|  |
| --- |
| 西华大学 |
| ***Front End*** |
| Web前端笔记 |
|  |
| **周和涛** |
| **2017/2/14** |

|  |
| --- |
| TEL:18482163550 |

**WEB前端基础**

1、基础

HTML5 : H5 --> HTML

CSS3 : 样式表

UI : 用户交互

Project : 项目

2、核心

JS : JavaScript

DOM : Document Object Model

3、高级

HTML5高级

jQuery :

AJAX : 阿贾克斯,页面无刷新(异步提交请求)

4、框架

Angular JS

BootStrap

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

1、WEB工作原理

请求

响应

HTTP协议: http://www.baidu.com http://www.google.com

ftp://10.0.0.1

Hyper Text Transfer Protocal（http）：超级文本传输协议

作用：规范了数据是如何传递的

规范了传递的数据的格式是什么

**HTML**

1、标记语法

1、使用尖括号扩起来(<>)

2、有封闭标签也有非封闭标签

封闭:成对出现，有开始，有结束(双标签)

<a> : 开始，<标签名字>

</a> :结束，</标签名字>

非封闭:只有开始没有结束,(单标签)

<br> : 即表示开始，又可以表示结束

<br/>: 单标签，开始结束即为一体

<p></p> : 双标签中，可以将一些文本写在标签内，这些文本受当前标签的样式控制

注意:封闭标签，必须有成对出现，如果忘记结束，将会产生意想不到的结果

<h1>Hello Some Text<h1>

这是一段HTML文档

<h1>Hello Some Text

这是一段HTML文档

2、元素嵌套：一个元素包含在另外一个元素内

注意:1、要注意嵌套的标签顺序

<a><p></p><a>

<p><a></a></p>

<a><p></a></p>（错误）

2、代码要以缩进的方式体现出来

<a>

<p>

<b>

</b>

<p>

</a>

3、属性：用来修饰元素

<标记名称 属性名="属性值"></标记名称>

一个标签内属性可以有多个，排名不分先后,多个属性间通过空格来区分

<p align="center">这是个p</p>

align功能:控制文本在标签内的水平对齐方式(left,center,right)

标准属性(公共属性、通用属性)：

id : 定义所在标签的唯一标识名称

title : 鼠标移入到当前标签所在区域时所显示的文字

class : 引用样式表中的指定类样式

style : 定义当前标签的行内样式

4、注释：<!-- 这是一段注释 -->

1、声明标签

2、声明标签内的属性

3、 。。。。 。。。。

-->

5、文档类型声明

HTML5的文档类型声明

<!DOCTYPE html>

Web 与 Internet

1、<head>元素

1、作用:网页头部，主要包含网页的说明信息

1、网页标题

2、网页的编码格式

3、声明内部样式表

4、引入外部样式表

5、声明内部JS(JavaScript)脚本

6、引入外部JS

7、声明其他元素,如关键字、描述等... ...

1、网页标题

<title></title>

--标题内容会出现在浏览器最上方(分浏览器)

--没有任何属性

--只能出现在head中

--一个网页最多只能有一个标题

2、网页编码格式

默认编码:ISO-8859-1

1、通过meta标签

<meta http-equiv="content-type" content="text/html;charset=utf-8" />

2、通过meta标签

<meta charset="utf-8" />

字符集：文本编码

3、添加网页的描述和关键字

meta : 一般做说明信息的声明

常用属性:

http-equiv,

content,

name

charset

1、添加关键字

<meta name="keywords" content="HTML,CSS,JS,WEB前端" />

2、添加描述

<meta name="description" content="描述内容" />

练习:

创建一个网页，添加标题为 "我的第一个HTML文档",设置编码为utf-8,添加网页关键字HTML,CSS,WEB前端,添加网页描述"这是我的第一个HTML文档，包含了我所学到的一部分内容"

1、创建文档并且添加html结构

doctype,html,head,body

2、添加title标签

3、添加 meta ，说明编码

4、添加两个 meta ，指定关键字，和描述

keyworkds,description

2、特殊字符

通过转义字符来表示特殊符号,如空格，<,>

转义字符:改变了单次的原有的意思，被解释成一种全新的含义

&.....; &:与

空格:&nbsp;

< : &lt; ---> less than

> : &gt; ---> greater than

企业标识:&copy; ---> company

钱币:&yen;

3、标题

<hn> : n :1-6 一级标题 - 六级标题

1级最大

6级最小

使用场合：多数情况下会用在网页中文章的标题(h2)

4、段落元素

<p></p>

注意：成对出现

p提供了结构化文本的方式,具备一定的格式

格式:

---与其他文本用换行区分

---添加垂直空白距离，作为段落的间距

常用属性: align : 文本在p标签内的水平对齐方式

5、换行标记：<br>

在任何地方创建手工换行

6、分区元素

多用于布局

块分区元素:

该元素会自动占一行

<div>

行内分区:

众多的行内元素不会换行，在一行内显示

<span>

7、块级元素与行内元素

块级:独占一行

div,p,hn, ... ...

作用:多数在布局上使用

布局：网页上元素所摆放的位置

行内:一行内共存好多个

span,b,i,u,sub,sup

作用:包围文本，方便为文本添加效果

8、分割线

<hr> --> horizontal水平

作用：在网页构建一条水平线

空标记

块级元素

常用属性:

size:尺寸,水平线的粗细程度

width:水平线的长度

可以给具体值

百分比,可随浏览器打下而调整

align:水平对齐方式

color:颜色

9、预格式化标签

<pre></pre>

保留源文档中的格式，保留原来的换行和空格

10、URL

1、绝对路径

1、什么是绝对路径

文件从最高级目录下开始的完整的路径

E:\apache-tomcat-7.0.57\temp\safeToDelete.tmp

特点:无论当前资源路径是什么，使用绝对路径总能找到需要的资源

完整的URL组成:

假设:百度的服务器上有一个文件夹 img , 里面有logo图像logo.gif

http://www.baidu.com/img/logo.gif

组成:

1、协议

http://

2、主机名/域名

www.baidu.com

3、目录路径

img

4、资源名称(音视频、图片、html文档)

logo.gif

2、相对路径

users/admin/images/admin.png

../../users/image/u.jpg:向上两级目录，进入到users文件夹，再进入到image文件夹找到u.jpg的图像

平级用名字

子级进目录

父级向上返

3、根相对路径

所有的路径表示，都使用 /

/images/user.jpg

11、图像

<img>

属性:

src

width

height

title:

alt:

1、图片不显示的时候的文字提示

12、链接元素

<a></a>

属性:

href:链接资源的URL(绝对、相对、根相对路径)

target:目标，新网页打开的位置

\_blank : 在新标签页中打开

\_self : 在自身页面窗口中打开

name : 定义锚点名称

链接的各种表现形式:

1、下载资源

<a href="abc.rar"></a>

2、页面跳转

<a href="abc.html"></a>

3、电子邮件链接

<a href="mailto:zhaoxu@tarena.com.cn">联系我们</a>

4、返回页面的空链接

<a href="#"></a>

5、链接到Javascript

通过a定义锚点:

html中的锚点相当于是html文档中某个位置的记号

1、定义锚点

<a name="anchorName"></a>

2、跳转到锚点处

跳转到本页的锚点处:

<a href="#anchorName">Return</a>

跳转到其他页面的锚点处:

<a href="页面URL#anchorName"></a>

练习:

1、在之前的练习基础上完成图像的插入以及超链接的跳转

2、插入图-1 以及文本

3、插入图-2 以及文本，并实现页面跳转 路径:http://www.tmooc.cn/

table

定义表格用 table

定义行 tr

定义列 td （单元格标记）

所有放在table里的内容必须放在td里

边框属性：

border 边框宽度

bordercolor 边框颜色

尺寸

width 宽 height 高

\*背景

bgcolor 背景色

background 背景图

细节

align table在父元素中的对齐方式

\*cellspacing 单元格的外边距

\*cellpadding 单元格的内边距

默认不为0。

tr height

align（内容在单元格的水平位置）

valign（内容在单元格的垂直位置）

td width、height

bgcolor、align、valign

colspan 跨列

rowspan 跨行

结构标记

header 文档头、页头……

使用率较高

nav 导航。一般 用于制作页面中的主导航

使用率不高

section 文档的节。主体内容

使用率较高

article 文章、帖子、博客……

指引用的一段文字。

aside 独立于主体内容之外的内容

边栏、侧导航……

footer 页脚、文章结尾

使用率较高

列表

有序列表 ol

无序列表 ul

列表项 li

列表可以嵌套。无序列表嵌套时，列表

项目编号会自动变化。

start 定义有序列表的起始值

type 定义有序列表和无序列表的列表项样式

type=A/a/1/I/i

type=disk/circle/square

定义列表

dl

dt 定义标题

dd 定义详情（内容）

表单

定义表单 form

表单控件

input type 定义其功能

type=text 默认文本框

type=password 密码框

type=submit 提交按钮

type=checkbox 复选框

name 定义控件名。

没有name不能提交

value 当控件为按钮时，value

为按钮上的文字。

1、表单元素

1、form

1、作用

用于定义表单中的基本信息，如提交地址，提交方式等

2、注意：如果表单元素没有出现在<form>中，那么将不能被提交到服务器

3、属性

1、action

动作

服务器处理数据的程序地址，由服务器端开发人员来提供。默认提交到本页。

2、method

表单将以什么样的方式将数据提交给服务器(明文还是密文的)

默认值:get

常用值:

1、post 以密文的方式发送数据到服务器不限制传递数据的长度

2、get 以明文的方式发送数据到服务器,有传递数据的长度限制2KB

3、name

定义表单的名称

作用：方便使用JS来获取表单

4、enctype(了解)

表单数据的编码方式

不同的数据将使用不同的编码方式，主要体现在普通文本的传输和文件传输上

值:

1、application/x-www-form-urlencoded(默认值)，普通文本的编码方 式

2、multipart/form-data

上传文件时的编码值

3、text/plain

文本编码方式(不推荐)

2、表单元素

1、input元素

text,password,checkbox,submit

1、属性

1、type

用于区分不同的input元素的样式与功能

2、value

当前表单元素的值(默认值)

3、name

当前表单元素的名称，用于提交给服务器去使用

4、disabled

禁用控件

注意：只要该属性出现在表单元素中，那么该元素就不能用

2、文本框与密码框

文本框:<input type="text" />

密码框:<input type="password" />

属性:

maxlength:最大长度

readonly:只读

3、单选按钮、复选框

1、单选按钮

<input type="radio" />

属性:

1、name

如果一组单选按钮中只有一个能被

选中上，那么它们的name

属性值必须一样

2、复选框

<input type="checkbox" checked />

注意：作为一组复选框，它们的name属性值必须要一致

4、按钮

1、提交按钮

作用：将表单数据提交给服务器端

<input type="submit" />

2、重置按钮

作用:将所有的表单元素恢复到默认值状态

<input type="reset" />

3、普通按钮

作用：不带有任何功能的按钮。主要去执行自定义的脚本内容。

<input type="button" />

5、隐藏域和文件框

1、隐藏域

作用:在页面看不见的地方保存一段信息

<input type="hidden" />

一般情况下，会将安全系数较高的并且不想让用户看到的数据保存在隐藏域中

2、文件框

作用：能够实现文件选择的功能

使用场合：上传文件(头像、rar、txt......)

<input type="file" />

注意:要将form的enctype属性改位multipart/form-data

2、textarea多行文本域

<input type="text" />

语法:

<textarea>文本内容</textarea>

属性:

name

id

cols:指定文本区域的列数(定义宽度)

rows:指定文本区域的行数(定义高度)

使用场合:

1、注册信息时的详细条款(只读)

2、接受用户输入的多行数据时

3、select和option

作用:允许用户从下拉选项框中选择内容(而不是输入)

分类:

1、下拉列表

2、滚动列表

语法:

1、<select></select> 表示下拉(滚动)列表

属性:

name:

size:显示的备选信息的数量

multiple:是否允许多选

2、<option></option> 表示下拉列表中的选项

属性:

value : 选项值

selected:预选中

4、其他元素

1、label元素

作用:与表单元素相关联，点击该label时就如同点击那个表单元素

语法:

<label>文本</label>

属性:

for:用于关联表单元素的id

2、为控件分组

<fieldset> : 为控件分组

<legend> : 分组的标题

3、其他常用标记

1、浮动框架

可以在一个html页面中显示多个html文档内容,也就是将其他的页面嵌入到当前页面中

语法:

<iframe></iframe>

属性:

src:浮动框架中的网页URL

height:高度

width:宽度

2、摘要与细节

作用:将网页的一部分信息通过类似于下拉列表的方式进行显示和隐藏

<details></details> : 用于定义细节

<summary></summary> : 显示当前details中的标题

注意:summary必须要出现在details中的第一个子元素的位置

3、度量元素

<meter></meter>

多数表示进度的显示、比例的显示

主要属性:

1、min : 范围的最小值，默认为0

2、max : 范围的最大值，默认为1

3、value : 度量值，默认为0

4、时间元素(了解)

<time></time>

用来定义公历时间(24)小时制或日期

语法:

<time>显示内容</time>

属性:

datetime:规定日期/时间，日期与时间之间，通过T来表示分割

5、高亮显示文本元素

<mark>

作用:用于突出显示文本

语法:

<mark>文本</mark>

**CSS**

1、什么是CSS

Cascading Style Sheet

层叠样式表

级联样式表

样式表

2、作用

实现了内容与表现的分离

提高了代码的可重用性和可维护性

2、CSS样式表使用方式

使用方式共分三种，使用任何一种都可以更改页面的样式。

1、内联方式

内联样式

将CSS样式定义在HTML的标签中

特点：不通用，只能定义某一标签的单独样式。

2、内部样式表

将一系列样式定义在HTML的<head>中

特点:当前网页任何一个标签都可以使用，体现出可重用性。

3、外部样式表

将样式定义在外部的css文件中，可以由任何一个页面进行样式表的引用

特点：真正意义的实现了样式表的可重用性

3、内联方式的CSS

定义在HTML标签中的style属性里

<h2 style="样式规则"></h2>

CSS语法:

样式规则:属性名称:属性值

color:red

background-color:blue

多个规则之间，用;进行区分

background-color:blue;color:red;

4、内部样式表

样式内容出现在<head>中的<style>中

<head>

<title></title>

<style>

/\*

这里是注释

\*/

/\*这里是注释\*/

</style>

</head>

样式组成:选择器和样式声明

选择器:决定哪些元素能够使用定义好的规则

样式声明:由一对大括号包围着所有的样式规则

选择器

{

/\*样式规则\*/

属性名称:属性值;

属性名称:属性值;

}

5、外部样式表

step1:创建一个文件已css作为后缀

step2:在文件中写样式相关代码

选择器

{

属性名称:属性值;

}

step3:在页面中对css文件进行引用

<head>

<link rel="stylesheet" type="text/css" href="引入样式表的地址" />

<style>

@import "css/style.css";

/\*其他样式定义\*/

</style>

</head>

6、CSS样式表特征

1、继承性

大部分的样式属性是可以被继承的

<body>

<p></p>

</body>

body

{

font-size:12px;

}

2、层叠性

可以为一个标签定义多个样式

不冲突时，多个样式可以合并成一个

h1

{

color:red;

}

h1

{

background-color:yellow;

}

h1

{

font-size:20px;

color:green;

}

合并后:

h1

{

color:red;

background-color:yellow;

font-size:20px;

}

3、优先级

样式定义冲突时，会根据样式规则的优先级进行应用样式

级别最高：内联样式

其次：内部样式表和外部样式表,就近原则

4、调整优先级

!important

语法 ：

选择器{属性:属性值 !important;}

7、选择器

规定了哪些元素能够使用定义好的样式

1、通用选择器

表示:\*

可以与任何一个元素匹配

多数用在设置网页整体的默认样式

\*

{

font-size:12px;

font-family:"Microsoft Yahei";

}

2、元素选择器

由html元素名称作为选择器

作用：用于修改当前选择器所对应的元素的默认样式

表示:元素名称

p{font-size:12px;}

div{color:red;}

练习:

1、通过通用选择器，设置字体大小为12px，字体为微软雅黑

2、通过元素选择器，将p元素的文本颜色改为蓝色

3、在页面创建p、div标签，查看效果

3、类选择器

语法:.className{}

注意:类名不能以数字开始

.1234 错误

.font

使用:能够附带class属性的元素都能使用类选择器,将元素的class属性设置为类名即可

一个标签也可以同时引入多个类选择器

<div class="c1 c2">

表示div同时引用了c1类样式和c2类样式

类选择器和元素选择器结合使用:

能够实现对某种元素中不同样式的细分控制。又称之为"分类选择器"

语法:元素选择器.类选择器{}

p.important{}

4、id选择器

特点:作用于指定id属性值得元素上

语法:#idName{}

#header{ .... }

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

id属性:

1、标识元素的唯一的值

2、引用样式表中的id样式

5、群组选择器

选择器的声明是以逗号","隔开的选择器列表

作用:将相同的规则用于多个元素中

语法:选择器1,选择器2,..{}

6、后代选择器

<div>

<span>

<span class="s1"></span>

</span>

</div>

作用:根据元素的位置关系(层次)找到元素，然后改掉样式

语法:选择器1 选择器2{}

div span{}

div .s1{}

7、子代选择器

子代：只描述父子关系的,标签嵌套也局限于父子关系的嵌套

<fieldset>

<span class="s1"></span>

<div>

<span class="s2"></span>

</div>

</fieldset>

span.s1是fieldset的子代元素

span.s2是fieldset的子代元素吗?不是!是div的子代元素,是fieldset的后代元素。

语法:

选择器1>选择器2{}

fieldset>span{}

fieldset>.s1{}

8、伪类选择器

作用:为一些选择器添加特殊的效果,多数表示的是一个元素(选择器)的不同状态

语法: 通过 : 作为结合符

选择器:伪类选择器

伪类的分类:

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

链接伪类:

:link 尚未访问的链接

:visited 访问过的链接状态

a:link{}

a:visited{}

动态伪类:

:hover 鼠标悬停在元素上的状态

:active 元素被激活时的状态

:focus 元素获取焦点时的状态

需要掌握:

:link

:visited

:hover

:active

8、选择器的优先级

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

div{color:blue;}

#d1{color:green;}

.red{color:red;}

优先级:

内联>ID>类(伪类)>元素

9、颜色

#rrggbb : 6位16进制数字

每一位的范围:0-9 A-F

#3A7765

#000080 : 蓝色

#B9E5FC :

#FFAA11 -> #FA1

10、尺寸

在HTML中哪些元素适合使用尺寸属性

1、所有的块级元素

div,p,h1...ul ol li

2、大部分行内元素不适合使用尺寸属性

span

3、存在width与height属性的元素

img,table ... ...

11、边框

border:边框

方向:

left,right,top,bottom

大小:

width

样式:

style

颜色:

color

border:四个方向的 大小 样式 颜色

border:大小 样式 颜色;

border-方向:大小 样式 颜色;

border-颜色:

border-样式:

border-大小:

border-方向-颜色:

border-方向-样式:

border-bottom-width:10px;

border-style:solid;

border-bottom:1px solid black;

12、边框阴影

box-shadow:向方框添加阴影

box-shadow:h-shadow v-shadow blur spread color inset

h-shadow:必选，水平阴影偏移距离

v-shadow:必选，垂直阴影偏移距离

blur : 可选，模糊程度

spread : 可选，阴影尺寸

color : 可选阴影颜色

inset : 可选，将外部阴影改为内部阴影(outset)

13、图片边框

由图像作为元素的边框显示效果

border-image:src width repeat

repeat:

repeat: 平铺

round : 铺满

stretch : 拉伸

border-image-source:图片的路径

border-image-source:url(img/aa.jpg);

border-image-width:图片边框宽度

border-image-repeat:边框平铺

repeat,round,stretch

14、轮廓

轮廓(outline)是绘制于元素周围的一条线，在边框的外围

语法:

outline:color style width;

outline-color:

outline-style:

outline-width:

1、外边距-margin

1、取值

margin:10px; -->上下左右各10px

margin:5px 10px; -->上下5px 左右10px

margin:5px 10px 5px;-->上5px 左右10px 下5px

margin:5px 10px 5px 10px;

上5px 右10px 下5px 左10px

2、外边距合并

当上下外边距相遇时，它们将形成一个外边距，称之为合并外边距。最终的边距距离，以数值大的为准。

2、内边距 - padding

1、什么是内边距

内容区域与边框之间的空间

注意:会扩大元素边框所占用的区域

2、赋值

padding:value;

内边距属性值可以为像素、百分比，不能是负数

padding-left:

padding-right:

padding-bottom:

padding-top:

padding:value(上下) value(左右);

padding:v1(上) v2(左右) v3(下);

padding:v1(上) v2(右) v3(下) v4(左);

3、CSS重写

目的：去除有些标签默认显示效果

hn

p

ul

ol

dl dt dd

方式：将margin padding 全部设置为0 , 将 list-style-type:none;

4、背景图像

1、背景图像尺寸

属性:background-size

取值:

v1 v2 : 宽度 高度

v1% v2% : 原始大小的百分比

cover : 扩展背景图像，使背景图完全覆盖背景区域

contain:将图像扩展至最大尺寸，使宽度和高度自适应内容区域(按比例拉伸)

2、背景图片固定

属性:background-attachment

值:

scroll:滚动，默认值

fixed : 背景图像固定

3、背景定位

属性:background-position

值:

left

right

top

bottom

center

x% y%:第一个值表示水平偏移量，第二个值，垂直偏移量

x y:x表示水平偏移量，y表示垂直偏移量

4、背景绘制区域(颜色)

属性:background-clip

取值:

border-box:背景被裁剪到边框,默认值

padding-box:背景被裁剪到内边距框

content-box:背景被裁剪到内容框

5、背景定位区域(图像)

属性:background-origin

取值:

border-box:

padding-box:

content-box:

6、背景属性

在以个属性中声明所有的相关的背景属性

语法:

background:color url() repeat attachment position;

background:red url(user.jpg) no-repeat -35px 25px;

5、渐变

raidal-gradient([size at postion])

size:半径

postion:

left

top

bottom

right

x y : 距离左上方圆心定位

1、控制字体

1、指定字体

属性:font-family:value value value;

2、字体大小

font-size:value;

3、字体加粗

font-weight:

取值:

bold : 加粗 <b></b>

normal : 正常

value :

100

200....900...

400 : normal

900 : bold

4、字体样式(斜体)

font-style:

取值:

normal

italic <i></i>

5、小型大写字母

font-variant

取值:

normal

small-caps

6、字体属性

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

font:12px Times,New York,微软雅黑

2、控制文本格式

1、文本颜色

属性:color

2、文本水平对齐方式

text-align:left/center/right

3、文字线条修饰

属性:text-decoration

none

underline 下划线

overline 上划线

line-through 删除线

4、行高

注意:如果行高的高度高于文本自身的高度的话，那么文本将在这段区域内垂直居中显示

属性:line-height

取值:

常用值 与所在容器设置相同高度

5、首行文本缩进

属性:text-indent

取值:缩进的距离 比如:20px

6、文本阴影

属性:

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

7、文本溢出

1、处理空白

white-space:normal/nowrap

2、文本溢出

text-overflow:

取值:

clip:不显示

ellipsis:使用 ... 替代溢出文本

注意:text-overflow属性，配合着 overflow:hidden 联用

否则，text-overflow无效果

8、文本换行

1、长单词换行

long long long

longlonglong

属性:word-wrap

取值:

noraml

break-word

2、文本换行

word-break:

取值:

normal 正常

break-all:不关心单词显示完整的问题，该换就换

keep-all:当单词到达边界处后根据半角空格" "和连字符"-"

3、表格

1、表格特有属性

1、边框合并

属性:border-collapse

取值:

separate(默认值)

collapse(边框合并)

2、边框边距

table的属性cellspacing

注意:必须是border-collapse为separate的时候才有作用

属性:border-spacing

取值:

v1 : 水平和垂直方向的距离都是v1

v1 v2 : v1表示水平间距,v2表示垂直间距

3、标题位置

属性:caption-side

取值:

top: 默认

bottom: 靠下

4、显示规则

固定布局

属性:table-layout

取值:

auto : 列宽度由单元格内容来决定

fixed : 列宽度由表格宽度和列宽决定，不受内容所影响

练习:

1、添加导航

由a标签组成

浮动

更改宽度

行高/内边距

1、框模型

框:盒子

框模型:外边距 内边距 边框

对象实际宽度:左外边距+左边框+左内边距+宽度+右内边距+右边框+右外边距

1、外边距

元素边框周围的空白区域

2、表示

margin:value

margin:auto;处理元素居中显示的方式之一

margin:v1 v2;

margin:v1 v2 v3;

margin:v1 v2 v3 v4;

3、CSS重写

\*{

margin:0px;

padding:0px;

list-style-type:none;

}

4、外边距合并

两个垂直外边距相遇时，以最大值作为外边距

5、内边距

内容区域与边框之间的距离

表示:

padding:value;

2、背景

1、背景颜色

background-color:颜色值;

2、背景图像

background-image:url("image/1.jpg");

3、背景平铺

属性:background-repeat

取值:

repeat

repeat-x

repeat-y

no-repeat

4、背景尺寸

属性:background-size

取值:

v1 v2

v1% v2%

cover

contain

5、背景固定

属性:background-attachment

取值:

scroll

fixed

6、背景定位

属性:background-position

取值:

top

bottom

left

right

center

x% y%

x y

7、背景绘制区域(背景色)

属性:background-clip

取值:

border-box

padding-box

content-box

8、背景的定位区域(背景图像)

属性:background-origin

取值:

border-box

padding-box

content-box

9、背景属性

属性:

background:color url repeat attachment position;

3、渐变

属性:background-image

取值:

线性渐变

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

径向渐变

radial-gradient([size at position],color point,color point)

重复线性

repeating-linea-gradient

重复径向

repeating-radial-gradient

4、浏览器兼容

火狐Firefox : -moz-

Chrome\Safari : -webkit-

欧鹏Opera : -o-

1、文本

font-family

font-size

font-weight

font-style

font-variant:small-caps;

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

color:

text-align

vertical-align

text-decoration

line-height:文本垂直居中方式

text-indent:20px;

text-shadow:h,v,blur,color

溢出

1、处理空白

处理文本换行还是不换行

white-space:

normal

nowrap

2、文本溢出

text-overflow:

clip

ellipsis

注意:要与overflow:hidden联用

3、换行

1、长单词

word-wrap:

normal

break-word

2、文本换行

word-break

normal

break-all

keep-all

2、table

1、border-collapse

separate

collapse

2、border-spacing

3、caption-side

4、table-layout

3、浮动

1、什么是浮动

元素脱离文档流，从文档中浮动起来，靠在某一个边框上

2、属性

float:left/right

clear:left/right/both

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

1、display

none : 隐藏

block : 行内元素按块显示(显示)

inline : 块元素按行内显示(显示)

inline-block : 行内块元素,本身是行内元素，允许改变宽和高

2、visibility

hidden

visible

collapse

3、vertical-align

1、td中修改文本的垂直对齐方式

2、图像周围的文本相对于图像的垂直对齐方式

练习:京东导航条

1、通过nav实现导航栏

2、完成 "全部商品分类"

div - id : category

<p></p>

超链接

下箭头

3、样式

1、文本

12px

#666

Arial,"Microsoft yahei"

2、nav

width:900px

heigin:40px;

边距：上下10 左右居中

背景色:

3、category

width:210px;

浮动

背景色:

4、a

行内块级元素

字体 ....

5、b

大小 20 X 20

背景图

背景偏移

margin

4、列表

场合:使用列表做导航

<ul>

<li></li>

</ul>

5、

<ul id="cate\_box">

<li>

<h3>

<a>图书</a>、

<a>音像</a>、

<a>数字产品</a>

</h3>

</li>

</ul>

练习

1、cate\_box

border:2px

width:203

padding-left/right:3px

background:

2、ul ：margin

3、li

width

height : 28

border-bottom

4、li>h3

margin:0px;

padding-left:

font-size

宽

高

行高

加粗

5、h3:hover

背景色 白色

上下边框 #ddd

6、h3 a

去下划线

颜色

练习:

1、在 #cate\_box 的第一个li中装一个div :id -> #sub\_cate\_box

2、在 #sub\_cate\_box 中放三个div

div -->#sub\_cate\_items

div -->#sub\_cate\_banner

div -->#close

3、样式

1、#cate\_box

更改成相对定位

2、#sub\_cate\_box

更改成绝对定位

left: 190~200之间

top:2px

边框

背景

宽 700

高 400

阴影

padding

3、#sub\_cate\_items

左浮动

宽

高

border

4、#sub\_cate\_banner

右浮动

宽

高

border

4、#close

修改定位为绝对定位

width

height

**JavaScript**

1. 什么是js

2. 如何编写、调试js

3. \*\*\*变量

4. \*\*\*数据类型

1. 什么是js：

前端三大语言：

HTML：专门编写网页内容的语言

CSS：专门美化网页内容样式的语言

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

交互：用户操作 程序处理或响应操作 返回结果

JavaScript：专门编写交互程序的语言

运行于js解释器/引擎中

解释执行——逐行读取，先读到先执行

后读到后执行

js发展史：

Netscape：navigator

1995 JavaScript——像极了Java，但比Java简单

Java "write once run anywhere"

1996 JavaScript 1.1-->ECMA

\*ECMAScript标准-->定义了js语言的核心语法

Netscape-->遵照标准，实现了JavaScript语言

Microsoft-->遵照标准，实现了JScript语言

W3C：DOM标准-->专门操作网页内容的标准

所有浏览器都兼容的标准

BOM：专门操作浏览器窗口的编程接口

\*js=ECMAScript标准+DOM标准+BOM

2004年 Netscape-->Mozilla-->Firefox

2. 如何使用js：

如何运行：2种：1. 浏览器内置js引擎

2. 独立安装js引擎: nodejs

如何编写：4种方式：

1. 直接在浏览器控制台(F12)中，编写js脚本：

Console: 控制台：专门编写和调试js程序的窗口

比如：console.log("Hello World!");

console特指控制台窗口

.：的

log：console中提供的一个功能

功能接受一个字符串参数，在控制台输出该字符串

字符串：一串字符的集合

字符串必须用引号包裹，但单双引号都行

\*程序中的关键字区分大小写！

\*字符串中的字符，不要求大小写。

2个按键：

多行程序：shift+Enter

上下方向键：切换已执行过的代码——避免重复编写

2. 在网页中编写js脚本程序：3处

1. <script>元素中：

直接编写在<script>中的js程序，随网页加载过程解释执行

document: 当前网页

write: 写入功能，直接写入body元素中。

不带换行

write中的字符串必须符合html语法

强调：write中换行：<br>

2. 单独js文件保存js脚本程序：2步：

1. 创建.js文件，保存js脚本程序

2. <script src="url"></script>引入外部js文件

脚本程序：不需要预编译，边解释边执行的程序

3. 元素的事件处理程序中编写——了解

不随网页加载过程执行，

只有触发事件时，才执行。

window:指代当前窗口

alert:弹出警告框

如何调试：

bug：程序中出现的一切错误

debug：发现错误，解决错误

js程序出错：希望的交互效果没出来——>F12-->console

console：错误信息：3部分：

错误的类型：错误的原因 出错位置的链接

出错现象：出错位置同<script>下的程序：

出错前的程序正常执行

出错位置及其之后的程序不执行

其他<script>中的程序，不受影响

1. \*\*\*变量

2. 常量

3. \*\*\*数据类型

1. \*\*\*变量：

What：内存中存储1个数据的存储空间，再起个名字

When：程序中的数据都要先用变量保存，再处理

如何使用变量：声明 赋值 取值

声明：在内存中创建一个新变量

如何声明：var 变量名;

强调：仅声明未赋值的变量，默认值自动是undefined

命名规则：

1. 不能以数字开头

2. 不能用保留字

3. \*见名知义\*

4. 驼峰命名：首字母小写，之后每个单词首字母大写

比如：backgroundColor

listStyleType

studentName

赋值：将一个数据存入变量中

如何赋值：变量名=值

=：专门表示将右边的值，存入左边的变量中保存

\*对已有值的变量赋新值，新值会替换旧值\*。

强烈建议：

声明同时初始化：第一次给变量赋值

如何声明同时初始化: var 变量名=值;

取值：从变量中取出数据，进行运算

如何取值：在程序任何位置，只要使用变量名等效于直接使用变量中存储的数据

特殊情况：2种：

1. 为一个从未声明过的变量赋值：不会出错！

会自动创建同名变量，再保存数据

2. 尝试从一个未声明过的变量中取值：会出错！

ReferenceError：未找到或未定义

\*\*\*声明提前：在程序正式执行前，都会将所有var声明的变量提前到开始位置，集中创建。\*\*\*赋值留在原地\*\*\*。

2. 常量：一旦声明并初始化后，值不可改变的量！

When：只要保存一旦创建，不可擅自改变的量时

比如：π g

How：const 常量名=值;

常量名全大写

3. \*\*\*数据类型：

What:内存中存储不同数据的存储格式

js中数据类型：2大类

1. 原始类型：值保存在变量本地的数据类型

5种：Number 专门保存数字的类型

String 专门保存字符串的类型

Boolean 专门保存真/假二选一的类型

undefined 只有一个值undefined

null 表示不指向任何地址

2. 引用类型：值不保存在变量本地的数据类型

Number类型：专门保存数字的类型，不加引号

When：要参与算数计算或比较大小的数据

比如：价钱 年龄

所占空间：64位二进制数保存

64Bit=8bytes

强调：数字所占空间和数值大小无关

　　　　　所有数字都占8字节

1GB=1024MB

1MB=1024KB

1KB=1024bytes

1byte=8Bit

\*\*\*舍入误差：因为计算机中无法精确表示1/10，导致计算中产生的误差。不可避免！

解决：四舍五入——n.toFixed(2)：将n按2位小数四舍五入

String类型：专门保存一串字符的序列，必须带引号

When：保存文字时，或保存仅用于显示的数字时

比如：姓名 身份证号 手机号 QQ号

所占空间：js的程序内存中的字符都是用unicode标识的

unicode：对全球主要语言中的每个字符都编一个号

每个字母、标点：1字节

每个汉字：2字节

比如："Hello马里奥"

111112 2 2 =11个

\*\*\*字符串内容一旦创建，不可改变！

要想改变只能创建新字符串，替换旧字符串

+：如果参与+运算的值中包含字符串类型数据

+法运算自动变为字符串拼接！

Boolean类型：只有两个值true/false。

When：如果表示二选一的值时，就用Boolean类型

回顾：

1. 变量：声明(var) 赋值(=)

取值(使用变量名等效于使用变量中的值)

\*\*\*声明提前：程序开始执行前，

将所有var声明的变量提前到开始位置

赋值留在原地

console.log(a); //undefined

var a=100;

console.log(a); //100

\*\*\*使用变量报错：尝试从\*未声明\*的变量中\*取值\*

\*\*\*给未声明的变量赋值，\*不会报错\*！会自动创建同名变量

2. 数据类型：2大类

原始类型：值保存在变量本地的数据类型

5种：Number String Boolean undefined null

Number：8bytes 舍入误差-->四舍五入

String：每个英文字符 1byte

每个汉字字符 2bytes

一旦创建内容不可改变

Boolean: true false

引用类型：值不保存在变量本地的数据类型

js中的变量 vs 数据库

内存中 硬盘上的文件

临时的 持久的

客户端 服务器端只存一份

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是弱类型编程语言：3方面：

1. 声明变量时，不用规定变量存储的数据类型

2. 赋值时，动态决定变量的数据类型

同一个变量，先后可保存不同类型的数据

3. 运算时，js会根据需要，动态转换数据的类型

2大类：

1. 隐式转换：无需程序员干预，js自动完成的类型转换

仅讨论+运算中的隐式转换：2句

算数计算中，一切类型都隐式转为number类型，再计算

比如："2"-->2 true/false-->1/0

特例：+运算中，只要有一方是字符串，两数据都转为字符串，且+运算变为字符串拼接！

表达式：由数据，变量和运算符组成的一个公式

每个表达式默认都从左向右，两两计算

\*每个表达式有且仅有一个运算结果！\*

NaN：Not a Number-->说明运算中包含了无法转为数字的值

不是数字 的 数字

NaN和任何数据做计算，永远返回NaN

typeof(x): 专门用于判断x的数据类型，返回数据类型名

typeof(NaN): number

隐式转换，仅影响表达式的运算结果，不影响变量中存储的实际值。

2. 强制转换：程序员通过主动调用专门函数，执行转换

1. 任意类型 to String: 2种：

var str=x.toString();

js中，一切数据都有toString方法

var str=String(x);-->隐式转换

何时使用：程序在需要时，自动调用

\*\*\*2. 任意类型 to Number: 2种：

任意类型 to Number: var n=Number(x);-->隐式

String to Number: 2种：

var n=parseInt(str);

原理：从str开始位置逐个读取每个字符

直到碰到第一个不是数字的字符时，停止读取

自动忽略开头碰到的空格

比如：var width="12px";

width+"2px"-->"12px2px"

Number(width)+Number("2px")-->NaN

parseInt(width)+parseInt("2px")

parseInt("12px")+parseInt("2px")

width=14+"px";

var money="$400"; NaN

"34.56"; 34

var n=parseFloat(str);

原理和parseInt完全一致！但是，parseFloat,只认第一个小数点

比如：parseFloat("34.56"); 34.56

parseFloat("34.56.78"); 34.56

Number(x) vs parseInt(str)

比如：Number("12px")-->NaN

parseInt("12px")-->12

tring to Number-->parseInt

Number(true)-->1

parseInt(String(true))

parseInt("true")-->NaN

x to Number-->Number()

3. 任意类型-->Boolean: 1种：

var bool=Boolean(x);-->隐式转换

"" NaN undefined null 0 --> false

其余-->true

课堂练习：

输入框：var input=[window.]prompt("输入提示");

[window.]alert();

\*\*\*凡是从页面上获得一切数据，都是字符串类型

2. \*\*\*运算符和表达式：

程序：让计算机按照人的想法去执行任务

运算符：程序中模拟人的思维运算或判断的符号

运算符：算数运算符：+ - \* / % ++ --

%运算：取余数 6789%100-->89

被除数/除数,不取商，而是取除不尽的余数部分

何时使用：2大用途：

1. 判断奇偶数：n%2-->1 奇数 否则 偶数

2. 限制一个运算的结果，不超过某个最大值（除数)

正课：

运算符：

1. 算数运算符：++ --

2. 关系运算

3. 位运算——了解

4. 赋值运算

5. 三目运算

1. \*\*\*算数运算：仅++

++：将当前变量中的值递增1

比如：n++; ++n; <==> n=n+1;

何时使用：今后，只要对变量中的值递增1，还存回变量中

3种情况：n++单独使用,前++和后++都一样

++运算参与到其他表达式中：

前++(++n):将n中的值+1，然后返回\*新值\*

后++(n++):将n中的值+1，然后返回\*旧值\*

--：将当前变量中的值递减1

比如：n-- <==>n=n-1;

2. \*\*\*关系运算：将两个值做比较：> < >= <= == !=

只能返回两个值之一：如果成立，返回true；否则返回false

隐式转换：将所有类型都转为number类型，再比较！

特殊情况：3种：

1. 两字符串做比较：依次PK每一位字符的unicode编号,只要有一位字符，分出大小，就不再比较

比如："3">"10" -->true

"Smith">"Scott" -->true

"m" "c"

109 99

"张东">"张无忌" -->false

"东" "无"

19996 26080

2. NaN: NaN和任何数据做大小或等于比较永远返回false

NaN和任何数据做\*不等于\*比较，永远返回true

isNaN(num):专门判断num是否是NaN

如果是NaN返回true，否则返回false

When：今后只要判断一个数值是否是数字或能否被转为数字，都用isNaN。如果是数字,返回false！否则返回true

3. undefined vs null

undefined == null true

===: 全等：数据类型要相同，再比较值是否相等

不带隐式转换

何时使用：只要不希望关系运算中自动隐式转换时

3. \*\*\*逻辑运算：综合多个关系运算的结果得出最终结论

隐式类型转换：自动将每个条件转为Boolean类型

鄙视题：

alert(4&&5);//?

alert(4||5);//?

返回值只能二选一：满足条件返回true，否则返回false

3个：

关系运算1&&关系运算2：

而且

两个条件必须都是true，才返回true，否则返回false

关系运算1||关系运算2：

或者

只要有一个条件为true，则返回true

只有两个条件都为false时，才返回false

！关系运算：颠倒关系运算的true/false结果

\*\*\*短路逻辑：如果前一个条件已经可以得出结论，

则后续条件不再执行！

作业：5\_login.html：鄙视题

\*\*\*利用短路逻辑：2种：

1. 利用&&的短路逻辑：

关系运算&&操作：只有满足关系运算时，才执“操作”

2. 利用||的短路逻辑：

关系运算||操作：如果前一个条件不满足，才执行后一个"操作"

作业：逻辑运算：课堂练习：2个问题

问题1：如何判断一个字符是汉字：

第一个汉字"一"-->"\u4e00"

最后一个汉字"\u9fa5"

4. 位运算：2个

左移：n<<m 将n的二进制，左移m位

比如：1<<3 -->1\* 2的3次方

右移：n>>m 将n的二进制，右移m位

比如：64>>3 -->64/ 2的3次方 ==> 8

5. 扩展赋值运算：5个：+= -= \*= /= %=

仅讨论+=: n+=2; ==>n=n+2

将n中的值+2后，再保存回n中

何时使用：今后只要对变量中的值进行累加时，都要用扩展赋值运算。

6. \*\*\*三目运算：根据不同的条件，多选\*一\*，返回不同的结果

语法：条件1?值1:

条件2?值2:

...?...:

默认值;

无论三目运算多么复杂，结果只能是一个值。

回顾：

1. 隐式转换：

算数计算：一切都转为Number

特例：+运算，碰到字符串，都转字符串，改拼接

关系运算：一切都转为Number

特例1：两个字符串比较，逐位PK每个字符的unicode

特例2：NaN不等于任何数值

NaN做> < ==比较永远返回false

isNaN(num)——专门判断一个数值是否是NaN

反着用:专门判断num是否是一个数字

特例3：null==undefined ==> true

null === undefined ==> false

逻辑运算：一切都转为boolean类型

2. 强制转换：

1. 任意 to String:

var str=x.toString();

var str=String(x); -->隐式转换

2. 任意 to Number:

var num=Number(x); -->隐式转换

string to Number:

parseInt/Float(str);

3. 任意 to Boolean:

var bool=Boolean(x);

"" NaN null undefined 0-->false

3. 运算符：

1. 算数运算：% ++ --

前++：将变量中的值+1，返回新值

后++：将变量中的值+1，返回旧值

2. 关系运算：> < >= <= != ==

3. 逻辑运算：&& || !

短路逻辑：

根据条件判断是否执行某个操作：

1. 条件&&操作：只有满足条件才执行“操作”

2. 条件||操作：如果不满足条件，才执行“操作”

4. 位运算： << >>

5. 扩展赋值运算：+= -= \*= /=

6. 三元运算：

条件1?值1:

条件2?值2:

...?...:

默认值;

正课：

1. \*\*\*函数

2. 全局函数——了解

3. \*分支结构

1. \*\*\*函数: 封装一个专门任务的步骤清单的代码段

起一个任务名

何时使用：只要定义一个专门的任务，都要封装一个函数

使用函数: 声明 定义 和调用

如何声明并定义一个函数：

function 函数名([参数变量列表]){

函数体:包含任务的步骤清单（代码段）

[return 返回值]

}

如何调用函数：[var 返回值=]函数名([参数值列表]);

函数只有调用时才执行！

反复调用，会反复执行相同操作！

为什么定义函数：一次定义，反复使用！——代码重用！

一次修改，处处生效！

函数的参数：专门接收传入函数内的数据的\*变量\*

何时定义参数：如果一个函数必须某些数据才能正常执行时

参数可让函数变得更灵活

如何定义参数变量：function 函数名(变量1,变量2,...)

如何传递参数值：调用时: 函数名(值1,值2,...)

强调：一般值的个数和顺序，应与声明时的参数列表一致

返回值：方法调用后，返回的执行结果！——1个数据

如何返回数据：函数体中：return 要返回的数据；

强调：1. return单独使用，表示退出函数执行的意思

2. 函数仅负责返回值，不负责保存返回值！

如何获得返回值：调用时：var 变量=函数名();

\*\*\*变量作用域：一个变量的可用范围

js中2种作用域：

1. 全局作用域：一个变量可以在程序的任何位置被访问。

2. 函数作用域：一个变量仅能在函数调用时，内部被访问

js中2种变量：

1. 全局变量：定义在全局作用域中的变量

2种：1. 直接在任何函数外声明的变量

全局变量都属于window——全局对象

2. 无论在任何位置，为从未声明过的变量赋值时

自动在全局创建同名全局变量！

2. 局部变量：定义在函数作用域中的变量

2种：1. 在函数定义内部声明的变量

2. 参数变量天生就是局部变量

函数其实是引用类型的对象；

函数名其实是指向函数对象的变量；

函数的定义和调用过程：

1. 声明和定义函数时：

window中创建了函数名变量

window外创建了函数定义的对象-->封装函数的定义

函数名变量中保存了指向函数对象的地址

2. 调用时，创建局部变量：2种

3. 调用过程中，如果局部有，就不用全局的

如果局部没有，才会用全局的

如果全局也没有，则报错！

4. 调用结束后，释放所有局部变量！

犀牛书：JavaScript权威指南

望远镜：JavaScript高级程序设计(红皮)

微信公众号:前端大全

js1k——国际大赛

Chrome断网-->看到下恐龙-->按空格

正课：

1. \*\*\*函数：\*声明提前; \*\*\*按值传递

2. 全局函数——了解

3. \*分支结构：

1. \*\*\*函数:

\*声明提前：在程序执行前或函数被调用前

将var声明的变量和function声明的函数

提前到\*当前作用域\*的顶部集中创建

强调：仅声明提前，赋值留在原地

\*\*\*按值传递：js中无论变量间赋值或使用变量传递参数时，

都是将变量中的值，复制一个副本给对方！

2. 全局函数——了解

什么是：\*ES标准中\*已经定义好的，开发者可直接调用的函数

parseInt/Float(str)

isNaN(num)

alert(str)X

prompt(str)X

编码/解码:

编码：将url中的非法字符，改为合法字符表示

http://..../s?word=%E5%BC%A0

张

utf-8格式编码

汉字占3字节

url中不允许出现多字节字符！

解决：将url中的多字节，变为单字节

url中不允许参数值中出现保留字符：比如: /

http://tmooc.cn -->http%3A%2f%2ftmooc

解决：encodeURIComponent

解码：将url中的非法字符编码后的内容，恢复成原文

decodeURIComponent

eval: 计算字符串格式的表达式的值

或执行字符串格式的js语句

3. \*分支结构：

程序结构3大类：

顺序：程序默认都是从上向下逐行执行

分支：根据条件判断的结果，有选择的执行不同代码段

循环：程序可以反复执行同一代码段，到达临界时退出

需求：Input Process Output

Input: 单价(price) 数量(count) 收款金额(money)

Output: 总价(total) 找零(change)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Process:

总价=单价\*数量

如果总价>=500，则打8折

找零=收款金额-总价

分支结构：3种：

1. if结构: 1件事 满足条件就执行，否则不执行

语法：if(条件){

满足条件才执行的代码段

}

2. if...else...结构: 2件事 满足条件执行if中的代码段

不满足条件执行else中代码段

语法: 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. else if结构：多件事，根据条件，多选\*一\*执行！

语法：if(条件1){

满足条件1才执行的代码段1

}else if(条件2){

满足条件2才执行的代码段2

}else if(条件3){

... ...

}[else{

万一之前所有条件都不满足，才执行默认代码段 }]

如果程序已经引入后一个条件判断，就已经暗示前一个条件肯定不满足！

分支结构 vs 三目/短路

如果只是返回值-->三目/短路

如果操作复杂-->分支结构

作业：PPT 14页 2题

作业：使用if...else if结构实现可莱托指数

正课：

1. \*分支结构：switch case

2. \*循环结构：3种

1. switch case结构：都是根据不同条件执行不同的代码段

用switch中的表达式的值和case中的值做\*全等\*比较

语法：switch (表达式){

case 值1:

代码段1;

case 值2:

代码段2;

...

...

[default:

默认代码段;

]

}

break: 停止当前结构的执行，并跳出当前结构

switch vs else if:

switch: 条件是全等比较时，首选switch case

else if: 除全等比较外，希望灵活定义条件时

2. 循环结构：让程序反复执行一段代码段

只有达到临界条件时，停止循环

3要素：

1. 循环条件：继续循环的条件：圈数<3

2. 循环变量：在循环条件中，用作比较的变量：圈数

循环变量一般会向着不满足循环条件的趋势不断变化

比如：圈数：每转一圈+1

3. 循环体：反复执行的代码段

3种：

1. while循环：

语法：var 循环变量=初值;

while(循环条件){

循环体;

迭代变化循环变量;

}

退出循环：2种方式：

1. 自然退出：不满足循环条件时，自动退出循环

2. 手动退出：程序员在循环体中用break强行退出循环

2. do while循环：

语法：var 循环变量=初值;

do{

循环体;

迭代变化循环变量;

}while(循环条件);

差别：当第一次条件满足时，while和do while完全一样！

当第一次条件就不满足时，

do while:至少会执行一次

while: 一次都不执行！

何时使用do while代替while？

只要希望哪怕第一次条件都不满足，至少也能执行一次

\*js中没有块级作用域：分支/循环结构中声明的变量，出了结构，照样使用！\*

1. \*for循环：

语法:

for(var 循环变量=初值;循环条件;迭代变化循环变量){

循环体;

}

何时使用：只要循环变量的变化规律固定！

优先选择for循环

continue: 结束本轮循环，继续下一轮循环

continue一般都可以使用否定条件代替！

回顾：

1. 分支：

短路逻辑：

条件&&操作：1件事，满足条件就做，否则不做

只限于操作简单的情况下

值1||值2：如果值1有效，就返回值1，否则返回值2

三目运算：多个值，按条件判断，多选一

条件1?值1:

条件2?值2:

...?...

默认值;

if...else if结构：当操作复杂时，就用if结构。3种：

if(条件){代码段}：1件事，满足条件就做，否则不做

if(条件){：2件事，满足条件做代码段1，否则做代码段2

代码段1

}else{

代码段2

}

if(条件1){：多件事，根据条件判断，多选一执行

代码段1

}else if(条件2){

代码段2

}...

}[else{

默认代码段

}]

switch case：如果条件是全等比较时，就用switch case

switch(表达式){

case 值1:

代码段1;

[break;]

case 值2:

代码段2;

[break;]

[default:

默认代码段]

}

2. 循环：3种：

while: 循环变量的变化规律不确定时

do-while: 如果第一条件就不满足，也希望至少能执行一次时

for: 循环变量的变化规律固定时

三要素：

1. 循环条件：继续循环的条件

2. 循环变量：循环条件中用作比较的变量

从几开始，每次增/减几，到几结束

3. 循环体：循环反复执行的代码段

循环体中都会向着不满足循环条件的趋势改变循环变量

正课：

1. \*\*\*数组：

程序=数据结构+算法

好的数据结构，可极大提高程序的执行效率

相关的多个数据应集中存储，集中管理

分类和排序

数组：一组连续的变量组成的集合——统一起一个名字

批量管理多个数据

如何使用：创建 赋值 取值

1. 创建：4种：

1. var 变量名=[];-->创建一个空数组对象

2. var 变量名=[值1,值2,...]-->创建数组同时，为数组指定初始元素(数组中每个值)

数组是引用类型的对象

变量中保存了数组对象的地址值，也成引用了数组对象

3. var arr=new Array();-->创建一个空数组对象

new：创建一个新对象，并返回新对象地址

Array：js中的数组类型

new Array()-->创建一个数组类型的新对象，并返回新对象的地址。

4. var arr=new Array(n);-->创建n个元素的数组

new Array(值1,值2,...)

找到数组：使用变量等效于直接使用变量引用的数组对象

数组对象中，每个元素都有一个下标：每个元素的位置号

下标从0开始，到元素个数-1结束

使用数组中的每个元素：

数组变量[i]-->获得i位置元素中的值

\*数组变量[i]的用法和普通变量完全一样！\*

2. 赋值：数组变量[i]=新值;

3. 取值: 数组变量[i]

null和undefined:

undefined: 是所有没有值得变量的默认值，自动赋值

null: 主动释放一个变量引用的对象

何时使用null：

当使用完一个较大的对象时，主动释放对象总是好的习惯

垃圾回收器：专门释放对象内存的一个程序

在底层，后台，伴随当前程序同时运行

引擎会定时自动调用垃圾回收期

总有一个对象不再被任何变量引用时，才释放

js数组 vs 其他语言的数组：

\*js数组下标越界不会出错！\*

为不存在的位置赋值：不会出错！会自动创建指定下标的新元素。

从不存在的位置取值：不会出错！也不会增加新元素

而是返回undefined

正课：

1. \*.length属性

2. \*数组遍历

3. \*关联数组

4. \*\*\*冒泡排序——手写\*

5. \*数组API

1. 数组是对象：封装了一组数据，并提供了对数据的操作方法

.length属性：获得数组中的元素个数!=实际的元素个数

何时使用:3种：

arr[arr.length-1]: 获得任意长度数组中的最后一个元素

arr[arr.length]=新值: 向数组末尾追加一个新元素

改小length的值，可删除末尾元素

2. 数组遍历：从下标0位置开始，依次取出每个元素，反复执行相同的操作。

三要素:

1. 循环条件：下标i<arr.length

2. 循环变量：下标i,

从0开始，每次增1，到arr.length-1结束

3. 循环体：xxx

3. 关联数组：可以自己定义下标名称的数组

索引数组：自动生成下标的数组都是索引数组

为什么：["范冰冰", 91, 65, 95]

sname shx yuw eng

如何创建关联数组: var fbb=[];

key value 对儿

fbb["sname"]="范冰冰";

fbb["shx"]=91;

如何访问关联数组中的元素：fbb["sname"]

关联数组中的.length属性失效！

关联(hash)数组: 下标是不能重复的

优势：利用Hash算法，精确定位某个下标的位置

不用遍历

索引数组：缺：下标自动分配，无意义

要想按内容查找元素，只能从头开始遍历！

遍历关联数组：依次遍历arr中每个元素，

将当前元素的下标存入key中

for(var key in arr){

key，仅得到当前元素的下标

arr[key]，得到当前元素的值！

}

4. \*\*\*冒泡排序：——手写

5. \*数组API：数组=数据+操作数据的API

1. 数组 to String:2种：

var str=arr.toString();

-->输出数组中的内容，每个元素用逗号分隔

-->自动调用

var str=arr.join("连接符")

-->输出数组中的内容,自定义元素间连接符

何时使用：2种固定套路

1. 将字符组成单词：arr.join("")

2. 将单词组成句子：arr.join(" ")

2. 拼接和截取：\*API无法直接修改原对象，而返回新对象\*

拼接：var newArr=arr.concat(另一个数组,值1,值2...)

复制子数组：var subArr=arr.slice(starti,endi+1);

含头不含尾

3. splice: 删除 插入 替换

1. 删除：直接修改原数组对象

[var deleted=]arr.splice(starti,删除个数)

返回本次删除元素组成的数组

2. 插入：

arr.splice(starti,0,新值1,...)

3. 替换：

arr.splice(starti,删除个数,新值1,...)

4. arr.reverse(); 颠倒数组中所有元素

回顾：

1. API: 应用程序编程接口

已经实现的，现成的对象和方法。——提高开发效率

2. 数组API：

1. arr to String:

var str=arr.toString()-->自动，逗号分隔

var str=arr.join("连接符")-->自定义连接符

固定套路：1. 将字符拼接为单词: arr.join("")

2. 将单词拼接为句子: arr.join(" ")

\*优化：问题：频繁字符串拼接，造成内存浪费\*

解决：先将要拼接的字符串放入数组

最后，arr.join("")

var arr=["a"];

arr[arr.length]="b";//1个arr 2个str

arr[arr.length]="c";//1个arr 3个str

arr[arr.length]="d";//1个arr 4个str

[ "a" , "b" , "c" , "d"]

arr.join("")

2. 拼接和截取：原对象保持不变，返回新对象

var newArr=arr.concat(新值1,另一个数组,新值2...)

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

第二个参数可省略，如果省略表示从start一直取到结尾

两个参数都可以取负值，就近使用下标

含头不含尾

3. arr.splice(): 删除 插入 替换

4. arr.reverse();

正课：

1. \*\*\*数组API

\*\*\*数组排序

\*栈和队列

2. \*二维数组

3. \*\*\*String

1. 数组API：

1. 数组排序：arr.sort();

问题：默认将所有元素转为字符串升序排列

\*\*\*比较器函数：专门比较任意两值大小的函数

2特点：1. 2个参数，分别表示要比较的任意两值

2. 无论比较逻辑是什么，都要返回一个数字

\*\*\*函数对象作为参数：

js中函数本质是一个对象

函数名本质是指向函数对象的变量

如何自定义sort方法的排序？

arr.sort(比较器函数)

\*颠倒比较器结果的正负，可改升序为降序排列\*

正课：

1. \*栈和队列

2. \*二维数组

3. \*\*\*String

1. \*栈和队列：js中没有真正的栈和队列的类型

一切都是用数组对象模拟的

栈：只能从一端进出的数组，另一端封闭

FILO

何时使用：今后只要仅希望数组只能从一端进出时

如何使用：2种情况：

1. 末尾出入栈：已入栈元素的下标不再改变

入栈: arr.push(新值1,...)

出站: var last=arr.pop()

2. 开头出入栈：每次入栈新元素时，已入栈元素的位置都会向后顺移。

入栈：arr.unshift(新值1,...);

出站：var first=arr.shift();

队列：只允许从末尾进入数组，必须从开头出数组

FIFO

结尾入队列：arr.push();

开头出队列：var first=arr.shift();

2. \*二维数组：数组中的元素，又引用了另一个子数组对象

何时使用二维数组：2种：

1. 只要保存横行竖列的二维数据

2. 保存上下级关系的数据时

如何创建二维数组：2种：

1. 先创建空数组，再赋值:

var arr=[];

arr[0]=["石家庄","保定","唐山"];

arr[1]=["德州","济南","烟台"];

2. 创建数组时，同时初始化子数组：

var data=[

[0,2,2,4],

[2,4,4,8],

[16,32,16,8],

[4,2,128,4]

];

如何访问二维数组中的数据：arr[行下标][列下标]

比如：data中第2行，第3个元素

data[1][2]

强调：二维数组中，行下标不能越界！

console.log(data[1][5]); //undefined

console.log(data[5][1]); //

//undefined[1]

　　遍历二维数组：

固定套路：外层循环遍历行，内存循环遍历列

var arr=[[...],[...]...]

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

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

}

}

3.\*\*\*String: 字符串：

什么是字符串：多个字符组成的一个\*只读\*的集合(数组)

凡是数组对象中，不修改原对象的API，字符串都能用！

比如：.length属性：字符个数；都可用[i]访问每个字符

比如：slice, indexOf

凡是数组对象中，直接修改原对象的API，字符串都不能用！

比如：push, sort , splice

内置对象：ES标准中已经定义好的，

由浏览器厂商已经实现的对象

11个：String Number Boolean

Array Date Math RegExp

Error

Function Object

Global（全局）

包装类型：临时封装原始类型数据，

提供对原始类型数据的操作方法

3种：String Number Boolean(不能new)

何时使用包装类型：自动使用

为什么使用包装类型？原始类型的数据本来没有任何方法

当程序中使用原始类型调用方法时，程序会自动创建包装类型对象封装该数据。

函数调用完，包装类型自动释放

String API：\*所有API都无法直接修改原字符串\*

1. 转义字符：当字符串的内容和程序的特殊符号冲突时

或字符串中包含功能字符时，都用\转化字符的意义

\n：换行 \t: 一次缩进Tab键

2. 大小写转换：将字符串中所有字符都统一转为大写或小写

何时使用：不区分大小写时

比如：用户名 验证码 电子邮件

转大写:str=str.toUpperCase();

转小写:str=str.toLowerCase();

3. 获取指定位置的字符：3种：

1. str[i]

2. str.charAt(i)

3. str.charCodeAt(i)-->获得字符的unicode号

4. 查找关键字的位置：2种：

1. var i=str.indexOf(kword[,starti]);

从左向右，查找kword首字符在字符串中的下标

如果没找到，返回-1

starti：从starti位置开始向后查找下一个

如果省略starti，默认从0开始找

2. var i=str.lastIndexOf(kword[,starti])

从starti位置开始，向左找！下一个kword

作业：day01-->素材-->preview

-->Homework\_string\_mario

回顾：

1. \*\*\*String：多个字符组成的只读字符数组

2. \*\*\*String API：所有API不可能直接修改原字符串

1. 大小写转换：

都转大写：str.toUpperCase();

都转小写：str.toLowerCase();

2. 获取指定位置字符：3种：

1. str[i]

倒数第二个:str[str.length-2]

2. str.charAt(i);

3. str.charCodeAt(i);-->返回字符的unicode号

3. 查找关键字的位置：2种：

1. var i=str.indexOf("kword"[,starti]);

如果找不到，返回-1

starti默认是0

2. var i=str.lastIndexOf("kword"[,starti]);

查找starti位置左侧第一次出现的关键字

starti默认是length-1

正课：

1. \*\*\*String API：

2. \*\*\*正则表达式：

1. \*\*\*String API:

1. 获取子字符串：3种：

1. var subStr=str.slice(starti[,endi+1])

pid.slice(6,-5+1);

2. var subStr=str.substring(starti[,endi+1])

不支持负数参数

pid.substring(6,pid.length-5+1);

3. var subStr=str.substr(starti,取几个);

pid.substr(6,8)

正课：

1. \*\*\*String API:

2. \*\*\*正则表达式

1. \*\*\*String API:

1. 分隔字符串：var arr=str.split("分隔符")

arr中以数组形式保存每段子字符串

何时使用：今后只要分段处理字符串时

都要先分隔，再遍历

固定套路：将字符串切割为字符数组: //Hello

var chars=str.split("");

2. 模式匹配：按照规则查找或替换字符串中的子内容

查找：2种：

1. 只能从开始位置，找第一个匹配的关键字

var i=str.search(/kword/i);

何时使用：仅判断有没有关键字时

vs str.indexOf:

search专门用于模式匹配

indexOf不支持模式匹配

2. 获得所有关键字的内容，或个数：

var arr=str.match(/kword/ig)

arr中保存了所有关键字的内容

arr.length表示匹配的关键字的个数

\*如果没找到，返回null\*

\*都要先判断是否为null，再处理！\*

\*match无法获得关键字的下标位置！\*

\*\*\*替换：var newStr=str.replace(模式,"替换内容")

2. \*\*\*正则表达式：专门规定字符串中字符\*格式规则\*的表达式

何时使用：只要定义字符串格式规则，都用正则表达式

最简单的正则：一个关键词的原文，就是最简单的正则

1. 备选字符集：规定某\*一位\*字符可选的备选文字列表

语法：[备选字符列表]

强调：无论备选字符集中由有多少字符，

都必须且只能选1个

一个[]只代表1位字符！

比如：6位数字：[0123456789][0123456789][0123456789][0123456789][0123456789][0123456789]

^:除了xxx之外，都行——了解

2. -：如果备选字符连续，可用-表示"到"

3. 预定义字符集：专门表示常用的连续的字符集

\d ==>[0-9] : 1位数字

\w ==>[0-9a-zA-Z\_] : 1位数字，字母或下划线

\s ==>[空字符] : 代表一位空字符：空格，Tab符

.:除了换行外，其他所有的字符

\D,\W,\S-->除了\d,\w或\s --了解

4. 量词：规定一个字符集出现的次数：2种

明确数量：3种：{min,max}:最少min位，最多max位

{min,}:至少min位

{n}:必须n位

身份证号:至少15位数字

2位数字

最后一位数字或x X

后三位可有可无，但最多出现1次

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

不确定数量：3种：

? ==>{0,1}: 可有可无，最多1次

\* ==>{0,}:可有可无，多了数量不限

+ ==>{1,}:至少一次，多了数量不限

手机号：+86或0086 可有可无，最多1次

空字符可有可无，多了数量不限

1

从3,4,5,7,8选一个

最后必须是9位数字

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

正课：

1. \*\*\*RegExp

2. \*\*\*String中支持正则的API:

split、search、match、replace

3. \*Math

4. \*Date

1. \*\*\*RegExp对象：封装了正则表达式，提供了利用正则表达式执行验证和查找的API。

何时使用：只要利用正则表达式执行验证和查找功能时

创建：2种：

1. var reg=/正则表达式/ig;

何时使用：正则表达式在运行时不会改变

2. var reg=new RegExp("正则表达式","ig");

何时使用：在运行时正则表达式可能被改变时

强调：所有\都要改为\\

var str1="\\d{6}";

var str2="\\d{15}(\\d\\d[0-9xX])?";

var reg=new RegExp(str2,"");

如何使用：2种操作：

1. 格式验证：var bool=reg.test(str)

验证str的格式是否符合reg的要求

验证：要求从头到尾必须完整匹配！

问题：test方法默认只要部分匹配即可！

解决：都要在正则表达式前加^，后加$

^表示开头位置的xxxx

$表示结尾位置的xxxx

^和$联用表示从头到尾必须完整匹配

密码强度要求：6密码，只能是字母，数字的组合

至少包含一位大写字母，

至少包含一位数字

预判：1. 不全由小写字母或数字组成

可能包含大写或符号

(?![a-z0-9]\*$)

2. 不全由大写字母或小写字母组成

可能包含数字或符号

(?![a-zA-Z]\*$)

正式规则：只能是字母数字的组合-->屏蔽了符号

[a-zA-Z0-9]{6}

完整：(?![a-z0-9]\*$)(?![a-zA-Z]\*$)[a-zA-Z0-9]{6}

2. 查找API：

var arr=reg.exec(str)：在str中查找所有符合reg规定的关键字

要加g

其中:arr[0]:保存了当前关键字的内容

arr.index:保存了当前关键字的位置

reg.lastIndex:保存下次开始查找的位置

何时使用：只要即查找内容，又查找位置，只能用reg.exec

总结：查找关键字：

1. var i=str.search(reg)-->判断有没有

缺点：不能指定开始位置

2. var arr=str.match(reg)-->获得所有关键字内容，个数

缺点：不能返回位置

3. var i=str.indexOf("kword"[,starti])

-->从指定位置开始，查找下一个关键字的位置

缺点：不支持正则

4. var arr=reg.exec(str)-->即获得内容，又获得位置

缺点：复杂

正则：贪婪模式与懒惰模式

默认贪婪模式：默认总是匹配最长的符合条件的字符串

原因：.\*或.+等引起

懒惰模式：仅匹配最短的符合条件的字符串

贪婪-->懒惰: (.\*?)

从正则表达式匹配结果中获取子内容：2步：

1. 正则表达式中：用()包括要获取子内容的部分

2. 程序中,本次查找后: RegExp.$n

取\*本次\*匹配结果中第n个()匹配的子内容

正则中n从1开始

2. \*\*\*String中的模式匹配API：4个

2个查找：str.search(reg) str.match(reg)

1个替换：str.replace(reg);

1个切割：

正课：

1. \*\*\*String中的正则API:str.replace() str.split()

2. \*Math

3. \*Date

1. \*\*\*String中的正则API:

1. 替换：str.replace(reg,"替换值")

\*固定套路：利用replace格式化数据

"替换值"中，也可使用$n,和RegExp.$n的用法完全相同

2. 切割：var arr=str.split(reg);

2. \*Math:专门执行数学计算的对象

封装了数学计算中常用的常量

1. 取整：3种：

上取整: Math.ceil(n)

下取整: Math.floor(n)

四舍五入取整: Math.round(n);

round vs toFixed

toFixed Number对象 按任意小数位数 返回字符串

round Math对象 只能取整！ 返回数字

2. 乘方/开平方:

乘方：Math.pow(n,m):计算n的m次方

开平方: Math.sqrt(n):计算n的平方根

3. 获取最大值/最小值;

Math.max(a,b,c...)

Math.min(a,b,c...)

\*\*\*固定套路：变相实现获取数组中最大值

Math.max.apply(Math,arr);-->Math.max(arr[0],...)

4. 随机数：Math.random(); 0<=n<1

任意min到max之间取一个随机整数

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

3. \*Date: 封装了一个时间点，提供了对时间和日期的操作API

Date中封装了？从1970年1月1日0点0分0秒至今的毫秒数

创建Date对象：4种：

1. var now=new Date(); //创建一个新日期对象同时，保存了\*客户端\*当前时间点的毫秒数。——获得当前时间

自定义时间对象：

2. var time=new Date("xxxx/xx/xx [xx:xx:xx]")

3. var time=new Date(年,月-1,日[,时,分,秒)

date.getTime(): 获得日期对象中的毫秒数

Date API：

JavaScript-->左侧js对象

-->js日期

-->中间找：JavaScript Date对象参考手册

1. 每个分量都有一对儿get/set方法

2. 方法的命名：年月日星期 都不带s

时分秒毫秒 都以s结尾

比如：设置小时: setHours()

获取月份：getMonth()

特殊：年: get/setFullYear()-->使用4位完整年份

星期: getDay()：获得星期的序号

日 一 二 。。。六

0 1 2 6

\*星期没有set方法！

月中的日: get/setDate()

3. 返回值：只有月中的日从1开始到31结束

其它分量都是从0开始到减1结束

月份：1月 2月 3月...11月 12月

0 1 2 10 11

日期如何计算？2种：

1. 两日期对象直接相减，结果是毫秒差！

2. 对任意分量做加减：先用get取出来

做计算

再用set放回去

日期转字符串输出：

1. date.toLocaleString(); 转为完整日期字符串

2. date.toLocaleDateString(); 仅转为日期部分

3. date.toLocaleTimeString(); 仅转为时间部分

回顾：

Date API

1. 每个分量都有一对儿get/set方法

getDay()获得星期，没有setDay()

2. 命名：年月日星期,不加s

时分秒毫秒,加s

get/setFullYear()

3. 返回值：月中的日取值范围从1开始到31结束

其余都是从0开始到减1结束

星期：日 一 二 六

0 1 2 6

月份：1月 2月 11月 12月

0 1 10 11

Date计算：2种：

1. 两日期对象可直接相减，结果毫秒差

2. 对任意分量做加减:先get出来，计算，再set回去

\*直接修改原日期对象\*

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

补：创建日期对象第四种方式：复制日期对象

var date1=new Date();

var date2=new Date(date1.getTime());

//new Date(毫秒数)

date2.setXXX(date2.getXXX()+-n)

正课：

1. \*错误/异常处理

2. \*\*\*Function对象

\*\*\*\*闭包

1. \*错误/异常处理

什么是错误：导致程序无法继续执行的异常状态

js中一旦发生错误，就会自动创建一个Error类型对象

js中有几种错误类型（6种）

SyntaxError: 语法错误

ReferenceError: 引用错误，找不到变量或对象

TypeError: 类型错误，错误的使用了对象中的方法时

RangeError: 范围错误，参数超范围

EvalError: 调用eval函数时出错

URLError: URL错误

错误处理：在程序发生错误时，保证程序不退出或正常退出

如何错误处理：

语法：tryCatch块：3部分

try{

正常代码段中，可能出错的代码段

}catch(err){//仅在发生错误时才执行

//一旦发生错误err中就会自动存入Error对象

1. 记录/显示错误的信息

2. 继续向调用者抛出异常

}[finally{

无论对错，一定都会执行的代码段

释放资源

}]

何时需要定义错误处理？某段只要有可能出错，都要包在tryCatch块中。

建议：尽量优先使用if...else结构判断已经预知的错误

只有无法预知的错误，才用try catch。

tryCatch用法之一：解决浏览器兼容性问题

抛出自定义异常：throw new Error("自定义错误消息")

比如：如果程序员甲定义了一个方法，被程序员乙调用

如果程序员乙错误的使用甲的方法，

甲的方法中就要抛出自定义错误消息

乙在调用时，就要将甲的方法包在try块中

正课：

1. \*\*\*Function对象

1. \*arguments对象

2. \*\*\*\*Function对象的本质

3. \*\*\*匿名函数

4. \*\*\*\*闭包

1. \*\*\*Function对象：js中一切都是对象

连函数也是对象

函数名其实是引用函数定义对象的变量

1. \*arguments对象：

重载：程序中可定义多个相同函数名，不同参数列表的函数

调用者不必区分每个函数的参数

执行时，程序根据传入的参数个数，自动判断选择哪个函数执行。

tmooc/java/java面向对象/类中的方法

js语法不支持重载！但可用arguments对象模拟重载效果

arguments对象：函数对象内，自动创建的专门接收所有参数值得类数组对象。

arguments[i]: 获得传入的下标为i的参数值

arguments.length: 获得传入的参数个数！

即使定义了参数变量，arguments对象同样会收到所有参数值

2. \*\*\*\*函数对象本质：

1. 创建函数对象：3种：

1. 声明方式：function 函数名(参数){

函数体;

return 返回值

}

函数名和函数定义都被提前

在调用前后位置定义都行

2. 函数直接量：var 函数名=function(参数){

函数体;

return 返回值

}

仅函数名变量声明会提前，

函数定义留在本地

必须定义在调用前！

3. 使用new创建函数类型对象：

var 函数名=new Function("a","b",...,"函数体");

2.\*\*\*内存中的函数对象\*\*\*

创建函数对象时：同时创建2个对象：

函数对象:函数的定义

作用域链对象：保存了函数对象可用变量的位置的对象（栈）

默认第一项指向window对象

调用函数时：又会创建1个新对象：

活动对象：专门保存局部变量的对象

在作用域链对象中追加指向活动对象的引用

调用后：作用域链中活动对象的引用出栈

活动对象因无人引用而释放

3.\*\*\*匿名函数：定义时，不指定函数名的函数

何时使用：2大用途：

1. 匿名函数自调：定义完，立刻执行

执行完立刻释放

何时使用：只有确定函数只执行一次时！

如何自调：(function(参数){

函数体

})(参数值);

自调：定义在哪儿，就在哪儿执行，不提前

2. 匿名函数回调：向将函数作为对象传递给另一个函数

由另一个函数自主决定在需要时调用

何时使用：只要将一个函数对象传递给其他方法调用时

如何回调：直接将匿名函数的声明传入另一个函数中。

4.\*\*\*闭包：

问题：全局变量和局部变量的缺陷

全局变量：容易全局污染

局部变量：无法共享，不能长久保存

既可以共享，长久保存，又不会全局污染——闭包

解决：3步：闭包三特点：

1. 定义外层函数，封装被保护的局部变量

2. 定义内层函数，执行对外层函数局部变量的操作

3. 外层函数返回内层函数的对象

并且外层函数被调用，结果被保存在全局变量中

何时使用：即反复使用局部变量，又避免全局污染

就要用闭包

回顾：

1. 闭包：

判断闭包3特点：

1. 嵌套函数

2. 内层函数操作了外层函数的局部变量

3. 外层函数将内层函数返回到外部

被全局变量保存住

判断闭包执行结果：

\*1. 外层函数被调用几次，就有几个受保护的局部变量副本

2. 来自一个闭包的函数被调用几次，受保护的局部变量就变化几次

正课：

1. 面向对象：

1. 创建自定义对象

2. \*\*\*\*继承

1. 面向对象：在程序中都是用一个对象来描述现实中一个具体的东西。

现实中的一个东西都包含属性和功能：

属性：描述一个东西特点的变量，一个值

功能：东西可以执行的操作

什么是对象：封装多个数据和方法的存储空间

什么是自定义对象：封装现实中一个东西的属性和功能的存储空间。

现实中东西的属性会成为对象中的属性变量。

现实中东西的功能，会成为对象中的方法(函数)

2. 创建自定义对象：3种方式：

1. var obj={'属性名1':值1,

'属性名2':值2,

'功能名1':function(){...}

};

js中一切都是对象！所有对象的底层都是hash数组

属性：如何访问属性:2种：obj.属性名 obj["属性名"]

访问对象中不存在的属性

(访问数组中不存在的下标): 不会出错，返回undefined

强行给不存在属性赋值，不报错！js会自动创建同名属性

如何判断某个对象是否包含指定成员：3种

1. obj.hasOwnProperty("成员名")

2. "属性名" in 对象

如果找到，返回true，否则返回false！

3. 直接使用obj.属性名作为条件:

arr.indexOf!==undefined

如果不包含,返回undefind-->false

如果包含,返回值或function-->true

何时省略：判断方法是否存在时，可省略!==

如果确定属性值一定不是null,0,"",NaN

也可省略

方法：如何在方法中，访问当前对象自己：

\*\*\*\*this关键字：运行时，指代正在\*调用\*方法的对象

（.前的对象）

this本质是window下唯一的一个指针，指向当前正在调用方法的对象

如何在方法内，访问当前对象自己的属性：this.属性名

\*\*\*在方法内访问当前对象自己的属性，必须用this.属性名

\*\*\*this和定义在哪儿无关！仅和调用时使用的当前对象有关

\*\*\*如果无主的调用或赋值，默认this都是window！

正课：

1. \*\*\*面向对象：继承

面向对象三大特点：封装 继承 多态

封装：将描述同一个东西的属性和方法，定义在一个对象中

继承：父对象中的属性和方法，子对象可直接使用

多态：同一个对象，在不同情况下，呈现不同的状态

重载：同一方法名，传入参数不同，执行不同的操作

重写：子对象觉得父对象的成员不好用，可自己定义一个，覆盖父对象的成员

创建对象：3种：

1. 对象直接量

var obj={"属性名":值,

...:...,

"方法名":function(){ ...this.属性名... }

}

2. var obj=new Object(); //创建一个空对象

obj.属性名=值;

obj.方法名=function(){...this.属性名...}

3. 利用构造函数\*反复\*创建\*相同结构\*的对象：

构造函数：描述一类对象结构的特殊函数

2步：

1. 定义构造函数

function 构造函数名|类型名(属性参数1,.....){

this.属性名=属性参数1;

//在当前正在创建的对象中添加一个属性名

//赋值为属性参数1的值

...

this.方法名=function(){

...this.属性名...

}

}

2. 利用构造函数创建对象:

var obj=new 构造函数名|类型名(属性值1,...);

new: 1. 创建一个空对象：new obj={ };

2. 用空对象，调用构造函数

构造函数在空对象中添加属性和方法

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

4. 返回新对象的地址。

继承：js中一切继承都是用原型对象实现的！

原型对象：每个函数对象都有一个原型对象

　　　　构造函数的原型对象负责保存所有子对象共享的成员！

建议：所有子对象共享的方法，都应定义在构造函数的原型对象中。——避免重复定义方法对象，浪费内存。

说明：其实所有内置类型的API都是定义在类型.prototype

扩展对象属性：2种扩展：

1. 扩展共有属性：通过构造函数.prototype添加的属性

2. 扩展自有属性：通过某一个具体子对象添加的属性

判断自有属性或共有属性：

1. 判断自有属性：obj.hasOwnProperty("属性名")

2. 判断共有属性：

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

在原型关系中包含 且 子对象自己\*没有\*

删除属性：delete 对象.属性名

\*仅能删除当前对象自己的属性，无法删除共有属性

全局变量：3种：

var n=1; | window.n=1; | window["n"]=1

不能delete 不能delete 可被delete

原型链：由各级对象的\_\_proto\_\_逐级继承形成的关系

获得任意对象的父级原型对象：

Object.getPrototypeOf(子对象)

==>子对象.\_\_proto\_\_

检查对象的父对象：父对象.isPrototypeOf(子对象)

DOM&BOM

Day 01：

正课：

1. DOM概述

2. \*\*\*DOM树

3. \*查找

1. DOM概述：

DHTML：动态网页技术的统称

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：动态网页技术的统称，=HTML+CSS+JS

XML：可扩展的标记语言，可自定义标签

专门用来存储/传输自描述的结构化数据

逐渐被json替代了

<演员>

<姓名>范冰冰</姓名>

<数学>91</数学>

<语文>65</语文>

<英语>95</英语>

</演员>

"{"姓名":"范冰冰","数学":91,...}"——json

BOM VS DOM

BOM：浏览器对象模型(API),专门操作浏览器窗口的API

没标准！

DOM：文档对象模型(API),专门操作网页内容的API

可以对网页中任意对象，做任何修改！

DOM是标准，90%以上浏览器都严格兼容

核心DOM：操作所有结构化文档(XML,HTML)的通用API

HTML DOM：针对HTML文档的简化API

HTML DOM不能完成所有功能，实际开发中都是核心DOM与HTML DOM配合使用。

HTML DOM：网页中一切都是对象(元素，属性，文字)

同一网页中的所有对象，在内存中父子相连，形成一棵DOM树。

2.\*\*\*DOM树：

文档中的每个元素，属性，文字，注释，都被看做一个节点对象——Node——所有节点对象的父类型

当网页被加载进内存时，浏览器会为网页创建一个document对象。所有节点对象都是document对象的子节点。

document，封装了对网页中所有子节点的增加，删除，查找

Node类型定义了3个公共属性：

nodeType: 节点的类型的数值

何时使用：专门用于判断获得的节点类型

如果是元素节点，返回1

如果是文本节点，返回3

nodeName: 节点的名称

何时使用：专门用于判断获得的标签名

如果是元素节点，返回标签名

\*\*强调：nodeName返回的都是全大写标签名

如果是文本节点，返回#text

nodeValue: 节点的值：

元素节点，返回null

文本节点，返回文本的内容

childNodes: 获得当前节点对象下的所有子节点

类数组对象，[i] .length

DOM树6种关系：6个属性：

父子：4个：parentNode childNodes firstChild lastChild

兄弟：2个：previousSibling nextSibling

回顾：

1. DOM树：

节点树：网页中每一个元素，属性，文本，注释都是节点对象

同一网页中的所有节点对象通过父子关系形成树结构

树根：document对象

节点数中：节点间关系：6个属性

父子关系4种：parentNode childNodes firstChild lastChild

兄弟关系2种：previousSibling nextSibling

childNodes:类数组对象，

\*\*\*动态集合\*\*\*：自己不保存任何实际数据

每使用一次都重新查找一遍

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

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

正课：

1. \*\*\*\*遍历：

2. \*查找

1. \*\*\*\*遍历：从指定父元素开始，按照深度优先的原则

遍历所有各级子节点

2步：

1. 定义一个函数，查找任意父节点下的所有直接子节点

2. 以深度优先为原则，递归调用函数本身

何时使用递归调用：2个场景：

1. 遍历\*不确定层级深度\*的树形结构时：

比如：网页中的元素，网盘的文件夹结构

2. \*不确定层级深度\*的多级管理结构：

元素树：仅由元素节点组成的树结构

其实有一组和节点树6个属性对应的元素树属性

节点树 元素树

父对象 parentNode parentElementNode

所有子对象 childNodes children

第一个子对象 firstChild firstElementChild

最后子对象 lastChild lastElementChild

前一个兄弟 previousSibling previousElementSibling

后一个兄弟 nextSibling nextElementSibling

何时使用：只要仅希望遍历元素节点时，就用元素树

问题：IE8不兼容，children可用

DOM Level2 遍历API：2个——选学

1. 深度优先遍历：NodeIterator

节点迭代器

如何使用：2步：

1. 创建遍历API对象：

var iterator=document.createNodeIterator(

开始的父节点对象,

whatToShow,

null,false

);

whatToShow: NodeFilter.SHOW\_ELEMENT

NodeFilter.SHOW\_ALL

2. 用while循环，反复调用iterator.nextNode()方法

强调：1. 只要nextNode()，就向下一个移动一次

2. iterator.previousNode()，后退一次

\*\*\*作业：为NodeIterator遍历结果，添加缩进

2. 自有遍历：TreeWalker：

使用几乎相同，只不过TreeWalker比Iterator多个别方法

总结：4种：节点树 元素树

API（NodeIterator，TreeWalker）

2. \*查找：5个API：

1. 按id查找：

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

2. 按标签名查找：（向下爬树的主要手段）

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

\*\*\*elems也是动态集合\*\*\*

\*不仅查直接子节点，同时可获得间接子节点\*

3. 按name属性查找：（专门用于查找表单中的元素）

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

\*\*\*elems也是动态集合\*\*\*

4. Selector API：jQuery的核心

var elem=parent.querySelector("选择器");

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

2特点：1. 内置API：执行效率高

2. elems：包含完整对象属性的集合

不会反复查找！

Day 02：

回顾：

1. DOM中一切都是节点对象：Node类型

元素节点对象：Element类型

文本节点对象：Text类型

三大属性：nodeName nodeType nodeValue

六个关系：parentNode childNodes firstChild lastChild

previousSibling nextSibling

两种树：节点树 元素树

2. \*\*\*\*遍历：4种：默认都是\*深度优先\*

1. 遍历节点树——手写,带缩进

2. 遍历元素树——手写,带缩进

3. NodeIterator：nextNode() previousNode()

4. TreeWalker:

3. \*查找：5种：

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

2. var elems=parent.getElementsByTagName("标签名");

3. var elems=document.getElementsByName("name属性");

4. var elems=parent.getElementsByClassName("class属性")

elems：动态集合，每使用一次都重新查找

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

5. Selector API: jQuery的核心

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

var elems=parent.querySelectorAll("任意选择器");

只能从父节点向下找。

找平级：先parentNode向上，再querySelector向下

elems:包含所有属性和方法的完整对象的集合

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

正课：

1. \*修改\*元素的内容或属性：3种东西：

1. 元素的内容

2. \*\*元素的属性/特性

3. 元素的样式：2处：

1. 修改内联样式

2. \*修改样式表中的css规则

1. 获取或修改元素内容：3个属性:

1. innerHTML：获得/设置元素开始标签和结束标签之间的html原文

何时使用：只要获得html原文内容时

固定套路：2个：

1.删除父元素下所有子元素：

parent.innerHTML="";

2. 批量替换父元素下所有子元素

parent.innerHTML="所有子元素标签组成的html" 比如：ul.innerHTML="<li>电影</li><li>剧集</li>"

2. textContent/innerText: 获得开始标签和结束标签之间的

DOM标准 IE8 纯文本内容，不包含标签

何时使用：只要希望去掉标签，仅获得文本时

3. 文本节点的内容：nodeValue

正课：

1. 元素内容

2. \*\*元素属性：标准 自定义

3. 元素样式：内联样式 样式表中的css规则

1. 元素内容：

2. \*\*元素属性：get/set/has/removeAttribute()

所有元素都有attributes属性,[i]访问每个属性——了解

读取属性：4种方法：

1. element.attributes[下标].value

2. var value=element.attributes['属性名']

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

\*\*\*4. var value=element.getAttribute("属性名")

何时使用：只要获得任意属性的值

修改属性：2种：

\*\*\*1. element.setAttribute(name, value);

IE8不支持

只能：element.attributes['属性名']=value

2. element.setAttributeNode(attrNode);

移除属性：2种：

\*\*\*1. element.removeAttribute( '属性名');

2. element.removeAttributeNode(attrNode);

判断元素是否包含属性：2种：

\*\*\*1. element.hasAttribute('属性名') //true或false 2. element.hasAttributes( );

\*\*\*Property vs Attribute

属性 HTML特性

Property: 对象在内存中存储的属性

用.访问

Attribute: 元素对象在开始标签中定义的HTML属性和自定义属性

访问HTML标准属性时。二者完全一致：

比如：<a href="http://tmooc.cn"...

a.href-->属性 -->HTML DOM

a.getAttribute("href")-->特性 -->核心DOM

如果访问自定义属性时，二者不通用！

<li /\*data-age="29"\*/>Eric</li>

读取自定义属性：li.data-age? X

li.getAttribute("data-age");

设置自定义属性：li.age=29-->网页？

li.getAttribute("age")找不到;

li.setAttribute("data-age",29);

3. \*元素的样式：

1. 要修改的样式在哪儿？

2. 优先级

1. 获取或修改内联样式：style对象

在style对象中设置的样式属性，优先级最高！

设置：style.属性名="值"

移出：2种：

style.属性名="";

style.removeProperty("CSS属性名")

问题：仅能操作style属性中定义的内联样式

无法获取或设置样式表中的样式

2. 获取或修改样式表中的属性：内部 外部（）

3步：

1. 获得要修改的样式表对象：

var sheet=document.styleSheets[i];

styleSheets:获得当前网页的所有样式表对象

2. 获得要修改的cssRule：

cssRule：样式表中一个大括号就是一个cssRule对象

var cssRule=sheet.cssRules[i]

cssRule可能嵌套。

3. 获得cssRule中的属性

cssRule.style.属性名

Day 03：

正课：

1. \*创建和删除节点

2. \*HTML DOM常用对象: Table Select

1. \*创建和删除节点

1. 创建节点：

创建元素节点：3步：

1. 创建空元素对象：

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

比如：var a=document.createElement("a");

<a></a>

2. 设置必要属性：

newElem.属性名="值";

newElem.innerHTML="文本";

比如：a.href="http://tmooc.cn";

a.innerHTML="go to tmooc";

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

3. 将元素对象挂载到指定父元素下：2种：

追加：parent.appendChild(newElem);

强调：只有向已经在页面中的父元素追加新节点，才导致渲染

比如: div.appendChild(a);

<div>

...

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

</div>

2. 创建文档片段：documentfragment

文档片段：内存中，临时存储多个子节点的存储空间

何时使用文档片段？反复追加多个平级元素

解决：先将所有平级元素先追加到文档片段中

将文档片段一次性追加到父元素下

\*文档片段不参与追加\*

正课：

1. \*创建、删除节点：

2. \*常用HTML DOM对象

1. \*创建、删除：

插入新元素：parent.insertBefore(newElem,oldElem);

删除节点：parent.removeChild(oldElem);

oldElem.parentNode.removeChild(oldElem);

替换节点：parent.replaceChild(newElem,oldElem);

课堂练习：级联下拉列表：

1. onchange:当内容发生改变时触发

2. select对象：selectedIndex属性：当前选中项的下标

2. \*常用HTML DOM对象：Table Select Form

Table对象：

属性：

tHead tFoot tBodies

rows: 获得表中所有行对象

rows[i]: 获得表中小标为i的一个行对象

方法：

var newRow=insertRow(rowIndex)：

rowIndex写-1，表示在末尾追加

比如：insertRow(-1)

核心DOM：var newRow=document.createElement("tr") table.appendChild(newRow)

deleteRow(rowIndex)：

比如：currRow.parentNode.removeChild(currRow);

table.deleteRow(currRow.rowIndex)

TableRow对象：代表table对象中的某一个tr对象

table.rows集合，就是一组TableRow对象的集合

属性：

cells: 当前行中所有td对象

cells[i]: 获得当前行中下标为i的td

rowIndex: 当前行的下标位置，专用于删除行

方法：

var newCell=insertCell(index)

比如：insertCell(3)

核心DOM：var td=document.createElement("td");

tr.appendChild(td);

deleteCell(index)

TableCell对象：

属性:cellIndex

Select对象:

属性：

options: 获得当前select下所有option

options[i]: 获得当前select下i位置的option

selectedIndex: 获得当前选中的option的下标

方法：

add(新option对象)

比如: select.appendChild(newOpt);

select.add(newOpt);

remove(index)

Option对象：指代select下某一个option元素

如何创建：var newOpt=new Option(innerHTML,value)

创建option对象同时，设置对象的innerHTML和value属性

相当于：var newOpt=document.createElement("option");

newOpt.innerHTML="内容"

newOpt.value="值";

一句话：创建，设置，追加

select.add(new Option(innerHTML,value));

属性：index: 获取当前option的下标位置，专用于删除

selected: 可当做bool用

如果当前option被选中，返回true

否则，返回false

Form对象：

如何找到一个form对象

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

如何找到form中的一个数据采集元素:

var elem=form.elements[i/name]

事件：onsubmit：在正式提交表单前自动触发

Day 04：

回顾：

1. Form对象：

如何找到：document.forms[序号|name]

如何找到数据采集的元素：

document.forms[序号|name].elements[序号|name]

让元素获得或失去焦点：elem.focus()

elem.blur()

获得/失去焦点的事件：onfocus onblur

获得当前正在获得焦点的元素：document.activeElement

事件：onsubmit 在表单正式提交前自动触发

对整个表单执行验证

form.onsubmit=function(){

只要任意一个验证未通过，

就\*取消事件\*:2步：

1. 获得event对象e:

var e=window.event||arguments[0];

if(e.preventDefault){

e.preventDefault() //DOM

}else{

e.returnValue=false //IE8

}

}

表单提交：2种：直接点submit按钮；

如果当前form中任意元素获得焦点，可按回车自动提交

只要自动表单提交前，都会先触发onsubmit，可做验证

BOM

1. \*BOM：

window对象

\*\*\*\*定时器——动画

2. 常用BOM对象：

\*\*\*Navigator

1. BOM：专门操作浏览器窗口的对象

window对象：2个角色：

1. 充当全局对象！

2. 包含BOM常用对象

属性：

大小与位置：

innerHeight/Width: 文档显示区的大小

outerHeight/Width: 窗口大小

pageYOffset：文档左上角到文档显示区左上角的距离

1. 打开新链接：4种效果：

1. 在当前窗口打开新链接，可后退

html:

js:[window.]open("url","\_self")

2. 在当前窗口打开新链接，禁止后退

js:location.replace("新url");

3. 在新窗口打开新链接，可同时开多个

html:

js:

4. 在新窗口打开新链接，只能打开一个

target-->目标窗口的名称

\_self: 自动获得当前窗口名称

\_blank: 创建一个新窗口，随机生成一个不重复的名字

\*窗口名：内存中同一个窗口名只能打开一个

后打开的，会替换先打开的

html: <a href="xxx" target="自定义窗口名"></a>

window.name="自定义窗口名"

正课：

1. window对象：

窗口大小与定位

\*\*\*\*定时器

1. 窗口大小与定位：

大小：

1. window.innerHeight/Width: 文档显示区宽高

outerHeight/Width: 整个窗口的宽高

2. screen.height/width: 桌面完整分辨率宽高

screen.availHeight/availWidth: 去掉任务栏后剩余分辨率的宽高

3. 调整大小：window.resizeTo(width,height)

调整到

resizeBy(变化的width,变化的height)

位置：

1. 左上角x坐标：window.screenLeft||window.screenX;

y坐标：window.screenTop||window.screenY;

2. 将窗口移动到指定坐标：window.moveTo(x,y)

3. 事件发生时，鼠标相对于整个屏幕的坐标：

event.screenX|screenY

2. \*\*\*\*定时器：让浏览器按指定时间间隔反复执行同一任务

2种：

周期性定时器：让浏览器按指定时间间隔反复执行同一任务

如果不手动停止，则一直反复执行

一次性定时器：让浏览器等待一段时间间隔，执行一次任务

自动停止。

在一次性定时器的结尾，每次都重新启动一个一次性定时器

建议：尽量使用一次性定时器，代替周期性定时器

如何使用：周期性和一次性用法完全相同的

周期性：3件事：

1. 动画的每一步要执行的任务(函数对象)

function step(){

每一步要做的事情

}

2. 将一步的任务放入定时器，反复调用

timer=setInterval(step,间隔毫秒数)

3. 必须用全局变量，临时存储定时器的编号

clearInterval(timer)

一次性：3件事

1. 动画的每一步要执行的任务(函数对象)

function step(){

每一步要做的事情

/\*根据条件判断是否有必要继续启动下一个定时器\*/ }

2. 将一步的任务放入定时器，反复调用

timer=setTimeout(step,间隔毫秒数|等待毫秒数)

3. 必须用全局变量，临时存储定时器的编号

clearTimeout(timer)

停止正在等待的定时器

Day 05：

回顾：

1. 定时器

一次性定时器：3件事：

1. \*\*\*每一次要执行的任务\*\*\*

任务函数结尾：都要判断是否继续启动下一次

2. 将任务函数放入定时器中自动反复执行

timer=setTimeout(任务函数,时间间隔)

3. 停止定时器：2句话：

clearTimeout(timer)

timer=null;

正课：

1. 常用BOM对象：navigator history location screen

2. \*\*\*事件：

3. cookie：

1. 常用BOM对象：

1. navigator: 保存浏览器配置信息的对象

cookieEnabled: 判断当前浏览器是否启用cookie

plugins: 保存所有插件对象的集合

userAgent: 保存了浏览器名称，版本号的字符串

正课：

1. window常用子对象：history location screen

2. \*\*\*事件

3. cookie

1. window常用子对象：

history: window对象中保存当前窗体访问过的url的历史记录栈

history.go(1): 前进1次

go(-1): 后退1次

go(0): 刷新当前页

go(n): 前进/后退n次

location：当前窗口正在打开的url地址对象

location.search：获得url中的查询字符串

如果进一步获得参数名和参数值？

先按&分隔，再按=分隔

location.replace("新url"): 在当前窗口打开新链接

不可后退

（history中旧url被新url替换了）

使用location在当前窗口打开新链接，可后退：2种：

location.href="新url";

location.assign("新url");

刷新：location.reload();

screen: 封装当前屏幕的显示信息

screen.height/width: 完成屏幕宽高

availHeight/Width: 去掉任务栏后的剩余宽高

window7下任务栏透明

2. \*\*\*事件：

浏览器自动触发的或用户手动触发的对象状态的改变

DOM Level2 Event标准

IE8：自成体系！

事件处理函数：当事件触发时，自动执行的函数

比如：<button onclick="函数/js语句"></button>

//btn.onclick();

事件处理：

1. 事件定义（绑定事件处理函数）：3种

html： <标签 on事件名="fun()">

d1.onclick=function(){

eval("fun()");

//[window.]fun();

}

强调：fun()中this-->window

如果获得当前目标元素对象：

html: onxxx="fun(this)"

js中定义函数时：fun(elem)

js: elem.on事件名=函数对象;

一个元素的一个事件处理函数，只能绑定一个函数对象

DOM标准：elem.addEventListener(

"事件名",函数对象,是否在捕获阶段触发)

true/false

一个元素的一个事件处理函数，可add多个不同函数对象IE8: elem.attachEvent("on事件名",函数对象)

实际执行的：elem.on事件名(); this-->elem

2. \*\*\*事件周期：从浏览器捕获到事件后，一直到最后一个事件触发完，经历的过程。

DOM标准：3个阶段：

1. （由外向内）捕获：从最外层html元素开始，向内逐层记录每层元素绑定的事件处理函数。到目标元素为止

2. 目标触发：自动执行目标元素上绑定的事件处理函数

3. （由内向外）事件冒泡：从目标元素的父元素开始，逐层向上执行每层的事件处理函数，到最外层html结束。

IE8的事件周期：2个阶段：没有捕获

3. event对象：

当事件发生时，自动创建，封装了事件信息

比如：keyCode

screenX/Y

获得event对象：

html: onclick="fun(event)"

实际调用时: event会自动获得当前事件对象

fun(e){

e中获得的就是事件对象

}

js：elem.onxxxx=fun;

fun(){

//DOM标准：自动创建event对象，默认以第一个参数传入！

//IE8：window全局的event属性，

当事件发生时，也会自动创建event对象,

但会保存在window.event中

}

event对象中包含：

1. 目标元素对象：var src=e.srcElement||e.target

2. \*\*\*取消/利用冒泡:

取消：DOM标准：e.stopPropagation()

IE8：e.cancelBubble=true;

一般用在当前事件处理函数执行结尾

\*\*\*优化：如果多个子元素中定义了相同的事件处理函数

其实，只需要在共同的父元素上定义一次即可！

3. \*取消事件：

if(e.preventDefault){

e.preventDefault(); //--DOM

}else{

e.returnValue=false; //--IE8

}

何时取消：比如:表单提交前，如果验证未通过，

则取消之后的自动提交

事件坐标：3种坐标系

1. 相对于显示器:

最大值: screen.availHeight/availWidth

鼠标位置: e.screenX/Y

2. 相对于文档显示区

最大值：window.innerHeight/Width

鼠标位置：e.clientX/x; e.clientY/y

3. 相对于父元素左上角

最大值：父元素的宽和高

鼠标位置：e.offsetX/Y

jQuery

jQuery

Day 01：

QUERY DAY01:

\* 讲师简介

\* 姓名 - 金云龙

\* 负责第三阶段课程 - JQUERY|HTML5|SERVER|AJAX

\* 从事企业级开发(JavaEE)

\* JAVASCRIPT类库

\* javascript类库,简称为JS库

\* 作用 - 为了简化javascript的开发

\* 目的 - 预定义了很多对象(属性和方法)和函数

\* 特点 - 兼容各大浏览器

\* JQUERY

\* 概述

\* jQuery其实就是一个JS文件

\* 分类

\* jQuery - Web版本(最主要)

\* jQuery UI(User Interface) - 集成UI内容

\* jQuery Mobile - 移动版本(WebApp)

\* QUnit - 用于测试

\* 版本

\* 1.4.2版本 - 企业开发使用

\* 1.8.3版本 - 企业使用最新

\* 1.11.3版本 - 目前jQuery最新

\* 2.x.x版本 - 不再支持IE浏览器(8以前)

\* 特点

\* HTML代码(结构) - 标签

\* HTML内容 - 浏览器运行页面后显示的

\* 如何使用jQuery

\* 在HTML页面中引入jQuery文件

\* 使用jQuery的选择器定位(获取)页面元素

\* 利用jQuery的API方法完成需求

\* 基本内容

\* $(selector) - jQuery的工厂函数

\* 该函数返回jQuery对象

\* jQuery的约定 - jQuery对象前增加"$"符号

\* DOM对象与jQuery对象

\* DOM对象 - 通过DOM获取的元素,称之为DOM对象

\* jQuery对象 - 通过jQuery包装DOM后产生的对象

\* jQuery对象的底层还是DOM对象

\* DOM对象与jQuery对象的转换

\* DOM对象转换为jQuery对象

\* $(DOM对象)

\* jQuery对象转换为DOM对象

\* jQuery对象是数组对象 - 角标

\* jQuery对象提供get(index)方法

\* 注意 - DOM对象与jQuery对象之间不能相互调用

\* jQuery事件

\* jQuery将DOM的事件封装成对应的方法

\*

\* 选择器 - 是jQuery的根基

\* 基本选择器

\* 层级选择器

\* 过滤选择器

\* 基本过滤选择器

\* 子元素过滤选择器

\* 内容过滤选择器

\* 可见性过滤选择器

\* 属性过滤选择器

\* 表单对象属性过滤选择器

\* 表单选择器

\* 扩展内容

\* 方法与函数的区别

\* 方法 - 定义对象时,对象的方法(出现在对象中)

\* 函数 - 函数其实也是方法

\* Web2.0

\* Web1.0 - 以内容为主的网站(门户网站)

\* Web2.0 - 以人的关系为主的网站(社交网站)

\* Web3.0 - 基于移动互联网的社交网站(微信)

\* Web前端 - 全端开发(javascript)

\* 客户端 - PC端和移动端

\* 服务器端 - NodeJS

\* 数据库端 - MySQL

\* 浏览器(内核 - javascript引擎)

\* IE - IE6/7/IE8 IE9/10/11

\* 其他浏览器(WebKit-Apple)

\* Safari浏览器

\* Chrome浏览器

\* Firefox浏览器

\* 众多国内浏览器

\* QQ浏览器 - 号称自主内核X5|V5

\* 百度浏览器 - 号称自主内核X5

\* 遨游浏览器 - 号称自主内核

\* 测试javascript代码时

\* 使用console.log()进行打印

\* error() - 输出错误

\* warn() - 输出警告

\* info() - 输出信息

\* log() - 输出日志

\* 使用alert()打印输出

\* 断点调试

\* 打断点 - 在对应代码前的行标处,鼠标点击即可

\* 断点调试 - 重新加载页面,自动停止在对应的代码中

\* 功能 - 从左到右

\* 暂停|运行功能

\* 单步跳过

\* 单步跳入

\* 单步跳出

Day 02：

回顾JQUERY第一天内容:

\* 概念

\* javascript类库 - 预定义对象和函数

\* jQuery - 实现HTML页面的内容与结构的分离

\* 如何使用jQuery

\* 拷贝jQuery的文件到指定目录

\* 在HTML页面的head标签内引入jQuery

\* 使用jQuery提供的选择器定位(获取)页面元素

\* 利用jQuery的API方法完成需求

\* 选择器 - 是jQuery的基础

\* 基本选择器

\* 层级选择器

\* 过滤选择器

\* 基本过滤选择器

\* 子元素过滤选择器

\* 属性过滤选择器

\* 可见性过滤选择器

\* 内容过滤选择器

\* 表单对象属性过滤选择器

\* 表单选择器

\* 总结

\* (使用)jQuery是目前全球使用率最高的

\* (为什么)jQuery本身的代码非常值得学习的

\* 提供学习方法(3W)

\* WHAT - 明确目的(需求)

\* HOW - 如何完成(思路)

\* WHY - 举一反三(总结)

\* DOM操作

\* 回顾原生DOM操作

\* 获取节点

\* 创建节点

\* 遍历节点

\* 替换节点

\* 删除节点

\* 插入节点

\* 复制节点

\* jQuery中的DOM操作

\* 基本操作

\* html() - 类似于原生DOM的innerHTML属性

\* 获取 - html()

\* 设置 - html(html代码)

\* val() - 类似于原生DOM的value属性

\* 获取 - val()

\* 设置 - val(value)

\* text() - 类似于原生DOM的textContent属性

\* 获取 - text()

\* 设置 - text(文本内容)

\* attr() - 获取或设置指定元素的属性

\* 获取 - attr(attrName) - 类似于getAttribute()

\* 设置 - attr(attrName,attrValue) - 类似于setAttribute()

\* removeAttr(attrName) - 类似于removeAttribute()

\* 样式操作 - css

\* attr("class",classValue)

\* addClass() - 追加样式

\* removeClass() - 删除样式

\* toggleClass() - 切换样式

\* hasClass() - 判断样式

\* css() - 操作样式

\* 设置

\* css(name,value)

\* css(options)

\* options - 选项

{

name : value,

name : value,

...

}

\* 获取 - css(name)

\* 遍历节点

\* parent() - 获取指定节点的父节点

\* children() - 获取指定节点的所有子节点

\* next() - 获取指定节点的下一个兄弟节点

\* prev() - 获取指定节点的上一个兄弟节点

\* siblings() - 获取指定节点的所有兄弟节点

\* find(expr) - 在指定节点中查找指定内容

\* 注意 - 查找指定节点的后代节点

\* 创建节点

\* 元素节点 - $(HTML代码)

\* 文本节点 - text()

\* 属性节点 - attr()

\* jQuery - $(HTML代码)

\* 删除节点

\* remove() - 删除自身及后代节点

\* empty() - (清空)删除后代节点,但保留自身节点

\* 插入节点

\* 内部插入

\* append

\* prepend

\* appendTo

\* prependTo

\* 外部插入

\* before

\* after

\* insertBefore

\* insertAfter

\* 替换节点

\* repalceWith

\* replaceAll - 就是颠倒了的repalceWith

\* 复制节点

\* jQuery - clone(boolean)

\* boolean参数 - 表示是否复制事件

\* DOM - cloneNode(boolean)

\* boolean参数 - 表示是否复制后代节点

\* 事件

\* ready() - 类似于window.onload的作用

\* 写法

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

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

\* $(function(){});

\* ready与onload的区别

\* ready

\* 具有简写方式

\* 在一个HTML页面中允许多个

\* 等待HTML页面中所有DOM结构加载完毕后就可以执行

\* onload

\* 没有简写方式

\* 在一个HTML页面中只能一个

\* 必须等待HTML页面中所有内容全部加载完毕后才能执行

\* 事件绑定与解绑

\* bind(type,data,fn) - 绑定事件

\* type - 指定绑定的事件名称

\* 如果绑定多个事件时,使用空格隔开

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

\* fn - 绑定事件的处理函数

\* unbind() - 解绑事件

\* type - 指定解绑的事件名称

\* 默认不传递任何内容 - 解绑所有事件

\* 指定单个事件名称 - 解绑单个事件

\* 指定多个事件名称 - 解绑多个事件

\* 绑定与解绑允许指定哪些事件?

blur, focus, focusin, focusout, load, resize, scroll, unload, click, dblclick, mousedown, mouseup, mousemove, mouseover, mouseout, mouseenter, mouseleave, change, select, submit, keydown, keypress, keyup, error 。

\* 注意

\* 所谓的click()事件是bind('click')的简写方式

\* 事件对象(event)

\* 事件对象被封装在事件对应的处理函数的参数

ele.onclick = function(event){

event

}

\* 常用用法

\* pageX/clientX/offsetX/x - 当前x轴

\* pageY/clientY/offsetY/y - 当前y轴

\* target - 当前绑定事件的源对象(元素)

\* returnValue - 返回值,Boolean类型

\* return false - 阻止页面的默认行为

\* which/keyCode/charCode - 键盘对应值

\* 事件冒泡

\* 捕获

\* 冒泡

\* 模拟事件(了解)

\* trigger()

\* 事件切换

\* hover(over,out)方法

\* over - onmouseover事件的处理函数

\* out - onmouseout事件的处理函数

\* 回顾内容

\* 原生DOM绑定事件三种

\* 在HTML页面元素中指定事件名称

\* 在javascript代码中

\* element.onclick = function(){}

\* element.addEventListener("click",function(){});

Day 03：

回顾JQUERY第二天内容:

\* DOM操作

\* 基本操作

\* html() - 获取或设置指定元素的HTML代码

\* text() - 获取或设置指定元素的文本内容

\* val() - 获取或设置指定元素的value属性值

\* attr() - 获取或设置指定元素的属性

\* removeAttr() - 删除指定元素的属性

\* 样式操作

\* attr("class",classValue) - 设置样式

\* 无论之前是否具有样式,重新设置

\* addClass(className) - 追加样式

\* removeClass() - 删除样式

\* 不传参 - 删除所有样式

\* 传参 - 删除指定样式

\* toggleClass() - 切换样式

\* 是在没有样式与指定样式之间切换

\* hasClass(className) - 判断是否含有指定样式

\* css() - 类似于style属性的作用

\* 获取 - css(attrName)

\* 设置

\* css(attrName,attrValue)

\* css(options)

\* options - 选项

{

attrName : attrValue,

attrName : attrValue

}

\* 创建节点

\* 根据原生DOM

\* 元素节点 - $("HTML代码")

\* 文本节点 - text()

\* 属性节点 - attr()

\* jQuery

\* $("HTML代码") - 涉及字符串的拼串

\* 删除节点

\* remove() - 删除自身及后代节点

\* empty() - (清空)删除后代节点,但保留自身节点

\* 插入节点

\* 内部插入 - 插入在指定元素内

\* append和prepend

\* appendTo和prependTo

\* 外部插入 - 插入在指定元素外

\* before和after

\* insertBefore和insertAfter

\* 替换节点

\* replaceWith()

\* 前面的元素是被替换的

\* replaceAll - 其实就是颠倒了的replaceWith

\* 复制节点

\* DOM - cloneNode(boolean)

\* boolean参数 - 是否复制后代节点

\* jQuery - clone(boolean)

\* boolean参数 - 是否复制事件

\* 事件

\* ready() - 类似于window.onload的作用

\* 写法

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

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

\* $(function(){});

\* ready与onload的区别

\* 事件绑定与解绑

\* bind(type,data,fn) - 事件绑定

\* type - 指定绑定的事件名称

\* 单事件绑定 - 指定单个事件名称

\* 多事件绑定 - 事件名称之间使用空格隔开

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

\* fn - 绑定事件的处理函数

\* unbind(type) - 事件解绑

\* type - 指定解绑的事件名称

\* 默认为空 - 默认解绑所有事件

\* 指定单事件 - 解绑单事件

\* 指定多事件 - 解绑多事件

\* 注意

\* click()其实是bind('click')的简写方式

\* 模拟事件 - trigger()

\* 事件切换

\* hover(over,out) - 模拟鼠标悬停事件

\* 效果上与mouseover+mouseout一致

\* 扩展内容

\* 字符串(string)

\* 定义字符串的方式

\* var str1 = ""|'';

\* var str2 = new String();

\* ""和''的含义是一样的

\* 其他开发语言中,""和''的含义不一样

\* "" - 字符串

\* '' - 字符

\* length属性 - 获取当前字符串包含字符的个数

\* 动画效果

\* 显示与隐藏

\* show() - 显示

\* 无参版本 - 不具有动画效果

\* show(speed,callback)有参版本 - 具有动画效果

\* speed - 设置动画执行的时长,单位为毫秒

\* 三个预定义值 - slow|normal|fast

\* callback - 当动画执行完毕后执行的函数

\* hide() - 隐藏

\* 无参版本 - 不具有动画效果

\* hide(speed,callback)有参版本 - 具有动画效果

\* speed - 设置动画执行的时长,单位为毫秒

\* 三个预定义值 - slow|normal|fast

\* callback - 当动画执行完毕后执行的函数

\* 滑动效果

\* slideUp() - 向上滑动

\* 可以不传递参数 - 底层具有默认时长

\* 执行的效果依然具有动画效果

\* 参数

\* speed - 设置动画执行的时长,单位为毫秒

\* 三个预定义值 - slow|normal|fast

\* callback - 当动画执行完毕后执行的函数

\* slideDown() - 向下滑动

\* 可以不传递参数 - 底层具有默认时长

\* 执行的效果依然具有动画效果

\* 参数

\* speed - 设置动画执行的时长,单位为毫秒

\* 三个预定义值 - slow|normal|fast

\* callback - 当动画执行完毕后执行的函数

\* 淡入淡出

\* fadeIn() - 淡入

\* 可以不传递参数 - 底层具有默认时长

\* 执行的效果依然具有动画效果

\* 参数

\* speed - 设置动画执行的时长,单位为毫秒

\* 三个预定义值 - slow|normal|fast

\* callback - 当动画执行完毕后执行的函数

\* fadeOut() - 淡出

\* 可以不传递参数 - 底层具有默认时长

\* 执行的效果依然具有动画效果

\* 参数

\* speed - 设置动画执行的时长,单位为毫秒

\* 三个预定义值 - slow|normal|fast

\* callback - 当动画执行完毕后执行的函数

\* fadeTo() - 将当前元素从透明度一个值到另一个值

\* 自定义动画

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

\* params - 设置动画样式(CSS属性)

\* duration - 设置动画执行时长,单位为毫秒

\* callback - 动画执行完毕后的函数

\* animate(params,options)

\* 并发与排队效果

\* 并发效果 - 多个动画效果同时执行

animate({

attrName1 : attrValue1,

attrName2 : attrValue2,

...

