# **前端笔记**

[HTML 6](#_Toc5146)

[day01 6](#_Toc11391)

[Day02 12](#_Toc22543)

[Day03 17](#_Toc15763)

[CSS 24](#_Toc16668)

[Day01 24](#_Toc5168)

[1、CSS概述 24](#_Toc21529)

[2、CSS语法 24](#_Toc12898)

[3、尺寸 与 边框 28](#_Toc8489)

[Day02 31](#_Toc19683)

[1、CSS概述 31](#_Toc25272)

[2、CSS语法 31](#_Toc4763)

[1、边框属性 33](#_Toc14713)

[2、框模型 34](#_Toc16698)

[3、背景属性 37](#_Toc5401)

[Day03 39](#_Toc30601)

[1、渐变 40](#_Toc6083)

[2、文本格式化 42](#_Toc31591)

[3、表格属性 44](#_Toc29364)

[Day04 46](#_Toc6287)

[1、定位 - 浮动 46](#_Toc27752)

[2、显示 47](#_Toc9874)

[3、列表 49](#_Toc1316)

[4、定位 50](#_Toc30641)

[Day05 52](#_Toc5359)

[1、复杂选择器 52](#_Toc16250)

[2、内容生成 54](#_Toc22070)

[3、弹性布局 55](#_Toc14165)

[Day06 59](#_Toc10929)

[1、CSS Hack 59](#_Toc14432)

[2、CSS3-转换 60](#_Toc5184)

[3、转换-2D转换 60](#_Toc32075)

[4、转换-3D转换 62](#_Toc27093)

[3、CSS3-过渡 63](#_Toc5153)

[4.Css动画animation 64](#_Toc20660)

[Bootstrap 66](#_Toc1804)

[Day01 66](#_Toc1151)

[Day02 69](#_Toc15522)

[1、Twitter Bootstrap 69](#_Toc3316)

[2、Bootstrap - 起步 69](#_Toc21311)

[3、Bootstrap 第二部分 - 全局CSS样式 70](#_Toc13893)

[4、Bootstrap 默认将屏幕分成四大类别 71](#_Toc6256)

[5、Bootstrap 提供的两种容器 71](#_Toc2408)

[6、全局Class - 按钮 71](#_Toc1154)

[7、全局Class - 列表 71](#_Toc11235)

[8、全局Class - 图片 72](#_Toc9128)

[9、全局Class - 表格 table 72](#_Toc2767)

[10、全局Class - 文本 72](#_Toc24934)

[11、全局Class - 栅格布局系统(重中之重) 72](#_Toc4911)

[Day03 74](#_Toc4117)

[1、全局Class - 栅格布局系统(下) 74](#_Toc2514)

[2、全局Class - 表单 74](#_Toc5563)

[1、什么是组件 76](#_Toc26447)

[2、组件 76](#_Toc12441)

[Day04 78](#_Toc223)

[1、组件-进度条 80](#_Toc6415)

[2、组件 - 警告框(alert) 80](#_Toc19753)

[3、组件 - 缩略图 81](#_Toc13262)

[4、 组件 - 媒体对象 81](#_Toc19508)

[5、组件 - 列表组 81](#_Toc24177)

[6、组件 - 面板 81](#_Toc5147)

[7、组件 - 导航条(navbar) 82](#_Toc29164)

[8、JS插件 - Plugins 82](#_Toc20538)

[9、JS插件 - 工具提示 83](#_Toc12667)

[10、JS插件 - 弹出框 83](#_Toc12012)

[11、JS插件 - 标签页 83](#_Toc20974)

[12、JS插件 - 模态框(modal) 83](#_Toc23061)

[Day05 85](#_Toc25743)

[1、插件-折叠效果(Collapse) 86](#_Toc14595)

[2、插件-图片轮播(carousel) 86](#_Toc2544)

[Bootstrap 定制 & Less 88](#_Toc28281)

[Day06 90](#_Toc13408)

[1、插件-折叠(Collapse) 90](#_Toc23511)

[2、附加导航 & 滚动监听 91](#_Toc16092)

[3、插件-轮播(Carousel) 91](#_Toc31423)

[1、Less 93](#_Toc1204)

[2、Bootstrap 定制 94](#_Toc10407)

[Javascript 95](#_Toc18245)

[Day01 95](#_Toc7224)

[一、什么是JavaScript 95](#_Toc23145)

[二、变量、常量 97](#_Toc14955)

[三、数据类型: 98](#_Toc5523)

[Day02 100](#_Toc29121)

[一、数据类型转换: 100](#_Toc14006)

[二、运算符和表达式:(重点) 101](#_Toc13322)

[Day03 105](#_Toc3425)

[一、\*\*\*函数:（重点、难点） 105](#_Toc11159)

[二、全局函数:了解 107](#_Toc14260)

[三、分支结构: 107](#_Toc28365)

[Day04 110](#_Toc18467)

[1.\*\*\*循环: 110](#_Toc19821)

[2.\*\*\*数组: 112](#_Toc8733)

[Day05 114](#_Toc27924)

[补：遍历 114](#_Toc3340)

[1.数组API: 115](#_Toc22204)

[2.栈和队列: 117](#_Toc14419)

[3.二维数组: 118](#_Toc25928)

[Day06 119](#_Toc47)

[1.String 119](#_Toc17755)

[2.\*\*\*\*正则表达式:Regular Expression 121](#_Toc676)

[Day07 124](#_Toc30992)

[1.RegExp正则表达式对象 124](#_Toc19684)

[2.Math: 125](#_Toc9807)

[3.Date 126](#_Toc27852)

[DOM 128](#_Toc27848)

[Day01 128](#_Toc20333)

[1.什么是DOM 128](#_Toc10468)

[2.\*\*\*查找:4种: 129](#_Toc23739)

[3.修改: 132](#_Toc29487)

[Day02 133](#_Toc473)

[1.修改: 133](#_Toc26621)

[2.添加/删除/替换: 135](#_Toc9881)

[Day03 136](#_Toc24927)

[1.HTML DOM常用对象: 136](#_Toc12329)

[2.什么是BOM:Browser Object Model 138](#_Toc23976)

[Day04 143](#_Toc520)

[1.\*\*\*\*event 143](#_Toc24323)

[Javascript core 145](#_Toc14258)

[Day01 145](#_Toc13557)

[1. 错误处理: 145](#_Toc25769)

[2.\*\*\*\*Function 146](#_Toc26308)

[3.匿名函数: 147](#_Toc21954)

[4.\*\*\*作用域(scope)和作用域链(scopechain) 147](#_Toc28169)

[5.\*\*\*\*\*闭包(closure): 148](#_Toc26081)

[Day02 149](#_Toc11012)

[1.\*\*\*\*\*面向对象OOP 149](#_Toc18822)

[2.\*\*\*\*\*继承: 151](#_Toc2158)

[3.多态:同一个函数在不同情况下，表现出不同的状态 153](#_Toc4061)

[Day03 155](#_Toc13007)

[1.\*\*\*\*ES5: 155](#_Toc19755)

[2.ES6: 159](#_Toc10157)

[Jquery 162](#_Toc16851)

[Day01 162](#_Toc17166)

[1.什么是jQuery: 162](#_Toc10820)

[2.如何使用: 162](#_Toc26143)

[3.jquery 选择器---\*\*\*查询/查找:2种: 164](#_Toc1931)

[Day02 167](#_Toc26622)

[1.修改:一个函数两用 167](#_Toc28012)

[2.查找:(按节点间关系查找) 168](#_Toc18944)

[3.添加，删除，替换，克隆 168](#_Toc2591)

[4.事件处理: 169](#_Toc13025)

[Day03 170](#_Toc32044)

[1:jquery对象 170](#_Toc4293)

[2:jquery选择器选择器-->查找元素方法 170](#_Toc32255)

[3:jquery方法 170](#_Toc32756)

[4:事件 170](#_Toc6175)

[Day04 171](#_Toc17583)

[1.事件处理: 171](#_Toc12855)

[2.鼠标事件: 171](#_Toc22688)

[3.模拟触发: 172](#_Toc32339)

[2.动画:2大类 172](#_Toc20641)

[3.简化类数组对象操作 173](#_Toc9461)

[4.封装自定义插件 173](#_Toc17908)

[HTML5 174](#_Toc18931)

[Day01 174](#_Toc1583)

[Day02 178](#_Toc16709)

[Day03 182](#_Toc10067)

[Day04 186](#_Toc15139)

[Day05 189](#_Toc18935)

[Day06 193](#_Toc10784)

[Day07 197](#_Toc24938)

[AJAX 198](#_Toc31532)

[Day01 198](#_Toc23092)

[Day02 201](#_Toc13989)

[Day03 204](#_Toc31077)

[Day04 208](#_Toc6873)

[Day05 210](#_Toc25488)

[Day06 212](#_Toc934)

[Day07 213](#_Toc28771)

[Day08 215](#_Toc20513)

[Day09 217](#_Toc22586)

[NodeJS 218](#_Toc15974)

[Day01 218](#_Toc29479)

[Day02 221](#_Toc14085)

[Day03 224](#_Toc15673)

[Day04 225](#_Toc3233)

[Angular 229](#_Toc12816)

[Day01 229](#_Toc6588)

[Day02 233](#_Toc25088)

[Day03 236](#_Toc13773)

[Day04 238](#_Toc30027)

[Webapp 241](#_Toc1038)

[Day01 241](#_Toc13890)

[Day02 244](#_Toc29088)

[Day03 246](#_Toc16711)

[Day04 249](#_Toc9726)

[Day05 252](#_Toc12807)

[Day06 255](#_Toc32722)

[Day07 255](#_Toc8041)

[React 259](#_Toc18859)

[Day01 259](#_Toc1024)

[Day02 264](#_Toc27940)

[Vue 267](#_Toc24322)

[Day01 267](#_Toc24696)

[Day02 271](#_Toc31255)

[微信开发 276](#_Toc10657)

[Day01 276](#_Toc25214)

[Day02 279](#_Toc27499)

[Day03 283](#_Toc22603)

# HTML

## day01

g-yangyan@tedu.cn

一.什么是web前端？

1.什么是web？

使用浏览器去访问的程序，叫网页

2.什么是web前端？

web代码存放在服务器

代码分为两种：

一种运行在浏览器端：前端代码

一种运行在服务器端：后端代码

3.完整优秀的网站包括两部分：前端+后台

二.技术

1.静态页面(按照产品设计图来编写网页.html,网页的特点它是运行在浏览器里)

技术：html,css

2.添加功能：动态特效，页面交互

技术：javascript,jQuery,DOM

3.和服务器交互

技术：server,php,ajax

4.复杂开发：html5高级，框架（bootstrap,node.js,angular.js）,移动端开发，微信开发

今天目标：

1.web基础知识

**1.web与internet**

1.internet

internet指的是全球计算机互联网络

通常称：因特网，国际互联网，交互网，互联网

2.提供服务

www:world wide web(万维网)通过网络访问网页

telnet:远程登录

email:电子邮件

ftp:上传下载文件

BBS:电子公告板，俗称论坛

3.基本实现技术

1.分组交换原理

2.TCP/IP协议

**2.web的工作原理**

web指的是在网络环境下的一种应用程序，俗称网页

工作原理:

由浏览器，服务器，通信协议组成

通信协议：http

浏览器：IE Chrome firefox safari opera

浏览器功能：

1.提交请求

2.解析HTML，css,js

3.能以图形化的方式显示文档

服务器：

1.存储web上的信息

2.响应浏览器请求并执行服务器程序

3.具备基本的安全性功能

服务器产品：

Tomcat iis apache

服务器程序：java,php,.net,c,node.js

前端数据展示：HTML，CSS，js

2.HTML快速入门（重点）

**1.HTML概述**

1.什么是HTML？

Hyper text markup language 超文本标记语言

ex:

普通文本a:无特殊意义

超级文本a:超级链接

超文本：文本具有特殊功能

标记：超文本的组成形式<a>

语言：拥有自己的语法结构

2.特点

1.以.html或者.htm为后缀

2.由浏览器解析执行

3.可以嵌套脚本语言（javascript,vbscript）

4.用带有尖括号的"标记"来标识

**2.HTML基础语法**

1.标记

1.什么是标记？

在HTML中，用于描述功能的符号，称为标记

2.语法

标记在使用时必须用尖括号括起来

标记分为：

1.封闭类型

<标记>内容</标记>

也称为双标记

ex:

<a>网页</a>

必须有开始有结束标记，否则会出现意想不到的结果

2.非封闭类型

语法：

<标记>或<标记/>

也称为单标记

ex:<img>或<img/>

2.元素 元素，即标记（标签）

3.元素嵌套

元素之间可以相互嵌套，来形成更为复杂的页面结构

元素嵌套：在一个元素中，又出现另外一个元素

ex:

<p>

<a>

<img>

</a>

</p>

以上是正确的嵌套

注意：

1.注意嵌套顺序（完整嵌套）

2.注意嵌套代码的缩进关系

ex:<p><a><img></a></p>正确，不推荐

4.属性和值

1.什么是属性？

属性是用来修饰元素的

2.语法

属性的声明必须位于开始标记里

<标记 属性="值" 属性="值" ...></标记>

ex：设置p标记的align属性，值为center,设置id属性，值为p1

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

3.标准属性

id:元素在页面中的唯一标识

title:鼠标移入到元素上时，所提示的文字

class:CSS中，引入类选择器

style:CSS中，定义元素内联样式

5.注释

在页面中编写完成后，浏览器不会解析执行的内容

可以将代码的解释说明写在网页中便于被其它程序员查看

语法：<!--注释内容-->

注意：

1.本身不能嵌套

2.不能写在标记中

6.HTML和XHTML和HTML5

w3c负责定制和推广HTML

1999年12月24日w3c推出的HTML4.01版本

2000年1月26日w3c推出的XHTML1.0版本

如:

单标记

<br>必须<br/> 如果在XHTML中<br>错误

2014.9月HTML5

目标：实现更为简洁的HTML代码

能少写就少写，能不写就不写

<p align="center"></p>

<p align=center></p>

<input readonly="readonly">

<input readonly>

**3.HTML标准结构**

1.HTML标准结构

1.文档类型声明

<!doctype html>

作用：指定HTML的版本和风格

2.HTML页面

由一对根标记组成

<html></html>

在根标记中包括两部分：

1.文件头<head></head>定义网页全局信息

2.文件主体<body></body>显示页面的主要内容

2.head元素

1.页面的头部内容

作用：定义全局信息

2.是其他头元素的容器

1.定义网页标题

<title>标题内容</title>

2.定义网页编码格式

<meta charset="utf-8">

<meta name="keywords" content="关键字的位置">

<meta name="descoration" content="描述内容">

3.<script><script>定义或引用JS文件

4.<style></style>定义内部样式

5.<link>引入外部样式

3.body元素

语法：<body></body>

属性：

1.text

作用：控制当前网页的文本颜色

取值：表示颜色的英文单词（red）

2.bgcolor

作用：控制当前网页的背景颜色

取值：颜色英文单词

3.文本（重点）

1.特殊符号

&nbsp; 空格

&lt; <

&gt; >

&copy; @

&yen; ￥

2.文本标记

1.文本样式

<i></i>斜体

<u></u>下划线

<s></s>删除线

<b></b>加粗

<sup></sup>上标

<sub></sub>下标

2.标题元素

以标题的形式显示文本内容

语法：<hn></hn> n:1-6数字

<h1></h1> 一级标题

...

<h6></h6> 六级标题

特点：

1.改变字体大小

2.加粗

3.上下文之间垂直空白间距

3.段落标记

语法：<p>内容</p>

以段落的形式显示文本

文字大小采用默认大小，段落元素独占成行，而且距离其它元素会有垂直空白间距

4.换行元素

语法：<br>或<br/>

5.分割线元素

语法：<hr>

属性：

size:取值以px为单位的数值

color:颜色

width:宽度 以px为单位的数值

align:线条的水平对齐方式

6.预格式化

保留源文件中的格式，保留源文件的换行和空格效果

语法：<pre></pre>

7.分区元素

1.块分区

语法：<div></div>

形式：

1.独立成行

2.无任何文本显示效果

作用：布局

2.行分区

语法：<span></span>

形式：

1.多个span元素在一行中显示

2.无任何特殊效果

作用：处理一行文字的不同效果

3.行内元素与块级元素

1.行内元素

多个元素在一行内显示

ex:span,i,b,u,s,sup,sub,a,img

作用：大部分的行内元素为了处理文本的显示效果

2.块级元素

每个块级会独自成行，即前后有换行的效果

ex:div,p,h1-h6,(结构标记)

作用：做布局

补充：Editplus的操作

1.指定创建模板

1.创建模板页（\*.html）

2.Tools->Preferences->File->Templates->HTML->file name重新选定模板

2.快速查看网页效果

1.快捷键：ctrl+B

2.Tools->Preference->Tools->Browser->指定浏览器

3.代码折叠

view->Code Folding->use code folding

4.代码缩进线条

View->Indent Guide

5.代码行号显示

view->Line Number

6.取消.bak文件

Tools->Preference->File->取消Creat backup file when saving

7.修改编码格式

Tools->Preference->file->Default Encodeing 设置为UTF-8即可

## Day02

1.图像和链接

1.url

Unifrom resource locator统一资源定位器，俗称：路径。

作用：用来标识资源的位置

2.url表现形式

1.绝对路径

1.什么是绝对路径？

是完整路径的表现形式

1.网络资源

通信协议：ex:http:

主机:域名/IP地址 ex:www.baidu.com

文件目录结构：ex:jd.com/index/css

文件名称：myStyle.css

2.本机资源

从最高盘符位置处开始查找，一直到资源文件所在的目录

D:\1703\资料\HTML素材\Day02~Day03 素材\COOKBOOK\

2.相对路径

1.什么是相对路径？

从当前文件位置处开始，一直资源文件所在的位置，他所经过的路径就是相对路径

1.当前文件和资源文件在同目录下

直接通过资源文件名称进行引用

jd\_logo.png

2.资源文件在子目录中

先进入到子目录，然后再引用

images/jd\_logo1.png

3.资源文件在父目录中

先返回到父级目录，然后再引用

../表示返回上一级

../jd\_logo2.png

3.根相对路径

永远都是从web站点的根目录处开始查找

由/作为开始，表示web站点的根目录

/index/images/a.jpg

3.图像

1.图像格式

jpg(jpeg):压缩

png:背景透明

gif:动画

2.图像元素

语法：<img>

属性：

src:要显示图片的路径（url）

width:宽度

height:高度

title:鼠标移入到元素上时，提示文字

alt:图片出错，显示文字

4.链接

1.作用

链接，又称为超级链接，设置页面中允许被点击的内容，完成页面间的跳转。

2.语法

标记：<a>内容</a>

属性：

href:链接的url(要跳转到的页面的地址)

target:目标，指定打开网页的形式

取值：

1.\_self在自身的标签页中打开（默认值）

2.\_blank在新的标签页中打开

title：鼠标放到链接上时的提示内容

3.链接的表现形式

1.目标为下载资源

<a href="\*.rar/\*.zip">下载</a>

2.电子邮件

<a href="mailto:g-yangyan@tedu.cn">打开邮箱</a>

3.返回当前页面的顶部

<a href="#">返回顶部</a>

4.链接到JS

<a href="javsscript:"></a>

4.锚点

1.作用

在页面中的某行处做一个标记

方便页面能够随时跳转到标记位置处

2.使用方式

1.定义锚点

1.通过a标记的name属性定义锚点

ex:<a name="锚点名称"></a>

2.通过任意标记的id属性定义锚点

ex:<any id="锚点名称"></any>

2.链接到锚点

跳转到本页面锚点处

<a href="#锚点名称"></a>

跳转到其它页面锚点处

<a href="其它页面URL#锚点名称"></a>

2.表格

1.表格的作用

按照一定的格式来显示数据

表格是由单元格组成，按照从左到右，从上到下的顺序排列的

表格中的数据最终要保存在单元格中

2.使用表格

1.创建表格

1.定义表格：<table></table>

2.创建表行：<tr></tr>

3.创建列（单元格）：<td></td>

练习：

创建一个3行4列的表格

步骤：

1.创建表<table></talbe>

2.创建3行<tr></tr>

3.每一行（<tr>）中添加4对<td>

2.表格的属性

1.table属性

1.width 设置表格的宽度

2.height 设置表格的高度

3.align 设置表格的水平对齐方式

取值：left/center/right

4.border 边框，指定表格边框

5.bgcolor 设置表格的背景颜色

6.cellspacing 设置单元格的外边距(单元格与单元格之间的间距)

7.cellpadding 设置单元格的内边距（单元格与内容之间的间距）

2.tr的属性

1.align 设置该行内容的水平对齐方式

2.valign 设置该行内容的垂直对齐方式

取值：top/middle/bottom

3.bgcolor 设置该行的背景颜色

3.td的属性

1.align 2.valign

3.width 4.height

5.colspan设置单元格跨列 6.rowspan设置单元格跨行

注意：

1.每行中的指定列的宽度，都是一致，默认情况下以最宽的为主

2.一行中所有列的高度都是一致的，默认以最高的为主

3.表格的标题

标记：<caption>标题</caption>

注意：

1.caption必须紧跟table标记之后

2.一个表格最多只能有一个标题

4.th行标题或列标题，字体有加粗效果，放在tr中

3.表格复杂使用

1.行分组

可以将表分为3个部分：

1.表头：<thead></thead>

2.表主体：<tbody></tbody>

3.表尾：<tfoot></tfoot>

2.不规则表格

通过td的colspan(跨列)和rowspan(跨行)属性设置

1.跨列

从指定的单元格位置开始，横向向右合并几个单元（包括自己）

注意：被合并单元格，要删除

属性：colspan

取值：数字（合并单元格的个数）

2.跨行

从指定的单元格位置开始，纵向向下合并几个单元格（包含自己）

注意：被合并的单元格，要删除

属性：rowspan

取值：数字

3.表格的嵌套

必须在单元格中嵌套另一个表格

<table>

<tr>

<td>

<table>

<tr>

<td></td>

<td></td>

<td></td>

</tr>

</table>

</td>

</tr>

</table>

3.列表

1.列表的作用

将具有相似特征或具有先后顺序的内容按照从上往下或者从左往右的顺序排列

2.列表分类

1.有序列表

2.无序列表

3.语法

列表组成：

1.列表类型

指定列表是有序还是无序

2.列表项

列表中所显示的内容

1.有序列表

列表类型：<ol></ol> order list

列表项：<li></li> list item

<ol>

<li></li>

<li></li>

...

</ol>

属性：

type 指定列表项前标志的类型，默认为数字排序

取值：

1 数字（默认值）

a 小写英文字母

A 大写英文字母

i 小写罗马数字

I 大写罗马数字

start 标志的起始值 默认1

ex: <body>

<ol type="A" start="5">

<li>葫芦娃</li>

<li>喜洋洋</li>

<li>美少女战士</li>

</ol>

</body>

2.无序列表

列表类型：<ul></ul> unorder list

列表项：<li></li>

<ul>

<li></li>

<li></li>

...

</ul>

属性：

type 列表项前的标识

取值：

disc:实心圆

circle:空心圆

square:实心方块

none:不显示标识

3.列表嵌套

允许在一个列表中再嵌套另一个列表

注意：在列表项（li）中去嵌套另一个列表

<ol>

<li>

<ul>

<li></li>

...

</ul>

</li>

</ol>

4.自定义列表

1.作用

往往用于要给出一类事物的定义情形

2.语法

<dl></dl>：创建一个定义列表

<dt></dt>:定义要解释的专有名词/术语

<dd></dd>:定义对专有名词/术语进行解释的内容

使用场合：图文混排的时候使用

## Day03

1.结构标记

1.什么是结构标记？

在HTML5中，专门提出的一组用来制作网页布局的

目的：取代div布局，提升布局代码的语义性和可读性。

ex:

<div></div> --> <header></header>

2.常用结构标记

1.header元素

语法：<header>内容</header>

作用：定义网页或其他部分内容的页眉信息（网页靠顶部的内容）

2.nav元素

语法：<nav>内容</nav>

作用：定义页面的导航内容

3.section元素

语法：<section>内容</section>

作用：定义网页的主体内容

4.aside元素

语法：<aside>内容</aside>

作用：定义网页任何一个位置的边栏

5.footer元素

语法：<footer></footer>

作用：定义网页的底部信息，用户不太关注的内容

6.article元素

语法：<article></article>

作用：定义与文章相关的内容部分，比如：论坛帖子，微博条目，简短新闻...

2.表单（重点、难点）

1.表单的作用

用于显示，收集数据，并提交信息给服务器

完整的表单处理包含以下两部分：

1.实现数据交互的可见界面元素（前端）

2.提交后的表单数据处理(服务器端)

2.表单元素

标记：<form></form>

属性：

1.action

定义表单提交时发生的动作

指定服务器端处理程序的地址(url)

具体取值要与服务器端人员协同完成

如果省略不写，默认提交给本页

ex:http://www.jd.com/index.jsp

2.method

定义表单数据的提交方式

取值：

1.get(默认值)

1.明文提交，提交的数据会显示在地址栏上

2.安全性不高

3.提交数据有大小限制-2kb

4.场合：向服务器要数据时使用：搜索关键字提交

2.post

1.隐式提交，不会将提交信息显示在浏览器的任何位置

2.安全性较高

3.提交数据无大小限制

4.场合：要传递数据给服务器进行处理时，使用post提交：注册，登录，上传文件等

3.delete,put...

3.enctype

作用：指定数据进行编码的方式

取值：

1.application/x-www-form-urlencoded

默认值，可以将表单中的普通文本，特殊字符，标点符号都进行二进制编码然后进行提交

2.multipart/form-data

可以将表单中的文件进行二进制编码再提交

字符，标点符号无法进行编辑提交

3.text/plain

可以将表单中的普通字符进行二进制编码再提交，其余都无法提交

4.name 定义表单名称，JS用到比较多

5.id 定义唯一标识

3.表单控件

1.作用：用于接收用户数据并依托于表单提交服务器

2.表单控件的分类

1.input元素：文本输入框（文本框，密码框），按钮，单选按钮，复选框（多选按钮）...

标记：<input>

2.textarea元素

多行文本域

标记：<textarea></textarea>

3.select和option元素

选项框

标记：

<select>

<option></option>

</select>

4.其它元素

3.<input>元素(重点)

1.作用：用于收集用户信息

2.语法

标记：<input>

主要属性：通有属性，所有的input元素都具有的属性

1.type

根据type的值，来创建各种类型的输入控件，比如：文本框，密码，按钮...

2.name

控件名称，提供给服务器用

注意：如果不设置控件名称，则数据无法提交

匈牙利命名法：控件缩写+功能名称

3.value

控件的值，最终提交给服务器的值

4.disabled

禁用控件

表示控件不可用（不能获取值，不能被提交）

注意：该属性无值

3.input控件详解

1.文本框和密码

文本框：<input type="text">

密码框：<input type="password">

专有属性:

1.maxlength

限制输入字符的数量

取值：数字

2.readonly

只读

该属性无值

辨析：disabled:禁用，不能被提交

readonly:只读，允许被提交

3.name

文本框和密码框的缩写：txt

ex：<input type="text" name="txtName">

<input type="password" name="txtPwd">

4.placeholder

占位符

2.单选按钮和复选框

单选按钮：<input type="radio">

复选框：<input type="checkbox">（多选按钮）

专有属性：

1.name

名称，同时具有分组作用

单选按钮：一组中只能有一个按钮被提交

复选框：一组数据后台获取时名称相同

名称缩写：

radio-->rdo

checkbox-->chk

2.value

值，被选中后提交的值

3.checked

设置默认被选中

该属性无值

3.按钮

1.提交按钮

<input type="submit">

作用：提交表单的数据给服务器

2.重置按钮

<input type="reset">

作用：将表单恢复到初始化的状态

3.普通按钮

<input type="button">

作用：由用户自己定义功能(JS)

按钮的共有属性：

1.name

定义按钮名称，缩写：btn

2.value

按钮上的文字

4.其它按钮：

1.图片按钮（提交功能）

<input type="image">

属性：src

ex: <input type="image" src="btn\_login.JPG">

2.按钮（提交按钮）

<button>内容</button>

4.隐藏域和文件选择框

1.隐藏域

<input type="hidden">

作用：要提交给服务器，但是用户不想看的数据，放在隐藏域中。

2.文件选择框

<input type="file">

作用：允许打开文件选择框，选择文件进行上传

共有属性：

name: 名称

缩写：txt

value: 值

3.注意

使用文件选择框上传文件是，有以下两点要求：

1.form的method属性必须是post值

2.enctype属性值必须是multipart/form-data

4.textarea元素

1.作用：完成多行文本的录入功能

2.语法

标记：<textarea>默认值</textarea>

属性：

1.name

名称，缩写为：txt

2.cols

指定文本区域的列数

变相设置控件的宽度

3.rows

指定文本区域的行数

变相设置控件的高度

4.readonly 只读，不能写，但能提交值给服务器

5.选项框

分为：

下拉选项框

滚动列表

<区别：取决于size值：=1，为选项框，>1，为滚动列表>

语法：

1.标记

<select></select>

创建选项框(滚动列表)

属性：

1.name

名称，缩写：sel

2.size

指定选项框默认能显示几项内容

默认值为1

如果取值大于1，则为滚动条列表

3.multiple

设置允许多选

该属性无值

2.<option></option>

作用：定义选项框的子选项

属性：

1.value

选项的值

2.selected

预选中，设置默认被选中的选项

该属性无值

6.其它元素

1.label元素

1.作用

关联文字与表单控件，关联后，点击文字如同点击表单控件一样

2.语法

标记：<label></label>

属性：

for：表与该元素相关联的表单控件的ID属性值

ex:

性别：<input id="rdoMale" type="radio" name="rdoGender" value="M" checked>

<label for="rdoMale">男</label>

2.为控件分组

标记：<fieldset></fieldset>

子元素：<legend></legend> 指定分组的标题

3.其它常用标记

浮动框架(多个HTML文件的嵌套调用，多用于页面头部、尾部的调用)

1.作用

可以在一个浏览器窗口中同时显示多个html文档

2.语法

1.标记

<iframe>文字内容</iframe>

注意：当浏览器不支持iframe元素时，文字内容就会显示

2.属性

1.src

浮动框架中的网页的url

即要引入的网页的地址

2.height

高度

3.width

宽度

4.frameborder

浮动框架的边框，如果不想要边框的话，设置为0

4.新表单元素（HTML5新标记） 缩写：txt

属性：required 非空限制

1.电子邮件类型

<input type="email">

2.搜索类型

<input type="search">

3.url类型

数据必须符合url规范

<input type="url">

4.电话号码

<input type="tel">

移动端：弹出数字键盘

5.数字类型

<input type="number">

属性：

min 定义控件接受的最小值

max 定义控件接受的最大值

step 控制控件递增的步长，默认为1

6.范围类型

作用：允许选择指定范围内的一个值

<input type="range">

属性：

min:指定范围的最小值（下限值）

max：指定范围的最大值（上限值）

step：值变化的步长

value：设置初始值

7.颜色类型

作用：颜色拾取控件

<input type="color">

8.日期类型

<input type="date">

作用：允许用户选择日期

9.周类型

作用：与date相似，但是只能选择周

<input type="week">

10.月份类型

作用：只能选择月份

<input type="month">

# CSS

## Day01

赵旭

zhaoxu@tedu.cn

CSS3Basic : C3基础

### 1、CSS概述

1、问题

1、在HTML中，所有的文本颜色变为红色

2、在HTML中，让所有的div变为蓝色

3、在HTML中，让所有的div变为黄色

HTML属性带来的弊端

1、相同的事情要用不同的属性完成

<body text="">

<font color="">

2、相同的事情重复的做(可重用性较低)

2、什么是CSS

Cascading Style Sheets

层叠样式表，级联样式表，通常称为：样式表

用户在HTML中定义元素样式

1、实现内容与表现相分离

2、提供代码的可重用性

3、CSS 与 HTML之间的关系

HTML ：定义网页结构

CSS ：构建页面的样式

注：HTML属性 与 CSS内容 如果能完成相同的事情，优先使用 CSS

### 2、CSS语法

1、使用CSS样式表的方式

1、内联方式(行内样式)

将 CSS样式 定义在单个HTML元素内

<ANY style="样式声明1;样式声明2;...样式声明n;"></ANY>

样式声明：(重点)

样式属性1:值;样式属性2:值;

ex:

样式属性 属性值

color 任意合法颜色值

background-color 任意合法颜色值

font-size 以px或pt为单位

2、内部样式表

在 <head>元素中的<style></style>里编写若干"样式规则"

<html>

<head>

<style>

样式规则1

样式规则2

...

样式规则n

</style>

</head>

</html>

样式规则：

选择器{

样式声明1;

样式声明2;

...

样式声明n;

}

ex：

h1{

样式声明 ....

}

特点：

提升了样式的可重用性和可维护性

3、外部样式表

将样式规则以及声明，存放在 独立的 样式文件中(\*\*.css)

使用步骤：

1、创建样式表文件并编写样式

2、在使用的网页中，使用 <link>标记链接到外部样式表

<head>

<link rel="stylesheet" href="样式表URL">

</head>

2、样式表的特征

1、继承性

直接使用父元素声明好的样式

大部分样式属性是可以被继承的

2、层叠性

可以为一个元素定义多个选择器

如果样式不冲突时，他们可以合并为一个，都应用在当前元素上

3、优先级

样式定义冲突时，会按照不同使用方式的优先级来应用样式

浏览器的缺省设置(UA Stylesheet) 最低

外部样式表或内部样式表 中

就近原则 - 后定义者优先

内联样式 高

4、!important 规则

!important 可以显示调整优先级，永远都以：!important 的为准

样式属性:值 !important;

注意：尽量少用

3、排错

1、Unknown property name

不认识的属性名称 - 属性名写错了

2、Invalid property value

未验证的属性值 - 属性值写错了

3、缺失分号

一部分样式 不识别

4、中文标点

一部分样式 不识别

4、选择器(重点&难点)

1、作用

p{} , div{} ,h1{}

规范了页面中哪些元素能够使用定义好的样式

为了匹配页面的元素

2、详解

1、通用选择器

作用：用于匹配页面中所有的元素

语法：\* { ... ... }

注意：

尽量少用 通用选择器

大部分属性是可以通过继承来取代 \*

2、元素选择器

又称为：标签选择器，标签样式，元素样式

作用：用于定义或重写页面中某元素的默认样式

语法：页面元素作为选择器 {}

h1{ ... }

div{ ... }

...

input{ ... }

3、类选择器

作用：用于定义一些元素的通用样式。类选择器定义好之后，允许被任意元素的class属性进行引用

语法：

.类名{}

类名规范：

1、英文，数字，下滑下(\_),连字符(-)

2、不能以数字开头

页面元素 通过class的属性值 ，可以对类选择器进行引用

<ANY class="类名"></ANY>

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

类选择器-多类选择器的引用方式

即 让一个元素同时引用多个类选择器

语法：

<ANY class="class1 class2 class3">

类选择器-分类选择器的定义方式

即 将元素选择器 和 类选择器结合到一起来定义

目的：为了实现对某种元素中不同样式的的细分控制

语法：元素选择器.类选择器{}

div.important{} : 匹配页面中 class 为 important 的 div元素

span.redBack{} :匹配页面中 class 为 redBack 的 span元素

4、ID选择器

作用：用于 匹配 页面中 指定 ID值的 元素（专属定制/私人订制）

语法：#ID值{}

<div id="container"></div>

样式表中：

#container{

}

5、群组选择器

作用：定义多个选择器中的通用样式

语法：以 , 隔开的选择器列表

ex:

.heavy,#container,div.important{

font-size:25px;

color:orange;

}

等价于

.heavt{font-size:25px;color:orage;}

#container{font-size:25px;color:orage;}

div.important{font-size:25px;color:orage;}

6、后代选择器

后代元素：出现在某元素的任意层级的子元素都可以称之为 "后代元素"

语法：

选择器1 选择器2{ .. ..}

ex:

#d1 span{

/\*定义 id为d1 中 出现的所有span 元素的样式\*/

}

7、子代选择器

子代元素：出现在某元素中，只具备一级层级关系元素称之为 "子代元素"

语法：

选择器1>选择器2{}

8、伪类选择器

作用：匹配页面元素不同状态的选择器

语法：

:伪类选择器{}

选择器:伪类选择器{}

分类：

1、链接伪类

1、:link , 定义未被访问的超链接状态

2、:visited,定义访问过的超链接状态

2、动态伪类

1、:hover,定义鼠标悬停在元素时的状态

2、:active,定义元素被激活时的状态

3、:focus,定义元素获取焦点时的状态

3、目标伪类

4、结构伪类

5、否定伪类

### 3、尺寸 与 边框

1、单位

1、尺寸单位

1、%

相对单位

2、in

1in=2.54cm

3、cm

厘米

4、mm

毫米

5、pt

磅(点) -> Point

1pt = 1/72in

多数用在 文字 大小

6、px

Pixel ，像素，计算机屏幕上的一个点

屏幕分辨率：1024\*768

7、em

倍数，父元素文字大小倍数

1em ：父元素字体原始大小

2em ：父元素字体大小的2倍

8、rem

倍数，相对于根元素(html)设置字体的尺寸倍数

2、颜色单位

在css中合法的颜色取值

1、rgb(r,g,b)

r:red (0-255)

g:green (0-255)

b:blue (0-255)

ex:

#d1{

color:rgb(0,0,0);/\*黑色\*/

}

2、rgb(r%,g%,b%)

3、rgba(r,g,b,alpha)

alpha : 透明度

0-1 之间的数字

4、#rrggbb

由6位 十六进制数字 表示一个颜色

十六进制：0-9 A-F 组成

A:10,B:11,C:12,D:13,E:14,F:15

ex:

#000000 : 黑色

#111111 : 深深深灰

#eeeeee : 浅浅浅灰

#ffffff : 白色

#1a2b3c : ?

5、#rgb

#rrggbb 的缩写，每两位数字相同时，可以采用缩写

#000000 -> #000

#aabbcc -> #abc

#1abbcc -> 无法缩写

6、颜色的英文表示法

red,blue,darkred,lightright,

2、尺寸属性

1、作用

用于设置元素的宽度和高度

单位通常为 px 或 %

2、语法

1、宽度

width : 指定元素的宽度

max-width : 最大宽度

min-width : 最小宽度

注意：

块级元素默认宽度：父元素宽度的100%

行内元素默认宽度：以内容为准

2、高度

height : 指定元素的高度

max-height: 最大高度

min-height: 最小高度

注意：

所有元素默认高度：以内容为准

3、哪些元素允许修改尺寸

1、块级元素 都允许修改尺寸

div,p,h1~h6,section,header,footer,...

2、大部分行内块元素允许修改

input 元素允许改，但 checkbox,radio 除外

3、存在 width，height 属性的HTML元素允许修改

<img width="" height="">

<table width="" height="">

4、溢出

1、什么是溢出

使用尺寸属性控制元素大小时，如果内容所需空间大于元素本身，会导致内容 "溢出"

2、溢出处理属性

属性：

overflow

overflow-x : 横向溢出处理

overflow-y : 纵向溢出处理

取值：

1、visible

默认值，溢出可见

2、hidden

隐藏，溢出隐藏

3、scroll

滚动，让元素显示滚动条，溢出时可用

4、auto

自动，溢出时显示滚动条并可用

3、边框属性（下节讲）

<div id="d1">

<span>#d1 中的 span</span>

<p>

<span>

<b>#d1 中的 p 中的 span 中的 b</b>

</span>

</p>

</div>

## Day02

### 1、CSS概述

CSS 与 HTML 之间的关系

<p align="center"></p>

css:

p{

align:center;/\*错误\*/

}

### 2、CSS语法

1、CSS的使用方式

1、内联方式

<ANY style="样式声明;">

样式声明的组成：样式属性:值;

2、内部样式表

<style>

样式规则：

选择器{

样式声明;

}

</style>

3、外部样式表

1、创建独立的 css 文件，并编写样式

编写 样式规则

2、在使用的页面中进行引入(链接)

<link rel="stylesheet" href="url"/>

2、选择器

1、通用选择器

\*{}

2、元素选择器

html{}

3、类选择器

.类名{}

多类选择器的引用方式

<ANY class="class1 class2 class3">

分类选择器的定义方式

元素选择器.类选择器{}

4、ID选择器

专属定制

#ID{}

5、群组选择器

选择器1,选择器2,选择器3,...{}

6、后代选择器

后代：无限子级

选择器1 选择器2{}

7、子代选择器

子代：一级层级 (下一级)

选择器1>选择器2{}

8、伪类选择器

1、链接伪类

:link

:visited

2、动态伪类

:hover (重点)

:active

:focus

3、尺寸属性

1、宽度

width

max-width

min-width

2、高度

height

max-height

min-height

3、允许设置尺寸属性的元素

1、块级元素

2、行内块

表单元素(radio,checkbox除外)

3、具备 width 和 height html属性的元素

<img>

<table>

4、溢出

空间已经包含不下所有的内容

属性：

overflow

overflow-x

overflow-y

取值

1、visible : 默认值，可见

2、hidden : 隐藏

3、scroll

4、auto

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

img : display:inline;(行内)

p: display:block;(块级)

div: display:block;(块级)

table : ?

ul : ?

li : ?

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

今日内容：

### 1、边框属性

1、边框

1、简写方式

border:width style color;

width:边框的宽度，以px为单位的数值

style:边框的样式

solid : 实线

dotted : 虚线(点状)

dashed : 虚线(线状)

color:边框的颜色，合法颜色值

特殊：transparent , 透明

ex:

border:1px solid #000;

border:0; 或 border:none;

2、单边定义

只定义某一方向的边框的 粗细，样式，颜色

border-方向:width style color;

方向：top/right/bottom/left

ex:

border-top:1px solid red;

3、单属性定义

定义四个边框的某一个属性值

border-属性:值;

属性：width/style/color

ex:

border-color:#e4393c;

4、单边单属性定义

定义某一方向的具体某一属性值

border-方向-属性:值;

方向：top/right/bottom/left

属性：width/style/color

ex:

1、下边框的颜色为 #ddd

border-bottom-color:#ddd;

2、右边框的样式为 dotted

border-right-style:dotted;

5、边框的组成

是由四个等边三角形组成的

2、边框倒角

将 元素的 四个直角 倒换成 圆角

语法：

属性：border-radius

取值：

1、具体数值

2、百分比

采用当前元素 宽度 和 高度的占比 来作为倒角圆的半径

单角定义:

border-top-left-radius:左上角倒角半径值;

border-top-right-raius:右上角...

border-bottom-left-radius:

border-bottom-right-radius:

3、边框阴影

语法：

属性：box-shadow

取值：

取值为多个值的列表

h-shadow v-shadow blur spread color inset;

h-shadow:必须的，阴影的水平偏移距离

取值为正，阴影右偏移

取值为负，阴影左偏移

v-shadow:必须的，阴影的垂直偏移距离

取值为正，阴影下偏移

取值为负，阴影上偏移

blur:模糊距离

spread:阴影大小

color:颜色

inset:将默认的外阴影改为内阴影

4、边框轮廓

1、什么是轮廓

是绘制与元素周围的一条线，位于边框之外，起到突出显示的作用

2、语法

outline:width style color;

outline-width:轮廓的宽度

outline-style:

outline-color:

outline:none; 或 outline:0;

### 2、框模型

1、框模型

框：Box，泛指页面中所有的元素

页面元素皆为框

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

外边距：元素(框) 与 元素(框)之间的距离

内边距：元素(框) 与 内容之间的距离

元素的占地尺寸

元素总宽度=左右外边距 + 左右边框 + 左右内边距 + width;

元素总高度=上下外边距 + 上下边框 + 上下内边距 + height;

元素的可视化尺寸(边框内)

边框内宽度=左右边框+左右内边距+width;

边框内高度=上下边框+上下内边距+height;

2、外边距

1、什么是外边距

元素边框(边缘)之外的距离

2、语法

属性：

margin:值;

margin-top/right/bottom/left:值;

取值：

1、具体数值 px

2、百分比 %

3、auto

默认情况下，可以自动计算左右外边距

实现 块级元素 水平居中对齐

注意：必须为元素设置宽度才能居中

4、负数

向相反的方向移动元素

margin-top

取值为正，元素下移动

取值为负，元素上移动

margin-left

取值为正，元素右移动

取值为负，元素左移动

简写方式：

margin:value; 四个方向外边距值

ex

margin:10px;

margin:v1 v2; v1:上下 , v2:左右

ex

margin:0 auto;

margin:v1 v2 v3;v1:上 v2:左右 v3:下

ex

margin:0 auto 15px;

margin:v1 v2 v3 v4;上 右 下 左

3、页面中具备默认外边距的元素

在页面中，通过 CSS Reset(CSS重写)的方式将默认的外边距全部都清除

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

margin:0;

}

4、特殊问题

1、外边距的合并

当两个垂直外边距相遇时，他们将形成一个外边距，合并后的外边距值等于两个元素外边距中较大的数值

2、外边距的溢出

在某些特殊的情况下，为子元素设置上外边距时，有可能会作用到父元素上

条件：

1、父元素没有上边框

2、必须为第一个子元素设置上外边距

解决方案：

1、给父元素加上边框

弊端：父元素高度会发生变化

2、通过给父元素增加上内边距取代子元素上外边距

弊端：改变父元素高度

3、在父元素中增加一个空table

弊端：多了一个空元素

4、... ...

后续完成...

3、为行内元素 和 行内块元素 设置 上下外边距

行内元素：上下外边距无效(img除外)

行内块元素：设置上下外边距时，整行元素都跟着动

3、内边距

1、什么是内边距

元素的边框(边缘) 到 内容之间的距离

注意：内边距会扩大元素边框所占用的区域

2、语法

属性：

padding

padding-top/right/bottom/left

取值：

1、px

2、%

3、auto

简写

padding:v1; 四个方向内边距

padding:v1 v2;上下 左右

padding:v1 v2 v3;上 左右 下

padding:v1 v2 v3 v4; 上 右 下 左

3、特殊效果

为行内元素 和 行内块元素 增加上下内边距时有特殊效果

特殊效果：只会影响自己，并不会影响其他元素

4、box-sizing

作用：指定 框模型的 计算方式

div{

width:471px;

border-left:2px solid #ddd;

border-right:6px solid #fff;

padding-left:18px;

padding-right:3px;

}

Element Width = 471 + 2 + 6 +18 + 3;

属性：box-sizing

取值：

1、content-box

元素的width属性值，不包含padding和border

Element Width = width + padding-left/right + border-left/right;

2、border-box

元素的width属性值，会包含padding 以及 border

Element Width = width;

ex:

border-sizing:border-box;

### 3、背景属性

1、背景颜色

属性：background-color

取值：合法颜色值 或 transparent

注意：背景颜色会填充到边框，内边距以及内容区域

2、背景图片

属性：background-image

取值：url("图片url")

ex:

background-image:url(images/jd\_logo.png);

背景图 与 <img> 有什么区别？

<img>是独立的元素

背景图：不是元素，是某元素背景的衬托

3、背景重复(平铺)

当元素区域大于背景图的话，默认是以平铺的方式出现

属性：background-repeat

取值：

1、repeat

默认值，水平和垂直方向都平铺

2、repeat-x

只在水平方向平铺

3、repeat-y

只在垂直方向平铺

4、no-repeat

不平铺

4、背景图片尺寸

属性：background-size

取值：

1、width height

2、width% height%

3、cover

覆盖，将背景图等比扩大，直到背景图覆盖到元素所有区域为止

4、contain

包含，将背景图等比扩大，直到背景图的一个边缘碰到元素右边或下边为止

5、背景图片固定

作用：让背景图一直保持在可视化区域中，不随着滚动条而发生改变

属性：background-attachment

取值：

1、scroll

默认值，滚动

2、fixed

固定

ex:

background-attachment:fixed;

注意：尽量为 body 设置背景

6、背景定位/位置

属性：background-position

取值：

1、x y

x ：背景图横向偏移距离

x 取值为正，背景图，向右偏移

x 取值为负，背景图，向左偏移

y ：背景图纵向偏移距离

y 取值为正，背景图，向下偏移

y 取值为负，背景图，向上偏移

2、x% y%

0% 0% : 左上角(默认:左上角)

100% 100% : 右下角

50% 50% : 正中间

3、关键字

x : left / center / right 取代

y : top / center / bottom 取代

CSS Sprites

雪碧图，精灵图

将若干幅小图像封装到一幅大图像中，以便减少http请求次数

实现步骤：

1、在页面中，根据要看的图像的尺寸，创建一个一模一样大小的元素

2、为该元素设置背景图，并且通过 背景图像定位 将要显示的图移动到 元素区域内

ex:

#Monster{

width:33px;

height:18px;

background-image:url(images/iconlist\_1.png);

background-position:-243px -112px;

/\*移动背景图像的位置\*/

}

(即：将url地址中的目标设为id=Monster，并将Monster移动到所设置的位置)

7、背景属性(简写)

background:color url() repeat attachment position;

ex:

background:#f00;

background:url(iconlist\_1.png) -243px -112px;

## Day03

\*\*\*\*\*\*\*\*\*\*\*复习\*\*\*\*\*\*\*\*\*\*

1、边框

1、边框

border:width style color;

border-方向:width style color;

border-属性:值;

border-方向-属性:值;

2、边框倒角

border-radius:值;

3、边框阴影

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

4、轮廓

outline:width style color;

outline:none;

2、框模型

1、外边距

属性：

margin

margin-top/right/bottom/left

注意：

1、外边距合并

2、外边距溢出

3、行内元素 行内块元素 设置垂直外边距

2、内边距

属性：

padding

padding-top/right/bottom/left

注意

1、行内元素 和 行内块元素 设置垂直外边距时，只影响自己

3、box-sizing

1、content-box

元素的 width和height 是不包含 border 以及 padding

2、border-box

元素的 width 和 height 会包含 border 以及 padding

3、背景

1、background-color

2、background-image

3、background-repeat

no-repeat,repeat,repeat-x,repeat-y

4、background-size

1、x y

2、x% y%

3、cover

4、contain

5、background-attachment

scroll,fixed

6、background-position

1、x y

2、x% y%

3、关键字

x : left / center / right

y : top / center / bottom

CSS Sprites 减少HTTP 的请求次数

7、background

background:color url repeat attachment position;

background:red;

background:url(a.jpg);

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

### 1、渐变

1、什么是渐变

两种或多种颜色平缓过渡的效果

2、重要元素

色标：指定每个颜色的值以及出现的位置

3、分类

1、线性渐变

以直线的方式填充渐变颜色

2、径向渐变

以圆形的方式填充渐变色

3、重复线性渐变

4、重复径向渐变

4、语法

1、属性

background-image

2、取值

1、线性渐变

linear-gradient(angle,color-point1,color-point2,color-point3);

1、angle

1、关键字

1、to top :

从下向上 填充渐变色

第一个色标，在最底下出现

最后一个色标，在最上面出现

2、to right

从左向右 填充渐变色

第一个色标，在最左边出现

最后一个色标，在最右边出现

3、to bottom

从上向下 填充渐变颜色

4、to left

从右向左 填充渐变颜色

2、具体角度值

0deg ~ 360deg

2、color-point

色标：包含颜色 及其 出现的位置

ex:

1、red 0%

从开始的位置 显示为红色

2、blue 50%

中间位置处 显示为蓝色

3、yellow 200px

4、red

省略位置，由浏览器自动计算

2、径向渐变

属性：background-image

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

1、size at position

size : 半径大小，以px或%为单位

at : 关键字，不能省略

position : 圆心所在位置

3、重复线性渐变

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

background-image:repeating-linear-gradient(to top,red 0%,blue 10%);

4、重复径向渐变

background-image:repeating-radial-gradient([size at position],color-point1);

5、浏览器兼容性

渐变：CSS3 提出的新属性

C3新属性，放在 老版浏览器中 支持性不好!

目前为止，主流浏览器的主流版本，对C3支持性特别好!但对于不支持C3新属性的版本浏览器，需要通过增加 "浏览器前缀"实现兼容性

常用浏览器前缀：

1、Firefox : -moz-

2、Chrome & Safari : -webkit-

3、Opera : -o-

4、IE : -ms-

浏览器前缀加在哪？？？

1、如果浏览器不支持属性的话，则将前缀加在属性名称前

animation : C3的新属性

兼容：

-moz-animation:值;

-webkit-animation:值;

-o-animation:值;

-ms-animation:值;

animation:值;

2、如果浏览器不支持属性值的话，则将前缀添加在属性值前

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

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

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

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

background-image:linear-gradient();

### 2、文本格式化

1、字体属性

1、指定字体系列

属性：font-family

取值：字体1,字体2,字体3;

注意：

如果字体中包含 中文 或 空格的话，最好用 "" 或 '' 括起来

ex:

font-family:"微软雅黑";

font-family:"microsoft yahei",arial;

2、字体大小

属性：font-size

取值：px,pt,em,rem

3、字体加粗

属性：font-weight

取值：

1、normal

正常体，非加粗

2、bold

加粗

3、value

400 ：相当于 normal

900 ：相当于 bold

4、字体样式

属性：font-style

取值：

1、normal

正常，非斜体

2、italic

斜体

5、小型大写字母

作用：将所有的小写字符改成大小，但是大小和小写字符一样大

属性：font-variant

取值：

1、nomal

正常

2、small-caps

小型大写字符

6、简写模式

font:style variant weight size family;

注意：

使用简写属性时，必须设置 family的值，否则无效!!!

font:12px;//无效果

font:12px "微软雅黑";//OK

2、文本属性

1、文本颜色

属性：color

取值：合法颜色值

2、文本排列

属性：text-align

取值：

1、left

2、center

3、right

4、justify

两端对齐

3、文本修饰(线条)

属性：text-decoration

取值：

1、none

没有线条显示

2、underline

下划线

3、overline

上划线

4、line-through

删除线

4、行高

作用：一行文本所占高度,如果行高大于文字高度大小，那么文字将在行高范围内垂直居中

属性：line-height

取值：px为单位

ex:

line-height:24px;

line-height:1.5;

5、首行文本缩进

属性：text-indent

取值：以 px 为单位的数值

6、文本阴影

属性：text-shadow

取值：h-shadow v-shadow blur color;

### 3、表格属性

1、表格常用属性

1、边距属性

padding

2、尺寸属性

width height

3、文本格式化

font,text

4、背景属性

5、边框属性

border

6、垂直方向对齐

属性：vertical-align

取值：top / middle / bottom

2、表格特有属性

1、边框合并

属性：border-collapse

取值：

1、separate

默认值，即分离边框模式

2、collapse

边框合并模式

2、边框边距

属性：border-spacing

取值：

1、指定一个值

水平 和 垂直间距相等

2、指定两个值

第一个值：水平间距

第二个值：垂直间距

注意：保证 border-collapse 的值为 默认值(separate),否则该属性无效

3、标题位置

属性：caption-side

取值：

1、top

默认值，标题在表格之上

2、bottom

标题在表格之下

4、显示规则

指定表格布局的算法

属性：table-layout

取值：

1、auto

自动，默认值

列宽度由单元格内容来决定

2、fixed

固定

固定表格布局，列宽度由设定的值为准，不以数据来决定

自动表格布局 VS 固定表格布局

1、自动表格布局

1、单元格的大小会适应内容的大小

2、表格复杂时，加载速度慢(缺点)

3、灵活(优点)

4、不确定每列大小时，优先使用自动表格布局

2、固定表格布局

1、单元格大小取决于css中设定的值，与内容无关

2、加速显示表格(优点)

3、不够灵活(缺点)

4、已知每列大小时，可以使用固定表格布局

## Day04

### 1、定位 - 浮动

1、什么是定位

定位，就是改变元素在页面中的默认位置

2、定位的分类

1、普通流定位 - 页面默认定位方式

2、浮动定位 - 多个块级元素在一行内显示

3、相对定位

4、绝对定位

5、固定定位

3、定位-普通流定位

普通流定位，又称为 "文档流定位", 是页面中默认的定位方式

特点：

1、所有的元素都是从父元素的左上角开始出现

2、每个元素都有空间，默认不能重叠

3、块级元素 - 从上到下显示

4、行内 & 行内块 - 从左到右，显示不下换行

4、浮动定位（重点）

1、什么是浮动

如果将元素的定位方式设置为浮动定位，将具备以下特征：

1、浮动，会将元素排除在文档流之外(脱离文档流),不占据页面空间，其它元素要上前占位

2、浮动元素会停靠在父元素的左边或右边,或其它平级的已浮动元素的边缘上

3、浮动元素只能在当前行浮动

4、浮动元素依然在父元素中

5、浮动定位处理的问题-让多个块级元素在一行内显示

2、语法

属性：float

取值：

1、none

默认值，即元素无任何浮动效果

2、left

左浮动，让元素停靠在父元素的左边，或者挨着左侧已有浮动元素的右边上

3、right

右浮动，让元素停靠在父元素的右边，或者挨着右侧已有浮动元素的左边上

3、浮动引发的特殊效果

1、父元素中，如果显示不下所有的已浮动子元素，最后一个会换行，有可能会被卡住

2、元素一旦浮动起来之后，元素宽度将由内容来决定(非指定情况下)

3、元素一旦浮动起来之后，那么都将变成块级元素，尤其对行内元素影响较大

块级元素：允许修改尺寸

行内元素：不能修改尺寸

4、文本，行内元素，行内块元素 是采用环绕的方式来排列的，不会被浮动元素压在低下，而会巧妙的避开浮动元素

4、清除浮动所带来的影响

元素一旦浮动起来，会对后面的元素带来位置的影响，如果后面的元素不想被影响的话，可以通过 "清除" 的方式 来解除影响

属性：clear

取值：

1、none

默认值，即不做任何清除浮动

2、left

清除 前面 元素左浮动所带来的影响，即前面元素左浮动的话，当前元素不会上前占位

3、right

清除 前面 元素右浮动所带来的影响，即前面元素右浮动的话，当前元素不会上前占位

4、both

清除 前面 元素任何一种浮动所带来的影响

5、浮动元素对父元素的影响

由于浮动元素会脱离文档流，所以会导致元素不占据空间.如果一个元素中所有的子元素全部都是浮动的，那么该元素的高度就是 0 ，父元素的高度，是以未浮动元素的高度为准

解决方案：

1、直接设置父元素的高度

弊端：必须要知道父元素的高度

2、设置父元素也浮动

弊端：对后续元素时有位置影响的

3、为父元素设置 overflow 取值为 hidden 或 auto

弊端：如果有要溢出显示的内容，也一同被隐藏

4、在父元素的最后一个子元素位置处，增加 一个空块级元素，并且设置其 clear:both

5、... ..

### 2、显示

1、显示方式

作用：指定了元素在页面中默认的布局方式

属性：display

取值：

1、none

让生成的元素不显示并且脱离文档流

2、block

让元素变的与块级元素一样

3、inline

让元素变的与行内元素一样

4、inline-block

让元素变的与行内块元素一样

特点：

1、多个行内块以及行内元素能在一行内显示

2、行内块元素允许修改尺寸(除radio,checkbox)

5、table

让元素变的与 表格 一样

特点：

1、独占一行

2、宽度以内容为准

6、table-cell

让元素变的与 单元格 一样

2、显示效果

1、可见性

属性：visibility

取值：

1、visible

默认值，元素可见

2、hidden

隐藏，没脱离文档流

display:none 与 visibility:hidden的区别

display:none 会脱离文档流，所以不占空间

3、collapse

用在表格元素上，删除一行或一列时，表格整体布局不受影响

2、透明度

属性：opacity

取值：0.0(完全透明) ~ 1.0(完全不透明)

rgba(r,g,b,alpha);

div{

background-color:rgba(255,0,0,0.5);

opacity:0.5;

}

3、垂直对齐

属性：vertical-align

作用：

1、作用在表格(table,tr,td,th,...)元素上

指定表格中的内容的垂直对齐方式

取值：top / middle / bottom

2、作用在 图像 以及 行内块元素上时

指定 图像 或 行内块 元素 两端的文本的垂直对齐方式

取值：

1、top

2、middle

3、bottom

4、baseline

基线对齐

对于行内块元素：基线在最后一行文本的底部

对于图片元素：基线是在图片底部向下3px位置处(默认情况)

vertical-align:bottom;解决图像多占3px高的问题

3、光标

属性：cursor

取值：

1、default

默认值

2、pointer

小手

3、crosshair

+

4、text

I

5、wait

等待

6、help

帮助

### 3、列表

1、列表项标识

属性：list-style-type

取值：

1、none

... ...

2、列表项图像

属性：list-style-image

取值：url(图像的url)

3、列表项位置

属性：list-style-position

取值：

1、outside

默认值，标识位于列表项之外

2、inside

将标识位置改为列表项之内

4、列表属性

属性：list-style

取值：type url() position;

常用形式：list-style:none;

### 4、定位

1、定位属性

1、定位属性

属性：position

作用：指定元素的定位方式

取值：

1、static

默认值

2、relative

相对定位

3、absolute

绝对定位

4、fixed

固定定位

注意：如果元素设置为 相对定位/绝对定位/固定定位的话，则会被称之为 "已定位元素"

2、偏移属性

作用：对已定位属性实现位置的移动

属性：

top:以上边为基准边改变元素位置

right:以右边为基准边改变元素位置

bottom:以下边为基准边改变元素位置

left:以左边为基准边改变元素位置

3、堆叠顺序

属性：z-index

取值：无单位的数字

2、定位-相对定位

1、什么是相对定位

元素会相对于它原来的位置偏移某个距离

2、语法

属性：position

取值：relative

配合偏移属性 完成位置的移动

3、注意

相对定位元素 原来的位置 会被保留，不能被其它元素所占据（相对定位 与 margin的区别）

3、定位-绝对定位

1、什么是绝对定位

1、绝对定位元素会脱离文档流-不占据页面空间

2、绝对定位的元素会相对于 离他最近的 已定位的 祖先元素 去实现位置初始化

3、配合 偏移属性 实现位置的改变

4、如果 绝对定位元素 没有 已定位的祖先元素，那么它的位置就相对于最初的包含框(body 或 html)实现初始化

2、语法

属性：position

取值：absolute

配合 偏移属性 实现元素的移动

3、特点

1、会脱离文档流

2、绝对定位元素会变成块级元素

3、绝对定位元素在正常情况下，margin:auto会失效，其它正常

4、非正常情况，auto会生效

绝对定位元素的 top,right,bottom,left 同时为0时，auto会生效

4、堆叠顺序

1、什么是堆叠顺序

只能设置 已定位(relative/absolute/fixed) 元素的堆叠顺序

2、语法

属性：z-index

取值：无单位的数值

数值越大越靠前

数值越小越靠后

如果设置为负数的话，那么将在页面的所有内容之下

3、注意

1、默认堆叠顺序，平级元素，是后来者居上

2、对于父子级元素来说，永远都是子压在父上，是不能通过 z-index 来调整的

5、固定定位

1、语法

属性：position

取值：fixed

配合着 偏移属性 实现位置初始化

2、注意

1、固定定位元素会脱离文档流

## Day05

### 1、复杂选择器

1、兄弟选择器

兄弟选择器：

具备相同父元素的元素，称之为 兄弟元素

兄弟选择器，只能向后找，不能向前找

1、相邻兄弟选择器

相邻：紧紧挨着

语法：选择器1+选择器2

ex:

1、div+p

2、#container+.menu

2、通用兄弟选择器

通用：后面所有

语法：选择器1~选择器2

ex: div~p

2、属性选择器

允许通过元素所附带的属性及其值来匹配页面的元素

<input type="text" id="" class="" title="" reaonly>

语法：

基本语法：[attr]

attr:表示任意属性

1、[attr]

作用：匹配页面中附带 attr 属性的元素

ex:

[id]

2、elem[attr]

elem : element，表示页面任意元素

作用：匹配页面中附带 attr 属性的 elem 元素

ex:

div[id]

3、[attr1][attr2]

作用：匹配同时附带 attr1 属性 以及 attr2 属性的元素

ex:

[id][class]

4、[attr=value]

value : 某一具体属性值

ex:想获取页面中所有的文本框

input[type=text]

input[type="text"]

input[type='text']

5、[class~=value]

在多类选择器引用中使用

class属性值是一个由空格隔开的值列表，并且value是该值列表中的一个独立的选择器

[class="important"](有且只有) 和 [class~=important]（包含）

ex：

将页面中所有包含 important 类选择器的元素匹配出来

6、[attr ^= value]

^= : 以 ... 作为开始的

<div class="col-md-5"></div>

<div class="col-sm-3"></div>

... ...

匹配 页面中所有class以 col 作为开始的元素

[class^=col]

7、[attr \*= value]

\*= : 包含该 ...

<div class="col-md-1"></div>

<div class="col-md-3"></div>

<div class="col-md-5"></div>

匹配 页面中所有class值中包含md的元素

[class\*=md]

8、[attr $= value]

$= : 以 ... 作为结尾的

3、伪类选择器

1、目标伪类

作用：匹配 页面中 被激活的 html 锚点元素

语法： :target

2、结构伪类

1、:first-child

作用：匹配 属于其 父元素中的 首个子元素

<table id="tbl">

<tr>

<td></td>

<td></td>

</tr>

<tr>

<td></td>

<td></td>

</tr>

<table>

ex:

1、td:first-child

2、#tbl td:first-child

2、:last-child

作用：匹配 属于其 父元素中的 最后一个子元素

3、:nth-child(n)

作用：匹配 属于其 父元素中的 第n个子元素

4、:empty

作用:匹配没有子元素的元素(不能包含文本，空格，回车)

<div></div> --> div:empty{}

5、:only-child

作用：匹配属于其父元素中的唯一子元素

3、否定伪类

作用：将满足指定选择器的元素排除在外

语法：:not(selector)

div:not(".container") : 获取页面中所有 除class 为container 的 div元素

4、伪元素选择器

伪类：匹配到的是 "元素"

伪元素：匹配到的是元素中的某部分 "内容"

1、:first-letter 或 ::first-letter

作用：匹配 某元素中的首字符

2、:first-line 或 ::first-line

作用：匹配 某元素中的首行

3、::selection

作用：匹配被用户选取的部分

注意：只能改 文本颜色 和 背景颜色

: VS ::

1、在CSS2中，: 即表示伪类选择器，也表示伪元素选择器

2、在CSS3中，: 只代表伪类选择器，::只代表伪元素选择器

推荐使用 : 取代 ::

:first-child : 在 CSS3 以及 CSS2中都支持

::first-child: 只在CSS3中支持

### 2、内容生成

1、什么是内容生成

通过css，动态的向某元素区域内增加东西

2、伪元素选择器

1、:before 或 ::before

定位到某元素的内容区域之前

<div>(内容区域之前)这是一个div</div>

2、:after 或 ::after

定位到某元素的内容区域之后

<div>这是一个div(内容区域之后)</div>

3、属性 - content

配合着 :after 或 :before 一起使用

作用：生成内容

语法：

属性：content

取值：

1、url() : 生成图像

2、字符串 : 用 "" 或 ''引起来的纯文本

3、计算器

解决问题：

1、浮动元素父元素高度

<div>

<div style="float:left">111</div>

<div style="float:left">222</div>

<div style="clear:both;"></div>

<!--

1、最后一个子元素位置处

div:after

2、空子元素:生成空元素

content:"";

3、块级元素

display:block;

4、clear:both

clear:both;

-->

</div>

2、外边距溢出

<div>

<!--

1、第一个子元素 - :before

2、空子元素 - content:"";

3、table元素 - display:table

-->

<table></table>

<div style="margin-top:20px;"></div>

</div>

### 3、弹性布局

1、什么是弹性布局

弹性布局(Flexible box)，简称为 ：flex布局。指定某元素中的子元素的排列布局方式。

弹性布局容器：简称 "容器"，使用弹性布局属性的元素，其子元素能够按照弹性布局的方式进行排列

弹性布局项目：简称 "项目"，弹性布局容器中的子元素，允许按照弹性布局的方式进行排列

语法：

属性：display

取值：

1、flex

作用：将块级元素设置为弹性布局的容器

2、inline-flex

作用：将行内元素设置为弹性布局的容器

注意：

1、元素设置为flex布局后，子元素的 float,clear,vertical-align,text-align全部失效

2、容器中的项目，允许直接修改尺寸

3、容器中的项目，外边距合并问题被解决

2、容器属性

将属性设置给容器元素，从而实现所有子元素的排列布局方式

1、flex-direction

作用：决定主轴以及项目在主轴上的排列方向

取值：

1、row:主轴为水平方向的轴，起点在左端，默认值

2、row-reverse

主轴为水平方向的轴，起点在右端

3、column(重点)

主轴 更改为 交叉轴，起点在上端

4、column-reverse

主轴 更改为 交叉轴，起点在底端

2、flex-wrap

作用：默认情况下，项目们都在一条线上排列(轴线)，如果项目们在一条轴线上排不下时，如何换行?

取值：

1、nowrap

默认值，不换行,但是会缩放项目

2、wrap

换行

3、wrap-reverse

换行，第一行在下方

3、flex-flow

作用：flex-direction 和 flex-wrap 的缩写

取值：

row nowrap : 默认值

direction wrap;

4、justify-content(重点) 作用：指定项目在主轴上的对齐方式

注意：

如果主轴为横轴，则指定横向的排列对齐方式

如果主轴为交叉轴，则指定纵向排列对齐方式

取值：

1、flex-start

在轴的起点对齐

2、flex-end

在轴的终点对齐

3、center

居中对齐

4、space-between(终点)

两端对齐，项目之间的间隔都相等

5、space-around

每个项目两侧的间隔是相等的

5、align-items

作用：定义项目在交叉轴上如何对齐

取值：

1、flex-start

交叉轴的起点对齐

2、flex-end

交叉轴的终点对齐

3、center

交叉轴的中间对齐

4、baseline

基线对齐，以最大的文本第一行的基线为准

5、stretch

默认值，如果项目未设置高度或高度为auto的话， 将占满整个容器的高度

6、align-content

作用：该属性定义项目们有多跟轴线时的对齐方式(仅在项目自动换行时有效)

取值：

1、flex-start

在轴的起点对齐

2、flex-end

在轴的终点对齐

3、center

在轴的中间对齐

4、space-between

与轴的两端对齐，轴线之间的间隔相等

5、space-around

每个轴线两侧的间隔相等

6、stretch

默认值，占满轴线

3、项目属性

1、order

作用：排序,取值越小，排列越靠前。默认值为 0

取值：整数数字

2、flex-grow(重点)

作用：父元素有剩余空间的条件下,定义项目放大比例,默认为0,即便有剩余空间，也不放大

取值：整数数字

3、flex-shrink

作用：父元素剩余空间不足的情况下，定义了项目如何缩小即缩小比例，默认值为1

取值：整数数字

ex:

如果一个项目的 flex-shrink

设置为0，其它项目设置为1，如果空间不足时，如何排列？？

flex-shrink 为0的元素 宽度不变，为1的元素会缩小

4、flex-basis

作用：定义项目所占据容器空间的大小(主轴)

取值：

1、auto

项目本身大小

2、具体数值

5、flex

作用：flex-grow flex-shrink flex-basis 的简写方式

取值：

1、auto

等同于 1 1 auto;

2、none

等同于 0 0 auto;

3、grow[,shrink,basis]

6、align-self

作用：定义 自己 与其他项目不一样的在交叉轴的对齐方式,可以覆盖父元素的 align-items 属性

取值:

1、auto

默认值，继承父元素align-items的值,如果没有父元素的话，则等同于stretch

2、flex-start

3、flex-end

4、center

5、baseline

6、stretch

## Day06

### 1、CSS Hack

1、作用

做IE浏览器的兼容性

2、IE浏览器中的3种模式

1、混杂模式

没有标准的模式

2、准标准模式

即支持标准，同时又向前兼容非标准代码

3、标准模式

只支持W3C的标准规范，不支持非标准代码

不同的模式主要是对 CSS(框模型) 以 JS解析会有不同的解析效果

3、CSS Hack

根据不同的IE浏览器版本,去执行不同的CSS代码

原理：使用CSS属性的优先级来实现CSSHack

4、CSS Hack的实现方式

1、CSS 类内部Hack

通过 属性和值的前缀或后缀来实现浏览器版本的识别

2、选择器Hack

在 浏览器前加前缀来实现不同的浏览器版本的识别

\* : 识别IE6

\*+ 前缀： 识别IE7

ex:

\*div{

background-color:pink;

}

3、HTML头部引用Hack(掌握)

原理：使用 IE的条件注释来完成浏览器版本的判断

语法：

<!--[if 条件 IE 版本]>

满足浏览器版本时要执行的操作

<![endif]-->

版本：6-10，如果省略版本的号的话，则判断是否为IE

<!--[if IE 6]>

只在IE6中要执行的操作

<![endif]-->

条件：

1、gt

判断当前浏览器版本是否大于指定版本

2、gte

判断当前浏览器版本是否大于等于指定版本

3、lt

判断当前浏览器版本是否小于指定版本

4、lte

判断当前浏览器版本是否小于等于指定版本

5、!

非，取反，不是

判断当前浏览器版本是否不等于指定版本

6、不写

判断浏览器是否为IE或IE的指定版本

### 2、CSS3-转换

1、什么是转换

改变元素在页面中的位置，大小，角度或形状的一种方式

在CSS3中，允许对元素进行 2D转换 和 3D转换

2D转换：只允许在 x轴 和 y轴上进行转换操作

3D转换：在2D基础上，增加对 z轴 的转换效果

2、转换属性

1、转换

属性：transform

取值：

1、none

2、由 一组转换函数 组成

一个 或 多个转换函数,如果是多个转换函数，中间用 "空格" 分开

2、转换的原点

属性：transform-origin

默认情况：转换原点在元素正中间位置处

取值：

数值 ：以元素左上角为(0,0)点进行计算的

百分比：以元素左上角为(0%,0%)点进行计算

关键字：top/right/bottom/left/center

取值时，多个值之间，用 空格 隔开

取值数量：

2个值：表示原点在 x轴 和 y轴的位置

3个值：表示原点在x，y，z轴上的位置

### 3、转换-2D转换

1、位移

改变元素在页面中的位置

语法：

属性：transform

函数：

1、translate(x)

元素在x轴上的位移距离

x取值为正，向右移动

x取值为负，向左移动

2、translate(x,y)

元素在x轴和y轴上的位移距离

y取值为正，向下移动

y取值为负，向上移动

3、translateX(x)

4、translateY(y)

2、缩放

改变元素在页面上的大小

语法：

属性：transform

函数：

1、scale(value)

x轴和y轴将同时进行缩放

value取值：

1 ：原始大小

大于1 ：放大

0~1 ：缩小

2、scale(x,y)

3、scaleX(x)

4、scaleY(y)

3、旋转

改变元素在页面上的角度

语法：

属性：transform

函数：

rotate(ndeg)

n取值为正，顺时针旋转

n取值为负，逆时针旋转

注意：

1、元素旋转后，坐标轴也跟着旋转

2、转换原点会影响旋转的效果

4、倾斜

改变元素在页面中的形状

语法：

属性：transform

函数：

1、skewX(xdeg)

向着x轴(横向)的方向产生倾斜,其本质是改变y轴的倾斜角度

x 取值为正，y轴逆时针倾斜

x 取值为负，y轴顺时针倾斜

2、skewY(ydeg)

向着y轴(纵向)的方向产生倾斜,其本质是改变x轴的倾斜角度

y 取值为正，x轴顺时针倾斜

y 取值为负，x轴逆时针倾斜

3、skew(x)

4、skew(x,y)

4、转换-3D转换

1、什么是3D转换

允许元素在 x，y，z轴上进行转换

2、属性

透视距离

属性：perspective

作用：模拟人眼到3D转换物体之间的距离

兼容性：-webkit-perspective

注意：该属性要加在 3D转换元素的 父元素上。

3、3D旋转

允许让元素在 x，y，z轴上分别实现旋转效果

语法：

属性：transform

函数：

rotateX(xdeg)

以 x 轴为中心轴，旋转元素

rotateY(ydeg)

以 y 轴为中心轴，旋转元素

rotateZ(zdeg)

以 z 轴为中心轴，旋转元素

rotate3D(x,y,z,ndeg)

x,y,z 如果取值为1，则该轴参与旋转，取值为0则不参与旋转

rotate3D(1,0,0,45deg)

rotate3D(1,1,0,45deg)

rotate3D(1,1,1,45deg) 不等于 rotatex(45deg) rotatey(45deg) rotatez(45deg)

4、3D位移

语法：

属性：transform

函数：

translateZ(z)

属性：transform-style

作用：如何显示子元素的3D位置

取值：

1、flat

默认值，不保留其子元素的3D位置

(子元素不能以3D位置摆放)

2、preserve-3d

保留其子元素的3D位置

(允许让子元素以3D方式摆放位置)

### 3、CSS3-过渡

1、什么是过渡

使得CSS属性值，在一段时间内，平缓变化的一个效果

2、过渡的要素 & 属性

1、指定过渡属性(必须的)

作用：指定哪个 css属性值，在变化时要使用过渡的效果

语法：

属性：transition-property

取值：

1、none

默认值，不使用过渡

2、all

能使用过渡效果的属性全部都使用过渡

3、具体属性名称

ex:

transition-property:border-radius;

页面中允许设置过渡的属性：

1、颜色属性

2、取值为数值的属性

3、转换

4、渐变属性

5、visibility属性

6、阴影属性

2、指定过渡时长(必须的)

属性：transition-duration

取值：以 s 或 ms 为单位的数值

1s = 1000ms

注意：该属性值，默认为0，即不会有任何过渡效果

3、指定过渡速度时间曲线函数

指定速率

属性：transition-timing-function

取值：

1、ease

默认值,慢速开始，快速变快，慢速结束

2、linear

匀速

3、ease-in

慢速开始，快速结束...

4、ease-out

快速开始，减速结束...

5、ease-in-out

慢速开始和结束，先加速后减速

4、指定过渡延迟时间

延迟：用户激发操作后，等待多长时间再显示效果

属性：transition-delay

取值：以 s 或 ms 为单位的数值

5、过渡的简写方式

transition:property duration timing-function delay;

transition:prop1 dur1 timing1 delay1,prop2 dur2;

### 4.Css动画animation

1、问题

一个矩形元素

先变成 红色背景，再变成蓝色背景，再变成绿色背景，再变成黄色背景

2、什么是动画

使得元素从一种样式逐渐变化为另一种样式

(多个样式之间的平缓变化的过程)

通过 "关键帧" 来定义动画的每一个步骤

关键帧：动作(样式) 和 时间

3、动画的使用方式

1、声明动画

为动画指定 名称 以及 定义关键帧

在样式表中，通过 @keyframes 规则来声明动画

语法

@keyframes 动画名称{

/\*声明若干关键帧(样式 和 时间)\*/

0%{

/\*动画开始时的样式\*/

}

...

50%{

/\*动画执行到一半时的样式\*/

}

...

100%{

/\*动画结束时的样式\*/

}

}

@-webkit-keyframes 动画名称{}

@-moz-keyframes 动画名称{}

2、为元素调用动画

1、animation-name

作用：指定调用动画的名称

animation-name:changeBack;

2、animation-duration

作用：动画完成一个周期的执行时长

3、animation-timing-function

作用：指定执行速率

取值：ease,linear,ease-in,ease-out,ease-in-out

4、animation-delay

作用：指定动画延迟时长

5、animation-iteration-count

作用：指定动画的播放次数

取值：

1、具体数字

2、infinite : 无限次播放

6、animation-direction

作用：播放方向

取值：

1、normal

正常，从0% ~ 100%

2、reverse

逆向播放 ，从 100% ~ 0%

3、alternate

轮流播放，单数播放次数时，从0% ~ 100%,偶数播放次数时,从100%~0%

7、animation

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

8、animation-fill-mode

作用：指定动画在播放前或播放后的填充状态

取值：

1、none

2、forwards

当动画完成后，将保持在最后一个帧的状态上

3、backwards

在动画播放前(延迟时间内)，将元素保持在第一帧的状态上

4、both

forwards 和 backwards 的结合体

9、animation-play-state

作用：指定动画的播放状态

取值：

1、paused 动画暂停

2、running 动画播放

# Bootstrap

## Day01

Bootstrap:由HTML，CSS组成

响应式网页

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

能访问网页的设备：

1、PC - Personal Computer

2、移动电话 - iPhone,三星，小米，华为 ... ...

3、其它移动终端 ：Pad, ... ...

4、其它设备 ：电视... ...

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

1、什么是响应式网页

Responsive Web Page,响应式网页/自适应网页，即一个页面即可以下PC浏览器中浏览，同时也可以在各种移动终端中浏览。并且会根据不同的设备有不同的响应结果。

响应式网页的特点：

1、不同的终端布局不同

2、页面上的图片 和 文字都会随着屏幕的尺寸而发生改变

2、如何测试响应式网页

1、使用真实的物理设备

优势：测试结果真实可靠

不足：设备太多，成本太大，测试任务量繁重

2、使用第三方模拟设备

优势：无需使用更多设备

不足：效率低

3、使用浏览器自带的模拟设备(Emulator)

优势：轻量级，功能丰富

不足：非真实设备，与真实设备有一定的差异

3、视口 - Viewport(移动端重点)

注意：视口，与PC端无关

IOS 在 Safari 中提示的概念

视口：在移动设备中，浏览器中显示网页的一块区域(虚拟)

默认视口大小(宽度)：

Google，Safari : 980px

IE : 1024px

理想中的视口：

1、视口的宽度：要与设备宽度保持一致

2、缩放倍率：设置为1倍大小

3、视口的手动缩放：通常不允许手动缩放

在HTML中指定视口的信息：(响应式中必备)

<meta name="viewport" content="">

1、指定视口的宽度与设备保持一致

视口宽度：width

取值：

1、具体数值

2、device-width

ex:

<meta name="viewport" content="width=device-width">

2、指定视口的初始化缩放倍率

缩放倍率：initial-scale

取值：

1 : 原始大小

大于1 ：放大

小于1 ：缩小

ex:<meta name="viewport" content="width=device-width,initial-scale=1">

3、不允许手动缩放网页

手动缩放：user-scalable

取值：1/0/yes/no

1,yes : 表示允许缩放视口

0,no : 表示不允许缩放视口

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

记住：以上代码，在移动端是必备的!

4、如何编写响应式网页(重点)

1、必须声明视口(已解决)

2、文字尽量使用相对尺寸(em,rem)，尽量不用绝对尺寸(px,pt)

PC端：浏览器 文字默认大小是16px

移动端：浏览器 文字默认大小改为10px

em : 采用的是父元素的字体大小

rem : 采用的根元素(html)的字体大小

3、容器元素使用相对尺寸(%,auto),尽量不用绝对尺寸(px)

4、图片使用相对尺寸(%,auto),尽量不使用绝对尺寸(px)

推荐 使用 ：max-width 设置图像的宽度

img{

width:500px; /\*不推荐\*/

width:50%;/\*推荐使用，占父容器宽度的50%，但可以无限缩放\*/

max-width:50%;

/\*推荐使用，占父容器宽度的50%,但最大不能超过图片原始大小\*/

}

图片大小：100px 宽

父元素尺寸：1000px

为图片设置：

width:50%; /\*结果为：500px\*/

max-width:50%;/\*结果为：100px\*/

5、页面元素尽量使用"流式布局"

流式布局特点：

1、默认下，元素靠近容器的左上方(浮动根据方向决定)

2、默认横向排列，排列不下在换行

方法：

1、float ：浮动定位

2、display:inline-block

浮动，行内块，一定会配合着元素的尺寸一起做布局

6、响应式网页都要使用 CSS3 Media Query 技术 - 最重要的技术

响应式网页：不同的屏幕分辨率下，要有不同的显示效果

ex:

1、在宽度 小于 768px 屏幕中，背景设置为 红

2、在宽度 大于等于 768px 并且 小于等于 991px时，背景设置为 蓝色

3、在宽度 大于等于 992px 时，背景颜色设置为 黄色

5、CSS3 Media Query

媒体查询技术

作用：可以根据不同的媒体类型以及特性执行不同的CSS

语法：

1、样式表中声明 媒体查询

通过 @media 规则进行声明

@media MEDIA-TYPE and (MEDIA-FEATURE) and (MEDIA-FEATURE){

样式规则

}

MEDIA-TYPE:媒体类型

取值：

1、all,默认值，所有设备

2、screen:屏幕，电脑屏幕，手机屏幕，平板电脑屏幕

3、tv：电视设备

MEDIA-FEATURE:媒体特性

取值：

1、width ：指定浏览器的窗口宽度是否为指定的数值

@media screen and (width:768px) {

/\*必须是屏幕并且屏幕宽度必须是768时，才会执行以下内容\*/

...

}

2、min-width : 指定浏览器宽度的最小值

@media screen and (min-width:767px){

}

3、max-width : 指定浏览器宽度的最大值

问题：判断 屏幕的尺寸在767~ 991之间时，要执行某些操作

@media screen and (min-width:767px) and (max-width:991px){

/\*\*/

}

2、在 <link>中 做媒体查询判断

语法：

<link rel="" href="" media="MEDIA-TYPE and (MEDIA-FEATURE) and (MEDIA-FEATURE)">

不足：

即使不满足当前设备条件的CSS文件也会被请求,但不会生效

<link>

## Day02

1、响应式网页

1、视口 - Viewport

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

2、文字尺寸 - em,rem

3、容器尺寸 - %,auto

4、图片尺寸 - %,auto

width:50%; 无限放大

max-width:50%; 最大不能超过图像原始大小

5、页面元素使用流式布局

1、float

2、display:inline-block;

3、display:flex;

一定要配合宽度一起调整布局

6、媒体查询技术 - Media Query

语法：

@media MEDIA-TYPE and (MEDIA-FEATURE) and .{

选择器{

}

...

}

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

**Bootstrap 框架**

### 1、Twitter Bootstrap

官网：http://getbootstrap.com

中文：http://www.bootcss.com

文档：http://v3.bootcss.com

Bootstrap 分为5部分：

1、起步 - Startup

2、全局CSS - Global CSS

3、组件 - Component

4、JS插件 - JS Plugin

5、定制 - Customize

API : Application Programming Interface（应用程序编程接口）

### 2、Bootstrap - 起步

1、为 html 标记 指定 lang 属性

lang : language , 语言

指定当前文档的基础语言

作用：

1、为浏览器自动翻译功能指定翻译基础

2、为读评软件指定基础发音

<html lang="zh-cn">

</html>

2、<meta name="viewport"> 必须

3、指定 跨IE浏览器兼容性

<meta http-equiv="x-ua-compatible" content="IE=edge">

x-ua-compatible : Cross UserAgent Compatible

作用：指定使用 哪个IE的内核进行页面的渲染

IE=6 : 指定使用IE6内核渲染页面

IE=7 : 指定使用IE7/6内核渲染页面

IE=8 : 指定使用IE8/7/6内核渲染页面

...

IE=edge : 指定使用当前浏览器中支持的最高版本的内核渲染页面

4、引入 bootstrap.css

5、引入两个 兼容性的 JS 文件

1、html5shiv.min.js

第三方JS，自调函数，用于让老IE(IE8以及以下)支持新HTML5标记

2、respond.min.js

第三方JS，自调函数，用于让老IE(IE8以及以下)支持CSS3媒体查询技术

<!--[if lt IE 9]>

<script src="url/\*\*.js"></script>

<script src="url/\*\*.js"></script>

<![endif]-->

6、引入2个JS插件文件

作用：制定元素的行为(动作)

1、jQuery.1.11.3.js 引入到页面(先)

2、bootstrap.js 引入到页面(后)

建议：尽量将以上两个文件放在页面最底端</body>之上进行引入

7、引入 bootlint.js 文件

作用：用于检查页面结构是否符合 bootstrap 的规范

1、引入 bootlint.js

2、编写检查代码

<script>

bootlint.showLintReportForCurrentDocument([]);

</script>

### 3、Bootstrap 第二部分 - 全局CSS样式

Bootstrap.css 分为两部分

1、CSS Reset

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

html{font-size:10px;}

... ...

2、上千个 class

.btn .btn-default .btn-danger

### 4、Bootstrap 默认将屏幕分成四大类别

1、大型屏幕(lg) : w >= 1200px

2、中型屏幕(md) : 1199px => w >= 992px

3、小型屏幕(sm) : 991px >= w >= 768px

4、超小型屏幕(xs) : 767px >= w

### 5、Bootstrap 提供的两种容器

1、定宽容器

在不同大小的设备上提供不同的width固定值

类: .container

lg : width:1170px

md : width:970px

sm : width:750px;

xs : width:100%

2、变宽容器

在任何设备中，宽度都是 100%

类：.container-fluid

### 6、全局Class - 按钮

所有的按钮都依赖于 .btn

与 按钮相关的 类

.btn-default : 白底深字

带颜色按钮：

.btn-danger : 红色按钮

.btn-success : 绿色按钮

.btn-warning : 黄色按钮

.btn-info : 蓝色按钮(浅)

.btn-primary : 蓝色按钮（深）

danger/success/warning/info/primary : 表示颜色

指定按钮尺寸:

btn-lg : 大型按钮

btn-sm : 小型按钮

btn-xs : 超小型按钮

块级按钮：

btn-block

页面中允许做按钮的元素：

<a class="btn btn-default"></a>

<button type="button" class="btn btn-info"></button>

<input type="submit" class="btn btn-default">

### 7、全局Class - 列表

.list-unstyled 不带标识的列表(ol,ul)

.list-inline 行内列表(ol,ul)

.dl-horizontal 定义列表水平排列(dl)

### 8、全局Class - 图片

.img-rounded ?

.img-circle ?

.img-thumbnail ?

.img-responsive ?

### 9、全局Class - 表格 table

基本表格：.table

带边框：.table-bordered

隔行表色（条纹）：.table-striped

鼠标悬停：.table-hover

响应式表格: .table-responsive

### 10、全局Class - 文本

1、文本颜色相关

.text-danger/success/warning/info/primary

.bg-danger/success/warning/info/primary

2、文本大小写

.text-uppercase : 全大写

.text-lowercase : 全小写

.text-capitalize : 每个单词首字符大写

3、文本对齐方式

.text-left : 文本左对齐

.text-right: 文本右对齐

.text-center:文本居中对齐

.text-justify:文本两端对齐

### 11、全局Class - 栅格布局系统(重中之重)

1、网页中实现布局的技术(标记)

1、div + css(float)

好处：效率高

不足：灵活，不易控制

2、table 布局

好处：简单，容易控制

不足：效率低

3、栅格布局系统

好处：效率高，容易控制，实现响应式

不足：none;

本质就是由div组成的table样式的响应式结构

2、使用方法

1、栅格布局系统的最外层，必须是bootstrap提供的两个容器类之一

.container 或 .container-fluid

2、允许在容器的任意位置放置若干"行"

class ：.row

ex:

<div class="container">

<div class="row"></div>

<h3></h3>

<div class="row"></div>

</div>

每行中，最多会被等分为12列

12列：每列宽 ：8.33%

3、行(div.row)中只能放置列,但需要指定每列所占的宽度(即当前列要占据几列的宽1,2,3,...,12)

列 ：

占一列的宽：8.33%

占两列的宽：16.66%

4、列 根据适用的屏幕分成4种类型

类：.col-\*-\*

.col-xs-\*:在超小屏幕下，所占的列宽数

.col-xs-1:超小屏幕下，占一列宽(8.33%)

.col-xs-2:超小屏幕下，占两列宽(16.66%)

.col-xs-12:超小屏幕下，占12列宽(100%)

.col-sm-\*:在小屏幕下，所占的列宽数

.col-sm-1:

.col-sm-6:

.col-md-\*:在中等屏幕下，所占的列宽数

.col-lg-\*:在大屏幕下，所占的列宽数

5、指定列偏移数量

每个列都可以指定 向右 偏移几列的位置

.col-xs-offset-n:在xs屏幕下，当前列向右偏移 n 列的位置

.col-sm-offset-n:在sm屏幕下，当前列向右偏移 n 列的位置

.col-md-offset-n:

.col-lg-offset-n:

## Day03

### 1、全局Class - 栅格布局系统(下)

1、栅格布局系统的嵌套

.container>.row>.col-\*-\*

在栅格布局系统中，.col-\*-\* 允许当做容器(.container/.container-fluid)去看待

.container>.row>.col-\*-\*>.row>.col-\*-\* ...

2、适用于不同屏幕的列的class(xs/sm/md/lg)，可以兼容更大的屏幕

col-xs-8 : 在 超小屏幕中，占8列

.col-xs-\* : 适应于 xs/sm/md/lg

.col-sm-\* : 适应于 sm/md/lg

.col-md-\* : 适用于 md/lg

.col-lg-\* : 是英语 lg

大屏幕的列的class不能兼容小屏幕的，所以大屏幕的内容放在小屏幕中 占100%的宽

3、可以在一个div中，指定在不同屏幕下的宽度占比

即在一个列中指定不同屏幕下占的列宽

<div class="col-md-3 col-sm-6 col-xs-12">

在 md 中，占3列的宽(一行显示4列)

在 sm 中，占6列的宽(一行显示2列)

在 xs 中，占12列的宽(一行显示1列)

4、可以指定在特定屏幕下不显示某个列

.hidden-lg : 在 lg 屏幕下 隐藏

.hidden-md : 在 md 屏幕下 隐藏

.hidden-sm : 在 sm 屏幕下 隐藏

.hidden-xs : 在 xs 屏幕下 隐藏

### 2、全局Class - 表单

1、表单分类

1、默认(基本)表单

纵向排列

1、表单组

.form-group

2、标签

<label>

3、表单控件

<input class="form-control">

定义表单控件的基本样式(radio,checkbox除外)

4、提示文本(可选)

.help-block : 定义提示文本的样式

2、行内(内联)表单

横向排列

1、<form class="form-line">

2、<label class="control-label">

3、水平排列的表单(难点)

组与组之间，纵向排列

组内的内容，横向排列

水平表单 = 栅格布局系统 + 表单

栅格布局系统：

最外层：.container / .container-fluid

行：div.row

列：div.col-\*-\*

水平表单：

最外层：

.container / .container-fluid

或

<form class="form-horizontal">

行：

div.row

或

div.form-group

列：

div.col-\*-\*

或

<label class="col-\*-\*">

或

<span class="help-block col-\*-\*">

注意：表单控件，只能放在 <div class="col=\*-\*">中

.sr-only : Screen Reader Only

仅针对屏幕读取器有效

属性：aria-label="xxxx"

<label for="" aria-label="用户名:">

作用：当label对应的控件获取焦点时，屏幕阅读器会读出响应的 aira-label的值

属性：aria-labelledby

<label for="" aria-labbeldby="某元素ID">

作用：当label对应的控件获取焦点时，屏幕阅读器会读出 某元素对应的文本

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Bootstrap 组件**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 1、什么是组件

由多个元素 以及 class 组成的复杂的结构

### 2、组件

1、下拉菜单

1、外层元素

必须是 .dropdown / .dropup / position:relative

2、内层（子元素）

1、触发器元素

由 button 或 a 来组成

1、class="dropdown-toggle" : 点击触发器元素后，触发器元素的效果是什么样

2、增加 data-toggle 属性 ：切换菜单的显示和隐藏(动作)

data-toggle="dropdown"

ex：<a href="#" class="btn btn-default dropdown-toggle" data-toggle="dropdown">显示/隐藏 菜单</a>

2、菜单

由 <ul> 或 <div> 组成

class="dropdown-menu"

li.divider : 分割线

li.disabled : 禁用菜单项

li.dropdown-header : 菜单项标题

2、导航

1、所有的导航都依赖于 .nav

2、导航分类

1、标签页式导航

<ul class="nav nav-tabs">

<li class="active">

<a href="#">北京市</a>

</li>

<li>

<a href="#">海淀区 </a>

</li>

</ul>

2、胶囊式导航

<ul class="nav nav-pills">

<li class="active">

<a href="#">北京市</a>

</li>

<li>

<a href="#">海淀区 </a>

</li>

</ul>

3、实现点击切换效果

<li>

<a href="#" data-toggle="tab"></a>

</li>

3、图标字体

在页面中，显示为图标，但本质是文字，允许设置颜色，大小，粗体，斜体 ... ...

Web程序中常用的图标字体：

1、Glyphicons 收费

2、FontAwesome 675个 免费

由于客户端不具备 bootstrap 中提供的图标字体，所以在使用时，必须先声明图标字体(声明字体的名称，以及提供字体文件让用户去下载)，然后再使用

再使用自定义图标字体时，需要进行以下设置：

1、在 \*\*\*.css中 ，声明图标字体

/\*声明字体\*/

@font-face{

font-family:"自定义字体名称";

src:url('字体文件路径');

}

2、在元素中

选择器{

font-family:"自定义字体名称";

}

注意：以上两个步骤仅在 使用自定义 图标字体时使用，在bootstrap 中，以上两个步骤已经被实现。

Bootstrap中的图标字体使用方式：

1、创建一个元素，必须保证为空

<span></span>

2、class="glyphicon glyphicon-\*"

练习：

在页面中创建多个按钮，在每个按钮上显示一个图标

首页，配置，刷新，定位，购物车，放大镜，发邮件，照相机，播放相关(暂停，播放，上一曲，下一曲，快进，快退)

4、按钮组

1、基本按钮组

<div class="btn-group">

<button>

<button>

</div>

2、允许将多个按钮组(btn-group) 合并进一个 .btn-toolbar(按钮工具栏)中

## Day04

1、栅格布局系统

1、嵌套

.container>.row>.col-\*-\*>.row

2、小屏幕class可以兼容大屏幕

col-lg-5 : 在 xs 中，占100%

3、在一个div中指定在不同屏幕下占比(重要)

<div class="col-lg-3 col-md-6 col-xs-12">

4、在某种屏幕下，不显示

.hidden-lg

.hidden-md

.hidden-sm

.hidden-xs

2、表单

1、默认表单

<form>

<div class="form-group">

<label>xxx</label>

<input class="form-control">

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

</div>

</form>

2、行内(内联)表单

<form class="form-inline">

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

<input class="form-control">

</form>

3、水平表单

水平表单：表单 + 栅格布局系统

容器：

div.container / container-fluid

form.form-horizontal

行：

div.row

div.form-group

列：

label.col-\*-\*

<div class="col-\*-\*">

<input>

</div>

span.col-\*-\*

3、组件-下拉菜单

<div class="dropdown / dropup">

1、触发器元素

<a class="dropdown-toggle" data-toggle="dropdown"></a>

2、菜单

<ul class="dropdown-menu">

<li class="divider"></li>

<li class="disabled"></li>

<li class="dropdown-header"></li>

</ul>

</div>

4、组件 - 导航

1、所有的导航都依赖于 .nav

2、导航分类

1、标签页式导航

<ul class="nav nav-tabs">

<li class="active">

<a data-toggle="tab"></a>

</li>

</ul>

2、胶囊式导航

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

5、组件 - 图标字体

1、自定义图标字体

xx.css

@font-face{

font-family:"";

src:url();

}

选择器{

font-family:"";

}

2、Bootstrap中的图标字体使用

1、创建一个纯空元素

2、class="glyphicon glyphicon-\*"

6、组件 - 按钮组

<div class="btn-group">

<button>

<button>

<button>

</div>

<div class="btn-toolbar">

<div class="btn-group">

<button>

<button>

<button>

</div>

</div>

7、组件 - 大杂烩

1、面包屑导航 - 路径导航

<ul class="breadcrumb">

<li>

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

</li>

<li class="active">

<a></a>

</li>

</ul>

2、分页

.pageination : 分页

.active : 被激活的

3、翻页

.pager

4、标签

所有的标签都依赖于 .label

.label-default

.label-success

...

.label-primary

5、徽章

.badge

6、巨幕

.jumbotron

7、页头

.page-header

8、水井- Well

.well

.well-lg/sm/xs

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

### 1、组件-进度条

外层：.progress

内层：

.progress-bar

.progress-bar-success

...

.progress-bar-primary

### 2、组件 - 警告框(alert)

所有的警告框都依赖于 .alert

基础类：.alert

可选类：

.alert-success

.alert-danger

.alert-info

.alert-warining

1 、带超链接的警告框

<a class="alert-link"></a>

2、允许关闭的警告框

1、警告框元素 必须依赖于 .alert-dismissible

2、激发关闭的元素 - 触发器元素

data-dismiss="alert"

### 3、组件 - 缩略图

1、基本缩略图

<a class="thumbnail">

<img>

</a>

2、复杂的缩略图

<div class="thumbnail">

<img>

<div class="caption">

<h3>Heading</h3>

<p>Some text ...</p>

<p>Some text ...</p>

<p>

<button>

<button>

</p>

</div>

</div>

### 组件 - 媒体对象

### 5、组件 - 列表组

ul : .list-group

li : .list-group-item

### 6、组件 - 面板

允许呈现出 头部，主体，脚注的组件结构

所有的面板都依赖于 .panel

可选 .panel-default/success/info/... ...

<div class="panel panel-default">

<!-- 面板头部(可选) -->

<div class="panel-heading">

<h3 class="panel-title">标题</h3>

</div>

<!-- 面板主体(必须的) -->

<div class="panel-body"></div>

<!-- 面板脚注(可选) -->

<div class="panel-footer"></div>

</div>

### 7、组件 - 导航条(navbar)

1、基本导航条

1、由nav来组成导航条

1、向nav元素增加 .navbar 以及 .navbar-default

2、nav 中 必须包含容器 (.container / .container-fluid)

3、容器中 允许添加 div.navbar-header,允许添加 a.navbar-brand 元素

2、导航条内 增加 导航

增加 .nav 以及 .navbar-nav

3、导航条中的表单

导航条中的表单不适用与bootstrtap表单默认类

需要为表单增加一个类 .navbar-form

可以通过 navbar-left / navbar-right 来实现浮动

4、导航条中的按钮

独立的按钮，脱离 <form>

通过 .navbar-btn 为不在表单中的按钮增加样式

5、导航条中的文本

通过 .navbar-text 为普通文本增加样式

6、导航条中内容的浮动

.navbar-left : 实现左浮动

.navbar-right : 实现右浮动

7、实现导航条的固定

固定在页面的顶端：.navbar-fixed-top

固定在页面的底端：.navbar-fixed-bottom

注意：最好为body设置 至少50px的内边距

### 8、JS插件 - Plugins

Bootstrap提供了一组行为如：下拉菜单

Bootstrap提供的行为 全部都是 基于 jQuery(JS库),Bootstrap是在 jQuery的基础上 提供了十几个插件函数,每个插件函数是一个独立的JS文件,可以一次性全部引入JS插件函数 - bootstrap.js

想使用 JS 插件，则必须引入两个 js 文件

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

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

每个插件函数都提供了2种调用方式：

1、data-\*

<a data-toggle="dropdown">

<button data-dismiss="alert">

2、JS编程方式

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

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

<script>

//通过 选择器 获取页面元素

$("选择器");在JS中获取指定选择器所对应的元素

$("#top") : 在JS中获取 页面上 ID为top的元素

$(".container") : 在JS中获取 页面上 class为 container 的元素

$("#top>div") : ......

</script>

### 9、JS插件 - 工具提示

为元素添加

data-toggle="tooltip"/\*鼠标移入行为\*/

title="提示的文本"

data-placement="top/right/bottom/left"

必须配合 JS 代码 一起使用

$("选择器").tooltip();

### 10、JS插件 - 弹出框

为元素添加

data-toggle="popover";

data-placement="top/right/bottom/left"

title="弹出框的标题"

data-content="弹出框内容"

必须配合 JS 代码

$("选择器").popover();

### 11、JS插件 - 标签页

标签页由两部分组成：

1、标签页式导航(nav nav-tabs)

<ul class="nav nav-tabs">

li 中的 a 标记：

1、data-toggle="tab"

2、href="#对应显示内容的模块ID"

2、内容部分(tab-content)

<div class="tab-content">

<div id="">内容模块1</div>

<div id="">内容模块2</div>

</div>

内容模块：

1、class = "tab-pane"

2、必须设置id，提供给对应的标签导航的a元素

### 12、JS插件 - 模态框(modal)

模态框：一旦被打开后，必须要处理掉之后才可以操作页面的其它元素

模态框由两部分组成：

1、触发器元素

通常由 a / button 充当 触发器元素

<a data-toggle="modal" href="#ID"></a>

<button

data-toggle="modal"

data-target="#ID"

></button>

2、模态框元素(必须提供 ID)

由四部分(层级)组成

//class="modal" : 提供半透明遮罩层

//data-backdrop="static" 保证模态框不能通过其他方式关闭

<div id="" class="modal" data-backdrop="static">

//modal-dialog:提供内容区域宽，高，定位

<div class="modal-dialog">

//modal-content:提供，背景，边框，倒角

<div class="modal-content">

<div class="modal-header"></div>

<div class="modal-body"></div>

<div class="modal-footer"></div>

</div>

</div>

</div>

关闭模态框：

<a data-dismiss="modal"></a>

## Day05

1、导航条

<nav class="navbar navbar-default navbar-fixed-top">

<div class="navbar-header">

<a class="navbar-brand"></a>

</div>

<ul class="nav navbar-nav"></ul>

<form class="navbar-form"></form>

<button class="navbar-btn navbar-left"></button>

<span class="navbar-text navbar-right"></span>

</nav>

2、工具提示

data-toggle="tooltip"

data-placement="top/right/bottom/left"

title="提示文本"

配合

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

3、弹出框

data-toggle="popover"

data-placement="top/right/bottom/left"

title="标题"

data-content="内容"

配合

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

4、标签页

<ul class="nav nav-tabs">

<li>

<a href="#关联的内容模块ID"></a>

</li>

</ul>

<div class="tab-content">

<div class="tab-pane"></div>

</div>

5、模态框

1、触发器

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

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

2、模态框元素

<div class="modal" id="" data-backdrop="static">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button class="close" data-dismiss="modal">&times;</button>

</div>

<div class="modal-body"></div>

<div class="modal-footer"></div>

</div>

</div>

</div>

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

### 1、插件-折叠效果(Collapse)

1、基本的折叠效果

1、触发器

<a data-toggle="collapse" href="#ID"></a>

或

<button data-toggle="collapse" data-target="#ID"></button>

2、折叠元素

<div class="collapse"></div>

2、手风琴-Accordion

3、响应式导航条-Responsive navbar

当屏幕尺寸>=768时，可以正常显示所有的内容，当屏幕尺寸<=767px时，一部分内容被隐藏，通过折叠点击的方式进行显示

响应式导航条由两部分组成

1、.navbar-header

1、.navbar-brand

任何情况下，都会显示

2、折叠按钮

.navbar-toggle

屏幕尺寸>=768px,不显示

屏幕尺寸<=767px,显示

2、折叠内容(受响应式控制)

.navbar-collapse .collapse

屏幕尺寸 >=768px,全部正常显示

屏幕尺寸 <=767px,则隐藏，需要通过 .navbar-toggle的点击 来完成显示 和 隐藏

### 2、插件-图片轮播(carousel)

1、基本结构

<div class="carousel slide" data-ride="carousel">

<div class="carousel-inner">

<div class="item active">

<img src="">

</div>

</div>

</div>

2、指定轮转的时间

增加属性：

<div class="carousel" data-interval="2000">

3、带文本的轮播图片

<div class="item">

<img src="">

<div class="carousel-caption">

<h3>标题</h3>

<p>文本</p>

</div>

</div>

4、带反向按钮的轮播

<div id="myCar" class="carousel">

<div class="carousel-inner"></div>

<!-- 左右方向按钮 -->

<a href="#myCar" class="carousel-control left/right" data-slide="prev/next">

<span class="glyphicon glyphicon-chevron-left"></span>

</a>

</div>

5、带导航圆点的轮播

<div class="carousel">

<ul class="carousel-indicators">

<li

data-target="#Carousel的ID"

data-slide-to="0"

class="active"

></li>

</ul>

</div>

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

### Bootstrap 定制 & Less

1、样式语言的分类

1、静态样式语言 : CSS

可以直接被浏览器解析处理；

但CSS并不是合格的 "语言"，它缺少编程语言的最基本要素 ：变量，运算符，函数，....

静态语言的缺点：可维护性差

2、动态样式语言 ：

如：Less , Stylus , SCSS ...

不能被浏览器直接编译处理；必须经过编译(Compile)得到一个CSS文件，再交给浏览器去使用

2、Less语言

官网：http://www.lesscss.org

中文：http://less.bootcss.com

Less是一个CSS预处理语言，它扩充了CSS，在静态的CSS基础上增加了动态的内容,Less中完全支持和兼容CSS

在 Web 中使用Less的两种方式

1、客户端浏览器中编译Less - 不推荐

1、编写 xxx.less

2、编写 xxx.html

在网页中，可以引入 xxx.less,同时还需要引入编译文件 less.js

3、客户端访问 xxx.html 会自动下载 xxx.less 和 less.js ，然后进行编译，解析 ...

2、在服务器端编译Less - 推荐

1、编写 xxx.less

2、在服务器端 搭建 Less编译器 ，将 xxx.less 编译成 xxx.css

3、网页中，正常引入 css 文件即可

3、搭建Less编译器(乱)

Less编译器是由JS进行编写的...

1、安装独立的JS解释器 - NodeJS

查看Node版本：在控制台中 输入 ：

node -v

2、安装Less编译器程序

确保 lessc.cmd 文件存在即可

3、将 Less 转换为 Css(重点)

1、在 控制台中 找到 lessc.cmd 并打开

2、编译 less文件

lessc.cmd E:\1.less > E:\1.css

4、在 WS 中，配置 FileWatchers(文件监视器)

通过 FileWatchers 自动检测Less文件的编写与更改，自动进行编译再得到CSS

WS --> File-->Settings-->Tools-->FileWatchers --> 添加 选择 Less --> 指定 lessc.cmd 文件地址

4、Less语法

1、Less 完全支持 CSS

2、Less支持单行注释 和 多行注释，但是，只有多行注释能够编译到CSS中

多行：/\* \*/

单行：//

3、Less 支持 变量(Variable)

变量：在 Less 中，变量是一个可以变化的数据

语法：

@变量名:值;

变量名：英文，数字(不能开头)，\_，-

ex:

@jd-color:#E4393C;

@border:1px solid #000;

@base-font:"microsoft yahei";

4、Less 中允许使用 运算符

+,-,\*,/

@width:5px + 1;

5、在 Less 中，支持在一组样式中 混入(Mixin) 另一种样式

选择器1{ ... }

选择器2{ ... ; 选择器1 ;}

选择器2中 会包含 选择器中 定义好的样式

带参数的混合:

声明选择器的时候，允许使用"参数"来表示暂时不确定 或 可以灵活变化 的数据 ，在最终 "混合"的时候，将确定的数据传递进来

选择器1(@参数名1,@参数名2){

background-color:@参数名1;

border:@参数名2 solid #005aa0;

}

带参数混合的调用

选择器2{

width:200px;

...

选择器1(值1,值2);

}

## Day06

### 1、插件-折叠(Collapse)

1、触发器元素

<a data-toggle="collapse" href="#ID"></a>

<button data-toggle="collapse" data-target="#">

2、被折叠元素

<div class="collapse" id=""></div>

3、其它效果

1、面板的折叠

<div class="panel panel-default">

<div class="panel-heading">

<a data-toggle="collapse" href="#pnl"></a>

</div>

<div id="pnl" class="panel-collapse collapse">

<div class="panel-body"></div>

<div class="panel-footer"></div>

</div>

</div>

2、手风琴效果

<div class="panel-group" id="pg">

<div class="panel panel-default">

<div class="panel-heading">

<a data-parent="#pg" data-toggle="collapse" href="#pnl"></a>

</div>

<div id="pnl" class="panel-collapse collapse">

<div class="panel-body"></div>

<div class="panel-footer"></div>

</div>

</div>

...

</div>

3、响应式导航条

<nav class="navbar navbar-default">

<div class="navbar-header">

<a class="navbar-brand"></a>

<button data-target="#d1" data-toggle="collapse" class="navbar-toggle">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

</div>

<div id="d1" class="navbar-collapse collapse">

... ...

</div>

</nav>

### 2、附加导航 & 滚动监听

<body data-spy="scroll" data-target="#nav">

... ...

<div id="nav" data-spy="affix" data-offset-top="300">

<ul class="nav nav-pills">

<li></li>

<li></li>

<li></li>

</ul>

</div>

</body>

### 3、插件-轮播(Carousel)

<div id="cc" data-interval="2000" class="carousel slide" data-ride="carousel">

<!-- 轮播广告内容 -->

<div class="carousel-inner">

<div class="item active">

<img>

<div class="carousel-caption">

<h4></h4>

<p></p>

</div>

</div>

</div>

<!-- 左右方向按钮 -->

<a href="#cc" class="carousel-control left" data-slide="prev"></a>

<a href="#cc" class="carousel-control right" data-slide="next"></a>

<!-- 导航小圆点 -->

<ul class="carousel-indicators">

<li class="active" data-slide-to="0" data-target="#cc"></li>

<li data-slide-to="0" data-target="#cc"></li>

</ul>

</div>

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

**Less:**

1、语法

1、Less 完全支持 CSS

2、Less中的注释

单行：// 不会被编译到CSS中

多行：/\* 正常会被编译到CSS中 \*/

3、变量

@变量名:值;

使用变量：

background:@变量名;

4、运算符

+，-，\*，/，... ...

5、混合(Mixin)

在一组样式中 混入 另一种样式

选择器1{}

选择器2{

... ...

选择器1;

}

带参数的混合:

选择器1(@参数1,@参数2){

width:@参数1;

background-color:@参数2;

}

选择器2{

... ...

选择器1(50px,#e4393c);

}

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

### 1、Less

1、语法

1、嵌套

在less中，允许嵌套定义 选择器,以便方便的完成父子结构 或 后代结构

语法：

选择器1{

... ...

选择器2{

... ...

选择器3{

...

}

}

>选择器4{

... ...

}

}

编译完成后：

选择器1{ 只包含选择器1的内容 }

选择器1 选择器2{ 只包含选择器2的内容 }

选择器1 选择器2 选择器3{ 选择器3 }

选择器1>选择器4{ 选择器4的内容 }

2、函数

是Less中提供的一组预定义的功能

1、lighten()

颜色减淡函数

ex:

background:lighten(red,20%);

意思:将red颜色，减淡20%，作为背景色

2、darken()

颜色加深函数

ex:

background:darken(red,20%);

3、img-width('URL')

获取指定图片的原始宽度

4、img-height('URL')

获取指定图片的原始高度

5、ceil(number)

对 number 进行向上取整

ex:

ceil(35.1) : 36

6、floor(numer)

对 number 进行向下取整

ex:

floor(38.6) : 38

3、文件导入

在一个less文件中，可以引入另外一个less文件

1、在CSS中

在 all.css 文件中

@import "top.css";

2、在Less中

在 all.less 文件中

@import "new.less";

### 2、Bootstrap 定制

1、Bootstrap Less 的结构

1、基础样式(保留)

variables.less ：用于声明变量

mixins.less ：用于混合

normalize.less ：实现最基本的样式重写

print.less ：设置打印的效果

glyphicons.less ：图标字体

scaffolding.less ：设置基础的全局样式

2、全局CSS样式文件-Core CSS (建议保留)

type.less ：全局CSS的基础样式

code.less ：与代码相关的样式

grid.less ：栅格布局系统样式

tables.less ：表格样式

forms.less ：表单相关样式

buttons.less ：按钮相关样式

3、组件-Components(按照需要保留或删除)

# Javascript

## Day01

张东

zhangdong@tedu.cn

一本书: 犀牛书

微信公众号: 前端大全

上届的笔记: 打印

正课：

1. 什么是JavaScript

2. 变量/常量

3. 数据类型

### 一、什么是JavaScript

1.前端三大语言:

1.HTML 专门编写网页内容的语言

2.CSS 专门编写网页样式的语言

3.JavaScript 专门添加网页交互行为的语言

2.什么是交互: 3步:

1.用户输入数据或执行操作

2.程序接受并处理数据，响应用户的操作

3.返回处理结果

今后，所有静态页面必需添加交互行为后，才能让用户使用

3.使用JavaScript:

1.背景：HTML和CSS都是静态语言，无法动态执行计算和判断

2.何时使用：所有网页必须使用js添加动态交互行为后，才能给用户使用典型用途:

1.客户端数据计算

2.客户端表单验证

3.动画效果

4.JavaScript发展史:

笔试题:ECMAScript JavaScript JScript

1.ECMAScript: 由ECMA组织制定的JavaScript语言国际标准

2.JavaScript: NetScape按照ECMAScript标准，实现的JavaScript语言版本

3.JScript: 微软按照ECMAScript标准，实现的JavaScript语言版本

5.JavaScript包含三大部分:（原生JS: 不需要下载任何第三方文件就可直接使用的js）

1.核心语法:ECMAScript:

ES 版本:3.1 + 5 + 6/2015

2.DOM标准: 规定了专门操作网页内容的程序

3.BOM: 专门操作浏览器窗口的程序

6.JS语言四大特点:

1.运行在脚本解释引擎中:

脚本解释引擎:专门解析js语言，并执行js程序的小软件。

如何:2种:

1.浏览器自带脚本解释引擎:

浏览器包括了两个小软件:

1.内容排版引擎:专门解析HTML和CSS，并显示网页内容和样式的小软件。

2.脚本解释引擎:专门解释并执行js程序

笔试题:做动画选择CSS vs JS

如果不需要交互行为的动画，首选CSS，效率高

如果必须交互行为的动画，只能用js

2.独立安装的脚本解释引擎:Node.js

从chrome浏览器中剥离的独立的脚本解释引擎

以前:前端js 后端 PHP/JAVA 数据库 SQL

现在:前端js 后端 nodeJS 数据库 MongoDB

2.解释执行:

边解释边执行,后解释的相同内容会覆盖先解释的

3.弱类型:

声明变量时，不必提前规定变量的数据类型一个变量，先后可保存不同类型的数据

4.基于对象:CORE

7.如何使用JavaScript:（写在哪儿: 2处）

1.页面中的script标签里

一旦进入script标签内，就要使用js的语法

问题:不符合内容与行为分离的原则-->不便于维护和共用

2. 独立的js文件中:

1.引入:<script src="js文件的url"></script>

2.强调:script标签不能同时引入外部js和包含内部js。

-->结果: 内部js自动失效

8.基本语法:

1. 区分大小写

2. 每条语句必须用分号(";")结尾

3. 单双引号都行

4. 字符串可用"或'包裹都行

5. 注释:

//单行注释

/\*多行注释\*/

9.输出:3个地方:

1.向页面中添加内容:

document.write("html代码段");

1.问题:向页面中随意输出内容会破坏页面的布局

2.解决:调试程序时，不轻易使用document.write

2.弹出对话框

alert("提示消息")

1.问题:浏览器预定好的样式，不可修改

2.解决:对话框都用div+css模拟

3.向控制台中输出内容:

1.控制台:console，专门用于调试程序，临时显示调试结果的小窗口

2.何时:只要调试程序，都要在控制台进行

3.如何:F12-->console

//在控制台输出一行记录

ex：console.log("Hello console!")

4.好处:调试的内容不影响页面布局

5.问题:编辑极其不方便

6.解决:在WS（webstorm）中编写独立的js文件，然后，选择在node中运行

7.强调:每log一次，都自动换行

8.更多控制台操作:

1.编写多行代码: Shift+Enter仅换行暂不执行

2.清空控制台: 左上角圆形叉号

3.调出刚写过的代码段:上下键切换

10.输入:输入框: 专门请用户输入一个数据的对话框

1.如何:prompt("输入提示")

2.对话框缺点:

1.样式不可修改

2.会阻断程序继续执行以及用户操作

### 二、变量、常量

1. 变量:

什么是变量: 内存中存储一个数据的存储空间，再起一个名字

1.何时: 只要反复使用的数据，都要先保存在变量中，再反复使用变量。

2.如何:

1.声明: 在内存中创建\*一个\*存储空间，再起一个名字

1.何时: 只要使用变量前，必须先声明变量，才能使用

2.如何: var 变量名;

1.强调:一切变量都用var声明，不必提前规定数据类型

2.变量的默认值: undefined

undefined: 专门表示一个从未使用过的变量

3.简写: 可用一个var同时声明多个变量, 用逗号分隔:

var 变量1,变量2,...;

4.变量名:

1.使用字母，数字，下划线组成，不能以数字开头

2.不允许使用js语言的保留字/关键词

（即：不能使用js语言已经提前占用的有特殊意义的名字）

3.见名知义

4.驼峰命名:

首字母小写，之后单词首字母大写

比如: backgroundColor

listStyleType

2.赋值: 将等号右边的值保存到等号左边的变量中

1.如何: 变量名=值;

2.简写:

1.声明同时，初始化变量的值:

var 变量名=值;

2.同时声明并初始化多个变量:

var 变量1=值1, 变量2=值2,....;

3.强调：

1.只有等号才能改变变量中的值

2.=（等号）会覆盖变量中的原值

3.变量赋值前，必须先声明

4.特殊:

1.普通模式: 强行给未声明变量赋值, 不会报错！自动创建新变量并保存值

2.ES5严格模式: 比普通js运行模式要求更严格的模式

1.规定：禁止给未声明的变量赋值

2.如何启用:在代码开始位置:"use strict";

3.取值:任何情况下，使用变量名，等效于直接使用变量中的值

3.强调: 严格模式下：强行从未声明的变量中取值, 报错！

显示--> ReferenceError: 要用的东西，没找到

2.常量: 一旦创建后，值不允许改变的量

1.何时: 只要一个数据一旦定义，就不允许改变

2.如何:

1.声明: const 常量名=值;

强调: 声明常量时，必须初始化

习惯上，常量名要全大写！

2.赋值: 常量不允许赋值

特殊: 强行修改常量的值：

1.普通模式: 不报错，但也不能修改——静默失败

2.严格模式（ES5:声明："use strict"）: 报错！

将所有静默失败都升级为错误！

3.取值：常量的用法和普通变量完全一样

### 三、数据类型:

1.什么是: 数据在内存中的存储结构

2.为什么: 不同的数据类型，擅长执行的操作不一样，就需要配套合适的存储结构，给操作提供便利

3.包括: 2大类:

1.原始类型（的值）: 值 直接保存在变量本地的数据类型

[5种: number、string、boolean、null、undefined]

1.number: 专门用于存储一切数字

1.何时: 只要经常用作算数计算或比较的数值

2.如何: 程序中只要不加引号的数值，默认就是number类型

3.存储空间: 8字节二进制(整数4字节)

1字节(Byte)=8位二进制数(Bit)

1KB=1024字节B

1MB=1024KB

将十进制数转二进制: n.toString(2);

4.强调: number即包含整数，又包含浮点数(小数)

2.string: 保存一串字符组成的字符串

1.何时: 用于显示或记录一段文字

凡是用"或'包裹的一串文字，自动就是字符串类型

2.存储空间:

1.每个字符，数字，或英文标点占1字节，每个汉字占2字节

2.js中采用unicode编码存储每个字符:

3.unicode是对全球主要语言的每个文字编一个号

4.为什么: 计算机只认数字，不认字符

5.查看一个字的unicode号: str.charCodeAt();

6. 数字0~9: 48~57

大写字母A~Z: 65~90

小写字母a~z: 97~122

汉字: \u4e00~\u9fa5

3.转义字符: \

何时: 2种:

1. 当字符串中包含和js语言标点符号冲突的内容，用\将冲突的字符转为原文，不再按程序解析

2. 要表示特殊意义:

\n 换行 \t 制表符(tab键)

\u unicode

3.特殊情况: 网址 或 路径中的\

2种:

1. 前再加\ -->不建议，只有windows才认\

2. 将\换成/ -->强烈建议，任何操作系统都认

3.boolean: 布尔类型: 只有两个值的类型: true/false

何时: 专门用于表示一个判断的结论

4.null: 空 让程序员手动清空一个变量

5.undefined: 空 由程序自动使用，为变量赋初值

2.引用类型（的值）: 值 无法保存在变量本地的复杂数据类型

1.如果一个复杂的数据结构，同时保存多个数据，就不能直接保存在变量中

2.存储在window之外的一块独立存储空间

3.独立存储空间中可同时存储多个数据，拥有唯一的地址

4.变量名仅保存存储空间的地址-----例：数组

## Day02

正课:

1. 数据类型转换:

2. 运算符和表达式:

### 一、数据类型转换:

1.什么是:将不需要的数据类型，转化为需要的数据类型

2.js四大特点之一:弱类型: 3大特点:

1.声明变量时，不需要提前指定变量的数据类型

2.同一个变量先后可保存不同类型的数据

3.不同数据类型间可相互转化（程序会根据自身的需要自动改变数据的类型）

3.何时:只要给定的数据类型不是想要的，都要转换

4.如何:2种:

1.隐式转换:程序根据自身的需要，不需要程序员干预，由js自动完成的数据类型转换

1.何时:只要给定的数据类型和js程序需要的类型不相符，js就会根据自己的需要，自动转化数据类型

2.比如:算数计算中

3.规则:

1.默认:一切数据都转为number类型，再做计算（原因：因为number类型最适合计算）

bool: true -> 1

false-> 0

2.特殊:+运算中，如果碰到一个字符串，则不再转数字，而是一切都转字符串，+运算变为字符串拼接

2.强制转换:程序员主动调用函数功能，完成的数据类型转换

1.何时:只要给定的数据类型不是想要的，且自动转换的结果也不是想要的，就要强制转换。

2.如何（3种）:

1.[转数字]其他类型转number:（2种）

1.[将纯数字或bool转为number]

var n=Number(x); //将x转为number类型

1.强调:不直接修改原变量中的值，而是返回新值。

2.何时:都是隐式转换，其实相当于自动调用Number(x)，很少主动使用。

3.问题:只能转换纯数字组成的字符串和bool类型

4.如果转不了:就返回NaN

1.NaN:不是一个数字的任何值--> Not a Number

2.何时:专门用于判断类型转换或算术计算错误。

3.特点:NaN参与任何计算，结果只能是NaN

2.[将任意字符串转number]

var n=parseFloat/parseInt(str);

//将str转为number类型，自动去除末尾非数字字符

1.parseFloat可保留小数部分

2.parseInt 去掉小数部分

3.何时:只要将字符串转number，首选parseFloat

如果确实需要去掉小数，才选parseInt

如果转不了，也返回NaN

4.强调:参数应该是string类型

1.可去掉结尾的非数字字符

但无法去掉开头的非数字字符

2.如果给定的值不是string类型，则先执行隐式转换，转为string类型，再转number

比如:parseFloat(true) --> parseFloat("true") --> NaN

Number(true) --> 1

3.Number/parseFloat/parseInt都能去掉字符串中的空格后，再转数字。

5.特殊：

Number("") --> 0

Number(null) --> 0

Number(undefined) --> NaN

2.[转字符串]其他类型转string:（2种）

1.x.toString()将x转为字符串类型

x不能是null或undefined --> 不是万能

2.String(x)将x转为字符串类型 --> 万能（首选）

其实，隐式转为string时，都是由程序在隐式转为字符串时自动调用String

3.[转布尔]其他类型转bool类型:1种

Boolean(x)

1.何时:在隐式转为bool时，自动调用Boolean(x)

2.规则:

1.只有5个值会被转为false：[这5个值均表示无意义]

0、""、NaN、null、undefined

2.其余都转为true

5.练习:

1.凡是从页面上获得的一切都是字符串类型

2.typeof 变量，可输出变量中值的数据类型名

输出结果：ex：string number

### 二、运算符和表达式:(重点)

1.程序:人的想法在计算机中的执行

2.运算符:程序中模拟人的想法的特殊符号

3.表达式:由数据、变量和运算符组成的完成一项单一任务的语句。

4.算数运算:+、-、\*、/、%

%:模运算/取余数:被除数/除数，不要商，要除不尽的余数部分

1.何时:

1.取余数

2.判断能否整除

判断偶数:能被2整除 n%2==0

判断奇数:不能被2整除 n%2!=0

3.拓展：判断闰年

2.隐式转换:

1.默认:都转number

2.特殊:+运算中，碰到字符串，都转字符串，+运算变为字符串，再拼接。

3.舍入误差:计算机中，也有计算不尽的数值

4.解决:

1.按指定位数四舍五入:n.toFixed(2)

今后几乎所有显示钱数的地方都要toFixed(2)

2.存储:保存很长位数的小数:0.39999999

5.关系运算:用两个值做比较，判断结果-->true/false

1.包括:> < >= <= == !=

2.返回值:bool类型的true/false

3.隐式转换:

1.默认:都转number

2.特殊:

1.两个值都是字符串，则不再转数字，而是依次比较每个字符的unicode号（Ascall码）

2.判断NaN:（NaN:不大于 不等于 不小于 一切，甚至自己）

1.NaN和任何值做 > 、< 、>=、<=、== 5种比较时，永远false

2.NaN和任何值做!=比较时，永远是true

3.问题:NaN==NaN =>false 无法用普通关系运算==或!=来判断是不是NaN

1.解决:isNaN(num)专门代替==，用来判断num是不是NaN

ex：var input=prompt("请输入一个数字:");

console.log(

"是数字吗？"+(isNaN(input)!=true)

);

2.何时:只要判断是不是NaN，都用isNaN(num)

3.固定套路:判断一个值是不是数字:

!isNaN(num) 成立 说明num是数字

3.区分null和undefined

1.问题：用普通的==无法区分null和undefined（原因：==比较时，会自动将undefined隐式转为null）

null==undefined --> null=null

2.解决:===全等:要求类型必须先相同，值再相等

其实===就是不带隐式转换的==

3.建议:今后尽量用===代替==

4.总结:= == ===

=赋值:只有=可改变变量中的值

==和===都是做比较的

==自带隐式转换

===不带隐式转换

4.两个对象做比较，不进行任何转换，而是直接比较两个对象的地址值。

6.逻辑运算:将多个关系运算，综合起来得到最终结论

返回值:也是bool的true/false

1.何时:只要根据多个条件， 综合得出最终结论时

2.包括:&& 而且 ||或(要么) !不(没有) --> [且 或 非]

1.条件1 && 条件2...:

1.要求所有条件都为true，结果为true

2.只要一个条件为false，结果为false

2.条件1 || 条件2...:

1.只要一个条件为true，结果为true

2.除非所有条件都为false，结果才为false

3.!条件:颠倒条件的判断结果

4.隐式转换:默认将每个条件都转为bool类型

5.短路逻辑:如果前一个条件已经可以得出最终结论，则后续条件不再执行。

&&:只有前一个条件为false时，后续条件才不执行

||:只要前一个条件为true时，后续条件不再执行

利用短路:

1.利用&&短路:简单分支结构:

1个条件，1件事，只有满足才做，不满足就不执行

1.如何:条件&&(操作)

2.练习中:运算符优先级:优先级高的运算，先计算

改变优先级:用()

2.利用||短路:实现默认值/备用值:

如果给定的值有效，则使用给定的值，否则，使用默认值代替

1.如何:给定值||默认值

无效:只要给定值为 "" 0 null undefined NaN，就启用默认值

2.如果值1可以被转为true，就选择给定值使用

3.如果值1不能转为true，就选择默认值使用

4.返回值:如果逻辑运算最后执行的是一个关系运算，则返回bool的true/false

如果逻辑运算最后执行的是一个值，则返回值，不再返回bool的true/false

7.位运算:了解

1.左移/右移:

m<<n m左移n位: 相当于m\*2的n次方

1<<3=8 1左移3位 相当于1\*2的3次方

2<<3=2\*2的3次方=16

8>>3=1 8右移3位 相当于8/(2的3次方)

2.下取整:省略小数部分

相当于：parseInt(n)

固定套路:

n^0 n|0 n>>>0

3.不声明第三个变量，而交换两变量的值:

var a=3,b=5;

方法一: var t=b; b=a; a=t; [用了第三变量t，不推荐！！！]

方法二: a+=b; b=-a-b; a-=b;

方法三: a^=b; b^=a; a^=b; [二进制，按位异或-阶]

8.赋值运算:

1.扩展赋值运算:对特殊赋值运算的简写:

a=a+b 将a的值+b的值，再存回a中

-->将b的值累加到a上，可简写为a+=b；

a=a-b -> a-=b;

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

a=a/b -> a/=b;

a=a%b -> a%=b;

2.更简化:递增递减运算:如果每次累加1/累减1

a+=1 -> a++

a-=1 -> a--

3.前++和后++:

ex：m++和++m:

1.如果m++单独使用，则前++和后++效果完全一样

2.如果m++嵌入其它表达式中:

1.相同:m中的值一定会被+1

2.不同:返回值:m++，返回+1前的\*旧\*值

++m，返回+1后的\*新\*值

4.笔试:

var n=3;

console.log(n++ + ++n + n++);

// 3 + 5 + 5 =13

//console.log(++n + n++ + ++n);

// 4 + 4 + 6=14

console.log(n);//6

提示:所有表达式都是从左向右依次执行

如果前一个表达式修改了变量的值，则会影响后续表达式的执行

## Day03

正课:

1. \*\*\*函数

2. 全局函数

3. 分支结构

### 一、\*\*\*函数:（重点、难点）

1.\*\*\*函数:

1.什么是:程序中封装一项任务步骤清单的代码段，再起一个名字。

2.为什么:代码重用

3.何时:只要一段代码，可能被反复使用，都要先封装在函数中，再反复调用函数。

4.如何:2步:

1.声明:

function 函数名(参数列表){

函数体;

return 返回值;

}

1.参数:函数运行时，接收传入函数的数据的变量

参数用法和普通变量完全一样，只不过不用var声明/创建

2.参数列表:多个参数间用逗号分隔

1.何时:当一项任务必须有某些数据才能正常执行时，就必须定义参数

2.作用:

1. 可用于提醒调用者，如何正确使用函数

2. 让函数变得更灵活！

3.如何使用:

function 函数名(参数变量1,参数变量2,...)

3.返回值:函数执行的结果

1.何时:如果函数的调用者需要获得函数执行结果时，就用return返回结果

2.return的2个作用:return 返回值;

1. 退出函数执行——可单独使用

2. 退出函数时，顺便返回结果

如果没有返回值的函数，其实也返回东西:默认返回undefined（即：如果函数没有返回任何结果，就返回undefined）

2.调用:让引擎按照函数的步骤清单，执行一项任务

1.何时：函数不调用不执行，只有调用才执行

2.如何：var 返回值=函数名(参数值列表)

3.参数值列表:传入函数的执行时必须的数据列表

ex：function 函数名(参数变量1,参数变量2,...)

4.只要函数定义时规定了参数变量，调用时都要传入参数值。且顺序和个数保持一致

5.每个参数值之间用逗号分隔

2.\*\*\*作用域scope:一个变量的可用范围

1.什么是: 变量的可用范围

2.为什么: 避免不同范围的变量之间互相污染

3.包括:2种:

1.全局作用域:window

全局变量:直接定义在全局中，不属于任何函数的变量

-->程序中的任何位置都可以访问的范围

1.特点:可反复使用，随处可用

2.何时:如果一个变量需要反复使用，或跨函数随处可用时可用（即：只要希望随处可用的公共变量便可用）

3.如何：只要不属于任何函数的变量，自动都是全局的

2.函数作用域:仅函数内可用的范围

-->保存着局部变量：保存在函数作用域中，仅函数内可用的变量

1.局部变量:2种:

1.函数内var声明的变量

2.函数的参数变量也是局部变量

2.特点:仅在函数内可用，出了函数不能使用

不可重用

3.何时:如果一个变量，仅希望在函数内有效时

4.总结:

优先定义并使用局部变量

尽量少使用全局变量-->避免被污染

5.变量的使用顺序:

优先使用函数内的局部变量

只要局部声明了，就不用全局的

除非局部没有声明，才去全局找

3.\*\*\*声明提前(hoist):【很重要，笔试面试经常会考！！】

1.什么是:在开始执行程序前，引擎会将var声明的变量和function声明的函数，提前到\*当前作用域\*顶部集中优先创建，赋值留在原地，再开始执行程序

2.笔试:如果先使用，后声明，一定在考声明提前就要将所有var和function提前到当前作用域的顶部，先创建，再判断程序的输出

3.问题：破坏了程序默认的执行顺序

4.解决:

1.传统:建议将所有的变量和函数声明都集中在当前作用域顶部

2.ES6:let代替var声明变量

1.let要求:在let之前，不允许提前使用该变量

2.强制将声明集中在当前作用域顶部创建

3.限制:不能用let反复创建相同名称的变量（let只能用一次）

5.函数:第2种创建函数的方式

var 函数名=function(参数变量列表){...}

1.本质：

1.函数名其实仅是一个普通的变量

2.函数其实是保存代码段的引用类型的对象

3.函数名通过函数对象的地址引用着函数对象

2.★优势：用以上方式创建的函数不会被声明提前

4.\*\*\*按值传递:

1.什么是:两变量间赋值，或向函数中传入参数时，其实只是将原变量中的值复制一个副本给对方。

2.结果:原始类型的值: 修改新变量，不影响原变量的值

### 二、全局函数:了解

1.什么是:ES标准中规定的，不需要任何前缀"."就可直接调用的函数。

1.比如:

String()/Number()/Boolean()

parseInt/parseFloat()

isNaN()

2.反例:alert()/prompt()...

2.编码解码:特指对地址栏中的url内容进行编码和解码

1.什么是:将url中多字节字符或保留字符编码为单字节

2.为什么:url中不允许出现多字节字符以及和保留字符冲突的字符，如果包含，会乱码

3.何时:只要url中包含多字节字符以及和保留字冲突的字符时，都要先编码为单字节。

4.如何:

1.编码:var code=encodeURI(url)

2.解码:var text=decodeURI(url)

1.解码:将编码后的单字节字符串解码回多字节原文

2.何时:只要收到的是编码后的字符串，都要先解码再使用

5.问题:url中不但不允许多字节字符，而且还不允许出现保留了字符:比如：“:” “/” 但是encodeURI无法对保留字编码

解决:

1.编码:var code=encodeURIComponent(url);

2.解码:var text=decodeURIComponent(url);

3.eval函数:执行一段字符串格式的js表达式

★程序三大结构:顺序，分支和循环

### 三、分支结构:

1.什么是:让程序根据不同的条件执行不同的操作

2.何时:只要让程序根据不同的条件执行不同的操作

3.如何:

1.一个条件，一件事，只有满足才做，不满足就不做

1.如果操作简单:短路:条件&&(操作)

2.如果操作复杂:if(条件){

操作

}

2.一个条件，二件事，二选一执行:

1.如果仅两个值二选一:

(条件?值1:值2)

2.如果操作简单:三目/三元/条件

条件?操作1:操作2

3.如果操作复杂:

if(条件){

操作1;

}

else

{

操作2;

}

3.多个条件，多件事，多选一执行：

1.如果多个值，多选一:三目

1.条件1?值1:

2.条件2?值2:

...?...:

默认值——强调:默认值不能省略！

2.如果操作简单:三目

1.条件1?操作1:

2.条件2?操作2:

...?...:

默认操作——强调:默认操作不能省略！

3.如果操作复杂:if...else if结构

1.if(条件1){

操作1 //如果当前条件满足，则不再向后执行

}else if(条件2){

//如果进入条件2，暗示条件1不满足

操作2

}else if(...){

...

}else{ —— 如果没有默认操作，则else省略

默认操作

}

2.简写:如果if/else if/else下只有一句话，可省略{} ——强烈不建议

3.特殊:如果所有条件都是等于比较，可简写为

switch(表达式){ //先计算表达式的值

//用表达式的值和每个case后的值做\*全等\*比较

case 值1: //如果表达式的值\*全等\*于case后的值1

操作1; //就进入case下执行操作

break;

case 值2: //否则：如果表达式的值全等于case后的值2

操作2;

break;

...

default: //如果所有case的值都不等于表达式的值

默认操作; //则执行默认操作（最后一个default可省略，但不推荐）

}

4.问题:只要前一个case满足条件，则后续所有case都被执行。

5.原因:switch是入口

6.解决:break;负责终止并退出当前结构继续向后执行

7.如何:每个case之间，都要加break

作业：100题:2、4、 5

## Day04

正课:

1. \*\*\*循环

2. \*\*\*数组

### 1.\*\*\*循环:

1.什么是:让程序反复执行同一段代码

2.何时:只要让程序反复执行同一段代码

3.如何:三要素:

1.循环条件:让循环可以继续执行的条件

2.循环变量:在循环条件中用作比较和判断的变量

考虑：

1.从几开始

2.每次递增/递减几

3.到几结束

3.循环体:循环反复执行的代码段

几乎都要修改循环变量，向着不满足循环条件的趋势。

如果循环条件始终为true，则循环永远无法退出——死循环

4.包括:3种:

1.while循环:先判断循环条件，再决定是否执行循环

var 循环变量=初始值; //声明并初始化循环变量

while(循环条件){ //当满足(循环条件)时

循环体;

修改循环变量的值；

}

循环不满足的结果；

何时使用:必须先验证循环条件，才能执行循环体时

2.do while循环:先至少执行一次循环，再判断循环条件，决定是否反复执行

var 循环变量=初始值; //声明并初始化循环变量

do{

循环体;

修改循环变量的值；

}while(循环条件); //当满足(循环条件)时

循环不满足的结果；

何时使用: 希望即使不满足，也至少可以执行一次时

辨析：while vs do while

while: 先验证后执行

do while: 先执行一次，再验证

1.如果第一次循环都满足，两者的输出是完全一样的

2.如果第一次条件不满足，while是一次都不执行，

do while至少执行一次。

3.for:其实就是while循环的简化

何时使用：只要循环变量的变化有规律

for(var 循环变量=初始值;循环条件;修改循环变量){

循环体;

}

注：for循环和while循环的原理是完全一样的。

辨析：for vs while: 如果循环变量的变化有规律: 首先for

如果循环变量的变化没有规律: 首先while

5.简写:

1.第一部分:可同时声明并初始化多个变量，多个变量间用逗号分隔。

2.循环体:(分支/循环结构中)

1.如果条件之下只有一条语句，则可省略{}，但强烈不建议这样做-->容易出歧义

2.如果for循环的第三部分可先后执行多个短小的操作，每个操作之间用逗号分隔。-->但是，不能改变原来的执行顺序

6.强调:js中没有块级作用域:

1.放在if/else if/else/for/while（即：分支结构/循环结构）中声明的变量，出了结构，依然可用

2. vs java所有{}都认为是块级作用域，块中的变量出了块，不能使用。

3.解决: 让循环中的变量不要泄露到循环外，用let代替var: let声明的变量仅能在{}内使用

7.break vs continue:

1.break:中断并退出当前循环

何时:当判断条件非常复杂时，就可用死循环+break的方式灵活控制循环退出。——降低循环编写的难度

2.continue:跳过本轮循环，继续执行下一轮

只要颠倒判断条件，就可避免使用continue

3.强调:break不能参与到任何简写中，必须独立一句。

100题: 1,2,4,9,11,12,14,17

8.嵌套循环:一个循环内，又执行了另一个内层循环

如何:1.截取片段找规律

2.用外层循环反复调用规律的公式

9.作业:打印反三角和等腰三角，九九乘法表

### 2.\*\*\*数组:

1.什么是:内存中连续存储多个数据的一块存储空间，再起一个统一的名字。

2.为什么:便于维护和查找数据

1.程序=数据结构+算法

1.数据结构:数据在内存中的存储结构

2.算法:解决问题的步骤

2.规律：好的数据结构可极大提高程序的执行效率

3.何时:今后只要保存多个同一类型的数据时，都必须用数组

4.如何:3件事:创建、赋值、取值

1.创建:3种:

1.创建空数组:2种:

var arr=[];

var new Array();

何时:创建数组时，暂时不知道数组内容

2.创建数组并初始化数组内容:

var arr=[值1, 值2, ...];

var new Array(值1,值2,...)

何时:创建数组时已知数组的内容

3.创建n个空元素的数组:

var arr=new Array(n);

何时:创建数组时仅知道数组元素个数，暂时不知道内容。

4.访问:概念:

1.元素:数组中每个数据，就称为元素

2.下标index:数组中唯一标识一个元素存储位置的序号

3.默认都是数字从0开始，连续不重复

2.取值:数组名[下标]

1.数组中每个元素的用法和普通的变量完全一样

2.数组也称为一组变量的集合，起一个统一的变量名

3.赋值:将数据保存到数组的某个元素中

数组名[下标]=新值;

arr[i]=新值;

4.注：js的数组三个不限制:

1.不限制元素的数据类型

2.不限制下标越界:不会报错！

1.取值时越界: 仅返回undefined

2.赋值时越界: 在指定的新位置自动创建新元素保存新值

3.如果添加后，数组的元素下标不连续——稀疏数组

3.不限制元素个数:

在任何时候，都可在任何位置添加新元素

5. .length属性:记录理论上数组中的元素个数

.length属性永远等于最后一个数字下标+1

.length属性永远指向最后一个元素的下一个新位置

.length可修改数组大小——缩容

固定套路:1.获取数组最后一个元素:

arr[arr.length-1]

2.获得倒数第n个元素

arr[arr.length-n]

3.末尾追加一个新元素:

arr[arr.length]=新元素

4.删除数组最后一个元素

arr.length-=1;

5.删除数组末尾倒数n个元素

arr.length-=n;

6.\*\*\*本质：数组是引用类型的对象:

按值传递:

1.原始类型的值:修改新变量，不影响原变量

原因:复制的是值本身

2.引用类型的对象:通过新变量修改对象，等效于直接修改原对象。新旧变量都受影响。

原因:

1.仅复制的是地址值，原对象始终只有一个

2.新旧变量使用相同的地址值，引用同一个对象

3.任何一方修改对象，另一方都同时受影响。

7.遍历:

1.什么是:对数组中每个元素执行相同的操作

2.何时:只要对数组中每个元素执行相同的操作

3.如何:

1.索引数组：【下节讲】

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

arr[i] //当前元素

}

2.关联数组：【下节讲】

8.dir问题:不会立刻输出结果

1.只有手动点击三角时，才动态查找内存中对象的最终结构。会打乱程序的输出结果顺序。

2.解决:如果向查看数组中间状态的内容:

console.log(String(arr)); //对数组拍照

3.总结:如果查看中间状态:console.log(String(arr));

如果查看结构:console.dir(arr);

4.ex:

console.log(String(arr));

console.log(arr);

console.dir(arr);

console.log(arr);

先输出dir的结构，刷新后变为String(arr)

仅输出内容，不关心结构: console.log(String(arr))

查看数组存储结构：console.dir(arr)

## Day05

正课:

补:\*\*\*关联数组

1.数组API:

2.栈和队列:

3.二维数组:

### 补：遍历

1.什么是索引数组:下标都为数字的数组

2.\*\*\*关联数组:

1.什么是:可自定义下标名称的数组

2.为什么:为了让每个元素都有一个专门的名称:

好处:1.便于维护

2.便于查找

3.何时:

1.希望每个元素都有专门的意义时

2.希望快速查找元素时

4.如何:2件事：

1.创建:2步:

1.先创建空数组：var ym=[];

2.向空数组中添加新元素，要使用自定义的下标名称

ym["name"]="杨幂";

ym["math"]=81;

ym["chs"]=59;

ym["eng"]=89;

2.访问:访问关联数组和访问索引数组中的元素，同变量的用法完全一样

只不过要使用自定义的下标名称

5.关联数组中length属性失效，永远等于0

因为关联数组中没有数字下标,length无法+1

6.遍历关联数组:

for(var key in ym){//in依次取出ym中每个可以

key //当前房间号

ym[key] //当前元素值

强调: key不能加引号, 因为每次循环都在变化

}

7.关联数组原理:

1.问题:索引数组，无法提前预知元素的具体位置

只能遍历查找: 受元素个数和元素存储位置影响

2.解决:hash算法:

1.根据一个字符串，计算出尽量不重复的一个序号。相同的字符串计算出的序号一定相同

向关联数组中保存元素；

2.将自定义下标名交给hash算法，计算一个散列的位置序号。将元素保存到序号指定位置

从关联数组中取值时；

3.将自定义下标名交给hash算法，计算出和存入时完全相同的序号，引擎直接去序号位置获得元素。

8.优点: 查找极快，

1.因为:

1.直接通过下标定位元素位置

2.查找速度和元素个数以及元素存储位置无关

2. vs 索引数组: 只能靠遍历查找

缺点: 受元素个数和元素存储位置的影响极大

### 1.数组API:

1.什么是数组:连续保存一组数据，并提供了操作数据的简化API

API:别人已经实现的，咱们用现成的程序

2.包括:

1.数组转字符串:2种:

1.String(arr):将arr中每个元素都转为字符串，用逗号连接

何时:给数组拍照，用于比较操作前后数组是否发生了变化。

2.arr.join("连接符"):将arr中每个元素都转为字符串，用自定义的连接符连接元素

1.何时:只要希望使用逗号以外的其它连接符时

2.固定套路:

1.把单词拼接为句子:arr.join(" ")

2.无缝拼接:（把字母拼成单词）

错误：arr.join() 等效于--> String(arr) 表现出-->逗号连接

正确：arr.join("")

重要用途:判断数组是否为空！

3.动态生成页面元素2步

1. var html="<ANY>"+arr.join("</ANY><ANY>")+"</ANY>";

2. elem.innerHTML=html;

2.拼接和选取:都无权修改原数组，只能返回新数组对象

1.拼接:将其他数据或另一个数组，和当前数组拼接为新数组。

1.如何:var newArr=arr1.concat(值1,值2,arr2,...);[注：英文：concat:粘]

2.强调:

1.不修改原数组，只返回新数组

2.可打散数组类型参数为单个数据，再拼接--> concat独有！

2.选取:复制原数组中指定开始位置到结束位置之间的元素，组成新数组-->原数组保持不变

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

2.强调：

1.不修改原数组，仅复制出想要的元素组成新数组

2.如果一个函数两个参数都是下标时，就会含头不含尾。

3.简写：

1.支持负数参数:代表倒数第n个位置

1.选择到倒数第n个位置:

arr.slice(starti,arr.length-n+1);

2.可简写为：arr.slice(starti,-n+1);

3.原理: 其实slice自动用length-n

2.可省略第二个参数，表示一直选取到结尾!

arr.slice(starti);从starti开始一直选到结尾

3.两个参数都省略:从头到尾完整复制一个数组

var arr\_clone=arr.slice();

3.修改数组：splice(英文译：修改)

强调:直接修改原数组

1.删除元素：arr.splice(starti,n);

1.删除starti位置开始的n个元素

2.强调:不必考虑含头不含尾

3.其实:有返回值:返回被删除的元素组成的临时数组

var deletes=arr.splice(starti,n);

4.简写:

1.支持负数下标，表示倒数第n个

2.省略第二个参数(省略n):一直删到结尾

2.插入新元素：arr.splice(starti,0,值1,值2,...);

1.在starti位置插入值1,值2,...

2.原starti位置的值及其之后的值被向后顺移

3.强调:不支持打散数组参数,如果插入一个数组，则变为二维数组

3.替换:先删除旧的，再在同一位置插入新的

arr.splice(starti,n,值1,值2,...)

1.先删除starti开始的n个元素

2.再在starti位置插入值1,值2,...

3.强调: 删除的个数和插入的个数不必相同

4.翻转: arr.reverse();

3.\*\*\*排序:

1.什么是:将数组中的元素按从小到大或从大到小的顺序重新排列

2.何时:所有要给用户展示的数据必须都要先排序

3.如何：arr.sort();

1.默认排序规则:将所有元素都转为字符串，再按字符串升序排列

2.何时:只有按字符串升序排列时，才用默认的sort()

3.问题1:只能按字符串升序排列

解决:（2步）

1.自定义比较器函数:【难点】

1.什么是:专门比较任意两值大小的函数

2.何时:如果sort默认的排序规则不是想要的，可自定义比较器代替sort中默认的排序规则

3.要求:2个参数：a,b

返回值:如果a>b，就返回正数

如果a<b，就返回负数

否则返回0

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

2.最简单的数字升序比较器:

1.如何使用:将比较器函数对象作为参数传入sort()

arr.sort(cmp);

2.强调:cmp后不加()

3.回调:自己定义的函数，自己不调用

而是传入另一个函数中，被另一个函数反复调用。

4.何时:只要对数字元素排序，都要自定义比较器函数

3.简写：arr.sort(function(a,b){return a-b;})

4.问题2:如何降序:

1.解决:只要颠倒比较器结果的正负号，就可改升序为降序

2.最简单的数字降序比较器:

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

### 2.栈和队列:

1.说明:js中没有专门的栈和队列结构，都是用普通数组模拟的

2.栈stack:

1.什么是栈:一端封闭，只能从另一端进出的数组

2.何时:只要希望始终使用最后进入数组的新元素时

3.如何:【补充】

1.结尾出入栈:

1.结尾入栈:arr.push(值) 相当于--> arr[arr.length]=值

即：在末尾追加一个新值

强调:1.其实push可压入多个值

2.不支持打散数组参数

2.结尾出栈:var last=arr.pop();

即：移出数组末尾的最后一个元素

pop是popup的简写，译：弹出

2.开头出入栈:

1.开头入栈:arr.unshift(值) 相当于--> arr.splice(0)...

强调:开头入栈后的元素顺序和结尾入栈后的元素顺序是相反的

2.开头出栈:var first=arr.shift();

4.总结:

1.向数组中添加元素4种:

1.concat():

1.不修改原数组，返回新数组

2.在结尾拼接元素

3.支持打散数组类型参数

2.splice():

1.直接修改原数组

2.在任意位置插入新元素

3.不支持打散数组类型参数

3.push():

1.直接修改原数组

2.只能在结尾拼接元素

3.不支持打散数组类型参数

4.shift():

1.直接修改原数组

2.只能在开头拼接元素

3.不支持打散数组类型参数

2.取出数组元素:4种

1.slice():

1.可获取任意位置的任意个元素

2.不修改原数组，返回选中的元素组成的新数组

2.splice():

1.删除任意位置的任意个元素

2.直接修改原数组

3.返回被删除的元素组成的新数组

3.pop():只能从结尾删除一个元素，并返回

4.shift():只能从开头删除一个元素，并返回

3.队列queue:

1.什么是:只能从结尾进入，从开头出的数组

2.何时:只要希望按照先来后到的顺序使用数组元素时

3.如何:

1.从结尾入队列:arr.push(值);

2.从开头出队列:var first=arr.shift();

### 3.二维数组:

1.什么是:数组中的元素，又引用了另一个子数组

2.何时:2种:

1.保存横行竖列的二维数据

2.一个大的数组中，还需要对元素进行更细致分类

3.如何:

1.创建:2种:

1.先创建空数组，再添加子数组:

var arr=[];

arr[0]=[0,0,0,0];

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

2.简写：创建数组同时，初始化子数组

var arr=[

[0,0,0,0],

.... ,

[0,0,0,0]

];

2.访问:arr[r][c] 每个元素的用法和普通数组的元素完全一样

3.越界:二维数组的行下标r，不能越界！越界报错！！

4.遍历二维数组:外层循环控制行，内层循环控制列

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

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

data[r][c] //当前正在变量的元素

}

}

## Day06

正课:

1.String

内置对象

包装类型

字符串API

2.\*\*\*\*正则表达式

### 1.String

1.什么是:由多个字符组成的\*只读\*字符数组

vs 数组:相同:1.下标; 2. .length; 3. slice

不同:类型不同，API不通用！

2.内置对象:ES标准中规定的，浏览器厂商已经实现的对象

1.包括:11个:

String Number Boolean -->包装类型

Array Date Math RegExp

Error

Function Object

Global (在浏览器中被window代替)

2.包装类型:

1.什么是:专门包装一个原始类型的值，并提供操作原始类型值的API

2.为什么:原始类型的值本身不具有任何功能，才需要包装类型对象的帮助，完成功能

3.何时:一般不用手动使用

只要试图用原始类型的值调用函数时，都会自动创建对应包装类型的对象，封装要操作的值，并使用函数执行功能。

比如：

1.保留两位小数：

n.toFixed(2) （=> typeof n=>number）

=>new Number(n).toFixed(2)

2.字符串转为unicode号

str.charCodeAt() （=>typeof str=>string）

=>new String(str).charCodeAt()

3.String API:强调:所有String API都无权修改原字符串，只能返回新字符串

1.大小写转换:

1.将字符串中所有字母，统一转为大写或小写

str.toLowerCase() 转小写

str.toUpperCase() 转大写

2.何时:不区分大小时，都要先转为一致的大小写，再比较或判断

例如：验证码，用户名，电子邮件

2.获得指定位置的字符

1.获得指定位置的字符:

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

2.获得指定位置字符的unicode号【编码】

str.charCodeAt(i) --->即：获得i位置的字符的unicode号

3.将unicode号，反向转回原文的字符串【解码】

String.fromCharCode(unicode号)

强调:一次只能转一个字

3.获取子字符串:[！！！注：starti/endi 指的是当前字符串的开始、结束的下标]

str.slice(starti,endi+1)

str.substring(starti,endi+1)用法同slice，但不支持负数

str.substr(starti,n)

-->即：从starti开始，获取n个字符

-->相当于 str.slice(starti,starti+n);

4.查找关键词:4种:[查水表]

1.查找一个固定关键词的位置:

var i=str.indexOf("关键词",fromi)-->即：从fromi位置开始，找str中下一个"关键词"的位置

1.返回值:找到的关键词的下标位置i

如果没找到，返回-1

2.简写:省略fromi，默认从0开始

3.特殊:查找\*最后一个\*关键词的位置:

var (last)i=str.lastIndexOf("关键词");

4.问题:只能查找一个固定关键词的位置

5.解决:正则表达式 模糊匹配 多种关键词

2.判断字符串中\*是否包含\*符合规则的关键词

var i=str.search(/正则规则/i)

1.返回值:返回关键词的下标的位置i

如果没找到，返回-1

2.问题:正则表达式默认都是大小写敏感的

3.解决:在第二个/后加i,表示忽略(ignore)

4.何时:只要仅判断有没有关键词时，首选search

5.问题:

1.只能查找一个关键词

2.只能返回位置，无法返回关键词内容

3.查找所有关键词的内容:

var kwords=str.match(/正则规则/ig);

1.返回值:返回所有关键词组成的数组

如果没找到，返回null

2.何时:仅希望获得关键词内容时

3.问题1:正则表达式默认都只匹配第一个关键词

解决:在第二个/后加g，表示全部(global)

4.问题2:仅能获得内容，无法获得每个关键词的位置

解决:见1.3.4.4（即：下一行）

4.即查找关键词内容，又查找位置:

要依据regExp.exec() -->次日讲！

5.替换:将字符串找到的关键词，替换为指定新内容

1.简单替换:将所有关键词都替换为统一的内容

str=str.replace(/正则规则/ig,"新值");

强调:所有字符串API都无权修改原字符串，只能返回新字符串

2.高级替换:根据每个关键词的不同，动态选择替换不同的新值

str=str.replace(/正则规则/ig,function(kw){

kw //可自动获得本次找到的关键词

return //根据kw的不同，动态返回不同的值

})

6.衍生:删除:将关键词替换为""

str=str.replace(/正则规则/ig,"")

7.切割:按指定字符，将一个字符串切割为多段子字符串

1.简单:分隔符是固定的:

var subStrs=str.split("分隔符")

返回:多个子字符串的数组

切割后的结果中，不包含分隔符的

2.复杂:分隔符不是固定的:

var subStrs=str.split(/正则/)

3.固定套路:将字符串打散为字符数组:var chars=str.split("");

### 2.\*\*\*\*正则表达式:Regular Expression

1.什么是:规定一个字符串中字符出现规律的规则

2.何时:2种:

1.模糊匹配多种关键词

2.密码格式

3.如何:

1.最简单的正则表达式，是关键词原文本身

2.字符集:

1.什么是:规定一位字符可用的备选字符列表的集合

2.何时:只要一位字符，有多种可能备选字时

3.如何:[备选字符列表]

4.强调:一个字符集[]内，只能规定一位字符的备选字符列表

5.简写:如果字符列表中部分字符是连续的，可用-省略中间字符

比如:[0-9]一位数字

[a-z]一位小写字母

[A-Z]一位大写字母

[A-Za-z]一位字母

[0-9A-Za-z]一位字母或数字

[\u4e00-\u9fa5] 一位汉字

6.特殊:除了...

[^字符列表] 除了4、7 例：[^47]

3.预定义字符集:

什么是:对常用特定字符集的简写:(4个）

\d 一位数字 =>[0-9]

\w 一位字母,数字或下划线\_ -->[0-9A-Za-z\_]

\s 一位空字符:空格,Tab...

.通配符：匹配任意字符

4.量词:

1.什么是:专门规定字符集出现次数的规则

2.何时:只要规定字符集出现的位数/次数

3.如何:量词必须紧跟在字符集之后，修饰相邻的前一个字符集。

4.包括:2大类:

1.有明确数量边界的:

字符集{n,m} 至少n个，最多m个

字符集{n,} 至少n个，多了不限

字符集{n} 必须n个

2.没有明确数量边界的:

字符集? 可有可无，最多1个

字符集\* 可有可无，多了不限

字符集+ 至少一个，多了不限

5.选择和分组:

1.选择:或 -->即：多个规则，只要匹配其一即可！

规则1|规则2|...

|在正则中，优先级最低

2.分组:用()将多个规则分为一组

1.何时:如果希望量词同时修饰多个字符集时，就要先将多个字符集分为一组，再用量词修饰（）分组

2.为什么:默认字符集仅修饰相邻的前一个字符集

3.身份证号:15位数字 2位数字 1位数字或Xx

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

后两部分，可有可无，最多1次

4.手机号:

+86或0086 可有可无，最多一次--> (\+86|0086)?

空字符 可有可无，多了不限--> \s\*

1 --------------------> 1

3,4,5,7,8 选一个--------------> [34578]

9位数字 --------------------> \d{9}

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

5.微信 wei xin w x

(微|w(ei)?)\s\*(信|x(in)?)

(微|[Ww]([Ee][Ii])?)\s\*(信|[Xx]([Ii][Nn])?)

6.指定匹配位置:3种:

1. ^ 字符串开头 比如:字符串开头的空字符:^\s+

2. $ 字符串结尾 比如:字符串结尾的空字符:\s+$

开头或结尾的空字符:^\s+|\s+$

3. \b 单词边界:

1.开头^

2.结尾$

3.空格

4.标点符号

例:查找单词no，不包含单词内的no

\bno\b -->即：给no的前后边界都加\b

## Day07

正课:

1.RegExp

2.Math

3.Date

### 1.RegExp正则表达式对象

1.什么是:专门封装一条正则表达式，并使用正则表达式执行查找和验证的API

2.何时:

1.验证用户输入的字符串的格式

2.既查找关键词内容，又查找关键词位置

3.如何:

1.创建:2种:

1.直接量:var reg=/正则/ig;

何时:如果正则表达式是固定不变的

问题: //之中不支持js表达式,也就是不支持动态生成正则表达式

2.用new:var reg=new RegExp("正则","ig");

何时:如果正则表达式需要动态拼接(.join)生成

强调:""中再出现\或"，都要改为\\和\"

2.API:2个:

1.验证格式:

var bool=reg.test(str) 验证str是否符合reg规则要求

1.验证通过返回true，不通过返回false

2.问题:默认正则只要找到部分匹配就返回true

3.解决:凡是验证，正则内容必须前加^，后加$

2.即查找内容，又查找位置:

var arr=reg.exec(str) 在str中查找下一个符合reg规则的关键词的内容和位置

1.返回值:返回一个数组arr:[0:关键词内容,index:下标位置]

arr[0] 本次找到的关键词的内容

arr["index"] 本次找到的关键词的位置

如果没找到，就返回null

2.问题:每次只返回本次找到的一个关键词

3.解决:用循环反复调用reg.exec

4.原理:每次调用exec，做3件事:

1.将本次找到的关键词保存在arr的0位置

2.将本次找到的关键词位置保存在arr的index位置

3.修改reg对象的lastIndex=index+关键词的字数，

reg.lastIndex:记录下次开始查找位置，默认为0

每执行完一次exec，exec会自动修改lastIndex为当前位置+关键词长度

ex:console.log("在"+arr["index"] +"位置找到关键词："+ arr[0]+",下次从"+ reg.lastIndex+"位置找起");

### 2.Math:

1.什么是:专门保存数学计算的常量和函数的对象

2.何时:只要执行数学计算

3.如何:

1.创建:Math不用创建，不能new，用直接调用！

2.API:

1.取整:3种:

1.上取整:只要超过，就取下一个整数

Math.ceil(num) -->ceil 译：天花板

2.下取整:只要超过，就省略小数部分

Math.floor(num)---->只能对纯数字

vs parseInt(str)--->能去掉数字之后非数字字符，再取整

3.四舍五入取整:

1.Math.round(num)

缺点:只能取整,不能指定小数位数

优点:返回number, 可直接计算

2.vs n.toFixed(d)

优点:可按任意小数位数四舍五入

缺点:返回字符串，要先转化，再计算

3.解决办法：可自定义round函数:[详见示例代码文件]

ex:function round(num,d){

num=num\*Math.pow(10,d);

num=Math.round(num);

num=num/Math.pow(10,d);

return num;

}

2.乘方、开平方:

Math.pow(底数,幂)乘方

ex:10的2次方: Math.pow(10,2)=100

Math.sqrt(n)开平方

Math.sqrt(9)=3；

3.最大值和最小值:

Math.max(值1,值2,值3,...)

Math.min(值1,值2,值3,...)

1.问题:max/min不支持数组参数

2.解决:Math.max.apply(null,arr)

其中:apply表示"调用"的意思，还可自动打散数组类型参数！

4.随机数:

Math.random() 0<=r<1-->默认值：小数

1.公式:在任意min~max之间取一个随机整数:

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

ex:在1-16之间随机生成：

var r=parseInt(Math.random()\*(16+1)

2.简写:如果0~max之间:随机整数

var r=parseInt(Math.random()\*(max+1))

### 3.Date

1.什么是:封装一个时间值，并提供操作时间的API

2.何时:今后只要保存，操作时间，都要使用Date

3.如何:

1.创建:4种:

1.自动获得当前系统时间:

var now=new Date();

强调:只能获得客户端本地时间

2.创建日期对象并封装自定义时间:

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

如果只关心日期，不关心时间，后半部分时分秒可省略。省略后，默认是00:00:00

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

3.使用毫秒数创建日期对象:

1.原理:日期对象中保存的是1970年1月1日0点至今的毫秒数——整数

2.为什么:时间段，不受时区影响

相同的毫秒数:在不同时区可显示不同的时间点

3.总结:今后在数据库中存储时间或在程序中存储时间都用毫秒数，不要用文字。

获取当前日期的毫秒数(ms):var ms=now.getTime();

获得日期中的毫秒数:var ms=date.getTime();

4.问题:人看不懂毫秒数

5.解决:将毫秒数转化为程序中的日期对象，再输出

var date=new Date(ms);

4.复制一个日期对象:

1.为什么:日期的计算只能直接修改原日期

2.后果:计算后，原日期无法保存

3.解决:今后如果希望同时保存开始时间和结束时间，

都要先将开始时间复制一份，再用副本计算结束时间。

4.如何:var date2=new Date(date1);

2.API:3句话:

1.单位:

FullYear Month Date Day

Hours Minutes Seconds Milliseconds

2.每个单位都有一对儿getXXX()/setXXX()方法

getXXX() 获取指定单位的值

setXXX() 设置指定单位的值

1.※特例:Day没有setXXX()方法[因为：不能随便修改星期]

2.其中:get负责获得指定单位上的数值

set负责修改指定单位上的数值

3.比如:date.getFullYear() date.getMonth()...

date.setFullYear(n) date.setMonth(n)...

4.强调:setXXX(n)可自动调整时间进制

3.取值范围:

FullYear:和现实中年份一致

Month:取值：0~11 现实中:1~12

计算机中的月份值，比现实中少1

需要修正

Date:1~31 和现实一样

Day:0~6 现实:日~六 和现实中一致[不能set！！！]

Hours:0~23 和现实一样

Minutes/Seconds:0~59

3.日期计算:

1.两日期对象可直接相减计算时间差(ms差)-->应用：倒计时

2.对任意单位做加减:3步:

1.取值:var date=now.getDate();

2.做加减:date+=30;

16+30=46

3.放回去:now.setDate(date);

setDate可自动调整进制

4.简写:now.setDate(now.getDate()+30)

5.问题:setDate直接修改原日期对象，无法保留开始时间。

6.解决:在计算前，都要先将开始时间复制一个副本，再用副本计算结束时间。

4.日期格式化:

toString(date) 将date转为当地时间的完整格式

date.toLocaleString() 将date转为当地时间简化版格式

date.toLocaleDateString() 将date转为当地时间简化版格式，仅保留了日期部分.

date.toLocaleTimeString() 将date转为当地时间简化版格式，仅保留了时分秒部分.

date.toGMTString() 将date转为0时区国际标准时间

# DOM

## Day01

正课:

1.什么是DOM

2.\*\*\*查找

3.修改

### 1.什么是DOM

1.DOM:Document Object Model 文档对象模型

2.什么是:专门操作网页内容的API标准--W3C

W3C指定的标准，所有浏览器厂商遵照实现

3.何时:今后只要操作网页内容，就必须用DOM

4.为什么:为了统一操作网页内容的API

用DOM标准操作网页内容几乎100%兼容

5.分为:核心DOM和HTML DOM

1.核心DOM:最初制定的操作一切结构化文档的统一API

优点:万能

缺点:繁琐

2.HTML DOM:专门操作HTML文档的API，对核心DOM中部分常用API进行了简化

优:简洁

缺:不是万能

3.总结

实际开发中不必区分核心DOM和HTML DOM

都是优先使用简单的API

如果简单地实现不了，用复杂的补充

6.\*\*\*DOM树:

1.网页中的一切内容在内存中都是以树形结构存储

2.所有内容(元素，属性，文本)在树上，都是一个节点对象

3.为什么:树形结构是保存不确定层级深度的上下级包含关系最好的结构

4.根节点:树结构都有唯一的一个树根节点:document节点

即：DOM有唯一的一个根节点document

5.所有内容节点都是document的后代节点（子节点）

7.\*\*\*节点对象:

1.网页中的所有内容，都是DOM树上的节点对象

2.类型:Node 译：节点

3.三大属性:

1.nodeType:表示节点的类型

值都是整数，共4个类型:

document 9[是编号]

element 1※

attribute 2 属性

text 3※

1.何时:只要区分节点类型时

2.为什么:不同类型的节点可执行的操作不同

3.问题:无法进一步区分不同元素

4.解决:nodeName:节点名

2.nodeName:节点名(元素的标签名)

1.何时:只要进一步细致区分元素种类时

2.包括:

document #document

element 元素的标签名(全大写)(BODY、ID)

attribute 属性名

text #text

3.其实，nodeName可代替nodeType来鉴别节点类型

3.nodeValue:节点值(了解)包括:

document null

element null

attribute 属性值

text 文本内容

8.DOM操作的流程:增删改查+事件处理

查找触发事件的元素-->绑定事件处理函数->查找要修改的元素-->修改属性/样式,添加,删除

### 2.\*\*\*查找:4种:

1.不需要查找就可直接得到:3种

document

document.documentElement --> 获得html的节点

document.head --> head

document.body --> body

2.按节点间关系查找:

1.何时:只要已经获得一个节点，要找周围节点时

2.包括:2大类关系:父子关系、兄弟关系

1.节点树:包含一切内容节点的树结构

※问题:包括看不见的空字符文本节点的干扰

1.父子:

elem.parentNode 获得一个元素的父节点

elem.childNodes 获得所有\*直接\*子节点的集合

elem.firstChild 获得父节点的下的第-一个\*直接\*子节点

elem.lastChild 获得父节点的下的最后一个\*直接\*子节点

2.兄弟:

elem.previousSibling 获得一个节点相邻的前一个节点

elem.nextSibling 获得一个节点相邻的下一个节点

2.元素树:只包含元素节点的树结构

※说明:元素树不是一棵新树,只是节点树的部分子集

1.父子:

elem.parentElement 获得一个节点的父元素

elem.children 获得\*直接\*子元素的集合

elem.firstElementChild 获得父元素下的第一个直接子元素

elem.lastElementChild 获得父元素下的最后一个直接子元素

2.兄弟:

elem.previousElementSibling 获得一个元素相邻的前一个兄弟元素

elem.nextElementSibling 获得一个元素相邻的下一个兄弟元素

优:不受空文本的干扰

缺:IE8不兼容 IE9+才满足

3.强调:childNodes和children返回的不是数组，而是类数组对象。

1.类数组对象:长的像数组的对象

vs 数组:相同:1.下标，

2. .length

不同:类型不同，API不通用

2.简称:集合

其实，childNodes和children都返回动态集合

3.动态集合:不实际存储数据，每次访问集合，都重新查找DOM树。

1.问题:反复访问集合，会导致反复查找DOM树

2.遍历动态集合[live collection]: 方法：先缓存长度，再遍历

错误的遍历:for(var i=0;i<children.length;i++){...}

正确的遍历:for(var i=0,len=children.length;i<len;i++)

3.笔试:用节点间关系，遍历：查找父节点下所有后代节点

1.递归遍历:2步:[平时写代码不用，笔试面试会考到]

Step1:先仅查找直接子节点

Step2:对每个直接子节点，调用和父节点完全相同的函数

1.算法:深度优先遍历:每个节点都优先遍历子节点，子节点遍历完，才遍历兄弟节点。

2.问题1:递归时，函数内的函数名不能写死

解决:用arguments.callee指代当前函数自己。

※arguments.callee:

1.什么是:专门指代当前正在调用的函数自己

2.何时:只要递归，在函数内调用自己，必须用arguments.callee代替

2.为什么:避免修改外层函数名后，还要重复修改内部调用的函数名

3.强调:因为递归算法效率极低

所以ES6或新版浏览器中已经禁止使用arguments.callee, 意味着不再推荐使用递归算法。

3.问题2:遍历查找的问题：递归的执行效率极低

1.因为:遍历的步骤非常繁琐，而查找功能又会频繁使用

2.解决:使用专门的查找API

2.用循环代替递归: [iterator译：迭代器]

1.节点迭代器对象:专门按照深度优先遍历的顺序依次访问每个子元素的对象，并返回每个节点对象的小工具-->内置深度优先算法

2.如何:2步:

1.创建节点迭代器对象:

var iterator=document.createNodeIterator(

parent, NodeFilter.SHOW\_ALL, null, false

SHOW\_ELEMENT

);

2.循环反复调用迭代器NodeIterator的nextNode()函数，调到下一个节点。直到返回null结束。

做2件事:

1.返回当前节点

2.蹦到下一个节点

到返回null退出

do{

var node=iterator.nextNode();

if(node!=null){

console.log(node.nodeType!=3?node.nodeName:node.nodeValue);//输出node

}else break;

}while(true);

4.按HTML查找:4种:

1.按id查找1个元素:[各种id名]

var elem=document.getElementById("id")

强调:

1.必须用document调用！

2.只返回1个元素！

2.按标签名查找多个元素:[ex: div button td ...]

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

强调:

1.可在任意父元素上调用，仅找当前父元素下的指定元素。

2.不但找直接子元素，而且找所有后代元素

3.返回动态集合【原理 见本笔记2.2.3.2】

3.按name属性查找:——了解[后面有更好地方法代替]

<input type="text" name="txtName">

1.何时:在表单中查找有name属性的表单元素时

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

2.强调:

1.只能用document调用

2.返回动态集合【原理 见本笔记2.2.3.2】

4.按class属性查找:[各种含class的类名-->class="类名";]

var elems=parent.getElementsByClassName("class");

1.强调:

1.在任意父元素上调用

2.返回动态集合【原理 见本笔记2.2.3.2】

3.不但找直接子元素，而且在所有后代中查找

4.不要求完整匹配class，只要包含就找到

2.问题:每次只能按一个条件查找，如果条件复杂，代码会很繁琐

3.解决:当查找条件复杂时，要用选择器查找

5.按选择器查找:Selector API:2个:

1.只找一个元素: query 译：查询

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

2.找多个元素:

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

注：返回值：找到：返回一个值，没找到：null

3.强调:

1.可在任意父元素上调用

2.不仅查找直接子元素，且查找所有后代元素

3.返回非动态集合※※

非动态集合:数据直接保存在集合本地，无序反复查找。【原理：因为实际存储数据，即使反复访问，也不会导致反复查找DOM树。所以：遍历时: 不用先缓存length】

注：与动态集合对比理解【见本笔记2.2.3.2】

4.选择器兼容性，受制于当前浏览器的兼容性的要求

4.笔试:按HTML查找 vs 按选择器(Selector API)查找

1.返回值:

1.按HTML查找:返回动态集合

优点:效率高！只需返回需要的数据即可，不需要返回完整数据

缺点:造成反复查找

2.按选择器查找:返回非动态集合

优点:不会反复查找

缺点:首次执行效率低，首次会返回完整的数据，但现如今的浏览器性能都很好，可忽略

2.效率:

1.首次查找按HTML效率快！

2.按selector查找稍慢!

3.易用性:当查找条件复杂时

1.按HTML查找:繁琐

2.按选择器查找:简洁

5.总结:

1.如果已经获得一个元素，找周围元素:用节点间关系查找

2.如果通过一个条件就可找到想要的元素:按HTML查找

3.如果查找条件复杂时:用selector API

6.总结返回值:

1.凡是返回一个元素的API，如果没找到，都返回null

2.凡是返回多个元素的API，如果没找到，都返回空元素的集合

7.练习:事件绑定://当事件发生时，自动执行保存的事件处理函数

elem.on事件名=function(){

this //自动获得触发事件的.前的当前元素

}

ex:点击事件 elem:已定义好的元素名称

elem.onclick=function(){}

### 3.修改:

1.内容:

1. .innerHTML:获取或设置元素开始标签到结束标签之间的 HTML代码 片段。

2. .textContent:获取或设置元素开始标签到结束标签之间的 纯文字 内容。

1. 2件事:

1.去掉文本中内嵌的标签

2.将所有转义字符翻译为正文

2.强调:IE8不兼容, 用.innerText代替

3. .value:获取或设置表单元素的内容，因为表单<input/>是单标记

2.属性:

3.样式:

内联样式:elem.style.css属性名="值"

强调:css属性名都在<script>或js文件里，命名要去横线变驼峰

ex:CSS中：background-color:-->改为：js中：backgroundColor:

4.过渡动画:

1.js中:用支持过渡的css属性修改样式:(不要用display!!!)

width height opacity ....

2.css中:

起始样式(设置)

overflow:hidden

transition:all .5s linear

凡是带transition的元素，无论以任何手段修改样式属性值，都自带过渡效果

## Day02

正课:

1.修改:

属性，样式

2.添加/删除/替换

### 1.修改:

1.内容:.innerHTML .textContent .value

1.属性:3类:

1.标准属性:2种方式访问:

1.核心DOM:4个API:——繁琐

1.所有属性节点都保存在元素的attributes集合中

1.获取属性值:elem.getAttribute("属性名")

2.修改属性值:elem.setAttribute("属性名","属性值")

3.是否包含:elem.hasAttribute("属性名")

4.移除属性:elem.removeAttribute("属性名")

2.状态属性:selected checked disabled

不能用核心dom访问，只能用HTML DOM访问

值都是bool类型

2.HTML DOM:——简洁——首选

HTML DOM：将所有HTML标准属性都封装为了元素对象的属性，可直接用.访问。

1.获取属性值:elem.属性名

2.修改属性值:elem.属性名=值

3.是否包含:elem.属性名!==""

4.移除属性:elem.属性名=""

特殊:class属性:class是js语言的保留字，所以DOM中的class属性被迫改名为className

2.自定义扩展属性:2种:

1.核心DOM:——繁琐

不能用HTML DOM访问

2.HTML5语法:——简洁 data-toggle="dropdown"

定义自定义属性:data-属性名="值"

获取/修改属性值:elem.dataset.属性名=值

2.样式:2种:

单独修改一个css属性

1.内联:写在元素的style属性中的样式

1.获取或设置内联样式:

elem.style.css属性

2.强调:

1.css属性名都要去横线变驼峰:

比如:background-color->backgroundColor

list-style-type->listStyleType

2.所有css属性值都是字符串(可能带单位)

计算前都要先去单位，转数字再计算。

3.设置内联:优:优先级最高，当前元素独有，不影响其它元素

获取内联:

1.问题:实际开发中可能几乎不包含内联样式。用elem.style可能无法获得任何样式。

2.解决:用getComputedStyle(elem)代替elem.style

ComputedStyle:计算后的样式: 最终应用到元素上的所有样式的综合.

3.读取:不能用style.css属性名

因为style只能获得内联样式，无法获得样式表中继承/层叠来的样式

解决:今后凡是读取样式，都要读取计算后的样式(Computed)。

计算后的样式:最终应用到元素上的所有样式的综合，并将相对单位计算为绝对单位。

4.如何2步:

1.获得计算后的所有css属性的总和：style对象

var style=getComputedStyle(elem)

2.从style中获得想要的css属性

style.css属性

5.强调:计算后的css属性都是只读

因为计算后的css属性可能原来是共用的，一旦修改，牵一发而动全身。

2.内部/外部样式表:——危险:3步:

1.找样式表对象:var sheet=document.stylesheets[i]

2.找cssRule对象:var cssRule=sheet.cssRules[i]

如果是keyframes,就需要继续找下级cssRule

3.修改cssRule的style下的css属性值:

cssRule.style.css属性=值

3.问题:内联样式中，每次只能修改一个css属性

解决:如何批量修改一个元素的多个属性:2步:

1.先在css中将多个属性定义为一个class

2.在js中修改元素的className为指定的class

总结：即：通过修改元素的class属性，来批量应用多个css属性

### 2.添加/删除/替换:

1.添加:3步:

1.创建空元素:

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

比如:var a=document.createElement("a")

相当于<a></a>

2.设置关键属性:

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

a.innerHTML="go to tmooc";

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

3.将新元素添加到DOM树指定父元素下:

1.新建的元素，必须添加到DOM树才能显示出来（3种）:

1.末尾追加:parent.appendChild(elem)

2.中间插入:parent.insertBefore(elem, child)

3.替换:parent.replaceChild(elem,child)

2.优化:尽量减少操作DOM树

1.原因：每改变一次DOM树，都会反复触发layout降低页面的响应速度

2.原理:页面加载过程:

html -> DOM Tree

↓

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

↑ 计算每个元素的精确布局

css -> CSSOM 耗时

3.解决:2种:

1.如果需要同时添加父元素和子元素时，就要在内存中先将子元素添加到父元素，再将父元素整体一次性添加到DOM树

2.如果父元素已经在页面上，要添加多个平级子元素，就要借助文档片段。

1.文档片段: 内存中临时保存多个子元素的虚拟父元素

2.何时:要添加多个平级子元素

3.如何:3步:

1.创建文档片段:

var f0rag=document.createDocumentFragment();

2.将平级子元素添加到frag中

frag.appendChild(elem);

3.将frag整体添加到页面父元素下

parent.appendChild(frag);

说明:将子元素添加到DOM树后，frag自动释放，不占用页面空间。

2.删除:parent.removeChild(child);

## Day03

正课：

1.HTML DOM常用对象:

Image Select/Option Table/.... Form/...

2.什么是BOM

1.打开和关闭窗口

2.窗口大小

3.\*\*\*\*定时器（重点）

4.BOM常用对象:

history location navigator screen

### 1.HTML DOM常用对象:

1.Image:指代页面上一个img元素

创建:var img=new Image();

【创建空元素:相当于：var elem=document.createElement("标签名")】

2.Select:指代页面上一个<select>元素

1.属性:

sel.selectedIndex 获得当前选中项的下标位置

sel.value 获得当前选中项的值(value)

如果当前option没有value，则用innerHTML（内容）代替

sel.options 获得当前sel下所有option元素的集合

sel.options.length 选项的个数

sel.options.length=0 清除所有选项

sel.length 相当于=> sel.options.length选项的个数

sel.length=0 清除所有选项

2.方法:sel.add(option) 向sel末尾追加一个option

相当于-->parent.appendChild(elem)

强调:不支持文档片段

sel.remove(i) 移除i位置的选项

3.事件:sel.onchange 当选中项发生改变时触发

3.Option:指代页面上一个option元素

1.创建:var opt=new Option(text,value)

2.属性:opt.text代替opt.innerHTML

4.Tabel:指代一个table元素:

1.table管着行分组:创建，删除，获得

1.创建:

var thead=table.createTHead();

var tbody=table.createTBody();

var tfoot=table.createTFoot();

2.删除:

table.deleteTHead();

table.deleteTFoot();

3.获取:

table.tHead

table.tBodies[i]

table.tFoot

2.行分组:tHead tBody tFoot

管着tr:

1.创建:var tr=tXXX.insertRow(i);-->插入行

1.说明:如果i位置有行，则原位置的行向后顺移

2.常用操作:

1.末尾追加一个新行:tXXX.insertRow();

2.开头插入:tXXX.insertRow(0)

2.删除:var tr=tXXX.deleteRow(i)

强调：删除当前行分组tXXX中下标为i的行

3.获取:var tr=tXXX.rows

3.行:

管着格td:

1.创建:var td=tr.insertCell(i)

强调:insertCell只能创建td，不能创建th

方法：用tr代替th创建th--> var th=tr.insertCell();

2.删除:var td=tr.deleteCell(i)

3.获取:var td=tr.cells[i]

4.删除行:2种:

1.行分组.deleteRow(i)

i 是行在行分组内tXXX中的相对位置, 无法直接获得

2.table.deleteRow(i)

i 是行在整个表中的绝对位置，可直接获得:

tr.rowIndex 表示tr在整个表中的位置

3.总结:今后凡是删除行:table.deleteRow(tr.rowIndex)

4.三大对话框：

1.alert() 警告框

2.prompt() 输入框

3.confirm() 确认框：有确认的取消操作的对话框

1.何时：只要执行危险的操作前(提交更新、删除)都要先确认再执行操作

2.如何:var bool=confirm("提示消息");

如果点确定，就返回true，就执行操作

否则，就返回false，就不执行任何操作

5.Form:代表页面上一个form元素

1.获取:var form=document.forms[i/id]

2.属性:

1.form.elements 包含所有表单元素: input select button textarea

2.form.elements.length 获得表单中表单元素的个数

form.length => form.elements.length

3.方法:form.submit(); 2种://用程序手动提交表单

1.input button 代替input submit

何时:只要希望通过自定义的验证决定是否提交表单

在input button的单击事件中自己调用form.submit();

2.input submit 配合 onsubmit事件 （下节讲）

4.表单元素:

1.获取:form.elements[i/id/name]

2.更简写:如果表单元素有name属性，可直接:form.name

特例：没有name属性的元素:

固定套路:获得倒数第二个按钮:form.elements[form.length-2]

3.方法:

elem.focus(); 让elem获得焦点

elem.blur(); 让elem失去焦点

DOM:实现对网页内容的增删改查

### 2.什么是BOM:Browser Object Model

1.BOM:专门操作浏览器窗口/软件的API-->没有标准

1.window:2个角色:

1.代替ES中的Global充当全局作用域对象

2.封装所有浏览器内置的/DOM/BOM的API

2.window的功能:打开和关闭窗口，弹出对话框...

包括:

1.history:专门保存历史记录,控制前进后退

2.location:专门保存浏览器正在打开的url地址

3.navigator:专门保存浏览器的配置信息

4.document:就是DOM中的document节点

5.screen:专门保存显示设备的信息

6.event:专门保存事件信息

2.打开和关闭窗口:

1.打开一个新窗口:/\*window.\*/open("url","name")

1.其中:

name:新窗口在内存中的名称

浏览器规定:相同name的窗口只能开1个，后开的会替换先开的。

2.预定义值:

1.\_self:用当前窗口的name打开新窗口

结果:新窗口替换当前窗口

2.\_blank:用空name打开新窗口

1.浏览器会自动为每个空name属性的窗口随机分配name属性

2.结果:每个窗口的name都不一样，就可打开多个

2.打开一个超链接:4种:

1.在当前窗口打开，可后退前进:

HTML:<a href="url" target="\_self"></a>

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

2.在当前窗口打开，不可后退:

当前窗口每打开一个新url，都会将新url保存在history中。

如果新url是追加进history中，则可后退

如果新url将当前url替换掉，则无法后退

js:location.replace(url)

3.在新窗口打开，可打开多个:

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

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

4.在新窗口打开，只能打开一个:-->[付款界面]的链接

HTML:<a href="url" target="自定义name"></a>

js:open("url","自定义name")

1.内存中每个窗口都有一个唯一的name属性

2.浏览器规定:同一个name属性的窗口只能打开一个

3.后打开的同名窗口会覆盖先打开的

4.其实，target属性值，就是窗口的name属性

3.窗口大小

文档显示区的大小:

宽:innerWidth 高:innerHeight

4.\*\*\*\*定时器：2种:【很重要，案例全要会】

1.周期性定时器:

1.什么是:让程序每隔一段时间间隔反复执行一次任务

2.何时:让程序每隔一段时间间隔反复执行一次任务

3.如何:3件事:

1.任务函数:function task(){...}

定义了定时器每次要执行的任务

2.启动定时器:var timer=setInterval(task,间隔ms)

1.强调:task是回调，不用加()

2.timer:定时器在内存中的序号:从1开始的整数

3.何时:只要一个定时器可能被停止，都要在启动时先保存住序号

3.停止定时器:clearInterval(timer);timer=null;

1.其中:timer是要停止的定时器序号

2.问题:停止定时器不会自动清除timer变量中的序号，有可能影响下次启动。

3.解决:

凡是停止定时器，都要手动清除timer（2种方式）:timer=null;

1.手动停止:用户通过操作来停止定时器 timer=null;

2.自动停止:

在任务函数中判断临界值

只要达到临界值，就自动停止定时器

4.回调函数--bind(this)

在call,apply,bind中绑定this，【记】在替换回调函数中的this一定要用bind绑定!

1.回调函数的要求:

1.不止执行一次

2.不会立刻执行

2.call的特点:(不用call的原因)

1.只执行一次

2.立刻执行

3.因为在回调函数中，默认的this是指的window

(任何一个函数不用.不用new调用，其中的this默认指向window)

ex:特别是在定时器回调函数中，setInterval函数就没有用.或者用new调用

2.一次性定时器:[闹铃]

1.什么是:程序先等待一段时间后，再自动执行一次任务

2.何时:先等待一段时间再自动执行一次

自动执行一次后，自动停止

3.如何:3件事:

1.任务函数:task()

2.启动定时器:var timer=setTimeout(task,等待ms)

3.停止定时器:clearTimeout(timer);timer=null;

在任务执行前，就停止等待

辨析(自己理解)：

clearInterval(timer);-->多次停止，也可恢复倒计时，直到倒计时全过程完成才终止

clearTimeout(timer); -->一次停止，之后倒计时全过程中断并终止

3.原理:定时器中的回调函数只能在主程序所有语句执行完才能开始执行。

笔试:

ex1:var a=10;

setTimeout(function(){

console.log(a);------------------------//输出：a=11;

},0);

a++;

即：先执行a++;在执行setTimeout函数

ex2:var a=10;

function fun(){

a=100;

console.log("a2="+a);

}

setTimeout(fun,0);--------------------//fun()就是定时器的task()函数

console.log("a1="+a);-----------------------//输出：a1=10;a2=100;

即：先执行页面主程序，之后再执行setTimeout里的回调函数fun的输出

ex3:ex2:var a=10;

function fun(){

a=100;

}

setTimeout(fun,0);--------------------//fun()就是定时器的task()函数

console.log("a1="+a);-----------------------//输出：a1=10;

因为fun()未输出值，与ex2对比理解

5.BOM常用对象:history location navigator screen

1.history:保存当前窗口打开后成功访问过的url的历史记录栈

history.go(n):history.go(0) 刷新

history.go(-1) 后退

history.go(1) 前进

也可前进/后退多步

2.location:保存当前窗口正在打开的url对象

1.属性:

.href 获取或设置完整url，不但可获取，还可设置，实现在当前页面打开新连接

.protocol 协议-----http 或 https

.host 主机名 包含:主机名+端口号

.hostname 仅主机名

.port 端口号

.pathname 相对路径

.hash 获得#锚点地址

.search 获得表单提交后（"?"后）的查询字符串

2.方法:location.assign("url")在当前页面打开新url

其实:=>location.href="url"

=>location="url"

location.replace("url") 在当前窗口打开，替换history中当前url，实现禁止后退

location.reload(fales/true) 刷新

3.笔试:2种刷新:

1.浏览器缓存:浏览器在首次加载页面时，会将变化不大的静态资源(css,图片,html)缓存在客户端本地硬盘。

2.为什么:避免重复加载，节约网络流量，提高加载速度

3.加载页面时:优先在本地找可用的资源，如果本地资源比服务器上资源旧，才从服务器下载新资源。

4.刷新:2种:

1.普通刷新/默认刷新:优先使用本地缓存中的文件，除非文件比服务器上的旧，才被迫下载新文件。

F5

history.go(0)

location.reload(/\*false\*/)

force

2.强制刷新:每次强制跳过浏览器本地缓存，总是从服务器下载新文件。

location.reload(true)

force

3.navigator:封装了浏览器的配置信息的对象

1.cookieEnabled

1.cookieEnabled:返回bool值(true/false), 判断浏览器是否启用cookie

2.什么是:客户端持久保存用户私密信息的小文件

3.为什么:程序内存中的数据都是临时的

4.何时:只要在客户端持久保存用户数据时

5.典型:记住密码，游戏最高分

2.plugins:保存所有插件信息的集合

1.plugins["插件名"]!==undefined 说明装了！

2.什么是: 为浏览器添加新功能的小软件

3.userAgent:保存浏览器名称和版本号的字符串

1.何时:鉴别浏览器的名称和版本号时

2.作业:查找网上辨别浏览器名称和版本号的方法

4.screen:保存客户端显示设备信息

笔试:判断屏幕宽

1.css:媒体查询

2.js:屏幕宽:screen.width 屏幕高:screen.height

## Day04

正课:

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

### 1.\*\*\*\*event

1.什么是:用户手动触发的或浏览器自己触发的页面状态的改变。

事件处理函数: 当事件发生时，自动触发的函数——回调

2.绑定事件处理函数:3种:

1.在HTML中绑定:<ANY on事件名="js语句"

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

问题:不符合内容与行为分离的原则，不便于维护、重用

2.在js中绑定:

1.on事件名:elem.on事件名=function(){

this-->当前触发事件的.前的元素对象elem

}

局限:一个事件，是能绑定一个处理函数

2.addEventListener:可一个事件同时绑定多个处理函数，增加事件监听

1.elem.addEventListener("事件名",fn);

2.还可移除:

elem.removeEventListener("事件名",函数名fn)

3.总结:

1.如果一个事件只有一个处理函数，且不会被移除时，首选onXXX——简单

2.如果一个事件需要同时触发多个处理函数，或处理函数可能被移除时，首先addEventListener——灵活

3.其实addEventListener可以用匿名函数绑定

4.问题:绑定时如果使用匿名函数绑定，移除时无法找到原函数

4.解决:如果一个处理函数可能被移除，就必须用有名的函数

3.事件模型/周期:

DOM: 3个节点

1.捕获:

由外向内，记录各级父元素上绑定的事件处理函数——只记录，不触发

2.目标触发:

优先触发目标元素上的事件处理函数

目标元素(target):最初实际触发事件的元素

3.冒泡:

由内向外，按捕获顺序的反向，依次执行父元素上的事件处理函数

4.事件对象:e

1.什么是:事件发生时，自动创建的，封装事件信息的对象，提供了操作事件的API

2.何时:2种:

1.获得事件信息

2.修改事件的行为时

3.如何:

获取:事件对象，默认作为事件处理函数的第一个参数传入。

function 事件处理函数(e)

//在触发事件时，参数e会自动收到事件对象

4.API对象:

1.取消冒泡:e.stopPropagation()

2.利用冒泡:

1.优化:尽量少的绑定事件处理函数

2.为什么:因为浏览器是用遍历方式查找事件监听并执行，所以事件监听的个数越多，导致页面响应速度慢

3.如何:如果多个平级子元素绑定相同的事件时，仅需要在父元素绑定一次，所有子元素公用即可！

2个难题:

1.如何获得最初触发事件的目标元素:

问题:this不再指实际单击的子元素

而是指向公共父元素

解决:e.target代替this

2.目标元素一定是想要的吗?

问题:可能点在子元素，也可能点在父元素

解决:利用冒泡的事件处理函数中，都要先判断目标元素是否想要，再决定是否触发事件。

5.阻止默认行为:e.preventDefault()

1.何时:只要默认行为不是想要的，就可取消。

2.如何:2个固定场景:

1.阻止a元素作为按钮时，向url末尾添加锚点地址

2.配合submit按钮阻止表单提交

表单的onsubmit事件会在以任何方式提交表单时自动触发。是表单提交的最后一关。专门用于表单中所有验证，决定最终能否提交。

3.总结:2中控制表单提交的方式

1.type="button" 只有验证通过才form.submit()

2.type="submit"

只要验证未通过就e.preventDefault();

3.HTML5 拖拽API 也要用e.preventDefault()阻止浏览器自带的默认拖拽行为

5.事件坐标:3种:

1.相对于屏幕左上角:e.screenX/screenY

2.相对于文档显示区左上角:e.clientX/clientY

3.相对于当前元素左上角:e.offsetX/offsetY

6.页面滚动:

1.事件:window.onscroll

2.获得页面滚动的距离:

document.body.scrollTop

# Javascript core

## Day01

正课:★笔试

1.错误处理

2.\*\*\*\*Function

1.创建

2.重载

3.匿名函数

4.\*\*\*作用域和作用域链

5.\*\*\*\*\*闭包

### 1. 错误处理:

1.错误:程序运行过程中，导致程序无法继续执行的异常情况。

现象：程序一旦出错，默认会报错，并强行退出

2.错误处理:即使程序发生错误，也保证不强行退出的一种机制

3.为什么:避免程序强行中断，导致极差的用户体验

4.何时:只要希望即使程序发生错误，也保证不强行退出

5.如何:

try{

可能发生错误的语句

}catch(err){ //只有发生错误时，才执行

错误处理代码: 1.保存数据; 2. 提示用户; 3. 记录日志

}

其中:err是错误对象:错误发生时，自动创建的，记录错误信息的对象

6.性能问题（效率）:

放入try catch中的代码，可能执行效率会降低，且一旦出错，还要额外创建error对象，占用更多内存空间。

解决:如果可提前预知错误的原因，可用if...else...来代替try catch-->对于开发人员的经验要求非常高

try catch的优点:

1.不影响程序执行效率

2.节约内存

7.主动抛出错误:

1.什么是:在本来程序不出错的情况下，为了以防万一，主动创建一个错误，并抛出。

2.何时:在协作开发中，函数的作者(函数的定义者)，可用主动抛出错误的方式提醒调用者如何正确使用函数。

3.如何:throw new Error("错误信息提示");

8.★笔试:js中错误的类型:6种

1.SyntaxError :语法错误

2.ReferenceError:引用错误，要找的变量不存在(找不到对象)

3.TypeError :类型错误，使用错误的类型调用函数或访问元素(错误的调用了API或错误的使用了()或[])

4.RangeError :范围错误，特指：参数的取值超范围

5.URIError; :统一资源标识符错误

[URI:(Uniform Resource Identifier)即：统一资源标识符，是一个用于标识某一互联网资源名称的字符串。]

[Web上可用的每种资源-HTML文档、图像、视频片段、程序等由一个通用资源标识符(URI)进行定位。]

6.EvalError; :表达式错误

[eval函数:执行一段字符串格式的js表达式]

9.bug:只要导致程序出现问题的原因

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

1.创建函数:3种:

1.声明:(用得最多)

function 函数名(参数列表){函数体; return 返回值}

特点:会被声明提前

2.直接量:

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

1.优点:不会被声明提前

2.解释:

3.何时:只要希望函数的定义不被声明提前时

3.用new:

var 函数名=new Function("参数1","参数2","函数体;return 返回值")

1.强调:所有参数变量和函数体，必须用""(字符串)包裹

2.笔试:以下函数创建正确的是:

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

var fun=function(a,b){return a-b}

var fun=new Function(a,b,"return a-b") XXX

var fun=new Function("a","b","return a-b")

2.重(chong)载(overload):

1.什么是:相同函数名，不同参数列表的多个函数，在调用时可根据参数的不同，自动选择对应的函数执行。

2.何时:如果一项任务，根据不同的参数，执行不同的操作流程时

3.为什么:减少函数(API)的数量，减轻调用者的负担

4.问题:js语法，默认不支持重载语法，即：不允许多个同名函数同时存在！

后果：最后一个函数会覆盖之前所有函数

5.如何变通:

1.仅定义一个函数，不要写参数列表

2.arguments:函数内自动创建的 （argument 译：参数）

用于接收传入函数的所有参数值的类数组对象

类数组对象:长的像数组的对象

vs 数组:

相同:1.下标, 2.length, 3.for遍历

不同:类型不同-->API不通用

3.函数内，根据arguments中参数的\*个数\*或类型，动态决定执行不同操作。

4.问题:参数是否还需要

答:需要:

1.参数用于提示调用者如何正确使用函数

2.参数名一般都比arguments简洁且见名知义

3.匿名函数:

1.什么是:创建函数时，不指定函数名的函数

2.何时:2种:

1.当一个函数仅使用一次时

2.划分临时作用域，避免全局污染时

3.为什么:

1.节约内存

1.垃圾回收:垃圾回收器程序会自动释放不被任何变量引用的对象

2.垃圾回收器:专门检查并回收不再被使用的对象的小程序

2.避免全局污染

4.如何:2种:

1.回调:自己创建函数后，不调用，传递给其它函数，让其它函数使用。

比如:

arr.sort(function (a,b){return a-b})

str.replace(reg,function(kw){return 替换值})

btn.addEventListener("click",function(){...})

绑定事件处理函数

2.自调:定义匿名函数后，立刻调用自己，调用后立刻释放

1.如何:2种:

(function(...){...})(...)---->推荐

+function(...){...}(...)----->不推荐

2.何时:只要不希望造成全局污染时，都可将代码封装在匿名函数中，自调执行。

3.为什么:划分临时作用域，避免创建全局变量

4.\*\*\*作用域(scope)和作用域链(scopechain)

1.什么是作用域:一个变量可用范围，也是一个变量的实际存储位置。

1.包括:2种:

1.全局作用域:window

保存全局变量:

优点:随处可用，可重复使用

缺点:易造成污染

2.函数作用域:AO(active object)(活动对象)

保存局部变量

优点:仅函数内可用，不会造成污染

缺点:不可重用

2.\*\*\*\*\*函数的生命周期:

1.函数(程序)开始执行前:

1.创建执行环境栈[ECS](数组)，用于保存将要调用的素有函数

2.在栈中添加第一个函数调用:浏览器的主程序main()

3.主程序创建全局对象window

2.定义函数:

1.用函数名创建全局变量

2.创建函数对象封装函数体

3.函数名变量引用函数对象

4.函数对象的隐藏属性(籍贯):scope指回window

3.调用函数:

1.在ECS中添加本次函数调用的记录

2.创建本地函数调用所需的作用域对象AO(活动对象)

3.在AO中添加函数中定义的所有局部变量(var 声明的和参数变量)

4.AO的隐藏属性parent指向父级作用域对象

5.变量使用顺序:

优先使用函数作用域AO中的局部变量

AO中有，就不去全局找

AO中没有，才去全局找(局部没有，才去父级作用域找)

4.函数调用后:

1.ECS中本函数调用的记录出栈

2.导致函数作用域AO没人引用，被释放

3.导致AO中的局部变量一起释放

3.作用域链:由多级作用域对象连续引用形成的链式结构

职责:(2个)

1.保存着所有变量

全局变量都保存在window中

局部变量都保存在AO中

2.控制着变量的使用顺序:先局部，再全局

5.\*\*\*\*\*闭包(closure):

1.什么是:即重用变量，又保护变量不被污染的一种机制

2.为什么:全局变量和局部变量都有优缺点:

1.全局变量:

优点:可反复使用

缺点:随处可用——易被污染

2.局部变量:

优点:仅在函数内可用，不会被污染

缺点:不可重用

3.何时:即重用变量，又保护变量不被污染

4.如何:3步:

1.用外层函数，包裹住受保护的变量和内层函数

2.外层函数将内层函数返回到外部

3.使用者调用外层函数，获得返回的内层函数对象

5.★笔试:

闭包形成的原因:外层函数的作用域对象(AO)无法释放

6.闭包的问题(缺点):

比普通函数占用更多内存空间-->多的是外层函数的AO

建议:如果闭包不再使用，就要及时释放，如何释放?

