复习：

HTML5新特性：

(1)新的语义标签

(2)表单2.0

(3)视频和音频

(4)Canvas绘图

(5)SVG绘图

(6)地理定位

navigator.geolocation.getCurrentPosition(succ, err)

(7)拖放API

源对象可以触发的事件：

dragstart / drag / dragend

目标对象可以触发的事件：

dragenter / dragover / dragleave / drop

(8)WebWorker

(9)WebStorage

(10)WebSocket

今日目标：

(1)作业&扩展小知识

(2)WebWorker

(3)WebStorage

1.如何在拖动的源对象和目标对象间传递数据

**方法1：使用全局变量**

简单；但会污染全局对象

**方法2：使用H5拖放API的数据运输对象——dataTransfer**

H5的拖放API为每个拖放事件都提供了一个e.dataTranfer(拖拉机)属性，专用于在一次拖放过程中的7个事件间传递数据

源对象的事件——保存数据

e.dataTransfer.setData( key, value );

目标对象的事件——获取数据

var value = e.dataTransfer.getData( key );

2.扩展小知识：如何拖动客户端的一张图片显示在服务器端下载的网页中？

|  |
| --- |
| H5中指定的用于文件IO的新对象：  File：表示一个文件/目录对象  FileList：类数组对象  FileReader：文件读取器，用于读取文件内容  FileWriter：文件写出器，用于向文件中写出内容 |

**div.ondrop = function(e){**

var f0 = e.dataTransfer.files[0]; //获取被拖动的文件

var fr = new FileReader(); //创建文件读取器

fr.readAsDataURL(f0); //从文件中读取内容

fr.onload = function(){ //文件内容读取完成

fr.result; //读取到的文件数据

}

**}**

课下思考题：实现异步文件上传，带进度条显示

3.面试题：请谈一下Chrome浏览器的线程模型？

Chrome浏览器是如何下载网页资源？

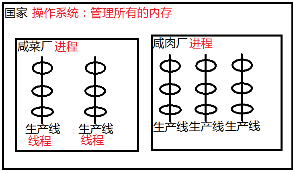
操作系统相关背景知识：

**程序**：可供CPU执行的代码，存储在磁盘上

**进程**：将程序从磁盘调入到内存，准备执行，称为进程

**线程**：Thread，是进程内部执行代码的基本单位

**多线程并发**：为了同时执行多个任务，在每个进程中可能同时存在多个线程。这些线程是“并发执行”的——宏观上看是“同时”执行，微观上看是“依次轮询”执行



进程和线程的关系：

(1)进程是操作系统分配内存的基本单位

(2)线程是执行代码的基本单位——由CPU调度执行线程中的代码

(3)线程处于进程内部；一个进程中至少要有一个线程，也可有多个同时存在的线程

Chrome浏览器中的线程模型：

每个Chrome进程中至少有6个线程。

**为了保证浏览器请求网页内容的速度，Chrome会同时启用这6个线程——请求线程**。

**而执行代码(HTML/CSS/JS/图片渲染...)仅有一个线程来完成——UI主线程**。

测试：看下面的代码执行的效果

<button onclick="console.log(111)">

<script>执行一个很耗时的JS代码</script>

<button onclick="console.log(222)">

4.HTML5新特性之八 —— WebWorker

若Web页面中执行了很耗时的JS任务（如加解密、复杂路径计算、多参数计算等），由于所有的HTML/CSS/JS代码都在同一个UI主线程中执行，会导致页面“卡死”。

根本解决办法：创建一个新的并发线程，在新的线程中执行耗时JS任务，让UI主线程继续渲染页面/监听事件。

var t = new Thread('x.js') //其它语言创建新线程的语句

var w = new Worker('x.js'); //H5中创建的新“线程”的语句

注意：Worker线程不同于普通线程的天然限制！！！

**Worker线程不允许使用任何的BOM和DOM对象！**——浏览器只允许UI主线程渲染页面，操作DOM元素！

Worker线程若想从DOM树上读取数据、或把处理结果显示在DOM树上，不能直接操作DOM元素；但可以给UI主线程发消息，让UI主线程来操作。

**UI主线程给Worker线程发消息：**

UI主线程：

var w = new Worker('x.js');

w.postMessage('stringMsg');

Worker线程：

onmessage = function(e){ e.data }

**Worker线程给UI主线程发消息：**

UI主线程：

var w = new Worker('x.js');

w.onmessage = function(e){ e.data }

Worker线程：

postMessage('stringMsg');

练习：在页面中有两个输入框，点击按钮后开始计算这输入框中的数字的和——假设非常耗时！必须放在Worker线程中！window.alert()出最终的计算结果.

