[十.表单控件 1](#_Toc11311636)

[6.其他元素 1](#_Toc11311637)

[7.新表单元素 1](#_Toc11311638)

[Ajax 2](#_Toc11311639)

[一.HTTP协议 2](#_Toc11311640)

[1.URL 2](#_Toc11311641)

[2.HTTP协议 3](#_Toc11311642)

[3.web请求的详解 4](#_Toc11311643)

[4.消息/报文Message 4](#_Toc11311644)

[5.请求消息Request Message 5](#_Toc11311645)

[6.响应消息 Response Message 6](#_Toc11311646)

[7.缓存 7](#_Toc11311647)

[8.HTTP性能优化 9](#_Toc11311648)

[二.DOM操作 9](#_Toc11311649)

day04

# 十.表单控件

## 6.其他元素

③浮动框架---不属于form内容

|  |
| --- |
| <iframe></iframe>  width="100%"  height="6500px"  src=<http://www.codeboy.com> 要引入的网页路径  scrolling="no" 设置滚动条yes no auto  frameborder="0" 设置框架边框  由于我们还没有学习dom和bom  无法让iframe自适应高度  所以只能写一个固定的高度 |

## 7.新表单元素

①邮箱

|  |
| --- |
| <input type="email" name="email">  提交时，验证有没有@，以及@后面有没有内容 |

②搜索类型

|  |
| --- |
| <input type="search" name="search">  自带快速清除功能 |

③url类型

|  |
| --- |
| <input type="url" name="url">  提交时,验证开头是http://或者https:// |

④tel类型

|  |
| --- |
| <input type="tel" name="tel">  在移动端使用时，弹出虚拟键盘 |

⑤数字类型

|  |
| --- |
| <input min="10" max="20" step="2" type="number" name="number">  min="10" 能接收的最小值  max="20" 能接收的最小值  step="2" 步长 |

⑥范围类型

|  |
| --- |
| <input min="10" max="20" step="2" type="range" name="range">  min="10" 能接收的最小值  max="20" 能接收的最小值  step="2" 步长 |

⑦颜色类型

|  |
| --- |
| <input type="color" name="color"> |

⑧日期类型

|  |
| --- |
| <input type="date" name="date">  提供一个日期选择控件 |

⑨月份类型

|  |
| --- |
| <input type="month" name="month"> |

⑩星期类型

|  |
| --- |
| <input type="week" name="week"> |

# Ajax

# 一.HTTP协议

## 1.URL

|  |
| --- |
| 结构：协议+主机名称+目录结构+文件名称  URL完整的结构  <scheme>://<user>:<pwd>@<host>:<port>/<path>;  <params>?<query>#<frag> |

①scheme

|  |
| --- |
| 方案，协议，以哪种方式获取服务器资源  不区分大小写，常见的协议 http:// https:// ftp:// |

②user和pwd

|  |
| --- |
| <user>:<pwd> 用户名和密码，这种写法已经被淘汰 |

③host

|  |
| --- |
| <host>主机名：域名或者IP地址 |

④port

|  |
| --- |
| <port>端口号，对应计算机上某一个服务器 0~65535 2^16-1  0~1024 不要自己修改 |

⑤path

|  |
| --- |
| <path>路径，资源在服务器上具体存放位置 |

⑥params

|  |
| --- |
| <params>参数 跟踪状态参数 session/cookie |

⑦query

|  |
| --- |
| <query>查询字符串，get方法提交请求时，传输参数的查询字符串 |

⑧frag

|  |
| --- |
| #<frag>锚点 |

## 2.HTTP协议

|  |
| --- |
| HTTP:HyperText Transfer Protocal 超文本传输协议  规范了数据是如何打包和传输的（专门用于传输html文件的） |
| HTTP的历史 |

## 3.web请求的详解

|  |
| --- |
| 请求与响应流程： |

## 4.消息/报文Message

|  |
| --- |
| 1.请求消息Request Message(请求起始行，请求头，请求主体)  2.响应消息 Response Message(响应起始行，响应头，响应主体) |

## 5.请求消息Request Message

|  |
| --- |
| 请求消息，客户端发送给服务器的数据库  由三部分组成：请求起始行，请求头，请求主体 |

①请求起始行

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1.请求方法 restful Api   |  | | --- | | get: 客户端向服务器要数据的时候使用（查询）  靠地址栏传递查询字符串，有大小限制  无请求主体 | | post: 客户端传递给服务器数据，新建添加数据库条目的时候使用  有请求主体 | | put: 编辑服务器上的资源（文件和数据库内容）  使用请求主体传递数据 | | delete: 删除服务器上的资源（文件和数据库内容）  不使用请求主体，使用查询字符串 | | 请求方法—后台获取方法—数据库操作  get----req.query----delect  post---req.query----insert  put----req.query---update  delete-req.query---delete |   2.HTTP版本号 http 1.1  3.请求的url |

②请求头

|  |
| --- |
| 1.Host.www.tmoo.cn  浏览器告诉服务器请求的是哪个主机  2.Connection：keep-alive  浏览器告诉服务器，开启持久连接  3.User-Agent:  浏览器告诉服务器，我这个浏览器能够兼容的类型和版本号  4.Accept-Language:  浏览器告诉服务器，我这个浏览器可以接收的自然语言类型  5. Accept-Encoding:  浏览器告诉服务器，我这个浏览器可以接收的压缩文件的类型  6. Referer: http://www.tmooc.cn/  浏览器告诉服务器，我这个请求来自于哪个网页 |