},time);

aniamte({attrName:attrValue},{duration:time})

.aniamte({attrName:attrValue},{

duration : time,

queue : false

});

\* 排队效果 - 多个动画效果按照先后顺序执行

animate({attrName:attrValue},time).animate({attrName:attrValue},time).animate({attrName:attrValue},time)...

\* 切换动画

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

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

\* 类(似)数组操作 - jQuery对象是数组对象

\* 查看jQuery底层代码 - 证明jQuery是数组对象

var jQuery = function( selector, context ) {

return new jQuery.fn.init( selector, context );

}

var init = jQuery.fn.init = function(selector,context){

return jQuery.makeArray( selector, this );

}

var makeArray = function(arr,results){

var ret = [];

return ret;

}

window.jQuery = window.$ = jQuery;

\* 属性

\* length - 获取当前数组的长度(当前jQuery对象包含多少个DOM对象)

\* 方法

\* get(index) - 根据角标返回DOM对象

\* eq(index) - 根据角标返回DOM对象

:eq()选择器的作用一致

\* index(obj) - 根据DOM对象返回对应角标

\* 隐式迭代

\* 概念

\* 隐式迭代 - jQuery的遍历

\* 在遍历的过程中,只关注入口和出口

var arr = [];// 数组就是入口

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

var ele = arr[i];// 数组的每个元素就是出口

}

\* 显式迭代 - 例如for循环

\* 在遍历的过程中,从开始到结束控制遍历的所有过程

\* jQuery的遍历

\* $().each(callback) - 对象方法

\* $.each(obj,callback) - 全局函数

\* 支持插件

\* 作用 - 为了扩展或简化jQuery的开发

\* 特点 - 数量多

\* 地址 - http://plugins.jquery.com

\* 目的

\* 掌握jQuery插件的使用方法

\* 掌握如何学习jQuery插件的使用

\* 日历插件 - My97DatePicker(国产)

\* 目录说明

\* lang - 存储国际化(i18n)文件

\* 本地化(i10n)

\* skin - 皮肤(日历控件长什么样子)

\* calendar.js - 日期库主文件

\* WdatePicker.js - 配置文件

\* 如何使用My97日历插件

\* 将My97DatePicker文件夹整体拷贝到工程目录中

\* 注意 - 不能随意修改My97DatePicker目录中任何内容

\* 使用My97日历插件的步骤

\* 拷贝My97DatePicker文件夹

\* 在HTML页面引入文件

\* 引入jQuery文件

\* 引入插件文件

\* 在HTML页面定义容器(元素)

\* 作用 - 用于在页面中显示的效果

\* 在javascript代码使用插件提供的API方法

\* 表单验证插件

\*

\* 扩展内容

\* Web前端

\* 页面设计师 - UI设计师

\* Web前端 - 逻辑(页面特效)

\* this的用法

\* 在具有上下文环境中使用this

\* this指代上下文环境

\* 在javascript创建构造器时使用this

\* this指代new之后的对象

\* 在jQuery中的隐式迭代中使用this

\* this指代DOM对象(页面元素)

\* 注意

\* this的用法是javascript整个内容中最难的部分

\* 建议

\* 会就用,不会就别用

Day 04：

回顾JQUERY第三天内容:

\* 动画效果

\* 显示与隐藏

\* show() - 显示

\* 无动画版本 - show()

\* 有动画版本 - show(speed,callback)

\* speed - 设置动画执行的时长,单位为毫秒

\* callback - 动画执行完毕后调用的函数

\* hide() - 隐藏

\* 滑动效果

\* slideUp(speed,callback)

\* speed - 设置动画执行的时长,单位为毫秒

\* callback - 动画执行完毕后调用的函数

\* slideDown()

\* 淡入淡出

\* fadeIn(speed,callback) - 淡入

\* speed - 设置动画执行的时长,单位为毫秒

\* callback - 动画执行完毕后调用的函数

\* fadeOut() - 淡出

\* fadeTo(speed,opty,callback)

\* speed - 设置动画执行的时长,单位为毫秒

\* opty - 设置透明度(0-1)

\* 切换效果

\* toggle(speed,callback)

\* slideToggle(speed,callback)

\* 自定义动画

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

\* params - 设置自定义动画的样式(CSS属性)

\* duration - 设置自定义动画的时长

\* callback - 自定义动画执行完毕后的函数

\* aniamte(params,options)

\* params - 设置自定义动画的样式(CSS属性)

\* options - 选项

\* duration - 设置自定义动画的时长

\* queue - Boolean值,如果false的话,该动画不会被加入在动画队列中

\* 并发与排队效果

\* 并发效果 - 多个动画同时执行

animate({

attrName1 : attrValue1,

attrName2 : attrValue2

},time)

animate({attrName1 : attrValue1},{duration : time})

.animate({attrName1 : attrValue1},{

duration : time,

queue : false

})

\* 排队效果 - 多个动画按照先后顺序执行

\* 类数组操作

\* 查看了jQuery的底层代码 - jQuery对象是数组对象

\* 属性

\* length属性 - 获取jQuery对象中包含DOM对象的个数

\* 方法

\* get(index) - 根据索引值返回对应的DOM对象

\* eq(index) - 根据索引值返回对象的DOM对象

\* index(obj) - 根据DOM对象返回对应的索引值

\* 隐式迭代

\* $().each(callback) - 对象方法

\* callback - function(index,domEle){}

\* index - 遍历过程中的索引值

\* domEle - 遍历之后的结果(DOM对象)

\* $.each(obj,callback) - 全局函数

\* obj - 需要遍历的对象或数组

\* callback - function(index,domEle){}

\* index - 遍历过程中的索引值

\* domEle - 遍历之后的结果(DOM对象)

\* jQuery插件

\* 概念

\* 基于jQuery进行扩展或简化(JS文件)

\* 使用

\* 在HTML页面中引入外部文件

\* 引入jQuery文件

\* 引入jQuery插件文件

\* 在HTML页面中定义容器(元素)

\* 在javascript代码中使用插件提供的API方法

\* 插件

\* My97日期插件

\* 表单验证插件

\* jQuery插件

\* 如何学习(研究)jQuery插件

\* 提供插件的帮助文档

\* 提供demo(示例)代码

\* 表单验证插件

\* 树菜单插件

\* jQuery UI - 集成页面特效

\* 开发插件 - 原理

\* 对象方法插件 - $().each()

\* $.fn.extend()

\* 全局函数插件 - $.each()

\* $.extend()

\* 选择器插件 - 目前几乎不用

\* 这一类插件扩展jQuery选择器的用法

\* jQuery官方推出专门用于扩展选择器的JS库

\* 举例 - XPath插件 - 专门用于解析XML文件

\* 目的

\* 将来实际开发需要独自开发插件(入门级)

\* 为了面试

HTML**5**

HTML5 DAY01:

\* 基本内容

\* HTML5目前最新的规范(标准)是2014年10月推出

\* 2005年左右出现HTML5版本(非标准)

\* W3C组织(两个组织定义H5规范)

\* 学习(研究)HTML5是学习未来(将来主流)

\* HTML版本 - 第一阶段主要学习还是4版本(包含5版本)

<header><nav>

\* HTML5版本之后,声明不再出现版本信息

\* 有意地版本,以后可能不再会有新版本

\* HTML5的规范内容实时更新

\* 注意

\* HTML5永远都不可能离开javascript.

\* HTML5在移动端支持好于PC端

\* HTML5新表单

\* input新类型

\* email类型 - 验证是否包含"@"

\* url类型 - 验证是否包含"http://"

\* tel类型 - 只在移动端显示

\* number类型

\* range类型

\* date类型

\* color类型

\* 表单新元素

\* datalist元素

\* progress元素

\* meter元素

\* output元素

\* 表单新属性

\* placeholder - 提供默认提示内容

\* multiple - 允许输入多个值

\* 多个值之间使用","

\* autofocus - 自动获取焦点

\* form - 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

\* 验证当前元素值是否为空

\* pattern属性

\* 使用正则表达式验证当前元素值是否匹配

\* 注意 - 并不能验证当前元素值是否为空

\* min和max属性

\* 验证当前元素值最小值或最大值

\* 一般适用于number、range等元素

\* minlength和maxlength属性

\* minlength - 验证当前元素值的最小长度

\* 输入值时,允许输入小于指定值

\* 提交表单时,验证元素值最小长度

\* 注意

\* minlength并不是HTML5新属性

\* maxlength - 验证当前元素值的最大长度

\* 输入值时,长度不能大于指定值

\* validity属性

\* 表单验证HTML5提供一种有效状态

\* 有效状态通过validityState对象获取到

\* validityState对象可通过validity属性得到

\* 验证(有效)状态

\* validityState对象提供了一系列的有效状态

\* 通过这些有效状态,进行判断

\* 注意

\* 所有验证状态必须配合上述的验证属性使用

\* 作用

\* 用来替换原本手工实现的逻辑

\* 验证状态

\* valid

\* 作用 - 判断当前元素值是否正确

\* 注意

\* 该状态返回true,表示验证成功

\* 该状态返回false,表示验证失败

\* valueMissing

\* 作用 - 判断当前元素值是否为空

\* 用法 - 配合required属性使用

\* typeMismatch

\* 作用 - 判断当前元素值的类型是否匹配

\* 用法 - 配合email、number、url等

\* patternMismatch

\* 作用 - 判断当前元素值是否与指定正则表达式匹配

\* 用法 - 配合pattern属性使用

\* tooLong

\* 作用 - 判断当前元素值的长度是否正确

\* 用法 - 配合maxlength属性

\* 注意

\* 使用maxlength属性后,实际不可能出现长度大于maxlength的值

\* tooLong很难出现这种情况

\* rangeUnderflow

\* 作用 - 判断当前元素值是否小于min属性值

\* 用法 - 配合min属性

\* 注意 - 并不能与max属性值进行比较

\* stepMismatch

\* 作用 - 判断当前元素值是否与step设置相符

\* 用法 - 配合step使用

\* 注意 - 并不能与min或max属性值进行比较

\* customError

\* 用法 - 配合setCustomValidity()方法

\* 如果使用该方法,该状态返回true

\* setCustomValidity()方法

\* 作用 - 设置自定义的错误提示内容

\* 问题 - 使用该方法设置错误信息

\* 当输入正确时,调用该方法将信息设置为空("")

\* 不能使用上述有效状态的任何一种判断输入是否正确(所有状态返回false)

\* 扩展内容

\* 代码的编写 - 逻辑的完整性

\* 判断value值是否为空

value==""||value==null

\* tooLong状态

Day 02:

回顾HTML5第一天内容:

\* HTML5新表单

\* input新类型

\* email类型

\* url类型

\* tel类型

\* date类型

\* number类型

\* range类型

\* 表单新元素

\* datalist元素 - 类似于select元素

\* datalist元素 - 预定义数据内容(option)

\* 使用<input>元素的list属性

\* progress元素 - 进度条

\* max - 设置最大值

\* value - 设置当前进度

\* meter元素 - 刻度

\* min - 设置最小值

\* max - 设置最大值

\* value - 设置当前值

\* low - 设置低预警值

\* high - 设置高预警值

\* output元素 - 输出

\* 表单新属性

\* placeholder - 给出默认提示

\* autofocus - 自动获取焦点

\* multiple - 允许输入多个值,多个值之间使用","

\* form - 允许表单元素定义在表单外

\* 表单新验证

\* 验证属性

\* required - 验证是否为空

\* pattern - 正则表达式是否匹配

\* min和max - 验证最大/小值

\* minlength和maxlength

\* minlength - 并不是H5的新特性

\* maxlength - 不允许输入超过maxlength长度内容

\* 有效状态

\* valid - 验证是否成功

\* true - 表示验证成功

\* valueMissing - 验证是否为空(true)

\* patternMismatch - 验证正则是否匹配

\* typeMismatch - 验证类型是否匹配

\* rangeUnderflow - 验证是否小于min

\* stepMismatch - 验证是否与step匹配

\* customError - setCustomValidity()方法

\* 音视频处理

\* 视频处理

\* 基本内容

\* 使用Flash技术处理HTML页面中的视频内容

\* 包含音频、动画、网页游戏等

\* 特点

\* 浏览器原生不支持(IE浏览器要求安装ActiveX组件)

\* 性能不好(不能过多地使用)

\* 智能移动端并不支持Flash技术

\* 命运

\* Flash的母公司Adobe公开宣布放弃

\* 目前用于替代Flash技术最好的选择 - HTML5

\* 几乎所有浏览器原生支持<video>元素

\* 性能更高

\* 智能移动端支持非常好

\* 如何实现视频处理

\* <video>元素

\* 如果当前浏览器不支持<video>元素

\* 在<video>元素内编写提示内容

\* 属性

\* src - 引入视频文件的路径

\* autoplay - 自动播放视频

\* 使用<source>元素

<video>

<source src="一种视频格式" />

<source src="一种视频格式" />

<source src="一种视频格式" />

</video>

\* <video>支持的视频格式

\* MP4格式 - 目前比较主流

\* OGG格式 - 多用于移动端

\* WebM格式 - 目前唯一支持超高清格式

\* 在HTML页面中支持超高清格式(HTML5)

\* 由Google公司推出的

\* <video>元素的属性

\* src - 引入视频文件的路径

\* autoplay - 自动播放视频

\* controls - 提供控制面板

\* loop - 表示循环播放

\* poster - 设置播放之前显示的图片

\* width和height - 设置显示视频的宽度和高度

\* preload - 预加载

\* auto - (默认值)自动加载

\* none - 不加载

\* metadata - 只加载视频的基本信息(不含视频)

\* 高级内容

\* 方法

\* play() - 播放视频

\* pause() - 暂停视频

\* load() - 加载视频

\* canPlayType() - 判断当前浏览器是否支持指定视频格式

\* 事件

\* play - 当视频播放时被触发

\* pause - 当视频暂停时被触发

\* ended - 当视频结束时被触发

\* error - 当视频错误时被触发

\* canplay - 不考虑整体情况下,只要能播放,就播放

\* canplaythrough - 考虑整体情况下,只要能播放,就播放

\* progress - 表示视频加载的进度

\* 属性 - 用于判断

\* paused - 表示判断当前视频是否暂停

\* 返回Boolean值,true表示暂停,false表示播放

\* ended - 表示判断当前视频是否播放完毕

\* 返回Boolean值,true表示完毕

\* duration - 表示当前视频的时长

\* currentTime - 表示当前视频播放的位置

\* 音频处理

\*

\* 扩展内容

\* Web前端 - 移动端

\* 移动智能终端

\* iPhone - IOS系统 - Object-c|Swfit

\* Android系统 - Java

\* Windows Mobile系统 - .net平台

\* BlackBerry系统 - 企业级应用 QNX

\* WebOS系统 - 全键盘+触摸屏

\* 塞班 - 诺基亚

\* MeeGo - Inter和诺基亚

\* 移动跨平台 - HTML|CSS|JAVASCRIPT

\* 一次编写,到处运行(phoneGap)

\* 游戏分类(画面)

\* 2D效果

\* 2.5D效果 - 实际还是2D技术,模拟3D效果

\* 3D效果

\* 作为前端Web开发人员的原则

\* 眼见为实 - 不适用我们前端人员

\* HTML5的难度

\* HTML5的核心API相对来讲,都不难

\* HTML5的案例真正难在逻辑(JS代码)

\* 总结 - HTML5离不开javascript

\* 回答疑问

\* 第二阶段的javascript,抓紧复习

\* 个人并不建议 - 留级

\* 编写简历 - 项目经验

\* 多做网上的案例

\* 建议买哪些书籍

\* 个人不建议任何书籍

\* 如何成为优秀的程序员

\* 自学能力很强

\* 网上的案例

\* 网上的资料

\* 善于归纳总结

\* 师傅领进门,修行在个人

\* 英文对于开发功能的影响

\* 英文 - 书面+口语

\* 口语 - 外企+合资

\* 计算机英文

\* class - 类|教室

\* bug - 错误|臭虫

\* ant - 版本|蚂蚁

Day03:

回顾HTML5第二天内容:

\* 音视频处理

\* 视频处理

\* 基本内容

\* 目前主流HTML页面视频处理 - Flash技术

\* HTML5提供的视频处理 - <video>元素

\* 使用<video>

\* 提供一种视频格式

<video src="引入视频文件的路径">

不好意思,你的浏览器不支持video元素.

</video>

\* 提供多种视频格式

<video>

<source src="引入视频文件的路径" />

<source src="引入视频文件的路径" />

<source src="引入视频文件的路径" />

</video>

\* <video>元素的属性

\* src - 引入视频文件的路径

\* width和height - 设置当前视频的宽度和高度

\* autoplay - 自动播放

\* controls - 提供控制面板

\* loop - 循环播放

\* poster - 在播放视频前,显示一张图片

\* preload - 预加载

\* auto - 默认值,尽快地加载视频

\* none - 不加载(缓存)

\* metadata - 只加载除视频以外的基本信息

\* 高级内容

\* 事件

\* play - 当视频播放时被触发

\* pause - 当视频暂停时被触发

\* ended - 当视频结束播放时被触发

\* 自动播放下一个视频功能

\* 方法

\* play() - 播放视频

\* pause() - 暂停视频

\* 属性

\* paused - 判断视频是否暂停

\* Boolean值,true表示暂停,否则播放

\* ended - 判断视频是否结束

\* duration - 返回当前视频的时长

\* currentTime - 返回视频播放的当前位置

\* 音频处理

\* <audio>元素

\* 提供一种音频格式

<audio src="" controls></audio>

\* 提供多种音频格式

<audio>

<source src="" />

<source src="" />

<source src="" />

</audio>

\* <audio>元素的属性

\* autoplay

\* loop

\* controls

\* 高级内容

\* 事件

\* 方法

\* 属性

\* Canvas画布

\* 基本内容

\* Canvas的作用 - 为HTML页面提供画布的功能

\* Canvas的效果 - 在画布中绘制图形

\* 注意 - 绘制的图形与HTML页面无关

\* 扩展内容 - 所见非所得(眼见不为实)

\* 使用canvas的步骤

\* 在HTML页面定义<canvas>元素

\* 在javascript代码中

\* 获取<canvas>元素

\* 使用width和height属性 - 没有任何问题

\* 使用CSS样式

\* 通过style属性设置 - 绘制图形被拉伸

\* 通过style元素设置 - 按比例放大

\* 创建画布对象 - getContext("2d")

\* 该方法返回2D和3D效果两种画布对象

\* 注意

\* 参数必须是"2d"或"3d"

\* 目前只绘制2d效果

\* 使用Canvas提供的API

\* 绘制图形

\* 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

\* clearRect(x,y,width,height)

\* 清除指定区域的矩形

\* 设置颜色

\* fillStyle - 设置填充颜色

\* strokeStyle - 设置描边颜色

\* globalAlpha - 设置透明度(0-1)

\* 渐变效果

\* 线型渐变

\* createLinearGradient(x1,y1,x2,y2)

\* x1和y1 - 基准线的起点坐标值

\* x2和y2 - 基准线的终点坐标值

\* 基准线

\* 扇形(射线)渐变

\* createRadialGradient(x1,y1,r1,x2,y2,r2)

\* x1,y1 - 第一个基准圆的元素坐标值

\* r1 - 第一个基准圆的半径

\* x2,y2 - 第二个基准圆的元素坐标值

\* r2 - 第二个基准圆的半径

\* 两个基准圆

\* 设置渐变颜色

\* addColorStop(position,color)

\* position - 设置渐变颜色的位置(0-1)

\* color - 设置渐变颜色

\* Canvas

\* 绘制图形

\* 绘制文字

\* 方法

\* fillText(text,x,y) - 实心文字

\* text - 绘制的文字内容

\* x和y - 绘制的坐标值

\* strokeText(text,x,y) - 空心文字

\* 属性

\* font - 类似于CSS中的font属性

\* textAlign - 设置文字的水平方向对齐

\* left - 左对齐

\* center - 水平居中

\* right - 右对齐

\* textBaseline - 设置文字的垂直方向对齐

\* top - 顶部对齐

\* middle - (垂直)居中对齐

\* bottom - 底部对齐

\* hanging - 悬挂基线

\* alphabetic - 字母基线

\* 注意

\* 无论是水平方向还是垂直方向对齐,基准线对齐,并不是文字对齐

\* 无论是水平方向还是垂直方向对齐,并不是必要的属性(不使用也是可以的)

\* 阴影效果

\* shadowColor - 设置阴影颜色

\* shadowOffsetX - 设置水平方向阴影

\* shadowOffsetY - 设置垂直方向阴影

\* shadowBlur - 设置阴影的模糊程度

\* 创建路径

\* (标识)方法

\* beginPath() - 表示开始创建路径

\* closePath() - 表示结束创建路径

\* 设置方法

\* rect(x,y,width,height) - 设置矩形形状

\* x和y - 设置矩形的左上角坐标值

\* width和height - 设置矩形的宽度和高度

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

\* x和y - 设置圆形的圆心坐标值

\* radius - 设置圆形的半径

\* startAngle和endAngle - 设置圆形的起始位置

\* direction - 按照顺时针或逆时针绘制

\* 绘制方法

\* stroke() - 绘制轮廓

\* fill() - 绘制填充

\* 绘制线条(直线和折线、多边形) - 创建路径

\* moveTo(x,y) - 设置这条线的起点坐标值

\* lineTo(x,y) - 设置这条线的终点(折点)坐标值

\*

\*

\* 作业

\* 位置、大小、颜色都随机的圆形

\* 设置线条

\* lineWidth - 设置线条的宽度

\* 默认值为1(px)

\* lineCap - 设置线条端点的形状

\* butt - 默认值,平直

\* round - 圆角

\* square - 正方向

\* lineJoin - 设置两条线焦点的形状

\* miter - 默认值,尖角

\* round - 圆角

\* bevel - 斜角

\* miterLimit - 配合lineJoin使用

\* lineJoin设置为miter,该属性值设置尖角的延伸范围

\* Canvas处理图片

\* 绘制图片

\* drawImage(img,x,y) - 按照图片原大小加载

\* img - 当前加载(绘制)的图片

\* x和y - 绘制图片的坐标值(左上角)

\* drawImage(img,x,y,width,height) - 按照指定大小加载图片

\* img - 当前加载(绘制)的图片

\* x和y - 绘制图片的坐标值(左上角)

\* width和height - 设置绘制图片显示的宽度和高度

\* 注意

\* 必须保证图片加载完毕(onload事件)后,再绘制图片

\* 平铺图片

\* createPattern(img,type)

\* img - 平铺的图片

\* type - 平铺的方式

\* repeat - 平铺

\* no-repeat - 不平铺

\* repeat-x - 水平方向平铺

\* repeat-y - 垂直方向平铺

\* 注意

\* 必须保证图片加载完毕(onload事件)后,再绘制图片

\* 切割图片

\* clip() - 切割(按照创建路径使用)

\* 画布方法

\* scale(x,y) - 缩放(缩小或放大)

\* x - 表示水平方向的缩放

\* y - 表示垂直方向的缩放

\* 参数的取值

\* 如果为1的话,表示不缩放(原大小)

\* 如果小于1的话,表示缩小

\* 如果大于1的话,表示放大

\* translate(x,y) - 重新定位(x,y)

\* x和y - 新的坐标值

\* 注意 - x和y是相对于上次定位坐标值

\* rotate(旋转角度) - 旋转画布

\* 公式为 degrees \* Math.PI / 180;

\* Chart.js - Canvas的JS库

\* 作用 - 提供各种图表

\* 如何使用

\* 在HTML页面中引入Chart.js文件

\* 在HTML页面中定义<canvas>元素

\* 在javascript代码中

\* 获取<canvas>元素

\* 创建画布对象

var context = canvas.getContext("2d");

\* 通过画布对象,创建Chart对象

var chart = new Chart(context);

\* 利用Chart对象调用API方法

var data = [];

chart.Pie(data);

\* 提供6种图表

\* 柱状图 - Bar(data,options)

\* 饼状图 - Pie(data,options)

\* 曲线图 - Line(data,options)

\* 环形图 - Doughnut(data,options)

\* 雷达图 - Radar(data,options)

\* 极地区域图 - PolarArea(data,options)

\* 作业

\* 位置、大小、颜色都随机的圆形

\* 将Chart.js其他4种图表实现一遍

\* 扩展内容

\* 前端Web开发时,建议使用以下几种浏览器

\* Chrome - 两个版本 - 版本更新过于频繁

\* 开发版本

\* 功能较新

\* 不稳定

\* 正式版本

\* 功能较旧

\* 稳定

\* Chrome OS

\* Firefox - 两个版本

\* 开发版本 - (开发)功能多

\* 正式版本

\* FireFox OS - 移动操作系统

\* 开发使用HTML|CSS|JAVASCRIPT

\* Safari

\* 内向问题

\* 面试 - 你认为你的性格,是外向还是内向?

\* 内向 - 沟通能力比较差,团队协作出问题

\* 外向 - 担心工作时,不够专心

\* 良好的沟通能力

\* 程序员 - 形成程序员思维

\* 都是学习和培养

\* Markdown编辑器

\* 支持Markdown文件的扩展名为".md"

\* 前面的内容忘记

\* 个人建议 - 在案例中复习

Day04:

回顾HTML5第三天内容:

\* Canvas

\* 绘制文字

\* 方法

\* fillText(text,x,y) - 绘制实心文字

\* text - 绘制的文本内容

\* x和y - 绘制文字的坐标值

\* strokeText(text,x,y) - 绘制空心文字

\* 属性

\* font - 类似于CSS中的font属性的作用

\* textAlgin - 水平方向对齐(基准线)

\* left

\* center

\* right

\* textBaseline - 垂直方向对齐(基准线)

\* top

\* middle

\* bottom

\* 阴影效果

\* shadowColor - 设置阴影的颜色

\* shadowOffsetX - 设置阴影的水平方向

\* 0 - 没有阴影

\* 负值 - 阴影向左

\* 正值 - 阴影向右

\* shadowOffsetY - 设置阴影的垂直方向

\* 0 - 没有阴影

\* 负值 - 阴影向上

\* 正值 - 阴影向下

\* shadowBlur - 设置阴影的程度

\* 创建路径

\* 标识方法

\* beginPath() - 标识开始创建路径(必要)

\* closePath() - 标识结束创建路径

\* 绘制不完整的圆形时,效果不同

\* 设置方法

\* rect(x,y,width,height)

\* arc(x,y,r,startAngle,endAngle,direction)

\* 绘制方法

\* fill()

\* stroke()

\* 绘制线条(直线、折线和多边形)

\* moveTo(x,y) - 设置起点坐标值

\* lineTo(x,y) - 设置终点(折点)坐标值

\* 设置线条

\* lineWidth - 设置线条宽度,默认值为1

\* lineCap - 设置线条端点的形状

\* round - 圆角

\* butt - 默认值,平直

\* lineJoin - 设置两个线交点的形状

\* round - 圆角

\* miter - 尖角,默认值

\* bevel - 斜角

\* miterLimit - 配合lineJoin=miter使用

\* 绘制图片

\* 绘制图片

\* drawImage(img,x,y)

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

\* img - 加载的图片

\* x和y - 绘制图片的坐标值

\* width和height - 绘制图片的宽度和高度

\* 注意 - 加载图片后,绑定onload事件

\* 平铺图片

var ptn = context.createPattern(img,type);

\* img - 平铺的图片

\* type - 平铺方式

\* repeat - 全方向平铺

\* repeat-x - x轴平铺

\* repeat-y - y轴平铺

\* no-repeat - 不平铺

context.fillStyle = ptn;

context.fillRect();

\* 注意 - 上述代码编写onload事件内

\* 切割图片

context.beginPath();

context.rect()或arc()

context.clip();

\* 画布方法

\* scale(x,y) - 缩放

\* x - 表示水平方向的缩放

\* y - 表示垂直方向的缩放

\* translate(x,y) - 平移

\* x和y - 相对值(上次定位坐标值)

\* rotate(旋转角度) - 旋转画布

\* Chart.js

\* 作用 - 绘制图表

\* 6种图表

\* 柱状图

\* 饼状图

\* 曲线图

\* 环形图

\* 极地区域图

\* 雷达图

\* SVG

\* 基本内容

\* SVG并不属于HTML5专有内容

\* HTML5提供有关SVG原生的内容

\* 在HTML5出现之前,就有SVG内容

\* SVG,简单来说就是矢量图

\* SVG文件的扩展名为".svg"

\* SVG使用的是XML语法

\* 概念

\* SVG是一种使用XML技术描述二维图形的语言

\* SVG的特点

\* SVG绘制图形可以被搜索引擎抓取

\* SVG在图片质量不下降的情况下,被放大

\* SVG与Canvas的区别

\* SVG

\* 不依赖分辨率

\* 支持事件绑定

\* 大型渲染区域的程序(例如百度地图)

\* 不能用来实现网页游戏

\* Canvas

\* 依赖分辨率

\* 不支持事件绑定

\* 最合适网页游戏

\* 保存为".jpg"格式的图片

\* 用途

\* 网页中一些小的图标

\* 网页中动态特效(动画效果)

\* 绘制图形

\* 矩形元素

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

\* 圆形元素

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

\* 椭圆元素

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

\* 直线元素

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

\* 折线元素

<polyline points="">

\* 多边形元素

<polygon points="" />

\* 特效元素

\* 渐变

\* 滤镜 - 高斯模糊

\* \* TWO.js

\* 基本内容

\* JS库介绍

\* three.js - 专门用于绘制三维图形

\* two.js - 专门用于绘制二维图形

\* two.js支持的格式

\* SVG - 默认

\* Canvas

\* WebGL - 专门用于绘制图像

\* 如何使用two.js

\* 在HTML页面中引入two.js文件

\* 在HTML页面中定义容器(<div>)

\* 在javascript代码中

\* 获取HTML页面中的容器

\* 创建Two对象,将该对象添加到容器中

new Two(params).appendTo(Element);

\* 使用two.js提供的API方法进行绘制

\* 利用two.js提供的方法,设置图形

\* 利用update()方法进行绘制

\* 创建Two对象

\* 构造器 - new Two(params)

\* params参数 - 设置当前对象的信息

\* type - 设置当前使用的格式(Two.Types.svg)

\* svg - 默认值

\* canvas

\* webgl

\* width和height - 设置宽度和高度

\* fullscreen - 设置是否全屏

\* Boolean值,true表示全屏

\* 图形方法

\* makeLine() - 绘制线条

\* makeRectangle() - 绘制矩形

\* makeCircle() - 绘制圆形

\* makeEllipse() - 绘制椭圆

\* 动画方法

\* update() - 更新动画

\* play() - 添加动画(循环)

\* pause() - 删除动画

\* 设置绘制图形的样式

\* 调用Two对象的绘制方法绘制图形时,返回该图形对象

\* 通过该图形对象,设置相关属性值

\* 分组操作

\* Two.Group

\* 动画效果

\* bind(event,callback)方法 - 事件绑定

\* event - 绑定事件名称

\* update - 对应update()方法的作用

\* 所有的DOM事件都可以绑定

\* callback - 事件处理函数

\* 扩展内容

\* 前端开发工具

\* Aptana Studio 3 - 代码提示

\* Webstrom - 国内前端开发人员神器

\* 实际开发中

\* 多用SVG

\* 不失真

\* 可被搜索

\* 页面优化 - 减少外部链接

\* <a href="">

\* <img src="">

\* Canvas - HTML绘制图形

\* 实际运行中,是以图片方式出现(.png)

\* 不能被搜索引擎抓取

\* 放大后失真

\* SVG内容

\* 内容量非常大

\* 静态绘制图形

\* 动态动画效果

\* 专门提供事件

\* 网上关于SVG的资料很少(没有书籍)

\* SVG的规范(W3C英文)

\* 作业(思考)

Day 05:

回顾HTML5第四天内容:

\* SVG

\* 基本内容

\* 概念 - SVG是一种使用XML技术描述二维图形的语言

\* 支持的格式

\* 矢量图

\* 图片

\* 文本

\* SVG文件的扩展名为".svg"

\* SVG与Canvas的区别

\* SVG

\* 不依赖分辨率

\* 可被搜索、压缩或脚本化

\* 适用于大型渲染区域的程序(百度地图等)

\* 不适用于网页游戏

\* Canvas

\* 依赖分辨率

\* 可以以图片形式存在(.png)

\* 适用于网页游戏

\* 不支持事件绑定

\* 设置样式

\* fill - 填充颜色

\* stroke - 描边颜色

\* stroke-width - 描边宽度

\* 绘制图形

\* 绘制矩形 - <rect />

\* 绘制圆形 - <circle />

\* 绘制椭圆 - <ellpise />

\* 绘制直线 - <line />

\* 绘制折线 - <polyline />

\* 绘制多边形 - <polygon />

\* 特效元素

\* 渐变

\* 高斯模糊

\* two.js

\* 基本内容

\* 该JS库支持三种格式

\* svg

\* canvas

\* webgl

\* 特点 - 具有SVG解释器

\* 如何使用

\* HTML页面

\* 引入two.js文件

\* 在页面中定义容器(元素)

\* <svg></svg>

\* <canvas></canvas>

\* <div></div>(推荐)

\* javascript代码

\* 获取页面中的容器(元素)

\* 创建Two对象,添加到容器中

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

\* 设置要绘制的图形(API)

\* 绘制方法 - update()

\* 设置样式

\* fill

\* stroke

\* linewidth

\* 绘制方法

\* 绘制矩形

\* 绘制圆形

\* 分组操作

var group = two.makeGroup(objects);

\* 动画效果

\* update() - 更新

\* play() - 添加动画(循环)

\* pause() - 删除动画

\* Geolocation(地理定位)

\* 基本内容

\* 地理定位 - 地球的经度和纬度的相交点

\* 实现地理定位的方式

\* GPS - 美国的,依靠卫星定位

\* 北斗定位 - 纯国产,惯性定位技术和卫星定位

\* 基站定位 - 移动运营商创建基站(提供信号源)

\* 基于互联网 - IP地址(PC端和移动端)

\* 目前很多浏览器都具有定位功能

\* HTML5中地理定位

\* 地理定位功能并不是属于HTML5专有内容

\* HTML5的地理定位技术,由Google公司提供的

\* Google Map产品

\* 中国 - 国内不能使用Google公司所有服务和产品

\* 百度地图和高德地图

\* 百度地图

\* 百度地图 - http://developer.baidu.com/map/

\* 注册百度开发者账户

\* http://developer.baidu.com/

\* 条件 - 必须能连接互联网

\* 目的 - 掌握如何使用百度地图

\* JS库或百度地图

\* 提供的API帮助文档

\* 提供的Demo示例代码

\* 学习目的

\* 学习的百度地图的功能

\* 百度地图的使用特点

\* 如何使用百度地图

\* 在HTML页面中

\* 引入百度地图的JS

http://api.map.baidu.com/api?v=2.0&ak=秘钥

\* 定义显示地图的容器

<div id="" style=""></div>

\* 在javascript代码中

\* 创建百度地图Map对象

var map = new BMap.Map(容器id);

\* 进行地图的初始化

map.centerAndZoom();

\* 百度地图的组件

\* 核心类 - Map类

\* 构造器 - BMap.Map(容器id);

\* 方法

\* centerAndZoom() - 初始化方法

\* addControl() - 添加控件

\* addOverlay() - 添加标注

\* Control类 - 控件类

\* ScaleControl类 - 表示地图的比例尺

\* 构造器 - 创建比例尺对象

\* NavigationControl类 - 表示移动缩放控件

\* 构造器 - 创建移动缩放控件

\* Overlay类 - 遮盖物类

\* Marker类 - 表示地图的一个标注

\* 构造器 - Marker(point)

\* Point类 - 标注类

\* 拖放(拖拽)API

\* 实现拖拽效果

\* 要拖拽的文件是什么? - 源元素

\* 要拖拽到哪里去? - 目标元素

\* 目前实现拖拽效果

\* 使用原生DOM就能实现 - 最麻烦

\* 使用jQuery的插件 - 拖拽效果

\* HTML5中提供的拖拽功能

\* HTML5中拖拽

\* 源元素事件

\* dragstart - 当鼠标开始拖放时被触发

\* drag - 当鼠标拖放过程中,类似于mousemove事件

\* dragend - 当鼠标结束拖放时被触发

\* 目标元素事件

\* dragenter - 当鼠标拖放进入到目标元素内被触发

\* dragover - 当鼠标到达目前元素被触发

\* 为该事件增加event.preventDefault();

\* drop - 当鼠标实现拖放效果时被触发

\* 默认情况下,该事件没有被触发

\* 原因 - HTML页面默认情况下,不允许拖放

\* 称之为HTML页面的默认行为

\* 解决 - 阻止页面的默认行为

\* 事件对象event.preventDefault()方法

\* dragleave - 当鼠标拖放离开目标元素被触发

\* dataTransfer对象

\* 作用 - 类似于window系统的剪切板的功能

\* 功能

\* 可以将源元素的信息(数据),存储在这里

\* 将存储在该对象的源元素信息,提供给目标元素

\* 方法

\* setData() - 设置(源元素)数据

\* 在源元素事件中使用

\* getData() - 获取设置的数据

\* 在目标元素事件中使用

\* clearData() - 清除(设置的)数据

\* 所有的数据内容,存储在浏览器内存中

\* 当使用完毕数据内容时,清除

\* setDragImage()方法

\* 作用 - 修改拖放过程中,鼠标跟随的图片效果

\* 用法 - drag、dragstart等事件

\* 注意 - 实际操作中,该方法几乎不用

\* 扩展内容

\* 建立自己的技术博客

\* 作用

\* 归纳学习技术知识和经验总结等

\* 帮助检查技术专业性或是否存在错误

\* 技术博客提供与别人交流平台

\* 面试时,公司会问是否拥有技术博客

\* 技术博客网站

\* CSDN - 技术圈知名度最大的

http://blog.csdn.net/

\* 博客园 - 老牌的技术博客

http://www.cnblogs.com/

\* iteye - 专注于技术博客

http://www.iteye.com/blogs

\* 开发社区

http://segmentfault.com/

\* 将作品放在网上可以访问(Web前端)

\* hexo - 使用nodejs编写的静态博客程序

\* 地址:https://hexo.io/

\* 搭建博客网站

\* github

https://pages.github.com/

\* gitcafe

https://gitcafe.com/

\* git软件的使用

Day 06:

回顾HTML5第五天内容:

\* 百度地图

\* HTML5中的地理定位

\* 使用Google公司的Map产品(无法使用)

\* 目前实际开发中,非常常见

\* 百度地图

\* 注册百度开发者

\* 创建应用(服务器)

\* IP白名单(0.0.0.0/0)

\* 得到应用秘钥

\* 学习目的

\* 不去学习研究百度地图的逻辑

\* 最主要的目的就是如何使用百度地图

\* 百度地图

\* 提供API帮助文档(类参考)

\* 提供Demo示例代码

\* 拖放(拖拽)API

\* 源元素事件

\* dragstart - 当鼠标开始拖放时被触发

\* drag - 类似于mousemove,表示拖放过程

\* dragend - 当鼠标结束拖放时被触发

\* 目标元素事件

\* dragenter - 第一次进入目标元素时被触发

\* dragover - 进入目标元素时被触发(反复触发效果)

\* 类似于mousemove(在目标元素中)

\* 在该事件的处理函数中,利用event事件对象的preventDefault()

\* drop - 在目标元素中投放源元素时被触发

\* 在默认情况下,该事件不会被触发

\* 原因 - HTML页面默认不允许实现拖放

\* 解决 - 阻止HTML页面的默认行为

\* dragleave - 离开目标元素时被触发

\* dataTransfer对象

\* 作用 - 类似于剪切板的功能

\* 存储源元素的数据

\* 提供给目标元素

\* 方法

\* setData(type,data)方法

\* 作用 - 设置(源元素的)数据

\* 该方法在源元素事件中使用

\* getData(type)方法

\* 作用 - 获取(源元素的)数据

\* 该方法在目标元素事件中使用

\* clearData(type)方法

\* 作用 - 清除(源元素的)数据

\* 优点 - 提高性能

\* 实现拖放效果

\* 源元素

\* dragstart事件

\* 利用dataTransfer对象,存储源元素数据

\* 目标元素

\* dragover事件

event.preventDefault()

\* drop事件

\* 利用dataTransfer对象,获取源元素数据

\* 完成拖放的效果逻辑

\* setDragImage()方法

\* 作用 - 修改鼠标拖放过程中跟随的图片

\* Web Worker

\* 基本内容

\* 单线程与多线程

\* Worker可以模拟多线程的效果

\* 定义 - 运行在后台的javascript

\* 注意 - 不能使用DOM

\* 在Worker中只能使用javascript中的ECMA

\* 目前主流浏览器都支持Worker,除IE8之前

\* Worker提供API

\* 检测当前浏览器是否支持Worker

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

说明当前浏览器支持Web Worker

}else{

说明当前浏览器不支持Web Worker

}

\* 创建WebWorker对象

new Worker(worker文件)

\* Worker对象

\* onmessage事件

\* postMessage()方法

\* terminate()方法

\* Web存储API

\* 基本内容

\* 数据仓库 - 用于存储数据

\* HTML5存储系统

\* localStorage(本地存储) - 替代Cookie

\* sessionStorage(会话存储) - 替代Session(类似于内存)

\* sessionStorage

\* setItem(key,value)方法

\* 作用 - 设置数据(新增|修改数据)

\* 新增数据 - key值是新的

\* 修改数据 - key值是存在的

\* 参数

\* key - 作为存储数据的标识(唯一,不可重复)

\* value - 存储的数据内容(number|string)

\* getItem(key)方法

\* 作用 - 获取数据(读取数据)

\* 参数

\* key - 根据key获取对应的数据内容

\* 返回值 - key对应的value值

\* key(index)方法

\* 作用 - 根据索引值返回对应key

\* 参数

\* index - 索引值

\* 返回值 - 返回key值

\* removeItem(key)方法

\* 作用 - 删除数据

\* 参数

\* key - 根据key删除指定数据内容

\* clear()方法

\* 作用 - 将存储系统的所有数据删除(清空)

\* length属性

\* 作用 - 返回当前存储系统的数据个数

\* localStorage

\* 提供的属性和方法与sessionStorage一致

\* storage事件

\* 作用 - 实现多个窗口之间共享数据内容

\* 问题

\* 数据安全性低

\* 目前除Safari浏览器支持该事件,其他所有浏览器都不支持

\* 对于数据的操作

\* 新增 - setItem()

\* 查询 - getItem()

\* length属性

\* key(index)

\* 删除 - removeItem()

\* clear()

\* 修改

\* key(index) - key

\* getItem() - value

\* setItem(key,新value)

\* sessionStorage与localStorage

\* sessionStorage存储系统存储数据的特点

\* 当浏览器窗口关闭时,数据全部丢失

\* 当再次打开浏览器窗口时,数据不能使用

\* localStorage存储系统存储数据的特点

\* 当浏览器窗口关闭时,数据依旧存储

\* 当再次打开浏览器窗口时,数据继续使用

\* 数据只能由用户删除

\* Web Socket(了解)

\* 基本内容

\* socket - 是一种网络协议

\* 网络协议 - SERVER&HTTP课程中的HTTP

\* WebSocket

\* 表示在HTML5页面中允许支持socket协议

\* 浏览器-服务器

\* 使用的网络协议 - http协议

\* http协议的问题

\* 短连接

\* 无状态

\* WebSocket提供的API

\* 创建WebSocket对象

var socket = new WebSocket(URL);

\* URL - 是以"ws://"开始的

\* 创建Socket对象,建立连接

\* WebSocket对象的方法

\* send() - 向服务器端发送数据内容

\* close() - 关闭socket链接

\* WebSocket对象的事件

\* onmessage事件 - 当服务器端向客户端发送返回数据时被触发

\* onopen事件 - 当客户端与服务器端建立连接时被触发

\* 完成初始化功能

\* onclose事件 - 当客户端与服务器端关闭链接时被触发

\* 完成资源释放功能

\* WebSocket对象的属性

\* readyState属性 - 表示服务器端的通信状态

\* CONNECTING(数字值为0)，表示正在连接。

\* OPEN(数字值为1)，表示已建立连接。

\* CLOSING(数字值为2)，表示正在关闭连接。

\* CLOSED(数字值为2)，表示已关闭连接。

CSS**3**

***Day 01:***

练习1:

1、创建四个元素 ，分别定义边框、宽度、内边距

p div p p

2、div定义class为s1

3、通过相邻兄弟选择器 将 div下一个p更改为 红色字体

div.s1+p

div+p

4、通过通用兄弟选择器 将 div后所有的p 背景改为 #ccc色

div~p

2、属性选择器

将元素属性用于选择器中，从而获取指定元素属性或者值得元素

语法：

[属性] : 具备 指定属性 的所有元素全部匹配出来

元素[属性] : 匹配具备 属性的 指定元素

p[id] : 匹配具备id属性的p元素

div[class] :

元素[属性1][属性2] :

p[id][class] : 匹配出所有即具备id属性，又具备class属性的p元素

元素[属性=值] :

input[type="text"] : 匹配type的值为text的input元素

元素[属性~=值] :

input[class ~= second] :

<input class="first second" />

<input class="myseconddiv" />

~= : 包含指定的数据（独立）

= : 只有指定的数据

元素[属性^=值] : 匹配指定属性以指定值开始的指定元素

p[class^=f] : 匹配class以f开始的p元素

<p class="first"></p>可以匹配

<p class="fast"></p>可以匹配

<p class="second"></p>不能匹配

元素[属性\*=值] : 属性包含值元素

p[class\*="valid"];

<p class="myvalid"></p>

元素[属性$=值] : 匹配属性以指定值结束的元素

元素[属性!=值] : 匹配属性不等于具体值得元素

eg:

input[id] : 获取具备id属性的input元素

<input type="text" />

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

练习2:

过 .....

========================1、伪类选择器

回顾：什么是伪类选择器

更改元素的不同的状态的样式

:hover :link :visited :active

1、目标伪类

1、什么是目标伪类

突出显示活动的HTML锚，用于选取当前活动的目标元素

2、语法

:target

p:target

2、元素状态伪类

1、什么是元素状态伪类

主要匹配元素的禁用、启用、选中状态

多数应用在表单元素上

2、常用的状态伪类

:disabled 匹配每个被禁用的元素

:enabled 匹配每个已启用的元素

:checked 匹配每个被选中的元素(只用于单选按钮和复选框)

3、结构伪类

1、什么是结构伪类

主要从元素的结构(层级结构)上来进行划分的

2、什么时候使用结构伪类

找 第一个子元素、最后一个子元素、没有子元素、仅仅只包含一个子元素 优先使用结构伪类

3、如何使用结构伪类

:first-child 匹配属于其父元素的首个子元素

:last-child 匹配属于其父元素的最后一个子元素

:empty 匹配没有子元素(包含文本)的元素

<p>这是一个段落</p> 非empty

<div></div> empty

:only-child 匹配属于其父元素的唯一子元素

<div>

<span></span>

<span></span>

</div>

<div>

<span></span>

</div>

4、否定伪类

匹配非指定元素/选择器的每个元素

语法:

:not(selecotr)

eg:

1、获取 input 但是 type不是text的所有元素

input[type=text]

input:not([type=text])

2、获取除第一个div子元素意外的其他div子元素

div:first-child

div:not(div:first-child)

2、伪元素选择器

主要针对元素中的文本内容进行匹配的。

1、:first-letter

选取指定选择器的首字母

eg:

p:first-letter{

}

一般用于 排版、首字符突出等操作

2、:first-line

选取指定选择器的首行

3、::selection

匹配用户选取的部分

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

1、内容生成

1、什么是内容生成

通过css 向元素的前面或后面增加一部分内容

2、伪元素

:before 匹配到某一元素的最前面

p:before

:after 匹配到某一元素的最后面

p:after

3、内容生成

content : 配合 :before 或 :after伪元素，插入生成内容

content的常用取值:

1、字符串:纯文本，插入到指定的位置处

p:before{

content:"至尊宝:";

}

2、url:生成的图像

a:before{

content:url(Images/i1.jpg);

}

3、计数器

2、计数器

1、计数器的作用

向已存在的html文本内容增添序号

2、如何使用计数器

1、定义计数器

属性:counter-reset

作用:定义一个计数器，并且设置初始值为0

body{

counter-reset:计数器名称 初始值 计数器名称 初始值;

}

body{

counter-reset:count 10;

}

2、设置计数器增量

可以单独设置某一选择器使用计数器时的增量，默认值是1

属性:counter-increment

div{

counter-increment:count 10;

}

3、使用计数器

通过 counter(计数器名称) 来使用计数器

div:before{

content:"第"counter(count)"章";

}

3、多列

1、分隔列

把一段文本拆分成几列

column-count:规定元素被分隔的列数

2、列间隔

设置列与列之间的距离

column-gap:间隔;

3、列规则

可以在列与列之间设置一条间隔线，列规则可以设置间隔线的样式、颜色、粗细

column-rule:大小 样式 颜色;

column-rule-width:

column-rule-style:

column-rule-color:

4、浏览器兼容

1、IE10 以及 Opera 浏览器支持多列属性

2、FireFox 需使用前缀 -moz-

-moz-column-count:

-moz-column-gap:

-moz-column-rule:

3、Chrome 和 Safari

-webkit-column-count:

-webkit-column-gap:

-webkit-column-rule:

4、CSS Hack

1、为什么需要CSS Hack

浏览器兼容性所引发的问题

2、浏览器运行模式

1、混杂模式

2、标准模式(Standard Mode)

3、准标准模式(Almost Standard Mode)

3、浏览器主要通过 DOCTYPE 进行模式选择

触发混杂模式:不声明DOCTYPE

触发标准模式:

4、浏览器兼容问题

1、margin和padding

ie6中 默认的 maring 偏大

解决方案:body{margin:0px;}

2、居中布局

ie6以及低版本浏览器:通过父元素的text-align:center;完成居中

高版本浏览器(ie6,ie7+,chrome,firefox,opera,safari):通过当前元素的margin实现即 margin:0 auto;

3、元素高度与内容

ie6 : 元素的高度至少包括内容

其他:元素高度会超出容器

解决方案:overflow属性

4、子元素设置上外边距时

ie6:正常显示

其他版本:子元素的上外边距会认为是父元素的上外边距

解决方案:

1、给父元素设置边框

2、设置父元素的 padding-top

5、HTML头部引用Hack

通过HTML条件注释完成

<!--[if 条件]>

<![endif]-->

IE6:

<!--[if IE 6]>

编写HTML

引入css

<![endif]-->

大于IE6的版本:

<!--[if gt IE 6]>

<![endif]-->

练习：

1、创建4个超链接分别链接4个锚点

<a href="#img1">img1</a>

2、创建4个锚点 ，分别是四幅图像

<img id="img1" />

3、让4个锚点全部隐藏

4、哪个锚点被激活，哪副图像显示出来

Day 02:

1、变形 - transform 迅速变化

2、过渡 - transition 缓慢变化

3、动画 - animation 关键帧动画

1、转换原点

默认情况下，原点是在当前物体的中心点上

修改原点:

属性:transform-origin

值:

1、x坐标 y坐标

当前物体的左上点为 x:0 y:0

2、宽度百分比 高度百分比

0% 0% 左上点

50% 50% 中心点

3、关键字

top:上

bottom:下

left:左

right:右

center:中间

center center

left top

transform-origin:0; 表示 所有轴的位置都将归到0点处

transform-origin:50px 50px; 表示 x轴在50像素处，y轴在50像素处

transform-origin:50px 50px 50px; 表示 x轴，y轴，z轴

2、变形:2d

旋转:rotate()

位移:translate()

缩放:scale()

倾斜:skew()

旋转:围绕一个参照原点(transform-origin),旋转指定角度，默认为顺时针

语法:transform:rotate(ndeg);

n为正 则按顺时针旋转

n为负 则按逆时针旋转

注意:rotate在旋转的过程中，不但能够旋转元素，同时能够旋转绘图的坐标系方向。如果配合着其他的变形一起来做的话，rotate在前的话，会影响之后延坐标轴的其他变形。

解决方案:如果配合变形一起来做，最好将rotate放在最后一个函数上

位移:延 坐标方向 移动指定的距离

语法:translate(x轴移动距离,y轴的移动距离);

x : 正为右，负为左

y : 正为下，负为上

transform:translate(50px,50px);

注意:位移不会影响其他元素位置,单可能会盖住周围元素

其他两个单方向位移:

translateX(距离);

translateY(距离);

缩放:将指定坐标轴上的坐标缩放指定的倍数

语法:scale(倍数)

倍数 : 0-1 之间的小数,缩小

>1 放大

scale(倍数) 表示等比缩放

其他两个单方向缩放:

scaleX(倍数);

scaleY(倍数);

transform:scale(2);

倾斜:沿着坐标轴方向，倾斜指定角度

语法:skew(ndeg);仅沿x轴倾斜

skew(ndeg,ndeg);沿着x轴和y轴同时倾斜

单方向倾斜:

skewX(ndeg)

skewY(ndeg)

x轴方向:角度为正，向左倒

角度为负，向右倒

y轴方向:角度为正向上倾斜

角度为负向下倾斜

3、3d变形

元素,要从立体角度观察

坐标轴,x轴，y轴，z轴

属性:

perspective

设定假定的人眼位置到投影平面的距离

如何使用：设置在父元素上的

浏览器兼容性:

Chrom,Safari : -webkit-perspective

Firefox : -moz-perspective

位移:

3D位移可以改变元素在z轴上的位置

translateZ(z);

translate3d(x,y,z);

旋转:

transform:

rotateX(ndeg);

rotateY(ndeg);

rotateZ(ndeg);

缩放:

transform:

scaleZ(z);

scale3d(x,y,z);

1、过渡

transition

1、什么是过渡

CSS属性在一段时间内进行平滑的过渡

2、过渡四要素

1、过渡属性:background,color,transform

2、过渡所需时间:

3、过渡函数:速度和方式

4、过渡延迟时间:激发操作后的执行间隔(s|ms)

3、过渡属性

1、过渡属性

属性名:transition-property

transition-property:background,color;

2、过渡时间:

属性名:transition-duration:

以秒(s)或毫秒(ms)为单位

transition-duration:5s;

3、过渡函数

属性名:transition-timing-function

备选值:

ease:默认值，慢速开始，快速变快，慢速结束

linear:匀速过渡

ease-in:慢速开始，加速效果

ease-out:以慢速结束，减速效果

ease-in-out:以慢速开始和结束，中间先加速再减速

4、过渡延迟

属性名:transition-delay

以秒或毫秒为单位

5、整合transition属性

transition:属性名 持续时间 过渡函数 [延迟],

属性名 持续时间 过渡函数 [延迟];

2、关键帧动画

关键帧:动画执行过程中，物体在某一位置上的特殊状态

关键帧动画:使用连续的关键帧，控制物体联系的状态变化

什么时候使用关键帧动画:

1、连续，有规律的过渡 -- 过渡 transition

2、无规律的连续变化 -- 动画 animation

如何实现:

2大步:

1、定义关键帧

@keyframes 动画名{

from{

0%

css样式

动画开始的状态

}

percent{

}

to{

100%

css样式

动画结束的状态

}

}

@keyframes change{

from{

background:red;

}

10%

{

background:yellow;

}

50%

{

background:green;

}

to

{

background:pink;

}

}

浏览器兼容性:

@keyframes : IE , 火狐

@-webkit-keyframes : Chrome Safari

@-o-keyframes : Opera

2、触发动画(调用动画)

属性:animation

animation:动画名 持续时间 速度类型;

如何:

1、写在非伪类选择器中，页面加载则执行动画

2、写在伪类中，伪类触发时则播放动画

浏览器兼容性:

-webkit-animation: Chrome，Safari

-o-animation:Opera

3、动画子属性

1、animation-name : @keyframes(动画)名称

2、animation-duration : 动画时长

3、animation-timing-function:动画速度函数

4、animation-delay:动画的延迟时间

animation:name duration timing-function;

5、animation-iteration-count : 播放次数

取值:

1、具体数值

2、infinite(无限次播放)

6、animation-direction : 动画播放方向

取值:

1、normal 正常播放

2、alternate 轮流播放

奇数次数：正向播放

偶数次数：反向播放

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

7、animation-fill-mode

动画播放前后的效果

备选值:

1、none:不改变默认行为

2、forwards:动画完成后保证最后一个属性状态

3、backwards:动画播放前，显示开始属性值

4、both

8、animation-play-state

控制动画播放与暂停

备选值:

1、paused : 暂停动画

2、running : 播放动画

使用场合:

1、配合伪类选择器

2、结合Javascript使用

@keyframes second{

from{

transform:rotate(0deg);

}

to{

transform:rotate(360deg);

}

}

调用

#second{

animation:second 60s linear 0s infinite;

}

SERVER&HTTP

***SERVER&HTTP DAY01:***

\* 学习SERVER的原因

\* 招聘 - Web前端(熟悉或了解某种服务器端技术优先)

\* 工作 - 与服务器端人员配合

\* 学习 - 掌握或了解Web应用的整体架构

\* 服务器概念

\* 服务器(SERVER)其实就是一台PC机(硬件)

\* 服务器的分类

\* 硬件服务器 - PC机

\* 电脑硬件 - PC机/小型机/刀片机/中型机/大型机/超级计算机

\* 小型机 - IBM(AIX)/HP/联想(Linux)

\* 软件服务器 - 中间件

\* 为了运行Web应用的一种软件

\* 软件架构

\* B/S - 浏览器(browser)/服务器端(server)

\* 互联网-企业级

\* 互联网 - 网易、腾讯、百度等

\* 企业级 - 银行系统、医院系统等

\* 好处

\* 软件升级 - 服务器端的升级

\* C/S - 客户端(client)/服务器端(server)

\* 先出现,例如QQ、Email等

\* 原因 - 网络带宽/电脑硬件普遍偏低

\* 问题

\* 软件升级 - 客户端\服务器端都要升级

\* 软件服务器

\* Web(应用)服务器 - 用于运行Web应用

\* 数据库服务器 - 用于运行数据库产品

\* 访问Web应用

http://www.baidu.com

网络协议-IP地址-端口号

\* XAMPP软件

\* Apache - 软件服务器(运行PHP)

\* 启动失败

\* 原因 - 端口号被占用

\* 错误信息 - Error: Apache shutdown unexpectedly.

\* 解决

\* config选项-<Browse>[Apache]-conf目录-httpd.conf文件

\* 修改 - Listen 8888(端口号)

\* 如何访问

\* http://localhost:端口号

\* http://127.0.0.1:端口号

\* 设置虚拟地址

\* 目录 - C:\Windows\System32\drivers\etc

\* 打开hosts文件

\* 关于80端口号

\* 在开始菜单输入"cmd"运行命令行窗口

\* 在命令行窗口输入命令

netstat -ano

\* 找到占用80端口号对应PID值

\* 在任务管理器中的"进程",找到对应的进程,结束

\* 在本地搭建Web应用程序

\* 目录 - XAMPP软件的安装路径-htdocs

\* 将该目录的所有文件删除

\* 创建index.html页面文件

\* 重新在浏览器中访问虚拟地址,页面内容

\* MySQL - 数据库服务器(运行MySQL)

\* MySQL默认使用的端口号 - 3306

\* 不建议修改该端口号

\* 命令行方式登录(打开)数据库

\* 登录数据库 - mysql -u用户名 -p密码

\* 退出数据库 - exit;

\* 同时启动Apache和MySQL服务

\* 访问地址 - http://localhost:8888/phpmyadmin

\* 上述地址的端口号是Apache的端口号

\* phpMyAdmin服务是Apache提供的(连接数据库)

\* Tomcat - 软件服务器(运行Java)

\* 数据库

\* 基本概念

\* 数据库 - 数据仓库,用于存储或操作数据内容

\* 两大阵营

\* 关系型数据库(SQL) - 是目前主流数据库

\* 是以表(行和列)的形式存储数据

\* 非关系型数据库(NoSQL) - 是新潮流数据库

\* 是以文档方式存储数据

\* 是以key:value形式存储数据

\* ...

\* NoSQL

\* 是一种运动(抗议关系型数据库)

\* 逐步发展之后,出现很多这种产品

\* 主流的产品

\* mongoDB - JSON格式

\* ...

\* 关系型数据库

\* Oracle - 甲骨文(Oracle)公司的产品

\* 企业级开发98%市场份额都是使用这款产品

\* Oracle认证

\* 初级 - 月薪10K - 15K

\* 中级 - 月薪15K - 20K

\* 高级 -

\* MySQL - 甲骨文(Oracle)公司的产品

\* 互联网开发98%市场份额都是使用这款产品

\* SQL Server

\* 是微软公司推出的

\* 只提供Windows操作系统版本

\* Access(Office套件的组件)

\* 是微软公司推出轻量级数据库

\* DB2

\* MySQL产品

\* 是瑞典MySQL AB公司开发的

\* 后期被SUN公司收购(SUN公司最主要产品-Java)

\* SUN公司被Oracle公司收购

\* 特点

\* 免费

\* 开源

\* 目前两种版本

\* 社区版本 - 免费

\* 商业版本 - 收费

\* LAMP组合 - 目前开发互联网网站

\* L - Linux

\* A - Apache

\* M - MySQL

\* P - PHP

\* SQL语言 - 无论数据库产品,使用SQL语言

\* DDL - 数据定义语言(数据库+表)

\* DCL - 数据控制语言(权限)

\* DQL - 数据查询语言

\* DML - 数据操作语言

\* 注意

\* SQL语言并不区分大小写(官方建议大写)

\* SQL语句编写完毕后,一定增加";"结束符

- 命令行中

\* SQL语言使用字符串时,建议使用单引号'

\* DDL(了解) - CREATE|ALTER|DROP

\* 数据库操作

\* 创建数据库

\* 用法 - CREATE DATABASE 数据库名称 DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci;

\* 查看数据库

\* 用法 - SHOW DATABASES;

\* 修改数据库

\* 用法 - ALTER DATABASE 数据库名称 CHARACTER SET utf8;

\* 删除数据库

\* 用法 - DROP DATABASE 数据库名称;

\* 使用(切换)数据库

\* 用法 - USE 数据库名称;

\* 注意

\* 数据库一旦被创建,很少修改或删除

\* 创建、查看和切换数据库

\* 数据表操作

\* 数据类型

\* 数值(Number)数据类型

\* INT - 整数

\* FLOAT/DOUBLE - 浮点型(小数)

\* DECIMAL - 精确值(金额等)

\* 日期(Date)数据类型

\* DATE - 日期(默认格式:yyyy-MM-dd)

\* DATETIME - 日期时间(yyyy-MM-dd hh:mm:ss)

\* TIMESTAMP - 时间戳(标识:唯一)

\* 字符串(String)数据类型

\* CHAR - 长度固定的字符串

\* 定义一个字符串的长度为10,实际存储的内容为"abc",未被字符占用的位置会以空格补位

\* VARCHAR - 长度可变的字符串

\* 定义一个字符串的长度为10,实际存储的内容为"abcde"

\* 创建数据表

CREATE TABLE (

字段名称1 数据类型,

字段名称2 数据类型,

...

);

\* 约束

\* 主键约束 - PRIMARY KEY

\* 作用 - 唯一,不可重复

\* 主键自增约束 - AUTO\_INCREMENT

\* 作为主键的字段,自增

案例:创建用户表(id,name,pwd,age,email,addr)

CREATE TABLE myuser (

//标识(唯一,不可重复)

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(30),

pwd VARCHAR(30),

age INT,

email VARCHAR(30),

addr VARCHAR(30)

);

练习:创建订单表myorder(orderid,ordername,price,addr,ordertime)

\* orderid - 订单表的标识(唯一)

\* ordername - 订单表的名称

\* price - 金额

\* addr - 收货地址

\* ordertime - 订单日期

\* 删除数据表

\* 用法 - DROP TABLE 表名;

\* 查看表结构

\* 用法 - DESC 表名;

\* DML(增删改) - INSERT|UPDATE|DELETE

\* 插入(新增)数据

\* 用法一

INSERT INTO 表名 VALUES(字段值1,字段值2,...);

\* 注意

\* 当前表具有多少字段,VALUES输入多少字段值

\* 如果哪个字段是主键自增的话,使用NULL补位

\* 用法二

INSERT INTO 表名(字段名1,字段名2,...) VALUES(字段值1,字段值2,...)

\* 注意

\* 表名后定义多少字段,VALUES后输入多少字段

\* 当前数据表的字段是允许为空的

\* 练习 - myorder表

1. 插入orderid,ordername,price,addr,ordertime

INSERT INTO myorder VALUES(NULL,'linghuchong',3.5,'huashan','2015-11-17 15:44:40')

2. 插入ordername,addr,ordertime字段值

INSERT INTO myorder(ordername,addr,ordertime) VALUES('renyingying',10,'heimuya')

\* 更新(修改)数据

\* 用法一 - UPDATE 表名 SET 字段名=字段值;

\* 注意 - 修改所有数据(指定字段值)

\* 用法二

UPDATE 表名 SET 字段名=字段值 WHERE 字段名=字段值;

\* SET后面的"字段名=字段值",为设置的值

\* WHERE后面的"字段名=字段值",为查询的值

\* 用法三

UPDATE 表名 SET 字段名1=字段值1,字段名2=字段值2 WHERE 字段名=字段值;

\* 练习:修改myorder表

\* 修改addr字段值为"光明顶"

UPDATE myorder SET addr='光明顶';

\* 修改id为2记录的price修改为100

UPDATE myorder SET price=100 WHERE orderid=2;

\* 修改addr为"光明顶"记录的ordertime为"2020-11-11 00:00:00"

UPDATE myorder SET ordertime='2020-11-11 00:00:00' WHERE addr='光明顶'

\* 删除数据

\* 用法一 - DELETE FROM 表名;

\* 注意 - 删除指定表中所有数据

\* 用法二 - DELETE FROM 表名 WHERE 字段名=字段值;

\* 问题 - 实际操作

\* 实际的开发中基本不使用DELETE语句

\* 以防我们的用户反悔

\* SQL操作(删除)

\* 物理删除 - 执行DELETE语句

\* 逻辑删除

\* 简单来说,对于用户来讲是删除的,对于实际来讲并没有删除

\* 为指定表,增加一个字段(state|status),表示当前这条记录是什么状态

\* 值为1的话,表示这条记录是正常的

\* 值为0的话,表示这条记录是删除的

\* DQL(查)

\* 基本查询 - 新版本一样

\* 用法一 - SELECT \* FROM 表名;

\* 用法二 - SELECT 字段名1,字段名2,... FROM 表名;

\* 条件基本查询

\* 用法 - SELECT \* FROM 表名 WHERE 字段名=字段值;

\* 复杂条件查询

\* AND - 表示多个条件同时满足

\* OR - 表示其中一个条件满足

\* IN(SET) - 表示一个字段包含多个值

\* SET - 多个值,之间使用","

\* = - 表示字段值为指定值

\* BTWEEN AND - 等于 >= AND <=

\* IS NULL - 匹配NULL值

\* 排序查询 - ORDER BY 字段名

\* ASC - 正序排序,默认值

\* DESC - 倒序排序

SELECT \* FROM 表名 WHERE 条件 ORDER BY 字段名;

\* 练习 - myorder

1. 查询addr为"光明顶"的所有记录

SELECT \* FROM myorder WHERE addr='光明顶';

2. 查询id为1或ordername为"renyingying"的所有记录

SELECT \* FROM myorder WHERE id=1 OR ordername='renyingying';

3. 查询id为1,2,3的所有记录

SELECT \* FROM myorder WHERE id in(1,2,3);

\* 命令行方式登录MySQL

\* 输入mysql命令,提示"mysql"不是内部命令

\* 原因 - 系统环境变量中没有配置MySQL的变量

\* 解决

\* 进入到MySQL的安装目录中

\* 进入到bin目录中,找到mysql.exe文件

\* 执行mysql.exe文件

\* 在命令行中输入命令

\* cd mysql的安装目录/bin

\* mysql命令

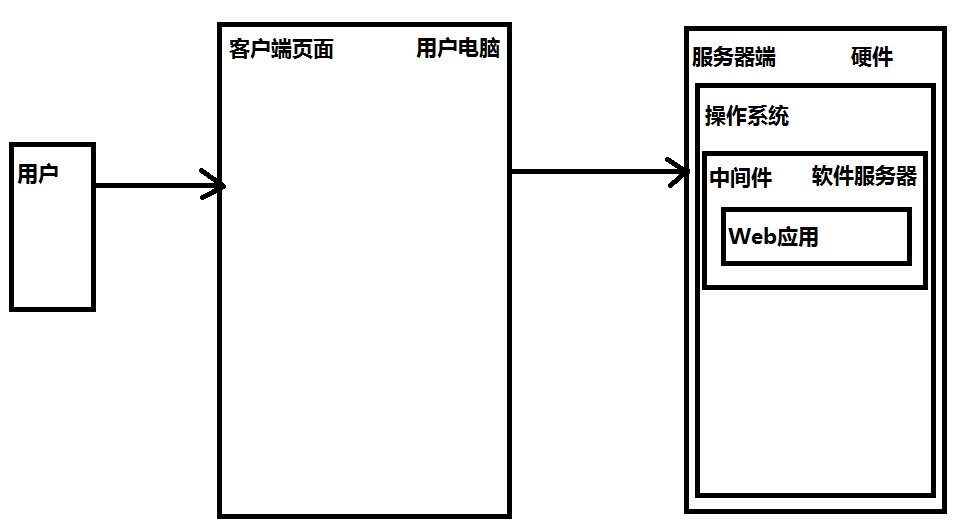
\* 数据库操作原则 - 只增不删

\* 数据库的空

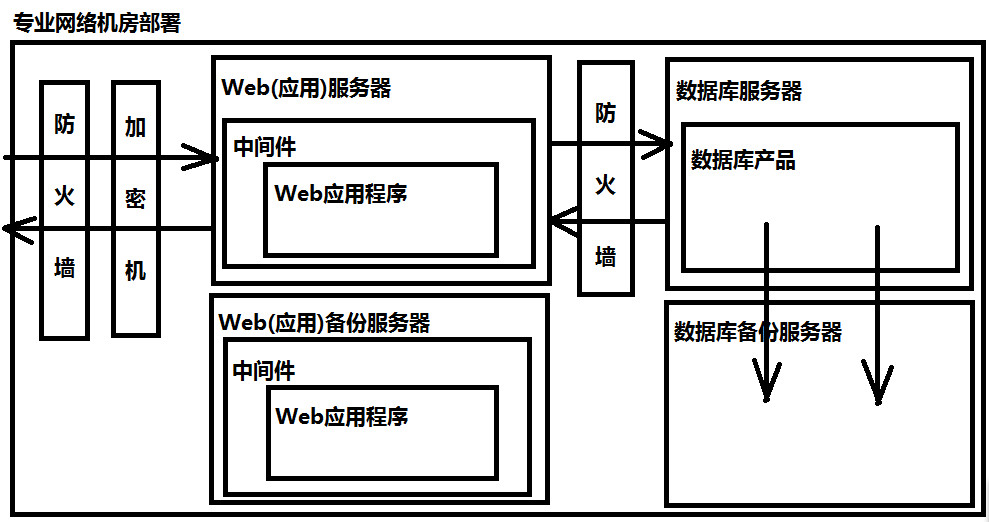
\* '' - 空字符串

\* NULL - 值不存在

**软件与硬件服务器：**



专业网络机房部署图：



***回顾SERVER&HTTP第一天内容:***

\* 基本概念

\* 服务器

\* 硬件服务器 - PC机/小型机等

\* 软件服务器 - 中间件(运行Web应用的软件)

\* Web架构

\* C/S架构 - Client(客户端)/Server(服务器端)

\* B/S架构 - Browser(浏览器)/Server(服务器端)

\* XAMPP软件

\* Apache - 运行PHP的服务器

\* Tomcat - 运行Java的服务器

\* MySQL - 数据库产品

\* 数据库

\* 数据库 - 数据仓库,用于存储或操作数据

\* 关系型数据库 - 简单来说,就是表(行和列)

\* Oracle

\* MySQL

\* NoSQL - 非关系型数据库

\* SQL语言 - 应用在各种数据库产品中

\* DDL - 数据定义语言(数据库|表)

\* DCL - 数据控制语言

\* DML - 数据操作语言(增删改)

\* DQL - 数据查询语言(查)

\* DDL - CREATE|ALTER|DROP

\* 数据库

\* 创建数据库 - CREATE DATABASE 数据库名称

\* 查看数据库 - SHOW DATABASES

\* 修改数据库 - ALTER DATABASE 数据库名称 修改内容

\* 删除数据库 - DROP DATABASE 数据库名称

\* 使用(切换)数据库 - USE 数据库名称

\* 数据表

\* 数据类型

\* 数值(Number)类型

\* INT - 整型

\* FLOAT/DOUBLE - 浮点型

\* DECIMAL - 精确值

\* 日期(Date)类型

\* DATE - 日期(yyyy-MM-dd)

\* DATETIME - 日期/时间(yyyy-MM-dd hh:mm:ss)

\* TIMESTAMP - 时间戳(yyyy-MM-dd hh:mm:ss)

\* 字符串(String)类型

\* CHAR - 长度固定的字符串

\* VARCHAR - 长度可变的字符串

\* 创建数据表

CREATE TABLE 表名(

字段名1 数据类型,

字段名2 数据类型,

...

