千峰学习笔记

目录

[学费 1](#_Toc7151_WPSOffice_Level1)

[单词 1](#_Toc11187_WPSOffice_Level1)

[Html专业知识及相关概念 2](#_Toc6768_WPSOffice_Level1)

[1、W3C制定的结构和表现的标准: 2](#_Toc27216_WPSOffice_Level1)

[2、WHATWG: 3](#_Toc983_WPSOffice_Level1)

[3、ECMA制定的行为的标准 3](#_Toc17280_WPSOffice_Level1)

[Html 5 3](#_Toc10427_WPSOffice_Level1)

[1)常规标记（对儿标记、双标记） 3](#_Toc29077_WPSOffice_Level1)

[2)空标记（单标记） 3](#_Toc15835_WPSOffice_Level1)

[CSS 13](#_Toc29501_WPSOffice_Level1)

[选择器 14](#_Toc6251_WPSOffice_Level1)

[a:visited 鼠标访问（点击）后的状态 15](#_Toc23831_WPSOffice_Level1)

[H5页面窗口自动调整到设备宽度，并禁止用户缩放页面 41](#_Toc9696_WPSOffice_Level1)

[Javascript概念 58](#_Toc11467_WPSOffice_Level1)

[Javascript基础 60](#_Toc17189_WPSOffice_Level1)

[Javascript结构 65](#_Toc20982_WPSOffice_Level1)

[if else 65](#_Toc30294_WPSOffice_Level1)

[函数 67](#_Toc878_WPSOffice_Level1)

博客园 三生勿忘~zq68955710819

学费

学费：分期17000（20476.5）前6个月总共722.46元； +1900=22466.5元。

单词

Hyper 透视

Text 文本

Markup 标记

Language 语言

EXtensible 扩展

HyperText 超文本

WHATWG

script 脚本

World

Wide

Web

Consortium 联合

index 索引

cellspacing 单元格之间间隙

cellpadding

align 排列

colspan 合并列

rowspan 合并行

center 居中

width

height

border 边框

href 超链接

table 表格

self 默认

blank 新窗口

form 表格

method 函数

action 动作

password 密码

submit 登录

reset 重置

radio

checkbox 复选框

disabled 使...失去能力

checked 检查

select 查询

option 选项

textarea 文本框

cols 报表

rows 行数

type 类型

button 按钮

span 跨度

value 数值

cascading

style

sheets 层叠样式表

rel

import

link 初始状态 连接

element 元素

visited

hover 滑动时

active 按住时

float 浮动

family

bold 粗体

bolder

normal 正常

italic 斜体

oblique 斜角

line 排列

none 默认

underline 下划线

overline 上

line-through 划掉

indent 缩排

letter 字母

disc

circle 循环

square 方形

position 位置

repeat 平铺

attachment 附件

scroll 卷形

fixed 固定

solid 固体

dashed 虚线

dotted 薄的

double 双倍

微信小程序http://we.wwei.cn/alipay.html

HTML专业知识及相关概念

HTML 5

web前端包含： pc端页面 移动端页面

1：web前端首先要解决用户体验的问题

2：web前端工程师必备条件：

（1）能上网，会打字，懂的互联网是什么

（2）兴趣

（3）肯做练习

3：要学哪些东西：

（1）软件（浏览器（浏览器插件F12），开发工具）

（2）语言:90%的精力

doctype html

<!DOCTYPE html>声明为HTML5文档。

1、<html>、</html>是文档的开始标记和结束的标记。是HTML页面的根元素，在它们之间是文档的头部（head）和主体（body）。

2、<head>、</head>定义了HTML文档的开头部分。它们之间的内容不会在浏览器的文档窗口显示。包含了文档的元（meta）数据。

3、<title>、</title>定义了网页标题，在浏览器标题栏显示。

4、<body>、</body>之间的文本是可见的网页主体内容。

<!DOCTYPE> 声明必须是 HTML 文档的第一行，位于 <html> 标签之前。

<!DOCTYPE> 声明不是 HTML 标签；它是指示 web 浏览器关于页面使用哪个 HTML 版本进行编写的指令。

<span>Doctype</span>作用

定义文档类型，让浏览器解析器知道应该用哪个规范来解析文档。<!DOCTYPE>声明必须在 HTML 文档的第一行，这并不是一个 HTML 标签。

严格模式与混杂模式如何区分？它们有何意义？

1.严格模式：又称标准模式，是指浏览器按照 W3C 标准解析代码。

2.混杂模式：又称怪异模式或兼容模式，是指浏览器用自己的方式解析代码。

3.区分方法：

a.如果文档包含严格的 DOCTYPE ，那么它一般以严格模式呈现。

b.包含过渡 DTD 和 URI 的 DOCTYPE ，也以严格模式呈现，但有过渡 DTD 而没有 URI （统一资源标识符，就是声明最后的地址）会导致页面以混杂模式呈现。

c.DOCTYPE 不存在或形式不正确会导致文档以混杂模式呈现。

d.HTML5 没有 DTD ，因此也就没有严格模式与混杂模式的区别，HTML5 有相对宽松的语法，实现时，已经尽可能大的实现了向后兼容。

DTD:是一套为了进行程序间的数据交换而建立的关于标记符的语法规则。它是标准通用标记语言和可扩展标记语言1.0版规格的一部分，文档可根据某种DTD语法规则验证格式是否符合此规则。

div标签和span标签

div标签用来定义一个块级元素，并无实际的意义。主要通过CSS样式为其赋予不同的表现。  
span标签用来定义内联(行内)元素，并无实际的意义。主要通过CSS样式为其赋予不同的表现。

块级元素与行内元素的区别：  
所谓块元素，是以另起一行开始渲染的元素，行内元素则不需另起一行。如果单独在网页中插入这两个元素，不会对页面产生任何的影响。  
这两个元素是专门为定义CSS样式而生的。

注意：

关于标签嵌套：通常块级元素可以包含内联元素或某些块级元素，但内联元素不能包含块级元素，它只能包含其它内联元素。

认识网页

网页主要由文字、图片和按钮等元素构成。当然，除了这些元素，网页中还可以包含音频、视频以及Flash等。

Web标准

w3c 万维网联盟组织：

定义：用来制定web标准的机构（组织）

web标准：制作网页要遵循的规范

web标准规范的分类：

结构标准；表现标准；行为标准；

结构： html；

表现： css；

行为：javascript；

1、W3C制定的结构和表现的标准:

W3C( World Wide Web Consortium )万维网联盟，创建于1994年是Web技术领域最具权威和影响力的国际中立性技术标准机构。(制定了结构和表现的标准，非赢利性的。)

2、WHATWG:

网页超文本应用技术工作小组是一个以推动网络HTML 5 标准为目的而成立的组织。在2004年，由Opera、Mozilla基金会和苹果这些浏览器厂商组成。

3、ECMA制定的行为的标准

欧洲电脑场商联合会。

HTML相关概念的介绍

HTML 指的是超文本标记语言 (Hyper Text Markup Language)。

XHTML指可扩展超文本标记语言（标识语言）（EXtensible HyperText Markup Language）。“超文本”就是指页面内可以包含图片、链接，甚至音乐、程序等非文字元素。

HTML5指的是HTML的第五次重大修改（第5个版本）。

XML可扩展标记语言。（主要用于数据的处理与表达）了解。

相关准备工作

编辑器的应用DreamWeaver。

PC端调试工具的使用 -浏览器。

创建站点

1)站点的作用:

用来归纳一个网站上所有的网页、素材以及他们之间的联系。

2)创建站点的步骤:

3)文件的命名规则:

文件命名规则：用英文，不用中文

名称全部用小写英文字母、数字、下划线的组合，其中不得包含汉字、空格和特殊字符；必须以英文字母开头。

首页必须命名为index.html。

HTML5

基本语法

1)常规标记（对儿标记、双标记）

<标记 属性=“属性值” 属性=“属性值”></标记>；

2)空标记（单标记）

<标记 属性=“属性值” />；

说明：

1.写在<>中的第一个单词叫做标记，标签，元素。

2.标记和属性用空格隔开，属性和属性值用等号连接，属性值必须放在“”号内。

3.一个标记可以没有属性也可以有多个属性，属性和属性之间不分先后顺序。

4.空标记没有结束标签，用“/”代替。

WHATWG网页超文本应用技术工作小组

ECMA制定的行为的标准: 欧洲电脑场商联合会。

HTML 指超文本标记语言

XHTML指可扩展超文本标记语言

XML指可扩展标记语言

HTML基本语法

1)常规标记

<标记 属性=“属性值” 属性=“属性值”></标记>

2)空标记（单标记）

<标记 属性=“属性值” />

HTML常用标记

1)文本标题

< h1>一级标题</ h1>

< h2>二级标题< /h2>

……

< h6>六级标题</ h6>

2)段落标记

< p>段落文本内容< /p>

3）空格

&nbsp;（所占位置没有一个确定的值,这与当前字体字号还有浏览器都有关系。

4）加粗

<b >加粗内容< /b>

<strong>加粗内容< /strong>

5）倾斜

<em></em>

<i></i>

6） 强制换行

<br />

7）水平线

< h r />

8）列表(ul,ol,dl)

HTML中有三种列表，分别是：无序列表，有序列表，自定义列表

插入图片<img src="目标文件路径及全称" alt="图片替换文本" title="图片标题" />

\*注:所要插入的的图片必须放在站点下:

1.title的作用: 在你鼠标悬停在该图片上时显示一个小提示，鼠标离开就没有了，HTML的绝大多数标签都支持title属性，title属性就是专门做提示信息的.

2.alt的作用:alt属性是在你的图片因为某种原因不能加载时在页面显示的提示信息，它会直接输出在原本加载图片的地方。

10）超链接的应用

语法：

<a href="目标文件路径及全称/连接地址"> 链接文本/图片< /a>

属性：target:页面打开方式

默认属性值：\_ self

属性值：\_blank 新窗口打开

11）数据表格的作用及组成

作用：显示数据

组成：

<table>

<tr>

<td></td>

<td></td>

.........

</tr>

</table>

表单的作用：用来收集用户的信息的;

表单框 ：

<form name="表单名称" method="post/get" action="">

</form>

文本框

<input type="text" value="默认值"/>

密码框

<input type="password" />

提交按钮

<input type="submit" value="按钮内容" />

重置按钮

<input type="reset" value="按钮内容" />

单选框/单选按钮

<input type="radio" name="ral"/>

单选按钮里的name属性必须写，同一组单选按钮的name属性值必须一样。

复选框

<input type="checkbox" name="like" />

<input type="checkbox" name="like" disabled="disabled" />

(disabled="disabled" :禁用)

(checked="checked" :默认选中)

下拉菜单

<select name="">

<option>菜单内容</option>

</select>

多行文本框（文本域）

<textarea name="textarea" cols="字符宽度" rows="行数">

</textarea>

按钮

<input name="'" type="button" value="按钮内容" />

div的用法

<div></div>文档区域，文档布局对象

div是区块，分割的意思,作用：用来把页面划分出不同的区域

span的用法

<span></span>

文本结点（某一小段文本，或是某一个字）

Title和Alt的作用

title: 在你鼠标悬停在该图片上时显示一个小提示，鼠标离开就没有了，HTML的绝大多数标签都支持title属性，title属性就是专门做提示信息的。

Alt :alt属性是在你的图片因为某种原因不能加载时在页面显示的提示信息，它会直接输出在原本加载图片的地方。

post和get的区别

1. get是从服务器上获取数据，post是向服务器传送数据。

2. get是把参数数据队列加到提交表单的ACTION属性所指的URL中，在URL中可以看到。post是通过HTTP post机制，用户看不到这个过程 。

3. get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。

4. get安全性非常低，post安全性较高。但是执行效率却比Post方法好。

button和submit的区别

submit是提交按钮，起到提交信息的作用，button只起到跳转的作用，不进行提交。

常用标记

1)文本标题

<h1>一级标题</h1>

<h2>二级标题</h2>

……

<h6>六级标题</h6>；

属性color red属性值（html属性值是放在双引号里面的）

2)段落标记

<p>段落文本内容</p>

排列方式：自上而下排列的

3）空格

&nbsp;（所占位置没有一个确定的值,这与当前字体字号还有浏览器都有关系。

4）加粗

<b>加粗内容</b>

<strong>加粗内容</strong>

5）倾斜

<em></em>

<i></i>

6） 强制换行

<br />

7）水平线

<hr />

<hr color="red"/>

8）列表(ul,ol,dl)

HTML中有三种列表，分别是：无序列表，有序列表，自定义列表；

type属性 I、a、A、数字 如果不写属性的话，默认是数字；

把标记外层的叫父元素，里面的叫子元素

\*无序列表组成：

<ul>

<li></li>

<li></li>

．．．．．．

</ul>

\*有序列表组成：

<ol>

<li></li>

<li></li>

．．．．．．

</ol>

\*自定义列表

<dl>

<dt>名词</dt>

<dd>解释</dd>

．．．．．．

</dl>

9) 插入图片

<img src="目标文件路径及全称" alt="图片替换文本" title="图片标题" />

注:所要插入的的图片必须放在站点下。

1.title的作用: 在你鼠标悬停在该图片上时显示一个小提示，鼠标离开就没有了，HTML的绝大多数标签都支持title属性，title属性就是专门做提示信息的。

即：提示信息（标题）

2.alt的作用:alt属性是在你的图片因为某种原因不能加载时在页面显示的提示信息，它会直接输出在原本加载图片的地方。

即：提示信息（当图片由于某种原因（路径错误、和谐了、网速比较慢的时候）不能加载的时候，显示在图片的位置）

\*相对路径的写法：

1)当当前文件与目标文件在同一目录下，直接书写目标文件文件名+扩展名；

2)当当前文件与目标文件所处的文件夹在同一目录下，写法如下：

文件夹名/目标文件全称+扩展名；

3)当当前文件所处的文件夹和目标文件所处的文件夹在同一目录下，写法如下：

../目标文件所处文件夹名/目标文件文件名+扩展名。

即：

当当前文件与目标文件在同一目录下，直接书写目标文件文件名+扩展名；

当当前文件与目标文件所处的文件夹在同一目录下，写法如下：文件夹名/目标文件全称+扩展名；

10）超链接的应用

语法：

<a href="目标文件路径及全称/连接地址">链接文本/图片</a>

**属性：target:页面打开方式**

默认属性值：\_self。

属性值：\_blank 新窗口打开

即：

a超链接标记 横向排列的

属性：href 地址（路径）

target="\_blank" 新窗口打开

target="\_self" 默认值

11）数据表格的作用及组成

作用：显示数据

组成： <table>

<tr>

<td></td>

<td></td>

.........

</tr>

</table>

1、当设置在表格上面时，表示整个表格的位置；

2、当设置在投入上面时，表示整行内容位置；

3、当设置在td上面时，表示单元格内容的位置;

4、table布局在以前网页开发中经常使用，现在基本不用；

5、现在使用div+css布局方式来替代table布局，更简单、更快捷、方便。

Eg: <table border="1" width="600" height="200" align="center" cellspacing="0" cellpadding="0" bgcolor="orange">

<tr bgcolor="yellow">

<td>1</td>

<td>2</td>

<td>3</td>

</tr>

</table>

\*数据表格的相关属性 ：

width="表格的宽度"

height="表格的高度"

border="表格的边框"

bgcolor="表格的背景色"

cellspacing="单元格与单元格之间的间距"

cellpadding="单元格与内容之间的空隙"

bordercolor=”表示边框颜色”

水平对齐方式：align="left/center/right"

\*合并单元格属性：

colspan=“所要合并的单元格的列数"合并列；

rowspan=“所要合并单元格的行数”合并行；

表单的作用

表单标记

表单控件

下拉菜单

多行文本域

表单标记

表单的作用：用来收集用户的信息的; 表格当填入内容时会默认进行伸缩;

表格标题：<caption></caption>；

Eg:<!--表格的标题-->

<!--<caption>小说排行榜</caption>-->;

表单框

<form name="表单名称" method="post/get" action=""></form>

1、form属性：

form表单标记 自上而下排列的 作用：收集用户信息。

name 作用：表单的名称。

method 作用：表单的提交（传输）方式 get和post方式（如果设置method属性默认是get方式）。

区别：get不安全 post安全；

action 作用：地址(后台地址-其他地址)；

target 作用：新窗口打开；

2、input标记 横向排列的；

input属性：

name 属性：名称

type 属性：表单类型 type属性的属性值不同决定了表单框类型（功能）的不同

value 属性：初始值

placeholder 属性：初始值（当输入内容的时候，默认的提示会自动消失）；

文本框

<input type="text" value="默认值"/>；

密码框

<input type="password" />；

提交按钮

<input type="submit" value="按钮内容" />；

重置按钮

<input type="reset" value="按钮内容" />；

单选框/单选按钮

<input type="radio" name="ral"/>

<input type="radio" name="ral" />

单选按钮里的name属性必须写，同一组单选按钮的name属性值必须一样。

复选框

<input type="checkbox" name="like" />

<input type="checkbox" name="like" disabled="disabled" />

(disabled="disabled" :禁用)

(checked="checked" :默认选中)；

1、同一组的单选按钮上面的name属性必须设置，而且属性值必须一样

2、不同组的单选按钮的name属性值不能一样

#表示空链接

作用：可以返回顶部；

checked默认选中

eg: checked="checked"