③请求主体 form data

|  |
| --- |
| Formdata get/delete 没有请求主体  post/put 有请求主体 |

## 6.响应消息 Response Message

|  |
| --- |
| 响应消息，服务器发送给浏览器的数据库  由三部分组成：响应起始行，响应头，响应主体 |

①响应起始行

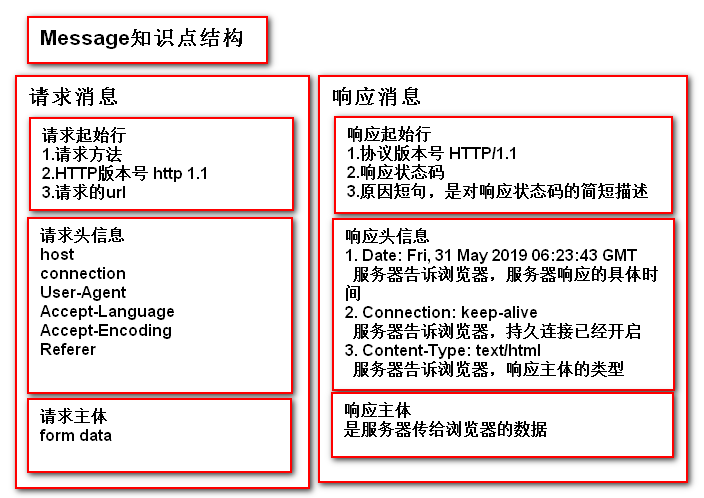
|  |  |
| --- | --- |
| 1.协议版本号 HTTP/1.1  2.响应状态码   |  | | --- | | 服务器高速浏览器，服务器的响应状态码是什么  1XX：正在请求，提示信息  2XX：200 响应成功  3XX：重定向  301 永久重定向  302 临时重定向  304 请求没有修改，命中了缓存  4XX：404 请求资源不存在  403 权限不够  405 请求的方法不被允许  500：服务器代码错误 |   3.原因短句，是对响应状态码的简短描述 |

②响应头

|  |
| --- |
| 1. Date: Fri, 31 May 2019 06:23:43 GMT  服务器告诉浏览器，服务器响应的具体时间  2. Connection: keep-alive  服务器告诉浏览器，持久连接已经开启  3. Content-Type: text/html  服务器告诉浏览器，响应主体的类型  text/html：响应回来的数据是html文本  text/plain：响应回来的数据是普通字符文本  text/css：响应回来的数据是一个css文本  application/javascript：响应回来的数据是js  image/jpeg png gif：响应回来的数据是图片  application/json：响应回来的数据是json字符串  application/xml：响应回来的数据是xml字符串 |

③响应主体

|  |
| --- |
| 是服务器传给浏览器的数据 |



## 7.缓存

|  |
| --- |
| 客户端将服务器响应回来的数据进行自动保存  当再次访问的时候，优先使用保存的数据 |

①缓存的优点

|  |
| --- |
| 1.减少冗余的数据传输，节省客户端流量  2.节省服务器的带宽  3.降低了对服务器的资源的消耗和运行的要求  4.降低了由于远距离传输而造成的加载延迟 |

②缓存的新鲜度和过期

|  |
| --- |
| 1.请求---无缓存---连接服务器---存缓存---浏览器渲染  2.请求---有缓存---够新鲜---使用缓存---浏览器渲染  3.请求---有缓存---不新鲜---连接服务器确认是否过期---没过期---更新缓存的新鲜度---浏览器渲染  4. 请求---有缓存---不新鲜---连接服务器确认是否过期---已经过期---再连接服务器要数据---存缓存---浏览器渲染 |

③与缓存相关的消息头

|  |
| --- |
| cache-control:max-age=1800  从服务器将文档传输到客户端之时起  此文档处于新鲜的秒数，这是一个相对事件  语法：cache-control:max-age =处于新鲜的秒数 |

④在页面中添加缓存需要修改消息头

|  |
| --- |
| <meta http-equiv=”消息头属性” content=”消息头的值”>  <meta http-equiv=” cache-control” content=”max-age=3600”> |

## 8.HTTP性能优化

|  |
| --- |
| HTTP连接的过程：  发起请求🡪建立连接🡪服务器接收处理请求🡪访问资源🡪构建响应  🡪发送响应🡪记录日志 |

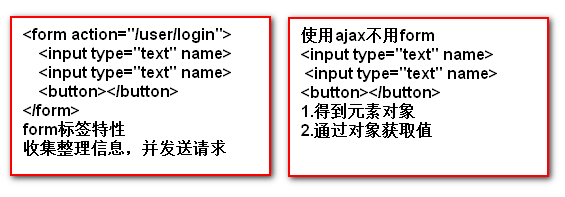
①HTTP连接性能的优化

|  |
| --- |
| 1.减少连接创建次数（开启持久连接）  2.减少请求次数（优化css、js文件，使用缓存）  3.提高服务器端运行速度  4.尽量减少响应数据的长度 |

②安全的HTTP协议

|  |
| --- |
| HTTPS：安全版本的HTTP协议  SSL：为数据通信提供安全支持  1.客户端发送请求--->SSL层加密--->服务器接收加密文件--->在SSL层解密，得到请求的明文，对请求做处理  2.服务器发送响应--->SSL层加密--->浏览器客户端得到加密文件--->在SSL解密，得到响应明文，解析运行响应内容 |

# 二.DOM操作



作业：

1.nodejs接口问题

2.完成下图，table布局

