，通常只需在readyState值为4时做数据的获取和处理工作。
- `responseText` 属性
    - `responseText` 属性包含客户端接收到的 HTTP 响应的文本内容。当 `readyState` 值为 0、1 或 2 时，`responseText` 包含一个空字符串 ;当 `readyState` 值为3 时，响应中包含客户端还未完成的响应信息;当 `readyState` 为4（已加载)时，该 `responseText` 包含完整的响应信息。

2.2. 方法

方法	描述
abort()	停止当前请求
getAllResponseHeaders()	获取所有 HTTP 头部，以键/值对形式返回
getResponseHeader("header")	返回指定 HTTP 头部的串值
open(method, url)	建立对服务器的调用，method 参数可以是 get、post或put，url参数可以是相对 URL 或绝对 URL。
send(content)	向服务器发送请求
setResrequestHeader("header", "value")	把指定请求头设为提供的值，再设置任何请求头之前必须先调用 open() 方法

- `void open(method, url, asynch, username, password)`
- `open()` 方法会建立对服务器的调用
    - `method` ： 特定的请求方法
    - `url` ： 调用资源的 URL
    - `asynch` ： 指定异步还是同步
    - `username` ： 指定一个特定用户么
    - `password` ： 指定密码
- `void send(content)`
    - 向服务器发出请求。如果请求声明为异步的，这个方法就会立即返回，否则会等待直到接受到响应为止，参数可以是 DOM 对象的实例
- `void setRequestHeader(header, value)`
    - 用于为 HTTP 请求中一个给定的请求头设置值，必须在调用 `open()` 之后才能调用
        - `header` ： 设置的请求名称
        - `value` ： 设置的值

- `void abort()`
    - 用于停止请求
- `String getAllResponseHeaders()`
    - 返回一个字符串，包含 HTTP 请求的所有响应头部

3. Ajax 实例

- 创建步骤
    - 在页面中定义 Ajax 请求的触发事件;
    - 创建 XMLHttpRequest 对象;
    - 确定请求地址和请求参数;
    - 调用 XMLHttpRequest 对象的 `open()` 方法建立对服务器的调用;
    - 通过 XMLHttpRequest 对象的 `onreadystatechange` 属性指定响应事件处理函数;
    - 在函数中根据响应状态进行数据获取和数据处理工作;
    - 调用 `XMLHttpRequest` 对象的 `send()` 方法向服务器发出请求。

4. JSON

`JSON` 有两种结构:

- `名/值` 对的集合 :在不同的语言中，它被理解为对象、结构、关联数组等;

```
{
  "first" : "second",
  "name" : "test"
}
```

- 值的有序列表(`An ordered list of values`): 在大部分语言中，它被理解为数组。
    - 声明一个名为 `user` 的数组对象

```
var usersArray =
{
  "user" : [
    {"firstName" : "h", "LastName" : "yx", "email" : "..."},
    {},
    {}
  ]
}
usersArray.user[0].firstName
```

JSON 在 Ajax 中的应用

- `XMLHttpRequest` 对象可以通过 `responseText` 属性获取字符串格式的响应数据

5. jQuery

5.1. 基本操作

- DOM 的遍历和操作

```
$("button.continue").html("Next Step...")
```

- 事件处理：id 为 `banner-message` 隐藏对象，在其按钮被单击时，使其变为可见

```
var hiddenBox = ${"#banner-message"}
$("#button-container").on("click", function(event)) {
  hiddenBox.show();
}
```

5.2. 实现 Ajax

- `ajax()`
  - `$.ajax(options)` ：返回创建的 `XMLHttpRequest` 对象
    - `options` ：表示 `ajax` 请求设置，所有选项都是可选的
    - 方法返回 `XMLHttpRequest` 对象
- `load(url, data, callback)` ：从服务器加载数据，把返回的数据放入被选元素中
  - `url` ：必需参数，指定需要加载的 URL
  - `data` ：可选参数，规定与请求一同发送的查询字符串键/值对集合
  - `callback` ：可选参数，请求成功完成时的回调函数
- `get() / post()`
- `getJSON(url, data, callback)`
  - 通过 HTTP GET 请求载入 `JSON`
  - `url` ：请求地址
  - `data` ：连同请求发送的数据