下拉菜单

<select name="">

<option>菜单内容</option>

</select>；

select标签一般用于知道指定数量的 selected 作用：默认选中让其排在第一位-。

多行文本框（文本域）

<textarea name="textarea" cols="字符宽度" rows="行数">

</textarea>；

按钮

<input name="'" type="button" value=“按钮内容” />

（他和submit的区别是 ，submit是提交按钮 起到提交信息的作用，button只起到跳转的作用，不进行提交。）

扩展知识点：post和get的区别

1. get是从服务器上获取数据，post是向服务器传送数据。

2. get是把参数数据队列加到提交表单的ACTION属性所指的URL中，在URL中可以看到。post是通过HTTP post机制，用户看不到这个过程 。

3. get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。

4. get安全性非常低，post安全性较高。但是执行效率却比Post方法好。

13）div的用法

<div></div>

文档区域，文档布局对象

div是区块，分割的意思,作用：用来把页面划分出不同的区域

14） span的用法

<span></span>

文本结点（某一小段文本，或是某一个字）

标记知识点

自上而下排列：p、form、

水平横向排列：b、strong、i 、em、img、a、input、span、

[单词](课程/day1/day1预习单词.txt) [html基础](课程/知识点/day1-HTML基础.ppt) [CSS基础](课程/知识点/DAY3：CSS基础.png)

字体大小：Font-size

eg: <font size="1">字体标记</font>

<font size="6" color="greenyellow">字体标记</font>；

表单：table tr（行） td（列-单元格）

border 边框

width 宽

height 高

cellspacing 指的是单元格与单元格之间的间隙

cellpadding 指的是内容到边框之间的距离

数据表格的相关属性 ：

width="表格的宽度"

height="表格的高度"

border="表格的边框"

bgcolor="表格的背景色"

cellspacing="单元格与单元格之间的间距"

cellpadding="单元格与内容之间的空隙"

bordercolor=”表示边框颜色”

\*合并单元格属性：

colspan=“所要合并的单元格的列数"合并列（左右）；

rowspan=“所要合并单元格的行数”合并行（上下）；

水平对齐方式 ：align="left/right/center"；

align

valign

背景颜色：bgcolor

边框颜色：bordercolor

div标记 (区域，分割)

作用：网页布局（使用div标记可以把网页划分成不同的区域,现在使用最多的就是div+css布局方式）

span标记 文本结点 横向排列的，没有默认样式

让自上而下排列的元素可以水平居中的方法：margin:0 auto；

margin-top/left/right/bottom 上/左/右/下边距；

background-color 背景颜色；

边框颜色border:1px solid red；

float:left/right/none (让竖着的东西横着来)float浮动；；

list-style:none 清除列表符号；

line-height 文本垂直居中；

行高line-height： 就是行与行之间的高度（距离）

行高显示原则：

1、当行高大于容器高度时，文本垂直往下

2、当行高小于容器高度时，文本垂直往上

3、当行高等于容器高度时，文本可以实现垂直居中

text-align:center /justify文本水平居中/两端对齐；

text-decoration:none 去除a的下划线；

实现划过效果：使用：hover；

text-decoration:none 去除超链接的下划线；

导航结构布局： ul li a形式；

margin 外边距（边界） 指的是元素与元素之间的距离

margin:0 auto;让元素水平居中 缩写形式：值1表示上下 值2表示左右 auto表示自动

margin-top 上边距

margin-bottom 下边距

margin-left 左边距

margin-right 右边距（margin上下值会发生重叠）

padding指的是内容到边框直接的距离

padding 两个值 值1表示上下 值2表示左右

padding-left 左填充

padding-right 右填充

padding-top 上填充

padding-bottom 下填充

overflow: hidden超出隐藏

text-indent首行缩进

font-weight: bold / bolder/ narmol加粗/ 更粗/ 取消加粗

font-style: italic /oblique 斜体字 / 倾斜的字

font-style: normal

font-family 字体

text-decoration：underline/ overline / line through/ 下划线/ 上划线 /删除线

list-style-type: circle/ square/ none 列表符号属性

list-style-image: url(images/gt.jpg) 让图片作为列表符号

list-style-position: inside 列表符号的位置

background-image: url(地址) no-repeat x值 y值（center/ left/ right）；插入背景图

background-repeat: no-repeat 背景图平铺

background-attachment: fixed 背景图固定属性

border-width:1px； 边框的宽度

border-style: dashed ；边框的样式（线型）

border-color: red； 边框的颜色

border: none 取消边框

border-left /right 左/ 右边框

border: 字号 类型dotted /dashed /double 颜色； 点画线/ 虚线 /双线（要出现双线边框的宽度必须大于等于3px）。

overflow: inherit 继承

overflow: auto 超出显示滚动条

box-sizing: border-box 把标准盒模型转成怪异盒模型

pre 预格式化 块级标签

code代码标签 专门放置代码的 内联标签（一般在工作中使用pre标签替代code的标签）。

white-space: nowrap强制不换行

把其他标签转成了pre标签\*/

white-space: pre;转换成pre

white-space: pre-wrap;

white-space: pre-line

强制不换white-space:nowrap;超出隐藏overflow:hidden;显示省略号text-overflow:ellipsis

display: inline把块级元素转成内联元素（内联块元素不能转成内联元素 - 因为行内块元素默认有内在的样式）

display: inline-block把块级元素转成了内联块元素

display: block 1、把其他元素转成块级元素 2、表示让一个隐藏的元素让其显示

display: none 元素隐藏

display :list-item 把块级类型转成了列表类型

&copy ©符号

display :table 把块级元素转成了表格类型

[z-index](课程/day9/11-z-index.html) 层级属性 (没有单位)

[锚点语法](课程/day9/17-锚点.html)：

<a href="#名称"></a>

<标签 id="名称"></标签>

vertical-align: top / middle/ bottom文本垂直居中属性 /解决图片间隙

font-style: normal 取消倾斜

vertical-align: top(垂直往上)/bottom（垂直往下）/middle（垂直居中）

[透明](课程/day9/18-透明属性.html) 兼容写法

Opacity: .5；(0-1)

filter:alpha(opacity=50);(1=100)

[滚动字幕](课程/day9/19-字幕滚动.html)：

<marquee> </marquee>

<marquee behavior="alternate"> </marquee>

<marquee direction="right" scrollamount="10">

Novalidate 对表单不进行验证提交 在H5中的属性，如果属性和属性值一样的话，属性值可以省略

<form action=" " target="\_blank" method="get">

智能表单 - 就是表单自身增加了验证功能

<input type="email" value="" />-

<input type="text" name=" " id=" " value=" " disabled/>-

[单行表单名称](课程/day12/03-表单.html)

<fieldset >

<legend> </legend> </fieldset>

[状态栏图标](课程/day13/01-状态栏图标.html)

<link rel="shortcut icon" href=" "/>

文档的某个区域

<section> </section>

[添加视频](课程/day13/05-多媒体标签.html)

<embed src=" "> </embed>

[地址标签](课程/day13/08-标签.html)

<address>武侯区科华北路力保大厦</address>

标题

<figcaption> </figcaption>

画布

<canvas> </canvas>

CSS

样式表

样式表包括内部样式表、外部样式表、内联（行内）样式表。

一、内部样式表:

type属性 作用：定义文档类型 type属性可以省略，但是不能写错，代码执行顺序：从上往下执行，从左到右执行。

样式表：<style type="text/css"></style>；

二、外部样式表:

1、rel表示关联样式表 type表示定义文档类型（可以省略，但是不能写错） href表示路径

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

2、@import url(css/style.css)；

三、内联（行内）样式表：双引号里面放置的是css代码。

Eg: <h1 style="color:green;">文本</h1>。

样式表的优先级

1、内联样式表的优先级最高；

2、内部样式表和外部样式表的优先级和书写顺序有关，后面的会把前面的给覆盖掉。

3、优先级覆盖的是相同的样式，不相同的属性依旧会正常显示，同一个外部样式表可以给多个html文件引用。

选择器

Class选择器

class选择器：（10）

语法：

css: .名称{ }；

html: <标签 class="名称"></标签>；

标签选择器或者叫类型选择器，把标签名选择出来就行,相同的标签样式都起作用。

ID选择器

id选择器：

唯一性、 一个标签只能对应有一个id名。（100）

语法：

html：<标签 id="名称"></标签>；

css： #名称{ }；

包含选择器

包含选择器（后代选择器）：

作用：可以限制样式的范围。

语法:

标签 标签{ }；

群组选择器

群组选择器：

作用：可以一次性有范围的选择多个；

语法：

标签1,标签2,标签...{ } ；

注意点:最后一个标签不能写逗号。

通配符

作用：重置样式（重新把默认的样式给清掉重新设置

语法:

\*{ }；

星号表示所有（all）。

重置样式：

\*{margin:0;padding:0;}

有些标签会自带默认的样式，不利于开发需要把它给清除掉，一个个去清除很麻烦，这个时候就可以使用通配符

margin 外边距 元素与元素之间的距离

padding内边距 元素内容到边框直接的距离

子选择器

>

只选择子元素

交集选择器

只对一类元素起名起作用限制了范围

div.box{

background-color: yellow;

}

class 包含（后代） 子选择器

序列选择器

nth-child( )不建议外层使用 不能被其他元素给隔开；

伪类选择器

伪类选择器（类似选择器）

伪类选择器是超链接里面的一种：

a:link 初始状态

a:visited 鼠标访问（点击）后的状态

a:hover 鼠标划过（悬停）时的状态 - 使用最多的是hover，因为hover可以给其他标签使用

a:active 鼠标激活（鼠标按着不动）时的状态

注意点：

1、鼠标访问过后的状态，有浏览器缓存问题，只有把缓存清除才可以回到初始状态。

2、就近原则。

权重

选择器默认都有权重 - 相对于人的体重 - 每个选择器都有不同的权重。

id 100 class 10 类型 （标签）1

优先级的权重 样式表叫优先级 选择器叫权重；

作用：

1、当发生冲突的时候，高的覆盖低的；

2、无论是优先级还是权重，覆盖的都是相同属性的；

3、超链接文本颜色不能继承父元素的。

背景图：

背景图 默认情况下背景图会平铺

背景显示原则：

1、当背景图大于容器大小时，背景图的某些部分显示不完整

2、当背景图小于容器大小时，背景图会平铺

3、当背景图等于容器大小时，刚好显示下

background-image:url(images/s8.png);

/\*背景平铺属性 repeat(平铺) no-repeat(不平铺) repeat-x(横向平铺) repeat-y（纵向平铺）\*/。

background-repeat:no-repeat;

/\*背景图的位置属性 left/right/top/bottom/center 具体的数值 值1表示水平 值2表示垂直\*/。

使用浮动布局

1、浮动问题：高度塌陷

需求：在实际的网页开发中，需要容器的高度自适应，父元素不设置高度，让子元素撑开。需要使用浮动属性(在标准流中子元素可以把父元素的高度撑开，但是浮动流不能把容器的高度撑开)，这个就发生高度塌陷

2、解决问题：清除浮动 - 所谓的清除浮动是清除浮动的负面影响 - 高度塌陷问题

清除浮动的方法：

1、使用overflow:hidden;属性 给父元素设置 1、原来作用：文本溢出隐藏 2、作用：清除浮动

2、使用空（盒子）元素的方法 在需要清除浮动的元素同级下加一个空元素（这个元素不能有 其他任何样式）设置clear:both()这个使用最多/right/left/none;属性。

在程序中数字和字母不能自动换行，英文单词可以换行。

什么时候清除？出现高度塌陷的时候清除，平时不需要清除。

字体设置

字体可以设置多个，多个字体之间是有逗号连接；

font-family:"微软雅黑","黑体","Times New Roman",Aria；

font-size 在浏览器中默认的字体大小是16px；

在谷歌浏览器下默认支持最小的字体大小是12px；

在谷歌浏览器下默认的字体是微软雅黑；

同一文本大小，由于字体的不同，呈现的效果不一样；

9pt = 12px 30px = ?pt 30\*0.75；

3:4 分数化到最简 0.75；

em是相对于父元素的单位 变化的值；

盒模性

无论什么类型的元素，都有一个盒模型 - 盒模型是css布局的一个基础或者叫基石

盒模型的组成：border（边框）、margin（边界、外边距）、padding（填充、内边距、补白）、content（宽和高）

盒模型规定了网页中的元素如何显示以及元素相互之间的关系；

margin 外边距 指的是盒子与盒子之间的间距

padding 内边距（填充）指的是盒子内容到边框之间的距离；

盒子内容 - 文本、元素

使用区别：

1、当元素内容是文本的时候，必须使用padding

注意点：当使用padding时，会把元素的原有大小给改变，如果不想改变元素的原有大小，又想出现效果必须从width和height减去相应的padding值 当元素没有设置width和高的时候，paddin的值不需要减去

2、当元素的内容是元素时，这个时候既可以使用margin也可以使用padding，怎么选择？那个简单使用那个

盒模型的大小指的是一个元素在页面中真正呈现的大小

w3c标准的盒模型：

盒模型宽 = 左右margin + 左右padding + 左右border + width

IE盒模型：

盒模型宽 = 左右margin + width（padding + border）

怎么把标准盒模型转成IE（怪异）盒模型？

box-sizing:border-box(怪异盒模型)/content-box（标准盒模型）; 这是css3新增的盒模型属性

元素类型

大方向元素类型划分：

1、块级元素

2、内联元素（内联元素、内联块元素）

比较标准的划分：

1、块级元素 独占一行、自上而下排列的、可以设置宽和高、一般作为其他元素的容器

2、内联元素 横向排列的，不能设置宽和高（标准流中不能设置宽和高 - 脱离文档流就可以设置宽和高）

3、内联块元素 横向排列的、可以设置宽和高

元素类型本身的注意点：

1、一般使用div去嵌套其他元素 - div网页布局，划分网页区域的

2、p标签里面不能嵌套其他块级元素

3、内联元素使用margin-top或者padding-top有问题，不起作用或者不规范。

元素隐藏

1、中文和英文必须找两个元素放置

2、无论是中文还是英文都能点击（一个连接），中文就是英文的翻译

3、先把em的默认样式给取消，让其先隐藏 - display:none;

4、当鼠标划过li时，让它下的em出现（display:block;），让span隐藏

[链接](课程/day8/11-demo.html)

定位

一、定位属性的介绍：

1、position 定位属性(告诉浏览器这是定位属性) - 什么是定位？就是把元素定那个位置上

2、语法：position：static（默认值 - 静态定位） /absolute（绝对定位）/relative（相对定位）/fixed（固定定位）

在标准流中，元素排列 - 根据元素类型的不同，排列方式不同 - 自上而下、横向排列

需求？需要元素层叠排列、元素固定到某个位置不动。。。使用定位属性可以做到

二、定位

1、什么是定位流？就是给元素设置了position属性的，然后成为定位流

2、定位流和标准流不是一个层次上面的，可以理解为在标准流之上

3、定位属性必须配合top、left、right、bottom来使用

4、定位属性可以层叠 - 定位有层级关系

5、定位属性层级关系 - 后面会把前面的给覆盖 - 后来居上

三、相对定位的特点

1、相对定位 不脱离文档流的，占空间的（占位置）、区分元素类型的 - 就是元素之前是什么类型，设置了相对定位还是什么类型

2、相对定位是相对于自己以前在标准流中的位置进行定位

3、对元素进行微调、子绝父相（父元素设置相对定位、子元素设置绝对定位）

四、绝对定位的特点：

1、绝对定位 脱离文档流的，不占空的、不区分元素类型的

2、相对于祖先元素进行定位

注意点：绝对定位去相对于谁定位时，默认先去上一级，如果有定位流就相对于定位，如果没有继续往上找，直到找到为止。

如果它的上级都没有定位流，就以body进行定位。

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>无标题文档</title>

<style>

\*{margin:0;padding:0;}

/\*子绝父相\*/

.box{

width:300px;

height:300px;

background:blue;

margin:50px auto;

position:relative;

}

.box div{

width:100px;

height:100px;

background-color:yellow;

position:absolute;

left:0;

bottom:0;

right:0;

top:0;

margin:auto;

}

</style>

</head>

<body>

<div class="box">

<div></div>

</div>

</body>

</html>

[几张图片和合在一起](课程/day9/09-demo.html) [在导航标题单个添加小图片](课程/day9/10-demo.html)

H5

1、H5是不断发展和完善的，增加了语义化的标签 header/nav/main/footer... 强大的功能（多媒体标签）视频、音频。。。

2、H5新增的一些标签和属性，让编程简单化了

3、DTD规定了标记语言的规则，这样浏览器才能正确地呈现内容（记忆）

4、SGML标准通用标记语言，是一种定义电子文档结构和描述其内容的国际标准语言（记忆）

5、扩展：语义化的重要性

① 当页面加载失败的时候，还能够呈现出清晰的结构

② 有利于SEO优化，利于被搜索引擎收录（即便于网络爬虫的识别）

③ 在项目开发及维护时，语义化的也很大程度上降低开发难度，节省成本

<!--<input type="text" placeholder="请输入你的用户名"/>

HTML5基础

文档类型：过渡型、严格型、框架型、手机浏览器DTD mobile；

当不对doctype定义时会触发怪异模式，

特点：更简洁、但是实际开发中要注意书写规范，利于后期维护；

语义化的重要性

1、当页面加载失败是还能呈现清晰的结构

2、有利于SEO优化利于网络爬虫的识别

3、在项目开发及维护时，语义化也很大程度降低开发难度节省成本

4、语法跟宽松

5、多设备跨平台

6、自适应网页设计

新增语义化标签

header头部、nav导航、section章节页眉栏目、artcle文章、aside侧边栏、footer页脚、figure文档插入图像、video定义视频（ogg、MPEG4、WebM）、audio音频、canvas图像

表单新增元素

email

URL 地址

number

date pickers 时间

range 最大最小值

H5新增表单验证

placeholder 文本框显示

autofocus 焦点

required 验证不能为空

list 结合datalist使用

自动验证 max min step数字间隔

不自动验证 novalidate 给form添加

多媒体播放

source标签是路径 src路径 type文件类型

object对象标签 作用：放flash .swf表示flash

<video width="800" height="" controls="controls">

<source src="video/movie.mp4" type="video/mp4"></source>

<source src="myvideo.ogv" type="video/ogg"></source>

<source src="myvideo.webm" type="video/webm"></source>

<!--给低版本浏览器提示作用-->

<!--当前浏览器不支持 video直接播放，点击这里下载视频： <a href="myvideo.webm">下载视频</a>-->

src 路径

controls 视频控件属性

loop 循环播放

autoplay 自动播放

muted 静音

poster 封面

视频标签只支持这三种格式：

Ogg - ogg

MPEG4 - mp4

WebM - webm

<!--在谷歌浏览器下进度条不兼容 autoplay属性在谷歌浏览器下禁用了-->

<video src="video/movie.mp4" controls="controls" poster="bjk.jpg"></video>

<video>

<!--音频video标签支持三种格式的-->

<source src="myvideo.mp4" type="video/mp4"></source>

<source src="myvideo.ogv" type="video/ogg"></source>

<source src="myvideo.webm" type="video/webm"></source>

H5表单

1、在H5里面新增的表单，比普通表单多了验证功能（内置在表单上面的）和属性

2、在工作中怎么做表单验证呢？使用js里面的正则表达式

multiple 多选属性

required 表单为空不能提交

min 最小值是多少

max 最大值是多少

step 间隔值是多少

maxlength 表单允许输入最大字符数

邮箱:<input type="email" value="" multiple="multiple"/>

地址:<input type="url" value="" />

<form action="">

<!--取色器-->

<input type="color" name="" id="" value="" />

<!--搜索-->

<input type="search" value="" />

<!--图像按钮-->

<input type="image" src=""/>

<!--定义用于输入电话号码的字段-->

<input type="tel" name="" id="" value="" />

<!--定义隐藏输入字段。-->

<input type="hidden" name="" id="" value="" />

</form>

input 类型是 HTML5 中的新类型：color、date、datetime、datetime-local、month、week、time、email、number、range、search、tel 和 url。

表格标签

单元格间距border-spacing

合并相邻单元格边框border-collapse：separate、collapse

无内容单元格的设置empt-cells：show、hide

隔行变色 双行even、2n；单行odd、2n+1；

表格标题：<caption><caption>

CSS层叠

1、什么是css层叠？指的是样式表的优先级，当发生冲突的时候以优先级高的为标准

2、无论是css2还是css3都是样式语言，只是css3在css2的基础上增加了很多强大的功能而已（动画、2D、3D。。。）

3、网站优化分为两个部分： 技术手段 花钱 结合起来使用

4、CSS新特性：

CSS3选择器、图片的视觉效果（圆角、阴影、渐变背景、图片边框等）、背景的应用（background-size）、盒模型的变化、阴影效果（文本阴影、盒子阴影）、

多列布局和弹性盒布局、web文字和font图标、颜色和透明度、圆角和边框的新特效、2D和3D变形、CSS3过渡和动画效果、媒体查询和Responsive布局。

属性选择器：

E[attr]：只使用属性名，但没有确定任何属性值

E[attr="value"]：指定属性名，并指定了该属性的属性值

E[attr~="value"]：指定属性名，并且具有属性值，此属性值是一个词列表，并且以空格隔开，其中词列表中包含了一个value词，而且等号前面的“〜”不能不写

E[attr$="value"]：指定了属性名，并且有属性值，而且属性值是以value结束的

UI状态伪类选择器：

元素 ::selection

可以让鼠标选中文字的时候，给其设置背景和文本颜色，不能设置其他的属性。 选择器必须是两个冒号

p::selection

给处于可用状态的元素设置样式

input:enabled

给处于禁用状态的元素使用

input:disabled

给处于选中状态的元素使用；

CSS

CSS3是一种语言样式，告诉浏览器是如何渲染的，除上述css3被划分为几个模块组并且都有哦自己的规范，不会因为部分印象其他模块的推进

优势：

让页面更加炫酷用户体验高；利于开发维护提高网站性能，适配跟多的设备，利于SEO网站优化，提高网站搜索排名；

新特性：

CSS3选择器、图片的视觉效果（圆角、边框、阴影）、背景应用background-size、多列布局和弹性盒布局、2D 3D 过渡和动画效果；

选择器：

属性选择器：E[attr ?=" value "]

~ 词列表，以空额隔开

^ 以....开头的

$ 以....结束的

\*包含了某元素的；

伪类选择器

1、UI元素状态伪类

enabled可用状态、disabled禁用状态、checked选中状态

2、CSS3序列选择器

：first-child

：last-child

：nth-child(n)

：first-child

：nth-last-child(n)

: only-child

3、否定选择器

：not

4、：root选择文档的根元素s

1、后代选择器（E F）

2、子元素选择器（E > F）

3、相邻兄弟选择器（E + F）

4、通用兄弟选择器（E ~ F）

组成：选择器和声明（属性、属性值）

样式：

外部样式、内部样式、内联样式

样式的优先级

内联样式优先级最高；内部样式表和外部样式表与书写的顺序有关，后书写的样式最高。

CSS选择器

元素/类型选择器：元素名称{属性：属性值}

ID选择器：#ID名{属性：属性值’}

class选择器： . {属性：属性值}

通配符：\*{属性：属性值}

群组选择器: 选择符1，选择符2{属性：属性值}

包含选择器： 选择器1 选择器2{属性：属性值}

伪类选择器：

a:link{ }初始

a:visited{　　}访问后

a:hover{ }鼠标悬停

a:active{ }鼠标按下时

CSS文本属性

（浮动 10+背景颜色背景图）

文本大小、颜色

font-size:value; 单位：px(像素 ) pt (打印) 1em=16px 1px=0.75pt （em>pt>px）

文字倾斜

font-style : italic / oblique /normal

文字加粗

font-weight：bold /bolder

水平对齐方式

text-align: left /right /center

文字行高

line-height:normal /value 文字缩写形式：font:字号，行高，字体

文本修饰

text-decoration: none /underline 下划线 / overline 上划线 / line-through添加删除线

首行缩进

text-indent:value;

字间距

letter-spacing:value

定义列表符号样式：list-style-type:disc(实心圆) /circle（空心圆/）square(实心方块)/none（去列表符号）

。。。

。。。

元素类型

1、块状元素

自上而下，块状元素会占据一行不会出现并列情况；并且可以设置width和height；是其他元素的容器。自上而下

eg: div dl , dd, dt, ol, ul , fieldset, h1-h6 ,p, form ,hr,colgroup,col,table,tr,td;

2、行内元素

始终以行内逐个显示，不能设置width和height；横向；

eg; a, span, i ,em ,strong,

3、内联块状元素

其他元素都在一行，height和width可以设置；横向；

eg ; img, input，select;

图片格式

jpg / gif / png

Link和import的区别

Import老浏览器不支持，只支持IE5以上的浏览器；

Link属于XHTML的标签，import是css提供的一种方式；

Link应用的标签会同事加载，而import会等到页面下载完后再加载；

清除浮动

1. 添加空盒子clear：both；

2. Overflow：hidden；

3. 万能清除法；

clear:after{display:block; clear:both;content:" ." visibility:hidden;height:0;}

clear{zoom:1}

浮动只能清楚元素本身，不能印象其他元素。

可为负值的属性

Text-indent 字间距

Letter-spacing 首行缩进

省略号

White-space：norwap强制不换行

overflow-hideen 超出隐藏

text-overflow:ellipsis显示省略号;

……要实现单行文本溢出时产生的省略号的效果还须定义：

1、容器宽度

2、强制文本行内显示white-space: nowrop

3、溢出隐藏overflow：hidden

4、显示省略号text-overflow:ellipsis;

透明的写法

标准写法：Opacity: ( )0-1;

IE写法：Filter:alpha(Opacity:value)1-100；

隐藏写法

Visibility：hidden；

Display:none;

Opacity:0;

文字隐藏：text-indent

数据分组

数据行分组：thead tbody tfoot;

数据列分组：<colgroup> <colgroup>； <col>；

图片整合技术（CSS精灵图）

定义：将导航背景图片，按钮背景等有规则的合并成一张图片，然后利用background-position来实现背景图片的定位技术

优点：

1、通过图片整合来减少对服务器的请求次数从而提高页面加载速度；

2、通过图片整合来减少图片体积。

实现技巧：

使用ps进行小图片整合；

宽高自适应

宽度自适应：元素宽度设置为100%；width：100%；

高度自适应：height：100%；高度不写

自适应窗口：html，body{width100%，height：100%}

三角形

| | <!doctype html>

| | <html>

| | <head>

| | <meta charset="utf-8">

| | <title>无标题文档</title>

| | <style>

| | div{

| | width:0;

| | height:0;

| | border-top:50px solid #fff;

| | border-left:50px solid orange;

| | border-right:50px solid #fff;

| | border-bottom:50px solid #fff;

| | }

| | </style>

| | </head>

| |

| | <body>

| | <div></div>

| | </body>

| | </html>

五大浏览器内核

Trident（MSHTML）IE 遨游、腾讯、360

Gecko：Linux、 MacOS X 开源

Presto：opera

Webkit ： safari、chrome 开源

blink: Google

浏览器兼容问题：

浏览器由不同的浏览器厂家开发，所用核心架构和代码也很难重和；各大厂商处于自身利益考虑而设置的种种技术壁垒让CSS应用起来麻烦。

浏览器格式

OGG、MPEG4、WebM

常见的CSSbug和CSShack

1、图片间隙

display：block

2、dt ,li中图片间隙,

将img转为快长元素添加声明display：block

3、双倍浮向

display：inline

4、图片在IE中有蓝色边框

给imgde边写成0，img{border：0}

5、默认高度

IE6及以下版本部分元素快有默认高度

font-size：0；overflow：hidden；

6、表单元素距离顶部间距不一致

float：left‘

7、按钮元素默认大小不一

统一大小，input外边套一个标签

8、百分比BUG clear：right；

9、鼠标指针BUD

corsor:pointer;

10、子元素没有设置浮动，设置margin-top属性后会错误的吧margin-top的属性添加给父元素

overflow:hidden;

私有前缀

wedkit 谷歌

moz 火狐

ms- IE

o 欧朋

HTML重要属性

colspan 合并列 （左右）

rowspan 合并行 （上下）

valign=top、bottom、middle、baseline垂直对齐方式

rules=rows、cols、all、none添加分割线

媒体查询

优雅降级

渐性增强

定义：

媒体查询可以让我们根据设备显示器的特性（如视口宽度、屏幕比例、设备方向：横向或纵向）为其设定CSS样式，媒体查询由媒体类型和一个或多个检测媒体特性的条件表达式组成。媒体查询中可用于检测的媒体特性有 width 、 height 和 color （等）。使用媒体查询，可以在不改变页面内容的情况下，为特定的一些输出设备定制显示效果。

结构：

@media all and (min-width:320px) {

div{ background-color:blue;}

}

​ \*表示设备宽度小于320px时就不执行蓝色背景了

screen 是媒体类型里的一种，CSS2.1定义了10种媒体类型；

and 被称为关键字，其他关键字还包括 not(排除某种设备)，only(限定某种设备)；

(min-width: 400px) 就是媒体特性，其被放置在一对圆括号中。

默认样式 ：

注意：默认样式要写在最前面

/\* 打印样式 \*/

@media print {}

/\* 手机等小屏幕手持设备 \*/

@media screen and (min-width: 320px) and (max-width: 480px) {}

/\* 平板之类的宽度 1024 以下设备 \*/

@media only screen and (min-width: 321px) and (max-width: 1024px) {}

/\* PC客户端或大屏幕设备: 1028px 至更大 \*/

@media only screen and (min-width: 1029px) {}

/\* 竖屏 \*/

@media screen and (orientation:portrait) {对应样式}

/\* 横屏 \*/

@media screen and (orientation:landscape){对应样式}

CSS2 Media用法

其实并不是只有CSS3才支持Media的用法，早在CSS2开始就已经支持Media，具体用法，就是在HTML页面的head标签中插入如下的一段代码

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

想知道现在的移动设备是不是纵向放置的显示屏，可以这样写：

<link rel=“stylesheet” type=“text/css” media=“screen and (orientation:portrait)” href="style.css">

第一段的代码也用CSS2来实现，让它一样可以在页面宽度小于960的执行指定的样式文件：

<link rel="stylesheet" type="text/css" media="screen and (max-width:960px)" href="style.css">

animation和transition的区别

相同点：都是随着时间改变元素的属性值。

不同点：

transition需要触发一个事件(hover事件或click事件等)才会随时间改变其css属性；

而animation在不需要触发任何事件的情况下也可以显式的随着时间变化来改变元素css的属性值，从而达到一种动画的效果，css3的animation就需要明确的动画属性值。

响应式网页设计

固定布局：以像素作为页面的基本单位，不管设备屏幕及浏览器宽度，只设计一套尺寸；

可切换的固定布局：同样以像素作为页面单位，参考主流设备尺寸，设计几套不同宽度的布局。通过识别的屏幕尺寸或浏览器宽度，选择最合适的那套宽度布局；

弹性布局：以百分比作为页面的基本单位，可以适应一定范围内所有尺寸的设备屏幕及浏览器宽度，并能完美利用有效空间展现最佳效果；

混合布局：同弹性布局类似，可以适应一定范围内所有尺寸的设备屏幕及浏览器宽度，并能完美利用有效空间展现最佳效果；只是混合像素、和百分比两种单位作为页面单位。

布局响应：对页面进行响应式的设计实现，需要对相同内容进行不同宽度的布局设计，有两种方式：pc优先（从pc端开始向下设计）；移动优先（从移动端向上设计）；无论基于那种模式的设计，要兼容所有设备，布局响应时不可避免地需要对模块布局做一些变化（发生布局改变的临界点称之为断点），

布局方案

（1）模块中内容：挤压－拉（布局不变）

（2）模块中内容：换行－平铺（布局不变）

（3）模块中内容：删减－增加（布局不变）

（4）模块位置变换（布局改变）

（5）模块展示方式改变：隐藏－展开（布局改变）

（6）模块数量改变：删减－增加（布局改变）

设计特点：

面对不同分辨率设备灵活性强

能够快捷解决多设备显示适应问题

缺点：

兼容各种设备工作量大，效率低下

代码累赘，会出现隐藏无用的元素，加载时间加长

其实这是一种折中性质的设计解决方案，多方面因素影响而达不到最佳效果

一定程度上改变了网站原有的布局结构，会出现用户混淆的情况

设置meta标签

Meta标签

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

参数解释：

width = device-width：宽度等于当前设备的宽度

initial-scale： 初始的缩放比例（默认设置为1.0）

minimum-scale：允许用户缩放到的最小比例（默认设置为1.0）

maximum-scale：允许用户缩放到的最大比例（默认设置为1.0）

user-scalable：用户是否可以手动缩放（默认设置为no，因为我们不希望用户放大缩小页面）

标签设置：

H5页面窗口自动调整到设备宽度，并禁止用户缩放页面

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

忽略将页面中的数字识别为电话号码

<meta name="format-detection" content="telephone=no" />

忽略Android平台中对邮箱地址的识别

<meta name="format-detection" content="email=no" />

当网站添加到主屏幕快速启动方式，可隐藏地址栏，仅针对ios的safari

<meta name="apple-mobile-web-app-capable" content="yes" />

<!-- ios7.0版本以后，safari上已看不到效果 -->

将网站添加到主屏幕快速启动方式，仅针对ios的safari顶端状态条的样式

<meta name="apple-mobile-web-app-status-bar-style" content="black" />

<!-- 可选default、black、black-translucent -->

渐变

语法：

background: linear-gradient(direction, color-stop1, color-stop2, ...);

说明：

direction：默认为to bottom，即从上向下的渐变；

stop：颜色的分布位置，默认均匀分布

种类：

线性渐变、径向渐变、重复渐变

CSS3 过渡

transition-property：检索或设置对象中的参与过渡的属性

transition-duration：检索或设置对象过渡的持续时间

transition-delay：检索或设置对象延迟过渡的时间

transition-timing-function：检索或设置对象中过渡的动画类型

简写：transition:all(全部或所有)/具体属性值 运动时间s/ms 延迟时间s/ms 动画类型

linear匀速 ease逐渐慢下来 ease-in加速 ease-out 减速 ease-in-out先加后减

CSS3 2D

transform 该属性允许我们对元素进行旋转、缩放、移动

1、translate ( )

将元素向指定的方向移动，类似于position中的relative。

如：transform: translate(100px, 0px);

第一个参数:水平方向

第二个参数:垂直方向

2、scale()

让元素根据中心原点对对象进行缩放。默认的值1。因此0.01到0.99之间的任何值，使一个元素缩小；而任何大于或等于1.01的值，让元素显得更大。

如：transform: scale(0.5, 0.5);

第一个参数:水平方向

第二个参数:垂直方向

注意点:

如果取值是1, 代表不变

如果取值大于1, 代表需要放大

如果取值小于1, 代表需要缩小

如果水平和垂直缩放都一样, 那么可以简写为一个参数

transform: scale(1.5);

3、rotate()

旋转rotate()通过指定的角度参数对元素根据对象原点指定一个2D旋转。它主要在二维空间内进行操作，接受一个角度值，用来指定旋转的幅度。如果这个值为正值，元素相对原点中心顺时针旋转；如果这个值为负值，元素相对原点中心逆时针旋转。

注意点:

1.如果需要进行多个转换, 那么用空格隔开

2.2D的转换模块会修改元素的坐标系, 所以旋转之后再平移就不是水平平移的 如：transform: rotate(45deg) translate(100px, 0px) scale(1.5, 1.5);

rotateX() 方法，元素围绕其 X 轴以给定的度数进行旋转

rotateY() 方法，元素围绕其 Y 轴以给定的度数进行旋转

rotateZ() 方法，元素围绕其 Z轴以给定的度数进行旋转，默认就是Z轴旋转。

perspective: 数值；透视属性：近大远小

transform-origin

transform-origin是变形原点，也就是该元素围绕着那个点变形或旋转，该属性只有在设置了。transform属性的时候起作用；

第一个参数:水平方向

第二个参数:垂直方向

注意点 取值有三种形式

具体像素 transform-origin: 200px 0px;

百分比 transform-origin: 50% 50%;

特殊关键字 transform-origin: center center;

transform-origin: left top;

CSS3 3D

Translate3d（x.y.z）

Transfrom-style：preserve-3d

立方体：1.先平移在旋转2.先旋转在平移（会改变主轴）。

动画三要素：

动画时间：animation-duration

动画名称：animation-name

动画执行体：@keyframes play{from{}to{}}

动画类型：animation-iteration-count

相对定位

position-relative

特点：

1、相对定位不脱离文档流会占用空间；

2、给相对定位设置margin、padding时会影响标准流布局；

3、区分块状、行内、内联块状元素

作用：

1、用于元素进行微调；

2、配合绝对定位使用；

绝对定位

特点：

1、相对定位脱离文档流占用空间；

2、无论有么有父元素都会以body作为参考点定位；

3、不区分块状、行内、内联块状元素；

定位元素层级属性

z-index:auto/ number

说明：较大数值的对象会覆盖较小的对象上，如果两个绝对定位对象的此属性具有同样的值，那么将依据他们在html文档声明的顺序层叠，此属性仅仅作用于position属性值relative或absolute，fixed对象。

粘性定位

position: sticky; 粘性定位

粘性定位是相对定位和固定定位的混合。元素在跨越特定阈值前为相对定位，之后为固定定位。

屏幕名称

分辨率，PPi,DPI,DPR,物理像素，逻辑像素

屏幕尺寸：指的是屏幕对角线的长度

分辨率：是指宽度上和高度上最多能显示的物理像素点个数

点距：像素与像素之间的距离，点距和屏幕尺寸决定了分辨率大小

PPI:屏幕像素密度，即每英寸(1英寸=2.54厘米)聚集的像素点个数，这里的一英寸还是对角线长度

DPI:每英寸像素点，印刷行业术语。对于电脑屏幕而言和PPI是一个意思

设备像素(又称为物理像素): 指设备能控制显示的最小物理单位，意指显示器上一个个的点。从屏幕在工厂生产出的那天起，它上面设备像素点就固定不变了，和屏幕尺寸大小有关，单位 pt。

设备独立像素(也叫密度无关像素或逻辑像素)：可以认为是计算机坐标系统中得一个点，这个点代表一个可以由程序使用的虚拟像素(比如: css像素)，这个点是没有固定下小的，越小越清晰，然后由相关系统转换为物理像素。

css像素(也叫虚拟像素)：指的是 CSS 样式代码中使用的逻辑像素，在 CSS 规范中，长度单位可以分为两类，绝对(absolute)单位以及相对(relative)单位。px 是一个相对单位，相对的是设备像素(device pixel)

DPR(设备像素比)：设备像素比 = 设备像素 / 设备独立像素。(在Retina屏的iphone上，DPR为2，1个css像素相当于2个物理像素)

XHTML和HTML的区别

1.XHTML 元素必须被正确地嵌套。

2.XHTML 元素必须被关闭。

3.标签名必须用小写字母。

4.XHTML 文档必须拥有根元素。

属性补充

a、text-transform属性

1.none 默认。定义带有小写字母和大写字母的标准的文本。

2.capitalize 文本中的每个单词以大写字母开头。

3.uppercase 定义仅有大写字母。

4.lowercase 定义无大写字母，仅有小写字母。

5.inherit 规定应该从父元素继承 text-transform 属性的值。

b、font-variant 把文本设置为小型大写字母字体

1、normal 默认值。浏览器会显示一个标准的字体。

2、small-caps 浏览器会显示小型大写字母的字体。

多列布局

规定多列布局之间的间隔：column-gap

规定多列布局之间列与列之间的边框：column-rule；

设置元素是否可以横跨所有列：column-span；

规定元素被分割的次数：column-count；

设置多列的列宽：column-with；

网页优化

1.网站结构（重构）优化，网页的中结构、表现、行为三者分离；

2.H1标签的使用，一个页面设置一个H1标签，有助于搜索引擎明白网站架构，因为H1标签的搜索权重比其他标签高，一般用于logo区域。

3.页面头部优化（meta）

<meta name="keywords" content="" />向搜索引擎说明你的网页的关键词

<meta name="description" content=""/>告诉搜索引擎你的站点的主要内容

“描述”部分应该用近乎描述的语言写下 一段介绍你网站的文字，在这其中，你应该适当的对你网站的特色内容加以重复以求突出；

“关键字”部分应该列出你认为合适的，能突出网站内容的关键字就可以了，关键字不要设置太多，可设置10---8个，搜索引擎只会浏览靠前的几个关键字。

4、超链接优化

搜索引擎为何可以能够索引全世界的网站，是因为各个搜索引擎程序中都有一个会自动“爬行”于互联网上的智能机器人程序，这个机器人就是顺着网站之间的链接游览世界的，那么我们就应该为它创造一个良好的爬行通道——合理的设置链接。

a.采用纯文本链接，少用，最好是别用Flash动画设置链接，因为搜索引擎无法识别Flash上的文字

b.按规范书写超链接，这个title属性，它既可以起到提示访客的作用，也可以让搜索引擎知道它要去哪里

c.最好别使用图片热点链接，理由和第一点差不多。

5、图片优化(alt属性，title属性)

图片优化并不是修改图片的大小、颜色，而是你应该为每个标签加上alt属性，alt属性的作用是当图片无法显示时以文字作为替代显示出来，而对于SEO来说，它可以令搜索引擎有机会索引你网站上的图片，对于一些确实没什么意义的图片，最好也不要省略alt，而应该留空，即 alt=""。

5、避免大“体积”的页面

6、命名规则化，使用公共代码

BFC

在了解BFC之前，先了解两个个概念：

1.box就是盒模型。box是css布局的对象和基本单位，说白了就是咱们网页布局的基石，简单来说一个页面是由很多box（盒模型）组成的。

2.元素的类型也就是display属性。元素的类型和display属性决定了这个 Box的类型。不同类型的Box，会参与不同的Formatting Context（一个决定如何渲染文档的容器-人话就是网页布局的一个范围），因此Box内的元素会以不同的方式渲染（显示）。

概念：BFC全称Block Formatting Context ，中文直译为块级格式上下文。它是W3C CSS 2.1 规范中的一个概念，它决定了元素如何对其内容进行定位，以及与其他元素的关系和相互作用。

BFC布局规则

1.在BFC下，内部的Box会在垂直方向，一个接一个地放置。

2.Box垂直方向的距离由margin决定。属于同一个BFC的两个相邻Box的margin会发生重叠

3.在BFC中，每一个盒子的左外边缘（margin-left）会触碰到容器（父元素）的左边缘(border-left)（对于从右到左的格式来说，则触碰到右边缘），即使存在浮动也是如此。

4.BFC的区域不会与float box重叠。例子：两个div，一个进行浮动，一个不浮动，这时浮动的元素会覆盖没有浮动的元素。给没有浮动的元素加overflow：hidden。就不会重合。

5.计算BFC的高度时，浮动元素也参与计算

如何触发 BFC

1.浮动元素，float 除 none 以外的值

2.position的值不为static或者relative

3.display为inline-block、table-cell、table-caption

4.overflow 除了 visible 以外的值

BFC的应用

1.解决浮动塌陷问题

2.自适应两栏布局（我们还可以运用BFC可以阻止元素被浮动元素覆盖的特性来实现自适应两栏布局。方法：给没有浮动的元素加overflow：hidden。）

3.解决设置margin值重叠问题。

总结：BFC就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素。它规定了元素与其他元素的关系和相互作用。

弹性盒布局介绍

1)弹性盒也叫伸缩布局盒模型

它是\*\*<span>css3</span>\*\*引入的一种新的布局模式——\*\*<span>flexbox</span>\*\*布局，即伸缩布局盒模型，用来提供一个更有效的方式制定、调整和分布一个容器里的项目布局。相比之前的布局方式，更便利于开发。

2)主要思想是让容器有能力让其子项目能够改变其宽度、高度（甚至顺序），以最佳的方式填充可用空间（主要是为了适应所有类型的显示设备和屏幕大小）。flex容器会使子项目扩展来填充可用空间，或缩小他们以防止溢出容器。

3)专业术语解释

a、主轴-水平方向或者X轴

b、侧轴-垂直方向或者Y轴

c、伸缩容器（父元素）和伸缩项目（子元素）

弹性盒属性

1)伸缩流方向 flex-direction 主要用来创建主轴，定义伸缩项目在伸缩容器中的方向

row：从左向右

row-reverse：与row相反

column：从上到下

column-reverse：与column相反

2)主轴对齐 justify-content 主要用来定义伸缩项目沿主轴线的对齐方式

flex-start：伸缩项目向一行的起始位置靠齐

flex-end：伸缩项目向一行的结束位置靠齐

center：伸缩项目向一行的中间位置靠齐

space-between：伸缩项目会平均的分布在行里

space-around：伸缩项目会平均的分布在行里，两端保留一半的空间

3)侧轴对齐 align-items 伸缩项目行在侧轴上的对齐方式

flex-start：伸缩项目在侧轴起点边的外边距 紧靠住 该行在侧轴起始边

flex-end：伸缩项目在侧轴终点边的外边距 紧靠住 该行在侧轴终点边

center：伸缩项目的外边距盒 在该行的侧轴上居中放置

baseline：伸缩项目根据伸缩项目的基线对齐

stretch：伸缩项目拉伸填充整个伸缩容器

4)换行 flex-wrap

nowrap 不换行 默认值，不管超出还是不超出都不会换行

wrap 换行 一旦伸缩项目超出伸缩容器，那么就会换行

wrap-reverse 换行反向 主轴水平时，上下反向，主轴垂直时，左右反向

5)伸缩流方向属性与换行属性缩写形式 flex-flow

flex-flow：换行 伸缩方向; 两个值同时定义或者单独定义都生效，不分先后顺序

6)align-self（加在子元素上）：主要用来设置单独伸缩项目在侧轴的对齐方式。注意点：align-self可以覆盖align-items

flex-start：伸缩项目在侧轴起点边的外边距 紧靠住 该行在侧轴起始边；（元素位于容器的开头）

flex-end：伸缩项目在侧轴终点边的外边距 紧靠住 该行在侧轴终点边；（元素位于容器的结尾）

center：伸缩项目的外边距盒 在该行的侧轴上居中放置；（元素位于容器的中间）

stretch：伸缩项目拉伸填充整个伸缩容器。（元素被拉伸以适应容器）

7)堆栈伸缩行 align-content(行与行之间的对齐方式) 定义多个伸缩行的对齐方式； 往往要与换行同时应用，没有换行就不存在多行的情况

flex-start：各行向伸缩容器的起点位置堆叠；（没有行距）

flex-end：各行向伸缩容器的结束位置堆叠；（底部对其没有行距）

center：各行向伸缩容器的中间位置堆叠；（居中没有行距）

space-between：各行在伸缩容器中平均分布；（两端对齐，中间自动分配）

space-around：各行在伸缩容器中平均分布，两端保留一半的空间；（自动分配距离）

8)显示顺序 order （加在子元素上）默认状态是按照标准流的顺序排列，在<span>flexbox</span>模型里，可以通过order改变伸缩项目的顺序

a、不定义order的伸缩项目会排到前面

b、order：1； 排第一 order：2； 排第二

9)flex（设置在子元素上） 作用：可以让元素自动占用容器的可用空间，等比划分

flex:1;

flex:2;

1. flex-grow属性定义项目的放大比例，默认为0，即如果存在剩余空间，也不放大；

如果所有项目的flex-grow属性都为1，则它们将等分剩余空间（如果有的话）。如果一个项目的flex-grow属性为2，其他项目都为1，则前者占据的剩余空间将比其他项多一倍。

11)flex-shrink属性定义了项目的缩小比例，默认为1，即如果空间不足，该项目将缩小，

如果所有项目的flex-shrink属性都为1，当空间不足时，都将等比例缩小。如果一个项目的flex-shrink属性为0，其他项目都为1，则空间不足时，前者不缩小。

负值对该属性无效。

12)flex-basis属性定义了在分配多余空间之前，项目占据的主轴空间（main size）

浏览器根据这个属性，计算主轴是否有多余空间。它的默认值为auto，即项目的本来大小。

它可以设为跟width或height属性一样的值（比如350px），则项目将占据固定空间。不常用，还在不断变化中。

flex属性是flex-grow, flex-shrink 和 flex-basis的简写，默认值为0 1 auto

弹性盒案例

1)支付宝

2)导航

3)自适应网页

4)表单

5)网页内容区域

背景尺寸属性

背景尺寸属性是CSS3中新增的一个属性, 专门用于设置背景图片大小.

语法：background-size:200px 100px;

（第一个参数是宽度，第二个参数是高度）

background-size:100% 80%;

（也可以写百分比）

background-size:cover;

情况：1.默认 2.具体情况 3.百分比 4.宽度等比拉伸 5.高度等比拉伸

6.cover 7.contain

cover含义:

专业属语：把背景图像扩展至足够大，以使背景图像完全覆盖背景区域。

背景图像的某些部分也许无法显示在背景定位区域中。

1.告诉系统图片需要等比拉伸

2.告诉系统图片需要拉伸到宽度和高度都填满元素

background-size:contain;

专业术语：把图像扩展至最大尺寸，以使其宽度和高度完全适应内容区域。

1.告诉系统图片需要等比拉伸

指定了背景图像的位置区域。

background-origin: padding-box;

告诉系统背景图片从什么区域开始显示,

默认情况下就是从padding区域开始显示

border-box背景图像相对于边框盒来定位。

content-box背景图像相对于内容框来定位。

CSS3中background-clip背景剪裁属性是从指定位置开始绘制。

background-clip属性 padding-box border-box content-box

background: url("images/animal1.png") no-repeat left top,url("images/animal2.png")；

多张背景图片之间用逗号隔开即可

注意点:

先添加的背景图片会盖住后添加的背景图片

建议在编写多重背景时拆开编写

状态栏图标

<link rel="shortcut icon" href=" "/>

综合

1.html的英文全称

hyper text Markup language

2.html的语法：

<标记 属性=“属性值”></标记>

<标记 属性=“属性值”/>

3.网页的组成

html+css+js

4.get与post

get是从服务器上获取数据

post是向服务器发送数据

5.css的英文全称

cascading style sheets

6.css的语法

选择符{属性：属性值；}

7.样式表的优先级

内联>内联？外联 （就近原则）

8.样式表的连接

外联：<link href="路径" rel="stylesheet" />

<style> @import url();</style>

​ 内部：<style>css 样式</style>

​ 内联：<div style="属性:属性值;"></div>

9.伪类选择器

​ link（初始）---visited（访问后）---hover（划过）---active（激活）

10.选择符权重

​ class/属性/伪类/伪元素-----0010

​ id----------------------------0100

​ 类型--------------------------0001

​ 内联--------------------------1000

​ 继承/子元素------------------0000

11.文本属性

​ font-size字体大小

​ color

​ font-family字体

​ font-weight粗细

​ font-style倾斜

​ text-align水平对齐

​ vertical-algin垂直对齐

​ line-height行高

​ font：字体/行高 字体

​ text-indent首行缩进

​ text-transform大小写字母切换

​ letter-spacing字间距

​ font-variant小型大写字母

12.背景属性

​ background-attachment固定

​ background-repeat 平铺

​ background-position 图片位置

13.空余空间

​ white-wrap：normal,nowrap,pre,pre-wrap,pre-line,inhiert

14.省略号

​ text-overflow：clip/ellipsis

15.省略文本条件： whith/overflow:hidden;/white-warp:nowrap;/text-overflow:ellipsis

16.定位：position:relative/absolute/fixed/static sticky

17.透明

​ opacity（0---1）

18.精灵图的优势

​ 减少服务器访问次数，提高网页加载速度。减少图片体积

​

19世界上第一个浏览器：

​ 网景 Netscape navigator

20.浏览器内核及其代表作

​ trident ie -m-

​ gecko firefox -moz-

​ presto opera -o-

​ webkit safari -webkit-

​ blink chrome

21.原点：origin

22.圆角：border-radius

23.盒子阴影：box-shadow

24.渐变

​ 线性渐变：linear-gradient

​ 径向渐变：radial

25.过渡：transition

第二阶段

Javascript概念

Javascript是一门面向对象的、跨平台的脚本语言；

对象：属性和方法的结合；类是对象的抽象，对象是类的实例；(你做事，我放心)；

跨平台：JS代码；（浏览器windows 、linux、unix、Android、IOS）；

脚本语言：即依赖外部环境才能运行的语言。

脚本语言特点：

​ 1、HTML文件必须在浏览器里执行；

​ 2、JS文件嵌入到HTML文件里才能执行；

​ 3、凡是不能独立执行需要依赖其他程序的，通常都叫做脚本，也叫动态语言。（宿主语言）；

- （浏览器>HTML文件>JS文件）

​ 真+假=假

​ 学习逻辑方法：类比、归类；

学习内容

1、完全掌握各版本的-JS-ES3，5，6；（ECMAscript）

2、利用JS完成复杂编程 -页面交互效果；

3、面向游戏对象编程思想 -游戏的制作；

4、前后端交互方式 -AJAX；

5、交互化构建工程化前段 -gulp。

JAVA历史

1995年，网景公司Netscape由brendan eich设计，

IE 的脚本语言：Jscript；

Nombas:cenvi中的scriptEase。

-1997年，ECMA发布262号标准文件（ECMA-262）的第一版；

1999年，ECMAScript 3.0发布；

2015年6月，ECMAScript 6 正式通过（ES6）；

ECMAScript 可以为不同种类的宿主环境提供核心的脚本编程能力，因此核心的脚本语言是与任何特定的宿主环境分开进行规定的... ...

标准化：W3C； ECMAscript

组成：ECMAScript；BOM：DOM；

JS特点

1、解释性脚本语言；

2、运行在浏览器（浏览器带有js解释器，Chrome V8引擎）；

3、弱类型语言（松散型）；

4、事件驱动（动态）；

5、跨平台；

JS用途

（视觉交互、数据交互、node.js）

嵌入动态文本于HTML页面;

对浏览器事件做出响应;

读写HTML元素;

在数据被提交到服务器之前验证数据;

检测访客的浏览器信息（可以使用js代码判断浏览器类型）;

控制cookies，包括创建和修改等;

基于Node.js技术进行服务器端编程;

Cookie的特点

1. 储存在用户本地硬盘上的数据；
2. Cookie可能被删除，因为每个cookie都是硬盘上的一个文件都有可能被用户删除；
3. Cookie最对储存50个；
4. Cookie不能跨域访问；

JS和H5的关系

页面三要素

- 结构：HTML

- 表现：CSS

- 行为：JS

H5是一种新的技术，就目前而言，我们所知的H5都是一些标签，但是有了JS之后，这些标签深层的扩展功能才得以实现。 比如video标签；我们对其理解为一个简单的标签；但是实际上，video标签还有更深层次的扩展功能；

Javascript基础

Javascript书写

1、内部JS

<script>

document.write("hello world");文本显示

console.log("hello world");控制台显示

alert("hello world");弹窗显示

document.write("<h1>hello world</h1>");

document.write("&lt;h1&gt;hello world&lt;/h1&gt;");

/\* 多行注释 \*/

</script>

2、外部JS

index.js文件 document.write("hello world");

​ 内部引入：<script src="path:to/index.js"></script>；

<script src="01.js">

<！--这里不能写js代码，因为这个script是用来引入外部js文件的-->

</script>

向页面输出内容：

document.write("hello world");

document.write("< strong >我很man</strong>")；

转义字符：

&lt； || &gt；

document.write("&lt;strong&gt;我很man&lt;/strong&gt;");

变量声明：

var a = "hello world"；

| var | 声明变量的关键字 |

| a | 变量名 |

| = | 赋值 |

| "hello world" | 变量的值 |

我们所写的程序运行在内存中，当我们使用关键字var 声明一个变量的时候，计算机会从内存中留出一定储存空间，为我们存放不同类型的内容。

地址指的是内存的编号；

指针就是指向某个地址的工具，去取到这个地址对应的内存里面的值；

- 分清左右两侧的意义：

JavaScript中的 ‘’=‘’ 和我们数学中的 ‘’ = ‘’不一样，在JavaScript中‘’=‘’的作用是赋值（取别称）。

var a = "hello world"; 这句话分成两个部分。

​左值：在等号左侧，是声明一个变量并且起名；

​右值 ：在等号右侧，是存放进变量中的东西（给变量赋值）；

注释

​ // ？

变量类型

**变量**：存储数据的容器；

弱类型语言

（因为存放进变量中的数据是放在内存中的，计算机只识别0，1，所以需要对不同类型的数据进行编码，编码的结果会导致所需空间不同，所以JavaScript中分成很多种数据类型，以便计算机储存时使用）

数字类型(number)：1234567890 int：整型 float：浮点型（小数）

字符类型(string)：'123456',"字符串类型",.....（有单引号或者双引号）

未定义类型(undefined)：这是一种特殊类型，当变量被声明但是没有被赋值的时候，那么该变量的类型为undefined；

布尔值类型（boolean）：true，false；(boolean会隐式转换，true会转换成1、false会转换成0；)

对象类型：object；

函数类型：Function；

Symbol

引用类型：Function：object

判断数据类型

关键字：typeof；用typeof后得到的是字符串；

例:console.log（typeof ？）

-- typeof返回有六种值： number、string、boolean、undefined、object（数组也是对象）、function；

-- js有五种基本数据类型：number、string、boolean、undefined、null、symbol；

--两种引用类型：object、function；

--两种特殊数据类型：null、undefined；

变量类型转换

**隐式转换**

**显式转换**

​ parseInt( )、parseFloat( )、Number( )、toString( )、String( )；

​ toFixed( )保留几位小数之后数据类型变为字符串；

​ 进制转换toStering（进制数）

parseInt( ) ：

方法首先查看位置 0 处的字符，判断它是否是个有效数字；如果不是，该方法将返回 NaN，不再继续执行其他操作。 但如果该字符是有效数字，该方法将查看位置 1 处的字符，进行同样的测试。这一过程将持续到发现非有效数字的字符为止，此时 parseInt() 将把该字符之前的字符串转换成数字。

(从左往右看，直到特殊符号和字母结束，保留前面的整数，若是前面第一位出现字母或字符这回不出现NaN,)

parseFloat( )：

跟parseInt()极其相似，不过第一个出现的小数点是有效字符。如果有两个小数点，第二个小数点将被看作无效的。parseFloat ( ) 会把这个小数点之前的字符转换成数字。

（与parseint一样，但会保留一个小数）

NaN：Not a Number：

​ 是一个特殊值，说明某些算术运算（如求负数的平方根）的结果不是数字（本来期望得到一个数字，但是结果没有计算成功）。方法 parseInt() 和 parseFloat() 在不能解析指定的字符串时就返回这个值。

​判断是否是NAN时必须要用 isnan 来判断；

number

（整数小数可以转换，但有字符就会出现NAN）

ECMAScript 中可用的 3 种强制类型转换如下：

Boolean(value) - 把给定的值转换成 Boolean 型；

Number(value) - 把给定的值转换成数字（可以是整数或浮点数）；

String(value) - 把给定的值转换成字符串；

关键字

| break | do | instanceof | typeof

| case | else | new | var

| catch | finally | return | void

| continue | for | switch while

| debugger\* | function | this |with

| default | if | throw delete

| in | try

保留字

未来可能作为，标识符存在

| abstract | enum | int | short

| boolean | export | interface | static

| byte | extends | long | super

| char | final | native | synchronized

| class | float | package | throws

| const | goto | private | transient

| debugger | implements | protected | volatile

| double | import | public | name

命名

字母、数字。下划线和$，但是不能以数字开头和关键字、保留字命名；连接可以用下划线，或者是第二个大写（user\_name / userName驼峰命名）。

- js语言里面的分号可以省略

运算符

程序：数据结构+算法

1、算数运算符

加、减、乘、除、求余/取模（+ 、- 、\* 、/ 、%）；

+：加好除了做数字运算还能字符串连接，数组跟字符相加，会把数字转换为字符串并且连接到一起；

取余：大数对小的那个数取余结果为整除之后的余数；小的那个数对大数求余的结果就是小的那个数本身；

2、关系运算符

= =（相等） 、! =（不等于）、<（小于） 、>（大于） 、< =（小于或者等于） 、> =（大于或者等于）

​ \*\*判断运算符两侧的结果是否满足规则，满足结果即为true，否则false；

3、逻辑运算符

逻辑与：&&

​ “和”的意思；必须两个条件均为true结果才为true；

逻辑或：||

​ “或”的意思 ；只要有一个条件为true结果就为true，必须两个条件均为false结果才为false；

逻辑非：! ；“不”的意思， 对本身的结果取反。

与逻辑在两个条件均为true时结果才为true，其余情况均为false；

或逻辑在两个条件均为false时结果才为false，其余情况均为true；

补充：

​ 异或：两个条件结果不同时（一个true，一个false），结果才为true。

1. 三目运算符

？

写法：条件？ 条件成立时：条件不成立时 ；

自增

++a a++、- -a a-

-自增自减前置后置是不一样的，前置是先加后用，后置的先用后加。

= 和 == 以及 === 的区别

即：= 赋值 、== 相等 、=== 全等、！== 不全等 ；

（1）一个等号是赋值，将等号右边的值赋给左边；

（2）两个等号是判断相等，相等结果为true，不等为false，不考虑数据类型，只判断值；

（3）三个等号要求全等，不仅值要相等，类型也必须相等；

Javascript程序执行结构

分支结构：根据不同的条件判断来决定程序执行走向的结构（也叫选择结构）

循环结构：需要重复执行同一操作的程序结构称为循环结构。

顺序结构：按照由上到下的顺序一行一行地执行的程序结构。

**注意：0、-0、null、""、false、undefined 或者 NaN在if条件里结果为false；**

if else

单分支：if（判断条件）{

​ 条件成立时执行的代码

​ }else{

​ 条件不成立时执行的代码

​ }；

双分支：if（判断条件1）{

​ 条件1成立时执行的代码

​ }else if（判断条件2）{

​ 条件1不成立但条件2成立时执行的代码

​ }else if（ 判断条件3）{

​ 条件1和条件2不成立但条件3成立时执行的代码

​ }......else{

​ 全都不成立

​ }；

如果 if 里面只有一句话，大括号可以省略；

eg:

var a = (num % 10); // 个位数

var b = parseInt((num / 10) %10); // 十位数

var c = parseInt(num / 100); // 百位数

var d = parseInt((num / 100) %10); // 千位数

弹出输入框: prompt( )；

switch，case

全等匹配，适用于确定几种情况之间的情况；

有多个确定值需要判断的时候；注意case穿透，要加break语句（如果程序没有发现break语句，那么解析器会继续向下解析；

例：var num=4；

​ switch（ ）{

​ case 1：...............................break

​ case 2：...............................break

​ case 3：...............................break

​ default :

​ }

点击事件

onclick=“ calc（）”

white循环

循环的三大因素：

1、循环变量的初始值

2、步进；

3、终点值；

var b = “ ”内容

var a = i；起始条件

white（ 判断条件 （终止条件））{

​ 条件成立时反复执行的代码

​ i++ （步进）

}

do white循环

不管条件是否成立一定花先执行一次；

do{

}white（ ）

For循环

for（ 定义初始变量；判断条件；步进 ）{

​ 循环体

}

for循环的嵌套：可以简单的理解为行和列的关系（外层for循环控制行，内层循环控制列）。

断点的使用

右键检查或者F12，单击sources选择调试的文件，选择调试程序对应的序号，刷新进入后进行调试。单步调试 F11；

break和continue

break：跳出循环体，整个循环结束；

continue：结束本次循环，但继续下次循环；

三种循环的联系和区别

1、都是会反复执行的代码块；

2、大部分情况下可以互相替换；

3、do...while至少执行一次，while和for有可能0次，while不太能确定执行次数，for可以。

函数

函数在创建的时候会创建两个对象，一个是函数对象本身，另一个是作用域链对象

函数在调用的时候会创建一个执行环境对象（活动对象）

function

系统函数（封装、调用）

函数表达方式：

​ 立即函数表达式：

（function（）{ }）（）；只能调用一次

eval（）

可以使字符串当做代码使用；

parseInt( ) alert( ) prompt( ) eval( )

封装函数关键字：

function fn （ 传递参数 形参 ）{

​ 函数体（可重复执行）

}；

如何调用函数：fn ( 实参 )；

回调函数：把函数作为参数

获取样式

1、\_\_\_样式采用内联样式；

2、（外面）定义一个开关；

var a（取名）=（赋值） true、false、 数字。。。。a = ！a；

函数的参数

形参、实参；

- 形参：形式参数，声明函数的时候写在小括号里面的参数，无需var

- 实参：实际参数，在函数调用的时候需要传递实际有值得参数

- 实参个数大于形参，多余实参自动舍弃

- 形参个数大于形参，多余形参默认为undefined

- 调用的时候讲实参传递形参，形参的值就是实参的值。

作用域

1、全局作用域 外边

2、局部作用域 内部声明

全局变量

作用范围为整个程序的执行范围

在函数体外部定义的变量就是全局变量

在函数体内部不使用var定义的也是全局变量

局部变量

作用范围是某个函数体内部

在函数体内部通过var关键字定义的变量或者形参，都是局部变量

当局部变量与全局变量重名时，在函数体内部局部变量优先于全局变量

变量提升

变量的声明会提升至当前作用域的最顶端，但不会提升赋值

eval

Eval：把字符串当作js代码去执行

var str = "1+1";

var res = eval(str);

console.log(res);

副本

arguments

return

1.结束函数的执行

2.交回函数执行权

3.返回一个结果函数调用位置

堆栈

存储数据的模式

即--- 是一种数据结构，指的是数据存取的方式，当定义一个变量时，内存会开辟一段空间。

栈：先进后出 栈底删除，插入

队列：先进先出 对头删除，对位插入

作用域链

内层环境可以访问外层中的变量和函数，而外层环境不能访问内层的变量和函数。

执行环境站

也称执行上下文

当JavaScript解释器初始化执行代码时，它首先默认进入全局执行环境，从此刻开始，函数的每次调用都会创建一个新的执行环境，每一个执行环境都会创建一个新的环境对象压入栈中。

当执行流进入一个函数时，函数的环境对象就会被压入一个环境栈中（execution stack）。在函数执行完后，栈将其环境弹出，把控制权返回给之前的执行环境。

递归

程序调用自身的编程技巧称为递归（recursion）。

递归具备的条件：

1. 子问题须与原始问题为同样的事，且更为简单；

2. 不能无限制地调用本身，须有个出口，化简为非递归状况处理。

由于递归是函数本身一层一层压栈，导致先入栈的不能出栈，空间占满以后就会造成堆栈溢出。

逻辑短路

如果第一部分已经能够决定真个表达式的结果，那第二部分就不会执行

与逻辑： 一部分false下一步就不会执行；

或逻辑： 一部分false下一步会继续执行；

变量提升

解释器会扫描代码进行预编译查看是否有语法错误。

变量和函数提升当前作用域的顶部，但是变量的赋值不会提升。

（函数与变量冲突时会保留函数，两个函数冲突时会按照书写顺序，顺序靠后的会覆盖前面的。）

JS运行和编译

1、语法分析：

查找基本语法有没有错误。

2、预解析：

执行之前进行预解析。

var、function关键字提前到当前作用域的顶部，变量默认值为undefined，函数默认值为函数体代码块，当函数与变量重名时，保留函数。

3、解释执行：

变量生命周期：

全局变量的生命周期直至浏览器卸载页面才会结束。

局部变量只在函数的执行过程中存在，而在这个过程中会为局部变量在栈或堆上分配相应的空间，以存储它们的值，然后再函数中使用这些变量，直至函数结束

事件

用户的行为：onclick、ondblclick、onfocus、onblur、window.onload；

是用户跟页面的交互，当用户跟页面进行一些“交流”的时候，页面通过js就会触发一些事件，比如鼠标点击的时候就会触发onclick事件，给这个事件绑定一个函数，那么这个时候函数就会被调用，代码就会被执行。

文本框事件：onchange

元素失去焦点：onblur

元素获得焦点：onfocus

**事件类型**：

鼠标事件：click，dbclick，mousedown，mouseup，mouseover，mouseout，mouseenter、mouseleave、mousemove

scroll mousewheel 鼠标滚轮

contextmenu：鼠标右键（上下文菜单：在不同环境下右键菜单不一样）

mouseover：鼠标在元素身上移动穿过子元素的时候会被反复触发

mouseenter：只是在进入元素的时候触发

**键盘事件**：keydown，keyup，keypress

**表单事件**：对表单元素操作之后会触发的事件

单选框、多选框、下拉菜单 状态改变的时候会触发 onchange 事件

表单提交的时候会触发 onsubmit 触发在<form>元素身上。

Math/date

敏感词过滤

替换replace（ ， ）

Math内置对象

对象属性：

Math.E

返回算术常量 e，即自然对数的底数（约等于2.718）。

Math.LN2

返回 2 的自然对数（约等于0.693）。

Math.LN10

返回 10 的自然对数（约等于2.302）。

Math.LOG2E

返回以 2 为底的 e 的对数（约等于 1.414）。

Math.LOG10E

返回以 10 为底的 e 的对数（约等于0.434）。

Math.PI

返回圆周率（约等于3.14159）。

Math.SQRT1\_2

返回 2 的平方根的倒数（约等于 0.707）。

SQRT1\_2.SQRT2

返回 2 的平方根（约等于 1.414）。

对象方法：

abs(x)

返回数的绝对值。

sin(x)

返回数的正弦。

cos(x)

返回数的余弦。

tan(x)

返回角的正切。

ceil(x)

对数进行上舍入。

floor(x)

对数进行下舍入。

round(x)

把数四舍五入为最接近的整数。

max(x,y)

返回 x 和 y 中的最高值。

min(x,y)

返回 x 和 y 中的最低值。

pow(x,y)

返回 x 的 y 次幂。

sqrt(x)

返回数的算术平方根。

random( )

返回 0 ~ 1 之间的随机小数，包含0不包含1。

生成 min ~ max （min < max）的随机数公式：

Math.random( ) \* (max - min) + min；

参数对应的是弧度：Math.PI / 180;

Eg：Math.sin(30 \*Math.PI / 180 )就等于sin30的值。

对象

JavaScript 中的所有事物都是对象：字符串、数值、数组、函数...........

删除对象属性：delete；

javaScript内置对象(十个)

1. Object对象 所有JavaScript对象的超类(基类)

2. Array对象 数组对象---定义数组属性和方法

3. Boolean对象 布尔对象---布尔值相关

4. Date对象 日期对象---日期时间相关

5. Error对象 错误对象---处理程序错误

6. Function对象 函数对象---定义函数属性和方法

7.Math对象 数学对象---各种数学运算工具(不是构造函数)

8.Number对象 数字对象---定义数字属性和方法

9.RegExp对象 正则表达式对象---定义文本匹配与筛选规则

10.String对象 字符串对象---定义字符串属性和方法

Date内置对象

时间戳：时间戳是指格林威治时间1970年01月01日00时00分00秒(北京时间1970年01月01日08时00分00秒)起至现在的总毫秒数（时间戳）。

get系列API

getFullYear( )

返回年；

getMonth( )

返回月份0--11；

getDate( )

返回某一天；

getDay( )

返回星期0-6；

getHours( )

返回小时；

getMinutes( )

返回分钟；

getSeconds( )

返回秒；

getTime( )

返回1970年1月1日午夜到指定日期（字符串）的毫秒数；

set系列API

setFullYear( )

设置年份；

setMonth( )

设置月；

setDate( )

设置天；

setHours( )

设置小时；

setMinutes( )

设置分钟；

setSeconds( )

设置秒；

setTime( )

使用毫秒的形式设置时间对象；

注意：

0 为上个月最后一天；

-1 为上个月最后一天之前的天数；

如果当月有31天，32 为下个月的第一天；

如果当月有30天：32 为下一个月的第二天。

定时器：setInterval（内容“可以为函数”，时间“单位：毫秒”）。

InnerHTML (前面的内容满足条件则输出HTML文本)。

对象数组

对象{ }

obj，属性和方法的结合；类是对象的抽象，对象是类的实例；JS在开发前没有对象；

创建对象的方式：

  （1）字面量  var obj = {

        键值对

        key:value

    }

  （2）通过new运算符 var obj = new object( )

对象遍历：for...in循环（隐式迭代）

For（var key in obj）{ }

键值对：key：value；

{ （属性、方法）

属性名 ： 属性值，“用逗号隔开”

}

例：

var obj= new object( );

obj.name = "zhangsan";

obj.age = 18;

obj["gender"] = "male";

for(var key in obj){

console.log(key);

console.log(obj[key]);

}

For-in或for-of

For in可以遍历对象，for of不能变量对象

For of可以遍历map集合，for-in不能遍历map集合

For-in遍历数组得到的是数组的下标，for of遍历数组得到是数组的元素

For in遍历键，for of遍历值；

数组[ ]

稀疏数组、二维数组

Array（引用类型），一组数据（一个变量来承载）。

数组的长度 arr.length

数组的索引（下标） arr[0] --- arr[arr.length-1]

（1）稀疏数组：数组中部分元素没有值的数组。

var arr1= new A rray（n）； “此时两位数组长度”

arr1[3] = 10； “给第四个赋值”

（2）var arr2 = new A rray（1,2,3,4,5,6,） “定义数组”

传除数字以外的一个参数

（3）var arr2 = new A rray（“verr”）

遍历数组for

1. for in

for(var key in arr){

}

（5）for of

for(var value of arr){

}

API：数组方法

concat( )

连接两个或更多的数组，并返回结果。

join( )

把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。

pop( )

删除并返回数组的最后一个元素

push( )

向数组的末尾添加一个或更多元素，并返回新的长度。

shift( )

删除并返回数组的第一个元素

unshift( )

向数组的开头添加一个或更多元素，并返回新的长度。

reverse( )

颠倒数组中元素的顺序。

slice( )

从某个已有的数组返回选定的元素

sort( )

对数组的元素进行排序

sort会循环的取数组里的每一个元素function里返回值如果大于0,就会交换顺序：

即：var arr1 = arr.sort(function (a,b) {

return a-b;

});

splice( )

删除元素，并向数组添加新元素。

splice(？，？，？)

“从下标？的位置删除？个在从删除的位置新增一个”。

toString( )

把数组转换为字符串，并返回结果。

排序

1. 冒泡排序：“相邻两个元素进行比较，若是小于当前比较数值则向左换序，直至最后一个”

<script>

function bubble(arr){

for(var i=0; i<arr.length-1; i++){

for(j=0; j<arr.length-1-i; j++){

if(arr[j]>arr[j+1]){

var temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

return arr;

}

console.log(bubble([12,12,32,54,02,78,54,41]));

</script>

1. 选择排序：“左边数值为单位向右查找小于的数，第一个值先记录直至搜索完毕找到最小的值，然后排序”。

<script>

function choose(arr){

for(var i=0; i<arr.length-1; i++){

var minIndex=i;

for(var j=i+1; j<arr.length; j++){

if(arr[minIndex]>arr[j]){

minIndex=j;

}

}

var temp=arr[i];

arr[i]=arr[minIndex];

arr[minIndex]=temp;

}

return arr;

}

console.log(choose([12,20,15,56,24,34]));

</script>

1. 快速排序：
2. 插入排序：

二维数组

<script>

var arr = [

[1,2,3],

[4,5,6],

[7,8,9]

];

console.log(arr[1][1]);

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

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

console.log(arr[i][j]);

}

}

</script>

去重

去重的方法：

1. 查找数组中是否有两个数有则删除

<script>

function distinct (arr){

for(var i=0; i<arr.length-1; i++){

for(var j=i+1;j<arr.length;j++){

if(arr[i] === arr[j]){

arr.splice(j--,1);

}

}

}

return arr ;

}

console.log(distinct([ ]));

</script>

1. 通过对象的属性值不能重复的特点去重：

<script>

function distinct(arr){

var obj={};

obj.name;

obj["name"];

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

if(obj[arr[i]] === undefined){

obj[arr[i]]=1;

}else{

arr.splice(i--,1);

}

}

return arr;

}

console.log(distinct([ ]));

</script>

3、直接去重；

<script>

function distinct(arr){

return Array.from(new Set(arr));

}

console.log(distinct([ ]));

</script>

4、reduce去重：

var arr=[ ];

var arr1= arr.reduce(function(perv,next){

if(!perv.includes(next) perv.push(next);

return perv;

},[])

console.log(arr);

5、set结构去重：

var newarr = [...new Set(array)];

ES5新增数组常见方法

数组：

2个索引方法：indexOf() 和 lastIndexOf()；

5个迭代方法：forEach()、map()、filter()、some()、every()；

2个归并方法：reduce()、reduceRight()；

**字符串**

trim( ); // 去掉字符串前后空格

trimLeft( )

trimRIght( )

**索引**：

indexOf( ？ )

找到为？的索引；

lastIndexOf( )

倒着找从右往左为？的索引；

**迭代**：

forEach( )

通过传函数应用；item为数组，index为索引。

例：arr.forEach(function (item, index) {

console.log(item, index);

})

map( )

通过函数给个返回值item，

var arr1 = arr.map(function (item, index) {

return item\*2;

})

filter( )

过滤，通过函数给个返回值item，返回一个条件，条件为ture则这个值会保留；

some( )

通过函数给个返回值item，返回一个条件，只要部分条件为ture则这个值会保留；只要有一个item满足条件,那么结果就为true, 并且不会再往后去查找了

every( )

通过函数给个返回值item，返回一个条件，全部条件为ture则这个值会保留；必须所有的都满足条件才返回true, 所以一定会找到最后一个元素

**归并**

reduce( function(prev,next){} ) 上次迭代的结果作为下次迭代的开始

reduceRight( )；从右往左

JSON.parse(str); json序列化，将符合json格式的字符串转换为json

JSON.stringify( ); json转换为字符串

Object.assign 合并对象

Date.now( ); //日期对象得到当前日期的毫秒数

Object.defineProperties(obj, props); //给obj设置属性

Object.keys(obj); //获取obj的所有属性名称，返回数组

Object.values(obj); //获取obj的所有属性值，返回数组

字符串

ASCII码和字符集

charAt( )

返回在指定位置的字符。

indexOf( )

检索字符串，返回下标

lastIndexOf( )

从后向前搜索字符串。

charCodeAt( )

返回在指定的位置的字符的 Unicode 编码。

fromCharCode( )

从字符编码创建一个字符串。

concat( )

连接字符串。

match( )

找到一个或多个（正则表达式的）匹配。

replace( )

替换与正则表达式匹配的子串。

search( )

检索与正则表达式相匹配的值。

slice( )

提取字符串的片断，并在新的字符串中返回被提取的部分。

split( )

把字符串分割为字符串数组。

substr( )

从起始索引号提取字符串中指定数目的字符。

substring( )

提取字符串中两个指定的索引号之间的字符。

toLowerCase( )

把字符串转换为小写。

toUpperCase( )

把字符串转换为大写。

trim( )

去掉字符串前后空格（ES5）

startsWith( )

字符串是否以某个字符开头（ES6）

endsWith( )

字符串是否以某个字符结尾（ES6）

includes( )

字符串是否包含某个字符（ES6）

repeat( )

重复某个字符串几次（ES6）

字符串常见API(charCodeAt\fromCharCode)；字符串不能相减

遍历字符串

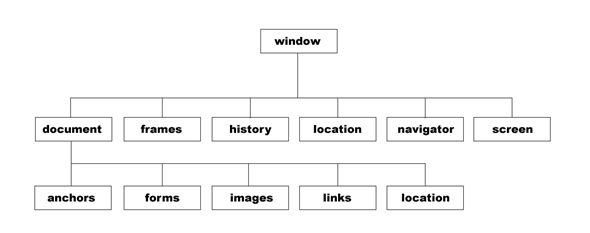
For-in语句

For-of语句

For循环

BOM/DOM

BOM

BOM：浏览器对象模型

1. 全局默认对象是挂在windows上的；
2. Windows下的子对象：

（1）location 刷新

刷新的方法：window.location.reload( );

window.location.href 当前页面的 URL，可以获取，可以修改（页面跳转）。

window.location.hostname web主机的域名

window.location.pathname 当前页面的路径和文件名

window.location.port web主机的端口 （80 或 443）

window.location.protocol 所使用的 web 协议（http:// 或 https://）

window.location.search 请求参数（？后面的内容）

（2）window.navigator

navigator.appName 返回获取当前浏览器的名称。

navigator.appVersion 返回获取当前浏览器的版本号。

navigator.platform 返回当前计算机的操作系统。

navigator.userAgent 返回浏览器信息

（3）history

history.go(1) 参数可写任意整数，正数前进，负数后退

history.back( ) 后退

history.forward( ) 前进

（4）screen: 屏幕

window.screen.width 返回当前屏幕宽度(分辨率值)

window.screen.height 返回当前屏幕高度(分辨率值)

（5）window下的弹框方法

alert( ) prompt( ) confirm( )

（6）定时器

超时定时器 间隔定时器

setTimeout setInterval

clearTimeout clearInterval

（7）window.onload 页面加载

（8）window.onscroll 滚动条兼容

（9）window.onresize 浏览器窗口大小触发事件

DOM

DOM：文档对象模型

**获取DOM节点：**

1、document.getElementById(id名)

2、getElementsByTagName(标签名)

得到的是一个集合（不止一个，是一堆）

3、getElementsByName( ) 通过Name值获取元素，返回值是集合，通常用来获取有name的input的值；

注：1、不是所有的标签都有name值；

2、低版本的浏览器会有兼容问题；

4、children属性，获得DOM元素的所有子元素；返回值是集合

5、parentNode属性，获得DOM元素的父级元素

6、getElementsByClassName(class名称)，但是：IE8以下不能用

7、ES5选择器：

document.querySelector ( ) 一旦匹配成功一个元素，就不往后匹配了

document.querySelectorAll ( ) 匹配到所有满足的元素, 支持IE8+；

**属性获取和操作**：

1、getAttribute( )获取元素的属性值，他是节点的方法！所以前缀必须是节点！

document.getElementById( ID值 ).getAttribute( )

什么是元素属性呢？ class就是元素属性，写在标签内的所有东西都是标签属性， 比如link的href比如img的src....都是元素属性。

元素自带的属性可以直接用语法获取，但是自定义属性需要 getAttribute() 和 setAttribute( ) 方法。

2、setAttribute( )设置元素的属性。同上；

有些小小的兼容性问题，低版本IE不兼容；

设置的属性永远都是字符串类型

3、removeAttribute( )删除属性；同上；

兼容性问题同上；

**DOM元素类型**

（元素、文本和属性）

1. node.Name 节点名称
2. node.Type 1=元素节点、2=属性节点、3=文本节点

元素节点：标签名（大写）

属性节点：属性名称

文本节点：#text

适用场所：网页换肤、隔行变色。

**操作DOM**

作用：增、删、克隆节点

创建节点

var oDiv = document.createElement("div");

克隆节点

clonedNode = Node.cloneNode(boolean)只有一个参数，传入一个布尔值，true表示深客隆，复制该节点下的所有子节点；false表示浅克隆，只复制该节点

插入节点

parentNode.appendChild(childNode); 将新节点追加到子节点列表的末尾

parentNode.insertBefore(newNode, targetNode); 将newNode插入targetNode之前

替换节点

parentNode.replaceChild(newNode, targetNode); 使用newNode替换targetNode

移除节点

parentNode.removeChild(childNode); 移除目标节点

node.parentNode.removeChild(node); 在不清楚父节点的情况下使用

childNode.remove( ) IE不支持

node节点

根据 W3C 的 HTML DOM 标准，HTML 文档中的所有内容都是节点：元素是节点的别称，节点包含元素当然节点还有好多细化的种类；

**节点操作**：（通过父子系关系）

childNodes 获取当前元素的所有子节点；

nodeType 节点种类，返回值是数字；

nodeValue 获取（文字）节点的文本内容；

nodeName 返回node节点名称（#text，注释，标签....）；

获取节点元素

nodeValue 不会进行解析，会将标签名转译成字符串，直接输出；

innerHTML 解析标签，获取元素节点标签间的元素内容 前面的内容满足条件则输出HTML文本。

innerTEXT 表示节点及其后代的“呈现”文本内容。

tagName 获取元素节点间的标签名

childNodes 获取当前元素节点的所有子节点

firstChild 获取当前元素节点的第一个子节点

lastChild 获取当前元素节点的最后一个子节点

previousSibling 获取当前节点的前一个同级节点

nextSibling 获取当前节点的后一个同级节点

firstElementChild 获取当前元素节点的第一个元素子节点

lastElementChild 获取当前元素节点的最后一个元素子节点

ownerDocument 获取该节点的文档根节点，相当与 document

parentNode 获取当前节点的父元素

**创建文档碎片**

Document.createDocumentaFragment( )

例：

var cache = document.createDocumentFragment( );

for( var i = 0 ; i < 1000; i ++ ){

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

opt.type="button";

opt.value = "删除";

cache.appendChild(opt);

}

document.body.appendChild(cache);

<body>

<ul></ul>

<script>

var ul = document.querySelector("ul");

var cache = document.createDocumentFragment();

for(var i = 0; i < 100; i++){

var li = document.crecliateElement("li");

li.innerHTML = i;

cache.appendChild(li);

}

ul.appendChild(cache);

</script>

</body>

DOM尺寸和位置

**DOM尺寸**

box.style.width

box.style.height

只能获取到内联style属性的CSS样式中的宽和高，如果有，获取;如果没有，则返回空。

getStyle(box,"width")

getStyle(box,"width")

通过计算获取元素的大小，无关你是否是行内、内联或者链接，它经过计算后得到的结果返回出来。如果本身设置大小，它会返回元素的大小，如果本身没有设置，非IE浏览器会返回默认的大小，IE浏览器返回auto。

box.clientWidth

box.clientHeight

返回了元素大小，但没有单位，默认单位是px，如果设置了其他的单位，比如100em之类，返回出来的结果还会转换为px像素（不含边框）。

box.scrollWidth

box.scrollHeight

获取滚动内容的元素大小（当元素出现滚动条时，此属性指全部滚动内容的宽高）

返回了元素大小，默认单位是px。如果没有设置任何CSS的宽和高度，它会得到计算后的宽度和高度。

box.offsetWidth

box.offsetHeight

返回了元素大小，默认单位是px。如果没有设置任何CSS的宽和高度，他会得到计算后的宽度和高度，包含盒模型中除margin以外的宽高（包含边框）。

**DOM 位置坐标**

box.clientLeft

box.clientTop

获取左边框和上边框的宽度

box.offsetLeft

box.offsetTop

获取元素当前相对于offsetParent父元素的位置

box.scrollTop

box.scrollLeft

获取滚动内容上方的位置(就是隐藏的内容的高度)

获取滚动内容左方的位置

offsetParent

这个属性的返回值是它根据谁定位的，如果它的所有父元素都没有定位，那么返回body

**获取分行内样式**

function getStyle(obj,attr){ //获取非行间样式，obj是对象，attr是值

if(obj.currentStyle){ //针对ie获取非行间样式

return obj.currentStyle[attr];

}else{

return getComputedStyle(obj,false)[attr]; //针对非ie

};

};

事件

事件是给浏览器定义一个预处理函数，当事件触发的时候，执行函数，这就是事件。

右键菜单：oncontextme 表单提交事件：onsubmit

鼠标滚轮事件onmousewheel模拟滚动条

格式：

浏览器中的节点(对象) . on + 事件句柄 = function ( ) {

要干什么？（放在浏览器中，不执行，当事件发生的时候再执行。）

}

**鼠标事件及方法**

JS缺德定律：事件对象有兼容问题：e = e || window.event

e.buttons

返回鼠标点击按键（1左键，2右键，4中键滚轮）

e.offsetX / offsetY

获取事件触发最近的盒子（事件源）的坐标

e.clientX / clientY

获取可视区的坐标（根据浏览器的定位）

e.screenX / screenY

获取整个屏幕的坐标

e.pageX / e.pageY

获取文档的坐标（包含滚动条）

**键盘事件**

keydown、keyup、keypress

格式：

document.onkeydown = function(e){

console.log(e.keyCode)

}

键盘事件有兼容问题：e.keyCode || e.which

**阻止默认事件**

ruturn false

格式：

if(e.preventDefault) {

e.preventDefault();

}else{

window.event.returnValue = false;

// return false;

}

**判断当前键码**

格式：document.onkeydown=function(e){

e=e || window.event;

var cade=e.keyCode || e.which;

console.log(code);

}

**可视化宽高**

/\* 获取整个body的宽高

\*/ @return <object> {width, height}

getBody : function () {

return {

width : document.documentElement.clientWidth || document.body.clientWidth,

height : document.documentElement.clientHeight || document.body.clientHeight

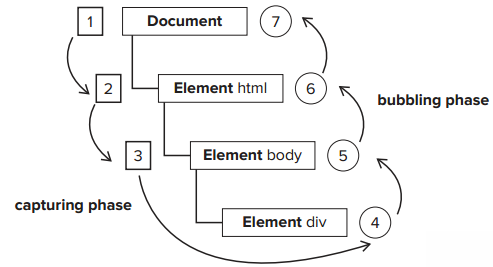
}

}

事件流

完整事件流包含： 捕获阶段 、 目标阶段 、 冒泡阶段。

子元素的事件被触发时，父级也会被触发（冒泡）。



**阻止冒泡**

if(e.stopPropagation) e.stopPropagation();

else e.cancelBubble = true;

事件监听

DOM0级事件处理，是一种赋值方式，是被所有浏览器所支持的，简单易懂容易操作;

DOM2级事件处理是所有DOM节点中的方法，可以重复监听，但是浏览器兼容存在问题；

添加事件监听：

addEventlistener( “click”，fn，是否捕获isCaptrue = true / false默认)

attchEvent 兼容IE

移除事件监听：

第二个参数为必须移除的事件处理函数

oDiv.onclick = function(){ .... } //DOM0级

//DOM2级

if(window.attachEvent){

oDiv.attachEvent("onclick", function(){ ... }); // IE只有冒泡阶段,所以没有第三个参数，而且需要加on；

}else{

oDiv.addEventListener( "click", function(){ ... },false); // false指冒泡阶段

}

oDiv.removeEventListener( "click",fn）

oDiv.detachEvent("onclick",fn)

事件委托

事件委托就是利用事件冒泡，只指定一个事件处理程序，就可以处理某一类型的所有事件

使用场景主要用于事件源不确定的情况，可以把事件委托给父级。

特点：

1. 减少了事件绑定浏览器重绘的次数，提高了程序的执行效率；

2、减少了事件的亢余绑定，节约了事件资源；

3、可以解决动态添加元素节点无法绑定事件的问题；

判断事件源：

e.target || e.srcElement

拖拽

<body>

<div></div>

<script>

var div = document.querySelector("div");

div.onmousedown = function (e) {

e = e || event;

// 记录鼠标跟元素的偏移量

var disX = e.offsetX,

disY = e.offsetY;

document.onmousemove = function (e) {

console.log(disX, disY);

e = e || event;

// 获取鼠标坐标

div.style.left = e.clientX - disX + "px";

div.style.top = e.clientY - disY + "px";

}

document.onmouseup = function () {

// 停止拖动 --- move事件解绑

document.onmousemove = null;

}

return false; // 阻止默认圈选文字事件

}

</script>

</body>

正则表达式

regExp：定义字符串规则的表达式

写法：

字面量/reg/；

var reg = new RegExp；

test方法：

该方法用来测试某个字符串是否与正则匹配，匹配就返回true，否则返回false；

compile方法：

该方法的作用是能够对正则表达式进行编译，被编译过的正则在使用的时候效率会更高，适合于对一个正则多次调用的情况下，如果对一个正则只使用一两次，那么该方法没有特别显著的效应；

exec方法：

返回的是一个数组，数组元素为匹配的子字符串

[ ] 中括号：匹配其中的某一个字符

( ) 小括号：分组：小括号里面的内容作为整体进行匹配；小括号、竖线不要放在 [ ] 内（无意义）

|：或，跟js中的(||)一样

^：排除（除了） 类似js中的（!）

^：(不在中括号里)匹配字符串开头

$ 匹配结尾

/^ $/ 完整匹配全等匹配

**转义字符**：元字符

\d ---- [0-9] 数字

\w -----[a-z0-9\_A-Z] 数字，字母，下划线

\s -----空白字符

\D -----[^0-9] 非数字

\W -----[^a-z0-9\_] 非数字，字母，下划线

\ S -----非空白字符

. -----全部字符

\b -----匹配单词边界

\B -----匹配 非 单词边界

\0(数字0) -----匹配 NUL 字符

\n -----匹配 换行符

\f -----匹配 换页符

\r -----匹配 回车符

\t -----匹配 制表符

\v -----匹配 垂直制表符

**量词**

{n} ---匹配n次

{n,m} ---最少n次，最多m次；

{n,} ---最少n次，最多不限

+ {1,}

? {0,1} 可有可无，最多一个

\* {0,} 可以有也可以没有，个数不限

i：忽略大小写

g：全局查找

m：多行查找

**直接量字符**

\/ 匹配 /

\\ 匹配 \

\. 匹配 .

\\* 匹配 \*

\+ 匹配 +

\? 匹配 ?

\| 匹配 |

\( 匹配 (

\) 匹配 )

\[ 匹配 [

\] 匹配 ]

\{ 匹配 {

\ } 匹配 }

\’ 匹配 单引号

\” 匹配 双引号

\xxx 查找以八进制数 xxx 规定的字符

\xdd 查找以十六进制数 dd 规定的字符

\uxxxx 查找以十六进制数 xxxx 规定的 Unicode 字符

\u4e00 - \u9fa5 验证中文

贪婪模式和惰性模式

贪婪： /[ab]+b/ 结果：abbbb

惰性： /[ab]+?b/ 结果：ab

**支持正则的字符串方法**

search

查找第一次匹配的子字符串的位置，如果找到就返回一个number类型的index值，否则返回-1

replace

该方法用来将字符串中的某些子串替换为需要的内容，接受两个参数，第一个参数可以为正则或者子字符串，表示匹配需要被替换的内容，第二个参数为被替换的新的子字符串

split

将一个字符串拆分成一个数组，它接受一个正则或者子字符（串）作为参数，返回一个数组

match

接受一个正则作为参数，用来匹配一个字符串，返回一个数组。

正则扩展

1.u

ES6对正则表达式添加了u修饰符，含义为“Unicode模式”，用来正确处理大于\uFFFF的Unicode字符。也就是说，会正确处理四个字节的UTF-16编码。

/^\u{61}$/.test('a') // false {61}会识别成量词

/^\u{61}$/u.test('a') // true {61}识别大括号的Unicode字符

2.y (“粘连”修饰符)

ES6添加了y修饰符和g类似，不过y 修饰符在下次匹配的时候需要紧跟上次匹配成功之后的结果匹配，确保匹配必须从剩余的第一个位置开始，这也就是“粘连”的含义。而g则是全局匹配，只需要剩余位置中存在匹配就可;

实际上，y 修饰符号隐含了头部匹配的标识 ^，y修饰符的设计本意，就是让头部匹配的标识 ^ 在全局匹配中都有效。

exec( ) 方法在匹配全局对象的时候，多次匹配会在上一次结束的地方继续匹配;

ES5严格模式

进入严格模式 "use strict";

严格模式行为变更：

1.全局变量声明时 必须加var

2.this 无法指向全局对象;

3.函数内重名属性；

4.arguments对象，arguments对象不允许被动态改变;被自调用;

5.新增保留字； implements, interface, let, package, private, protected, public, static, yield。

ES6

声明变量

let/const：

块级作用域：一种普遍存在于各个语言中的作用域范围，let不能提升声明。

暂时性死区：

ES6规定在某个区块中，一旦用let或const声明一个变量，那么这个区块就变成块级作用域，用let 或者const声明的变量即为该区块绑定，该变量不受任何变量影响。在该变量使用let声明前不可以用。在语法上，我们叫这种情况为：暂时性死区 (temporal dead zone，简称 TDZ)。

Const：

声明常量；引用类型可以修改内部结构,但是不能修改引用。

扩展运算符...：

三个点号，功能是把数组或类数组对象展开成一系列用逗号隔开的值；

rest运算符：

也是三个点号，不过其功能与扩展运算符恰好相反，把逗号隔开的值序列组合成一个数组；

字符串扩展

1、字符串的 Unicode 表示; 规则为\u + 四位十六进制;

2、字符串的新增方法:

2.1 repeat( )重复功能

'x'.repeat(3) //xxx；

重复字符串;

2.2 includes( )、startsWith( )、endsWith( );判定字符串中是否存在某个字符串;

eg：var s = 'Hello world!';

s.startsWith('Hello') // true 以参数开头

s.endsWith('!') // true 以参数结尾

s.includes('o') // true 包括参数;

第二种方法接受第二个参数，第二个参数表示从第几位开始;

3、for of

一种新的遍历方式;for of 可以用于遍历字符串：

**字符串模板扩展**：

ES6中存在一种新的字符串， 这种字符串是 以 ` ` (波浪线上的那个字符 > 反引号)括起来表示的；

拼接一个带有标签的字符串， 是用这样的方式:

bianliang + " <strong>这是一个文字" + obj.name + "</strong> " + bianliang；

1、用 ${ } 扩住变量让拼接变得非常容易，解析成变量；

2、在字符串中使用 `反引号的时候我们需要进行转义；

3、模板还可以调用函数；

箭头函数

使用箭头函数： var 函数名 = 参数 => 运算规则;：

参数只有一个时可以省略括号，运算规则只有一句话时可以省略大括号和return。

1、箭头函数自绑this， this = window；function的this = function的父元素，

2、箭头函数不能New；

3、箭头函数如果要返回一个JSON对象，json函数用小括号把键值对括起来 “（{}）”，或者大括号+ return “{return{ }}”；

4、自动绑定this；

解构赋值

从数组和对象中提取值，对变量进行赋值，这被称为解构（Destructuring）

var arr = [2,4,5,6];

var [a,b] = arr;

console.log(a,b);

var [p1, {userName,ag:[a,b]}] = json;

作用：

1. 一次性可以定义多个变量；
2. 可以作用在函数传参上，以对象的方式传递，参数顺序也无需保持一致；
3. 可以轻松实现两个数的交换；
4. 可以实现剩余和扩展运算；

Symbol类型

var s1 = Symbol()不等于var s2 = Symbol();，只有状态进行比较

const TYPE = {

SMALL: Symbol(),

MIDDLE: Symbol(),

LARGE: Symbol() //他们的值都不相等

}

Set和Map结构

Set集合，本质上就是对数组的一种包装；Set集合是默认去重复的，但前提是两个添加的元素严格相等 。

去重：var newarr = [...new Set(array)];

Set遍历方法：set集合没有下标for循环。

var imgs = new Set(['a','b','c']);

//根据KEY遍历

for(let item of imgs.keys()){

console.log(item);

} //a //b //c

//根据VALUE遍历

for(let item of imgs.values()){

console.log(item);

} //a //b //c

//根据KEY-VALUE遍历

for(let item of imgs.entries()){

console.log(item);

} //['a','a'] //['b','b'] //['c','c']

//普通for...of循环(for...of跟for-in的区别很明显，就是直接取值，而不再取下标了)

for(let item of imgs){

console.log(item);

} //a //b //c

SET集合没有提供下标方式的访问，因此只能使用for来遍历。

Map集合,即映射。

获取：map.get（）

循环遍历，配合解构赋值 for(let [key,value] of map){ console.log(key,value); }

兼容问题

1. 事件对象的创建；
2. 事件冒泡
3. 浏览器的默认行为
4. 事件委托中事件源的获取

面向对象、原型和this

创建对象的方式

1、字面量 var obj = {}

2、通过new运算符 var obj = new Object()

3、构造函数

用来构造（创建）对象的函数

他在声明的时候跟普通函数没有区别

用new运算符加上函数的调用，调用的结果就是一个对象

构造函数中的this指的是即将要new的那个对象

1. ES6语法糖

原型

prototype原型是函数的伴生体，是实例共享方法和属性的地方；

创建的每个函数都有一个prototype(原型)属性，这个属性是一个指针，指向一个对象，而这个对象的用途是包含可以由特定类型的所有实例共享的属性和方法prototype（原型）属性指向的对象就是原型对象。

**属性和方法：**

constructor prototype里面的constructor指向当前对象的构造函数

\_\_proto\_\_ === [[prototype]] 实例的\_\_proto\_\_指向父类的prototype

prototype 指向当前对象的原型对象

instanceof 运算符，判断当前对象是否是另一个对象的实例

hasOwnProperty 判断对象上是否存在某个属性，并且这个方法会过滤到原型上的属性

isPrototypeOf 检查一个对象是否存在于另一个对象的原型链上

**原型注意点：**

1.改写面向对象程序，要让所有函数不能嵌套；

2.提取变量，让函数正常运行；

3.改写面向对象，将方法写在prototype上；

4.注意this指向问题。

对象的特点

封装、继承、多态（重载、重写）

**封装**：写对象用对象；

**继承**：子对象可以使用父对象的属性和方法；

**多态**：（重载）就是根据不同参数类型，参数个数实现不同的功能。JS本身不支持重载；

（重写）父类的方法不好用，子元素重新定义一个方法名相同的方法。

类和对象的关系： --> 模板 --> 产品;

Class

构造函数constructor say，构造函数是一个模板，每new一次构造函数就会被调用一次。

语法糖：

class Fn {

constructor (name) {

this.name = name;

}

say () {

console.log(this.name);

}

}

var num=new Num（“name”）;

num.say();

OOA、OOD、OOP

**OOA面向对象分析**

面向对象分析: 将大问题拆分成小问题，并试图用分工协作来完成 的思维方式。

面向对象不是编程独有思维方式。

面向对象分析方法（Object-Oriented Analysis，OOA），是在一个系统的开发过程中进行了系统业务调查以后，按照面向对象的思想来分析问题。OOA与结构化分析有较大的区别。OOA所强调的是在系统调查资料的基础上，针对OO方法所需要的素材进行的归类分析和整理，而不是对管理业务现状和方法的分析。