)

\* 为字段增加约束

\* 主键约束 - PRIMARY KEY

\* 主键自增约束 - AUTO\_INCREMENT

\* 修改数据表

\* 删除数据表 - DROP TABLE 表名

\* DML - INSERT|UPDATE|DELETE

\* 插入(新增)数据

INSERT INTO 表名(字段名1,字段名2,..) VALUES(字段值1,字段值2,...)

INSERT INTO 表名 VALUES(所有字段值)

\* 更新(修改)数据

UPDATE 表名 SET 字段名1=字段值1,字段名2=字段值2 WHERE 条件

\* 删除数据

\* 物理删除 - DELETE FROM 表名 WHERE 条件

\* 逻辑删除 - 实际并没有执行删除操作

\* 增加一个字段,作为一个标识(是否为删除)

\* 通过修改标识的值,在逻辑上表示删除

\* DQL

SELECT \* FROM 表名 WHERE 条件 ORDER BY 字段名

***SERVER&HTTP DAY02:***

\* PHP语言

\* 基本内容

\* PHP语言 - 类似于javascript语言的

\* javascript是客户端(HTML)的脚本语言

\* PHP是服务器端的脚本语言

\* PHP文件的扩展名为".php"

\* 注意

\* PHP文件是不能鼠标右键,使用浏览器直接打开

\* PHP文件是应该运行在服务器(Apache)上的

\* PHP文件允许的内容

\* 允许编写HTML代码

\* 允许编写PHP代码

\* PHP页面与HTML页面的区别

\* HTML页面 - 静态页面

\* 静态资源 - HTML|CSS|JAVASCRIPT

\* PHP页面 - 动态页面

\* 动态资源 - 根据用户的操作,动态变化

\* 第一个PHP文件

\* 创建PHP文件 - 扩展名为".php"

\* 如何编写PHP代码

\* 是以"<?php"开始,是以"?>"结束的

\* 使用echo进行输出

\* PHP语法

\* 常量与变量

\* 常量 - 一旦定义并初始化后,值不会改变

\* 使用const关键字

const 常量名=常量值

\* define(常量名,常量值)

\* 变量 - "$"符号只是表示当前定义的是变量

$变量名=值;

\* 数据类型

\* 四种标量类型

\* boolean - 布尔类型

\* integer - 数值类型(整型)

\* float/double - 数值类型(浮点型)

\* string - 字符串

\* '' - 定义固定字符串

\* 性能比较高

\* "" - 可以识别变量名的

\* 性能相对低(有匹配的过程)

\* 两种复合类型

\* array - 数组

\* object - 对象

\* 两种特殊类型

\* resource - 资源

\* 作用 - 用于保存外部资源的一个引用

\* 使用场景

\* 在文件上传中,保存上传的文件

\* NULL

\* 运算符

\* 基本与javascript的运算符保持一致

\* 字符串连接符 - "."

\* 在实际开发中,很少使用

\* 循环结构

\* while

\* do...while

\* for

\* foreach

foreach(数组 as key => value){}

\* 分支结构

\* if...else if...else

\* switch...case

\* break与continue的区别

\* break - 结束循环

\* continue - 结束本次循环

\* PHP预定义

\* 预定义变量

\* $\_GET - 接收客户端以请求类型为GET方法发送的数据内容

\* $\_POST - 接收客户端以请求类型为POST方法发送的数据内容

\* $\_REQUEST - $\_GET、$\_POST等

\* $\_FILES - 专门用于文件上传

\* $\_COOKIE - 接收客户端保存的Cookie数据

\* 预定义函数

\* 数据库扩展

\* PHP连接MySQL数据库

\* 准备工作

\* 在XAMPP软件的安装目录/php/ext目录中

\* php\_mysql.dll和php\_mysqli.dll文件必须存在

\* 在XAMPP软件的安装目录/php目录中

\* php.ini文件 - php的主配置文件

extension=php\_mysql.dll

extension=php\_mysqli.dll

\* 在php中配置有关MySQL数据库

\* PHP如何连接MySQL数据库

\* 过程化风格

\* 建立与MySQL数据库的连接

$conn = mysqli\_connect(host,username,passwd,dbname,port);

\* 该方法返回数据库的连接对象

\* 定义SQL语句

$sql = "";

\* 发送SQL语句 - MySQL数据库

$result = mysqli\_query($conn,$sql);

\* 该方法返回执行SQL语句的结果

\* (可选)解析结果集对象

\* 结果集对象 - mysqli\_result对象

\* 关闭与MySQL数据库的连接

mysqli\_close($conn)

\* 面向对象风格

\* 创建mysqli或mysql对象

$mysqli = new mysqli(host,username,passwd,dbname,port);

\* 相当于与MySQL数据库建立连接

\* 定义SQL语句

\* 调用mysqli对象的query()方法,向MySQL数据库发送SQL语句

$mysqli->query($sql);

\* 如果SELECT语句,解析mysqli\_result对象

\* 调用mysqli对象的close()方法,关闭与MySQL的连接

\* 中文乱码问题

\* 执行mysqli\_query($conn,'SET NAMES UTF8');

\* 执行$mysqli->query('SET NAMES UTF8');

\* 结果集对象 - 执行SELECT语句

\* 属性

\* num\_rows - 记录数量

\* field\_count - 字段数量

\* 方法

\* mysqli\_fetch\_array(结果集对象)

\* 返回数组

\* mysqli\_fetch\_object(结果集对象)

\* 返回object

\* 案例 - 开发完整的Web应用(客户端|服务器端|数据库)

\* 用户登录功能

\* 用户注册功能

\* 先查询用户名是否存在

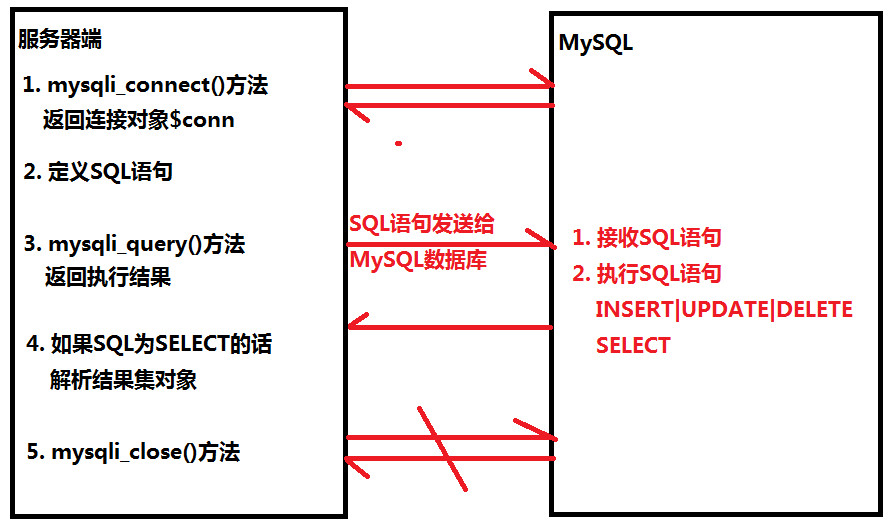
SELECT \* FROM 表名 WHERE name=用户名

\* $result->num\_rows - 0(不存在)

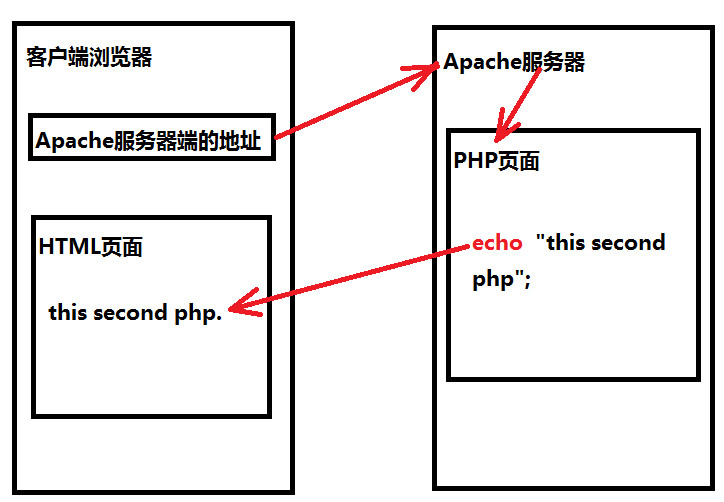
\* $result->num\_rows - 不为0(存在)

\* 再插入数据内容

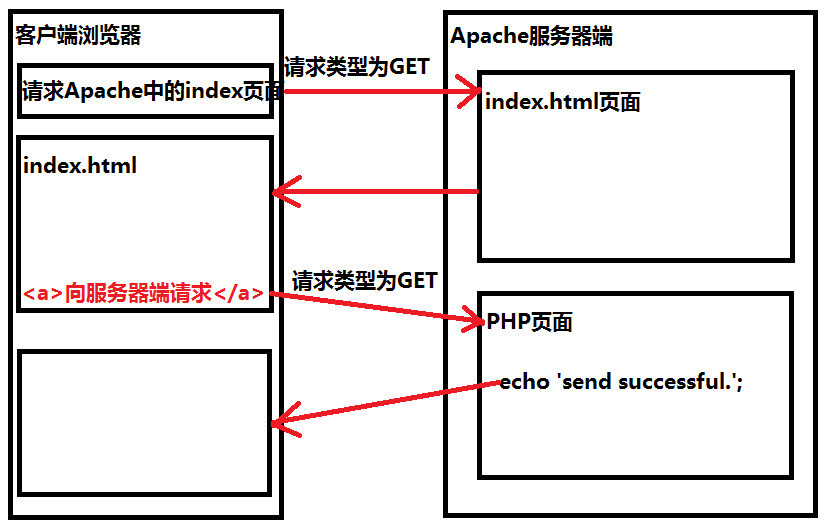
php操作mySQL的执行流程图：



php文件的运行流程图：



客户端与服务器端的交互流程图：



***回顾SERVER&HTTP第二天:***

\* PHP语言

\* PHP - 类似于javascript语言

\* javascript是客户端的脚本语言

\* PHP是服务器端的脚本语言

\* PHP文件的扩展名为".php"

\* 第一个PHP文件

\* 创建PHP文件 - 扩展名为".php"

\* 编写PHP代码时

\* 是以"<?php"开始,是以"?>"结束

\* PHP基础语法

\* 常量与变量

\* 常量 - 定义后值不变(不允许重复赋值)

\* const 常量名 = 常量值;

\* define(常量名,常量值);

\* 变量

$变量名=值;

\* 数据类型

\* 四种标量类型

\* boolean - true|false

\* true响应给客户端页面 - 1

\* false响应给客户端页面 - ''

\* integer - 整型

\* float/double - 浮点型

\* string - 字符串

\* '' - 定义固定内容的字符串

\* 性能相对高

\* "" - 识别变量名称

\* 两种复合类型

\* array - 数组

var arr1 = [];

var arr2 = array(key=>value,key=>value)

\* object - 对象

class 类名{}

var 对象 = new 类名;

\* 两种特殊类型

\* resource - 资源类型

\* NULL

\* 运算符

\* PHP中的大部分运算符的使用方式与javascript中的元素符类似

\* 字符串的连接符 - "."

\* 循环结构

\* while

\* do...while

\* for

\* foreach - 类似于javascript中的forin

\* 分支结构

\* if...else

\* switch...case

\* break与continue的区别

\* PHP预定义

\* 预定义变量

\* $\_GET - GET方式的请求数据

\* $\_POST - POST方式的请求数据

\* $\_REQUEST

\* $\_FILES

\* $\_COOKIES

\* 预定义函数

\* 数据库相关扩展内容

\* PHP连接MySQL数据库

\* 准备工作

\* PHP\_HOME/ext目录 - php\_mysql.dll和php\_mysqli.dll

\* php.ini主配置文件 - 配置相关数据库库文件

\* 连接MySQL数据库的步骤

\* 与MySQL数据库建立连接

mysqli\_conntect(host,username,passwd,dbname,port)

\* host - MySQL数据库所在电脑的IP地址

\* username - 登录MySQL数据库的用户名

\* passwd - 登录MySQL数据库的密码

\* dbname - 登录的数据库名称

\* port - MySQL数据库的端口号

\* 该方法返回MySQL数据库的连接对象$conn

\* 定义SQL语句

\* 在PHP代码定义SQL语句 - 字符串

\* 如果SQL语句中涉及字符串类型的话,记得加''

\* 将SQL语句发送到MySQL数据库执行

mysqli\_query(link,sql)

\* link - 与MySQL数据库的连接对象

\* sql - 定义的SQL语句

\* 如果执行的是SELECT语句的话,返回结果集对象

\* 类型为mysqli\_result类型

\* 属性

\* num\_rows - 记录数

\* field\_count - 字段数

\* 方法

\* mysqli\_fetch\_array()

\* mysqli\_fetch\_object()

\* 关闭与MySQL数据库的连接

mysqli\_close(link);

SERVER&HTTP DAY03:

\* HTTP

\* URL概念

\* URL - 统一资源定位符

\* URI - 统一资源标识符

\* URL与URI的区别

\* 所有的URL都是URI,但URI不一定是URL

\* 完整的URL

网络协议:IP地址(虚拟地址):端口号/路径;参数?查询数据#锚点

http://127.0.0.1:8080/11\_SERVER&HTTP;JSESSION=123123213?key=value#mylink

ftp://code.tarena.com:80/14\_SERVER&HTTP;JSESSION=123123213?key=value

\* 一般实际使用时

\* http://www.baidu.com/news

\* http://127.0.0.1:8888?key=value

\* http://localhost:80#mylink

\* 具有哪些元素

\* 网络协议

\* IP地址 - 每个电脑都具有

0.0.0.0

\* 端口号 - 每个电脑开放给外部电脑访问接口

\* 最常用的端口号 - 80

\* 数据库的端口号 - 3306

\* 路径 - 绝对路径|相对路径

\* 绝对路径

\* 无论当前位置在哪,目标元素的路径是固定不变的

\* 相对路径

\* 根据当前位置的变化,路径也会变化

\* 参数

\* 查询数据

\* 格式为key=value&key=value

\* 锚点

\* 网络协议

\* 网络协议 - 客户端与服务器端之间的协议

\* 主流协议

\* http协议 - https加密(openssl)协议

\* socket协议 - HTML5的web socket ws://

\* ftp协议、pop3协议等

\* Http协议

\* Http协议 - 多用于B/S架构

\* 一些问题

\* 短连接

\* 每次客户端与服务器端交互时

\* 先建立连接

\* 交互完毕后,关闭链接

\* 无状态

\* 服务器端只能记得住当次请求状态

\* 版本历史

\* HTTP 1.0 - 目前主流版本

\* HTTP 1.1 - 较新版本(向下兼容)

\* HTTP 2.0 - 最新版本

\* 请求(Request)协议

\* GET请求方式 - server.php?user=zhangwuji

\* 请求行

\* 请求类型 - GET

\* 请求地址 - URL?请求参数

\* 状态码 - 服务器端的状态

\* 协议版本

\* 请求头

\* 请求头的格式

\* key : value

\* key : value,value,value,...

\* 请求头的含义

\* Accept - 表示服务器端接受的MIME类型

\* Accept-Encoding - 表示服务器端是否接受压缩

\* gzip - 是指一种服务器端的压缩格式

\* 问题 - 客户端请求的数据内容越大

\* 对网络带宽的要求越高,流量占用大

\* 用户体验不好 - 速度慢、对服务器造成的压力大

\* Accept-Language - 表示服务器端接受的语言

\* zh-CN - 简体中文

\* zh-TW - 繁体中文

\* zh - 中文

\* us - 英文

\* Connection - 表示当前的连接状态

\* keep-alive - 表示保持连接

\* Host - 表示当前电脑的地址(IP:端口号)

\* Referer - 表示当前的请求来源于哪里

http://127.0.0.1:8888/http/index.html

\* 实现防盗链接

\* User-Agent - 获取用户的浏览器信息

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.90 Safari/537.36

Mozilla/5.0 (Windows NT 6.1; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0

Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko

\* Cookie - 将Cookie自动携带请求头

\* 请求体 - 空

\* POST请求方式

\* 请求行

\* 请求类型 - POST

\* 请求地址 - URL

\* 状态码 - 200 OK

\* 协议版本

\* 请求头

\* Cache-Control - 缓存控制

\* max-age=0 - 设置缓存最大活动周期

\* 0 - 表示没有缓存

\* 设置缓存保存的最大时间的单位为毫秒/秒

\* Content-Length - 请求数据的长度(大小)

\* Content-Type - 请求的MIME类型

\* application/x-www-form-urlencoded

\* <form>元素提交请求时默认的类型

\* 一般文件上传时,类型一定这种格式

\* 请求体

\* 请求参数

\* 响应(Response)协议

\* 响应行

\* 状态码

\* 协议版本

\* 响应头

\* Connection - 表示当前的连接状态

\* keep-alive - 表示保持连接

\* Content-Length - 响应数据的长度(大小)

\* Content-Type - 响应数据的MIME类型

\* 一般情况下 - text/html格式

\* 设置响应页面编码 - charset=UTF-8

\* Date - 响应的日期时间

Thu, 19 Nov 2015 08:17:03 GMT

yyyy-MM-dd hh:mm:ss

yyyy年MM月dd日 hh:mm:ss

\* Keep-Alive - 设置保持连接的超时和最大存活时间

\* timeout=5, max=99

\* 一般都是在Connection的值设置为keep-alive时

\* Server - 响应服务器端的信息

Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.6.3

\* 响应体

\* 服务器端向客户端进行响应的数据内容

\* HTTP基本优化

\* 域名解析

\* 在HTML页面尽量减少外部链接

\* <a hred=""></a>

\* <img src="" /> - HTML5的Canvas|SVG

\* <script src=""></script>

\* <link href="" />

\* <video src=""></video>

\* 尽量减少与其他网址交互

\* 创建连接

\* 尽量减少客户端与服务器端之间的交互

\* 减少请求的次数

\* 使用Ajax异步交互

\* 使用例如socket这种协议(实时连接)

\* 发送请求

\* 请求数据进行压缩

\* 等待响应

\* 主要从服务器端进行优化

\* 将响应数据进行压缩

\* 将缩短等待响应的时间 - timeout

\* 浏览器对HTML|CSS的解析,对javascript的解析

\* 安全的HTTP协议

\* HTTP协议

\* http://

\* 默认端口号为80

\* HTTP安全协议 - OpenSSL安全协议

\* https://

\* 默认端口号为443

\* 举例

\* 电商网站

\* 银行网站

\* ...

\* 扩展内容

\* 请求类型

\* GET - 最常使用的

\* POST - 最常使用的

\* HEAD

\* PUT

\* DELETE

\* OPTIONS

\* TRACE

\* 请求类型的面试题

\* 请求类型只有两种(错)

\* 请求类型常用的有两种(对)

\* 请求类型共有七种(错)

\* 请求类型至少有七种(对)

\* 标准API - 标准的请求方式

\* GET - 获取数据

\* POST - 修改数据

\* PUT - 新增数据

\* DELETE - 删除数据

\* 状态码

\* 1xx - 服务器端的信息

\* 一般实际开发中很少出现

\* 2xx - 表示请求成功

\* 200 - 请求成功

\* 201 - 创建连接

\* 202 - 接收请求

\* 3xx - 表示重定向

\* 302 - 表示重定向

\* 304 - 表示服务器端资源没有变化,访问缓存

\* 问题 - 这次请求没有真正地访问服务器端

\* 305 - 表示使用代理

\* 4xx - 客户端错误

\* 400 - 错误的请求

\* 403 - 拒绝访问

\* 404 - 网页找不到

\* 405 - 请求类型不允许

\* 5xx - 服务器端错误

\* 500 - 服务器端错误

\* 502 - 路径错误

\* 504 - 请求超时

\* 505 - HTTP版本不支持

\* MIME类型 - 文件类型

\* MIME类型是如何定义的

// 定义MIME类型

<mime-mapping>

// 文件的扩展名

<extension>jpg</extension>

// 对应MIME类型

<mime-type>image/jpeg</mime-type>

</mime-mapping>

\* 常见的MIME类型

\* html - text/html

\* htm - text/html

\* xhtml- application/xhtml+xml

\* css - text/css

\* js - application/javascript text/javascript

\* json - application/json

\* jpg - image/jpeg

\* jpeg - image/jpeg

\* png - image/png

\* text - text/plain

\* webm - video/webm

\* mp4 - video/mp4

\* 请求参数

\* GET请求类型

\* 将请求数据 - URL?key=value(浏览器地址栏)

\* 安全相对比较低

\* 请求地址的长度是有限制的

\* 请求数据内容过多

\* 如果请求数据中包含中文的话,需要转码的

\* 更容易出现中文乱码问题

\* POST请求类型

\* 浏览器地址栏URL,不包含请求类型

\* 安全相对比较高

\* 请求数据在请求体中

\* 对请求数据的大小(长度)没有要求

\* 如果请求数据中包含中文的话,可以不转码

\* 相对来讲对于中文的处理比较好

\* unicode码

\* 金云龙 - %E9%87%91%E4%BA%91%E9%BE%99

\* 将中文内容转换成unicode码

\* 服务器端接收后,将unicode码转换为中文

\* Web应用的中文乱码问题

\* GET与POST请求方式的请求协议的区别

\* GET请求方式

\* 请求行

\* 请求类型 - GET

\* 请求地址 - URL?请求参数

\* 请求体 - 空

\* POST请求方式

\* 请求行

\* 请求类型 - POST

\* 请求地址 - URL

\* 请求体

\* 请求参数

\* 头信息

\* 通用(请求与响应)头信息

\* Connection

\* Content-Length

\* Content-Type

\* Date

\* 特殊响应头

\* 控制是否缓存的头信息

\* Cache-Control - 0 表示不缓存

\* Expires - 0 表示不缓存

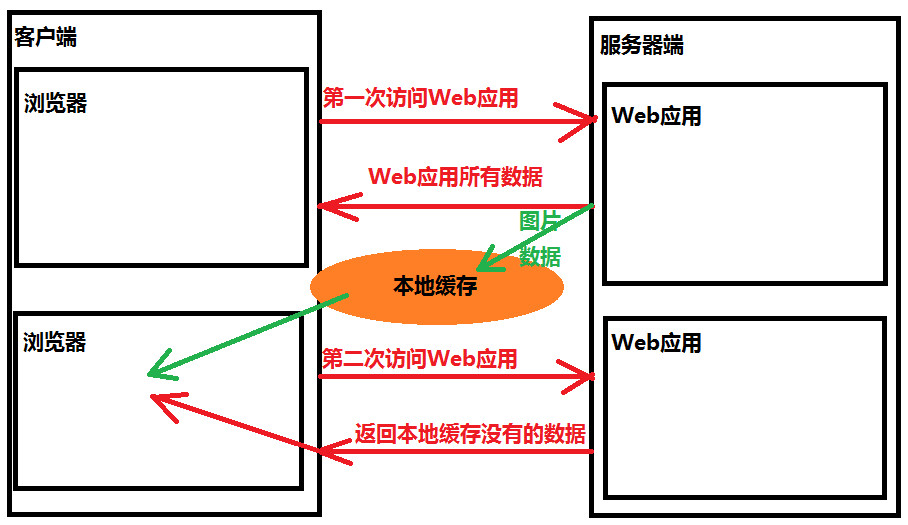
\* 注意 - 是允许缓存

\* 控制头信息

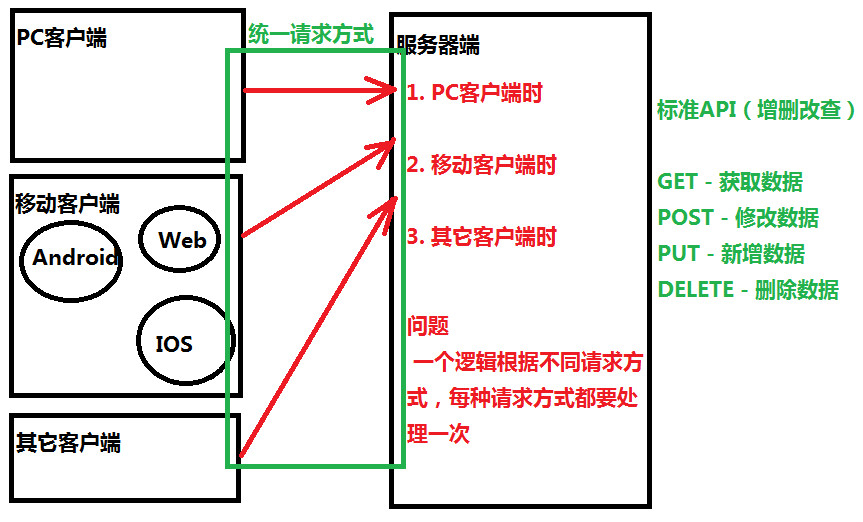
\* HTML页面 - 使用<meta />元元素

\* PHP页面 - header()函数设置头信息

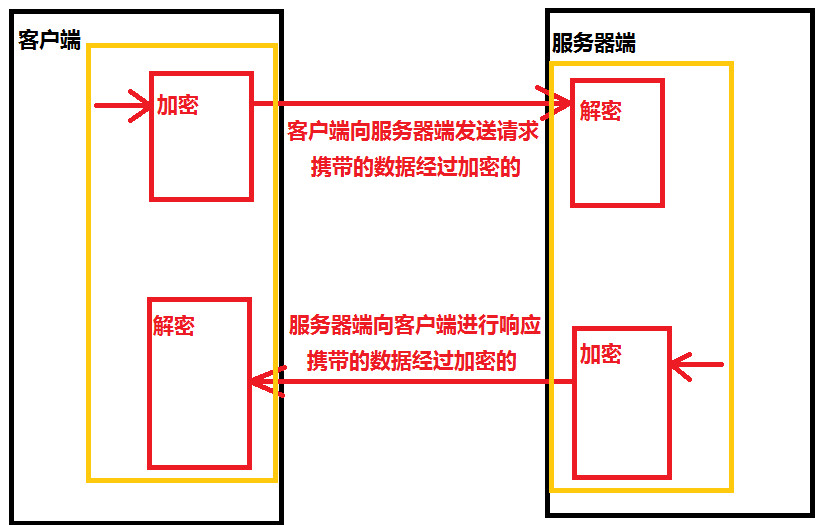
**本地缓存的实现原理：**



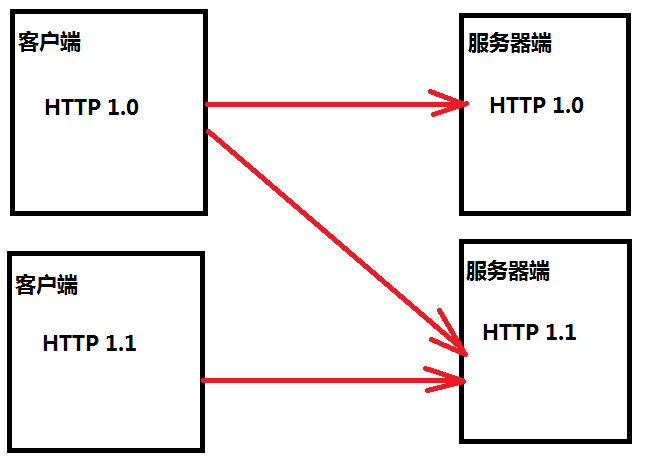
**标准式API的实现原理：**



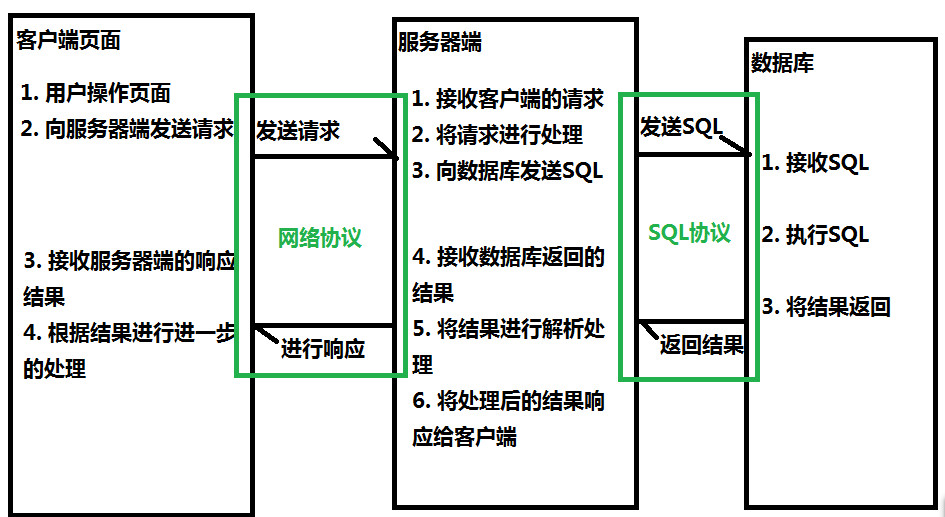
**加密协议的传输过程：**



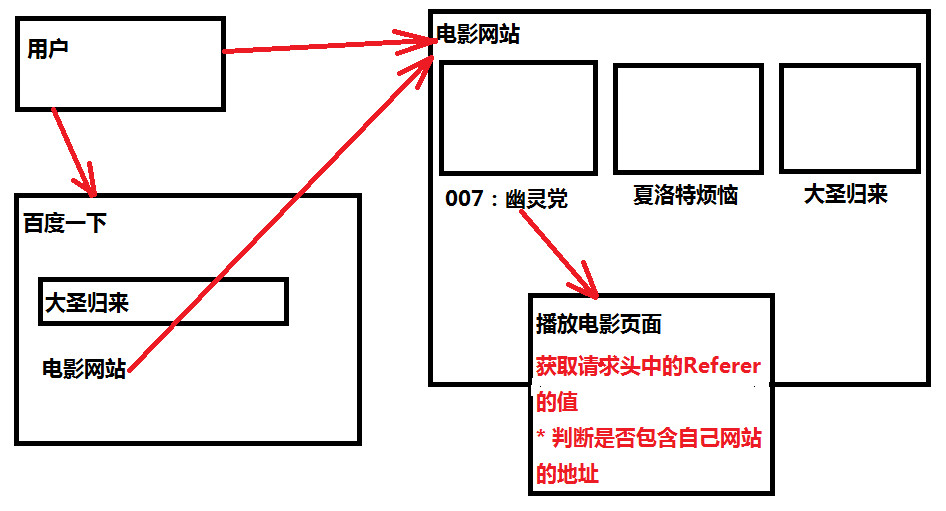
客户端与服务器端的HTTP版本对应：



**完整的Web应用执行流程图：**



网站防盗链接的原理分析：



**AJAX**

AJAX DAY01:

\* 基本内容

\* 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,到服务器端进行响应,这个过程中,用户是不能做任何其他事情(等).

\* 异步交互

\* 客户端向服务器端发送请求，直到服务器端进行响应，这个过程中，用户可以做任何其他事情(不等)

\* AJAX

\* Asynchronous JavaScript and Xml

\* 直译中文 - javascript和XML的异步

\* (不严格的定义)客户端与服务器端进行交互,而无需刷新当前页面的技术,称之为Ajax

\* Ajax实现的是B/S架构下的异步交互

\* 实现异步交互的技术

\* <iframe src="">元素

\* 同步与异步的区别

\* 同步交互

\* 执行速度相对比较慢

\* 响应的是完整的HTML页面

\* 异步交互

\* 执行速度相对比较快

\* 响应的是部分数据

\* Ajax具有核心对象

\* XMLHttpRequest对象

\* 创建XMLHttpRequest对象

function getXhr(){

// 声明XMLHttpRequest对象

var xhr = null;

// 根据浏览器的不同情况进行创建

if(window.XMLHttpRequest){

// 表示除IE外的其他浏览器

xhr = new XMLHttpRequest();

}else{

// 表示IE浏览器

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

}

return xhr;

}

// 创建核心对象

var xhr = getXhr();

\* Ajax的核心对象

\* 属性

\* readyState

\* status

\* 方法

\* open() - 与服务器端建立连接

\* send() - 向服务器端发送请求

\* 事件

\* onreadystatechange事件

\* 作用 - 监听服务器端的通信状态改变

\* 实现Ajax的异步交互步骤

\* 创建XMLHttpRequest核心对象

\* 固定写法 - 独立编写

\* 与服务器端建立连接

\* 使用XMLHttpRequest对象的open(method,url)方法

\* method - 设置当前请求的类型

\* url - 设置当前请求的地址

\* 向服务器端发送请求

\* 使用XMLHttpRequest对象的send(请求参数)方法

\* 请求参数的格式 - key=value

\* 接收服务器端的响应数据

\* 使用XMLHttpRequest对象的onreadystatechange事件,监听服务器端的通信状态

\* 使用XMLHttpRequest对象的readyState属性,判断服务器端的当前状态(0-4)

\* 使用XMLHttpRequest对象的status属性,判断服务器端的状态码(200)

\* 使用XMLHttpRequest对象的responseText属性,接收服务器端的响应数据

\* GET与POST的区别

\* GET请求类型

\* send()方法不起作用,但是不能被省略

xhr.send(null);

\* 请求参数 - 添加到URL?key=value

\* POST请求类型

\* send()方法起作用

\* 在send()方法被调用前,使用setRequestHeader()方法设置请求头信息

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

\* 使用Ajax的原则

\* 小则怡情,大则伤身

\* 另外的实现Ajax的步骤

\* 创建XMLHttpRequest对象

\* 注册监听

\* 建立连接

\* 发送请求

\* 实现Ajax的六步是什么?(不标准)

\* 创建XMLHttpRequest对象

\* 建立连接

\* 发送请求

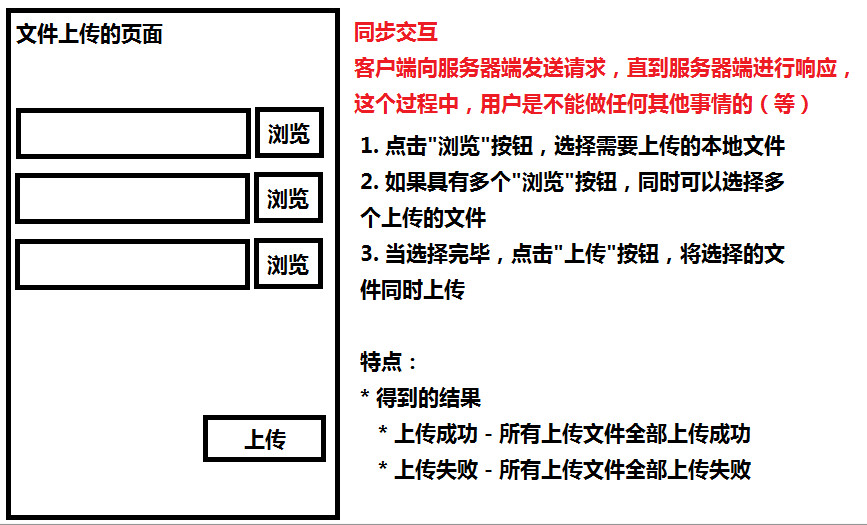
\* 注册监听 - onreadystatechange事件

\* 获取服务器端的通信状态 - readyState

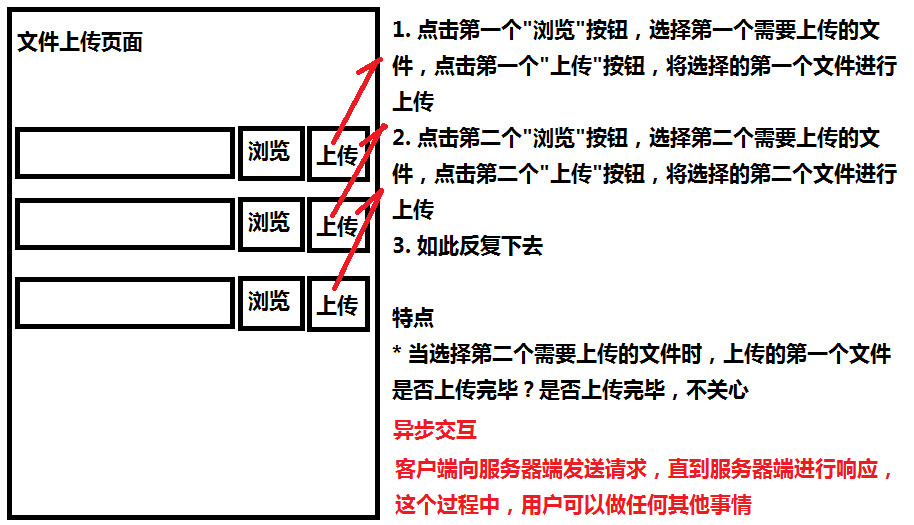
\* 获取服务器端的状态码

\*

**同步交互举例分析：**



异步交互举例分析：



***回顾AJAX第一天内容:***

\* 基本内容

\* 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，直到服务器端进行响应，这个过程中，用户不能做任何其他事情

\* 异步交互

\* 客户端向服务器端发送请求，直到服务器端进行响应，这个过程中，用户可以做任何其他事情

\* Ajax

\* Ajax实现B/S架构下的异步交互

\* (不严格定义)客户端向服务器端发送请求,而无需刷新当前页面的技术

\* 实现异步交互的其他技术

\* <iframe src="">元素

\* Ajax的核心对象 - XMLHttpRequest对象

\* XMLHttpRequest对象

\* 创建核心对象

function getXhr(){

var xhr = null;

if(window.XMLHttpRequest){

xhr = new XMLHttpRequest();

}else{

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

}

return xhr;

}

\* 属性

\* readyState属性 - 表示当前服务器端的通信状态

\* 0 - (服务器端)尚未初始化

\* 1 - (服务器端)正在接收

\* 2 - (服务器端)接收完毕

\* 3 - (服务器端)正在响应

\* 4 - (服务器端)响应完毕

\* status属性 - 表示当前服务器端的状态码

\* 200 - 请求成功

\* 404 - 网页找不到(请求路径不正确)

\* 500 - 服务器端错误

\* 方法

\* open(method,url)方法

\* 作用 - 与服务器端建立连接

\* 参数

\* method - 设置请求类型(GET或POST)

\* url - 设置请求地址

\* send()方法

\* 作用 - 向服务器端发送请求参数

\* 事件

\* onreadystatechange事件

\* 作用 - 监听服务器端的通信状态

\* 实现Ajax的步骤

\* 创建XMLHttpRequest对象

var xhr = getXhr();

\* 与服务器端建立连接 - open()

\* GET方式 - open("get")

\* POST方式 - open("post")

\* 向服务器端发送请求 - send()

\* GET方式

\* send()方法不起作用(不能使用send()方法发送请求数据)

\* send()方法不能被省略 - send(null)

\* 请求数据 - 增加在URL?key=value

\* POST方式

\* send()方法起作用

\* 注意

\* 在send()方法调用前,调用setRequestHeader()方法

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

\* 接收服务器端的响应

\* 使用onreadystatechange事件监听服务器端

\* 判断readyState属性值等于4(响应完毕)

\* 保证status属性值为200(表示请求成功)

\* 使用responseText属性接收服务器端的响应数据

\* 面试题

\* 另外四步实现

\* 创建核心对象

\* 注册监听

\* 建立连接

\* 发送请求

\* 实现Ajax的六步是什么?

