第一阶段HTML与CSS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*7月29日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 第一章HTML

### WEB基础知识

1、WEB与Internet

**1、Internet**

互联网

若干台计算机通过网线所连接而成的物理设备

主要服务：

1、Telnet

2、Email

3、WWW(WordWideWeb)

4、BBS

5、FTP

信息共享

基本实现技术：

1、TCP/IP协议簇

2、分组交换原理

**2、WEB**

1、什么是WEB

网页应用

2、主要作用

将各类信息(文本，图像，音频，视频)和服务进行连接

3、网站与网页

4、WEB的工作原理

1、基于B/S结构模式的程序

B/S:Browser/Server，基于浏览器/服务器

C/S:Client/Server,如：QQ，游戏

2、组成

1、服务器

WEB中的所有的信息都保存在服务器上

2、浏览器(UA:UserAgent)

3、通信协议

http:hypertexttransferProtocol超级文本传输协议

功能：规范了数据是如何打包的以及如何传递的。

http://www.baidu.com

http://www.baidu.com

5、WEB服务器

1、功能

1、存储WEB上的信息

2、处理用户的请求，执行服务器端程序，并且给出响应

3、安全功能

2、主要产品

1、TOMCAT(APACHE)

2、IIS(InternetInformationService,MS)

3、APACHE

3、技术

1、php

2、jsp

3、asp

4、asp.net

6、浏览器

1、功能

1、代理用户提交请求

2、作为html解释器以及内嵌脚本(js)程序的执行器

3、以图形化方式显示文档内容

2、产品

1、IE

2、Firefox

3、Chrome

4、Opera(欧朋)

5、Safari

3、技术

1、HTML:网页技术

2、CSS：样式表

3、JavaScript：脚本

2、HTML快速入门

1、什么是HTML

HyperTextMarkupLanguage

超级文本标记语言

该语言所组成的文件以.html或.htm作为后缀

2、HTML基础语法

1、标记语法

1、什么是标记

在html中用于描述功能的符号称之为"标记"

<a>:超链接

<b>:加粗显示文本

2、语法

使用时，必须使用尖括号(<>)

分类：

1、封闭类型

也叫双标记，必须成对出现

<标记>内容</标记>

<b>加粗显示</b>

注意：

1、双标记必须成对出现

<b>

2、

<b>

fdsafdsa

</b>

2、非封闭类型

又称为：空标记或单标记

语法：<标记>或<标记/>

<img>:

<img/>

<img></img>错误

2、元素

元素，即标记

<a></a>

<a>百度</a>

元素的嵌套

<标记>

<标记></标记>

</标记>

注意：

1、注意嵌套顺序

<div><a></a></div>正确

<div><a>...</div></a>

2、用缩进表示层级关系

<div>

<a></a>

</div>

<div>

<a>

<b>

<u>.....</u>

</b>

</a>

</div>

3、属性和值

1、什么是属性

属性是用来修饰元素的

<p></p>

2、语法

1、属性声明必须位于开始标记中

2、允许出现多属性，多个属性之间用空格来分开，排名不分先后

值：

属性和属性值之间用=连接

属性="值"

属性='值'

<标记属性=值属性=值></标记>

<标记属性=值属性=值>

<palign="center">...</p>

3、标准属性

1、id

作用：定义元素在页面中的独一无二的标识

2、title

作用：鼠标移入时所提示的文字

3、class

作用：引用类选择器

4、style

作用：定义内联样式

ex：定义一个p元素，设置其id为p1，并且鼠标移入的时候，提示这是一个p

<pid="p1"title="这是一个p"></p>

4、注释

1、作用

被注释的东西是不会被浏览器所解释的。

<!---->

2、注意

1、注释不能被嵌套

<!--

<!--

-->

-->

错误

2、注释不能出现在标记中

<p<!---->></p>

错误

3、HTML版本

1、1999.12.24HTML4.01

<img>

<img/>

<palign="center"></p>

2、2000.1.26XHTML1.0

规范与html4.01几乎相同

是更严格更纯净的HTML版本

<img/>

3、HTML5

目标：更为简洁的HTML代码

<palign="center"></p>

<palign='center'></p>

<palign=center></p>

<inputreadonly>

4、文档结构

文档类型声明

作用：指定网页版本和风格

html5:

<!doctypehtml>

w3c:万维网联盟

html页面

由一对html根标记组成

<html></html>

练习

1、创建一个html文件

步骤：

1、创建文本文件

2、将后缀名改为html

2、搭建网页结构

步骤：

1、添加文档类型声明

2、创建html根标记

5、<html>元素

1、作用

描述整个html网页内容的

2、2个子元素

1、head元素

1、作用

用于定义页面"全局信息"

2、语法

<head></head>

3、可包含

1、<title></title>

作用：网页标题

2、<metacharset="utf-8">

3、<meta>

4、<style>

作用：定义内部样式表

5、<link>

作用：引入外部样式表

6、<script>

作用：声明或引入JS代码(文件)

2、body元素

1、作用

显示的主体内容

2、语法

<body></body>

3、文本(标记)

1、特殊字符

网页中的空格，<>页面中的表现方式

通过转义字符表示特殊字符

1、&nbsp; 空格

2、&lt; lessthan（小于）<

3、&gt; greaterthan（大于）>

4、&copy; copyright ©（版权）

5、&yen; ￥

2、文本标记

1、文本样式标记

<b></b>：加粗

<i></i>：斜体

<s></s>：删除线

<u></u>：下划线

<sub></sub>：下标

<sup></sup>：上标

2、标题元素

语法：

<hn></hn>

n:1~6即1级标题~6级标题

效果：

改变文字大小，加粗，以及上下垂直空白距离

作用：突出显示文本

3、段落元素

语法：<p></p>

显示效果：上下有垂直空白

属性：

align：控制段落，标题的文本的水平对齐方式

4、换行元素

语法：<br>或<br/>

5、块分区元素

语法：<div></div>层

效果：独占一行

作用：做布局

6、行内分区元素

语法：<span></span>

作用：显示文本的不同效果

7、分割线元素

语法：<hr>

作用：显示一条水平线

属性：

1、size：尺寸

2、width：宽度

默认100%

3、align

4、color：颜色

8、预格式化

语法：<pre></pre>

作用：保留源文档中的空格和换行

9、行内元素与块级元素

**1、块级元素**

**特点：独占一行，即元素的前后都会自动换行**

**如：div,p,pre,h1~h6,......**

作用：都能做布局

注意：

**所有的块级标记都具备align属性**

**2、行内元素**

**特点：不会换行，多个元素可在一行内显示**

**作用：包裹文本设置文本样式**

如：span,i,u,s,b,sub,sup......

注意：、

**行内元素没有align属性**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月1号\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 图像和链接URL目录结构

文件夹

WEB站点中保存资源文件夹的名称

2、URL（uniform Resourse Locator）

统一资源定位器 俗称：路径

功能：用来标识网络中的资源的位置

1. URL的分类

1、绝度路径

1、网络资源

网络资源：协议名、主机名（域名、IP地址）、目录路径、资源名称

2、本机资源

从文件 所在的最高级目录想爱开始查找

1. 相对路径

什么是相对路径

相当于从当前文件所在的位置出发，找到资源文件 所经过的所有路径 就是 相对路径

1、同目录直接用

2、子目录，先进入，在引用

3、父目录，先返回，在引用

../:返回上级目录

3、根相对路径

作用：永远都是从 WEB站点的根目录下开始查找

表示：/ 作为开始

ex : /img/index/jd\_logo.png

1. 图像

1、图像元素

<img/>

属性：

1、src

要显示的图像的路经（URL）

路径 严格区分大小写（服务器用）

1. Width

图像宽度

1. Heigth

图像的高度

以上两属性 可以取数值，以PX作为单位，px可以省略不写

以上两个属性如果只声明一个属性的话，那么另一个会等比缩放

1. title

链接

1. 链接元素

链接：又叫超链接，用户点击时能够发生页面的跳转等其他操作

1. 语法

<a>内容</a>

1. 属性
2. href 属性

链接 URL

注意：只有设置了href属性后，元素才能被点击

1. target 属性

目标，打开新网页的方式

取值：

1、\_self

在自身标签页中打开

默认值

2、\_blank

在新标签页中打开

1. 链接的表现形式

1、，目标文档为下载资源

<a href=”\*\*\*.zip/rar”></a>

1. 电子邮件链接

联系我们

前提：计算机中必须先安装邮件客户端以及绑定账号

<a href=”mailto:邮电地址”>\*\*\*</a>

1. 链接到javascript

<a href=”javascript:js操作代码”>\*\*\*</a>

锚点

1. 作用

某个位置处做记号

1. 锚点使用方式
2. 定义锚点
3. a元素的name属性定义

<a name=”名称”>\*\*\*\*</a>

1. 任何一个元素的id属性

<标签id=”值”></标签>

1. **链接到锚点处**
2. 链接本页锚点
3. <a href="#锚点名称">\*\*\*</a>

链接到其他页面的锚点

<a href="页面url#锚点名称"></a>

表格

由“单元格”按照从左到右，从上到下的方式排列

1. 表格的作用

按照一定的格式来显示数据

1. 创建表格

定义表格<table ></table>

创建表行：<tr></tr>

创建单元格列：<td></td>

注意：保证每行的列数都是一致的

1. 表格属性

1、width:宽度

2、height:高度

3、align:

作用：设置表格元素的水平对齐方式

取值：left/right/center

4、border:边框宽度

5、cellpadding

内边距一单元格边框与内容的距离

1. Cellspacing

外边框一单元格之间的距离

1. bgcolor:背景颜色

Tr属性

1. Align

当前行文本 水平对齐方式

1. Valign

当前行的文本 垂直对齐方式

3、bgcolor

当前行的背景颜色

Td属性

1. Width
2. Height
3. Align
4. Valign
5. **Colspan:设置单元格跨列**
6. **Rowspan:设置单元格跨行**

table的子元素

1. <caption></caption>

作用定义表格的标题

2、行标题/列标题

显示效果：水平居中，并且加粗显示

<td></td><th></th>

**表格的复杂应用**

1、行分组

将数据 分成不同的组 以便设置样式

1. 表格行分组,表头

<thead></thead>

2、表主体行分组

<tbody></tbody>

1. 表尾行分组

<tfoot></tfoot>

注意：如果不设置分组的话，那么所有的行默认tbody中的元素

不规则表格

通过td的colspan,rowspan属性

1. Colspan

跨行，合并

横向向右合并几个单元格包括自己

**注意：**被合并的td，一定要删除

2、rowspan

跨行，合并行

纵向向下 合并几个单元格(包含自己)，并删除合并

3、表格的嵌套

表格可以嵌套其他的表格

被嵌套的表格 要 出现在 <td></td>中

3、列表

1、作用

按照一定的结构显示数据

将 一组数据按照从上到下的顺序来进行排列

2、列表的组成

由 列表类型 （有序和无序 ）和 列表项

3、语法

1、有序列表

标记：<ol></ol> --> Order List

列表项：<li></li> --> List Item

属性：

1、type

列表项前的列表类型

取值：

1、1 数字(默认值)

2、a 小写字母

3、A 大写字母

4、i 小写罗马字符

5、I 大写罗马字符

2、start

列表项前的标识，从第几个字符开始显示

2、无序列表

标记：<ul></ul> -> Unorder List

列表项：<li></li>

属性：

1、type

取值：

1、disc ，实心圆(默认值)

2、circle ，空心圆

3、square，实心矩形

4、none，不显示列表项

3、嵌套列表

在 <li>中，可以嵌套另一个列表

4、定义列表

1、作用

用于要给出一类事物的定义情形，比如：名词

2、语法

<dl></dl> : 定义 一个定义列表

<dt></dt> : 定义 要描述的名词术语（情形）

<dd></dd> : 定义 名词术语的解释

3、使用场合

图文混排

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月2日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 结构标记

1、作用

通过一组特殊的标记来完成 网页的布局

目的：提升语义性

2、结构标记

1、<header></header>

作用：定义网页或页面某部分的头部显示内容

替换：<div id="header"></div>

2、<nav></nav>

作用：在页面中，负责定义导航部分内容

替换：<div id="nav">

3、<section></section>

作用：表示页面中的主体内容

替换：<div id="main"></div>

4、<article></article>

作用：显示 论坛中的帖子、具体的新闻、博客的条目、用户评论、微博信息等 ....

5、<aside></aside>

作用：表示边栏信息

替换：<div id="left\_side"></div>

6、<footer></footer>

作用：定义页面最底部的信息，用户关注度不太高的

替换：<div id="footer"></div>

2、**表单(重难点)**

1、作用

表用用于收集、显示用户信息，并且将信息提交给服务器

2、表单组成

1、表单元素

2、表单控件

3、表单元素

1、定义表单

<form></form>

2、属性

1、action

表单提交时的动作

指定服务器端处理程序的地址

通常都是有服务端开发人员提供

默认为提交给本页

2、method

表单数据的提交方式

取值：

1、get

1、显示提交：会将表单中所有的提交信息显示在地址栏上

2、安全性较低

3、大小限制：2KB

4、场合：向服务器要数据的时候使用 get 方式,根据指定关键字 进行搜索时

2、post

1、安全性较高

所提交的数据不会显示在任何地方上

注册信息或者涉及到密码，卡号等 一定要用post

2、无长度限制

3、场合：传递数据给服务器去处理使用

默认值为 get

3、enctype

作用：指定表达数据的编码方式，表单中的那些数据可以提交给服务器

取值：

1、**application/x-www-form-urlencoded**

默认值，允许将普通字符和特殊字符提交给服务器，文件不行

2、multipart/form-data

特点：允许提交 文件,会影响普通上传数据

3、text/plain

特点：只允许进行普通字符的提交，特殊字符无法提交

如：?,=,&

4、name

表单名称，JS允许使用name属性获取表单

4、表单控件

1、作用

1、接收用户数据，与用户交互

2、提供了可视化的外观

2、表单控件分类

1、**input 元素 行内元素**

提供了 ：文本输入(文本，密码)，按钮，单选按钮，复选框，文件选择框，隐藏域

1、语法

<input>

2、属性

1、type(重点)

根据不同的type属性值，创建各种类型的输入控件。

注意：如果不写 type值或者 写错了 type值得话，那么默认就是文本框

2、value

提交给服务器的值

3、name

控件的名称,服务器使用

注意：如果不设置name属性的话，数据是无法提交给服务器的

4、disabled

禁用

注意：该属性 无值

3、具体表单控件元素

1、文本框与密码框

文本框:type="text"

密码框:type="password"

缩写：txt

属性：

1、name

规范：匈牙利命名法

由控件缩写+控件作用组成

2、value

控件的值，以及默认显示的默认值

3、maxlength

限制输入的字符数

4、readonly

只读

不需要设置值

2、单选按钮和复选框

单选按钮：type="radio"

缩写：rdo

特点：一组中，只能有一个被选中。name值相同则为一组。

复选框：type="checkbox"

缩写：chk

属性：

1、name ：设置名称，分组。

2、value ：尽量设置

3、checked ：设置默认被选中

3、按钮组

1、提交按钮

type="submit"

2、重置按钮

type="reset"

3、普通按钮

type="button"

属性：

1、name

2、value

按钮上显示的文字

4、隐藏域和文件选择框

1、隐藏域

type="hidden"

缩写：txt

作用：将不想展示给用户，但要提交给服务器的数据 保存在隐藏域中

2、文件选择框

type="file"

name必不可少

缩写：txt

注意：

1 、method 必须为 post

2、enctype 必须为 multipart/form-data

2、textarea 多行文本域

1、语法

<textarea>文本</textarea>

2、属性

1、name

当前表单元素的名称

2、cols

指定文本域的列数

数字(英文状态下允许显示的列数)

3、rows

指定文本域的行数

4、readonly：只读

3**、select和option 选项框 行内元素**

1、表现方式

1、下拉选项框

2、滚动列表

2、标签

1、<select></select>

创建 选项框

属性：

1、name

缩写：sel

2、size

值>1的话，则为滚动列表，否则就是下拉选项框

3、multiple

设置多选

2、<option></option>

选项框中的选项

属性：

1、value

选项的值

2、selected

预选中

4、其他控件

1、为控件分组

1、标记

**<fieldset>**

**<legend></legend>作用：为控件分组**

**</fieldset>**

作用：为分组指定标题

2、**<label>元素 行内元素**

作用：关联 文本 与 表单元素。点击文本 就如同点击 所关联的表单元素

语法：

<label>文本</label>

属性：

for：表示与该元素相关的 表单控件的ID值

3、其他标记

1、浮动框架

1、什么是浮动框架

可以在一个浏览器窗口中同时显示多个页面文档(一个浏览器中显示多个网页)

2、语法

<iframe>您的浏览器不支持浮动框架</iframe>

3、属性

1、src

浮动框架中的网页URL

2、width

浮动框架的宽度

3、height

浮动框架的高度

4、frameborder

浮动框架边框

设置为0，则没有边框

2、摘要与细节

1、作用

允许用户动态的展开 或 收缩 页面某部分内容

2、语法

<details>

<summary>标题</summary>

内容1

内容2

</details>

3、度量元素

1、作用

在页面中定义度量衡

多数用于表示数据的比例

2、语法

<meter>文本</meter>

3、属性

1、min

定义度量范围的最小值，默认为0

2、max

定义度量范围的最大值，默认为1

通常 改成 100

3、value

度量值，显示在度量元素上的值。默认为 0

4、时间元素

1、作用

关联时间的不同表现形式

2、语法

<time>文本</time>

3、属性

1、datetime

规定 日期与时间 ，日期与时间用T分割

5、高亮文本显示

语法：

<mark>文本</mark>

第二章CSS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月3日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 1、CSS概述

1、问题

1、相同的效果用不同的属性

<body text="blue">

<font color="red">

2、可维护性和可重用性不高

**2、CSS**

1、什么是CSS

Cascading Style Sheets

层叠样式表、级联样式表、样式表

2、作用

1、实现 内容与样式 相分离

2、提高代码的可重用性和可维护性

3、CSS与HTML之间的关系

HTML ：人 - 搭建网页结构

CSS ：衣服装扮 - 构建网页样子

4、CSS与HTML使用原则

尽量使用css属性 取代 HTML 属性

能取代：

Html ：align

css ：text-align

html：color

css：color

**不能取代：**

colspan

rowspan

2、使用样式表

3、使用CSS样式表

1、使用样式表方式

**1、内联方式**

将 样式定义在 元素的style属性中

语法：

<标记 style="样式声明;样式声明"></标记>

样式声明：样式属性:样式值

ex:

color:red; 当前元素文本为红色

background-color:blue;背景颜色

font-size:20px;字体大小

弊端：

1、没有体现出 可重用性和可维护性

**2、内部样式表**

1、what

将样式定义在 head 元素中的 style 元素中

2、语法

<head>

<style>

若干"样式规则"

</style>

</head>

样式规则组成：

1、选择器

规定了页面中[哪些]元素能够使用定义好的样式

2、若干样式声明

样式属性:值;

选择器{

样式属性:值;

样式属性:值;

}

h1{

color:red;

font-size:12px;

}

特点：

1、提升了样式的可重用性和可维护性

**3、外部样式表**

1 、作用

将样式声明在独立的样式文件中(\*\*.css),并且独立于任何HTML文档

2、步骤

1、创建css文件并声明样式规则

2、在页面中对 css 文件进行引入

<head>

<link rel="stylesheet" href="css样式表文件url">

</head>

2、CSS样式表特征

1、继承性

大部分的CSS样式属性是可以被继承

2、层叠性

可以为一个元素定义多个样式规则

样式属性不冲突时，可以将多个样式规则中的属性 层叠为一个

3、优先级

样式属性定义冲突时，按照不同样式表或样式规则的优先级来应用样式

浏览器缺省设置(UA) 低

外部样式表与内部样式表 中

-就近原则 ：后定义的优先

内联样式 高

4、!important 规则

1、作用

调整样式属性的优先级

2、用法

选择器{属性:值 !important;}

3、建议

少用

1、IE6 及一下了浏览器不支持

2、打破了原有优先级规则

5、排错

1、Unknown property name

属性名称 写错了

2、Invalid property value

属性值 写错

3、选择器写错

什么样式都没有

**2、CSS语法(重点)**

1、选择器

1、作用

规范页面中哪些元素能够使用定义好的样式

匹配页面元素

2、详解

1、通用选择器

1、作用

匹配页面所有的元素

2、语法

\*{}

2、元素选择器

1、作用

匹配 页面中指定的元素的元素

2、语法

元素名{}

div{}

p{}

h1{}

span{}

3、类选择器

1、作用

允许元素使用附带的class属性进行引用

2、语法

.类名{}

**注意：**不能以数字开始

引用

<标记 class="类名"></标记>

3、多类选择器

允许一个元素 引用 多个类选择器

<标记 class="类名1 类名2"></标记>

4、分类选择器

将类选择器和元素选择器结合使用，以便实现对某种元素中不同样式的细分控制

语法：元素选择器.类选择器{}

.redBack{} //匹配页面中所有class 为 redBack 的元素

div.redBack{}//匹配页面中所有class为redBack的div元素

4、id选择器

1、作用

只匹配页面指定id值得元素

2、语法

#idValue{}

3、ex

<div id="d1"></div>

<div id="d2"></div>

#d1{color:red;}

5、群组选择器

1、what

声明时，以 逗号 隔开的选择器列表

2、语法

选择器1,选择器2,选择器3,..{

}

6、后代选择器

1、作用

根据元素间的后代(出现在该元素中的所有元素)关系来匹配元素

2、语法

选择器1 选择器2{}

7、子代选择器

1、作用

根据元素间的子代(一层层级关系)关系来匹配元素

2、语法

选择器1>选择器2{}

8、伪类选择器

1、作用

匹配元素不同状态时的样式

2、分类

1、链接伪类

2、动态伪类

3、目标伪类

4、元素状态伪类

5、结构伪类

6、否定伪类

3、语法

:伪类{}

选择器:伪类选择器{}

1、链接伪类

1、:link 适用于尚未访问的超链接

2、visited 适用于访问过的超链接

2、动态伪类

1、:hover 适用于鼠标悬停在HTML元素时

2、:active 适用于元素被激活时的状态

3、:focus 适用于html元素获取焦点时的状态(text,password)

3、优先级

选择器 是通过 权值 表示优先级

元素选择器 1

类选择器 10

伪类选择器 10

ID选择器 100

内联样式 1000

选择器冲突时，将权值加到一起，大的优先。权值相同时，后定义优先

3、尺寸与边框

1、CSS单位

1、尺寸单位

% : 相对单位,属性值会占据父层容器对应属性的 百分比

<div style="width:500px;">

<p style="width:50%;"></p>

</div>

in : 英寸 1in=2.54cm

cm : 厘米

mm : 毫米

pt : 磅 point 1pt=1/72in,多数用于 字体大小

px : 像素(计算机屏幕上的一个点)

em : 1em当前字体尺寸，2em，当前字体尺寸2倍

注意：有些属性值是有单位的，单位不能省略。0除外。

2、颜色单位

1、rgb(r,g,b)

r:red 0-255

g:green 0-255

b:blue 0-255

color:rgb(0,0,0); 黑色

background-color:rgb(255,255,255); 白色

2、rgb(r%,g%,b%)

3、#rrggbb

由6位16进制的数字组成的一种颜色

每位由 0-9 A-F 组成

#000000 : 黑色

#111111 ：深灰色

#EEEEEE : 浅灰色

#FFFFFF : 白色

color:#E64916;

4、#rgb

简写方式，每两位数字相同时，可用该方式表示

#000000 -> #000;

#ff55ee -> #f5e;

#fe55cd无替换

5、表示英文颜色的单词

6、rgba(r,g,b,alpha)

alpha : 透明度，0-1之间的数字

0 完全透明

1 完全不透明

rgba(255,0,0,0.5);

**2、尺寸属性**

1、作用

主要用于设置元素的宽和高

2、属性

1、宽度

width

max-width

min-width

2、高度

height

max-height

min-height

**3、哪些元素可以设置尺寸属性**

**1、块级元素 可以设置**

**div,p,h1~h6,ul,ol**

**2、行内块**

**input**

**type = text ， password ...**

**3、行内元素**

**不能修改尺寸**

4、本身具备 width 和 height 属性的 html元素 可以修改

**<img><table> img行内块元素 table像块元素，但是不是块元素，是table类型**

4、溢出处理

1、什么是溢出

使用尺寸属性限制元素大小时，如果内容所需空间大于元素大小本身，会导致内容溢出

2、属性

overflow

overflow-x : 横向溢出处理

overflow-y : 纵向溢出处理

取值：

1、visible

默认值，溢出可见

2、hidden

溢出隐藏

3、scroll

右边，底边出现滚动条，溢出时可用

4、auto

自动，溢出时显示滚动条并且可用

3、边框属性

1、边框属性

1、简写方式

border:width style color;

width : 宽度 ，取值px的数值

style : 边框线条样式

取值：

1、solid：实线

2、dotted ：虚线

3、dashed ：虚线

color : 边框线条颜色

作用：元素的 上下左右 四条边的 宽度，样式，颜色

border:1px solid black;

2、单边定义

1、语法

border-方向:width style color;

方向：

top : 上

right : 右

bottom : 下

left : 左

3、单属性定义

1、语法

border-属性:值;

属性：

1、width ：宽度

2、style ：样式

3、color ：颜色

2、作用

设置四条边框某一属性值

4、单边单属性定义

1、语法

border-方向-属性:值;

2、ex

border-top-color :上边框颜色

border-left-style:左边框样式

5、注意

1、边框颜色 可取值为 transparent(透明色)

2、边框特殊用法

border:none;

或

border:0;

2、边框倒角

1、什么是倒角

将边框四个直角 倒成 圆角

2、属性

border-radius

取值：

1个值 ：设置四个角的倒角半径

可以是 绝对数值 或 百分比

单角定义:

border-top-left-radius:左上角

border-top-right-raidus:右上角

border-bottom-right-radius:右下角

border-bottom-left-radius:左下角

3、边框阴影

1、属性

box-shadow

2、取值

是由多个值所组成的值列表，多个值之间用 空格 隔开

1、h-shadow : 必须，阴影的水平偏移距离，取值为正，向右偏，取值为负，则向左偏

2、v-shadow : 必须，阴影的垂直偏移距离，取值为正，向下偏，取值为负，则向上偏

3、blur : 可选，模糊距离

4、spread : 可选，阴影尺寸

5、color : 可选，阴影颜色

6、inset : 可选值，将默认的外部阴影改为内部阴影

3、常用模式

box-shadow:0 0 blur color;

4、轮廓

1 、属性

outline:width style color;

outline-width:

outline-style:

outline-color:

2、常用方式

outline:0;或outline:none;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月4日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**1、框模型**

1、框模型

1、框页面元素皆为"框"

2、框模型

框模型(Box Model)定义了元素框处理元素内容、内边距、外边距以及边框的方式

width和height 指定内容区域的宽和高

增加了内外边距、边框 不会影响内容区域的尺寸，有可能会增加元素框的总尺寸

元素实际宽度=左右外边距 + 左右边框 +左右内边距 + width;

元素实际高度=上下外边距 + 上下边框 +上下内边距 + height;

2、外边距

1、什么是外边距

围绕在元素边框周围的空白区域，默认是不允许被其他元素所占据的。

2、作用

主要拉开元素与元素之间的距离

3、语法

1、属性

margin:value;

单边设置

margin-top/right/bottom/left:value;

取值单位：

1、px 值

2、%

3、auto ：左右方向有效

4、负值 ：目的为移动元素

2、取负值

让元素向反方向移动

通常只设置

margin-top : 负值

margin-left: 负值

**3、margin：auto**

**能够自动计算外边距(左右)**

**特点：控制非 行内元素 水平居中时使用**

**前提：必须设置元素宽度**

4、外边距的简便写法

margin:value;(四个方向外边距相同)

margin:v1 v2;v1 上下外边距 v2 左右外边距

margin:v1 v2 v3;v1 上外边距，v2 左右外边距，v3 下外边距

margin:v1 v2 v3 v4;上 右 下 左

5、有默认外边距的元素

body,h1,h2,h3,h4,h5,h6,p,ul,ol,pre,dl,dd

具备默认外边距

通过 css重写(CSS Reset) 方式，改变默认外边距为 0

6、特殊情况

1、外边距合并

当两个垂直外边距相遇时，它们将形成一个外边距

合并后的外边距的高度等于两个外边距中高度较大者的值

2、外边距溢出

子债父还

在某些特定情况下，给子元素设置外边距时，效果会作用在父元素上。

特定情况：

1、上下外边距 能产生溢出

2、父元素无边框，并且设置的是第一个元素的上外边距或最后一个元素的下外边距

解决方案：

1、为父元素增加边框

2、由父元素的内边距来取代子元素的外边距

4、注意

上下外边距 对行内元素 是不起作用的

上下外边距 对行内块元素 有特殊效果

3、内边距

1、什么是内边距

内容区域与边框之间的距离

会扩大元素边框所占的区域

2、语法

1、属性

padding:value;

取值 可以为 像素px，百分比%

padding-top/right/bottom/left:val;

2、简写方式

padding:value;(四个方向内边距)

padding:v1 v2;(上下左右)

padding:v1 v2 v3;(上 左右 下)

padding:v1 v2 v3 v4(上右下左)

3、注意

内边距的效果只能影响元素本身，并不会给其他元素带来影响效果

2、背景属性

1、作用

背景颜色以及背景图像

注意：背景是从边框的位置处开始绘制的

2、背景属性

1、背景色

语法：background-color:value;

取值：

任意合法颜色值

transparent

2、背景图片

语法:background-image

取值:url(图片URL);

3、背景重复

语法:background-repeat

取值:

1、repeat ：默认值，即垂直又水平方向的重复

2、no-repeat : 不重复

3、repeat-x : 仅在水平方向重复

4、repeat-y ：仅在垂直方向重复

4、背景图片尺寸

语法：background-size

取值：

1、value1 value2

宽度 高度

2、value1% value2%

百分比

3、cover

覆盖，将背景图像等比缩放，直到背景能够覆盖到元素的所有区域

4、contain

包含，将背景图像等比缩放，直到背景图像碰到右边或下边时，停止缩放。

5、背景图片固定

语法：background-attachment

取值：

1、scroll，默认值，背景图会随着文档而滚动

2、fixed，背景图像固定，背景图不会随着文档而发生滚动

6、背景定位

1、作用

改变背景图像在元素中的位置

2、语法

属性：background-position

取值：

1、x y

x 表示 水平偏移距离,取值为正，图像向右偏移，取值为负，图像向左偏移

y 表示 垂直偏移距离，取值为正，图像向下偏移，取值为负，图像向上偏移

2、x% y%

0% 0% : 原始位置

100% 100% :右下角

50% 50% : 正中间

3、关键字

x ：left center right

y ：top center bottom

3、CSS Sprites

CSS精灵

将若干幅小图像 拼合成 一幅大图像

7、背景属性

属性：background

取值：color url() repeat attachment position;

ex

background:red url(a.jpg) no-repeat fixed center center;

常用：

background:url() no-repeat position;

background:red;

3、渐变

1 、作用

多个颜色之间，逐渐变换的效果

2、渐变分类

1、线性渐变

2、径向渐变

3、重复渐变

1、重复线性

2、重复径向、

3、重要元素

1、色标

每种颜色值 及其 出现的位置

4、渐变语法

1、属性

background-image

取值：

1、linear-gradient() : 线性渐变

2、radial-gradient() : 径向渐变

3、repeating-linear-gradient()

重复线性渐变

4、repeating-radial-gradient()

重复径向渐变

2、线性渐变

1、语法：

background-image:linear-gradient(angle,colo-point,color-point);

1、angle

方向 或 角度

取值：

1、to top--> 0deg

从下向上填充颜色

2、to right --> 90deg

从左向右填充颜色

3、to bottom --> 180deg

从上向下填充颜色

4、to left --> 270deg

从右向左填充颜色

2、color-point

color position

ex:

red 0px

green 50%

注意：省略 位置

3、径向渐变

1、语法

background-image:radial-gradient([size at position],color-point,color-point);

1、size at position

size ：半径大小

position ：圆心所在位置

0px 0px：圆心在元素左上角

50% 50%: 圆心在元素中间处

left bottom:圆心在元素坐下角

ex:50px at center center

4、重复渐变

repeating-linear-gradient();

repeating-radial-gradient();

注意：colo-point 颜色的位置 必须给 px为单位的数值。

repeating-linear-gradient(to top,red 0px,yellow 20px,blue 50px);

5、浏览器兼容性

主流浏览器的新版本 肯定支持渐变

不支持的版本，尝试着加浏览器前缀

Firefox : -moz-

Chrome & Safari : -webkit-

Opear : -o-

IE : -ms-

background-image:-moz-linear-gradient();

background-image:-webkit-linear-gradient();

background-image:-o-linear-gradient();

background-image:-ms-linear-gradient();

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月5日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

box-sizing : 规定了元素的宽度从哪算起

content-box ：内容区域的宽

border-box ：边框以及框内的宽度 为 width 值

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

文本格式化属性

1、文本格式化

字体样式：加粗，斜体 ，大小

文本格式：排列方式，缩进，颜色

建议：尽可能使用 css 属性取代 html 标记

2、控制字体

1、字体系列

属性：font-family

取值：value1,value2,....,valuen;

注意：字体值中包含特殊符号(空格)、中文时，需要用 "" 引起来

ex ：font-family:"microsoft yahei",Arial,simsun;

2、字体大小

属性：font-size

取值：px 或 pt 或 em

3、字体加粗

属性：font-weight

取值：

1、normal 正常

除 i ，strong，h1~h6 之外 ，其他都是 normal

2、bold

加粗

3、value

无单位的数值

100

~

1200

常用值：400 ~ 900

4、字体样式

1、作用

斜体--><i></i>

2、属性

font-style

3、取值

1、normal 正常

2、italic 斜体

5、小型大写字母显示

Aa Bb Cc Ee

1、作用

英文，按照大写字母显示，个头和小写字母一样高

2、属性

font-variant

3、取值

1、normal

2、small-caps

6、字体属性 font

font:style variant weight size family;

注意：font 属性值中，如果没有 family 的值， 则无效

font:12px; 无效

font:12px "微软雅黑"; 有效

3、文本属性

1、文本颜色

color

2、文本排列

文本的水平对齐方式

1、属性

text-align

2、取值

left/center/right/justify

3、文字修饰(线条)

1、属性

text-decoration

2、取值

1、none

无线条样式

除 a、s、u 其余都是none

2、underline

下划线

3、overline

上划线

4、line-through

删除线

4、首行文本缩进

1、属性

text-indent

2、取值

以 px 为单位的数值

5、文本阴影

1、属性

text-shadow

2、取值

h-shadow v-shadow blur color;

6、行高

1、作用

设置一行文本的高度。

如果 行高的高度值 大于 文本本身的高度，那么 该行文本将在指定的行高高度内 呈现出 垂直居中的效果

2、表格属性

1、表格常用属性

1、内边距属性 ： padding

2、尺寸属性 ：width height

3、文本格式化/字体属性

font-\*

text-\*

4、背景属性

5、border属性

6、vertical-align

1、允许使用 vertical-align属性的元素

1、tabletd

2、img

3、行内块

2、作用

垂直对齐方式

3、取值

top/middle/bottom

2、表格特有属性

1、边框合并属性

属性：border-collapse

取值：

1、separate

默认值，分离边框

2、collapse

边框合并

2、边框边距

属性：border-spacing

作用：设置相邻单元格的边框间的距离

要求：border-collapse 必须为 separate

取值：

1、取一个数值 ：水平和垂直间距相同

2、取两个数值 ：第一个值 表示水平间距，第二个值 表示垂直间距

两个值之间用 空格 隔开

3、标题位置

<caption></caption>

属性：caption-side

取值：

1、top，默认值

2、bottom

4、显示规则

1、作用

规定 单元格 的宽度 和 高度的处理方式

2、属性

table-layout

取值：

1、auto

默认值，自动表格布局，单元格宽度由单元格内容决定

2、fixed

固定表格布局。单元格宽度 由表格和单元格设置的值为主。

3、对比

1、自动表格布局

table-layout:auto

1、单元格大小会适应内容的大小

2、加载复杂的表格式比较慢

3、适用于不确定每列大小时使用

2、固定表格布局

table-layout:fixed

1、列宽度取决于表格宽度，列宽度，与单元格内容无关

2、会加速显示表格

固定表格布局 ，算法较快，不够灵活

自动表格布局 ，算法较慢，传统表格

3、浮动(重点)

1、定位概述

1、什么是定位

元素应该出现在网页中的哪个位置处

2、分类

1、普通流定位

2、浮动定位

3、相对定位

4、绝对定位

5、固定定位

3、普通流定位

1、what

又称为 文档流定位

每个块级：占用一行的空间，并且上下排列

每个行内：从左到右排列，不单独占一行

2、浮动定位

1、什么是浮动 & 特点

1、会将元素排除在文档流之外-脱离文档流

2、元素不占据页面空间

3、其余在文档流中的元素要上前补位

4、浮动元素会停靠在父元素的左边或右边，或者停靠在已经浮动起来的元素的边缘上

2、处理问题(作用)

1、让块级元素在一行内显示

3、属性

属性：float

取值：

1、none

默认值，即没有浮动效果

2、left

左浮动

3、right

右浮动

4、特殊处理

1、如果父元素的宽度已经装不下所有的已浮动子元素，那么最后一个将会换行

2、浮动元素如果不设置宽度的话，那么元素的宽度将自适应

3、所有的元素浮动起来都将变成块级元素

块级元素：允许修改 尺寸

4、浮动元素 对 文字、行内元素、行内块元素的影响 （产生环绕效果）

5、清除浮动

1、属性

clear

作用：清除左右两端浮动元素给当前元素带来的影响

取值：

1、left

清除 当前元素 左边的浮动元素带来的影响

2、right

清除 当前元素 右边的浮动元素带来的影响

3、both

清除 当前元素 两边的浮动元素带来的影响

4、none

默认，不清除

6、浮动元素对父元素高度所带来的影响

1、父元素浮动

2、设置父元素的高度

3、设置父元素 overflow 值为 auto 或 hidden

4、父元素中，追加一个 空的 块级 子元素，并且 设置其 clear 属性为both

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月8日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**显示**

1、显示方式

块级元素： "块框"

行内元素： "行内框"

1、属性

1、display

作用：定义元素的显示方式

2、取值

1、none

让生成的元素没有框

可以理解为隐藏

会脱离文档流-不占据页面空间

2、block

让 元素 表现的和块级元素一样

3、inline

让 元素 表现的和行内元素一样

注意：不要试图将 块级 元素改变成 行内元素

4、inline-block

让 元素 表现的和行内块一样的

特点：

位置摆放于行内元素一致

允许修改元素的宽和高

2、显示效果

1、显示/隐藏

1、属性

visibility

2、取值

visible : 默认值，元素可见

hidden : 元素不可见，但是依然占据页面空间

问题：display:none 与 visibility:hidden之间的区别

1、display 为none 改变元素显示方式，并且“脱离文档流”，所以不占据页面空间

2、visibility 为 hidden，只是改变隐藏性，不脱离文档流，空间依然占据，没有东西显示

3、collapse

用在表格时，删除一行或一列时，不影响表格的整体布局。

2、元素透明度设置

1、属性

opacity

2、取值

0(完全透明)~1(完全不透明)之间的小数

3、垂直对齐

1、属性

vertcial-align

2、注意\*

该属性只能使用在 td，img，行内块元素中

3、取值

1、baseline ：基线对齐

2、top

3、middle

4、bottom

4、注意

垂直对齐 用于 图像元素时，控制的图像两边的文本 相对于图像的垂直对齐方式

3、光标

1、属性

cursor

2、取值

1、default

默认值

2、pointer

小手

3、crosshair

+

4、text

I

5、wait

等待

6、help

?

2、列表

1、列表项标志

属性：list-style-type

取值：

none ：不显示

disc ：实心圆

...

2、列表项图像

属性：list-style-image

取值：url();

ex:list-style-image:url(图像路径);

3、列表项位置

属性：list-style-position

取值：

1、outside ：默认值，标记位于文本区域之外。

2、inside ：标记放置文本区域之内

4、列表属性

属性：list-style

取值：type url() position;

常用：list-style:none;

ul{

list-style:none;

margin:0;

padding:0;

}

5、常用场合

从上到下 或 从左到右(列表项浮动) 的显示数据时，优先使用列表

3、定位

1、定位属性

1、属性-position

作用：实现元素的特殊定位方式

取值：

1、static - 默认值

2、relative - 相对定位

3、absolute - 绝对定位

4、fixed - 固定定位

relative、absolute、fixed 被称为“已定位”元素

2、偏移属性

作用：根据不同的定位方式去实现元素的位置摆放(偏移)

top 属性

bottom 属性

left 属性

right 属性

3、属性-z-index

作用：已定位元素的堆叠顺序

2、相对定位

1、什么是相对定位

元素框会相对于它原来的位置偏移某个距离

2、属性&值

position:relative;

通过 偏移属性 ，来实现位置的微调

3、使用场合

1、位置微调

3、绝对定位

1、什么是绝对定位 & 特点

1、绝对定位的元素会脱离文档流-不占据页面空间

2、会相对于最近的已定位的祖先元素实现位置的初始化

已定位：relative，absolute，fixed

祖先元素：无限级的父级元素

<div id="d1">

<div id="d2">

<div id="d3">

<span id="s1"></span>

</div>

</div>

</div>

3、如果元素没有已定位的祖先元素，那么它的位置就相对于最初的包含块实现位置初始化

最初的包含块：比如 body

2、属性

position:absolute;

通过 偏移属性 实现位置初始化

3、注意

1、绝对定位的元素 都将 变成块级元素

2、margin 的处理，除auto外，其他数值均正常显示

3、能够处理 元素 堆叠的特殊显示效果

4、堆叠顺序

属性：z-index

作用：显示调整已定位元素的堆叠顺序效果

取值：无单位数字，值越大越靠前

注意：

堆叠顺序无法调整父子元素之间的堆叠效果，父子元素间，永远都是 子 压在 父上

5、固定定位

1、什么是固定定位

将元素的内容固定在页面的某个位置处

会脱离文档流-不占据页面空间，可以实现堆叠效果

滚动页面时，固定定位元素不会随着一起滚动

2、属性

positioin:fixed;

通过 left top right bottom 实现位置的摆放

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月12日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CSS核心

1、复杂选择器

1、作用

匹配 页面的元素 ... ...

2、选择器分类

1、兄弟选择器

1、作用

通过 元素的位置关系匹配元素

位置关系：兄弟关系(平级元素)

<div>

<p id="p1"></p>

<p id="p2"></p>

<p id="p3"></p>

<p id="p4"></p>

</div>

2、相邻兄弟选择器

1、作用

匹配指定元素的相邻兄弟元素

被匹配的元素要 紧紧的跟在 前一个元素的后面

2、语法

选择器1+选择器2

#p1+p{}

3、通用兄弟选择器

1、作用

2、语法匹配某元素后面所有的兄弟元素

选择器1~选择器2{}

2、属性选择器

1、作用

使用元素所附带的 属性及其值，来匹配页面的元素

ex ：获取页面中所有 type 值为 text 的元素

2、语法

语法：[]

1、[属性名]

作用：匹配页面中 所有 附带 指定属性的元素

[class] : 匹配页面中所有附带 class 属性的元素

[type] :

2、元素[属性名]

作用：页面中所有 附带 指定 属性的 指定元素

ex:

p[id]:匹配 所有 附带 id 属性的p元素

input[type]:匹配 所有 附带 type 属性的 input元素

input[type][class]:匹配 所有附带 type属性 和 class 属性的 input元素

3、元素[属性名=值]

作用：匹配 所有 属性名 = 指定值得 元素

ex:

1、获取页面上所有的文本框(type=text)

[type=text]{}

input[type=text]{}

4、元素[class~=值]

<div class="c1 c2 c3 c4"></div>

作用：匹配 附带 class属性的 元素，其中class属性的值是以 空格 隔开的值列表。"值" 是列表中的一个独立单词

div[class~=c2]

5、元素[属性^=值]

作用：匹配 属性值 以 值作为开始的元素

ex:

div[class^=c] : 匹配页面中 class属性值以c作为开始的div元素

获取页面中，所有 name 属性值 以 txt 作为开始的 input元素

input[name^=txt]

6、元素[属性\*=value]

作用：匹配页面中 属性值中 包含value字符的元素

ex:

div[class\*="ort"]

<div class="important"></div>

7、元素[属性$=value]

作用：匹配页面中 属性值 以value字符作为结尾的元素

3、目标伪类

1、作用

突出显示活动html 锚点 元素

2、语法

:target

4、元素状态伪类

1、匹配 启用、禁用、被选中状态

2、语法

:enabled : 匹配每个已启用元素(表单元素中)

:disabled : 匹配每个已禁用元素(表单元素中)

:checked : 匹配每个被选中的元素(checkbox,raido)

5、结构伪类(重点)

1、通过结构关系进行元素匹配

2、语法

1、:first-child

匹配 属于其父元素中的首个子元素

td:first-child{background:red;}

2、:last-child

匹配 属于其父元素中的最后一个子元素

3、:nth-child(n)

匹配 属于其父元素中的第n个子元素

:nth-child(1) --> :first-child

4、:empty

匹配 没有子元素(文本)的元素

<div>

<p>xxx</p>

</div>

非 empty

<div>xxxx</div>

非 empty

<div>

</div>

非 empty

<div></div>

empty

div:empty

5、:only-child

匹配 属于去 父元素中的 唯一子元素

6、否定伪类

语法：

:not(选择器)

作用：将 指定选择器匹配的元素 排除在外

**7、伪元素选择器**

1、伪元素 与 伪类的区别

1、伪类 ：匹配 元素

2、伪元素 ：匹配元素中的文本内容

2、分类

1、:first-letter 或 ::first-letter

作用：用于匹配 元素中的首字符

2、:first-line 或 ::first-line

作用：用于匹配元素中的 首行

3、::selection

作用：匹配用户选取的部分

3、: 与 ::之间的区别

W3C规范

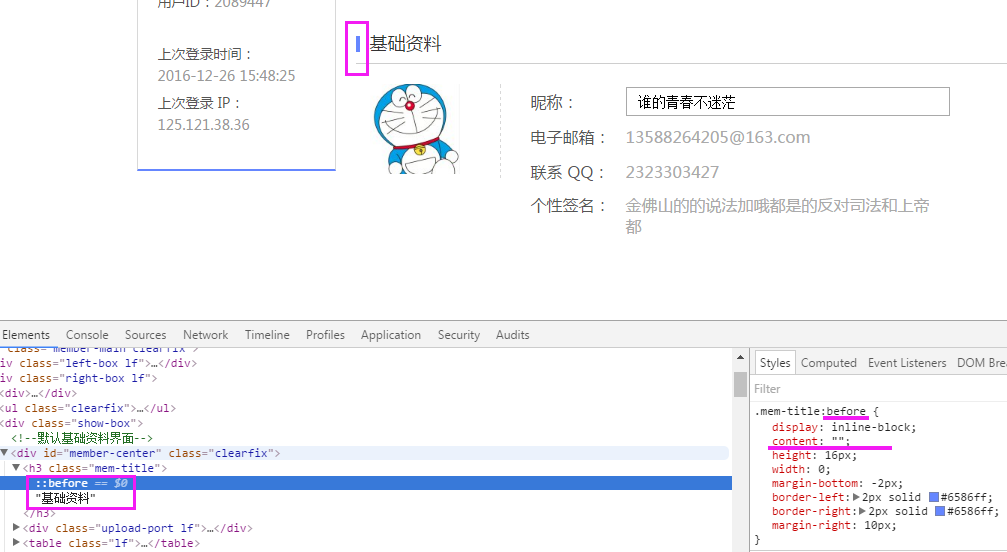
: 表示的 是伪类选择器

:: 表示的是 真正的 伪元素选择器

在 CSS2.1的时候，伪类 和 伪元素 都是用 : 来进行表示

现阶段，推荐使用 : 即表示伪类，页表示伪元素

1. **内容生成**



1、作用

允许使用CSS的方式，在匹配元素内容区域之前或之后 插入新的数据

<div>普通的div</div>

2、选择器(伪元素)

**1、:before**

**匹配 元素 内容区域之前**

2、:after

匹配 元素 内容区域之后

3、属性

属性：content

作用：配合 :before 和 :after 一起插入生成的内容

取值：

1、图像 ：url()

2、字符串：纯文本

3、计数器

4、计数器

1、作用

通过 CSS 生成 一组有序的数字，通过 内容生成的方式，插入到元素中

2、属性 和 函数

1、counter-reset

counter：计数器

reset：重置，复位

1、作用

定义 或 复位 一个计数器，并设置其初始值

2、语法

counter-reset:名称 值;

注意：

1、默认值 可以省略，省略则为 0

2、声明位置：不要放在使用计数器的元素中声明

3、声明多个计数器

counter-reset:名称 值 名称 值;

2、counter-increment

1、作用

设置某个选择器每次每次出现的计数器 增量

2、语法

counter-increment:名称 增量值;

注意：

1、增量值可省，默认值为1

2、增量之可以为正(递增),也可以为负(递减)

3、什么地方声明：哪个元素使用，放在哪个元素中声明

4 、设置 多计数器 增量

counter-increment:名 增量 名 增量;

3、函数:counter()

语法：content:counter(计数器名)

3、练习

完成以下内容展示(通过计数器)

1、html

1.1 块级元素

1.2 行内元素

1.3 图片元素

2、CSS

2.1 行内样式

2.2 内部样式表

2.3 外部样式表

3、多列

1、分隔列

属性：column-count

取值：数字

2、列间隔

属性：column-gap

取值：px为单位的数字

3、列规则

作用：每两个列之间的边框的大小 样式 颜色

属性：column-rule

取值：width style color;

4、兼容性

IE10+ , Opera 支持 多列

Firefox ：-moz-

Chrome、Safari ：-webkit-

4、CSS Hack

1、IE 标准模式和 混杂模式

运行模式：

1、标准模式

2、混杂模式

3、准标准模式

不同模式

1、CSS 框模型 有影响

2、JS解析 有影响

依靠 <!doctype>进行模式选择

2、CSS Hack

浏览器的类型以及版本 不同 会造成 CSS效果 显示不同

1、不同厂商的浏览器

IE ，Firefox ，Chrome，Safari，Opera

2、同一厂商浏览器的不同版本

IE6 IE7 IE8 ... ...

3、CSSHack 工作原理

1、通过 选择器器 或 样式的优先级 来解决兼容性问题

2、实现方式

1、CSS 类内部Hack

通过 前后缀的 方式解决兼容性

2、选择器Hack

在选择器前加前缀 解决兼容性问题

3、HTML头部引用Hack

通过 条件注释 来解决兼容性问题

根据 条件注释 来判断浏览器的类型和版本，再决定执行哪些代码

语法：

<!--[if 关键字 IE 版本]>

满足条件的话可以被正常执行

否则，就被注释掉

<![endif]-->

1、关键字

1、省略

判断是否为 IE 或 是否为IE的指定版本

ex

<!--[if IE]>

该段内容只有在 IE浏览器中被运行

<![endif]-->

<!--[if IE 6]>

该段内容只有在 IE6 浏览器中被运行

<![endif]-->

2、gt

greater than

选择条件版本以上版本浏览器，不包含条件版本

<!--[if gt IE 6]>

在 IE6(不包含) 以上版本的浏览器中显示

<![endif]-->

3、gte

greater than or equal

大于等于

包含指定版本浏览器

4、lt

小于指定版本(不包含)

5、lte

小于等于指定版本(包含)

6、!

选择条件版本以外的所有版本

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月15日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1、转换

1、转换的简介

1、什么是转换

改变元素在网页中的 形状，尺寸，位置和角度的一种方式

元素能够实现2D转换或3D转换

2D:使元素在 x轴 和 y轴发生变化

3D:2D基础上，增加在Z轴的变化

2、转换属性

1、属性

transform

取值：

1、none

默认值，无转换效果

2、transform-function

一个或多个 转换函数，中间用 空格 分开

2、转换的原点

1、属性

transform-orgin

默认情况下：转换原点在元素中的中心点

2、取值

1、value1 value2

value1 ：在x轴上的位置

value2 ：在y轴上的位置

2、value1 value2 value3

value1 ：在x轴上的位置

value2 ：在y轴上的位置

value3 : 在z轴上的位置

3、取值单位

1、数值

以元素的左上角 为 0px 0px 开始计算

0px 0px

2、百分比

以元素的左上角 为 0% 0% 开始

0% 0% ：左上角

50% 50% : 元素中间

3、关键字

top,right,bottom,left

ex:

div{

transform-origin:center center; /\*\*原点在元素中心\*/

transform-orgin:0% 100%;/\*原点在元素左下角\*/

}

2、2D转换

1、2D位移

1、作用

改变元素在页面中的 位置

2、属性 和 函数(值)

属性：transform

函数：

translate(x)

translate(x,y)

translateX(x)

translateY(y)

取值：

1、数值

2、百分比

3、负数

2、2D缩放

1、作用

改变元素的 大小

根据缩放比例 改变元素大小

2、属性 和 函数

属性：transform

函数：

scale(value)

scale(x,y)

scaleX(x)

scaleY(y)

注意：一个参数时，第二个参数与第一个参数默认相等

x : x轴上的缩放比例

y : y轴上的缩放比例

取值：

默认值 为 1

缩小：0-1之间的小数

放大：大于1的数值

3、2D旋转

1、作用

改变元素显示的角度

2、属性 和 函数

属性：transform

函数：rotate(ndeg)

deg : 单位 ， 不能省略

n : 旋转的角度

+ ：顺时针旋转

- ：逆时针旋转

3、注意

元素旋转后，连同坐标轴也跟着旋转

4、转换原点

transform-origin:value1 value2;

4、2D倾斜

1、作用

让元素向x轴或y轴产生一定倾斜角度(倒)

2、属性 和 函数

属性：transform

函数：

skew(xdeg)

skew(xdeg,ydeg)

skewX(xdeg)

skewY(ydeg)

3、注意

1、skewX(x)

元素向x轴的方向产生倾斜，实际上是改变y轴的倾斜角度(逆时针)

2、skewY(y)

元素向y轴的方向产生清下，实际上是改变x轴的倾斜角度(顺时针)

3、3D转换

1、3D

空间距离 --> 空间轴 --> z轴

2、空间距离

1、属性

perspective

-webkit-perspective : 500px;

2、作用

定义人眼 到 3D投射平面的距离

3、定义位置

该属性要定义在 实现3D转换元素的父元素上

3、3D转换效果

1、旋转

1、属性和函数

属性：transform

函数：

1、rotateX(xdeg);

2、rotateY(ydeg);

3、rotateZ(zdeg);

4、rotate3D(x,y,z,ndeg);

x,y,z 的取值 如果 为0 ， 则该轴不参与旋转

rotate3d(0,0,1,45deg)-->rotatez(45deg)

transform:rotate3D(1,1,1,45deg)

与

transform:rotatex(45deg) rotatey(45deg) rotatez(45deg);

2、位移

1、属性 和 函数

属性：transform

函数：

translatex()

translatey()

translate(x)

translate(x,y)

translatez(z);

translate3d(x,y,z);

3、属性：transform-style

取值：

1、flat

默认值，以2D元素位置摆放子元素

当前元素还是2D元素

2、preserve-3d

将当前元素变成3D元素

2、过渡

1、什么是过渡

过渡(transition),使CSS属性值，在一段时间内平滑过渡，能够观察到变化的过程和最后的结果。

2、过渡要素 & 属性

1、过渡属性

属性：transiton-property

取值：

1、none

2、all

3、property(具体的属性名称)

ex :

transition-property:background;

transition-property:width;

可以设置过渡的属性:

1、颜色属性

2、取值为数值的属性

3、转换属性

transform

4、渐变属性

5、visibility属性

6、阴影

2、过渡时长

1、作用

指定 整个过渡 所需要的时间

2、属性

transition-duration

取值：

以 s | ms 作为单位的数值

1s = 1000ms

0.3s = 300ms

3、注意

该属性 在过渡中 必须设置，否则默认为 0 ，不会产生过渡效果

3、过渡速度时间曲线函数

1、属性

transition-timing-function

取值：

1、ease：默认值，慢速开始，快速变快，慢速结束

2、linear：匀速

3、ease-in ：慢速开始，加速效果

4、ease-out: 快速开始，减速效果

5、ease-in-out:慢速开始和结束，先加再减

4、过渡延迟

1、作用

当用户激发过渡效果后，等待多长时间后开始 显示效果

2、属性

transition-delay

取值： 以 s | ms 作为单位的数值

默认值 ：0s

3、触发过渡

用户特殊行为 进行 触发

如：

:hover

:active

:focus

或JS

4、简写属性

transition:property duration timing-function delay,property duration... ...;

3、动画

1、什么是动画

动画(animation)，使元素从一种样式逐渐变化为另外一种样式的过程

2、完成动画所需要的步骤

1、声明动画

在特点的时间点上设置动画的样式

时间点：是个模糊时间，而不是具体时间

样式：元素的 样式

2、为元素调用动画

通过 animation 属性 调用动画

并且指定动画的播放时长、次数等

3、声明动画

1、作用

通过 @keyframes 关键字 声明动画的"关键帧"

关键帧：某个时间点上的特殊效果

2、语法

@keyframes 动画名{

0% | from{

/\*动画开始时的样式效果\*/

width:100px;

height:100px;

background:red;

}

...

50%{

background:green;

}

...

100% | to{

/\*动画结束时的样式效果\*/

width:100px;

height:100px;

border-radius:50%;

background:blue;

}

}

兼容性：

@-webkit-keyframes 名{}

@-moz-keyframes 名{}

4、调用动画(兼容性)

1、animation-name

指定调用动画的名称

2、animation-duration

完成一个动画周期的时间

必须设置属性

s | ms 作单位

3、animation-timing-function

完成动画时的速度时间曲线函数

4、animation-delay

动画播放之前的延迟

5、animation-iteration-count

动画播放次数

取值：

1、数值

2、infinite ：无限次播放

6、aniamtion-direction

动画播放方向

取值：

1、normal,默认值，正向播放(0%~100%)

2、reverse,逆向播放(100%~0%)

3、alternate,奇数播放次数是正向播放，偶数播放次数时，逆向播放

7、简写方式 - animation

animation:name duration timing-function delay interation-count direction;

8、animation-fill-mode

规定动画在播放之前或之后，动画效果是否可见。

取值：

1、none

不改变默认行为

2、forwards ：动画播放完成后，保持在最后一个 帧 的状态上

3、backwards ：动画播放之前(延迟时间内)，保持在第一个 帧 的状态上

4、both：动画播放前后都采用填充模式

9、animation-play-state

规定动画的播放状态(运行或暂停)

取值：

1、paused : 暂停

2、running : 播放

4、CSS优化

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月19日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

JAVAScript

1、Javascript 概述

1、什么是Javascript (JS)

Javascript 是一种运行于 JS解释器/引擎 中的解释型脚本语言

JS解释器/引擎 ：JS运行环境

1、独立安装的解释器 NodeJS

2、浏览器内核中嵌入的JS解释器

PC机，手机，平板，电视 。。。。

解释性脚本语言：

解释型：运行之前不会编译

编译型：运行之前先编译C,C++ ...

2、JS的发展史

1、1992年 ， Nombas 为自己的软件写一款脚本语言 C----》 ScritEase ,可以嵌入在网页中运行

2、1995年 ，Netscape(网景) Navigator2.0 --> LiveScript --> Javascript ,Java 与 Javascript之间没有任何关系

3、1996年 ，Microsoft 在 IE3.0 中发布 JS的克隆版 --> JScript

4、1997年 ，Javascript 1.1 版本 作为 草案 提交给了ECMA(欧洲计算机制造商联合会) ECMA-262标准ECMAScript 作为 JS的 标准核心。 ES5ES6

完整的JS的组成：

1、核心(ECMAScript)

2、文档对象模型(DOM,Document Object Model) , 让JS有能力与页面元素进行对话

3、浏览器对象模型(BOM,Browser Object Model) ，让JS有能力与网页进行对话

3、JS的特点

1、编写简单 - 任何文档编辑工具都可以编辑JS

2、无需编译，直接运行

3、弱类型编程语言

强类型编程语言： C,C++，Java ...

特点：由数据类型来决定数值

弱类型编程语言：JS

特点：由数值 来决定数据类型

4、面向对象的编程语言

4、JS的用途

1、客户端的JS

1、客户端的数据计算

2、表单输入验证

3、网页中的事件触发和处理

onclick-- 单击元素时做的操作

4、网页特效制作

5、服务器端异步数据提交(AJAX)

2、服务器端的JS

1、分布式运算

2、实时服务器

3、窗口应用

4、网络应用

2、Javascript基础语法(重点)

1、浏览器内核

1、作用

负责页面内容的渲染

2、引擎

1、内容排版引擎-解析HTML/CSS

2、脚本解释引擎-解析JS

3、常用浏览器的内核和引擎

1、Microsoft IE

内核：Trident

排版引擎：-

脚本引擎：Chakra

2、Mozilla Firefox

内核：Gecko

排版引擎：-

脚本引擎：猴子(Monkey)系列

3、Apple Safari

内核：Webkit

排版引擎：Webcore

脚本引擎：Nitro

4、Google Chrome

内核：Webkit

排版引擎：Webcore

脚本引擎：V8

5、Opera

内核：Presto

排版引擎：-

脚本引擎：Carakan

2013 以后

内核：Webkit

排版引擎：Webcore

脚本引擎: V8

2、JS运行环境

1、独立安装的JS解释器(了解)

NodeJS

代码：

console.log("Hello World");

退出 Node

Ctrl + C ： 2次

2、嵌入在浏览器内核中的JS引擎(重点)

1、直接在Console 中输入脚本并执行

2、将脚本嵌入在HTML页面中执行

1、html元素事件

onclick --> 单击时执行的操作

onmouseover

onmousemove

on....

2、<script>元素中编写JS脚本

<script></script>

可以出现在网页的任何位置处

<script>

JS脚本

</script>

document.write("内容");

3**、外部脚本文件**

1、创建脚本文件并编写 脚本

脚本文件：\*\*\*.js

2、引入外部脚本文件

<script src="脚本文件路径"></script>

注意：

1、必须是成对的 script

2、该对标记中，不允许出现任何内容

3、JS调试

1、代码错误

解释型语言，若某行代码错误，则解释器终止此次执行

但不会影响后续块的执行，以及后续HTML解析

<script></script>

<script></script>

... ...

4、语法规范

1、代码组成

1、语句 - 能够被JS引擎所执行的代码

通常由 ; 作为结尾

表达式、关键字、运算符 来组成

console.log("");

var s=1+2;

2、注释

1、单行注释

// 单行注释

2、多行注释

/\*

\* 多

\* 行

\* 内

\* 容

\*/

2、严格区分大小写

console.log("");

Console.log(""); //错误

3、变量和常量(重难点)

1、变量

内存:保存程序在运行过程中所需要用到的数据(临时性)

变量：内存中的一段内存空间

值：变量中(内存空间中)所保存的数据

变量名：内存空间的别名，自定义

1、变量的声明

1、声明

var 变量名;

2、赋值

变量名=值;

3、声明变量并赋初始值

var 变量名=值;

4、注意

1、声明变量时可以不加 var 关键字

var stuname="张三丰";

==

stuname="张三丰";

如果省略了 var 关键字， 那么声明的就是"全局变量"，有可能造成 "全局污染" 的效果

2、如果声明变量没有赋值的话，那么将自动赋值为 undefined

5、一次性声明多个变量

声明 stuname，stuage，stugender 变量

var stuname;

var stuage;

var stugender;

一次性声明多个变量 使用一个 var 关键字， 多变量名之间 用 , 分隔

var stuname, stuage,stugender;

2、变量名的规范

1、标识符

表示 变量 或函数的名称

2、要求

1 、不允许使用语言的关键字和保留关键字作为标识符的名称

3、命名规范

1、可以包含字母、数字、下划线、$

var +name; //错误 ，不能有 +

2、不能以数字开头

var 1name; //错误

var name1; //正确

var $name; //正确

var \_name; //正确

3、尽量见名知意

var a;

var b,c,d,e,f,g,aab,aac;

//以上方式， 不推荐

var username;

var salary;

4、可以采用 "匈牙利命名法","驼峰命名法","下划线命名法"

1、匈牙利命名

表单控件缩写+功能

var txtUsername= ....;

2、驼峰命名法

如果 变量名 是由多个单词组成的合成词，从第二个单词开始，每个单词的首字符变大写。

var age;

var gender;

var salary;

var userName;

var userSalary;

var stuJavaScore

3、下划线命名法

var \_userName;//一般用于全局变量命名

var user\_name;

3、变量的使用方式

1、未经初始化的变量

1、变量声明之后，从未赋过值

var userName;

console.log(userName);

结果为：undefined

2、变量未被声明而直接使用

console.log(userName);

2、变量的存取操作

1、存 - SET操作

将数据保存到 内存中的过程

var useName = "张三丰";

userName = "张翠山";

userName = "张无忌";

特点：变量名成出现在 赋值符号(=) 的左边，一律是 赋值操作

2、取 - GET操作

var userSalary = 8000; //赋值操作

console.log(userSalary);//

var newSalary = userSalary + 10;

特点：只要不是 赋值操作，那么就都是取值。

= : 如果出现 赋值符号， 先算 右边的数据的结果，再赋值给 左边的变量

var age=18;

18=15;//错误,=的左边必须是变量

3、练习

套餐名称：

香辣鸡腿堡

汉堡名称：

香辣鸡腿煲

配餐名称：

烤翅

饮料名称：

可口可乐

2、运算符

1、算术运算符

+,-,\*,/,%

% : 模 , 获取两个数字的余数

5 % 2==> 5 / 2 = 2 ... 1

ex：

1、判断 5 是否为 偶数

2、获取1234 的最后 一位数

/ : 除法

\* : 乘法

- : 减法

+ :

1、两个 数字 做 + ，就是 求和

var num1=15;

var num2=18;

var result=num1+num2;//33

2、如果+两边有一个不是数字的话，那么就 做 字符串的累加

var s1="你好";

var s2="世界";

var result=s1+s2;//你好世界

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

var num1 = 18;

var num2 = "15";

var result=num1+num2;

// 1815

2、算术运算符的优先级

1、正常 从左向右 运算

2、先 \*/% 再 +-

var result = 25+4\*8/2+6%3;

25+16+6%3;

25+16+0

41

var result = "你好"+13+25;

结果为：

你好1325

var result = 13+25+"你好";

结果为：

38你好

3、提升优先级

var result = 25+4\*8/2+6%3;

使用 () 提升优先级

var result = (25+4)\*8/((2+6)%3);

== 29\*8/(8%3) == 29\*8/2 = 116

3、常量

1、什么是常量

一旦声明好，就不能被修改的数据 叫做常量

π = 3.14

hourOfDay=24

2、声明常量

const 常量名=值;

注意：常量声明时，必须赋值

常量的名称 一般采用 全大写 方式

ex:

const PI=3.14;

3、练习

程序中，声明一个变量(r),表示圆的半径，并且赋值，计算 该圆的周长(l),和面积(s)

l = 2\*π\*r;

s = π\*r\*r;

半径为10的圆的周长为62.8

半径为10的圆的面积为314

4、数据类型

1、什么是数据类型

保存在内存的数据的类型 是什么

8bit(位) = 1byte(字节)

1024byte = 1kb(千字节)

1024kb = 1MB

1024MB = 1GB

1024GB = 1TB

2、数据类型分类

1、基本数据类型(原始类型)

1、number 类型

数字类型

表示 32位的整数 也可以表示64位的浮点数

整数：

1、十进制

var num = 125;

2、八进制

var num = 0125;

3、十六进制

var num = 0x123AF

小数

1、小数点

var num = 123.456;

2、指数记录

var num = 5.6e10;

2、string 类型

字符串类型

表示 一系列的文本字符的数据

由 Unicode 字符、数字、标点来组成

汉字的范围：

\u4E00 ~ \u9FA5

必须使用"" 或 '' 引起来数据

var n = 15;//number 类型

var n1 = "15";//string 类型

转义字符

\n : 换行

\r : 回车

\t : 制表符

\" : "

\' : '

\\ : \

3、boolean 类型

布尔类型

仅有两个值：true 和 false

var result = true; //真

var result = false; //假

var result = "false"; //string 类型

做算术运算时，true = 1，false=0

var num = 45;

console.log(num+true);//46

4、undefined 类型

语义：不存在该数据

场合：

1、声明变量未赋值

2、对象属性不存在

2、引用类型

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月22日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1、数据类型转换

1、隐式转换

var num;

num = 15; //number类型

num = "你好!";//string类型

弱类型：是由值来决定数据类型

var num1 = 15;

var num2 = 30;

var result = num1 + num2;

===============================

var str1 = "Hello";

var num1 = 30;

var result = str1 + num1;//Hello30

1、typeof 函数/运算符

语法

var r1 = typeof 变量;

var r2 = typeof(变量);

2、NaN

Not a Number ：不是一个数字

isNaN() : 是不是一个数字

返回结果：true 或 false

true ：不是一个数字

false ：是一个数字

3、结论

1、数字 + 字符串 ：将数字转换为字符串

2、数字 + boolean ：将 boolean 转换为 number类型

3、数字 + undefined ：结果为 NaN，NaN的类型为 number

4、字符串 + boolean ：将boolean 转换为 字符串

5、boolean + boolean ：都转换成number在做相加运算

不同类型的数据在计算过程中会进行"自动转换"

2、转换函数

显示转换/强制转换

1、toString()

将任意类型的数据转换为字符串类型

var newVar=变量.toString()

2、parseInt()

作用：将一个数据转换成"整数"

语法：var result=parseInt(变量);

注意：

1、碰到第一个非数字字符则停止转换，已经转换的则保留

2、如果第一个字符就是非数字字符，结果为 NaN

3、parseFloat()

作用：将指定的数据转换为 小数

语法：var result = parseFloat(变量);

4、Number()

作用：将一个数据转换为 number 类型

注意：

只要包含非法字符(除数字和小数点)，则返回NaN

3、练习

console.log("");//控制台打印

document.write("");//页面输出

window.alert();/alert();//页面弹框

输入：

window.prompt();/prompt();//页面输入框

prompt("提示文本");

2、运算符和表达式

1、分类

1、单目运算符

只需要一个操作数即可

2、双目运算符

需要两个操作数的的运算符

a + b;

5 % 2;

a >= 25；

3、三目运算符

需要三个操作数的运算符

?: 列子：8>4?tue：false; 8>4表达式1必须是一个布尔值，判断是否为真，为真就选true，不然就选false

2、详解

1、算术运算符

+,-,\*,/,%

10 % 3 : 1

10.5 % 3 :1.5

...

**1、当递增递减运算单独使用，前++和后++完全等效都是加1**

**2、当递增递减运算参与到另一个表达式内时鄙视题：**

**++和--有前后之分 var n=3;**

**++m 和 m++ n++ + ++n + n++ =**

**相同：m的值一定都+1 3 + 5 + 5 =13**

**不同：后++，返回加之前的旧值**

**前++，返回加之后的新值**

++,-- 运算符，单目运算符

++:自增运算,增量为1

var i=5;

i++;i=i+1;

++i;i=i+1;

++在后 ：先使用变量的值，再对变量进行自增。

var i=5;

console.log(i++);//先打印i的值，打印完成后，i再增加

++在前 ：先对变量进行自增操作，然后再使用变量的值

var i=5;

//1、先将i的值自增(+1)

//2、打印i的值 6

console.log(++i);

--:自减运算,减量为1

变量--; //先使用 变量值，然后再自减

--变量; //先自减，然后再使用变量值

四舍五入：toFixed(保留位数);

语法：数字.toFixed(n);

由算术运算符所组成的表达式，称之为 算术表达式

var a;

var b;

var c = a + b; //算术表达式，赋值表达式

2、关系运算符

1、作用

判断数据之间的大小或相等关系

2、关系运算符

>,<,>=,<=,==,!=

由 关系运算符所组成的表达式 称之为 关系表达式，关系表达式的值为boolean类型

==，!= : 只判断数值是否相等，与类型无关

全等：===

类型相同

数值相同

不全等：!==

3、isNaN函数

1、作用

判断其参数是否为一个 "非数字" 值

返回值：

true ：值不是一个数字

false：值是一个数字

4、逻辑运算符

1、作用

判断多条件 以及 条件取反时使用

2、语法

1、逻辑 非

对现有条件 取反

语法：!

2、逻辑 与

两个条件，两条件同时为真时，&& 表达式才为真。

语法：&&

ex：

考试通过条件：

1、机试 >= 60分

2、笔试 >= 60分

3、逻辑 或

两个条件，只要有一个为真时，表达式即为真

语法：||

3、运算结果

true && true : true

true && false : false

false && true : false

false && false :false

true || true : true

true || false :true

false || true : true

false || false :false

练习：输入一个数字，判断该数字是否在 0 ~ 10 之间

练习：判断字符

中文 : \u4e00~ \u9FA5

英文 ：'a'~'z' 或者 'A'~'Z'

数字 ：'0'~'9'

闰年：

1、年份能被4整除但是不能被100整除是闰年

2、年份能够 400 整除 也是闰年

4、"短路逻辑"

对于 && : false &&? : false

如果第一个表达式已经为假的话，那么第二个表达式就不需要再判断，并且将第一个表达式的值 作为 整个逻辑表达式的值。

对于 || : true || ? :true

如果第一个表达式已经为真的话，那么第二个表达式就不需要再判断

当做假看待的 数值：

"" : 空字符串

0 :

0.0

undefined :

NaN

null:空

5、位运算符

1、二进制

由 0,1组成，逢二进一

二进制十进制

0 0

1 1

102

113

100 4

内存中：

4在内存中的形式：

0000 0000 0000 0000

0000 0000 0000 0100

1在内存中的形式

0000 0000 0000 0000

0000 0000 0000 0001

2、按位 与

语法：&

作用：将两边得到操作数都转换成二进制，每位上的数字，只有都为1的时候，该位的结果才为 1，否则 该位按0处理

ex : 1 & 3

1 : 01

3 : 11

r : 01

5 & 3

5 : 101

3 : 011

r : 001

场合：判断一个数字的奇偶性。n&1,结果为1，n为奇数，否则n为偶数

5 & 1

5 ：101

1 ：001

r ：001

6 & 1

6 : 110

1 : 001

r : 000

3、按位

或 : |

非 : ~

4、按位异或 : ^

语法：^

作用：两个数的二进制中，该位数字只有一个1的时候，该位结果才为1，其他情况都是0

ex:

5 ^ 3

5 : 101

3 : 011

r : 110

使用场合：互换两个数字时，不借助第三方变量时使用 异或

var num1=1,num2=2;

var temp = num1;

num1 = num2;

num2 = temp;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

num1=num1^num2;

num2=num2^num1;

num1=num1^num2;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

num1=5,num2=3;

num1 : 101

num2 : 011

num1=num1 ^ num2

num1 : 101

num2 : 011

num1= 110 = 6

num2=num2 ^ num1;

num2 : 011

num1 : 110

num2= 101 = 5

num1=num1 ^ num2;

num1 : 110

num2 : 101

num1= 011 = 3

5、左移

语法：<<

作用：将 对应的二进制数，向左移动几位，后面以 0 补齐位置

2<<1

2:10

0000 0000 0000 0000

0000 0000 0000 1000

2<<1 : 100= 4

2<<2 : 1000 = 8

2<<3 : 10000=16

2<<4 : 100000=32

6、右移

语法：>>

作用：将 对应的二进制数，向右移动几位，前面以 0 补齐位置

2>>2 :

0000 0000 0000 0000

0000 0000 0000 1000

6、条件运算符(三目运算符)

1、语法

? :

表达式1?表达式2:表达式3;

表达式1，值应该为 boolean 类型

如果表达式1的值为true，那么表达式2的值则作为整个表达式的值

如果表达式1的值为false，那么表达式3的值则作为整个表达式的值

ex ：输入考试成绩，如果成绩大于等于60 ，则输出及格，否则输出 不及格

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月23日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1、赋值运算符

1、=

变量 = 表达式|数值;

var salary = 2000;

var result = salary >= 5000;

var a=b=c=10; //可以，不推荐该写法

2、扩展赋值表达式

+=,-=,\*=,/=,%=,^=

声明一个变量salary表示工资，并赋值为1000，为salary+500，作为涨工资后的结果，打印工资

1、

var salary = 1000;

var newSalary = salary + 500;

2、

var salary = 1000;

salary = salary + 500;

3、语法

a += b; ==> a = a + b;

a \*= b; ==> a = a \* b;

4、练习

a=18

b=25

a=25，b=18

2、函数(重难点)

1、什么函数

函数:function，也叫 方法(method)或过程(procedure)

是一段预定义好，并且可以被反复执行的代码块。

预定义：提前先声明好，不是马上使用。

反复执行：可以被循环利用

代码块：允许包含多条可执行语句

函数本质实际上是功能完整的对象。

2、定义 函数(方法)

1、语法

function 函数名(){

若干可执行语句(代码块)

}

函数名：遵循标识符的规范

2、函数调用

允许出现JS脚本的地方都可以实现 "函数调用"

语法：函数名();

parseInt();

parseFloat();

Number();

console.log();

3、定义带参函数

1、语法

function 函数(参数列表){

代码块

}

参数列表：由一个或多个 参数名称 组成，多个参数间，用 , 来 分隔。定义时的参数，称之为 "形参(形式上的参数)"

2、调用

函数名(参数列表);

参数列表：由一个或多个 值 组成，按照定义好的顺序，往里传递。调用函数时所传递的参数值，称之为 "实参(实际参数)"

function parseInt(value){

.......

}

4、带返回值的函数

var num=parseInt("25.5");

语法：

function 函数名([参数列表]){

代码块;

//如果有返回值

return 值/表达式;

}

return:返回

1、将指定的值或表达式带出到函数调用的位置处

2、并终止当前函数的执行

5、练习

1、定义一个可以接受两个数字的方法，返回这两个数字的和，并调用方法，打印输出最终结果

3、变量的作用域

1、什么是作用域

作用域就是变量或函数的可访问范围。控制变量或函数的可见性和生命周期

作用域可分为：

1、函数作用域：局部变量(函数)，只能在当前函数内访问

2、全局作用域：全局变量(函数)，一经定义，代码的任何位置都能访问

2、函数作用域的变量

将 变量 或 函数 声明在某函数内，有效范围就在该函数中，出了该函数，变量 或 函数 就失效（因为函数不调用不执行，所以出了函数就没用了）

3、全局作用域的变量

将 变量定义在 独立于 任何 function 的位置处(<script>中)，在代码的任何位置都可以访问到

1、独立于任何 function 声明变量

2、函数中，通过不写var的方式声明全局变量

4、注意

1、全局变量和局部变量冲突时，优先使用的是 局部变量

5、声明提前

JS程序在正式执行前，会将所有的var声明变量和function的声明函数，预读到所在作用域的顶部。

但是，对变量的赋值依然保留在原位置上。

6、按值传递

基本数据类型 在做参数传递时，实际上将数据进行复制 得到一个副本，将副本传递到函数参数中

4、函数作用域

1、分类

与变量相似，分为

1、全局作用域(全局函数)

2、函数作用域(局部函数)

2、常用全局函数

1、parseInt()

2、parseFloat()

...

3、encodeURI()

uri:Uniform Resource Identifier(统一资源标识符)

作用：对统一资源标识符进行编码，并且返回 编码后的URI字符串

针对 URI 中 非 ASCII码的字符进行编码，转换为16进制的转义字符

ASCII : American Standard Code for Information Interchange

4、decodeURI()

作用：对已编码的URI进行解码，返回解码后的数据

5、eval()

作用：计算并执行以字符串表示的JS代码

3、分支结构(重点)

1、程序的流程控制

程序 = 数据 + 算法

任何复杂程序都可以通过以下三种结构实现

1、顺序结构

将可执行代码自顶向下执

2、选择结构(分支)

根据条件 选择某一块代码去执行

3、循环结构

根据条件 选择某一块代码反复执行

问题：当 总价 大于或等于 500时，享受8折优惠

分支结构：

可根据条件，选择执行某些语句

1、当条件满足时，运行某些语句

当条件不满足时，则不运行这些语句

----if结构

2、当条件满足时，运行某些语句

当条件不满足时，运行另外一些语句

----if - else 结构

2、if 结构

1、语法

if(条件){

满足条件要执行的语句块;

}

流程：

1、判断 条件的值 ：

如果为 true， 则执行if语句块中的语句

如果为 false，则不执行if语句块中的语句

2、注意

1、条件 是 boolean 的

如果 出现的 条件 不是 boolean，则会自动转换

出现以下几种情况，都作为假处理:

if(0){}

if(0.0){}

if(undefined){}

if(NaN){}

if(null){}

if(""){}

if("我帅吗"){

console.log("恭喜你，答对了!");

}

2、可以省略 if 后面的 {}

如果 省略 {} ,那么if 只控制它下面的第一条语句

3、问题

异常情况：如果 收款金额 小于 应收金额

3、if ... else 结构

1、语法

if(条件){

满足条件所执行的代码块(语句块1)

}else{

不满足条件时所执行的代码块(语句块2)

}

流程：

1、判断 条件

如果值为true，则执行语句块1

如果值为false，则执行语句块2

4、成绩计算 程序

手动输入一个考试分数 (0~100)

如果成绩 为 100分 ，奖励一台兰博基尼

如果成绩 大于等于90分，奖励一台 法拉利

如果成绩 大于等于80分，奖励一台 奥迪Q7

如果成绩 大于等于70分，奖励一台 奥拓R8

如果成绩 大于等于60分，奖励一台 自行车

否则 奖励 俩大嘴巴子

4、else.... if 结构 (多重if结构)

1、语法

if(条件1){

语句块1;

}else if(条件2){

语句块2;

}else if(条件3){

语句块3;

}...else if(条件n){

语句块n;

}else{

else 语句块;

}

练习：

1、可莱托指数，通过 选择 结构完成

2、日期计算器

输入 出生的 年，月，日，计算 生日那一天，是那一年的第多少天!!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月24日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1、分支结构-2

1、条件判断

1、各种 if 结构 (if、if...else...、else...if)

2、短路逻辑 条件 && 操作

3、条件运算符（三目运算符）:条件?满足 :不满足;

-- if(){} else{} 相当于如果条件为true，选满足，否者选 不满足

2、做等值判断的 分支 结构

1、switch-case 结构

语法：

（括号里面值为true和false）语义：如果表达式符合值1，就执行case值1 语句

switch(变量/表达式){

case 值1:

语句块1;

break; //可选

case 值2:

语句块2;

break; //可选

... ...

default: //以上任何一块都没匹配上的时候

语句块n;

break; //可选

}

// 直落形式 : 多个 case 执行相同的语句块。

switch(变量/表达式){

case 值1:

case 值2:

case 值n:

语句块;

break;

....

}

2、循环结构-1

1、问题

1、控制台中输出 一句 Hello World

2、控制台中输出 10句 Hello World

3、控制台中输出 1000句 Hello World

4、控制台中，将 1000句Hello World 变成 你好 世界!

5、控制台中，将1000句 你好 世界 的前面加上

第 1 句 你好世界

第 2 句 你好世界

第 1000 句 你好世界

2、循环

1、什么是循环

一遍又一遍的反复执行相同的操作!

生活循环：

1、跑圈 - 1、重复跑 2、圈数限制(12,25)

2、上学 - 1、重复学习过程 2、4个月 时间限制

3、地球公转自转 - 1、重复转 2、限制

特点：

1、循环操作 ：循环体，要重复做的事情是什么

2、循环条件 ：重复执行的次数，或继续执行循环的条件

2、while 循环

1、语法

while(循环条件){

//循环操作

}

语义：当 循环条件 为真时，执行循环操作

流程：

1、判断 循环条件 (boolean表达式或值)

2、如果 结果为 真

2.1 执行循环操作

2.2 再回到条件判断位置处，继续判断条件

3、如果条件为 假

结束循环

3、循环流程控制-break

1、作用

提前 结束 循环

ex : 要求用户 循环的 从网页输入内容，打印在控制台上，直到 输入 exit 为止

循环条件：死循环,输入 exit 则退出循环

循环操作：

1、输入数据

2、判断 如果为 exit ， 则 退出 循环

3、打印输出在控制台上

2、循环和switch 中的 break

while(true){

var n = prompt();

switch(n){

case "1":

console.log(1);

break;

case "2":

console.log(2);

break;

}

}

3、每日一练

猴子吃桃:

猴子第一天摘下若干个桃子，当即吃了一半，还不爽，又多吃了一个

第二天 ，将剩下的桃子吃了一半，又多吃一个

...

以后每天早上都吃了前一天的剩下的一半零一个

第十天 ，只剩下一个桃子了

求：第一天共摘了多少桃子

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月25日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

回顾:

1、分支结构 - switch

1、场合

等值判断

2、语法

switch(变量/表达式){

case 值1:

语句块;

break;

......

default:

语句块;

break;

}

1、default

任何case 都没被匹配上要执行的操作

可有可无

2、结束 switch...case运行

1、break

2、走到switch结束

switch(n){

case 1:

console.log("1");

case 2:

console.log("2");

}

2、循环结构 - while

1、循环特点

1、循环操作 ：重复做的事情是什么!

2、循环条件 ：控制循环的开始与结束!

2、while

2、语法

while(条件){

操作

}

流程：

先判断条件

如果条件为真，则执行操作，然后再判断条件 ... ...

如果条件为假，则退出循环结构

3、循环流程控制 - break

碰到 break 则提前结束循环

练习：猴子吃桃

一半零一个

第10天 剩1个

第10天 剩1x/2-1=1x : 4 (第09天的量)

第09天 剩4x/2-1=4x : 10(第08天的量)

第08天 剩10 x/2-1=10 x : 22(第07天的量)

第07天 剩x

... ...

第02天 剩left x/2-1=left : x ? 第一天的总量

第01天 剩x

循环条件：9次 从10开始，到2结束

循环操作：

left : 1- 第10天所剩的桃子数量

total: ? - 第一天所剩的桃子总数

公式：total/2-1=left

total/2 = left+1

total = (left+1)\*2

total = (left+1)\*2;

left=total;

输入 年 月 日 判断 ，该日是概念的第多少天，用循环来做

提示：月份，是循环判断的!

循环

1、do...while 循环

1、问题

要求用户从网页中输入文本，在控制台上打印输出，直到输入 exit 为止

方式1:

while(true){

//输入数据

//判断是否为 exit，不是exit 打印输出

}

方式2:

var input="";

while((input=prompt("input")) != "exit"){

//打印

}

2、特点

先 执行 循环体

再 判断 条件

3、语法

do{

循环操作;

}while(循环条件);

4、while 和 do...while 的区别

1、while 循环

先判断，再执行

最少 循环体 一次都不被执行

2、do...while 循环

先执行，再判断

任何情况下，至少都要执行一次循环体

两种循环只在第一次执行时有差别，其他情况，while与do...while 是完全一致。

5、练习

将 while 版的 猜数字 游戏，改版成 do...while版

1、条件

1、用户输入的数字与生成的随机数一样(结束)

2、用户输入的exit(结束)

2、操作

1、循环录入数据

2、判断(exit?大了?小了？相等)

2、for 循环

1、while

通过 while 打印 1-100 之间所有的数字

条件：从1 开始 到 100结束

操作：打印 循环变量

var i=1;//循环条件 起始数值(表达式1)

while(i<=100){ //条件判断(表达式2)

console.log(i); //循环操作(循环操作)

i++;//更新循环变量(表达式3)

}

特点：固定循环次数

2、for 循环

1、特点

将 while 循环中的 几个重要因素放在一起来声明,通过一个大的完整的表达式来声明 循环所需要用到的东西

2、语法

for(表达式1;表达式2;表达式3){

循环操作!

}

表达式1：循环变量的声明表达式.

表达式2：循环条件判断，条件为真，执行循环体，否则退出循环。

表达式3：更新循环变量

流程：

1、计算表达式1的值

2、计算表达式2的值(boolean),如果条件为真，则执行循环体，否则 退出

3、执行循环体

4、计算表达式3的值

5、计算表达式2的值(boolean),如果条件为真，则执行循环体，否则 退出

6、同3

7、同4

3、场合

一般用于 固定循环次数时 使用

3、for语句三个表达式的特殊用法

1、表达式1允许为空

允许在循环外，将第一个表达式不足

2、表达式2允许为空

死循环

尽量不要为空，如果为空的话，尽量在循环内部将条件补充完整额。

3、表达式3允许为空

死循环

在 循环 内部 将表达式补充完整。

4、表达式1 和 表达式3 内容多样化

表达式1 可以一次声明多个变量

表达式3 可以一次改变多个变量的数值

多个表达式之间 用 , 分开

4、循环流程控制

1、break

可以提前终止循环的执行

能用在循环结构中，也可以用在switch结构中

2、continue

1、只能用于循环中

2、作用

跳过本次循环中的剩余语句，而继续执行下次循环

3、三种循环的使用场合

1、确定循环次数的时候 优先使用 for循环

2、不确定循环次数的时候，优先使用while和do..while

1、先判断条件，再决定是否走循环体：while

2、先走循环体，再根据条件判断是否执行下次循环：do...while

4、循环嵌套

在一个循环中 又嵌入了另外一个循环

for(var i=1;i<=10;i++){

for(var j=1;j<=10;j++){

console.log("Hello World");

}

}

1\*1=1

1\*2=2 2\*2=4

1\*3=3 2\*3=6 3\*3\*=9

......

1\*5=5 2\*5=10 3\*5=15 4\*5=20 5\*5=25

1,2,3,4,5 : 循环条件

操作：循环条件 和 打印行数相乘 计算结果

1,1,2,3,5,8,13,21,34,55.....

n1 = 1,n2 = 1

console.log(n1);console.log(n2);// 1,1

n1=n1+n2;

n2=n1+n2;

console.log(n1);console.log(n2);// 2,3

n1=n1+n2;

n2=n1+n2;

f(n) : 第 n 个月 兔子总量

f(n) = f(n-1)+f(n-2)

f(5) = f(4)+f(3)

f(4) : 3

f(3) : 2

f(2) : 1

f(1) : 1

\*第1行1个

\*\* 第2行2个

\*\*\*第3行3个

\*\*\*\* 第4行4个

\*\*\*\*\*第5行5个

\* 第1行1个星第1行4个空格5-1=4

\*\* 第2行2个星第2行3个空格5-2=3

\*\*\* 第3行3个星第3行2个空格5-3=2

\*\*\*\* 第4行4个星第4行1个空格5-4=1

\*\*\*\*\* 第5行5个星第5行0个空格5-5=0

j : 行数

k : 空格 k<=(5-j)

i : 星星 i<=j

\* 第1行1个星第1行4个空格

\*\*\*第2行3个星第2行3个空格

\*\*\*\*\* 第3行5个星第3行2个空格

\*\*\*\*\*\*\*第4行7个星第4行1个空格

\*\*\*\*\*\*\*\*\* 第5行9个星第5行0个空格

第二阶段

JAVASCRIPT核心

张东

zhangdong@tedu.cn

犀牛书

微信公众号: 前端大全

js1k

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月26\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

正课:

1. \*\*\*数组:

什么是:内存中，连续存储多个数据的存储空间，再起一个名字

一组变量的集合

为什么:现实中存储多个相关数据，都是集中存储，共用同一个名字。

程序=数据结构+算法

好的数据结构可以极大的提高程序的执行效率

何时:今后只要连续存储多个相关的数据都要用数组

如何使用:

创建数组: 3种:

1.创建空数组: 2种:

1.数组直接量: var arr=[];

2.用new: var arr=new Array();

2.创建数组同时，初始化数组的数据:

1.数组直接量: var arr=[值1,值2,...]

2.用new:

var arr=new Array(值1,值2,...);

3. 创建n个空元素的空数组:

只能用new: var arr=new Array(n);

访问数组中的元素:

下标: 数组中唯一标识一个元素存储位置的序号

默认从0开始，连续递增，不重复

arr[i]: 访问下标为i位置的元素

数组中，每个元素的用法和普通变量完全一样。

数组的length属性:

标识了数组中理论上的元素个数

任何情况下: length永远是最大下标+1

固定套路:

1.最后一个元素: arr[arr.length-1]

2.倒数第n个元素: arr[arr.length-n]

3.追加新元素:arr[arr.length]=新值;

4.缩容: arr.length--;

特殊情况: js中的数组:3个不限制

1. 不限制元素的数据类型

2. 不限制下标越界

获取元素值:

下标越界: 不报错，返回undefined

3. 不限制元素的个数

设置元素值:

下标越界:

自动在指定位置添加新元素

自动修改length为最大下标+1

稀疏数组: 下标不连续的数组

遍历数组: 依次访问数组中每个元素

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

arr[i]//当前元素

}

数组是引用类型的对象:

正课:

1. \*\*\*数组

垃圾回收

关联数组

数组API

垃圾回收: 定期回收不再被任何变量引用的对象。释放内存。

1. 垃圾回收器:

伴随主程序，并行运行的一个小程序

定期回收内存中的不再被任何变量引用的对象。

2. 引用计数器:

每个对象上，标记正在引用对象的变量个数的一个属性

每多一个变量引用对象，计数器就+1

如果一个变量通过赋值为null释放对对象的引用，则计数器-1

直到计数器为0，表示不再有任何变量引用该对象，则该对象被回收/释放

建议: 如果使用完较大的对象，都要主动赋值为null。

索引数组: 下标为数字的数组

关联数组:

什么是:可自定义下标名称的数组

为什么:普通的索引数组的下标都是无意义的数字。不便于快速查找元素。也容易出现歧义.

[林心如, 81, 53, 86]

Namemath chs eng

何时:希望每个元素有专门的名称时

通常用于快速定位元素

如何使用:

创建: 2步:

1.创建空数组: var lxr=[];

2.向空数组中添加新元素:

使用自定义的字符串元素名作为下标

lxr["name"]="林心如";

lxr["math"]=81;

lxr["chs"]=53;

lxr["eng"]=86;

访问元素:

lxr["name"]

强调: 关联数组的length失效

遍历: 不能用for循环

用for in循环：

其中in,表示依次取出数组中每个下标

for(var key in hash){

key//每个元素的下标名称

hash[key]//当前元素值

}

关联/hash数组原理:

hash算法: 接收一个字符串，并计算出一个尽量不重复的序号

相同的字符串，计算出的号总是一样的

不同的字符串，计算出的号大多数不重复

存入数据:

hash算法接收一个字符串的下标名，计算出一个不重复的序号。将数据存储在序号对应的数组位置上

获取数据:

将下标名称交给hash算法，算出和存入时完全相同的序号，直接去序号位置查找元素

优点: 查找极快!

和数组中元素个数，以及元素在数组中的位置无关。

数组API:

对象: 集中存储多个数据的一块存储空间

并提供了操作数据的API

函数: 不属于任何对象的函数

方法: 必须用对象才能调动的函数，叫方法

1. 转字符串: 2种

1. 将数组中每个元素都转为字符串，再用逗号分隔: var str=String(arr);

vars = "abc,abcd,aaa";

2. 将数组中每个元素都转为字符串，再用自定义下标连接每个元素:

var str=arr.join("自定义连接符")

强调:如果省略参数，默认等效于String

固定套路:

1. 无缝拼接: arr.join("")

2. 将单词拼接为句子: arr.join(" ")

3. 判断空数组: arr.join("")==""

4. 根据数组中的元素，动态生成页面元素的内容

2. 拼接和截取:

拼接:

var newArr=

arr1.concat(值1,值2,arr2,....)

将值1,值2,以及arr2中每个元素依次拼接到arr1结尾，返回新数组。

何时使用: 将多个元素或其他数组，拼接到当前数组上时

强调:

1.concat无权修改原数组，只能返回新数组

所以，必须用变量接住拼接后的返回值

2.concat可打散数组类型的参数为单个元素

Var arr2=[2,3,45,67,76];

Var arr=[78,43,3,56,232];

Var newarr=arr.concat(arr2);

Console.log（newarr）；//78,43,3,56,232,2,3,45,67,76

截取:(复制)

var subArr=arr.slice(starti,endi+1);

复制arr中starti位置开始到endi结束的元素组成子数组返回。

何时: 选取数组中指定位置的子数组时

强调: 1. 无权修改原数组，只返回新子数组

2.含头不含尾

3.其实可使用负数参数,表示倒数第n个

简写:

1. 省略第二个参数，表示一直截取到结尾

2. 两个都省略，表示完整复制原数组中的元素到新数组。

不能只省略第一个参数——报错

3.splice: 删除，插入，替换

删除数组中指定位置的元素：

arr.splice(starti,n):

删除arr中starti位置开始的n个元素

其实: var deletes=arr.splice(starti,n)

deletes: 返回的被删除的元素组成的临时子数组。

在数组指定位置插入新元素:

arr.splice(starti,0,值1,值2,...)

在starti位置插入新值

原starti位置的值被向后顺移

强调: splice，不支持打散数组参数

插入: splice vs concat

concat只能在开头或结尾拼接，不能中间插入。但可打散数组类型参数

splice可在任意位置插入，但不可打散数组类型参数。

替换数组中指定位置的元素

arr.splice(starti,n,值1,值2,...)

先删除starti位置的n个旧元素

再在starti位置插入新值

强调: 删除的个数和插入的个数不必一致

splice可自动调节数组的长度和位置

翻转: arr.reverse();

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月29\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

回顾:

数组:

创建：

1. 创建空数组:

1. 直接量: var arr=[];

2. 用new: var arr=new Array();

2. 创建数组同时，初始化数组元素:

1. 直接量: var arr=[值1,值2,...]

2. 用new:

var arr=new Array(值1,值2,...);

3. 创建n个空元素的数组:

用new: var arr=new Array(n);

访问元素:

arr[i]

三个不限制:

1. 不限制元素的数据类型

2. 不限制元素个数——添加元素时

不报错, 在任意位置添加新元素

3. 不限制下标越界——获取元素时

不报错，返回undefined

length: 标识理论上的元素个数

固定套路:

1. 最后一个: arr[arr.length-1]

2. 倒数第n个: arr[arr.length-n]

3. 末尾追加: arr[arr.length]=值

4. 缩容: arr.length-=n;

遍历: for(var i=0;i<arr.length;i++){

arr[i]//当前元素

}

索引数组: 下标是数字的数组

稀疏数组: 下标不连续的数组

关联/hash数组: 可自定义下标名称的数组

创建: 2步:

Step1: 创建空数组:

var lxr=[];

Step2: 向空数组中添加新元素，使用自定义的下标名称

lxr["name"]="林心如"

访问元素:lxr["name"]

length属性: 失效，始终为0

遍历: for(var key in hash){

key//依次获得hash中每个下标名称

hash[key]//当前元素

}

数组API:

1. 转字符串:2种

1. var str=String(arr)

2. var str=arr.join("自定义分隔符")

固定套路:

1. 无缝拼接: arr.join("")

2. 将单词拼接为句子: arr.join(" ")

3. 判断是否空数组: arr.join("")==""

4. 将数组内容动态生成页面元素:

<标签>+

arr.join("</标签><标签>")

+</标签>

2.拼接和选取:

拼接:

var newArr=arr1.concat(值1,值2,arr2,..)

强调: 可打散数组参数为单个元素

选取:

var subArr=arr.slice(starti,endi+1)

强调: 含头不含尾

支持负数(倒数)下标

其实,arr.length-n

简写: 省略第二个参数，一直选取到结尾

两个参数都省略, 完整复制原数组

3. splice:

删除: var deletes=arr.splice(starti,n)

强调: 直接修改原数组

返回被删除的元素组成的临时数组

插入: arr.splice(starti,0,值1,值2,...)

强调: 不能打散数组类型参数

替换: arr.splice(starti,n,值1,值2,...)

强调: 插入的新元素个数不必和删除的元素个数一致。

4. 翻转: arr.reverse();

正课:

1. \*\*\*数组:

\*\*\*\*排序

栈和队列

二维数组

2. \*\*\*String:

\*\*\*\*排序:

自定义排序算法: 冒泡，插入，快速

冒泡: 从头开始遍历数组中每个元素

如果当前元素>下一个元素

就交换两数的位置

交换两变量值:a,b

1. 声明第三个变量:

var t=a;

a=b;

b=t;

2. 不声明第三个变量: 仅能交换数字

a+=b; b=a-b; a-=b;

3. 不声明第三个变量: 万能

b=[a,a=b][0];

使用排序API:

arr.sort(); 默认，仅能对字符串升序排列

解决: 自定义比较器函数

什么是比较器函数:

专门用来比较任意两数大小的函数

要求:

1. 2个参数,a,b

2. 如果a>b，就返回正数

如果a<b，就返回负数

如果a==b,就返回0

最简单的数字比较器:

function cmp(a,b){return a-b;}

如何使用比较器函数：

将比较器函数对象，作为sort方法的参数

arr.sort(cmp);

强调: cmp后，不要加()

区别: 不加()，是将函数对象放入sort中，sort可反复调用执行。

加()，仅是将函数的执行结果，放入sort中，sort方法无法反复调用。

降序排列: 颠倒比较器返回值的正负

比如:

function cmp(a,b){return -(a-b)};

栈和队列:

js中没有专门的栈和队列类型，都是用普通该数组模拟的。

何时: 只要希望按照顺序使用数组元素时

栈: 一端封闭，只能从另一端进出的数组

顺序: FILO

何时: 如果希望始终操作最后进入数组的最新的元素时。

如何:

结尾出入栈:

入: arr.push(值);

出: var last=arr.pop();

优: 每次出入栈不影响其余元素的位置

开头出入栈:

入: arr.unshift(值);

出: var first=arr.shift();

缺: 每次出入栈都会影响每个元素的位置

队列: 只能从一端进入，从另一端出

顺序: FIFO

何时: 只要按照先来后到的顺序使用元素

如何:

从结尾入队列: arr.push(值)

从开头出队列: var first=arr.shift()

正课:

1.二维数组

2.\*\*\*String

1. 二维数组:

什么是: 数组中的元素，又引用了另一个子数组。

何时:

1. 存储横行竖列的二维数据

2. 对一个大的数组中的元素，进一步分类

如何:

创建:

1. 先创建空数组，再添加子数组:

var arr=[];

arr[0]=[2,2,4,0];

arr[1]=[0,2,2,0];

2. 在创建数组同时，初始化子数组

var arr=[

[2,2,4,0],//0

[0,2,2,0],//1

...

];

访问元素:

arr[r][c]:访问arr中r位置的子数组中的c位置的元素

越界: r不能越界

遍历: 外层循环控制行，内层循环控制列

for(var r=0;r<arr.length;r++){

for(var c=0;c<arr[r].length;c++){

arr[r][c]//当前元素

}

}

2. \*\*\*String:

什么是: 一串字符组成的**只读字符数组**

vs数组:

1. 下标访问每个字符

2. length属性标识字符个数

3. 用for循环遍历每个字符

4. 数组中不直接修改原数组的API，字符串也可用concat,slice

内置对象: ES标准中规定的，浏览器已经实现的对象。包括11个:

String Number Boolean——包装类型

Array Math Date RegExp

Error

Function Object

Global(在浏览器中被替换为window)

包装类型:专门封装原始类型的值，并提供对原始类型的值进行操作的API 的对象

为什么:原始类型的值，本身什么API都没有

何时: 只要试图用原始类型的值调用方法时

都会自动创建包装类型对象

如何使用: 自动创建和调用。

StringAPI: 所有StringAPI，都无权修改原字符串，必须返回新字符串

1. 大小写转换:

何时使用: 不区分大小写时

str.toUpperCase();

str.toLowerCase();

2. 获取指定位置的字符:

str.charAt(i) => str[i]

str.charCodeAt(i)

获得str中i位置的字符的unicode号

互逆操作: 将unicode号转为原字符

var char=String.fromCharCode(unicode);

3. 获取子字符串:

str.slice(starti,endi+1);

str.substring(starti,endi+1);

用法和slice完全一样。

不支持负数参数:

解决: str.length-n

str.substr(starti,n);

截取str中starti开始后的n个字符

4.检索关键词:

1. 检索一个固定的关键词出现的位置

var i=str.indexOf("关键词",fromi);

从str中fromi位置开始，找下一个"关键词"出现的位置。

如果找不到了，返回-1

简写: 省略fromi，表示从0位置开始找

var i=str.lastIndexOf("关键词",fromi)

从str中fromi位置开始查找前一个"关键词"出现的位置

如果找不到，返回-1

简写: 省略fromi, 表示从末尾开始找

何时: 只要查找最后一个xxxx

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月30日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

回顾:

1.数组:

排序: arr.sort();

function cmp(a,b){return a-b};

arr.sort(cmp);

function cmp(a,b){return b-a};

栈和队列:

栈: FILO

结尾: 入: arr.push(值)

出: var last=arr.pop(值);

开头: 入: arr.unshift(值)

出: var first=arr.shift();

队列:FIFO

结尾入:arr.push(值)

开头出:var first=arr.shift();

二维数组:

遍历:

for(var r=0;r<arr.length;r++){

for(var c=0;c<arr[r].length;c++){

arr[r][c]//当前元素

}

}

2. String:

大小写转换:

str.toUpperCase();

str.toLowerCase();

获得指定位置的字符:

str.charAt(i) => str[i]

var unicode=str.charCodeAt(i)

var char=String.fromCharCode(unicode)

获取子字符串:

str.substring(starti,endi+1)

=> str.slice(starti,endi+1)

str.substr(starti,n)

检索关键词:

1. 仅检索一个固定关键词出现的位置

var i=str.indexOf("关键词",fromi);

找所有位置:

while(

(i=str.indexOf("关键词",i+1))!=-1{

i//就是找到关键词的位置

}

找最后一个:

var i=

str.lastIndexOf("关键词",fromi)

问题:每次只能检索一个固定的关键词，无法实现按规则模糊检索

正课:

1. \*\*\*\*\*正则表达式：

2. \*\*\*String中的正则API

查找

替换

切割

1. 正则表达式（Regular Expression）

什么是: 规定字符串中字符出现规律的规则

何时使用:

1. 只要按规则模糊匹配多个关键词时

2. 只要按规则验证字符串是否符合要求

查看正则的逻辑: regexper.com

语法:

1. 最简单的正则，就是关键词原文

2. 字符集: 规定1位字符可用的备选字符的列表: [备选字符列表]

何时使用: 如果一位字符，有多种选择时

简写: 如果字符列表中部分字符连续,就可用-省略中间的字符

比如: 1位字母:[A-Za-z]

1位数字:[0-9]

除了: [^xxx] 除了xxx都行

3. 预定义字符集:

对部分常用字符集的简化

1位数字 \d=> [0-9]

1位字母，数字,\_ \w => [A-Za-z0-9\_]

1位空字符(空格，换行，Tab): \s

. 匹配除回车换行外的所有字符

强调: 只有需求和预定义字符集完全匹配时，才能使用预定义字符集。

4. 量词: 专门规定一个字符集出现次数的规

则。

有明确数量限制:

字符集{n,m}字符集可出现至少n次，最多m次

字符集{n,}字符集至少出现n次，多了不限

字符集{n}字符集必须出现n次

没有明确数量显示:

字符集? 可有可无，最多一次

字符集\* 可有可无，多了不限

字符集+ 至少一次，多个不限

强调: 字符集，默认仅修饰相邻的前一个字符集。

身份证号:

前15位数字

再跟2位数字

最后一位可以是数字或x,X

\d{15}(\d{2}[0-9xX])?

5. 分组和选择:

分组: 将多个字符集用()包裹为一组

何时: 为了让一个两次同时修饰多个字符集。

选择: 其实就是程序中的或者 |

规则1|规则2只要满足其一即可

手机号:

+86或0086? 可有可无，最多一次

空字符可有可无，多了不限

1

在3,4,5,7,8之中挑一个

9位数字

^(\+86|0086)?\s\*1[34578]\d{9}$

6.指定匹配位置:

^ 字符串开始位置

$ 字符串结尾位置

何时: 1. 验证时，都要前加^，后加$

2. 仅匹配开头的或结尾的关键词

密码强度:

6~8位字母，数字的组合，至少包含一位大写字母和一位数字

预判: 从头到尾判断字符串是否符合指定规则。

1. 如果不都由字母组成:

说明，可能有数字或特殊符号

(?![A-Za-z]+$)

2. 如果不都由小写字母和数字组成:

说明，可能有大写字母或特殊符号

(?![a-z0-9]+$)

3. 必须由字母和数字组成:

排除特殊字符的干扰

[A-Za-z0-9]{6,8}

^(?![A-Za-z]+$)(?![a-z0-9]+$)[A-Za-z0-9]{6,8}$

2. String正则API

检索关键词:

1. indexOf: 不支持正则

2. 判断是否包含符合规则的关键词:

var i=str.search(/正则/);

查找str中是否包含符合正则要求的关键词。

如果找到，返回关键词的位置

如果找不到，返回-1

强调: search没有第二个参数fromi，不能指定下次开始查找的位置，始终只能找第一个关键词的位置。所以，只能用作判断有没有。

正课：

1. String中的正则API

2. RegExp

1. String中的正则API:

检索: 4种:

1. 查找一个固定关键词的位置

var i=str.indexOf("关键词",fromi);

var i=str.lastIndexOf("关键词",fromi)

问题: 不支持正则

2. 判断字符串中是否包含符合规则的关键词

var i=str.search(/正则/);

始终返回第一个关键词的位置

如果没找到，返回-1

问题: 1. 只能找第一个

2. 只能获得位置，无法获得内容

3. 获得每个关键词的内容:

var kwords=str.match(/正则/ig);

获取str中所有和正则匹配的关键词

返回: 一个包含所有关键词的数组

没找到，返回null

强调: 只要一个方法可能返回null，就都要先验证不是null，再使用。

强调: 所有正则表达式默认只匹配一个关键词。除非在第二个/后加g,表示全部。

如果希望忽略大小写: 加i

问题: 无法获得每个关键词的位置

如果match不加g:

仅返回第一个关键词的内容和位置

kwords:[0:"关键词",index:i]

4. 即获得每个关键词的内容，又获得每个关键词的位置？

替换: 将str中和正则匹配的敏感词替换为指定内容。

str=str.replace(/正则/ig,"替换值");

高级: 如果希望根据关键词的不同，替换不同的内容时:

str=str.replace(/正则/ig,

function(kwd,$1,...){

return 根据kwd返回不同的值;

}

)

$1,$2,...:如果需要对关键词中部分内容单独处理时:2步:

1. 在正则中用()，将要单独操作的部分分组。在匹配时，每个分组会自动获得从1开始，连续不重复的序号

2. 在replace内部的函数的参数中可用$1,$2...获得第i个分组的子内容。

练习:

补:正则:

\b: 单词边界: ^ 空格 标点 $

何时使用: 今后，只要匹配单词，必须前后加\b。

扩展:

1. 删除:将关键词替换为空字符串:

2. 格式化: 2步:

1. 用正则将原字符串分组

2. 在replace第二个参数中用$i方式拼接新的格式。

补:正则: 一位汉字:[\u4e00-\u9fa5]

切割: 将字符串，按匹配的分隔符，分割为多个子字符串

var subs=str.split(/正则/);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8月31日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

正课：

1. String中的正则API

2. RegExp

1. String中的正则API:

检索: 4种:

1. 查找一个固定关键词的位置

var i=str.indexOf("关键词",fromi);

var i=str.lastIndexOf("关键词",fromi)

问题: 不支持正则

2. 判断字符串中是否包含符合规则的关键词

var i=str.search(/正则/);

始终返回第一个关键词的位置

如果没找到，返回-1

问题: 1. 只能找第一个

2. 只能获得位置，无法获得内容

3. 获得每个关键词的内容:

var kwords=str.match(/正则/ig);

获取str中所有和正则匹配的关键词

返回: 一个包含所有关键词的数组

没找到，返回null

强调: 只要一个方法可能返回null，就都要先验证不是null，再使用。

强调: 所有正则表达式默认只匹配一个关键词。除非在第二个/后加g,表示全部。

如果希望忽略大小写: 加i

问题: 无法获得每个关键词的位置

如果match不加g:

仅返回第一个关键词的内容和位置

kwords:[0:"关键词",index:i]

4. 即获得每个关键词的内容，又获得每个关键词的位置？

替换: 将str中和正则匹配的敏感词替换为指定内容。

str=str.replace(/正则/ig,"替换值");

高级: 如果希望根据关键词的不同，替换不同的内容时:

str=str.replace(/正则/ig,

function(kwd,$1,...){

return 根据kwd返回不同的值;

}

)

$1,$2,...:如果需要对关键词中部分内容单独处理时:2步:

1. 在正则中用()，将要单独操作的部分分组。在匹配时，每个分组会自动获得从1开始，连续不重复的序号

2. 在replace内部的函数的参数中可用$1,$2...获得第i个分组的子内容。

练习:

补:正则:

\b: 单词边界: ^ 空格 标点 $

何时使用: 今后，只要匹配单词，必须前后加\b。

扩展:

1. 删除:将关键词替换为空字符串:

2. 格式化: 2步:

1. 用正则将原字符串分组

2. 在replace第二个参数中用$i方式拼接新的格式。

补:正则: 一位汉字:[\u4e00-\u9fa5]

切割: 将字符串，按匹配的分隔符，分割为多个子字符串

var subs=str.split(/正则/);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月1日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

回顾：

1. RegExp:

检索: var arr=reg.exec(str);

while((arr=reg.exec(str))!=null){

arr[0]//本次找到的关键词

arr[i],或RegExp.$i //第i个分组的子内容

arr.index//本次找到的关键词位置

}

验证: var bool=reg.test(str);

前加^，后加$

2. Math:

取整: var n=4.5

上取整: Math.ceil(n); //5

下取整: Math.floor(n); //4

四舍五入取整: Math.round(n);

vs n.toFixed(d)

function round(num,d){

return Math.round(num\*Math.pow(10,d))

/Math.pow(10,d)

}

乘方和开平方:

Math.pow(底数,幂);

Math.sqrt(n)

正课:

1. Math:

max，min

random

2. Date:

1. Math: 不能new！

最大值和最小值:

Math.max(值1,值2,...)

获得给定值中的最大值

Math.min(值1,值2,...)

获得给定值中的最小值

问题: max和min不支持数组参数

解决: Math.max.apply(null,arr)

获得arr中的最大元素

随机数:

Math.random() -> 返回0<=r<1之间的一个随机小数。

在0~max之间取随机整数:

Math.floor(Math.random()\*(max+1));

在min~max之间取随机整数:

Math.floor(Math.random()\*(max-min+1)+min)

2. Date: 封装一个时间点，并提供操作时间的API的对象

何时使用: 只要存储日期或时间，或者要对时间进行计算时。

如何使用:

创建: 3种:

1. 创建一个日期对象，同时自动获得客户端当前系统时间:

var now=new Date();

2. 创建一个日期对象，并封装自定义时间:

var date=

new Date("yyyy/MM/dd hh:mm:ss");

var date=

new Date(yyyy,MM-1,dd,hh,mm,ss)

3. 复制一个日期对象:

为什么:为了保留计算之前的原时间

何时: 只要要求同时保留计算前后的两个时间时，都要先复制原时间，再用副本计算新时间。

如何: var date1=new Date("xxx")

var date2=new Date(date1);

日期对象的原理:

日期对象中封装的是一个从1970年1月1日0点至今的毫秒数

**Date API**:

**分量:** FullYear Month Date Day

Hours Minures Seconds Milliseconds

1. 每个分量都有一对儿getXXX/setXXX方法

getXXX方法负责获取分量的数值

setXXX方法负责将新值更新到分量

强调: set可自动调整时间进制

特殊：Day 星期，没有set

2. 年月日星期，都是单数，不带s

时分秒毫秒，都是复数，带s

3. 返回值:(取值范围)

除了月中的日(Date)，是从1开始到31结束

其余都是从0开始，到进制-1结束

比如:

Month: 0~11 在赋值或取值时都要修正

Date: 1~31

Hours: 0~23

Minutes/Seconds: 0~59

Day: 0~6 日,一,二,...,六不用修正

计算:

1. 两日期对象可相减,结果毫秒差

**分量:** FullYear Month Date Day

Hours Minures Seconds Milliseconds

1. 每个分量都有一对儿getXXX/setXXX方法

getXXX方法负责获取分量的数值

setXXX方法负责将新值更新到分量

2. 对任意分量做加减:3步:

1. 取分量: date.getXXX();

2. 做加减:

3. set回去: date.setXXX(n)

简写: **date.setXXX(date.getXXX()+n)；**

**例子：var now=new Data();//获取当前时间**

**将当前时间加3年**

**Now.setFullyear(now.getFullyear()+3)//加3年**

date to string: 日期对象转字符串

date.toString()/toGMTString();

date.toLocaleString(); 转为当地时间格式，即包含日期，又包含时间

date.toLocaleDateString(); 转为当地时间格式，仅包含日期

date.toLocaleTimeString(); 转为当地时间格式，仅包含时间

正课：

1. 错误处理

2. \*\*\*\*\*Function

1. 错误处理:

错误: 程序在执行过程中遇到的异常中断

当发生错误时，程序会报错，退出。

错误处理: 即使发生错误，也保证程序不中断退出的机制。

如何:

try{

可能发生错误的正常逻辑

}catch(err){

只有发生错误时才执行的错误处理代码:

//包括: 1. 记录日志:

//2. 提示用户

//3. 保存数据

}finally{

//无论是否出错，都一定会执行的代码

//包括: 释放内存

}

其中err:

一旦try中的程序发生错误，js会自动创建一个Error类型的对象。

Error:专门封装错误信息的对象。

6种:

SyntaxError: 语法错误

ReferenceError: 引用错误，要使用的变量，或对象，没找到。

TypeError: 类型错误,指错误的使用了类型的方法。

RangeError: 参数超范围

URIError,EvalError

如果可以提前预知错误的原因，就可用if...else...代替try catch

判断浏览器兼容性:

try{

new XMLHttpRequest

();

提示: 支持Ajax

}catch(){

提示: 不支持Ajax

}

也可用if...else代替

鄙视题: try catch中的return

如果finally中有return，则使用finally中的return返回

如果finally中没有return，try中的return，会先获得要返回的值，但挂起，暂不返回。要等待finally中代码执行完，才返回。

主动抛出错误:

何时: 当API的开发者，向API的调用者提示错误时。

如何抛出: throw new Error("信息")

2.\*\*\*\*\*Function

js中的一切函数都是一个对象

创建函数: 3种:

1. 声明方式:

function 函数名(参数列表){

函数体;

return 返回值;

}——被声明提前(hoist)

**其中:function是创建一个新函数的意思**

函数其实是一个对象

函数对象中封装的不是数据，而是代码段。

函数名仅是引用函数对象的普通变量

2. 函数直接量:

var 函数名=function(参数列表){

不会被声明提前。

何时: 如果不希望函数定义被提前时，就必须用直接量。

3. 用new:

var 函数名=

new Function("参数名1",...,"函数体")

何时: 如果函数的参数名和函数体可能在运行时动态生成。

\*\*\*重载(overload): 相同函数名，不同参数列表的多个函数，在调用时，可根据传入的参数不同，自动选择对应的函数执行。

为什么: 为了减少函数的个数，减轻使用者的负担。

如何: js中默认不支持重载的语法

变通: arguments对象

函数中自动创建的，接收所有传入函数的参数值得类数组对象。

类数组对象: 长的像数组的对象

vs 数组: 相同: 1. 用下标访问元素

2. length属性

不同: 不是数组类型，无法使用数组类型的API。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月2日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

回顾:

1. Math:

Math.random()

0~max:

Math.floor(Math.random()\*(max+1))

min~max:

Math.floor(Math.random()\*(max-min+1)+min)

2. Date:

创建:3种:

1. 创建日期对象并自动获得当前客户端系统时间:

var now=new Date();

2. 创建日期对象保存自定义时间:

var date=

new Date("yyyy/MM/dd hh:mm:ss");

var date=

new Date(yyyy,MM-1,dd,hh,mm,ss);

3. 复制一个日期对象:

var date2=new Date(date1);

Date API:

FullYear Month Date Day

Hours Minutes Seconds Milliseconds

1.每个分量都有一对儿get/set方法

Day没有set

2.年月日星期 单数,不带s

时分秒毫秒 复数,带s

3.月中的日(Date): 1~31

其余都是: 0~进制-1

只有Month需要修正: 计算机中的月比现实中少1

计算:

1. 两日期对象可相减,结果是毫秒差

2. 对任意分量做加减:

date.setXXX(date.getXXX()+n)

to String:

date.toString();

date.toGMTString();

date.toLocaleString();

date.toLocaleDateString();

date.toLocaleTimeString();

3.错误处理:

try{

可能出错的正常代码

}catch(err){

只有出错才执行的错误处理代码

}finally{

无论是否出错总会执行的代码

}

Error:

SyntaxError,ReferenceError,TypeError

RangeError, URIError, EvalError

主动抛出异常:

throw new Error("错误信息");

4.Function:

创建：

1. 声明:

function 函数名(参数列表){

函数体

return 返回值;

}——被声明提前(hoist)

2. 直接量:

var 函数名=function(参数列表){

——不会被声明提前

3. 用new:

var 函数名=

new Function("参数名",...,"函数体")

js中，函数其实就是一个对象。

封装了一段可反复执行的代码段。

函数名其实只是一个引用函数对象的变量

重载:(overload)

何时: 如果一件事，根据不同的参数，执行不同的逻辑时

如何:

arguments: 函数中自动创建的

自动接收所有传入函数的参数值得

类数组对象

arguments[i] //获得第i个参数值

arguments.length //获得参数个数

正课:

**1.\*\*\*\*\*Function**

**匿名函数**

**\*\*\*\*作用域和作用域链**

**\*\*\*\*\*闭包**

1.匿名函数:

什么是: 在创建时，不被任何变量引用的函数。

为什么: 节约内存

何时: 只要一个函数只用一次，就释放

如何使用:

1.回调: 创建一个函数后，将函数对象传递给其他函数调用。

比如:

arr.sort(

function(a,b){return a-b}

);

arr.replace(reg,function(){

return xxx;

})

2.自调: 创建一个函数，然后立刻调用执行

执行后，立刻释放

(function(参数列表){

函数体;

return 返回值;

})(参数值);

何时: 如果一项任务仅执行一次，且使用了仅属于当前任务的变量时，就要将变量和任务代码封装在一个匿名函数中，自调

\*\*\*\*作用域和作用域链:

作用域(scope): 一个变量的可用范围

本质: 存储变量的对象

函数的原理:

程序开始执行:

创建ECS(Execution Context Stack):

顺序保存正在调用的函数的执行环境

向ECS中压入第一个EC:

EC:函数调用时，引用所有变量的元素。

创建全局作用域对象window:

EC引用window

函数定义时: 创建函数对象

在全局创建函数名变量引用函数对象

在函数对象中添加一个scope属性:

引用函数来自的作用域对象

函数调用时:

向ECS中压入要调用函数的执行环境EC

创建函数作用域对象(活动对象AO):

AO:保存本次函数调用时，使用的所有局部变量的对象。

在AO中添加parent属性，引用函数对象的scope属性指向的作用域。

EC中的scope chain属性引用AO对象

函数调用后: 本次调用的函数的EC出栈

导致：AO释放 Ao局部作用域

导致: 局部变量一同释放 所以说函数一调用，他里面的局部变量就释放了，下次来这个局部变量还是原来的值

作用域链(scope chain):

由各级作用域对象，逐级引用形成的链式结构

控制变量的使用顺序:

优先使用AO中的局部变量

如果局部没有，才延作用域链向下找，到window

如果window也没有才报错

\*\*\*\*\*闭包(closure):

**为什么说重用局部变量，因为局部变量，在函数调用后就释放了不可重用**

重用局部变量，并保护变量不受污染的机制

为什么:

全局变量: 随处可用，可反复使用

缺: 极易被污染

局部变量: 不会被污染

缺:仅函数内可用，且**不可重用在函数调用后就释放了不可重用**

何时: 只要希望重用一个局部变量，又不希望局部变量被污染时

如何: 3步:

Step1: 将受保护的变量和操作变量的函数封装在一个外层函数中定义。

Step2: 外层函数，要将内层函数队形返回

Step3: 使用者调用外层函数，获得内层函数对象。

正课:

1. \*\*\*\*\*Function

闭包

2. \*\*\*\*\*OOP

1. \*\*\*\*\*Function:

闭包:

如何: 3步:

1. 用外层函数封装受保护的变量和操作变量的内层函数

2. 外层函数返回内层函数的对象

3. 使用者调用外层函数，获得内层函数的对象。

三大特点:

1. 外层函数

2. 受保护的变量

3. 内层函数

笔试:

1. 先找受保护的变量:

1. 外层函数的局部变量

2. 内层函数中被使用

判断当外层函数结束时，受保护的变量最终的值。

2. 找外层函数返回的内层函数对象:

1. return function

2. 直接给全局变量赋值一个function

3. 将多个内层函数放入数组或对象中，批量返回。

结论:

同一次外层函数调用返回的多个内层函数对象共用同一个受保护的变量。

先后两次外层函数调用返回的两个内层函数对象，分别使用各自的受保护的变量副本，互不影响。

优: 重用变量，且保护其不被篡改

缺: 占用更多的内存空间。

无法自动释放

**2.\*\*\*\*\*面向对象:**

什么是: 程序中都是用**对象**来描述现实中一个具体事物。

什么是对象: 程序中封装现实中一个具体事物的属性和功能的程序结果

事物的属性，会成为对象的属性

事物的功能，会成为对象的方法

为什么: 更符合人的想法。

如何使用: 2步:

1. 先用对象描述一个事物的属性和功能

-创建对象/封装

封装: 将事物的属性和功能集中定义在一个对象中。

3种:

1.对象直接量:

var obj={

属性名:值,

方法名:function(){

//对象的方法中，要使用对象自己的属性，必须写成"this.属性名"

//当前对象的...

}

}

this: 指正在调用方法的.前的对象

\*\*\*js中一切对象底层都是hash数组

2.用new:

var obj=new Object();

obj.属性名=值;

obj.方法名=function(){

...this.属性名...

}

其中new可省，()也可省，但不能同时省略。

3.反复创建多个相同结构的对象:

2步:

1.定义构造函数:

什么是: 专门描述一类对象统一结构的函数。

何时: 只要反复创建多个相同结构的对象时。

为什么: 代码重用——优

如何:

function 类型名(属性参数){

this.属性名=属性参数;

this.方法名=function(){

...

}

}

2.用new调用构造函数:

var obj=new 类型名(属性值);

new: 4件事儿:

1. 创建了一个空对象:

2. ?

3. 用空对象调用构造函数

4. 返回新对象地址给变量

缺: 没有解决内存浪费的问题

2. 再按照逻辑的需要，依次调用对象中的功能来操作对象的属性。

访问对象的成员(属性和方法):

操作对象的属性: obj.属性名

**对象的属性名其实就是一个放在对象中的普通变量而已。**

特殊:

访问对象中没有的属性,不报错,返回undefined.

为对象中不存在的属性赋值，不报错，会自动添加属性，保存属性值。

调用对象的方法: obj.方法名();

**对象的方法名**其实就是一个放在对象中的普通函数而已。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月3日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8

回顾:

1.OOP:

创建:

1. 直接量:

var obj={

属性名:值,

方法名:function(){

this.属性名

}

}

2. 用new:

var obj=new Object();

obj.属性名=值;

obj.方法名=funciton(){

this.属性名;

}

3. 用构造函数，反复创建多个相同结构的对象

2步:

1. 定义构造函数:

this.属性名=属性参数;

this.方法名=function(){

this.属性名...

}

}

优: 代码重用

缺: 没有解决内存浪费的问题

2. 用new调用构造函数:

——实例化(instance)一个xxx类型的对象。

var obj=new 类型名(属性值列表)

其中:obj也成为xx类型的一个实例

访问对象成员:

obj.属性名 => obj["属性名"]

obj.方法名();

**本质: js中一切对象底层都是hash数组，属性名是字符串类型的key，属性值是value。**

正课:

1.\*\*\*\*\*OOP:

面向对象的理解:

面向对象三大特点:

**封装**:将一个/一类事物的属性和功能集中定义在一个对象中

为了代码重用！

**继承:**父对象中的成员，子对象无需重复创建，即可直接使用。

为了即代码重用，又节约内存

**多态:**同一个方法,在不同情况下，表现出不同的状态。

为了体现子对象与父对象之间的差异。

继承: js中的继承都是通过原型对象实现的

也成为: 原型继承/继承原型

原型对象(prototype): 集中存储一类对象共有成员的父对象。

何时: 只要多个子对象中，拥有相同的成员，就都要集中定义在父对象(原型对象)中一份，然后所有子对象共用!

如何:

创建: 在创建构造函数时，会自动创建该类型的原型对象prototype

何时继承:

new: 2. 设置新对象的\_\_proto\_\_属性继承构造函数的原型对象

如何向原型对象中添加共有成员:

类型名.prototype.共有成员名=值

原型链: 由各级父对象逐级继承，实现的链式结构。

任何对象都有\_\_proto\_\_继承其父对象

Object.prototype是所有对象的顶级父对象。Object.prototype.\_\_proto\_\_是null

Function.prototype是所有函数对象的父对象。Function.prototype.\_\_proto\_\_是Object.prototype

原型链控制着成员的使用顺序:

优先使用当前对象本地的自有成员

如果自己没有，才延原型链向父级找

如果到Object.prototype没找到，返回undefined。

vs 作用域链: 控制着变量的使用顺序

不带.直接访问的变量，走作用域链

带.访问的对象成员，都走原型链

正确的访问对象的成员:

原则:

修改原型链上的共有成员:必须用父对象改

修改自有成员: 必须用当前子对象本身

错误: 禁止使用子对象修改共有成员

如何判断成员是自有还是共有:

自有: 直接定义在当前对象本地的成员

共有: 对象本地没有，从原型链上继承来的成员。

判断自有:

var bool=obj.hasOwnProperty("成员名")

判断"成员名"是否直接定义在obj本地

如果是自有，就返回true，

否则返回false

判断共有:

不是自有，且原型链上有

!obj.hasOwnProperty("成员名")

&&(obj.成员!==undefined)

或 "成员名" in obj

in: 判断"成员名"是否包含在obj的原型链上。（可检查整个原型链）

删除: delete obj.属性名

不能用子对象删除共有属性

正课:

1.\*\*\*\*\*OOP

扩展内置对象的API

原型API

自定义继承

1.扩展内置对象的API:

为什么: 解决浏览器兼容性问题

何时: 如果想在旧的浏览器上，使用新浏览器的API，就需要再旧浏览器扩展内置对象的API

如何: 都是将自定义的API添加到指定类型的原型对象上。

2.原型API:

获得原型对象:2种:

通过构造函数获得: 构造函数.prototype

通过子对象获得: obj.\_\_proto\_\_ 兼容性

Object.getPrototypeOf(obj)

获得obj对象的父对象(原型对象)

判断指定对象是否是子对象的父对象:

father.isPrototypeOf(child)

判断father对象，是否是child对象的父对象。

如果father是child的父对象，就返回true，否则返回false

强调: 检查整个原型链

判断指定对象是否是指定构造函数的实例:

child instanceof 构造函数(mather)

判断child是不是被构造函数实例化出的

强调: 也检查整个原型链

鄙视题: 判断一个对象是不是数组类型

有几种办法?

typeof: 只能识别原始类型和函数

无法进一步区分不同的引用类型

1.isPrototypeOf

2.instanceof

3.class属性: 记录了对象创建时的类型名

一旦对象创建，不可更改

只有Object.prototype中的toString方法才能输出class属性

问题: 内置对象的原型对象中都重写了新的toString方法，阻碍了子对象直接调用原始的toString方法

解决: 用call强行调用!

要调用方法的完整签名.call(调用方法的对象)

4.Array.isArray(obj)

专门用于检测obj是不是数组类型

原理同第三种方法

总结:

call: 用一个对象强行调用一个本来无法调用到的函数。

何时: 希望打破原型链的限制，直接调用一个本不可调用的函数

如何:

要调的函数.call(对象)

添加一个新方法,如何选择加在构造函数上，或者加在原型对象中?

如果希望指定类型的所有子对象都可访问时，就放在原型对象中

——必须通过实例化的对象调用

也成为实例方法

如果无法确定将来调用函数时的对象的类型，就不能放在原型对象中。要放在构造函数上。——static静态方法

多态:

重写(override): 如果子对象觉得父对象继承来的成员不好用，就可在本地定义和父对象成员同名的自有成员——优先使用自有成员，而屏蔽了父对象的共有成员。

3. 自定义继承:

3种:

1. 修改一个对象继承另一个父对象:

child.\_\_proto\_\_=father;

=> Object.setPrototypeOf(

child,father

)

设置child继承father

2. 批量修改所有子对象的父对象:

时机: 在开始创建子对象之前，就修改原型对象。

构造函数.prototype=father

3. 两种类型间的继承:(最像Java的继承，extends)(inherits&&extends)

问题: 两种类型间，有部分相同的属性结构，和共同的方法。

解决: 抽象出一个公共的父类型

将子类型中相同的部分属性和方法，定义到父类型中。

如何实现两种类型间的继承:

1. 设置子类型的原型对象继承父类型的原型对象。(inherits)

Object.setPrototypeOf(

子类型的原型对象,

父类型的原型对象

);(先有子类型再有父类)

2. 借用父类型的构造函数:

父类型构造函数.call(this,参数,...)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月5日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

回顾:

1. OOP:

面向对象三大特点:

封装:将现实中一个事物的属性和功能集中定义在一个对象中。

3种:

1. 直接量:(创建一个单独的对象)

var obj={

属性名:值,

方法名:function(){

...this.属性名...

}

}

2. 用new:

var obj=new Object();

obj.属性名=值;

obj.方法名=function(){

...this.属性名...

}

3. 反复创建多个相同结构的对象: 2步

1. 定义构造函数:

function 类型名(属性参数列表){

this.属性名=属性参数;

/\*浪费内存，已被否定

this.方法名=function(){

...this.属性名...

}\*/

}

2. 用new调用构造函数: 实例化一个对象。

var obj=new 类型名(属性值列表);

new: 1.创建新的空对象

2.设置新对象的\_\_proto\_\_继承构造函数的原型对象

3.用新对象调用构造函数，为新对象添加规定的属性和方法。

4.将新对象的地址保存到变量中

js中一切对象底层都是hash数组

继承:父对象中的成员，子对象无需重复定义，即可直接使用。

js中的继承都是原型继承

原型对象:集中存储一类子对象共有成员的父级对象。

原型链: 由各级对象的\_\_proto\_\_属性逐级继承，形成的链式结构

控制着对象中属性的使用顺序

API:

1. 判断指定"属性名"是否是obj的自有属性

var bool=obj.hasOwnProperty("属性名")

2. 判断是否是共有属性:

!obj.hasOwnProperty("属性名")

&&("属性名" in obj)

3. 获取原型对象: 2种:

构造函数.prototype

Object.getPrototypeOf(child)

4. 删除属性: delete 对象.属性名

5. 判断一个对象是不是数组类型:

1. father.isPrototypeOf(child)

判断father是否是child的原型链上的父级对象。

2. child instanceof 构造函数

判断child是否是构造函数的实例

3. 输出对象的内部属性class:

Object.prototype.toString.call(obj)

输出obj的class属性

4. Array.isArray(obj)

判断obj是否是数组类型

自定义继承: 3种:

1. 直接修改一个对象继承另一个对象:

Object.setPrototypeOf(child,father);

2. 修改构造函数的原型对象，批量修改之后所有子对象的父对象

构造函数.prototype=father

时机: 在开始创建子对象之前就修改。

3. 两种类型间的继承: inherits&&extends

**抽象: 如果多个子类型之间有相同的属性结构和方法定义，就要抽象出一个父类型，将相同的属性结构和方法定义统计定义在父类型中。**

在子类型构造函数中借用父类型构造函数：extends

function 子类型构造函数(...){

//new 子类型构造()->this->新对象

父类型构造函数.call(this,参数1,参数2,...)

...扩展新属性

}

其实: 父类型构造函数.call(this,参数1,参数2,...)可简写为:

父类型构造函数.apply(this,arguments)

让子类型原型继承父类型原型:inherits

Object.setPrototypeOf(

子类型原型, 父类型原型

)

call vs apply:

相同: 都是强行借用任意函数，并替换函数中的this为指定对象。

不同: 参数:

call,只能将传入借用的函数的参数，单独传入。

apply,可将传入借用的函数的参数，以数组方式传入。apply可自动打散数组为单个参数。

多态:

重写(override):如果子对象觉得父对象继承来的成员不好用，可子定义同名的自有成员，来覆盖父对象的成员。

正课:

**1. \*\*\*\*\*ES5:**

\*\*\*\*\*对象的属性

Object.create

数组API

\*\*\*\*\*函数的bind()

严格模式

1.\*\*\*\*\*对象的属性:

ES认为对象就是属性的集合

属性分为:

命名属性:自定义的或可直接用.访问到的属性

数据属性: 直接存储属性值得属性

将每个属性都定义为四大特性:

value: 实际存储当前属性值

writable: 标识属性是否可修改

enumerable: 标识属性是否可被for in遍历到。即使改为false，依然可用.访问到

configurable: 标识能否删除属性或修改属性的其他特性

为什么: 特性是对属性的保护

configurable保护其他特性

如何访问特性:

Object.getOwnPropertyDescriptor(

obj（对象）,"属性名"

)

返回一个对象，对象中包含了属性的四大特性

如何修改属性的特性:

1. 只修改一个属性的四大特性:

Object.defineProperty(obj,"属性名",{

writable:false,

configurable:false

})(改谁写谁)

强调: 为了防止其他人再将writable改回true，可将configurable一并设置false。

configurable一旦改为false，不可逆。

2. 同时修改多个属性的四大特性:

Object.defineProperties(obj,{

属性名:{

特性名:值,

...

},

属性名:{

特性名:值,

...

}

})

关于特性的默认值:

defineProperty/defineProperties，如果要修改的属性不存在，会自动添加新属性。

但是，被defineXXX添加的新属性，所有特性默认值为false！

\*\*\*\*\*访问器属性: 不直接存储属性值，而是对其他属性提供保护的一种特殊属性。

何时: 如果使用自定义的验证规则保护属性时。

四大特性:

set:function(val){

验证val

如果验证通过，才将val赋值给受保护的数据属性

}//只要对访问器属性赋值时，自动调用访问器属性的set方法

get:function(){

return 受保护的数据属性的值

}//只要使用访问器属性取值时，自动调用访问器属性的get方法

enumerable: 同数据属性

configurable: 同数据属性

如何定义:

都用defineProperty/defineProperties

内部属性:不建议或不允许访问的属性

正课:

1. ES5:

对象的属性

Object.create

数组API

\*\*\*\*\*函数的bind()

严格模式

**1.对象的属性:**

命名属性:

数据属性:

访问器属性:

核心问题: 实际属性值存在哪儿?

解决: 使用闭包，封装一个受保护的局部变量。

内部属性:

比如: \_\_proto\_\_ class

extensible: 标识一个对象是否可扩展新属性。默认值true。

防篡改: 限制对已经创建的对象中的属性进行重新配置。

为什么: 严格的语言中，要求对象一旦创建，属性不允许擅自增减。

何时: 今后，只要不允许对已经创建的对象的属性进行添加和删除操作时。

如何: 3个级别:

1. 防扩展: 禁止向对象中添加新属性

判断一个对象是否可扩展:

Object.isExtensible(obj)

设置一个对象不可扩展:

Object.preventExtensions(obj)

相当于修改obj的extensible为false

2. 密封: 在兼具防扩展的基础上，进一步禁止删除对象中的属性

判断当前对象是否密封:

Object.isSealed(obj);

修改当前对象为密封

Object.seal(obj);

相当于:修改obj的extensible为false

将所有属性的configurable特性都改为false。

3. 冻结: 在兼具密封的基础上，进一步禁止修改所有属性的值

何时: 只要希望一个对象中的属性值一旦创建，不可改变时，就要冻结对象——

判断一个对象是否被冻结:

Object.isFrozen(obj)

冻结一个对象:

Object.freeze(obj)

**2. Object.create()**

设置继承有几种方法:

1.Object.setPrototypeOf(child,father)

2.构造函数.prototype

3.两种类型间的继承:

抽象父类型

借用构造函数

设置子类型原型继承父类型元素

4.创建一个新对象，继承父对象，并扩展自有属性。

如何:

var child=Object.create(father,{

新属性:{

四大特性

},

新属性:{

四大特性

}

})

**3.数组API:**

1. 判断数组中每个元素是否符合要求:

arr.every():判断数组中每个元素是否都符合指定规则的要求。

arr.some():判断数组中是否包含符合指定规则的元素。

如何使用:

arr.every(function(val,idx,arr){

return 判断条件

})

2. 遍历API:

3. 过滤和汇总:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月6日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

正课:

1.\*\*\*\*\*ES5

数组API:

\*\*\*\*\*bind函数

严格模式:

**1.数组API:**

1. 判断数组中的元素是否符合要求

arr.every(): 每个元素是否都符合要求

arr.some(): 是否包含符合要求的元素

如何使用:

arr.every(function(val,idx,arr){

return 判断条件

});

2.**遍历API:**对数组中每个元素执行相同的操作

arr.forEach(): 对原数组中每个元素，直接做相同的操作——直接修改原数组

arr.map(): 依次取出原数组中每个元素，执行相同操作后，放入一个新数组。

——返回新数组

如何使用:

arr.forEach(function(val,idx,arr){

直接修改arr中idx位置的当前元素

});

arr.map(function(val,idx,arr){

return 一个新值

});

其实: map也可实现forEach的功能，用法和forEach一样。

3.过滤和汇总:

arr.filter(): 筛选出原数组中符合条件的元素，组成新数组。

arr.filter(function(val,idx,arr){

return 判断条件

});

汇总: 将数组中每个元素统计起来，得出一个结果。

var r=arr.reduce(

function(prev,val,idx,arr){

return prev+val;

},

startVal

)

其中: prev自动接收截止到目前已经统计的临时汇总结果。

startVal是开始汇总前的初始值

2. bind函数:基于一个现有函数，创建一个新函数，并永久绑定this为指定对象，同时，还可提前绑定部分参数。

何时使用: 在使用回调函数时，只要希望固定回调函数中的this时。

鄙视题: call apply bind

call和apply: 临时借用构造函数，并临时替换函数中的this为指定对象。

立刻执行函数

调用后，this回复原样

call: 参数单独传入

apply: 参数以数组传入

bind: 创建一个函数，并永久绑定this

不会立刻执行新函数

一旦被bind创建的函数，其中的this，无法再被call/apply替换

3. 严格模式:

如何使用:

在当前范围的开头: "use strice";

写在2个地方:

1. 写在一个函数内部的开头:

仅函数内使用严格模式，函数外依然使用普通模式

2. 写在一个script元素或js文件的开头

整个js文件或script代码段都采用严格模式

1. 旧项目升级，先逐个函数启用严格模式

2. 新项目，开始就使用严格模式

规定:

1. 将静默失败升级为了错误！

2. 不允许对未声明的变量赋值！

3. 尽量少的或不使用arguments及其API

4. 新增了eval作用域:

在eval中声明的变量，eval结束后，不能使用。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月7日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DOM

正课:

1.什么是DOM:

2.DOM Tree

3.遍历DOM树:

1.什么是DOM:

原生js=ECMAScript(核心语法)+

DOM(专门操作网页内容的API)+ 3天

BOM(专门操作浏览器窗口的API)+ 2天

DOM: Document Object Model

专门操作网页内容的API

DOM标准: W3C 规定了操作一切结构化文档的API。

核心DOM: 操作一切结构化文档(包括HTML和XML)的通用API

万能，但繁琐

HTML DOM: 专门操作HTML文档的简化版API

只对部分常用的API提供了简化

不是万能

XML DOM: 专门操作XML文档的API

鄙视题: HTML XHTML DHTML XML

HTML:专门编写网页内容的语言

XHTML: 更严格的HTML标准

DHTML: 一切实现动态网页效果的技术统称

DHTML=HTML+CSS+JS

XML: 可扩展的标记语言

语法同HTML,但标签可自定义

专门用于持久保存或在网络中传输结构化数据

<student>

<name>杨幂</name>

<math>81</math>

<chs>56</chs>

</student>

JSON: JavaScript Object Notation

'{"name":杨幂,"math":81,"chs":56}'

DOM可执行的操作:

\*\*\*查找 修改 添加 删除

2. \*\*\*DOM Tree:

什么是DOM树:

网页中一切内容在内存中都是以树形结构组织在一起的。

网页加载过程:

当接收到HTML文档时:

创建一个树根节点:document对象

读取HTML内容时:

每读取一项内容就创建一个节点对象

将节点对象挂到document下

**网页中的一切内容都是一个节点对象：**

节点对象:Node

三大属性: 1.**nodeType2．nodeName3.attribute**

1.**nodeType**:节点类型:

document: 9

element: 1

attribute: 2

text: 3

何时: 辨别获得的节点类型时

**2．nodeName**: 节点名:

document: #document

element: 标签名(全大写)

何时: 辨别元素的具体标签名

**3.attribute:** 属性名

text: #text

nodeValue: 节点值:

document: null

element: null

attribute: 属性值

text: 文本内容

**节点间**关系: 分：**节点树** 和 **元素树**

**节点树**: 包含所有html内容的DOM树

1. 父子关系:

parentNode: 父节点

childNodes: 所有直接子节点

返回一个动态集合-类数组对象

firstChild: 第一个子节点

lastChild: 最后一个子节点

2. 兄弟关系:

nextSibling: 下一个兄弟

previousSibling: 前一个兄弟

**何时使用:**

**已经获得一个节点，要找它周围的相邻的节点。**

问题: 容易受看不见的空文本干扰

解决: 如果**只需要获取元素**，而**不关心文本**，可用**元素树**

**元素树**: 仅包含元素节点的树结构

强调: 不包含一切文本节点

1. 父子关系:

parentElement: 父元素

children: 所有直接子元素

返回一个动态集合-类数组对象

firstElementChild: 第一个子元素

lastElementChild: 最后一个子节点

2. 兄弟关系:

nextElementSibling: 下一个兄弟

previousElementSibling: 前一个兄弟

问题: 浏览器兼容性: IE9+

3.遍历节点:

2步:

1. 先定义函数仅遍历直接子节点

2. 对每个直接子节点应用和父节点相同的操作

算法: 深度优先遍历

childNodes和children:

live collection(动态集合):

不实际存储节点的完整属性

问题:每次访问集合时，都会重新查找DOM树

遍历:

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

造成反复查找DOM树

解决:

for(var i=0,len=childNodes.length;i<len;i++)

问题:递归的效率低

解决:用遍历API+循环

遍历API:

1. 节点迭代器:

按照深度优先的原则，逐个返回每个子节点。

何时: 只要按照深度优先的顺序，遍历所有子代节点时。

如何:2步:

1.用指定父元素创建节点迭代器对象:

var iterator=

document.createNodeIterator(

parent,NodeFilter.SHOW\_ALL,

.SHOW\_ELEMENT

null,false

);

2.反复调用iterator的nextNode方法

返回正在遍历的当前节点对象

直到返回null为止

2.TreeWalker:

比节点迭代器更灵活的API:

基本用法和NodeIterator一样

只不过扩展了:

walker.parentNode()

walker.firstChild();

walker.lastChild();

walker.nextSibling();

walker.previousSibling();

如何: 2步:同NodeIterator

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月8日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

回顾:

1. DOM树:

节点对象: Node

**nodeType**

document: 9

element: 1

attribute: 2

text:3

**nodeName**

document: #document

element: 标签名(全大写)

**attribute: 属性名**

text: #text

nodeValue

document: null

element: null

attribute: 属性值

text: 文本内容

节点间关系:

节点树:

1. 父子关系:

elem.parentNode

elem.childNodes (live collection)

elem.firstChild

elem.lastChild

2. 兄弟关系:

elem.previousSibling

elem.nextSibling

元素树:

1. 父子关系:

elem.parentElement

elem.children (live collection)

elem.firstElementChild

elem.lastElementChild

2. 兄弟关系:

elem.previousElementSibling

elem.nextElementSibling

遍历:

1. function getChildren(parent){

//输出parent

var children=parent.childNodes;//children

for(var i=0,len=children.length;i<len;i++){

getChildren(children[i]);

}

}

2. 遍历API:

NodeIterator:

var iterator=document.createNodeIterator(

parent,NodeFilter.SHOW\_ALL, null, false

.SHOW\_ELEMENT

);

var curr=iterator.nextNode();//获得下一个节点

while((curr=iterator.nextNode())!=null){

//curr就是当前节点

}

TreeWalker:

var walker=document.createTreeWalker(

parent,NodeFilter.SHOW\_ALL, null, false

.SHOW\_ELEMENT

);

基本用法同iterator，外还扩展了:

walker.parentNode();

walker.firstChild();

walker.lastChild();

walker.nextSibling();

walker.previousSibling();

正课:

**1. \*\*\*查找:**

**1.** 利用节点间关系查找周围的元素

**2.** 用HTML属性查找:

按id:var elem=document.getElementById("id");

只能找到一个元素

如果没找到，返回null

强调: 必须用document调用。

按标签名:

var elems=

parent.getElementsByTabName("标签名");

在指定父元素parent下查找指定标签名的元素

返回live collection

强调: 查找所有子代节点

按name:

var elems=

parent.getElementsByName("name");

在指定父元素parent下查找指定name属性的元素

返回live collection

按class:

var elems=

parent.getElementsByClassName("class");

在指定父元素parent下查找指定class属性的元素

返回live collection

如果getElementsXXX没找到，返回空集合[]

**3. 用选择器查找:**

1. 只查找一个符合选择器要求的元素:

var elem=parent.querySelector("selector");

如果没找到，返回null

2. 查找多个符合选择器要求的元素:

var elem=parent.querySelectorAll("selector");

如果没找到，返回空集合[]

强调: 返回no-live collection:

**包含**完整属性的集合，反复访问不会导致重复查找DOM树

鄙视题: getXXX vs querySelector

1. 执行效率:

如果通过一个属性即可找到的元素

getXXX更快返回live collection

querySelector 更慢no-live collection

2. 代码量:

如果需要通过多次查找才能获得想要的元素时

querySelector 更简单

getXXX 繁琐

4. VIP通道:

document.body

document.head

document.documentElement : html

回顾:

1. 查找:

1. 节点间关系:

节点树:

1. 父子关系:

parentNode

childNodes

firstChild

lastChild

2. 兄弟关系:

previousSibling

nextSibling

元素树:

1. 父子关系:

parentElement

children

firstElementChild

lastElementChild

2. 兄弟关系:

previousElementSibling

nextElementSibling

2. 按html属性找:

document.getElementById()

parent.getElementsByTagName();

parent.getElementsByName();

parent.getElementsByClassName();

3. 按选择器查找:

parent.querySelector();

parent.querySelectorAll();

4. VIP通道:

document.body

document.head

document.documentElement - html

正课:

**1. 修改:**

内容:

属性: 标准、自定义

样式: 内联样式、内/外部样式表

1. 修改内容:

elem.innerHTML: 元素开始标签到结束标签之间的html内容 （包括元素标签）

例子：**elem.innerHTML=”XXXXX”; 先找到元素，然后修改内容**

**P.innerHTML=”<a>年化啊发生的书法家</a>”**

elem.textContent: 元素开始标签到结束标签之间的文本内容 （不包括元素标签）

去掉html标签

IE8.innerText

练习: 技巧: 如果遍历过程中会删除数组的元素，则要从后向前反向遍历。

2. 访问或操作**元素的属性**:

标准属性:

核心DOM: 万能

1. 访问属性节点对象:

var node=elem.attributes[i/"属性名"];

var node=elem.getAttributeNode("属性名");

node.value->获得属性值

**2. 获得/获取属性值**:

var value=elem.getAttribute("属性名");

**3. 修改属性值:**

elem.setAttribute("属性名","值");

**4. 移除/删除属性:**

elem.removeAttribute("属性名");

**5. 判断该元素是否包含指定属性:**

var bool=elem.hasAttribute("属性名");

**HTML DOM:将html标准属性封装到了元素对象中**

**elem.属性名**

问题: 仅能访问HTML标准属性，无法修改html标准属性

自定义属性:

核心DOM: 万能

HTML5 规定:

1, 定义自定义属性时: data-属性名="值" 例子：<p data-tang=”ding”></p>

2, 读取自定义属性: elem.dataset.属性名 例子：p.dataset.tang

在jquery中 获取自定义属性值：$(“p”).data(“tang”) //获取到属性值ding

property vsattribute

显式写在开始标签中的: attribute

强调: getAttribute只能获得在页面上显式定义的属性值

在内存中对象里保存的: property

标准属性: 既是property也是attribute

自定义属性: 只是attribute

input type="checkbox" 要获得选中状态checked

不能用getAttribute，一定要用elem.checked

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月9日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

回顾:

1. \*\*\*查找:

1. 节点间关系:

节点树:

1. 父子关系:

parentNode

childNodes

firstChild

lastChild

2. 兄弟关系

previousSibling

nextSibling

元素树:

1. 父子关系:

parentElement

children

firstElementChild

lastElementChild

2. 兄弟关系

previousElementSibling

nextElementSibling

2. 按HTML查找:

按id: document.getElementById();

按标签名: parent.getElementsByTagName();

按name: parent.getElementsByName();

按class: parent.getElementsByClassName();

3. 按选择器:

找一个: parent.querySelector()

找多个: parent.querySelectorAll()

4. VIP:

document.body

document.head

document.documentElement

2. 访问属性:

标准属性:

核心DOM:

1. 获得属性值: getAttribute("属性名")

2. 设置属性值: setAttribute("属性名",值)

3. 移除属性: removeAttribute("属性名")

4. 判断是否包含: hasAttribute("属性名")

HTML DOM:

elem.属性名

缺点：html Dom只能访问标签中的标准属性，不能修改属性

自定义属性:

核心DOM:

HTML5: 1. 定义时: data-属性名="值"

2. 访问时: elem.dataset.属性名

propertyvsattribute

正课:

1. **修改:**

样式: 内联样式, 内/外部样式表

2. 增加和删除:

1.**Dom 修改样式:**

访问内联样式:

**修改样式:** elem.style.css属性名

例子：P．style.display=”none”；

**获取样式**: elem.style.CSS属性名 p.sytyle.displsay

问题: 仅能访问内联样式

解决: var style=getComputedStyle(elem)

style.css属性名 ——只读

其中,css属性名都要去横线，变驼峰:

比如: background-color:backgroundColor

list-style-type: listStyleType

问题: 一条语句只能修改一个样式

解决: 将一套样式，放入class

用程序操作class

课堂练习:

获得焦点onfocus: 元素从不可操作到可操作的状态变化

失去焦点onblur: 元素从可操作变回不可操作的变化

修改内/外部样式表中的属性:

1. 获得样式表对象: var sheet=document.styleSheets[i]

2. 获得cssRule对象: var rule=sheet.cssRules[i];

特殊: rule是一个keyframes:

var subRule=rule.cssRules[i]

3. 修改cssRule中的属性: rule.style.css属性名=值

Chrome: 不允许sheet对象访问本地的css样式表文件

正课:

1. 添加和删除:

2. \*\*\*HTML DOM常用对象:

ImageSelect Table Form

Option

1. 添加和删除:

添加: 3步:

1. 创建空元素对象:

var a=document.createElement("a");

相当于: <a></a>

2. 设置元素的关键属性:

a.href="http://tmooc.cn"

a.innerHTML="go to tmooc";

3. 将元素挂到指定父元素下:

末尾追加: parent.appendChild(a);

插入: parent.insertBefore(a,old);

替换: parent.replaceChild(a,old);

\*\*\*优化: 尽量少的操作DOM树:

html->DOM Tree

↓

Render Tree->layout->paint

↑

css->cssRules

如何: 1. 如果同时添加父元素和子元素，要在内存中先将所以子元素添加到父元素下，将父元素一次性加入DOM树

2. 如果同时添加多个平级子元素，要先将平级子元素加入文档片段，再将文档片段整体加到页面。

练习: select元素: onchange事件: 选中项发生改变时触发

selectedIndex属性: 当前选中项的下标

文档片段: 内存中临时存储多个平级子元素的虚拟父元素

何时: 添加多个平级元素时，都要先将子元素追加到文档片段中缓存，再将文档片段一次性添加到页面的父元素下

如何: 3步:

1. 创建文档片段：

var frag=document.createDocumentFragment();

2. 将平级子元素，先追加到frag下: 用法同普通父元素

3. 将文档片段，整体添加到页面

删除: parent.removeChild(child);

child.parentNode.removeChild(child);

2. HTML DOM常用对象:

Image: <img />

创建: var img=new Image();

Select: <select>

属性:

selectedIndex:当前选中项的下标

options: 获得select下所有option元素的集合

Option:<option>

创建: var opt=new Option(text,value);

相当于: var opt=

document.createElement("option");

opt.innerHTML=text;

opt.value=value;给对象添加属性

属性: index、text、value、selected

value: 如果选中项有value，则select的value等于选中项的value

如果选中项没有value，则select的value等于选中项的内容

方法:

add(option): 向select下追加一个option对象

appendChild(option)

remove(i): 移除select下i位置的option

事件: onchange: 当选中项发生改变时触发

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月12日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

回顾:

1. 添加和删除:

添加: 3步:

1. var a=document.createElement("a")

**添加属性：a.name=”dingding”; 获取属性：a.getAttributed(“name”) 修改属性：a.setAttributed(“name”,”tang”)**

2. 设置关键属性

3. 将元素添加到DOM树:

parent.appendChild(a);

parent.insertBefore(a,old);

parent.replaceChild(a,old);

优化: 尽量少的操作DOM树，避免反复layout

1. 如果同时添加父元素和子元素，应在内存中先将子元素拼到父元素下，再将父元素一次性添加到DOM树

2. 如果同时添加多个平级子元素，应先将所有子元素放入文档片段，再将文档片段一次性添加到DOM树

文档片段:

1. var frag=document.createDocumentFragment();

2. frag.appendChild(child);

3. parent.appendChild(frag);

删除: parent.removeChild(child);

child.parentNode.removeChild(child);

2. HTML DOM常用对象:

Image: var img=new Image();

Select:

属性: selectedIndex, options, value

方法: add(opt), remove(i)

事件: onchange

Option:

创建: var opt=new Option(text,value)

正课:

1. HTML DOM常用对象:

Table

tHead:

var thead=table.createThead();

table.deleteThead();

rows:

var tr=thead.insertRow(i);

//省略i，表示末尾追加

thead.deleteRow(i);

//i不能省略

cells:

var td=tr.insertCell(i);

//省略i，表示末尾追加

tr.deleteCell(i)

tBodies

tBody:

var tbody=table.createTBody();

//没有table.deleteTBody!

tFoot: 同tHead

Table.rows:

table.insertRow(i);

table.deleteRow(i);

row.rowIndex: 标识row在整个表中的下标位置

table.deleteRow(row.rowIndex)

强调: 所有creatxxx/insertxxx方法都会自动将新元素添加到父元素下，不需要appendChild

练习: 确认框: confirm 两个按钮

点确认，就返回true，否则返回false

Form

获取: var form=document.forms[i/id/name];

属性:

length: 表单中，表单元素的个数

elements[i/id/name]

获得表单中的元素:

form.name => form.elements["name"]

方法:submit(); 手动提交表单

事件:

每个表单元素都有两个方法:

elem.focus(); //让elem获得焦点

elem.blur(); //让elem失去焦点

BOM

正课：

1. BOM:

打开和关闭窗口

窗口大小和窗口定位

\*\*\*\*\*定时器

1. 什么是BOM: Browser Object Model

**BOM是专门操作浏览器的API**

window: 两个角色

1. 代替global，充当全局对象

2. 封装操作浏览器窗口的API

history: 保存当前窗口打开后成功访问过的url的历史记录栈

navigator: 保存浏览器配置信息

location: 保存当前窗口正在打开的url的对象

screen: 保存当前显示器或桌面分辨率信息

event: 在事件发生时，自动保存事件相关信息的对象

2. 打开和关闭窗口:

window.open() 打开，window.close()关闭当前窗口

打开超链接的方式总结:

1. 在当前窗口打开新链接，可后退

html: <a href="url"></a>

js: open("url","\_self")

2. 在当前窗口打开新链接，不可后退

js: location.replace("url");

3. 在新窗口打开新链接，可打开多个

html: <a href="url" target="\_blank"></a>

js: open("url")

4. 在新窗口打开新链接，只能打开一个窗口

内存中，每个窗口都有一个唯一的名字name属性

name属性是在内存中唯一标识一个窗口的名称

浏览器规定: 相同name的窗口的只能打开一个

后打开的会刷新先打开的。

在打开新窗口时，可自定义name属性

预定义name属性值: \_self\_blank ...

html: <a href="url" target="name"></a>

js: open("url","name")

3. 窗口大小和窗口定位(了解):

窗口大小:

获取窗口大小:

1. 获取完整窗口大小: outerWidth/outerHeight

2. 获得文档显示区大小: innerWidth/innerHeight

修改窗口大小:

1. 在打开窗口时，就可定义新窗口的大小和位置:

2步:

1.定义配置字符串:

var config="top=?,left=?,width=?,height=?"

//只是文档显示区的大小

2. 打开窗口时，传入配置字符串

open("url","name",config);

2. 通过API调整窗口大小:

resizeTo(width,height)

width/height定义的是完整大小

resizeBy(width的增量,height的增量)

强调:

1. 只能修改在打开时，就修改了位置和大小的弹出窗口

2. 无法修改当前窗口自己的大小

通过一个窗口修改另一个窗口的大小

窗口定位:

获取窗口左上角定位:

window.screenLeft||window.screenX

window.screenTop||window.screenY

修改窗口的位置:

moveTo(x,y);

moveBy(x的增量,y的增量)

练习:

1. 屏幕的大小:

完整大小: screen.heightscreen.width

去掉任务栏之后的剩余大小:

screen.availHeightscreen.availWidth

2. 事件发生时，鼠标的位置:

e.screenX; 相对于屏幕左上角的x坐标

e.screenY; 相对于屏幕左上角的y坐标

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月13日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

回顾:

1. \*\*\*HTML DOM常用对象:

Table

创建: createTHead() createTBody()createTFoot

删除: deleteTHead() deleteTFoot()

tHead

insertRow(i) deleteRow(i)//i相对于当前行分组内的位置

rows:

tr

insertCell(i)deleteCell(i)

cells: td

tBodies

tBody

tFoot

insertRow(i) deleteRow(i)//i相对于整个表中的位置

rows:

tr.rowIndex: tr相对于整个表的位置

Form:

获得Form: document.forms[i/id/name]

form:

属性: length

方法: submit();

获得表单元素: form.elements[i/id/name]

form.name

elem:

方法: elem.focus() elem.blur()

2. BOM

打开和关闭窗口:

打开: var pop=open("url","name",config)

其中: config="top=?,left=?,width=?,height=?"

//文档显示区大小

关闭: close()

打开连接四种方式:

1. 在当前窗口打开，可后退:open("url","\_self")

2. 在当前窗口打开，不可后退: location.replace("url");

3. 在新窗口打开，可重复打开: open("url")

4. 在新窗口打开，只能打开一个: open("url","name")

窗口大小与窗口定位

大小: outerWidth/outerHeight

innerWidth/innerHeight

修改: resizeTo(width,height)//完整大小

resizeBy(width的增量,height的增量)

定位: window.screenLeft||window.screenX

window.screenTop||window.screenY

修改: moveTo(left,top)

moveBy(left的增量,top的增量)

正课:

1. \*\*\*\*\*定时器

什么是: 让程序按照指定的时间间隔自动执行任务

何时: 动画

周期性定时器: 让程序按照指定的时间间隔反复自动执行同一任务。

何时: 只要反复自动执行一项任务

如何: 3件事:

1. 任务函数: 定义每次要执行的任务

2. 启动定时器:timer=setInterval(任务函数,interval)

让引擎每隔interval ms自动执行一次任务函数

其中: timer: 是当前定时器任务的一个序号

3. 停止定时器: clearInterval(timer)

如果需要停止定时器，都要在启动时就提前保存序号

如何自动停止定时器:

在任务函数的结尾，判断一个临界值，如果达到临界值，就停止定时器.

一次性定时器: 让程序先等待一段时间，再自动执行一次任务。

执行一次后，自动停止。

何时: 只要先等待一段时间，再执行一次任务

用法: 同周期性定时器

1. 任务函数:

2. 启动: timer=setTimeout(task,wait);

3. 停止: clearTimeout(timer);

不是停止执行，而是停止等待

正课:

1. **BOM常用对象**: history, location, navigator

2. \*\*\*event

1. BOM常用对象:

**history:保存当前窗口打开后成功访问过的url历史记录栈**

history.go(n): 前进n步

前进一步: history.go(1)

后退一步: history.go(-1)

刷新: history.go(0)

location: 封装当前窗口正在打开的url的对象

属性:

href: 完整url

protocol: 协议

host: 主机名+端口

hostname: 主机名

port: 端口

pathname:路径

hash: #锚点名

search: ?表单变量

方法:

在当前窗口打开新链接:

location.href="url";

location="url";

location.assign("url");

在当前窗口打开，不可后退:

location.replace("url");

重新加载当前页面:

location.reload(force)

其中: force: 表示是否强制从服务器硬盘获取文件。

true/false

navigator: 封装浏览器配置信息的对象

cookieEnabled: 判断浏览器是否启用cookie

**cookie:** 客户端本地，持久存储用户私密数据的文件

plugins: 包含了所有插件的信息

userAgent: 包含浏览器名称和版本号的字符串

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月14日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

回顾:

1. 定时器:

1. 任务函数:

周期性: 任务函数结尾可能需要判断临界值，自动停止

2. 启动定时器:

周期性: timer=setInterval(任务函数,interval);

一次性: timer=setTimeout(任务函数,wait);

3. 停止定时器:

周期性: clearInterval(timer);timer=null

一次性: clearTimeout(timer); timer=null

动画:

DISTANCE DURATION STEPS

step interval

timermoved

start(): setInterval(moveStep.bind(this),interval);

moveStep(): 将主角移动一个step

在moveStep结尾，判断临界值，自动停止

2. BOM常用对象:

history.go(n)

location:

属性:

href,protocol, host, hostname,port,pathname, hash, search

方法:

1. 在当前窗口打开新连接，可后退:

location="url";

location.href="url";

location.assign("url");

2. 在当前窗口打开,不可后退:

location.replace("url");

3. 重新加载当前页面:

location.reload(false/true)

navigator:

cookieEnabled

plugins

userAgent

正课:

1**. \*\*\*event:**

1. 什么是事件: 用户手动触发的或浏览器自动触发的页面或元素状态的改变。

所有事件: https://developer.mozilla.org/en-US/docs/Web/Events

当事件发生时，浏览器会自动调用事件处理函数:

事件处理函数: 当事件发生时，自动调用的函数

on事件名 ——其实就是一个元素的特殊属性

特殊: 需要绑定一个函数

如何绑定事件处理函数: 3种:

1. 在元素的开始标签中绑定:

<button onclick="js语句(this)"></button>

button.onclick=function(){//this->button

eval("js语句(this)")

}

当单击按钮时:button.onclick()

问题: 不符合内容与行为分离的原则，不便于维护

解决:

2. 在js中为元素的事件处理函数属性赋值:

btn.onclick=function(){//this->button

...

}

问题: 一个事件处理函数属性，只能绑定一个函数对象

解决:

3. 使用API添加事件监听:

btn.addEventListener("事件名",函数对象)

btn.removeEventListener("事件名",函数对象)

优: 1. 同一事件处理函数属性同时可绑定多个函数对象

2. 可移除绑定的函数对象

3. 可改变事件触发的顺序

事件周期:

DOM: 3个阶段:

1. 捕获: 从最外层元素，向内层元素，逐个记录绑定的事件处理函数。默认,暂不触发任何事件。

2. 目标触发:优先触发目标元素上绑定的事件处理函数

目标元素: 实际点击的元素

3. 冒泡: 按照捕获的顺序,由内层，向外层逐层触发捕获的事件处理函数

修改事件触发的顺序:

btn.addEventListener("事件名",函数对象,capture)

其中:capture表示是否在捕获阶段就提前触发

默认false, 只在冒跑阶段才触发

改为true, 在捕获阶段提前触发

事件对象: event

事件发生时，自动创建的，封装事件信息的对象

还提供了对事件进行操作的API

获得event对象:

DOM: 事件发生时，会自动创建event对象

会将event对象作为事件处理函数的第一个参数自动传入。

阻止冒泡: e.stopPropagation();

**利用冒泡:**

优化: 尽量少的创建事件监听对象

事件监听对象容易形成闭包，很难释放

解决: 如果多个平级子元素需要绑定相同的事件处理函数时，只要在父元素上绑定一次即可。所有子元素共用

核心问题:

1. 如何获得目标元素: e.target

2. 屏蔽不需要的元素的干扰

取消事件(阻止默认行为): 当事件执行过程中，遇到问题，可取消事件。不再触发

如何: e.preventDefault();阻止默认行为

练习: form对象: 事件: onsubmit当提交前触发

回顾:

1. \*\*\*event

elem.addEventListener("事件名",函数,capture);

elem.removeEventListener("事件名",函数);

e.stopPropagation(); //取消冒泡

e.target //获得目标元素

e.preventDefault(); //阻止默认行为

正课

1. \*\*\*event

**事件坐标:**

参照屏幕左上角: e.screenX, e.screenY

参照文档显示区左上角: e.clientX||e.x, e.clientY||e.y

参照所在父元素的左上角: e.offsetX , e.offsetY

页面滚动事件:

onscroll:

获得页面滚动过的距离:

document.body.scrollTop|| document.documentElement.scrollTop

2. cookie:

创建: document.cookie=

"变量名=值;expires="+date.toGMTString();

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月20日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

jQuery

正课:

1. 什么是jQuery:

2. 如何使用jQuery:

3. 选择器:

1. 什么是jQuery: 快速的，简洁的JavaScript库

为什么:

1. DOM操作的终极简化:

核心DOM: 万能，API繁琐

HTML DOM: 简洁，不是万能

jQuery: 终极简化:

1. 增删改（内容,属性,样式）查

2. 事件处理:简化和兼容性

3. 动画操作:

4. AJAX操作:

2. 屏蔽浏览器兼容性问题:

何时:

总结: 终极简化——write less,do more

2. 如何使用:

1. 版本:

1.11.x:兼容IE8

jquery-1.11.3.js 未压缩版本-可读性好，便于学习

但文件大，不便于网络传输

jquery-1.11.3.min.js 压缩版本-文件小，便于网络传输，但是可读性差，不便于学习——开发用

2.xx.x:不在兼容IE8

2. 手册:

3. 编程步骤: 2步:

1. 将jquery-1.11.3.js文件保存在当前项目的文件夹下

2. 在html页面的body元素底部引入jquery-1.11.3.js文件

强调: 在自定义脚本之前引入

4. jQuery对象:

什么是: 封装找到的DOM对象

并提供操作DOM对象的简化版的API

的类数组对象

鄙视题: DOM对象 vs jQuery对象的

DOM对象: 由W3C标准规定的，浏览器厂商已经实现的，使用DOM API的标准对象

优: 万能 缺: 繁琐

jQuery对象: 由jQuery.js函数库定义的第三对象/API

DOM对象的API和jQuery对象的API不通用！

今后，只要使用jquery，都要先获得jquery对象:

如何获得jquery对象: 2种

1. 先获得DOM对象，再用$函数封装到jQuery对象

var $jQuery=$(DOM对象);

2. 直接使用$函数查找到的DOM对象，被自动封装到一个jQuery对象中。

var $jQuery=$("selector");

强调: 只有放入jQuery对象中的DOM对象，才能使用简化版API。

特殊: 万一jQuery对象不支持某项功能:

从jQuery对象中取出DOM对象，使用原生DOM API

$jQuery[i] -> $jQuery.get(i) 获得i位置的DOM对象

3. 选择器:jQuery支持几乎所有CSS选择器

提前:

绑定: DOM: elem.addEventListener("事件名",fn);

jq: $("selector").事件名(fn);

修改样式: DOM: elem.style.css属性名=值

jq: $("selector").css("css属性名",值)

回顾:

1. 基本选择器:

#id .class 元素\*群组

2. 层次选择器:

选择器1 选择器2

parent>child

前一个兄弟+后一个兄弟

前一个兄弟~之后所有兄弟

新选择器:

3. 过滤选择器: 通过特定的过滤规则选出所需的DOM元素

包括: 基本过滤(新), 内容过滤, 可见性过滤, 属性过滤

子元素过滤， 表单过滤

基本过滤: 也称为位置过滤:

强调: 位置: 指DOM元素在jQuery元素中的下标位置

包含: :first, :last,

:not(selector),

:even, :odd,

:eq(i)

:gt(i) :lt(i)

何时: 只要选择查询结果中指定位置的元素时

vs first-child/last-child/nth-child(n)

何时: 只要选择指定父元素下的第几个子元素时

内容过滤: （了解）

:contains(text) 比如: span:contains("购物车")

:empty比如: span:empty 选择内容为空的span

:parent比如: span:parent 选择包含子内容或元素的span ——是:empty的反义

:has(selector)比如: div:has(.active) 选择包含class为active的子元素的父元素div

可见性过滤:（了解）

:visible匹配所有可见的元素

:hidden 所有不可见元素

4种: css: display:none; visibility:hidden; opacity:0

html: type="hidden"

强调: :hidden只能匹配 display:none; 和type="hidden"

回顾: 属性过滤——其实就是css中的属性选择器

[属性名] 匹配包含指定属性的元素

[属性名=值] 匹配指定属性的值为指定值得元素

[属性名!=值] 匹配指定属性的值不是指定值得元素

[属性名^=值] 匹配指定属性的值以指定内容开头的元素

[属性名$=值] 匹配指定属性的值以指定内容结尾的元素

[属性名\*=值] 匹配指定属性的值包含指定内容的元素

[属性选择器1][属性选择器2]... 匹配同时满足属性选择器1，,2的元素

回顾: 子元素过滤:

包括: :first-child :last-child :nth-child(n) :only-child

选择在父元素下作为指定为位置的子元素

位置相对于父元素而言。

vs 位置过滤: 选择查询结果集合中指定下标位置的元素

和元素在其父元素下的位置无关

表单选择器: 利用表单中元素标签和type属性选择表单中的元素

何时:今后，只要选择表单中的元素，都要用表单元素选择器:

为什么: 代替DOM中的form.name

包括:

:input选择所有inputtextareaselectbutton

vs input: 仅选择input元素

:text选择type为text的元素

:password

:radio

:checkbox

:submit

:reset

:button

:file

:hidden

不仅选择type="hidden"，还选择display="none"

:image 选取所有图形按钮

表单属性过滤选择器:

:enabled 选择所有可用的元素

:disabled 选择所有不可用的元素

:checked 选择被选中的checkbox

:selected 选择被选中的option

课堂练习:

修改标准属性: $("selector").attr("标准属性名","值")

等效: getAttribute和setAttribute

所有这类方法，只要省掉第二个参数，就是读取属性值

\*\*\*问题: attr只能获取或设置元素开始标签中的attribute

无法获取或设置不在开始标签中的property

解决: prop方法专门读取内存中对象的属性

比如: checked属性只能用prop读取

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月21日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

正课:

1. jQuery中基本操作: 操作元素

1. 操作元素:

属性(attr prop)

attr: 在开始标签中显式定义的属性

读: $("selector").attr("属性名")=>getAttribute("属性名")

改: $("selector").attr("属性名",值)

=>setAttribute("属性名",值);

问题: 无法访问不在开始标签中的内存中的prop属性

比如: checked selected disabled

解决:

prop: 读取不在开始标签中定义的内存中的元素属性prop

两用:

读: $("selector").prop("属性名")=>elem.属性名

改: $("selector").prop("属性名",值)

移除属性(了解):$(...).removeAttr("属性名")

=>removeAttribute("属性名");

总结: jquery中很多方法都是两用:

如果不提供新值，就读取；如果提供新值，就修改

内容:

html原文:

读: $(...).html()elem.innerHTML

改: $(...).html('html片段')elem.innerHTML='html片段'

简便: 清空元素内容:

$(...).empty(); elem.innerHTML="";

纯文本

读: $(...).text() elem.textContent/innerText

改: $(...).text('文本')

表单元素的value

读: $(...).val()

改: $(...).val(值)

样式:

1. 直接操作CSS属性:

$(...).css("css属性名") => getComputedStyle()

$(...).css("css属性名",值) => elem.style.css属性名=值

总结: css()能读所有属性，但只能改内联样式中的属性

**简化: 同时修改多个属性值**

**$(...).css({属性名1:值1, 属性名2:值2, ...})**

强调: 属性名都要去横线变驼峰

2. 修改class属性:

完整修改class属性: $(...).attr("class","类名")

追加class属性: $(...).addClass("类名")

class="cell" : $(...).addClass("n8")=> class="cell n8"

移除class属性: $(...).removeClass("类名")

class="cell n8": $(...).removeClass("n8")=>class="cell"

清除class: $(...).attr("class","")或$(...).removeClass();

判断是否包含指定class: $(...).hasClass("类名")

技巧: 因为class有时容易发生变化

所以，如果class有可能发生变化时，尽量不要用class查找。

解决: 可使用自定义属性来保存固定值作为查找条件

练习: 补: 获得父元素: DOM: elem.parentNode

elem.parentElement

jq: $(...).parent()

获得兄弟元素: DOM: elem.nextElementSibling

elem.previousElementSibling

jq: $(...).next();$(...).prev();

过渡: transition

3步: 1. 定义开始样式: 同时定义transition属性

2. 定义结束样式:

3. 用addClass将结束样式追加到class属性中

正课:

1. 修改:

样式：

2. 遍历:

3. 插入,删除,替换

4. 事件:

1. 修改:

样式: 1. 修改css属性: $(...).css()

2. 修改class:

$(...).addClass()$(...).removeClass() $(...).hasClass()

$(...).toggleClass("类名"): 在有或没有指定类名之间切换。

2. 遍历节点: 节点间关系

1. 父子关系:

$(...).parent()

$(...).children([selector]) 只获得直接子元素

$(...).find(selector) 获得所有子代元素

强调: children可不加selector，但find必须加selector

2. 兄弟关系:

$(...).next([selector])下一个兄弟元素

$(...).prev([selector])前一个兄弟

强调: 如果加selector，必须满足:

1. 必须是相邻，2. 必须满足selector的要求

$(...).siblings([selector]) 其它兄弟

3. 添加，删除，替换:

添加:

DOM: 3步:

1. 创建空元素对象，

2. 设置关键属性,

3. 添加到父元素下

jq: 2步:

1. 创建节点: var $elem=$("完整html元素代码段");

2. 将节点添加到指定父元素下:

追加: $(parent).append($elem); 追加到parent下的所有子节点末尾——appendChild

插入: $(parent).prepend($elem); 作为parent下的第一个子节点插入

$(child).after($elem); 插入到child之后

$(child).before($elem); 插入到child之前

删除: $(要删除的元素).remove();

强调: 删除时，不必查找父元素

替换和复制(clone):

替换:$(old).replaceWith(新元素)

$(新元素).replaceAll(old)

复制: var $clone=$(...).clone()

浅克隆: 仅复制普通属性，不复制事件处理函数

$(...).clone(true)

深克隆: 不但复制普通属性，而且还复制事件处理函数

总结: jQuery是对DOM操作的终极简化函数库

学jQuery，其实还是在学DOM

查找——选择器

修改:

属性: attr() prop()

内容: html() text() val() empty()

样式: css()

addClass() removeClass()hasClass() toggleClass()

添加: 2步:

1. 创建: var $elem=$(html代码段)

2. 追加: $(parent).append($elem); ...

删除: remove();

克隆: clone([true])

以上是jQuery的核心和本质

以下是jQuery的扩展部分: 1. 事件绑定，2.动画, 3.AJAX封装

4. 事件处理:

鄙视题: jQuery有几种绑定事件处理函数的方法，有什么区别:

1. $(...).bind("事件名",fn)->addEventListener

特点: 为同一个元素的事件处理函数绑定多个函数对象

$(...).unbind("事件名",fn)->removeEventListener

强调: 如果有可能移除事件处理函数，则绑定时，必须用有名的函数绑定，不能用匿名函数。

重载1:不带任何参数: 移除元素上所有事件处理函数绑定

重载2: 只带一个事件名称参数: 移除元素上指定事件名称绑定的所有函数对象

重载3: 带两个参数: 仅移除事件元素上指定事件名称绑定的执行的一个函数对象。

问题: 只能为页面上现有的元素绑定事件

动态新添加的元素无法自动绑定事件处理函数

解决: delegate

**2. 事件代理: 让指定父元素下，所有符合要求的子元素，都能使用事件处理函数。（包括已有的元素和新生成的元素）**

原理: **利用冒泡:**

1. 仅将事件处理函数绑定在父元素上一次

2. 获得目标元素: e.target

3. 仅响应符合条件的元素的事件

优: 减少事件监听的个数

可让后生成的新元素自动响应事件

何时: 只要希望一个父元素下指定的所有子元素都能响应事件时

如何使用:

$(parent).delegate("selector","事件名",fn)

将事件绑定到parent上（利用冒泡）

只有符合selector条件的元素才能响应事件

其中: fn: function(e){e.target//获得目标元素}

$(parent).undelegate("selector","事件名",fn)

delegate vs bind:

1. 优化: 创建事件监听对象的个数

delegate: 仅在父元素创建一个事件监听，所有子元素公用

bind: 为每个子元素分别创建事件监听

2. 效果:

delegate: 新生成的子元素也可自动共享父元素的事件

bind: 新生成的子元素无法自动获得事件处理函数

3. 一次性事件: $(...).one("事件名",fn)

只能执行一次事件处理函数，执行后，自动解除绑定

4. $(...).live("事件名",fn), 本质:同delegate

问题: 极端，将所有元素的所有事件集中绑定到document

$(...).die("事件名",fn), 本质:同undelegate

5. 终极简化:

问题: 仅为一个元素绑定事件处理函数: bind

利用冒泡: delegate

解决: 统一成了on

1. 仅为一个元素绑定事件处理函数:

$("target").on("事件名",fn)

例子：$(“Li”).on(“click”,functuion(){}) 给每个li绑定单机事件

2. 利用冒泡:

$("parent").on("事件名","selector",fn)

例子：$(“Ul”).on(“click”,“li”，function(){………});给Ul下每个li绑定单机事件，包括后生成的元素

解除绑定: off(...)

$(“ul”).off(“cilick”,“li”，fun)

6. 简写: $(...).事件名(function(){...})

问题: 仅少量常用的事件，可简写

如果遇到无法简写的事件，就要用on

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月22日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

正课:

1. 事件

2. 动画

3. jQuery UI

1. 事件:

模式触发事件:

DOM: elem.onxxx();

问题: 只能触发直接用onxxx绑定的事件处理函数

用addEventListener添加的事件监听，无法模拟触发

jq: $(...).trigger("事件名")

其实可简写为: $(...).事件名

页面加载后执行:

特殊: $(document).ready(function(){...})

鄙视题: vs DOM: window.onload=function(){...}

ready的底层事件: DOM中的ondocumentcontentloaded

只要DOM树加载完成，js执行完毕就会触发。

而onload: 必须等到html,css,js,img全部加载完成才触发

优化: 可将原来在load之后才执行的任务(事件绑定，修改DOM树)，提前到ready之后执行，可缩短页面整体加载时间。

如何使用: $(document).ready(function(){...})

简: $().ready(function(){...})

更简: $(function(){...});

其实: 只要将script放在body的结尾，就可实现DOM加载后立刻执行。

一个事件: hover: 其实就是mouseover和mouseout的合体

所以,hover要绑定两个事件处理函数

何时使用: 只要同时响应mouseover和mouseout时，就可简写为hover

2. 动画:高级函数: 基于底层函数又进行了封装

两大块: 简化版动画函数 和万能动画函数

简化版动画函数:

显示/隐藏: $(...).show();$(...).hide(); $(...).toggle()

强调: 无参数的show()/hide()使用的是display属性

瞬间隐藏和显示

动画: 参数: speed:

2种: 1. 三档: fastnormalslow

2. 用毫秒数自定义动画时长

动画的速度变化: 参数: easing:linearswing

伸缩: slideUp()slideDown() slideToggle()

淡入淡出:fadeIn()fadeOut() fadeToggle();

问题: 效果受局限

解决: 万能动画函数: animate()

animate()可对数值类型的CSS样式执行定时器动画

包括: 宽高,位置,透明度,边框宽度,字体大小,

强调: 不能对非数值类型属性做动画

包括: (颜色,背景图片,字体,display)

如何:

$(...).animate(params,speed,easing,fn)

其中: params: 所有变化的css属性的目标值:

比如: {css属性1: 值1, css属性2:值2,...}

强调: css属性名要去横线变驼峰

值要加单位

speed: 动画持续时间/速度

easing: 速度变化效果

fn: 动画结束后，自动调用的回调函数

动画中的排队和并发:

排队: 多个动画顺序，先后执行

$(...).animate({属性1:值1,...},ms).animate({属性2:值2,...})

先对属性1执行动画效果，再对属性2执行动画效果

并发: 多个动画同时执行

$(...).animate({属性1:值1,属性2:值2},ms)

同时修改属性1和属性2

正课:

1. 类数组操作:

2. jQuery UI:

1. **类数组操作: jquery是一个自带遍历的类数组对象，所以有length属性 $(…).lenght**

.each(function(i){ i->当前元素的下标;this->当前元素})

.each vs forEach

原理是一样的

forEach是原生js中Array类型的方法

each是jquery中第三方定义的类数组对象的方法

$("要查找的元素").index("所有元素")

返回，要查找的元素在所有元素中的下标位置

2. jQuery UI: 一组可复用的更简化版API或插件库

何时: 如果觉得jQuery还不够简化，或需要复用一些插件时

比如: 日历选择框 验证

插件: 官方插件第三方插件: 自定义插件

什么是jQuery UI: 基于jquery编写的一套UI组件库，专用于PC端浏览器。

jQuery Mobile: 基于jquery编写的一套UI组件库, 专门用于移动端浏览器的网页

官网: jqueryui.com

问题: 公司要使用一项新的技术或库，如何快速上手:

官网->什么是->DEMO->API 手册

下载: jQuery UI: 默认的色调是蓝色

希望获得自定义主题的UI库: 官网->Themes->Gallary

需要更细致的定制主题: Gallary->edit

正式学习jquery UI: 三部分: Effectsinteractions Widgets

Effects: 动画效果

addClass/removeClass/toggleClass() 默认没动画，现在有

show/hide/toggle 扩展了动画效果

color animation: 实现颜色动画

练习:

强调: 因为jquery ui依赖于jquery，所以必须先引入jquery.js，再引入jquery ui.js。

jquery ui中的$(...).animate(): 支持颜色动画

jquery ui中的$(...).toggle(Effects,speed)

toggle的effects参数:

blind(遮蔽)bounce(弹跳) clip(缩短) drop(丢弃)

explode(爆炸) fold(折叠) highlight(高亮) puff(膨胀)

pulsate(心跳) shake(震动) size slide

Interactions:交互

1. 拖拽: $(...).draggable()

问题: API会悄悄的为元素添加class->侵入性

对于有侵入性API，不但要引入js，还要引入css

2. 可选中: $(...).selectable();

问题: 可自动向选中的li添加ui-selected类，但jquery ui并未定义该类的样式

解决: 必须手动定义同名样式类ui-selected，定义选中项的样式

3. 可排序: $(...).sortable();

总结: jquery ui中的很多API具有侵入性:

通常都要引入jquery ui附带的css文件

有的还需要自己补充同名的样式类

要求: 对jquery ui不但要知道表面现象，更要知道Element中的变化原理。

Widget: 小部件: (小重点)

1. Accordion(手风琴部件)

何时: 节约页面空间

如何使用:

引入: jquery-ui.css

HTML: 父元素>标题+内容div

js:$(父元素).accordion();

2. AutoComplete(自动完成)

何时: 避免重复输入,减少键入次数

如何:

html: input文本框

js: 一个保存所有备选向的数组

$(文本框).autocomplete({source:数组})

3. datepicker:（日期选择框）

如何使用:

html: input文本框

js中: $(文本框).datepicker({dateFormat:'yyyy年mm月dd日'})

vs h5中<input type="date"...

h5有兼容性问题，且不能修改样式

4. dialog:对话框:

为什么: BOM中的alert,confirm,prompt样式死板，无法定制

何时: 今后，只要弹窗，都用dialog部件

如何:

html中:定义外层元素，包含提示信息

其中外层元素要定义title属性

title属性会出现在对话框的左上角

js: $("外层元素").dialog();

强调: 不带参数的dialog(), 只负责将容器变成对话框

不控制显示与否

如何隐藏，再弹出:

配置对话框:

$(...).dialog({

autoOpen:false,//默认隐藏

buttons:[

{text:"文本",click:function(){this->dialog}},//按钮

...

]

});

打开: $(...).dialog("open");

关闭: $(...).dialog("close");

5. 按钮部件: $(button/input/a).button();

6. tabs: 标签页:

如何:

html: 父级div>标签页ul>li>a href="#id"

内容页div#id

说明: li>a的href中的#id是为了匹配div的#id

js: $(父元素).tabs();

7. tooltip: 工具提示

为什么: 普通的title属性，浏览器之前有不一致

何时: 只要希望所有浏览器统一title提示的样式时

如何: tooltip，必须应用到document上

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月23日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

正课:

1. 第三方插件:

日期，验证，瀑布流

2. \*\*\*开发自定义jQuery插件

1. 第三方插件:

日期: jquery ui datepicker

h5 input type="date"

My97

layDate

layDate:不依赖于jquery

如何:

html: <input class="laydate-icon"

css: 引入laydate/skins/default/laydate.css

js: laydate({

elem:"selector",

format:"YYYY年MM月DD日",

其他属性

})

表单验证: jquery.validator依赖于jquery

如何使用:

$(表单).validate({

rules: {//规则

name1: "required",//必填,如果只有一个简单规则时

name2: {//如果一个字段，需要多个规则验证时

规则1: 值,

规则2: 值,

},//所有规则列表: https://jqueryvalidation.org/documentation/

List of built-in Validation methods

},

message:{//提示信息

name1: "name1验证失败的信息",

name2:{

规则1: "name2的规则1验证失败的信息",

规则2: "name2的规则2验证失败的信息",

}

}

});

如何添加自定义验证规则:

jQuery.validator.addMethod(规则名,method[,默认提示])

使用自定义规则: field: "规则名"

Masonry: 瀑布流

特点: 不规则大小的砖块，拼凑成完整的砖墙

什么是Masonry: 是一个javascript网格布局库.

作用: 在一个垂直区域内，将元素按最优化的空间利用率自动摆放。

如何使用: $(容器父元素).masonry();

2. \*\*\*自定义jquery插件:

需求: 添加一个专门的函数sum，来计算数组中所有元素的和

问题: 直接定义在全局，易被污染

解决: 定义在一个自定义的对象中: 2步:

1. 定义一个全局对象;

2. 在自定义的全局对象中添加自定义方法

优: 不易被污染

html如何使用自定义方法: 2步:

1. 先引入方法所在的js文件

2. 用库对象.自定义方法()

添加jquery全局函数:

正课:

1. \*\*\*自定义jQuery插件:

jQuery的全局方法

添加jQuery的对象方法——自定义插件

1. jQuery全局方法:

jQuery库对象: window.jQuery=window.$=jQuery

向jQuery库中添加全局方法: 2步:

1. jQuery.命名空间={}

2. jQuery.命名空间.方法名=function(){...}

原生js的全局方法vs jQuery的全局方法

原生js的全局方法: 直接定义在window下，不需要任何对象即可直接访问。

jQuery的全局方法: 定义在jQuery对象内部，必须用jQuery对象才能调用

2. 添加jQuery对象方法:

为所有jQuery查找结果对象添加的方法——插件

vs jQuery全局方法:

全局方法: $.xxx.方法名()

对象方法: $(selector).方法名()

何时: 如果希望对查找结果对象调用方法时

如何定义: $(...)的原型对象是jQuery.fn

鄙视题: $.each(arr,function(i,val){...})

arr.forEach(function(val,i,arr){...})

vs$(...).each(function(i,elem){...})

$.each()是jQuery全局方法，存储在jQuery全局对象上

主要用于遍历普通数组

**$(...).each()是jQuery对象方法，存储在原型对象jQuery.fn上**

**主要用于对查询结果中的每个DOM元素执行操作**

3. 自定义jQuery插件:

何时: 如果一个效果需要反复使用，就要封装为插件

优: 重复利用, 提高发开效率

1. dropdown:

第三阶段

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月27日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PHP基础**

1.什么是PHP？

PHP：PHP Hypertext Preprocessor超文本预处理器，服务器端的脚本语言

2.PHP环境

WAMP

Windows+Apache+PHP+MySQL

LAMP

Linux+Apache+PHP+MySQL

独立安装

集成环境

xampp/wampserver/phpstudy

访问WEB服务器

http://localhost:80

http://127.0.0.1:80

apache服务器主目录

htdocs

apache配置文件

httpd.conf

3.PHP基础

**PHP文件**

**.php结尾的**

PHP文件组成

html

css

js

jquery

php

PHP语法结构

标准模式

<?php

代码段;

?>

注意

如果只有PHP代码，结束标记?>可以省略

如果文档中不只有PHP代码，<?php ?>成对出现

Parse error（解析错误）:

syntax error(语法错误),

unexpected '<' in

C:\xampp\htdocs\server\day01\demo\5-test.php on line 3

PHP变量

变量就是在程序执行期间可以变化的量，保存数据的

**声明变量以$变量名称=值;**

注意

变量名称以字母或者下划线开始，后面跟上数字字母下划线

变量名称最好含义明确

变量名称遵循驼峰标记法或者下划线法

变量名称严格区分大小写

**变量如果重名产生覆盖**

PHP也是弱类型语言，可以不用声明直接使用

PHP中数据类型

8种主要数据类型

标量类型

整型

int|integer,整数存储范围-21亿~21亿，0~42亿

浮点型

float|double|real,小数，12.3,2e3

字符串型

string,字符串需要写到定界符，''或者""

注意

单引号不解析变量

双引号解析变量

布尔类型

bool|boolean,true|TRUE,false|FALSE

复合类型

数组

array

对象

object

特殊类型

资源

resource

空

null|NULL

注意

声明变量并且赋值为null

未声明的变量值为NULL

经过unset()注销过的变量值为NULL

常量

自定义常量

定义常量

define(name,value)

const NAME=VALUE;

注意

常量名称不要包含$,最好大写

常量一经定义不能改变

使用常量

直接写常量的名称

系统常量

PHP\_VERSION

PHP版本

PHP\_OS

运行环境

魔术常量

\_\_LINE\_\_

得到当前行号

\_\_FILE\_\_

得到文件的绝对路径包含文件名

\_\_DIR\_\_

得到文件的绝对路径不包含文件名

预定义变量

$\_GET

接收地址栏以?形式传递的参数

超链接典型的?形式传递参数

表单以get形式发送数据

$\_GET['名称']

$\_POST

接收表单以post方式发送数据的值

$\_POST['名称']

$\_REQUEST

$\_GET+$\_POST+$\_REQUEST

PHP中数据类型转换

隐式转换(自动转换)

其他类型转换成数值型

true->1

false->0

'31king'->31

'true'->0

null->0

其他类型转换成字符串型

true->字符串1

false->空字符串

null->空字符串

数值型->字符串的数值型

其他类型转换成布尔类型的假的有

0->false

0.0->false

''或者'0'->false

null->false

空数组array()|[]->false

显示转换(强制转换)

临时转换

小括号的形式

(int)|(integer)

(float)|(double)|(real)

(string)

(bool)|(boolean)

(unset)

(array)

(object)

函数的形式

intval($var)

floatval()|doubleval()

strval()

boolval()

注意

临时转换不影响变量本身

永久转换

永久转换会真正改变变量的类型

settype($var,$type)

设置变量的类型

gettype($var)

返回变量的类型

注意

不要使用gettype获取变量类型

PHP运算符

算术运算符

+ - \* / % ++ --

字符连接符

.拼接字符串的

赋值运算符

= += -= \*= /= %= .=

比较运算符

>>= <<= == === != !==

注意

比较运算符的结果是true|false

逻辑运算符

三元运算符

其它常用

echo

输出一个或者多个字符串

输出多个的时候通过英文的逗号分隔

var\_dump($var,...)

打印变量的详细信息

Js 中：console.dir();

打印变量的具体信息

unset($var,...)

销毁变量

header('cotnent-type:text/html;charset=utf-8')

告诉浏览器以什么编码方式解析什么类型文档

常见错误

Parse Error(解析错误),Syntax Error(语法错误)

Notice(通知)，可以设置PHP的错误级别

需要修改PHP的配置文件,php.ini这是php的配置文件，搜索error\_reporting=E\_ALL&~E\_NOTICE

需要重新启动web服务器，重启apache生效

CSS3地址

链接：http://pan.baidu.com/s/1eRAQnDK 密码：hkhx

1.什么是PHP？

PHP：PHP Hypertext Preprocessor超文本预处理器，服务器端的脚本语言

2.PHP环境

WAMP

Windows+Apache+PHP+MySQL

LAMP

Linux+Apache+PHP+MySQL

独立安装

集成环境

xampp/wampserver/phpstudy

访问WEB服务器

http://localhost:80

http://127.0.0.1:80

apache服务器主目录

htdocs

apache配置文件

httpd.conf

3.PHP基础

PHP文件

.php结尾的

PHP文件组成

html

css

js

jquery

php

PHP语法结构

标准模式

<?php

代码段;

?>

注意

如果只有PHP代码，结束标记?>可以省略

如果文档中不只有PHP代码，<?php ?>成对出现

Parse error（解析错误）:

syntax error(语法错误),

unexpected '<' in

C:\xampp\htdocs\server\day01\demo\5-test.php on line 3

PHP变量

变量就是在程序执行期间可以变化的量，保存数据的

声明变量以$变量名称=值;

注意

变量名称以字母或者下划线开始，后面跟上数字字母下划线

变量名称最好含义明确

变量名称遵循驼峰标记法或者下划线法

变量名称严格区分大小写

变量如果重名产生覆盖

PHP也是弱类型语言，可以不用声明直接使用

PHP中数据类型

8种主要数据类型

标量类型

整型

int|integer,整数存储范围-21亿~21亿，0~42亿

浮点型

float|double|real,小数，12.3,2e3

字符串型

string,字符串需要写到定界符，''或者""

注意

单引号不解析变量

双引号解析变量

布尔类型

bool|boolean,true|TRUE,false|FALSE

复合类型

数组

array

对象

object

特殊类型

资源

resource

空

null|NULL

注意

声明变量并且赋值为null

未声明的变量值为NULL

经过unset()注销过的变量值为NULL

常量

自定义常量

定义常量

define(name,value)

const NAME=VALUE;

注意

常量名称不要包含$,最好大写

常量一经定义不能改变

使用常量

直接写常量的名称

系统常量

PHP\_VERSION

PHP版本

PHP\_OS

运行环境

魔术常量

\_\_LINE\_\_

得到当前行号

\_\_FILE\_\_

得到文件的绝对路径包含文件名

\_\_DIR\_\_

得到文件的绝对路径不包含文件名

预定义变量

$\_GET

接收地址栏以?形式传递的参数

超链接典型的?形式传递参数

表单以get形式发送数据

$\_GET['名称']

$\_POST

接收表单以post方式发送数据的值

$\_POST['名称']

$\_REQUEST

$\_GET+$\_POST+$\_REQUEST

PHP中数据类型转换

隐式转换(自动转换)

其他类型转换成数值型

true->1

false->0

'31king'->31

'true'->0

null->0

其他类型转换成字符串型

true->字符串1

false->空字符串

null->空字符串

数值型->字符串的数值型

其他类型转换成布尔类型的假的有

0->false

0.0->false

''或者'0'->false

null->false

空数组array()|[]->false

显示转换(强制转换)

临时转换

小括号的形式

(int)|(integer)

(float)|(double)|(real)

(string)

(bool)|(boolean)

(unset)

(array)

(object)

函数的形式

intval($var)

floatval()|doubleval()

strval()

boolval()

注意

临时转换不影响变量本身

永久转换

永久转换会真正改变变量的类型

settype($var,$type)

设置变量的类型

gettype($var)

返回变量的类型

注意

不要使用gettype获取变量类型

PHP运算符

算术运算符

+ - \* / % ++ --

字符连接符

.拼接字符串的

赋值运算符

= += -= \*= /= %= .=

比较运算符

>>= <<= == === != !==

注意

比较运算符的结果是true|false

逻辑运算符

三元运算符

其它常用

**echo**

**输出一个或者多个字符串**

**输出多个的时候通过英文的逗号分隔**

var\_dump($var,...)

打印变量的详细信息

unset($var,...)

销毁变量

header('cotnent-type:text/html;charset=utf-8')

告诉浏览器以什么编码方式解析什么类型文档

常见错误

Parse Error(解析错误),Syntax Error(语法错误)

Notice(通知)，可以设置PHP的错误级别

需要修改PHP的配置文件,php.ini这是php的配置文件，搜索error\_reporting=E\_ALL&~E\_NOTICE

需要重新启动web服务器，重启apache生效

CSS3地址

链接：http://pan.baidu.com/s/1eRAQnDK 密码：hkhx

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月28日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 注释内容

-- 注释内容

# 创建数据库

CREATE DATABASE db\_name;

# 得到所有已创建

SHOW DATABASES;

# 打开指定的数据库

USE db\_name;

# 得到当前打开的数据库名称

SELECT DATABASE();

# 删除指定数据库

DROP DATABASE db\_name;

# 创建数据表

# 在test数据库下创建user表

# primary key 主键

# auto\_increment 自增长

CREATE TABLE user1(

id int primary key auto\_increment,

username varchar(20),

password char(32),

age tinyint,

addr varchar(50),

salary float(8,2)

);

#insert user1(id,username,password,age,addr)

values(4,"卫盈",'1234','18','衢州'),

(5,"大哥",'1234','28','黑龙江'),

(6,"吵吵",'1234','18','宁波');

#该表格中的指定用户的信息

update user set age=20, password='132356' where id=2;

#将用户名为king 的并且密码为123的用户年龄改为18；

update user1 set age=18 where password=123&&username=‘king’;

#改所有用户的指定信息

update user1 setage=30;

#将编号为3的用户地址改为上海；

update user1 setaddr='上海' where id=3;

#将编号大于等于5的用户age设置为20；

update user1 setage=20 where id>=5;

#将年龄小于等于40的或者地址为bj的用户

#修改密码为123

**update表名setpassword='123' where age>=20 or addr='bj';**

update表名setpassword='123' where age>=20 or addr='bj';

#彻底清空表

TRUNCATE TABLE user1；

#删除所以记录delete FROMtable\_name where;

#删除编号为1 的用户

delete FROMuser1 where id=1;

#删除年龄小于等于30 的用户

delete FROMuser1 where age<=30;

#删除用户名为king并且密码为12345的用户

delete FROMuser1 where username='大哥'&&password=‘12345’;

#删除表中所以记录

delete FROMuser1

#向表中添加5条记录

#insert user1(id,username,password,age,addr)

values(1,"大饼",'1343534','38','杭州'),

(2,"军军",'3445234','58','金华'),

(3,"统统",'1234','15','甘肃'),

(4,"卫盈",'1234','18','衢州'),

(5,"大哥",'1234','28','黑龙江'),

(6,"吵吵",'1234','18','宁波');

#一次清空表中所以记录

TRUNCATE TABLE user1;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月29\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CREATE DATABASE cms;

SHOW DATABASES;

USE cms;//打开数据库、链接数据库

SELECT DATABASE();

DROP DATABASE cms;//删除数据库

# 创建数据表user

# unsigned 无符号，也没有负数，从0开始

CREATE TABLE user(

id int primary key auto\_increment,

username varchar(30),

password char(32),

age tinyint unsigned,

addr varchar(50)

);

# 插入记录

# INSERT INTO tbl\_name(字段名称,...) VALUES(值,...)

INSERT user(id,username,password,age,addr)

VALUES(1,'king','123456',22,'bj');

INSERT user(id,username,password,age,addr)

VALUES(1,'king','123456',12,'bj');

# 注意：第一次插入中文之前需要临时改变客户端的编码方式，

# SET NAMES GBK;

INSERT user(id,username,password,age,addr)

VALUES(2,'曹云金','123',34,'北京');

# 插入值不指定id，id是自增长的

INSERT user(username,password,age,addr)

VALUES('王宝强','1234',34,'北京');

# 查询记录

# SELECT \* FROM tbl\_name;

SELECT \* FROM user;

修改编码方式：

my.ini

找到[client]

default-character-set=utf8

找到[mysqld]

character\_set\_server=utf8

重启mysql服务器

[client]

default-character-set=utf8

[mysqld]

character\_set\_server=utf8

# 临时转换客户端的编码方式

SET NAMES GBK;

# 创建数据库的同时指定编码方式

CREATE DATABASE cms

DEFAULT CHARACTER SET 'UTF8';

# 查看指定数据库的编码方式

SHOW CREATE DATABASE cms;

CREATE TABLE user(

id int primary key auto\_increment,

username varchar(30),

password char(32),

age tinyint unsigned,

addr varchar(50)

)DEFAULT CHARSET=UTF8;

# 查看指定数据表详细信息

SHOW CREATE TABLE user;

# 向user表插入记录

INSERT user(id,username,password,age,addr)

VALUES(1,'king','123',12,'bj');

# 查询表中所有记录

SELECT \* FROM user;

INSERT user(id,username,password,age,addr)

VALUES(2,'张三','1234',12,'上海');

# 测试自动增长

INSERT user(username,password,age,addr)

VALUES('是对方的发生','123',12,'SH');

# 一次向表中插入多条记录

INSERT user(username,password,age,address)

VALUES('A1','A1',12,'bj'),

('B2','B2',22,'SS'),

('B3','B4',32,'1SS'),

('B4','B4',32,'2SS');

# 修改记录UPDATE

# 修改了表中的所有记录

UPDATE tbl\_name SET 字段名称=值,...;

# 修改了指定条件的记录

UPDATE tbl\_name SET 字段名称=值,字段名称=值,...

WHERE 条件;

# 将用户表中的age字段改为20

UPDATE user SET age=20;

# 将所有用户的age改为30 addr改为bj

UPDATE user SET age=30,addr='bj';

# 将id=4的用户age=20,username='lisi',addr='sh'

UPDATE user SET age=20,username='lisi',addr='sh'

WHERE id=4;

# 将编号大于等于6的用户年龄改为35

UPDATE user SET age=35 WHERE id>=6;

# 将地址为bj的用户年龄该位10

UPDATE user SET age=10

WHERE addr='bj';

# 将用户名为king的并且密码为123的用户年龄改为18

UPDATE user SET age=18

WHERE username='king1' AND password='123';

# 将所有用户年龄改为30

UPDATE user SET age=30;

# 将编号为3的用户地址改为上海

UPDATE user SET addr='上海'

WHERE id=3;

# 将编号大于等于5的用户age设置为20

UPDATE user SET age=20 WHERE id>=5;

# 将年龄小于等于40的或者地址为bj的用户，

# 密码修改为123

UPDATE user SET password='123456789'

WHERE age>=20 OR username='king';

# 删除记录 DELETE

# 删除表中的所有记录

DELETE FROM tbl\_name ;

# 删除指定条件的记录

DELETE FROM tbl\_name WHERE 条件;

# 删除编号为6的用户

DELETE FROM user WHERE id=6;

# 删除年龄为10的用户

DELETE FROM user WHERE age=10;

# 删除所有记录

DELETE FROM user;

INSERT user(username,password,age,address)

VALUES('A','A',12,'BJ');

# 彻底清空数据表 TRUNCATE

# 把表中的所有记录删除，而且将auto\_increment的值设置为1

TRUNCATE TABLE tbl\_name;

INSERT user(username,password,age,address)

VALUES('A21','A1',12,'BJ'),

('A11','A21',12,'BJ'),

('A111','A21',12,'BJ'),

('A1111','A21',12,'BJ'),

('A11111','A21',12,'BJ');

# DELETE FROM tbl\_name WHERE 条件

# 彻底清空数据表

TRUNCATE TABLE user;

# 删除编号为1的用户

DELETE FROM user WHERE id=1;

# 删除年龄小于等于30的用户

DELETE FROM user WHERE age<=30;

# 删除用户名为king并且密码为123456789的用户

DELETE FROM user WHERE username='king' AND password='123456';

# 删除表中所有记录

DELETE FROM user;

# 向表中添加5条记录

INSERT user(username,password,age,address)

VALUES(...),

(...);

# 一次清空表中所有记录

TRUNCATE TABLE user;

# 查询记录 SELECT

SELECT \* FROM tbl\_name;

# 查询指定字段的值

SELECT 字段名称,.. FROM tbl\_name;

SELECT username FROM user;

SELECT id,username,age FROM user;

# 可以查询出满足条件的记录

SELECT \* FROM user WHERE id=1;

# 查询用户名为king并且密码为123456的用户

SELECT \* FROM user WHERE username='king'

AND password='123456';

SELECT \* FROM tbl\_name;

# 查询id ,username,age字段的数据

SELECT id,username,age FROM user;

# 查询用户名为xxx 并且密码为xxx的用户

SELECT \* FROM user WHERE username='xx' AND password='xx';

# 查询地址为bj的用户

SELECT \* FROM user WHERE addr='bj';

# 查询年龄小等于30 并且 地址为bj的用户

SELECT \* FROM user WEHRE age<=30 AND addr='bj';

MYAQL

MySQL数据库

1.MySQL数据库

存储数据库的仓库

通过SQL(结构化查询语言)来操纵数据库

DDL:数据定义语言，创建库、创建表、选择

DML:数据操作语言，完成数据的增、删、改

DQL：数据查询语言，完成数据的查询

DCL：数据控制语言，像授权、回收权限

连接MySQL

mysql -uroot

mysql -uroot -p密码

SQL命令书写规范

SQL是不区分大小写的

SQL语句默认以分号;结束

SQL语句支持折行操作

数据库

创建数据库

CREATE DATABASE db\_name;

如果数据库不存在则创建

CREATE DATABASE IF NOT EXISTS db\_name;

创建数据库的同时指定编码方式

CREATE DATABASE db\_name

DEFAULT CHARACTER SET 'UTF8';

查询当前服务器下所有数据库

SHOW DATABASES;

打开指定数据库

USE db\_name;

得到当前打开的数据库

SELECT DATABASE();

删除数据库

DROP DATABASE db\_name;

DROP DATABASE IF EXISTS db\_name;

数据表

创建数据表

CREATE TABLE tbl\_name(

字段名称 字段类型 [约束条件]

);

当前数据库下的数据表

SHOW TABLES;

查看数据表的表结构

DESC tbl\_name;

删除指定数据表

DROP TABLE tbl\_name;

MySQL的数据类型

整型

tinyint

-128~127,0~255

1字节

int

-21亿~21亿，0~42亿

4字节

浮点型

float(M,D)

M总长度，D小数点后几位

float(8,2)

字符串型

char()

定长字符串

char(50)

varchar()

变长字符串

varchar(50)

数据操作

插入记录

INSERT

插入单条记录

INSERT tbl\_name(字段名称,...) VALUES(字段值,...);

插入多条记录

INSERT tbl\_name(字段名称,...) VALUES(字段值,...)

,(字段值,...),...;

修改记录

UPDATE

更新表中所有记录

UPDATE tbl\_name SET 字段名称=值,...;

按照指定条件更新记录

UPDATE tbl\_name SET 字段名称=值,...

WHERE 条件;

删除记录

DELETE

删除所有记录

DELETE FROM tbl\_name

按照条件删除指定记录

DELETE FROM tbl\_name WHERE 条件

彻底清空数据表

TRUNCATE TABLE tbl\_name;

查询记录

SELECT

查询表中所有记录

SELECT \* FROM tbl\_name;

指定字段

SELECT 字段名称,... FROM tbl\_name

按照条件查找指定数据

SELECT 字段名称,... FROM tbl\_name WHERE 条件;

注意

解决phpmyadmin缺少扩展问题

打开PHP的配置文件

搜索extension=

extension=php\_mbstring.dll

extension=php\_mysql.dll

extension=php\_mysqli.dll

extension=php\_pdo\_mysql.dll

保存，重启启动apache服务器即可

1.MySQL数据库

存储数据库的仓库

通过SQL(结构化查询语言)来操纵数据库

DDL:数据定义语言，创建库、创建表、选择

DML:数据操作语言，完成数据的增、删、改

DQL：数据查询语言，完成数据的查询

DCL：数据控制语言，像授权、回收权限

连接MySQL

mysql -uroot

mysql -uroot -p密码

SQL命令书写规范

SQL是不区分大小写的

SQL语句默认以分号;结束

SQL语句支持折行操作

**数据库**

**创建数据库**

CREATE DATABASE db\_name;

如果数据库不存在则创建

CREATE DATABASE IF NOT EXISTS db\_name;

创建数据库的同时指定编码方式

CREATE DATABASE db\_name

DEFAULT CHARACTER SET 'UTF8';

查询当前服务器下所有数据库

SHOW DATABASES;

打开指定数据库

USE db\_name;

得到当前打开的数据库

SELECT DATABASE();

删除数据库

DROP DATABASE db\_name;

**DROP DATABASE IF EXISTS db\_name;**

数据表

创建数据表

CREATE TABLE tbl\_name(

字段名称 字段类型 [约束条件]

);

当前数据库下的数据表

SHOW TABLES;

查看数据表的表结构

DESC tbl\_name;

删除指定数据表

DROP TABLE tbl\_name;

MySQL的数据类型

整型

tinyint

-128~127,0~255

1字节

int

-21亿~21亿，0~42亿

4字节

浮点型

float(M,D)

M总长度，D小数点后几位

float(8,2)

字符串型

char()

定长字符串

char(50)

varchar()

变长字符串

varchar(50)

数据操作

插入记录

INSERT

插入单条记录

INSERT tbl\_name(字段名称,...) VALUES(字段值,...);

插入多条记录

INSERT tbl\_name(字段名称,...) VALUES(字段值,...)

,(字段值,...),...;

修改记录

UPDATE

更新表中所有记录

UPDATE tbl\_name SET 字段名称=值,...;

按照指定条件更新记录

UPDATE tbl\_name SET 字段名称=值,...

WHERE 条件;

删除记录

DELETE

删除所有记录

DELETE FROM tbl\_name

按照条件删除指定记录

DELETE FROM tbl\_name WHERE 条件

彻底清空数据表

TRUNCATE TABLE tbl\_name;

查询记录

SELECT

查询表中所有记录

SELECT \* FROM tbl\_name;

指定字段

SELECT 字段名称,... FROM tbl\_name

按照条件查找指定数据

SELECT 字段名称,... FROM tbl\_name WHERE 条件;

注意

解决phpmyadmin缺少扩展问题

打开PHP的配置文件

搜索extension=

extension=php\_mbstring.dll

extension=php\_mysql.dll

extension=php\_mysqli.dll

extension=php\_pdo\_mysql.dll

保存，重启启动apache服务器即可

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*9月30日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

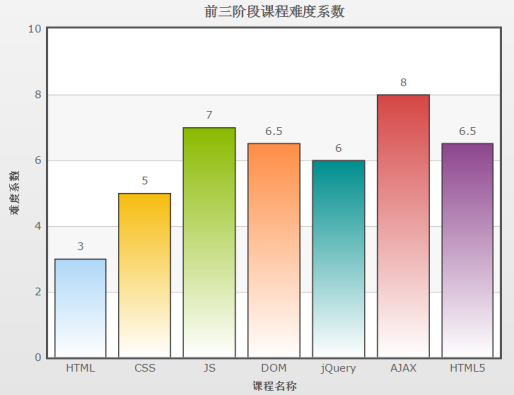
李文华

第一阶段：HTML + CSS =>静态页

第二阶段：JS + DOM + jQuery =>静态页面添加交互

第三阶段：Server+ HTTP + AJAX + HTML5 =>动态页面

第四阶段：前端框架



第三阶段课程：

(1)AJAX

1.1)数据库服务器

1.2)Web服务器

1.3)HTTP协议

1.4)原生AJAX

1.5)jQuery中的AJAX

(2)HTML5高级特性

今天课程特点：没有一行代码！纯理论！

今日课程内容：

(1)URL和URI

(2)HTTP背景

(3)HTTP请求消息格式——重点

(4)HTTP响应消息格式——重点

1.学习HTTP协议目的

(1)调试AJAX的错误——看不见摸不着

(2)指出网站访问优化方案——高水平前端必备

2.URL和URI、URN

Unified Resource Locator：统一资源定位符

Unified Resource Identifier：统一资源识别符

Unified Resource Naming：统一资源命名符

整数=负整数+ 非负整数

**URI= URL +URN**

URL:http://tmooc.cn:80/index.html?xxx=yyy

URL:jdbc:mysql://127.0.0.1:3306

URN:mailto:admin@tmooc.cn

URN:tel: 13501234567

<a href="http://tmooc.cn/index.html?xxx=yyy ">

<a href="mailto:admin@tmooc.cn">

<a href="tel: 13501234567">

<a href="javascript: void(0)">

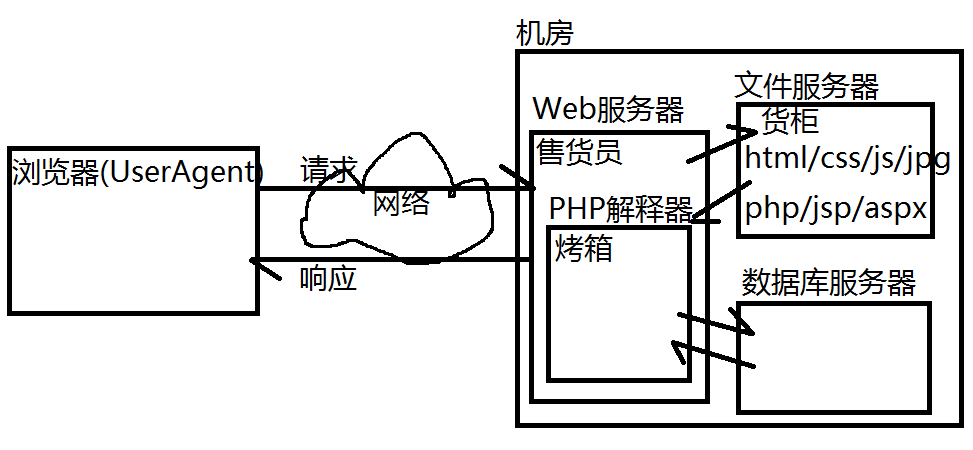
3.HTTP协议概述

历史：

1996年： HTTP/1.0

1999年： HTTP/1.1

|  |
| --- |
| 网络连接：三次握手- 发送请求 - 返回响应 - 四次挥手  面试题：HTTP/1.1比1.0有哪些改进？  (1)支持持久连接  Connection: keep-alive  不是永久连接  (2)支持虚拟主机  Host: tmooc.cn  一台服务器中可以同时存在多个网站，共用IP地址，但域名不同  (3)支持代理连接  Proxy: xxxx  客户端可以通过中间代理服务器访问目标服务器 |



4.HTTP协议的内容

HTTP协议的内容由IETF制定，

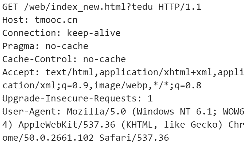
RFC Request For Comment

RFC-1945HTTP/1.0

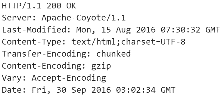
RFC-2616HTTP/1.1

指定了两种消息：

请求消息(request)：客户端发给服务器的



响应消息(response)：服务器发给客户端的



两种消息的具体格式参见“控制台中的Network”、思维导图。

|  |
| --- |
| <form action="" method="GET"> |
| 面试题：GET和POST请求的区别？  (1)二者语义不同：  GET:客户端想获取  POST:客户端想传递数据  (2)二者的安全级别完全一样——都不安全！  (3)发起GET请求的方法：  地址栏输入URL、超链接、JS跳转、GET方式提交表单、AJAX-GET提交  发起POST请求的方法：  POST方式提交表单、AJAX-POST请求  (4)提交表单时，二者的区别？  GET把请求数据追加在URI后面，以?开头——大部分浏览器/服务器对URI总长度有限制（如1024个字节）  POST把请求数据放在请求主体中——长度没有限制 |

|  |
| --- |
| GET /index.php HTTP/1.1  Host: www.baidu.com |
| HEAD /index.php HTTP/1.1  Host: www.baidu.com |
| DELETE /index.php HTTP/1.1  Host: www.taobao.com |

5.MIME类型

Multipurpose Internet Mail Extention，多用途互联网邮件扩展。

靠文件名的后缀无法可靠分辨的类型——靠MIME类型.

.html / .htm / .xhtml —— text/html

.jpg / .jpeg —— image/jpeg

.mp3 / .mpeg3 —— audio/mpeg3

6.如何控制响应消息头部

(1)方法1：修改Web服务器的配置文件，麻烦且有限制

(2)方法2：使用服务器端运行的动态语言来设置——最常用

如PHP：header('Content-Type: image/png');

header('Cache-Control: no-cache');

(3)方法3：修改HTML页面的meta标签

<meta http-equiv="Cache-Control" content="no-cache">

<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">

提示：刚才的验证码图片生成涉及到很多的PHP函数，没有深入讲解，感兴趣的同学可以仿照例子进一步研究，不作为必学内容！！

7.面试题：如何提高一个网站访问性能？ 如何进行网站的访问优化？

(1)域名解析

尽可能减少域名解析次数——减少跨站外部资源的引用

(2)创建连接

努力减少连接创建次数——使用持久连接避免重复连接

(3)发送请求

尽力减少请求次数——资源合并，合理的使用缓存

(4)等待响应

提高服务器端运行速度——使用服务器端缓存技术，提高数据运算及查询速度——后端程序员必学！

(5)接收响应

缩短响应消息下载时间，尽可能减小响应数据长度——启用压缩——修改Web服务器的配置文件

8.HTTP和HTTPS

HTTP：80 明文传输

HTTPS：443 加密传输，保证安全，但访问速度一般慢于HTTP协议

课下作业：

(1)通过百度，自学<base>标签的作用

(2)通过百度，自学表单实现文件上传时请求消息的主体格式

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月8日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

复习：

AJAX

数据库服务器 —— SQL

Web服务器 —— PHP

HTTP协议 —— 请求/响应

原生AJAX

jQuery中的AJAX

HTML5

Bootstrap

今日目标：

(1)复习数据库服务器

(2)复习PHP Web服务器

(3)原生AJAX概述

(4)使用AJAX发起异步请求——重点&难点

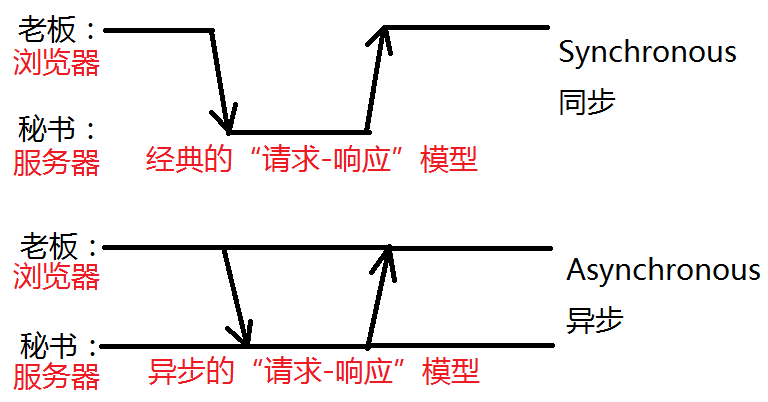
练习1：编写一个.sql脚本中，创建数据库名bbs，创建一个表user保存注册用户的信息(uid、uname、upwd、regTime)，插入三行记录，tom、mary、king，查询出所有的记录行

练习2：编写一个checkuname.php，接收客户端提交的uname数据，执行数据库查询，向客户端输出该用户名是否已经存在，如果已经存在，则向客户端输出cunzai，否则输出bucunzai

1.AJAX概述

Asynchronous Javascript And Xml，异步的JS和XML。

2001年，Google在搜索引擎应用中为了提高客户端的浏览体验，提出了AJAX的概念——Google Suggest。实现的目标：在无刷新无提交的页面内容局部的更新。涉及到的技术：HTML、CSS、JS、DOM、XML、HTTP —— 纯客户端技术的大集合。



浏览器发起的请求消息的类型：

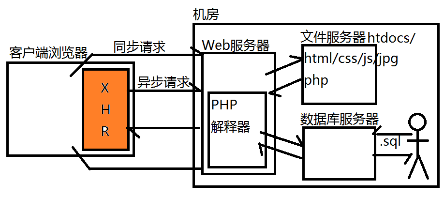
**(1)同步请求：**地址栏输入URL、超链接跳转、表单提交...

**(2)异步请求：**使用AJAX发起

提示：浏览器发起异步请求，依靠的是底层的XMLHttpRequest对象

|  |
| --- |
| 面试题：谈一下对AJAX理解 |

|  |
| --- |
| XHR对象的创建——注意浏览器兼容性  var xhr = null;  if(window.XMLHttpRequest){//新IE，其它浏览器  xhr = new XMLHttpRequest();  }else { //老IE  xhr = new ActiveXObject('Microsoft.XMLHTTP');  } |

****

2.XMLHttpRequest对象的常用成员：

注意：XHR对象的用途：异步的发起HTTP请求消息，并接收服务器返回的响应消息。



**成员属性：**

readyState：int，初始值为0；表示当前请求-响应进行到哪个步骤了，可取值0 - 1 - 2 - 3 - 4

0： UNSENT请求消息尚未发送

1： OPENEDXHR已经打开了到服务器的连接

2： HEADERS\_RECEIVED XHR已经接收到了响应消息起始行和头部

3： LOADING XHR正在加载响应消息的主体

4： DONE XHR已经接收完成响应消息

responseText：初始值为""，readyState值变为3开始才有响应消息主体文本

responseXML：初始值为null，readyState值变为3开始才有响应消息主体内容——XMLDOM树对象

status：初始值为0，保存着响应状态码，readyState值变为2开始才有值

statusText：初始值为""，保存着响应消息中的原因短句，readyState值变为2开始才有值

**成员事件：**

onreadystatechange：每次readyState值发生改变就触发

**成员方法：**

open(method, url)：打开到服务器的连接

send()：发送请求消息

setRequestHeader(name,value)：设置请求消息头部

getAllResponseHeaders()：获取所有的响应消息头部

getResponseHeader(name)：获取某个响应消息头部

使用XHR对象发起请求的步骤——重点：

(1)创建XHR对象

(2)监听XHR的就绪状态改变事件

(3)连接到服务器

(4)发起请求消息

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月9日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

复习：

AJAX：异步（浏览器和服务器同时工作）的JS和XML，Google Suggest中提出，实现(1)“无刷新无提交的页面内容局部更新”、(2)“动态网页(PHP)静态化(HTML)”

AJAX=HTML+CSS+JS+DOM+XML+HTTP

核心对象：XMLHttpRequest —— 用于发起异步HTTP请求，并接收服务器端返回响应消息

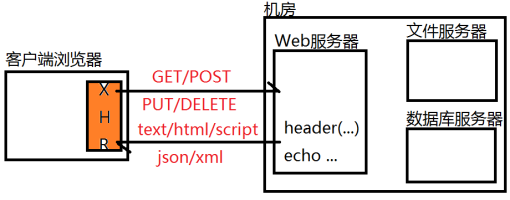
使用步骤：

(1) var xhr = new XMLHttpRequest();

(2) xhr.onreadystatechange = function(){ }

(3) xhr.open('GET/POST', 'x.php', true)

(4) xhr.send( null/data )



今日目标：

(1)使用XHR接收text类型响应消息 —— 掌握

(2)使用XHR接收html类型响应消息 —— 小重点

(3)使用XHR接收script类型响应消息 —— 掌握

(4)使用XHR接收xml类型响应消息 —— 了解

(5)使用XHR接收json类型响应消息 —— 重点

练习：实现一个无提交的商品信息录入系统

(1)编写一个jd.sql文件，数据库名：jd，产品信息表：product(pid, pname, price, pic, birthday)，插入三行测试数据，并查询出来；

(2)编写一个product\_add.php，接收客户端提交的商品信息，执行INSERT插入到数据库，返回插入成功的商品的自增编号$pid = mysqli\_insert\_id($conn)；

(3)编写一个product\_add.html，包含四个input，一个button，点击按钮异步提交四个数据（四步骤），给用户提示出添加的结果。 10:05

1.使用XHR接收服务器返回的数据——text

**服务器端：**

header('Content-Type: text/plain');

echo 'xxx';

**客户端：**

xhr.responseText; //xxx

示例：商品数据录入系统

2.使用XHR接收服务器返回的数据——html

注意：AJAX应用中，服务器端异步返回的html不是完整的HTML文档，而只是部分标签的片段，如：<li>XX</li>

**服务器端：**

header('Content-Type: text/html');

echo '<li>XX</li>';

**客户端：**

ul.innerHTML = xhr.responseText; //<li>XX</li>

示例：搜索建议 —— 重点！

实现步骤：

(1)编写SQL，创建数据库：jd，产品信息表：product

(2)编写PHP，接收客户端提交的搜索关键字kw，到产品表中进行模糊查询，获得包含指定关键字的所有产品名称，以HTML片段形式输出给客户端

SELECT pname FROM product WHERE pname LIKE'%奶%'; %匹配任意多个任意字符

(3)编写HTML，随着用户在输入框中的输入(onkeyup)，发起异步请求，获得对应的搜索建议 input.onkeyup = function(){}

3.使用XHR接收服务器返回的数据——script

**服务器端：**

header('Content-Type: application/javascript');

echo "alert(123);alert(456)";

**客户端：**

eval( xhr.responseText ) //"alert(123);alert(456)"

示例：实现具备国际化(i18n)功能的网页

(1)创建PHP，读取请求头部，获取客户端的首选语言，根据首选语言的不同，输出不同形式的欢迎消息

(2)创建HTML，点击“获取欢迎消息”，异步请求服务器端的PHP页面(四步骤)，得到不同形式不同语言的欢迎消息。

4.思考：如何处理 异步请求获取的批量数据

**服务器端：**

$list = [ [ ],[ ],[ ],[ ] ];

//echo $list; //PHP中的数组不能输出！且HTTP协议不能传输数组

提示：服务器和客户端只能传输字符串。批量数据必须转换为某种形式的字符串，才能使用HTTP协议进行传输。

**(1)text/plain**

金士顿#30#1.jpg@闪迪#35#2.jpg@三星#36#3.jpg

特点：简单，但容易出错

**(2)text/html**

<div><h3>金士顿</h3><b>30</b><img ="1.jpg"></div><div><h3>闪迪</h3><b>35</b><img ="3.jpg"></div><div><h3>三星</h3><b>36</b><img ="3.jpg"></div>

特点：直接可以使用，但灵活性不足

**(3)application/xml** —— W3C推荐字符串数据格式

**(4)application/json** —— W3C推荐字符串数据格式

5.使用XHR接收服务器返回的数据——XML

XML：eXtensible Markup Language，可扩展的标签语言——标签名称可以随意指定。语法要求：

(1)标签有开始，就必须有结束

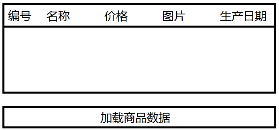
(2)标签可以有属性，属性值必须用单引号或双引号括起来

(3)标签可以嵌套，不能交叉

(4)一个大的XML文档/字符串片段有且只能有一个根标签

|  |
| --- |
| HTML和XML的区别？  (1)HTML语法比较松散不严格；XML语法要求严格  (2)HTML标签固定有100+个；XML标签名需要自定义  (3)HTML专用于网页，用于描述页面的内容；XML用于描述数据，尤其是批量的数据 |

示例：异步加载批量数据



实现步骤：

(1)编写SQL，数据库名：jd，商品信息表：product

(2)编写PHP，读取商品信息（保存在一个二维数组中），把数组中的数据转换为XML字符串格式，输出给客户端

(3)编写HTML，内容如上所示，点击“加载商品数据”按钮时，异步请求服务器端的PHP页面，读取其中的XML数据.......试着把XHR对象输出，看看XML数据的形式

|  |
| --- |
| 复习：DOM分为几种：  **核心DOM**：可以操作任意的DOM树  **HTMLDOM**：用于操作HTML文档  **XMLDOM**：操作XMLDOM树可以核心DOM方法 |

**服务器端：**

header('Content-Type: application/xml');

echo "<productList><product></product></productList>";

**客户端：**

//xhr.responseText string类型，不便于从其中读取数据

var document = xhr.responseXML; //一个XMLDOM树的根

提示：解析XMLDOM树可以使用核心DOM方法

6.使用XHR接收服务器返回的数据——JSON

JSON：JavaScript Object Notation，JS对象表示法，是一种字符串数据格式，作用于XML类型，用于描述数据，比XML更加精简；早期的语法来自于JS，但目前已经被几乎所有的语言所支持。语法：

(1)数据可以是对象形式： {}

也可以是数组形式：[]

(2)数据类型只有数字、字符串、布尔、null类型，没有其它类型

(3)字符串必须使用双引号，不能用单引号

(4)对象的属性名必须使用双引号，不能省略

(5)一个JSON字符串只能是一个{}或一个[]—— 根元素只能有一个！

**服务器端：**

header('Content-Type: application/json');

$list = [ ... ];

Echojson\_encode($list) ;//把PHP数组编码为JSON字符串

**客户端：**

JSON.parse( xhr.responseText ) //把JSON字符串解析为JS数组/对象

练习内容：

功能要求：

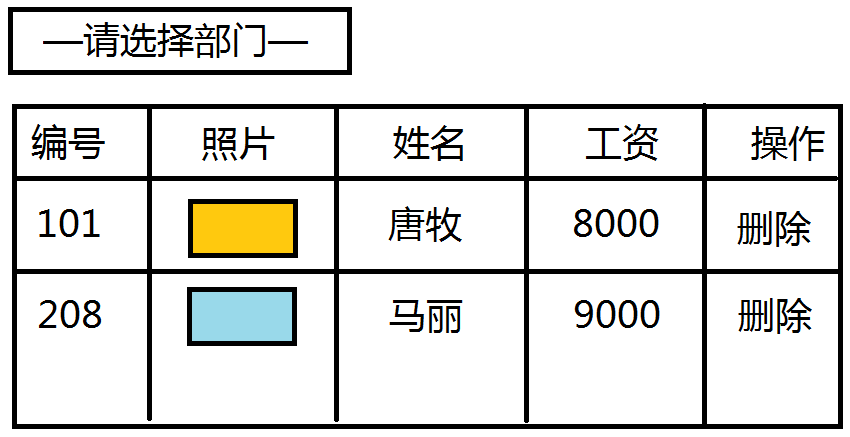
(1)编写SQL，数据库名: tarena，员工信息表emp(eid,ename,pic, salary,deptID)，保存多个员工的信息；

(2)编写PHP，emp\_select.php，根据客户端提交的部门编号(deptID)，返回该部门下的员工信息；

(3)编写HTML，emp\_select.html，下拉框中选择不同的部门，则下方的员工列表显示出该部门的员工信息；

(4)编写HTML，emp\_delete.html，根据客户端提交的员工编号，从数据库中删除该员工记录，返回succ或err；

(5)点击“删除”，发起异步请求删除该员工信息，服务器返回删除成功信息后，前端页面将表格中的当前行删除。



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月10日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

复习：

使用XHR发起两种请求

(1)1-2-3-4

(2) 1-2-3-3.5-4

使用XHR接收五种响应

(1)text

服务器端：

header('Content-Type: text/plain');

echo "succ";

客户端：

xhr.responseText //succ

(2) html

服务器端：

header('Content-Type: text/html');

echo "<li></li><li></li>";

客户端：

ul.innerHTML = xhr.responseText

(3) script

服务器端：

header('Content-Type: application/javascript');

echo "alert('123');alert(456)";

客户端：

eval( xhr.responseText )

(4) xml

服务器端：

header('Content-Type: application/xml');

echo "<plist><p></p><p></p></plist>";

客户端：

var xmlDocument = xhr.responseXML

(5) json

服务器端：

header('Content-Type: application/json');

echo "[{ },{ }]";

/\*$list = [ ];echo json\_encode($list);\*/

客户端：

JSON.parse( xhr.responseText )

练习：

(1)编写SQL，数据库名：tedu，表名student( sid, sname, photo, birthday, classId )，插入8条记录，分处于三个不同的班级

(2)编写PHP，student\_select.php，根据客户端提交的classId，返回该班中所有的学生信息

(3)编写HTML，student.select.html，下拉列表中选择不同的班级，则下方的学生表格中异步呈现出该班的学生

(4)编写PHP，student\_delete.php，接收客户端提交的sid，从数据库中删除该学生，返回succ或err

(5)修改HTML，student.select.html，为每个删除超链接添加事件监听，单击后异步提交请求

今日目标：jQuery中的AJAX封装函数

(1)load()

(2)$.get() —— 重点

(3)$.post()

(4)$.getScript()

(5)$.getJSON() —— 重点

(6)$.ajax ——重点

1.小知识点：

PHP中的json\_encode可以把PHP中的数组编码为一个JSON字符串，规则：

PHP索引数组 编码为 JSON：[]

PHP关联数组 编码为 JSON：{}

2. jQuery中的AJAX封装函数之一——load

使用方法：

$('选择器').load(url, [data], [fn])

$('选择器').load(url)

含义作用：

创建XHR对象，向指定的URL发起异步请求，请求参数为data，如果响应完成且成功，自动调用指定的回调函数fn。**把响应主体数据（必须是HTML片段）替换为当前选定元素的innerHTML，已有内容会被覆盖**。

使用限制：(1)服务器端返回的只能是HTML片段 (2)返回的内容会覆盖掉当前已有内容

应用示例：异步加载全站所有页面中相同的页头和页尾

待每个页面加载完成后，异步请求页头(header.php)和页尾(footer.php)的内容，放置到当前页面中

3. jQuery中的AJAX封装函数之二——$.get

使用方法：

$.get(url, [data], [fn])

含义作用：

创建XHR对象，向指定的URL发起异步GET请求，请求参数为data（k=v&k=v形式或{k:v, k:v}形式），如果响应完成且成功，自动调用指定的回调函数fn——必须在此回调函数中对响应数据进行处理——doResponse。

应用示例：注册学生信息时，输入框一失去焦点，立即异步验证该学生姓名是否已经存在

实现步骤：

(1)编写SQL， 数据库名：tedu，表名：student

(2)编写PHP，check\_sname.php，接收客户端提交的sname，返回它在数据库中是否存在，输出cunzai或者bucunzai

(3)编写HTML，student\_add.html，sname输入框一失去焦点，立即异步请求验证该学生名是否存在

$.get( url, {sname: '丁丁'},

function( txt, msg, xhr ){ } )

4. jQuery中的AJAX封装函数之三——$.post

使用方法：

$.post(url, data, [fn])

含义作用：

创建XHR对象，向指定的URL发起异步POST请求，请求参数为data（k=v&k=v形式或{k:v, k:v}形式）追加在请求主体中，如果响应完成且成功，自动调用指定的回调函数fn——必须在此回调函数中对响应数据进行处理——doResponse(txt, msg, xhr)。

应用示例：

(1)编写SQL， 数据库名：tedu，表名：student

(2)编写PHP，student\_add.php，接收客户端提交的学生信息，保存到数据库中，返回新记录在数据库中自增编号

(3)编写HTML，student\_add.html，点击“保存”按钮时异步提交用户输入的数据，提示出来：新记录保存成功！在数据库中的编号为：99

$.post( url, {sname: '丁丁', photo:'', birthday:'' },

function( txt, msg, xhr ){ } )

|  |
| --- |
| **表单序列化(Serialize)：**把表单中所有具备name属性的输入域及值拼接为一个k1=v1&k2=v2形式的字符串，可以直接用于HTTP请求消息var data = $('#formId').serialize( ); |

5. jQuery中的AJAX封装函数之四——$.getJSON

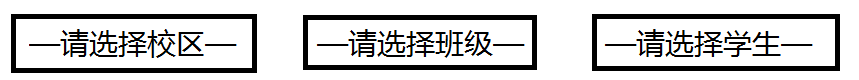
使用方法：

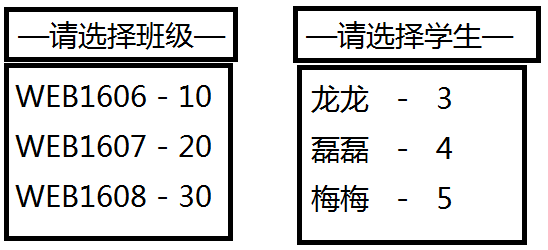
$.getJSON(url, [data], [fn])

含义作用：

创建XHR对象，向指定的URL发起异步GET请求，请求参数为data（k=v&k=v形式或{k:v, k:v}形式）追加在URL后，如果响应完成且成功，自动调用指定的回调函数fn——必须在此回调函数中对响应数据进行处理——doResponse(txt, msg, xhr)。**该函数要求相应数据必须是application/json类型，会自动调用JSON.parse(xhr.responseText)加以解析**。

应用示例：动态的级联下拉菜单





实现步骤：

(1)编写SQL， 数据库名：tedu，表名：student16:45

(2)编写PHP，student\_select.php，根据客户端提交的classId，向客户端返回该班的学生信息，形如：[ {"sid":3,"sname":"东东"},{}]

(3)编写HTML，student\_select.html，当班级列表中的选项发生改变时，异步请求服务器端返回的学生信息，转换为option添加到第二个select中

$.getJSON('xx.php', {k:v, k:v},

function(obj, msg, xhr){ console.log(arguments) })

|  |
| --- |
| **$.getJSON()**不论响应消息是什么类型，都会自动调用JSON.parse()把xhr.responseText进行解析，得到JS对象；  **$.get()** 取决于响应消息声明的内容类型，若声明为application/json，会自动调用JSON.parse()；但若声明为text/plain等，不会自动解析。 |

6 jQuery中的AJAX封装函数之五——$.getScript

使用方法：

$.getScript(url, [data], [fn])

$.getScript(url)

含义作用：

创建XHR对象，向指定的URL发起异步GET请求，请求参数为data（k=v&k=v形式或{k:v, k:v}形式）追加在URL后，如果响应完成且成功，自动调用指定的回调函数fn。**该函数要求响应数据必须是application/javascript类型，会自动调用eval(xhr.responseText)加以执行**。

应用示例：实现一个“i18n国际化”的欢迎消息 17:35

实现步骤：

(1)编写PHP，i18n.php，获取所有的请求头部，读取Accept-Language头部，截取前两个字母——客户端浏览器的首选语言，根据此语言向客户端输出不同形式不同语言的欢迎消息

(2)编写HTML，i18n.html，点击按钮，异步请求服务器端返回的欢迎消息 $.getScript('xx.php')

|  |
| --- |
| **$.getScript()**不论响应消息是什么类型，都会自动调用eval把xhr.responseText进行执行；  **$.get()** 取决于响应消息声明的内容类型，若声明为application/javascript，会自动调用eval()；但若声明为text/plain等，不会自动执行。 |

总结：

$('选择器').load(url) —— 只接收html响应

$.get(url,data,fn)

$.post(url,data,fn)

$.getJSON(url, data, fn) —— 只接收json响应

$.getScript(url) —— 只接收script响应

上述五个方法共同的缺陷：只能处理成功的响应消息(status:200)

面试题：在jQuery中，如何处理AJAX的失败响应？

7. jQuery中的AJAX封装函数之六——$.ajax

提示：上述五个函数都是$.ajax()的简化版本，但功能上都没有$.ajax()强大！

使用方法：

$.ajax( {

type: 'GET',//POST/PUT/DELETE/HEAD

url: 'x.php',

data: 'k=v&k=v',//{k:v} 请求数据

beforeSend: function(){},//请求发送前的回调

success: function(){}, //成功响应的回调

error: function(){}, //失败响应的回调

complete: function(){} //响应完成的回调

} )

对应原生AJAX代码：

var xhr = new XMLHttpRequest();

xhr.onreadystatechange = function(){

if(xhr.readyState===4){

if(xhr.status===200){

success();

}else{

error();

}

complete();

}

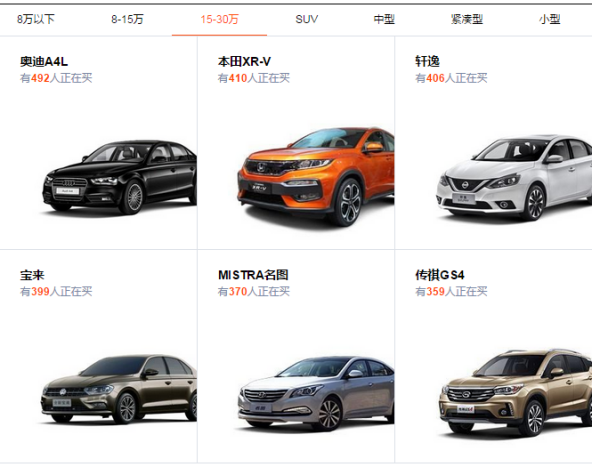
}

xhr.open();

beforeSend();

xhr.send();

课后练习：仿beijing.huimaiche.com中的车辆展示





必需的图片已经共享到FTP上

(1)编写SQL，创建huimaiche.sql，保存车辆数据（cid,cname,price,count,type, orderPrice, city）

(2)编写PHP，创建car\_list.php，根据客户端提交的车辆类别，返回该类别下所有汽车，以JSON格式

(3)编写HTML，创建car\_list.html，用户鼠标悬停在不同的车辆类别上，异步请求该类别下所有的汽车

(4)编写PHP，创建car\_select.php，根据客户端提交的cid，向客户端输出该车辆的所有信息，以JSON格式

(5)修改car\_list.html，当用户点击某个车辆的图片时，在下方显示出该车辆的全部信息

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月11\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

复习：

见思维导图

练习：

1)创建SQL，数据库名：autohome，表名：auto（aid, aname, thumbImg, fullImg, fullPrice, discountPrice, buyCount, type(lt8, lt15, lt30, suv)），插入12行记录，在phpMyAdmin中执行

2)创建PHP，auto\_select.php，接收客户端提交的汽车类别，返回该类别下的汽车列表（编号、名称、缩略图、购买数量）

3)创建HTML，auto\_select.html，有一个类别列表，随着鼠标位置的改变，异步请求该类别下的汽车数据

4) 创建PHP，auto\_getbyid.php，接收客户端提交的汽车编号，返回该汽车的详情，JSON：{ }

5)修改HTML，auto\_select.html，点击某个汽车的图片，异步请求该汽车的详情

今日目标：

(1)补充面试题：跨域请求 —— 掌握

(2)AJAX阶段项目

1.跨域请求和JSONP

Cross Domain Request：跨域名的HTTP请求，浏览器从某个域名下的资源访问了另一个域名下的另一资源，可能是协议、域名、端口号不同。

有HTML：

http://www.tmooc.cn/index.html

其中有代码：

<img src="http://www.baidu.com/logo.png">



浏览器哪些情形下允许跨域请求：

<img><link><script><iframe>

浏览器哪些情形下禁止跨域请求：

XHR —— 浏览器默认出于安全考虑会禁止XHR跨域请求

|  |
| --- |
| 面试题：  我们公司项目较大，把不同资源放在不同的服务器上，都有自己独立的域名  HTML:http://static.tedu.cn  PHP: http://data.tedu.cn  提问：如何从某个html页面中发起异步请求，访问PHP提供的数据？ |

解决方案——baidu

方案1：....

方案2：....

方案3：....

方案4：....

方案5：....

方案6：....

方案7：给被跨域访问的资源添加响应消息头部，设定允许来自于哪个域名下的页面访问当前页面：

header('Access-Control-Allow-Origin: http://xxxxx');

方案8：JSONPJSONP：JSON with Padding，填充式JSON， JSON是一种字符串数据格式，而JSONP与JSON完全不同，JSONP是使用JSON数据的一种巧妙的手段。

**思路：使用动态生成的<script>代替XHR，请求服务器端动态页面返回的填充了函数名的JSON数据**。

echo '**doResponse( {"ename":"Tom", "age":20} )**';

2.了解：jQuery中可以发起JSONP请求的方法

**(1)$.getJSON的两种用法：**

1) 使用XHR接收服务器端返回的JSON响应：

$.getJSON('x.php', function(obj){})

2) 使用script标签执行服务器返回的SCRIPT响应——JSONP：

$.getJSON('http://其他域名/x.php?callback=?', doResponse)

**(2)$.ajax的两种用法：**

1) 使用XHR接收服务器端返回的响应：

$.ajax( {} )

2) 使用script标签执行服务器返回的SCRIPT响应——JSONP：

$.ajax( {

type: 'GET',

url: 'x.php',

data: {k:v},

dataType: 'jsonp', //设定服务器端返回JSONP数据

success: fn

})

3.AJAX阶段项目 —— 京东购物车

项目目标：

(1)熟练掌握基于多表的数据库操作

(2)熟练掌握PHP访问数据库

(3)熟练掌握AJAX异步请求

所用技术：

SQL-MySQL、PHP、AJAX、JSON、jQuery

项目功能点：

(1)浏览商品

(2)异步分页显示

(3)添加到购物车

(4)浏览并修改购物车

实现步骤： SQL => PHP => HTML

(1)编写SQL，数据库名：jd，包含如下的表：

用户信息表： jd\_user( uid, uname, upwd )

插入两行记录： 1qiangdong123456

2naicha 456789

商品信息表： jd\_product( pid, pname, price, pic )

插入36行记录

购物车表： jd\_cart( cid, userId )

插入1行记录：100,1

购物车详情表： jd\_cart\_detail( did, cartId, productId, count)

插入3行记录：1, 100,10,3

2, 100,15,1

3, 100,18,2 16:50

(2)创建header.php，包含京东所有页面共有的头部内容；

创建footer.php，包含京东所有页面共有的尾部内容

(3)创建productlist.js，在productlist.html加以引入。编写JS，待页面加载完成后，异步请求页头和页尾内容，添加到div#header和div#footer

(4)创建login.php，接收客户端提交的用户名和密码，验证是否正确，向客户端输出ok或err

SELECT uid FROM jd\_user WHERE uname='?' AND upwd='?'

(5)编写JS，读取用户在输入框中的输入，异步提交给login.php，验证用户名和密码是否正确，错误则给用户提示；正确则关闭登录对话框

-------------------------------------------------------------------

(6)创建product\_select.php，向客户端输出所有的商品信息，以JSON格式

(7)修改productlist.js，页面加载完成后，异步请求商品信息，展示在商品列表中

(8)创建cart\_add.php，接收客户端提交的用户名和产品编号，把相关信息保存入需要的表——实现比较复杂，挑战一下！

SQL1：根据用户名查找用户编号

SQL2：根据用户编号查找购物车编号

SQL3：若用户编号没有对应的购物车编号，则执行添加语句生成购物车，得到购物车编号

SQL4：根据购物车编号和产品编号，到详情表查询是否有该记录

SQL5：若详情表中没有该商品记录，则执行插入，购物数量为1

SQL6：若详情表中已有该商品记录，则执行更新，购买数量+1

(9)修改productlist.js，为每个商品下的“添加到购物车”按钮添加单击事件监听，异步提交当前登录的用户名和当前商品的编号，保存购买信息

(10)在购物车详情页面中，查询出当前登录用户的购物车内容.......

INSERT INTO jd\_product VALUES

(1,'小米 Note 全网通 白色 移动联通电信4G手机 双卡双待',1199.00,'images/phone/phone\_01.jpg'),

(2,'Apple iPhone 6s (A1700) 16G 玫瑰金色 移动联通电信4G手机',3999.00,'images/phone/phone\_02.jpg'),

(3,'PPO R9 4GB+64GB内存版 玫瑰金 全网通4G手机 双卡',2499.00,'images/phone/phone\_03.jpg'),

(4,'小米 红米 3S 高配全网通 3GB内存 32GB ROM 经典金色',899.00,'images/phone/phone\_04.jpg'),

(5,'金立M6 Plus 香槟金 4GB+64GB版 移动联通电信4G手机 双卡双待',2999.00,'images/phone/phone\_05.jpg'),

(6,'Apple iPhone 6s Plus (A1699) 64G 玫瑰金色 移动联通电信4G手机',5799.00,'images/phone/phone\_06.jpg'),

(7,'vivo X7 全网通 4GB+64GB 移动联通电信4G手机 双卡双待',2499.00,'images/phone/phone\_07.png'),

(8,'小米 红米Note3 高配全网通版 3GB+32GB 金色 移动联通电信4G手机',1099.00,'images/phone/phone\_08.jpg'),

(9,'【六个月碎屏换新】荣耀8 4GB+32GB 全网通版 魅海蓝 双镜头，双2.5D玻璃',2499.00,'images/phone/phone\_09.jpg'),

(10,'荣耀7 (PLK-AL10) 3GB+64GB内存版 荣耀金 移动联通电信4G手机',1799.00,'images/phone/phone\_10.jpg'),

(11,'荣耀 V8 全网通 高配版 4GB+64GB 香槟金 移动联通电信4G手机 双卡双待双通',2799.00,'images/phone/phone\_11.jpg'),

(12,'荣耀 畅玩5X 3GB内存版 落日金 移动联通电信4G手机 双卡双待 炫酷指纹',1099.00,'images/phone/phone\_12.jpg'),

(13,'Apple iPhone 6 (A1586) 64GB 金色 移动联通电信4G手机',4199.00,'images/phone/phone\_13.jpg'),

(14,'TCL 初现 750 雅金 移动联通电信4G手机 双卡双待 后置1600万摄像，美姿拍照！',4199.00,'images/phone/phone\_14.jpg'),

(15,'华为 P9 plus 64GB 琥珀灰 移动联通电信4G手机 双卡双待',3988.00,'images/phone/phone\_15.jpg'),

(16,'Apple iPhone 5s (A1530) 16GB 金色 移动联通4G手机',2198.00,'images/phone/phone\_16.jpg'),

(17,'vivo X7Plus 全网通 4GB+64GB 移动联通电信4G手机 双卡双待 金色',2798.00,'images/phone/phone\_17.jpg'),

(18,'华为 畅享5S 金色 移动联通电信4G手机 双卡双待 10万好评手机！',1099.00,'images/phone/phone\_18.jpg'),

(19,'Apple iPhone 6 Plus (A1524) 16GB 银色 移动联通电信4G手机',3899.00,'images/phone/phone\_19.jpg'),

(20,'华为 麦芒5 全网通 4GB+64GB版 香槟金 移动联通电信4G手机 双卡双待',2599.00,'images/phone/phone\_20.jpg'),

(21,'小米5 全网通 标准版 3GB内存 32GB ROM 白色 移动联通电信4G手机',1799.00,'images/phone/phone\_21.jpg'),

(22,'华为 P9 全网通 3GB+32GB版 流光金 移动联通电信4G手机 双卡双待 麒麟955',3188.00,'s/phone/phone\_22.jpg'),

(23,'金立 金钢 标准版 爵士金 移动联通电信4G手机 双卡双待 4G全网通',999.00,'images/phone/phone\_23.jpg'),

(24,'360手机 N4 全网通 4GB+32GB 阳光白 移动联通电信4G手机 双卡双待',999.00,'es/phone/phone\_24.jpg'),

(25,'小米 Max 全网通 标准版 3GB内存 32GB ROM 金色 移动联通电信4G手机',1299.00,'es/phone/phone\_25.jpg'),

(26,'华为 P9 全网通 4GB+64GB版 金色 移动联通电信4G手机 双卡双待 后置1200万',3688.00,'es/phone/phone\_26.jpg'),

(27,'乐视（Le）乐2（X620）32GB 原力金 移动联通电信4G手机 双卡双待 5.5英寸',988.00,'images/phone/phone\_27.jpg'),

(28,'努比亚(nubia)【3+64GB】小牛5 Z11mini 黑色 移动联通电信4G手机',1299.00,'ges/phone/phone\_28.jpg'),

(29,'乐视（Le）乐2Pro 32GB 金色 移动联通电信4G手机 双卡双待 5.5英寸In-Cell屏',1399.00,'ges/phone/phone\_29.jpg'),

(30,'华为 Mate 8 3GB+32GB版 玫瑰金 移动联通电信4G手机 双卡双待 麒麟950芯片',2799.00,'ges/phone/phone\_30.jpg'),

(31,'小米 4c 标准版 全网通 白色 移动联通电信4G手机 双卡双待 高通骁龙808畅销机',799.00,'images/phone/phone\_31.jpg'),

(32,'vivo X7 全网通 4GB+64GB 移动联通电信4G手机 双卡双待 星空灰 vivox7',2498.00,'ages/phone/phone\_32.jpg'),

(33,'联想 乐檬3 （K32C36）16GB 金色 移动4G手机 双卡双待 刀锋致敬经典',599.00,'images/phone/phone\_33.jpg'),

(34,'华为 荣耀 畅玩4X 晨曦金 移动联通电信4G手机 双卡双待 5.5英寸大屏看片利器',749.00,'mages/phone/phone\_34.jpg'),

(35,'三星 Galaxy On5（G5500）金色 移动联通4G手机 真皮质感后盖，2600毫安大容量',699.00,'mages/phone/phone\_35.jpg'),

(36,'OPPO A37 2GB+16GB内存版 玫瑰金 全网通4G手机 双卡双待 【赠品任你选】',1299.00,'images/phone/phone\_36.jpg');

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月12\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.扩展：表和表之间的关系

(1)一对一关系

emp(eid,ename,computerId)

<=>

computer(cid,price,empId)

(2)一对多关系

dept(did,dname)

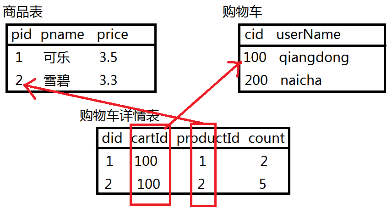
<=>

emp(eid,ename,deptId)

(3)多对多关系

product(pid,price)

cart(cid, userName)



2.AJAX阶段项目——京东购物车

实现步骤： SQL => PHP => HTML

(1)编写SQL，数据库名：jd，包含如下的表：

用户表： jd\_user( uid, uname, upwd )

商品表： jd\_product( pid, pname, price, pic )

购物车表： jd\_cart( cid, userId )

购物车详情表：jd\_cart\_detail( did, cartId, productId, count)

(2)创建header.php，包含京东所有页面共有的头部内容；

创建footer.php，包含京东所有页面共有的尾部内容

(3)创建productlist.js，在productlist.html加以引入。编写JS，待页面加载完成后，异步请求页头和页尾内容，添加到div#header和div#footer

(4)创建login.php，接收客户端提交的用户名和密码，验证是否正确，向客户端输出ok或err

(5)编写JS，读取用户在输入框中的输入，异步提交给login.php，验证用户名和密码是否正确，错误则给用户提示；正确则关闭登录对话框

(6)创建product\_select.php，向客户端输出所有的商品信息，以JSON格式

(7)修改productlist.js，页面加载完成后，异步请求商品信息，展示在商品列表中

(8)创建cart\_add.php，接收客户端提交的uname和pid，把相关信息保存入需要的表——实现比较复杂，挑战一下！

SQL1：根据用户名查找用户编号

SELECT uid FROM jd\_user WHERE uname='?'

SQL2：根据用户编号查找购物车编号

SELECT cid FROM jd\_cart WHERE userId='?'

SQL3：若用户编号没有对应的购物车编号，则执行添加语句生成购物车，得到购物车编号

INSERT INTO jd\_cart VALUES(NULL, '?')

SQL4：根据购物车编号和产品编号，到详情表查询是否有该记录

SELECT \* FROM jd\_cart\_detail WHERE cartId=? AND productId=?

SQL5：若详情表中没有该商品记录，则执行插入，购物数量为1

INSERT INTO jd\_cart\_detail VALUES(NULL,'?','?', 1)

SQL6：若详情表中已有该商品记录，则执行更新，购买数量+1

UPDATE jd\_cart\_detail SET count=count+1 WHERE cartId='?' AND productId='?'

PHP页面最终返回：{"msg": "ok","uid":1,"cid":100,"pid":10,"count":4}

(9)修改productlist.js，为每个商品下的“添加到购物车”按钮添加单击事件监听，异步提交当前登录的用户名和当前商品的编号，保存购买信息

(10)创建shopppingcart.js，在shoppingcart.html中引用；待shoppingcart.html页面加载完成，异步请求公用的页头和页尾

(11)创建cart\_detail.php，接收客户端提交的用户名，向客户端输出该用户的购物车详情

SELECT did,cartId,productId,count, pname,price,pic

FROM jd\_cart\_detail, jd\_product

WHERE cartId=100 AND productId=pid ;

(12)修改shoppingcart.js，待html加载完成，异步请求当前登录用户的购物车详情

--------------------------------------------------------

(13)创建cart\_update.php，接收客户端提交的购物车详情记录编号，以及最新的购买数量，更新到数据库，返回succ或err

UPDATE jd\_cart\_detail SET count=? WHERE did=?

(14)修改shoppingcart.js，为+和-按钮添加事件监听，修改购买数量并异步提交到服务器端

(15)创建cart\_detail\_delete.php，接收客户端提交的did，从购物车详情表中删除该记录，返回succ或err

DELETE FROM jd\_cart\_detail WHERE did=?

(16)修改shoppingcart.js，为产品后面的“删除”按钮添加事件监听，异步提交到服务器端删除该购买记录，服务器返回删除成功后从table中删除tr。

3.项目中的分页查询 —— 难点&重点

分页查询：项目中的数据可能有很多行，不可能一次性发送给客户端；一次只能发送其中的若干条——称为“1个页(page)”。

**服务器端分页查询中客户端的请求数据：**

product\_select.php?pageNum=3

**服务器端分页查询中服务器端的响应数据：**

{

recordCount: 36, //数据库中满足条件的总记录数

pageSize: 8, //页面大小，一页中最大记录数

pageCount: 5， //总页数

pageNum: 3, //当前是哪一页中的数据

data: [{},{}...{}] //当前页中的数据内容

}

服务器端如何返回上述数据：

(1)查询满足条件的总记录数

SELECT COUNT(\*) FROM jd\_product;

(2)如何计算总页数

ceil( recordCount / pageSize )

(3)如何获取当前要显示的页号

$\_REQUEST['pageNum']

(4)如何获取指定页中的数据——SQL分页查询

SELECT \* FROM jd\_productLIMIT start, count;

第1页： ... LIMIT 0, 8

第2页： ... LIMIT 8,8

第3页： ... LIMIT 16,8

第4页： ... LIMIT 24,8

第5页： ... LIMIT 32,8

....

第pageNum页： ... LIMIT (pageNum-1)\*pageSize,pageSize

(4)分页条最终呈现的外观

**1** 2 3

1 **2** 3 4

1 2 **3** 4 5

2 3 **4** 5

3 4 **5**

|  |
| --- |
| 补充：SQL语句中的分组函数：  SELECT MAX(price) FROM product;  SELECT MIN(price) FROM product;  SELECT SUM(price) FROM product;  SELECT COUNT(price) FROM product;  SELECT AVG(price) FROM product; |

4.如何在两个页面间传递数据

方法1：在页面跳转是追加查询字符串

页面1：location.href="2.html?loginName=xx"

---------------------------------------

页面2：location.search

方法2：使用Cookie技术

方法3：使用HTML5本地存储 —— 后面会学

方法4：使用服务器端Session技术 —— 超纲

5.扩展SQL知识——查询嵌套

在某个SELECT中继续嵌套另一个SELECT语句：

SELECT cid FROM jd\_cart WHERE userId=(

SELECT uid FROM jd\_user WHERE uname='xxx'

)

6. 扩展SQL知识——跨表查询

一个查询结果集中的数据需要来自于多个表：

SELECTempName, deptName FROM dept,emp;

上述语句的插件结果会对表1中的任一行记录与表2中的任一行记录进行匹配，可能性有"表1.size \* 表2.size"——笛卡尔积。必须剔除无效的数据

SELECTempName, deptName FROM dept,emp **WHERE emp.deptId = dept.did**;

CREATE TABLE dept(

did INT ,

deptName VARCHAR(32)

);

INSERT INTO dept VALUES(10, '研发部'),(20,'市场部');

CREATE TABLE emp(

eid INT ,

empName VARCHAR(32),

deptId INT

);

INSERT INTO emp VALUES(1, 'Tom', 10),(2,'Mary',20);

HTML5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月13日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第三阶段知识体系：

(1)AJAX异步请求

数据库、PHP、HTTP、原生AJAX、jQuery中的AJAX

(2)HTML5高级特性

九大新特性

(3)Boostrap框架

五部分

1.前端开发常用的工具软件

(1)轻量级(>3MB)： 小巧快速，但功能简单

Editplus、Notepad++、VIM

(2)中量级(xxMB)：功能相对丰富

SublimeText

(3)重量级(xxxMB)：启动速度慢，但功能丰富

IDE(Integrated Development Environment)

IDE = 编辑器(代码提示) + 调试器 + 文件管理+ 项目管理+ 文件转换程序

Dreamweaver

ApatanaStudio(基于Eclipse)

WebStorm(基于IntelJ IDEA)

常用快捷键：

(1)Ctrl+Alt+↓ 快速复制当前行

(2)Alt+↑/↓ 移动当前行

(3)Ctrl+D 删除当前行

(4)Ctrl+/ 快速（取消）注释当前行

(5)Ctrl+Alt+L 格式化当前文档

WebStorm的多行编辑模式：

开始多行编辑：按住Alt，点击鼠标左键

退出多行编辑：ESC或点击鼠标左键

今日目标：

1)HTML5高级新特性

|  |
| --- |
| 面试题：HTML5中添加了哪些新的标签？废弃了哪些标签？ |
| 面试题：HTML5中添加了哪些新的标签属性？废弃了哪些标签属性？ |

2.HTML5高级新特性——九大新特性

1)表单新特性(表单2.0)

2)视频和音频

3)Canvas绘图

4)SVG绘图

5)地理定位

6)拖放API

7)WebWorker

8)WebStorage

9)WebSocket

提示：上述特性彼此没有关联！学习过程中有些凌乱！

|  |
| --- |
| normalize.css：用于统一浏览器中标签的默认样式值——CSS Reset（重置） |

3.HTML5新特性之表单新特性——新的inputtype

<input type="?">

**已经学过的input type：**

text、password、radio、checkbox、button、reset、submit、image、 file、hidden

**HTML5新添的input type：**

(1)number：显示一个数字输入框

<input type="number" min="1" max="9" step="2"/>

(2)email：显示一个邮件地址输入框，提交时会进行格式验证，中间一个@符号，前后至少一个字符

<input type="email"/>

(3)url：显示一个URL地址输入框，提交时会进行格式验证，冒号前存在任意字符

<input type="url"/>

(4)tel：显示一个电话号码输入框，没有格式验证，仅在手机浏览器中弹出数字输入键盘

<input type="tel"/>

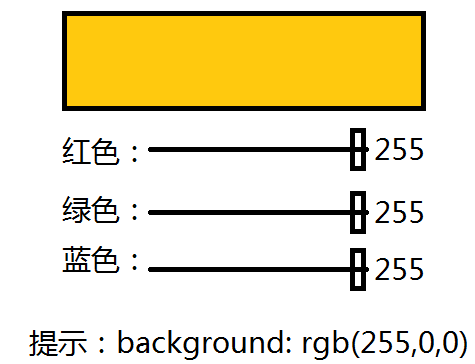
(5)search：显示搜索输入框，PC中Chrome下有X号；手机中弹出键盘右下角按钮显示文字“搜索”

<input type="search"/>

(6)range：显示一个范围选择滑块，让用户在特定范围内选择一个整数值

<input type="range" min="0" max="255" value="50" step="5"/>

练习：实现一个调色板，修改某个滑块的位置，可以改变背景颜色



(7)color：显示一个颜色选择窗口，浏览器会调用操作系统提供颜色选择窗口，提交的是颜色值

<input type="color">

(8)date：显示一个日期选择窗口，浏览器弹出一个日期选择窗口，不是DIV，不能定制样式

<input type="date" min="1949/10/01">

(9)month：显示一个月份选择器

<input type="month">

(10)week：显示一个“星期/周”选择器

<input type="week">

4.HTML5新特性之表单新特性——新的表单元素

**HTML5之前已有的表单元素——可与用户交互并提交：**

input、select/option、textarea、button

**HTML5中新增的表单元素——不能与用户交互，不能提交，仅用于信息展示：**

(1)datalist：数据列表，为某个输入框提供输入建议

<datalist id="dish">

<option>京酱肉丝</option>

<option>鱼香肉丝</option>

</datalist>

<input list="dish">

(2)progress：进度条，显示前进的进度提示条

<progress value=""></progress>

练习：使用定时器，修改progress的进度，实现不断前进的效果

(3)meter：度量衡，刻度尺，使用不同的颜色标示出数据所处的区间

<meter min="0" low="30" value="0" high="70" optimum="50" max="100"></meter>

最优值在上下限： 黄 - 绿 - 黄

最优值小于下限： 绿 - 黄 - 红

最优值大于上限： 红 - 黄 - 绿

(4)output：输出值，用于表示数据的计算结果，是一个语义标签

<output for="price count"></output>

5.HTML5新特性之表单新特性——表单元素新的属性——重点

**HTML5之前表单标签的常用属性：**

name、value、readonly、disabled、checked、for

**HTML5中表单标签的新属性：**

(1)placeholder：占位消息，为输入框中添加提示消息，注意：占位消息仅用作提示，与value属性根本不同，不能被提交

<input placeholder="提示性文字">

(2)autofocus：自动获得焦点，标识了该属性的输入框会自动的获得输入焦点

<input autofocus>

(3)multiple：多项，用于email和file输入框，允许其中输入多个值，用逗号分隔

<input type="email/file" multiple>

(4)autocomplete：自动完成，可取值**on**/off，指定输入域是否记录上一次提交的输入，下次输入时给予提示

<input autocomplete="off">

(5)form：表单，指定当前输入域所属的表单的ID，可以将声明了form的输入框放到表单元素的外面，仍然可以一同被提交

<form id="formID"></form>

<input form="formID">

=========与输入验证相关新属性========

(6)required：必填项，若没有输入则无法提交

<input required>提交时才会验证

(7)min：最小值，若输入值小于该值则无法提交

<input type="number/date" min="18">

(8)max：最大值，若输入值大于该值则无法提交

<input type="number/date" max="60">

(9)minlength：最小长度，若输入的字符串长度小于该值无法提交，此属性不是H5标准属性，FF不支持，Chrome支持

<input minlength="6" required>

(10)maxlength：最大长度，若输入的字符串长度大于该值无法提交

<input maxlength="9">

(11)pattern：正则表达式样式，若输入的字符串不符合指定的正则表达式则无法提交

<input pattern="^1[3578]\d{9}$">

|  |
| --- |
| 归纳总结练习——表单新特性：  (1)新的input type——10个  number、email、url、tel、search、range、color、date、month、week  (2)新的表单元素——4个  datalist、progress、meter、output  (3)表单元素新的属性——11个  placeholder、autofocus、autocomplete、multiple、form  ----------------------  required、min、max、minlength、maxlength、pattern |

提示：上述红色的特性可能在提交表单时弹出错误提示。

6.如何自定义表单错误提示消息

HTML5为每个表单输入域添加了一个属性：validity（有效性）

ValidityState {

valid : true

badInput : false 无效的输入，number

customError: false 自定义错误，setCustomValidity('')参数字符串有内容，curstomError就变为true；参数值是空字符串，customError变为false

patternMismatch : false 样式不匹配，pattern

rangeOverflow : false 范围上溢出，max

rangeUnderflow : false 范围下溢出，min

stepMismatch : false 步长不匹配，step

tooLong : false 字符串太长，maxlength

tooShort : false 字符串太短，minlength

typeMismatch : false 类型不匹配，email/url

valueMissing : false 值缺失，required

}

该对象中的属性值随着输入域中值的改变而立即改变，不会等到表单提交。其中只有valid为true时，表示输入域中的值是有效的，才能被提交；只要其它的某个属性值为true，valid属性值就会变为false。

注意：只要自定义了错误消息，浏览器默认的错误消息就不再显示。

晚间练习：

在输入域失去焦点时，使用其validity属性的各个布尔类型的值，验证用户的输入是否合法。下面的效果图可以放大。





\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月14日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

复习：

第三阶段课程

(1)AJAX：数据库、PHP、HTTP、AJAX、jQuery-AJAX

(2)HTML5：

表单新特性

视频和音频

Canvas绘图

SVG绘图

地理定位

拖放API

WebWorker

WebStorage

WebSocket

(3)Boostrap框架

表单新特性：

(1)新的input type——10个

number、email、url、tel、search、range、color、date、month、week

(2)新的表单元素——4个

datalist、progress、meter、output

(3)表单元素新的属性——11个——重点

placeholder、autofocus、autocomplete、multiple、form

--------------------------------------------

required、min、max、minlength、maxlength、pattern

修改错误提示消息的弹出时间、修改错误提示的内容

input.validity {

valid : true,

customError: false, //setCustomValidity('XX')

valueMissing: false,

rangeOverflow: false,

rangeUnderflow: false,

tooLong: false,

tooShort: false,

typeMismatch: false,

.....

}

作业回顾

今日目标：

(1)视频和音频 —— 小重点

(2)Canvas绘图—— 最重点

1.HTML5取代Flash体现在哪些方面

Flash-绘图、动画、游戏 —— HTML5- Canvas & SVG

Flash-视频、音频 —— HTML5 - Video & Audio

Flash-客户端存储 —— HTML5 -WebStorage

2.HTML5高级新特性——视频播放

<video src=""></video>在HTML5中专用于播放视频。

VIDEO标签默认是一个300\*150的inline-block；

Video标签/对象常用的成员：

|  |
| --- |
| 成员属性：  autoplay: false，是否自动播放  controls: false，是否显示播放控件  currentTime: 1.4901，当前播放到的时间  duration: 60.1012，总时长  defaultMuted: false,默认是否静音  loop: false，是否自动循环播放  muted: false，当前是否静音  paused: false，当前是否处于暂停状态  poster：''，仅在影片开始播放之前显示的“电影海报”  volumn：1, 当前播放音量  preload: 'auto'，如何预加载影片内容，可取值有三个：  auto：自动预加载影片元数据，并缓冲一定的长度  metadata：仅预加载影片元数据（宽高、时长）  none：不预加载影片的任何数据 |
| 成员方法：  play()让影片开始播放，paused属性变为false  pause()让影片开始暂停，paused属性变为true |
| 成员事件：  onplay：fn当调用了v.play()方法时触发  onpause: fn当调用了v.pause()方法时触发 |

练习：禁用Video的默认播放控件，自定义播放和暂停按钮，悬停在影片的正上方，点击则播放，再点击则暂停——仿优酷的效果

<div>

<video></video>

<a><img></a>

</div>

练习：不使用Video的海报(poster)属性，自定义一个广告图片，当影片播放时候，广告要隐藏；影片暂停时，显示广告。

v.onplay = function(){}

v.onpause = function(){}

3.HTML5高级新特性——音频播放

<audio src=""></audio>在HTML5中专用于播放声音。

AUDIO标签默认是一个300\*30的inline-block；若不显示播放控件则display为none。

AUDIO标签/对象常用的成员：

|  |
| --- |
| 成员属性：  autoplay: false，是否自动播放  controls: false，是否显示播放控件  currentTime: 1.4901，当前播放到的时间  duration: 60.1012，总时长  defaultMuted: false,默认是否静音  loop: false，是否自动循环播放  muted: false，当前是否静音  paused: false，当前是否处于暂停状态  volumn：1, 当前播放音量  preload: 'auto'，如何预加载音频内容，可取值有三个：  auto：自动预加载音频元数据，并缓冲一定的长度  metadata：仅预加载音频元数据（宽高、时长）  none：不预加载音频的任何数据 |
| 成员方法：  play()让音频开始播放，paused属性变为false  pause()让音频开始暂停，paused属性变为true |
| 成员事件：  onplay：fn当调用了v.play()方法时触发  onpause: fn当调用了v.pause()方法时触发 |

练习：根据复选框是否勾选，暂停/播放背景音乐 14:27



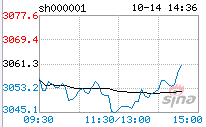
说明：

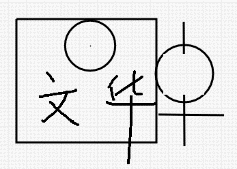
(1)早期的IE浏览器中，可以使用body的bgsound属性控制背景，但此属性不是HTML标准属性，且无法精确的控制播放和暂停：

<body bgsound="bg.mp3">

(2)iOS系统自带的Safari浏览器目前不支持AUDIO标签，只能使用隐藏的VIDEO标签来代替。

4.前端技术中可以实现绘图的技术





(1)WebGL技术 —— 当前还没有纳入HTML5标准

(2)Canvas技术 —— 在HTML5标准中提出的技术

(3)SVG技术 —— 早就存在的技术，纳入了HTML5标准

5.Canvas绘图技术 —— 比较难

提示：有两个难点

(1)属性/方法比较多，不好记忆

(2)小学数学计算(坐标轴，坐标点的位置计算)

Canvas：画布，在HTML5中默认是一个300\*150的inline-block。设定画布的宽和高不能使用CSS Style，只能使用width和height属性。

<canvas width="500" height="400">

您的浏览器不支持Canvas标签！

</canvas>

画布本身不能绘制内容，只用于承载被绘制的内容——使用画笔来往画布绘制内容。获得画笔使用原生JS：

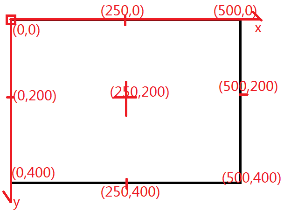
var ctx = canvas.getContext('2d'); //绘图上下文对象

画布上所有内容的绘制都要依靠“绘图上下文”对象。

Context对象常用的属性和方法：

|  |
| --- |
| 常用属性：  fillStyle:"#000000"填充样式  strokeStyle:"#000000"描边/轮廓样式  font:"10px sans-serif"  textAlign:"start"  textBaseline:"alphabetic"文本基线  globalAlpha:1全局不透明度  lineWidth:1 线/描边的宽度  shadowOffsetX:0 阴影在X轴上的偏移量  shadowOffsetY:0  shadowColor:"rgba(0, 0, 0, 0)"  shadowBlur:0 阴影模糊半径 |
| 常用方法：  arc()绘制一个拱形/圆形  beginPath()开始绘制一条路径——PS:钢笔工具  closePath()闭合一条路径——PS  fill()对路径进行填充  stroke() 对路径进行描边  moveTo() 移动到某一点  lineTo() 到另一个点绘制一条直线  fillRect()填充一个矩形  strokeRect()描边一个矩形  clearRect()清空一个矩形范围内所有内容  fillText() 填充一个字符串  strokeText()描边一个字符串  drawImage()绘制图像 |

Content Context



6.使用绘图上下文——绘制矩形(rectangle)

提示：矩形的定位点在自己的左上角

ctx.fillStyle = '#000'/渐变对象 填充样式

ctx.strokeStyle="#000"/渐变对象 描边样式

ctx.lineWidth = 2; 描边的宽度

ctx.fillRect(x, y, w, h) 填充一个矩形

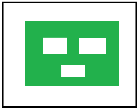
ctx.strokeRect(x,y,w,h) 描边一个矩形

ctx.clearRect(x,y,w,h) 清除一个矩形范围内的内容

|  |
| --- |
| 创建一个线性渐变对象：  var g = ctx.createLinearGradient(x1,y1, x2, y2);  g.addColorStop(offset1, color1);  g.addColorStop(offset2, color2); |

练习1：画布500\*400，左上角填充一个红色的矩形100\*80；右上角描边一个绿色的矩形100\*80；描边+填充一个矩形100\*80；右下角描边+填充另一个颜色的矩形100\*80；正中央描边+填充与右下角完全一样的矩形。

练习2：绘制如下的图形

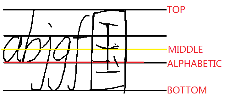


练习3：绘制一个可以在画布上左右晃动的矩形——最基础的动画模型

练习4：绘制一个上下晃动的矩形

练习5：练习一个斜向右下角30度晃动的矩形

7.使用绘图上下文——绘制文本(text)



提示：文本的定位点在整个字符串的文本基线的最左边

ctx.textBaseline = 'alphabetic';//文本基线，可取值为top/middle/alphabetic/bottom

ctx.font = "10px sans-serif";

ctx.fillText( txt, x, y )

ctx.strokeText( txt, x, y )

ctx.measureText( txt ).width //测量文本的宽度

课后练习：

(1)使用视频做元素的背景



提示：Video自动播放、循环播放、静音，绝对定位到目标元素下面，z-index为负值即可

(2)假设使用AJAX从服务器端获取到如下的数据：

[

{"label": "部门1", "value":300},

{"label": "部门2", "value":500},

{"label": "部门3", "value":150},

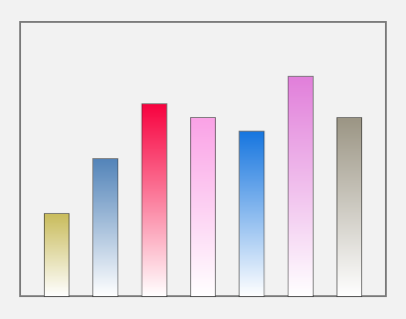
{"label": "部门4", "value":400},

{"label": "部门5", "value":600},

{"label": "部门6", "value":250}

]

根据这样的数据，绘制出如下图所示的统计图：



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月17日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

复习：

HTML5新特性

(1)表单新特性

新的input type 10

新的标签 4

标签的新属性 11

(2)视频和音频

videoaudio

(3)Canvas绘图

(4)SVG绘图

(5)地理定位

(6)拖放API

(7)WebWorker

(8)WebStorage

(9)WebSocket

使用Canvas绘图——JS绘图

<canvas id="c1" width="500" height="400"></canvas>

c1.width = 500;

c1.height = 400;

提示：Canvas的尺寸不能使用CSS来设置

var ctx = c1.getContext('2d');

常用属性：

ctx.fillStyle = '#f00'/gradient;

ctx.strokeStyle = '#f00'/gradient;

ctx.lineWidth = 1;

ctx.font = '10px sanse-sarfi';

ctx.textBaseline = 'alphabetic';

ctx.shadowBlur = '';

常用方法：

绘制矩形

ctx.fillRect()

ctx.strokeRect()

ctx.clearRect()

绘制文本

ctx.fillText()

ctx.strokeText()

ctx.measureText(txt).width

绘制路径

绘制图像

今日目标：

(1)绘制路径 —— 麻烦

(2)绘制图像 —— 抽象

(3)Chart.js —— 掌握

1.使用Canvas进行绘图——绘制路径

路径：Path，类似于Photoshop中的钢笔工具，可以绘制直线、各种曲线；但路径本身是不可见的，有三种用途：

(1)创建选区(clip)，对画布内容进行裁剪

(2)进行描边(stroke)，绘制任意形状的折线

(3)进行填充(fill)，填充出任意形状的图形

相关函数：

ctx.beginPath() 开始一条新的路径

ctx.arc() 绘制一条圆形/椭圆/拱形路径

ctx.moveTo(x,y) 移动到指定点

ctx.lineTo(x,y) 从上一点到指定点绘制直线

ctx.busierCurve() 绘制贝塞尔曲线

ctx.closePath() 闭合路径

---------------------------------

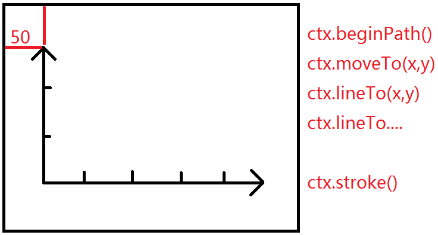
ctx.clip() 基于当前路径进行裁切

ctx.stroke() 基于当前路径进行描边

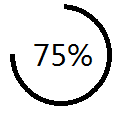
ctx.fill() 基于当前路径进行填充

练习：

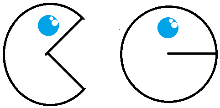
(1)使用路径绘制一个坐标轴



(2)使用定时器配合路径实现一个圆环状进度条



(3)创建两个函数，openMouth()和closeMouth()，分别绘制如下的两幅图形



2.使用Canvas绘图——绘制图像

提示：图像的定位点在自己的左上角

ctx.drawImage( img, x, y ) //原始大小绘制

ctx.drawImage( img, x, y, w, h ) //使用指定的宽高绘制图像——可能进行图像大小缩放

**注意：绘制图像时，必须等待图片异步加载完成**

var img = new Image();

img.src = "xx.png"; //向服务器异步请求图片

img.onload = function(){//图片加载完成

ctx.drawImage( img, x, y )

}

练习：

(1)在画布的左上角、右上角、左下角、右下角绘制小星星；正中央绘制一个2倍大小的星星

(2)在画布上随机绘制大大小小随机出现的20个星星

(3)在画布上绘制一个从左上角到右下角移动的小星星

(4)在画布上原地旋转的小星星

图像的旋转问题：

(1)Canvas中的旋转不是画布旋转！而是“绘图上下文（画笔）”旋转。

ctx.rotate(deg)

(2)旋转必须有一个轴点——默认是坐标轴的原点；若想以某个固定的点旋转，就必须平移画布的坐标原点：

ctx.translate(x, y) //指定新的坐标轴原点

练习(比较难)1：绘制四个小飞机，各在一个角落以自己为轴点原地旋转

提示：绘制每个小飞机都要平移一次坐标轴原点，平移后再恢复到原来的位置供下一次使用。练习(比较好玩)2：绘制一个可以随着鼠标移动的飞机

3.总结：Canvas绘图技术相关属性和方法——重点 17:18

ctx.fillStyle = 颜色/渐变色/样式;

ctx.strokeStyle =颜色/渐变色/样式;

ctx.font = "10px sanse-sarif";

ctx.textBaseline = '';

ctx.lineWidth = 1;

-------------------------------

(1)绘制矩形

ctx.fillRect()ctx.strokeRect()ctx.clearRect()

(2)绘制文本

ctx.fillText()ctx.strokeText()ctx.measureText()

(3)绘制路径

ctx.beginPath()

ctx.moveTo()ctx.lineTo()ctx.arc()ctx.rect()

ctx.stroke()ctx.fill()ctx.clip()

(4)绘制图像

ctx.drawImage(img,x,y,[w],[h])

ctx.rotate( deg )

ctx.translate( x, y )

4.基于Canvas的图表绘制框架/工具库

(1)Chart.js —— 免费开源，有九类图表可供使用

(2)Echart.js —— 免费的，百度提供，中文手册易上手

(3)FusionCharts.js —— 收费的，功能强大

5.第三方图表绘制工具——Chart.js的使用

(1)寻找官网：http://www.chartjs.org

(2)看官网介绍：

Simple yet flexible JavaScript charting for developers

(3)看官网提供的使用文档(API Document)

http://www.chartjs.org/docs/

(4)根据官网上的实例编写代码：

<canvas id="c2" width="600" height="400"></canvas>

<script src="js/Chart.js"></script>

<script>

new Chart(c2, {

type: 'bar', //图表类型

data: { }, //图表数据

options: { } //可选参数

});

</script>

课后练习：

1. 使用Canvas绘制一个随机改变的验证码图片——注意：真正项目中验证码图片都是由后台程序，如PHP生成的，此例子仅仅是为了练习Canvas的使用。



var str = 'ABCDEFGHJKLMNPQRSTWXY3456789';

var char = str[ 0~字符串长度间的随机数 ];

要求：

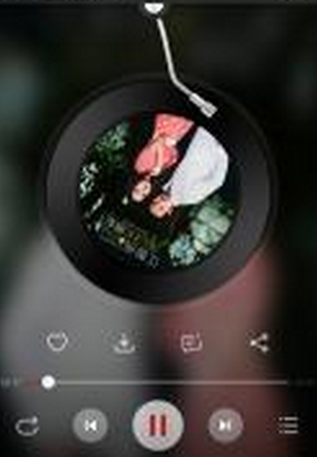
画布背景颜色随机(浅色)ctx.fillRect()

文字内容随机、大小随机，颜色随机(深色)、旋转角度随机。

5条随机干扰线（每个颜色随机深色），处于文字上方。

100个杂色点（半径为1为圆），处于文字上方。

2.使用Canvas仿写网易云音乐的播放界面



点击播放按钮，碟片开始旋转，背景音乐开始播放；

再次点击播放按钮，碟片停止旋转，背景音乐停止播放。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月18日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

复习：

HTML5新特性

(1)表单新特性 —— 重点

新的input type、新标签、标签的新属性

(2)视频和音频 —— 重点

video、audio

(3)Canvas绘图 —— 最重点

绘制矩形、文本、路径(裁剪/描边/填充)、图像

(4)SVG绘图 —— 最重点

(5)地理定位

(6)拖放API —— 次重点

(7)WebWorker —— 次重点

(8)WebStorage —— 重点

(9)WebSocket —— 次重点

Canvas绘图使用方法：

<canvas id="c2" width="" height=""></canvas>

var ctx = c2.getContext('2d');

ctx.fillStyle = 颜色/渐变/样式;

ctx.strokeStyle =颜色/渐变/样式;

ctx.lineWidth = 1;

ctx.font = '10px sans-serif';

ctx.textBaseline = 'alphabetic';

ctx.showdowXxx =

---------------------------

ctx.fillRect()ctx.strokeRect()ctx.clearRect()

ctx.fillText()ctx.strokeText()ctx.measureText()

ctx.beginPath()

ctx.moveTo() ctx.lineTo() ctx.arc()

ctx.closePath

ctx.clip()ctx.fill()ctx.stroke()

ctx.drawImage()

ctx.rotate()ctx.translate()

1.补充知识点：

若Canvas绘图中需要绘制多张图片，必须等待所有图片加载完成才能开始绘制；而每张图片的加载都是异步请求，彼此没有先后顺序，哪一张先加载完成完全无法预测。

var progress = 0;//全局加载进度

var img = new Image();

img.src = 'xx.jpg';

img.onload = function(){

progress += 10;//该图片权重

if(progress===100){

startDraw();

}

}

2.补充知识点：如何为Canvas中的某个图形/图像添加事件监听

HTML中，只有标签/元素才能添加事件监听。

Canvas绘图技术，只有一个标签——Canvas。

为某个部分中的图形/图像添加事件监听，只能委托给Canvas，获取事件发生的坐标，是否处于目标图像/图形范围内。

结论：为Canvas的某个图形/图像添加事件监听非常麻烦！

今日目标：

(1)SVG绘图 —— 重点

(2)Two.js第三方绘图工具库

1.SVG绘图概述

**矢量图：** Flash/AI，每个线条/色块有独立的颜色，可以无限缩放，不善于描述颜色细节。—— SVG

**位图：** PS每个点都是一个独立的颜色，善于表现丰富的颜色细节，放大后会失真。—— Canvas

SVG：Scalable Vector Graphics，可缩放的矢量图

SVG和Canvas的比较

|  |  |
| --- | --- |
| **SVG** | **Canvas** |
| 绘制的是矢量图 | 绘制的是位图 |
| 每个图形/图像都是独立的标签 | 只有画布是标签 |
| 容易为每个图形添加事件监听 | 不能为每个图形添加独立的事件监听 |
| 可以无限缩放，适合地图 | 不能无限放大，适合表现颜色细节，例如游戏 |

2.SVG的使用方法

SVG标准诞生于2001年，远早于HTML5，后来纳入H5标准，去除了一些原有的特性。

SVG图形的使用方法

**(1)HTML5之前：**

把SVG标签写在独立的XML文档中；

在HTML文档中使用IMG、IFRAME、EMBED、OBJECT等标签引入svg文件

**(2)HTML5之后(SVG标签直接纳入H5标签库)：**

可以直接在HTML文档中使用SVG标签——HTML解释器已经可以直接解析SVG标签了。

3.使用SVG绘图——绘制矩形

注意：SVG图形的属性可以使用HTML标签属性来声明，也可以使用类似CSS的形式来声明——SVG标签专用样式；但这两种方式本质不一样。

<rect></rect>

可用属性：

width 矩形的宽

height 矩形的高

x 定位点的X坐标

y 定位点的Y坐标

fill 填充颜色

fill-opacity 填充颜色透明度

stroke 描边颜色

stroke-width 描边宽度

**提示：上述属性不属于HTMLDOM范畴，不能直接rect.x读写。但可以使用核心DOM的setAttribute()方法来操作。**

练习：

(1)点击一个矩形，它就隐藏

(2)一个矩形，从左向右移动

(3)随机生成20个大大小小的矩形

结论：使用JS创建新SVG元素有两种方法：

1)svg.innerHTML = '<rect></rect>';

2)document.createElementNS('http://www.w3.org/2000/svg', 'rect');

4.使用SVG绘图——绘制圆形

<circle></circle>

可用属性：

r 半径

cx 圆心的X坐标

cy 圆心的Y坐标

fill 填充颜色，默认为#000

fill-opacity 填充颜色透明度

stroke 描边颜色，默认为transparent

stroke-width 描边宽度

练习：

(1)使用HTML在SVG四个角各创建一个圆形

(2)使用JS创建20个随机大小、位置、颜色随机的圆形，点击某一个变大变淡，直至消失，从DOM上删除它

(3)点击SVG的某处，生成一个圆形，变大变大直至消失

5.使用SVG绘图——绘制椭圆

<ellipse></ellipse>

可用属性：

rx 横向半径

ry 纵向半径

cx 圆心的X坐标

cy 圆心的Y坐标

fill 填充颜色，默认为#000

fill-opacity 填充颜色透明度

stroke 描边颜色，默认为transparent

stroke-width 描边宽度

6.使用SVG绘图——绘制直线

<line></line>

可用属性：

x1 起点的横坐标

y1 起点的纵坐标

x2 终点的横坐标

y2 终点的纵坐标

stroke 描边颜色，默认为transparent，必须进行修改

stroke-width 描边宽度

练习：使用line元素实现下述小图标



提示：可以使用“元素组”<g></g>把多个元素包含起来，这样每个组员可以共享<g>元素声明的属性。 16:21

7.使用SVG绘图——绘制折线

<polyline></polyline>

可用属性：

points 折线上的点，值形如："0,010,050,100..."

fill 填充颜色，默认为#000

fill-opacity 填充颜色透明度，必须设置为0，否则会自动填充

stroke 描边颜色，默认为transparent，必须设置

stroke-width 描边宽度

练习：根据下列数据，绘制折线图，鼠标悬停时，线条的颜色改变

var data = [ [ 260, 180, 280, 150, 270 ], [130,180,220,210, 260] ];



8.使用SVG绘图——绘制多边形

<polygon></polygon>

可用属性：

points 多边形上的点，值形如："0,010,050,100..."

fill 填充颜色，默认为#000

fill-opacity 填充颜色透明度

stroke 描边颜色，默认为transparent

stroke-width 描边宽度

练习：使用polygon绘制下述图标



提示：真正项目中，如果需要这样的SVG图标，可以使用AI进行手绘，再使用插件导出为svg文件即可。

9.使用SVG绘图——绘制文本

<text>文本内容</text>

可用属性：

x

y

font-size

font-family

fill 填充颜色，默认为#000

fill-opacity 填充颜色透明度

stroke 描边颜色，默认为transparent

stroke-width 描边宽度

10.使用SVG绘图——绘制图像

提示：若绘制了位图在SVG上，这个SVG图放大也会产生失真。

<image></image>

可用属性：

x

y

xlink:href 指定图片的URL

width 图片的宽，默认为0，不显示

height 图片的高，默认为0，不显示

|  |
| --- |
| 矩形、圆形、椭圆、直线、折线、多边形、文本、图像 |

11.在绘制图形时使用渐变色

渐变对象是SVG中的特效对象——特效对象都必须定义在<defs></defs>标签内：

<svg id="s1" width="500" height="400">

<defs> <!--定义特效元素-->

<linearGradient id="rainbow" x1="0" y1="0" x2="100%" y2="0">

<stop offset="0" stop-color="red"></stop>

<stop offset="1" stop-color="purple"></stop>

</linearGradient>

</defs>

<rect x="50" y="100" width="400" height="200" fill="url(#rainbow)"></rect>

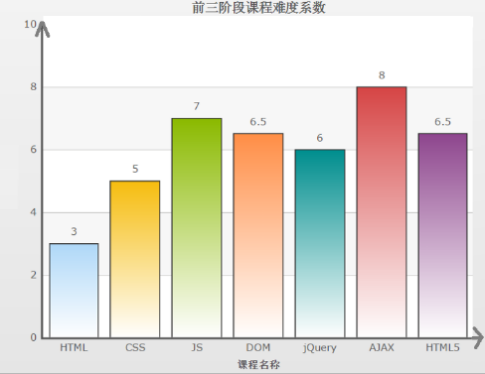
</svg>

课后练习：

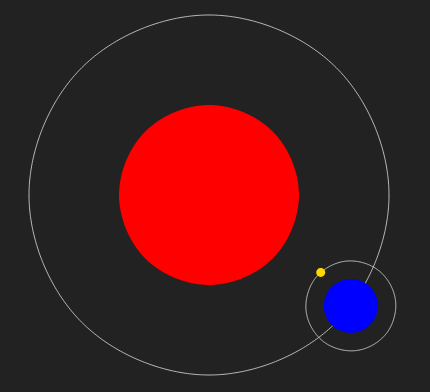
1. 用折线绘制五角星
2. 假设前端页面获取到后台返回了如下的JSON数据：

'[{"label":"HTML",value:3}, {"label":"CSS",value:5},....]'

根据这段JSON字符串，绘制出如下的统计图，要求每个柱在鼠标悬停时都会加粗显示边框：



3)提前自学 Two.js工具的使用，理解其作用，仿写官方示例代码，学会使用方法，实现如下效果——月亮绕着地球转，地球绕着太阳转：



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月19日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

复习：

HTML5新特性

(1)表单新特性

(2)视频和音频

(3)Canvas绘图

(4)SVG绘图

(5)地理定位

(6)拖放API

(7)WebWorker

(8)WebStorage

(9)WebSocket

SVG绘图：

方法1：1.svg <img src="1.svg">

方法2：<body><svg></svg></body>

绘制矩形：

<rect x="" y="" width="" height=""></rect>

绘制圆形：

<circle cx="" cy="" r=""></circle>

绘制椭圆：

<ellipse cx="" cy="" rx="" ry=""></ellipse>

绘制直线：

<line x1="" y1="" x2="" y2=""></line>

绘制折线：

<polyline points="x1,y1 x2,y2 ..."></polyline>

绘制多边形：

<polygon points="x1,y1 x2,y2 ..."></polygon>

绘制文本：

<text x="" y="" font-size="">XXXX</text>

绘制图像：

<image xlink:href=""></image>

使用渐变：

<svg>

<defs>

<linearGradient id="g1" x1="" y1="" x2="" y2="">

<stop offset="0" stop-color="" stop-opacity="">

</linearGradient>

</defs>

<rect fill="url(#g1)" stroke="url(#g1)"></rect>

</svg>

今日目标：

(1)补充：SVG中使用滤镜

(2)Two.js第三方工具

(3)地理定位

(4)拖放API —— 比较重要

1.SVG补充知识点

在SVG中使用滤镜(filter)——对图像图像进行像素化处理。使用方法：

<defs>

<filter id="f1">

<feGaussianBlur stdDeviation="5">

</feGaussianBlur>

</filter>

</defs>

<rect filter="url(#f1)"></rect>

2.三种绘图技术

(1)WebGL —— 3D渲染

(2)Canvas —— 2D动画游戏

(3)SVG —— 矢量图无限缩放

上述三个技术都可以用于绘制统计图——线、柱、圆...。

各自的使用方法不同，增加学习和使用难度。

two.js 2D绘图函数库

three.js 3D绘图函数库

3.第三方绘图工具库——Two.js

**官网：**https://two.js.org/

**官方介绍：**

Two.js is a two-dimensional drawing api geared towards modern web browsers. It is renderer agnostic enabling the same API to draw in multiple contexts: svg, canvas, and webgl.

一个2D绘图函数库，可以使用完全一样的API实现在不同的环境(svg/canvas/webgl)下绘制图形

**使用方法：**

<div id="container"></div>

<script src="js/two.js"></script>

<script>

var two = new Two({}).appendTo(container);

two.makeCircle(.....);

two.makeRectangle(....);

//two.update();

//two.play();

</script>

4.HTML5新特性——地理定位

window.navigator.geolocation：获取当前浏览器所在的地理位置，如

经度 longitude

纬度 latitude

海拔 altitude

速度 speed

浏览器如何定位：

(1)手机浏览器：使用内置GPS模块或通过信号基站

(2)PC浏览器：使用IP地址反向解析出对应的地址

浏览器地理定位涉及到个人隐私，所以浏览器都会询问用户，是否给予某个网页读取的权限。

navigator.geolocation {

getCurrentPosition: fn, //一次性的获取定位信息

watchPosition: fn, //周期性的监视定位信息

clearWatch: fn//清除定位监视器

}

使用方法：

navigator.geolocation.getCurrentPosition(

function(pos){ //获取成功

console.log(pos.coords.longtude); 经度

console.log(pos.coords.latitude); 纬度

console.log(pos.coords.altitue); 海拔

console.log(pos.coords.speed); 速度

},

function(err){ //获取失败

console.log(err.code);

console.log(err.message);

}

);

5.扩展知识点：如何在网页中使用百度地图

(1)注册一个百度开发者账号

http://lbsyun.baidu.com/

(2)登录百度账号

(3)查看使用JS调用百度地图说明文档：

http://lbsyun.baidu.com/index.php?title=jspopular

(4)申请使用密钥



(5)创建HTML页面，调用百度地图API

查看使用手册——非常通俗易懂。 16:10



6.HTML5新特性——拖放API——今日重点

Drag & Drop：HTML5为拖放事件定义了7个事件

**被拖动的对象——源对象(source)可以触发的事件：**

(1)ondragstart 拖动开始

(2)ondrag 拖动中

(3)ondragend 拖动结束

**可以拖动着进入上方并松手的对象——目标对象(target)可以触发的事件：**

(1)ondragenter 拖动着进入上方

(2)ondragover 拖动着在上方悬停

(3)ondrop 松开

(4)ondragleave 拖动着离开

提示：ondragover事件后续的默认行为是ondragleave，即ondragover后默认必然触发ondragleave，必须阻止浏览器此事件默认行为！！

练习：

(1)创建一个可以随着拖动而移动的飞机图片

提示：IMG必须定位！拖动过程中读取事件发生的坐标，及时修改IMG的left和top。

(2)垃圾箱默认是半透明，拖动小飞机到垃圾箱上方时浏览器完全不透明；拖动着离开浏览器继续变为半透明；

若飞机在垃圾箱上方释放，则从DOM树上删除飞机；垃圾箱继续变为半透明

(3)把练习1 和 练习2 组合在一起

课后练习： 英雄选择



要求：

(1)拖动某架飞机到目标区域(随鼠标移动)，则在上方显示出该飞机；下方没有改飞机了；

(2)拖动另一架飞机到目标区域，则显示出该飞机，之前选中的飞机重回待选区域；

(3)从目标区域将某架飞机拖走，则该飞机重回待选区域；目标区域变回？飞机。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月20日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

复习：

HTML5新特性

(1)表单新特性

(2)视频和音频

(3)Canvas绘图

(4)SVG绘图

(5)地理定位

navigator.geolocation.getCurrentPosition

(6)拖放API——7个事件

源对象：

ondragstart

ondrag

ondragend

目标对象：

ondragenter

ondragover - e.preventDefault()

ondragleave

ondrop

(7)WebWorker

(8)WebStorage

(9)WebSocket

今日目标：

(1)补充拖放API：dataTransfer

(2)扩展拖放API应用 —— 掌握

(3)WebWorker

(4)WebStorage —— 重点

1.拖放API补充知识点

如何在源对象的事件和目标对象的事件间传递数据？

**方法1：创建一个全局变量——**污染全局对象

var 全局变量= null;

src.ondragstart = function(){

全局变量= 数据值;

}

target.ondrop = function(){

console.log(全局变量);

}

**方法2：使用H5拖放API提供的dataTransfer对象**

提示：源对象事件的dataTransfer与目标对象事件的dataTransfer不是同一个对象！但二者之间可以传递数据。

src.ondragstart = function(e){

//利用数据传递对象保存数据

e.dataTransfer.setData('key', 'value');

}

target.ondrop = function(e){

//读取数据传递对象中的得到的数据

var data = e.dataTransfer.getData('key');

}

|  |
| --- |
| 提示：H5标准中dataTransfer对象可以保存的数据必须有key，key只能叫'text/html'、' text/uri-list'、'Files'三者之一。但经过实测，该key名称可以为任意名称。 |

2.扩展知识点：如何拖放客户端本地的图片到网页中显示

一般来讲，网页中的图片都应该来自于服务器；一般情况下，网页中不能使用客户端的图片。

可以使用拖放的方式把客户端的图片在网页中显示出来：

客户端的图片文件是拖放事件的源对象；

网页中的元素是拖放事件的目标对象；

container.ondrop = function(e){

var f0 =e.dataTransfer.files[0]; //File对象

var fr = new FileReader(); //读取文件中的内容

fr.readAsDataURL(f0); //把图片作为dataURL来读取

fr.onload = function(){ //文件读取完成

var img = new Image(); //创建IMAGE元素

img.src = fr.result; //dataURL

container.appendChild(img);

}

}

|  |
| --- |
| HTML5新增的文件读取相关的对象：  File 代表一个文件  FileList 代表一个文件列表  FileReader 用于读取一个文件中的内容  FileWriter 用于向一个文件中写出内容 |

3.面试题：请描述一下浏览器中的线程模型。

Program：程序，编写好的代码经过处理，可以在计算机上执行，放在文件系统(磁盘)上

Process：进程/任务，程序从磁盘中调入内存，分配必须可执行代码空间、数据空间，随时准备被CPU执行

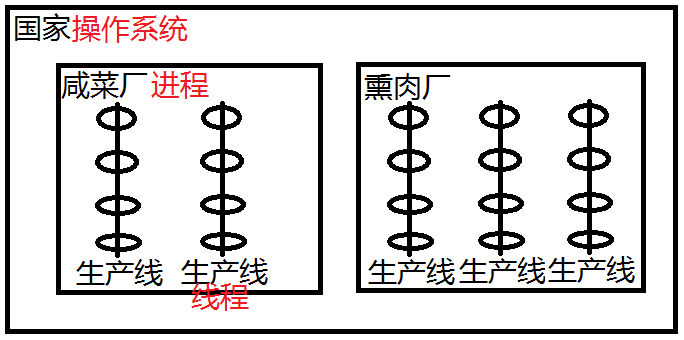
Thread：线程，是进程内部执行代码的基本单位

**进程和线程的关系：**

进程是操作系统分配内存基本单位；

线程是执行代码(JS/HTML/CSS)的基本单位；

线程处于进程内部，一个进程内部必须至少有一个线程，也可以有多个线程，这些线程间彼此可以没有影响——并发执行（宏观上看是同时执行，微观上看是在CPU上交替执行）



Chrome.exe进程中至少有6个线程，可以并发的向Web服务器发起请求（有的请求HTML、有的请求CSS、有的请求JS）。但是最终负责内容渲染(HTML解析/JS执行)的只有1个线程——UI主线程。

4.HTML5新特性——Web Worker

由于浏览器中负责渲染/监听只有一个UI主线程，所有的HTML/CSS/JS的执行都在这一个线程内，若页面中加载了非常耗时（算法复杂）的JS操作，会阻塞后续的HTML/CSS/JS的渲染和事件监听。

如何解决？—— 创建一个新的并发线程来执行这样的耗时操作。

JS中没有创建新线程的方法，替代方案——Web Worker

var w = new Work( 'xx.js' );

含义：在当前UI主线程中创建并启动一个新的并发的工作线程，该线程执行耗时操作，可能阻塞；但不会对当前UI主线程产生影响。

注意：Worker线程的致命问题——不能执行任何DOM操作，不能使用任何DOM&BOM元素——浏览器中只允许UI主线程修改DOM树。jQuery之类的JS文件决不能使用Worker来执行！！

**(1)如何让UI主线程给Worker线程传递数据：**

UI主线程：

var w = new Work('xx.js');

w.postMessage('data');

Worker线程：

onmessage = function(event){

var data = event.data;

}

**(2)如何让Worker线程把运算结果传递给UI主线程：**

Worker线程：

postMessage('data');

UI主线程：

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

w.onmessage = function(event){

var data = event.data;

}

练习：UI主线程中读取用户输入，发送给Worker线程；Worker线程开始运算，完毕后，把运算结果发送回给UI主线程，显示在DIV元素中。

**总结：Worker用于执行耗时的JS任务，在一个独立的线程中，可以避免UI主线程的阻塞问题。**

5.HTML5新特性——WebStorage——重点

项目中的数据如何保存？

**方式1：保存在服务器端——商品信息、用户信息、帖子**

**方式2：保存在客户端——浏览记录、登录信息、内容定制**

Web的客户端端存储技术有哪些？

(1)Cookie

优势：兼容性好

不足：操作繁琐，数据长度有限制（如4KB）

(2)Flash

优势：大小没有限制

不足：依赖于Flash允许环境

(3)Web Storage

优势：大小可达到8MB，操作简单

不足：HTML5新特性

(4)IndexedDB —— 课下自学

优势：大小没有限制，使用JS操作的一种客户端的数据库

不足：操作稍显复杂

HTML5中的Web Storage技术涉及两个新对象：

window.sessionStorage——会话级存储，其中的数据可以在一次会话中的多个页面中共享——数据存储在浏览器进程内存中。

sessionStorage.setItem(key, value)

var value = sessionStorage.getItem(key)

sessionStorage.removeItem(key)

sessionStorage.clear()

sessionStorage.key(i)

sessionStorage.length

window.localStorage——跨会话级存储/本地存储，其中的数据即使关闭浏览器/电脑，下次仍然可以访问——数据存储在文件系统的磁盘文件中。

localStorage.setItem(key, value)

var value = localStorage.getItem(key)

localStorage.removeItem(key)

localStorage.clear()

localStorage.key(i)

localStorage.length

注意：如果localStorage中的数据发生了改变，所有已打开的当前网站的浏览器窗口，都会自动触发window.onstorage事件，从而可以得到本地存储的数据已被修改。

|  |
| --- |
| Session：一个会话，指客户端连接到服务器后，在一段时间内的先后发起的多个请求，即一个会话中可能包含多个页面内容。只要浏览器不关闭，此次会话就一直存在；反之浏览器一关闭，会话就结束了。 |

练习：

(1)创建productlist.html页面，显示登录输入框，登录成功后可以点击“查看购物车”跳转到shoppingcart.html，此页面中显示出“欢迎回来：xxx”。还有一个超链接“退出登录”，删除登录的用户名。

(2)创建index.html页面，包含一个下拉列表，让用户选择自己喜欢网页风格，如：蔚蓝天空、芭比公主、杀马特等；接下来跳转到usercenter.html，该页面也呈现上一页面所选的风格；即使重启浏览器，再访问这些页面仍然是用户之前所选的风格。

提示：在客户端存储选中的主题的className

课下任务：

(1)挑战性任务：不使用传统的表单，使用AJAX+拖放API实现文件上传。

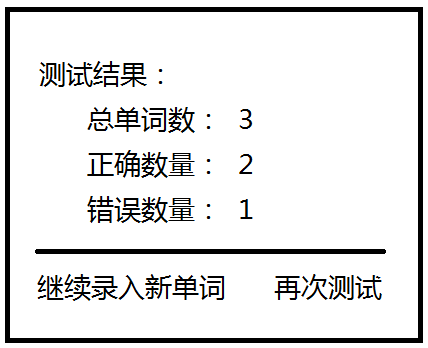
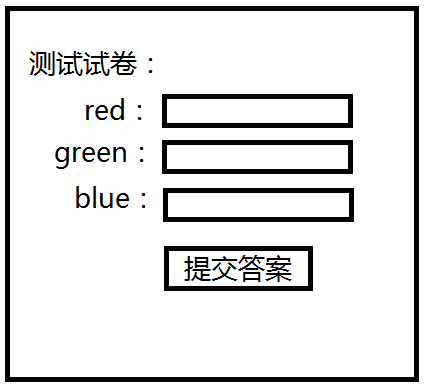
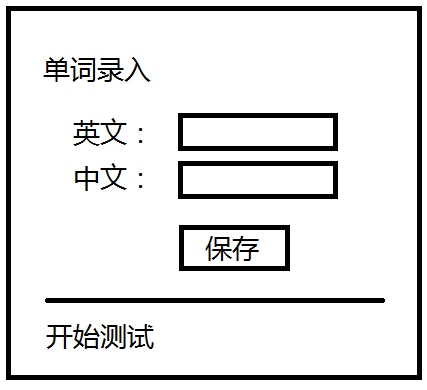
课后练习： 单词测试系统

(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

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月22日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.HTML5阶段项目

所用技术：

MySQL + PHP + AJAX + HTML5-Form + HTML5-Video + HTML5-Canvas + HTML5-SVG + HTML5-WebStorage

实现功能：

(1)用户注册

(2)用户登录

(3)浏览商品列表

(4)查看商品详情

(5)添加到购物车

(6)修改购物车

(7)确定下单 难点

(8)用户中心 - 我的订单 难点

(9)用户中心 - 消费统计

(10)用户中心 - 幸运抽奖

实现步骤：

(1)修改SQL，为jd\_user表添加新的字段：

uid,uname,upwd,email,homepage,age,birthday(bigint)

(2)创建PHP，user\_add.php，接收客户端提交的注册信息，保存入数据库，返回 {"msg": "succ", "uid": 3} 或 {"msg":"err", "sql":"INSERT...."}

(3)创建HTML，register.html（需要编写register.css），使用视频做背景，所有的输入域都要有输入验证，点击“注册”，使用AJAX异步提交注册信息，成功则跳转到产品列表页面（productlist.html）

(4)修改SQL，添加jd\_order（订单信息表），包含字段：oid, orderNum, shopName, rcvName, price, payment(付款方式，INT), orderTime(下单时间,BIGINT), status(订单状态,INT), userId(下单用户的ID号)

订单表 <=多对多=> 产品表

(5)修改SQL，添加jd\_order\_detail（订单详情表），包含字段： did, orderId, productId, count(购买数量)

(6)创建PHP，order\_add.php，接收客户端提交的订单信息，添加到订单表和订单详情表，返回{"msg":"succ","oid",12, "orderNum": 1234567890 } 或 {"msg":"err", "sql":"INSERT ...."}

SQL1: INSERT INTO jd\_order VALUES... 获得oid

SQL2: INSERT INTO jd\_order\_detail VALUES... 循环执行

(7)修改之前的productlist.html和shoppingcart.html，把登录后的用户名保存在sessionStorage

(8)创建HTML，addorder.html，根据当前购物车中的内容，生成订单，点击“下单购买”异步提交给服务器，进行订单生成，成功后在一个新的页面(addorder\_result.html)中显示出订单编号——sessionStorage

(9)创建HTML，usercenter.html，页头和页尾异步加载，主体内容分为左侧的“附加导航”和“右侧主体”形式的布局——使用CSS实现该布局

(10)创建JS，usercenter.js，随着附加导航的切换，右侧主体的内容会随之改变——类似“标签页签”效果

(11)创建PHP，order\_select.php，接收客户端提交的用户名，查询出该用户所有的订单，以JSON格式返回给客户端——难点

---------------------------------------------------------

(12)修改HTML，usercenter.html，页面加载完成后，异步请求当前登录用户的订单信息，追加在TABLE中

(13)选做：实现“我的订单”分页显示

(14)创建PHP，buy\_stat.php(消费统计)，根据客户端提交的用户名，模拟创建过去12个月中的消费总金额（无需访问数据库，直接伪造出JSON数据即可），返回数据形如：[{"label":"10月","value":3500}, {"label":"11月","value":3000}...]

(15)修改JS，usercenter.js，待页面加载完成后，异步请求当前登录用户的消费统计信息，根据这些信息绘制Canvas统计图——注意：所有图形都要使用动态变量创建

(16)修改JS，usercenter.js，待页面加载完成后，异步请求当前登录用户的消费统计信息，根据这些信息绘制SVG统计图——注意：所有图形都要使用动态变量创建

2.扩展JSON字符串在项目中的应用

**PHP服务器给客户端JS发送JSON响应数据：**

PHP： $arr = [ ... ]; echo json\_encode($arr);

JS： var obj = JSON.parse( str )

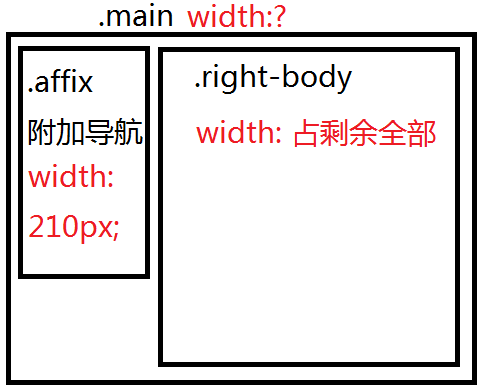
**客户端JS给发送PHP服务器JSON响应数据：**

JS： var arr = [ ... ]; var str = JSON.stringify(arr);

PHP： $arr = json\_decode( $\_REQUEST['str']);

提示：JSON中的{}会被PHP解析为Object,使用->符号读取对象的成员

3.面试题：使用纯CSS实现如下的布局



.main { }

.left {

width: 210px;

float: left;

}

.right-body {

margin-left: 210px;

}

3.如何根据订单编号查询出其对应的产品信息



需要的数据都在产品表中，不是跨表查询；

SELECT pid,pname,pic FROM jd\_product

WHERE pid = 10 OR pid=15 OR pid=18;

--------------------------------------------

SELECT pid,pname,pic FROM jd\_product

WHERE pid IN (10, 15, 18);

--------------------------------------------

SELECT pid,pname,pic FROM jd\_product

WHERE pid IN (SELECT productId FROM jd\_order\_detail WHERE orderId=101);

此处根据订单编号，查询出多个产品编号，再根据这些产品编号查询产品信息——查询中嵌套另一个查询——子查询

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月24日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

复习：

需求 -> PSD -> HTML -> PHP ->数据库

京东用户中心项目设计技术：

jQuery + AJAX + PHP/MySQL + Form + Video + Canvas + SVG + WebStorage

今日目标：

(1)京东用户中心——消费统计（Canvas版本）

(2)京东用户中心——消费统计（SVG版本）

(3)京东用户中心——幸运抽奖

1.项目实现步骤：

(14)创建PHP，buy\_stat.php(消费统计)，根据客户端提交的用户名，模拟创建过去12个月中的消费总金额（无需访问数据库，直接伪造出JSON数据即可），返回数据形如：[{"label":"10月","value":3500}, {"label":"11月","value":3000}...]

(15)修改JS，usercenter.js，待页面加载完成后，异步请求当前登录用户的消费统计信息，根据这些信息绘制Canvas统计图——注意：所有图形都要使用动态变量创建

(16)修改JS，usercenter.js，待页面加载完成后，异步请求当前登录用户的消费统计信息，根据这些信息绘制SVG统计图——注意：所有图形都要使用动态变量创建

(17)创建SQL，添加jd\_lottery表，保存用户的抽奖记录，包含如下列：(lid, uid,lotteryTime, level(INT,所中奖项))，添加三行测试数据

(18)创建PHP，lottery\_stat.php，接收客户端提交的用户名，返回该用户的抽奖统计情况，形如：{"uname":"qiangdong", "uid":1, "totalCount":39, "usedCount": 3, "leftCount":36}

总抽奖次数根据订单总金额来计算，每有1000元消费，就自动有一次抽奖机会

SELECT SUM(price) FROM 订单表 WHERE userId=?

SELECT COUNT(\*) FROM 抽奖表 WHERE userId=?

(19)修改HTML，待页面加载完成，异步请求当前登录用户的抽奖统计情况，显示在“开始抽奖”按钮上。剩余抽奖次数为0，则按钮保持禁用状态，否则按钮显示为可用状态。

--------------------------------------------

2.第三方绘图工具库

使用原生的Canvas或SVG实现统计图表的绘制比较麻烦。

许多第三方统计图表绘图工具产生：

(1)Chart.js：免费，小巧，八种图形

(2)EChart.js：百度提供的，免费，容易上手

(3)FreeChart：免费开源

(4)FusionCharts：收费的，功能强大，90+种图形

FC使用步骤：

(1)登录官网，查看介绍

(2)找示例程序，使用手册(API Document)

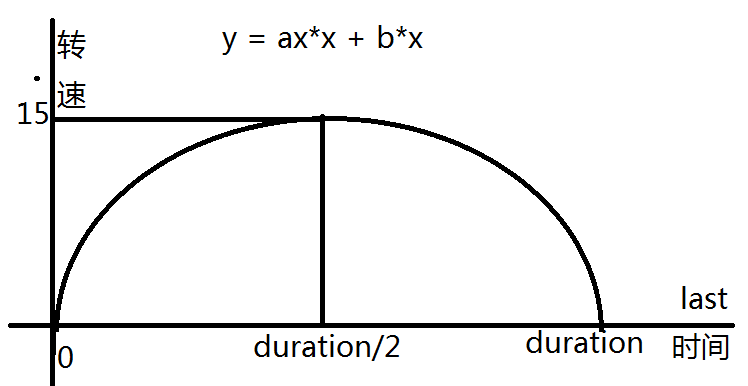
(3)上手编写简易程序

var c = new FusionCharts( {....} );

c.render('divId');

3.如何让旋转先加速再减速

本质是一个二次函数，X轴为旋转的持续时间，Y轴是旋转角度：



课后练习：

继续完善幸运抽奖，把抽奖结果异步提交给服务器，可以继续下一次抽奖；并让旋转先加速再减速。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月25日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

响应式框架

复习：

(1)AJAX

数据库、Web服务器、HTTP、原生AJAX、jQuery中的AJAX

(2)HTML5

表单、视音频、Canvas、SVG、地理定位、拖放、WebWorker、WebStorage、WebSocket

(3)Bootstrap

今日目标：

(1)响应式网页 —— 重点

(2)Bootstrap起步 —— 掌握

2G——不能上网

2.5G——GRPS——只能浏览WML网页

3G——智能操作系统——浏览HTML网页

1.什么是响应式网页？

Responsive Web Page，一个网页，可以根据浏览设备的不同，而自动更改布局、图片、文字效果，不会影响用户的浏览体验。

响应式网页的构成：

(1)布局：不能固定宽度，必须是流式

(2)文字和图片大小随着容器大小而改变

(3)CSS3 Media Query

2.如何测试响应式网页？

(1)使用真实的物理设备来测试

优势：测试效果真实可靠

不足：测试任务量太大

(2)使用设备模拟软件来测试

优势：快速，不需要准备大量的物理设备

不足：测试结果有待进一步验证

(3)使用Chrome自带的设备模拟器

优势：简单直观，功能比较丰富

不足：测试结果有待进一步验证

|  |
| --- |
| viewport：iOS诞生时网页都是为PC编写，只能进行强制缩放后才能在手机中显示——后果是所有的图片、文字都小到无法看清。苹果公司提出了Viewport（视口）的概念——可以在远比物理屏幕更宽的范围内浏览网页，不经缩放。后果是用户不得不左右滑动屏幕。后来的Android也借鉴了视口的概念。 |

3.如何编写响应式网页？——重点

手写响应式网页必须注意的六点：

(1)必须声明viewport元标签

<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no"/>

避免用户左右滑动屏幕，禁止浏览器初始时缩放显示

(2)容器的宽度尽量不使用固定值(px)，使用相对宽度(%或auto)

(3)文字尽量不使用绝对尺寸(px)，使用相对尺寸(em或rem)

(4)图片要指定大小，如下方式：

img {

/\*width: 400px;\*/

/\*width: 100%; 可随容器无限缩放\*/

max-width: 100%; 可随容器缩小，但最大不会超过图片的原始大小——防止马赛克

}

(5)页面布局时不能固定，而要用流式布局，如浮动、inline-block

(6)灵活的使用CSS3 Media Query技术——响应式网页必备元素！

4.使用CSS3 Media Query技术——重中之重

Media：指浏览网页的设备，如screen、tv、projection、tty(字符终端)、print、braille、speech等

Query：指自动获取当前浏览设备物理特性，如色彩深度、orientation、width、height等

CSS3媒体查询技术：可以自动根据浏览设备的类型及特性，执行不同的CSS代码。

CSS3MediaQuery有两种具体用法：

**(1)根据媒体查询的结果，执行不同的外部CSS文件**

<link media="screen and (max-width:991px) and (min-width:768px)" rel="stylesheet" href="css/pad.css"/>

不足：即使当前设备不执行的CSS文件，也会被浏览器加载；每个CSS文件中可能存在很多重复代码

**(2)在一个CSS片段中，有选择的执行某些选择器**

@media screen and (min-width: 768px) and (max-width:991px) {

选择器 { 名:值； }

}

午间练习：使用媒体查询的第二种用法，实现下面的响应式网页



PC下的布局效果 PAD下的布局效果PHONE

5.Twitter Bootstrap框架

Bootstrap：起步

**Twitter Bootstrap，是一个HTML/CSS/JS框架，用于简化响应式网页的编写。**

官网：http://getbootstrap.com/

中文镜像：http://www.bootcss.com/

主要内容分为5部分：

(1)起步

(2)全局CSS样式

(3)组件

(4)jQuery插件

(5)定制

提示：Bootstrap学习没有难以理解的知识点，但难点在1)词汇量偏大，记忆方面有难度 2)常用的class特别的使用要求，需要熟练应用。

6.Bootstrap第一部分：起步

下载

基本模板

|  |
| --- |
| <html lang="zh-cn">  <html lang="en">  <html lang="en-us">  <html lang="ja">  HTML标签的lang属性(language)，用于指定当前网页的自然语言。作用：1)为浏览器翻译功能指名原始语言 2)为屏幕阅读软件指明应该使用的发音 |

|  |
| --- |
| <meta http-equiv="X-UA-Compatible" content="IE=edge">  HTTP响应消息头部：X-UA-Compatible  Cross-UserAgent-Compatible：IE浏览器专用头部，告诉新版本的IE兼容其中的哪个版本的内核，edge表示启用能够使用的最新的版本内核  IE6： 内核6  IE7： 内核7/6  IE8： 内核8/7/6  IE9： 内核9/8/7/6  IE10：内核10/9/8/7/6  IE11：内核11/10/9/8/7/6 |

|  |
| --- |
| html5shiv.js：由第三方提供的通用JS，让老IE识别HTML5新的语义标签。 |
| respond.js：由第三方提供的通用JS，让老IE支持响应式网页——CSS3 Media Query |

练习：查找bootstrap.css对HTML基本标签做了哪些样式重置：

\* { box-sizing: border-box; }

html { font-size: 10px;}

body { font-size: 14px; margin:0; background:#fff; color:#333;}

h1 { font-size: 36px; margin-top:20px; margin-bottom:10px;}

...

h6 { font-size: 12px; margin-top:10px; margin-bottom:10px; }

p { margin: 0 0 10px; }

a { color: #337ab7; text-decoration: none; }

img { border: 0; }

table { border-spacing:0; border-collapse: collapse; }

div { ... }

span { ... }

7.Bootstrap第二部分——全局CSS样式

按钮相关class： 17:13

.btn 按钮基础样式

.btn-default 白底黑字的按钮

.btn-danger 红色按钮

.btn-warning 黄色按钮

.btn-success 绿色按钮

.btn-info 浅蓝色按钮

.btn-primary 深蓝色按钮

.btn-lg

.btn-sm

.btn-xs

.btn-block 块级按钮

图片相关class：

.img-circle 圆形图片

.img-rounded 圆角图片

.img-thumbnail 缩略图片

.img-responsive 响应式图片

文本相关class：

.text-五种颜色 文本颜色

.bg-五种颜色 文本背景颜色

.text-uppercase 文本转换为大写形式

.text-lowercase 文本转换为小写形式

.text-capitalize 文本转换为首字母与大写形式

.text-left 文本左对齐

.text-center 文本居中对齐

.text-right 文本右对齐

.text-justify 文本两端调整对齐

课下任务：

(1)自学“全局样式”—— 辅助类、排版、代码

(2)仿写如下的响应式网页 ——手写媒体查询，不直接使用bootstrap.css，但可以借鉴其中的class

PC下的布局效果：



PAD下的布局效果：



PHONE下的布局效果：



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月26日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

复习：

响应式网页：

概念：

如何编写： 六条原则

如何测试： 三种方法

Bootstrap框架：

是一个HTML/**CSS**/JS框架，适用于创建响应式Web页面。主要内容分为五部分：

(1)起步

下载、模板、实例、Bootlint、兼容性

(2)全局CSS样式

按钮、图片、文本、辅助类

(3)组件

(4)JS插件

(5)定制

今日目标：

(1)全局CSS样式

表格

栅格系统——重点

表单——难点

(2)组件

|  |
| --- |
| Boostrap指定的四种屏幕尺寸：  (1)超大PC屏幕——lg(large)：  w>=1200px  (2)中等PC屏幕——md(medium):  1200px>w>=992px  (3)PAD屏幕——sm(small)  992px>w>=768px  (4)PHONE屏幕——xs(extra small)  768px>w |

1.Boostrap中的两种容器

定宽容器：

.container:

1170px //lg

970px //md

750px //sm

100% //xs

变宽容器：

.container-fluid

100%

此外，两种容器都有:before和:after，可以清除子元素margin和float造成的影响。

2.全局CSS样式——表格 —— 小重点

.table 基础表格

.table-bordered 带边框的表格

.table-striped 隔行变色的表格

.table-hover 带鼠标悬停效果的表格

.table-responsive 响应式表格，必须使用在TABLE的父元素DIV上!

练习：如何给某个td或者tr变为红色/绿色...背景？.table-responsive的正确使用方法？

3.Bootlint

Bootlint 是 Bootstrap 官方所提供的 HTML/CS 检测工具。在使用了 Bootstrap 的页面上，它能自动检查某些常见的 HTML和class的使用错误——浏览器本身不具备这样的检查功能。

4.全局CSS样式——栅格布局系统——重点

前端页面可用的布局方法：

**(1)使用TABLE实现页面布局**

优势：简单，容易控制

问题：语义表达错误，渲染呈现的效果不友好

**(2)使用DIV+CSS实现页面布局**

优势：语义正确，渲染效果好

问题：不容易控制，尤其是响应式页面中

**(3)使用Bootstrap提供栅格布局系统(Grid Layout)**

优势：简单，容易控制，语义正确，支持响应式

问题：没有！

特点：类似于表格，有行(.row)和列(.col)的概念，底层实现却是DIV+CSS。

栅格布局系统使用方法：

(1)最外层必须是容器类： .container或.container-fluid

(2)容器中放置行： .row

(3)行中放置列： .col

(4)针对不同的屏幕有不同的列：

.col-lg-\* 适用于超大PC屏幕

.col-md-\* 适用于中等PC屏幕

.col-sm-\* 适用于PAD屏幕

.col-xs-\* 适用于PHONE屏幕

一个div可以同时声明多个不同屏幕下的列宽

<div class="col-lg-\* col-md-\* col-sm-\*">

(5)一行均分为12份，每个列都需要指定自己所占的份数

<div class="col-lg-2 col-md-6 col-xs-8">

(6)每个列都可以指定向右的偏移量，可以实现右错位的效果

.col-lg/md/sm/xs-offset-1/2/3...

(7)不同的列在不同的屏幕下有不同的适用性

col-lg-\* 只适用于LG屏幕

col-md-\* 适用于MD/LG屏幕

col-sm-\* 适用于SM/MD/LG屏幕

col-xs-\* 适用于XS/SM/MD/LG屏幕

列的偏移适用同样的规则，对当前屏幕以及更大的屏幕有效；

(8)可以指定某列在特定尺寸的屏幕下隐藏

.hidden-lg

.hidden-md

.hidden-sm

.hidden-xs

隐藏只对当前指定的屏幕有效！

(9)栅格系统可以嵌套，如：

<div class="container">

<div class="row">

<div class="col-xs-6">

**<div class="row">**

**<div class="col-xs-6"></div>**

**</div>**

</div>

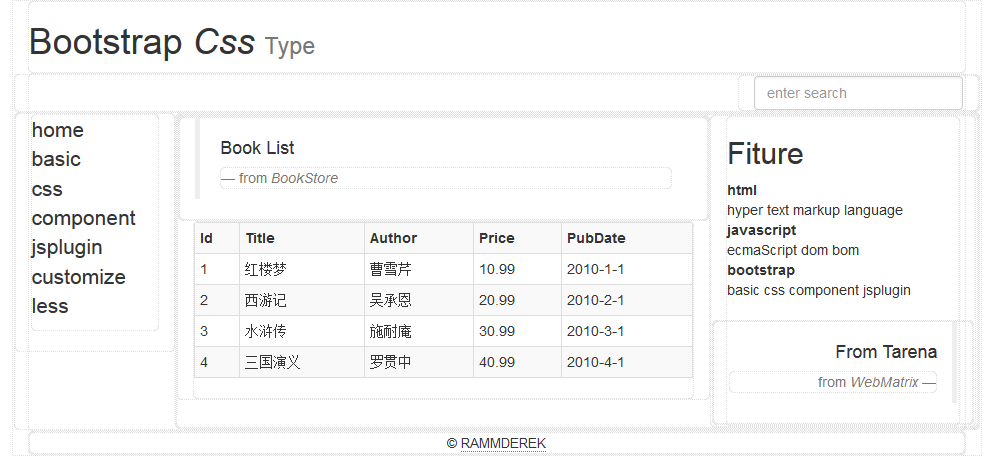
</div>

</div>

午间练习：

(1)一行中如何分出1:3:1(2:6:2)的三列？

(2)使用栅格布局系统实现如下布局：



input class="form-control"

练习：适用栅格布局系统实现：



PC下的布局效果 PAD下的布局效果PHONE

导航条： 4/12 4/12 6/12 6/12 12/12 12/12

主体： 1:2:1 1:2:0 100%:100%:0

5.全局CSS样式——表单——难点

HTML5标准有几个表单相关的推荐写法：

1)button必须指定type

2)输入组件都必须有对应的label——为无障碍阅读准备

<label for="xx">用户名</label><input id="xx">

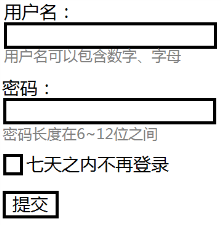
<label>

<input type="radio/checkbox">男

</label>

Bootstrap提供了几种形式的表单？

**(1)默认表单**



<form>

<div class="form-group">

<label></label>

<input class="form-control">

<span class="help-block"></span>

</div>

</form>

**(2)行内表单**



<form class="form-inline">

<div class="form-group">

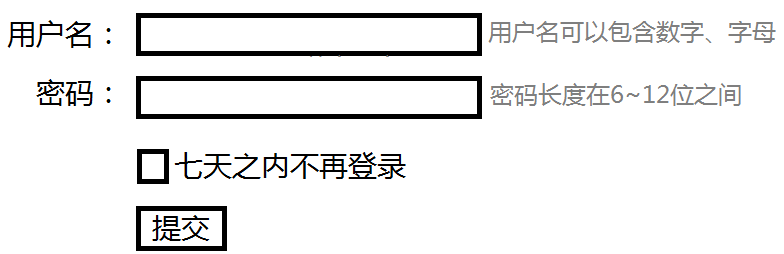
<label class="sr-only"></label>

<input class="form-control">

</div>

</form>

**(3)水平表单 —— 最难点**



提示：水平表单= 栅格布局（变种） + 表单

|  |  |  |
| --- | --- | --- |
|  | 标准栅格系统 | 表单中的栅格系统 |
| 容器 | div.container | form.form-horizontal |
| 行 | div.row | div.form-group |
| 列 | div.col-\*-\* | div.col-\*-\* |

<form class="form-horizontal">

<div class="form-group">

<div class="col-\*-\*">

<label class="control-label"></label>

</div>

<div class="col-\*-\*">

<input class="form-control">

</div>

</div>

</form>

6.Bootstrap第三部分：组件 —— 图标字体

图标字体是字体，可以无限缩放、修改文本颜色、背景颜色、阴影...，显示的内容却是图形图标。

前端领域常用的图标字体：

(1)FontAwesome —— 免费的

(2)Glyphicons Halfling —— 收费的

图标字体在Web项目中都是作为服务器端字体来使用——当客户端在访问网页时，即时从服务器上下载必需的字体文件：

@font-face {

font-family: 'Glyphicons Halflings';

src: url('../fonts/glyphicons-halflings-regular.eot');

}

.glyphicon {

position: relative;

top: 1px;

display: inline-block;

font-family: 'Glyphicons Halflings';

}

===================================

具体使用方法：

<span class="glyphicon glyphicon-\*"></span>

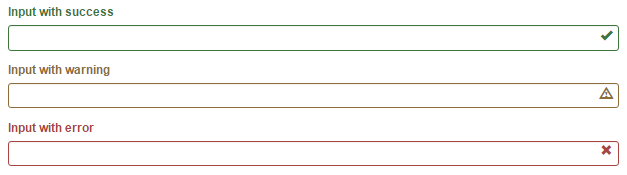
注意：span中不能有任何文本或子标签！

课后练习：

(1)创建按钮，上面有如下的常用图标：

主页、配置、汉堡包、定位、后退、前进、加号、减号、对勾、叉号、上一张、下一张、发邮件、星星、心形、播放、暂停、停止、上一首、下一首、放大镜

(2)自己参考手册，实现如下效果：



还有搜索框——在输入框中有个放大镜

(3)实现下面所示的响应式版权声明条

PC下的布局效果：



PAD下的布局效果：



PHONE下的布局效果：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月27日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

复习：

见思维导图

练习：为音乐播放器创建一系列的按钮：

上一首 快退 播放/暂停 停止 快进 下一首

1.Bootstrap第三部分：组件——图标字体

2.Bootstrap第三部分：组件——按钮组

.btn-group > .btn \* 5

.btn-group-justified

.btn-group-lg/sm/xs

.btn-group-vertical 竖直按钮组

3.Bootstrap第三部分：组件——下拉菜单

下拉菜单必需的三级结构：

<div class="dropdown">

<a data-toggle="dropdown">触发元素</a>

<ul class="dropdown-menu">

</ul>

</div>

4.Bootstrap第三部分：组件——导航

Bootstrap提供了三种形式的导航(nav)：

**(1)标签页式导航**

<ul class="nav nav-tabs">

<li class="active">

<a data-toggle="tab"></a>

</li>

</ul>

**(2)胶囊式导航**

<ul class="nav nav-pills [nav-justified nav-stacked]">

<li class="active">

<a data-toggle="tab"></a>

</li>

</ul>

**(3)导航条中的导航**

<ul class="nav navbar-nav">

<li class="active">

<a data-toggle="tab"></a>

</li>

</ul>

5.Bootstrap第三部分：组件——警告框

<div class="alert alert-四种颜色alert-dismissible">

<span class="close"data-dismiss="alert">&times;</span>

<p></p>

</div>

6.Bootstrap第三部分：组件——进度条

<div class="progress">

<div class="progress-bar" style="width: 30%"></div>

</div>

练习：使用定时器修改进度条宽度，让它真正的动起来

午间任务：

自学手册中的组件：路径导航(面包屑)-breadcrumb、分页条-pagination、分页器-pager、徽章-badge、标签-label、巨幕-jumbotron、页头-pager-header、水井(well)

7.Bootstrap第三部分：组件——列表组

**使用ul>li实现的列表组**

<ul class="list-group">

<li class="list-group-item"></li>

</ul>

**使用div>a实现的列表组**

<div class="list-group">

<a class="list-group-item"></a>

</div>

8.Bootstrap第三部分：组件——缩略图

缩略图一般配合栅格系统使用，实现批量的商品展示

<a class="thumbnail">

<img src="">

</a>

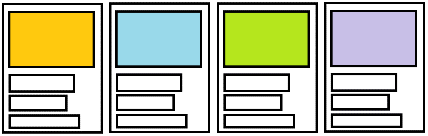
<div class="thumbnail">

<img>

<div class="caption"></div>

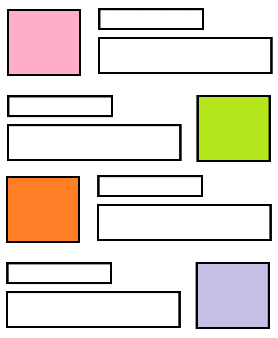
</div>

练习：商品展示，PC中一行四列，PAD中两行两列，PHONE中四行一列



9.Bootstrap第三部分：组件——媒体对象

左中右结构的商品/帖子展示：左边/右边有一幅图像，中央是说明文本



10.Bootstrap第三部分：组件——面板和面板组

面板组件把内容分为三部分：头、主体、尾部；

多个面板组合(称为面板组)可以实现“手风琴”组件。

<div class="panel panel-五种颜色/default">

<div class="panel-heading"></div>

<div class="panel-body"></div>

<div class="panel-footer"></div>

</div>

11.Bootstrap第四部分：JS插件 —— 折叠效果

折叠效果(collpase)本身很简单，

<button data-toggle="collapse" data-target="#b"></button>

<a data-toggle="collapse" href="#b"></a>

--------------------------------------------------------

<div id="b" class="collapse in"></div>

折叠效果有两个重要的扩展应用：

**(1)手风琴 —— 重点**

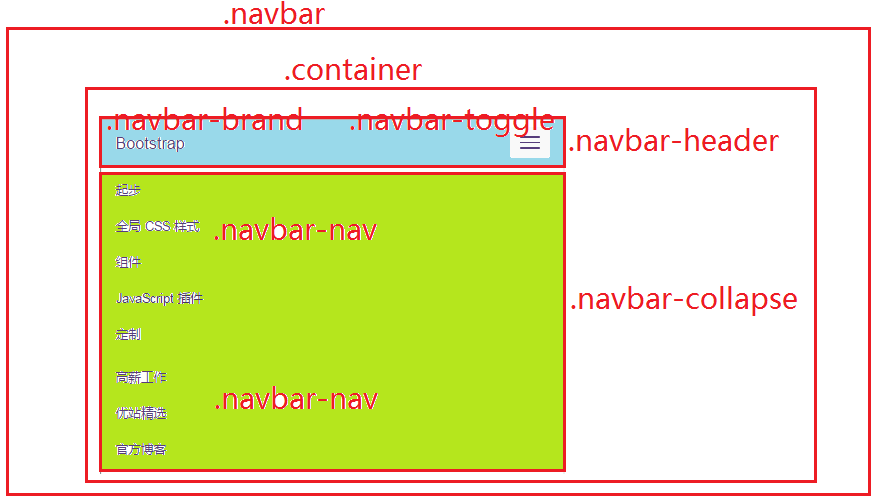
面板组+ 折叠效果插件

**(2)响应式导航条 —— 重点&难点**

12.Bootstrap第三部分：组件——响应式导航条

“响应式导航条”必须配合折叠效果插件使用。

响应式导航条在PHONE中只显示一个LOGO + 一个汉堡包按钮，其他菜单项全在折叠下拉菜单中；PAD和PC中，下拉菜单要实现绝对对位，定位到LOGO后面去。



Bootstrap中导航条分类：

(1)按颜色分：

白底黑字： .navbar-default

黑底白字： .navbar-inverse

(2)按定位方式分：

相对定位： 默认

固定定位： .navbar-fixed-\*

(3)按所在位置分：

固定在顶部： .navbar-fixed-top

固定在底部： .navbar-fixed-bottom

<div class="navbar navbar-default">

<div class="container">

**<!--导航条头部 = brand + toggle-->**

<div class="navbar-header">

</div>

**<!--导航条折叠部分 = 导航 + 链接 + 表单 + 按钮 ...-->**

<div id="my-menu" class="collapse navbar-collapse">

</div>

</div>

</div>

晚间练习：英特尔中文官网首页

注意：只完成页面布局和内容，**不做自定义样式设置**！！

难点提示：

(1)响应式导航条 —— 尤其是导航菜单中的下拉菜单。

(2)主体中的“大小小”的第二行如何实现——参考手册“栅格系统”中“列排序”相关知识。

(3)响应式版权声明栏。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月28日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

复习：

见思维导图

今日目标：

(1)Boostrap插件函数 —— 掌握

(2)Less语言 —— 重点

(3)Bootstrap定制 —— 比较乱

1.补充：列偏移 vs 列排序

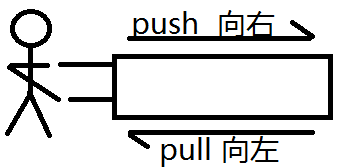
**列偏移**：控制一列出现的位置，某列偏移后，后续的列都会随之偏移——只能往右偏移

col-lg/md/sm/xs-offset-\*

**列排序**：控制一列出现的位置，某列可以往左拉或往右推——可以往右往左调整——某列排序后，其他列没有任何影响！

col-lg/md/sm/xs-push-\*

col-lg/md/sm/xs-pull-\*



|  |
| --- |
| 定义jquery对象插件：  jQuery.fn.函数名= function(){ }  调用jquery对象插件：  $('选择器').函数名(); |

2.Boostrap第四部分：jQuery插件

Bootstrap基于jQuery提供了十几个插件函数，可以有两种调用方法：

**(1)JS调用方式：** $('a').dropdown();

**(2)data-\*调用方式：**<a data-toggle="dropdown"></a>

提示：官方推荐使用方式2！但有两个插件提交特别：需要方式1和2同时使用才能奏效。

3. Boostrap第四部分：jQuery插件——下拉菜单

<div class="dropdown">

<a data-toggle="dropdown">触发元素</a>

<ul class="dropdown-menu">

</div>

4. Boostrap第四部分：jQuery插件——导航

<ul class="nav nav-tabs">

<li class="active"><a href="#">二十元套餐</a></li>

<li><a data-toggle="tab" href="#">二十元套餐</a></li>

</ul>

5. Boostrap第四部分：jQuery插件——警告框

<div class="alert">

<span class="close" data-dimiss="alert">&times;</span>

</div>

6. Boostrap第四部分：jQuery插件——折叠

<a data-toggle="collapse" href="#box">展开收起</a>

<div id="box" class="collapse"></div>

7. Boostrap第四部分：jQuery插件——工具提示

对HTML元素的title属性的呈现效果加以改观。

<ANY title="工具提示的内容" data-toggle="tooltip">

<script>

$('[data-toggle="tooltip"]').tooltip();

</script>

注意：此插件必须同时声明data-\*和js调用！还有一些可选参数，参考手册即可。

8. Boostrap第四部分：jQuery插件——弹出框

<ANY title="弹出框的标题" data-content="弹出框的内容"data-toggle="popover">

<script>

$('[data-toggle="popover"]').popover();

</script>

注意：此插件必须同时声明data-\*和js调用！还有一些可选参数，参考手册即可。

|  |
| --- |
| Module：模块  Model：模型  Modal：模态对话框 |

9. Boostrap第四部分：jQuery插件—— 模态框 —— 重点

Modal：模态对话框——在父窗口中弹出的子窗口，只要不关闭，则父窗口无法再获得输入焦点。

<a data-toggle="modal" href="#mid">打开模态框</a>

<button data-toggle="modal" data-target="#mid">打开模态框</button>

模态框必需的HTML结构：

<div id="mid" class="modal">半透明遮罩层

<div class="modal-dialog">尺寸位置

<div class="modal-content">背景/边框/倒角/阴影

<div class="modal-header">

<span data-dismiss="modal" class="close">&times;</span>

</div>

<div class="modal-body"></div>

<div class="modal-footer"></div>

</div>

</div>

</div>

午间练习：

(1)如何一刷新页面立即显示模态框——使用已经学过的技术

$('a').click(); $('a').trigger('click');

(2)完成一个完整的登录摸态框——包含表单和按钮

10. Boostrap第四部分：jQuery插件——轮播广告 —— 重点

提示：轮播广告的结构比较复杂！只要记得根元素div.carousel（旋转木马），剩余的全靠Bootlint的错误提示。

<div class="carousel" data-ride="carousel">

<div class="carousel-inner">

<div class="item active">

<img>

</div.

</div>

</div>

练习：

(1)为每个广告添加说明文字

(2)添加“上一条”、“下一条”按钮

(3)添加序号指示器

11.样式语言的分类

**静态样式语言**：CSS，可以直接被浏览器所理解；但是作为一门语言，CSS不称职，缺少语言必需的基本要素：数据类型、变量、运算、逻辑结构、函数、继承等。导致CSS可维护性低。

**动态样式语言/样式预处理语言**：在CSS基础之上添加了动态语言必需的要素，如：数据类型、变量、运算、逻辑结构、函数、继承等，从而极大的提高了样式的可维护性和可修改性。常见动态样式语言有三种：

(1)Sass / SCSS

(2)Stylus

(3)Less

注意：浏览器只能理解CSS！所有的动态样式代码必需转换为CSS才能被浏览器所解析！**x.less => x.css**必需有相应的转换程序——**动态样式预处理器/编译器**。

12.Less语言概述

官网：http://lesscss.org/

中文教程：http://www.bootcss.com/p/lesscss/

浏览器不能直接理解Less样式！

必需使用编译程序(Compiler)，把Less内容转换为CSS内容！

具体有两种使用形式：

**(1)在客户端使用Less —— 效率低，仅做了解**

1)编写.less文件

2)编写.html文件，引入.less文件

3).html中继续引入lessc.js文件

4)当浏览器下载html文件时，同时下载.less和lessc.js，自动执行lessc.js，把.less编译为CSS内容

**(2)在服务器端使用Less —— 重点掌握**

1)编写.less文件

2)程序员在自己的电脑上安装一款less编译器，把.less文件编译得到.css文件

3)编写.html文件，引入.css文件

4)客户端浏览html下载css即可

========================================

在自己的电脑上安装Less编译环境的步骤：

(1)下载并安装独立的JS解释器：Node.js

确保命令中 node -v 有正确的版本提示

(2)下载并解压缩less编译器：lessc.js

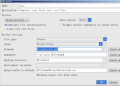
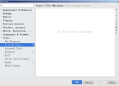
比如保存在e:/npm/node\_modules/less/bin/lessc

(3)运行JS解释器，执行lessc.js文件，把.less转换为.css

可以在命令中直接运行；

**node e:/npm/node\_modules/less/bin/lessc e:/x.less e:/x.css**

也可以在WebStorm中配置“文件监视器”自动运行；



配置好后，新建的或者修改后的.less文件都会自动被编译为.css文件

**使用WebStorm“文件监视器”需要注意： 项目路径中不允许有中文和空格！项目文件编码必须为UTF-8！Less必须新建或修改后才能自动编译！**

13.Less语法学习

(1)Less支持所有的CSS语法，即可以把.css重名为.less都是可以的！

(2)Less支持单行/多行注释，只有多行注释会被编译到.css中——推荐多多使用单行注释。

(3)Less中支持变量（variable）的概念：

定义变量：@变量名: 变量值;

使用变量：选择器 { color: @变量名;}

变量的值可以是任意合法的CSS属性值。

(4)变量和常量可以进行算术运算：加、减、乘、除、取余；

以及比较运算： ><>= <= !=

(5)Less中支持样式混入(mixin)的概念：

选择器1 {样式; }

选择器2 {选择器1;样式;}

(6)混入其它样式时可以带参

选择器1(@参1 , @参2) { 样式; }

选择器2 {选择器1(值1, 值2);}

(7)Less支持选择器的嵌套

选择器1 {

样式1;

选择器2 {

样式2;

}

}

编译后的结果为：

选择器1 {样式1; }

选择器1 选择器2 {样式2; }

(8)Less内置了几十个样式操作函数：

image-width() 返回图片的宽

image-height() 返回图片的高

ceil() 上取整

floor() 下取整

lighten() 颜色变亮

darken() 颜色变暗

(9)Less支持文件导入，

CSS中也支持文件导入的概念，由于会增加文件请求次数，不推荐使用；Less中的文件导入，则强烈推荐使用——被包含的.less会与当前文件进行拼接！

@import "xx.less";

@import "xx"; //可以省略.less后缀名

14.Bootstrap第五部分：定制

定制Bootstrap有三个目标：

**(1)减肥瘦身**：删除不需要的组件相关样式 —— 删除bootstrap.less不必要的@import即可；

**(2)全局定制**：定制大体的样式 —— 修改variables.less中变量的值，400+变量；

**(3)细节定制**：修改某个组件细节样式 —— 修改组件对应的.less的样式。

课下任务：

(1)自学插件——Affix(附加导航)

(2)自学插件——ScrollSpy(滚动监听)

(3)按照最新的Intel官网的效果，定制导航条组件、摸态框组件

(4)“我的大项目”中必须有一个响应式项目

第四阶段

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*10月31日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NG框架

1、ng

angularJS 是一个开源的js框架，用来开发单一页面应用，以及数据操作比较频繁的场景。

2、设计原则和设计模式

2.1 设计原则：

①YAGNI原则

you aren't gonna need it

不要写不需要的代码

②KISS原则

keep it simple and stupid

代码保持简介和具有表现力

语义话的标记、注释、变量或者方法的命名、减少嵌套

③DRY原则

Don't Repeat Yourself

模块的封装

提高代码的复用率 降低测试难度 方便后期的迭代和维护

④high cohesion low coupling

高内聚低耦合

内聚：一个组件内部 不同组成部分之间的关系

耦合：不同组件之间的关系

⑤SRP（single responsibility principle）

单一责任原则

⑥OCP（open closed principle）

开闭原则

⑦LCP 最少知识原则

2.2 设计模式

23种设计模式：

设计模式分为三大类：

创建型模式，共五种：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式。

结构型模式，共七种：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式。

行为型模式，共十一种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。

工厂模式、单例模式、迭代器模式、适配器模式。。

牢记7大设计原则，了解23种设计模式

2.3 MVC

mvc设计模式 组合型的设计模式

mv\*

M: model 模型数据

V: view html

C: controller 控制器

view 接收用户的操作 然后通知（controller）控制器当中对应的方法，对数据（model）操作，反过来更新了视图（View）

3、angularJS

3.1 概述

what？ 是一个开源的基于MVC的js框架

when？ 用在以数据操作为主的SPA应用

why？ ng的四大特征方便数据的处理以及模块方便实现SPA应用

ng的四大特征：

①采用MVC的设计模式

符合当前互联网的发展趋势：职责的细分

模块化开发：提高代码复用率 降低测试难度 维护方便

②双向数据绑定

是angularJS区别于其他框架的一个重要特性

③依赖注入

通过注入某些服务或者对象 直接调用

④模块化设计

ng框架本身就是符合模块化设计，使用框架结合模块、控制器、服务等来实现模块化开发

3.2 如何使用ng

基础语法：{{expression}} 可有执行括号中的表达式 将结果输出到innerHtml中

ng-app的作用：自动载入并启动ng应用

练习：在表达式中 是否能够执行算术、比较、逻辑、三目运算

ng-init的作用：初始化数据（变量、对象、数组。。。）

练习：使用ng-init指令 初始化一个对象数组（5个），

对象中包含stuName、stuAge、stuSex、stuChinese,

将数组显示在table中

[{},{},}{}].

注意事项：

ngApp 用来启动ng应用的，同时指定angular应用的作用范围，出了范围 ng中表达式 指令 是无法被执行的；只允许调用ngAPP指令一次。

如果通过调用一次ngInit 初始化多条数据，用逗号隔开

<p ng-init="name='Michael';age=20">

name = {{name}}

age = {{age}}

</p>

3.3 ng中常用的指令

ngApp ngInit

问题：闪烁{{}}有些时候会出现然后又变成正常数据

ngBind解决闪烁问题

ngRepeat让HTML支持循环

tr ng-repeat

练习：完成学生数组展示在table重构：ngRepeat

ngRepeat用法有两种

<ANY ng-repeat="tmp in set"/>

<ANY ng-repeat="(key,value) in set"/>

ngIf 让HTML支持选择（判断）

<ANY ng-if='表达式'/>

练习：完成学生数组展示在table，只显示语文成绩在80分以上的数据

ngShow 显示 true/隐藏 false

ngHide 隐藏 true / 显示 false

3.4 ng的MVC具体用法

Model View Controller

①声明模块

var app = angular.module("myApp",['ng']);

②注册模块

ngApp指令： ng-app="myApp"

③声明控制器

app.controller（"myCtrl",function(){}）

④使用控制器

ngController指令：ng-controller='myCtrl'

⑤在控制器的回调函数中，注入$scope对象，指定模型数据：

$scope.name = 'Michael' ;

$scope.age = 20;

⑥显示

{{}} ng指令

$scope 作用域控制对象，将模型数据和视图建立联系的桥梁

初始化数据：

ngInit

$scope对象

练习：

按钮 段落

模型数据 num = 0 ; --》显示在段落中

按钮--》点击 修改num 自增操作（onClick--》ng-click）

ng-click

ng-dblclick

ng-focus

ng-blur

掌握ng中MVC的基本步骤。

注意：

①定义模型数据，必须注入$scope

②ng-controller所在的标签内部的作用范围 才能识别控制器所声明的变量、方法

$scope.url='img/1.jpg';

<img src=''></img>

<img ng-src="{{url}}">

练习：

定义一个对象数组productList，对象中包含图片的路径、菜品的名称以及价格--》 $scope

通过ngRepeat，将productList展示出来

按钮（ng-click）--》 productList插入两条数据： push

、

每日一练：

简单的购物车功能，支持添加和删除数据

要求：通过MVC，结合ng指令。

push

splice（index,1）;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*11月1日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 1、指令

ng-\*\* --> angularJs内置的指令

自定义指令：

app.directive(name,function(){});

指令命名：采用驼峰式 前缀一般是工程名字的缩写 后缀是指令的作用

E:Element 元素 <ts-hello></ts-hello>

A：attribute 属性 <div ts-hello='''></div>

C: class <div class='ts-hello'></div>

M: comment <!-- directive: ts-hello --> 结合replace

restrict: 'EAC'

传参：

scope:{

testName:'@'

}

练习：创建自定义指令 tsDirective,传入ts-name='Hello World', 通过属性来调用,并显示出来Hello World

<div ts-directive='' ts-name='HelloWorld'></div>

在通过ngRepeat遍历数组，如果数组中有重复内容，会报错，通过指定track by $index

2、双向数据绑定

方向1

model ------------》view

方向2

view ------------》model

2.1 方向1：将模型数据绑定到视图中。

$scope.num = 1;

<p>{{num}}</p>

<p ng-bind="num"></p>

如何绑定？双花括号+常用的ng指令

绑定的作用：修改了模型数据 对应的视图就会跟着更新

DOM操作：先查找元素--》修改元素

ng： 创建model数据，绑定到视图，内部有脏数据检查机制，只有数据发生了变化，视图就会更新

$scope.num = 0;

p 显示num

button 点击 num做自增操作

练习：

定义模型数据 name='Michael'

放置两个按钮1、2，

点击按钮1 , 修改模型数据 name=''

点击按钮2，恢复模型数据 name=''

p显示name

2.2 方向2：将视图中组件的数据 绑定 到模型数据中

如何绑定：ngModel

使用：input 复选 单选 多行文本

练习： 方向2的绑定

复选框：是否同意 ng-model='isAgree'

单选框 ：选择性别

多行文本：留言本

$watch

双向数据绑定：

页面中既有方向1的绑定 又有方向2的绑定

model

num=0；

练习：使用双向数据绑定，实现如下页面效果

3、过滤器

过滤器： 实现对表达式结果的 刷选 过滤 格式化，达到更好的表现效果

过滤器用法：支持多重过滤以及传参

{{ 表达式 | 过滤器：param | 过滤器2：param }}

currency:'￥'

date:'yy-MM-dd'

uppercase/lowercase

limitTo:3

orderBy:'age':true 按照age进行降序排列（默认是升序）

练习：

模拟数据：对象数组（age\name\score） 5个

输出3个列表：

①按照age降序

②按照name增序

③按照score降序 只显示前三名

4、方法

angular.forEach

toJson:将一个json对象序列化 转化为一个json格式的字符串

fromJson：反序列化 转化为一个json对象

uppercase

lowercase

5、服务

在ng 服务是一种单例对象

$scope 作用域控制对象 id是叠加的

$rootScope 根作用域控制对象 id = 1

$rootScope是所有$scope的父元素

不同的控制器之间 所定义的模型数据 默认情况下，不允许被其他控制器调用的，但是共享的数据 放在$rootScope中，或者做一个嵌套

<div ng-controller="myCtrl02">

<p>myCtrl02 : {{num}}</p>

<p>myCtrl02 school: {{school}}</p>

<div ng-controller="myCtrl03">

<p>myCtrl03 school: {{school}}</p>

</div>

</div>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*11月3日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

##### 1、服务

1.1 $location

创建一个定时器，num 自增，绑定到视图。

$watch('num',function(oldValue,newValue){})

1.2 双向数据绑定的工作原理：

当一个模型数据num，绑定到视图之后，$scope添加一个监听$watch, 判断num是否发生变化，如果变化就执行func。

$digest 判断数据是否发生了变化,$scope.$digest();

$scope.$apply();==> $rootScope.$digest()

1.3 $interval $timeout

练习：

$interval

使用周期性定时器 来实现一个图片轮播；

$scope.imgList=['1.jpg','2.jpg','3.jpg'];

$scope.index = 0;

<img ng-src='{{imgList[index]}}'/>

同时实现一个自增的计数器，当计数器值大于30，自动停止。

1.4 $http

普通用法

$http({method:'get',url:'myUrl'}).success().error();

快捷操作

$http.get('myUrl').success()

post

put

delete

练习：完整版购物车

$http 与服务器的文件交互

①将apache服务跑起来

②将服务器端的php文件

放到c:/xampp/htdocs

如果想用post给服务器传数据：

app.run(function ($http) {

$http.defaults.headers.post =

{'Content-type':'application/x-www-form-urlencoded'};

})

$http.post('myUrl'，data).success(function(){}).error(function(){});

data数据 需要处理：$.param(obj);//序列化

2、自定义服务

app.factory('服务名字'，function(){

return {

}

})

app.service('服务名字'，function(){

this.\*\* = function(){}

this.\*\*\* = xyz;

});

app.constant('服务名字',value);

app.value('服务名字',value);

当常量服务注入控制器之后，都可以通过angular.extend方法进行重置。

constant常用于创建配置数据，而value常用于创建对象

3、ng四大特征的依赖注入

3.1 文件压缩

$injector 注入器

文件压缩：

方式1：通过命令行来进行文件压缩

①java的运行环境

windows+r ->cmd

②执行压缩操作

java -jar \*\*\*.jar test.js > test.min.js

方式2：借助webStorm来实现

①将jar放到C盘根目录下

②webStorm左上角的file，点击，选中settings，找到tools，在tools 找到file watcher

文件压缩的作用：

①删除所有的注释

②删除所有的空白字符

③尽可能的简化变量名、函数名、形参名==》 混淆

3.2 依赖注入的类型

①推断式（猜测式）

//创建一个服务 factory

app.factory('$show', function () {

return {

showWindow:function(){

alert('推断式依赖注入');

}

}

});

app.controller('myCtrl', function ($scope,$show) {

$scope.clickFunc = function () {

$show.showWindow();

}

});

②标记式

//创建一个服务 factory

app.factory('$show', function () {

return {

showWindow:function(){

alert('标记式依赖注入');

}

}

});

var func = function ($scope,$show) {

$scope.clickFunc = function () {

$show.showWindow();

}

}

func.$inject = ['$scope','$show'];

app.controller('myCtrl',func);

③行内式

在声明控制器的时候，第二个参数必须是一个数组，而且数组中内容是要用到的服务的字符串，最后一个元素是一个方法（控制器被调用时的回调函数）

app.controller('myCtrl',['$scope','$show',

function ($scope,$show) {

$scope.clickFunc = function () {

$show.showWindow();

}

}]);

注意事项：标记式和行内式依赖注入，服务的顺序需要保持一致

建议 采用行内式依赖注入。

练习：

采用service方法自定义一个服务 $addService，

在服务中一个方法：用来对传两个参数求和；

要求是进行行内式依赖注入。

input\input\button

每日一练：

使用service方法，创建一个服务，

在服务中有两个方法，

方法1：通过周期性定时器，每隔1s，在控制台输入‘Hello service’

方法2：结束周期性定时器

通过行内式依赖注入，在html有两个按钮，分别是开始和结束，调用服务中方法1 和方法2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*11月4日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 注入器

1、依赖注入，$injector 注入器

作用：快速的定位到要用的各种服务，定位和创建（实例化）

常用API：

①has方法

$injector.has(name);

根据传入的服务的name，判断是否在注册列表中，能找到返回true，否则返回false

②get方法

$injector.get(name); 返回指定名称的服务实例

2、ng四大特征之模块化设计

①采用MVC

②双向数据绑定

③依赖注入

④模块化

模块化好处：高内聚低耦合、提高开发速度、提高代码的复用率，降低测试难度

ng中模块：

①内置的模块：ng核心模块、ngRoute、ngAnimate。。。

②自定义模块：userModule、orderModule、payModule。。。

模块化开发，ng是支持自定义多个模块，myApp01,myApp02,

如果myApp02模块需要用到myApp01模块中的服务或者其他：

在声明模块 angular.module('myApp02',['ng','myApp01'])

面试：一个ng模块可以包括哪些组件？？

①服务 service

②指令 directive

③方法 function

④控制器 controller

⑤其他 过滤器、提供者等等

3、ng路由模块的学习

SPA：single page application

3.1 单页面应用工作原理：

①页面url： http://127.0.0.1/index.html#/路由地址

②浏览器解析完整的页面，再解析路由地址

③根据路由地址，查找真实的页面（路由词典）

④客户端发起异步Ajax请求，获取页面并插入DOM树，实现局部刷新

需要用到ngRoute模块（①ngView指令：盛放代码片段 ②$routeProvider配置路由词典）

3.2 步骤：

①引入ng的angular.js、angular-route.js

②创建模块，声明依赖于ng和ngRoute

var app = angular.module('myApp',['ng','ngRoute']);

③使用ngView指令 指定容器

<div ng-view></div>

④创建模板页面

⑤配置路由词典($routeProvider)

app.config(function($routeProvider){

$routeProvider

.when('/start',{templateUrl:'myUrl'})

.otherwise({redirectTo:'/start'})

});

练习：一个完整index.html,3个代码片段 分别是：start.html main.html person.html

要求：通过ngRoute来实现SPA应用，ngView $routeProvider：通过修改地址栏中的路由地址，来实现页面的切换

3.3 页面跳转：

①直接修改地址栏中的路由地址

②使用超链接

<a href='#/路由地址'/>

<a href='#/person'/>

③js

$location.path('/路由地址');

练习：一个完整index.html,3个代码片段 分别是：start.html main.html person.html

start.html: 按钮-》跳转到main.html

person.html： 超链接-》跳转到start.html

3.4 跳转的传参

①明确发送方和接收方

start-》main

发送：start =>$location.path('/main/123') <a href='#/main/333'>

接收：main

②针对接收方做设置:路由

$routeProvider.when('/main/:id',{

templateUrl:'tpl/main.html'

})

③设置接收参数

$routeParams.id

3.5 引入页面 ngInclude指令

<div ng-include="'tpl/include/header.html'"></div>

注意：引入时，在双引号中有一个单引号

3.6 调用控制器的时候，在实现SPA应用中有两种方式：

① ng-controller='ctrl'

②.when('/路由地址',{

controller:'ctrl'

})

练习：

实现一个单页面应用，

完整：manager.html

代码片段：check.html pay.html send.html

要求：能够实现不同页面的跳转，

check.html可以通过超链接跳转到pay.html，传入price，pay.html接收参数并显示在页面中

pay.html 可以通过按钮跳转到send.html（超链接点击，跳转到check.html）

创建SPA基本步骤、传参、页面跳转

小技巧：

body是div（盛放代码片段）的父元素，父元素中对应控制器中的方法，在代码片段中 是可以直接调用的。

4、动画模块

基本步骤：

①angular-animate.js引入

②angular.module('myApp',['ng','ngAnimate']); 使用动画模块

③调整四个状态来实现动画效果

ng-enter ng-enter-active

ng-leave ng-leave-active

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*11月8日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## webApp

一、WebApp、NativeApp、HybridApp区别

1、webApp 网页应用

使用前端技术开发，体验就像传统的应用

2、NativeApp 原生应用

采用SDK（software Develop Kit）说提供的原生组件或者服务进行开发

Android开发、iOS开发

3、HybridApp 混合开发

国外：phoneGap+jQM/ionic

国内：369Clound/appCan/...

二、如何搭建Android的开发环境

①配置jdk（java development kit）---》java能够运行

下载、安装、配置环境变量

②配置SDK

adt（Android Develop Tools） Google给Android开发者提供一整套工具（sdk、eclipse）

三、如何创建一个Android项目

新建项目：File->new->Android Application Project->配置应用名字-》图标-》。。finish

虚拟设备管理器-》对虚拟机的增删改查

注意：

① ActivityManager: Warning: Activity not started, its current task has been brought to the front

原因是由于对工程没有做任何修改，就去重复编译运行

②工程有×号

根据x号 找到是哪个目录下的具体的文件的问题，找到出错的那一行根据提示解决问题

③window->Reset Perspective（恢复默认视图）

工程目录：

src-> 存放的java代码 处理业务逻辑

gen -> 自动生成的文件（R.java）

assets-> 存储一些不希望被编译成流文件的一些文件（前端代码）

libs->存放jar包，提供的api

res->①drawable 存储图片 ②layout 布局文件

values->变量

AndroidManifest.xml 配置版本号、准备支持手机操作系统的版本、权限的配置、应用的。。。

四、如何构建一个HybridApp

前端代码+原生组件（WebView） -》混合编程

步骤1：编写前端代码

步骤2：

①本地加载 （前端放到assets）

②通过部署在服务器端，去访问url

步骤3：原生组件--》WebView 加载页面

WebView组件：

what？ 是一个网络视图组件，加载对应的html

when？ onCreate方法

where? 混合开发

how？

WebView wv = new WebView(getApplicationContext());

wv.loadUrl("file:///android\_asset/test.html");

setContentView(wv);

允许网络通信：

androidManefiest.xml文件，在标签页中找到permissions，点击add添加 INTERNET。

五、混合编程

练习：

有一个html文件，p、button，就是点击按钮，然后隐藏段落

①编写html文件

②新建一个Android工程，把html文件放到assets中

③通过webView，加载html并显示在app中。

重要：允许执行js

wv.getSettings().setJavaScriptEnabled(true);

如果想要在hybrid能够查看console日志信息。

//查看控制台的日志输出

wv.setWebChromeClient(new WebChromeClient(){

@Override

@Deprecated

public void onConsoleMessage(String message, int lineNumber,

String sourceID) {

// TODO Auto-generated method stub

super.onConsoleMessage(message, lineNumber, sourceID);

Log.e("test", "内容为"+message+" 行号为"+lineNumber);

}

});

结合ng来实现一个login.html，输入用户名和密码，底部还有一个按钮。点击按钮，将用户名和密码发送给服务器 login.php(解析到用户名，然后给客户端 name==》 欢迎登陆+name)，并显示出来分析：

①需要apache服务跑起来，然后将工程建在c xampp htdocs目录，新建login.html login.php,

②login.html：方向2的数据绑定、$http

③login.php 接收参数并返回

④查看本机的ip地址：

windows+r -->cmd->ipconfig 构造服务器url地址

http://localhost/chaptor4/code/webApp/day1/login.html

==》

http://172.163.100.161/chaptor4/code/webApp/day1/login.html

⑤载入url：WebView loadUrl()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*11月9日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一、PhoneGap的特点、历史、开发环境

1.1 介绍

what？

phoneGap 是一个 允许使用前端开发技术 快速的开发平台 ，能够调用底层的核心功能（加速度传感器、摄像头。。。）

where？ 实现hybridApp的开发

1.2 历史：

09 phoneGap（在开发者大会，尝试使用前端技术来代替oc）

09年底 支持了Android、BlackBerry等

11年 被ADOBE收购，将phoneGap核心技术开源了--》 apache cordova

12年 PhoneGap Build --》云端编译打包

1.3 区分PhoneGap、Cordova的区别和联系

cordova是借助phoneGap的核心技术进行开源的项目

cordova与phoneGap --》 webkit与chrome

1.4 站点：

phonegap.com

cordova.apache.org

http://phonegap-plugins.com/

http://www.phonegap100.com/

1.5 特点：

①兼容性

written once,run everywhere

②标准化

采用w3c标准

③大众化

不需要手机编程技术，只要会html、js就能实现原生开发的效果

1.6 搭建phoneGap开发环境

①通过命令行

1、安装cordova

npm install -g cordova

想要通过国内的npm镜像快速安装：

npm install -g cordova --registry=https://registry.npm.taobao.org

2、使用cordova去创建工程

cordova create test

3、cordova默认是空白工程，如果针对平台:

cordova platform add android

cordova platform add ios

cordova platform remove android 取消对android平台的支持

4、判断是否具备运行的条件

cordova requirements

5、编译运行生成安装包

cordova emulate android

cordova run android

②通过图形化

1、pc

phonegap

2、移动端

https://github.com/phonegap/phonegap-app-developer/blob/master/resources/release/android/PhoneGap-1.7.6-release.apk

启动模拟器

ADB（Android Debug Bridge）

adb install \*\*.apk 安装指定的apk包

adb version 查看adb的版本号

adb start-server

adb kill-server

...

windows+r ->cmd->输入命令

二、PhoneGap 常用API

事件列表中：电池状态、网络状态、按钮状态、ready等

document.addEventListener('事件名字',方法，false);

核心API：

插件：device

源码：github--》www device.js -->getInfo->exec

cordova.js-->NativeApiProvider 找到各平台中 对应的文件（android-》device.java）

练习：

index.html中 放置一个按钮，点击按钮 可确认对话框

notification

(alert/prompt/beep/vibrtate)

accelerometer

(getCurrentAcceleration/watchAcceleration/clearWatch)

capture

(navigator.device.capture.captureAudio/Image/Video())

file

(fileSystem/fileEntry/fileReader/fileWriter)

File

该对象包含单个文件的属性。

属性：

name：文件的名称。（DOMString类型）

fullPath：文件的完整路径，包含文件名称。（DOMString类型）

type：文件的mime类型。（DOMString类型）

lastModifiedDate：文件最后被修改的时间。(日期类型)

size：以字节为单位的文件大小。（长整型

FileEntry 该对象代表文件系统中的一个文件，W3C目录和系统规范对其进行了定义。

方法 createWriter：创建一个可以写入文件的FileWriter对象。

FileReader

FileReader是一个允许用户读取文件的对象。

onloadstart：读取启动时调用的回调函数。（函数类型）

onprogress：读取过程中调用的回调函数，用于汇报读取进度（progress.loaded和progress.total）。（函数类型） 不支持

onload：读取安全完成后调用的回调函数。（函数类型）

onabort：读取被中止后调用的回调函数，例如通过调用abort()方法。（函数类型）

onerror：读取失败后调用的回调函数。（函数类型）

onloadend：请求完成后调用的回调函数（无论请求是成功还是失败）。（函数类型）

readAsText：读取文件，结果以文本字符串返回。

FileWriter对象：

write：向文件中写入数据。

onwritestart：写入操作启动时调用的回调函数。（函数类型）

onprogress：写入过程中调用的回调函数，用于汇报写入进度（progress.loaded和progress.total）。（函数类型）— 不支持

onwrite：当写入成功完成后调用的回调函数。（函数类型）

onabort：写入被中止后调用的回调函数，例如通过调用abort()方法。（函数类型）

onerror：写入失败后调用的回调函数。(函数类型)

onwriteend：请求完成后调用的回调函数（无论请求是成功还是失败）。（函数类型）

两个按钮，读/写

①写

a.txt ->Hello File

②读

a.txt alert();

分析：

fileEntry fileReader fileWriter

getFile

每日一练：

根据加速度计，摇一摇-》改变背景色

①watchAcceleration

②判断当前和上一次加速度信息值的变化，临界值:20 -> 随机的

颜色

③定时器每隔0.5s 实现判断

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*11月10日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 、Ionic介绍

①what

一个强大的开源的h5应用程序的 开发框架，使用前端技术来开发混合编程应用。

angularJS android phoneGap/Cordova Ionic

ng 以数据操作为主的js框架--》SPA

android -》 apk （原生开发或者借助WebView组件混合开发）

Cordova/phoneGap--》平台（将调用底层硬件的借口封装成jsAPI）

Ionic 重点：外观和体验 轻量的手机UI库

Ionic（负责处理UI）+Ng（处理数据）+phoneGap（提供调用原生组件的jsAPI）

②特征

完美融合ng

专注原生，结合cordova调用底层硬件

设计风格漂亮

提供强大了强大的命令行（CLI command line）操作

性能优越 运行速度特别快

对于Android4.2 Ios6.0 以下的版本，ionic 不支持

③站点

ionicframework.com //官网

ionicons.com // 图标

http://www.ionic.wang/ 中文网站

appcamp.io //ng ionic 在线学习教程

二、Ionic 环境搭建

①命令行

1、确保npm能够使用

npm -v

(nodejs.org 下载nodejs 并安装即可使用)

2、确保cordova是可用

cordova -v

（npm install -g cordova）

3、安装ionic

npm install -g ionic

4、ionic的指令

ionic start 应用名字 ---》 创建ionic项目

②压缩包

1、找到ionic\_v1.3.1.zip

2、解压缩，文件夹引入到工程

3、引入ionic.css ionic.bundle.js

三、Ionic组件的使用(CSS 组件)

3.1 基础页面：header+content+footer

3.2 颜色：

bar-light

bar-stable

bar-positive 天蓝色

bar-calm 湖蓝色

bar-assertive 红色

bar-energized 浅黄色

bar-royal 紫色

bar-dark 深黑色

3.3 button

button-block button-full

button-small button-large

button-outline button-clear

3.4 list

class='list/item/list-inset/item-icon-left/item-icon-right/badge/item-note'

3.5 form

复选框

<ion-list>

<ion-checkbox ng-model="filter.red">Red</ion-checkbox>

<ion-checkbox ng-model="filter.yellow">Yellow</ion-checkbox>

<ion-checkbox ng-model="filter.pink">Pink</ion-checkbox>

</ion-list>

单选

<ion-list>

<ion-radio >男</ion-radio>

<ion-radio >女</ion-radio>

</ion-list>

3.6 tabs 标签页

class="tabs/tab-item/tabs-icon-top/bottom/left/right/only"

3.7 grid

class="row/col/col-20/col-offset-33"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*11月11日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一、下拉刷新

①使用对应的标签 <ion-refresher>

②设置对应的方法 on-refresh=''

③发出广播 结束刷新动作 $scope.$broadcast();

scroll.refreshComplete

二、上拉加载更多

① ion-infinite-scroll

②on-infinite=''

③scroll.infiniteScrollComplete

练习：

list 支持下拉刷新、上拉加载更多

class='list list-inset'

item 内容不做限制

三、$ionicLoading

$ionicLoading.show({template:'正在刷新'});

$ionicLoading.hide();

四、ActionSheet

var sheet = $ionicActionSheet.show();

sheet();

五、Popup

$ionicPopup.alert/confirm/prompt

六、侧边栏菜单

练习：实现一个侧边栏菜单（右边） 内部:按钮 关掉侧边栏

在header中的右边 放置一个按钮，点击按钮从右边打开侧边栏菜单

<ion-side-menus>

<ion-side-menu-content></>

<ion-side-menu side='left/right'></>

</ion-side-menus>

打开侧边栏菜单：

①js

$ionicSideMenuDelegate.toggleLeft/Right(true)

②扩展属性

menu-toggle='left/right'

关闭：

①js

$ionicSideMenuDelegate.toggleLeft/Right(false)

②扩展属性

menu-close

七、ion-tabs 标签页

<ion-tabs class='tabs-positive tabs-icon-left/right/top/bottom/only'>

<ion-tab on-select=''></>

</ion-tabs>

八、列表

ion-list

ion-item

ion-option-button

$ionicScrollDelegate.scrollTop/Bottom();

$ionicScrollDelegate.getScrollPosition();

$ionicScrollDelegate.scrollTo()

九、uiRouter

使用步骤：

① 引入ionic-bundle.js

②声明模块 将ionic写在依赖列表中

③容器

ui-view

④创建模板文件

⑤设置状态

app.config(function($stateProvider,$urlRouterProvider){

state

otherwise

})

跳转：$state.go('路由地址') ui-sref='路由地址'

每日练习：login register main

login:

按钮（没有账号？去注册）=》 register

按钮（登录）->main

register:

按钮（注册完成，去登录）=》login

main：

按钮（退出登录）-》login

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*11月14日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1、项目的搭建

①、创建项目，添加css、js、img、tpl文件夹，添加必须引用的css、js文件，添加自定义的css、js文件以及img图片，并创建完整的引导页面kaifanla.html；

②、编写kaifanla.html文件：

定义模块ng-app=”kaifanla”

引入ionic.css以及自定义的css文件

定义用于替换模板的视图

引入ionic\_bundle.js文件，并引入自定义的js文件。

③、添加模板文件：添加 start/main/detail/order/myorder页面，每个页面中删掉原有内容，添加一个文字；

④、编写 kaifanla.js 文件

定义各自的控制器

定义路由：为所有模板定义路由，默认跳转到start

⑥、测试：跳转是否正常，是否有错误

2、所有静态页面的编写---》运行通畅，数据静态死数据

3、调用php页面，联调

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*11月15日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

目的：实现移动端的app（开饭啦）

技术构成： ionic(ui库-》内容+ng-》数据)

1、项目的搭建

①、创建项目，添加css、js、img、tpl文件夹，添加必须引用的css、js文件，添加自定义的css、js文件以及img图片，并创建完整的引导页面kaifanla.html；

②、编写kaifanla.html文件：

定义模块ng-app=”kaifanla”

引入ionic.css以及自定义的css文件

定义用于替换模板的视图

引入ionic\_bundle.js文件，并引入自定义的js文件。

③、添加模板文件：添加 start/main/detail/order/myorder页面，每个页面中删掉原有内容，添加一个文字；

④、编写 kaifanla.js 文件

定义各自的控制器

定义路由：为所有模板定义路由，默认跳转到start

⑥、测试：跳转是否正常，是否有错误

2、所有静态页面的编写---》运行通畅，数据静态死数据

①start.html

②main.html 模拟数据

可以通过ng-include包含页头和页尾

③detail.html 详情页

card

④order.html 表单提交页

⑤myOrder.html 个人中心页

通过grid 模拟 table 展示数据

3、调用php页面，联调

条件：

①工程在c:\xampp\htdocs

②xampp的apache和mySql跑起来

③测试-》 localhost: / (不要直接在webStorm中打开)

①main.html

ng-model

-->

初始化模型数据：$scope.inputTxt = {kw:''};

$watch : $scope.$watch('inputTxt.kw',function(){})

方向2的数据绑定： ng-model='inputTxt.kw'

尝试： ion-infinite-scroll 上拉加载更多

②detail.html

发送： main->a

接受： detail

1、配置detail状态中的url /detail/:id

2、$stateParams

提交订单 id--》 order.html

③order.html

序列化：

$httpParamSerializerJQLike

④myOrder.html

根据手机号 去查找所有的订单

注册邀请：http://t.cn/RqG1Nja

混合编程：

打开eclipse，将模拟器，新建一个Android应用，通过webView的loadUrl

wv.loadUrl('http://172.163.0.1/ionic\_kaifanla/kaifanla.html');

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*11月16日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

$ionicConfigProvider.tabs.position('bottom');

# React 概述

what? 是一个开源的 为数据渲染视图的，由fb，Instagram 社区共同维护的 js库

why？ fb和Instagram，

频繁的数据操作-》 大量的DOM操作（自动操作DOM），逻辑会比较复杂（状态与内容对应起来）

历史：

11 --》 fb newsfeed

12 --》 Instagram

13 --》 宣布开源

15 --》 ReactNative

蚂蚁金服、airbnb、大搜车。。。。

特点：

①声明式设计

②高效 ： React虚拟DOM，最大限度的减少与DOM的交互

③灵活： 方便与其他库搭配使用

④组件：构建组件

⑤JSX：js语法的扩展 允许在js中写HTML

⑥状态： 内容与状态对应起来

网址：

reactjs.com --》 facebook.github.io/react

https://www.gitbook.com/book/hulufei/react-tutorial/details

代价：

①React 不会支持IE8以下的浏览器

②如果你的应用或者站点 不需要频繁的更新页面 --》React 很小的功能写了很多代码

③React 年轻的库，需要开发者去造轮子

④极大的减慢你的速度 react state props

二、React的核心思想和概念

2.1 核心思想

封装组件

各个组件 维护自己的状态和UI，当状态改变，自动更新组件

积木 -》 拼接-》功能复杂的应用：组件驱动的开发

2.2 核心概念

①组件

②JSX（用jsx语法取代HTML，在js中描述UI）

jsx 浏览器不识别，需要转换为js代码 （babel 将jsx转换为js）

③虚拟DOM

组件结构--》虚拟DOM

diff算法--》 当更新组件的时候，会通过diff算法寻找到需要更新的DOM节点，再把修改应用到浏览器实际的DOM节点

虚拟DOM是存在内存的 js数据结构 性能比原生DOM好

④数据流

单向数据流 状态与内容

三、搭建React的开发环境、使用React

3.1 HelloReact

引入react.js react-dom.js browser.js 三个js文件

<script type="text/babel">// 可以让浏览器识别jsx的语法

ReactDOM.render(

<h1>Hello React</h1>,

document.getElementById('example')

);

//ReactDOM.render(渲染的内容，渲染的元素);

</script>

3.2 JSX

基本语法：

①遇到HTML标签 (<)，使用HTML来解析

②遇到代码块（{}），使用js来解析

如果有注释，写在花括号中，块注释

练习：

新建一个html文件，测试jsx中表达式：

算术运算、逻辑运算、三目运算。。

3.3 组件

3.3.1 创建组件

var HelloMsg = React.createClass({

})

3.3.2 使用组件

<HelloMsg/>

注意事项：

①组件的命名：首字母必须是大写的。

②在渲染组件，返回要求只有一个标签，但是这个标签可以包含多个子元素

3.4 复合组件

复合组件：将其他的组件 整合在一起 创建的新组件

积木+积木--》乐高机器人

使用复合组件来实现登录窗口：

3.5 props

<HelloMsg name='daxu' age='20'/>

this.props.name

this.props.age

在创建组件的时候，可以指定key 和value ，value的值必须是一个表达式或者字符串

练习：使用复合组件以及props来重构登录窗口

props的注意点：

this.props.children 返回值有3种（①undefined ②object ③array）

React.Children.map(this.props.children,function(child){

})

3.6 ref

如果从组件获取真实DOM的节点，--》ref属性

//通过指定ref=‘ref名字’ --》 this.refs.ref名字

<input type='text' ref='inputTxt'/>

<button onClick=''/>

function(){

this.refs.inputTxt-->拿到真实的DOM节点

}

练习：

页面：有两个input标签 ，点击按钮-》 将标签中输入的数值相加-》控制台输出

组件、ref

组件构成：

定义组件a （input）

定义组件b （button）

复合main包含a组件和b组件

要求：通过b组件 按钮 是否能够拿到a组件中input里用户所输入的内容，直接输出到控制台？

分析：

a组件和b组件本身是没有关联的，可以借助main组件来实现。

a组件中用户输入的内容--》main组件

b组件 按钮点击 --》 main组件中

（方法的传递）

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*11月17日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1、state 状态

组件被认为是一个状态机，一开始 有一个初始状态，然后用户做了交互，状态变化，从而触发ui的更新。

①初始化状态

getInitialState（）

②设置状态

this.setState（）

③读取状态

this.state.状态名字

练习：使用state来读写数据，将a b main 重构！

2、组件的生命周期

2.1 三个阶段

Mounting 插入

Update 更新

UnMount 移出

2.2 声明周期的处理函数

mounting

①componentWillMount（）

②componentDidMount（）

update

①componentWillUpdate（nextProps,nextState）

②componentDidUpdate（preProps,preState）

unmount

①componentWillUnmount()

ref如果要想能够使用，必须是在组件插入到真实的DOM之后，也就是在componentDidMount之后，才能生效使用。

练习：在组件 插入到真实DOM节点之后，启动定时器，每隔0.1s 修改透明度 0~1

分析：componentDidMount state(opcacityValue)

不可以 style="opacity:{this.state.opcacityValue}"

推荐：style={{opacity:this.state.opacityValue}}

3、事件

handleClick:function(){

}

onClick={this.handleClick}

handleChange:function(){

}

onChange={this.handleChange}

组件中的方法

①react内置的方法

render

getInitialState

ComponentWillMount/Update/Unmount

....

②自定义方法(符合命名规范 handle+事件名称)

handleClick handleChange 驼峰式命名...

练习： onWheel 结合状态 来设置 div的背景颜色--》随机颜色

分析： 创建组件 div onWheel handleWheel

颜色--》状态

4、表单

受控组件：

表单元素

①input textarea 设置了value

②checkbox radio 设置了checked

③option 设置了selected

非受控组件：

input value-->输入内容，数据并没有变化

解决：结合状态 onChange --> 状态 value=状态值

5、ToDoList

①创建组件，使用组件 构造整个应用（模拟数据）

②state--》数组 ToDoBox完成的初始化

点击add按钮时，需要往状态所对应的数组中插入用户输入的内容 console.log(this.state)

③将状态中的数组 显示出来

将状态中数组 传递给 ToDoList， 通过一个方法：遍历数组，返回ToDoItem（值的传递）

④删除功能

ToDoBox 定义方法 根据参数 删除数组中的内容

需要将方法传递给ToDoList-》ToDoItem(根据遍历数组的index)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*11月18日\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# zeptojs

一、概述

①what

zeptojs是一个轻量的js库，它与jQuery有类似的API

②why

zepto的设计目的是不到10k的通用库，快速下载，有一个熟悉的api--》精力专注在开发

原因：

1、轻量

2、只支持现代浏览器

3、非常方便的搭配其他框架（phoneGap）来编写代码

4、优秀的源代码，性能良好

③优势

1、学习成本很低

2、比较轻量

④对比

浏览器兼容：zepto偏移动端 jQuery偏PC

文件大小：zepto 10k jQuery 30k

部分API接口：参数和执行结果有可能不一致

生态圈：jquery更好

核心库：zepto核心库的性能要比jQuery优秀

⑤如何选择

1、webApp：zepto

2、用户群体使用浏览器的分布

⑥网站

http://zeptojs.com/

https://github.com/madrobby/zepto

http://github.e-sites.nl/zeptobuilder/

二、搭建环境与使用

1 搭建环境

1.1 命令行（CLI commandLine）

npm install

npm run-script dist

# do a custom build

MODULES="zepto event data" npm run-script dist

# on Windows

c:\zepto> SET MODULES=zepto event data

c:\zepto> npm run-script dist

The resulting files are:

dist/zepto.js

dist/zepto.min.js

1.2 压缩包

可以直接去官网下载（自定义构建其他模块的--》(http://github.e-sites.nl/zeptobuilder/) builder）

2 zepto模块

默认：核心模块、event、ajax、form、ie

练习：$.parseJSON 将一个json字符串 转换为json对象

选择符的支持：

标签、id、属性、class、父子选择符、后代选择符

节点进行增删改查

插入： insertAfter insertBefore append appendTo prepend prependTo

删除: remove

改： html('') css addClass removeClass toggleClass...

查： next prev children parent parents find

3、event

练习：

count=0 button 点击按钮 ->控制器输出count

当值为5的时候，解除绑定（加上 one:只调用一次 输出'only once'）。

4、Ajax

$.Ajax({})

$.get();

$.post()

$('h1').load()

练习： 使用post请求

name='web1607'

①按照$.ajax() 发起请求

②$.post()

demo.php 接收参数 返回json数组 :name(接收的值) class（'class1'）

接收到服务器返回的结果：结果打印出来。

注意：apache服务，工程在于htdocs

5、Touch

//swipeLeft swipeRight swipeUp swipeDown

//doubleTap tap longTap

建议：将touch模块 与 detect、gesture搭配使用 结合默认模块完成自定义的构建