解决:将引用闭包函数的变量赋值为null

导致内层函数对象先释放

导致外层函数的AO一并释放

7.笔试:画简图:2步:

1.找受保护的变量:

外层函数的局部变量(var的或参数变量)

确定在外层函数调用后，受保护变量的最终值

2.找抛出的内层函数:2个途径:

1.直接return function

2.给全局变量赋值:全局变量=function

3.隐藏:

将内层函数放入一个数组对象中，整体隐藏返回

一次外层函数调用，返回的多个内层函数，共用同一个受保护的变量。

## Day02

正课:

### 1.\*\*\*\*\*面向对象OOP

三大特点:封装，继承，多态

1.什么是面向对象编程:程序中都是用对象来描述显示中一个具体事物。

1.为什么:便于维护和使用数据

2.何时:今后只要维护的数据量或种类多时，都要用对象的方式，集中管理每个事物的属性和功能。

3.什么是对象:内存中集中存储一个事物的属性和功能的程序结构

4.如何:三大特点:封装，继承，多态

1.封装:创建一个对象

1.什么是:将一个事物的属性和功能集中保存在一个对象中，再起一个统一的名字

事物的属性会成为对象中的属性

事物的功能会成为对象中的函数

2.为什么:便于维护和查找事物的属性和功能

3.何时:只要使用面向对象，都要先将事物的属性和功能集中定义在对象中。

4.如何封装:3种:

1.只创建一个单独的对象:2种:

1.对象直接量:

var obj={//创建一个对象

属性名: 属性值,

... : ... ,

方法名(){...this.属性名...}, //ES6----->//方法名: function(){...},//旧

}

1.属性和方法，统称为对象的成员

2.对象的方法如何访问对象自己的属性:

1.错误:直接使用不加.的属性名

1.因为不加.的属性默认只能在作用域链中找

2.无法自动进入对象中查找属性

2.正确:this

1.什么是this:自动获得正在调用当前函数的.前的对象的关键词

2.为什么:不带this的普通变量，无法进入当前对象内查找属性

3.何时:只要对象的方法，想访问自己的属性，都必须加this.

3.如何访问对象的成员:

1.如何访问对象的属性:对象.属性名

其实属性就是保存在对象中的变量，用法和普通变量完全一样！

2.如何调用对象的方法:对象.方法名()

其实方法就是保存在对象中的函数，用法和普通函数完全一样！

2.用new:2步:

1. 先创建空对象: var obj=new Object();

2. 向空对象中添加新属性和方法:

obj.属性名=值;

obj.方法名=function(){

...this.属性名...

}

1.何时:创建对象时，暂时不知道对象中的成员

2.强调:js中的对象，可在任何时候，添加任何新属性和方法。

3.如何:只要尝试给不存在的属性和方法赋值时，对象会自动创建该属性。

2. 反复创建多个相同结构的对象:[辅助-->constructor文件夹图片]

3.构造函数

1.问题:用直接量创建，会产生大量重复的代码，不便于维护

解决:用构造函数:

2.构造函数:描述一类对象统一结构的函数

3.何时:只要反复创建多个相同结构的对象时，都要先用构造函数定义一类对象的统一结构。

4.如何:2步:

1.定义构造函数：

function 类型名(属性参数列表){

this.属性名=属性参数;

... = ... ;

this.方法名=function(){

... this.属性名 ...

}

}

2.用new调用构造函数创建新对象:

var obj=new 类型名(属性值列表);

1.new做4件事:

1.创建新的空对象

2.让新的空对象继承构造函数的原型对象

3.调用构造函数:

1.设置构造函数中的this->新的空对象new

2.通过强行赋值的方式，为新对象添加新属性和方法

4. 返回新对象的地址保存到变量中

2.总结:构造函数2个角色:

1.图纸:描述一类对象的统一结构

2.装修队:在空对象中添加固定的属性和方法

5.如何访问对象的成员:

1.成员=属性+方法

2.访问属性:对象.属性名 用法和普通的变量完全一样

属性其实就是保存在对象内的变量

3.调用方法:对象.方法名() 用法和普通的函数完全一样

方法其实就是保存在对象内的函数.读作"的"

其实，也可用对象["属性名"]方式访问

6.问题:方法定义在构造函数内。每创建一个新对象，都会重复创建相同的函数副本-->浪费内存！

7.总结:

构造函数:

优点:代码重用！

缺点:无法节约内存

解决:继承

5.js中对象的本质:

js中的对象其实就是关联数组。只不过比关联数组使用简单而已。

vs 关联数组:1.下标:关联数组:obj["成员名"] <=相当于> JS对象:obj.成员名

2.都可随时添加新属性

3.遍历:for in

2.\*\*\*\*\*继承:

1.什么是:父对象的成员，子对象无需重复创建就可直接使用

2.为什么:不但可代码重用，且还可节约内存

3.何时:如果多个子对象中拥有相同的属性值或方法定义，其实只需要将相同的属性和方法定义集中定义在父对象中一份即可。所有子对象公用！

4.如何:js中的继承都是继承原型对象

原型对象(prototype):专门集中保存同一类型的多个子对象，共有成员的父对象。

1.如何获得原型对象:[辅助记忆:母亲:构造函数 父亲:原形对象]

1.买一赠一:构造函数.prototype --> 所有子对象的父对象[即:原型对象(prototype)]

创建构造函数同时，已经自动创建了该类型的原型对象。构造函数的prototype属性引用着原型对象。原型对象的constructor也引用着构造函数对象。

ex:function Student(sname,sage){ //this->new

this.sname=sname;

this.sage=sage;

}

Student.prototype.intr=function(){

console.log("I'm "+this.sname+",I'm "+this.sage);

};

即：构造函数：Student

2.自动继承:创建子对象时，会自动设置新对象的

\_\_proto\_\_属性继承构造函数的原型对象

2.如何向原型对象中添加共有属性和方法:

构造函数.prototype.属性名=值;

构造函数.prototype.方法名=function(){...}

强调:原型对象中的方法，要想访问对象自己的属性，也必须加this.

3.总结:

每个子对象，值不同的属性，都要定义在构造函数中，所有子对象共有的相同方法和属性值，都要集中定义在原型对象中。

5.共有属性和自有属性:

1.自有属性:直接保存在当前对象本地的属性

2.共有属性:保存在原型对象(prototype)中，所有子对象共用的属性

3.相同:取值时，对象.属性

4.不同:修改时:

1.自有属性可直接通过子对象修改:

子对象.自有属性名=值

2.共有属性只能通过构造函数的原型对象修改:

构造函数.prototype.共有属性名=值

ex:文件3\_prototype.html中修改构造函数(Student)

Student.prototype.className="初二2班";

6.原型链:由多级父对象逐级\*继承\*形成的链式结构

1.任何对象都有 \_\_proto\_\_ 属性指向其父对象

2.保存着:所有对象的属性和方法

3.控制着对象成员的访问顺序:

优先在对象本地找自有属性 使用

自己没有才沿原型链向父级找

只要找到就不再向上找

4.原型链 VS 作用域链

作用域链:由多级作用域对象逐级\*引用\*形成的链式结构

1.保存着:所有的变量

2.控制着:变量的使用顺序:

优先在函数作用域对象AO中查找使用

局部没有，才沿作用域链向父级作用域查找

3.凡是不加.的变量，都在作用域链中找

4.凡是用.才能访问的属性，都在原型链中找

7.内置对象的原型对象:

1.其实每种内置对象也都有一对儿构造函数和原型对象

2.其中:

1.构造函数:负责创建新对象:

比如:

var arr=new Array();

var now=new Date();

var reg=new RegExp();

var str=new String();

...

特例:Math和window不是构造函数，不能new！

ex:function Array(){}

-->Array、Date、RegExp、String 属于构造函数，可以new

-->Array.prototype、Date.prototype、RegExp.prototype、String.prototype 属于原型对象

2.原型对象:负责将所有该类型的API都保存在构造函数的中

Array.prototype中:sort()

splice()

push()

...

Date.prototype中:getXXX()

setXXX()

toLocaleString()

...

3.用处:解决浏览器兼容性问题:旧浏览器无法使用新API

2步:

1.判断:当前浏览器的指定类型的原型中是否包含想要的API

方法一：if(!("indexOf" in Array.prototype))

[即:如果Array.prototype里面不包含"indexOf"时]

1.其中in用于检查左边的成员名"indexOf"是否在右边的对象中或对象的原型链上(Array.prototype)

2.[用indexOf在数组中查找置顶元素的位置，找到返回下标i，未找到返回-1]

方法二：if(typeof Array.prototype.indexOf!=="function")

[即:类型名为Array.prototype的里面找到的indexOf不是函数function]

-->[需要indexOf是函数function才能兼容]

2.如果没有就向原型中添加一个新函数

Array.prototype.indexOf=function(){

this //代表当前数组对象

}

4.问题:从父对象继承来的成员不一定都是想要的

解决:

3.多态:同一个函数在不同情况下，表现出不同的状态

1.重写(override):如果子对象觉得父对象的成员不好用，就可在子对象本地定义同名成员，覆盖父对象的成员。

2.为什么:从父对象继承来的成员不一定都是想要的

3.何时:如果子对象觉得父对象的成员不好用

4.如何:只要在子对象中定义和父对象中同名的成员即可

5.总结:面向对象三特点:

1.封装:将事物的属性和功能集中定义在一个对象中

为什么:便于维护

2.继承:父对象的成员，子对象无需重复创建，可直接使用

为什么:代码重用，节约内存--偷懒

3.多态:如果父对象的成员不好用，可在子对象中重写同名成员。

为什么:为了体现父子对象间的差异

6.自定义继承:

1.什么是:修改一个对象的父对象

2.何时:只要希望获取其它对象现成的成员时

3.如何:3种:

1.直接修改一个对象的\_\_proto\_\_属性指向新父对象

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

问题:\_\_proto\_\_是内部属性，不推荐使用

解决:Object.setPrototypeOf(child,father)

2.通过修改构造函数的原型对象来批量修改所有子对象的父对象:

构造函数.prototype=father

强调:时机:在开始创建子对象之前，就要换。

3.两种类型间的继承:更像Java的继承

1.何时:如果多个类型之间存在相同的属性结构和方法定义时，就可抽象出一个公共的父类型

2.为什么:代码重用和节约内存

3.问题:如果两种类型间拥有部分相同的属性结构和方法定义。

4.解决:抽象出一个公共的父类型:

1.父类型的构造函数中包含子类型相同的部分属性

2.父类型的原型对象中包含子类型相同的部分方法

3.在子类型构造函数中借用父类型构造

1.错误:直接调用父类型构造:

任何一个函数不用.不用new调用，其中的this默认指向window

ex:定时器的setInterval函数就没有用

.或者用new调用

2.解决:用call强行调用，并替换this为指定对象:

父类型构造.call(this,参数...)

3.call:可强行调用一个函数，并替换函数中的this

call的特点:

1.只执行一次

2.立刻执行

4.何时:如果函数中默认的this不是想要的

5.如何:fun.call(obj,参数值...)

调用fun，替换fun中的this为obj

4.让子类型原型继承父类型原型

Object.setPrototypeOf(

子类型.prototype, 父类型的prototype

)

## Day03

正课:

1.\*\*\*\*ES5

\*\*\*\*保护对象:保护属性，防篡改

Object.create()

\*\*\*\*call/apply/bind()

数组API

严格模式

2.ES6

模板字符串

let

箭头函数

for...of

\*\*\*\*class

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

ECMAScript:ECMA组织制定的JavaScript语言的国际标准，所有浏览器厂商遵照实现。规定了JS语言的核心语法。

1.\*\*\*\*保护对象:

1.什么是:

1.保护对象的属性和属性值始终有意义

2.防止篡改对象的结构

2.为什么:

1.JS中的对象的属性，默认可随意修改赋值

2.对象可随意添加删除属性

3.何时:严格来说，今后所有对象，都要有自我保护的抵抗力。

4.如何保护:

1.保护属性:保护对属性值的修改

对象属性分为:

1.命名属性:可直接用.访问到的属性

1.数据属性:直接存储属性值的属性

1.包括:四大特性:

1.value:实际存储属性值

2.writable:true/false

控制是否可修改属性值

3.enumerable:true/false

控制是否可被for in遍历，

强调:仅限制无法for in遍历到，依然可用.访问到。

4.configurable:true/false

控制着禁止删除该属性

控制禁止修改其它特性

2.说明:configurable一旦改为false，不可逆

所以，在修改其它特性是，通常都用configurable为false，作为双保险

3.获取四大特性:

Object.getOwnPropertyDescriptor(obj,"属性名")

4.设置四大特性:

Object.defineProperty(obj,"属性名",{

要修改的特性: 特性值,

... : ...

})

5.问题:一次只能修改一个属性

1.解决:用defineProperties代替defineProperty

Object.defineProperties(obj,{

属性名:{

要修改的特性

},

...:{ ... ... }

})

2.添加属性:其实defineProperty也可添加新属性:

只要defineProperty要修改的属性不存在，就会自动添加。

※强调:用defineProperty添加的新属性，四大特性默认值都为false

而用.添加的新属性，四大特性默认值都为true。

2.访问器属性:不直接存储属性值，仅提供对其它数据属性的保护

1.何时:只要使用自定义的规则保护属性值

2.为什么:数据属性的四大特性，保护规则是固定的，无法自定义。

3.如何定义:2步:

1.定义一个隐藏的数据属性:用于实际存储属性值

2.再定义一个访问器属性，来保护数据属性:

只能通过defineProperty或defineProperties[参考代码文件2\_getter\_setter.html]

不能通过直接量或.创建

Object.defineProperty(obj,"属性名",{

get(){ //在试图获取属性值时自动调用

return this.\_sage; //返回受保护的数据属性值

},

set(val){//在试图修改属性值时自动调用

//参数val会自动获得要修改的新值

//如果验证val符合规则

//才将val赋值给受保护的属性

//否则

//报错！

},

enumerable:true,(可允许遍历到)

configurable:false(禁止:删除)

})

4.如何使用:(访问其属性，不直接存值，即:无value值)

只要定义了访问器属性，就优先使用访问器属性，而不用受保护的数据属性.

访问器属性的用法和普通数据属性完全一样:

只不过:在取值时，自动调用get()

在赋值时，自动调用set(val)，val会自动获得要赋的新值

2.内部属性:无法通过.直接访问的属性

比如: \_\_proto\_\_

2.防篡改:保护对对象结构的修改:3个级别:

1.防扩展:禁止添加新属性

Object.preventExtensions(obj)

原理:每个obj内部都有一个隐藏属性:

Extensible，默认为true

preventExtensions将obj的Extensible改为false

2.密封:在防扩展基础上，进一步禁止删除现有属性[用的最多]

Object.seal(obj)

原理:修改obj的Extensible为false

将所有属性的configurable都改为false

3.冻结:在密封基础上禁止修改任何值

Object.freeze(obj)

原理:修改obj的Extensible为false

将所有属性的configurable都改为false

还将所有属性的writable都改为false

2.Object.create():

1.什么是:基于一个现有父对象，创建一个新的子对象继承当前父对象。并扩展新属性。

2.何时:今后如果没有构造函数，只有父对象的情况下，也想创建子对象

3.如何:var child=Object.create(father);

创建一个新的子对象child，继承father

其实还可为新对象扩展新的自有属性:

var child=Object.create(father,{

新属性:{四大特性},

... :

});

2.强调:只要添加到对象中的属性，四大特性默认为false，必须显式写为true。

3.原理:

1.创建一个空对象

2.让新对象自动继承father

3.为新对象扩展新的自有属性

3.\*\*\*\*call/apply/bind()

1.共同:为了替换函数中不想要的this

2.何时:只要函数中的this不是想要的

3.如何:

1.call/apply:

什么是:

1.\*调用\*一个函数

2.将函数中的this\*临时\*替换为指定的新对象

1.call:要求传入函数的参数必须单独传入

2.apply:要求传入函数的参数必须放在数组中集中传入

其中:apply可打散数组类型参数，单个传入

何时:只要调用函数时，函数中的this不是想要的，就用call/apply临时替换

2.bind:基于现有函数\*创建\*一个新函数，并\*永久\*绑定this

1.强调:被bind永久绑定的this，不能再用call/apply临时替换。

2.何时:创建一个新函数，并且希望永久绑定this

3.固定套路:替换回调函数中的this时，都用bind

4.回调函数--bind(this)

在call,apply,bind中绑定this，【记】在替换回调函数中的this一定要用bind绑定!

1.回调函数的要求:

1.不止执行一次

2.不会立刻执行

2.call的特点:(不用call的原因)

1.只执行一次

2.立刻执行

3.因为在回调函数中，默认的this是指的window，(任何一个函数不用.不用new调用，其中的this默认指向window)

ex:特别是在定时器回调函数中，setInterval函数就没有用.或者用new调用

4.数组API:

1.判断:

1.every:用于判断数组中所有元素是否都满足要求

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

val//获得当前元素值

i//获得当前元素位置

arr//获得当前数组对象

return 判断条件;

});

2.some:用于判断数组中是否包含满足条件的元素

arr.some(function(val,i,arr){

return 判断条件;

});

2.遍历API:

什么是:对数组中每个元素执行相同的操作

包括:

1.forEach:对原数组中每个元素执行相同操作

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

arr[i]=新值;

})——直接修改原数组

2.map:取出原数组中每个元素，执行相同操作后，再放入新数组返回。

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

return 新值;

})——不修改原数组，而是将新值放入新数组中返回

3.过滤和汇总:

1.过滤:选取原数组中符合条件的元素，组成新数组

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

return 判断条件

})——将原数组中符合判断条件的元素，复制到新数组中返回

2.汇总:将数组中每个元素，统计出一个最终结果

arr.reduce(function(prev,val,i,arr){

prev //截止到当前元素之前的临时汇总值

return prev+val;

}/\*,start\*/) //start表示汇总开始的基数

5.严格模式:

1.什么是:比普通js运行模式更严格的模式

2.为什么:解决普通js运行模式中固有的弊端和缺陷

3.何时:2种:

1.新项目，建议必须使用严格模式

2.旧项目，建议逐个函数/模块向严格模式迁移

4.如何:2种:

1.整个代码段启用严格模式:

在script元素或js文件的开头加入:"use strict";

2.仅对单个函数启用严格模式

在function内，函数体的顶部加入:"use strict";

5.要求:

1.不允许对未声明的变量赋值

2.静默失败升级为错误

3.不建议使用arguments和arguments.callee实现递归

### 2.ES6:

1.模板字符串:对字符串拼接的简化:

1.何时:只要字符串拼接时，需要动态执行表达式

2.如何:`反引号——ESC键的正下方

模板字符串必须用``包裹

``中支持:换行, 变量, 表达式

强调:每个变量和表达式都要用${...}包裹

2.let:创建一个仅在当前块内有效的变量

1.为什么:

1.js中没有块级作用域，导致块内的变量会污染外部。

2.js的声明提前会打乱程序的正常执行顺序

2.何时:今后强烈建议用let代替var

3.如何:

强调:let必须配套严格模式使用

1.解决声明提前: 检查let a之前不允许提前使用a

变相强制将所有变量的声明放在当前作用域的顶部

2.解决块级作用域:

let声明的变量，仅在当前{}内有效，出了块失效。

3.箭头函数:简化回调函数:

1.何时:今后，几乎所有的回调函数都可用箭头函数简化

2.如何:

1.所有回调函数都可:去function改=>

2.如果函数体只有一句话:可省略{}

如果这一句话还是return，可省略return

3.如果只有一个参数:可省略()

但是，如果没有参数，必须保留空()

3.更大用途:箭头函数内外共用同一个this-->取代bind

4.特殊:如果不希望内外共用this，就不能用箭头函数

比如事件处理函数:

elem.addEventListener("click",function(){this->elem})

elem.addEventListener("click",()=>{ this->不是elem})

变通解决:

elem.addEventListener("click",e=>{ e.target->elem})

4.for of:简化了for循环遍历:[示例]

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

sum+=arr[i] //当前元素

}

1.※可简写为:

for(var val of arr){

val //当前元素

}

2.局限:

1.只能遍历索引数组和类数组对象

不能遍历关联数组和对象:for in

2.无法获得房间号

3.只能逐个遍历所有，不能控制循环的开始和结束以及步调。

4.如果数组中保存的是原始类型的值，那么，修改val，不会影响数组元素

——按值传递:其实val只是数组元素值的副本。

3.何时:只要仅遍历元素值，不关心下标时，才可用for of简写。

5.\*\*\*\*class:对面向对象(OOP)的简化

1.定义类型:

1.用class结构包裹构造函数和原型对象方法:

2.构造函数名提升为类名(类型名提升到class之后，称为类名)

3.构造函数统一用constructor定义,其余保持不变

3.直接定义在class中的函数直接量会自动保存在当前类型的原型对象中

2.示例：

class Flyer{

constructor(fname,speed){

this.fname=fname;

this.speed=speed;

}

fly (){ //Flyer.prototype.fly

console.log(...);

}

}

6.继承:

1.在class 类名后加 extends 父类型名

2.在子类型构造函数中不允许直接用call调用父类型构造

3.正确:super(属性参数值列表)

super:

1.可自动指代父类型的构造函数

2.可自动将子类型正确的this传入父类型构造函数中

示例://让Plane继承Flyer函数

class Plane extends Flyer{

constructor(fname,speed,score){

super(fname,speed); //super()指代父类型构造，且自动替换this

this.score=score;

}

getScore(){...}

}

# Jquery

## Day01

正课:

1.什么是jQuery

2.如何使用jQuery

3.jquery 选择器---\*\*\*查询/查找

### 1.什么是jQuery:

jQuery是第三方的简化DOM操作的函数库

第三方:

即不是浏览器内置的，也不是开发人员自定义的。由第三方个人或组织定义的现成的API。必须下载后引入项目才能使用

1.何时:几乎所有大型的项目和流行的框架都是用jQuery简化DOM开发。

2.简化DOM操作

学jQuery还是在重复学习DOM,只不过API极简化而已。

3.为什么:2大目的:

1.DOM操作的终极简化:4个方面

DOM操作:查找，修改，增加，删除

事件绑定

动画

Ajax

2.解决了几乎所有浏览器兼容性问题:

只要 jQuery让用的，都没有兼容性问题

4.函数库:一组现成的API函数的集合

### 2.如何使用:

1.版本:

1.x

兼容旧浏览器 代码稍大

jquery-1.11.3.js

.js 未压缩版本，可读性好，便于学习

体积大，下载速度慢

仅适合学习阶段使用

.min.js 压缩版本，可读性差:去掉空字符、去掉注释;极简化变量名;

体积小，下载速度快，适合生产环境使用

2.x

不再兼容旧浏览器 代码较小

3.x

不再兼容旧浏览器 代码更小，功能更丰富

笔试:3.x新特性:10个

1.3.x所有函数默认运行在严格模式

2.支持ES6的新语法:for...of promise ...

2.加载:2种:

1.将js文件下载到网站本地，在页面中使用<script>相对路径引入

<script src="js/jquery-1.11.3.js"></script>

2.利用CDN网络:CDN:内容分发网络:

网络中包含多处服务器

每处服务器中都保存着资源的副本

客户端访问时，CDN可自动根据客户端到服务器的网络状况自动选择最优的服务器下载资源

<script src="CDN网络地址"></script>

3.jQuery的本质:

<script src="jquery-1.11.3.js"></script>

向window中添加了一种新的内置类型jQuery，包括构造函数和原型对象

1.构造函数jQuery，专门用于创建jQuery对象封装DOM对象

jQuery不用new调用，而是直接调用

可简写为$: window.jQuery=window.$=jQuery

2.原型对象jQuery.fn, 专门保存所有jQuery对象可用的简化版API

jQuery.fn = jQuery.prototype

4.创建jQuery类型的子对象（简称jq对象）:

1.为什么:普通的DOM元素，不是jQuery类型，无法使用jQuery类型的简化版API

2.要使用jQuery简化版API，必须先创建jq对象，封装找到的DOM对象，才能使用jQuery简化版API

3.如何创建jQuery对象:3种:

1.用$("选择器")查找DOM元素，并封装进jQuery对象中:

1.什么是jQuery对象:专门封装一个或多个DOM对象的类数组对象

2.何时:暂未获得任何元素，需要首次查找时

2.用$(DOM元素)封装一个已经得到的DOM元素为jQuery对象

1.何时:如果已经自动获得一个DOM对象，要用简化版API，都要用$封装为jq对象

2.比如:$(this) $(e.target) ...

3.比如:

$(...).html() 获取元素的innerHTML内容

$(...).html(xxx) 修改元素的innerHTML内容为xxx

3.用$(...)创建新的DOM元素，封装在jq对象中... ...

5.DOM操作的流程:

1.构建DOM树

2.查找触发事件的元素:$("选择器")

3.绑定事件:$(...).事件名(事件处理函数)

1.查找要修改的元素: $("选择器")

获得当前元素: $(e.target) $(this)

2.修改元素:

获取或修改元素的innerHTML:$(...).html() 一个函数两用

6.※※jQueryAPI三大特点:(重点)

之一:一个函数两用:(重点)

如果没给新值，默认就读取

如果给了新值，就变为修改

之二:自带遍历效果

之三:每个函数都返回正在使用的jQuery对象——链式操作

#注意一:

css方法添加属性

css("border-right","none"); ok

css("borderRight","none"); ok

#注意二:多次使用css方法

1:jquery支持链式写法

$("#btn1").css("","")

.css("","")

.css("","");

2:jquery{}

$("#btn1").css({

属性名:值,

属性名:值,

属性名:值...

});

### 3.jquery 选择器---\*\*\*查询/查找:2种:

准备：今后都是先找元素，再调用jqueryAPI操作元素；

只要用jquery找元素，都用选择器！

用jquery查找元素，返回直接就是jquery对象，无需再封装

如何:var $jq对象 = $("选择器");

#区别 jquery对象<-->dom对象

var $obj; //提示当前对象jquery对象

var obj;

1.按选择器查找:

jQuery支持一切css选择器,且不用考虑兼容性问题

jQuery扩展了少量专有的非CSS选择器:

非css选择器都是用程序模拟，效率比CSS选择器低

1.基本选择器:——同CSS

5个:#id .class elem \* 选择器1,选择器2,

2.层级选择器:——同CSS

4个:ancestor descendant

parent child ==>

parent > child ==>

prev + next ==>

prev ~ siblings ==>

练习:

1.jQuery事件绑定:

$("selector").事件名(function(){...});

2.访问元素的css样式

1.读取选中元素css属性

$("selector").css("css属性名");

2.修改选中元素css属性

$("selector").css("css属性名","值");

3.强调:jquery多数简化API自带forEach功能:

$("selector").css("css属性名","值");

相当于对找每个元素都css操作

$("selector").事件名(function(){....});

相当于对找到每个元素都绑定事件

3.过滤选择器

3.1子元素过滤选择器:——同CSS

:first-child

:last-child

:nth-child(i/even/odd/2n/2n+1)

:only-child

规律:

1.在每个父元素内分组编号

2.下标从1开始

3.2基本过滤(位置过滤):CSS没有

1.基本过滤:位置过滤，根据元素查找结果的位置来选择

2.何时:只要选择特定位置的元素时,都用基本过滤选择器

3.包含:

:first 选择查询结果中排在第一个位置的元素

不考滤元素在父元素位置，仅考在jquery对象中下标位置

:last 选择查找结果中排在最后位置一个元素

:even/odd 选择器查找结果排偶数/奇数位置元素，下标从0开始排序

:eq(i) 选择查询结果中的一个元素，i下标，下标从0开始排序

##jquery查找元素:规则深度优先

:gt(i) 选择查找结果中i位置之后的所有元素

:lt(i) 选择查找结果中i位置之前的所有元素

规律:

1.先取出符合条件的元素集中放在一个集合中，统一编号，和元素所在父元素内的位置无关。

2.下标从0开始

3.3\*\*\*属性过滤选择器:——同CSS,但多了一个!=

[属性名] 选择包含指定属性元素

例 div[id]

[属性名=值] 选择包含指定属性,并且属性值精确等于"值"的元素

例 [id=btn]

[属性名!=值] 选择包含指定属性,并且值不等于"值"的元素

模糊查找:

[属性名^=值] 选择包含指定属性,并且属性值以"值"开头的元素

例 [class^=b]

[属性名\*=值] 选择包含指定属性，并且属性值中包含"值"的元素

3.4内容过滤选择器:CSS没有

:contains(text) 查找内容中包含指定文本的元素

比如:span:contains('购物车')

:has(selector) 选择内容中包含符合选择器要求[父]元素

:emtpy 查找内容为空的元素

比如:span:emtpy

:parent 选择包含内容或子元素的父元素

比如:span:parent 其实就是:empty反义

3.5可见性过滤选择器:

:visible 匹配所有可见的元素

:hidden 匹配所有不可见的元素

只能选择:=>display:none和type=hidden的

隐藏元素:4种:

css:display:none;

opacity:0;

visibility:hidden;

html:type="hidden";

3.6表单过滤元素选择器:

:input 选择所有表单元素包括(input select textarea button)

:type类型:选择指定type类型表单元素

:text :password :radio :checkbox

:button :submit :reset

:file :image :hiddens

其实是:[type='xxx']简化写法

1.表单属性过滤选择器是:

:disabled 禁用

:enabled 可用

:checked 选中

:selected 选中(下拉菜单)

2.#小心

<input type="button"/>

<button></button>少用:在表单自带功能<==>submit

练习:

访问标准属性

获取:$("..").attr("标准属性名");

修改:$("..").attr("标准属性名",值);

不能访问三大状态属性

selected/checked/disabled

获取:$("..").prop("属性名");

修改:$("..").prop("属性名",值);

2.按节点间关系查找:[下节讲]

## Day02

正课:

1.修改

2.查找:(按节点间关系查找)

3.添加，删除，替换，克隆

4.事件:

学jQuery还是在学DOM！

### 1.修改:一个函数两用

1.内容:

Jquery DOM

$(...).html() .innerHTML

$(...).text() .textContent

$(...).val() .value

2.属性:

标准属性:

$(...).attr("属性名"[,值]) .属性名

状态属性:

$(...).prop("属性名"[,值]) .属性名 --prop 全称：property

自定义扩展属性data-\*:

$(...).data("属性名"[,值]) .dataset.属性名

3.样式:

1.读取或修改一个css属性:

$(...).css("css属性名"[,值]) 修改: .style.css属性

读取: .getComputedStyle()

2.同时修改多个css属性:

$(...).css({

css属性名:值,

... : ...

})

3.最高效的修改样式的办法:class

$(...).addClass("class1 class2")

$(...).removeClass("class")

$(...).hasClass("class")

固定套路: .toggleClass()

if( .hasClass())

.removeClass()

else

.addClass()

### 2.查找:(按节点间关系查找)

1. 父子:

$(...).parent() .parentNode

$(...).children([selector]) .children 直接子元素

$(...).children().first() .firstChild

$(...).children().last() .lastChild

$(...).find(selector) 查找所有后代元素中符合selector条件的子元素

2. 兄弟:

$(...).next() .nextElementSibling

$(...).nextAll() 之后所有兄弟

$(...).prev() .prevElementSibling

$(...).prevAll() 之前所有兄弟

$(...).siblings() 前后所有兄弟

### 3.添加，删除，替换，克隆

1.添加

1.DOM:3步:

.createElement()

设置关键属性

.appendChild()

2.jQuery:2步:

1.用$(html片段)创建新元素并设置关键属性

2.将新元素追加到父元素:

1. $(父).append($新元素/html片段)

$(新).appendTo(父)

2. $(父).prepend(...)

$(新).prependTo(父)

3. $(child).after(...)

$(child).before(...)

2.删除:

$(...).remove(); 删除当前选中元素

$(...).empty(); 清空所有子内容

3.替换:

$(旧).replaceWith(新)

$(新).replaceAll(旧)

4.克隆:

$(...).clone(false/true)

false(可省略) 仅克隆属性，不克隆行为——浅克隆

true 即克隆属性，右克隆行为——深克隆

### 4.事件处理:

笔试题:jQuery有几种事件绑定方式，差别

DOM:addEventListener/removeEventListener()

jQuery:

1. .bind/unbind() 用法和DOM完全一样

.unbind() 解除绑定 3种重载:

.unbind("事件名",fn) 移除事件上的fn处理函数

.unbind("事件名") 移除事件上所有函数

.unbind() 移除所有事件

——废弃了

2. .one() 仅触发一次事件绑定，触发后自动解绑

3. .live/die() ——废弃了 因为所有事件都集中直接绑在document根节点上

4. $(父).delegate/undelegate() 简化版利用冒泡

.delegate("selector","事件名",fn)

"selector" 筛选当前元素是否想要的

只有符合selector要求的子元素才能触发事件

fn中this->e.target

——已废弃

5.$(...).on/off()

2种用法:

1.代替bind:

$(...).on("事件名",fn)

off() 同 unbind()

2.代替delegate:

$(父).on("事件名","selector",fn)

6. 简化版:

.事件名(fn)

比如:.click .dblclick .mouseover ....

强调:仅对最常用的事件进行了简化

## Day03

### 1:jquery对象

如何获取jquery对象

工厂函数专门创建jquery对象

$("选择器");<--->

$("#menu1");

### 2:jquery选择器选择器-->查找元素方法

基本、层次

$("#menu"); id

$(".base"); class

$("ul>li");\*\*\*\*\*\*\* 子代

$("#menu li");\*\*\*\* 祖先后代

$("div,h1,span"); 组件

### 3:jquery方法

获取修改元素html内容

<li>hello</li>

$("li").html(); 获取

$("li").html("js");修改

获取修改元素属性

<form action="#" method="post">

$("form").attr("method");-->post

$("form").attr("method","get");

获取修改元素css

<div></div>

$("div").css("color","red");

$("div").addClass("base");

$("div").removeClass("base");

创建元素修改元素

<li id="btn1" >js</li>

$("#btn1").html("css");

### 4:事件

bind

on

<li id="btn1" >js</li>

$("#btn1").bind("事件名",fn);

$("父元素").on("事件名","子元素选择器",fn);

第一天代码:看-->写()

## Day04

正课:

1.事件处理:

页面加载后事件

鼠标事件

模拟触发

2.动画:

3.简化类数组对象操作

4.封装自定义插件

### 1.事件处理:

1.页面加载后事件:2个

1.window.onload=function(){...}

必须等待所有网页内容(html,css,js,图片)加载完成，才触发

2.DOMContentLoaded

只要html和js加载完成就可提前触发

不必等待css和图片加载完成

1.DOMContentLoaded比onload更早触发

2.总结:

1.绝大多数不依赖css的初始化操作和事件绑定都应该在DOMContentLoaded阶段提前执行

2.只有依赖于css的初始化操作，必须放在loaded中等待所有内容加载完成才能执行

3.如何在DOMContentLoaded阶段执行操作:

$(document).ready(()=>{

... ...

})

4.鄙视:window.onload vs $(document).ready()

1.window.onload必须等待所有内容(html,css,js,图片)加载完成才能触发——晚

$(document).ready()只要DOM内容(html和js)加载完成就提前触发——早

2.onload只能添加一个处理函数

$(document).ready() 可反复添加多个处理函数

5.更简化:$().ready(()=>{...})

更简化:$(()=>{...})

总结:直接放入$()中的回调函数自动在DOM内容加载后提前触发

更简单:放在body结尾的script中的代码，自动就是DOM加载后提前触发

2.鼠标事件:

mouseover mouseout

进出子元素，同样会触发父元素的进出事件

1.问题:进入子元素，同样也在父元素内，没必要反复触发进出事件

2.解决:mouseenter mouseleave

进出子元素，不再触发父元素的进出事件

3.如果同时绑定mouseenter和mouseleave:

$(...).hover(

e=>{...}, //mouseenter

e=>{...} //mouseleave

)

4.更简化:如果将进入和移出，统一为一个函数

$(...).hover(e=>{...})

### 3.模拟触发:

用程序代替用户的操作，触发其它元素的上的事件

2种:

1.可直接调用其它元素上的事件处理函数:

$(...).事件名()

2.用trigger()

$(...).trigger("事件名")

### 2.动画:2大类

1.简单动画:3种效果:

1.显示/隐藏:

$(...).show/hide/toggle()

1.如果没有参数，默认使用display:none/block瞬间显示隐藏

2.如果要加动画:参数:(动画持续时间)

可以为:slow/normal/fast/ 毫秒数

原理:

用定时器逐渐修改width/height/opacity

1.问题:

1.程序中定时器的效率比css中transition的效率要低

2.效果是写死的，无法维护

2.总结:今后优先选择css transition和boot中的效果实现动画。复杂动画才用程序中的定时器

2.上滑下滑:

$(...).slideUp/slideDown/slideToggle()

3.淡入淡出:

$(...).fadeIn/fadeOut/fadeToggle()

2.万能动画

1.动画函数的回调函数:专门在动画结束后自动执行

$(...).动画函数(时间,fn)

问题: 如果连续执行的函数特别多，会导致回调的层级很深——callback hell

解决:动画队列:

jQuery会为同一个元素上反复调用的多个动画函数创建执行队列。前一个函数执行完，后一个函数才能开始。

2.万能动画:可对任意css属性执行动画效果

$(...).animate({

css属性:目标值,

... : ... ,

},时间,callback)

1.仅支持纯数字值的属性:

width/height top/left margin/padding

opacity borderWidth fontSize ...

不支持:颜色, 字体, CSS3中的transform

2.排队与并发:

排队:对同一个元素反复调用的动画函数，自动排队执行。

并发:放在同一个animate内的要修改的多个css属性，同时改变。

3.停止动画:

transition:无法控制停止

1.解决:animate()+stop()

$(...).stop() 停止正在执行的动画

2.问题:无法清除队列中剩余的动画，剩余动画依然会执行

解决:

$(...).top(true) true:清空队列

3.延迟执行:$(...).delay(时间).XXX()

### 3.简化类数组对象操作

1.为什么:

1.jQuery对象本身就是类数组对象

2.类数组对象无法使用数组强大的API

2.所以，jQuery中提供了操作类数组对象的简化版API

1.包括:

$(...).size() $(...).length

$(...).get(i) $(...)[i]

$(...).each((i,elem=)>{...})

==>$(...).each(function(i,elem){...}) 类似于数组的forEach

2.可对jQuery对象中封装的每个DOM元素执行相同的操作

其中:i当前位置，elem 当前DOM元素

在jQuery3.0中可用for...of代替

$(...).index(elem) 类似于数组的indexOf

查找elem在整个jQuery结果集合中的下标位置。如果找不到，返回-1

### 4.封装自定义插件

1.何时:如果jQuery提供的函数不够用，就可添加自定义函数:

2.如何:

1.必须放在jQuery.fn原型对象中

jQuery.fn.自定义函数=function(){

this//指将来调用函数的.前的jQuery对象

}

2.判断是否引入了jQuery类型

if(typeof jQuery!=="function")

throw new Error(...)

# HTML5

## Day01

今天学习的内容

1:复习总结Nodejs

1)与前端JS区别

JS属于前端技术;NodeJS后端技术

JS数据类型:值类型+对象类型(ES)+NodeJS官方模块

2)NodeJS模块系统

(1)官方提供模块 querystring url fs http Buffer ..

(2)第三方模块 npm i mysql

(3)用户自定义模块

文件模块:

user.js module.exports={};

app.js var u = require("./user");

目录模块

node\_modules/oracle/package.json

{"main":"./index.js"}

node\_modules/oracle/index.js

module.exports={}

app.js var o = require("oracle ");

3)NodeJS核心

querystring.parse();

url.parse(str,true);

Buffer

fs.readFileSync(); //同步

fs.writeFileSync(); //同步

fs.appendFileSync(); //同步追加

fs.readFile();

fs.writeFile();

http.createServer();

4)第三方模块

mysql

1:加载模块 var mysql = require("mysql");

2:创建连接池 var pool = mysql.createPool({});

3:获取其中连接 创建SQL发送SQL

pool.getConnection((err,conn)=>{

conn.query("select insert ..",(err,result));

conn.release();

});

express

const http = require("http");

const express = require("express");

var app = express();

var server = http.createServer(app);

server.listen(8080);

#定义路由

app.get("/user",(req,res)=>{});

app.post("/user",(req,res)=>{});

2:作业

3:NodeJS中间件--重点&难点

Express中间件

Express是一个自身功能极简，完全是由路由和中间件构成的web开发框架:从本质上说，一个Express应用就是调用各种中间件.

中间件(Middleware)是一个***函数***.它可以访问请求对象(req request)，响应对象(res response)，中间件工作于请求和响应中间.

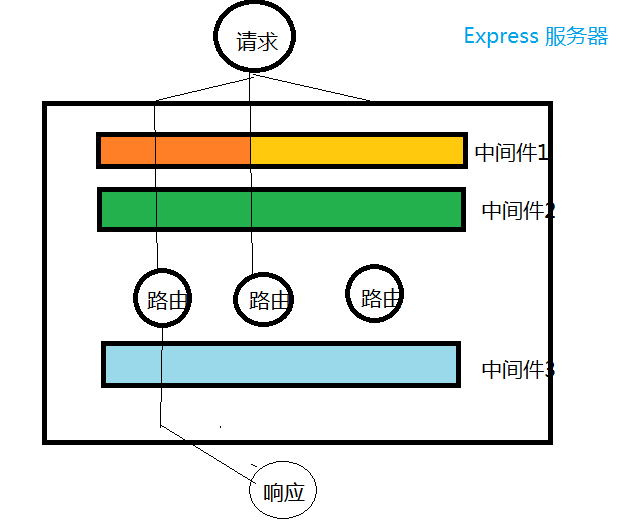
中间件的功能包括

(1)执行任何代码

(2)修改请求和响应对象

(3)终结请求-响应循环

(4)调用堆栈中的下一个中间件



标准语法

app.use(url,(req,res,next)=>{

//中间件要执行的代码

next();//调用下一个中间件或者路由

});

除了可以自定义中间件以外，Express 4.x官方提从一个中间件函数.

app.use(express.static("目录")); //public

若客户端请求了public目录下某个资源(html/css/jpg/js),它可以直接向客户端返回，不会再调用后续的路由.

客户端JS与服务器的NodeJS整合在一个项目中

(1)客户端请求静态HTML页面

(2)服务器返回客户端请求静态资源(express.static())

(3)客户端加载完成，AJAX异步请求动态数据

(4)服务器返回动态数据(一般数据JSON格式)

(5)客户端异步动态数据，解析出来，挂载到DOM树

4:HTML5新特性 --- 十个新特性:简单,凌乱

(1)新的语义标签

(2)增强表单(表单2.0)

(3)音频和视频

(4)Canvas 绘图 --重点&难点

(5)SVG 绘图 --重点&难点

(6)地理定位

(7)拖放API

(8)Web Worker

(9)Web Storage

(10)Web Socket

5:HTML5中表单的新特性

(1)新的input type <input type="?">

H4 input type :

text/password/radio/checkbox/file/hidden/submit/reset/image

H5 input type :(2014年10月发布)

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

(2)新表单元素

H4中表单元素:input/textarea/select/option/label

H5新表单元素:datalist/progress/meter/outer

(3)表单元素***新属性***

6:H5中新增表单元素--datalist

<datalist id="list3"> datalist本身不可见

<option>XX</option>

<option>YY</option>

</datalist>

<input type="text" list="list3" />

7: H5中新增表单元素-progress

显示一个进度条

<progress></progress> 左右晃动的进度条

<progress value="0.7"></progress>具有指定进度值进度

练习:使用定时器+进度条实现一个可以动态前进的

进度条，到100%时弹出"下载完成"提示

8: H5中新增表单元素-meter

Meter:度量衡，刻度尺，用于标示一个值所处范围：

不可收受(红色) 可以接受(黄色) 非常优秀(绿色)

<meter min="可取最小值" max="可取最大值"

low="合理的下限值" high="合理上限值"

optimum="最佳值" value="当前实际值"></meter>

练习:使用定时器+meter实现可以动态变化刻度尺，观察颜色.

9: H5中新增表单元素-output

output:输出,语义标签，没有任何外观样式，样式上等同span

商品单价 ￥3.5

购买数量<input type="number" value="1">

小计<output>￥3.5</output>

10: H5中新特性--表单元素新属性

H4:id/class/title/style/type/value/name/readonly/disabled

/checked

H5:中表单元素新属性

(1)placeholder :占位字符

<input value="tom" placeholder="请输入用户名" />

(2)autofocus 自动获取输入焦点

<input autofocus />

(3)multiple: 允许输入框中出现多个输入(用逗号分隔),

如:邮箱输入域

<input type="email" multiple/>

(4)form:用于输入域放置在Form外部

<form id="f5"></form>

<input form="f5"/>

========输入验证相关的新属性

(5)required: 必填项,内容不能为空

<input required>

(6)maxlength: 指定字符串最大长度

<input maxlength="9" />

(7)minlength: 指定字符串最小长度

<input minlength="3" />

(8)max:指定数字最大值

<input max="60" />

(9)min:指定数字最小值

<input min="3" />

(10)pattern 指定输入必须符合正则表达式

<input pattern="1[3579]\d{9}">

练习:

创建数据库bbs,包含留言表msg(mid,uname,content,pubtime)

使用NodeJS创建一个"留言板"应用,服务器可以接收如下请求

静态内容

/public/js/jquery-1.11.3.js

/public/msg\_add.html 显示"添加留言"的表单,包含

姓名-留言内容(AJAX)

/public/msg\_list.html 在Table中显示所有留言内容(AJAX)

动态内容

POST /msg 接收客户端异步提交uname/content 添加数据库

返回JSON {code:1,msg:"留言成功"}

GET /msg 向客户端返回所有留言JSON

## Day02

今天学习的内容

1:复习总结H5知识重点

(1)增强型表单

新input type:email/number/tel/url/search/range/color/

date/month/week

新的表单元素:datalist/option progress meter output

表单元素的新属性:

placeholder autofocus multiple form

required/minlength/maxlength/min/max/pattern

(3)音频视频

(4)Canvas绘图

(5)SVG绘图

(6)地理定位

(7)拖放API

(8)Web Worker

(9)Web Storage

(10)Web Socket

2:作业

3:今日目标

(1)视频和音频--好玩&简单

(2)Canvas绘图--难点&重点

1:Flash被H5取代体现在哪些方面

Flash绘图(AS/Flex) ==> Canvas/SVG

Flash动画 ==> 定时器/Canvas

Flash 视频和音频播放 ==> VIDEO/AUDIO

Flash 客户端存储 ==> WebStorage

2:H5新特性--视频播放

H5提供一个新的标签用于播放视频

<video src="res/birds.mp4"></video>

<video>

<source src="res/birds.mp4" />

<source src="res/birds.ogg" />

<source src="res/birds.webm" />

您的浏览器不支持VIDEO播放!