***AJAX DAY02:***

\* 接收服务器端的响应数据

\* 使用XMLHttpRequest核心对象的responseText属性

\* 该属性只能接收文本(HTML)格式

\* 问题

\* 解析过程比较复杂(拆串)

\* 拆串或拼串极容易出错

\* XML格式

\* 基本内容

\* HTML、XHTML、DHTML和XML的区别

\* HTML就是网页 - 元素定义大小写

\* XHTML就是严格意义的HTML - 元素定义小写

\* DHTML - BOM|DOM

\* XML - 配置文件|数据格式

\* XML文件的扩展名为".xml"

\* XML的定义方式与HTML非常相似

\* HTML的元素预定义好的

\* XML允许自定义元素

\* XML的版本

\* 1.0版本 - 目前唯一版本

\* 1.1版本 - 几乎没有人使用

\* 注意

\* 版本不会再更新

\* XML的作用

\* 作为数据格式 - 存储数据

\* XML语法

\* 声明

<?xml version="1.0" encoding="UTF-8" ?>

\* version - 设置当前XML文件的版本

\* encoding - 设置当前XML文件的编码

\* 注意 - 出现在0行0列上

\* 定义元素

\* 根元素

\* 必须是起始标签

\* 只能定义一个

\* 定义元素

\* 元素名可以自定义

\* 分类

\* 起始标签或单标签

\* 定义属性

\* 定义注释

\* 练习 - 使用XML文件定义省份和城市信息

\* 如何定义都可以,符合XML语法即可

\* 决定着后面使用Javascript解析的难易程度

\* DOM解析XML

\* 创建XML的解析器

function parseXML(xml){

// 声明解析XML后的DOM对象

var xmlDoc = null;

// 根据不同的浏览器

if(window.DOMParser){

// 其他浏览器

var parser = new DOMParser();

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

}else{

// IE浏览器

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

xmlDoc.async = false;

xmlDoc.loadXML(xml);

}

return xmlDoc;

}

\* 解析XML与解析HTML一致

\* 很少使用ById和ByName两个方法

\* 注意

\* 浏览器不允许读取外部的XML文件

\* 浏览器解析符合XML格式的字符串

\* Ajax中的XML格式

\* 请求的数据格式 - XML

\* 客户端如何构建符合XML格式的数据

\* 构建的数据类型 - 字符串(string)类型

\* 字符串的内容符合XML格式的语法要求

\* 服务器端如何接收符合XML格式的数据

\* 接收客户端的请求数据 - 字符串(string)类型

\* PHP集成了DOM的相关内容

\* DOMDocument类

\* DOMElement类

\* DOMNode类

\* 响应的数据格式 - XML

\* 服务器端如何构建符合XML格式的数据

\* 设置服务器端的响应头"Content-Type"值为"text/xml"

\* 构建符合XML格式数据内容

\* 手动方式构建字符串(string)内容

\* DOMDocument对象的方法

\* loadXML(符合XML格式的字符串);

\* saveXML()方法进行响应

\* 客户端如何接收符合XML格式的数据

\* 使用XMLHttpRequest对象的responseXML属性接收

\* 接收回来的就是XML DOM对象(不需要使用XML解析器进行解析)

\* 直接使用DOM解析XML DOM对象即可

\* 练习 - 二级联动(服务器端响应数据格式为XML格式)

\* JSON格式

\* 基本内容

\* JSON - JavaScript Object Notation(JS原生支持)

\* JSON数据格式源于javascript

\* 特点

\* 易于程序员阅读和编写

\* 易于计算机解析和生成

\* JSON目前是网络上使用最广泛的数据格式之一

\* JSON的结构

\* Array - 数组

\* Object - 对象

\* 支持的数据类型

\* String字符串

\* Number数值

\* Boolean - true|false

\* Object

\* Array

\* null

\* Ajax中的JSON格式

\* 请求格式为JSON

\* 客户端向服务器端发送请求为JSON格式的数据

\* 构建符合JSON格式的字符串

\* 保证定义字符串时,使用单引号(里面使用双引号)

\* 服务器端接收JSON格式的数据

\* 接收客户端的数据

\* 使用json\_decode()函数进行解析

json\_decode($json,true)

\* 响应格式为JSON

\* 服务器端向客户端发送响应为JSON格式的数据

\* 使用json\_encode()函数将PHP变量(array)转换成符合JSON格式的字符串

\* 客户端接收JSON格式的数据

\* 使用XMLHttpRequest对象的responseText属性接收

\* 没有responseJSON属性

\* 使用eval()函数进行转换

\* 作业 - 使用JSON格式完成二级联动

\* HTML(文本格式)、XML格式及JSON格式的优缺点

\* HTML格式

\* 优点 - 简单

\* 缺点 - 解析复杂

\* XML格式

\* 优点 - 易于构建复杂数据

\* 缺点 - 构建、解析复杂

\* JSON格式

\* 优点 - 轻量级

\* 缺点 - 可能转换失败

\* 回顾内容

\* DOM - 独立于任何开发语言的

\* DOM的分类

\* DOM CORE

\* DOM HTML

\* DOM XML

\* DOM CSS

\* DOM API

\* 获取元素

\* 创建元素

\* 替换元素

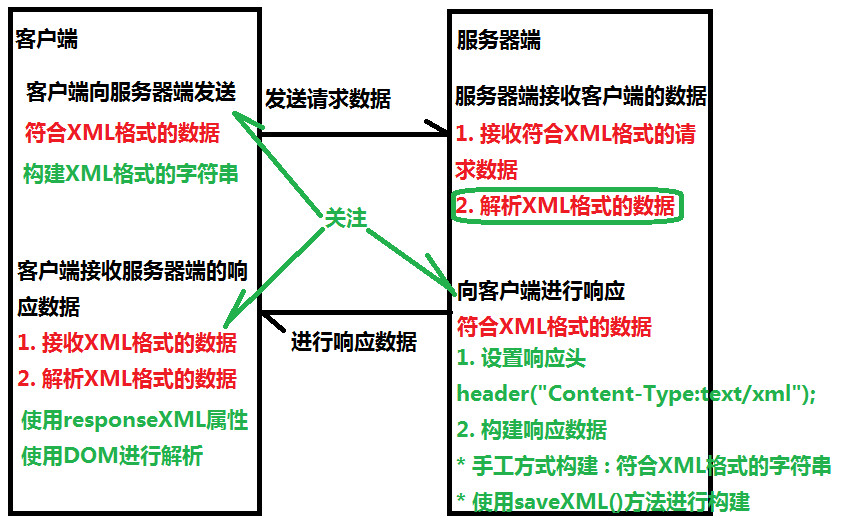
\* 删除元素

\* 插入元素

\* ...

\*

AJAX中使用XML数据格式：



***回顾AJAX第二天内容:***

\* XML格式

\* 基本内容

\* XML译为可扩展标记语言

\* XML文件的扩展名为".xml"

\* XML定义元素与HTML页面类似

\* HTML页面的元素是预定义的

\* XML的元素是自定义的

\* XML的用途

\* 配置文件

\* 存储数据

\* XML基础语法

\* 声明

<?xml version="1.0" encoding="utf-8"?>

\* version - 设置当前XML文档的版本

\* encoding - 设置当前XML文档的编码

\* 注意 - 必须出现在0行0列

\* 定义元素

\* 起始元素 - <></>

\* 单元素 - < />

\* 定义属性

\* 使用注释

\* DOM解析XML

\* 创建XML解析器

function parseXML(xml){

// 解析后得到的XML DOM对象

var xmlDoc = null;

// 根据不同浏览器操作

if(window.DOMParser){// 其他浏览器

// 创建XML解析器

var parser = new DOMParser();

// 通过解析器得到DOM对象

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

}else{// IE浏览器

// 得到XML DOM对象

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

// 关闭异步加载

xmlDoc.async = false;

// 指定加载XML

xmlDoc.loadXML(xml);

}

// 返回XML DOM对象

return xmlDoc;

}

\* 利用DOM API解析XML - 类似于DOM解析HTML

\* Ajax中的XML格式

\* 客户端向服务器端发送请求 - XML格式

\* 客户端如何构建XML格式数据

\* 请求数据格式 - key=value

\* 构建符合XML格式的字符串(string)

\* 服务器端如何接受XML格式数据

\* 利用$\_GET或$\_POST接收请求数据

\* PHP集成了DOM内容

\* DOMDocument对象

\* DOMElement对象

\* DOMNode对象

\* 服务器端向客户端发送响应 - XML格式

\* 服务器端如何构建XML格式的数据

\* 手工方式构建符合XML格式的字符串

\* 通过saveXML()方法将PHP变量构建成符合XML格式

\* 客户端如何接收XML格式的数据

\* 使用responseXML属性接收

\* 利用DOM进行解析

\* JSON格式

\* 基本内容

\* JSON - JavaScript Object Notation

\* 定义

\* 是一种轻量级(相对于XML)的数据交换格式

\* JSON源于JavaScript,原生JavaScript支持JSON

\* 特点

\* 易于程序员阅读和编写

\* 易于计算机生成和识别

\* JSON的结构

\* Array - 数组

\* Object - 对象

\* 注意

\* 数组和对象之间可以相互嵌套(多层嵌套)

\* 数据类型

\* String

\* Number

\* Boolean

\* Array

\* Object

\* null

\* Ajax中的JSON格式

\* 客户端向服务器端发送请求 - JSON

\* 客户端如何构建JSON格式数据

\* 请求数据格式 - key=value

\* 手工构建符合JSON格式的字符串

'{"name":"zhangwuji","pwd":"123"}'

\* 服务器端如何接收JSON格式的数据

\* 利用$\_GET或$\_POST接收请求数据

\* 使用json\_decode()函数将符合JSON格式的字符串,转换为PHP变量

json\_decode($json,true) - array数组

\* 服务器端向客户端进行响应 - JSON

\* 服务器端如何构建JSON格式的数据

\* 手工方式构建符合JSON格式的字符串

\* 使用json\_encode()函数将PHP变量转换为符合JSON格式的字符串

\* 客户端如何接收JSON格式的数据

\* 使用responseText属性接收数据

\* 使用eval()函数进行转换

var json = eval("("+data+")");

***AJAX DAY03:***

\* jQuery中的Ajax

\* 封装第一层 - 类似于原生Ajax的用法

\* $.ajax() - 最复杂

\* 选项

\* url - 请求地址

\* type - 请求类型,默认为GET

\* async - 是否异步,默认为true

\* data - 请求数据,格式为key:value

\* dataType - 响应数据格式

\* HTML格式

\* XML格式

\* JSON格式

\* success - 请求成功后的回调函数

function(data,textStatus){}

\* data - 响应数据内容

\* textStatus - success,请求状态

\* error - 请求失败后的回调函数

function(XMLHttpRequest,textStatus,errorThrown){}

\* XMLHttpRequest - Ajax的核心对象

\* textStatus - 请求状态

\* error、timeout及notmodified等

\* errorThrown - 错误异常信息

\* 封装第二层 - 基于第一层再次封装

\* $().load(url,data,callback) - 最简单、局限性最大

\* 参数

\* url - 设置当前Ajax的请求地址

\* data - 设置当前Ajax的请求数据

\* 格式要求为key:value,构建Object即可

\* callback - Ajax请求成功后的回调函数

\* 该回调函数的形参(data),表示服务器端响应的数据内容

\* 问题

\* 请求类型由是否发送请求数据决定

\* 没有发送请求数据时,请求类型为GET

\* 如果发送请求数据时,请求类型为POST

\* 默认接收服务器端的数据内容

\* 是以字符串类型(HTML格式)进行接收

\* 无法使用XML格式或JSON格式

\* $.get(url,data,callback,type) - 请求类型是GET

\* 参数

\* url - 设置当前Ajax的请求地址

\* data - 设置当前Ajax的请求数据

\* 格式要求为key:value,构建Object即可

\* callback - Ajax请求成功后的回调函数

\* 该回调函数的形参(data),表示服务器端响应的数据内容

\* type - 设置服务器端响应数据的格式

\* 默认值 - HTML格式

\* xml - XML格式

\* json - JSON格式

\* $.post() - 请求类型是POST

\* 使用方式与$.get()方法一致

\* 封装第三层 - 特殊用法

\* $.getScript(url,callback) - 动态读取脚本(JavaScript代码)

\* url - 读取脚本的地址(本地或服务器)

\* callback - 读取成功后的回调函数

\* $.getJSON() - 接收JSON格式数据

\* 表单的Ajax异步请求

\* 表单的序列化

\* serialize()方法

\* 返回JSON字符串

\* 格式 - {key:value,key:value,..}

\* serializeArray()方法

\* 返回JSON对象

\* 格式 - [obj1,obj2,obj3,...]

\* 注意

\* 表单元素必须具有name属性值

\* jQuery.form插件

\* 作用 - 实现表单的异步提交

\* 两个核心方法

\* ajaxForm()方法

\* ajaxSubmit()方法 - 使用Ajax异步提交表单

\* 底层机制

\* 表单提交机制

\* 表单异步提交的方式

\* 不再使用submit按钮,而是使用button按钮

\* 通过为button按钮绑定click事件

\* 表单的序列化

\* 表单的异步提交

\* (常用)依旧使用submit按钮

\* 在<form>元素绑定onsubmit事件

\* 在onsubmit事件的处理函数中

\* 表单的序列化

\* 表单的异步提交

\* 阻止表单默认行为(return false)

\* 跨域请求 - $.getJSON()方法

\* 跨域

\* 完全跨域 - IP不同

\* http://www.baidu.com

\* http://www.tedu.cn

\* 跨子域 - IP相同,但端口号不同

\* http://127.0.0.1:8888

\* http://127.0.0.1:9999

\* 域名

\* 顶级域名

\* http://www.baidu.com

\* 二级域名

\* http://wenku.baidu.com

\* http://www.baidu.com/kongjian

\* 万维网协议

\* 默认是不允许跨域请求的

\* 实现跨域

\* Cookie

\* 基本内容

\* 浏览器的缓存

\* 存储在浏览器内存中

\* 关闭浏览器(窗口)后,数据会被自动销毁

\* 存储在用户电脑硬盘中

\* 关闭浏览器(窗口)后,数据不会被销毁

\* Cookie(小甜饼)

\* 缓存文件 - 一些用户数据存储在此

\* 问题

\* 用户数据是以明码(明文)来存储的

\* Cookie谁都可以读取

\* A网站生成的Cookie文件,归属到百度

\* 如何操作操作

\* 读取Cookie - 将用户名或密码从Cookie读取

\* document.cookie

\* 写入Cookie - 将用户名或密码写入到Cookie文件

\* 格式 - key=value;expires=时间

\* document.cookie

Bootstrap

李文华

阶段课程安排：

1. Bootstrap（4+2）—— 背单词
2. AngularJS（3+2）—— 思路转变

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

1. WebApp（3+2）—— jQueryMobile 玩儿
2. 微信（2）—— 玩儿
3. 其他框架（2个0.5） ExtJS —— 玩儿
4. 面试题——什么是响应式网页

2G：

2.5G：GPRS 塞班、黑莓 WirelessML

3G： iOS、Android——Linux/浏览器 HTML

响应式/自适应网页：一个页面，可以在各种不同的设备下浏览，都有不错的浏览体验。响应式网页的三个特征：

(1)流式布局

(2)可伸缩的图片和字体

(3)CSS3 Media Query

|  |
| --- |
| 历史上(2009年左右)，HTML网页基本都是为PC设计的，宽度比较大。  iOS/Safari可以显示这些为PC而编写的网页——把PC网页缩小后放到手机屏幕上——文字/图片都会强制缩小，影响浏览体验。  iOS引入了viewport（视口）的概念：用于显示网页的内容的一个逻辑概念，其宽度/高度都可以任意指定，可以大到2000px都可以，网页不是仅仅显示在物理窗口，而是显示视口中——就可以实现大网页不经缩放，直接显示在手机中——只是需要用户左右滑动窗口即可。后来Android也引入了该概念。 |

1. 如何编写响应式网页——面试题

(1)设置viewport元标签

<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">

(2)避免使用绝对单位(px)，用相对单位代替(%、auto、em等)

(3)使用流式定位：float

(4)图片大小实现自适应：

img { max-width: 100%; }

会随着容器的改变而改变，且不会超过图片自身原始大小，防止失真。

**(5)根据浏览器屏幕的特征，有选择性的执行某些CSS样式——CSS3媒体查询技术(Media Query)**

1. 如何测试响应式网页

(1)使用真实物理设备——效果可靠但任务量太大

只要手机/平板/PC在同一个局域网/互联网即可测试。

(2)使用第三方测试/模拟软件——效果有待进一步的验证

(3)使用Chrome自带的浏览器模拟器测试

优势：可以模拟常见的设备、网速、经纬度、加速度....

不足：效果有待进一步的验证

1. WS的常用操作：

常用快捷键(Keymap/Eclipse)：

复制当前行： Ctrl+Alt+↓

向上/下移动当前行： Alt+↑/↓

删除当前行： Ctrl+D

注释/取消当前行： Ctrl+/

重新格式化当前文档： Ctrl+Alt+L

ZenCoding（代码补全）：

div+TAB

div.box+TAB

div#mybox+TAB

div\*3+TAB

ul>li\*3>a+TAB

假文生成：

lorem+TAB

多行编辑：

Alt+左键点击 开始多行同时编辑

ESC 退出多行编辑模式

5.CSS3提供的Media Query技术

作用：根据客户端浏览设备的特性，有选择性的执行部分CSS

Media：指浏览网页的设备，如screen(pc/pad/phone)、print、tv、projection、屏幕阅读器.....

Query：查询出当前浏览设备的特性，如类型、宽度、高度、分辨率、色彩位深、方向Orientation(Landscape/Portrait)，根据这些特性，执行特定的CSS样式。

CSS3MediaQuery有两种用法：

**(1)根据媒体的特性，执行不同的外部CSS：**

<link media="screen and (min-width:768px) and (max-width: 990px)" rel="stylesheet" href="xx.css">

不足：客户端会不管媒体特性，请求所有的CSS文件。

**(2)根据媒体的特性，执行某段CSS中的部分内容：**

@media screen and (min-width:768px) and (max-width:990px) {

h1 { ... }

.box { ... }

}

1. Twitter Bootstrap

Bootstrap( bootcss.com )是一个框架HTML/CSS/JS框架，适用于移动设备优先的响应式网页。

Bootstrap分为五部分：

1. 起步(Startup)
2. 全局CSS样式(Global CSS)
3. 组件(Component)
4. 插件(Plugin)
5. 定制(Customize)
6. Bootstrap第一部分：起步

(1)下载Bootstrap

(2)安装Bootstrap

(3)网页基本模板

|  |
| --- |
| lang属性的两个作用：(1)告诉浏览器翻译时如何确定当前网页的基础语言 (2)告诉读屏软件当前页面的基础发音 |

|  |
| --- |
| IE浏览器的兼容性问题：  X-UA-Compitable: Cross-UserAgent-Compatible，该元标签只有IE浏览器支持。  <meta http-equiv="X-UA-Compatible" content="IE=edge">  设置IE的兼容模式为edge——最新版，尽可能向行业标准看齐。  IE 6  IE 7  IE 8  IE 9  IE 10  IE 11 |
| Windows10 IE彻底被抛弃，新浏览器命名为 Edge |

|  |
| --- |
| **html5shiv.js**  由@afarkas @jdalton @jon\_neal @rem编写的一个JS自调用脚本，用于让老IE支持H5新标签。 |
| **respond.js**  由Scott编写的一个JS自调用脚本，用于让老IE浏览器支持CSS3MediaQuery技术，从而实现响应式网页的编写 |

作业：

(1)找一些免费/收费网站空间，上传自己的网页作品，使用PC/手机浏览

(2)完成完整版的响应式网页示例



(3)创建一个Bootstrap页面，查看对下述标签Bootstrap进行了哪些样式重置：

html

body

h1~h6

a

img

table

ul

(4)预习：Bootstrap第二部分：全局样式——按钮相关的样式

复习：

jQuery是一个JS框架；

jQueryUI是一个HTML组件库；

Twitter Bootstrap是一个HTML/CSS/JS框架，适用于移动设备优先的响应式网页。

V2：面向PC进行了样式设定，同时兼顾PAD和PHONE

V3：面向Phone进行样式设定，同时兼顾了PC

主要涉及：

(1)HTML：为H5已有的标签扩展了一些自定义属性 data-\*

(2)CSS： CSS Reset + 几千个class

(3)JS：基于jQuery提供了十几个插件函数

内容分为五部分：

(1)起步：下载、模板、Bootlint、示例、禁用响应式、

(2)全局CSS样式

(3)组件

(4)插件

(5)定制

<div id="" class="" style="" title="" data-old-width="500">

1.复习：使用Bootstrap页面模板

1. Bootstrap提供的CSS Reset

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

body { font ...; color: #333; background: ...; margin: 0;}

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

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

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

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

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

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

a { color:; text-decoration: ;}

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

p { margin-bottom:10px; }

......

|  |
| --- |
| CSS补充知识点：  盒子模型的计算方法 box-sizing  div {  box-sizing: content-box; /\*默认值\*/  box-sizing: border-box; /\*推荐使用\*/  } |
| content-box: 一个盒子的总宽度=margin+border+padding+width  border-box: 一个盒子的总宽度=margin+width |

1. Bootstrap全局CSS样式——按钮——简单&有趣

.btn { padding:; border: ;}

.btn-default { color:; background:; border-color:;}

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

.btn-danger

.btn-success

.btn-warning

.btn-info

.btn-primary

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

.btn-lg

.btn-sm

.btn-xs

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

.btn-block

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

.pull-left { float: left; }

.pull-right { float: right; }

1. Bootstrap全局CSS样式——图片——简单&有趣

.img-rounded

.img-circle

.img-thumbnail 缩略图片/拇指图片

.img-responsive 响应式图片

1. Bootstrap全局CSS样式——排版和代码——仅作了解

.text-danger

.text-success

.text.warning

.text-info

.text-primary

.bg-danger

.bg-success

.bg-warning

.bg-info

.bg-primary

.text-left

.text-right

.text-center

.text-justify 文本两端调整对齐

.text-uppercase

.text-lowercase

.text-capitalize

.list-unstyled

.list-inline

|  |
| --- |
| Bootlint工具:  是一个js，由Bootstrap官方提供，用于检测使用Bootstrap的页面中常见的HTML错误、class使用方面的错误——默认情况下浏览器是检查不出来的。 |

6.Bootstrap全局CSS样式——表格——简单&有趣

.table

.table-bordered 带边框的表格

.table-responsive 响应式表格 **注意：使用在table的父元素上，而不是table上**

.table-striped 隔行变色的表格

.table-hover 带悬停效果的表格

7.Bootstrap全局CSS样式——栅格布局系统——最重点&稍难

Web开发中页面布局可以采用的方式：

(1)使用TABLE做布局

优势：简单不易出错 不足：加载效率

(2)使用DIV+CSS做布局

优势：加载速度快、灵活 不足：不易控制

(3)使用Bootstrap提供的栅格(Grid Layout)布局系统

优势：加载速度快、灵活、支持响应式功能、容易控制（有行/列的概念，但使用DIV+CSS实现）

栅格布局系统的特点：

(1)所有的行必须放在容器中： .container或.container-fluid

(2)分为多行(row)，一行中平均分为12列(col)

(3)网页内容只能放在列(col)中，不能直接放在行(row)

(4)可以在col中再嵌套row

(5)col分为四大类： col-xs col-sm col-md col-lg

(6)col-md-\* \*值可为1~12，值就为某个列的宽度( \*/12 )

(7)可以为一个列指定不同屏幕下的不同宽度

(8) col-lg-\* 只对大PC屏幕有效

col-md-\* 对普通PC和大PC屏幕都有效

col-sm-\* 对平板、PC、大PC屏幕都有效

col-xs-\* 对手机、平板、PC大PC屏幕都有效

(9) .hidden-lg 当前列只在大PC屏幕下隐藏

.hidden-md 当前列只在PC屏幕下隐藏

.hidden-sm 当前列只在平板屏幕下隐藏

.hidden-xs 当前列只在手机屏幕下隐藏

(10) .col-md-offset-1~12——自学

练习：使用栅格系统创建“响应式网页示例”

|  |
| --- |
| .container的宽度问题：  当屏幕宽度>1200px（超大PC显示器-lg）： 容器宽1170px  当屏幕宽度>992px（普通PC显示器-md）: 容器宽970px  当屏幕宽度>768px（平板显示器-sm）: 容器宽750px  当屏幕宽度<768px（手机显示器-xs）: 容器宽auto  .container-fluid的宽度： width: auto; + before + after |
| **面试题：Bootstrap布局系统中容器的特点？**   1. 宽度做了媒体查询。 2. 添加了前置和后置内容生成，可以防止子元素的越界、浮动造成的影响。 |

1. 补充：CSS相关知识

(1)如何解决父元素的第一个子元素的margin-top越界问题

1. 为父元素加border-top: 1px;——有副作用
2. 为父元素指定padding-top: 1px;——有副作用
3. 为父元素指定overflow:hidden;——有副作用
4. 为父元素添加前置内容生成——推荐使用

.parent:before {

content: ' ';

display: table;

}

(2)如何解决所有的子元素浮动后父元素高度变为0，且影响后续元素

1)为父元素指定overflow:hidden;——有副作用

2)为父元素指定高度：height: xxx;——有局限性

3)为父元素添加后置内容生成——推荐使用

.parent:after {

content: ' ';

display: table;

clear: both;

}

9.Bootstrap全局CSS样式——表单——次重点&最难

作业：

1. 使用js为一个div元素声明一个自定义属性，并修改其值？

(2)完整的实现“响应式网页示例”

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网页。

主要分为5部分：

1. 起步：下载 模板 Bootlint 示例 禁用
2. 全局CSS样式：

按钮 .btn .btn-default

图片 .img-rounded .img-circle .img-thumbnail .img-responsive

文本 .text-五种 .bg-五种 .text-left/right/center/justify

排版和代码 .list-unstyled .list-inline

表格 .table .table-bordered .table-responsive .table-striped .table-hover

栅格布局系统

.container .container-fluild

.row

.col-xs-\* .col-sm-\* .col-md-\* .col-lg-\*

.hidden-xs .hidden-sm .hidden-md .hidden-lg

表单

1. 组件
2. 插件
3. 定制
4. 全局CSS样式——栅格布局系统

补充：列的偏移问题(offset)

.col-xs-offset-1~.col-xs-offset-12 在lg/md/sm/xs屏幕下偏移

.col-sm-offset-1~.col-sm-offset-12 在lg/md/sm屏幕下偏移

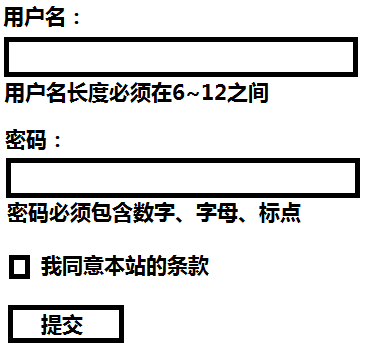
.col-md-offset-1~.col-md-offset-12 在lg/md屏幕下偏移

.col-lg-offset-1~.col-lg-offset-12 在lg屏幕下偏移

1. 全局CSS样式——表单——次重点&难点

Bootstrap中的表单分为三种：

(1)默认表单



<form>

<div class="form-group">

<label class="control-label"></label>

<input class="form-control">

<span class="help-block"></span>

</div>

</form>

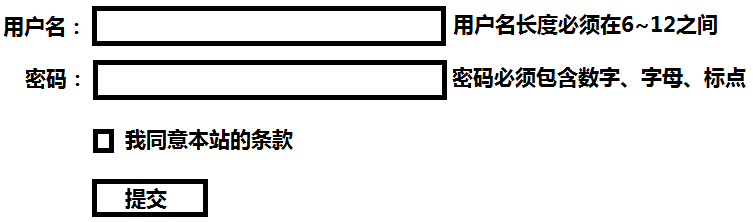
(2)行内表单



<form class="form-inline">

</form>

(3)水平表单



<form class="form-horizontal">

使用栅格系统来控制label/input/help-block的宽度

</form>

1. 组件——图标字体——愉快&简单

Glyphicons是一套收费的图标字体，提供了Web/移动开发中常用的小图标

Bootstrap中可以免费使用这套字体中的250+个；以服务器端字体形式出现的，即客户端若访问了使用Glyphicons字体的网站，会自动从服务器端下载对应的字体文件。

@font-face {

font-family: 'Glyphicons Halflings';

src: url('../fonts/glyphicons-halflings-regular.eot');

}

.glyphicon {

position: relative;

top: 1px;

display: inline-block;

font-family: 'Glyphicons Halflings';

font-style: normal;

font-weight: normal;

line-height: 1;

}

提示：(1)图标字体的本质不是图片，而是字体；故凡是可以使用文字的地方都可以使用不同字体

(2)glyphicon图标字体只能用于“空元素”——不包含任何内容或子元素！如：<span class="glyphicon glyphicon-\*\*\*"></span>

1. 组件——按钮组——简单

.btn-group 水平按钮组

.btn-group-vertical 竖直按钮组

.btn-group.btn-group-justified 水平且两端对齐的按钮组

.btn-group-lg

.btn-group-sm

.btn-group-xs

1. 组件——下拉菜单——小重点&简单

下拉菜单必须HTML结构：

**<div class="dropdown">** 相对定位的父元素

**<a data-toggle="dropdown">触发元素</a>**

**<ul/div class="dropdown-menu">** 绝对定位

**隐藏元素**

**</ul/div>**

**</div>**

1. 组件——导航——小重点&简单

提示：此处的导航不是指导航条！

Bootstrap提供了两种形式的导航：

(1)标签页式导航

<ul class="nav nav-tabs">

</ul>

(2)胶囊式导航

<ul class="nav nav-pills">

</ul>

此外，还有两种导航变种：

(1)两端对齐的导航 .nav.nav-tabs/pills.nav-justified

(2)竖直放置的胶囊导航 .nav.nav-pills.nav-stacked

7.组件——自学：路径导航(面包屑)/分页/标签/徽章/巨幕/水井/页头

面包屑： .breadcrumb



分页： .pagination .pager

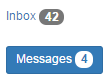




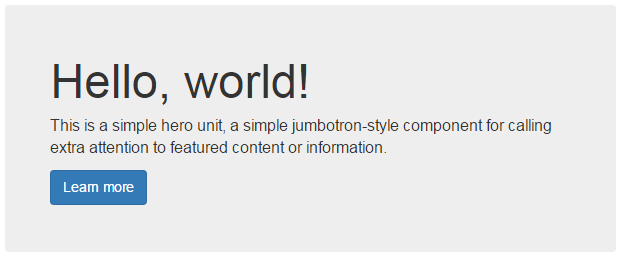
标签： .label



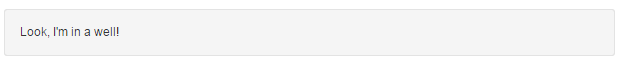
徽章： .badge



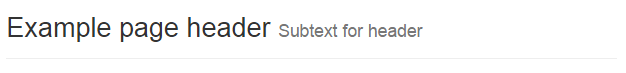
巨幕： .jumbotron



水井： .well



页头： .page-header



8.组件——响应式导航条——重点&难点

响应式导航条：在PC和平板中默认要显示所有的内容；但在手机中导航条中默认只显示“LOGO/Brand”，以及一个“菜单折叠展开按钮”，只有单击折叠按钮后才显示所有的菜单项。

基础class： .navbar

Bootstrap中导航条的按位置：

1. 顶部导航条
2. 底部导航条

Bootstrap中导航条的按颜色：

1. 浅色底深色的字 .navbar-default
2. 深色底浅色的字 .navbar-inverse

Bootstrap中导航条的按定位：

1. 相对定位position: relative 默认值
2. 固定定位position: fixed .navbar-fixed-top/bottom

导航条的结构：

<div class="navbar 颜色 定位">

<div class="container">

**<!--导航条的头部：商标+按钮-->**

<div class="navbar-header">

<a class="navbar-brand">

<button class="navbar-toggle">

</div>

**<!--导航条折叠菜单：菜单、按钮、搜索框、链接、文本...-->**

<div class="navbar-collapse">

<ul class="nav navbar-nav">

<form class="navbar-form">

<button class="navbar-btn">

<span class="navbar-text">

<a class="navbar-link navbar-text">

</div>

</div>

</div>

作业：

(1)自学“组件”——缩略图、**警告框、**进度条、**面板、媒体对象**

(2)复习表单和导航条

复习：

TwitterBootstrap是一个HTML/**CSS**/JS框架，适用于移动设备优先的响应式网页开发。主要涉及：

1. HTML：为H5已有的标签扩展了自定义属性 data-\*
2. **CSS： CSS Reset + 几千个class**
3. JS：基于jQuery提供了十几个插件函数

data-toggle="dropdown"

data-toggle="tab"

分为五部分：

1. 起步：下载 模板 Bootlint 示例 禁用
2. 全局CSS样式：按钮、图片、文本、排版、表格、栅格系统、表单
3. 组件：图标字体、下拉菜单、按钮组、导航、面包屑、分页、标签、徽章、水井、巨幕、页头、导航条
4. jQuery插件
5. 定制
6. 复习：组件——响应式导航条

|  |
| --- |
| 行业小知识：  盲人、智力低下、行动障碍...  H5标签中有两类属性与“无障碍阅读”相关：  (1)<ANY role=""> <b role="button"></b> VDA  (2)<ANY aria-\*=""> |

1. 插件——概述

Bootstrap基于jQuery提供了十几个插件函数（类似于jQueryUI插件库），每个插件对应一个.js文件，可以单独引用，也可以整体引用(bootstrap.js)。

调用上述十几个插件可以用两种格式：

(1)传统的JS方式调用： $(...).dropdown(); $().tab(...);

(2)使用data-\*扩展属性方式调用： <a data-toggle="dropdown">

1. 插件——下拉菜单

(1)$().dropdown( );

(2)<a data-toggle="dropdown">

1. 插件——标签页(tab)

(1) $().tab();

(2) <a data-toggle="tab">

1. 插件——Bootstrap提供的弹出框

(1)工具提示框(tooltip) data-toggle="tooltip"

(2)弹出框(popover) data-toggle="popover"

(3)警告框(alert) —— 小重点

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

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

xxxx

</div>

(4)模态对话框(modal) —— 小重点

模态框定义：在父窗体中弹出一个子窗体，子窗体若不关闭，父窗体就无法获得焦点，同时父子窗体间还可以传递数据。window.alert()/confirm()/prompt()就是典型例子。模态框必需的结构：

<div class="modal"> <!--半透明的黑色遮罩层-->

<div class="modal-dialog"> <!--宽/定位-->

<div class="modal-content"> <!--边框/背景色/阴影-->

<div class="modal-header">头部</div>

<div class="modal-body">主体</div>

<div class="modal-footer">尾部</div>

</div>

</div>

</div>

显示一个模态框：

1) <a href="#模态框ID" data-toggle="modal">

2) <button data-toggle="modal" data-target="#模态框ID">

1. 插件——折叠效果(collapse)

触发一个折叠效果：

1) <a href="#折叠元素ID" data-toggle="collapse">

2) <button data-toggle="collapse" data-target="#折叠元素ID">

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

<div id="折叠元素ID" class="collapse in">

</div>

折叠组件的两个扩展用途：

(1)Accordion（手风琴）

(2)响应式导航条在手机屏幕中的效果

1. 插件——轮播广告(carousel)——小重点

本身结构较复杂，编写时只需要记住根class: .carousel

<div class="carousel">

<div class="carousel-inner">

<div class="item active">

<img>

</div>

</div>

</div>

1. 插件——附加导航（Affix）

<div data-spy="affix" data-offset-top="100">

<ul class="nav nav-pills">

</div>

9.插件——滚动监听（scrollspy）——了解

随着页面内容的滚动，某个导航中的项目，会自动的更改.active位置。

实现思路：

1. 页面中必须首先有一个导航菜单(.nav)——其中可以定义一个菜单项为.active
2. 导航菜单中的超链接的href属性值必须和页面中的某个锚点名一样
3. 为页面添加滚动事件的监听函数

body.onscroll= function(){

if(body滚动的距离 === 某个锚点距离顶端的距离){

此锚点对应的超链接的父元素li添加.active；

}

}

作业：

1. 复习Bootstrap提供的所有class

栅格系统、导航条、表单、媒体对象、面板、表格

1. 归纳整理插件函数调用时对应的data-\*扩展属性有哪些

复习：

|  |
| --- |
| 思维导图：  用于整理思维过程，将思考的内容以图形化表示。 |

1. Bootstrap阶段项目

(1)day01：实现布局系统和内容

am：完成栅格系统的设计

pm：完成单元格中的内容

(2)day02：样式定制

am：less

pm：修改Bootstrap源代码

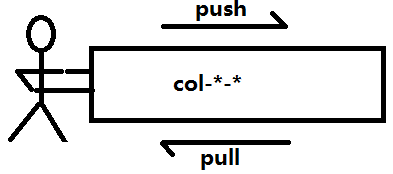
1. 补充：栅格布局系统中列的偏移和排序

**列偏移（offset）：** 用途-在不满的一行中某个列及后续的列向右偏移

col-xs/sm/md/lg-offset-\*

提示：某列偏移后，会影响后续的所有列。

**列排序（pull/push）：**用途-在一行中调整某列的位置，且不影响其他列



.col-lg-push-\* 在lg屏幕下将列向右推指定的宽度

.col-lg-pull-\* 在lg屏幕下将列向左拉指定的宽度

.col-md-push-\* 在md/lg屏幕下将列向右推指定的宽度

.col-md-pull-\* 在md/lg屏幕下将列向左拉指定的宽度

.col-sm-push-\* 在sm/md/lg屏幕下将列向右推指定的宽度

.col-sm-pull-\* 在sm/md/lg屏幕下将列向左拉指定的宽度

.col-xs-push-\* 在xs/sm/md/lg屏幕下将列向右推指定的宽度

.col-xs-pull-\* 在xssm/md/lg屏幕下将列向左拉指定的宽度

Bootstrap阶段项目：

day01:

am：响应式布局

pm：细节内容

day02:

am：Less语言

pm：定制Intel项目中的Bootstrap

1. 如何实现Bootstrap样式定制

(1)编写自定义的css，覆盖bootstrap.css中提供的样式

不足：产生大量的冗余/无用代码

(2)直接修改bootstrap.css文件

不足：任务量太大！——CSS通病

(3)修改Bootstrap开发者编写的Bootstrap源代码——bootstrap.less

不足：灭有！

1. 动态样式语言

**CSS：静态样式语言**，作为一门语言并不称职！缺少一般语言必需的基本要素：变量、运算、循环/选择、函数等，导致了CSS代码的修改和维护非常麻烦。

**动态样式语言：**在CSS的基础之上，添加了动态语言所必需的元素，如变量、运算、循环/选择、函数等，方便样式文件的修改和维护。

**注意：**浏览器默认只能处理静态样式语言，所有的动态样式语言必需设法转换为CSS才能被浏览器所理解！这个转换操作称为“编译”。

常见的动态样式语言：

(1)Sass/SCSS

(2)Stylus

(3)Less

3.Less的使用

是一种 动态 样式 语言.

