1. MDN文档

```
https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Overview
```

2. HTTP请求交互的基本过程

```
1). 前后应用从浏览器端向服务器发送HTTP请求(请求报文)
2). 后台服务器接收到请求后, 调度服务器应用处理请求, 向浏览器端返回HTTP响应(响应报文)
3). 浏览器端接收到响应, 解析显示响应体/调用监视回调
```

3. HTTP请求报文

```
1). 请求行:
    method url
    GET /product_detail?id=2
    POST /login
2). headers: 多个请求头
    Host: www.baidu.com
    Cookie: BAIDUID=AD3B0FA706E; BIDUPSID=AD3B0FA706;
    Content-Type: application/x-www-form-urlencoded 或者 application/json
3). body: 请求体
    username=tom&pwd=123
    {"username": "tom", "pwd": 123}
```

4. HTTP响应报文

```
1). 响应状态码: 200/404
2). 多个响应头
    Content-Type: text/html;charset=utf-8
    Set-Cookie: BD_CK_SAM=1;path=/
3). 响应体
    html文本/json文本/js/css/图片...
```

5. post请求体参数格式

```
1). Content-Type: application/x-www-form-urlencoded;charset=utf-8
    用于键值对参数, 参数的键值用=连接, 参数之间用&连接
    例如: name=%E5%B0%8F%E6%98%8E&age=12
2). Content-Type: application/json;charset=utf-8
    用于json字符串参数
    例如: {"name": "%E5%B0%8F%E6%98%8E", "age": 12}
3). Content-Type: multipart/form-data
    用于文件上传请求
```

6. 常见响应状态码

```
200  OK                    请求成功。一般用于GET与POST请求
201  Created               已创建。成功请求并创建了新的资源
401  Unauthorized          未授权/请求要求用户的身份认证
404  Not Found             服务器无法根据客户端的请求找到资源
500  Internal Server Error 服务器内部错误，无法完成请求
```

7. 不同类型的请求及其作用

```
1). GET: 从服务器端读取数据
2). POST: 向服务器端添加新数据
3). PUT: 更新服务器端已经数据
4). DELETE: 删除服务器端数据
```

8. API的分类

```
1). REST API:    restful
    发送请求进行CRUD哪个操作由请求方式来决定
    同一个请求路径可以进行多个操作
    请求方式会用到GET/POST/PUT/DELETE
2). 非REST API:   restless
    请求方式不决定请求的CRUD操作
    一个请求路径只对应一个操作
    一般只有GET/POST
```

9. json-server是什么?

```
用来快速搭建REST API的工具包
```

10. 使用json-server

```
在线文档: https://github.com/typicode/json-server
下载: npm install -g json-server
目标根目录下创建数据库json文件: db.json
    {
      "posts": [
        { "id": 1, "title": "json-server", "author": "typicode" }
      ],
      "comments": [
        { "id": 1, "body": "some comment", "postId": 1 }
      ],
      "profile": { "name": "typicode" }
    }
启动服务器
    执行命令: json-server --watch db.json
```

11. 测试访问1: 使用浏览器

```
 http://localhost:3000/posts
 http://localhost:3000/posts/1
```

12. 测试访问: 使用axios

```
<script src="https://cdn.bootcss.com/axios/0.19.0/axios.js"></script>
<script>
  // 30分钟内不再发预检请求
  // axios.defaults.headers["Access-Control-Max-Age"] = "1800"

  /* 1. GET请求: 从服务器端获取数据*/
  function testGet() {
    // axios.get('http://localhost:3000/posts') // 获取所有posts的数组
    // axios.get('http://localhost:3000/posts/1') // 获取id为1的数组
    // axios.get('http://localhost:3000/posts?id=1&id=2') // 获取id为1或2的数组
    // axios.get('http://localhost:3000/posts?title=json-server&author=typicode')
  }
  testGet()

  /* 2. POST请求: 向服务器端添加新数据*/
  function testPost() {
    // axios.post('http://localhost:3000/comments', {body: 'xxx', postId: 1}) // 保存数
据
  }
  testPost()

  /* 3. PUT请求: 更新服务器端已经存在的数据 */
  function testPut() {
    // axios.put('http://localhost:3000/comments/4', {body: 'yyy', postId: 1})
  }
  testPut()

  /* 4. DELETE请求: 删除服务器端数据 */
  function testDelete() {
    // axios.delete('http://localhost:3000/comments/4')
  }
  testDelete()
</script>
```