</video>

它本身是一个300\*150的inline-block元素

VIDEO标签/对象常用的成员:

**成员属性**:

autoplay:false 是否自动播放

controls:false 是否显示播放控件

loop:false 是否是循环播放

muted:false 是否是静音播放

poster:'' 在播放第一帧之前显示的海报

preload: 视频预加载策略，可取值

auto:预加载视频的元数据以及缓存一定时长

metadata:仅加载视频元数据

none:不预加载任何数据

##JS对象属性

currentTime: 当前播放的时长

duration: 总时长

paused:true 当前视频是否处理暂停状态

volume:1 当前音量

paybackRate: 播放速度

成员方法:

play(); 播放视频

pause(); 暂停播放

成员事件:

onplay 当视频开始播放时触发事件

onpause 当视频开始暂停时触发事件

练习:不使用Video自带controls,自定义播放/暂停按钮

鼠标移出视频区域，隐藏按钮，鼠标移入显示按钮

3:H5新特性--音频播放

H5提供了一个新的标签用于播放音频

<audio src="res/bg.mp3"></audio>

<audio>

<source src="res/bg.mp3" />

<source src="res/bg.wav" />

</audio>

它默认是一个300\*30的inline-block元素但若没有controls属性，AUDIO标签、对象常用成员

成员属性

autoplay:false 是否自动播放

controls:false 是否显示播放控件

loop:false 是否循环播放

muted:false 是否是静音

preload: 音频预加载策略

auto: 预加载音频元数据以及缓存一定时长

metadate: 仅预加载音频元数据

none: 不预加载作何数据

--JS属性

currentTime 当前播放时长

duration 总时长

paused:true 当前音频是否处理暂停状态

volume:1 当前音量 [0,1]

playbackRate : 播放速度 1 0.5 ..

成员方法

play(); 播放音频

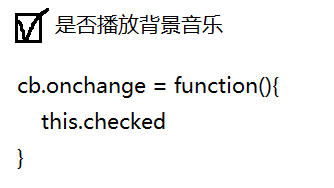
pause(); 暂停音频

成员事件

onplay 当音频开始播放时触发事件

onpause 当音频暂停播放时触发事件

练习:使用复选框控制网页背景音乐



15:30--15:40

挑战性要求:音量实现淡入和淡出volume

4:网页中可用的绘图技术

网页中实时走势图、统计图、在线画图板、网页游戏、地图应用都要使用绘图技术：

(1)SVG绘图:2D矢图绘图技术2000年出现，后纳入H5标准

(2)Canvas绘图:2D位图绘图技术，H5提出绘图技术

(3)WebGL绘图:3D绘图技术，尚未纳入H5标准

|  |
| --- |
| Canvas绘图难点所有  (1)坐标系  (2)单词比较多 |

5:H5新特性--Canvas绘图技术

Canvas:画布,是H5提供2D绘图技术

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

您的浏览器不支持canvas标签!

</canvas>

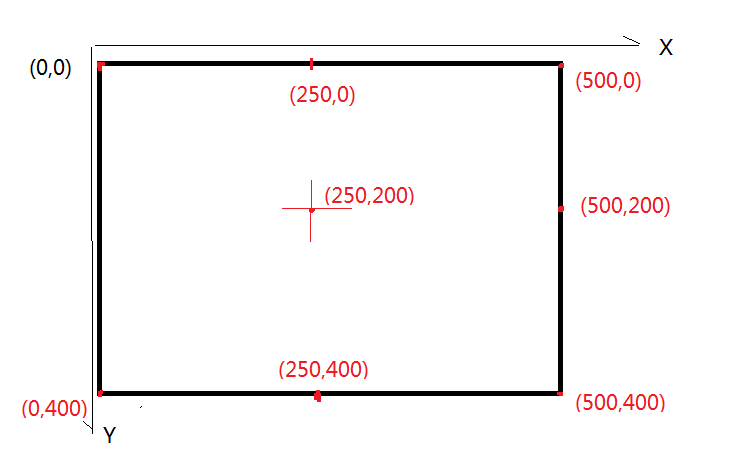
canvas标签在浏览器中默认300\*150的inline-block,

注意事项:

画布宽度和高度只能使用HTML/JS属性来赋值，不能使用CSS样式赋值!

每个画布上有且只有一个"画笔"对象--称为"绘图上下文"对象，--使用该对象进行绘图!

var ctx = canvas.getContext("2d"); //得到画布上的画笔对象



(1)使用canvas绘图矩形

矩形定位点在自己左上角

ctx.lineWidth = 1; 描边宽度(线宽)

ctx.fillStyle = "#ff0"; 填充样式/颜色

ctx.strokeStyle = "#000" 描边样式

ctx.fillRect(x,y,w,h); 填充一个矩形

ctx.strokeRect(x,y,w,h); 描边矩形

ctx.clearRect(x,y,w,h); 清除一个矩形范围所有绘图

练习1**:**在画布的右上角填充矩形 **100\*80**练习2**:**在画布的左下角填充矩形 **100\*80**练习3**:**在画布的右下角填充矩形 **100\*80**练习4**:**在画布的中央填充矩形 **100\*80**

在画布上描边一个可以左右移动的100\*80矩形

提示:使用定时器，先清除画布上己有内容,

再重新绘制一个(X不停增加)

在画布上描边一个可以上下移动的100\*80矩形(Y)

在画布上描边一个可以右下45度角移动的100\*80矩形(X,Y)

在画布上描边一个可以右下30度角移动的100\*80矩形(X,Y)

作业:

创建一个NodeJS服务器，可以接收客户端的如下请求

静态资源请求

/public/salesdate.html 向客户端输出一个只有空折页面的页面，加载完成后异步请求动态数据.

/public/jquery.js

动态资源请求

GET /salesdate 向客户端返回如下JSON数据;

[

{"label":"部门1","value":100},

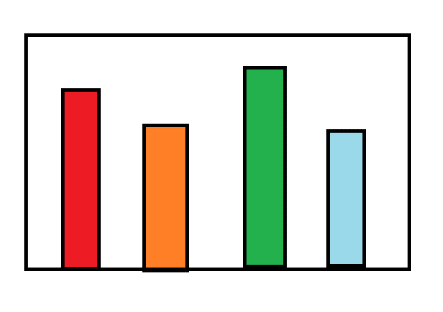
{"label":"部门2","value":200},

{"label":"部门3","value":150},

{"label":"部门4","value":190}

]

客户端根据这些数据，绘制如下图所示的统计图:



提示:为简化起见，可以把value值看做每个柱的高度.

## Day03

今天学习的内容

1:复习昨天的知识重点

HTML5新特性

(1)新的语义标签

(2)增强型表单

(3)视频、音频

<video src="x.mp4"></video>

<audio src="x.mp3"></audio>

(4)Canvas绘图 （位图/矢量图）

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

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

//绘制矩形 ctx.fillRect()/strokeRect()clearRect();

(5)SVG绘图

(6)地理定位

(7)拖放API

(8)Web Woker

(9)Web Storage

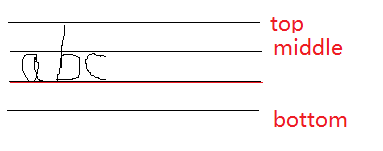
(10)Web Socket

2:作业

3:今天的目标

Canvas绘图

(1):使用Canvas绘制文本



一段文的定位点在文本基线的起点

ctx.textBaseline = "alphabetic"; **文本基线**

ctx.font = "12px sans-serif"; 文本大小和字体

ctx.fillText(str,x,y); 填充一段文本

ctx.strokeText(str,x,y); 描边一段文本

ctx.measureText(txt); 基于当前文字大小字体测量文本，返回一对象 {width:x}

练习一**:**

在画布左上角描边一段文本 **"tarena 2017"**

在画布左下角描边一段文本 **"tarena 2017"**

在画布右上角描边一段文本 **"tarena 2017"**

在画布右下角描边一段文本 **"tarena 2017"**

在画布正中央角描边一段文本 **"tarena 2017"**

(2)canvas绘图中使用渐变对象

线性渐变 linearGradient

径向渐变 redialGradient

var g = ctx.createLinearGradient(x1,y1,x2,y2);

g.addColorStop(offset,color);

....

g.addColorStop(offset,color);

ctx.strokeStyle = g;

ctx.fillStyle = g;

(3)使用canvas进行绘图--路径

Path:类似PS中的"钢笔工具",由多坐标点组成任意形状，路径不可见，可用于"描边","填充","裁剪"

ctx.beginPath(); 开始一条新路径

ctx.moveTo(x,y); 移动指定点

ctx.lineTo(x,y); 从当前到指定点画直线

ctx.arc(cx,cy,r,start,end); 绘制圆拱路径

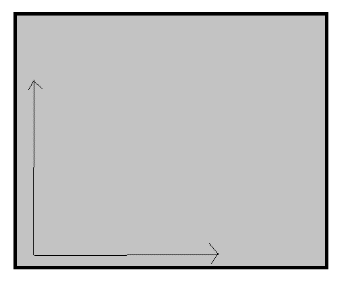
ctx.stroke(); 对当前路径描边

ctx.fill(); 对当前路径填充

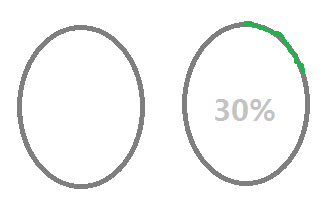
ctx.clip() 以当前路径裁剪

ctx.closePath() 闭合当前路径

练习:使用路径描边绘制一个坐标轴



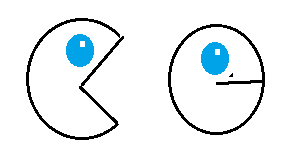
练习:使用圆拱形绘制可以前进的圆环进度条



练习:创建一个函数openMouth(),在画布上绘制左侧图形

练习:创建一个函数closeMouth(),在画布上绘制右侧图形

练习:使用定时器，每隔1s中交替调用openMouth()/closeMouth()



(4)使用canvas进行绘图---图像

canvas属于客户技术.图片在服务器中，所以浏览器必须先下载要绘制图片，且等待图片异步**加载完成**后再绘图.

var p3 = new Image();

p3.src = "img/p3.png"; //浏览器自动加载图片 下载

console.log(p3.width); //0

p3.onload = function(){ //图片加载完成

console.log(p3.width);

ctx.drawImage(p3,x,y); //原始图片大小

ctx.drawImage(p3,x,y,w,h);//拉伸绘图

}

练习**:**在画布的四个角各画一个原始大小飞机  
练习**:**在画布的中央画一个大小飞机400**\*200**练习**:**在画布的左上角画一个飞机 左右移动  
练习**:**在画布的左上角画一个飞机可以随鼠标移动而移动

|  |
| --- |
| Canvas绘图核心知识点--**重点**  **绘制矩形**  ctx.fillRect() ctx.strokeRect() ctx.clearRect()  **绘制文本**  ctx.fillText() ctx.strokeText() ctx.measureText().width  **绘制路径**  ctx.begingPath() ctx.closePath() ctx.moveTo() ctx.lineTo() ctx.arc()  ctx.stroke() ctx.fill() ctx.clip()  **绘制图像**  ctx.drawImage(); |

(5)第三方绘制统计图工具

a:chart.js 免费，提供了八种统计图表

b:FusionChars.js 收费，提供90+统计图表

c:ECharts. 百度提供免费的绘图工具，与地图整合很好

d:FreeChars..

提示:第三方工具使用无需记忆，重点掌握自学的过程

a:打开官网，查看说明

www.chatjs.org

b:仿照Demo编写示例

开源免费的8种图表，其于H5 Canvas

c:在实际项目中加以应用，查看详细的API说明

<canvas id="c15" width="500" height="400">

</canvas>

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

<script>

new Chart({

type:"bar",

data:{},

optioni:{}

});

</script>

作业:

(1) 自学 chart.js绘图统计图

(2) 使用canvas绘制一个随机改变的验证码图片

(3) 仿网易去音乐的播放界面

点击播放按钮，碟片开始旋转，背景音乐开始播放

再次播放按钮，碟片停止旋转，背景音乐停止播放



## Day04

今天学习的内容

1:复习昨天知识点(旋转/svg)

(1)新的语义标签

(2)增强表单

(3)视频和音频

(4)canvas绘图

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

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

//绘制矩形

ctx.fillRect();/ctx.strokeRect()/ctx.clearRect();

//绘制文本

ctx.fillText();/ctx.strokeText()/ctx.measureText().width

//绘制路径

ctx.beginPath()/ctx.closePath();

ctx.moveTo();/ctx.lineTo()/ctx.arc()/

ctx.stroke();ctx.fill();

//绘制图像

ctx.drawImage(p3,x,y,w,h);

(5)SVG绘图

(6)地理定位

(7)拖放API

(8)WebWorker

(9)WebStorage

(10)WebSocket

1.1:使用Canvas绘图时进行变形操作

Canvas绘图中也有变形的技术，可以针对某一个图像、图形的绘制过程进行变开

ctx.rotate(弧度); 旋转绘图片对象,**轴点是画布的原点** (0,0)

ctx.translate(x,y); 将整个画布的**原点**平移到指定点 (100,150)

ctx.save(); 保存画笔当前所有变形状态 (游戏中存盘)

ctx.restore(); 恢复画笔变形到最近一次保存 (游戏读取读数据从磁盘中);

练习:在画布左上角画一个绕自己为中心旋转飞机1  
练习:在同一个画布右上角画一个不旋转飞机2  
练习:在同一个画布左下角画一个旋转速度是飞机1速度2倍飞机3  
提示:每个飞机**"平移+旋转+绘制+逆向旋转+逆向平移"**  才不会影响后续的飞机绘制.

2:作业

补充:canvas如何按照特定的顺序绘制图片

cavnas绘图中若需要多张图片，他们的加载都是异步的，无法预测哪一张先加载完成！

但是绘图往往按照一定的顺序，如先绘背景，再绘上面的内容，所以必须等待所有图片全部加功完成，才能开始绘图.

var progress = 0;//所有图片的总加载进度

var imgBg = new Image();

imgBg.src = "img/bg.jpg";

imgBg.onload = function(){

progress += 25;

}

...

3:今天的目标SVG

可缩放的矢量图

|  |  |  |
| --- | --- | --- |
|  | canvas绘图 | SVG绘图 |
| 类型 | 2D位图 | 2D矢量图 |
| 如何绘图 | 使用JS代码绘图 | 使用标签图 |
| 事件绑定 | 每个图形不是元素，无法直接绑定事件. | 每个图形都是元素，可以直接绑定事件监听 |
| 应用场合 | 统计图/游戏 | 统计图/图标/地图 |

SVG技术诞生于2000年,早期作用XML的扩展出现，H5标准把**常用**SVG标签

采纳为标准，但有些被废弃.

使用SVG标签绘制矩形 <rect>

使用SVG 标签绘制圆形 <circle>

使用SVG 标签绘制椭圆 <ellipse>

使用SVG 标签绘制直线 <line>

使用SVG 标签绘制折线 <polyline>

使用SVG 标签绘制多边形 <polygon>

|  |
| --- |
| SVG绘图的特点---**着重注意**  (1)所有的图形默认只有填充色(黑色),没有描边  (2)SVG图形样式可以用元素属性来声明,也可以用CSS形式来声明,但用CSS声明时，只能使用SVG专用样式，不能使CSS样式，如:边框只能用 stroke,而不border  (3)图形可以使用JS来对属性赋值，但不能使用HTML DOM形式，只能用核心DOM操作如:  r0.x = 10; r1.width = 100;//无效  r0.setAttribute("x",10);  r1.setAttribute("width",100);  (4)动态添加SVG图形可以有两种方式  1)HTML字符串拼接  var html = "<rect></rect>";  svg.innerHTML = html;  2)使用  var 变量名 = document.createElementNS("http://www.w3.org/2000/svg","标签名"); |

练习:创建svg画布，绘制矩形高300宽30 加填充色  
 鼠标移入矩形透明度 1, 鼠标移出矩形透明度 0.3

练习:创建svg画布绘制两个柱子，初始高度者是0使用定时器不断修改它们的高

度，一个高度变为300停止一个变为150停止.

**提示**:用JS修改SVG图形属性只能使用核心的DOM方法

练习:使用AJAX从服务器端异步获取一段JSON数据,[{"label1":100},{"label2":120}]

根据这些数据创建矩形

4:使用SVG进行绘图--圆形

<circle r="" cx="" cy="" fill="" fill-opacity="" stroke="" stroke-opacity=""></circle>

练习:在SVG画布的左上角填充黑色圆形  
 在SVG画布的右上角填充蓝色圆形  
 在SVG画布的左下角填充黑色圆形  
 在SVG画布的右下角填充蓝色圆形  
 在SVG画布的中央描边(蓝色)+填充(黄色)一个圆形  
 练习:随机绘制30个实心圆形，大小随机，位置随机  
 填充颜色随机，透明度随机 点击某个圆形后，  
 它会慢慢变大/淡，直至消失从DOM删除

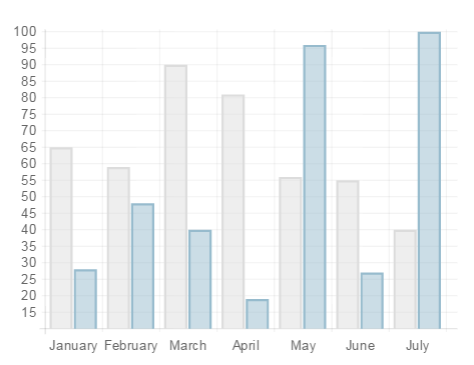
课程作业:

1)创建NodeJS web服务器,客户端先请求一个静态页面public/start.html,而加载后完成后发送AJAX请求，请求动态数据

动态数据: GET /start 该资源返回如下JSON数据

[{"label":"HTML",value:3},{"label":"js",value:9},...]

根据这段JSON字符串，绘制下图:



2)自学two.js工具的使用，理解其作用，仿写官方示例代码，学会使用方法，实现如下效果.

试着实现“月亮-地球-太阳”旋转的效果

## Day05

今天学习的内容

1:复习昨天知识重点

H5新特性

(1)新的语义标签

(2)增强表单

(3)视频、音频

(4)Canvas绘图

<canvas id="c1" width="" height="">

</canvas>

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

ctx.fillRect()/strokeRect()/clearRect() //绘图矩形

ctx.fillText()/strokeText()/measureText().width //绘制文本

//绘制路径

ctx.beginPath();/ctx.closePath();

ctx.moveTo();/ctx.lineTo()

ctx.arc();

ctx.stroke()/fill()

ctx.drawImage() //绘制图像

(5)SVG绘图

<svg id="" width="" height="">

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

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

</svg>

(6)地理定位

(7)拖放API

(8)Web Woker

(9)Web Storage

(10)Web Socket

2:作业

3:今天的目标

3.1： SVG

a:使用SVG进行绘图--椭圆

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

b:使用SVG进行绘图--直线

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

stroke-width="" stroke-linecap="round/butt/square"></line>

练习:使用SVG中直接绘制如下图形



|  |
| --- |
| 提示:若多个SVG图形有完全一属性，可以抽出来，放在一个公共的父元素中(小组)  <g stroke="#f00">  <line></line>  ...  </g> |

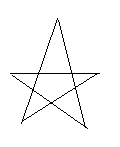
c:使用SVG进行绘图--折线

一条折线上可以有任意多个连续点.

<polyline points="50,50,100,300 ..." stroke="#000"

fill="transparent">

练习:使用折线绘制如下图标



d: 使用SVG进行绘图--多边形

<polygon points="50,50,.." fill="..."></polygon>

练习:使用多边形绘制如下两个图标



e: 使用SVG进行绘图--文本

SVG画布上不允许使用普通HTML元素绘制文件，SPAN/P等只能使用:

<text font-size="" fill="" stroke="" x="" y="" alignment-baseline="">文本内容</text>

f:使用SVG进行绘图--图像

不能使IMG设置于SVG画布!只能使用

<image width="" height="" x="" y="" xlink:href="p3.png"></image>

g:使用SVG进行绘图--渐变对象

定义特效对象:

1:渐变对象属于一种特效对象

<defs>

<linearGradient id="g3" x1="" y1= x2="" y2="">

<stop offset="0" stop-color="" stop-opacity="" />

....

<stop offset="1" stop-color="" stop-opacity="" />

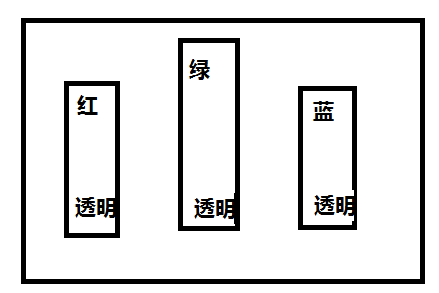
</ linearGradient>

</defs>

2:应用

<any fill="url(#g3)" stroke="url(#g3)"></any>

练习:使用渐变对象，为不同柱子分配不同渐变色



i:如何在SVG中使用滤镜

<defs>

<filter id="f2">

<feGaussianBlur stdDeviation="3" />

</filter>

</defs>

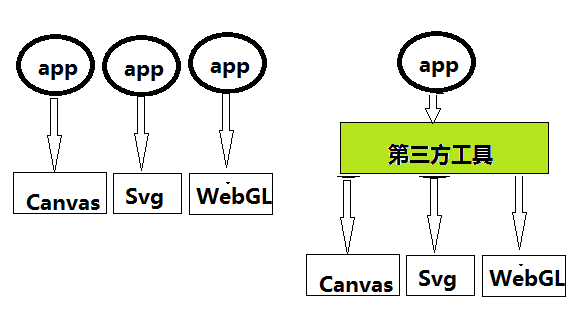
<any filter="url(#f2)">

网页中可用的绘图技术

(1)Canvas绘图

(2)SVG绘图

(3)WebGL绘图



第三方的绘图工具---Two.js

(1)打开官网，查看说明

https://two.js.org

是一个2D绘函函数库，可用于不同的技术下的绘图/如

SVG/CANVAS/WEBGL

(2)参考DEMO(演示案例集锦)编写示例

创建对象 var two = new Two({}).appendTo(body1)

绘制一个圆形 var c = two.makeCircle(200,200,100);

绘制一个矩形 var r = two.makeRectangle(200,200,50,50);

将绘制对象添加DOM树 two.update();

3.2:地理定位--了解&精确

Geolocation:地理定位，使用JS获取当前浏览器所在地理坐标(经度，纬度，海拔，速度)数据 ：如：饿了么？高德导航...

手机浏览器如何获取定位信息

(1)首选手机中的GPS芯片与卫星通信，定位精度在米.

(2)次选手机通信基站进行定位获取，定位精度公里

PC浏览器如何获得定位信息

通过IP地址进行反向解析，定位精度取决于IP地址库大小.

HTML5中提供一个新的对象，用于获取当前浏览器定位信息

window.navigator.geolocation{

getCurrentPosition:fn 获取当前定位信息

watchPosition:fn 监听定位数据改变

clearWatch:fn 取消监视

};

扩展小知识:在网页中如何嵌入百度地图

(1)注册百度开发者帐号

http://lbsyun.baidu.com/

(2)创建一个网站，申请一个地图AccessKey

[申请] facai.com ip地址限制 0.0.0.0/0

(3)示例:

百度地图坐标拾取器:

http://api.map.baidu.com/lbsapi/getpoint/index.html

http://lbsyun.baidu.com/index.php?title=jspopular

3.3:拖放API-- 掌握

Drag & Drop:拖动和释放

HTML5为拖放提了7个事件:分为两组

拖动的源对象(会动)可以触发事件

dragstart: 拖动开始

drag: 拖动中

dragend 拖动结束

拖动的目标对象(不动)可以触发事件

dragenter 拖动着进入

dragover 拖动着悬停在上方

dragleave 拖动着离开

drop 在上方释放

整个拖动过程1:dragenter\*1+dragover\*n+dragleave\*1

整个拖动过程2:dragenter\*1+drover\*n+drop\*1

注意:必须阻止 dragover的默认行为 dropleave才触发

**练习**:使用拖动源对象提供事件，实现"可以随鼠标拖动而移动小飞";

作业:

1:参考手册,使用百度地图API，在地图上用一张旗标注出你当前万寿路中心的地址.

2:英雄选择

要求:拖动某架飞机到目标区域，则在上方显示出该飞机;下方没有该飞机;问号飞机隐藏;拖动另一架飞机到目标区域，则显示出该飞机，之前选中的飞机重回下方列表区；若拖动选中飞机回下方列表区域，上方重新显示问号飞机(p0.png).

## Day06

今天学习的内容

1:复习昨天重点知识

HTML5新特性

(1)新的语义标签

(2)增强型表单

(3)视频和音频

(4)Canvas绘图

(5)SVG绘图

(6)地理定位

navigator.geolocation{}

(7)拖放API

拖动源对象 dragstart/drag{0,0}/dragend

拖动目标对象 dragenter/dragover{阻止默认行为}/dragleave/drop

(8)WebWorker

(9)WebStorage

(10)WebSocket

2:作业

3:今天的目标

HTML5新特性--WebWorker

程序:Progam 指可以CPU执行的代码，存储在外存中.

进程:Process/Task 指程序被OS调入内存， 分配执行空间，

随时供CPU调用执行.

线程:Thread :线程进程内执行代码基本单位

进程和线程:

(1)进程是操作系分配内存基本单位

(2)线程是CPU执行代码的基本单位

(3)线程必须处理某个进程内部

(4)一个进程内必须至少有一个线程也可以有多个

(5)一个操作系统可能同时存在几千个线程，它们是"并发执行的"--宏观看同时执行，微观上看依次执行.

Chrome浏览器中的线程模型

一个Chrome进程内，至少有6个线程，可以"并发"向WEB服务器发起HTTP请求，以获得所需的资源

---资源请求线程

还有**一**个线程负责将所有**内容绘制**到浏览器页面中--UI主线程--不允许多线程同时绘图，防止内容布局错乱.

<button>按钮1</button>  
<script src="3.js">很耗时的JS任务</script>  
<button>按钮2</button>

现象:JS执行过程中，按钮1可见，按钮2不可见

原因:浏览器中执行代码只有一个线程--UI主线程

解析办法:创建新的线程，由它来执行耗时JS任务;

UI主线程继续执行后续的HTML渲染.

<button>按钮1</button>  
<script>

var w = new Worker("3.js");

</script>

<button>按钮2</button>

3:Worker线程缺陷

浏览器不允许Worker线程操作任何的DOM、BOM对象

原因:浏览器只允许UI主线程操作DOM/BOM，若多个线程同时都可以DOM/BOM，页面混乱.

所以:类以JQUERY脚本不同使用Worker来加载执行.

Worker线程可以给UI主线程发数据消息

UI主线程

var w = new Worker("6.js");

w.onmessage = function(e){

e.data;//接收数据

}

Worker线程

postMessage(strinMsg);

UI主线程发数据消息Worker

UI主线程

var w6 = new Worker("6.js");

w6.postMessage(stringMsg);

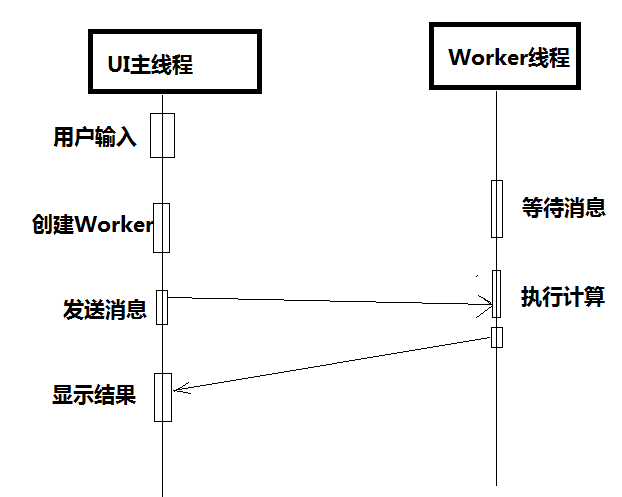
Worker线程

onmessage = function(e){e.data}

练习:在HTML中有一个INPUT,按钮"开始计算累加和",

点击按钮，创建一个Worker线程计算出用户输入数字

的累加和，下方一个div是计算结果.



|  |
| --- |
| 项目中Worker的使用场景  (1)只要js中有DOM&BOM就不能用Worker  (2)Worker适合于执行耗时的JS任务!如复杂计算,加密和解密，大数据统计，路径规则.... |

4:HTML5新特性---WebStorage

在浏览器中存储当前用户专有的数据，访问历史，内容定制，样式 定制...

在客户端存储数据可以使用技术

(1)Cookie技术: 浏览器兼容好，不能超过4KB,操作复杂

(2)Flash存储 依赖于Flash播放器

(3)H5 WebStorage 不能超过8MB，操作简单

(4)IndexDB 可存大量数据，还不是技术标准

|  |
| --- |
| Session:会话:浏览器从打开某个网站的一个页面开始，中间可能打开多个页面，直到关闭浏览器为止，整个过程称为"浏览器与web服务器的**一次会话**" |

WebStorage技术中，浏览器为用户提供了两个对象

(1)**sessionStorage 类数组对象，会话级数据存储;**

在浏览器进程所有分的内存中存储数据，可供此次会话中所有页面共同使用；浏览器一旦关闭了.

作用: 在同一个会话中所有页面共享数据:如登录用户名.

sessionStorage[key]=value //保存一个数据

sessionStorage.setItem(key,value); //保存一个数据

var v = sessionStorage[key] //读取一个数据

var v = sessionStorage.getItem(key);//读取一个数据

sessionStorage.removeItem(key); //删除一个数据

sessionStorage.clear(); //清除所有数据

sessionStorage.length; //数据数量

sessionStorage.key(i); //获取第i个key

练习:  
 创建index.html 右上角提示"请登录"超链接  
 创建login.html输入用户名，密码,提交按钮,用户点击  
 提交按钮提示 "登录成功" 3s后将自跳转回首页..  
 setTimeout+location.href  
 返回index.html 右上角提示"欢迎回来 xxx 退出登录"  
 创建logout.html,提示"您己退出登录",3s后自动跳转首页"  
 返回index.html,右上角提示"请登录" 超链接

**(2)localStorage:类数组对象，本地存储(跨会话存储)**  
在浏览器所有能管理外存(硬盘)中存储用户的浏览数据，  
可供此次会话以及后续的会话中的页面共同使用;即使  
浏览器关闭也不会消失--永久存在，  
作用:在当前客户端所对应的所有会话中共享数据.

localStorage[key]=value //保存一个数据

localStorage.setItem(key,value); //保存一个数据

var v = localStorage [key] //读取一个数据

var v = localStorage.getItem(key);//读取一个数据

localStorage.removeItem(key); //删除一个数据

localStorage.clear(); //清除所有数据

localStorage.length; //数据数量

localStorage.key(i); //获取第i个key

17:35--17:45

练习:创建index.html 有一个下拉菜单,"请选择您喜欢主题"

"蔚兰天空","芭比公主","黑暗主题", 各对应一个class名

.blue{background:#ddf,color:#33a}

.pink{background:#fdf,color:#a3a}

.dark{background:#333,color:#eee}

用户选中某个选项{select.onchange}为当前页面body使用指定的样式。

创建usercenter.html打开此页面即可应用index.html中的主题样式，

即使重启浏览器，直接访问返回index.html,也应该自动应用之前曾选的主题样式.

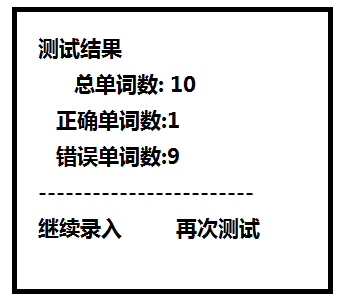
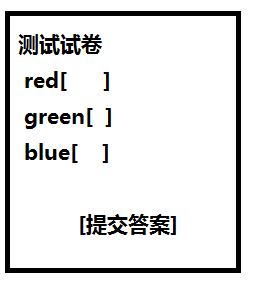
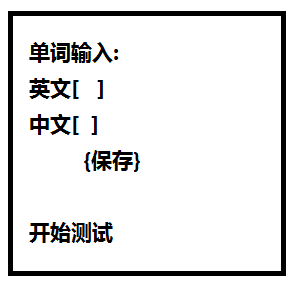
作业:单词测试系统

(1)用户可以在save.html中不停的录入新单词

(2)进入test.html开始测试，需要对之前录入过的所有单词测试

(3)提交答案后,在result.html中显示出测试成绩

提示:录入单词需要永就保存，而此次测试结果只需保存在当前会话中.



## Day07

今天学习的内容

1:作业

2:京东项目

模块一:用户登录

(1)数据库表:jd\_user(uid,uname,upwd)

(2)使用 NodeJS+Express创建web服务器/监听8080

(3)让web服务器向客户端提供如下静态资源

public/login.html 用户登录

public/register.html 用户注册

public/usercenter.html 用户中心

css/js/... 10:20--10:45

(4)让web服务器向客户提供如下动态资源

POST /login 执行用户登录验证

返回 {code:1,msg:"用户登录成功"}

返回 {code:2,msg:"用户名和密码有误"}

POST /register 执行用户注册

返回 {code:1,msg:"用户注册成功"}

(5)客户验证与AJAX请求

模块二:用户注册

数据库表:jd\_user(uid,uname,upwd)

模块三:用户中心

public/usercenter.html 用户中心

3:H5 Web Socket

4:H5 新特性

# AJAX

Day011:整体的课程  
 第一阶段 HTML(4)+CSS(5) = 静态网页  
 第二阶段 JS(7)+DOM(7.5) = 用户交互  
 第三阶段 AJAX(9)+H5(6.5)+NodeJS(7)= 动态页面(全栈)  
 第四阶段 框架  
2:AJAX内容安排  
 数据库服务器 --- SQL (背下)  
 WEB服务器 --- PHP(重点&难点)  
 HTTP协议  
 原生AJAX  
 JQUERY中AJAX  
 NodeJS  
3:服务器与数据库服务器  
 硬件服务器:高性能计算机  
 软件服务器:(web服务器/ftp/email/数据库)  
4:数据库服务器(软件分类)  
 网状数据库服务器  
 树型数据库服务器  
 关系型数据库服务器  
 MySQL/SqlServer/Oracle/DB2....  
5:关系型数据库数据层次  
 Server(服务器)-->Database(库)->Table(表)->Row(行)->Column(列)  
6:安装和使用mysql  
 1:下载mysql软件 www.oracle.com.cn 甲骨文  
 2:国内镜像 http://mirrors.sohu.com 搜狐  
 3:XMAPP 软件工具包(包括 mysql/php/apache)  
  
练习:进入mysql系统 mysql -uroot -p  
 退出 mysql系统 exit  
  
7:SQL语句---->重点&难点  
 Structured Query Language 结构化查询语言，专用于添加删除修改查询(CRUD)数据库服务器中的数据语言。由数据库客户端编写，提交给服务器来执行.  
 1:交互模式--一行执行一行  
 用户在客户端输入一行SQL命令，提交一行  
 2:脚本模式--一次性执行多行  
 把要执行的SQL命令编写在一个.sql文件中,一次性的全部提交给服务器执行  
 MariaDB [(none)]>source d:/a.sql  
  
8:SQL语句分类  
 1:DDL 数据定义语句(CREATE/DROP)  
2:DML 数据操作语句(INSERT/DELETE/UPDATE)  
3:DQL 数据查询语句(SELECT)  
4:DCL 数据控制语句  
  
9:DDL-->CREATE 创建库创建表  
#创建库标准语法  
 CREATE DATABASE 库名;  
 示例: CREATE DATABASE crm;  
 示例: CREATE DATABASE oa CHARSET=UTF8; //OK  
  
 #创建数据表(表)  
 #创建表标准语法  
 CREATE TABLE 表名(  
 列名1 列类型,  
 列名2 列类型,  
 ....  
 列名n 列类型 );  
 #列数据类型  
 int 整型 范围(-21亿~21亿) 年龄  
 varchar(n) 字符串 n几个字符(字母，数字，汉字,特殊字符)  
 double(n,m) 浮点数(小数) double(10,2)  
 总长度10位其中包括2位小数  
 datetime 日期时间  
  
10:DML数据操作语句(INSERT/DELETE/UPDATE)  
  
#INSERT 添加一条记录(添加一行数据)(增加一行数据)  
 标准语法:  
 INSERT INTO 表名 VALUES(值1,值2....);  
 注意事项  
 a:注意:顺序值1, 值2 和表列顺序相同  
 b:注意:字符串列类型或者日期类型数据加 ' 单引号  
  
 #DELETE 删除记录  
 标准语法  
 DELETE FROM 表名 WHERE 条件  
 注意事项  
 1:恢复很困难  
 2:一定要加条件，默认删除表中所有记录  
  
 #UPDATE 更新指令(更新数据)  
 标准语法  
 UPDATE 表名 SET 列1=新值1,列2=新值2 ..  
 WHERE 条件;  
 注意事项  
 1:一定要加条件，默认更新表中所有记录  
  
11:SELECT 查询表中记录  
 标准语法  
 SELECT 列名称1,列名称2,.... FROM 表名;  
 SELECT 列名称1,列名称2,... FROM 表名  
 WHERE 条件;  
 SELECT 列名称1,列名称2,... FROM 表名  
 ORDER BY 列名称;  
排序  
 SELECT \* FROM 表名; #\*所有列

## Day02

今天学习的内容

1:复习上周课程重点

什么是服务器

数据库服务器:作用保存数据

sql:结构化查询语言,专门用来操作数据库

DDL:CREATE/DROP

DML:INSERT/UPDATE/DELETE

INSERT INTO 表名 VALUES(值1,'值2');

UPDATE 表名 SET 列名=值 WHERE 条件;

DELETE FROM 表名 WHERE 条件;

DQL:SELECT

SELECT 列名1,列名2,...

FROM 表名

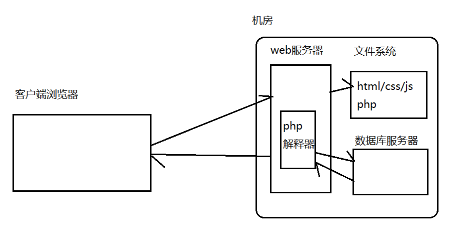
WHERE 条件

ORDER BY 排序

DCL

2:综合练习

3:web服务器/php-->重点&难点



作用:负责接收客户端请求，理解请求内容，找到请求资源，并且将资源返回客户端浏览器.

#web服务器种类

1)静态web服务器(apache)

提供的内容任何时间任何人访问都完全相同

属于静态web范畴的技术

HTML/CSS/JS/图片/视频/音频...

2)动态web服务器(apache)

提供的内容在不同的时间不同的人访问可能变化属于动态web范围技术;

1:JSP = HTML+JAVA

2:ASP.NET=HTML+C#

3:PHP =HTML+PHP(入门易)

4:NodeJS=HTML+NODEJS

##如何自学一门新的编程序语言

1)了解背景:历史，现状，发展，特点，***应用领域***

2)搭建环境:写出helloWorld

3)数据类型

4)变量，常量

5)运算符

6)逻辑结构

7)通用小程序

8)函数，对象

9)常用函数库，类库，中间件，框架

10)实用项目

#php背景知识

php是一种**运行**于**服务器端**编程语言，用于动态生成网页内容(优势：开发中小型web项目).

2)搭建php运行环境

#开发工具包(XAMPP) apache+php解释器

#注意:所有php程序只能保存 c:/xampp/htdocs/

#注意:程序后缀名 \*.php

#***注意***:运行 http://127.0.0.1/1.php

3)4)5)

#创建程序 \*.php

#声明变量 $age = 10;

#输出 echo $age;

#一个php程序是由html/css/js/php 代码混合组成

综合示例:

1:创建3.php 使用php向客户端输出 1个\*

2:创建4.php 使用php向客户端输出 50个\*

3:创建5.php 使用php向客户端输出 5行10列\*

4:创建6.php 使用php向客户端输出 九九乘法表

#数据类型

1:值类型/标量类型

string

bool/boolean int/integer float/double

2:复合类型

object php面向对象的程序

array 数组

3:特殊类型

null/NULL

resource 数据库连接,查询结果

#运算符

算术运算: + - \* / % ++ --

比较运算: < > == !=

逻辑运算:&& || !

字符串拼接运算符: .

#逻辑结构

选择:if if else switch case

循环:for while do..while foreach

foreach(数组 as $下标变量=>$值变量)

#php中预定义函数(mysql)--->重点&难点

php官网提供一组函数连接和操作mysql数据库

mysql\_xxx(); 旧版函数

mysqli\_xxx(); 新版函数

#***重点记忆***:php如何操作mysql数据库

1:创建到mysql数据库连接

$conn = mysqli\_connect(...);

2:向mysql数据库发送sql语句，并且等待mysql执行完成

$sql = "...INSERT/UPDATE/DELETE ";

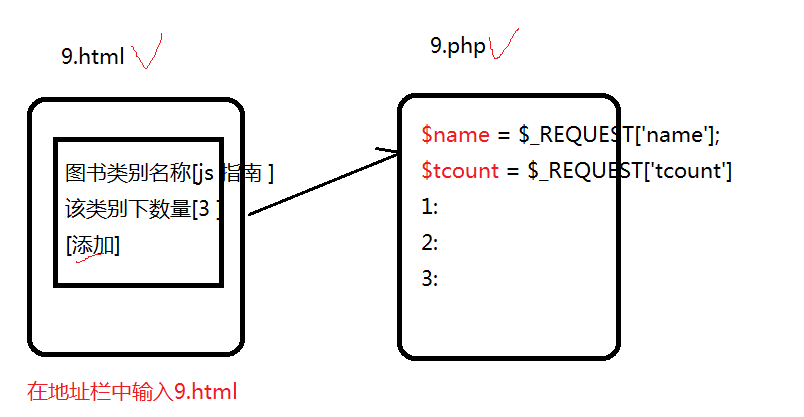
$result = mysqli\_query($conn,$sql);

3:读取mysql返回执行结果

if($result===true)执行成功

4:断开与mysql连接---(可以省略);

mysqli\_close($conn);



Day03  
今天学习的内容

1:复习总结昨天内容重点

php 运行在服务器程序设计语言

php 数据类型字符串 $str = 'tom'; $str = "jerry";

php 数据类型数组

count($arr) 返回数组长度

foreach($arr as $k=>$v){} 遍历数组元素

php mysql操作数据库 mysql

1:连接数据库$conn = mysqli\_connect('127.0.0.1','root','','dangdang');

2:创建sql 并且发送sql

$sql = "INSERT/UPDATE/DELETE ...";

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

3:if($result===true)

4:省略

2:作业 ifeng 10:30--10:35 运行作业

3:今天重点内容 php/mysql 查询(***重点***)与HTTP协议

#php mysql 查询

a:连接数据库

b:设置查询编码 utf8

c:创建一条查询的SQL，并且发送sql

d:获取返回结果

注意事项:

#返回结果$result不再是true/false

#$result结果集对象(查几条记录/查几列)

#如果需要获取最终数据结果，再一次操作-->抓取

e:抓取结果集中内容并转换数组

$row = mysqli\_fetch\_assoc($result); 抓取***一行***记录(转关联数组) 90%

$row = mysqli\_fetch\_row($result); 抓取***一行***记录(转索引数组)

f:抓取结果集中内容并转数组(抓取多行)

$rows = mysqli\_fetch\_all($result,MYSQLI\_ASSOC); 抓取多行记录

练习:查看所有商品记录、**删除**指定商品、更新指定商价格

1(js):为每一个商品添加 '删除'按钮

当用户点击'删除'按钮弹出'确认框'->(您是否要删除该商品)

是(自动执行***location.href***='product\_del.php?id=x') 否

2(php):product\_del\_02.php

获取商品编号并且删除商品,输出删除成功

#php 常用函数(了解)

die($str) 终止当前php程序的执行，并向客户端输出一行字符串,

终止原因.

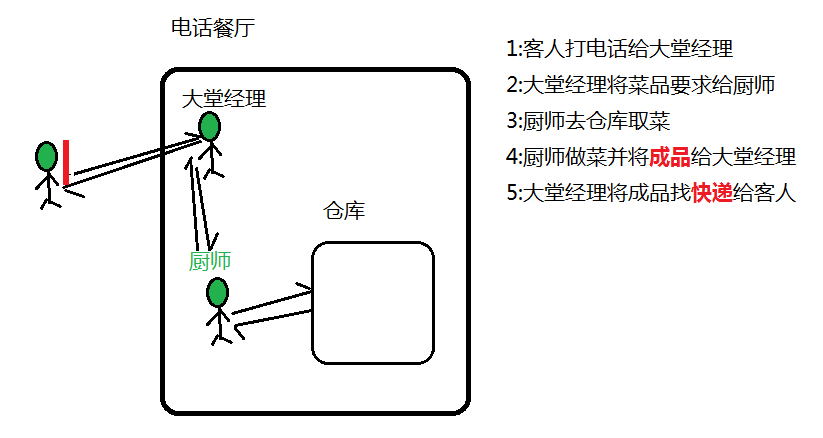
@ 压制住当前行代码的警告信息

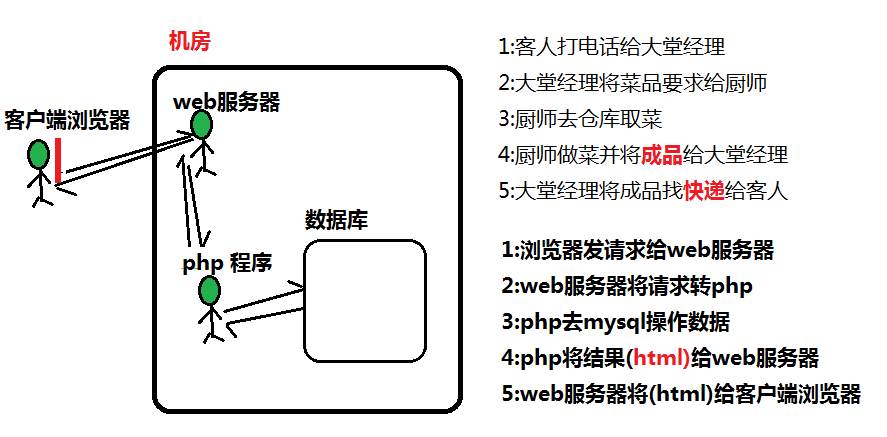
time() 返回当前系统时间，以秒为单位的整数

$id = mysqli\_insert\_id($conn) 返回连接上刚执行的insert产生的自增编号

$num = mysqli\_affected\_rows($conn); 返回连接上刚执行增删改语句影响行数

#php 运行原理(***重点理解***)





3http协议(知识点多，杂，乱，前后没关系)

学习http协议目标

1:调试AJAX程序中'看不见摸不着'的错误

2:进行web访问访问优化--高阶面阶题

3.1:URL

世界上任何一幢建筑物必须有一个"地址"才能被访问

互联网上的任何资源必须有一个"URL"才能被访问

**标准语法**

<scheme>://<user>:<pwd>@<host>:<port>/<path>;<params>?<query>#<frage>

scheme:方案,指定哪种协议从服务器上获取指定资源

常见方案:http/https/ftp/mailto/file/telnet等...

host:主机名,资源所在服务器ip地址或者域名(DNS转换IP)

http://127.0.0.1 {本地计算机}

port:端口号,每个程序在服务器运行都有一个端口号(整型数字)

ftp 21 文件上传下载

ssh 22 安全远程登录

telnet 23 远程登录

smtp 25 邮件传输

dns 53 域名解析baid.com-> 61.135.169.121

http 80 超文本传输协议(传送网页)

pop3 110 邮件接收协议

https 443 加密超文传输协议

今天学习的内容

1:复习昨天知识重点

php 操作 mysql数据库->查询

1:创建连接

2:设置编码 mysqli\_query($conn,"SET NAMES UTF8");

3:创建sql(SELECT)发送SQL语法

4:$result 不是true/false 结果集对象

5:抓取

一次抓取一行数据,如果没有数据返回NULL

$row = mysqli\_fetch\_row($result); 索引数组

$row = mysqli\_fetch\_assoc($result); 关联数组

一次抓取多行数据(关联数组)

$rows = mysqli\_fetch\_all($result,MYSQLI\_ASSOC);

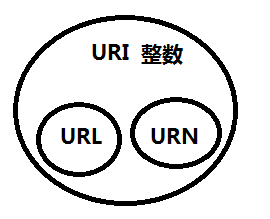
**2:http协议**

#分清一个概念URL/URN/URI

Unified Resource Locator:统一的资源定位符

Unified Resource Naming:统一的资源命名符

Unified Resource Identifier:统一的资源标识符



URI=URL+URN

URL: <a href="http://www.tmooc.cn/login.html" />

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

URN: <a href="mailto:admin@tom.com" />

<a href="tel:1399999999" />

#http协议

***作用***:用于传输网页的内容(html/css/js/img/mp3/avi....)

国际互联网工程任务组(IETF) 制定协议标准.

1991 http/0.9 在严重缺陷

1996 http/1.0

1999 http/1.1

#http协议如何工作(原理:请求响应式)

1:客户端浏览器发送请求给web服务器(请求)

2:web服务器接收请求处理后将结果发送给客户端浏览器(响应)

#http协议有哪个数据组成

***请求***[请求起始行、请求头信息、请求主体]

响应[响应起始行、响应头信息，响应主体]

***面试题***:http1.0/http1.1改进了哪些地方

1:支持虚拟主机技术，在一个web服务器上同时并存多个不同域名网站

Host:tmooc.cn

2:支持持久化连接技术

Connection:keep-alive (三次握手，四次挥手)

3:支持代理连接

Proxy:xxx

(增加功能:上传头像)

1：www.baidu.com http://cn.bing.com/

2: 博客园 csdn}

## Day04

今天学习的内容

1:复习昨天知识重点

php 操作 mysql数据库->查询

1:创建连接

2:设置编码 mysqli\_query($conn,"SET NAMES UTF8");

3:创建sql(SELECT)发送SQL语法

4:$result 不是true/false 结果集对象

5:抓取

一次抓取一行数据,如果没有数据返回NULL

$row = mysqli\_fetch\_row($result); 索引数组

$row = mysqli\_fetch\_assoc($result); 关联数组

一次抓取多行数据(关联数组)

$rows = mysqli\_fetch\_all($result,MYSQLI\_ASSOC);

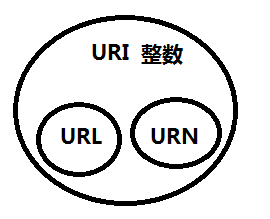
2:http协议

#分清一个概念URL/URN/URI

Unified Resource Locator:统一的资源定位符

Unified Resource Naming:统一的资源命名符

Unified Resource Identifier:统一的资源标识符



URI=URL+URN

URL: <a href="http://www.tmooc.cn/login.html" />

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

URN: <a href="mailto:admin@tom.com" />

<a href="tel:1399999999" />

#http协议

***作用***:用于传输网页的内容(html/css/js/img/mp3/avi....)

国际互联网工程任务组(IETF) 制定协议标准.

