day04

十.表单

4.表单元素

⑥其它元素

|  |
| --- |
| 1.label |
| 2.fieldset |
| 3.浮动框架  在一个html中引入其它html文本  <iframe></iframe>  src="00\_home.html" 要引入网页的路径  width="100%"  height="400px"  frameborder="2px" 框架的边框  scrolling="no" 设置滚动条 yes 或者 no  ex:  <iframe width="100%" height="400px" src="00\_home.html" frameborder="2px" scrolling="no"></iframe> |

5.新表单元素

①邮箱

|  |
| --- |
| <input type="email" name="uemail">  提交时验证填写的邮箱格式是否正确  正确的标准，有@符号，并且@符号前后有内容 |

②搜索类型

|  |
| --- |
| <input type="search" name="us">  自带快速清除的小叉叉 |

③url类型

|  |
| --- |
| <input type="url" name="uu">  验证以http://开头 |

④电话号码

|  |
| --- |
| <input type="tel" name="ut">  移动设备使用时，会弹出虚拟键盘 |

⑤数字类型

|  |
| --- |
| <input type="number" max="20" min="10" step="3">  max="20" 能接收的最大值  min="10" 最小值  step="3" 步长 |

⑥范围类型

|  |
| --- |
| <input type="range" max="50" min="30" step="5">  max="50"  min="30"  step="5" |

⑦颜色类型

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

⑧日期类型

|  |
| --- |
| <input type="date"> |

⑨月份类型

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

⑩周类型

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

**AJAX**

一.http协议

1.URL

|  |
| --- |
| http://www.codeboy.com/md/index.html  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、file |

②<user>:<pwd>

|  |
| --- |
| 登陆这个服务器的用户名和密码 |

③host

|  |
| --- |
| 主机名称 127.0.0.1 ip地址  www.baidu.com 域名  表示你要访问的服务器，在网络上具体的门牌号码 |

④port

|  |
| --- |
| 端口号 在主机上，程序开启服务，会对应一个唯一的端口号  端口号理论最大值 0~65535 不要修改0~1024的端口 |

⑤path

|  |
| --- |
| 路径，结构目录，资源在服务器上存放的位置 |

⑥params

|  |
| --- |
| 参数，跟踪状态的参数 保存在session/cookie |

⑦query

|  |
| --- |
| get方法提交请求时，问号后面的查询字符串 |

⑧#<frag>

|  |
| --- |
| 锚点 |

2.HTTP协议

|  |
| --- |
| HyperText Transfer Protocol 超文本传输协议  规范了数据是如何打包和传递的(专门用于传递html文件)  HTTP协议的历史，我们目前使用的版本 http/1.1 |

3.web请求原理

|  |
| --- |
|  |

4.消息/报文 Message

|  |
| --- |
| C:\Users\web\Desktop\web1905\web第二阶段练习效果图示\message.png  请求消息Request Message(请求起始行，请求头，请求主体)  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块  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.HTTP协议规定的请求的方法和Restful规则中的方法对比  Restful API是一种规则，利用http的已有的4个方法定义了4个行为，使用这个4个行为发送请求   |  |  | | --- | --- | | HTTP协议规定的请求的方法 | Restful规则中的方法 | | get 明文传参，上限2kb，向服务器要数据的时候使用，无请求主体 | get 查数据的时候  无请求主体  后台 req.query |  | | post 隐式传参,无大小限制，给服务器数据的时候使用，有请求主体 | post 创建(增)  有请求主体  后台 req.body | | put 放置文件到服务器，一般会被禁用 | put 修改，有请求主体  后台 req.body | | delete 把服务器上某个文件删除，一般会被禁用 | delete 删除，无请求主体  后台 req.query | |
| 2.协议版本号 |
| 3.请求的url |

②请求头

|  |
| --- |
| 1.Host: www.tmooc.cn  客户端告诉服务器。我要访问www.tmooc.cn这个主机。  2. Connection: keep-alive  客户端告诉服务器，请开启持久链接  3. User-Agent:  告诉服务器，我这个浏览器的类型和版本号  4. Accept-Encoding: gzip  告诉服务器，我这个浏览器可以接收的压缩文件的格式  5. Accept-Language: zh-CN  告诉服务器，我这个浏览器可以接收的自然语言的类型  6. Referer: http://www.tmooc.cn/  告诉服务器，当前的请求，是来源于哪个页面 |

③请求主体

|  |
| --- |
| 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，服务器发送给客户端的数据块  由三部分组成，响应起始行，响应头，响应主体 |

①响应起始行

|  |  |
| --- | --- |
| HTTP/1.1 200 OK  1.协议版本号 http/1.1  2.响应状态码   |  | | --- | | 服务器告诉客户端，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，301 永久重定向  302 临时重定向  304 请求未改变，命中缓存  4XX，404 not found 请求资源不存在  403 权限不够  405 请求方法不被允许  5XX, 500服务器代码出错 |   3.原因短句，对响应状态码简短的说明 |

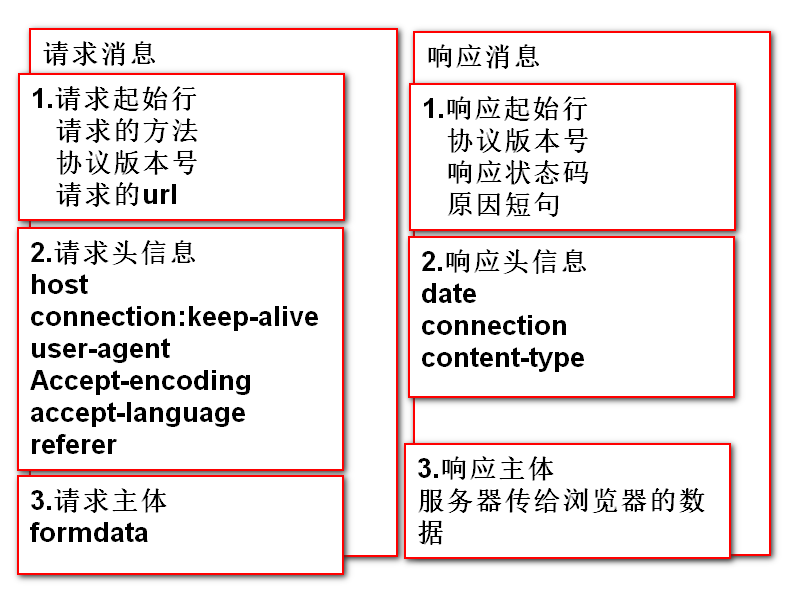
②响应头

|  |  |
| --- | --- |
| 1. Date: Thu, 04 Jul 2019 07:48:02 GMT 格林威治时间  告诉浏览器，服务器响应的时间  2. Connection: keep-alive  服务器告诉浏览器，我已经打开了持久链接  3. Content-Type: text/html  告诉浏览器，响应主体的类型是什么   |  | | --- | | text/html 响应回来的数据是html文本  text/css 响应回来的数据是css文本  application/javascript js文本  application/xml xml文本  application/json json 文本  image/jpg  image/png  image/gif | |

③响应主体

|  |
| --- |
| 服务器传递给浏览器的数据 |

总结



7.缓存

|  |
| --- |
| 客户端将服务器响应回来的数据  进行自动的保存  当再次访问的时候，直接使用保存的数据 |
| 缓存的优点  1.减少冗(rong)余的数据传输，节省了客户端的流量  2.节省服务器带宽  3.降低了对服务器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 |
| 缓存原理，新鲜度和过期  C:\Users\web\Desktop\web1905\web第二阶段练习效果图示\缓存流程.png  1.请求--无缓存--连接服务器--存缓存--客户端渲染  2.请求--有缓存--够新鲜--使用缓存--客户端渲染  3.请求--有缓存--不新鲜--连接服务器确认是否过期--没过期--更新缓存新鲜度--客户端渲染  4.请求--有缓存--不新鲜--连接服务器确认是否过期--过期--连接服务器--存缓存--客户端渲染 |

①操作缓存，需要前端和后端都写代码

|  |
| --- |
| 缓存相关的消息头  Cache-Control: max-age=0 单位是s  从服务器将文档传到客户端之时起  此文档处于新鲜的秒数，是一个相对时间 |

②修改请求消息头

|  |
| --- |
| <meta http-equiv="Cache-Control"  content="max-age=3600">  语法：  <meta http-equiv="消息头属性名称" content="值"> |

8.HTTP性能优化

|  |
| --- |
| HTTP的连接过程  发送请求-->建立连接-->服务器处理请求-->访问资源  -->构建响应-->发送响应-->记录日志 |

①http连接的性能优化

|  |
| --- |
| 1.减少连接创建的次数(开启持久连接)  2.减少请求的次数(代码设计更合理)  3.提高服务器端运行的速度  4.尽可能的减少响应数据的长度 |

9.安全的HTTP协议，HTTPS

|  |
| --- |
| HTTPS，是安全版本的http协议  S:SSL为数据通信提供安全支持  1.客户端发送请求--->ssl层加密--->服务器接收到加密文件--->SSL层解密，得到请求明文，对请求做处理  2.服务器发送响应--->SSL层加密---->客户端得到加密文件--->ssl层解密，得到响应明文，解析执行响应内容 |

二.Dom操作(简单dom操作)

|  |
| --- |
| ajax提交请求，不使用form表单  但是form表单自带收集数据的功能  如果不用form，就没有自动收集数据的功能了  我们需要使用js 的dom操作，手写代码，收集数据 |

1.完整的JavaScript的组成

|  |
| --- |
| 1.js核心代码：ECMA Script ES6  2.DOM Document object Model 文档对象模型  就是为了能够操作html中元素的(内容，值，样式)  3.BOM Browser object Model 浏览器对象模型  让js能够动态操作浏览器 |

2.使用js的dom获取页面数据

①找到元素对象

|  |
| --- |
| 获取html的元素对象  var obj=document.getElementById("元素id")；  document--->当前html文档的对象 |

②获取/修改这个元素的值/内容

作业：

1.nodejs 6个接口和app.js

2.在html中创建两个input，id分别为 d1和d2

再创建一个按钮

需求：点击按钮时，d1输入的内容，显示到d2中