LESS 为 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。

**(1)运行方式1：在客户端运行Less转换程序——了解**

在HTML中引入xx.less，同时再引入一个less.js——就是一个可以运行在客户端浏览器中的less编译程序——效率太低，不推荐使用。

**(2)运行方式2：在服务器端运行Less转换程序——推荐**

步骤：

1)下载并安装一款服务器端的JS解释器——NodeJS或者Rhino

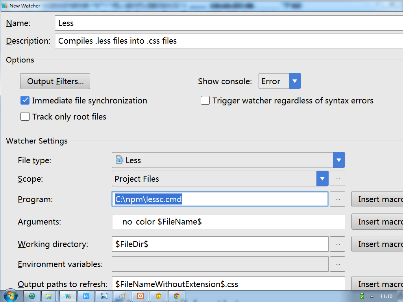
2)下载less文件的转换程序（lessc-Compiler）——一个JS脚本

3)在服务器端JS解释器中运行lessc转换器，把自己编写的.less文件转换为.css文件

1. 方式1：可以在命令行中使用转换程序

node.exe lessc my.less my.css

1. 方式2：可以在WebStorm中使用转换程序



1. 继续编写HTML文件，引用编译得到的.css文件即可。
2. Less语法学习

(1)Less完全支持CSS的所有语法

(2)Less支持单行和多行注释，但只有多行注释会被转换到css文件中

(3)Less支持变量(**Variable**)

语法：**@变量名: 值;**

使用：.class { 样式： @变量名; }

(4)Less支持样式混合(**Mixin**)——在一个样式中混入另一个样式

语法： **.class1{ ... }**

.class2{

...

.class1;

...

}

(5)带参混合

语法： **.class1(@参数1, @参数2, ...){ ... }**

.class2{

...

.class1(值1， 值2, ...);

...

}

(6)嵌套规则

语法：.class1 {

....

.class2 { ... }

}

转换的结果： .class1 { ... }

.class1 .class2 { ... }

(7)Less可以对变量、常量进行算术运算

语法： 变量/值 +-\*/ 变量/值

(8)Less为样式提供了几十个应用函数

lighten(颜色，亮度值)：将指定的颜色变亮指定的百分比

darken(颜色，亮度值)：将指定的颜色变暗指定的百分比

floor(数字)：对数值进行下取整

ceil(数字)：对数值进行上取整

(9)页面导入

尽量避免使用CSS文件中的@import指令——会增加HTTP请求次数；

为了可以将一个样式文件拆分为多个小的样式文件，由多人同时编写，可以使用LESS中的@import——less中导入其他less文件，转换时会拼接为一个大的完整的CSS样式文件，故推荐在Less中@import其他的Less文件。

语法： @import "xx.less";

@import "yy";

================示例：大型项目中的less文件结构=============

variables.less 放置所有的变量

mixin.less 放置所有的混合

reset.less 放置HTML元素重置样式

navbar.less 导航条相关样式

footer.less 页脚相关样式

....

jd.less => jd.css 总样式文件 一大堆@import

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

5.通过修改Bootstrap的Less源文件实现样式定制

定制的目标：

(1)删除不需要的样式，如轮播广告、模态框...

在bootstrap.less文件中，注释掉不需要的@import即可。

(2)定制需要的组件的默认样式，如修改导航条的默认背景色

修改variables.less文件中的变量即可。

(3)在Bootstrap提供的默认样式基础上创建新样式，如定制dropdown中的divider的样式——组件的深度定制

修改某个组件所对应的.less文件

课下练习：深度定制input中放大镜的颜色、导航条中的导航激活时有蓝色下边框。

AngularJS

jQuery是一个JS函数库，操作思路仍然是DOM操作思路：先查找元素，再操作元素。

jQueryUI是一个HTML组件库，简化了HTML/CSS的编写。

Bootstrap是一个CSS框架，主要提供了响应式布局、HTML组件、CSSReset。

AngularJS是一个JS框架，彻底颠覆了传统的DOM操作，所有的关注点集中在业务数据上，而不是DOM树。适用于以数据操作为主的SPA(Single Page Application)应用。

1. 软件工程——软件设计原则

(1)避免重复原则

(2)KISS原则，代码越简单越傻瓜越少

(3)YAGNI原则，不创建不需要的代码

(4)开闭原则，不让别人修改自己的代码，但可以扩展它

(5)单一责任原则

(6)高聚合低耦合原则，组件内部的逻辑关系越紧密越好，组件和组件之间关系越少越好。

(7)最少知识法则/迪米特法则，司机不必知道如何造汽车。

2.软件工程——设计模式

马走日象走田 => 双炮枪/马后炮

|| ||

对象/原型/闭包 => 设计模式 => 设计框架

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。

23+1种设计模式

+1的那种设计模式：MVC模式

|  |
| --- |
| modal： 模态框  model： 模型  module： 模块 |

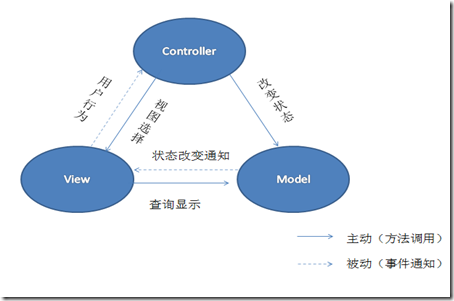
1. MVC设计模式

MVC模式根据逻辑关系，把前端项目的代码分为三个层次：

（1）Model：模型，就是业务数据，前端项目中就是JS变量；

（2）View：视图，就是业务数据在用户面前的展现，前端项目中就是HTML

（3）Controller：控制器，负责业务数据的增删改查，前端项目中就是function



|  |
| --- |
| 面试题：AngularJS在项目中的优缺点是什么？ |

3.AngularJS概述

AngularJS诞生于2009年，由Misko Hevery 等人创建，后为Google所收购。所有的操作思路都以“业务数据”为关注点，彻底颠覆了传统的DOM操作。适用于以数据的CRUD操作为主的SPA应用。

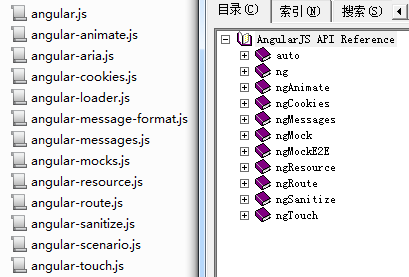
1. 面试题：AngularJS的四大特性

(1)MVC设计模式

(2)双向数据绑定

(3)依赖注入

(4)模块化设计



i18n: internationalization 国际化，Web项目若可以实现见中国人呈现汉语，见英国人呈现英语。

5.第一个AngularJS(ng)页面

引入angular.js文件即可

1. Angular表达式

语法：{{ 表达式 }}

含义：Angular会计算表达式的值，输出在当前位置。

练习：测试Angular表达式可以执行哪些运算？

1. 算数运算？ + - \* / % 都可以，唯独不能自增/自减
2. 比较运算？ > < 都可以
3. 逻辑运算？ && || ! 都可以
4. 三目运算？ ?: 可以
5. 调用字符串对象的成员方法？ 可以
6. 使用直接量表示法创建对象？ 可以
7. 可以使用数组吗？ 可以
8. 使用new关键字创建对象？ **不可以！不允许使用new和var关键字**
9. 调用ES全局函数？ **不可以！**

|  |
| --- |
| JavaScript中对象的分类：   1. ECMAScript标准对象 Global String Date RegExp Array Object ...   可以在任一个js解释器中使用   1. 宿主对象：   DOM: Node Element Attribute ....  BOM: window document ....  只能在浏览器中的js解释器中使用，不能在独立的服务器端js解释器(如NodeJS)中使用  (3)用户自定义对象 |

1. ng模块提供的指令(directive)

**(1)ngApp**：启动一个Angular应用——只有Angular应用中的表达式才会被Angular执行。

用法： <ANY ng-app>

注意：angular应用的范围仅限于声明它的元素范围；且默认情况下一个HTML中不允许声明多个ngApp元素。

**(2)ngInit**：声明模型变量(Model)——不是局部变量

用法： <ANY ng-init="变量名=值; 变量名=值;">

注意：ngInit声明变量不能声明var关键字！声明的变量可以在Angular表达式中进行输出

**(3)ngBind:** 计算一个表达式的值，输出为当前元素的innerHTML

用法： <ANY ng-bind="表达式"></ANY>

<ANY class="ng-bind: 表达式;'></ANY>

说明：ngBind指令作用与{{}}表达式基本类似，只是可以防止用户在一瞬间看到{{}}。ngBind指令计算完成表达式的值，会替换当前元素的innerHTML.

**(4)ngController**：调用Controller创建函数，实例化一个控制器对象，指定其作用范围

用法： <ANY ng-controller="控制器名">...</ANY>

**(5)ngRepeat**：用于在View实现循环输出

用法： <ANY ng-repeat="变量名 in 集合对象"></ANY>

含义：对于集合对象中的每一个元素，依次赋值给指定的变量名，对每次赋值都输出一遍当前元素。

**(6)ngIf：**用于在View实现判断输出，为false就不输出了

用法： <ANY ng-if="布尔表达式">

含义： 若布尔表达式为true则输出当前元素；否则当前元素在DOM树不存在

1. AngularJS中声明模型数据的方式

(1)使用ngInit指令来声明Model数据

<ANY ng-init="变量名=值;">

说明：此方式将Model声明在View中，严重违反了MVC模型的分工，不推荐使用

(2)使用Controller对象创建Model数据——符合MVC模型分工

新版本的AngularJS中创建Model的语法：

ngApp=>Module=>Controller=>Model

1. 声明一个AngularJS的应用程序： ngApp
2. 创建一个自定义的模块： angular.module('模块名', [依赖列表])
3. 在应用中注册自定义模块： ng-app="模块名"
4. 在模块中声明Controller函数
5. 在View中指定Controller对象的作用范围——调用控制器创建函数
6. 在Controller中声明Model数据

练习：**使用Controller创建Model数据：**

创建一个新的HTML页面，声明一个ngApp，创建一个模块，其中声明一个控制器，创建如下的几个Model数据，在View中显示这些数据：

1. string
2. number
3. boolean
4. array
5. object
6. array[object]

作业：

(1)关注developer.mozilla.com上有关H5/CSS3/JS/DOM

(2)使用Controller创建Model数据，学生列表5个学生，每个学生包含姓名、性别、生日、总分这几个属性，把上述数据显示在一个TABLE中，tr必须循环。

提高功能点：仅显示总分在600以上的学生的数据。

复习：

Google AngularJS是一个JS框架，主要思路不是传统的“先查找再操作DOM元素”，而是“以数据为中心”。适用于以数据操作为主的SPA应用。

Angular四大特性：

(1)MVC模型

Model：模型，即业务数据，在前端应用中就是保存在特定范围的JS变量

View：视图，即业务数据在用户面前的呈现，在前端应用中就是HTML

Controller：控制器，负责业务数据的获取、修改、删除等，在前端应用中由function来担当。

(2)双向数据绑定

(3)依赖注入

(4)模块化(Module)设计——体现“高聚合低耦合”设计原则

学习目标：

1. 复习并扩展MVC模型
2. 双向数据绑定——重点&偏难
3. 依赖注入——次重点&简单

|  |
| --- |
| 面试题： AngularJS与jQuery的关系？  jQuery操作思路：先找元素，再操作元素 $(....).xxx();  AngularJS操作思路：创建业务数据、绑定数据、维护数据  AngularJS已经把底层/低级的DOM操作，为开发者封装起来了    AngularJS在加载时会查看当前页面是否已经加载了jquery.js(就是判断window.jQuery是否存在)，若存在则所有的DOM操作全都使用jQuery提供的方法；若不存在，则anglarJS会使用自定义的jQuery精简版本——jqLite——只有jQuery的核心方法。 |

1.复习MVC模型

(1)在一个Angular应用中可以声明多个模块(module)

(2)某个模块可以依赖于其他的模块

(3)有一个模块必须注册给ngApp指令——启动模块

(4)一个模块中可以声明多种组件，如：

1. controller
2. directive
3. service
4. filter
5. ...

2.控制器的作用范围/作用域

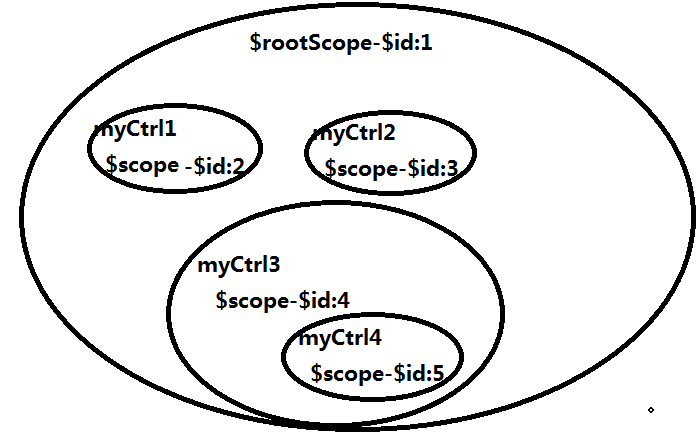
(1)每次调用ngController都会创建一个新的Controller对象

(2)每个Controller对象都有唯一的$scope对象

(3)$scope就表示当前控制器对象的有效范围/作用域

(4)声明在某个$scope中模型数据，一般情况下不能被其他的控制器所使用。

(5)**若想在多个控制器间共享/传递数据**，可以声明在根作用域中——$rootScope—每个Angular应用(ngApp)只有一个唯一的$rootScope对象



(6)控制器的本质用途：用于划分一个大型页面中的不同的DIV块——每个这样的块中都有自己专用的数据，以及可以与其他块共享的数据。

1. AngularJS四大特性之二——双向数据绑定——偏难&重点！

**(1)方向1：Model绑定到View，**此后不论何时只要Model发生改变，View会自动立即同步更新。

实现方法：{{ }}、ngBind、ngIf、ngRepeat、ngShow、ngChecked ... 等等几乎所有的显示数据的指令都实现了方向1的绑定。

练习：

1)创建一个点击计数器，有一个按钮，只要点击一次，立即显示出点击次数。

2)仿写轮播广告中的“前进/后退”按钮，提示：忘掉传统的轮播广告！注意img的src属性值可以赋值为一个Model数据，而此数据可以使用模型函数进行修改。

**(2)方向2：View绑定到Model**，把视图中用户可以修改的HTML元素——即表单控件——的值绑定到一个Model变量上。此后，不论何时只要用户修改了表单控件的值，后台模型变量的值会立即随之改变。

实现方法：**只有ngModel指令可以**！ 为了监视到Model变量真的被改变了，可以使用$scope.$watch()函数对Model数据的值进行监视。

·单行文本输入域，ngMode可以把value属性值绑定到Model变量

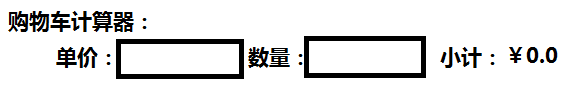
·复选框，ngModel可以把true/false值绑定到Model变量

·单选框，ngModel可以把当前选中的单选框的value值绑定到Model变量

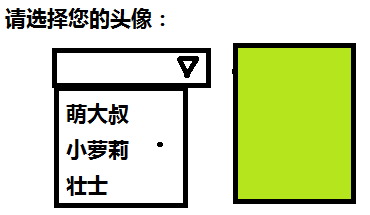
·下拉框，ngModel可以把当前选中的option的value值绑定到Model变量

练习：

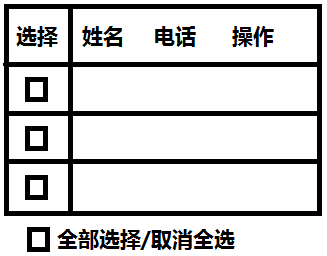
1. 实现一个简易版的购物车计算器



1. 用户“同意本站使用条款”则显示“注册”按钮；否则“注册”按钮消失
2. 下拉选择用户头像的名称，旁边立即显示用户选择的头像图片



4)表格外勾选“全部选择”复选框，则表格内的所有复选框全部选择



1. ng模块中提供的service组件

(1)$rootScope 用于在所有的控制器间共享数据的服务

(2)$interval 周期性定时器服务

(3)$timeout 一次性定时器服务

1. ng模块提供的directive组件

(1)ngApp

(2)ngInit

(3)ngBind

(4)ngIf

(5)ngRepeat

(6)ngClick： 为元素绑定单击事件的监听函数——只能是Model函数($scope.函数名=function(){})，不能是全局函数

(7)ngMouseOver ....

(8)ngSrc: 为IMG标签指定src属性，但可以防止404请求错误

(9)ngShow: 若赋值为true，则display:block；否则display:none;

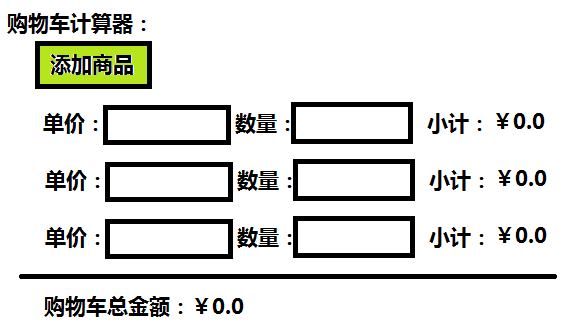
(10)ngHide: 若赋值为true，则display:none；否则display:block;

|  |
| --- |
| 如何压缩js文件？  Yahoo User Interface，YUI项目中提供了一个很好用的压缩程序：yui-compressor  可用于压缩CSS、JS文件。使用步骤：   1. 安装Java运行环境 2. 编写JS/CSS文件，用于压缩 3. 运行yuicompressor工具，执行压缩   提示：可以把yuicompressor配置为WebStorm中的FileWatcher，监视js/css文件，实现自动的压缩    注意：压缩工具会执行如下的压缩：  (1)删除所有的注释  (2)删除所有的无用的空白字符  (3)变量名、函数名、形参名尽可能的简化  -----------------------------------------------------------------  (4)注意：所有的数据和值不会进行简化！ |

6.AngularJS四大特性之三——依赖注入

作业：

1. 完成一个完整版的购物车计算器：



复习：

Google AngularJS是一个JS框架，适用于以数据操作为主的SPA应用。所有的操作都是以数据为出发点。四大特性：

(1)MVC模型

Model：即业务数据 $scope.xx $rootScope.xx

View：即业务数据的呈现 HTML + ngXxx

Controller：负责操作业务数据 .controller('', function(){})

(2)双向数据绑定

方向1：Model绑定到View，只要Model变View随着变

方向2：View(表单控件)绑定到Model，只要View变Model随着变

(3)依赖注入

(4)模块化设计

|  |
| --- |
| 面试题：AngularJS的最大的不足/应用时需要特别关注的地方？  原生ES/JS/DOM只有特定HTML元素的特定事件的监听机制，没有监听数据/对象/值改变的机制。 |
| $interval和window.setInterval()的区别？  $interval修改的任何Model数据，底层会立即遍历一遍$digest队列；  setInterval()即使修改了Model数据，也不会遍历$digest队列； |
| $interval(function(){ $scope.count++; }, 1000)  等价于  setInterval(function(){ $scope.count++; $digetst()/**$apply()**; }, 1000) |

1. 依赖注入

Dependency：若某个函数调用时需要其它的对象作为形参，就此函数依赖于形参对象。

function Driver( car ){ //司机依赖于一个car对象

car.start();

car.run();

car.stop();

}

如何解决依赖关系：

**(1)主动创建方式**

var c1 = new Car(); //主动创建依赖对象

var d1 = new Driver( c1 ); //传递依赖对象

**(2)被动注入(Injection)方式**

module.controller('控制器', function(**$scope, $interval**){})

Angular中的ngController指令在实例化控制器对象时，会根据指定的形参名，创建出控制器所依赖的对象，并注入给控制器对象——依赖注入（Dependency Injection，DI）现象。

注意：

(1)Angular在创建控制器对象时，会根据形参列表中的每个形参的名称来创建依赖的对象，*故控制器声明函数不能声明Angular无法识别的形参名*——Angular只提供了这一种依赖注入方式——根据形参名来注入依赖的对象！

(2)若项目JS文件使用了类似yuicompressor等压缩程序，默认会把函数的形参名精简为一个字母的形式，会导致Angular的依赖注入失败！解决办法：

module.controller('控制器名', [

'$scope',

'$interval',

'$http',

function(aaa,bbb,ccc){}

])

["num1", 'num2', 'num3', function add(n1, n2,n3){}]

var num1=10;

var num2=20;

var num5=50;

var num3=30

add(10, 20, 30);

1. 可以被注入的对象——所有的service/provider对象都可以被注入

(1)$rootScope：在多个控制器间共享数据的服务

(2)$interval：提供周期性定时器服务

(3)$timeout：

(4)$log：提供五个基本的日志输出服务

(5)$http：提供异步HTTP请求（AJAX）服务

用法： $http({method:'GET', url:'/xx.php'}).

success(fn).

error(fn);

简化版： $http.get('url').success(fn);

$http.post('url', data).success(fn);

(6)$location

1. ng模块中提供的过滤器(filter)

Filter: 把Model数据在显示时以某种特定的格式呈现。

(1)lowercase

语法： {{ 表达式 | lowercase }}

(2)uppercase

语法： {{ 表达式 | uppercase }}

(3)number

语法： {{ 表达式 | number }}

{{ 表达式 | number : 小数位数 }}

(3)currency

语法： {{ 表达式 | currency }}

{{ 表达式 | currency : '货币符号' }}

(3)date

语法： {{ 表达式 | date }} 默认格式： Sep 1, 2015

{{ 表达式 | date : '日期时间格式'}}

1. Web项目中多页应用和单页应用的比较

Mutiple Page Application

Single Page Application

|  |  |
| --- | --- |
| 多页应用 | 单页应用(SPA) |
| 项目中有多个完整的HTML页面 | 整个项目中只有一个完整的HTML页面；其它HTML文件都是HTML片段 |
| 使用超链接、JS实现页面跳转 | 使用超链接、JS实现“伪跳转” |
| 所有的页面请求都是同步的——客户端在等待服务器给响应的时候，浏览器中时一片空白 | 所有的“伪页面请求”都是异步请求——客户端在等待下一个页面片段到来时，仍可以显示前一个页面内容——浏览体验更好 |
| 不便于实现两个页面间切换过场动画 | 很容易实现两个伪页面间的过场切换动画 |
| 浏览器需要不停的创建完整的DOM树、删除完整的DOM树 | 浏览器只需要创建一个完整的DOM树，此后的伪页面切换其实只是在换某个div中的内容。 |
| 每个页面都需要加载自己的CSS和JS文件 | 整个项目的CSS和JS文件只需要加载一次 |

手动实现单页应用的步骤：

1. 创建一个完整的HTML页面（如index.html)，引入所需要的所有CSS和JS；body中只需要一个伪页面的容器元素，如<div></div>
2. 创建若干个伪HTML页面/模板页面：只需要声明HTML片段
3. 客户端请求完整的HTML页面，同时URL中再追加一个特殊的标记，如http://127.0.0.1/index.html#/start——指定要加载的伪页面的名称
4. 浏览器解析出伪页面名称，查找一个字典，找到该名称对应的模板页面的URL

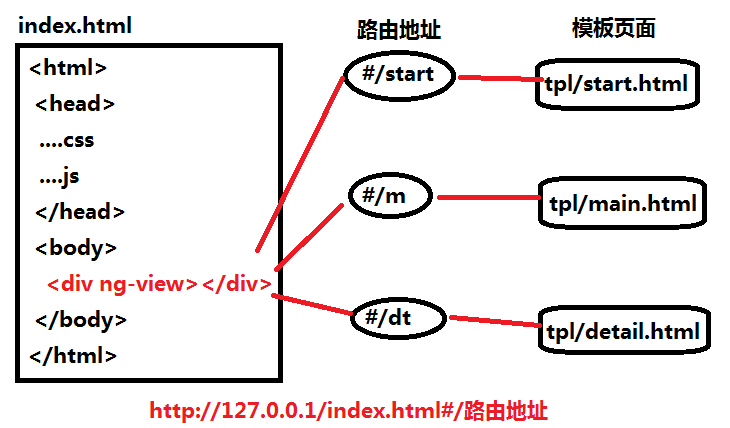
window.location.hash

#/start => template/start.html

#/m => template/main.html

#/detail => template/productdetail.html

(5)客户端发起异步的AJAX请求，获取模板页面的内容，加载到index.html的伪页面容器中即可



1. AngularJS提供的模块——ngRoute

Route：路由，通过某条线路找到目标内容。

ngRoute模块的用途：就是根据浏览器中URL中的一个特殊的地址标记(形如#/xxx)，查找到该标记所对应的模板页面，并异步加载到当前页面的ngView指令中。使用步骤：

(1)创建唯一完整的HTML页面，其中声明一个容器，ngView指令。引入angular.js和angular-route.js

(2)创建多个模板页面(习惯上放在一个特别的目录下，如tpl)

(3)创建Module，声明依赖于ng和ngRoute两个模块。

(4)在Module中配置路由字典。

(5)使用浏览器做测试：

http://IP地址/index.html#/路由地址

1. ngRoute模块中的伪页面跳转
2. **通过超链接跳转**

<a href="#**/路由地址**"> #不能省

1. **通过JS跳转**

<button ng-click="jump()"></button>

$scope.jump = function(){

//location.href="2.html" 不能使用多页面应用中的跳转

**$location.path('/路由地址'); //#不能有**

}

作业：

(1)自学RESTful相关知识

复习：

AngularJS知识点整理

1. 复习：ngRoute

使用步骤：

(1)创建index.html，引入css/js，声明ngView

(2)创建模板页面

(3)创建模块，配置路由字典

module.config(function($routeProvider){

$routeProvider.

when('/start', {templateUrl: 'xxx.html'})

})

(4)测试： http://IP/index.html#/start

1. ngAnimate模块的使用

Angular本身没有提供任何动画效果，但ngAnimate模块提供了一些“动画钩子(hooks)”，可以钩住用户自定义**JS(jQuery.animate())、Transition、Keyframes**动画效果。

ngAnimate模块为下述指令提供的动画钩子：ngRepeat, ngInclude, ngIf, ngSwitch, ngShow, ngHide, ngView and ngClass。

演示：使用ngAnimate提供的动画钩子调用自定义的Transition动画效果。

1. 引入angular.js、angular-animate.js
2. 自定义模块声明依赖ngAnimate模块，相关指令就会自动生成动画钩子
3. 针对动画钩子编写Transition动画

/\*要离开的元素动画开始时的样式\*/

.page.ng-leave {

left: 0;

opacity: 1;

}

/\*要离开的元素动画结束时的样式\*/

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

left: -100%;

opacity: 0;

}

/\*要进入的元素动画开始时的样式\*/

.page.ng-enter {

left: 100%;

opacity: 0;

}

/\*要进入的元素动画结束时的样式\*/

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

left: 0;

opacity: 1;

}

1. AngularJS阶段项目

day01:

am：完成基本页面切换效果

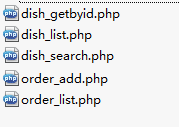
pm：每个模板页面的静态内容

提高：仿“饿了么”的样式实现Bootstrap定制

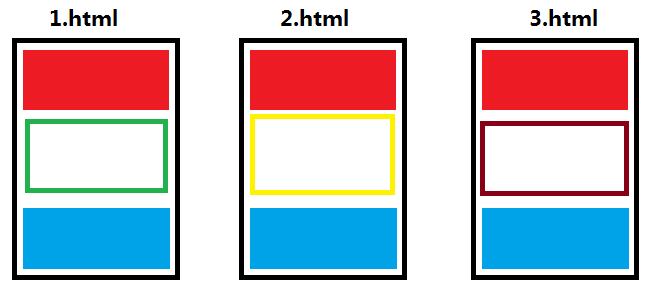
day02：

am：完成后台PHP页面

pm：发起AJAX请求，呈现动态数据



1. 补充：如何实现页面包含



项目中，习惯把多个页面中完全一样的内容，单独提取出来作为一个独立的文件(如header.html、footer.html)，凡是需要此文件的页面，引入该页面即可。页面包含可以采用多种方案：

(1)利用Web服务器的SSI命令：客户端请求一个页面，服务器一次返回多个页面——需要修改Web服务器配置文件。

(2)使用服务器端动态语言提供的页面包含函数：如PHP:

include('header.php');

....echo '主体';

include('footer.php');

客户端请求一个页面，服务器返回多个PHP页面组合后的一个页面。

(3)在客户端使用AJAX技术：先加载一个页面的主体内容，加载完成后，再去请求header.html、footer.html放到空容器中

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

<main>XXXXXXXX</main>

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

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

$.ready(function(){

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

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

})

提示：AngularJS中ng模块提供了一个指令：ngInclude，已经实现了方法3。

**<div ng-include=" 'tpl/header.html' "></div>**

|  |  |
| --- | --- |
| 面试题：说出下面几段代码运行后的效果： | |
| View：  <p>{{count}}</p>  Controller:  $scope.count=0;  setInterval(function(){  $scope.count++;  }, 1000) | View：  <p>{{count}}</p>  Controller:  $scope.count=0;  $interval(function(){  $scope.count++;  }, 1000) |
| View：  <p>{{count}}</p>  Controller:  $scope.count=0;  setInterval(function(){  $scope.count++;  $scope.$digest( );  }, 1000) | View：  <p>{{count}}</p>  Controller:  $scope.count=0;  setInterval(function(){  $scope.count++;  $scope.$apply( );  }, 1000) |
| View：  <p>COUNT1: {{count1}}</p>  <p>COUNT2: {{count2}}</p>  Controller:  $scope.count1=0;  $scope.count2=0;  setInterval(function(){  $scope.count1++;  }, 1000)  $interval(function(){  $scope.count2++;  }, 1000) | View：  <p>COUNT1: {{count1}}</p>  <p>COUNT2: {{count2}}</p>  Controller:  $scope.count1=0;  $scope.count2=0;  setInterval(function(){  $scope.count1++;  }, 1000)  $interval(function(){  //$scope.count2++;  }, 1000) |

WebAPP

学习目标：WebApp——把HTML5应用打包到Android手机中。

特点：代码量少！思路简单！——过程有点乱，很容易出错！

|  |
| --- |
| 面试题：面向过程(function)和面向对象(object)的区别？  面向过程的语言：C、C++、JavaScript、PHP —— 底层软件，如三大基础软件  面向对象的语言：C++、JavaScript、PHP、Java、C# —— 高层应用软件 |

Hybrid App = NativeApp + WebApp

1. 原生App和WebApp的比较

**原生(Native)App指：**

iOS(Linux-C)： Objective-C、Swift

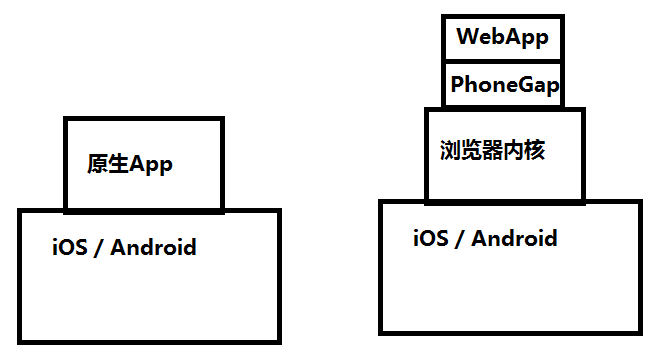
Android(Linux-C)： Java

**WebApp指：**

使用HTML5编写的移动Web应用。一个WebApp几乎可以不加修改的运行在PC、Android、iOS等平台。

优势：一套代码到处运行。

劣势：某些底层功能缺失，运行速度不如原生App。



提示：有Android手机的同学设置如下：

1. 保证在“我的电脑”中可以看到并打开我的手机
2. Android系统 => 设置 => 开发者选项 => 启用USB调试

就可以直接安装自己开发的移动应用到Android系统了

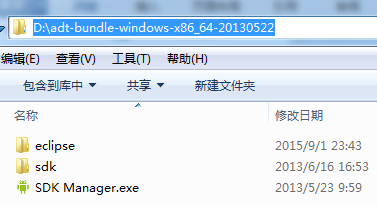
1. Android开发环境的搭建

(1)下载并安装Java程序的运行环境——JDK（JavaDevelopmentKit）

baidu：jdk安装

保证在命令行中输入 java.exe 可以运行

(2)下载并解压缩Android应用的开发环境——ADT（AndroidDeveloperTools）



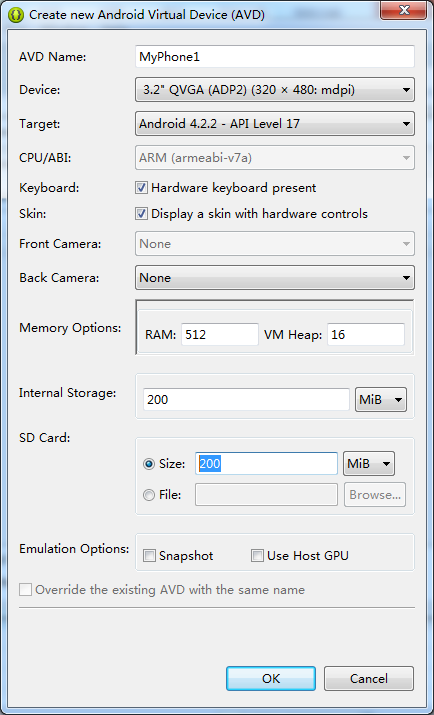
(3)启动ADT： eclipse/eclipse.exe

(4)创建虚拟的Android设备——Android Virtual Device Manager

注意：有真实Android手机可以省略此步，在

Eclipse->Window->OpenPerspective->DDMS

界面下可以查看到自己的手机



(5)创建Android应用程序，选择自定义图标，定制窗口中的内容，安装真实手机或虚拟手机中。

提示：若ADT无法发现真实手机，可以把项目运行一下，在bin目录下会生成Android应用程序的安装程序： xxx.apk 可以直接把此文件拷贝到手机的sd卡中，使用手机直接安装此apk文件。

