爱创课堂前端培训

HTML5

第1天课堂笔记（本课程共1天）

班级：北京前端训练营27期

讲师：蔡久明

日期：2019年10月26日

爱创课堂官网 ：[www.icketang.com](http://www.icketang.com)

# 本地存储

网站的架构：

B/S：browser server

浏览器：html、css、js

服务器：服务器、服务器端脚本语言、数据库

前端也可以进行简单的数据存储:存储的类型只支持字符串，如果想存储数组和对象，使用ES5中的JSON对象

前端存储一共有两大类

本地存储:localStorage 和 sessionStorage

本地数据库:websql 弃用

本地存储localStorage和sessionStorage的区别：生命周期不同

localStorage永久

sessionStorage关闭浏览器就消失（不是关闭页面）

localStorage和sessionStorage都可以跨页面读取信息，但是必须在同一个域名下

## localStorage

本身是一个对象：

|  |
| --- |
|  |

setItem:添加或修改数据

setItem(key, value)

key:属性名

value:属性值

getItem(key)：获得指定的数据

key:属性名

removeItem(key)：清除指定数据

key:属性名

clear():清空数据

|  |
| --- |
| 1. // 添加数据 2. // localStorage.setItem('meinv', '张虹'); 3. // 清除指定的数据 4. // localStorage.removeItem('meinv'); 5. // 获得数据 6. // console.log(localStorage.getItem('shuaige')); 7. // console.log(localStorage.getItem('meinv')); 8. // 清除所有数据 9. // localStorage.clear(); 10. console.log(localStorage); |

## sessionStorage

和localStorage的api一样，只是sessionStorage为临时存储，生命周期结束的时机为浏览器关闭

# 二、服务器

简单来说就是可提供服务的机器（响应）

这个服务器可以租、可以买、可以下载一些模拟软件

我们选择自己搭建的Nodejs

如果想要使用Nodejs,需要下载应用程序

平时我们查看的地址栏：

|  |
| --- |
| 1. file://…… |

有了服务器之后的地址栏：

|  |
| --- |
| 1. http:// …… |

## 2.1HTTP

http:是超文本传输协议，里面规定了前端如何发送请求到后端，后端如何返回数据，前端如何接收的规定

当有了服务器之后，我们可以发送http请求，来请求服务器中的资源

在很久之前，浏览器的目的只是为了渲染静态页面，此时，面临一个问题，前端发送请求，后端如何接收，

后端返回数据，前端如何解析

于是，HTTP就出现了

例如：前端想要请求一张图片到服务器端，此时，要按照HTTP协议中的规定去发送请求，后端接收请求并返回数据的时候要按照HTTP协议中的规定按照正确的格式返回数据

还有一个问题，服务器可连接的数量是有限的，比如一个服务器的连接数量是10， 此时，前端发送过来一个请求，并且还是有状态的（持久连接），服务器的连接数量最多只有10个，但是在当时浏览器只是为了渲染静态页面，所以没有必要是有状态的（持久连接），所以特意给HTTP设置为无状态（没有持久连接）

连接过程：

浏览器端发送HTTP请求，请求服务器中的资源，服务器得到响应，返回数据，断开连接

由于断开了连接，所以服务器可连接的数量远大于有状态的连接

浏览器接收数据，并渲染页面

服务器最大的作用是可以提供服务器环境

在前端很多时候要用到服务器环境

比如：事件推送，多线程，ajax、canvas相关功能都需要使用 服务器环境

## 2.2 如何使用服务器

第一步：

按住shift键 右键单击 选择cmd （在当前位置打开cmd）

|  |
| --- |
|  |

第二步输入命令:

node server.js

出现错误及解决办法：

|  |
| --- |
|  |

正常情况：

|  |
| --- |
|  |

访问的方式：

|  |
| --- |
|  |

实例：

|  |
| --- |
|  |

# 三、webSql

webSql本来是html5的规范，后来被移除

数据库的结构：

数据库（总称）-->库(具体的库)--->表（数据表）--->表头（设置字段--分类名称） + 表格的行(具体的内容)

每一条数据存储为一行

## 3.1 创建（打开）数据库

openDataBase(dbname, version, descript, size)

dbname:数据库名称

version:版本

descript:描述

size:大小 字节

执行代码：

|  |
| --- |
| 1. var db = openDatabase('grade27', 1.0, '27期的数据库', 1024 \* 1024); 2. console.log(db); |

结果：

|  |
| --- |
|  |

## 3.2 transaction

打开一个数据库之后可以获得一个对象，transaction一个方法，该方法可以执行一个回调函数

tr为trancaction执行回调函数注入的实参

tr.executeSql(sql语句，[], successCallback, errorCallBack)

successCallback：成功时执行的回调函数

errorCallBack：失败时执行的回调函数

执行代码：

|  |
| --- |
| 1. // 打开数据库---如果没有则创建，有则打开 2. /\* 3. openDataBase(dbname, version, descript, size) 4. dbname:数据库名称 5. version:版本 6. descript:描述 7. size:大小 字节 8. \*/ 9. var db = openDatabase('grade27', 1.0, '27期的数据库', 1024 \* 1024); 10. // console.log(db); 11. // transaction 12. db.transaction(function(tr) { 13. /\* 14. tr.executeSql(sql语句，[], successCallback, errorCallBack) 15. successCallback：成功时执行的回调函数 16. errorCallBack：失败时执行的回调函数 17. \*/ 18. // console.log(arguments); 19. // console.log(tr); 20. // 通过形参对象tr，执行sql语句（操作数据库的语句） 21. // 如果是创建表，只能创建一次 22. // tr.executeSql('create table class(id int, name char(10))', [], function() { 23. // console.log('建表成功'); 24. // }, function() { 25. // console.log('建表失败'); 26. // }); 27. // 添加数据 28. tr.executeSql('insert into class(id, name) values(11, "zh")', [], function() { 29. console.log('添加成功'); 30. }, function() { 31. console.log('添加失败'); 32. }); 33. }); |

结果：

|  |
| --- |
|  |

# worker

## 4.1 使用worker创建线程

使用方式:

1. 将复杂的js代码或者阻塞线程的代码放入到一个外部文件中（demo.js）
2. new Worker(‘引入外部js文件的路径’)

执行代码：

index.html

|  |
| --- |
| 1. // 开辟一个新的线程 2. var wk = new Worker('out.js'); 3. </script> 4. <button>按钮</button> |

out.js

|  |
| --- |
| 1. // 开辟一个新的线程 2. var wk = new Worker('out.js'); 3. </script> 4. <button>按钮</button> |

结果：

|  |
| --- |
|  |

## 4.2 线程之间的通信

开辟的新线程要想主线程传送消息: postMessage()

主线程接收消息：采用的是事件监听

wk.onmessage = function(e) {

console.log(e.data);

}

执行代码：

主线程:

|  |
| --- |
| 1. <script> 2. // 开辟一个新的线程 3. var wk = new Worker('out.js'); 4. // 接收消息 5. // console.log(wk); 6. wk.onmessage = function(e) { 7. console.log(e.data); 8. } 9. </script> |

开辟的新线程：

|  |
| --- |
| 1. var fibonacci =function(n) { 2. return n <2 ? n : arguments.callee(n -1) +arguments.callee(n -2); 3. }; 4. console.log(this); 5. console.log('计算开始'); 6. var num = fibonacci(40); 7. console.log('计算结束'); 8. // 告诉主线程运算结束，告诉他结果 9. postMessage('久哥真英俊'); |

结果：

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |