第一天

赵旭

zhaoxu@tedu.cn

1、WEB 前端(代码实现)

1、HTML5基础

CSS3基础(Basic)

PROJECT\_1(京东-首页)

Web UI : 切图

CSS3高级(Core)

PROJECT\_2(京东-详情页)

Javascript 基础(DOM)

20天

2、Javascrip 核心(Core)

DOM 核心(Core)

20天

3、jQuery 框架

HTML5高级

服务器端与HTTP

Ajax (阿贾克斯)

20天

4、Bootstrap (CSS框架)

AngularJS (JS框架)

Web APP

其他前端框架

20天

背

敲

name red

2、HTML5互联网工程师(偏UI实现)

1、同上

2、WEB 界面设计 、 WEBUI设计工具

3、Javascript 核心

DOM编程

jQuery框架

4、HTML5高级

Bootstrap

Web APP

学习方法：

1、背、练

2、复习、预习

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*下午\*\*\*\*\*\*\*\*\*\*\*

1、web相关知识

1、web

是Internet环境下的一个具体网页应用

简称：网页、web页面、页面

Internet:互联网，由若干个终端以及特殊的连接介质所组成

2、web的工作原理

1、B\S模型(结构)

B : Browser 浏览器

S : Server 服务器

PC : Personal Computer

将web应用全部保存在服务器(Server)上,由浏览器(Browser)发起请求进行访问

C/S 结构(模型)

C : Client

S : Server

通过指定的客户端来访问服务器的应用程序。eg:QQ、各种游戏

3、WEB 相关技术

1、服务器端技术

都提供了数据库的访问技术

php

jsp(Java Server Page)

asp (Active Server Page)

asp.net

2、客户端（浏览器端）技术

由浏览器负责解释运行

1、HTML

2、CSS

3、Javascript

2、HTML快速入门

1、HTML

Hyper Text Markup Language

超级 文本 标记(标签) 语言

a : 26个英文字符中的第一个字符

b : 26个英文字符中的第二个字符

超文本 a : 链接

超文本 b : 加粗

<> : 标记

Language : 语言,语法

web 页面 是由 html组成的，并且 以.html或.htm作为后缀的文档文件,各个页面间，是通过超链接组织到一起的。

开发web应用所使用的工具:

1、记事本(能编辑文字的工具)

2、浏览器(IE、Google Chrome、Firefox、Safari、Opera(欧朋))

2、HTML基础语法

1、标记语法

标记 又 称为 元素

语法:<标记内容>

注意：不同的标记，会有不同的表现结果。主要用于描述功能

分类：

1、封闭标记(双标记)

语法:<标记>内容</标记>

<b>内容</b>

<s>内容</s>

2、非封闭标记(单标记、空标记)

语法:<标记>或<标记/>

特点：不能包含内容，每一个独立的空标记就是一段独立的功能

<br/> : 换行

<hr/> : 水平线

2、标记的嵌套

在一个标记中出现另一对标记，形成更复杂的语法，或者让功能进行叠加

<标记1>

<标记2>

</标记2>

</标记1>

注意：

1、代码缩进

具备嵌套标记的标签，子级内容一定要通过 若干(一个制表符) 空格来表示缩进

<标记1>

<标记2>

</标记2>

</标记1>

<a>

<b>内容</b>

</a>

<a>这是一个超链接</a>

<a>这是一个超链接</a>

2、注意嵌套顺序

<a>

<b></b>

</a>

<a><b></a></b>

3、属性 和 属性值

可以通过属性和属性值，对标签的功能加以修饰

<p>一个段落</p>

语法:

1、属性的声明必须位于开始的标记里面

2、一个元素可能有多个属性，多个属性间用 空格 区分

3、多个属性之间，排名不分先后

属性值：

1、属性 和 属性值之间 通过 = 连接 属性=值

2、属性值 尽量用 引号 引起来

<p 属性名=值></p>

<p 属性名="值"></p>

<p 属性名='值'></p>

<p align="right" >段落</p>

标准属性：

又称为通用属性，基本上每个标签都会具备的属性。

id:标签在页面中的唯一标识

title:鼠标移入到标签上时所显示的文字

class:定义(引用)样式表中的类样式

style:定义当前标签的 行内样式

eg:<p id="p1" title="I'm a p markup" class="testp" style="color:red;"></p>

4、注释

1、什么是注释：指的是允许出现在html中，但不被浏览器所翻译执行的内容

2、注释的作用

1、对代码进行解释说明

2、整理思路

3、注释的语法

<!-- 这

是

注释

区域 -->

<a <!-- 写在这里就是错 -->>

<!-- 该处写注释是允许的 -->

</a>

3、html文档结构

1、html文档组成

1、文档类型声明

声明当前的html文档的版本以及类型的。

版本：

html5

xhtml1.0

html4.01

html3

html2

类型：

strict : 严格类型

transitional : 过渡类型

frameset : 框架集

出现的位置：位于 整个 html 文档的最顶端

HTML5的文档类型声明：

<!doctype html>

2、html页面组成

1、文档根标记

<html></html>

2、出现在 根标记里面的结构

1、网页的头部

<head></head>

2、网页的主体内容

<body></body>

3、head 元素

包含的都是全局信息

title : 网页的标题

meta :单标记， 编码格式、关键字、描述....相关的全局信息

script : 定义或引入 Javascript文件

link : 单标记，引入外部css样式表

style:定义内部样式表

4、body元素

1、body出现的位置 在 head元素之后，html元素之内。包含了网页要显示给用户去看的信息，称之为主体内容。

2、特殊字符的表示

回车 、若干空格 ，最终都会被解析成一个空格

特殊字符通过 转义字符 来实现的。

空格: &nbsp;

> : &gt; --> greater than

< : &lt; --> less than

© : &copy; --> Company

3、文本标记

1、文本样式相关

<b></b> : 加粗

<i></i> : 倾斜

<s></s> : 删除线

<u></u> : 下划线

<sub></sub> : 下标

<sup></sup> : 上标

2、标题元素

以醒目的方式进行显示,通常用作文章的标题

<hn></hn>

n:1-6

<h1></h1>

<h2></h2>

...

<h6></h6>

3、段落元素

提供了结构化文本的一种方式。

<p></p>

4、换行元素

<br> 或 <br/>

可以出现在任意需要换行的位置处

5、分区元素

常用于元素的分组，以及页面的布局

1、块分区元素

<div></div>

所有的效果需要开发者自己定义，主要通过 css 实现布局

2、行内分区元素

<span></span>

多个span会在一行内显示

span很少用在布局中。主要功能是 包含普通文本并且自定义样式。

6、行内元素 和 块级元素

块级元素：自己会独占一行，前后的内容会自动换行。

div、p、h1~h6

块级元素作用：多数都会用作布局中。

行内元素：可以和其他元素位于同一行。

s,b,u,i,span,

行内元素的作用：多数都会包裹文本，去实现文本的不同样式

注意：块级元素可以嵌套行内元素，但行内元素不允许嵌套块级元素。

<div>

<span></span>

<s></s>

<b></b>

</div>

<b>

<div></div>

</b>

注意：没强调的情况下，任何标记不允许嵌套div

<p>

<div></div>

</p>

7、水平线

<hr>

属性：

size:高度

width:宽度

align:水平对齐方式

color ： 颜色

8、预格式化

作用：保留文档中的格式，即保留文档中的 换行 和 空格

<pre></pre>

第二天

http://code.tarena.com.cn

用户名：tarenacode

密码：code\_2013

1、图像和链接

1、目录

文件夹

2、URL

URL : Uniform Resource Locator

统一 资源 定位器

资源：一个网页中要用到的不同类型的文件，比如：图片，音视频文件，文档

定位器：查找资源的一种方式

URL：就是俗称的【路径】

路径的分类：

1、绝对路径

2、相对路径

3、根相对路径

1、绝对路径

获取资源文件所在的从最高路径下开始的完整路径

查找本机资源文件：

E:\赵旭\01\_HTML5Basic\Day02\images\jd\_logo.png

如果访问的是互联网的资源：

协议 主机名称(ip，域名) 目录 文件名

http://www.baidu.com

http://www.baidu.com/img/bd\_logo1.png

2、相对路径

资源文件的位置是相对于【当前文件】的位置而查找出来的路径

1、同级目录：直接用

2、子级目录：先进入,再引用

3、父级目录：先返回(../),在引用

3、根相对路径

/ : 表示 当前web站点在服务器上的根路径

/images/jd\_logo.png

2、图像

语法：

<img />

属性:

src : 引用的图像的路径(URL),可以是绝对的，相对的，根相对的

可选属性：

width ：宽度

height：高度

title :

1、鼠标移入的时候显示的文本

2、图片不显示的时候，所显示的替代文本

注意：

1、width、height 如果只设置一个属性的话，那么图片会变成等比缩放

2、尽量不去修改宽度和高度

3、链接

用户通过 【鼠标点击】 的操作 能够完成【页面的跳转】 就是链接

语法：

<a>链接文本或图像</a>

属性：

href:链接URL（必须）

可选：

target : 目标，即打开新网页的方式.取值：\_self:默认值，当前页面打开，\_blank:新标签页打开

name : 定义 【锚点名称】

链接的表现形式：

1、链接到普通页面

<a href="http://www.baidu.com">百度</a>

2、下载文档资源

在页面中，除 图片、和txt以外的其他内容，都允许被下载

<a href="资源的路径">内容</a>

3、电子邮件链接

<a href="mailto:邮件地址">内容</a>

4、链接到 Javascript (后期讲解)

<a href="javascript:js代码">内容</a>

5、返回页面顶部的空链接

<a href="#">内容</a>

锚点：

在网页上任意一个位置去定义一个【标识】，随时随地都可以通过链接【跳转到这个位置】上来

使用方式：

1、定义锚点

<a name="锚点名称"></a> 其他标签需用id eg：<div id="锚点名称"></div>

2、链接到锚点

<a href="#锚点名称"></a>

<a href="页面URL#锚点名称"></a>

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

1、表格

1、表格的作用

表格通常是用来组织结构化信息

表格是有一些被称为【单元格】的矩形框 从左到右，从上到下排列到一起组成的。

数据可以保存在单元格中

2、创建表格

语法：

1、表格<table></table>

2、行 <tr></tr> --> Table Row

3、单元格(列) <td></td> --> Table Data

注意：html中的表格， 默认情况下，每行的列数都是统一化的。

需求：创建一个 3 行 4 列的表格

3、表格的属性

1、表格属性

width:宽度

height:高度，以 像素(px)为单位

align:设置表格的对齐方式(left,center,right)

border:边框,设置边框的宽度 px为单位

cellpadding:设置内边距(单元格边框与内容之间的距离)

cellspacing:设置外边距(单元格之间的距离) 以px为单位

bgcolor:设置表格的背景颜色

2、行属性

align:设置 当前行中【内容】的水平对齐方式，取值：left,center,right

valign:设置 当前行中【内容】的【垂直】对齐方式，取值：top,middle,bottom

3、列(单元格)属性

width:

height:

align:

valign:

colspan:跨列

rowspan:跨行

4、表格的标题

语法：<caption>标题内容</caption>

该标签直接写在<table></table>中，并且每个表格中最多只能有一个标题。

5、列标题

语法：<th></th> --> table head

注意：级别 与 td相同

6、表格的复杂应用

1、行分组

表格可以整体划分成3个部分：表头、表主体、表尾

表头:<thead></thead>

表主体:<tbody></tbody>

表尾：<tfoot></tfoot>

注意：

每组中，可以包含任意多对 tr

2、不规则表格

属性：

colspan : 跨列,横向向右的合并几个单元格，包含自己

rowspan : 跨行,纵向向下的合并几个单元格，包含自己

3、表格的嵌套

允许在 td 中嵌套另一个表格

练习1：

创建不规则表格

1、先创建 3\*3 的表格

2、跨行、跨列

2、列表

能够从上到下按照一定的顺序排列内容

列表的组成：

1、列表类型 -- 列表长什么样子

2、列表项 -- 列表中的内容

语法：

1、列表类型

1、有序列表

<ol></ol>

2、无序列表

<ul></ul>

2、列表项

要出现在 ol 或 ul 内部

<li>内容</li>

属性：

1、有序列表

type : 列表项类型

取值:

1,默认，数字

a,小写字母

A,大写字母

i,小写罗马数字

I,大写罗马数字

start : 起始编号

设置 type 的值从第几位数开始

2、无序列表

type :

取值：

disc : 默认，实心圆

circle : 空心圆

square : 实现矩形

none : 无

使用场合：所有的 从左到右显示的一组数据，或从上到下显示的一组数据，都可以使用列表来做。

列表的嵌套：

<ul>

<li>

<ol>

<li></li>

</ol>

</li>

</ul>

3、自定义列表

根据需求创建一个类似于列表的结构体。

语法：

dl : 声明自定义列表

dt : 定义列表中的一个标题(专业术语)

dd : 定义列表中的对标题的解释或对专业术语的解释。

使用场合：在 图文混排 的结构中，多数使用 dl,dt,dd 来表示,。

eg：<dl>

<dt>图</dt>

<dd>文</dd>

<dd>文</dd>

</dl>

第三天

1、结构标记

1、作用

为了提升代码的语义性

2、结构标记

1、header 元素

语法：<header></header>

作用：表示信息部分的头部内容,页眉

等同于 <div id="head"></div>

<header id="head"></header>

....

<header id="content\_header"></header>

2、nav 元素

navigation : 导航

语法：<nav></nav>

作用：表示或定义页面中的导航链接部分

等同于 <div id="nav"></div>

3、section 元素

语法：<section></section>

作用：定义文档中的节(主体内容中的某一部分)

等同于 <div id="section"></div>

4、article 元素

article ：文章

语法：<article></article>

作用：定义与文字性相关比较强的内容，如：帖子，报纸文章、博客条目、微博信息

5、footer 元素

foot ：脚

语法：<footer></footer>

作用：定义页面中靠下的部分内容，页脚内容，多数会将 版权信息、友情链接 等相关内容放在此标签中。

等同于：<div id="footer"></div>

6、aside 元素

语法：<aside></aside>

作用：定义页面的一些额外组成部分。

一般情况下，边栏可以使用 该标记

等同于:<div id="aside"></div>

2、表单

1、表单的作用

收集用户信息，并将用户信息提交给服务器

1、表单元素

2、表单控件

2、表单元素

语法：<form></form>

属性：

1、action

作用：定义表单被提交时所发生的动作。主要用于定义表单提交的地址

默认 为 提交给本页

2、method

方式、方法

作用：表单数据提交的方式

取值：get 和 post

get:获取，取得，获取

post:邮寄，发送

1、get ：表单将以【明文】的方式将数据发送给服务器。安全性较低。有大小限制，大小限制为 2KB。

常用场合：与搜索相关、无安全性要求的。

2、post ：表单将以 【隐式】的方式将数据发送给服务器。安全性较高。大小没有限制。

常用场合：登陆操作、注册操作

method 如果不写，默认值为 get

3、enctype

作用：表单数据进行编码的方式

取值：

1、application/x-www-form-urlencoded

默认的

2、multipart/form-data

适用场合：如果表单中有【文件】需要提交的话，那必须将enctype设置为此值

4、name

定义表单的名称

5、id

定义表单在页面中的唯一标识

3、表单控件

用不同的外观形式展现出来的用于接受用户数据的内容。

常用表单控件：

1、input 元素

2、多行文本域(textarea)

3、选项框 (select , option)

1、input 元素

语法：<input />

通用属性：

1、type ：根据 type 值 ， 来创建不同的input 元素

2、value ：值，当前控件的值(默认值)

3、name ： 控件的名称

使用场合：服务器

4、id : 控件的唯一标识

使用场合：当前页面

5、disabled : 禁用

input 中的具体元素

1、文本框与密码框

文本框:type="text"

<input type="text" />

密码框:type="password"

<input type="password" />

属性：

name:

value:

maxlength : 限制输入的字符数

readonly : 设置元素为只读

2、单选按钮 、复选框

单选按钮：<input type="radio" />

属性：

name ：除了定义名称为，还能够为单选按钮分组

checked : 设置 默认被选中

复选框：<input type="checkbox" />

属性：

name：定义名称 和 分组

checked：设置默认被选中

3、按钮

1、提交按钮

具备 提交 的功能,将表单中的数据提交给服务器

<input type="submit" />

2、重置按钮

具备 默认值功能 ,将表单中的所有表单元素全部恢复成初始化状态

<input type="reset" />

3、普通按钮

作用：无任何功能，可以由用户定义功能(js)

<input type="button" />

4、隐藏域 、文件选择框

1、隐藏域

对服务器 来讲非常重要，单又不想给用户去看的东西，可以放在隐藏域里

隐藏域是页面上看不到，但是能随着表单提交给服务器的控件

<input type="hidden" />

name : 定义名称

value : 定义值

2、文件选择框

打开本机 的 ”打开“对话窗口，并且选择文件，上传到服务器

<input type="file" />

属性：

name ：定义名称

注意：

1、method 必须 为 post

2、enctype 必须更改为 multipart/form-data

2、label 元素

作用：关联 文本 与 控件的。点击文本就等同于点击它所关联的那个控件。

语法：<label>关联的文本</label>

属性：

for : 与当前文本相关联的控件的【ID】值

3、选项框

分类：

1、下拉选项框

2、滚动列表

1、下拉选项框

<select></select>

属性：

name ：名称

id ：唯一标识

size : 大于1，则为滚动列表

multiple:设置多选,(针对滚动列表有效)

<option>显示的文本</option>

属性：

value ：选项的值

selected ：预选中

4、多行文本域 textarea

多行的文本框

语法：<textarea></textarea>

属性：

readonly : 设置为只读

cols : 设置文本域的 列数,一行能显示多少个字符,变相设置宽度

rows : 设置文本域的 行数,整个文本域默认显示几行,变相设置高度。

5、为控件分组

语法：

<fieldset></fieldset> : 分组

<legend></legend> : 为分组指定标题

3、其他标记

1、浮动框架 iframe

语法:<iframe></iframe>

属性：

src : 引入的网页URL

width:宽度

height:高度

frameborder:引入的网页的边框宽度

2、摘要与细节

可以对网页上的某部分信息进行展开和收缩

语法：

<details></details> : 定义细节内容

<summary></summary> : 定义细节内容的标题

3、度量元素

按照一定比例的方式去显示数值

语法：

<meter></meter>

属性：

min ：度量范围的最小值，默认为0

max ：度量范围的最大值，默认为1

value：当前的度量值，默认值为0

4、时间元素

语法：

<time>文本</time>

属性：

datetime:规定当前元素的日期和时间 用T进行分割

eg:

今年<time datetime="2016-2-14T15:00">情人节</time>，我跟你过！

5、高亮文本显示

以特殊的效果 显示 所包含的文本

特殊效果：默认情况下，背景颜色为黄色

语法：<mark>文本</mark>

test?userName=admin&userPwd=test&gender=female&hobby=drink

test?userName=dsfsafd&userPwd=dfsafsafsa&gender=female&hobby=eat&hobby=drink&hobby=play&hobby=happy

test?userPwd=dsadasf&gender=female&hobby=drink

test?userName=请输入用户名称&userPwd=&gender=male&userID=110120119

控件取名：适用于 name 和 id

name 属性值 ， 可以 与 id属性值取值相同

1、命名规范

匈牙利命名法

控件的缩写作为前缀

文本框：type="text" ==> txt

密码框：type="password" ==> txt

单选按钮: type="radio" ==> rdo

复选框： type="checkbox" ==> chk

按钮(提交、重置、普通) ： ==> btn

隐藏域、文件选择框: ==> txt

前缀后，跟着当前控件的 【作用名称】

作用名称：每个单词的首字符 大写，其余小写。

<input type="text" id="txtUsername" />

<input type="radio" id="rdoFemale" />

<input type="submit" id="btnSubmit" />

2、尽量保证见名知意

txtLoginName

3、尽量避免重复

第四天

1、CSS

1、css作用

提出统一的方式来修改页面元素的【样式】

2、什么是CSS

CSS ：Cascading Style Sheet

级联样式表 或 层叠样式表 俗称：样式表

3、css的优势

1、实现了 【内容】 与 【表现】 的分离

2、更大程度的去提高代码的【可重用性】 和 【可维护性】

4、CSS的使用方式

1、内联样式(行内样式)

将 样式 定义在单个的html元素中的

使用方式：将 样式 定义在 style 属性中

特点：只对当前标记有效。

语法：<p style="样式内容"></p>

样式内容：样式属性:属性值;样式属性:属性值;样式属性:属性值;

样式属性：

color : 文本颜色

取值:red,blue,yellow,green,pink,black..

background-color:背景颜色

取值:red,blue,yellow,....

2、内部样式表

将所有的样式定义在 HTML 的 头元素<head>中

语法：

<head>

<title></title>

<!-- 定义内部样式表 -->

<style type="text/css">

样式规则

</style>

</head>

样式规则：要修改的 标记 的 属性

样式规则语法：

选择器{

样式属性:属性值;

样式属性:属性值;

样式属性:属性值;

样式属性:属性值;

.....

}

选择器：规范了哪些元素能够使用当前定义好的样式。

元素选择器:(标记选择器),由元素的名称作为选择器，符合该选择器的元素，都将使用定义好的样式

eg:

div{

color:red;

background-color:green;

}

特点：能够实现 样式表的 可重用性 和 可维护性

3、外部样式表

将样式定义在一个外部的 css 文件中(\*.css文件)

由HTML页面对 外部样式表 进行引用

使用步骤：

1、创建一个 css 文件 保存 样式规则

2、在需要的html页面上引入 css 文件

在 <head>标记中

<link rel="stylesheet" type="text/css" href="style.css" />

5、CSS 样式表特征

1、继承性

大多数css样式规则是可以被继承的

2、层叠性

页面中可以定义多个样式表

如果不冲突时，多个样式表中的样式可以层叠为1个

3、优先级

样式定义冲突时，按照不同样式规则的优先级来应用样式

样式优先级:

低1、浏览器缺省设置(默认显示规则)

中2、外部样式表 或 内部样式表

就近原则

高3、内联样式(行内样式)

!important 规则 ! : 非

可以通过 !important规则调整样式优先级

!important 放在样式属性值之后，中间用 空格 分开，该值的优先级是最高的。

div{

color:red !important;

}

注意：谨慎使用，能不用就不用

练习

1、添加 h3 元素 定义默认样式

文本颜色 ：黑色 black;

文字大小 ：12px

2、使用内联样式 定义字体颜色为灰色

3、内部样式表中 ，定义字体颜色为红色，字体大小为 30pt

4、外部样式表 demo.css中，字体颜色 黄色， 字体大小 10pt，背景为 灰色(gray)

5、查看效果

6、改变 内部样式表 与 外部样式表的位置 再查看效果

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

1、CSS基础选择器

选择器：规范页面上哪些元素能够使用定义好的样式

1、通用选择器

语法：\*

作用：与页面上任何一个元素匹配

通常定义页面上所有的标签的默认样式

eg: \*{

font-size:12px;

color:red;

}

2、元素选择器

别名：标记选择器、标签选择器

语法：由html标记 作为选择器如，html,body,div, span,a,b,

作用：修改某个标记的默认样式

eg:

a{

text-decoration:none;/\*去下划线\*/

}

3、类选择器(类样式)

用户可以随意定义一个选择器，然后由标签 通过 class 属性来进行引用

语法：.className{样式属性:值;}

引用：通过 标记的class属性 对 类名 进行引用

<style>

.warning{

color:red;

}

</style>

<body>

<p class="warning">Test</p>

</body>

class 属性值 ， 支持列表形式，称为 值列表。即class属性后允许出现多个 class 值，每个值之间用 空格 区分

4、分类选择器

将类选择器 与 元素选择器结合起来使用。以实现对某种元素中的不同样式的细分控制。

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

<div class="warning"></div>

语法：元素选择器.类选择器{}

作用：精确定位到某个元素上

eg:

div.important{

}

5、id 选择器(id样式)

以一种独立的方式来定义样式

仅作用在 id 属性值 能与当前选择器名称匹配的元素上

语法:#idName

eg:

#nav{

background-color:blue;

}

<div id="nav"></div>

布局用id ， 内容实现用class

div#nav{

}

#nav{

}

6、群组选择器

作用：统一定义一组选择器中的通用样式

p{

color:green;

}

div{

color:green;

}

span{

color:green;

}

语法：用 , 隔开的选择器列表

p,div,span,#nav,.back{

color:green;

}

div{

background:yellow;

}

#nav{

background:red;

}

7、后代选择器

后代元素选择器

作用：通过 元素 的位置关系来匹配到元素

语法：选择器1 选择器2

div span{}

#d1 span{}

div.nav ul.red{}

8、子代选择器

子代元素选择器

作用：通过 元素 的位置关系来匹配到元素

缩小 匹配 的范围

特点：只通过一层 父子关系来查找元素

语法：选择器1>选择器2

匹配的是 选择器1【下一级】元素中，满足选择器2的元素

9、伪类选择器

作用：匹配元素 不同的状态的。

语法：

选择器1:伪类选择器{}

伪类分类：

链接伪类

动态伪类

目标伪类

元素状态伪类

结构伪类

否定伪类

1、链接伪类

:link , 定义尚未访问的链接样式

:visited , 定义访问过的链接的样式

2、动态伪类

匹配用户的一些动作

:hover , 定义 鼠标 悬停 在元素上的样式

:active , 定义 html元素被激活时的样式

:focus , 定义 html元素获取焦点时的样式

2、单位

1、尺寸单位

2、颜色单位

1、rgb(r,g,b)

r: red(0-255)

g: green(0-255)

b: blue(0-255)

rgb(0,0,0) : 黑色

rgb(255,255,255) : 白色

rgb(255,0,0) : 红色

rgb(0,255,0) : 绿色

rgb(255,255,0) :

rgb(128,35,92) :

rgb(r%,g%,b%)

rgba()

r:

g:

b:

a:alpha,透明度 0-1之间的小数

2、#rrggbb

十六进制：0-9 A-F

#000000 : 黑色

#ffffff : 白色

#ff0000 : 红色

#fe0000 : 红色

#e4393c : 京东红

#rgb : --> #rrggbb 缩写 ,仅限每两位数字都一样的情况使用

#000 --> #000000

#fa0 --> #ffaa00;

#000 : 黑色

#111 : 灰色（深）

....

#eee : 灰色（浅）

#fff : 白色

3、尺寸属性

用于描述 元素的 高度 和 宽度

单位 ：像素、百分比

宽度：width:给确定宽度

max-width: 给定 宽度范围最大值

min-width: 给定 宽度范围最小值

高度：height:给确定高度值

max-height:

min-height:

注意：一旦元素给定宽度 和 高度后 ， 如果元素中的内容超出范围，则会产生 溢出 的效果

4、溢出

overflow:当内容溢出元素框时如何处理

取值：

visible : 溢出可见,默认值

hidden : 溢出隐藏

scroll : 溢出后滚动,默认的话会有滚动条

auto : 自动，不溢出，正常显示，溢出后，显示滚动条

单向溢出：

overflow-x:横向溢出后的处理

overflow-y:纵向溢出后的处理

取值：

visible、hidden、scroll、auto

5、哪些HTML元素允许设置尺寸属性

块级元素 允许设置 尺寸相关属性

行内元素则不可以

具备 width ，height 属性的html元素 ， 允许通过css来修改

img table

6、边框

属性：

简写：border:width style color;

width : 取值 为 数值

style ：表示 线条样式 取值 solid(实线),dotted(虚线),dashed(虚线)

color : 设置颜色,可以取值为 transparent(透明色)

border:1px solid #ddd;

设置上下左右四个边框的宽度为 1px ， 样式为实线，颜色为 #ddd色

单边定义：

border-left/right/top/bottom : width style color;

border-width:四个边框宽度

border-left/right/top/bottom-width:

border-style:四个边框样式

border-left/right/top/bottom-style:

border-color:四个边框颜色

border-left/right/top/bottom-color:

border-方向-属性:

方向：left,right,top,bottom

属性：width,style,color

第五天

1、边框倒角

圆角矩形

属性：border-radius

按照顺时针的方向，从左上角开始，设置四个角的倒角半径

取值：

可以为1个 ， 也可以为 4个

取绝对值，也可以取百分比

单独定义：

border-top-left-radius:左上角

border-bottom-left-radius:左下角

border-top-right-radius:右上角

border-bottom-right-radius:右下角

2、边框阴影

属性:box-shadow

取值：多个属性值的 值列表， 中间用 空格 区分

h-shadow : 必需，水平阴影的位置

v-shadow : 必需，垂直阴影的位置

blur : 可选，模糊距离

spread : 可选，阴影的尺寸

color : 可选，阴影的颜色

inset : 可选，将默认的外部阴影改为内部阴影

常用方式：

box-shadow:h-shadow v-shadow blur color;

3、轮廓

绘制于元素周围的一条线，位于边框的外围,起到突出元素的作用。

属性：outline : width style color;

outline:0px; 取消轮廓，多用于表单元素上

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

1、框模型

框模型：内容（宽、高）、内边距、边框、外边距

对象实际宽度：左外边距 + 左边框 + 左内边距 + width +右内边距 + 右边框 + 右外边距

1、外边距

1、什么是外边距

围绕在元素 边框 周围的空白区域

会在元素 外围 创建空白区域，并且是透明的

2、属性

margin : value;

单边设置：

margin-left:左外边距

margin-right:右外边距

margin-top:上外边距

margin-bottom:下外边距

取值：

px

%

auto

负值

注意：对于【块级】元素来说 ， 设置【左右外边距】为【auto】的话，那么该元素会【居中】显示

值的数量

margin : value ; 上下左右统一值;

margin : v1 v2; v1 : 上下 ， v2 ：左右

margin : v1 v2 v3; v1 : 上 ， v2：左右，v3：下

margin : v1 v2 v3 v4;上 右 下 左

eg:

margin:15px 20px 8px;

解释：上边距 15px 左右外边距 20px 下边距 8px

margin:2px auto 8px auto;

解释：上边距2px 左右居中 ， 下边距8px

注意：在html 中，有一些元素会具备默认外边距

body

h1 - h6

p

ul li

dl dt dd

margin 所带来的影响：

1、外边距合并

两个垂直外边距相遇时，他们将形成一个外边距，称为外边距合并。最终外边距的值，以 大值 为准。

2、设置子元素的上外边距时，可能会成为 父元素的上外边距。

解决方案：

1、设置父元素的边框

2、通过 父元素的 上内边距

2、内边距

1、什么是内边距

内容区域和边框之间的空间

注意：内边距会扩大边框所占用的区域

2、语法

padding : value;

padding:v1;上下左右 四个内边距

padding:v1 v2;上下 左右

padding:v1 v2 v3;上 左右 下

padding:v1 v2 v3 v4;上右下左

padding-top

padding-bottom

padding-right

padding-left

取值单位:像素 、百分比

2、背景

1、背景颜色

2、背景图像

属性：background-image

取值：url(路径)

eg:background-image:url(路径);

3、背景重复

平铺

当图片大小小于 元素大小时，就会出现背景重复。

属性：background-repeat

取值：

repeat : 在垂直方向和水平方向平铺，默认

repeat-x:仅在水平方向平铺

repeat-y:仅在垂直方向平铺

no-repeat:不平铺，只显示一次

4、背景图片尺寸

属性：background-size

取值：

value1 value2 : 宽度，高度

value1% value2% :百分比

cover:覆盖，

contain:包含,背景图是能够完整的显示出来的，元素不一定会被完整覆盖。

5、背景图片固定

属性：background-attachment

取值：

fixed : 背景图片固定，背景图不会随着滚动条而发生滚动

scroll : 默认值，滚动

6、背景定位

背景图片在元素中应该出现的位置

属性:

background-position

取值：

x y : x 表示 水平位置 偏移量，+ 为右，-为左 , y表示 垂直位置 偏移量，+为下，-为上

x% y% : 偏移量取决于元素的 width 和 height,多数用于 背景图 水平和垂直居中的时候

x 取值

left

center

right

y 取值

top

center

bottom

7、背景属性(综合)

属性:background

取值:值列表

color url() repeat attachment position

eg:

background:#f00 url(images/pwd.png) no-repeat fixed 0px 0px;

常用：

background:url repeat position;

background:red;

3、渐变

1、渐变语法

属性:background-image

取值:

linear-gradient : 线性渐变

radial-gradient : 径向渐变

repeating-linear-gradient : 重复线性渐变

repeating-radial-gradient : 重复径向渐变

eg:

bakcground-image:linear-gradient(渐变参数);

1、线性渐变

语法：

linear-gradient(angle,color-point1,color-point2,...)

angle : 渐变方向或角度

to top : 向上 --> 0deg

to right :向右 --> 90deg

to bottom : 向下 --> 180deg

to left : 向左 --> 270deg

color-point : 颜色点

颜色值 位置

red 0%

blue 50%

yellow 100%

eg :

background-image:linear-gradient(to bottom,red 0%,blue 50%,yellow 100%);

2、径向渐变

语法:

radial-gradient([size at position],color-point1,color-point2,....);

size: 圆的半径

position：圆的圆心坐标

3、重复线性渐变

语法:

repeating-linear-gradient(angle,color-point);

第六天

1、文本格式化

1、字体属性

1、指定字体

属性：font-family

值：value1,value2 ....

eg:

font-family:"microsoft yahei","arial";

2、字体大小

属性:font-size

值：px ，pt 为单位

3、字体加粗

属性：font-weight

取值：normal/bold/400-900

4、字体样式

作用：斜体

属性：font-style

值：normal/italic

5、小型大写字母

属性：font-variant

取值：normal/small-caps

6、字体属性 font

属性:font

取值：font-style font-variant font-weight font-size font-family;

eg:

font:12px "microsoft yahei";

font:bold 12px "微软雅黑";

font:12px/24px "微软雅黑";

2、文本属性

1、文本颜色

属性：color

2、文本排列(水平)

属性:text-align

取值：left,center,right

注意：设置在外层元素上控制内层的【文本】 和 【行内元素】水平对齐方式

3、文本修饰

线条修饰

属性：text-decoration

取值：none/underline/line-through

4、行高

作用：设置 一行 文本的高度是多少，如果行高高于文本的高度，那么文本将在这段高度重垂直居中

使用场合：制作文本的垂直居中，将行高设置成与当前元素高度一致即可。缺点：只能有一行文本

属性：line-height

值: px

5、首行文本缩进

第一行的开始处，空出多大的【距离】

属性：text-indent

取值：以 px 为单位的值

6、文本阴影

属性：text-shadow

取值：h-shadow v-shadow blur color

eg:

text-shadow:0px 0px 1px red;

2、浮动定位

1、定位

定义元素框相对于其 正常位置 应该出现的位置

位置 可以是 相对于自身、父元素位置、其他元素以及浏览器窗口本身的位置。

1、定位分类

1、普通流定位(文档流定位、正常流)默认定位方式

2、浮动定位

3、相对定位

4、绝对定位

5、固定定位

2、普通流定位

块级元素：从上到下的一个接一个的排列

内联元素：在一行中从左到右的水平排列

2、浮动定位

1、什么是浮动定位

将元素排除在普通流之外(脱离文档流)

元素将不再占据页面空间

将元素放在【包含框】的【左边】或者【右边】

浮动元素依旧位于包含框之内

2、浮动属性

属性：float

取值：left/right/none

3、清除浮动元素所带来的影响

属性：clear

取值：left/right/both/none

定义了元素的哪边上不允许出现浮动元素

3、浮动元素对 父元素的影响

父元素的高度，与子元素(浮动的)无关

解决方案：

1、设置父元素的高度

2、通过 overflow 解决

在父元素内增加 overflow:hidden;属性撑起父元素高度。

4、元素只要浮动起来，就会变成 块级 的

行内元素 要是 浮动起来的话，就【允许】修改其宽度和高度

第七天

1、显示

1、显示方式

页面元素的显示模式

属性：display

取值：none / block / inline / inline-block

none : 让生成的元素没有框,即让生成的元素不显示（隐藏）。

block : 让元素表现的像块级一样,独立成行

inline : 让元素表现的像行内元素一样,多个元素占一行

inline-block : 行内块,还是行内元素,会具备块级元素的一些特点-->允许修改元素的宽和高

2、显示效果

1、可见性

真正意义上完成元素的隐藏

属性：visibility

取值：

visible ：可见的 ， 默认值

hidden ：元素不可见，但是依然占据空间

collapse：用在表格上，删除一行或一列时不影响表格布局

尤其注意：hidden（占空间），与 display:none;(不占空间)

2、透明度

属性：opacity

取值：0-1之间的小数

0.0(完全透明) - 1.0(完全不透明)

3、垂直对齐方式

属性：vertical-align

取值：

baseline : 默认，基线对齐,放在父元素基线上

top : 顶端对齐

bottom：底端对齐

middle：居中对齐

使用场合:

1、table里

2、img ：图片左右的文本相对于图片本身的垂直对齐方式

3、光标

属性：cursor

取值：

default 浏览器默认

pointer ：鼠标变成 手的 形状

crosshair ： 十字

text : I

wait : 等待

help : 帮助

2、列表样式

1、列表项标识

属性：list-style-type

无序列表：

取值： none

disc

cirdcle

square

有序列表：

取值：

none

decimal

lower-roman

...

li{

list-style-type:none;

}

2、列表项图像

将默认的标识替换成自己的图像

属性: list-style-image

取值：url( 图像路径);

3、列表项位置

属性：list-style-position

取值：

outside：默认值

inside : 将标识放在文本区域内

4、列表属性

属性：list-style :type url(路径) position;

常用方式：list-style:none;

3、定位

1、定位方式

属性：position

偏移属性：top，bottom，left，right

1、普通流定位

语法：

position:static;

所有元素的position默认值都为 static

2、相对定位

1、什么是相对定位

相对于元素原来的位置偏移某个距离

2、特点

元素仍然保持当前的形状

元素原本所占的空间依然保留

3、语法

position:relative;

属性：position

值：relative

通过 left,right,top,bottom 实现位置偏移

4、使用场合

1、元素位置会发生微妙的变化

2、配合着其他的(绝对定位)定位方式一起来使用

3、绝对定位

绝对定位 和 相对定位的差别，定位元素的起始位置不同

1、什么是绝对定位

将元素内容从普通流中完全移除(脱离文档流),不占据页面空间。

绝对定位的起始位置：相对于离它【最近的】【已定位的(relative,absolute)】祖先元素。

如果当前元素不具备已经定位的祖先元素，那么它的位置就相对于最初的包含块(body)。

2、语法

属性：position

值：absolute

通过 top、bottom、right、left 实现位置的偏移。

3、使用场合

1、适合配合着 relative 做 弹出菜单

4、固定定位

将元素固定在页面中的某个位置处，不随着滚动条发生位置的变化。

固定定位也从文档流中溢出，不占用页面空间

属性：position

值：fixed

通过 top,left,right,bottom属性修改定位位置

4、层叠顺序

属性：z-index

取值：不带单位的数字

使用场合：在 已定位 元素身上使用，值越大者越靠前

06\_CSS3CORE 第一天 2015.12.18

1、选择器

1、兄弟选择器

1、相邻兄弟选择器

匹配指定元素的相邻【下一个】兄弟

语法：由 + 充当结合符

选择器1+选择器2

2、通用兄弟选择器

匹配某元素 【后面的】【所有指定】兄弟元素

语法：由 ~ 充当结合符

2、属性选择器

通过元素的【属性以及值】去匹配不同的元素

语法：[]

1、[attribute]:匹配具有 atrribute属性的元素

[id] : 匹配所有有id属性的元素

div[id] : 匹配所有具备id属性的div元素

2、element[attr1][attr2]

匹配 所有 具备 attr1属性以及attr2属性的element元素

div[id][class]

3、element[attr=value]

匹配attr属性值为value的element元素

input[type=text]

input[type='text']

input[type="text"]

4、element[attr~=value]

匹配 attr属性值为一个值列表，并在在此列表中包含独立单词value的element元素

div[class~=warning]

<div class="content warning important lf"></div>

div[class~=content] : 能匹配

div[class~=lf] : 能匹配

div[class~=on] : 不能匹配

5、element[attr^=value]

匹配attr属性值，以value开始的element元素

^= : 匹配开始

div[class^=my] : 匹配class属性值以my开始的div元素

6、element[attr$=value]

匹配attr属性值，以value作为结束的element元素

$= : 匹配结束

div[class$="over"] : 匹配class属性值以over作为结束的div元素

7、element[attr\*=value]:匹配 attr属性值中【包含】value的element元素

注意：与 element[attr~=value] 的对比

~= : 匹配值列表中，包含完整单词的

\*= : 匹配值中，包含指定字符的

3、伪类选择器

伪类选择器：匹配元素的某种状态

1、目标伪类

突出显示活动的锚点元素

目标：活动(被激活)的锚点元素

语法：:target

a:target{}

div:target{}

2、元素状态伪类

多数用在表单控件上，去匹配表单控件的不同状态(启用、禁用、选中)

1、 :enabled

匹配每个已启用元素(所有表单控件)

2、 :disabled

匹配每个被禁用元素(所有表单控件)

3、 :checked

匹配每个已被选中的input元素(使用与radio和checkbox)

3、结构伪类

从标记的层次结构来匹配元素。

1、 :first-child

匹配属于其父元素中的首个子元素

table tr td:first-child

2、 :last-child

匹配属于其父元素中的最后一个子元素

3、 :empty

匹配没有子元素(包含文本内容)的元素

4、 :only-child

匹配属于其父元素中的唯一子元素

5、 :nth-child(n)

匹配属于其父元素中的第n个子元素

4、否定伪类

将匹配的元素排除在外

语法：:not(selector);

eg:

input:not([type=text]){

}

table tr td:not(:first-child){

}

4、伪元素选择器

伪元素：匹配元素的内容

伪类:匹配元素不同的状态

1、:first-letter

获取匹配元素的第一个字母(字符)

2、:first-line

获取匹配元素的首行

3、::selection

匹配用户选取的部分

:selection

::first-letter

:before

::before

W3C : 规定 所有的伪类选择器 全部使用 一个冒号 :

在CSS3中，所有的伪元素选择器，全部使用两个冒号 ::

2、内容生成

1、内容生成

1、什么是内容生成

通过css 属性，可以将 文本、图像等一些内容插入到已有的元素内容中去

2、伪元素选择器

:before / ::before

将选择器定位到指定的元素【内容】之前

:after / ::after

将选择器定位到制定的元素【内容】之后

<a href="#">百度</a>

a:before{

}

a:after{

}

3、内容生成属性

属性: content

取值：

1、普通文本 content:“我是文本”；

2、url() 函数,生成图像内容，url就是图像路径

3、计数器

2、计数器

1、生成计数器的属性

属性：

counter-reset:计数器名称 初始值;

注意：如果页面中多个元素要使用同一个计数器的话，那么要将计数器声明在 body 元素中，否则计数器永远都是初始值。

声明多个计数器:

counter-reset:计数器名称1 初始值1 计数器名称2 初始值2;

初始值，可以省略，如果省略的话，初始值为 0

body{

counter-reset:c1 c2;

}

2、设置计数器的增量

每次在使用之前，使计数器在原来值的基础之上增加(减少)多少。

如果不设置计数器的增量的话，那么每次使用时，数字都是相同的。

属性:

counter-increment:计数器名称 增量值;

注意：哪个元素用计数器，在哪个元素中设置增量

div{

counter-increment:c1 10;

}

3、使用计数器

函数：counter(计数器名称)

一般情况下，会配合内容生成属性一起使用

div:before{

content:"第"counter(c1)"章";

}

3、多列

column-count

column-gap

column-rule

4、CSSHack

1、padding 与 margin 问题

不同的浏览器，对默认的margin和padding解析方式不同

解决方案:

body{

margin:0px;

padding:0px;

}

2、居中布局

ff,chrome....通过 margin:auto方式完成块级元素居中显示

ie6及以下，主要通过 text-align:center;方式完成所有元素的居中(包含块级)

body{

margin:auto;

text-align:center;

}

3、元素高度与内容

内容高度超出定义高度后，ie6，自适应，其他浏览器，溢出。

解决方案:

overflow属性;

4、子元素设置上外边距时对父元素的影响

解决方案：

1、父元素 加 border

2、设置父元素的padding-top取代子元素的margin-top，父元素高度做相应加减。

条件注释：

gt : 选择大于条件版本以上的浏览器版本

gt ie 6 :条件是 大于 ie6 以上的浏览器版本才会识别

lt : 选择条件版本以下的浏览器版本，不包含条件版本

lt ie 6

gte : greater than or equal

lte : less than or equal

! :选择条件版本以外的所有版本

! ie 6

语法：

<!--[if 【判断】 浏览器版本]>

执行内容

<![endif]-->

eg:

<!--[if IE]>

只有在IE浏览器下，才能执行该段内容

<![endif]-->

<!--[if IE 6]>

只有在IE6浏览器下，才能执行该段内容

<![endif]-->

只有在 IE7 以上的版本才能执行的内容，条件注释?

<!--[if gt IE 7]>

执行的内容

<![endif]-->

06\_CSS3CORE 第二天 2015.12.21

1、基本数据类型

1、number

整数、小数

小数：浮点数 1.234 -》3.5E2

开辟空间：8个字节 --> 64位

2、string

字符文本

使用时需要通过 '' 或 ""进行包裹

3、boolean

布尔型

取值:true 或 false

使用场合：条件

4、null

空

5、undefined

未定义

var name="张三丰";

name = 18;

name=true;

2、打印输出

控制台 ：console.log()

页面:document.write()

对话框:window.alert() 或 alert();

3、类型转换

1、typeof() 函数

判断指定数据的类型

typeof(变量)

typeof(表达式)

typeof(数值)

表达式：由运算符连接而成的式子

var name="";

var age=18;

var result = name + age;

2、显示转换(强制转换)

1、toString()

变量.toString()

2、parseInt()

转换成整数

var parseValue = parseInt(变量/值);

var a = "123abc";

parseInt(a); --> 123

var b = "123.45";

parseInt(b); --> 123

var c = "abc123";

parseInt(c); --> NaN

var d = "abc";

parseInt(d); --> NaN

3、parseFloat()

var a = "123.45";

parseFloat(a) --> 123.45

输入提示框

window.prompt("请输入工资:","0");

函数:

是一段被预定义好，并且可以被反复使用的代码块。可以包含多条可执行的语句

函数也成为方法，或过程。其实就是一个独立的功能体。

函数只有在被调用时才会执行

如何把执行语句放在函数中：封装函数

如何去使用函数:调用函数

1、定义(封装)函数

function 函数名称(){

//函数体

//其实就是若干条的执行语句

}

//封装一个函数

1、接受用户输入的数据 window.prompt

2、打印输出 输入的数据

2、调用函数

1、在<script>中

直接使用 函数名称() 进行调用

2、在html元素中的事件里调用

onclick --> 单击事件

<input type="button" value="点我啊" onclick="函数名称()" />

事件名称="函数名称()"

练习：声明一个函数，要求在函数中能够完成 让用户分两次输入两个数字，并且能够实现两个数字的加法操作，最后打印结果。

创建一个按钮，在按钮 点击 时，执行以上操作

带参数的函数:

声明函数过程中，可以定义函数在运行过程中需要用到的参数

声明语法:

function 函数名称(参数列表){

}

参数列表：变量名称，多个值的话用 ,来区分

调用：

如果调用的函数具备参数，那么一定要传参

函数名称(具体数据);

var fruit = "苹果";

zhaZhiJi(fruit);

parseInt("25");

prompt("请输入一个数字");

alert();

分支结构：

1、什么分支结构

程序在运行过程中，根据不同的【条件】，选择执行某段语句(语句块)。

又叫选择结构。

1、当条件满足的时候运行某些语句

if 结构

2、条件满足运行一部分语句，条件不满足，运行另一部分语句

if .. else 结构

2、if 结构

语法：

if(条件){

//满足条件要做的事情

}

条件：一定是boolean类型的数值或表达式

如果条件 为 true ，则执行 {}里的内容

如果条件 为 false, 则不执行 {}里的内容(跳过大括号，继续向下执行)

比较运算符: > ,< ,>=,<=,==,!=

== : 判断等于，判断两个数值是否相等

!= : 不等于,判断两个数值是否不相等

5 == 5 : true

5 != 5 : false

3 >= 5 : false

5 >= 5 : true

以上运算符的运算结构全部为 boolean类型

= : 赋值

注意：

1、if结构中的 {} 是可以被省略的，如果省略的话，那么if只控制到它下面的第一条语句。建议不要省略{}，哪怕if结构中只有一句话。

2、有些值，放在条件中，会自动转换为false

if(0){}

if(0.0){}

if(null){}

if(undefined){}

if(NaN){}

if(""){}

if(''){}

除以上情况外，其余数值放在 条件中，都默认为真

if(123){} 真

if("Hello") 真

3、if ... else 结构

条件满足是，可以执行一个语句块，当条件不满足时，执行另外一个语句块。两个块中只能选择一个被执行

if(条件) {

//条件满足时执行

语句块1;

}else{

//条件不满足时执行

语句块2;

}

练习：

在之前程序上 ，判断用户输入的金额是否比总价高，如果比总价高的话正常进行计算(找零、提示)，否则，提示，付款金额不足 .....

问题：

输入一个考试的分数

如果分数 == 100 奖励一台兰博基尼

如果分数 >= 90 奖励一台 保时捷

如果分数 >= 80 奖励一台 奔驰

如果分数 >= 70 奖励一台 奥迪

如果分数 >= 60 奖励一台 奥拓

否则 ：奖励一 大嘴巴

多重 if 结构

一个完整的结构，允许出现多次条件判断，只能选择一个(匹配上的第一个)满足的条件去执行

语法结构:

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{}

克莱托指数：

体重(kg) / 身高(m) \* 身高(m)

指数：

< 20 偏瘦

20 < 指数 < 25 正常

> 25 偏胖

编写一个 个人所得税计算器函数，接受一个参数表示工资值，通过函数计算工资所对应的个人所得税是多少。

3500 以下 免征

3500 ~ 5000部分 缴纳 3%

5000 ~ 9000部分 缴纳 10%

9000 以上部分 缴纳 20%

06\_CSS3CORE 第三天

1、if结构

if(条件1){

语句块1;

}

if(条件2){

语句块2;

}

if(条件1){

}else if(条件2){

}else if(){

}

1、需求1

在控制台上打印 10 遍Javascript

console.log("javascript");

需求2

在控制台上打印 100 遍 Javascript

需求3

在控制台上打印 第1遍 Javascript

....

....

第100遍 Javascript

循环：重复执行 相同 或 相似的功能

生活中的循环:

1、活着

2、上学(上班)

上学： 毕业 4个月

上班： 退休

3、地球 公转 自转

循环的特点：

1、循环条件：规定了循环的开始 和 结束

2、循环操作：循环的过程中，做了什么

1、打印10遍Javascript

循环条件:从 第1遍 开始 ， 到 第10遍 结束

循环操作:打印 javascript

2、打印100遍Javascript

循环条件:从 第1遍 开始 ， 到 第100遍 结束

循环操作:打印 javascript

3、打印 100 遍 第 n 遍 Javascript

循环条件：从 第1遍 开始 ， 到 第100遍 结束

循环操作：遍数 打印 javascript

4、每打印10遍Javascript后，增加一条分割线

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

1、while循环

语法：

while(条件boolean类型表达式){

//循环操作

}

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

//声明开始的条件

while(结束条件){

//循环操作

}

注意：条件为真的时候能够执行循环体，直到条件变为假则退出循环

while ：最少 一遍 都不执行 0遍

不要让条件一直保持为真，否则会变成 死循环

练习1：

做两个函数

第一个函数 计算 1- 100之间 所有数字的和

第二个函数 计算 1 -100之间 所有偶数的和

2、提前终止循环

1、break

用于提前结束循环操作

while(i >= 1000){

...

if( ... ){

break;

}

...

}

...

让用户随意输入数字，输入 0 则退出，其余则打印，继续输入

3、

在确定循环次数的条件下 必须需要三要素

1、循环开始的条件

2、循环结束的条件

3、更新循环变量

for循环结构 --> while 循环替代品

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2：循环条件，条件为true、则执行循环体，条件为false，则跳出循环

表达式3：更新循环变量

for(var i=1;i<5;i++){

console.log(i);

}

使用场合:

1、while循环更适合使用在不固定循环次数的循环操作当中

while(true){

if(..){

break;

}

}

2、for循环更适合使用在 确定循环次数的循环操作当中

for(循环变量声明;判断条件;变量自增)

练习：

通过 for 循环语句操作

1、创建一个函数 实现 1+2+3 +....999的和

2、创建一个函数 实现 1+3+5 + .... + 999的和

for的特殊表示方式

1、省略第一个表达式

for(;表达式2;表达式3){

}

var i=0;

for(;i<100;i++){

console.log(i);

}

使用场合：循环变量不是固定的，而是通过一些计算得到。

2、省略第二个表达式

for(表达式1;;表达式3){

}

缺少表达式2 ， 则为死循环

注意：一定要在循环体中 指定循环退出的条件

3、省略第三个表达式

for(表达式1;表达式2;){

}

缺少表达式3 , 死循环,有可能不进入到循环结构中

4、for(;;){

}

死循环

在for循环之外、之内，一定要给足三个表达式

var i=0;

for(;;){

if(i == 10){

break;

}

i ++;

}

5、表达式多样化

for(初始化1,初始化2;条件1,条件2;更新1,更新2){

}

表达式1;

while(表达式2){

//循环操作

//更新循环变量

表达式3;

}

var i= 1;

while(i < 5){

console.log(i);

i ++;

}

数组：

保存 1200 名学员的 姓名

数组：一组数据 保存在一个变量里

数组元素 按照线性顺序排列，线性顺序就是 除第一个元素外，每个元素都有一个 前驱元素；除最后一个元素外，每个元素都有一个 后继元素

下标：索引，表示的是数组中元素的位置，从0 开始

定义一维数组(Array):

1、var arr1 = [];

创建一个 不包含 任何 元素的数组

2、var arr2 = ["张三丰","张翠山","张无忌"];

创建一个 包含 三个字符串元素的数组

3、var arr3 = new Array();

创建一个 不包含任何元素的数组

4、var arr4 = new Array("张无忌",28,true,"赵敏");

创建一个包含 4个元素的数组

数组属性:

1、用于获取数组中元素的个数，又称为数组长度

属性:length

下标范围:0 ~ (数组.length-1)

遍历数组:

1、通过 length 属性 来构建元素的下标

下标起始值:0

下标结束值:数组.length - 1

06\_CSS3CORE 第四天

1、DHTML 概述

DHTML :Dynamic HTML,动态HTML,允许让页面的元素产生变化。保证页面在下载后，仍然能够实时变换页面效果。

DHTML的功能：

1、动态改变页面元素

2、与用户进行交互

3、DHTML对象模型包含:BOM 和 DOM

DHTML = HTML + CSS + Javascript

2、获取页面元素

1、通过ID选取元素

document.getElementById("元素的id");

返回：得到一个DOM对象

2、通过标签名获取元素

document.getElementsByTagName("标签名");

返回：得到 由DOM元素 所组成的一个 数组

3、通过class名称获取元素

document.getElementsByClassName("class名称")

返回：得到由DOM元素所组成的一个数组

var div = document.getElementById("d1");

div.getElementById("d1\_1");

div.getElementsByTagName("span");

3、节点间关系

属性:

1、parentNode ：获取父节点

2、childNodes ：获取所有子节点(得到数组)

3、firstChild ：获取第一个子节点

4、lastChild ：获取最后一个子节点

5、previousSibling ：获取上一个兄弟节点

6、nextSibling ：获取下一个兄弟节点

7、nextElementSibling : 获取下一个【元素】节点

8、previousElementSibling : 获取上一个【元素】节点

9、children : 获取所有的子【元素】节点，不包含文本节点

4、元素内容

1、HTML内容

<div>

<span>Hello Div</span>

</div>

通过 innerHTML属性 【读取】或【设置】元素节点中的【html内容】

2、文本内容

读取或设置元素中的文本内容(不包含html)

属性：textContent IE8 及以下浏览器不支持

innerText : 火狐不支持

5、元素属性

1、属性集合

属性：attributes

返回：返回该节点所有属性的数组

<a name="" id="" href="" target="\_blank"></a>

2、读取属性值

1、element.attributes[下标].value;

2、element.attributes['属性名'].value;

3、element.getAttributeNode('属性名').value

4、element.getAttribute('属性名');

3、修改/设置属性值

1、element.setAttribute(name,value);

name ：属性名称

value : 属性值

document.getElementById("a").setAttribute("href","http://www.baidu.com");

2、element.setAttributeNode(attrNode);

//1、创建属性节点

var attr=document.createAttribute("href");

//2、为属性节点赋值

attr.nodeValue="http://www.baidu.com";

//3、将属性节点放置在 setAttributeNode()中

document.getElementById("a").setAttributeNode(attr);

4、element.属性名称

document.getElementById("a").href="http://www.baidu.com";

6、元素样式

DOM封装了针对不同的样式表的操作方式，最常用的就是 内联样式：style属性访问

1、style属性

注意：只能获取内联样式中的属性

属性名:style中的属性有连字符(-)，去横线变驼峰

css : javascript

width width

background-color backgroundColor

border-left-color borderLeftColor

获取或修改元素的内联样式：

获取:element.style.属性名;

获取的结果类型均为字符串

设置:element.style.属性名=value;

value也均为字符串类型，并且有单位的话要添加单位

element.style.width="200px";

element.style.display="none";

2、获取计算的样式

内部、外部样式表

DOM:

document.defaultView.getComputedStyle("元素对象").样式属性;

IE :元素对象.currentStyle.属性

7、修改样式表中的样式

1、获得样式表对象

var sheet = document.styleSheets[index];

2、获取样式表中所有的规则的集合

var rules = sheet.cssRules || sheet.rules;

3、获取规则数组中，想要修改的某一个规则

var rule = rules[index];

4、获取或设置 样式属性

var value = rule.style.属性名;

或

rule.style.属性名=value;

注意：使用时须谨慎。

JS第一天

张东

zhangdong@tedu.cn

回顾：

1. 什么是JavaScript：专门编写网页交互的语言

2. 什么是变量：内存中存储\*一个\*数据的存储空间，再起一个名字

声明: 创建一个变量, var 变量名;

赋值: 将等号右边的数据保存到等号左边的变量中。变量名=值;

取值: 在任何位置使用变量名等效于直接使用变量中的值

特殊：1. 简写：var 变量名=值;——建议

\*\*\*2. 声明提前：在正式执行程序前，都会预读所有var声明的变量，集中到顶部集中创建。\*\*\*赋值留在原地\*\*\*

解题：先改为声明提前后的状态，再判断输出

3. 出错：仅声明，但未赋值的变量，可以使用。

未声明的变量，可以直接赋值——强烈不建议

未声明的变量参与运算或表达式：

报错:Reference Error

3. 数据类型：2大类：

原始类型：值直接保存在变量本地的 数据类型

5个：Number String Boolean undefined null

true/false undefined null

Number: 64位=8字节，所占空间与数值大小无关

特殊：舍入误差：计算机中也有计算不尽的数值

　　　　　　　　　　解决：按指定小数位数四舍五入

　　　　　　　　　　　　　n.toFixed(d)

控制台：换行：shift+enter

切换写过的代码：上下键

清屏：点左上角圆叉号

避免新旧变量间干扰：刷新页面F5

String：一串字符组成的一句话

js中用unicode编号保存每个字符

unicode：对全球主要语言中的每个文字编一个号

查看任意字符unicode号: "字".charCodeAt(0)

内存占用：一个英文字母，数字或符号 占1字节

汉字 占2字节

特殊：字符串一旦创建无法改变，只能整体替换。

结果：每次字符串+拼接时都会浪费一个字符串

何时加""：

如果数据可能用作算数计算或比较时，不加引号-Number类型

比如：年龄 价格 数量

如果数据不可能用作算数计算或比较时，就加引号-String类型

比如：QQ 身份证 手机号

Boolean：只有两个值: true/false

何时使用：用作条件或结论时

引用类型：值无法直接保存在变量本地的 数据类型

正课：

1. \*\*\*数据类型转换

2. \*\*\*运算符和表达式

1. \*\*\*数据类型转换：

js是弱类型语言：3个特点：

1. 声明变量时，不用提前指定数据类型

2. 同一个变量，先后可保存不同类型的数据

3. js会根据自身的需要，自动转换数据类型使用

数据类型转换：2大类：

1. 隐式转换：不需要程序员干预，js自动完成的类型转换

算数运算中的隐式转换：

1. 默认都转为数字，再运算

2. +运算中，只要一个数据是字符串：

+变为字符串拼接，不是字符串的数据，被转为字符串

特殊：默认情况：bool类型：true-->1 false-->0

如果转不成数字，则转为NaN

NaN参与算数运算，结果永远为NaN

2. 强制转换：程序员主动调用函数实现的类型转换

正课：

1. \*\*\*数据类型转换

2. \*\*\*运算符和表达式

1. \*\*\*数据类型转换：

1. 隐式转换：

2. 强制转换：程序员通过调用函数完成的类型转换

任意->字符串：2个函数

var str=x.toString() x不能是null或undefined

（var a=20; a.toString(2); 把20变为2进制数）

var str=String(x) 万能——相当于隐式类型转换

一般都是自动调用，不需要手动调用

检测数据类型：typeof(x)-->返回类型名

其实String(x)的源码：

function String(x){

if(x===null){

return "null";

}else if(x===undefined){

return "undefined";

}else{

return x.toString();

}

}

补：var value=x.valueOf(),仅取出x内部的值，不做类型转换

鄙视题：区分String() toString() valueOf()

任意->数字：3个函数

任意->数字：var n=Number(x) -->相当于隐式转换

强调：Number中只能转换纯数字组成的内容

特殊：Number(null)->0 Number(undefined)->NaN

Number("")->0

何时使用：隐式转为数字时自动调用

字符串->数字：

var n=parseInt(str)

从左向右，依次读取str中的数字字符

直到碰到第一个非数字字符就退出

自动跳过开头的空字符

不认小数点，省略小数部分

var n=parseFloat(str)

认识第一个小数点

何时使用：字符串类型转数字时使用

Number vs parseXXX

对于bool类型：Number可以转，parseXXX转不了

对于结尾带单位的字符串: Number不能转换，parseXXX可以转

任意->bool: Boolean(x)——隐式转换

规则：除了0,NaN,null,undefined,""转为false外，

其余都转为true。

2. \*\*\*运算符和表达式：

程序：“人的想法”在计算机中的执行（数据+运算）

运算符：程序模拟人类想法的特殊符号

表达式：由变量，运算符和数据组成的公式

所有表达式都返回一个结果，都可当做一个值使用

1. 算数运算：

%模运算(取余数)：m%n,先m/n，取除不尽的余数部分

何时使用：1. 判断能否被整除

2. 限制运算结果的上限

++，--:

n++; ==> n=n+1; 单独使用，++写前写后都一样

如果参与其他表达式中：

都是将变量中的值+1

前++: 返回\*新\*值

后++: 返回\*旧\*值

2. 关系运算：做比较！返回bool类型结论

普通:> < >= <= != ==

隐式转换：默认都转为数字再比较

　　　　　　　如果参与比较的都是字符串，则按位PK unicode号

特殊：1. 判断是不是NaN：isNaN(num)

如果num是NaN，就返回true，否则返回false

为什么：NaN不等，不大于，不小于任何值，甚至自己

NaN和任何数据做!=比较，永远返回true

习惯用法：判断num是不是数字

!isNaN(num): 返回true，说明是数字

返回false，说明不是数字

2. null undefined

undefined: 所有变量的默认值，自动赋值

null: 主动清空一个变量(主动释放对象)

问题：用==无法判断null和undefined

解决：全等===：先数据类型必须相同，再数值相等

===不带隐式转换

何时使用：狭义：区分null和undefined时

广义：只要不希望隐式转换时，就用全等

3. 逻辑运算：将多个关系运算综合得出最终结论

做判断-->返回bool类型true/false

隐式转换：默认将所有条件转化为bool，再联合判断结果

&&:"且"，所有条件都为true，结果才为true,

只要一个条件为false，结果就为false

||:"或"，只要一个条件为true，结果就为true

只有所有条件都为false，结果才为false

! :"不"，颠倒任意关系运算和逻辑运算的结果

\*\*\*短路逻辑：只要前一个条件已经可以得出最终结论，后续条件不再执行。

利用短路逻辑：2种：

1.一个条件，一件事，满足条件才执行，不满足就什么也不做

total>=500&&(total=total\*0.8)

2.设置默认值：||

值1||值2 -->返回两值中有效的一个

4. 位运算：>> <<

左移: m<<n 将m的二进制数左移n位，相当于\*2的n次方

右移: m>>n 将m的二进制数右移n位，相当于/2的n次方

5. 扩展赋值运算：先修改变量中的值，再存回变量

+= -= \*= /= %=

m+=n --> m=m+n

6. \*\*\*三目/三元/条件运算：

什么是根据不同的条件返回不同的值/执行不同的操作

何时使用：

如何使用：console.log(

条件1?操作1/值1:

条件2?操作2/值2:

... ...:

默认操作/默认值;-->不能省！

)

三目 vs if...else

第一种情况：

如果操作简单，首选三目

如果操作复杂，用if...else

第二种情况：

如果操作只是获得不同的值，也可用三目代替if...else

第三种情况：

如果希望省略默认操作时，只能用if...else

输入一个字符是数字、字母、汉字、其他字符

var input=prompt("输入一个字符");

console.log(

input>="0"&&input<="9"?"是数字":

input>="A"&&input<="Z"

||input>="a"&&input<="z"?"是字母":

input>="\u4e00"&&input<="\u9fa5"?"是汉字":

"其它字符"

);

JS第二天

正课：

1. \*\*\*函数

2. 分支结构

1. \*\*\*函数：封装一项任务的步骤清单的代码段，再起一个名字

js中函数也是一个引用类型的对象，函数名其实也是引用函数对象的普通变量。

（function fun(){ }等效于fun=function(){ }, fun可以看成普通变量，fun=function(){ }函数 对象）

函数对象：内存中专门存储一个函数定义的存储空间

在window之外。

何时使用：一项任务可能反复使用时，就仅需要在函数中封装一次，反复调用函数即可。

如何声明函数：

function 函数名([参数1,参数2,...]){

函数体（步骤清单）

return 返回值

}

参数：专门接受函数调用时，必要数据的变量

如何使用参数：在函数内，参数变量的使用和普通变量完全一样

2个作用：1. 暗示调用者，必须的参数有哪些

2. 调用时，可接受传入的参数值

如何调用函数：调用者直接调用函数名

js引擎按照函数中步骤清单执行函数

定义时，不读取函数的内容

只有调用时，才读取函数内容，执行。

语法：函数名(参数值1,参数值2,...);

返回值：函数执行的结果

何时需要返回值：调用者是否需要有明确的执行结果

\*\*\*变量作用域：一个变量的可用范围

2种：1. 全局作用域：在window下直接声明的一切都属于全局作用域。全局作用域下的变量，叫全局变量

特点：可在任何地方被访问，可反复使用。

2. 函数作用域：在函数中创建的变量都属于函数作用域。属于函数作用域的变量，叫局部变量

特点：仅在本次函数调用时创建

函数调用完，销毁。——不可重用！

\*\*\*2种局部变量：

1. 参数变量

2. 在函数体中用var声明的变量

函数的执行过程：3个阶段：

1. 声明时：仅创建函数对象，封装步骤清单

2. 调用时：为\*本次\*函数调用创建执行环境，和活动对象。活动对象中再包含本次要用到的局部变量。

活动对象：在函数调用时，专门保存局部变量的空间

为什么叫活动对象：只有调用时创建，调用后就释放

变量的使用顺序：

如果活动对象中有，就用自己的

如果活动对象中没有，就用全局的

3. 调用后：执行环境弹出，活动对象释放。

\*\*\*按值传递：将变量作为参数传入函数时，仅将变量中的值复制一个副本给函数的参数变量。

对于原始类型的值，在函数内修改参数变量的值，不影响外部全局变量的值。

正课：

1. \*\*\*函数

2. 全局函数——了解

3. 分支结构

1. \*\*\*函数：

\*\*\*声明提前：在程序正式执行前，提前预读var声明的变量和function声明的函数，集中到\*当前作用域\*的顶部创建。赋值留在原地

2. 全局函数：ES标准中规定的，浏览器厂商已经实现的函数

不需要使用任何对象，就可直接调用

鄙视题：js的全局函数有哪些：

手册: JavaScript-->左侧JavaScript-->底部JavaScript 对象

-->JS Functions-->全局函数的列表

编码解码：

张东 编码==> %E5%BC%A0 %E4%B8%9C

多字节字符 单字节字符

问题一：URI语法中不允许出现多字节字符！

utf-8编码：一个汉字占3个字节

年货 编码==> %E5%B9%B4 %E8%B4%A7

如何编码解码：

解决：var code=encodeURI(多字节字符串)

返回编码后的单字节字符串

问题二：URI语法中不允许和保留字冲突，比如 : /

http://tmooc.cn ==> http%3A%2F%2Ftmooc.cn

解决：encodeURIComponent

decodeURIComponent

eval：执行字符串格式的代码或表达式

eg：var sc=prompt("输入算式");//"1+2-3+4\*5/6"

var result=eval(sc);

alert(result);

isFinit(num): 判断num是否在有限范围内

如果num是有效的，且在有限范围内，返回true

否则，返回false

js中/0不会报错，会返回infinity——无穷大

3. 分支结构：

1. 一个条件，一件事，满足就做(不满足就什么也不做)

如果操作简单：短路：条件&&(操作1,操作2,...)

如果操作复杂：

if(条件){

满足条件才执行的代码段

}

2. 一个条件，两件事，二选一执行

如果操作简单：三目: 条件?操作1:操作2

如果操作复杂：

if(条件){

满足条件才执行的代码段

}else{

不满足条件才执行的代码段

}

3. 多个条件，多件事，多选一执行

如果操作简单，或仅返回不同的值时：三目：

条件1?操作1/值1:

条件2?操作2/值2:

... ...:

默认操作/值;

如果操作复杂或可能一个都不执行时：

if(条件1){

满足条件1才执行的代码段

}else if(条件2){

满足条件2才执行的代码段

}...{

...

}else{

默认代码段(只有以上所有条件都不满足时才执行)（可以没有）

}

强调：只可能有一个if后的代码段被执行，后续if不再判断

（）

如果条件都是全等比较时，可用switch case代替else if结构

switch(表达式){

case 值1:

表达式的值===值1时执行的代码段

break;

case 值2:

表达式的值===值2时执行的代码段

break;

... :

...

[default:

表达式的值和以上值都不相等时执行的默认代码段]

}

switch case vs else if

switch case做全等比较，效率比else if高

else if可自定义比较条件，可解决一切问题

switch case限制条件只能是全等比较。

break：退出当前结构

何时不加break：相邻两个case执行相同代码时

JS第三天

正课：

1. \*\*\*循环

2. \*\*\*数组

1. \*\*\*循环：让程序反复执行相同代码段

三要素：1. 循环条件：可以继续进入循环的条件

2. 循环变量：循环条件中用作判断的变量

有规律：从几开始，到几结束，每次增/减几

3. 循环体：反复执行的代码段

循环变量的变化没有规律：while

var 循环变量=初始值;

while(循环条件){

循环体;

迭代变化循环变量;

}

do...while循环: 即使第一次条件不满足，也至少可以执行一次

var 循环变量=初始值;

do{

循环体;

迭代变化循环变量;

}while(循环条件);

while vs do while

如果第一次条件都满足，两者完全等效

如果第一次条件不满足，while是一次都不执行

do while至少可以执行一次

鄙视时，只看第一次条件是否满足！

循环变量的变化有规律: for

for(var 循环变量=初始值;循环条件;迭代变化循环变量){

循环体;

}

其实，for和while循环是通用的

for循环的特殊用法：

1. 声明并初始化变量部分可提前到for之前外部

2. 声明并初始化变量部分可同时声明并初始化多个变量

3. 迭代变化循环变量部分可同时修改多个变量

4. 迭代变化循环变量部分可省略或放在循环体结尾

5. 死循环：for(;;) while(true)

\*循环中的大括号不能省略！

js中的循环变量 vs Java中的循环变量

js中只有2个作用域:全局 函数

循环内的变量，出了循环可继续使用

Java中有三个作用域：全局 函数 块({})

循环内的变量，出了循环不可使用！

退出循环：2种选择：

1. 使用循环条件控制退出循环——优雅——要求高

2. 使用break随时退出循环——野蛮——随意

continue: 跳过本轮循环，继续下一轮

2. \*\*\*数组：一组变量的集合，再起一个统一的名字

内存中存储多个数据的引用类型的对象

程序=数据结构+算法

算法：程序执行的步骤，顺序，分支，循环

数据结构：数据在内存中的存储结构

好的数据结构，可极大的提高程序的执行效率

何时使用：程序中，集中分类管理多个数据时

如何使用：创建 取值 赋值

创建：5种：

1. var arr=[]; 创建空数组

2. var arr=[值1,值2,...]; 创建同时初始化数组元素

3. var arr=new Array(); 创建空数组

new: 在内存中开辟一块存储空间——创建一个新对象

Array: 数组类型名

new Array()：创建一个Array类型的新对象

4. var arr=new Array(n); 创建n个空元素的数组

5. var arr=new Array(值1,值2,...);创建同时初始化数组元素

获得数组中每个元素： 数组名[i]

数组中每个元素的用法和普通变量完全一样！

\*\*\*数组是引用类型的对象：

多个变量可同时引用同一个数组对象，通过任意一个变量修改数组，都会影响对方。

null: 原始类型的值，主动释放一个变量引用的对象

何时使用：只要使用完较大的对象，都要用null主动释放

垃圾回收器：js引擎中的一个小程序

伴随主程序执行

统计每个对象被引用的次数——引用计数器

只有一个对象不再被任何变量引用时，就会被回收

正课：

1. \*\*\*关联数组

2. \*数组API：已经实现的，直接使用的对象和函数

索引数组：下标自动生成且是数字的数组

arr.length属性始终是最大下标+1

固定套路：

1. 获得最后一个元素：arr[arr.length-1]

2. 末尾增加一个新元素：arr[arr.length]=新值

js数组特点：没有长度限制，可随时增加新元素。

可在任意下标位置增加新元素

数组会根据最大下标调整length属性

3. 缩容：减少length属性值，可减少元素个数

遍历索引数组：

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

1. 关联(hash)数组：可自定义下标名称的数组

何时使用关联数组：希望每个元素的值都有固定的意义，就可自定义下标名。

如何使用：创建 访问元素

创建：2步：

1. 创建空数组：var hash=[];

2. 向数组中添加新元素，自定义下标名: hash["下标名"]=值

关联的length属性始终是0，失效！

访问元素：hash["下标名"]——和索引数组每个元素用法相同

遍历关联数组：for in循环:依次取出hash数组中每个下标，保存在变量key中。

for(var key in hash){...}

关联数组 vs 索引数组：

\*\*\*js中一切都是关联(hash)数组：

其实索引数组底层也是关联数组，只不过索引数组的下标都是数字而已。而且索引数组的length属性有效。

为什么叫hash数组：

hash算法：将一个字符串，换算成一个尽量不重复的数值

不同的字符串，换算结果几乎不重复

同一个字符串，每次计算的结果，一定相同的。

hash数组：采用hash算法分配元素存储位置的数组。

hash算法负责将自定义的key换算成实际存储位置

关联数组查找效率极快，不需要遍历，和元素个数无关！

索引数组只能通过遍历查找元素！效率受元素个数的影响。

2. 数组API：

1. 数组 to 字符串：2个

var str=arr.toString();//输出每个元素的内容，用逗号分隔

何时使用：一般自动调用

var str=arr.join("连接符");//输出每个元素的内容，使用自定义的连接符连接。如果省略连接符，则等效于toString

固定套路：

1. 无缝拼接数组为字符串：

var chars=['H','e','l','l','o'];

var word=chars.join("");

2. 将单词，拼接成句子：

var words=['no','zuo','no','die']

var stmt=words.join(" ");

3. 将数组拼接成html元素！

\*\*\*优化：如果出现频繁拼接字符串，可用数组+join方法代替

2. 连接数组：var newArr=arr1.concat(值1,值2,arr2,...)

\*不修改原数组，而是返回新数组，必须用变量接住新数组

3. 获得子数组：var subArr=arr.slice(starti,endi+1);

starti: 开始位置的下标

endi: 结束位置的下标,含头不含尾

slice支持负数参数，如果参数为负，则自动用length-参数

4. splice：删除 插入 替换 ——直接修改原数组

1. 删除：[var deleted=]arr.splice(starti,n)

starti表示开始删除的位置, n表示删除的个数

deleted表示被删除的元素，临时组成的子数组

2. 插入：arr.splice(starti,0,值1,值2,...)

starti表示开始插入的位置，

0表示一个都不删除

值1,...:要在starti位置插入的新值。

3. [var deleted=]arr.splice(starti,n,值1,值2,...)

5. 颠倒数组的元素：arr.reverse();

JS第四天

正课：

1. \*\*\*\*数组排序

2. 栈和队列

1. \*\*\*\*数组排序：

1. 手写: 冒泡排序（手写）

2. sort方法：默认将每个元素转为字符串，再按升序排列

arr.sort();

自定义排序：2步：

1. 定义比较器函数:

比较器函数：专门比较任意两值大小的函数

规定：1. 必须有两个参数,分别代表要比较的两个值

2. 通过返回数值，说明两个参数值的大小关系

比如: a>b，就返回正数

a<b，就返回负数

a=b，就返回0

function compare(a,b){

return a-b;

}

2. 将比较器函数作为参数传入sort方法

arr.sort(compare);

原理：1. js中函数也是对象，函数也可作为参数传递

何时需要将函数作为参数？一项任务的执行过程出现不确定性时。就只能在运行时，临时传入执行步骤。

降序：颠倒比较器的正负号，可改升序为降序

正课:

1. \*数组API

栈和队列

二维数组

2. \*\*\*字符串

1. \*数组API：

栈和队列：

栈：一端封闭，只能从另一端进出的数组

FILO

何时使用：希望封闭数组一端，只能从另一端进出

如何使用：

1. 开头封闭，结尾出入栈：

入：arr.push(值);

出：var last=arr.pop();

出入栈操作不影响旧元素的下标

2. 结尾封闭，开头出入栈：

入：arr.unshift(值);

出：var first=arr.shift();

每次出入栈操作，都影响所有元素的下标

队列：只能从一端进入，必须从另一端出

FIFO

何时使用：希望只能从一端进入，从另一端出时

如何使用：

结尾入队列：arr.push(值)

开头出队列：var first=arr.shift()

concat拼接 vs unshift,push

unshift push直接修改原数组

concat不直接修改原数组，而是返回新数组对象

unshift push每次只能压入一个元素，不支持数组参数

concat可拼接任意个元素，且支持数组参数

总结：仅压入一个元素时，就用栈操作

同时拼接多个元素时，就用concat

二维数组：数组中的元素，又引用了另一个子数组

何时使用：1.只要存储横行竖列的二维数据时

2.存储上下级的包含关系

如何定义二维数组：

1. 创建同时，就初始化数组内容：

var arr=[

[元素1,元素2,....],

[元素1,元素2,....],

...

];

2. 先创建普通数组，再赋值：

var arr=[];

arr[0]=[元素1,元素2,....];

arr[1]=[元素1,元素2,....];

如何使用二维数组：

1. 访问每个元素: arr[r][c]

二维数组中，行下标r不能越界，否则报错！

r没越界，c越界，返回undefined，不报错！

2. 遍历：外层循环控制行，内存循环控制列

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

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

当前元素：arr[r][c]

}

}

2. \*\*\*字符串：由多个字符组成的只读数组

只读：所有API都不能直接修改原字符串，必须返回新字符串

数组：凡是数组中不直接修改原数组的API，字符串都能用！

比如：1. 都可用下标访问每个字符:str[i]

2. .length属性获得字符个数

3. 获取子字符串: str.slice(starti,endi+1)

1. js的内置对象：ES标准中规定的，浏览器厂商已经实现的对象

11个：

String Number Boolean ——包装类型：

Array Date Math RegExp

Error

Function Object

Global

\*\*\*包装类型：专门封装原始类型的值，提供对值进行操作的方法

为什么使用：原始类型的值不带任何操作方法，必须通过包装类型提供对原始类型操作的方法

何时使用：在试图对原始类型的值调用方法时，js会自动创建对应的包装类型对象，封装原始类型的值。

作业：100题中：1,9,11

Js第五天 字符串api 正则表达式

正课：

1. \*字符串API：

2. \*\*\*\*正则表达式：

1. \*字符串API：所有API都无法修改原字符串，都会返回新字符串

1. 大小写转换：

var newStr=str.toUpperCase();-将str中所有字母转大写

var newStr=str.toLowerCase();-将str中所有字母转小写

何时使用：大小写一视同仁时

2. 获得指定位置的字符：

var char=str[i]

var char=str.charAt(i);

获得指定位置字符的unicode号-number

var unicode=str.charCodeAt(i);

将unicode号翻译回字符正文:

var char=String.fromCharCode(unicode);

3. 检索敏感词位置：

var i=str.indexOf("敏感词"[,fromi])

从fromi位置向后找下\*一个\*敏感词出现的位置

找到：返回敏感词第一个字符的位置下标；没找到返回：-1

省略fromi,则默认从0位置开始

var i=str.lastIndexOf("敏感词"[,fromi])

从fromi位置向前找上\*一个\*敏感词出现的位置

找到：返回敏感词第一个字符的位置下标；没找到：-1

省略fromi,则默认从结尾位置开始

4. 获取子字符串：

var subStr=str.slice(starti[,endi+1]) 支持负数参数

var subStr=str.substring(starti[,endi+1])不支持负数参数

str.slice(-4,-2)

str.substring(length-4,length-2)

都可省略第二个参数，如果省略，表示一直截取到结尾

var subStr=str.substr(starti,n)

从starti位置开始，截取n个。

5. 切割字符串：

var arr=str.split("分割符");

切割后的数组中不包含分隔符的

固定套路：1. 将字符串打散成字符数组

var arr=str.split("")

2. 将英文句子，切割为单词

var arr=str.split(" ");

正课：

1. \*\*\*\*正则表达式:

2. \*\*\*String中支持正则的API:

1. 正则表达式：规定字符串中字符出现规律的规则。

何时使用：查找 验证 替换 切割

模糊查找，验证字符串格式，模糊替换，按照一定规律切割字符串

正则表达式语法：

1. 最简单正则：关键字正文

2. 备选字符集：字符串中某\*一位\*上可选的字符的集合

[备选字符列表]

强调：一个备选字符集，只能规定一位字符

如果备选字符列表是(部分)连续的：用-省略中间字符

仅表示\*除别字符外\*，所有字符均可：^除掉字符——了解

3. 预定义字符集：对常用字符集的简写方式

\d：1位数字 [0-9]

\w: 1位数字，字母或下划线 [0-9a-zA-Z\_]

\s: 1位空字符：空格 tab 回车换行

.: 1位任意字符，除回车换行外

预定义字符集 vs 自定义字符集

预定义字符集语法简单，规则是固定的，不可变！

自定义字符集，规则完全自定义，灵活！语法繁琐

4. 数量词：规定一个字符集出现次数的量词

如何使用：紧跟字符集之后，修改相邻的前一个字符集

1. 确定数量：3种：

{m,n} 最少m个，最多n个

{m,} m个以上

{m} 必须m个

2. 不确定数量：3种：

?: 可有可无，最多一个

\*: 可有可无，多了不限

+: 至少一个，多了不限

5. 选择和分组：

分组： (子表达式)

选择：或 表达式1|表达式2

手机号：+86或0086 可有可无，最多一个

空字符 可有可无，多了不限

1

3,4,5,7,8 选其一

9位数字，不能多，不能少

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

身份证号：15位数字

2位数字

从数字、x、X中选一个

后两部分整体可有可无，最多一次

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

匹配img元素：<img xxxx src="" xxxx />

<img 1位以上任意字符 src= 在'"间选一个 1位以上任意字符 在'"间选一个 1位以上任意字符 />

贪婪模式：默认匹配最长的字符

比如: .\* .+

懒惰模式：仅匹配最短的字符

贪婪改懒惰：.\*=>(.\*?) .+==>(.+?)

<img(.+?)src=['"](.+?)['"](.+?)/>

6. 指定匹配位置：

^：开头位置的xxx，专用在表达式开头

$：结尾位置的xxx，专用在表达式结尾

比如：^\s+ 匹配开始位置的所有空字符

\s+$ 匹配结尾位置的所有空字符

^\s+|\s+$ 匹配开头或结尾的空字符

前加^后加$(不带|):表示从头到尾完整匹配

今后只要验证格式，都要前加^，后加$

比如：验证密码：6位数字：^\d{6}$

\b:单词边界，每个单词前或后，必须有空字符分割——了解

预判：在正式匹配前，使用特定条件先预检查一次字符串

如果预判都未通过，则不再匹配

6位以上字母，数字的组合，至少包含一位大写字母和一位数字

1. 预判：不都由小写字母或数字组成

结论：可能包含大写字母，特殊符号

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

2. 预判：不都由大写字母或小写字母组成

结论：可能包含数字，特殊符号

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

3. 6位以上字母，数字的组合(排除特殊字符)

结论：肯定包含大写字母和数字

[a-zA-Z0-9]{8}

(?![a-z0-9]+$)(?![A-Za-z]+$)[a-zA-Z0-9]{8}

补充：汉字的备选字符集:[\u4e00-\u9fa5]

2. String中支持正则的API：

1. 查找：4种：

1. var i=str.indexOf/lastIndexOf("敏感词"[,fromi])

缺点：不支持正则表达式，每次只能找一个敏感词

优点：可指定开始查找的位置

何时使用：敏感词固定时，且仅获得位置时

2. var i=str.search(/正则表达式/)——不支持后缀g

缺：仅查找\*第一个\*和正则表达式匹配的关键词

仅返回关键词第一个字符的位置下标

如果没找到，返回-1.

无法返回关键词内容。

优：支持正则

何时使用: 需要使用正则表达式查找关键字时，且只判断有没有。

3. var arr=str.match(/正则表达式/g)

缺：可返回所有匹配关键词的内容，

无法获得每个关键词的位置。

优：找到所有关键词

默认正则表达式只匹配一个关键词

表达式后+g,让表达式匹配所有关键词

\*\*\*找不到，返回null!

都要先判断不是null，再调用属性或方法

4. 即可获得关键字内容，又可获得位置：reg.exec（待续）

2. 替换：var newStr=str.replace(/正则表达式/g,"替换内容")

作业：100题：14,16,18

Js第六天 String正则API RegExp对象 Math Date

正课：

1. \*\*\*String正则API

2. \*\*\*RegExp对象

3. Math

4. \*Date

1. \*\*\*String正则API

1. 查找：4种：

1. 固定关键词(不支持正则)，只能找位置：

var i=str.indexOf/lastIndexOf("敏感词"[,fromi])

2. 支持正则，只能判断有没有:

var i=str.search(/正则表达式/)

3. 支持正则，获得所有匹配的敏感词内容，无法获得每个位置

var arr=str.match(/正则表达式/g)

4. 支持正则，即可获得每个敏感词内容，又可获得位置:

var arr=reg.exec(str);

2. 替换：

var newStr=str.replace(/正则表达式/g,"替换内容");

删除：将匹配内容替换为""

比如：去掉字符串开头和结尾的空字符

格式化：2步：

1. 先定义正则表达式，用()分组

每个()，都会得到一个序号:$1,$2,...

2. 在replace中的替换内容部分，使用$n拼接结果

匹配时，会自动将$n换成本次匹配的实际内容

比如: var birth="19831226";

birth=birth.replace(

/(\d{4})(\d{2})(\d{2})/,"$1年$2月$3日");

3. 切割：

var arr=str.split(/分隔符正则表达式/);

2. \*\*\*RegExp对象：正则表达式对象

封装了一个正则表达式，提供了操作正则表达式的方法

创建正则表达式：2种：

1. 创建固定正则表达式对象：

var reg=/正则表达式/g;

何时使用：如果正则表达式不需要动态生成时

2. 创建可变正则表达式对象：

var reg=new RegExp("正则表达式","后缀");

何时使用：如果正则表达式由变量或字符串动态生成时

强调：所有\都要写成\\。比如："\d{6}"-->"\\{6}"

后缀：g: global 全局/全部

i: 忽略大小写

正则表达式对象的方法：2个：

1. 查找的第四种方法：即可找每个敏感词内容，又可获得位置

var arr=reg.exec(str);

每次只能找下\*一个\*敏感词，自动调整下次开始的位置！

如果找到返回一个数组对象：

数组的第一个元素保存本次找到的敏感词内容

数组的index属性保存本次找到的敏感词的位置

如果没找到，返回null！

RegExp.$n : 仅获得本次匹配中的$n部分

正课：

1. \*\*\*RegExp对象

验证

2. Math

3. Date

正则表达式: 查找(4种) 替换(删除/格式化) 切割 验证

1. \*\*\*RegExp对象：

验证：var bool=reg.test(str)

str符合reg的规则，就返回true，否则返回false

2. Math对象：仅专门提供数学计算的方法

Math对象没有构造函数，所以不能new！——唯一一个

所有API都直接通过Math类型名调用

1. 取整:

上取整：只要超过，就取下一个整数

Math.ceil(num)

下取整：去掉小数部分，取整数部分

Math.floor(num)

四舍五入取整：小数部分够5就进1，不够就舍掉

var num=Math.round(num)

缺点：只能取整

var str=num.toFixed(d); d:0-20之间

缺点：返回字符串，不能直接用于计算

自定义round方法：3步：

function round(num,d){

num\*10的d次方倍

Math.round(num.toFixed(2))

num/10的d次方倍

返回number

}

2. 乘方和开平方

乘方: Math.pow(底数,幂)

开平方: Math.sqrt(num);

3. 最大值和最小值

Math.max(值1,值2,...)

Math.min(值1,值2,...)

如何获取数组中的最大值：固定套路

Math.max.apply(Math,arr);

4. 随机数:Math.random() 0<=r<1的小数

任意min~max之间取随机整数

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

如果min从0开始,可简写为：

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

Js第七天 Date对象

正课：

1. \*Date对象：

1. \*Date对象：封装一个时间值，提供对时间的操作方法

Date对象中封装了1970年1月1日0点至今的毫秒数

何时使用：计算日期前，都要创建或获得日期对象

创建：4种：

1.var now=new Date();//创建日期对象,同时封装客户端系统时间

now.getTime();//获得日期对象中封装的毫秒数

2.var date=new Date("yyyy/mm/dd[ hh:MM:ss]");自定义时间点

3.var date=new Date(yyyy,mm,dd,hh,MM,ss);——了解

\*\*\*date对象中，月数从0~11结束。现实中的月份要-1修正

从日期对象中取出的月份都要+1修正

4.复制一个日期对象：在计算之前，将旧日期对象中的毫秒数取出，放入一个新的日期对象中保存副本。

Why：所有日期API都直接修改原日期对象，无法保留旧值

var old=new Date("yyyy/mm/dd");

var newDate=new Date(old.getTime());

结果：修改newDate，old不会受影响

日期API：手册：JavaScript->JavaScript 对象->JS Date

分量：FullYear Month Date Day(星期)

Hours Minutes Seconds Milliseconds

1. 每个分量都有一对儿get/set方法

比如:getFullYear,setFullYear

getMonth，setMonth

...

var 值=date.getXXX()，返回指定分量的数值

date.setXXX(值)，修改一个分量的值

自动调整时间进制

\*\*\*例外：Day没有set方法

2. 命名：年月日星期，不带s结尾

时分秒毫秒，带s结尾

比如：date.setDate(n)

date.setMinutes(n)

3. 返回值：只有Date，从1开始到31结束

其余都是从0开始到进制-1结束

比如：Hours 0~23 ——和现实中一样，不需要修正

Minutes 0~59 ——不需要修正

\*\*\*Month 0~11 ——比现实中少1，需要+1/-1修正

Day 0~6 ——和现实中一样，不需要修正

日 一...六

0 1 6

日期计算：2种：

1. 使用毫秒值直接计算：只适用于计算一天以内的时间

两日期对象可直接相减=毫秒差

利用毫秒差/进制获得时间差

2. 使用API计算：可计算任意范围内的任意时间

3步：

1. 取分量: var n=date.getXXX();

2. 做加减: n+=?;

3. 放回去: date.setXXX(n);

整合为一步:

date.setXXX(date.getXXX()+?);

问题：直接修改date值，原日期值丢了

解决：先复制日期对象的副本，再用新对象计算新值

日期转字符串：

1. toString()输出的是完整的标准日期时间格式，带时区

2. toLocaleString()输出浏览器当地时间的完整格式

3. toLocaleDateString()仅输出浏览器当地时间的日期部分

4. toLocaleTimeString()仅输出浏览器当地时间的时间部分

Js第八天 错误处理 函数对象 闭包

正课：

1. \*错误处理

2. \*\*\*\*函数对象

\*\*\*\*\*闭包

1. \*错误处理

错误：导致程序无法正常执行的异常状态

发生错误，程序立刻退出

错误对象：Error 封装错误信息的对象，在发生错误时自动创建

6种：

ReferenceError:找不到对象时

TypeError:错误的使用了类型或对象的方法时

RangeError:使用内置对象的方法时，参数超范围

SyntaxError:语法写错了

EvalError:错误的使用了Eval

URIError:URI错误

错误处理：即使程序发生错误，也保证不异常中断的机制

如何错误处理：

try{

可能发生错误的代码

}catch(err){

只有发生错误时才执行的代码——错误处理代码

}[finally{

无论是否出错，肯定都要执行的代码——善后工作(保存，释放

}]

问题：使用try包裹的代码，即使不出错，效率也比不用try包裹的代码低。

建议：在try块中，尽量少的包含可能出错的代码。

优化：尽量使用if判断代替try catch

无法提前预知错误类型的错误，必须用try catch捕获。

抛出自定义错误：

何时：如果函数的定义者，需要告知调用者使用过程中的错误

如何：throw new Error("参数必须是数字！");

2. \*\*\*\*函数对象：

js中的函数也是一个对象，封装了一项任务的步骤清单代码段，起一个名字。

函数名其实只是引用函数对象的一个变量

arguments对象：接收所有传入函数的参数值的\*类数组对象\*

默认所有函数都会自动创建，直接可用。

类数组对象：长得像数组的\*对象\*,不是数组类型的实例(对象)

arguments instanceof Array -->false

[] instanceof Array --> true

obj instanceof 类型名:专门判断obj是否是指定类型的实例

\*\*\*鄙视题：类数组对象 vs 数组: 类型不相同

相同点:

都可用下标访问每个元素

都有length属性，记录元素个数

不同点：

数组对象的类型是Array，类数组对象的类型是Object

不能直接调用数组API！

其实类数组对象可转为标准数组对象：

arguments=Array.prototype.slice.call(arguments);

\*\*\*重载overload: 程序中可同时定义多个同名，但不同参数的函数

调用时，可自动根据传入的参数不同，调用不同的函数

优点：方便调用！减少记忆的API个数

js在语法上不支持重载！但可用arguments模拟实现重载效果！

自定义参数变量的作用：1. 提醒调用者，如何正确使用函数

2. 使用比arguments简单

总结：函数尽量定义参数

创建函数对象：3种：

1. 声明方式创建：

\*\*\*只有以声明方式创建的函数才能被hoist（声明提前）

function 函数名(参数列表){函数体;return 返回值};

2. 函数直接量：

var 函数名=function (参数列表){函数体;return 返回值};

3. 实例化对象: 参数名和函数体都是字符串

var 函数名=new Function(

"参数名1","参数名2",...,"函数体; return 返回值");

鄙视题：js中创建函数的方式正确的是

正课：

1. \*\*\*匿名函数

2. \*\*\*\*\*闭包

1. \*\*\*匿名函数：没有函数名的函数定义

没有任何变量引用

何时：只要一个函数只执行一次时，就用匿名函数。

执行完，立刻释放！

为什么使用：匿名函数没有变量引用，调用完自动释放！

缺点：不可重用！

如何使用：2种：

1. 匿名函数自调: 定义后立刻调用

何时使用：只要一个函数只执行一次时

如何使用：(function([参数列表]){

函数体;

return 返回值

}([参数值列表]));

2. 回调：将函数作为对象传递给其他函数，由其它函数调用

2. \*\*\*\*\*闭包:

作用域: 一个变量的可用范围：2种:

1. 全局作用域:window

2. 函数作用域:Activation Object

作用域链: 多个作用域上下引用组成的链式结构

开头：当前函数的EC中的scope chain属性

scope chain->AO->window

结尾: 全局对象window

闭包: 实现可重用的局部变量，且保护其不受污染的机制

问题：函数作用域: 不可重用!

全局作用域: 极易被污染!

何时使用：反复使用一个变量，又不希望被污染时

如何使用：3特点：

1. 外层函数包裹受保护的变量和内层函数——函数嵌套

2. 内层函数专门负责操作外层函数保护的局部变量

3. 将内层函数返回到外层函数外部，反复调用

解题：1. 外层函数调用了几次，就有几个受保护的局部变量副本

2. 同一次外层函数调用返回的多个内层函数，共同同一个局部变量。

Js第九天 面向对象 继承 原型和原型链

正课：

1. \*\*\*面向对象：

1. 什么是面向对象

2. 创建对象

3. \*\*\*\*\*this

4. \*\*\*\*原型和原型链

5. \*\*\*\*\*继承

1. 什么是面向对象：程序中都是用对象结构描述现实中一个事物

为什么：现实中一个数据或一个功能都要包含在一个具体事物上才有意义！

何时使用：所有数据和功能都应该先包含在对象中，再使用！

对象：内存中存储多个数据的存储空间——存储

描述现实中一个事物的属性值和功能的程序结构——用法

属性：在对象中，描述当前对象某个特性值的一个变量

方法：在对象中，提供一项功能的函数

其实就是引用函数对象的属性而已

成员: 属性和方法统称为对象的成员！

\*\*\*封装：将属性变量和方法，集中定义在一个对象中

优点：1. 代码重用！

2. 便于维护！

2. 创建自定义对象：3种：

1. 创建一个单独的对象：2种：

1. 对象直接量: var obj={//强调: 属性和方法名都是字符串

// 成员名其实可省略引号，当含有特殊字符时不能省

"属性名":属性值,//强调:逗号分隔

...:...,

"方法名":function(...){

...

return ...;

},

...//强调:结尾不加逗号

}

访问对象中的属性和方法: . 读作"的"

\*\*\*\*\*对象的底层其实就是一个hash数组

\*\*\*\*\*js底层一切都是hash数组

相同：

1. 遍历对象中所有属性值：

for(var key in obj){

console.log(key+":"+obj[key])

}

2. 随时可向对象中增加新属性：

不同：类型不同：

\*\*\*对象的方法中如何访问当前对象自己的属性：

this. --> "当前对象的"

何时使用：只要对象的方法中使用当前对象自己的属性

就必须用this.属性名

2. 使用实例化新对象: 2步:

1. 创建空对象：var obj=new Object();//或{}

2. 向空对象中增加新属性:

obj.成员名=值;

或 obj["成员名"]=值;

2. 反复创建相同结构的对象：2步：

1. 先定义构造函数: （设计图）

构造函数: 描述创建一个对象的结构的函数

function 构造函数名/类型名(参数变量){

this.属性名=参数变量;

...

this.方法名=function(){

...

}

}

2. 使用new关键字，创建对象同时，调用构造函数

var obj=new 构造函数名(属性值列表)

\*\*\*\*\*什么是this：EC中记录当前正在调用函数的对象的属性

(如果有”. “this指”.”前的对象

如果没有”.”this指window)

this和定义在哪儿无关！

this只和调用时使用的对象有关！

new：4步:

1. 创建空对象: new Object()

2. 使用新对象，调用构造函数

(new Object()).构造函数();//this-->刚创建的空对象

3. ?

4. 将新对象地址，保存到等号左边的变量中

正课：

1. \*\*\*面向对象：

\*\*\*\*原型和原型链

\*\*\*\*\*继承

1. \*\*\*\*原型和原型链:

原型：专门保存所有子对象共有属性值和方法的对象

每个函数对象都有一个prototype属性，指向一个原型对象

在创建新子对象时，会自动设置子对象的原型属性(\_\_proto\_\_)为构造函数的原型对象。——子对象继承构造函数的原型

何时使用：今后只要所有子对象共有的成员值，都要保存在构造函数的原型对象中。——一次定义，反复使用

优点：1. 节省内存空间

2. 一次修改，处处生效——便于维护！

new：4步:

1. 创建空对象: new Object()

2. 使用新对象，调用构造函数

(new Object()).构造函数();//this-->刚创建的空对象

3. 设置新对象的\_\_proto\_\_指向构造函数的prototype对象

4. 将新对象地址，保存到等号左边的变量中

继承：子对象可以直接使用父对象中的成员。

所有对象都有\_\_proto\_\_属性

原型链：所有对象的\_\_proto\_\_自底向上连续继承形成的链式结构

访问一个对象的成员时，先在对象本地查找。

如果找不到会延原型链向父对象查找，直到找到为止。

原型相关API：

1. 获得原型对象: 2种：

1. 从构造函数获得原型对象:

构造函数名.prototype

2. 从子对象获得父级原型对象:

子对象.\_\_proto\_\_——有兼容性问题，不建议使用

var proto=Object.getPrototypeOf(子对象)

get 子对象 的 prototype

prototype vs \_\_proto\_\_

相同: 其实都指向同一个原型对象

不同: prototype,属于构造函数

\_\_proto\_\_,属于子对象

2. 判断父对象是否在子对象的原型链上:

父对象.isPrototypeOf(子对象)

父对象 是 子对象 的 prototype吗？

如果父对象在原型链上，就返回true，否则返回false

强调：不仅找直接父对象，且可在整个原型链上查找。

何时使用：专门用于判断子对象的类型

比如: 判断一个对象是否是数组:

反例: typeof({}) typeof([]) 都返回object

Array.prototype.isPrototypeOf({}) false

Array.prototype.isPrototypeOf([]) true

3. 自有属性和共有属性：

自有属性: 直接保存在对象本地的属性

共有属性: 保存在原型对象中，所有子对象共有的属性

如何判断一个属性是否可用：解决浏览器兼容性问题

1. in关键字："属性名" in 对象

在对象的\*原型链\*上查找指定"属性名"

找到，返回true，否则返回false

2. 判断对象的属性是否有效：

if(对象.属性名===undefined)

解决浏览器兼容性问题：看课堂练习1\_prototype.html

精确判断一个属性是自有还是共有:

1. 判断自有: obj.hasOwnProperty("属性名")

判断"属性名"是否是obj对象的自有属性

如果是，返回true，否则返回false

2. 判断共有: 自己没有，但是原型链上有

if(!obj.hasOwnProperty("属性名")&&"属性名" in obj)

4. 删除对象的属性：delete 对象.属性

只能删除自有属性，不允许删除共有属性

5. 如何修改原型对象中的属性：

共有属性\*只能\*通过原型对象修改！

Student.prototype.className="六年级2班" 正确

试图通过子对象修改共有属性，会在子对象本地创建共有属性的副本。

lilei.className="六年级2班" 在李雷本次创建className

从此李雷和共有className再无关系。

4. \*\*\*\*\*继承：

面向对象三大特点：封装 继承 多态

封装：将属性和方法集中定义在一个对象中

继承：父对象中的成员，子对象可直接使用

多态：统一个事物，在不同情况下表现出不同的状态

重载overload 重写override

重写：子对象觉得父对象的方法不好用！可在本地自定义同名方法，覆盖父对象的方法。

实现自定义继承关系：3种

为什么继承: 代码重用！

修改子对象的\_\_proto\_\_引用的对象

js中的一切继承关系都是由原型(\_\_proto\_\_)属性实现的

1. 一个对象，只继承自另一个对象：

Object.setPrototypeOf(子对象,父对象)

set 子对象 的 原型 指向 父对象

继承

2. 修改构造函数的原型对象：创建新对象之前修改

保证之后创建的所有子对象都继承同一个新父对象

构造函数.prototype=新父对象

3. 继承原型的同时，继承结构: 最像Java的继承

2步：

1. 将子类型的原型对象继承父类型的原型对象

2. 调用父类型构造函数,

同时强行指定父类型构造函数的this！

构造函数.call(this的值,参数1,...)

call在调用函数时，临时更换EC中的this引用的对象

Js第十天 面向对象 继承 原型和原型链 Bind和严格模式

回顾：

1. \*\*\*\*\*继承：

3. 继承原型，扩展结构：2步：

inherit extends

1. 设置子类型的原型对象，继承父类型的原型对象

2. 在子类型构造函数开始位置，用call调用父类型构造函数

2. call和apply：

都是调用一个函数的同时，临时修改EC中的this值为新对象

如果不考虑传入参数，则二者完全相同！

语法: 函数.call(替换this的对象,参数值1,....)