1991 http/0.9 在严重缺陷

1996 http/1.0

1999 http/1.1

#http协议如何工作(原理:请求响应式)

1:客户端浏览器发送请求给web服务器(请求)

2:web服务器接收请求处理后将结果发送给客户端浏览器(响应)

#http协议有哪个数据组成

***请求***[请求起始行、请求头信息、请求主体]

响应[响应起始行、响应头信息，响应主体]

***面试题***:http1.0/http1.1改进了哪些地方

1:支持虚拟主机技术，在一个web服务器上同时并存多个不同域名网站

Host:tmooc.cn

2:支持持久化连接技术

Connection:keep-alive (三次握手，四次挥手)

3:支持代理连接

Proxy:xxx

3:作业与综合练习--重点 14:40--15:00

综合练习(惠买车)

a:编写sql:创建数据库 huimaiche 创建数据表

汽车信息表:car(cid,cname,pic,price,pubtime,type)

注意事项:其中type范围 {'lt8' 'lt15' 'lt30' 'suv'}

用户信息表:user(uid,uname,upwd)

b:用户注册

编写php reg.php 接收客户端提交用户名和密码添加数据库

编写html reg.html 提交表单，录入用户名和密码提交给reg.php

(中文字母数字6~8)

c:用户登录

编写php login.php 接收客户端提交用户名和密码，查询用户名和密码是否正确.

SELECT \* FROM user WHERE **uname='tom' AND upwd='123'**;

编写html login.html 接收客户端提交用户名和密码，查询用户名和密码是否正确.(中文字母数字6~8)

d:添加汽车信息

编写php car\_add.php 接收客户端提交汽车名/汽车图片/汽车价格/汽车类型添加数据库

编写html car\_add.html 提交表单，录入汽车名/(汽车图片)/汽车价格/汽车类型提交car\_add.php

e:浏览器汽车信息列表

f:删除指定汽车

(增加功能:上传头像)

1：www.baidu.com http://cn.bing.com/

2: 博客园 csdn

Day05  
今天学习的内容

1:复习总结上周重点知识(php mysql/http)

php mysql

INSERT/UPDATE/DELETE

$conn = mysqli\_connect("127.0.0.1","root","","库名");

mysqli\_query($conn,"SET NAMES UTF8");

$sql = "INSERT ... UPDATE ...DELETE";

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

if($result===true)

SELECT

连接数据库/设置编码/创建sql "SELECT"/发送sql

$result 结果集对象(包含查询几条记录,几列)

$row = mysqli\_fetch\_row($result); 转换索引数组

$row = mysqli\_fetch\_assoc($result); 转换关联数组

if($row===null) 结束

$rows = mysqli\_fetch\_all($result,MYSQLI\_ASSOC);

http请求消息

2:作业(其中:上传图片功能)

3:http协议的响应消息

3.1:面试题:GET请求与POST请求区别

|  |  |  |
| --- | --- | --- |
|  | GET | POST |
| 语义 | 客户端想获取服务器(程序)上指定资源 | 客户端想传递数据给服务器端(程序) |
| 如何发起 | 浏览器地址栏输入URL、标签href/src,js跳转,表单GET，AJAX-GET(6) | 表单POST提交,AJAX-POST |
| 安全级别 | 不安全 | 不安全 |
| 表单提交 | 请求URL地址中 | 请求数据主体 |
| 数据长度 | 总长度(1KB) 字母数字1000个(请求url地址中) | 没限制(请求主体) |
| 是否需要编码 | 需要编码,一个汉字9个字节 | 自动编码 |

4:Ajax-->重点&难点

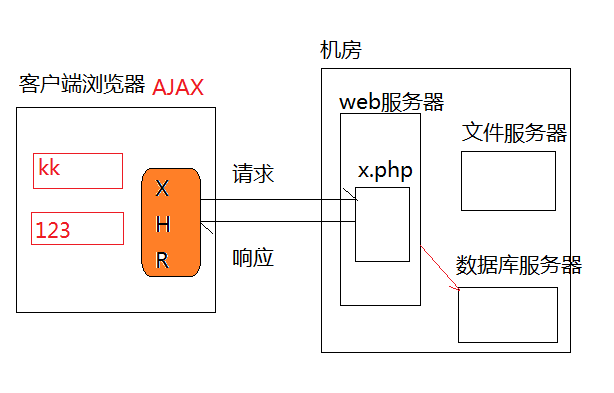
4.1:快速入门 ajax

添加用户示例 15:25---15:35

a:创建表:在数据库创建用户表 huimaiche user

b:创建php: add\_user.php (获取参数/创建sql/返回添加成功/失败)

c:创建 html:add\_user.html



4.2标准语法:XHR发送请求并且接收返回数据(**重点**)

(1)创建XHR对象

var xhr = new XMLHttpRequest();

(2)为XHR绑定事件，状态改变事件

xhr.onreadystatechange = function(){

alert(xhr.responseText);//接收服务器程序返回数据

}

(3)创建一个与服务器程序连接

xhr.open(请求方法,请求地址,是否为异步请求)

(4)发送请求

xhr.send(null);

4.3:概述

2002年左右，由Google在搜索引擎，提出GoogleSuggest 自动提示，提出AJAX概念(Asynchronous JavaScript And XML)

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

作用:无刷新，无提交，无跳转情况下网页内容局部更新.

应用场合:聊天室，股票走势图，搜索建议...

示例一:验证用户名是否存在(注册其中一个功能)

a:创建php existsuname.php (获取参数一个 uname;查询;'该用户名己存在',或者'该用户名可以使用')

b:创建html userreg.html (用户名失去焦点自动触发ajax)

作业:

1:编写sql:sohu.sql 库sohu 表user(uid,uname,upwd)

插入三行记录

2:编写 login.php 接收客户端提交的uname和upwd，验证用户名密码是否正确，返回"login-succ"或者"login-err"

3:编写HTML,输入用户名和密码后，点击"登录"按钮,通过AJAX 发送AJAX请求，若服务器返回"login-succ" 显示"登录成功，欢迎回来",若服务器返回"login-err" 显示"登录失败"

## Day06

今天学习的内容  
1:复习昨天重点  
 HTTP协议:作用(传送网页中数据 html/css/js/img/mp3..)  
 请求消息  
[请求方式:GET/POST/PUT/HEAD/DELETE/CONNECT/TRACE/OPTIONS]  
 响应消息  
 [响应状态码] 100 200 300 400 500  
[响应主体类型]text/plain;text/html;  
 GET/POST区别  
 AJAX 概述与工作流程  
 AJAX(Asynchronous JavaScript And XML)异步javascript and xml  
 核心对象:XHR 作用发送请求并且接收服务器返回数据  
 AJAX开发流程背  
 (1)创建对象 XHR var xhr = new XMLHttpRequest();  
(2)绑定事件 xhr.onreadystatechange = function(){  
xhr.responseText;  
}  
(3)创建服务器连接 xhr.open('GET','x.php',true);  
(4)发送请求 xhr.send(null);  
3:ajax 常用对象属性及处理不同数据格式 (JSON-重点&难点)  
 3.1 同步异步(工作方式)  
 \*同步请求:前面的请求方式均属于同步(5 GET 1 POST)  
 \*异步请求:AJAX  
 同步工作方式:(打电话)  
 客户端 服务器  
 填写表单 等待  
 点击提交发送(等待) 操作  
 接收数据 等待  
 异步工作方式:(微信)  
 填写表单 操作  
 点击提交发送(操作) 操作  
 接收数据 操作  
 3.2 AJAX常见对象、属性、方法、事件  
 #对象 XMLHttpRequest (作用:发送请求并且接收服务器返回数据)  
 注意事项:兼容性  
 IE 678 ActiveXObject("Microsoft.XMLHTTP")  
 IE 9/c/f/ XMLHttpRequest  
 if(window.XMLHttpRequest){  
 var xhr = new XMLHttpRequest();  
}else{  
 var xhr = new ActiveXObject("Microsoft.XMLHTTP");  
}  
 #属性:  
 readyState 表示当前XHR对象状态,不能赋值，可取值  
 其值会随着请求-响应过程进程变化.  
 0 UNSENT 请求消息尚未发送  
 1 OPENED XHR己经创建与WEB服务器连  
 2 HEADER\_RECEVICED XHR己经接收到响应消息头部  
 3 LOADING XHR正在收到响应消息主体  
 4 DONE XHR 接收完成响应消息主体  
 status 服务器响应状态码 200  
 responseText 保存响应消息主体(文本)  
 responseXML 保存响应消息主体(XML)  
 #方法:  
 open(method,url,isAsync);创建(打开)一个服务器连接  
 isAsync:发送请求方是异步(true)同步(false)  
 send();发送请求主体  
如果请求方式GET 参数 null  
如果请求方式POST 参数 自己拼接  
 #事件:当XHR状态改变时会触发事件  
 onreadystatechange()  
  
3.3 AJAX 接收服务器返回不同格式的数据  
 text/plain 纯文本--了解  
 text/html 文本与网页--了解  
 application/javascript JS代码--了解  
 application/json json格式数据--熟练掌握  
 application/xml xml 格式数据--了解

Day07

今天学习的内容  
1:复习昨天重点  
 AJAX开发流程(面试题)  
 (1)创建一个XHR var xhr = new XMLHttpRequest();  
(2)绑定事件 xhr.onreadystatechange = function(){}  
 事件触发时机: 当对象XHR状态改变触发  
 0-1 1-2 2-3 3-4 4次  
 2.1:二次判断 if(xhr.readState===4&&xhr.status===200)  
 2.2:xhr.responseText  
(3)打开连接 xhr.open('GET','x.php',true);  
 true:是否是异步请求  
(4)发送请求 xhr.send(null);  
 AJAX对象、属性、方法、事件  
 XMLHttpRequest  
 readyState  
 status  
 responseText  
 responseXML  
 open()  
 send();  
 onreadystatechange()

3:原生:AJAX 处理不同数据格式(JSON/JAVASCRIPT/XML)  
 JSON：是字符串数据格式，用于描述数据  
 JSON(JavaScript Object Notation)是一种轻量级的数据交换格式，易于阅读和编写，语法基于JS，但目前己经被各种语言支持.成为一种“异构系统交互数据的标准”  
  
 JSON: 语法要求  
 (1)一个JSON字符串有且只有一个根  
(2)JSON中可以表示数字，BOOL,NULL,字符串 (字符串必须是双引号)  
(3)数组中可以包含多个值，使用逗号分隔  
(4)对象中可以包含多个键和值，使用逗号分隔，键和值之间用冒号分隔，键必须用双引号.  
  
PHP-AJAX-JSON开发流程  
(1)服务器php:将数组转换JSON字符串发送  
 服务端: array--->json--->输出  
 $arr = ...;//手工创建数组，或者从数据中查询数组  
 $inputJson = json\_encode($arr);  
 echo $inputJson;  
(2)客户端js:将JSON字符串转JS对象  
 客户端:json字符串-->JS对象->拼接创建DOM  
 var obj = JSON.parse(xhr.responseText);  
  
4:JQuery中的AJAX  
 $('选择器').load(URL,[data],fn);  
 $.get(URL,[data],fn);  
 $.post(URL,[data],fn);  
 $.getJSON(URL,[data],fn);  
 $.getScript(URL,[data],fn);  
 $.ajax(URL,[data],fn); 万能AJAX  
  
 标准语法:  
 $.ajax({  
 type:'GET', //请求方式 POST/DELETE/PUB/HEAD  
 url:'x.php', //请求程序地址(必须)  
 data:'k1=v1&k2=v2' //请求提交服务器数据  
 beforeSend:fn, //请求消息发送之前回调方法  
 success:fn, //响应完成并且响应成功回调方法 200  
 error:fn, //响应完成并且出错回调方法 404 500..  
 complete:fn //响应完成后回调(无论成功失败)  
});  
 #注意事项:请求参数  
 二种写法  
 拼接字符串参数:data:'id='+id; //参数数量少可以使用  
 对象方式参数: data:{cname:"tom",age:10}

## Day08

今天学习的内容

1:复习昨天知识重点

AJAX处理JSON 格式的数据

JSON--> 原生AJAX

php 处理 json

header("Content-Type:application/json;charset=utf-8");

$input = json\_encode($rows);echo $input;

js 处理 json var obj = JSON.parse(xhr.responseText)

JQUERY中AJAX方法

$.ajax({

type:"GET",

url:"x.php",

data:{ename:"tom",age:19},

success:function(data){ //200 OK 自动JSON.parse();

},

error:function(){//404 500

}

});

2:jquery中ajax其它方法与跨域访问

**jquery中ajax其它方法**

$.get(url,data,fn); 发送get请求

fn:回调函数{发送请求并且接收返回数据后调用方法}

$.post(url,data,fn); 发送post请求

$("选择器").load(url,data,fn); \*\*\*

$.getScript(url,data,fn); 发送get请求，接收响应数据格式为script,会自动执行 eval(xhr.responseText)

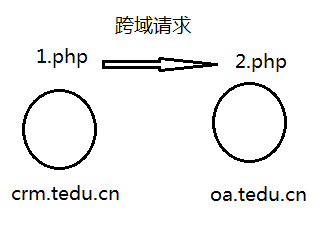
$.getJSON(url,data,fn); 发送get请求，接收响应数据格式为json,会自动执行JSON.parse();

$("选择器").load(url); 发送get请求，接收响应数据格式必须为HTML,会动将html保存选择器的标签中间.

2.2跨域访问

Cross Domain Request

从一个域名下的某个资源访问另一个域名下的某个资源



(1):两个域名不同 crm.tedu.cn--->oa.tedu.cn

(2):协议名不同 http://tedu.cn/1.php--> https://tedu.cn/2.php

(3):主机名/IP不同 http://localhost/1.php-->http://127.0.0.1/2.php

(4):端口号不同

http://127.0.0.1:8080/1.php->http://127.0.0.1:80/2.php

浏览器允许跨域请求情形

img/link/script/iframe 等标签可以

浏览器不允许跨域请求情形

浏览器默认禁止XHR跨域请求-跨域访问可能导致外来的数据对当前程序影响，造成安全风险.

解决方案

JSONP JSON with Padding 填充式JSON，是一种使JSON数据的方法，用于解决浏览器的XHR跨域请求限制.

JSONP 的思路

发起异步请求不使用XHR，使用一个动态创建script标签代替XHR，<script async src="跨域访问x.php"></script>要求服务器端必须回返回一段JS片段，包含了服务器数据

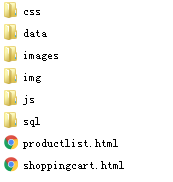
doResponse({ename:"tom"});

客户端特意提供了一个名为doResponse(data)函数;

06\_jsonp.html http://localhost/06\_jsonp.html

06\_jsonp.php http://127.0.0.1/06\_jsonp.php

3:京东首页 登录 产品列表



## Day09

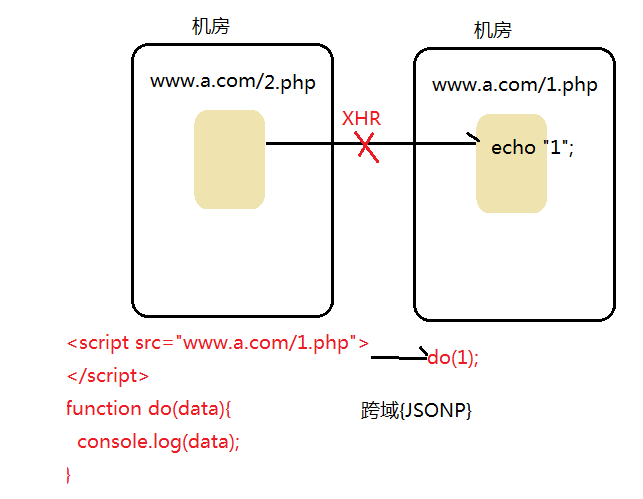
今天学习的内容

复习昨天知识重点

1:跨域

解决方案

JSONP JSON with Padding 填充式JSON，是一种使JSON数据的方法，用于解决浏览器的XHR跨域请求限制.



2:XML数据类型格式(text/html;script;json;...)

XML:可扩展的标记语言，所有标签都是自定义的，用于描述一段数据.

xml语法要求

(1)xml文档类型声明 <?xml version="1.0" encoding="utf-8"?>

(2)只能一个根元素 <books></books>

(3)有开始必须的结束

(4)可以嵌套不能交叉 <a><b><c></c></b></a> ok

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

(5)任意属性，属性必须用'或者"括起来

<book id="1001"></book>

服务器端程序创建 php

header("Content-type:application/xml;charset=utf-8");

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

echo "<book>";

echo "<name>js大全</name>";

echo "</book>";

前面js接收

var doc = xhr.responseXML; //原生AJAX

doc.querySelector("name");

新的内容京东项目

# NodeJS

Day01  
2:cookie  
 cookie:小甜饼  
 作用:保存(指定保存时间)用户添加数据.默认情况会话结束数据就删除了.  
 会话:是一个操作过程:当用户打开浏览器请求指定页面(会话开始),下一步操作,关闭浏览器(会话结束).  
 操作方式  
 (1)保存数据至cookie document.cookie='uid=10';  
 (2)从cookie中获取数据 var rs = document.cookie;  
 #注意事项:cookie数据保存在客户端浏览器，不要将安全性高数据保存cookie  
 #cookie保存数据通常是:用户编号/用户喜欢产品编号/昵称..  
  
  
3:NodeJS  
 1:阿里面试题:用户在浏览器中输入 www.taobao.com 直到看面之间发生了什么?  
 (1)操作系统(window)访问网络上DNS服务器，把或域名转换IP址地  
 DNS [www.taobao.com 60.28.242.250]  
 (2)浏览器发起HTTP请求  
 (3)web服务器接收并解析请求消息,查找指定资源，可能访问数据库，构建并返HTTP响应消息.  
 (4)浏览器接收并解析响应消息  
 (5)浏览器(缓存)接收到解析内容，并解析和渲染响应内容.  
  
 2:静态网页和动态网页  
 静态网页:网页内容任何人在任何时间访问都是不变的.  
 HTML/CSS/JS/Flash/音频/视频..  
 动态网页:网页内容任何人在任何时间访问可能是不同的.  
 JSP/PHP/NodeJS...  
 JSP =HTML+JAVA 功能强大可靠，适合大型企业级项目  
 PHP =HTML+PHP 简单易用，适合互联网项目  
 ASP =HTML+C# 易用，适合windows平台  
 NodeJS =HTML+"JS" 性能好，适合服务器IO密集(微博)型项目，不适合CPU密集(天气预报)型项目.  
  
 3:Nodejs概述  
 NodeJS不是JS，就一个软件开发平台(安装服务器端),它的竞争对手，PHP/JSP/ASP ! 历史上第一次有一种语言可以通吃前后端.  
 官网:www.nodejs.org  
 版本: 0.12(2016年初) 4.x(2016年中) 6.x(2016年底)  
 LTS: Long Term Support 长期稳定支持版本  
 Current:最新版本  
  
 4:nodejs两种运行模式  
 (1)交互模式--一般用于临时测试  
 REPL:输入一行代码执行一行  
 注意:交互模式自带输出功能，不必定 console.log(age)  
 node 回车 进入交互模式  
 .exit 退出交互模式  
 (2)脚本模式--正式项目中使用方式  
 把要执行所有语句编写在一个文本文件中(后缀名任意,没有都行)  
 一次性提交node解释器执行.  
 node 完整路径名/x.js 回车  
 提示:只要安装node.js 重启 webstorm, ws可以自动发现nodejs解析器程序。记得新建项目一开视频修改文件编码UTF-8.  
 练习:创建3.js文件，打印九九乘法表,以脚本模式运行.  
  
 5:面试题-如何掌握一门新语言  
 如果自学一门新语言--NodeJS  
 (1)了解背景 --百度百科  
 (2)搭建开环境，编写HelloWorld  
 (3)数据类型--重点  
前端JS的数据类型  
1:基本/原生/值类型/原始..  
 number/string/boolean/null/undefined  
2:引用/对象类型  
 ES对象:  
String/Number/Boolean/Math/Date/RegExp/Object/Function.  
 BOM:window/document/screen/location/navigator/event  
 DOM:Node/Element/Att...  
3:用户自定义对象  
后端NodeJS数据类型  
1: 基本/原生/值类型/原始..  
 number/string/boolean/null/undefine  
2:引用  
 ES 对象/Math/RegExp/Date/Object/Function/Boolean/...  
3:NodeJS官方与第三方对象 https://www.npmjs.com/  
4:用户自定义对象  
  
 (4) 变量和常量  
 var age = 20;  
 const PI = 3.14;  
 (5)运算符  
 算术运算符  
 比较运算符  
 逻辑运算符   
 位运算符  
 三目运算符  
 赋值运算符  
 特殊运算符  
 (6)逻辑结构  
 循环结构:while do..while for(;;) for(..in..) for(..of..)  
 练习:创建4.js 声明一个保存5个学生成绩数组，使用三种for循环获取每个学生成绩.  
 选择结构 if..else switch...case  
 练习:创建一个变量 var path = "/index";使用二种选择结构，判断path的值为哪种(/index、/search、/login)调用不同的执行函数.  
 (7)通用小程序 九九乘法表 100以内的质数 数组排序..  
 (8)函数和对象  
 (9)常用的组件、第三方工具、框架  
 (10)实际小项目  
  
Modal: 模态框  
Model: 模型  
Module:模块  
  
 6:模块(NodeJS中特有概念)--重点&难点  
 一个web项目功能可以分为很多个“模块”：如商品管理模块，支付模块，促销模块，商家管理....  
 NodeJS按照不同的功能，把函数，对象分别保存在不同文件，目录称为"Module".  
 快速入门:不同文件模块之间调用...  
 NodeJS中每个模块都是一个独立构造函数，解释器会为每个模块添加如下代码.  
 (function (exports, require, module, \_\_filename, \_\_dirname) {  
size(); //自己编写内容  
});  
 exports:{} 用于声明向外部导出自己的成员  
 require:fn 用于导入其它的模块，创建指定模块对象 new  
 module: 代表当前模块  
 \_\_filename 当前文件绝对路径;  
 \_\_dirname 当前文件所属目录绝对路径;  
  
NodeJS的模块中exports和module.exports对象区别是什么?  
二者都可以用于向外界导出自己的内部成员  
但是  
module指代当前模块对象，直正导出的是moulde.exports  
NodeJS底层有代码 exports = module.exports  
所以:若只是给exports对象添加新的成员, 则等价于给module.exports添加新成员，但是若修改了exports的指向，则不公产生实质作用.  
练习: 创建14\_app.js  
每个模块都可以使用自己的require()函数引入加一个模块--底层本质就是创建指定模块的一个对象实例.  
require("./模块文件名");  
每个模块可以使用exports对象向外导出/公开一些自己内部的成员供其它的模块使用.  
 exports.成员名=值  
  
NodeJS 模块的分类  
(1)nodejs官方提供的模块--安装在解释器内部  
 var 模块名 = require("模块名");  
(2)第三方编写的模块  
(3)用户自定义的模块  
 exports.x= y + require("./模块文件名");  
  
NodeJS预定义--Global  
 该模块提供的可以直接使用,而无需 require("global");  
 exports 用于向外部导出当前模块的内部成员  
 module 用于指代当前模块  
 \_\_filename 返回当前模块的绝对路径  
 \_\_dirname 返回当前模块所属目录绝对路径  
 console 控制对象,注意该以象与chorme中console  
 setInterval(time,fn);  
 setTimeout(time,fn);  
 setImmediate(fn); 等价于 setTimeout(0,fn);

Day02  
1:复习昨天知识重点  
 NodeJS:开发平台  
 适用范围:适合IO密集型(微博，微信)项目，不适合CPU密集(天气预报)项目.  
 前端JS和NodeJS区别  
 前端JS:运行在客户端浏览器中,存在兼容性问题，数据类型:值类+引用类型(ES+BOM+DOM+自定义)  
 NodeJS:运行于服务器(V8引擎),不存在兼容性，数据类型:值类+引用类型(ES+第三方对象+自定义)  
 Node模块系统  
 Nodejs中每个.JS文件都是一个"Module",每个模块都可以引入其它模块,也可以导出自己的成员(变量/函数/对象)供它的模块使用.  
 (function (exports, require, module, \_\_filename, \_\_dirname) {  
//自己编写内容  
 var age = 20;  
 var fn = function(){}  
  
 exports.age = age;  
 module.exports.fn = fn;  
 //源代码 exports = module.exports;  
 //##注意事项  
 module.exports = {age:10,f:fn}; ///OK  
 exports = {age:10,f:fn}; //ERROR  
});  
 exports与module.exports在使用上最大区别  
 (1)module.exports可以向外导出对象或者是成员  
 (2)exports只能向外导出成员

**Node模块的类型**  
 (1)官方提供原生模块 global/util/url/fs/http/....  
 (2)第三方模块 mysql/oracle/exports...  
 (3)自定义模块 文件模块和目录模块  
3:新的内容  
(1)npm和目录模块管理(有点乱)  
 (2)常用原生模块---重点&难点  
  
 3.1:自定义模块两种形式  
 (1)文件模块  
 创建一个js文件如 m3.js 导出需要公开数据，其它模块可以require("m3") 模块  
 (2)目录模块  
 方式1:创建一个目录,假设名为m4,其中创建名(必须)为index.js文件，导出公开的数据，主模块可以 require("./m4");  
 方式2:创建一个目录,假设名为m5,其中创建名package.json文件  
,其中声明main属性指定默认执行的js文件,如 5.js,其中导出需要公开的数据。其它模块可以 require("./m5");  
 方式3:创建一个目录 必须名为node\_modules,在此目录中再创建目录模块，假设m6,在m6目录再创建文件,必须名为package.json文件,其中声明main属性指定默认执行的启动文件名,如:6.js，其中导出公开数据。其它模块可以可以 require("m6")  
  
 2:NPM 包管理器  
 Node Package Manager :NodeJS第三方模块管理器，下载，更新，删除维护包依赖关系的工具.  
 npm 默认到 www.npmjs.org 网站下载所需第三方模块  
 标准语法:  
 (1)下载第三方软件包 npm install 包名  
 (2)删除指定软件包 npm uninstall 包名  
  
 3:NodeJS官方提供原生模块--querystring  
 querystring模块用于处理 HTTP请求URL中查询字符串  
 var obj = qs.parse(str); 把查询字符串解析js对象  
 var str = qs.stringify(ob); 把js对象转换为查询字符串  
  
 4:NodeJS官方提供原生模块-url  
 url模块用于解析http请求地址,获取其中各个部分  
 var obj = url.parse(str); 把一个url字符串解析为一个对象  
 var obj = url.parse(str,true); 把一个 url字符串解析为一个对象,并把其中查询字符串也解析为对象.

5:NodeJS官司方提供原生模块-Buffer  
 Buffer:缓冲区,本质是一个内存区域，用于暂存以后要用到数据(可能数字，字符串，图片、音频、视频...)  
  
(1)创建缓存区并且分配指定大小  
 var buf1 = Buffer.alloc(1024); //字节b  
(2)创建缓存区数字数组  
 var buff2 = Buffer.from([1,4,5,6]);  
(3)创建缓存区字符串  
 var buff3 = Buffer.from("abcde...");  
(4)把缓存中数据转字符串  
 var str = buff3.toString();  
  
6:NodeJS官方提供原生模块--fs----->重点  
 fs模块提供对文件系统的文件/目录进行增删改查，读写的功能.  
(1)同步读取文件中的内容  
 var data = fs.readFileSync(file);  
(2)同步向文件写入内容(删除己有内容)  
 fs.writeFileSync(file,str/buff);  
(3)同步向文件中追加写的内容(不删除己有内容)  
 fs.appendFileSync(file,str/buff);  
 (4)异步读取文件的内容  
 fs.readFile(file,function(err,data){})  
(5)异步向文件中写入内容  
 fs.writeFile(file,str/buf,function(err){});  
  
7:NodeJS官方提供原生模块--http----->重点  
 HTTP模块可用于编写基于HTTP协议客户端程序(即浏览器);也可以编写基于HTTP协议服务器(web服务器)  
 用http模块编写一个web服务器  
 (1)接收客户端http请求消息  
(2)解析客户端请求消息  
(3)读取客户端请求文件  
(4)向客户端发送HTTP响应消息，主体就是客户端请求文件  
 var server = http.createServer();  
 server.listen(8080);  
 server.on("request",function(req,res){  
 req 请求对象  
 res 响应对象});

## Day03

1:复习昨天知识重点  
NodeJS数据类型  
(1):基本值类型(string/number/boolean/null/undefined)  
(2):引用类型(ES原生类型/NodeJS/用户自定义类型)  
NodeJS模块(Module)系统  
(1)NodeJS官方提供模块系统  
(2)第三方模块---npm  
(3)自定义模块  
 文件模块:circle.js exports.size = size;  
 app.js require("./circle");  
 目录模块:  
 a:m4/index.js require("./m4");  
b:m5/package.json "main":"1.js" 1.js  
c:node\_modules/m3/package.json "main":"./index.js"  
 index.js  
NodeJS官方模块  
(1)jquerystring  
 var obj = qs.parse("name=tom&age=10");  
(2)url  
 const url = require("url");  
 var obj = url.parse("http://www.jd.com:8080/ad/?name=tom");  
 var obj = url.parse("http://www.jd.com:8080/ad/?name=tom",true);  
(3)Buffer  
 var buf1 = Buffer.alloc(1024);  
 var buf2 = Buffer.from("abc");  
(4)fs  
 const fs = require("fs");  
 //同步IO操作  
var data = fs.readFileSync(file);  
 fs.writeFileSync(file,str/buff);  
 fs.appendFileSync(file,str);  
 //导步IO操作  
 fs.readFile(file,function(err,data){  
});  
fs.writeFile(file,str,function(err){});  
(5)http  
 var http = require("http");  
 var server = http.createServer();  
 server.listen(8080);  
 server.on("request",function(req,res){  
});

今天新的内容NodeJS http模块、第三方模块 mysql-重点&难点  
 3.1：MySQL中SQL  
 SQL分类  
 DDL(数据定义语句) CREATE/DROP/ALERT/TRUNCATE  
 DML(数据操作语句)INSERT/UPDATE/DELETE  
 DQL(数据查询语句)SELECT  
 DCL(控制用户语句)GRANT/REVOKE  
  
 3.2：NodeJS访问MySQL服务器  
 为了精简NodeJS解析器，官方没有提供访问数据库模块，必须使npm工具下载第三方模块.在www.npmjs.org 上搜索mysql,可以得很多相关的模块，每个模块有使用说明.  
 npm i mysql  
  
  
 mysql模块使用步骤  
 (1)创建数据库连接  
 const mysql = require("mysql");  
 var conn = mysql.createConnection({...});  
 (2)创建sql.并且发送sql  
 conn.query("SQL...",function(err,result){});  
 (3)关闭连接  
 conn.end();

## Day04

NodeJS 模块分类  
(1)官方提供原生模块 querystring url Buffer fs http  
(2)第三方模块 npm mysql  
(3)自定义模块  
 文件模块:user.js exports.XX require("./user");  
 目录模块:node\_modules/db/package.json require("db");  
http模块的使用  
const http = require("http");  
var server = http.createServer();  
server.listen(8080);  
server.on("request",(req,res)=>{  
});  
mysql模块的使用  
#npm i mysql ./node\_modules/mysql/...  
const mysql = require("mysql");  
var pool = mysql.createPool({  
 host:"",user:"",password:"",database:"",connectionLimit:10  
});  
pool.getConnection((err,conn)=>{  
 conn.query("SELECT ...?,?",[x,y],(err,result)=>{  
 ....  
});  
});  
下面指定用来查询端口是否启动  
  
3:NodeJS框架 Express  
 使用官方提供http模块可以创建一个web服务器应用，但是此模块非常底层，要处各种情形，比较繁琐。推荐使http模块进一步封装简化模块---express---第三方模块。是一个基于NodeJS http模块而编写高层模块，简化web服务器应用的开发，V4版本官方  
  
该模块还有自己专用官方网站: http://expressjs.com  
 该模块有一个中文镜像网: http://www.expressjs.com.cn  
  
 使用方式  
 (1)下载express npm i express  
 (2)在自己的模块中引入express模块  
 const express = require("express");  
 (3)使用http模块创建web服务器对象,让express处理请求  
 var app = express();  
 var server = http.createServer(app);  
 server.listen(8080);  
  
HTTP协议--请求和响应  
 请求方法(方式):用于标明此次请求目的  
 GET 表客户端想"获得"指定的资源  
 POST 表客户端想"添加/上传"指定数据给服务器，相关数据在请求主体中.  
 PUT 表客户端想"放置/更新"服务器上指定资源，相关数据在请求主体中.  
 DELETE 表客户端想"删除"服务器上指定资源  
  
POST和PUT语义区别:  
所有请求方法中只有二个者有请求主体,都可以向服务器传数据  
POST:一般情况表示"添加",每次POST请求都在服务器上产生一条数据记录.  
PUT:一般情况表示"更新",多次请求也只能产生一条影响.  
  
GET /user 表示客户端想获取所有用户数据  
GET /user?pno=2&psize=10 表示客户端想分页获取数据  
GET /user/pno/2/psize/10 表示客户端想分页获取数据  
GET /user/10 表示客户端想获取编号为10数据  
POST /user 表示客户端想添加一条记录  
uname=tom&age=10  
PUT /user 表示客户端想更新一条记录  
uname=james&uid=1  
DELETE /user 表示客户端想删除服务器所有用户数据  
DELETE /user?uid=9 表示客户端想删除服务器编号为9用户  
DELETE /user/9 表示客户端想删除服务器编号为9用户  
  
浏览器中如何发起GET请求  
 地址栏输入URL/超链接/js跳转/src/href/表单GET/AJAX-GET  
浏览器中如何发起POST请求  
 表单-POST/AJAX-POST  
浏览器中如何发起PUT请求  
 AJAX-PUT  
浏览器中如何发起DELETE 请求  
 AJAX-DELETE  
  
 使用 Express处理客户端的GET请求  
 app.get(url,(req,res)=>{  
 res.send("向客户端返回HTML字符串响应");  
 res.sendFile(\_\_dirname+"/向客户端输出指定文件内容");  
 res.json(obj); //将obj转化为JSON字符串，输出给客户端  
})  
Express路由  
//路由(Route)=Method+Path+fn  
  
参数:  
 (1)接收GET请求中查询字符串  
 //GET /user?uid=10&loc=bj  
 app.get("/user",(req,res)=>{  
 //express为每一个req对象添加属性query  
 req.query.uid;  
 req.query.loc  
});  
 (2)接收GET 请求中请求参数  
//GET /user/jsj/60  
 app.get("/user/:type/:id",(req,res)=>{  
 //express为每个req对象添加属性 params属性  
 console.log(req.params.type);//jsj  
 console.log(req.params.id);//60  
});  
  
 疑问:如何使express处理静态资源(jpg/css/js...)  
 express 4.0 官方提供函数(中间件)  
 app.use(express.static('public'));  
 #若客户请求了/public目录某个指定资源,它可以直接向客户端返回，  
 #不会再调用后续路由  
  
 #使用Express处理客户端的POST请求  
 app.post(url,(req,res)=>{  
 //接收请求主体的数据  
 req.on("data",(data)=>{  
 var obj = qs.parse(data.toString());  
})  
});  
#使用Express处理客户端的PUT请求  
 app.put(url,(req,res)=>{  
 //接收请求主体的数据  
 req.on("data",(data)=>{  
 var obj = qs.parse(data.toString());  
})  
});  
#使用Express处理客户端的DELETE请求  
app.delete(url,(req,res)=>{});

# Angular

## Day01

alt+f2 方便在webStorm中预览页面效果的快捷键组合  
  
一、设计原则  
①YANGI  
you arn't gonna need it  
不写不需要的代码  
②KISS   
keep it simple and stupid  
越简单越好  
  
③DRY  
Dont't Repeat Yourself  
善于封装和重用  
④高内聚、低耦合  
耦合：不同的功能模块之间的关系  
低耦合：降低不同模块之间的关系  
⑤SRP  
single responsibility principle  
功能要细化，每一个方法或者模块，就用来实现一个指定的功能。  
⑥OCP  
open closed principle 开闭原则  
对于扩展开放 对于修改关闭  
⑦最少知识法则  
封装的功能单位要小，只与自己相关  
  
二、设计模式  
23种常用的设计模式。  
  
了解常见的设计模式：  
工厂方式模式、单例模式、组合模式、适配器模式、迭代模式。。  
MVC： model(数据) View（视图） Controller（控制器）  
MVVM：model view viewModel  
  
三、AngularJS 概述  
1.\*与angular2.0：  
angular2.0 typescript  
what？是一个用来实现SPA的基于MVC的框架  
when？实现SPA  
  
why?  
核心特点：  
①采用了MVC的设计模式  
②双向数据绑定  
③依赖注入  
④模块化设计  
how？  
采用引入js文件的方式  
  
四、基本使用  
1、表达式  
语法：<any>{{表达式}}</any>  
作用：将表达式的结果 输出 到当前位置  
  
算术运算、比较运算、逻辑运算、三目运算。。  
  
新建一个html文件，实现：  
通过ngInit来初始化一个对象数组(4个对象)，每个对象包含name\sex\score\age，将数组中的数据通过双花括号语法显示在table。  
  
ngApp/ngInit/ngRepeat/ngIf(添加/删除)/ngShow/ngHide/ngSrc(比较特殊，在读取变量值要加上双花括号)/ngDisabled/ngChecked/ngClass/ngStyle/ngClick/ngDblclick/ngChange...  
  
2、常用指令  
ng中指令：是由angularJS提供的，可以在标签中作为扩展属性、扩展标签（ngView作为标签来使用：<ng-view/>）  
  
指令都是采用驼峰式的命名方式，在使用的时候采用烤串式命名  
  
ngApp：  
①启动angularJS  
②指定ng的作用范围  
注意事项：第一次调用时有效的  
  
ngInit:初始化变量  
在这个指令中，声明变量时不需要写上var关键字；如果要声明多个变量，通过分号隔开  
<any ng-init="count=10;name='zhangsan'">  
  
ngBind：  
<any ng-bind="表达式"></any>  
将表达式的结果输出到当前元素位置去显示，与双花括号的区别是：  
双花括号在ng启动成功之前是会作为普通的字符串去显示，启动成功，替换表达式的结果，会有闪烁的情况出现  
ngBind指令由于是作为属性来使用的，如果ng没有启动成功，是不识别该属性，启动成功之后，直接显示值  
  
ngRepeat:  
实现一个集合的遍历，并同时为集合中的每一项 实例化一个模板  
用法：  
<any ng-repeat="tmp in myList">  
{{tmp}}  
</any>  
<any ng-repeat="(key,value) in myList">  
{{value}}  
</any>  
  
ngRepeat提供属性：  
$index 迭代的偏移量  
$dle/$last/.. 布尔类型  
  
ngIf指令  
选择挂载到DOM或者从DOM中移除:  
并不是隐藏或者显示，而是直接添加或者删除  
<any ng-if="表达式"></any>  
  
通过ngIf来显示成绩及格的行。  
  
3、常用其他指令  
ngSrc 配合img标签  
<img ng-src='img/{{imgUrl}}'></img>  
  
ngShow/ngHide  
根据表达式的值 决定一个元素是要显示还是隐藏  
ngDisabled、ngChecked  
  
4、MVC  
在ng中如何理解MVC？  
Model 通过js定义在$scope中的变量  
View 通过ng指令和双花括号增强后的html  
Controller：  
  
  
为什么会出现MVC？  
①工程化  
②代码量-》封装和重用  
  
mvc的工作原理：  
视图接收用户的操作，会触发控制中定义好的方法，在方法中实现对数据的增删改查的操作，数据和视图是解耦的，数据和视图是通过绑定进行显示的。  
M：定义的数据  
V：视图（绑定了数据）  
C：方法  
  
  
通过ng实现MVC的基本步骤：  
①创建模块  
var app = angular.module('模块名字',['依赖模块1','依赖模块2'])  
  
②调用模块  
ngApp--》 ng-app="模块名字"  
③创建控制器（C）  
app.controller('控制器的名称',func)  
  
④使用控制器  
ngController：  
ng-controller='控制器名称'  
  
⑤创建模型数据（Model），显示在视图（V）中  
  
$scope:是一个作用域对象  
  
实现一个demo：  
有一个p,还有一个按钮，  
点击按钮，count会自增，显示在p标签  
①创建MVC的脚手架  
②实现业务逻辑  
  
实现MVC的基本步骤：  
①模块的创建和使用 module  
②控制器的创建和使用 controller  
③定义模型数据 $scope  
④在视图中显示数据，接收用户的操作修改数据  
  
ngInit和$scope都能够完成数据的初始化，怎么选择？  
建议：多去使用$scope。  
把和功能模块相关的数据存储在$scope中，如果多个模块都要用到同一类数据，可以通过ngInit初始化。

## Day02

ng是由很多指令的：  
ngApp ngRepeat ngIf ngShow ngHide ngChecked ngDisabled ngSrc   
ngBind ngClass ngStyle  
  
在webStorm中创建一个文件模板的基本步骤：  
①拷贝要使用的代码  
②点击左上角的file，选择new  
③选择Edit File Templates  
④点击弹窗中右上角的+号  
⑤修改name和extension为创建的文件模板的名称以及格式，点击ok保存即可  
  
一、自定义指令  
1、如何自定义？？  
angular.module()  
app.controller()  
//创建指令  
app.directive('指令的名称'，function(){  
 return {}  
})  
  
指令的命令规则：指令的名称由两部分构成，前缀：一般使我们模块或者项目的名字缩写； 后缀：用来描述指令的功能  
  
属性：  
template：指定要显示的模板内容  
restrict: "EAC" //E(Element)A(Attribute)C(Class)M(Comment)  
replace: true,替换并显示出来  
scope  
  
2、如何使用自定义的指令？？  
  
注意事项：指令在命名时是要遵循驼峰式命名规则；在使用时要遵循烤串式用方  
'tsHello' --> ts-hello  
根据配置指令时的restrict属性所对应的值，在合适的场景下使用指令。  
  
3、自定义指令传递参数  
怎么传？  
在调用指令的时候可以传 将参数作为属性对应的值  
怎么接受？  
scope  
  
二、双向数据绑定  
方向1：从'数据'绑定到'视图'  
绑定方式：  
①常用指令： ngRepeat/ngIf/ngShow...  
②双花括号：{{}}  
  
方向2：从‘视图中用户的输入’绑定到‘数据’  
绑定方式：指令：ngModel  
  
如果要监听数据的变化：  
$scope.$watch('txt',function(){})  
  
在实现下拉框的过程中，如果数据是动态的，可以通过ngOptions将数据绑定到视图中。  
ngOptions语法：  
 <select  
 ng-model="city"  
 ng-options="city.cityName for city in cities">  
 </select>  
  
三、过滤器  
  
3.1 内置的过滤器  
ng中的过滤器为了实现对于表达式结果的筛选、过滤、格式化，达到更好的表现效果。  
过滤器的语法：支持多重过滤和传参  
{{expression | 过滤器名称 : '参数' | 过滤器名称2：‘参数’ }}  
  
| -》 管道  
  
常用的过滤器：  
currency 货币样式的过滤器  
date 日期  
uppercase/lowercase 大小写的处理  
orderBy 对指定的数组进行升序或者降序排列  
number 格式化数字为文本（对有小数点的数据的处理）  
limitTo 限定数组或者字符串要显示的个数  
  
3.2 自定义过滤器  
  
自定义过滤器方式：  
app.filter('过滤器名称',function(){  
 return function(input，arg){  
 //input是传递给过滤器的数据  
 //arg 是过滤器本身的参数  
 return '过滤后的结果'  
 }})  
使用自定义的过滤器:  
和内置的过滤器用法是一样的  
  
  
四、ng模块内置的方法  
序列化、反序列化  
toJson:将对象或者数组 序列化 为json格式的字符串  
fromJson：相反的过程  
  
  
五、服务  
服务的本质是一个单例对象，提供数据和对象。  
  
两大类：  
①内置服务  
$location、$http、$window、$scope...  
使用内置服务中提供的方法：  
 第一步 将需要用到的服务注入进来 function($scope,$location)  
 第二步 调用服务中提供的方法 数据。。  
②自定义的服务  
  
面试题：$scope与$rootScope之间的关系  
  
1、不同的控制器之间的数据是彼此隔离的  
2、不同的控制器之间如何共享数据？？  
  
$scope的id是从2开始依次往上递增，  
id为1的是谁？ $rootScope是根作用域对象，其余的$scope都是子对象  
  
方式1：借助于$rootScope  
把要共享的数据存在$rootScope,所有的子对象就可以读取该数据  
  
方式2：借助于控制器之间的嵌套  
在调用控制器的元素 之间的嵌套关系  
<div ng-controller='myCtrl02'>  
 <div ng-controller="myCtrl03"></div>  
</div>  
  
方式3：触发事件传递参数  
绑定事件：  
$scope.$on('事件名称',function(event,data){})  
  
触发事件：  
//从子作用对象 向 父作用域对象 传递事件  
$scope.$emit()  
//从父作用对象 向 子作用域对象 传递事件  
$scope.$broadcaset()  
①构造数据  
对象数组  
②显示出来  
③关联部分选中和全部选中  
双向数据绑定

## Day03

一、ng常用的内置服务  
$scope $location $rootSCope $interval $timeout $http...  
  
数据绑定的工作原理：  
 每一次完成数据的绑定，在ng框架的背后 都会watcher， 监听数据的变化，如果发生了变化就会更新DOM。  
 如果检测数据的变化？？ng在初始化之后，会有一个循环，在不断的判断数据的变化，将这个循环称之为$digest循环。  
 什么需要手工执行$scope.$digest()?  
 有冲突的时候，发现数据变化 没有引起DOM的更新  
 除了$digest()，有时还会使用$scope.$apply();  
 不同点在于$apply会触发$rootScope所有的子作用域对象的$digest循环。  
  
1、$interval $timeout  
var promise = $interval(func,delay) 开启一个周期性定时器  
$interval.cancel(promise) 取消一个周期性定时器  
var promise = $timeout() 开启一个定时器  
$timeout.cancel(promise) 取消一个定时器  
  
2、$http 处理网络请求  
发起get请求：  
$http({method:'get',url:''}).success().error()  
$http.get('url').success().error()  
发起post请求：  
$http.post(‘url’,data).success().error()  
$http({method:'post',url:''}).success().error()  
//如果post请求需要传参：  
 app.run(  
 function ($http) {  
 $http.defaults.headers.post = {'Content-Type':'application/x-www-form-urlencoded'};  
 })  
  
//实现表单的序列化  
var result = $httpParamSerizlizerJQLike(对象);  
//key1=value1&key2=value  
二、ng自定义服务  
1、factory方法  
创建的方式:  
 var app = angular.module();  
 app.factory('服务名称',function(){  
 return {  
 key:value,  
 funcName:func  
 }  
 })  
使用的方式:  
 和内置服务用法是一样的  
  
2、service 方法  
app.controller/filter/module/service..  
创建服务的方式：  
app.service('服务名称',构造函数);  
使用服务的方式：  
就像是使用内置的服务，引入到ng对象中就可以了  
  
3、const/value  
创建时一些常量或者变量服务  
app.constant('服务名称',{})//常量  
app.value('服务名称',{})//变量  
总结：factory 必须要求有一个返回值  
service 指定的构造函数 constant 创建的常量服务 value变量服务  
  
三、依赖注入  
  
依赖注入：将需要用到的对象或者服务 直接注入到对应的ng对象（服务，控制器..）,直接去使用，不用理会要注入的对象内部是怎么实现的。  
文件压缩： yui-compressor uglify..  
文件压缩：删除注释、没有语义的空白字符，同时做了代码的混淆（$scope -> a 这是对于代码的保护同时也是一种压缩）  
之前的服务注入的方式 在进行文件压缩之后，是无法找到对应的服务。  
考虑到在进行文件压缩时，字符串或者数组是不会发生变化的，从这里找到切入点。  
  
依赖注入：  
①推断式（猜测式）  
app.controller('myCtrl',function(服务1名称){  
})  
②标记式  
  
将依赖的各个服务的名称放到一个数组中作为依赖注入的标记，即使压缩之后，我们依然可以通过数组中的服务名称找到对应 的服务，提供给我们去使用。  
//通过一个数组对依赖的服务做标记  
funcHandler.$inject = ['$scope','$show'];  
由于标记式的写法是依赖一个字符串的数组，所以服务是有先后顺序的：在$inject中数组中服务的顺序要和控制器处理函数引入服务的顺序是要一致的  
funcHandler = function($scope,$show)  
③行内式（内联式）  
要求在创建一个ng对象时，第二个参数可以是一个数组，数组中存储的是服务的名称，最后一个必须是ng对象对应的处理函数  
app.controller('myCtrl',  
 ['$scope','$print',function($scope,$print){}]);  
 注意事项：最后一个元素必须是方法，而且服务的名称在数组中和在方法的顺序要保持一致。  
  
要求：注入服务，统一使用行内式依赖注入。  
  
进阶知识：注入器($injector)  
  
在ng启动的时候，就已经被初始化了，提供了常用的API(application interface 应用程序接口)，有两个经常用到的方法:  
has() 判断一个服务是否存在  
get ('$show') 得到一个指定的服务的对象

Day04  
  
一、模块化设计  
1、自定义模块以及相互调用  
angular.module('myApp01',['myApp02'])  
  
2、面试题：一个ng的模块可以包含哪些东西？  
控制器  
指令  
过滤器  
函数  
服务  
  
二、内置模块  
1、路由模块  
实现SPA。  
SPA：单一页面应用程序，对应的是多页面应用程序（传统）。  
  
SPA的工作原理：  
①页面url地址  
http://127.0.0.1/index.html#/路由地址  
②请求完整的html页面  
③解析路由地址  
④从路由词典找到该路由地址对应的配置信息--》代码片段，控制器。。  
⑤发起异步请求，加载代码片段到完整的页面中（有一个容器）  
  
  
2、在ng中如何实现SPA？  
基本步骤：  
①引入angular-route.js,在创建自定义模块指定依赖ngRoute  
②指定容器 盛放代码片段 ngView  
③创建需要的代码片段  
④配置路由词典  
$routProvider.when 添加一条路由信息  
$routProvider.otherwise 处理异常  
⑤测试  
  
3、在SPA页面跳转的方式  
①直接修改地址栏（测试）  
②优化后的超链接  
<a href='#路由地址'></a>  
<a href='#/mySettings'></a>  
③通过js的方式  
$location.path('路由地址');  
$location.path('/mySettings');  
  
不同的控制器之间通信方式：  
①$rootScope  
②控制器之间的嵌套  
③通过event ($scope.$broadcast/$emit/$on)  
  
路由传参：  
①明确 发送方和接收方  
②配置接收方的路由地址  
when('/mySettings',{})   
--->  
when('/mySettings/:id',{})  
③借助于$routeParams服务获取参数  
④发送  
<a href='#/mySettings/20'>  
$location.path('/mySettings/20')  
  
要求：  
从个人中心页 跳转到 设置页面时 传值，设置页面接收到值并显示在视图  
①发送：个人中心 接收：设置页面  
②设置接收方的路由  
when('/mySettings',{})  
-->  
when('/mySettings/:id')  
③$routeParams  
④发送  
a href='#/mySettings/20'  
  
指令：  
ngInclude  
用法： <div   
ng-include="'tpl/  
include/header.html'"></div>  
作用：在当前页面动态加载指定的代码片段

三、动画模块  
①直接引入对应的js angular-animate.js  
②在自定义模块时，依赖于ngAnimate  
③修改ng动画模块内置的样式类  
ng-Enter ng-Enter-Active   
ng-Leave ng-Leave-Active  
  
  
四、表单验证  
$pristine 原始的，没有操作过  
$dirty 操作过的  
$valid 有效  
$invalid 无效  
$error 对象：email,url,..  
  