提示：可以在一个消息中传递两个数字： "123-789"

5.Web项目中的数据存储的方式

**(1)业务数据——**如商品信息、订单、帖子、新闻、出入账

1)服务器端的数据库中；

2)服务器端的内存中；

**(2)用户数据——**如浏览历史、偏好设定、登录记录

1)Cookie存储：浏览器兼容性好，不能超过4KB，操作复杂

2)H5 WebStorage：不能超过8MB，操作简单

3)Flash 存储：依赖于Flash播放器

4)IndexedDB：目前还不是HTML5标准技术

|  |
| --- |
| Web会话：浏览器从第一次打开某个网站的页面开始，然后不停的在此站点中的页面中跳转（很多请求-响应过程），直到最后关闭浏览器——整个过程称为“浏览器与Web服务器间的一次会话”。 |

6.HTML5新特性之九 —— Web Storage

WebStorage：H5提供的客户端用户数据存储技术，分为两个对象：

**window.sessionStorage：**会话存储，浏览器在自己的进程内存中存储的当前用户的访问数据，可以供当前网站的所有页面使用。一旦会话结束(浏览器关闭)，这些数据全部会被删除。

sessionStorage.setItem(k, v) //保存k-v对数据

sessionStorage[k] = v //保存k-v对数据

var v = sessionStorage.getItem(k) //获取k对应的v

var v = sessionStorage[ k ] //获取k对应的v

sessionStorage.removeItem(k) //删除一个k-v对

sessionStorage.clear() //清除所有的k-v

sessionStorage.length

var k = sessionStorage.key( i ) //获取第i个key

**window.localStorage**：本地存储，跨会话存储，浏览器在当前文件系统中存储的当前用户的访问数据，可以供当前网站的所有页面使用。即使浏览器关闭甚至操作系统重启，这些数据也不会被删除，除非手工删除。

localStorage.setItem(k, v) //保存k-v对数据

localStorage [k] = v //保存k-v对数据

var v = localStorage.getItem(k) //获取k对应的v

var v = localStorage [ k ] //获取k对应的v

localStorage.removeItem(k) //删除一个k-v对

localStorage.clear() //清除所有的k-v

localStorage.length

var k = localStorage.key( i ) //获取第i个key

练习： 保存用户登录信息 —— sessionStorage

(1)创建index.html，右上角显示超链接

<a href="login.html">请登录</a>

(2)创建login.html，用户名和密码输入域，点击登录，假设用户输入都有效，alert("登录成功! 3s后自动跳转回首页")

setTimeout + location.href=''

(3)修改index.html，右上角变为：

欢迎回来：xxx，<a href="logout.html">退出</a>

(4)创建logout.html，删除用户的所有登录信息，提示登录退出成功！3s后自动跳转回首页

(5)再回到首页，右上角显示“请登录”超链接

练习：用户自定义网站样式 —— localStorage

(1)创建index.html，编写三个CSS: class

.blue { color: '#33a' background: '#eef' }

.green { color:'#3a3' background: '#efe'}

.pink { color: '#a3a' background: '#fef'}

页面中提供下拉列表：深邃蓝色主题、清新绿色主题、粉色公主主题，用户选择某个主题后，当前页面body的class修改为对应的值。

(2)用户可以跳转到usercenter.html，此页面中也会自动呈现出上一页面用户选择的主题色；

注意：用户关闭浏览器，重新打开任一页面，仍然显示出之前选择的主题色

localStorage中的数据若发生了改变，会触发window.onstorage事件，可以利用该事件通知当前打开的其它窗口localStorage数据改变了 —— sessionStorage数据不需要跨窗口使用，不会触发该事件。

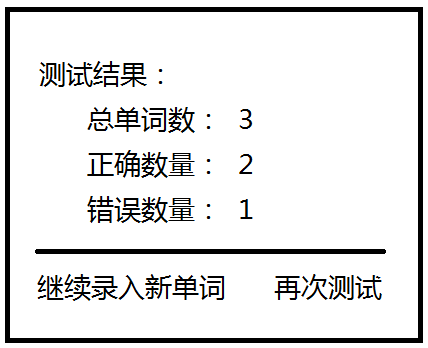
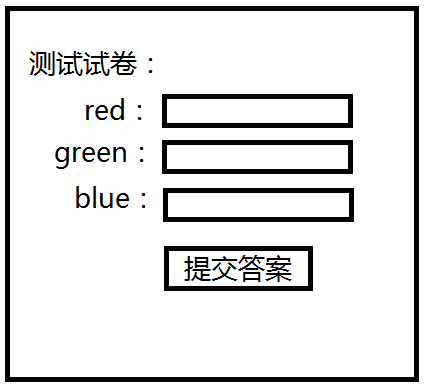
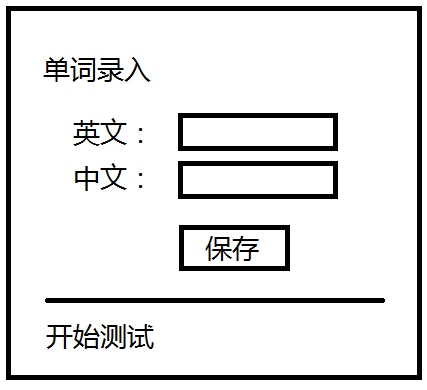
课后练习： 单词测试系统 18:45

(1)用户可以在save.html中不停的录入新单词；

(2)进入test.html开始测试，需要对之前录入过的所有单词进行测试；

(3)提交答案后，在result.html中显示出测试成绩。

提示：录入的单词需要永久保存；而此次测试结果只需要在当前会话中保存。



//遍历客户端存储的数据

for(var i=0; i<localStorage.length; i++){

var key = localStorage.key(i) //获取第i个key

}