函数.apply(替换this的对象,[参数值1,...])

差别：call，参数只能单独传入

何时使用：不需要传入参数或传入参数较少时

apply, 参数可以以数组方式传入

何时使用：传入参数较多，且可能发生变化时

正课：

1. ES5 特性

Object的新特性

数组的增强API

1个函数的API:bind

严格模式

1. Object的新特性：对象其实就是属性的集合

属性：命名属性：一切自定义的属性

内部属性：对象中自带，甚至不能直接访问的属性

命名属性：数据属性：专门保存一个数据的属性

访问器属性：专门保护一个属性的特殊属性

数据属性：每个属性(Property)其实都有4个特性(Attribute)

value: 专门保存属性值

writable: true/false 规定属性值能否被修改

enumerable: true/false 规定for in中能否遍历到该属性

configurable: true/false 规定该属性能否被delete删除

以及能否修改除value外的其他特性

一旦改为false，不可逆

API：获取指定对象的，指定属性的 特性(对象)

var descriptor=

Object.getOwnPropertyDescriptor(对象,"属性名");

修改属性的特性，只能用defineProperty方法

Object.defineProperty(对象,"属性名",{

value:?,

writable:?,

enumerable:?,

configurable:?

})

如果属性已存在，则修改旧属性；如果不存在，则创建新属性

鄙视题：clone新对象——见课堂练习

如果一个属性，禁用遍历，则无法被克隆到新对象

访问器属性：专门保护一个属性的特殊属性：

访问器属性，本身没有value和writable特性——不存值

保存一对儿函数：get/set

自动调用：

get: 专门负责获得受保护的属性值

当试图从访问器属性中取值时，自动调用

set: 专门设置受保护的属性值

当设置访问器属性的值时，自动调用

访问器属性的特性：

{

get:function(){return 属性值;},

set:function(value){

if(验证value)

throw new Error(...);

else

属性名=value;

},

enumerable:true/false,

configurable:true/false

}

何时使用：如果希望保护一个变量不被随便赋值或取值时

如何添加：必须用defineProperty

正课：

1. Object对象

2. 数组API

3. bind

4. 严格模式

1. Object对象：

一次定义多个属性及其特性：

Object.defineProperties(对象,{

属性1:{特性1:特性值,...},

...

});

\*\*\*凡是新添加的属性，除value外，其余三大特性默认都为false！

内部属性：对象自带的属性，甚至不能从外部直接用.访问

\_\_proto\_\_: 引用当前对象的父对象，实现继承

extensible: 可随时向对象中添加新属性，默认为true

如果改为false，不能为对象添加新属性

但仍可修改旧属性

class: 当前对象的类型名(构造函数名)

强行调用Object的原型中的原始toString方法！

Object.prototype.toString.call(要查看的对象)

Object构造函数: var obj=new Object();

1. var obj=Object(); ==> new Object(); new可省略

2. 将原始类型的值封装在包装类型对象中:

var obj=Object(100); ==>new Number(100);

obj ==> Number{PrimitiveValue:100}

\*\*\*\*鄙视题 Object.create(): 基于一个父\*对象\*，创建并扩展一个子对象

var newObj=Object.create(父对象,{

新属性:{四大特性},

...

})

//1. 创建一个新对象

var newObj=new Object();

//2. 设置新对象继承父对象

newObj.\_\_proto\_\_=父对象;

//3. 为新对象扩展新属性，及其四大特性

Object.defineProperties(newObj,{

新属性:{四大特性},

...

});

防篡改：3个层次：

1. 禁止扩展：不允许添加新的属性

还是可以修改旧属性的特性

内部:extensible 表示当前对象是否可扩展新属性

判断当前对象是否可扩展？

Object.isExtensible(要检测的对象)

其实就是返回extensible的值(true/false)

如何禁止扩展：

Object.preventExtensions(要禁止扩展的对象)

2. 密封：不允许添加新的属性：Object.preventExtensions()

不可以修改旧属性的特性

不可以删除旧属性

将所有属性的configurable改为false

仅可以修改属性值

判断对象是否密封:Object.isSealed(被检查对象)

如果密封，返回true，否则返回false

默认是不密封，false。

如何密封对象:Object.seal(要密封的对象)

3. 冻结：禁止修改对象的任何属性及属性值

判断对象是否被冻结:

Object.isFrozen(被检查对象);

如果没有被冻结，返回false，否则返回true

默认是false

如何冻结对象：

Object.freeze(要冻结的对象);

2. 数组API：

1.\*\*\*鄙视题：判断一个对象是不是数组：几种办法

typeof无法区分对象和数组：typeof [] object

typeof {} object

1. obj instanceof 构造函数名/类型名

2. Array.prototype.isPrototypeof(obj)

\*3. Object.prototype.toString.call(obj) [object Array]

4. ES5: Array.isArray(obj)

obj是数组，则返回true，否则返回false！

判断并实现自定义isArray方法: 见课堂练习

2. 迭代API;依次操作数组中每个元素的API

1. every: 判断数组中每个元素是否都满足条件

只有都满足条件才返回true，

只要有一个不满足就返回false

语法：arr.every(function(value,index,array){

return 条件

})

some: 判断数组中是否\*包含\*满足条件的元素

只要有一个满足就返回true

只有都不满足时才返回false

2. forEach: 对\*原数组\*中每个元素执行相同操作

强调：直接修改原数组！

参数：同every，不需要return！

arr.forEach(function(value,index,array){

操作当前元素

})

\*map: 将原数组中每个元素，经过加工，生成一个\*新数组\*

var newArr=arr.map(function(value,index,array){

return 要放入新数组的值

})

作业：查找 map的源代码，鄙视题：map的实现原理

3. 过滤: filter: 查找数组中满足条件的元素，返回新数组

var subArr=arr.filter(function(value,index,array){

return 对当前值的判断结果

})

只有条件返回true的元素，才保留

4. 汇总：reduce(): 从头开始，依次汇总每个元素

返回一个最终值！

var result=arr.reduce(function(prev,curr,index,array){

return 将curr累加到prev上

}[,初始prev])

正课：

1. \*\*\*bind方法

2. \*严格模式:

1. fun.bind方法: 基于现有fun函数对象，创建一个新的函数对象

\*\*\*同时封装一个固定的this值

——提前绑定this！

var newFun=fun.bind(替代this的对象[,参数值列表])

newFun(); //this-->替代this的对象

\*\*\*bind的实现原理：见课堂练习

补：为什么要用this：

如果在方法中写死对象名(变量名)——紧耦合

结果：对象名变化，要手动修改方法内容

var lilei={sname:"Li Lei",

intrSelf:function(){

console.log("I'm "+lilei.sname);

}

}

var lil=lilei;

lilei=null;

lil.intrSelf();//报错：找不到null的sname属性

this-->动态获得调用函数时的对象——松耦合——必须！

结果：对象名(变量名)发生变化，不用修改方法内容！

2. \*严格模式：比普通程序执行，要求更严格的一套全新的执行规定

严格模式和普通模式是完全不同的执行方式

为什么：消除陷阱——静默失败：

为将来新的更广泛的ES标准打基础

如何启用：2个范围：一个<script>内

一个函数内

"use strict";——代码段之前！

\*\*\*要求：1. 禁止对未声明过的变量赋值！

2. 将静默失败升级为错误!

3. 新增了eval作用域，在eval中声明的变量，eval结束后不能访问！

4. 局部变量改变，不影响arguments中的原参数值

Exercise Day09：\*\*\*\*\*第一题

Js DOM 第一天

正课：

1. DOM概述

2. \*\*\*DOM Tree

\*\*\*\*\*递归遍历DOM Tree

3. \*查找

1. DOM概述：

JavaScript=ECMAScript(核心语法)+

DOM(专门操作网页内容的API)+ 3

BOM(专门操作浏览器窗口的API) 2

DHTML: 所有实现网页动态效果的技术的统称 —— 宣传词汇

DHTML=HTML+CSS+JS

鄙视题: HTML XHTML DHTML XML

HTML: 超文本标记语言，专门编写网页内容的语言

XHTML: 严格的HTML语言标准

DHTML: 所有实现网页动态效果的技术的统称

XML: 可扩展标记语言，标签可自定义的

<student><!--自描述-->

<sname>杨幂</sname>

<math>81</math>

<chs>53</chs>

<eng>89</eng>

</student>

JSON: JavaScript Object Notation ——互联网中取代XML

"{'sname':'杨幂','math':81,'chs':53,'eng':89}"

DOM：操作一切结构化文档的API

即可操作XML，又可操作HTML

核心DOM: 可以兼容所有结构化文档的API

可执行所有操作: 查找，修改，插入，删除 CRUD

缺点：繁琐

HTML DOM: 基于核心DOM的API，扩展的专门操作网页的简化版API

缺点：不能完成所有CRUD操作：

更侧重于修改

查找，插入和删除还要靠核心DOM补充

2. \*\*\*DOM Tree: html文档内容加载进内存后，由所有节点对象组成的树形结构。

网页加载过程：

下载HTML-->DOM Tree——（jQuery加载）

|

附加-->\*\*\*layout\*\*\*-->Render Tree-->paint

| |

下载CSS -->CSSRules |

|

onload

网页中一切内容都是节点对象：

document对象是根节点

其他所有对象都是document的子代节点

节点对象的类型：4大类：

document节点：文档类型(Document)

元素节点：元素类型(Element)

文本节点：文本类型(Text)

属性节点：属性类型(Attr)——通常属性不当做节点看

节点对象的父类型：Node (所有节点对象都继承自Node.prototype)

三大属性：

1. node.nodeType: 获得当前节点对象的类型

返回一个数字，区分节点类型

ELEMENT\_NODE: 1

ATTRIBUTE\_NODE: 2

TEXT\_NODE: 3

COMMENT\_NODE: 8

DOCUMENT\_NODE: 9

2. node.nodeName: 获得当前节点的名称：

1. 元素节点：标签名 \*\*\*全大写\*\*\*

2. 文本节点：#text

3. document: #document

比如：判断一个元素是否是button

if(node.nodeName=="BUTTON")

3. node.nodeValue: 获得当前节点的值:

1. 元素节点：返回null - 无效

2. 文本节点：返回文本的内容

DOM Tree：6种关系：（节点树：包含所有节点）

1. 父子关系: node.parentNode - 每个节点只可能有一个父节点

node.childNodes - 返回直接子节点的集合

node.firstChild - node下第一个子节点

node.lastChild - node下最后一个子节点

2. 兄弟关系: node.previousSibling - 前一个平级的兄弟节点

node.nextSibling - 后一个平级的兄弟节点

除parentNode外，都会受看不见的空格，换行，制表符的干扰

元素树：（IE9+）仅包含元素节点的树结构 - 节点树的子集

1. 父子关系: node.parentElementNode

node.children - IE8兼容！

node.firstElementChild

node.lastElementChild

2. 兄弟关系: node.previousElementSibling

node.nextElementSibling

快捷通道：document.body 直接获得body节点

DOM API原型链：

HTML DOM 核心DOM

document-->HTMLDocument | -->Document-->Node-->Object

elem-->HTMLXXXElement |

-->HTMLElement | -->Element-->Node

正课：

1. \*\*\*DOM Tree

\*\*\*\*\*递归遍历DOM Tree

2. \*查找

1. \*\*\*DOM Tree

\*\*\*\*\*递归遍历DOM Tree： 手写

遍历：查看一个父节点下的所有子节点

递归遍历：对每个子节点执行相同的查找操作！

在函数内又调用了函数本身。

何时递归：不确定子节点层级深度时

如何递归：2步：

1. 先定义一个函数，仅遍历直接子节点

2. 在遍历过程中，对每个直接子节点执行相同的遍历操作！

深度优先：对每个子节点，都优先遍历下级节点，直到叶子节点才返回上一级。

缺点：占用更多内+5

02141414存空间，且执行效率低

vs 循环：循环是在一个EC内反复执行相同代码段，效率高！

遍历API：

1. NodeIterator：按照深度优先的属性，依次遍历每个节点对象

如何使用：2步：

1. 创建对象：var iterator=document.createNodeIterator(

开始节点对象,

显示何种节点,NodeFilter.SHOW\_ALL

NodeFilter.SHOW\_ELEMENT

null,false

);

2. 使用循环，推动迭代器移动到下一个节点：

2. TreeWalker: 基本用法和NodeIterator完全一样

比NodeIterator，多提供了灵活的跳转方法：

walker.parentNode()

walker.firstChild()

walker.nextSibling()...

2. \*查找：2大类：

1. 按HTML属性查找：

1. 按id查找：

var elem=document.getElementById("id值");

2. 按标签名查找所有子元素

var elems=elem.getElementsByTagName("标签名");

返回动态集合：不实际保存节点对象，仅保存节点对象的引用

每访问一次集合中元素或属性，都会重新查找！

还有：childNodes children都是返回动态集合

\*\*\*如何遍历：

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

强调：不仅查找直接子节点，而且还找所有子代节点

如果仅找到1个节点对象，也放在数组里返回，elems[0]

3. 按name属性查找:

var elems=elem.getElementsByName("name属性值");

4. 按class属性名查找: IE9+

var elems=elem.getElementsByClassName("class属性值");

2. 按选择器查找：Selector API - jQuery的核心！

1. 只找1个元素:

var elem=parent.querySelector("css选择器");

无论找到几个，仅返回第一个符合条件的

2. 找多个元素:

var elems=parent.querySelectorAll("css选择器")

返回非动态集合：实际保存节点对象

即使反复使用集合中的元素或属性，也不会反复查找.

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

\*\*\*鄙视题：Selector API vs getxxxByxxx()

getXXXBYxxx(): 优：执行效率高

缺：繁琐

返回值: 返回动态集合，不实际保存节点对象

只要访问集合的元素和属性都要重新查找

何时使用：如果经过一次查找即可获得结果时

Selector API: 优：易用

缺：执行效率低

返回值: 返回非动态集合，实际保存每个节点对象的属性

访问集合的元素和属性，不会导致重新查找

何时使用：需要经过多级选择器查找时

作业：素材: 7\_1

TMOOC0课后练习:3,4

鄙视题：

var emp={

work:function(){

var sum=0;

for(var i=0;i<arguments.length&&arguments[0]>0;i++){

//callee等效于当前函数对象——实现递归

sum+=arguments[i]+arguments.callee(--arguments[i])

}

return sum;

}

}

console.log(emp.work(3,2,1));//?

Js DOM 第二天

回顾：

1. 自定义属性：

elem.getAttribute("data-属性名") - 所有浏览器都兼容

html5： elem.dataset.属性名

正课：

1. \*\*\*修改样式：2处：

1. 仅获取或设置内联样式：

elem.style.属性名="值";

其实elem.style是CSSStyleDeclaration类型的对象

封装了一个元素所有css样式属性的值

固定套路：清除一个元素的所有内联样式:

elem.style.cssText="";

2. 获取元素最终应用的所有样式：计算后的样式

var style=getComputedStyle(elem);

style也是CSSStyleDeclaration类型的对象

IE8：elem.currentStyle;

style.属性名

只要设置：elem.style.属性名 -> 加入内联样式，优先级最高！

不影响其他元素的样式

只要获取：var style=getComputedStyle(elem)

3. 改样式表中的样式：3步：chrome不支持本地的外部css文件

1. 从样式表集合中获取第i个样式表对象

var sheet=document.styleSheets[i]

sheet是CSSStyleSheet类型的对象：

1. sheet.cssRules集合: 样式表中所有cssRule对象

cssRule对象：css中每个{}就是一个cssRule对象

cssRule下还可包含子Rule,比如: @keyframes

2. cssRule对象下，又包含:

1. cssText: 保存Rule完整的字符串

2. selectorText: 选择器名称

\*3. style: CSSStyleDeclaration

style.属性名

2. 获取CSSRule对象：

var rule=sheet.cssRules[i]

3. 通过rule获得style对象

var style=rule.style

style.属性名='值';

作业：4\_1

TMOOC 第三题

Js DOM 第三天

正课：

1. 创建和删除节点：

2. HTML DOM常用对象：Table Select Form

1. 创建和删除节点：——核心DOM

1. 创建单个元素节点：3步：

1. 创建空元素节点对象：

var elem=document.createElement("标签名");

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

html: <a></a>

2. 设置新元素的关键属性：

比如: a.href="http://tmooc.cn";

a.innerHTML="go to tmooc";

html: <a href="http://tmooc.cn">go to tmooc</a>

3. 将新节点挂到DOM Tree的指定父节点下：3种：

追加：parent.appendChild(elem);

插入：parent.insertBefore(elem,oldElem);

替换：parent.replaceChild(elem.oldElem);

2. 创建多个平级元素节点：3步：

1. 先创建一个文档片段对象：

var frag=document.createDocumentFragment();

文档片段：内存中临时保存一个DOM子树的对象

何时使用：只要添加多个平级节点时，都要先放入文档片段，再将文档片段整体挂到DOM Tree上

2. 将子节点先放入文档片段

3. 将文档片段整体挂到DOM Tree上

文档片段本身不会出现在DOM Tree上，只会将元素加入DOM树

3. 删除：parent.removeChild(子节点);

子节点.parentNode.removeChild(子节点);

2. HTML DOM常用对象：

1. Select对象：

事件：onchange 选中项发生改变时

属性：sel.selectedIndex 获得当前选中项的位置

sel.options 获得sel下所有option元素对象

方法：sel.add(opt) 将opt追加到sel下

sel.remove(i) 删除sel下i位置的option元素

Option对象：

创建一个新option：var opt=new Option(text,value);

相当于核心DOM三句话:

var opt=document.createElement("option");

opt.value=value;

opt.innerHTML=text;

属性：opt.index 获得当前opt所在的位置下标

opt.text opt.value: 分别获得innerHTML和value

opt.selected: 判断当前opt是否被选中！

强调：opt的selected不是bool值！

但是却可直接作为条件判断

创建option同时，追加到select中：

sel.add(new Option(text,value));

正课：

1. \*\*\*HTML DOM常用对象: Table Form

1. Table对象：

Table:createTHead() deleteTHead()

createTFoot() deleteTFoot()

createTBody()

tHead: insertRow(i),如果省略表示追加

deleteRow(i),如果省略优先删除首行

rows：统领当前行分组下的tr

row.rowIndex: 获得当前行的下标！专门用于删除当前行

强调：rowIndex是从整个表格的第一个tr开始计算

删除: table.deleteRow(row.rowIndex);

insertCell(i),如果省略表示追加

deleteCell(i),如果省略优先删除首格

cells: 当前列中的所有td

cell.cellIndex: 获得当前格的下标，专用于删除

tBodies

tBody

rows：统领当前行分组下的tr

cells: 当前列中的所有td

tFoot

rows：统领当前行分组下的tr

cells: 当前列中的所有td

rows: 统领table下所有tr

table.insertRow(i),如果省略: 默认追加在最后一个行分组内

table.deleteRow(i),如果省略: 优先删除首行

课堂练习：window.confirm("确认信息")

如果用户选择确认：返回true

如果用户取消：返回false

2. Form对象：

1. 快速获取表单元素:var form=document.forms[i/id/name];

2. 快速获取表单中的元素：var elem=form.elements[i/name]

elements中仅包含数据采集类的元素，比如:input,select...

其实.elements可省略

表单的方法:

手动提交: form.submit();

表单的事件:

onsubmit: 在正式提交前触发，主要做验证

如果验证没通过可取消事件！

表单元素的方法：

elem.focus(): 将elem设为焦点！

比如：一个elem验证失败时，希望直接定位到失败的元素

elem.blur(): 将elem失去焦点!——多数自动控制

JsBOM第一天

正课：

1. BOM：

window对象：

打开关闭链接

窗口大小和定位

\*\*\*定时器

1. BOM：专门操作浏览器窗口的API

window: 2个角色：

1. 带ES中的Global成为全局作用域

2. 封装了浏览器软件和窗口对象的属性和方法-BOM

打开新连接的方式总结: 4种情况：

1. 替换当前窗口，允许后退

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

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

2. 替换当前窗口，不允许后退

js:

history对象：保存当前窗口打开过的url历史记录的栈

location对象：保存当前窗口正在打开的url的对象

location.replace("url");

3. 在新窗口打开，可重复打开

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

js: open("url")

4. 在新窗口打开，不可重复打开

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

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

[var newWin=window.]open("url"[,"name","config"])

window.可省略

返回值，新打开的窗口对象！

newWin相当于新窗口中的window！

"name"，默认是\_blank,可省略

如果自定义name参数，则新窗口不可重复打开

每个window对象在内存中都有一个唯一的name属性

浏览器规定，相同name属性的窗口只能打开一个！

后打开的窗口，会替代先打开的

窗口大小和窗口定位：

1. 打开窗口时，设置窗口大小和位置：

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

config="top=?,

left=?,

width=?,

height=?,//仅包含文档显示区

resizable=yes,

location=no";

2. 获取窗口宽高：window.innerHeight/Width: 文档显示区

window.outerHeight/Width: 整个窗口

3. screen对象：当前桌面的分辨率信息

screen.height/width: 操作系统的桌面分辨率

screen.availHeight/Width: 桌面可用的大小范围

去掉任务栏

4. 调整大小：window.resizeTo(width,height)

window.resizeBy(宽度的增量，高度的增量)

5. 移动窗口位置：window.moveTo(left,top);

window.moveBy(left的增量,top的增量)

事件发生时，获取鼠标相对于桌面的x和y坐标:

e.screenX/screenY

正课：

1. \*\*\*\*定时器：

周期性：

一次性：

1. \*\*\*\*定时器：让程序\*每隔指定的时间间隔\*反复执行一项任务。

反复自动调用函数的机制

何时使用：让程序\*每隔指定的时间间隔\*反复执行一项任务

周期性：让程序\*每隔指定的时间间隔\*反复执行一项任务

何时使用：如果一项任务需要反复定时执行时

如何使用：3件事:

1. \*\*\*任务：函数对象

都要根据临界值判断是否需要停止定时器

2. 将任务放入定时器，设定时间间隔，启动定时器:

timer=setInterval(任务函数,时间间隔);

强调：timer应该定义在全局

3. 如果需要手动停止定时器，则必须保存定时器序号timer

序号只能在启动定时器时获得！

clearInterval(timer);

timer=null;

一次性：让程序先等待一段时间后，自动执行一次任务

何时使用：如果一项任务可能反复停止或启动时

如何使用：3件事:

1. \*\*\*任务：函数对象

根据临界值判断是否启动下一次定时器

如果不启动，自动停止了

2. 将任务放入定时器，设定等待时间，启动定时器:

timer=setTimeout(任务函数,时间间隔);

强调：timer应该定义在全局

3. 如果需要手动停止定时器，则必须保存定时器序号timer

序号只能在启动定时器时获得！

clearTimeout(timer);

timer=null;

JsBOM第二天

正课：

1. BOM常用对象: navigator history location

2. \*\*\*事件:

3. cooke:

1. BOM常用对象：

\*\*\*navigator: 保存了浏览器的名称，版本以及配置信息

cookieEnabled: 判断当前浏览器是否启用cookie

plugins: 保存了当前浏览器安装的所有插件的信息的集合

plugins[i].name

userAgent: 保存了浏览器名称和版本信息的字符串

history: 保存当前窗口访问过的url的历史记录栈

history.go(n): 操作history中的游标前后移动

n为正: 前进几步

n为负: 后退几步

n为0: 刷新

location: 保存当前窗口正在打开的url地址的对象

在当前窗口打开新页面,可后退

location="url";

location.href="url";

location.assign("url");

不可后退：

location.replace("url");

刷新页面:

location.reload(true/false): 刷新

true: 总重新从服务器下载文档

false: 默认如果服务器上文档的更新时间，在上次下载后未改变，则直接从缓存中获取文档。

2. \*\*\*事件：页面本身或页面的内容的状态发生了改变

DOM Level2 Event 标准：IE9+

IE8 自成体系:

事件处理函数：当事件发生时，自动执行的函数对象

其实就是页面或页面元素的属性而已

命名：on事件名

如何绑定事件处理函数: 3种：

1. 在html元素中绑定事件处理函数属性:

<elem on事件名="js语句"></elem>

比如: <button onclick="fun(this)"></button>

js: function fun(btn){

btn-->传进来的button

this-->window

}

构建DOM Tree:

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

eval("fun(this)"); //fun(this);

button

}

事件发生时，自动执行：button.onclick();

2. 在js中动态绑定事件处理函数：2种：

1. elem.on事件名=函数对象

当事件发生时: elem.on事件名();

在函数对象内，this-->elem

问题1：同一元素的同一事件处理函数属性只能绑定一个函数对象

问题2：无法修改事件的执行顺序

2. DOM 标准：

elem.addEventListener("事件名",函数对象,false/true)

当事件发生时: elem.on事件名();

在函数对象内，this-->elem

强调：1. 事件名，不带on

2. 函数对象中的this-->elem

3. 表示是否在捕获阶段触发，默认是false

IE8：？

正课：

1. \*\*\*事件：

\*\*\*事件周期

固定套路：

1. 取消冒泡

2. 利用冒泡

3. 取消事件

事件坐标：

1. \*\*\*事件:

\*\*\*事件周期：从事件触发到各级事件执行完毕的全过程

DOM 标准：3个阶段：

1. 捕获阶段： 从根元素-目标元素

捕获：记录每一级元素是否绑定了事件处理函数

2. 目标触发: 触发目标元素上的事件处理函数

目标元素：实际触发事件的元素

3. 冒泡触发：从实际发生事件的元素的父元素开始，依次触发各级父元素上捕获的事件处理函数

改变事件触发的顺序：

elem.addEventListener("事件名",函数对象,是否在捕获阶段触发);

第三个参数为true，可提前触发该事件处理函数

IE8 事件周期：没有捕获阶段，只有目标触发和冒泡

不支持addEventListener

attachEvent("on事件名",函数对象)

因为没有捕获阶段和第三个参数，所以无法改变事件触发顺序

事件对象：在事件发生时，自动创建，封装事件信息的对象

获取事件对象：

DOM 标准：默认event对象作为事件处理函数第一个参数传入

arguments[0]

IE 8：事件发生时创建事件对象，放入全局变量event

window.event

兼容：var e=window.event||arguments[0];

IE DOM

固定套路：

　　　1. 取消冒泡：

DOM 标准: 阻止事件继续向下传导和事件冒泡

e.stopPropagation();

IE 8: 仅阻止事件冒泡阶段

e.cancelBubble=true;

兼容模式：

if(e.cancelBubble){

e.cancelBubble=true;

}else{

e.stopPropagation();

}

\*\*\*2. 利用冒泡：

\*\*\*优化：如果多个子元素都定义了相同的事件处理函数

只要在父元素上定义一次事件处理函数即可

为什么：页面中事件绑定的次数直接影响页面执行效率！

减少事件绑定次数，都可提高页面执行效率

\*\*\*获得目标元素：

DOM 标准: e.target

IE 8: e.srcElement

兼容: var target=e.target||e.srcElement;

DOM IE

3. 取消事件：2种情况：

1. 在html中绑定的事件处理函数: 2个return

html: <elem onXXX="return fun()">

js: function fun(){

return true/false;

}

如果fun返回true，则继续执行事件,否则取消事件

2. 在js中动态绑定事件如何取消：

DOM 标准：e.preventDefault();

IE 8: e.returnValue=false;

兼容: if(e.preventDefault!==undefined){

e.preventDefault();

}else{

e.returnValue=false;

}

事件发生时，鼠标的坐标: 3种定位范围

左上角: 0,0

1. 相对于屏幕左上角: e.screenX/Y

最大值: screen.availWidth/availHeight

2. 相对于文档显示区左上角: var x=e.clientX||e.x;

var y=e.clientY||e.y;

最大值: window.innerWidth/innerHeight

3. 相对于父元素左上角: e.offsetX/Y

最大值: 父元素的width和height

页面滚动: window.onscroll

获得页面滚动的距离：页面顶部超出文档显示区顶部的距离

var scrollTop=document.documentElement.scrollTop

||document.body.scrollTop

jQuery 第一天

JQUERY DAY01:

\* JavaScript类库(JS库)

\* 定义 - 预封装了很多的对象和函数

\* 目的 - 简化JavaScript的开发

\* 作用 - 兼容各大浏览器

\* 主流的JS库

\* jQuery - 最多、经典

\* Prototype - 基于原型概念发展的JS库

\* YUI - Yahoo User Interface

\* ExtJS

\* ...

\* 目的

\* 掌握jQuery的用法

\* 掌握如何学习JS库

\* jQuery

\* 基础内容

\* 06年1月推出

\* 分类

\* jQuery - WEB版本

\* jQuery UI - 集成式UI组件

\* jQuery Mobile - 针对移动开发

\* QUnit - 测试JavaScript代码

\* 版本

\* 1.xx版本 - 浏览器全兼容版本

\* 1.4.2版本 - 企业级开发主流版本

\* 1.8.3版本 - 互联网开发主流版本

\* 1.12版本 - 最新版本

\* 2.xx版本 - 不再支持IE678

\* 宗旨 - write less, do more(写更少的代码,做更多的事儿)

\* jQuery的特性

\* 具有相对完善的处理机制

\* 隐式迭代 - 遍历循环

\* 链式操作

\* 如何使用jQuery

\* 引入jQuery文件

\* 定位页面元素(通过jQuery的选择器)

\* 使用jQuery提供的API

\* jQuery特点

\* $() - jQuery的工厂函数

\* 另外一种写法 - jQuery()

\* 作用

\* 使用jQuery的选择器

\* 将DOM对象转换为jQuery对象

\* "$"符号

\* 作用 - 等价于jQuery

\* 注意 - jQuery官方的约定

\* DOM和jQuery

\* DOM对象 - 通过DOM获取的页面元素

\* jQuery对象 - 封装了DOM对象后,所产生的对象

\* jQuery的底层代码还是DOM

\* DOM对象与jQuery对象之间的转换

\* DOM对象转换为jQuery对象

\* $(DOM对象)

\* jQuery对象转换为DOM对象

\* 通过jQuery的get(index)方法--(对象.方法)

\* index - 下标

\* jQuery对象是类数组对象

\* jQuery对象[角标]

\* 注意 - DOM对象与jQuery对象不能相互调用

\* jQuery的选择器

\* 基础内容

\* jQuery中的选择器

\* 设计原理来源于CSS中的选择器

\* 相对CSS的选择器有所改变和扩展

\* 注意

\* jQuery的选择器和CSS的选择器不能混淆

\* jQuery的选择器是jQuery的根基

\* 问题

\* 数量比较多 - 记不住(熟悉jQuery的帮助文档)

\* 难点多个选择器混合使用 - 没有标准写法

\* 选择器

\* 基本选择器

\* id选择器 - #id

\* 元素选择器 - element

\* class选择器 - .className

\* 通配符选择器 - \*

\* 组合选择器 - selector1,selector2,...