  
注意事项：  
①提交事件 ngSubmit  
②给每一个表单组件指定name，通过ngModel绑定到模型数据  
③表单本身也可以加上判断  
myForm.$valid 只要是表单中所有的表单组件通过验证，该值就是true，否则就是false

# Webapp

## Day01

一、移动端开发的趋势

①NativeApp 原生开发的app

借助于SDK（software development kit 软件开发包）提供的组件和服务进行开发生成的app

②WebApp 网页应用

第一种理解：基于前端技术开发的移动端的页面

第二种理解：基于前端技术和原生技术（浏览器组件webView）开发的可以安装在手机上的app

③HybridApp 混合编程app

基于前端技术和原生技术（调用智能设备底层硬件）开发的可以安装在手机上的app

混合编程App 是开发成本比较低、维护更新比较简单、扩平台性比较好。

jQueryUI -》pc

bootStrap -》 响应式

ionic -》 移动端的ui库

ZeptoJS -》 移动端操作DOM的js库

phoneGap -》 在移动端调用智能设备底层硬件

二、编写Android平台下的hybridApp

http://www.androiddevtools.cn/

搭建iOS平台的环境：

①MacOS

②苹果开发者账号

③xcode

1、搭建Android平台开发环境

①配置jdk(Android是以Java作为开发语言)

java development kit java开发工具

②配置adt（下载adt-bundle）

android develop tools 安卓开发工具（集成了eclipse、sdk）

(as=android studio Google官方建议使用的开发工具 )

2、使用Android开发工具 创建Android项目

使用eclipse的注意实现：

①显示工具条

window->show toolbar

②恢复视图

window -> reset perspective

③使用android sdk manager 更新sdk

直接连接google，配置镜像服务器

推荐网址:Android SDK在线更新镜像服务器

④创建虚拟设备

android virtual device manager

配置3个东西：1、起名字 2、选设备 4寸 3、指定sd大小 512M

其他的虚拟设备解决方案：genymotion

使用eclipse创建一个Android项目：

①点击file -> new ->Android Application Project

②给应用和包起名字

③设置图标

④运行：选中项目，点击右键，选择run as -->android application.

(打包生成apk格式的安装包-》借助于adb传输手机-》安装-》运行)

3、第一个原生App

①在xml文件中通过图形化布局的方式添加一个button

②在java文件中，查找button并设置回调

java语言、Android SDK提供的各个组件和服务

4、一个Android工程的目录结构

src -> source 存储的java代码，主要是处理业务逻辑

gen -> 自动生成的文件 R.java 记录了当前Android工程中所有的资源id

bin -> 存储可以安装在手机上的apk格式的安装包（如何在文件系统中找到该安装包？ 选中apk，点击右键，选中properties 查看location，复制到文件窗口）

assets -》资源文件夹 前端代码放到目录，放置被编译处理

libs-》libraries 存储第三方的jar包

res -》 resource 资源 图片、布局文件、菜单设置。。

AndroidManifest.xml 配置文件

versionCode

权限的配置

java文件的配置

..

如何给App添加网络权限？

①找到工程中的AndroidManifest.xml文件

②选中permissions标签

③添加网络权限

点击add，在弹窗中选中uses permission，在右侧的下拉菜单中选中需要用到的权限，保存即可

三、实现hybridApp

1、基本步骤：

①编写你的前端代码

②创建一个Android工程，将前端代码拷贝到Android工程

③使用原生组件WebView加载指定的页面

④运行

错误：

①安装超时

failed to install demo03.apk ..:timeout

原因：adb服务挂掉了

解决方案：重启eclipse和虚拟机

②库是无法运行的

android library can't be launched

解决方案：新建项目，java代码拷贝过去

③警告

ActivityManager: Warning: Activity not started, its current task has been brought to the front

如果代码没有做任何的修改，没必要去重新编译运行的

2、WebView的使用

WebView wv = new WebView(getApplicationContext());

wv.loadUrl("file:///android\_asset/test01.html");

setContentView(wv);

练习：

实现一个HybridApp，前端代码的html中有一个p、button,功能：点击按钮隐藏段落。

①编写前端代码

②拷贝到Android工程的assets目录

③配置webview

④运行

3、在ecilpse中查看js控制台的输出

wv.setWebChromeClient(new WebChromeClient(){

@Override

public boolean onConsoleMessage(ConsoleMessage consoleMessage) {

// TODO Auto-generated method stub

return super.onConsoleMessage(consoleMessage);

}

});

4、混合编程的常见两种方式

## Day02

一、Ionic概述

http://www.ionic.wang/

ionicframework.com

http://ionicons.com/

虽然我们已经能够实现hybridApp，但是①样式不好看 ②功能不够强大，所以做的app不符合使用的要求！！

Ionic

what？ 是一个免费的开源的非常受欢迎的框架

Ionic = 一套移动端的ui库+AngularJS+ApacheCordova+uiRouter

when？ 帮助通过前端技术实现移动端的hybridApp

why？

①完美融合了AngularJS

②专注原生，结合着ApacheCordova调用智能设备的底层硬件

③ui风格特别漂亮

④强大的CLI操作

⑤活跃的社区

⑥性能优越，运行速度快（iOS6 Android4.1以下的版本是不支持的）

how？

搭建环境使用Ionic？

方案1：CLI（command line interface GUI-->Graphical User Interface）

①安装

npm install -g cordova ionic

②创建ionic项目

ionic start myApp tutorail/blank/tabs..

③运行Ionic项目

ionic serve (前提条件：必须是在ionic项目中才可以调用该指令)

方案2：直接使用官方所提供的文件进行开发

使用压缩包，在压缩包中提供了编写ionic项目所必须的图标、css、js文件

二、使用Ionic

1、Ionic移动端页面的基本结构

ionHeaderBar ionContent ionFooterBar

2、内置的颜色

positive 蓝色

calm 湖蓝色

assertive 红色

royal 紫色

energized 黄色

balanced 绿色。。。

3、button 按钮

button

button-positive/calm..

button-large/small

button-outline

button-clear

icon-left/right

ion-home/reresh...

button-icon

4、list

list

item

item-divider

item-icon-left

item-icon-right

item-note

badge badge-assertive badge-balanced

list-inset

item-avatar

item-thumbnail-left/right

5、card

card

item item-text-wrap

item-divider

6、form

list

item

item-input

input-label

item-stacked-label

item-floating-label

7、tabs

tabs

tab-item

tabs-icon-only/top/bottom/left/right

在Android设备中，tabs默认是在上方，在iOS设备上，tabs默认在底部。

可以通过$ionicConfigProvider设置Ionic中显示的效果

$ionicConfigProvider.tabs.position = 'bottom'

8、grid

row

col

col-10/20/33..

col-offset-10/20/33..

col-center/top/bottom

row-center/top/bottom

每日一练：

col row

①构造数据源

对象数组 有20个对象

②按照一行四列的方式显示对象数组

div class='row'

## Day03

一、Ionic提供基于AngularJS的扩展

1、ionHeaderBar

align-title="top/center/left" 标题对齐的方式

2、ionRefresher

下拉刷新组件

①用在ionContent第一个子元素

②通过on-refresh去绑定刷新要执行的动作

③事件 $scope.$broadcast('scroll.refreshComplete')

3、ionInfiniteScroll

上拉加载更多组件

①用在ionContent最后一个子元素

②通过on-infinite去绑定加载更多时要执行的动作

③事件 $scope.$broadcast('scroll.infiniteScrollComplete');

④该组件默认会在首次加载时判断是否可以加载更多数据，如果可以，就会自动触发on-infinite；可以通过immediate-check='true/false'调整是否允许首次加载是否要自动触发加载更多

注意事项：如果没有更多数据可以加载了，通过ngIf指令移除该组件

练习：通过ionic实现一个列表，同时支持下拉刷新和上拉加载更多，下拉--》列表的头部插入一条新的数据，上拉-》列表的底部插入一条数据。

4、$ionicScrollDelegate

在服务中封装了控制滚动的方法

scrollTop scrollBottom scrollTo()

getScrollPosition

例子：滚动到列表项内容为20：

Object {left: 0, top: 1060, zoom: 1}

程序控制恢复到指定的位置：

scrollTo()

5、ionicTab 选项卡/标签页

css:tabs tab-item tabs-icon-top/bottom/left/right

通过指令实现选项卡

ionTabs: 通过class来设置tabs-icon-top tabs-positive

ionTab： 通过属性title icon icon-on icon-off on-select/deselect ng-click

注意事项：第一个标签页式默认被选中的。on-select只要是该标签页被选中就会触发

练习：实现点击不同的tab页面，更新头部标题的信息

音乐--》music

电影--》movie

游戏--》game

6、侧边栏菜单

ionSideMenus

ionSideMenu

ionSideMenuContent

ionSideMenu:

side/width/is-eneabled

属性：menu-toggle menu-close

方法：$ionicSideMenuDelegate.toggleLeft/right

7、窗口

①actionSheet(从下往上弹出的操作表)

$ionicActionSheet.show({})

参数中的对象:titleText/cancelText/destructiveText/buttons/cancel/destructiveButtonClicked/buttonClicked

②popup(在中间显示的弹窗)

$ionicPopup.alert/prompt/confirm()

③loading

$ionicLoading.show/hide()

练习：实现一个列表，支持下拉刷新，在下拉刷新的处理函数中加载数据（每次加载数据，都要显示一个加载中的窗口 loading...,只出现2s）

分析：

①构造数据并显示

②支持下拉刷新，每次在头部插入一条随机数据

③在下拉刷新时添加加载中的窗口

8、列表

ion-list/ion-item/ion-checkbox/ion-toggle/'ion-radio

ion-option-button

ion-delete-button(show-delete='true')

ion-reorder-button(show-reorder='true')

11 22

22 33

33 44

44 55

55 11

二、uiRouter

## Day04

一、ngRoute知识点回顾

1、使用ngRoute创建SPA基本步骤

①引入angular-route.js 创建模块指定依赖ngRoute

②指定容器 ngView

③创建代码片段

④配置路由词典

app.config(function($routeProvider){

$routeProvider

.when('/myStart',{

templateUrl:'',

controller:''

})

.otherwise({redirectTo:'/myStart'})

})

2、如何跳转

js:$location.path()

a <a href='#/myStart'/>

直接修改地址栏

3、如何传参

①发送 接收方

②配置接收方的路由

when('/myStart',{})

-->

when('/myStart/:id',{})

③接收参数 $routeParams

④发送参数

二、uiRouter用法

uiRouter帮助我们在ng、react项目中充当路由管理库，实现SPA。

最大的特点是可以实现嵌套路由。

1、使用uiRouter创建SPA的基本步骤

①引入angular-ui-router.js 并在创建模块时指定依赖ui.router模块

注意事项：在ionic模块中已经指定过依赖ui.router模块，在ionic的项目中直接指定依赖于ionic就可以使用uiRouter。

②uiView 指定盛放代码片段的容器

③创建代码片段

④配置状态机

app.config(function($stateProvider,$urlRouterProvider){

//添加状态

$stateProvider

.state('状态名字',{

url:'/\*\*'，

templateUrl:'',

controller:''

})

//处理异常

$urlRouterProvider.otherwise('某状态对应的url')

})

2、跳转的方式

直接修改地址栏

js: $state.go('状态名字')

属性：<any ui-sref="状态名字"></any>

a <a href='#url'/>

3、传参

①发送 接收

②配置接收方state对应的url

③接收参数

$stateParams

④发送参数

从send跳转到pay并传递价格40.

①发送：send 接收方pay

②配置接收方对应的url

.state('kflPay',{

url:'/pay',

templateUrl:'tpl/pay.html'

})

-->

.state('kflPay',{

url:'/pay/:price',

templateUrl:'tpl/pay.html'

})

③接收参数

$stateParams.price

④发送参数

http://127.0.0.1/index.html#/pay/20

$state.go('kflPay',{price:40})

<any ui-sref='kflPay({price:40})'>

a href='#/pay/40'

注意事项：通过js或者uiSref属性传参时，参数的key要和配置状态机中url的变量名称要保持一致

url:'/pay/:price' --> {price:40}

练习：（11:10 - 11:25）

实现SPA

代码片段：login register start

login： 按钮（没有账号，去注册）--> 注册

按钮（登录 js的方式跳转并传参） 传参id=10--》start

start： 接收到参数并显示在视图

a 点击退出登录 ---》 login

register：按钮--》跳转到login

三、uiRouter处理嵌套路由

嵌套：在一个代码片段中 还可以插入其他的代码片段

demo：有两个代码片段，login mail (加载inbox、outbox)

总结

如果要在一个代码片段(parent.html)中插入其他的代码片段(son.html)

方案1：

①在parent.html 设置uiView

②配置状态

如果parent.html的状态为parentState,

son.html的状态就应该是parentState.sonState.

举例： myMail.html -> 状态：mail

inbox.html -> 状态：mail.inbox

③触发状态

触发子代码片段的状态ui-sref="parentState.sonState"

举例:ui-sref='mail.inbox'

方案2：

①在parent.html 设置uiView

②配置状态

$stateProvider

.state('mail',{

url:'/myMail',

templateUrl:'tpl/myMail.html'

})

.state('inbox',{

url:'/inbox',

templateUrl:'tpl/inbox.html',

parent:'mail'

})

③触发状态

在触发状态时 直接指定状态名称为配置过的字代码片段的状态名字即可。

<ion-item ui-sref='inbox'>收件箱</ion-item>

响应式开发啦项目：

ui-》bootStrap

数据 网络 -> $http

服务器端-> php

移动端的开饭啦项目：

技术架构选型：

ui-> ionic

数据 网络 -> $http

服务器端-> php

## Day05

ionic2升级版：

https://github.com/kunyashaw/ionic2ExamplesV2

一、Zepto概述

zeptojs.com

http://github.e-sites.nl/zeptobuilder/

1、what

轻量的针对现代高级浏览器的js库，与jQuery有着类似的api，可以让使用过jQuery的程序员把精力放到开发

2、why

ionic适合中大型，zepto可以帮助实现简单的移动端的应用程序

①轻量

②低廉的学习成本

③支持现代高级浏览器，满足移动端浏览器的需求

3、where

移动端DOM操作比较频繁的的app

4、how

直接引入zepto.js文件到工程中

二、使用zeptojs

1、核心模块 (zepto)

①工具类方法

each/map/trim/parseJson...

②操作节点

选择器:标签、id、class、属性、父子、后代

selector.js 模块中还支持一些其他的选择器

添加

insertAfter insertBefore

append appendTo

prepend prependTo

删除

remove removeClass

修改

html/css/addClass/toggleClass

//查找兄弟、子、父、祖先。。

prev next children parent parents

2、detect模块

$.os 记录操作系统相关的信息

Object {android: true, version: "4.4.2", tablet: false, phone: true}

$.browser 记录浏览器相关的信息

Object {webkit: true, version: "55.0.2883.75", chrome: true}

3、event模块

on/off/one/trigger

练习：

计数器：点击button1按钮(on)，数字自增，从0开始，当数值超过5的时候，提示：再点击就解绑了（off），再点击，只提示一次“已经解绑了” (one);

自定义事件以及触发。

给button2添加一个自定义事件customCLick（处理函数：在控制台输出一句话）,当button被加载时，自动触发该事件

4、Ajax模块

$.ajax/$.get/$.post/$.param

练习：请求方式为post

var user = {name:'zhangSan'};

服务器解析传过来的name，返回Hello zhangsan。

注意:apache 端口

5、form

serialize()/serializeArray()/submit()

序列化 指定submit事件触发时对应的处理函数

注意：

①form必须指定name属性

②复选框/单选框 没有值，序列化会的结果会略过该值

6、touch

提供了移动端触控相关的事件支持

①tap

tap/longtap/doubletap

②swipe

swipe/swipeLeft/swipeRight/swipeUp/swipeDown

7、fx 动画模块

封装好的方法：

fadeIn fadeOut fadeTo fadeToggle

show hide toggle

animate();

## Day06

一、在线版简历

要求：使用zeptojs来完成数据的加载和显示

步骤

1、请求getWorkInfo.php拿到数据并显示在视图

get/post/ajax

找到容器，动态的生成节点，插入到容器

2、请求getTechInfo.php拿到数据并显示在视图

每日一练：

动手将pc端的开发啦项目 部署在新浪云。

## Day07

新浪云部署注意事项：

①选择的服务器端环境为php

②初始化数据库

数据库要与项目符合

③修改数据库连接方式

//$conn = mysqli\_connect('127.0.0.1','root','','kaifanla');（这个适合本地调试，在部署到新浪云里时不再使用，所以使用下面这个版本）

都要替换成下面这行：

$conn = mysqli\_connect(SAE\_MYSQL\_HOST\_M, SAE\_MYSQL\_USER, SAE\_MYSQL\_PASS, SAE\_MYSQL\_DB, SAE\_MYSQL\_PORT);

④访问出错

1、查看控制台有没有出错

2、出错了

如果涉及到服务器端 看network-》找到请求的服务器端地址-》点击headers、response 排除请求参数格式不对，返回的数据与预期有冲突等

如果只是本地的数据，到当前页面查看上下文

二、phoneGap

1、what？

http://phonegap.com

http://build.phonegap.com

phoneGap是一个基于前端技术来开发跨平台移动app的平台；phoneGap所开发的app与传统的webApp最大的区别是可以调用智能手机的核心本地功能。

当前适合hybridApp的公司：

phoneGap/cordova

http://www.wex5.com

http://www.appcan.cn/

https://dcloud.io/

http://www.369cloud.com/

2、where？

实现支持调用底层硬件的hybridApp

3、why？

①兼容性

write once ,run everywhere

②标准化

w3c标准

③大众化

不需要原生的app编程基础，就能使用html/css/js编程生成可以安装在手机去运行的app

4、搭建开发环境

phoneGap与cordova的关系：cordova是phoneGap的核心代码开源之后形成的开源项目。

类似chrome与webkit的关系

http://cordova.apache.org

所支持的插件：

http://cordova.apache.org/plugins

http://phonegap-plugins.com/

搭建环境：

①通过命令行CLI(command line interface )

②借助于phoneGap官方所提供的GUI(graphical user interface)工具

按照GUI来搭建：

①按照桌面端的软件

②按照移动端的app到模拟器

1、启动eclipse，找到avd manager 启动虚拟设备

2、找到adb

3、找到apk格式的安装包

4、安装

windows+R --> cmd

将app安装到真实手机：

ios-》appStore-》phoneGap

android-》GooglePlay->phoneGap

（https://github.com/phonegap/phonegap-app-developer/blob/master/resources/release/android/PhoneGap-1.7.9-release.apk）

三、使用phoneGap

1、事件

deviceReady 当所有的插件加载完毕

online/offline

\*\*\*button

pause/resume

battery\*

2、device插件

使用device插件所提供的对象 读取设备属性显示在当前的页面中。

device.model/platform/version/uuid/....

3、accelerometer 加速度传感器插件

注意事项：

①不是每一个插件都做到了支持所有的平台

②每一个插件在使用的过程中，在各个平台去用有可能会有一些差异

navigator.accelerometer.getCurrentAcceleration/watchAccleration/clearWatch

4、camera

camera.getPicture(func,func,{sourceType:})

注意事项：

sourceType:Camera.PictureSourceType.Camera/PHOTOLIBRARY/SAVEDPHOTOALBUM

拍照、相册、已保存相册

5、capture 采集音、视频、图片

navigator.device.capture.captureAudio/captureIamge/captureVideo

三个参数：成功处理函数、失败处理函数，相关的设置

6、compass 罗盘

compass.getCurrentHeading

compass.watchHeading

compass.clearWatch

罗盘是一个检测设备方向或朝向的传感器，使用度作为衡量单位，取值范围从0度到359.99度。

7、文件操作

写操作：

①得到文件系统（fileSystem）

window.requestFileSystem（）

②在指定的目录中创建一个文件(fileEntry)

该对象代表文件系统中的一个文件

getFile：创建或查找一个文件。

③得到一个写对象(FileWriter)

是一个允许用户写文件的对象。

fileEntry.createWriter()

④写入内容

fileWrite.write()

读操作：

①得到fileSystem

②得到fileEntry

③得到file

④通过fileReader读取file对象中的内容

8、notification 通知提醒类的插件

视觉、听觉、触觉的通知

notification.alert/confirm

notification.beep(3);//发出声音的次数

notification.vibrate(3000);//震动持续的毫秒数

模拟器必须是有SD卡的

练习：读取相册的图片信息并设置为头像

①在模拟器中 保存一张照片

打开apache服务，在模拟器中通过浏览器去访问当前PC的ip地址，找到项目中的img目录任意一张照片，点开，长按保存

②通过编码

camera插件中getPicture方法读取照片信息

# React

## Day01

一、移动端课程回顾：

①hybridApp

3种mobile开发模式 ：webApp nativeApp hybridApp

搭建Android开发环境

webView （new\setJSEnable\loadUrl\setContentView）

②ionic

ionic = ng + ui库（css样式类以及基于ng扩展的指令和服务）+ uiRouter（配置状态机）+cordova

ionic样式类：button/list/card/checkbox/form/...

ionic基于ng的扩展：ionRefresher/ionInfiniteScroll/$ionicLoading/$ionicPopup/$ionicScrollDelegate...

uiRouter:第三方的框架，可以用在ng1、ng2、React中，最大的特点就是不仅仅支持路由管理，支持路由嵌套

①基本步骤

引入angular-ui-router.js 依赖ui.router

$stateProvider.state() $urlRouterProvider.otherwise()

②跳转

$state.go()

ui-sref

a href='#url'

③传参

发送方、接收方

配置接收方状态所对应的url

.state('kflDetail',{

url:'/kflDetail/:id'

})

接收参数 $stateParams.id

发送参数

③zeptojs

移动端的jQuery，提供了常用的选择器和DOM操作方法，

同时支持在移动端的处理：touch、detect

zepto默认包含的5个模块

①zepto 核心模块

②event 事件模块 on off one trigger

③ajax 通信模块 $.ajax() $.get/post/load/jsonP

④form 表单模块 submit serialize serializeArray

⑤ie ie10+

detect： $.os $.browser

touch : tap/longTap/doubleTap swipe/swipeLeft...

fx : $.animate/show/hide/toggle/fade\*

①在线版简历

通过zepto完成服务器通信以及选择器和DOM操作

②将pc端2048游戏重构移动端的app

给项目添加滑动事件、计算出来宽高等属性设置适应不同的屏幕大小、打包成hybridApp、上传到应用商店（蒲公英）、新浪云部署（①将代码包传上去 ②指定服务器 ③配置数据库）

4、phoneGap/cordova

phoneGap是一个平台，提供丰富的插件（允许使用js调用设备底层的软硬件信息），最终实现接近于原生app体验的应用程序

device/acclerometer/notification/file/capure/camemra/...

二、React概述

what？React是一个开源的js库，将数据渲染成视图

why？

fb遇到：①浏览器性能 ②代码维护太复杂

创建的React，通过VirtualDOM来尽量减少对DOM的直接操作解决浏览器遇到性能瓶颈问题，通过单向数据流来解决代码维护的问题

where?

提供渲染视图，数据操作比较频繁的场景

how？？

①CLI

https://github.com/facebookincubator/create-react-app

npm install -g create-react-app

create-react-app my-app

cd my-app/

npm start

②直接引入对应的js文件

核心思想：组件化

核心概念：

①组件

②jsx （并不是强制要求使用jsx，只是建议使用jsx）

并不是一种新的语言，只是js的扩展：允许在js中直接来编写html标签

③VirtualDOM 虚拟DOM

结合着React当中的Diff算法，在于可以极大限度的减少对于真实DOM的操作

④组件相关的属性state、props。。

三、React使用

1、基本用法

react.js 核心文件

react-dom.js 处理dom

browser.js 处理浏览器兼容

babel编译器，支持es6新特点转换为普通的js，支持jsx转换为js

ReactDOM.render(arg1,arg2) 将arg1中的内容 插入到arg2容器中去渲染视图。

2、jsx语法

如果渲染多个元素，不允许直接渲染多个，但是可以将要渲染的元素放在一个顶层标签是允许的。

jsx语法：

jsx并不是新的语言，只是js的扩展。

如果遇到{就用js解析，

如果遇到<使用html来解析

编写的jsx语法，浏览器默认是不支持的，但是通过babel编译器，转化为了浏览器能够识别的js。

练习：

新建一个文件，引入react对应的js文件，使用jsx的语法来实现：逻辑运算、三目运算。

3、组件

组件创建和使用

怎么创建？

var Hello = React.createClass({

render:function(){

reutrn 渲染的标签

}

})

怎么使用？

就像使用普通的html标签使用自定义组件

<Hello></Hello>

<Hello/>

注意事项：

①组件名称首字母必须是大写

②返回多个元素 要放在一个顶层标签

③在组件的render方法返回元素时，不允许直接换行

思路切换--》组件化的思维模式：将看到的一切的页面 通过组件的方式来拆分，最终拼接起来构成一个复杂的应用程序

4、复合组件

就是一个普通的组件，组件的内部在渲染的时候，包含其他的组件。

可以类似玩积木一样，创建一个小组件，最终拼在一起，构成复杂的ui。

5、props（properties 属性）

①传值和接收值

在任意一个组件上，指定对应的属性以及对应的值；

如何读取属性中的值？this.props.属性名称

//调用组件并指定属性和对应的值

<PropComponent userName='zhangsan'>

</PropComponent>

//在组件内部通过this.props读取对应的属性的值

var PropComponent = React.createClass({

render:function(){

return <h1>Hello{this.props.userName}</h1>

}

})

练习：使用props来重构登录窗口

props 不仅可以传递普通的值，传递一些方法，就像是传递值传方法

怎么传？？

parentComponent:

handleLogin:func,

name:'zhangsan'

<ButtonCom btnName={this.name}/>

<ButtonCom funcClick={this.handleLogin}/>

怎么接受？

this.props.btnName

this.props.funClick()

②this.props.children

this.props对象中的属性和创建组件时指定的属性 是一一对应的，但是有一个比较特殊

this.props.children:当组件在被调用时，所有的子元素。当有一个子元素时，只返回一个对象；有多个子元素时，返回一个数组；没有任何子元素，返回undefined。

React.Children.map可以帮助来处理this.props.children

6、ref

VirtualDOM(VDOM)：是由各个组件构成一种数据结构。

ul

li 10

li 11

li 12

li 13

/ul

ul

li 09

li 10

li 11

li 12

li 14

/ul

VDOM有一个diff算法，当需要更新视图时，找到VDOM与真实DOM的区别，最后最大限度的减少对真实的DOM的修改。

在组件中，如果要拿到真实的DOM节点，必须是在插入到DOM之后，可以通过ref属性找到真实的DOM节点进行操作。

怎么指定引用？

<customComponent ref='refName'>

怎么通过引用找到元素呢？

this.refs.refName

练习：实现一个组件，组件中有两个输入框，一个求和按钮；点击按钮，将输入的值求和并弹窗显示。

要求：通过ref

综合练习：

A --> input

B --> button

Main:A+B

实现的功能：(5:03 - 5:30)

点击B组件的按钮，能够调用A组件中输入值

A组件和B组件 是兄弟组件要想通信，借助于Main组件

1、完成A组件中用户的输入能够存在main组件

①只要是用户输入，就可以拿到输入的值并打印出来

②如何传递个Main组件

Main组件是可以给A组件传值、传方法

Main

save:function(arg){}

A funcSave={this.save}

A onChange this.props.funcSave()

2、完成B组件按钮点击 读取Main组件中的值

在main中定义一个方法，传递给B组件，按钮点击弹窗显示结果

每日一练：完成综合练习！

## Day02

一、state 状态

初始化状态:返回一个对象的方式来初始化状态

getInitialState:function(){}

修改状态：

this.setState({状态1：值,状态2：值..})

读取状态

this.state.状态名称

例子：开关

React中的state是可以实现单向数据绑定的，将状态中的数据绑定到视图上，同时只要是状态中的值改了，React框架在最合适的时候，采用最优的算法来重新渲染视图。

练习：需要大家通过state来重构综合练习

步骤：将保存的result 通过state来初始化、读、写。

深入组件：自己的生命周期

3个阶段：

mount 挂载到DOM

update 更新

unmount 从DOM中移除

生命周期的处理函数：是在对应阶段会被自动调用的

mount：

componentWillMount

componentDidMount

update

componentWillUpdate(nextProps,nextState)

componentDidUpdate(prevProps,prevState)

unmount

componentWillUnmount

注意事项：

一些标识符像 class 和 for 不建议作为 组件属性名。作为替代，ReactDOM使用className和htmlFor来做对应的属性。

练习：在组件（div）插入到真实的DOM之后，启动定时器修改组件的透明度（0~1）

①指定生命周期处理函数

②初始化状态，读、写状态

③

style="opacity:this.state.状态名称"

==》

style={{opacity:this.state.状态名称}}

二、表单

受控组件：当表单元素指定以下属性，变成了受控组件（用户的输入操作是无效的，交给react来控制）

input value/checked

select selected

三、三连棋

1、组件

square 小方块

board 棋盘

game

（15:50 - 16:05）

①给square指定点击的处理函数 和 要显示的内容

②在board组件指定操作状态的处理函数

③决定输赢

四、ToDoList

1模拟显示

需要创建4个组件

2实现添加功能

①在点击添加按钮时，将输入的数据存在状态中的数组

在todobox中定义状态：数组

在todoInput中点击按钮时，往数组中写入数据（在todobox定义一个修改状态的方法传递给todoinput，去调用）

②显示出来

将状态中的数组中的数据，显示在todolist中

每日一练：

ToDoItem删除功能

①明确一点：状态中的数组已经绑定到了视图，视图元素的增删都是针对状态中的数组

②点击删除按钮，删除数组中指定下标的数据即可

# Vue

## Day01

一、Vue概述

what？

Vue是渐进式的框架，核心库关注视图层

why？

1、Vue继承了ng指令和react组件等优势开发出来的

2、Vue渐进式的框架，官方提供了全套的技术支持

3、Vue 效率比较高 代码组织非常轻松

where？

视图层、spa、部署到服务器提供了非常多的官方工具

二、搭建Vue开发环境

1、CLI

https://github.com/vuejs/vue-cli

(参考：https://github.com/kunyashaw/vue2Examples)

2、引入文件

引入vue.js文件到工程

三、使用vue

1、vue基本用法

el:element

data:数据

methods：方法

directives:

new Vue({

data:

el:})

2、vue自带指令

循环：v-for="临时变量名 in 集合"

v-for="(value,key) in 集合"

选择：v-if/v-else-if/v-else 添加分支处理不同的选 择结果

绑定：v-bind 将变量的值 绑定到组件的 某一个属性

a v-bind:href='linkUrl'

缩写：a :href='linkUrl'

双向数据绑定：

v-model 双向数据绑定的指令，模型数据可以绑定到视图，视图中用户操作的结果绑定到数据

修饰符：

.lazy - 取代 input 监听 change 事件

.number - 输入字符串转为数字

.trim - 输入首尾空格过滤

input v-model.number='num1'

事件绑定：

v-on 后边跟上事件的名称以及对应的处理函数

button v-on:click="handleClick"

button @click="handleClick"

3、vue自定义指令

自定义指令，需要在vue的实例中，定义一个directives,返回一个对象，在对象中完成指令的定义

每一个指令对应着都有自己的钩子函数：

bind

update

unbind

new Vue({

directives:{

change:{

bind:func,//绑定

update:func,//更新

unbind:func,//移除

}

}

})

change 指令的名称，在使用: v-change.

4、过滤器

vue没有自己的官方的过滤器

①方案1：自定义

过滤器的本质其实就是方法（有输入输出）。

过滤器通过管道符号（|）进行调用，支持多重过滤

和ng过滤器区别主要传参：

ng {{price| currency:'￥'}}

vue {{price| currency('￥')}}

new Vue({

filters:{

filterName：function(value,arg1,arg2){

return 过滤后的结果

}

}

})

②方案2：借助第三方的

https://github.com/wy-ei/vue-filter

5、Vue属性

el 将实例挂载到指定的视图

data 初始化

methods 方法

directives 指令

filters 过滤器

computed 计算属性 复杂的业务逻辑

watch 监听

components 定义一些局部组件

计算属性：

new Vue({

computed:{

handleMsg:function(){

}

}

})

与methods，不同的是计算属性是基于它们的依赖进行缓存的。计算属性只有在它的相关依赖发生改变时才会重新求值

监听属性：

watch：{

kw：function(newValue,oldValue){

}

}

6、生命周期

React中生命周期：mount update unmount

Vue的实例生命周期：create mount update destroy

create:

beforeCreate

created

mount:

beforeMount

mounted

update

beforeUpdate

updated

destroy

beforeDestroy

destroyed

7、组件的创建和使用

局部组件的创建和使用：

创建：

Vue({

components:{

'组件名称':{

template:''

}

}

})

使用：就像使用普通的html标签一样

<my-component></my-component>

全局组件：

创建

Vue.component('组件名称'，{

template:''

})

使用：和局部组件用法一样

注意事项：

组件的命名：必须是烤串的写法，要求都是小写。

全局组件和局部组件的区别：

全局组件可以在任意一个组件中去使用，可以构造更复杂的全局组件

局部组件只能够在vue实例所绑定的视图中调用。

按照作用域分类：

创建组件方式：

局部组件

全局组件

按照模板的写法分类：

template:'<h1></h1>'

template:`<h1>

</h1>`

script type='text/x-template' id='myid'

template:'#myid'

通过jsx的语法去指定模板（必须在命令行创建的模板中才可以--因为会有对应的依赖）

直接创建一个以vue结尾的文件（通过命令行创建的模板默认的方式）

React ui层 不等于 样式类的库

使用场景：数据操作非常频繁

<https://github.com/enaqx/awesome-react>

## Day02

注意事项：

在创建的全局组件中，需要初始化或者声明数据，指定下data属性，该属性对应的值是一个方法（有返回一个对象）

一、组件通信

props down ,events up.

1、父与子的数据传递：通过属性

①在子组件指定props，对应的值是数组，数组中的字符串就是将来从父组件传递过来的属性名称

Vue.component('son',{

props:['name']

})

②在父组件调用子组件 指定属性

Vue.component('father',{

template:` <son name='zhangsan'></son> `

})

③在子组件使用传递过来的属性的值

和data属性定义的变量用法是一致

Vue.component('son',{

props:['name'],

template:' <h1>{{this.name}}</h1>'

})

2、子与父通信

通过事件

①给子组件绑定事件

Vue.component('father',{

template:` <son @msgTF="revMsg"></son> `,

methods:{

revMsg:function(msg){

//msg就是在子组件传递来的值

}

}

})

②在子组件中触发事件

this.$emit('msgTF','123');

3、父子通信

父与子通信 $refs

子与父通信 $parent

用法： 通过$parent找到父组件

Vue.component('parent',{

template:' <son></son> '

})

Vue.component('son',{

template:' <button @click="handleClick">

</button>'

methods:{

handleClick:function(){

//与父组件通信

this.$parent.属性或者方法

}

}

})

用法： 通过$refs找到子组件

Vue.component('parent',{

template:`<div>

<son ref='mySon'></son>

<button @click="handleClick">

</button>

</div>`

methods:{

handleClick:function(){

this.$refs.mySon.属性或者方法

}

}

})

Vue.component('son',{

template:' <button> </button>'

data:function(){

return {

msg:'你好'

}

}

})

4、兄弟组件通信

有很多种方案，比如通过共同的父组件；

借助于事件

var bus = new Vue();

//事件的绑定

bus.$on('eventName',function(msg){//使用传过来的参数})

//触发事件

bus.$emit('eventName','it is an arugemnt')

练习：目标实现组件之间的通信

根组件：chat-room:是由列表以及两个聊天组件构成

聊天组件：mike

聊天组件：lucy

二、路由

使用vueRouter插件就可以实现SPA。

ngRoute：

①引入angular-router.js

②在创建自定义模块，指定依赖于ngRoute

var app =angular.module('myApp',['ng','ngRouter']);

③指定一个盛放代码片段的容器

ngView

④创建代码片段

⑤配置路由词典

app.config(function($routeProvider){

$routeProvider

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

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

})

$location.path()

href='/start'

传参：

$routeProvider

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

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

$routeParams.id

uiRouter:

引入angular-ui-router.js

指定依赖ui.router

指定个容器 uiView

创建代码片段

配置状态机

app.config(

function($stateProvider,$urlRouterProvider){

$stateProvider

.state('MikeState',{

url:'/myMike',

tempalteUrl:''

})

$urlRouterProvider.otherwise('/myMike');

})

触发状态：

$state.go('MikeState')

ui-sref="MikeState"

href="/myMike"

传参：

$stateProvider

.state('MikeStat',{

url:'/myMike/:id',

tempalteUrl:''

})

$stateParams.id

发送参数：

$state.go('MikeStat',{id:20});

ui-sref="MikeStat({id:30})"

href="/myMike/20"

VueRouter

引入vue-router.js

指定容器：

<router-view></router-view>

创建各个子组件

配置路由词典

①创建一个对象数组，每个对象指定path/component

(定义路由)

const myRoutes = [

{path:'/myStart',component:testStart},

{path:'/myStart',component:testStart}

]

②通过创建一个VueRouter的实例

const myRouter = new VueRouter({

routes:myRoutes

})

③在vue的实例中指定router属性即可

new Vue({

router:myRouter

})

跳转

<router-link to="/foo">Go to Foo</router-link>

js:

router.push()

a href=''

传参

①detail->order

②配置接收方path

/myOrder --> /myOrder/:id

③接收：

this.$route.params.id

④发送

router-link to='/myOrder/20'

myRouter.push('/myOrder/20')

href='#/myOrder/20'

三、网络请求

this.$http.get/post()

# 微信开发

## Day01

1.魏晓武 weixw@tedu.cn

QQ:3560501078

2.微信 3 1公众号维护开发

2小程序

3.UI、前端和后端

4.为何借助于微信开发或者维护

微信 火 流量大

接口丰富

兼容少 内嵌浏览器 webkit

android ios

运营成本低

4000-5000W 1200W

###团队 技术部 100W

###外包公司 5-10W 维护人员

静态 2000k

动态 CMS系统 500-2000

动态 5-10W

微信站 移动网站+微信接口

移动网站 xwkj.org.cn

###免费建站 u1278384.cnzz.cn

u1275384.cnzz.cn

###微信公众号 300 公共推广的平台

3500W

5.微信公众号是什么?

前端大全

web前端大全

前端早读课

JavaScript

前端开发攻城狮

W3cplus

前端工坊

菜鸟教程

魏武科技

6.如何做公众号

小白公众号运营 前台 软文

公众号开发

mp.weixin.qq.com

6.1注册过程

6.2区别

最低资质 功能 权限

订阅号 个人 阅读1天1条

服务号 企业资质 服务1周1条 支付

企业号 企业

小程序 个人 app 什么都可以

6.3推广 名称 和二维码

名称开始不可以搜索 点击头像 功能设置 通过名称搜索本账号

6.4被添加的发送的信息

6.5资源类型(文本 图片 视频 音频 图文)

6.6主动消息-我想要信息-点击自定义菜单

6.7被动消息-厂家推送

6.8交互消息-关键词

7.微信公众号开发

公众号设置里面--功能设置--js安全域名 购买

域名

http://bjwsl-001-pc/

IP地址 100.34.12.32

http://172.163.100.131 http://119.75.217.109/

域名: 不重复 而且还好记

100.200.101.102.201.101.100.201

预留ip

10.

172.

192.168.

224.

56.23.10.23

空间:公开的U盘

ftp:账号

ftp:用户名

ftp:密码

ip地址

阿里云 万网

腾讯云

51科技

新网

西部数据

ele.me

www.webmake.club

mp3.webmake.club

$$付费

$$认证 身份证正反面 身份证号码 名称 手机号 3-5

营业执照 法人

$$url解析 A记录

www.xwkj.org.cn A指向 IP地址

mp3.xwkj.org.cn

mail.xwkj.org.cn

8.调用微信接口 纯手机网站手机浏览器 微信

webapp都可以调用

@安全域名 空间 ftp

@微信接口是什么？

java封装好的js可以调用的window全局方法

手机网站调用摄像头的代码

调用了微信接口的网站只能在

微信中才能发挥功效

window.camera()

window.getLink()

@appID wx8460529c8128c4f8

appsecret 4caf2b48519951f2b6a6d40a0152fb00

wxf049334e39e3e42b

(appID 、appsecret)-(access)-(ticket)-(signature)

uYo-nh2AP-G5ttm8YPHyqFjKImhRRf9FkOXKtPH0Sosr0lvnVpJM72mpGYwd5p90Zf3qNetZ4dkAC3QZixlfbmAh6UtNSnhAOMfUASstY9yQrSsMz-W6a13Fr77qGS3eCHPgAGASXF

bxLdikRXVbTPdHSM05e5uwztXuhujSoCmzCBPsIZfnK0McgKwib6RkCQvGxFsjWDKpqwxae0qrPushTMr2e6Rw

signature

应用只能2小时

curl插件 发起请求

学习：学而习

learn study 探索

培训学校

传统学校 创新 解决问题 探索能力

微信站 1-2微信接口

1.<script src="http://res.wx.qq.com/open/js/jweixin-1.0.0.js"></script>

2.微信sdk帮助文档

http://mp.weixin.qq.com/wiki/7/aaa137b55fb2e0456bf8dd9148dd613f.html

3.根据基本配置里面的

appid

AppSecret

4.到开发者工具-在线接口调试工具

https://mp.weixin.qq.com/debug

https://api.weixin.qq.com/cgi-bin/token?grant\_type=client\_credential&appid=APPID&secret=secret

生成access\_token

date

5.根据这两个值到

https://api.weixin.qq.com/cgi-bin/ticket/getticket?access\_token=你的access\_token&type=jsapi

生成json数据 ticket也就是jsapi\_ticket

6.到

http://mp.weixin.qq.com/debug/cgi-bin/sandbox?t=jsapisign

进行其他变量的获取

signature

noncestr

timestamp

7.进行配置

8.进行调用

## Day02

1.回顾

微信开发 微信公众号微信站 （PHP js-sdk）

app开发 移动应用开发

网站应用开发

第三方平台开发

微信公众号运行

二次开发 js-sdk 增强型H5网站

不能针对普通浏览器网站

2.小程序

3.是什么?

运行于微信中的代码片段 app级别

小 <1M

页面 5-10页

下载 卸载方便

兼容性好

成本低

流量大

4.区分

@1小程序 京东购物

@2app 微信 QQ 浏览器 游戏

@3webapp 京东 淘宝 手机百度 知乎

@4手机网站 wap.baidu.com m.jd.com 3g.qq.com

微信-我-设置-关于微信 6.5.4

发现-小程序

@1 20 php 前端 设计 1M 1个月

@2 100 java oc 设计 15M-150M 4个月

@3 20 php 前端 设计 10M 2个月

@4 50 php 前端 设计 无 2个月

5.推荐一些小程序

京东购物

饿了么

美团外卖

小小房贷计算器

汇率助手

腾讯房产小助手

查地铁

车来了

摩拜单车

小蓝单车

北京薪资计算器

6.如何开发？

软件 微信开发者工具 qq.com

一路下一步 然后打开软件扫描二维码

选择小程序

添加项目

appID 不填 无

填写名字 选择目录 空

确定

最小网页html body head meta title

7.小程序的构成 4种后缀

wxml weixin markup language

view===div

text===span

image===img

wxss weixin stylesheet

css

js javascript

json json

8. app.js app.json app.wxss

app.json 全局配置

"pages" 键 页面注册 不注册会导致无法连接到此页面

第一个一定是首页

"window" 键 设置页面全局值

"tabBar" 键 下面导航

"networkTimeout"键 超时

"debug" 键 调试状态

app.js 全局事件 全局方法 全局属性

App({}) 调用函数 alert()

onLaunch

onShow

onHide

onError

自定义函数

自定义属性

每个子页面都可以使用,子页面的js文件中

var k=getApp();//script

app.wxss css样式文件 全局样式表

默认每个小wxml都先引入这个文件

//公共样式

//rpx rem 根大小 em父大小

特有单位 rpx 不是标准css3单位

小程序的时候

根px 750rpx 320px

750rpx 414px

DPR

无显示器 纸袋时代 电视机 LED灯 液晶

2500点 50px 50点

1600点

1000px 50px 125点

nokia 80\*80

iPhone 3g 320\*480点

4 320\*480 640\*960

sumsang C5 360\*640 3

meta name="viewport"

content="width=device-width"

设计师依然用真实分辨率来设计图稿，因为手机里面其他界

面依然是真实分辨率

给小程序添加背景颜色 container page上加

9.页面.wxss 页面.wxml 页面.js 页面.json

页面.wxml 雷同html

页面.wxss 雷同css

页面.json 容易

1.要么不写 要么有{}

2. app.json window键里面的值拿过来就可以了

如果页面里面的json设置了，会覆盖app.json

不设置就用app.json里面的设置

页面.js

var a= getApp()//调用app.js事件属性方法

Page({});

data ng变量的值

系统事件 onLoad onShow onHide

自定义事件

自定义属性

<div onclick="chg()">df</div>

<view bindtap="chg">df</view>

10.和前端的对比

@@@没有DOM

@@@没有window

@@@修改innerHTML 结构 this.setData()

@@@修改css style 变量

@@@修改class

@@@this问题

@@@定时器修改

this.setData({abc:"李四"})

宿主 ###浏览器 document window

###node.exe http path fs

###小程序 wx.

transform:rotate({{dd}}deg);

## Day03

1.回顾

微信公众号js-sdk开发

微信小程序---干掉app开发

wxml aaa.wxml

wxss app.wxss aaa.wxss

js app.js aaa.js

json app.json aaa.json

打开pages文件夹父亲就可以

@@互联网定位原理和精确度

###路由器定位 不太准确 得不到

###基站定位 准确

###卫星定位 gps模块 45451415412

45454544545

10m

2.前端开发小程序不爽

@@@ {{aaa}} this.setData({}) 修改innerHTML

@@@ var that=this;

@@@ 修改style

@@@ 修改class

@@@ 没有 DOM BOM

@@@ 获取innerHTML this.data.XXXX

@@@ 定时器 setInterval

只能 setInterval clearInterval

前端 window.setInterval setInterval

@@@ 传参 小程序无法传参

event target currentTarget

n个按钮 功能相似一样 n个方法 1个方法

onclick="chg(0)" function chg(a){ alert(n)}

onclick="chg(1)"

bindtap="chg" data-xx="0"

chg:function(e){

e.target.dataset.xx

}

@@@ 节点操作

appendChild

removeChild

createElement

insertBefore

cloneNode

wxml特殊语法

<view if()你就不显示>你好</view>

###<view wx:if="{{真}}">你好</view>

###<view wx:if="{{真}}">你好</view>

<view wx:else>你好</view>

###<view wx:if="{{真}}">你好</view>

<view wx:elif="{{真}}">你好</view>

<view wx:else>你好</view>

<view wx:if="{{真}}">你好</view>

<block wx:if="{{真}}">

<view>你好</view>

</block>

空标记 不会在页面显示

只控制条件

@@@条件结构 wx:if

@@@循环结构 wx:for

自带两个变量{{item}} {{index}}

万一全局也有这两个变量

我想在循环体内用别的变量替换 index item

那么需要写 wx:for-index="a"

wx:for-item="b"

结构之间可以相互无层深限制的嵌套

@@@ 本地存储 存储或获取

localStorage.getItem()

setItem()

wx.getStorage()

wx.setStorage()

@@@ 事件 e.changedTouches[0].clientX

2048

bindtap 轻轻触摸 onclick

bindlongtap 长按 ondbclick

bindtouchstart 触摸开始 onmouseover onmousedown

bindtouchend 触摸结束 onmouseup

bindtouchmove 触摸移动 onmousemove

bindtouchcancel 触摸取消

bindinput 当输入的时候

3.API的套路

wx.XXXX({

有的接口此处也有数据

success:function(res){},

fail:function(e){},

complete:funcion(e){}

})

4.北京薪资计算器

###注册小程序账号 wxa503da876bc540c8

###修改安全域名 https://你们公司的域名

###写好小程序拷贝一份

###新建一个项目 有APPID的

### 项目 上传操作 用户还是无法搜索到我们到我们小程序

### 登录网站 提交审核（3-7工作日）

### 北京薪资计算器

注意: data变量不一定简单变量

数组 对象

复杂数据类型

ajax请求的网址 本地测试可以是localhost

地址已定义是一个在小程序中写过的安全域名

项目: 模仿一款小程序 写入简历