\* 层级选择器

\* 父子选择器 - selector1>selector2

\* 祖先与后代选择器 - selector1 selector2

\* 下一个兄弟选择器 - selector1+selector2

\* 后面所有兄弟选择器 - selector1~selector2

\* 过滤选择器 - 具有前缀":"

\* 基本过滤选择器

\* :first - 匹配第一个元素

\* :last - 匹配最后一个元素

\* :even - 匹配索引值为偶数的元素

\* :odd - 匹配索引值为奇数的元素

\* :eq(index) - 匹配索引值等于xxx的元素

\* :gt(index) - 匹配索引值大于xxx的元素

\* :lt(index) - 匹配索引值小于xxx的元素

\* :not - 匹配属性不为xxx的元素(包含没有该属性的元素)

\* :header - 匹配h1~h6元素

\* :animated - 匹配正在执行动画的元素

\* 注意 - 只能匹配jQuery实现的动画效果

\* 可见性过滤选择器

\* :hidden - 匹配所有隐藏的元素

\* <input type="hidden"> - 隐藏域

\* CSS中的display:none

\* :visible - 匹配所有可见的元素

\* 子元素过滤选择器 - 前面增加空格

\* :first-child - 匹配第一个子元素

\* :last-child - 匹配最后一个子元素

\* :nth-child - 匹配第n个子元素

\* 注意 - 从 1 开始

\* :only-child - 匹配只有一个子元素

\* 属性过滤选择器(重要)

\* [attr] - 匹配包含指定属性的元素

\* [attr=value] - 匹配包含指定属性等于xx值的元素

\* [attr!=value] - 匹配包含指定属性不等xx值的元素

\* [attr^=value] - 匹配包含指定属性值以xx开始的元素

\* [attr$=value] - 匹配包含指定属性值以xx结束的元素

\* [attr\*=value] - 匹配包含指定属性值包含xx的元素

\* [][] - 组合属性过滤选择器

\* 内容过滤选择器

\* :contains(text) - 匹配包含指定文本的元素

\* :parent - 包含子元素或者文本内容的元素

\* :empty - 既不包含子元素又不包含文本内容的元素

\* :has(selector) - 包含指定selector元素的元素

\* 注意 - 查找指定元素的父元素

\* 表单对象属性过滤选择器

\* :disabled - 匹配包含属性disabled(不可用)的元素

\* :enabled - 匹配不包含属性disabled(可用)的元素

\* :selected - 匹配被选中的下拉列表中的选项

\* :checked - 匹配被选中的radio和checkbox元素的选项

\* 表单选择器

\* jQuery的底层逻辑

\* jQuery的工厂函数

var jQuery = function(selector){}

window.$ = window.jQuery;

\* jQuery对象是类数组对象

var jQuery = function(){

return new jQuery.fn.init();

}

init = jQuery.fn.init = function(){

return jQuery.makeArray();

}

makeArray = function(){

// 如果results存在,ret等于results;如果results不存在,ret等于[]

var ret = results || [];

return ret;

}

\* 尽量少定义全局变量和函数

\* 原因 - 全局变量和全局函数占用浏览器内存空间

\* 解决

\* 匿名自调函数

(function(){

var str = "abcderfg";//局部变量

function fn(){}//内部函数

}({}));

\* 扩展内容

\* 封装的概念

\* 只关注输入和输出,而不再关注过程

\* Web前端的发展 - 全栈工程师

\* PC客户端 - 页面

\* 移动端 - 浏览器|APP

\* 服务器端 - NodeJS

\* 其他终端

\* 类数组对象

\* Arguments对象 - 类数组对象

\* 定义 - 结构类似于数组的对象

\* 开发者工具

\* console - 控制台

\* 输出

\* log() - 日志

\* info() - 信息

\* warn() - 警告

\* error() - 错误

\* 输入 - 直接编写JavaScript代码

\* elements - 页面快速定位

\* 从页面定位代码

\* 从代码定位页面

\* sources - 断点调试(重要)

\* network - 抓取网络传输数据

\* timeline - 代码|性能优化

\* resource - 页面存储

\* HTML分为数据(内容)和结构

\* 数据 - 文本内容

\* 结构 - HTML元素(标签)

\* 作业 - 实现多页签切换效果

jQuery 第二天

JQUERY DAY02:

\* DOM操作

\* 基础内容

\* JQUERY是JS库,JQUERY就是封装了DOM

\* JQUERY中的DOM没有原生DOM的那些问题

\* 基本操作

\* html()方法 - 类似于原生DOM中的innerHTML属性

\* 获取 - html()

\* 设置 - html(html)

\* val()方法 - 类似于原生DOM中的value属性

\* 获取 - val()

\* 设置 - val(value)

\* text()方法 - 类似于原生DOM中的textContent属性

\* 获取 - text()

\* 设置 - text(text)

\* attr()方法 - 类似于原生DOM中的getAttribute()和setAttribute()

\* 获取 - attr(name)

\* 设置 - attr(name, value)

\* 删除 - removeAttr(name)

\* 样式操作

\* attr("class",className) - 设置样式

\* addClass(className) - 追加样式

\* removeClass() - 移除样式

\* 不传递参数 - 删除所有样式

\* 传递参数 - 删除指定样式

\* toggleClass(className) - 切换样式

\* 注意 - 是在没有指定样式与有指定样式之间切换

\* hasClass(className) - 判断是否含有指定样式

\* css()方法

\* 获取 - css(attrName)

\* 设置

\* css(attrName, attrValue) - 设置一个样式

\* css({ - 设置多个样式

attrName : attrValue,

attrName : attrValue

})

\* 遍历元素

\* 父元素 - parent()

\* 祖先元素 - parents()

\* 子元素 - children()

\* 兄弟元素

\* 上一个兄弟元素 - prev()

\* 下一个兄弟元素 - next()

\* 所有兄弟元素 - siblings()

\* 通用 - find(expr) - 查找指定元素的指定后代元素

\* 创建节点 - $(HTML代码)

\* 元素节点 - $(HTML代码)

\* 属性节点 - attr()和removeAttr()

\* 文本节点 - text()

\* 插入节点

\* 内部插入 - 插入子节点

\* append() - 后面的节点插入到前面的节点的子节点的最后

\* prepend() - 后面的节点插入到前面的节点的子节点的最前

\* appendTo()

\* prependTo()

\* 外部插入 - 插入兄弟节点

\* insertBefore()

\* insertAfter()

\* before() - 后面的节点插入到前面的节点的前面

\* after() - 后面的节点插入到前面的节点的后面

\* 删除节点

\* remove() - 删除自身节点及后代节点

\* empty() - (清空)删除后代节点,但保留自身节点

\* 替换节点

\* replaceWith()

\* 前面元素是被替换元素

\* 后面元素是替换元素

\* replaceAll() - 就是颠倒了的replaceWith()

\* 事件

\* 基础内容

\* JQUERY中的事件源于DOM中的事件

\* 页面载入 - ready()

\* ready() - 类似于window.onload事件

\* 写法

\* $(document).ready(function(){});

\* $().ready(function(){});

\* 简写用法 - $(function(){});

\* ready与window.onload的区别

\* ready

\* 具有简写写法

\* 一个HTML页面中允许编写多个

\* (效率高)需要在DOM加载后就可以执行

\* window.onload

\* 没有任何简写

\* 一个HTML页面中只能编写一个

\* (效率低)必须在页面中所有内容都加载完毕后,才能执行

\* 事件的绑定与解绑

\* 方法

\* bind(type,[data],fn)方法 - 事件绑定

\* type - 指定要绑定的事件名称,例如"click"、"dblclick"等

\* data - 作为event.data属性值传递给事件对象的额外数据对象

\* fn - 回调函数,绑定事件后的处理函数

\* 作用 - 类似于原生DOM中的addEventListener()方法

\* unbind([type],[fn])

\* type - 指定要解绑的事件名称

\* fn - 解绑事件后的处理函数

\* 场景

\* 绑定与解绑单个事件

\* bind(单个事件名称,function(){})

\* unbind()

\* 不传递参数 - 默认全部解绑

\* 传递参数 - 解绑指定单个事件

\* 绑定与解绑多个事件

\* bind(事件名称,function(){})

\* 事件名称为多个,中间用空格隔开

\* unbind()

\* 不传递参数 - 默认全部解绑

\* 传递一个参数 - 解绑指定单个事件

\* 传递多个参数 - 解绑指定多个事件

\* bind()、live()、one()和on()的区别?

\* bind() - 绑定事件

\* live() - 老版本具有,新版本移除

\* one() - 只绑定一次事件

\* on() - 是jQuery底层提供的机制

\* 模拟事件

\* 定义 - 用于模拟用户触发对应事件的效果

\* trigger(事件名称)

\* 注意 - 绑定什么事件,才能模拟什么事件

\* 回顾内容

\* 颜色设置

\* 单词进行设置 - red、blue

\* 三原色 - 红|绿|蓝

\* #xxxxxx

\* rgba(0,0,0,0)

\* 透明度 - 值范围 0~1

\* 原生DOM操作

\* 遍历节点

\* 父节点 - parentNode

\* 子节点

\* 所有子节点 - childNodes

\* 第一个子节点 - firstChild

\* 最后一个子节点 - lastChild

\* 兄弟节点

\* 上一个兄弟节点 - previousSibling

\* 下一个兄弟节点 - nextSibling

\* 创建节点

\* 元素节点 - createElement()

\* 文本节点 - createTextNode()

\* 属性节点 - 不是子节点

\* 设置属性 - setAttribute()

\* 获取属性 - getAttribute()

\* 删除属性 - removeAttribute()

\* 插入节点

\* appendChild()

\* insertBefore()

\* 没有insertAfter()

\* 删除节点 - parent.removeChild(child)

\* 替换节点 - replaceChild()

\* JavaScript对象

\* 原始类型

\* string

\* number

\* boolean

\* null

\* 原始对象

\* String - ''和""一致

\* Number - 32位和64位

\* 32位 - 整数 - Integer

\* 64位 - 小数 - Float/Double

\* 二进制、八进制和十进制(十六进制)

\* 得到的结果都是十进制

\* Boolean

\* 内置对象

\* Array

\* Math

\* RegExp

\* 宿主对象 - 浏览器环境

\* BOM对象

\* DOM对象

\* 面试题

\* 下列哪个选项是错误的?D

A var a = [];//数组

B var b = {};//对象

C var c = //;//正则表达式

D var d = ();

\* alert("xxxyyy"),xxx和yyy换行

\* event事件对象

\* target - 绑定该事件的目标元素

\* pageX/clientX/offsetX/x - 鼠标当前坐标值x

\* pageX - 相对于当前HTML页面的坐标值

\* offsetX - 相对于父元素的坐标值

\* pageY/clientY/offsetY/y - 鼠标当前坐标值y

\* keyCode/charCode - 键盘值

\* this的用法

\* 用法

\* 可以指代具有上下文环境的上下文对象

\* 可以指代new之后的对象

\* 原则 - 会就用,不会就别用

\* 尽量不要使用this

\* window.onload事件

\* 当JavaScript代码在<body>元素前引入时

\* 作用 - 当页面加载完毕后,再执行以下代码

\* JS中的特殊函数

\* 匿名函数 - 没有名的函数

\* 匿名自调函数 - 只定义不调用(自己调用)

\* 匿名回调函数

\* 回调函数 - 将一个函数作为另一个函数的参数,而作为参数的函数就叫做回调函数

\* 内部函数 - 在一个函数中定义的函数

\* 阻止默认行为

\* 默认行为 - HTML页面本身具有的某种行为

\* 定义<form>元素 - 提交表单

\* 定义<a>元素 - 跳转页面

\* 如何阻止?

\* 事件对象event.preventDefault()

\* return fasle

\* return fasle的作用

\* 阻止默认行为

\* 不再执行以下代码

\*

jQuery 第三天

JQUERY DAY03:

\* 动画效果

\* 显示与隐藏 - 同时改变高度和宽度

\* show() - 显示

\* 无动画 - show()

\* 有动画 - show(speed,callback)

\* speed - 设置动画执行的时长(单位为毫秒)

\* 三个预定义时长 - slow|normal|fast

\* callback - 动画执行完毕后调用的函数

\* hide() - 隐藏

\* 无动画 - hide()

\* 有动画 - hide(speed,callback)

\* speed - 设置动画执行的时长(单位为毫秒)

\* 三个预定义时长 - slow|normal|fast

\* callback - 动画执行完毕后调用的函数

\* 滑动效果 - 改变高度

\* slideUp(speed,callback) - 向上滑动(隐藏)

\* 注意

\* 默认不传递任何参数,并不报错

\* 代码底层逻辑预定义一个动画的时长

\* 参数

\* speed - 设置动画执行的时长(单位为毫秒)

\* 三个预定义时长 - slow|normal|fast

\* callback - 动画执行完毕后调用的函数

\* slideDown(speed,callback) - 向下滑动(显示)

\* 注意

\* 默认不传递任何参数,并不报错

\* 代码底层逻辑预定义一个动画的时长

\* 参数

\* speed - 设置动画执行的时长(单位为毫秒)

\* 三个预定义时长 - slow|normal|fast

\* callback - 动画执行完毕后调用的函数

\* 淡入淡出 - 改变透明度

\* fadeIn(speed,callback) - 淡入

\* 注意

\* 默认不传递任何参数,并不报错

\* 代码底层逻辑预定义一个动画的时长

\* 参数

\* speed - 设置动画执行的时长(单位为毫秒)

\* 三个预定义时长 - slow|normal|fast

\* callback - 动画执行完毕后调用的函数

\* fadeOut(speed,callback) - 淡出

\* 注意

\* 默认不传递任何参数,并不报错

\* 代码底层逻辑预定义一个动画的时长

\* 参数

\* speed - 设置动画执行的时长(单位为毫秒)

\* 三个预定义时长 - slow|normal|fast

\* callback - 动画执行完毕后调用的函数

\* 切换效果 - 页面中的特效

\* toggle() - show()+hide()

\* slideToggle() - slideUp()+slideDown()

\* 自定义动画

\* animate(params,duration,easing,callback)

\* params - 设置自定义动画的效果

\* 直接使用CSS中的属性样式

\* duration - 设置自定义动画的时长(单位为毫秒)

\* easing - 要使用的擦除效果的名称

\* callback - 自定义动画执行完毕后的处理函数

\* animate(params,options)

\* params - 设置自定义动画的效果

\* 直接使用CSS中的属性样式

\* options - 选项

\* duration - 设置自定义动画的时长(单位为毫秒)

\* easing - 要使用的擦除效果的名称

\* complete - 自定义动画执行完毕后的处理函数

\* queue - 设定为false将使此动画不进入动画队列

\* 不能使用的CSS属性

\* backgroundColor

\* borderBottomColor

\* borderLeftColor

\* borderRightColor

\* borderTopColor

\* Color

\* outlineColor

\* 动画排队与并发

\* 排队效果 - 多个动画按照先后顺序执行

\* 并发效果 - 多个动画同时执行

\* 类数组对象

\* length属性 - 返回所包含的DOM对象的个数

\* 方法

\* get(index) - 根据index值返回对应的DOM对象

\* eq(index) - 根据index值返回对应的JQUERY对象

\* 隐式迭代

\* $().each(callback) - 对象方法

\* callback - 回调函数

function(index,domEle){}

\* index - 遍历过程中的索引值

\* domEle - 遍历得到的DOM对象

\* $.each(object,callback) - 全局函数

\* object - 需要遍历的对象或数组

\* callback - 回调函数

function(index,domEle){}

\* index - 遍历过程中的索引值

\* domEle - 遍历得到的DOM对象

\* 与其他JS库共存 - "$"符号的使用权

\* 宗旨 - JQUERY放弃"$"符号的使用

\* JQUERY UI

\* 在线中文帮助文档

\* http://www.css88.com/jquery-ui-api/

\* 如何使用JQUERY UI

\* 引入相关文件

\* 引入JQUERY文件

\* 引入JQUERY UI的JS文件

\* 引入JQUERY UI的CSS文件

\* 按照规范定义HTML元素

\* 使用JQUERY UI提供的API

\* 提供哪些功能

\* 效果

\* animate()方法 - 自定义动画

\* JQUERY的animate()方法的区别,该方法支持所有CSS属性

\* 使用方式上与JQUERY中的animate()一致

\* effect(effect,option,duration,callback)方法

\* 作用 - 提供各种效果

\* 参数

\* effect - 执行的效果名称

\* option - 效果选项

\* duration - 执行的时长

\* callback - 执行完毕后的处理函数

\* 组件

\* 使用上

\* 工作中使用的并不多

\* 意义

\* 工作中都会使用到类似于JQUERY UI的JS库

\* 是经典的案例

\* 大公司会自己封装类JQUERY UI

\* 扩展内容

\* JQUERY中文帮助文档:http://www.jquery123.com/

\* JQUERY的选择器与方法对应

\* :eq(index) - eq(index)方法

\* :first - first()方法

\* :last - last()方法

\* 方法与函数的区别?

\* 方法 - 只存在于对象中

\* 函数 - 在对象之外定义的

\* 形参与实参的区别?

\* 形参 - 定义函数时的参数

\* 实参 - 调用函数时的参数

\* 软件设计原理

\* 封装

\* 低耦合 - 组件化

\* 浏览器内核

\* IE浏览器 - IE6/7/IE8 IE9/10/11

\* IE版本从 6 到 11 版本,至少有两个内核

\* 其他浏览器 - Webkit(Apple)

\* Chrome - V8引擎(Nodejs)

\* Firefox - 自主内核产品

\* Safari

\* 国内浏览器

\* QQ浏览器 - 号称自主内核(V5)

\* 百度浏览器 - 号称自主内核(X5)

\* 遨游浏览器 - 号称自主内核

\* 国际化

\* 国际化 - i18n

\* 本地化 - i10n

\* 技术学习

\* Web前端 - 不建议买书

\* 官方网站 - 帮助文档

\* 技术类网站 - Demo示例代码

\*

jQuery 第四天

JQUERY DAY04:

\* 使用插件

\* 基础内容

\* 插件 - 组件,热插拔

\* 官方插件 - http://plugins.jquery.com/

\* 目的

\* 掌握所学习的JQUERY插件的使用

\* 掌握如何学习(研究)JQUERY插件

\* 学习(研究)新插件还是新JS库

\* 官方提供的Demo示例代码

\* 官方提供的API帮助文档

\* 日期插件 - laydate

\* 如何使用laydate插件

\* 将laydate文件夹整体拷贝到工程目录中

\* 引入相关文件

\* 引入JQUERY文件

\* 引入laydate文件

\* HTML页面,定义<input>输入框元素

\* class="laydate-icon"

\* JavaScript代码

\* laydate(options)方法

\* 官网 - http://laydate.layui.com/

\* 表单验证插件 - validation

\* 如何使用validation

\* 在HTML页面中

\* 引入JQUERY文件

\* 引入validation插件的JS文件

\* 引入validation插件的CSS文件

\* 在JavaScript代码

\* 获取HTML页面中的表单元素

\* 使用validation插件的核心方法

validate()

\* 核心的验证方法 - validate()

{

rules : {//设置验证规则

key : value

\* key - 验证元素的name属性值

\* value - 验证规则名称

},

messages : {//设置错误信息

key : value

\* key - 验证元素的name属性值

\* value - 设置的错误信息

}

}

\* 自定义验证规则

$.validator.addMethod(name,function(value,element,param){},message);

\* name - 自定义验证规则的名称

\* method - 自定义验证规则的逻辑

function(value,element,param){}

\* value - 验证元素的value属性值

\* element - 验证的元素(DOM对象)

\* param - 参数(rules中的value值)

\* message - 自定义验证错误信息

\* 瀑布流插件 - masonry

\* 官网 - http://masonry.desandro.com/

\* 如何使用瀑布流插件

\* 引入相关文件

\* 引入JQUERY文件

\* 引入masonry文件

\* HTML页面

\* 定义容器元素 - <div></div>

\* 定义成员元素 - <div></div>

\* CSS样式

\* JavaScript

\* 获取容器元素

\* 调用masonry()

\* Masonry插件的核心方法 - masonry()

\* 选项

\* itemSelector

\* columnWidth

\* isFitWidth

\* ...

\* 方法

\* 事件

\* 开发插件(了解) - 面试

\* JQUERY插件种类

\* 对象方法的插件 - $().方法名()

\* 全局函数的插件 - $.方法名()

\* 选择器插件

\* 这一类插件 - 用于扩展JQUERY选择器

\* 这一类目前基本不用

\* 典型的代表 - XPath插件

\* JQUERY制作插件的机制

\* 制作对象方法 - $.fn.extend(object)

\* 制作全局函数 - $.extend(object)

\* 扩展内容

\* md文件 - Markdown格式

\* markdown是一种文本编辑的标记语言

\* markdownPad 2 - 收费的软件

\* json文件 - 数据格式

\* json - 目前最流行的数据交换格式

\* 这个内容会在Ajax的课程中

\* CDN加速

\* CDN引入与本地引入的区别?

\* CDN引入

\* 不需要下载JS库文件

\* 带宽和服务器性能一般比较高:加载速度快

\* 提供JS库所有的版本文件

\* 本地引入

\* 下载到本地对应的JS库文件

\* 由于服务器的配置和带宽都不很高:加载速度稍慢

\* JS库文件的版本管理

HTML5 第一天

HTML5 DAY01:

\* 基础内容

\* 发展历史

\* HTML5的最新规范是2014年10月底发布的

\* HTML5的规范经历了近8年的时间

\* HTML5成为HTML的最后一版本

\* 原因 - HTML5页面的声明猜测出来的

\* 注意 - 不再以版本，快速迭代方式

\* 说明

\* HTML5的内容，学习的很好：首先JavaScript学好

\* HTML5的特点

\* 融合之前所有的HTML技术

\* HTML5的发展方向

\* 非常适合完成移动端的功能

\* 视频类的技术

\* 网页游戏方向

\* 传统的网站和Web应用

\* 新表单（<form>）

\* INPUT新类型

\* email - 电子邮件

\* 作用 - 提供表单验证功能

\* 实际 - 验证当前字符串中是否包含“@”

\* search - 搜索

\* 注意 - 并没有发现提供提示框功能

\* url - 地址

\* 作用 - 提供表单验证功能

\* 实际 - 验证当前字符串中是否包含“http://”

\* number - 数字

\* 作用 - 提供表单验证功能

\* 实际 - 验证当前字符串是否为数字（Number）

\* 属性

\* max - 最大值

\* min - 最小值

\* step - 步长

\* range - 范围

\* 作用 - 提供页面效果（滑块）

\* 属性

\* max - 最大值

\* min - 最小值

\* step - 步长

\* value - 当前值

\* color - 颜色

\* date - 日期

\* 显示出来的 - yyyy/MM/dd

\* 实际日期格式 - yyyy-MM-dd（标准格式）

\* 表单新元素

\* <datalist>元素

\* 作用 - 用于封装HTML元素所使用的数据内容

\* 注意

\* 该元素本身不显示在HTML页面中

\* 配合<option>元素，预定义数据内容

\* 配合<input type="text">元素，显示在HTML页面中

\* list属性 - 关联<datalist>元素

\* <progress>元素

\* 作用 - 提供一个进度条

\* 注意

\* 主流浏览器目前都支持，但是样式不同

\* <meter>元素

\* 作用 - 提供一个刻度

\* <output>元素

\* 作用 - 提供输出元素

\* 表单新属性

\* placeholder属性

\* 作用 - 为输入框<input>元素提供默认提示信息

\* autofocus属性

\* 作用 - 自动获取焦点

\* 注意 - 只需要设置名称，而不需要值

\* multiple属性

\* 作用 - 允许单个输入框<input>元素输入多个值

\* 注意

\* 多个值之间使用逗号隔开

\* 只需要设置名称，而不需要值

\* form属性

\* 作用 - 允许表单元素定义在表单之外

\* 表单新验证

\* 验证属性

\* required属性

\* 作用 - 验证表单元素的值是否为空

\* pattern属性

\* 作用 - 验证表单元素的值是否与指定的正则表达式匹配

\* 注意

\* 不能为空逻辑的验证，不能使用pattern属性

\* min和max属性

\* 作用 - 验证表单元素的值的范围（包括两个边界）

\* 含义

\* min - 最小值

\* max - 最大值

\* minlength和maxlength属性

\* 作用 - 验证表单元素的值长度的范围（包括两个边界）

\* 注意

\* minlength其实并不是HTML5的新特性

\* minlength没有出现在W3C的标准中

\* maxlength准确算作是限制属性

\* 当用户输入时进行限制

\* 练习题

\* 用户注册表单 - INPUT新类型、默认提示

\* 用户名：自动获取焦点，不能为空，6-18位的英文+数字

\* 密码：不能为空，6-8位的英文

\* 确认密码：不能为空，6-8位的英文

\* Email：不能为空，格式正确

\* 年龄：不能为空，最小为18，最大为50

\* 个人主页：不能为空

\* 出生日期：不能为空

HTML5 第二天

年前课程内容回顾：

\* JQUERY

\* JQUERY选择器

\* JQUERY的DOM操作

\* 对比原生DOM操作来学习

\* 内容

\* 基本操作

\* html()

\* text()

\* val()

\* attr()

\* 样式操作

\* addClass()

\* removeClass()

\* toggleClass()

\* hasClass()

\* attr("class",className)

\* css()

\* 遍历元素

\* 父节点 - parent()

\* 子节点 - children()

\* 兄弟节点

\* 上一个兄弟节点 - prev()

\* 下一个兄弟节点 - next()

\* 替换元素

\* replaceWith()

\* replaceAll()

\* 删除元素

\* remove()

\* empty()

\* 创建元素

\* $(HTML代码)

\* JQUERY的事件

\* ready() - 类似于window.onload事件

\* 事件绑定与解绑

\* 模拟事件

\* 动画效果

\* 显示与隐藏 - show()和hide()

\* 滑动效果 - slideUp()和slideDown()

\* 淡入淡出 - fadeIn()和fadeOut()

\* 自定义动画 - animate()

\* 类数组操作

\* each()迭代

\* JQUERY UI

\* 效果

\* animate() - 扩展JQUERY的animate()

\* 组件

\* 使用JQUERY插件

\* 日期插件

\* 表单验证插件

\* 瀑布流插件

\* 开发JQUERY插件

\* 对象方法插件 - $.fn.extend(object)

\* 全局函数插件 - $.extend(object)

\* 选择器插件 - 几乎不用

\* SERVER&HTTP

\* MYSQL

\* PHP

\* HTTP

\* HTML5

\* 新表单

\* INPUT新类型

\* email类型

\* url类型

\* date类型

\* number类型

\* 表单新元素

\* datalist元素 - 数据与结构的分离

\* progress元素 - 进度条

\* meter元素 - 刻度

\* output元素 - 输出

\* 表单新属性

\* placeholder - 替换value属性

\* autofocus - 自动获取焦点

\* multiple - 允许输入多个值

\* form

\* 表单新验证

\* 验证属性

\* min和max属性

\* minlength和maxlength属性

\* required属性

\* pattern属性

\*

HTML5 DAY02:

\* 表单新验证

\* 验证属性

\* required属性 - 验证是否为空

\* pattern属性 - 匹配正则表达式

\* min和max属性 - 最小值和最大值

\* minlength和maxlength属性 - 最小长度和最大长度

\* validity属性

\* ValidityState 对象 - 属性

\* 是由HTML5底层逻辑所提供的（原本已经存在的）

\* 该对象开放了一个接口 - validity属性

\* 作用 - 通过该属性直接调用ValidityState对象

\* 有效状态 - 表单元素验证的逻辑

\* 有效状态

\* 基本内容

\* 作用 - 用于替代之前的手动逻辑判断

\* 获取 - 利用ValidityState对象的属性

\* 实际上通过Element.validity属性获取ValidityState对象

\* 分类

\* valid

\* 作用 - 验证表单元素是否正确

\* 返回值 - Boolean

\* true - 表示验证成功

\* false - 表示验证不成功

\* valueMissing

\* 作用 - 验证表单元素是否为空

\* 返回值 - Boolean

\* true - 表示为空（错误）

\* false - 表示不为空（正确）

\* 注意 - 必须配合required属性使用

\* typeMismatch

\* 作用 - 验证表单元素的类型是否匹配

\* 返回值 - Boolean

\* true - 表示不匹配

\* false - 表示匹配

\* 注意 - 必须配合email、url等类型使用

\* stepMismatch

\* 作用 - 验证表单元素的步长是否匹配

\* 返回值 - Boolean

\* true - 表示步长不匹配

\* false - 表示步长匹配

\* 注意 - 配合step属性使用

\* rangeUnderflow

\* 作用 - 验证表单元素是否小于最小值（min）

\* 返回值 - Boolean

\* true - 表示小于最小值

\* false - 表示不小于（大于等于）最小值

\* 注意 - 配合min属性使用

\* rangeOverflow

\* 作用 - 验证表单元素是否大于最大值（max）

\* 返回值 - Boolean

\* true - 表示大于最大值

\* false - 表示不大于最大值

\* 注意 - 配合max属性使用

\* tooLong

\* 作用 - 验证表单元素是否过长

\* 返回值 - Boolean

\* true - 表示过长

\* false - 表示正常

\* 注意

\* 配合maxlength属性使用

\* maxlength属性是限制属性，很难出现目前这种情况

\* tooShort

\* 作用 - 验证表单元素是否过短

\* 返回值 - Boolean

\* true - 表示过短

\* false - 表示正常

\* 注意 - 配合minlength属性使用

\* patternMismatch

\* 作用 - 验证表单元素是否与指定正则表达式匹配

\* 返回值 - Boolean

\* true - 表示不匹配

\* false - 表示匹配

\* 注意 - 配合pattern属性使用

\* customError

\* 作用 - 验证表单元素是否使用了自定义错误提示

\* 返回值 - Boolean

\* true - 表示曾经调用setCustomValidity()方法

\* false -

\* 注意 - 配合使用setCustomValidity()方法

\* setCustomValidity()方法

\* 作用 - 用于替换验证属性底层提供的默认错误提示信息

\* 注意 - 当表单输入正确时，这样编写setCustomValidity("");

\* 音视频处理

\* 视频处理

\* <video>元素

\* 所支持的视频格式

\* mp4格式 - 主流视频格式

\* ogg格式（.ogv） - 应用于移动端

\* webm格式 - 目前唯一支持超高清格式

\* 是由Google公司提供的

\* 引入单个视频文件

\* src属性 - 指定引入视频文件的路径

<video src="视频文件路径">

浏览器不支持视频的错误提示信息

</video>

\* 引入多个视频文件

\* 在<video>元素中使用<source />元素

<video>

浏览器不支持视频的错误提示信息

<source src="mp4文件" />

<source src="ogv文件" />

<source src="webm文件" />

</video>

\* 属性

\* width和height属性

\* 注意 - 并不像<img>元素（按比例进行缩放）

\* autoplay属性 - 自动播放

\* loop属性 - 循环播放

\* controls属性 - 提供一个控制面板

\* poster属性 - 播放视频之前提供一张图片显示

\* preload属性 - 表示是否自动加载视频

\* none - 表示不加载

\* auto - （默认）表示自动加载（尽快地视频加载完毕）

\* metadata - 表示除视频之外的所有信息（高宽等）

\* 高级编程用法

\* 事件

\* play - 当视频播放时被触发

\* pause - 当视频暂停时被触发

\* ended - 当视频播放完毕时被触发

\* error - 当视频播放错误时被触发

\* 方法

\* play() - 表示播放视频

\* pause() - 表示暂停视频

\* canPlayType(type) - 判断当前浏览器支持哪种视频格式

\* 属性

\* paused属性 - 表示是否暂停

\* ended属性 - 表示是否播放完毕

\* duration属性 - 返回当前视频的时长（单位为秒）

\* currentTime - 获取或设置媒体播放位置

\* 音频处理

\* <audio>元素

\* 所支持的音频格式

\* mp3格式

\* ogg格式

\* wav格式

\* 引入单个音频文件

<audio src="" autoplay></audio>

\* 引入多个音频文件

<audio>

<source src="" />

</audio>

\* 属性

\* autoplay属性

\* loop属性

\* controls属性

\* 扩展内容

\* 目前视频处理技术

\* 网页中几乎所有的视频都是由Flash

\* 浏览器原生并不支持Flash技术（第三方组件）

\* Flash技术的性能不好

\* Flash技术并不支持移动端（浏览器和APP）

HTML5 第三天

HTML5 DAY03：

\* Canvas

\* 基本内容

\* 是HTML5新特性中，目前最成熟技术（最重要）

\* HTML5提供一个新元素<canvas>

\* 作用 - 利用JS代码在HTML页面中绘制图形（二维图形）

\* 用途（实际应用）

\* Web应用 - 图表、动态动画

\* 网络游戏 - 网页游戏

\* 好处 - HTML页面中直接绘制图形（减少外部链接）

\* 如何使用Canvas

\* 实现步骤

\* 在HTML页面中定义<canvas>元素

\* 在JavaScript代码中

\* 获取<canvas>元素

\* 通过<canvas>元素创建画布对象

canvasElement.getContext('2d');

\* 通过画布对象进行绘制图形

\* 注意

\* 通过Canvas绘制的图形，最终是以图片格式出现

\* 通过Canvas绘制的图形与HTML页面是无关的

\* 通过Canvas绘制的图形

\* 不能通过DOM进行定位

\* 不能使用DOM事件（事件绑定到<canvas>元素）

\* 绘制基本图形

\* 绘制矩形

\* fillRect(x,y,width,height) - 实心矩形

\* x和y - 绘制矩形的左上角的坐标值

\* width和height - 绘制矩形的宽度和高度

\* strokeRect(x,y,width,height) - 空心矩形

\* x和y - 绘制矩形的左上角的坐标值

\* width和height - 绘制矩形的宽度和高度

\* clearRect(x,y,width,height) - 清除指定区域的矩形

\* x和y - 绘制矩形的左上角的坐标值

\* width和height - 绘制矩形的宽度和高度

\* 注意

\* 绘制图形的默认颜色为 黑色

\* 绘制空心矩形默认的边框宽度为 1px

\* 设置样式

\* fillStyle - 设置填充样式

\* strokeStyle - 设置描边样式

\* globalAlpha - 设置透明度（0 - 1）

\* 注意

\* 以上三个属性，一旦设置不再修改的话 - 始终起作用

\* 设置渐变

\* 线性渐变 - createLinearGradient(x1,y1,x2,y2)

\* 及其重要的概念 - 基准线（起点和终点）

\* 基准线的起点坐标值 - （x1,y1）

\* 基准线的终点坐标值 - （x2,y2）

\* 注意

\* 线性渐变的效果取决于基准线是怎么样的

\* 设置渐变颜色是怎么样的

\* 实现步骤

\* 设置线性渐变

var grad = context.createLinearGradient(x1,y1,x2,y2);

\* 通过渐变对象，设置渐变颜色

grad.addColorStop(position,color);

\* 将渐变效果，赋值给填充样式或描边样式

context.fillStyle = grad;

\* 进行绘制

context.fillRect(x,y,width,height);

\* 射线（扇形）渐变 - createRadialGradient(x1,y1,r1,x2,y2,r2)

\* 及其重要的概念 - 基准圆

\* 第一个基准圆 - （x1,y1）圆心，r1半径

\* 第二个基准圆 - （x2,y2）圆心，r2半径

\* 注意

\* 线性渐变的效果取决于基准圆是怎么样的

\* 设置渐变颜色是怎么样的

\* 实现步骤

\* 设置线性渐变

var grad = context.createRadialGradient(x1,y1,r1,x2,y2,r2);

\* 通过渐变对象，设置渐变颜色

grad.addColorStop(position,color);

\* 将渐变效果，赋值给填充样式或描边样式

context.fillStyle = grad;

\* 进行绘制

context.fillRect(x,y,width,height);

\* 设置渐变颜色 - addColorStop(position,color)

\* position - 设置渐变颜色的位置

\* 值范围 - （0 - 1）

\* color - 设置渐变的颜色

\* 绘制文字

\* 设置文字相关属性 - font

\* 绘制文字

\* 空心文字 - strokeText(text,x,y)

\* text - 绘制的文字内容

\* x和y - 绘制文字的坐标值

\* 实心文字 - fillText(text,x,y)

\* text - 绘制的文字内容

\* x和y - 绘制文字的坐标值

\* 对齐方式

\* 水平对齐 - textAlign

\* left - 左对齐

\* right - 右对齐

\* center - 居中对齐

\* 垂直对齐 - textBaseline

\* top - 上对齐

\* middle - 居中对齐

\* bottom - 下对齐

\* hanging - 悬挂基线

\* alphabetic - 字母基线

\* 设置阴影

\* shadowColor - 设置阴影颜色

\* shadowOffsetX - 设置阴影的水平偏移量

\* shadowOffsetY - 设置阴影的垂直偏移量

\* shadowBlur - 设置阴影程度

\* 扩展内容

\* 目前网页游戏的技术 - Flash

\* 网络游戏的要求

\* 技术

\* JavaScript - 基本用法、面向对象、设计模式、算法及数据结构

\* HTML5 - Canvas、WebWorker、WebSocket等

\* 物理引擎 - 模拟现实生活的场景

\* 粒子系统 - 绘制地图等

\* 设置颜色

\* 直接使用单词设置

\* 利用三原色（红绿蓝）原理

\* 使用rgb方式设置

\* 使用#xxxxxx设置

\* 事件对象event

\* pageX/offsetX/clientX/x - 鼠标的x轴值

\* pageY/offsetY/clientY/y - 鼠标的y轴值

\* 前端开发的难点

\* 所见非所得 - 看到的效果和实现的逻辑完全不同

HTML5 第四天

HTML5 DAY04：

\* 绘制图形

\* 创建路径

\* 特点 - 比较灵活，适用场景较多

\* 标识方法

\* beginPath() - 开始创建路径

\* closePath() - 结束创建路径

\* 设置方法

\* rect(x,y,width,height) - 设置矩形

\* arc(x,y,radius,startAngle,endAngle,direction) - 设置圆形（弧形）

\* x和y - 设置圆形的圆心坐标值

\* radius - 设置圆形的半径

\* startAngle - 设置圆形的起点（0 - 2PI）

\* endAngle - 设置圆形的终点（0 - 2PI）

\* direction - Boolean值，是顺时针还是逆时针

\* 默认值为false（顺时针）

\* moveTo(x,y) - 设置起点

\* lineTo(x,y) - 设置拐点或终点

\* 绘制方法

\* fill() - 绘制实心

\* stroke() - 绘制空心

\* 实现效果

\* 绘制矩形

context.beginPath();

context.rect(x,y,width,height);

context.closePath();

context.fill()或context.stroke();

\* 绘制圆形

context.beginPath();

context.arc(x,y,radius,startAngle,endAngle,direction);

context.closePath();

context.fill()或context.stroke();

\* 绘制弧形

context.beginPath();

context.arc(x,y,radius,startAngle,endAngle,direction);

context.closePath();

context.fill()或context.stroke();

\* 绘制直线

context.beginPath();

context.moveTo(x,y);

context.lineTo(x,y);

context.stroke();

\* 绘制折线

context.beginPath();

context.moveTo(x,y);

context.lineTo(x,y);

context.lineTo(x,y);

context.stroke();

\* 绘制多边形

context.beginPath();

context.moveTo(x,y);

context.lineTo(x,y);

context.lineTo(x,y);

context.closePath();

context.stroke();或context.fill();

\* 设置线型

\* 设置线宽 - 默认宽度为 1px

lineWidth

\* 设置端点形状 - lineCap

\* butt - 默认值，平角

\* round - 圆角

\* square - 正方向线帽

\* 设置交点形状 - lineJoin

\* miter - 默认值，尖角

\* miterLimit - 设置尖角的程度

\* round - 圆角

\* bevel - 斜角

\* 处理图像（图片）

\* 绘制图像（图片）

\* drawImage(img,x,y)

\* img - 当前绘制的图像

\* x和y - 绘制图像的坐标值

\* 注意 - 使用图像的原大小

\* drawImage(img,x,y,width,height)

\* img - 当前绘制的图像

\* x和y - 绘制图像的坐标值

\* width和height - 绘制的高度和宽度

\* 绘制步骤

\* 加载一张图片

\* 绘制加载的图片

\* 注意 - 在绘制图片之前，必须保证图片加载完毕

\* 平铺图像 - createPattern(image,type)

\* image - 平铺的图像

\* type - 平铺的方式

\* repeat - 全方向平铺

\* repeat-x - 水平方向平铺

\* repeat-y - 垂直方向平铺

\* no-repeat - 不平铺

\* 切割图像 - clip()

context.beginPath();

设置切割的图形

context.closePath();

context.clip();// 切割

\* 画布方法

\* 状态方法

\* save() - 保存当前的Canvas的设置

\* restore() - 恢复之前保存的设置

\* 转换方法

\* scale(x,y) - 缩放

\* 作用 - 缩小和放大

\* x和y - 分别指水平方向和垂直方向的

\* 值范围

\* 值小于 1 - 缩小

\* 值等于 1 - 原大小

\* 值大于 1 - 放大

\* 注意 - 缩放的是整个画布

\* translate(x,y) - 平移

\* 作用 - 将绘制的图形移动到指定位置

\* x和y - 表示水平方向和垂直方向

\* 注意 - 相对上一次绘制的位置的平移 //

\* rotate(角度) - 旋转方法

\* 角度的计算公式 - Math.PI / 180 \* 角度

\* 注意 - 旋转的是整个画布 //以画布的左上角为轴

\* Chart.js库 - 图表

\* http://www.bootcss.com/p/chart.js/docs/

\* https://d3js.org/

\* 扩展内容

HTML5 第五天

HTML5 DAY05：SVG

\* 基础内容

\* SVG技术并不专属于 HTML5

\* SVG本身是一套独立图像技术

\* HTML5 之后，SVG技术被集成到 HTML5 新特性

\* HTML5 之前和之后使用SVG的区别

\* HTML5 之前，将SVG作为一张普通的图片进行引入

\* HTML5 之后，允许在HTML代码直接编写SVG图像

\* SVG在HTML页面的作用

\* 提供一些小图标或动画

\* SVG的优势

\* 允许使用文本编辑器创建和编写SVG内容

\* 可被搜索、索引及压缩等

\* 并不依赖分辨率

\* SVG和Canvas的区别

\* SVG

\* 不依赖分辨率

\* 并不适合实现游戏类

\* 适用于大型密集类应用（例如百度地图）

\* 原生支持事件

\* Canvas

\* 依赖分辨率

\* 实现网络游戏

\* Canvas图形不支持事件

\* 最终是以图片格式出现

\* 注意

\* 虽然SVG可以与DOM配合使用

\* 在DOM的内容中提供专属于SVG的内容

\* SVG技术

\* SVG被简单地翻译成 矢量图

\* 全称Scalable Vector Graphics，译为可缩放矢量图形

\* 使用 XML技术 的语法进行编写

\* 作用 - 嵌入到 HTML页面 中的图像

\* SVG文件（了解）

\* 扩展名为“.svg”

\* HTML5如何使用SVG

\* 图形元素 - 基本六种图形

\* 矩形元素

<rect x="" y="" width="" height=""></rect>

\* x和y - 绘制矩形的左上角的坐标值

\* width和height - 绘制矩形的宽度和高度

\* 圆形元素

<circle cx="" cy="" r=""><circle>

\* cx和cy - 绘制圆形的圆心坐标值

\* r - 绘制圆形的半径

\* 椭圆元素

<ellipse cx="" cy="" rx="" ry=""></ellipse>

\* cx和cy - 绘制椭圆的圆心坐标值

\* rx - 绘制椭圆的水平方向半径

\* ry - 绘制椭圆的垂直方向半径

\* 直线元素

<line x1="" y1="" x2="" y2=""></line>

\* x1和y1 - 直线的起点坐标值

\* x2和y2 - 直线的终点坐标值

\* 折线元素

<polyline points=""></polyline>

\* points - 设置折线的起点、折点（允许多个）和终点

\* 格式 - （x1,y1 x2,y2 x3,y3 ...）

\* 多边形元素

<polygon points=""></polygon>

\* points - 设置折线的起点、折点（允许多个）和终点

\* 格式 - （x1,y1 x2,y2 x3,y3 ...）

\* SVG使用的问题

\* 提供的元素过多 - 记不住（建议不记忆）

\* 绘制图形需要计算所有的坐标值 - 容易出错

\* 编写SVG图形效果的代码量很大 - 成本大

\* 实际开发使用SVG

\* 对于图像的处理 - 提供专门的开发（设计）工具 - 软件

\* Inkscape - 开源免费的

\* Illustrator - 收费的

\* Batik - Apache提供的

\* 在实际工作使用SVG

\* 不是以编程方式来使用 - 不是由Web前端开发人员负责

\* 是以设计方式来使用 - 是由UI设计人员

\* 最终目的

\* 不会学习研究太多关于SVG本身的内容

\* 特效元素

\* 渐变效果

\* 线性渐变

<linearGradient x1="" y1="" x2="" y2="">

<stop offset="" stop-color="" />

</linearGradient>

\* 射线（扇形）渐变

<radialGradient cx="" cy="" r="" fx="" fy=""></radialGradient>

\* 滤镜效果 - 高斯模糊

\* Two.js库

\* 基础内容

\* 官网 - http://jonobr1.github.io/two.js/

\* 特点

\* 可以使用 HTML5 提供的 Canvas、SVG和WebGL 技术

\* 如何使用Two.js库

\* 在HTML页面中

\* 引入Two.js库文件

\* 定义容器元素

\* 在JavaScript代码中

\* 获取HTML页面中的容器元素

\* 创建Two对象，将其添加到容器元素中

var two = new Two(params).appendTo(element);

\* 设置要绘制的图形

\* 进行绘制 - update()

\* 绘制图形 - 返回对应图形的对象

\* 绘制矩形 - makeRectangle(x, y, width, height)

\* 绘制圆形 - makeCircle(x, y, radius)

\* 绘制椭圆 - makeEllipse(x, y, width, height)

\* 绘制直线 - makeLine(x1, y1, x2, y2)

\* 绘制多边形 - makePolygon(ox, oy, r, sides)

\* 绘制星形 - makeStar(ox, oy, or, ir, sides)

\* 设置样式

\* fill - 设置填充样式

\* stroke - 设置描边样式

\* linewidth - 设置边框的宽度

\* cap - 设置端点形状

\* join - 设置交点形状

\* 动态效果

\* play() - 类似于setInterval()的效果

\* pause() - 类似于clearInterval()的效果

\* update() - 更新当前的设置

\* 扩展内容

\* HTML或CSS技术是不具备编程性

\* 允许定义变量或函数等内容，称之为编程性

\* SVG资料的相关链接

\* 属性参考 - https://developer.mozilla.org/zh-CN/docs/Web/SVG/Attribute

\* 元素参考 - https://developer.mozilla.org/zh-CN/docs/Web/SVG/Element

\* 开发Web应用

\* 开发人员

\* 服务器端

\* 前端（PC客户端）

\* 移动端（浏览器）

\* DBA（数据库）

\* 设计人员

\* UI（User Interface）设计人员

\* UE设计人员

\* 美工人员

\* 其他人员

\* 项目经理

\* 架构师

HTML5 第六天

HTML5 DAY06：

\* Geolocation（地理定位）

\* 基础内容

\* GPS（APGS） - 经度和纬度的交点（美国的）

\* 误差大概在 5 米之内

\* GPS芯片依靠定位卫星（颗数）

\* 北斗定位 - （国产的）

\* 惯性定位

\* 依靠卫星（23颗卫星）

\* 移动运营商的基站

\* 网络化定位

\* IP地址（唯一的）

\* HTML5的地理定位

\* 地理定位技术是由Google公司提供的

\* 百度地图 - JavaScript

\* 地址 - http://lbsyun.baidu.com/

\* 百度地图入门使用

\* 在HTML页面

\* 引入百度地图的JS库文件（在线文件）

http://api.map.baidu.com/api?v=2.0&ak=您的密钥

\* 定义容器元素 - 显示百度地图

\* 在JavaScript代码中

\* 创建百度地图对象

var map = new BMap.Map(容器元素的id);

\* 初始化百度地图（设置中心点坐标和显示级别）

map.centerAndZoom(point,zoom)

\* 百度地图API

\* 核心类

\* Map类 - 创建一个地图对象

\* 构造器 - Map(container) -

\* 作用 - 在指定的容器内创建地图实例

\* 实现百度地图场景

\* 如何在百度地图中使用比例尺、全景等功能?

\* 如何在百度地图实现定位效果（标注、信息提示功能）

\* 拖放（拖拽 + 投放）API

\* 注意 - HTML默认情况下不允许实现拖放效果

\* 源元素 - 被拖拽的元素（拖拽效果）

\* 源元素事件

\* dragstart - 源元素开始被拖拽时触发（一次）

\* drag - 源元素被拖拽的过程中，实时触发（多次）

\* dragend - 源元素结束被拖拽时触发（一次）

\* 目标元素 - 投放到的元素（拖放效果）

\* 目标元素事件

\* （必要）dragover - 源元素到达目标元素时触发（多次）

\* 调用阻止HTML页面的默认行为

event.preventDefault();

\* 成功地触发drop（投放事件）

\* dragenter - 源元素到达目标元素时触发（一次）

\* （必要）drop - 源元素被投放到目标元素时触发

\* 默认该事件没有被成功触发

\* 该事件被成功地触发 - 阻止HTML页面默认行为

event.preventDefault();

return false;

\* dragleave - 源元素（到达目标元素，）离开时触发

\* dataTransfer对象

\* 作用 - 类似于剪切板的作用

\* 该对象已经集成在event事件对象中

\* 方法

\* setData(type,data) - 将源元素的数据保存在dataTransfer对象中

\* getData(type) - 从dataTransfer对象取出源元素的数据

\* setDragImage()方法

\* 作用 - 用于修改鼠标拖拽源元素时显示的效果

\* 扩展内容

\* 目前开始着手复习

\* HTML + CSS

\* 通过实际案例来复习

\* 通过实际面试题来复习

\* JAVASCRIPT

\* ECMAScript - 面向对象

\* BOM + DOM

\* JQUERY + HTML5 + AJAX + SERVER

\* JQUERY - 面试题

\* HTML5 - 系统复习|案例

\* AJAX

\*

HTML5第七天

HTML DAY07：

\* Web Workers

\* 基础内容

\* 什么是Worker？

\* 就是一段运行在后台的JavaScript代码

\* 特点

\* 利用多核CPU的优势

\* 对“多线程”支持较好

\* 不会影响页面性能

\* 问题

\* 不能访问Web页面和DOM API

\* Worker API

\* 检测浏览器对Worker的支持情况

if( typeof(Worker) !== "undefined" ){

// 表示当前浏览器支持Web Worker

}else{

// 表示当前浏览器不支持，给出错误提示

}

\* 创建Web Worker对象

var worker = new Worker(worker文件路径);

\* 与Web Worker（文件）的通信

\* onmessage事件

\* 作用 - 用于监听Worker文件的执行情况

\* postMessage(data)方法

\* 作用 - 用于传递消息（数据或结果）

\* 终止与Web Worker的通信

\* terminate()方法

\* Worker语法

\* Worker对象

\* 创建对象 - new Worker(worker文件)

\* 方法 - postMessage()

\* 事件 - onmessage事件

\* Worker文件

\* 简单来说，就是一个普通的JS文件

\* 特殊地方 - 允许直接使用Worker对象的方法和事件

\* 注意

\* Worker文件允许定时器和navigator对象

\* 定时器 - setTimeout()和setInterval()

\* Worker文件中使用定时器

\* 在Worker文件中，允许直接使用Worker对象的方法和事件

\* 在Worker文件中，允许直接调用setTimeout()和setInterval()

\* Web存储API

\* 基础内容

\* 存储方式

\* 浏览器内存

\* 优点

\* 便捷

\* 浏览器内置

\* 缺点

\* 容量不大

\* 占用量直接影响浏览器性能

\* 周期由浏览器决定的

\* Cookie（缓存）

\* 优点

\* 容量比内存大

\* 用户硬盘空间

\* 存储周期较长

\* 缺点

\* 不能存储中文

\* 大小限制（单文件不能超过4KB）

\* 数量限制（一个网站最多200多个文件）

\* 不安全

\* 数据库 - 独立于任何开发语言

\* Web存储的分类

\* session Storage（会话存储） - 用于替代Session（会话）

\* 存储数据的周期（时间） - 浏览器从打开到关闭

\* local Storage（本地存储） - 用于替换Cookie

\* 存储数据的周期（时间） - 由开发人员决定

\* 注意

\* session Storage与local Storage提供的API几乎一致的

\* Session Storage

\* setItem(key, value) - 设置指定数据

\* getItem(key) - 获取指定数据

\* removeItem(key) - 删除指定数据

\* clear() - 清空所有数据（慎用）

\* key(index) - 根据索引值返回key值

\* length属性 - 返回存储数据的数量

\* Local Storage

\* storage事件 - 实现多页面之间共享数据

\* 该事件目前只有Safari

\* Web Sockets

\* 基础内容

\* socket协议

\* socket与http的区别

\* socket协议

\* 多用于C/S架构的软件

\* 实现实时连接

\* http协议

\* 多用于B/S架构的软件

\* 实现短连接

\* 无状态协议

\* Web Sockets

\* 允许在B/S架构下实现Socket协议的应用

\* Web Sockets API

\* 创建WebSocket对象

var socket = new WebSocket(url);

\* url - 当前HTML页面所请求的地址

\* 注意

\* 这个url地址的协议是 ws:// 或 wss://（加密）

\* WebSocket对象

\* 属性

\* readyState属性 - 表示服务器端的通信状态

\* OPEN - 数字值为 1，表示建立连接

\* CLOSED - 数字值为 2，表示关闭连接

\* 方法

\* send(data)方法 - 向服务器端发送请求数据

\* close()方法 - 关闭与服务器端之间的连接

\* 事件

\* onmessage事件 - 用于监听服务器端的通信状态

\* 在服务器端向客户端响应数据内容时触发

\* 作用 - 接收服务器端响应的数据内容

\* onopen事件 - 用于监听客户端与服务器端建立连接

\* 在客户端与服务器端建立连接时被触发

\* 作用 - 必要的初始化工作

\* onclose事件 - 用于监听客户端与服务器端关闭连接

\* 在客户端与服务器端关闭连接时被触发

\* 作用 - 必要的资源释放工作

\* 在HTML（客户端）如何实现WebSocket

\* 客户端与服务器端建立连接

// 创建Web Socket对象

var websocket = new WebSocket(url);

// 完成必要初始化

websocket.onopen = function(){}

// 接收服务器端的响应数据

websocket.onmessage = function(){}

// 释放必要的资源

websocket.onclose = function(){}

\* 客户端向服务器端发送数据

websocket.send(data);

\* 客户端与服务器端关闭连接

websocket.close();

\* 扩展内容

\* "!="与"=="和"!=="与"==="的区别

"1" == 1 - true

"1" === 1 - false

\*

Ajax第一天

AJAX DAY01：

\* 基础内容

\* 同步交互与异步交互

\* 同步交互 - 客户端向服务器端发送请求，到服务器端将响应结果返回。这个过程中，用户不能做任何事情。

\* 异步交互 - 客户端向服务器端发送请求，到服务器端将响应结果返回。这个过程中，用户可以做任何事情。

\* 异步交互的由来

\* Google公司在搜索、Map、Gmail等产品使用异步交互技术

\* 目前来看，有哪些应用使用着异步交互：

\* 搜索引擎产品

\* 基于页面实现的聊天功能

\* 网站中具有评论的功能

\* Ajax的概念

\* （不严格）客户端向服务器端发送请求，而无需刷新页面的技术

\* Ajax - Asynchronous JavaScript And Xml，译为异步的JavaScript和XML

\* 实现异步交互的技术

\* <iframe src="">元素

\* Flash技术

\* Ajax的注意

\* Ajax不是一个（种）技术

\* Ajax更不是新技术

\* Ajax包含技术

\* JavaScript（DOM）

\* XML技术

\* XMLHttpRequest

\* Ajax的核心对象

\* XMLHttpRequest对象 - 异步对象

\* XMLHttpRequest对象

\* 如何创建该对象

function getXhr(){

var xhr = null;//声明变量

if(window.XMLHttpRequest){// 其他浏览器

xhr = new XMLHttpRequest();

}else{// IE浏览器

xhr = new ActiveXObject("Microsoft.XMLHttp");

}

return xhr;

}

要求：

\* 将上述创建XMLHttpRequest对象的代码默写出来

\* 方法

\* open()方法 - 与服务器端建立连接

\* send()方法 - 向服务器端发送请求

\* setRequestHeader() - 设置请求头信息

\* 事件

\* onreadystatechange事件

\* 作用 - 监听服务器端的通信状态

\* 触发 - 服务器端的通信状态变化时

\* 属性

\* readyState属性 - 获取服务器端的通信状态

\* status属性 - 获取服务器端的状态码

\* responseText属性 - 返回服务器端响应的文本格式数据

\* responseXML属性 - 返回服务器端响应的XML格式数据

\* 参考资料

https://developer.mozilla.org/zh-CN/docs/Web/API/XMLHttpRequest

\* Ajax的实现步骤

\* 创建核心对象（固定写法）

\* 与服务器端建立连接

核心对象的open(method, url, async)方法

\* method - 请求类型（GET或POST）

\* url - 请求地址

\* async - 是否异步（最新规范中只有异步）

\* 向服务器端发送数据

核心对象的send(data)方法

\* data - 请求数据

问题

\* GET方式的请求

\* send()方法无效 - 改写xhr.send(null)

\* 请求数据被添加在url（请求地址）后面

\* POST方式的请求

\* 默认情况下，send()方法无效

\* 在调用send()方法前，设置请求头信息

xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

\* 接收服务器端的响应内容

\* 核心对象的onreadystatechange事件

\* 作用 - 用于监听服务器端的通信状态

\* 核心对象的readyState属性 - 服务器端的通信状态

\* 0 - （服务器端）未初始化

\* 1 - （与服务器端）正在连接

\* 2 - （服务器端）正在接收

\* 3 - （服务器端）正在响应

\* 4 - （服务器端）响应完毕

\* 核心对象的status属性 - 服务器端的状态码

\* 核心对象的responseText属性 - 接收响应数据

\* Ajax的GET与POST请求的区别

\* GET方式的请求

\* open()方法指定请求方式为GET

\* open()方法的url参数需要手动添加请求数据

\* send()方法是无效的

\* POST方式的请求

\* open()方法指定请求方式为POST

\* open()方法的url只是简单请求地址

\* send()方法是有效的（先设置请求头信息）

\* Ajax的实现步骤（其它）

\* 获取readyState属性值（最多）

\* 创建XMLHttpRequest对象

\* 注册监听（服务器端）

\* 与服务器端建立连接

\* 向服务器端发送请求数据

\* Ajax的应用 - 二级联动

\* 作业（扩展） - 思考是否可以将2次Ajax逻辑合并成一次?

Ajax第二天

AJAX DAY02：

\* XML格式 - 企业级开发

\* XML格式的基础内容

\* XML技术的发展历史

\* 具有一个野心 - 替代HTML

\* XHTML - 严格的HTML

\* XML技术逐步退出技术领域

\* XML的概念

\* 全称eXtensible Markup Language

\* 译为可扩展标记语言

\* XML的用途

\* 用于存储数据 - 文件存储

\* 用于应用配置 - 配置文件

\* XML和HTML的区别

\* HTML

\* 预定义很多元素

\* 都是W3C的标准

\* XML

\* 自定义元素

\* 都是W3C的标准

\* XML的语法

\* XML文件的扩展名为".xml"

\* JavaScript如何使用XML格式

\* 解析XML的步骤

\* 使用XML解析器将XML转换为 XML DOM对象

\* 使用DOM解析 XML DOM对象

\* XML解析器

\* 目前主流浏览器都内置了XML解析器

\* 创建读取XML字符串的解析器

function getXmlDoc(xml){

var xmlDoc = null;

if(window.DOMParser){// 其他浏览器

var parser = new DOMParser();//创建解析器

// 从xml字符串读取并解析为XML DOM对象

xmlDoc = parser.parseFromString(xml,"text/xml");

}else{// IE浏览器

// 创建IE浏览器的XML DOM对象

xmlDoc = new ActiveXObject("Microsoft.XMLDOM");

// 关闭异步加载

xmlDoc.async = false;

// 读取并解析XML字符串

xmlDoc.loadXML(xml);

}

return xmlDoc;

}

\* 注意

\* 读取并解析XML文件的话，使用的是AJAX的同步技术

\* AJAX的同步交互方式，官方规范不再建议

\* 目前主流浏览器不支持读取本地文件

\* AJAX如何使用XML格式

\* 请求数据使用XML格式

\* 在客户端如何构建符合XML格式的数据

\* 在服务器端如何接收并解析符合XML格式的数据

\* 响应数据使用XML格式

\* 在服务器端如何构建符合XML格式的数据

\* 在客户端如何接收并解析符合XML格式的数据

\* JSON格式 - 互联网开发

\* 基础内容

\* JSON的概念

\* 全称JavaScript Object Notation

\* 定义：是一种轻量级的数据交换格式

\* JSON的特点

\* 易于程序员阅读和编写

\* 易于计算机解析和生成

\* JSON这种数据格式，实际是源于JavaScript

\* JSON的结构

\* 在JavaScript中，称之为对象（字面量）

\* 在JavaScript中，称之为数组

\* JSON所支持的数据类型

\* JavaScript如何使用JSON

\* 定义JSON - 对象和数组的嵌套

\* 调用JSON - 对象和数组的调用

\* Ajax如何使用JSON

\* 请求数据使用JSON格式

\* 在客户端如何构建符合JSON格式的数据

\* 在服务器端如何接收并解析符合JSON格式的数据

\* 响应数据使用JSON格式

\* 在服务器端如何构建符合JSON格式的数据

\* 在客户端如何接收并解析符合JSON格式的数据

\* Ajax中使用的数据格式

\* 文本格式

\* 优点

\* 简单，容易

\* 不需要任何额外内容

\* 缺点

\* 不适合复杂结构的数据

\* 解析过程容易出错

\* XML格式

\* 优点

\* 结构化存储数据

\* 具有通用化

\* 可扩展性

\* 缺点

\* 单独地构建

\* DOM解析XML很复杂

\* JSON格式

\* 优点

\* 更轻量（对比XML）

\* 解析容易（不易出错）

\* 主流语言都支持

\* 缺点

\* 不易过多嵌套

\* 开发语言具有特殊要求

\* 扩展内容

\* DOM的组成部分

\* DOM CORE - 核心DOM

\* HTML DOM - 针对HTML

\* XML DOM - 针对XML

\* CSS DOM - 针对CSS

\* 作业 - 二级联动使用JSON格式

Ajax第三天

AJAX DAY03：

\* JQUERY中的AJAX

\* 第一层 - 将原生Ajax的用法进行封装

\* $.ajax()方法 - 最接近原生Ajax的用法（最麻烦）

\* 第二层 - 基于第一层的再次封装

\* $().load(url, data, callback)方法 - 最简单（具有局限性）

\* 问题

\* 无法自定义请求方式（由是否发送请求数据决定）

\* 默认使用文本格式数据（XML和JSON可以手动转换）

\* 参数

\* url - 请求地址

\* data - 请求数据，格式为JSON格式中的对象结构

\* callback - 回调函数

\* 作用 - 用于接收服务器端的响应数据

function(data){} - 利用形参data进行接收

\* $.get(url, data, callback, type)方法 - 是以GET方式请求

\* url - 请求地址

\* data - 请求数据，格式为JSON格式中的对象结构

\* callback - 回调函数

\* 作用 - 用于接收服务器端的响应数据

function(data){} - 利用形参data进行接收

\* type - 设置响应数据的格式

\* 注意 - 需要服务器端进行配合

\* 备选项

\* text - 文本格式

\* xml - XML格式

\* json - JSON格式

\* $.post(url, data, callback, type)方法 - 是以POST方式请求

\* 第三层 - 基于第二层的再次封装

\* $.getScript()方法 - 动态获取JS脚本

\* $.getJSON()方法 - 获取JSON格式的数据

\* 表单的异步提交

\* 表单提交的方式

\* 通过submit按钮实现表单提交

\* 通过绑定onsubmit事件实现表单提交

\* 在JavaScript获取表单的方式

\* 通过通用获取方式

\* document.forms，获取当前页面中所有表单

\* document.表单名，获取表单

\* 表单的异步提交

\* 阻止表单内部提交机制

\* 不使用submit按钮

\* 表单的submit事件中使用"return false"

\* 为submit按钮绑定click事件中使用"return false"

\* 手动获取表单内所有元素的数据

\* 通过Ajax的异步方式进行提交

\* 原生Ajax

\* jQuery中的Ajax

\* 跨域请求

\* 基础内容

\* 域

\* 域名 - 虚拟地址（绑定真实IP）

\* 顶级域名 - www.baidu.com

\* 二级域名 - news.baidu.com | blog.baidu.com/tarena

\* 三级域名 - tarena.blog.baidu.com

\* 同域

\* 万维网默认协议具有同源策略（不允许跨域的）

\* 跨域

\* 完全跨域 - 不同IP地址

\* 跨子域 - IP地址相同，端口号不同

\* 实现跨域

\* JSONP（JSON with Padding，JSON的使用模式）

\* 利用HTML中的<script>元素的开放策略，动态获取不同域的JSON格式的资料

\* jQuery中的Ajax

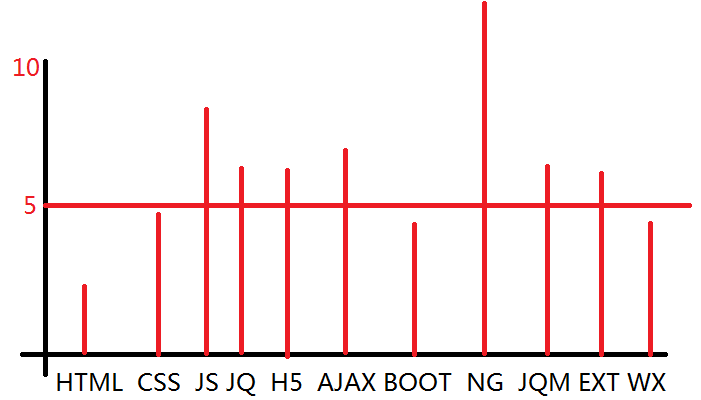
\* $.getJSON()方法

\* $.ajax()方法

第四阶段bootStrap第一天

第四阶段课程安排：

1. Bootstrap 4+2
2. AngularJS 3+2
3. WebApp 3+2
4. ExtJS 2
5. 微信 2半天

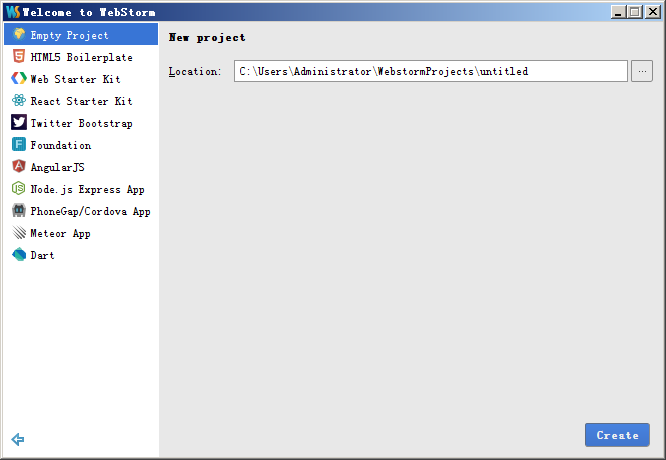


1. 常用的前端开发软件

(1)小型软件(xMB)：Notepad++ VIM Editplus

(2)中型软件(xxMB)：SublimeText

(3)大型软件(xxxMB)：Dreamweaver、Aptana(Eclipse)、WebStorm——编辑、代码提示、项目管理、调试、代码监视跟踪



1. 什么是响应式网页？

2000年：2.5G手机浏览网页——WML

2009年：3G智能手机自带浏览器浏览网页——HTML

问题：PC网页一般很大，放在手机中浏览必须缩小，影响浏览体验。解决方法：编写两套网页：一套宽度很大，一套宽度很小。

2010年：响应式网页概念提出——一个网页，可以适应各种不同的屏幕。响应式网页一般适合于企业门户、电子政务、资讯类....一屏下内容不是非常多的页面；不适合于京东首页等大型页面。

响应式网页应具备如下三个因素：

(1)流式布局

(2)图片和文字大小可变

(3)使用CSS3媒体查询技术

1. 如何测试一个响应式网页？

**(1)使用真实物理设备测试**

优势：测试结果真实可靠

劣势：测试任务量太大

**(2)使用模拟软件测试**

优势：无需购置大量设备，可以虚拟任意的硬件

劣势：部分功能无法测试，测试结果有待进一步的验证

**(3)使用浏览器(Chrome)自带的设备模拟器测试**

优势：无需购置大量设备，模拟常见的设备，模拟的功能选项丰富

劣势：部分功能无法测试，测试结果有待进一步的验证

|  |
| --- |
| viewport(视口)的由来：  2009年之前的网页都是针对PC屏幕而编写，布局框宽度很大且固定。3G智能手机可以浏览这样的HTML网页，但受手机屏幕宽度限制，只能把很宽的页面做压缩，放在手机屏幕中显示：    影响用户的浏览体验。iOS第一次提出了“viewport”概念——网页是在viewport中查看，而不是直接在屏幕中查看；视口的宽和高可以使用程序随意指定！    效果：大型网页可以不经缩放直接显示在手机——只是需要用户左右滑动屏幕，来看到更多的内容。  后来，android也引入了视口的概念；此概念只适用于只能手机，PC浏览器直接忽略。 |

4.如何编写响应式网页？——重点

(1)方法1：使用某种响应式网页框架，如Bootstrap

(2)方法2：手写响应式网页样式，具体技术点有：

1)声明viewport元标签——响应式网页必备！

<meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no">

1. 使用响应式图片

img {

/\*width: 150px;\*/

/\*width: 50%; 在父容器中的占比\*/

max-width: 50%; /\*在父容器中的占比，且不能超过图片自身原始大小——防止图片无限放大产生失真\*/

}

3)使用流式布局 布局元素使用浮动或行内块

4)避免使用绝对尺寸(如px)，使用相对单位(如em/rem/%)

5)使用CSS3MediaQuery技术

5.使用CSS3媒体查询编写响应式网页——最重点

Media：指浏览网页的设备，如：screen(pc/pad/phone/watch)、print、tv、projection、tty(针式打印机/cmd)、braille(读屏软件)...

MediaQuery：查询出当前浏览设备类型、分辨率、色彩深度、方向(landscape/portait)等等特性。

CSS3 MediaQuery：根据当前浏览设备的特性有选择性执行某些CSS样式，而忽略其他一些样式。

具体有两种用法：

**(1)有选择性的执行某些外部CSS文件**

<link media="screen and (min-width:768px) and (max-width:991px)" rel="stylesheet" href="css/pad.css"/>

特点：只有媒体查询结果为真的外部样式文件才会被执行，否则不执行！不足：即使不符合当前浏览设备的外部样式文件，也会被浏览器请求！

**(2)有选择性的执行某段CSS中的部分样式**

@media screen and (min-width:768px) and (max-width:991px) {

.box {

margin: 6px;

...

}

}

使用媒体查询完成如下练习：



|  |
| --- |
| 三大开源软件集散地：  (1) sourceforge.net  (2) code.google.com  (3) github.com |

5.Bootstrap概述

Bootstrap：引导、起步、引导程序

Twitter Bootstrap： 是一个由Twitter的两个工程师开源的CSS框架，托管在github.com。国内镜像网站：www.bootcss.com

主要内容分为五部分：

1. 起步
2. 全局CSS样式
3. 组件
4. JavaScript插件
5. 定制
6. Bootstrap第一部分：起步

(1)下载

(2)基本模板

(3)实例精选——第三天再回过头来看

(4)兼容性提示——Bootstrap不支持Windows下的Safari！

(5)初体验Bootstrap的魅力

|  |
| --- |
| 行业小知识： lang: language自然语言  <html lang="zh-CN"></html>  <html lang="zh-TW"></html>  <html lang="zh-HK"></html>  <html lang="zh-SG"></html>  <html lang="zh"></html>  <html lang="en-US"></html>  <html lang="zh-UK"></html>  <html lang="jp-JP"></html>  作用：(1)用于为浏览器确定当前页面的自然语言，可基于此执行网页翻译操作  (2)为盲人的读屏软件确定页面的基础语言和发音 |
| <meta http-equiv="X-UA-Compatible" content="IE=edge">  X-UA-Compitable： Cross-UserAgent-Compitable  跨IE浏览器版本兼容性；此元标签仅对IE系列浏览器有效。  IE 6：特性不兼容行业标准  IE 7：尽可能的兼容更多行业标准，同时保留了一个IE6内核  IE 8：尽可能的兼容更多行业标准，同时保留了一个IE6/7内核  IE 9：尽可能的兼容更多行业标准，同时保留了一个IE6/7/8内核  IE 10：尽可能的兼容更多行业标准，同时保留了一个IE6/7/8/9内核  IE 11：兼容行业标准的大部分特性，同时保留了一个IE6/7/8/9/10内核 |
| **html5shiv.js**：自调函数，无需手工参与，由@afarkas @jdalton @jon\_neal @rem编写，让老版本IE(<=IE8)能够理解HTML5新标签 |
| **respond.js：**自调函数，无需手工参与，由Scott Jehl编写，让老IE支持响应式网页(CSS3媒体查询技术) |

课后作业：

1.em和rem的区别？

2.继续完成“手工编写响应式网页示例”

3.预习手册“全局CSS样式”

第四阶段bootStrap第二天

复习：

Twitter Bootstrap是一个HTML / CSS / JS框架，适用于移动设备优先的响应式Web项目。涉及到：

HTML：为H5已有标签扩展了自定义属性 data-\*

CSS： (1)CSS Reset **(2)定义了千个class**

JS：基于jQuery创建了十几个插件函数

内容分为五部分：

(1)起步：下载、模板、示例、兼容性、禁用响应式

(2)全局CSS样式

(3)组件

(4)JS插件

(5)定制

1.全局CSS样式——CSS Reset

Bootstrap重置了HTML5标签的默认样式，屏蔽了浏览器的不兼容问题。

\* { box-sizing: border-box; }

html { font-family: sans-serif; font-size: 10px;}

body { margin:0;

font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;

font-size: 14px;

line-height: 1.42857143;

color: #333;

background-color: #fff;}

h1 { font-size: 36px; margin-top: 20px; margin-bottom: 10px;}

h2 { font-size: 30px; margin-top: 20px; margin-bottom: 10px;}

h3 { font-size: 24px; margin-top: 20px; margin-bottom: 10px;}

h4 { font-size: 18px; margin-top: 10px; margin-bottom: 10px; }

h5 { font-size: 14px; margin-top: 10px; margin-bottom: 10px; }

h6 { font-size: 12px; margin-top: 10px; margin-bottom: 10px; }

p { margin: 0 0 10px; }

a { background:transparent; color:#337ab7; text-decoration:none; }

a:hover { color: #23527c; text-decoration: underline; }

img { vertical-align: middle; border: 0; }

table { background: transparent; border-spacing:0; border-collapse:collapsed; }

div { }

span { }

2.全局CSS样式——按钮相关样式

三种形式的按钮： button input a

基础按钮： .btn .btn-default

五种常用色：

.btn-danger

.btn-warning

.btn-success

.btn-info

.btn-primary

四种大小：

.btn-lg

默认

.btn-sm

.btn-xs

块级按钮：

.btn-block

3.全局CSS样式——图片相关样式

.img-rounded 圆角图

.img-circle 圆形图

.img-thumbnail 缩略图

.img-responsive 响应式

4.全局CSS样式——字体和排版相关样式

修改文本颜色：

.text-五种颜色

修改文本背景颜色：

.bg-五种颜色

改变大小写：

.text-uppercase

.text-lowercase

.text-capitalize 首字母大写

修改文本对齐方式：

.text-left

.text-center

.text-right

.text-justify 两端调整对齐

文本隐藏：

.text-hide

5.全局CSS样式——表格相关样式

.table

.table-bordered 带边框的

.table-striped 条状的/隔行变色的

.table-hover 带悬停效果的

.table-responsive 响应式表格，注意：此class必须用于TABLE的父元素div上！不能用于TABLE本身！！

|  |
| --- |
| Bootlint：是Bootstrap官方提供的JS文件，用于检测HTML文档中的标签以及class使用方面的不恰当之处——默认情况下，浏览器会不会进行这些检查的。使用方法：  <script>  (function () {  var s = document.createElement("script");  s.onload = function () {  bootlint.showLintReportForCurrentDocument([]);  };  s.src = "js/bootlint.js";  document.body.appendChild(s)  })();  </script> |

5.5.全局CSS样式——辅助工具类

.pull-left

.pull-right

.clearfix

.center-block

.show

.hidden

.invisible

.list-unstyled

.list-inline

1. 全局CSS样式——栅格布局系统——最重点！！

**Bootstrap把浏览器屏幕分为四类：**

1. 大屏/大型PC屏幕——large/lg

width>1200px

1. 中等屏/普通PC屏幕——medium/md

992px<width<1200px

1. 小屏/平板电脑屏幕——small/sm

768px<width<992px

1. 超小屏/手机屏幕——extra small/xs

width<768px

**两个容器class:**

1. .container：宽度固定的容器

lg: 1170px md:970px sm: 750px xs:100%

1. .container-fluid：宽度可变的容器/液态容器

|  |
| --- |
| 面试题：Bootstrap中的两个容器类有哪些样式设定？  .container: 媒体查询设定几个宽度固定值、padding、margin  .container-fluid: 宽度始终是100%、padding、margin  重点：两个容器类都有前置和后置内容生成，可以防止子元素的margin越界、子元素的float对容器及后续元素造成影响。 |

网页内容布局方式：

1. TABLE布局：

优点：简单，容易控制

缺点：加载速度慢、语义有问题

1. DIV+CSS布局

优点：渲染速度快

缺点：不容易控制

(3)Bootstrap栅格布局：

优点：简单容易控制、渲染速度快、支持响应式

缺点：没有！

Bootstrap栅格布局系统的使用：

(1)栅格系统中的所有内容都必须放置的“容器”中。

(2)容器中可以放置 div.row 元素。

(3).row元素中可以放置 div.col 元素。

(4).col 元素中若想继续分为多份，可以进一步嵌套 .row>.col

(5)一个.row默认均分为12等分，每个.col可以指定其中的占比

(6)不同宽度的屏幕下，需要使用不同的列，如：

.col-xs-1 ~ .col-xs-12 col-xs-\*对xs/sm/md/lg屏幕都有效

.col-sm-1 ~ .col-sm-12 col-sm-\*对sm/md/lg屏幕都有效

.col-md-1 ~ .col-md-12 col-md-\*对md/lg屏幕都有效

.col-lg-1 ~ .col-lg-12 col-lg-\*对lg屏幕都有效

总结：col宽度对当前屏幕以及更大的屏幕有效，对更小的屏幕无效！

(7)一个DIV可以在不同的屏幕下指定不同的宽度

<div class="col-md-3 col-sm-6 col-xs-12">响应式列</div>

(8)col可以指定在某种屏幕下的偏移量(offset)

.col-xs-offset-1 ~ .col-xs-offset-12

.col-sm-offset-1 ~ .col-sm-offset-12

.col-md-offset-1 ~ .col-md-offset-12

.col-lg-offset-1 ~ .col-lg-offset-12

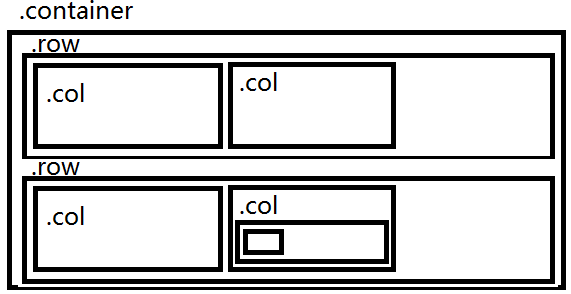
(9)col可以指定在某种特定的屏幕下隐藏

.hidden-xs 当前列仅在xs屏幕下隐藏

.hidden-sm 当前列仅在sm屏幕下隐藏

.hidden-md 当前列仅在md屏幕下隐藏

.hidden-lg 当前列仅在lg屏幕下隐藏



使用栅格布局系统练习编写响应式网页：

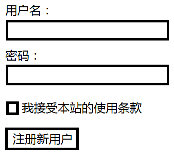


7.全局CSS样式——表单相关样式——难点！

Bootstrap提供了三种形式的表单：

**(1)默认表单**



<form>

<div class="form-group">

<label for=""></label>

<input class="form-control">

<span class="help-block">

</div>

</form>

**(2)行内表单(inline)**



<form class="form-inline" action="">

<div class="form-group">

<label class="sr-only" for="uname2">用户名：</label>

<input class="form-control" type="text" id="uname2" placeholder="用户名..."/>

</div>

</form>

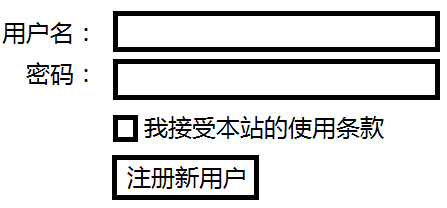
**(3)水平表单(horizontal)**

水平表单=表单+栅格布局系统

.container === form.form-horizontal

.row === .form-group

.col-\*-\* === .col-\*-\*



作业：

1. 如何设置手机中网页的默认字体？
2. 使用栅格布局系统练习编写响应式网页示例。
3. 查看手册：解决水平表单中label文字右对齐的问题——深究为什么！

第四阶段bootStrap第三天

复习：

Twitter Bootstrap是一个HTML/CSS/JS框架，适用于移动设备优先的响应式项目。内容涉及到：

HTML: 为H5已有标签自定义属性 data-\*

CSS: (1)CSSReset (2)千个class

JS: 基于jQuery创建了十几个插件函数

分为五部分：

(1)起步：下载、模板、示例、Bootlint、兼容性、禁用响应式

(2)全局CSS样式：Reset、按钮、图片、文本、列表、表格、栅格布局、表单

(3)组件： 基于HTML标签，创建一个特别的效果元素

(4)jQuery插件：

(5)定制

1.Bootstrap第三部分：组件——Glyphicon图标字体

图标字体：内容不是对应一个又一个文字，而是一个又一个的图标。



http://glyphicons.com/提供了一套可以适用于Web项目中常见的图标符号字体——Glyphicon Halflings。

Bootstrap中免费提供了这套字体中的260个，作为服务器端字体，在客户端浏览器Bootstrap页面时，可以从服务器端自动下载相关的字体文件。

使用方法： **<span class="glyphicon glyphcion-\*"></span>**

|  |
| --- |
| 如何在网页中使用服务器端字体(让客户端下载字体文件)：  @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';  } |

练习：创建几个按钮，每个按钮上放一个下述图标：

用户、主页、配置、下拉菜单、刷新、前进、后退、打分(星号)、我喜欢、删除、对勾、垃圾箱、定位、加号、减号、搜索、邮件

2.Bootstrap第三部分：组件——按钮组

.btn-group

.btn-group-justified 需要配合.btn-group

.btn-group-vertical 单独使用

3.Bootstrap第三部分：组件——下拉菜单——小重点

下拉菜单必需三级结构：

<div class="dropdown"> 相对定位

<a data-toggle="dropdown">触发元素</a>

<div/ul class="dropdown-menu"> 绝对定位

隐藏菜单

</div/ul>

</div>

1. Bootstrap第三部分：组件——导航(Nav)

Boostrap提供了两种形式的导航组件：

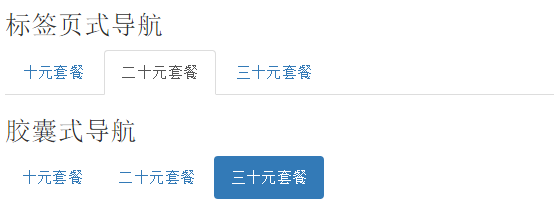
(1)标签页式导航

<ul class="nav nav-tabs">

(2)胶囊式导航

<ul class="nav nav-pills">

<ul class="nav nav-pills nav-stacked">



午间自学：面包屑(breadcrumb)、分页条(pagination)、翻页(pager)、徽章(badge)、标签(label)、巨幕(jumbotron)、页头(page-header)、缩略图(thumbnail)、水井(well)、进度条(progress/progress-bar)

**记住每个组件的class！！！**

5.Bootstrap第三部分：组件——警告框——小重点

<div class="alert alert-四种颜色 alert-dismissible">

<span data-dismiss="alert" class="close">&times;</span>

</div>

6.Bootstrap第三部分：组件——媒体对象——小重点

<div class="media">

<div class="media-left">图片</div>

<div class="media-body">文字</div>

<div class="media-right">图片</div>

</div>

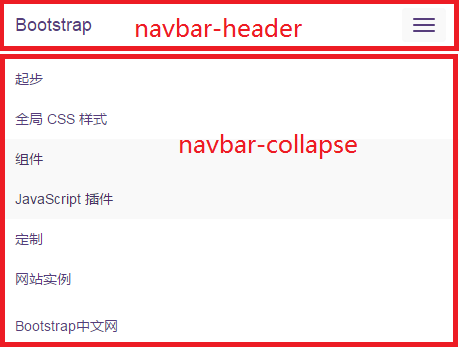
7.Bootstrap第三部分：组件——响应式导航条——最重点&难点

响应式导航条在平板/PC上的效果：



响应式导航条在PHONE上的效果：





Bootstrap中提供的导航条分为如下几类：

(1)按颜色分为：

浅色底深色字： navbar-default

深色底浅色字： navbar-inverse

(2)按定位方式分为：

相对定位(relative)： 默认

固定定位(fixed)： navbar-fixed-\*

(3)按所在位置分为：

页面顶部导航条： 默认 navbar-fixed-top

页面底部导航条： navbar-fixed-bottom

作业：

1. 仿写“起步”中的示例
2. 总结：今天学过哪些data-\*属性，各表什么含义？
3. 反复演练“响应式导航条”

第四阶段bootStrap第四天

复习：

TwitterBootstrap是一个HTML/CSS/JS框架，适用于移动设备优先的响应式项目。主要涉及到：

HTML: 为H5标签自定义属性 data-\*

**CSS: (1)CSSReset (2)千个class**

JS:基于jQuery提供了十几个插件函数

分为五部分：

1. 起步
2. 全局CSS样式
3. 组件：下拉菜单、按钮组、导航、面包屑、标签、徽章、巨幕、页头、进度条、水井、媒体对象、警告框、导航条
4. jQuery插件
5. 定制
6. jQuery插件函数概述

Bootstrap提供了十二个插件函数——大部分在jQueryUI库基本都对应的内容——为了精准控制样式，代码偏复杂。

这12个插件函数有两种使用方式：

**(1)传统JS调用方式：**

例如： $('xxx').dropdown( );

**(2)使用data-\*扩展属性方式：**

例如：<a data-toggle="dropdown">

在Bootstrap底层有如下代码：

$('[data-toggle="dropdown"]').dropdown();

2.Bootstrap提供的jQuery插件——下拉菜单

方式1：data-\*属性方式：

<a data-toggle="dropdown" href="#">产品大全</a>

方式2：JS编码方式：

<a id="bt" href="#">产品大全</a>

<script>

$('#bt').dropdown();

</script>

3.Bootstrap提供的jQuery插件——标签页

方式1：data-\*

<li><a data-toggle="tab" href="#">十元套餐</a></li>

方式2：js调用

<li><a href="#">十元套餐</a></li>

$('#list2 a').click(function(e){

e.preventDefault();

$(this).tab('show');

})

4.Bootstrap提供的jQuery插件——警告框

<div class="alert alert-danger alert-dismissible">

<span data-dismiss="alert" class="close">&times;</span>

</div>

·

|  |
| --- |
| 行业小知识：无障碍阅读  无障碍阅读网页：为了让阅读有障碍的人士(视力障碍、色盲、行动障碍、智力障碍)可以正常阅读的网页。  W3C在制定HTML5标准时，为所有的HTML标签指定了两组属性：  <ANY role="" aria-\*="" ></ANY  ARIA: Accessible Rich Internet Application |

1. Bootstrap提供的jQuery插件——弹出组件

(1)工具提示——Tooltip

(2)弹出框——Popover

(3)模态框——Modal——小重点

模态子窗体(Modal)：若弹出在子窗体不关闭，则父窗体无法获取焦点；且父子窗体间可以传递数据。浏览器默认提供了三个模态子窗体：

window.alert( )/confirm()/prompt();

Bootstrap使用div模拟实现了模态子窗体：

模态框必需4级6个div结构：

<div class="modal"> 半透明遮罩层

<div class="modal-dialog"> 宽高、定位层

<div class="modal-content"> 背景色、边框、倒角层

<div class="modal-header">头部</div>

<div class="modal-body">主体</div>

<div class="modal-footer">尾部</div>

</div>

</div>

</div>

午间练习：完成一个完整的注册模态框，给模态框加上动画



data-toggle="tab"

data-dismiss="alert"

data-toggle="dropdown"

data-toggle="collapse"

第四阶段bootStrap第五天

复习：

整理Boostrap知识点

思维导图绘制工具：MindManager、XMind...

下次讲解： 11:20

1. Bootstrap阶段项目——英特尔中国官网(intel.cn)

选题：http://v3.bootcss.com/ “基于 Bootstrap 构建的网站”

项目进度安排：

day01:

am: 实现响应式页面布局，不带任何内容和图片

pm: 填充布局中每个div的内容(文字、图片)

day02:

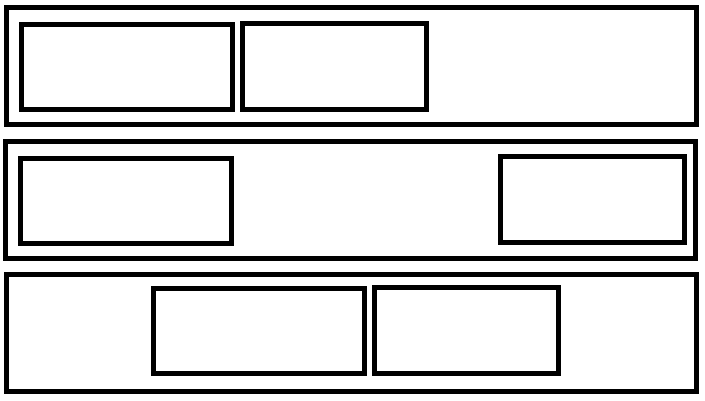
am: 学习LESS样式语言

pm: 修改Bootstrap源代码，“蓝色主题”的定制

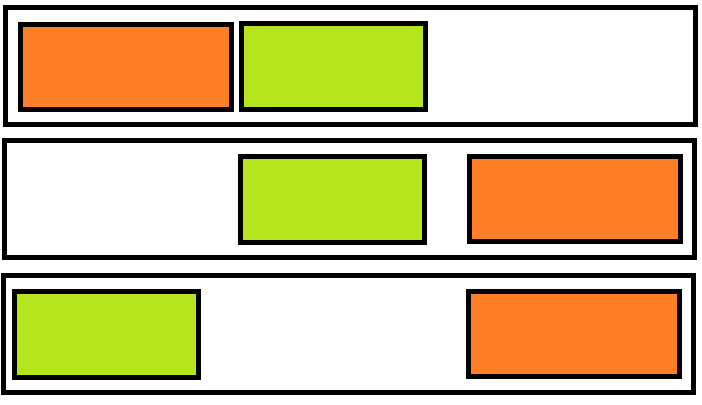
2.补充：Bootstrap全局CSS样式——栅格布局系统

列偏移和列排序：

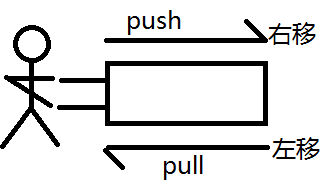
列偏移(col-offset)：偏移会影响当前列及后续的所有列。使用场合：让某一列及后续的整体向右错位



列排序：在源代码中的出现顺序不进行修改的情况下，某一列可以向右/左调整自身的位置，但不影响任何其它列。



列排序的方法：



.col-lg/md/sm/xs-push-1~12： 推向右

.col-lg/md/sm/xs-pull-1~12： 拉向左

提示： 列排序对当前宽度的屏幕及更大的屏幕都有效。

第四阶段bootStrap第六天

Bootstrap阶段项目进度安排：

day01:

am：页面布局

pm：布局框中的内容

day02:

am：LESS样式语言

pm：定制Bootstrap，实现蓝色主题

1. 动态样式语言

静态样式语言：即CSS，作为一门“语言”，CSS并不称职，缺少一门编程语言必备的基本要素：如变量、数据类型、运算、逻辑结构、函数、继承等；导致样式文件的修改和维护困难！

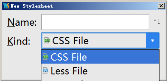
动态样式语言：在静态样式语言的基础上，添加了一门真正的语言所必需的元素：**变量、数据类型、运算、逻辑结构、函数、继承**等，从而大大提高样式的可修改和可维护性。

常见的动态样式语言：

(1)Sass/SCSS

(2)Stylus

(3)Less



注意：浏览器只能解析标准的CSS样式语言！所有动态样式语言的源文件必需转换为css文件——这一过程称为动态样式语言的“编译(Compile)”。

2.动态样式语言——Less概述

官网：http://www.lesscss.net/

Bootstrap提供的Less使用手册：http://www.bootcss.com/p/lesscss/

LESS 将 CSS 赋予了动态语言的特性，如 变量， 继承， 运算， 函数. LESS 既可以在 客户端 上运行 (支持IE 6+, Webkit, Firefox)，也可以借助Node.js或者Rhino在服务端运行。

Less的语法基于CSS，为CSS添加了很多动态语言的特性。

LESS源文件的两种使用方法：

**(1)在客户端使用.less源文件——仅供了解，不推荐使用**

1)编写.less源文件；

2)在.html链接该文件

<link rel="stylesheet/less" href="css/1.less"/>

3)在.html中链接一个less.js文件——自调函数，是一个Less的编译器。

**(2)在服务器端使用.less源文件——推荐使用**

1. 编写.less源文件
2. 程序员在自己的开发电脑上运行一个Less编译程序，把.less编译为.css文件
3. 编写.html文件，直接链接.css文件即可
4. 在服务器端编译Less源文件的步骤——易出错！

(1)下载并安装一个服务器端JS解释器：如NodeJS或Rhino

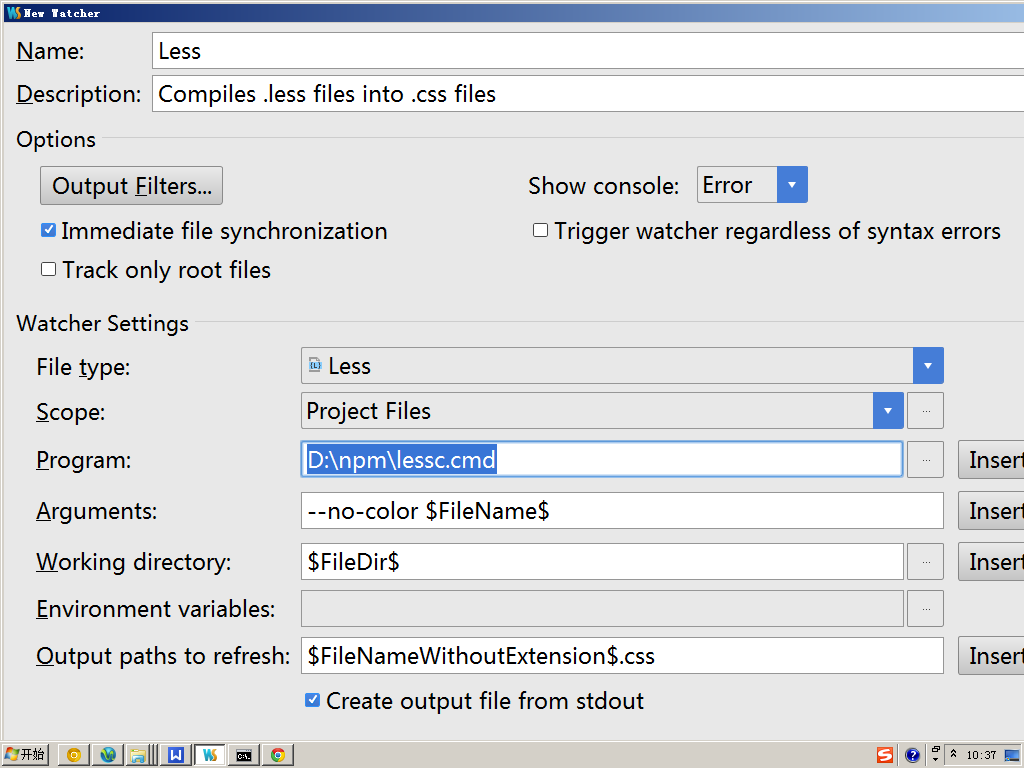
(2)下载一个Less源文件的编译器lessc.js(省略了.js后缀)

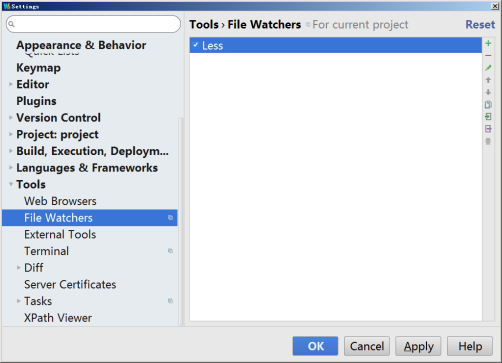
(3)使用JS解释器执行Less编译器，把.less文件转换为.css文件

1. **方式1：手工在cmd中敲入命令来执行**

node d:/npm/node\_modules/less/bin/lessc e:/1.less > e:/1.css

1. **方式2：把lessc配置为WebStorm里的一个文件监视器(File Watcher)，监视.less文件的创建和修改，立即自动调用lessc编译器，生成.css文件**





1. LESS语法学习

(1)Less支持所有的CSS语法。

(2)Less支持多行注释——会被转换到.css文件；还支持单行注释——不会被转换到.css文件，CSS不支持单行注释。

(3)Less提供了变量(Variable)的概念

语法： **@变量名: 值;**

提示：变量可以赋值为任何合法的样式值。

(4)Less提供了混入/混合(Mixin)的概念——超级变量

语法： .class1 { ... }

.class2 { .class1; }

(5)Less提供了带参混合

语法： .class1(@变量1, @变量2, ...) { ... }

.class2 { .class1(值1, 值2, ...); }

(6)Less支持样式嵌套

语法： .class1 {

样式声明1;

.class2 {

样式声明2;

}

}

会被编译为： .class1{ } .class1 .class2 { }

(7)Less支持算术运算

可以对各种变量和常量数值执行如下运算：+ - \* / %

(8)Less内置了几十个函数

round( )

ceil( )

floor( )

image-width( url )

image-height( url )

lighten( color, 百分比 )

darken( color, 百分比 )

(9)Less支持文件的导入

语法： @import 'xx.less'

提示：CSS中也支持文件的导入(在客户端导入)，但不推荐使用；Less中支持文件的导入(在服务器端导入)，严重推荐使用！——可以实现一个大项目中的多个样式文件分别编写和存储。 一般情况下，项目中可以按照代码的功能&人员的分工把样式文件分配多个.less文件中，最后由一个总的.less文件，导入全部其它文件，生成一个完整的.css文件。

5.通过修改Bootstrap LESS源代码实现Bootstrap定制

Bootstrap定制的三个目标：

**(1)瘦身：**精简bootstrap.less中无用的@import，删除自己项目中无用的样式；

**(2)定制：**修改variables.less其中定义的变量；

**(3)深度定制：**修改每个组件对应的.less文件，设置其中的样式。

Angularjs第一天

软件声明周期：

1)招投标

2)需求分析

3)概要设计

4)细节设计

5)编码实现

6)测试

7)部署维护

软件工程核心课程：

《软件过程》《软件测试》《软件质量保证》

《软件度量》《软件标准》 CMMI-5

1. 软件工程——软件设计原则

(1)DRY——不要Ctrl+C/V

(2)KISS——代码越简单越好

(3)YAGNI——不写不需要的代码

(4)OCP——你的代码可供别人进一步使用，但不能被别人修改

(5)High Cohesion，Low Coupling

(6)SRP——每个组件只负责一件事情

(7)迪米特法则/最少知识法则——不要让司机造车

(8)拥抱改变——提高代码的可修改性

1. 软件工程——软件设计模式

设计模式（Design pattern）是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性。 毫无疑问，设计模式于己于他人于系统都是多赢的；设计模式使代码编制真正工程化；设计模式是软件工程的基石脉络，如同大厦的结构一样。

马走日象走田 ——> 双炮枪/马后炮

|| ||

闭包/原型/对象 ——> 设计模式(23+1种)

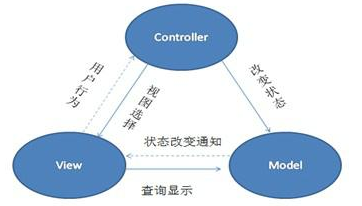
1. [工厂模式](http://baike.baidu.com/view/66964.htm#3_2)
2. ▪ [抽象工厂模式](http://baike.baidu.com/view/66964.htm#3_3)
3. ▪ [建造者模式](http://baike.baidu.com/view/66964.htm#3_4)
4. ▪ [原型模式](http://baike.baidu.com/view/66964.htm#3_5)
5. ▪ [单例模式](http://baike.baidu.com/view/66964.htm#3_6)
6. ▪ [适配器模式](http://baike.baidu.com/view/66964.htm#3_7)
7. ▪ [桥接模式](http://baike.baidu.com/view/66964.htm#3_8)
8. ▪ [组合模式](http://baike.baidu.com/view/66964.htm#3_9)
9. ▪ [装饰模式](http://baike.baidu.com/view/66964.htm#3_10)
10. ▪ [门面模式](http://baike.baidu.com/view/66964.htm#3_11)
11. ▪ [享元模式](http://baike.baidu.com/view/66964.htm#3_12)
12. ▪ [代理模式](http://baike.baidu.com/view/66964.htm#3_13)
13. ▪ [职责链模式](http://baike.baidu.com/view/66964.htm#3_14)
14. ▪ [命令模式](http://baike.baidu.com/view/66964.htm#3_15)
15. ▪ [解析器模式](http://baike.baidu.com/view/66964.htm#3_16)
16. ▪ [迭代器模式](http://baike.baidu.com/view/66964.htm#3_17)
17. ▪ [中介模式](http://baike.baidu.com/view/66964.htm#3_18)
18. ▪ [备忘录模式](http://baike.baidu.com/view/66964.htm#3_19)
19. ▪ [观察者模式](http://baike.baidu.com/view/66964.htm#3_20)
20. ▪ [状态模式](http://baike.baidu.com/view/66964.htm#3_21)
21. ▪ [策略模式](http://baike.baidu.com/view/66964.htm#3_22)
22. ▪ [模版模式](http://baike.baidu.com/view/66964.htm#3_23)
23. ▪ [访问者模式](http://baike.baidu.com/view/66964.htm#3_24)

|  |
| --- |
| Modal：模态框  Model：模型  Module：模块 |

MVC设计模式：

AngularJS中的MVC把前端项目代码分为三个层次：

1. **Model：模型**，即项目中的业务数据，在ng中由变量担当；
2. **View：视图**，负责业务数据的呈现，在ng中HTML+指令;
3. **Controller**：控制器，负责操作（增删改查）业务数据，在ng中由一种特殊的function来担当。



MVC设计模式的好处：把项目代码根据功能特性分为三个层次，结构更加清晰，更便于分工协作。

MVC设计模式的不足：会增加代码的复杂性。

1. AngularJS概述

AngularJS诞生于2009年，由Misko Hevery 等人创建，后为Google所收购。是一款优秀的前端JS框架，已经被用于Google的多款产品当中。AngularJS有着诸多特性，最为核心的是：MVVM、模块化、自动化双向数据绑定、语义化标签、依赖注入等等。

AngularJS是一个JS框架，适用于以数据操作为主的SPA(Single Page Application)项目。

面试题：AngularJS的四大特性

(1)采用MVC模式 —— day01

(2)双向数据绑定 —— day02

(3)依赖注入 —— day03

(4)模块化设计 —— day03

internationalization i18n 国际化：一个网站见中国人呈现中文、见美国人就呈现英文......

3.AngularJS的使用

ng的编码特点：

1. 忌讳声明全局变量和函数——避免污染全局对象！整个框架只有一个污染点： window.angular
2. AngularJS框架自己的对象都以$开头，如$http、$interval、$window、$timeout、$location；ng内部专用的对象(禁止程序员调用的对象)都以$$开头。

4.AngularJS基础概念——指令(directive)

Directive：HTML标签本身功能有限，不能声明变量、循环、选择等。AngularJS为HTML标签增加了变量、循环、选择等缺少的特征，称为“AngularJS指令”，具体可能有三种用法：

(1)直接作为标签使用： <ng-view />

(2)作为属性使用： <li ng-repeat=""></li>

(3)作为class内容使用： <div class="ng-view">

提示：AngularJS中，“HTML标签+增强型指令” 担当View的角色。

常用的Angular指令：

(1)ngApp：用于启动一个Angular应用，划定当前应用的有效范围

用法： <ANY ng-app=""></ANY>

(2)ngInit：用于在HTML中声明Angular变量

用法： <ANY ng-init="变量名=变量值; 变量名=变量值; ..."

<ANY class="ng-init: 变量名=变量值;">

(3)ngController：用于创建一个控制器对象

用法： <ANY ng-controller="控制器名">

(4)ngSrc：用于为图片等元素创建src属性，放置浏览器提前请求，产生404错误

用法： <ANY ng-src="NG表达式">

(5)ngRepeat：用于循环输出当前的HTML元素

用法： <ANY ng-repeat="变量名 in 数组">

5.AngularJS基础概念——表达式

**语法： {{ NG表达式 }}**

作用： 用于在HTML标签中穿插输出Model数据——可以是常量、变量、常见运算。

提示： {{}}必须放在ngApp指令范围内才会被NG处理！

午间练习：测试NG表达式可以执行哪些运算：

1. 算术运算： + - \* / %可以； ++ --不可以
2. 比较运算： 可以
3. 逻辑运算： 可以
4. 赋值运算： =可以； += -= \*= /=不可以
5. 三目运算： 可以
6. 特殊运算： 不可以
7. 调用对象的属性和方法：可以
8. AngularJS四大特性之一——MVC模式——重点

Model：模型，一种特殊的变量

View：视图，HTML+指令

Controller：控制器，一种特殊的function

**Module<=控制器<=Model<=呈现数据(View)**

使用AngularJS中的MVC模型的步骤：

(1)声明自定义的模块

angular.module('模块名', [依赖列表])

(2)把自定义的模块注册给当前应用

<html ng-app="模型名">

(3)在模块中声明控制器函数

module.controller('控制器名', function(){ ... });

(4)调用控制器函数，创建控制器对象

<ANY ng-controller="控制器名">

(5)在控制器中增删改查Model数据

$scope.xx = yy;

(6)在View中呈现Model数据

{{ xx }}

AngularJS中，Model数据可以是下述类型：

·数字

·字符串

·布尔

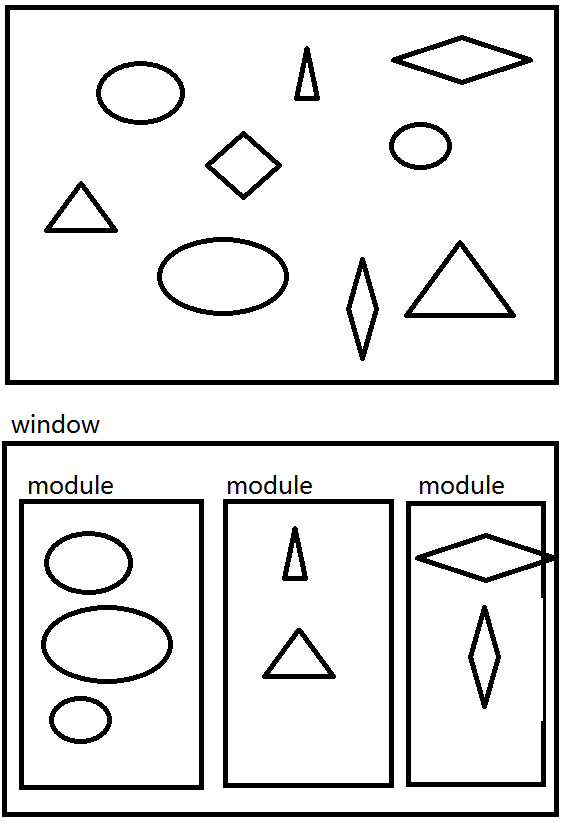
·日期

·对象

·数组

·对象的数组

1. AnguarJS四大特性之四——模块化设计



模块化设计体现的设计思想：高聚合低耦合

作业：把bookList呈现到一个TABLE中

Angularjs第二天

复习：

Google AngularJS是一个JS框架，适用于以数据操作为主的SPA应用。四大特征：

(1)采用MVC模式

Model：模型，即业务数据，ng中保存在特定作用范围内的变量 $scope.xx=yy;

View：视图，负责数据的呈现， ng中HTML+指令

Controller：控制器，负责操作数据， ng中function

(2)双向数据绑定

(3)依赖注入

(4)模块化设计，即一个箱子/容器，可以用于盛放全局函数、控制器等。

\*(5)指令系统

1. 面试题：DOM、jQuery、AngularJS三者的关系？

**DOM操作思路：先查找DOM元素，再操作元素**

DOM: document.getElementByXxx().innerHTML = "";

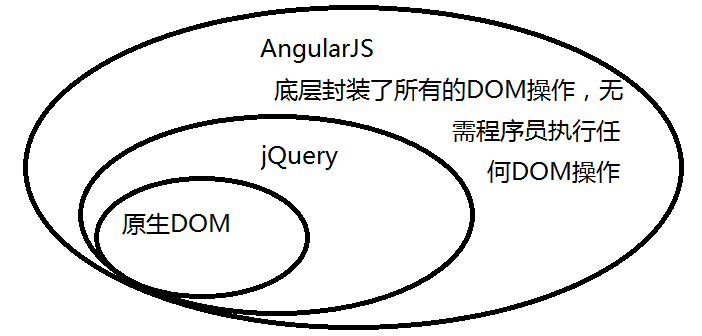
jQuery: $(..).html("")

**AngularJS操作思路：先创建数据，再操作数据**

$scope.bookList = [ {}, {}, {} ];

{{bookList[0].bname}}

**NG彻底改变了前端人员的思维方式。**



angular.element('#footer')

angular.element('.list1 li > a')

angular.element() === jQuery()

AnguarJS是对DOM操作的进一步封装，其底层会执行所有的DOM操作，NG在启动时会检查若存在jQuery对象，则使用它；否则使用一个自定义的jQuery简化版本——jQLite

使用MVC模式的步骤：

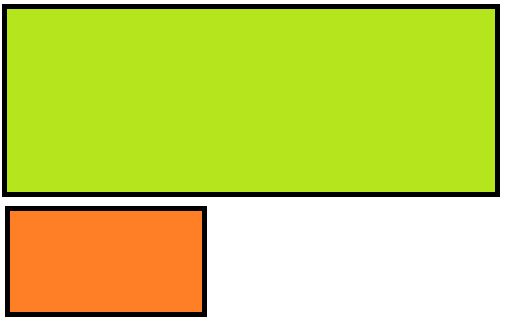
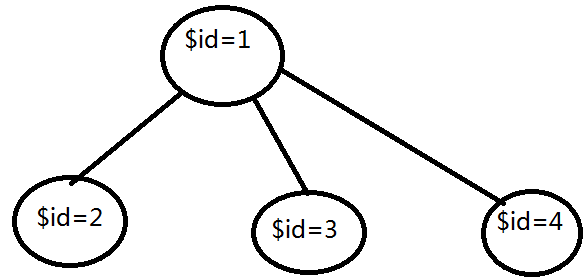
1. 创建自定义模块
2. 注册自定义模块
3. 创建控制器
4. 创建控制器对象
5. 声明Model
6. View中呈现Model

2.控制器和$scope、$rootScope

(1)每个控制器对象都有一个作用范围，每个作用范围都对应一个$scope对象

<div ng-controller=""> 控制器的作用范围 </div>

(2)一般情况下，每个控制器作用范围内只能使用当前$scope中保存的Model数据

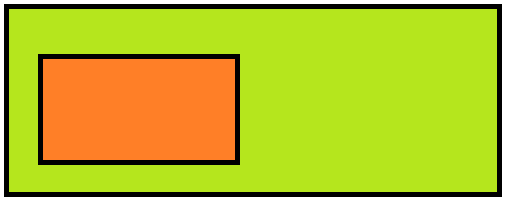
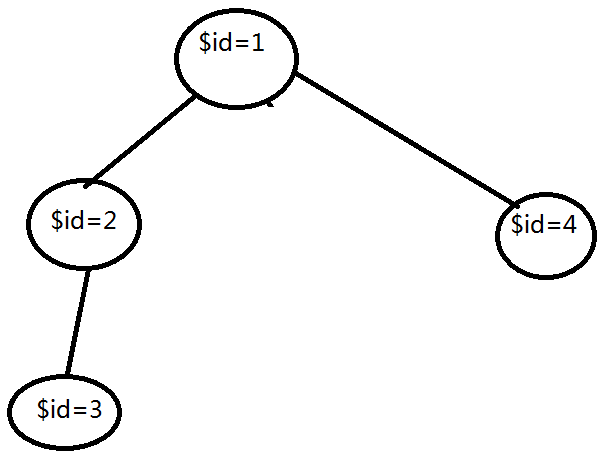
 

(3)若需要让控制器2可以使用控制器1中的Model数据，只需要把控制器2声明在控制器1内部即可：

<div ng-controller="c1">

<div ng-controller="c2"></div>

</div>

(4)若某个控制器想创建一个Model数据，让其它所有的控制器都可以共享，只需要保存在$id=1的那个根作用域中即可——$rootScope

作业回顾：

1. 补充：AngularJS基础概念——指令

(6)ngIf：为HTML添加条件判断功能

语法： <ANY ng-if="布尔表达式"></ANY>

用涂： 当布尔表达式的值为true，当前的元素则显示；否则当前元素从DOM树删除

(7)ngShow：为HTML添加条件判断功能

语法： <ANY ng-show="布尔表达式"></ANY>

用涂： 当布尔表达式的值为true，当前的元素则显示；否则就隐藏

(8)ngHide：为HTML添加条件判断功能

语法： <ANY ng-hide="布尔表达式"></ANY>

用涂： 当布尔表达式的值为true，当前的元素则隐藏；否则就显示

(9)ngClick/ngKeypress/ngBlur/ngMouseover：为元素绑定事件监听函数——也是一个Model数据

(10)ngBind：将一个表达式的值绑定为当前元素的innerHTML

(11)ngClass：为当前元素指定class属性值(由一个Model变量来确定)

语法： <ANY ng-class="Model变量名"></ANY>

1. AngularJS四大特性之二——双向数据绑定——最重点

**数据输出**：一个变量一旦被输出到DOM树，此后不论此数据如何改变，之前的输出都不会再变！

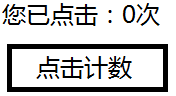
**数据绑定(bind)**：一个变量一旦被绑定到DOM树，此后不论此数据何时改变，之前的呈现都会随之改变！

**方向1：把Model绑定到View**——把模型数据绑定到View中，此后只要Model一发生改变，则View中的呈现会随之改变。

实现方法： {{ }}、ngBind、ngRepeat、ngIf、ngSrc......

练习：

1. 实现一个点击计数器



1. 实现一个闪动的小星星，提示：此题还可以考虑ngStyle、ngClass两个指令来实现
2. 使用ul>li显示一个成绩数组，点击按钮可以在该数组中添加一个新的成绩(随机数)
3. 仿写一个轮播广告，每1s自动换下一张图片

**方向2：把View(表单控件的值)绑定到Model**——只要浏览者改变了表单控件的值，Model数据会自动的随之改变。

实现方法： 使用ngModel指令

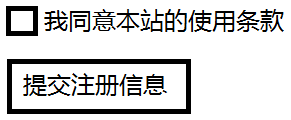
1. 单行输入域：把value属性的值绑定到ngModel指定的变量
2. 多行输入域：把value属性的值绑定到ngModel指定的变量
3. 下拉列表：把选中的option的value绑定到ngModel指定的变量
4. 单选框：把选中的radio的value绑定到ngModel指定的变量
5. 复选框：每个复选框必须绑定到唯一的一个模型变量上，不能多个复选框绑定到同一个Model变量！当复选框被选中，则模型变量的值变为true；否则变为false

|  |
| --- |
| 提示： ngModel是一个“方向2+方向1”的绑定  <input ng-model="userName">  若$scope.userName不存在，则ngModel会创建该对象；  若$scope.userName值存在，则ngModel会首先使用该对象； |

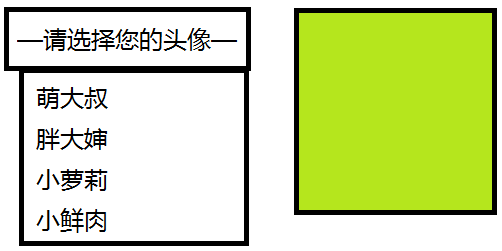
练习：

(1)用户在输入框中输入自己的名字，下方有一个div，其中显示出：你好：Xxx 提示：方向2 + 方向1

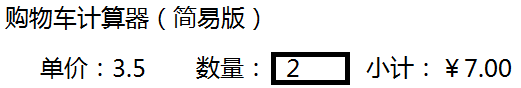
(2)用户同意注册条款，则可点击“提交按钮”；否则不可见或不可点击

 提示：可以使用ngShow或ngDisabled指令

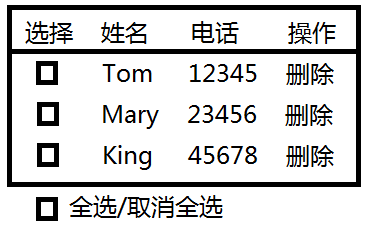
1. 让用户通过单选按钮选择性别，若选择“男”则整个页面变为蓝色背景；若选择“女”则整个页面变为粉色背景。
2. 通过下拉框选择头像名称，后方的图片立即呈现出对应的图片

 可以使用ngClass或ngStyle指定样式

1. 实现一个购物车计算器（简易版）

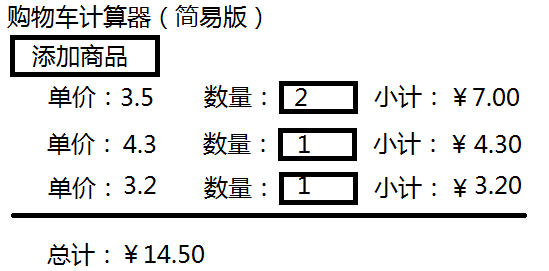


(6)全选和取消全选，以及删除 提示：ngChecked可用于为复选框指定是否选定

 练习：16:00~17:10

作业题：

实现一个购物车计算器（豪华版）



Angularjs第三天

复习：

Google AngularJS是一个JS框架，适用于数据操作为主的SPA应用。

NG是对jQuery的进一步封装，操作思路不再是“查找元素再操作元素”，解放为“创建数据再呈现数据”。

四大特性：

1. MVC设计模式

Model：数据

View：呈现数据

Controller：操作数据

1. 双向数据绑定

方向1：Model=>View

方向2：View=>Model

1. 依赖注入
2. 模块化设计

1.作业：

1. AngularJS基础概念——过滤器

Filter：过滤器，对数据进行筛选、过滤、格式化；本质是一个函数：

function(inputData){

//处理inputData

return ouputData;

}

NG中提供了几个好用的过滤器，用法：

{{表达式 | filter1 | filter2 | ...}}

“|”称为数据传递的管道

(1)uppercase

(2)lowercase

(3)date：把日期/整数转换为特定格式的字符串

(4)number：把数字转换为三位一逗号的字符串

(5)currency：把数字转换为货币显示格式的字符串

(6)orderBy：指定排序方法<ANY ng-repeat="x in 集合 | '排序属性' : 是否降序 "></ANY>

(7)limitTo：限制输出的数量

3.CSS和JS文件的压缩问题

YUI 库，全称Yahoo! UI Library。是一组工具和控件，用JavaScript写成, 为的是用DOM 脚本，DHTML和AJAX等技术创建丰富的网页交互式应用程序。

YUI 基于BSD协议，对所有的使用方式都是免费的。YUI 项目包括YUI 库和两个创建时工具： YUI Compressor (压缩) 和 YUI Doc （JavaScripts代码的文档引擎）。还有一些前端开发时进行页面性能优化指导方案——YUI 34条军规！

YUI-Compressor：是一个使用Java编写的CSS和JS文档的压缩和混淆工具。使用方法：

(1)安装Java运行环境软件——JDK

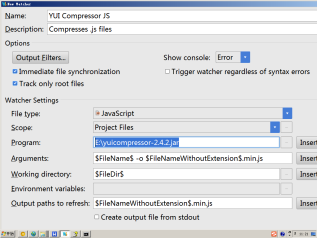
(2)下载yui-compressor.jar文件

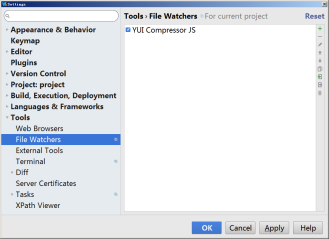
(3)运行yui-compressor压缩工具，对JS或CSS进行压缩

1)直接在命令行中运行——了解

java -jar e:/yuicompressor-2.4.2.jar e:/li.js > e:/li.min.js

2)配置为WebStorm中FileWatcher，只要用户编写了.js或.css即可以自动实现文件压缩——推荐





YUICompressor可以实现的压缩效果：

(1)删除所有的注释——代码压缩

(2)删除所有无用的空白字符——代码压缩

(3)尽可能简化所有的形参、变量名、函数名，但代码逻辑不变——代码混淆

(4)所有的关键字、数据(数字、字符串等)不做任何改变！

4.AngularJS四大特性之三——依赖注入

function Driver( car ){

car.start();

car.run();

}

**依赖(Dependency)**：如果要创建一个司机对象(new Driver())，就必须首先提供一个Car对象！没有Car对象就不能创建出Driver对象！——称Driver依赖于Car，或者Car对象是Driver的依赖对象！

被依赖的对象的创建方法：

1. **主动创建——适用于简单的被依赖对象的创建**

var c1 = new Car(); var d1 = new Driver(c1);

1. **被动注入(inject)——适用于复杂的被依赖对象的创建**

module.controller('myC1', function($scope, $interval){})

<div ng-controller="myC1"></div>

提示：AngularJS应用启动时，在底层会创建一个对象 $injector —— 专门负责ng调用了一个用户编写的function，此函数依赖的对象的创建。

**依赖注入(Dependency Injection)：**当一个对象必需的参数(称为依赖的对象)，由一个框架来自动提供，无需程序员手工创建，这一过程称为“依赖注入”。

**AngularJS在调用用户提供的function时，是根据其形参的名称来实现的以来注入——即在NG中function的形参名不能随意指定**！

注意：使用AngularJS的项目中，若编写的JS文件需要使用压缩程序进行压缩，function的形参名可能会被混淆为a、b、c这样名称，导致依赖注入失败！解决办法：

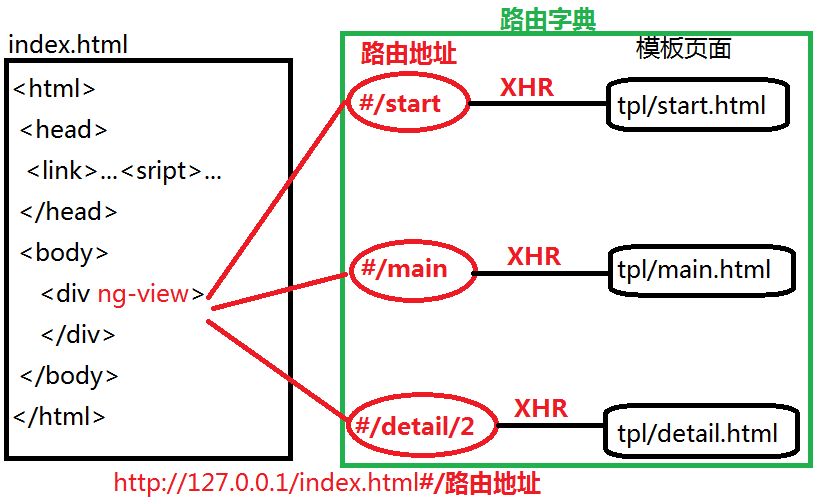
module.controller('控制器名', ['$scope', '$interval', '$http',function(a, b, c ){ }]);

1. 多页应用和单页应用的比较

|  |  |
| --- | --- |
| **多页应用** | **单页应用(SinglePageApplication)** |
| 一个项目中有多个完整的HTML文件 | 一个项目中只有一个完整的HTML页面（index.html），其它的HTML都只是标签片段而已——称为“模板(template)页面” |
| 可以使用超链接、JS实现页面间跳转 | 可以使用改进后的超链接、JS实现模板页面间的切换 |
| 传统的页面跳转是同步请求：在服务器生成响应内容时客户端是一片空白 | 模板页面间的切换属于典型的异步请求：直到下一个模板页面到来，前一个模板页面才从当前DOM树上删除。 |
| 页面跳转时，前一个DOM树无用了，需要从浏览器中全部删除；然后等待着下一个DOM树的到来.... | index.html到来时，浏览器创建一个DOM树。所有的模板页面请求都是XHR请求，到来时只需要作为一个片段挂在当前DOM树——浏览器从始至终只有一个DOM树。 |
| 当网速比较慢时(尤其是移动应用中)，用户的浏览体验不好！ | 不会出现一片惨白的情形，浏览体验好！ |
| 页面切换时，DOM被完整的删除，不可能有过场动画！ | 页面切换的本质是DIV的轮换，可以很容易实现漂亮的过场动画！ |

|  |
| --- |
| AngularJS官方提供的模块(箱子)：   1. ng：整个框架的核心功能：MVC模型、双向数据绑定、依赖注入 2. ngRoute：负责SPA应用 3. ngAnimate：负责动画效果 4. ngTouch：负责触屏相关的操作 |

6.ngRoute模块实现SPA应用原理——面试题



使用步骤：

(1)创建唯一完整的页面：index.html，引入需要的css和js文件，自定义的模块中必须声明使用到ngRoute模块

(2)在唯一完整的页面中声明一个容器元素(如ngView)，用于盛放模板页面

(3)创建所有必需的模板页面——只需要div片段即可——不能直接被浏览器在地址栏中请求！！！！！

(4)配置ngRoute中的路由字典——为每个模板页面取一个路由地址名称

(5)在浏览器地址栏中输入index.html再追加上某一个路由地址，回车.......

(6)ngRoute会自动解析location中的hash值，查找路由字典，获取到该路由地址对应的模板页面的真实URL，发起XHR异步请求，接收服务器返回的HTML片段，挂载到ngView元素中

ngRoute使用过程中的几个小问题：

1. 如何设置默认路由地址

$routeProvider.otherwise({redirectTo: '/start'})

1. 如何为每个模板页绑定不同的Model数据

$routeProvider.when('路由地址', {

templateUrl: '模板页面地址',

controller: '控制器名'

//该控制器对象的$scope限于整个模板页面

})

1. 模板页面中的所有外部资源（如图片）地址不能以模板页面为起点！都必须以index.html为起点——这才是唯一的DOM树
2. 模板页面之间执行跳转，有三种方式：

1)直接修改地址栏中的路由地址： ...index.html#/路由地址

2)使用超链接： <a href="#/路由地址"></a>

3)使用JS脚本： $location.path('/路由地址')；

(5)若多个模板页面中需要使用完全一样的Model变量/函数，则可以提取出来，在一个公共的父控制器中声明即可

<body ng-controller="parentCtrl">

面试题：

(1)面向过程和面向对象的区别？

(2)如何进行前端性能优化——34条！

作业题：

(1)仿照ngRoute模块的运行效果，写一个SPA应用框架。

Angularjs第四天

复习：

1.高阶面试题：AngularJS双向数据绑定的实现原理？ AngularJS最大的缺陷是什么？

**View:**

<div ng-controller="控制器名">

<p>{{emp.age}}</p>

<div ng-bind="emp.age"></div>

</div>

**Controller&Model:**

module.controller('控制器名', function($scope){

$scope.emp = {};

$scope.emp.age = 10;

})

Java语言中有监视对象数据改变的机制：

Object.observe( obj , function(){} )

监视任何一个对象值的改变，一变立即调用指定的回调函数。

JS中只有HTML对象才有这样的机制——事件处理：

div.onclick = function(){ }

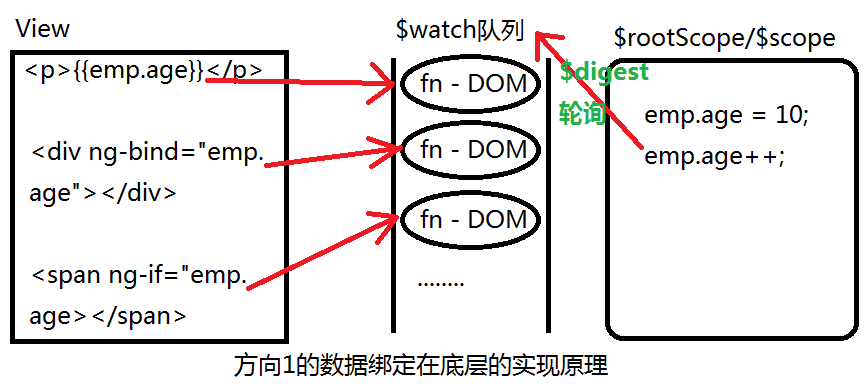
但JS没有监视某一个数据对象值改变的机制：

emp = { age: 10; }

emp.age++;

AngularJS中提供这样一种监视机制：

$scope.$watch('数据对象', function(to, from){ })



AngularJS最致命的缺陷是：每一次数据绑定都要生成一个监视函数放在$watch队列。若绑定次数是海量的，则监视函数也是海量的。只要有一个小的模型数据发生改变，AngularJS会调用$digest()轮询整个$watch对列，执行效率可能很低！

解决办法：

(1)把完全不相干的数据放在不同的ngApp中——使用angular.bootstrap函数

(2)对于不需要反复改变的数据，尽量避免创建其对应的监听函数—— {{::age}} ng-bind="::age"

面试题：window.setInterval（BOM相关对象）和$interval(Angular对象)有何区别？

$interval(function(){

////执行语句

//最后会自动添加一句：$scope.$digest();

})

面试题：AngularJS中发起AJAX请求通常使用：

$http.get('URL').success( function(data){} )

$http.post('URL', data).success( function(data){} )

上述两个方法可以用$.get()和$.post()代替吗？ —— 答案同上

2.复习：使用ngRoute模块实现SPA应用

3.使用ngAnimate模块添加过场动画

The ngAnimate module provides support for JavaScript, CSS3 transition and CSS3 keyframe animation hooks within existing core and custom directives.

ngAnimate模块没有提供任何动画效果，但是为常用的指令提供了很多的“动画钩子(Hooks)”，可以供JS动画、Transition动画、Keyframe动画使用。

The directives that support animation automatically are: ngRepeat, ngInclude, ngIf, ngSwitch, ngShow, ngHide, ngView and ngClass.

演示：使用ngAnimate模块为ngView指令提供的Transition动画钩子。

使用步骤：

(1)引入angular.js和angular-animate.js

(2)自定义模块声明使用到了ngAnimate模块

(3)观察ngAnimate模块为ngView指令提供的动画钩子

.page {

transition: all 5s linear;

position: absolute;

width: 100%;

}

/\*即将要离开的ngView动画开始时的样式\*/

.page.ng-leave {

left: 0;

opacity: 1;

}

/\*即将要离开的ngView动画结束时的样式\*/

.page.ng-leave.ng-leave-active {

left: -100%;

opacity: 0;

}

/\*即将要进入的ngView动画开始时的样式\*/

.page.ng-enter {

left: 100%;

opacity: 0;

}

/\*即将要进入的ngView动画结束时的样式\*/

.page.ng-enter.ng-enter-active {

left: 0;

opacity: 1;

}

1. AngularJS阶段项目——仿“饿了么”做订餐应用

所用技术：jQuery + Bootstrap + AngularJS + AJAX

项目进度安排：

day01:

am：实现SPA应用中的页面切换及过场动画

pm：实现每个模板页面中的静态内容——Boostrap

day02:

am：编写服务器端必需的PHP页面

pm：使用$http远程请求服务器端数据，使用数据绑定将数据呈现在View中

提示：推荐使用大型AngularJS项目所使用的目录结构——每个模板页面都独立为一个Model。

|  |
| --- |
| Bootstrap和AngularJS整合出现的冲突点：  Bootstrap导航条中的下拉菜单按钮 不能使用 <a href="#">形式  必须使用<button data-target="#">代替！  <!--  <a href="#my-collapse" class="navbar-toggle" data-toggle="collapse">  <span class="icon-bar"></span>  <span class="icon-bar"></span>  <span class="icon-bar"></span>  </a> -->  <button data-toggle="collapse" data-target="#my-collapse" class="navbar-toggle">  <span class="icon-bar"></span>  <span class="icon-bar"></span>  <span class="icon-bar"></span>  </button> |

1. 处理“多页面中相同的内容”——页面包含

把多个页面中完全一样的内容（如导航条、版权声明条等）担当提取出来作为一个独立的页面，其他需要这些内容的页面将其包含进来。具体方案：

(1)方案1： 使用SSI技术

Server Side Include，修改Web服务器中的配置，当接受到特定的HTML请求时，返回多个html文件的组合体——要求开发者必须能够修改Web服务器的配置文件。

(2)方案2：使用服务器端动态编程语言

PHP - include('header.php');

//当前页主体

include('footer.php');

当前页面以及被当前页面包含的页面的内容会形成的一个响应消息发送给客户端。

(3)方案3：使用纯客户端技术(AJAX) —— ng模块中ngInclude指令

<div id="header"></div>

//当前页面的主体

<div id="footer"></div>

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

<script>

$(function(){

$('#header').load('header.html');

$('#footer').load('footer.html');

})

</script>

好处：不依赖于服务器和服务器端特定技术

不足：需要多发起两次AJAX请求

AngularJS中提供了方案3的具体实现：

<div ng-include="被保护文件的路径"></div>

Angularjs第五天

1. AngularJS阶段项目——仿“饿了么”做订餐应用

所用技术：jQuery + Bootstrap + AngularJS + AJAX

项目进度安排：

day01:

am：实现SPA应用中的页面切换及过场动画

pm：实现每个模板页面中的静态内容——Boostrap

day02:

am：编写服务器端必需的PHP页面

pm：使用$http远程请求服务器端数据，使用数据绑定将数据呈现在View中

1. 导入项目所需要的数据结构

数据库名： kaifanla

表1： kf\_dish —— 8行记录

did

name

price

img\_sm

img\_lg

detail

material

表2： kf\_order

oid

phone

user\_name

sex

order\_time

addr

did

把SQL文件导入到本地数据库服务器，有两种方式：

**(1)使用命令行方式——mysql.exe**

c:/xampp/mysql/bin/mysql.exe -h127.0.0.1 -uroot -p [-Dkaifanla] -P3306

上述语句可以简化为：

mysql.exe -uroot

提交SQL文件中所有的语句给服务器执行： source e:/kaifanla.sql

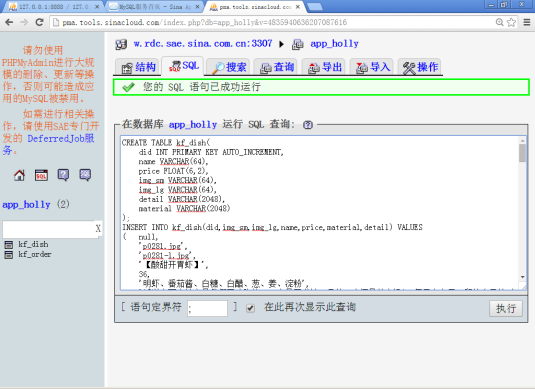
**(2)使用网页方式——phpMyAdmin**

http://127.0.0.1:8888/phpmyadmin

把SQL文件导入到新浪云数据库服务器，有一种方式：

(1)使用网页方式





注意：SAE中的MySQL不允许用户创建自定义数据库名，且每个应用只能有一个默认的数据库。

1. 根据项目功能需求点编写PHP页面

**main.html中需要如下的数据提供页面：**

(1)data/dish\_getbypage.php 分页向客户端返回菜品列表

(2)data/dish\_getbykw.php 根据搜索关键字返回菜品列表

**detail.html中需要如下的数据提供页面：**

(3)data/dish\_getbyid.php 根据ID返回一道菜全部详情

**order.html中需要如下的数据提供页面：**

1. data/order\_add.php 接收订单信息，保存订单

**myorder.html中需要如下的数据提供页面：**

(5)data/order\_getbyphone.php 根据电话号码返回订单列表

WebStorm PHPStorm

1. 补充：如何实现分页查询

SELECT 列名 FROM 表名 LIMIT 起始行, 行数 ;

提示：起始行从0开始； 行数指定此次返回的最多可能数。

1. 补充：SQL中的模糊查询

精确查询：

SELECT 列名 FROM 表名 WHERE did=3;

SELECT 列名 FROM 表名 WHERE name='鸡蛋';

模糊查询：

SELECT 列名 FROM 表名 WHERE 列名 **LIKE** '%关键字%'

1. 补充：跨表查询/多表查询

SELECT 表1.列1，表1.列2, 表2.列3, 表2.列4

FROM 表1, 表2

WHERE 表1.列=表2.列;

提示：多表查询一定要防止“笛卡尔积”

Bootstrap： CSS框架，UI组件库

AngularJS： JS框架， MVC、双向数据绑定

jQueryMobile：移动开发UI组件库

ExtJS：重量级JS框架

微信开发：

1. ngRoute模块中如何在两个模板页面间传递请求参数

main.html：

<a href="#/detail**/101**">第一道菜</a>

<a href="#/detail**/202**">第二道菜</a>

<a href="#/detail**/303**">第三道菜</a>

detail.html： 下次讲解： 17:10

如何获取地址栏的URL中提供的可变数据呢？

**路由参数(Route Parameter)**：在ngRoute模块中，请求一个模板页面必须指明一个路由地址，一个路由地址一般以#开头，后面分为两部分：一部分固定不变，一部分是可变的——可变的这部分相当于一个变量，称为路由参数。

如果一个模板页面的路由地址中有可变部分(即有路由参数)，必须修改其对应的路由地址的声明：

$routeProvider.when('/detail**/:did**', {

templateUrl: 'tpl/detail.html',

controller: 'detailCtrl'

})

带有路由参数的模板页面，若想读取地址栏中的路由参数，只需要在其对应的控制器中使用ngRoute模块中服务：$routeParams即可

.controller('detailCtrl', function($scope, $routeParams){

console.log('detailCtrl读取到的路由参数对象：');

console.log($routeParams);

})

|  |
| --- |
| 小知识点： jQuery.param(obj) 可用于把一个对象转换为： k1=v1&k2=v2&k3=v3...形式的字符串，用于GET/POST的请求消息主体。 |