OOA（面向对象的分析）模型由5个层次（主题层、对象类层、结构层、属性层和服务层）和5个活动（标识对象类、标识结构、定义主题、定义属性和定义服务）组成。在这种方法中定义了两种对象类之间的结构，一种称为分类结构，一种称为组装结构。分类结构就是所谓的一般与特殊的关系。组装结构则反映了对象之间的整体与部分的关系。

**OOD面向事件设计**

1、分析模块后，确定职责。

2、确定耦合关系；

3、为OOP做准备。

**OOP面向事件编程**

将内容包含在对象中 --> 信息传递速度更快，效率更高

系统升级 --> 通过增加和删除组件实现需求。（某个软件中的组件）

This

事件函数里面的this指的是事件触发对象；

他是根据执行上下文（执行环境）动态指向当前调用的对象；谁调用，就指谁；

**This的指向**：

1. 全局this指向window；
2. 对象方法的this指向对象的本身；（谁调用就指谁）
3. 构造函数的this指即将new的对象本身；
4. 自调用函数指向window； “IIFE，立即执行函数(function( ){})( )”
5. 事件里的this指向事件触发事件；
6. 箭头函数没有自己的this；
7. 定时器的this指向window；

call()、apply()、bind() 可以改变this的指向。

call传参依次传参，第一个参数数对应第二个数的下标；

obj1.say.call(obj2, 1, 2);

apply传参是一个数组，第一个参数对应数组的下标；

obj1.say.apply(obj2, [2,2]);

bind在函数封装绑定bind；bind绑定谁就是谁

符号

<< &laquo

>> &raquo

JavaScript

PHP+MySQL

应用程序分类：

B/S：browser（浏览器）/server（服务器）；

C/S：Client（客户端）/ server（服务器）

http协议：基于TCP / IP的无状态通信协议

前端（浏览器客户端）向后端（Web服务器端）发送http请求获取数据；

**http请求报文require**

请求报文包括：请求行（request line）、请求空行、请求头部（header）、请求数据。

|  |  |  |
| --- | --- | --- |
| Form的enctype属性编码方式 | application/x-www-form-urlencoded（默认） | multipart/form-data |
| action为get | x-www-form-urlencoded的编码方式把form数据转换成一个字符串（name1=value1&name2=value2...）把这个字串追加到url后面，用?分割，加载这个新的url。 | |
| action为post | 浏览器把form数据封装到http body中，然后发送到server。  没有type=file的控件，用默认的application/x-www-form-  urlencoded，有type=file的话，就用multipart/form-data。 | |
| action为post且Content-Type类型是multipart/form-data | 浏览器会把整个表单以控件为单位分割，并为每个部分加上Content-Disposition(form-data或者file),Content-Type(默认为text/plain),name(控件name)等信息，并加上分割符(boundary)。 | |

**http相应报文response**

Response Headers里的Content-Type指服务器告诉浏览器相应的数据，类型有：

1、application/x-www-form-urlencoded：数据被编码为名称/值对。这是标准的编码格式；

2、multipart/form-data： 数据被编码为一条消息，页上的每个控件对应消息中的一个部分；

3、text/plain： 数据以纯文本形式(text/json/xml/html)进行编码，其中不含任何控件或格式字符；

Status-Code表示服务器发回的响应状态代码；Reason-Phrase表示状态代码的文本描述。状态代码由三位数字组成，第一个数字定义了响应的类别，且有五种可能取值：

1xx：指示信息--表示请求已接收，继续处理。

2xx：成功--表示请求已被成功接收、理解、接受。

3xx：重定向--要完成请求必须进行更进一步的操作。

4xx：客户端错误 -- 请求有语法错误或请求无法实现。

5xx：服务器端错误 -- 服务器未能实现合法的请求。

常见状态代码、状态描述的说明如下：

200 OK：客户端请求成功。

400 Bad Request：客户端请求有语法错误，不能被服务器所理解。

401 Unauthorized：请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用。

403 Forbidden：服务器收到请求，但是拒绝提供服务。

404 Not Found：请求资源不存在，举个例子：输入了错误的URL。

500 Internal Server Error：服务器发生不可预期的错误。

503 Server Unavailable：服务器当前不能处理客户端的请求，一段时间后可能恢复正常。

PHP

PHP、Java、ASP(Mac：XAMP)等服务器程序;

PHP（HyperText Preprocessor）是一种创建动态交互性站点的强有力的服务器端脚本语言。

**PHP语法**

<?php

// PHP 代码

?>

**PHP格式**

1. 变量必须以$开头；
2. 拼接符号(一个点) . ；
3. 注释与JavaScript一致；
4. 用echo或者print打印；

**超级全局变量**

这种变量在脚本的全局都可以使用：

- $GLOBALS

- $\_SERVER

**- $\_REQUEST**  收集表单提交的信息

**- $\_POST** 收集表单提交数据

**- $\_GET**

- $\_FILES

- $\_ENV

- $\_COOKIE

- $\_SESSION

**PHP数组**

索引数组、关联数组（用=>连接eg：$testArray = array('name' => "zhangsan",'age' => 17 )）；

手动数组赋值$arr[0] = 11;

遍历数组：foreach（）

forEach ($arr as $key => $value)

（$arr：数组、as：数组中找到的值赋给后面的变量）

获取数组长度：函数count( )；

数组排序：sort（）:升序排列； rsort() :降序排列；

返回JSON

json\_encode()方法进行json格式的转码；

json\_decode()方法对json进行解析;------php

JSON.parse(str); json序列化，将符合json格式的字符串转换为json

JSON.stringify( ); json转换为字符串

json是一种数据格式，由对象，数组互相嵌套组成，属性名必须加双引号，数据传输格式；

cookie直接存中文会造成未知错误，所以一般使用encodeURIComponent()进行编码，decodeURIComponent进行解码。

Json对象：[{“name”:”18”}......];

XML对象：

<json>

<item>

<name>zhangsan</name>

<age>18</age>

</item> //<item>.....

</json>

数据库

数据库：按照数据结构来组织、存储和管理数据的仓库。

RDBMS：关系型数据库（MongoDB 非关系型数据库）

DB2

Oracle

SQLServer--MSSQL

MySQL

SQL：结构化查询语言

DDL：数据定义语言

DCL：数据控制语言

DML：数据操纵语言，INSERT（添加）、UPDATE（修改）、DELETE（删除）

DQL ：数据查询语言，SELECT（查询）

CRUD - 数据：添加（ Create）、查询（ Retrieve）、修改（ Update）、删除（ Delete）

概念：

数据库：存储数据的仓库

表：仓库里面的房间

列（字段）：房间里的箱子

行（记录）：箱子里的数据

主键：唯一标识记录

外键：

**创建数据库**

打开My sql软件--单击文件--打开登录信息--名称--host里面填写localhost，端口不变--确定；

右键localhost--新建--数据库--名称--字符集选择utf-8--确定；

名称右键--新建--表格--名称--类型--字符集utf-8--确定；

名称右键--新建--字段--名称--类型--是否允许空；

**PHP连接数据库操作**

1. 跟数据库创建连接

mysql\_connect(servername,username,password);

（servername 服务器名、username 登录数据库的用户名、password 密码）

1. 选择数据库

mysql\_select\_db(dbname)；

1. 设置编码

4、编写SQL语句

5、执行SQL语句

$result = mysql\_query($sql);

对 INSERT、UPDATE、DELETE 的SQL语句，返回的是布尔值：true表示SQL语句执行成功，false表示SQL语句执行失败。

对 SELECT ，返回的是查询的结果集（虚拟的表格）

6、处理执行结果

对于查询结果集，可以使用 mysql\_fetch\_assoc($result) 来读取查询结果集中的一行，放置到数组中保存

7、关闭连接

mysql\_close( );

例：

mysql\_connect("localhost","root","");

mysql\_select\_db("test");

mysql\_query("set charset 'utf8'");

mysql\_query("set character set 'utf8'");

$sql = "SELECT \* FROM user";

$result = mysql\_query($sql);

while ($arr =mysql\_fetch\_assoc($result)) {

echo $arr['name']." : ".$arr['age'];

}

mysql\_close( );

**使用MySQL**

a. 安装

b. 启动MySQL数据库服务

c. 使用Navicat管理MySQL数据库：NAVH-WK6A-DMVK-DKW3

1. 建立连接

2. 创建数据库

3. 在数据库中创建表

数据类型：

数字：int double decimal numeric

字符串：char varchar text

日期时间：date、time、datetime、timestamp

4. CRUD

**插入**：

INSERT INTO 表名(列名) VALUES (列值)

**修改**：

UPDATE 表名

SET 列名=新列值, 列名=新列值

WHERE 条件

注意，不给修改的条件（WHERE子句）则会修改整张表的数据，这种修改 是不可撤销的修改

**删除**：

DELETE FROM 表名

WHERE 条件

注意，不给删除的条件（WHERE子句）则会删除整张表的数据，这种删除 是不可撤销

**查询**：

SELECT 列名

FROM 表名

WHERE 条件

ORDER BY 排序列 ASC|DESC

LIMIT 起始索引, 限定行数

ASC-升序

DESC-降序

SELECT id, name, birthday

FROM students

ORDER BY id ASC

LIMIT 3, 3

模糊查询：%

Cookie会话跟踪技术

一个网站从打开到浏览（包括这个网站的其他子页面）到最后关闭浏览器整个过程叫一个“会话”。

cookie的存取：document.cookie

<script>

function setCookie(){

document.cookie="username=zhangsan"; //cookie是字符串

}

function getCookie(){

console.log(document.cookie);

}

</script>

**Cookie的时效**

expires

var d = new Date();

d.setxxx(); //将d设置到过期的日期

document.cookie = "username=zhangsan;expires="+d;

例：

<script>

function setCookie(){

var date = new Date;

//有效期设置成明天

date.setDate(date.getDate()+1);

document.cookie="username=zhangsan;expires="+date;

}

function getCookie(){

console.log(document.cookie);

}

</script>

function setCookie(){

var useinfo = {

"username":"zhangsan",

"age":"81",

"password":"123456"

}

document.cookie="userename="+JSON.stringify(useinfo);

}

function getCookie(){

console.log(document.cookie);}

**Cookie的特点**

1、只能使用文本；

2、单挑储存只有4KB；

3、数量限制，一般50 条左右；

4、读取有域名限制，不可跨域读取，只能由来自写入cookie的统一域名的网页可以进行读取； 5、时效限制，每个cookie都有时效，最短的有效期是，会话级别：就是当浏览器关闭，那么cookie立即销毁。

6、cookie直接存中文会造成未知错误，所以一般使用encodeURIComponent()进行编码，decodeURIComponent进行解码。

Web storage浏览记录

localStorage中的键值对总是以字符串的形式存储。(需要注意：和js对象相比，键值对总是以字符串的形式存储意味着数值类型会自动转化为字符串类型)无论数据存储在localStorage还是sessionStorage，它们都特定于页面的协议。

只读的localStorage属性允许你访问一个Document源（origin）的对象Storage；其存储的数据能在跨浏览器会话保留。localStorage类似sessionStorage，但其区别在于：存储在localStorage的数据可以长期保留；而当页面会话结束——也就是说，当页面被关闭时，存储在sessionStorage的数据会被清除。

即：用JSON.stringify( )或JSON.parse( )进行转换。

大小：5MB

Sessionstorage 会话storage （只在会话期内有效）；

Localstorage 本地 storage （只要不删除就有效）；

例：record：记录、storage：储藏

var record = localStorage.getItem("record");

            if(record){

                //有浏览记录,就在之前(头部)加入，再存进去

                record = JSON.parse(record);

                record.unshift(detail);

                localStorage.setItem("record",JSON.stringify(record));

            }else{

                //没有浏览记录就创建一个只有一个元素的数组

                localStorage.setItem("record",JSON.stringify([record]));

            }

设置setItem、获取getItem、清除clear、移除removeItem；

例：localStorage.setItem('myCat', 'Tom');

Ajax

同步sync：同步的代码是不能访问异步的代码--undefined；按顺序依次执行；

异步async：异步代码会放在异步队列中，此时同步代码还会执行，当同步代码执行完毕后再运行异步代码，同事分开执行；（定时器setTimeout、setInterva）。

服务器：代码上线就是上传到服务器。

JS是一个单线程的语言，即代码一行一行的执行，上面的执行完在继续下面的代码；

传统的web交互是用户触发一个http请求服务器,然后服务器收到之后,在做出响应到用户,并且返回一个新的页面。

AJAX是一种用于创建快速动态网页的技术。通过在后台与服务器进行少量数据交换，AJAX可以使网页实现异步更新。

AJAX不是单一技术，是一串技术的集合：

1. JavaScript，通过用户或其他与浏览器相关事件捕获交互行为；
2. XMLHttpRequest对象，通过这个对象可以在不中断其它浏览器任务的情况下向服务器发送请求；

3）服务器上的文件，以XML、HTML或JSON格式保存文本数据；

4）其它JavaScript，解释来自服务器的数据（比如PHP从MySQL获取的数据）并将其呈现到页面上。

AJAX的请求：post、get、put

AJAX的优点：

1）不需要插件支持（一般浏览器且默认开启JavaScript即可）；

2）用户体验极佳（不刷新页面即可获取可更新的数据）；

3）提升Web程序的性能（在传递数据方面做到按需放松，不必整体提交）；

4）减轻服务器和带宽的负担（将服务器的一些操作转移到客户端）；

Ajax的不足：

1）不同版本的浏览器度XMLHttpRequest对象支持度不足(比如IE5之前)；

2）前进、后退的功能被破坏（因为Ajax永远在当前页，不会记录前后页面）；

3）搜索引擎的支持度不够（因为搜索引擎爬虫还不能理解JS引起变化数据的内容）；

创建ajax 为post对象：

var ajax = new XMLHttpRequest();

ajax.open("POST","url",true);

**ajax.setRequestHeader("Content-type","application/x-www-form-urlencoded");**

ajax.send("name=zhangsan&age=18");

ajax.onreadystatechange = function(){

if(ajax.readyState == 4 && ajax.status == 200){

var json = JSON.parse(ajax.responseText);

fn(json);

}

创建ajax为get对象：



<script>

// 1、创建一个ajax核心对象

var ajax = new XMLHttpRequest();

// 2、open方法打开连接（请求方式get\post，请求的地址url，是否异步true代表异步）

// 作为get请求如果要传输数据，就拼接在url后面

ajax.open("GET", "ajax.php?id=2", true);

// ajax.open("GET", "data.json", true);

// 3、send方法，发送请求

ajax.send(null);

// 4、监听状态的变化

ajax.onreadystatechange = function () {

if(ajax.readyState === 4 && ajax.status === 200){

// 交易成功

//console.log(ajax.responseText);

var json = JSON.parse(ajax.responseText)

console.log(json);

}

}

</script>

AJAX的状态值：ajax.readyState

0 － （未初始化）还没有调用send()方法

1 － （载入）已调用send()方法，正在发送请求

2 － （载入完成）send()方法执行完成，已经接收到全部响应内容

3 － （交互）正在解析响应内容

4 － （完成）响应内容解析完成，可以在客户端调用了

HTTP状态值：ajax.status;

1\*\*：请求收到，继续处理

2\*\*：操作成功收到，分析、接受

3\*\*：完成此请求必须进一步处理

4\*\*：请求包含一个错误语法或不能完成

5\*\*：服务器执行一个完全有效请求失败

101——客户要求服务器根据请求转换HTTP协议版本

200——交易成功

304——客户端已经执行了GET，但文件未变化

306——前一版本HTTP中使用的代码，现行版本中不再使用

400——错误请求，如语法错误

404——没有发现文件、查询或URl

405——用户在Request-Line字段定义的方法不允许

500——服务器产生内部错误

505——服务器不支持或拒绝支请求头中指定的HTTP版本

同源策略

同源：协议、域名、端口号完全一致；

默认端口号：https:443；http:80、

解决资源跨域访问：

a. CORS

cross-origin resource sharing

服务器端：

设置响应头信息： Access-Control-Allow-Origin:\*

前端：

ajax

可解决GET/POST跨域需求

b.jsonp

<script src=""></script>

利用<script>在引入外部JS时不受同源策略限制的特性，来实现跨域。(src的开放性原则)

JSONP只能处理GET请求方式的跨域

后端：

由服务器端构建一个字符串：字符串中的内容是能够在JS中执行的函数调用的结构

前端：

1.创建<script>元素

2.设置src属性，传递callback参数指明全局回调函数的名称

3.添加到body中

4.创建全局函数，用于处理响应数据

5.删除<script>元素

c.xhr2

例：

ajaxJsonp : function(url,cb,query){

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

url+="?cb"+cb;

if(query){

for(var key in query){

url+="&"+key+"="+key[query];

}

}

script.src=url;

document.body.appendChild(script);

document.body.removeChild(script);

}

JSONP接口

淘宝：https://suggest.taobao.com/sug?code=utf-8&q=关键字&callback=回调函数名

百度：https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su?wd=关键字&cb=回调函数名

Bootstrap

Bootstrap 是最受欢迎的 HTML、CSS 和 JS 框架，用于开发响应式布局、移动设备优先的 WEB 项目。

Bootstrap：v3兼容IE8+

**全局样式**

1、bootstrap 中用到一些 HTML元素和CSS属性需要将页面设置为 HTML5 文档类型，即在页面顶部添加“<!DOCTYPE html>”

2、布局容器：Bootstrap 需要为页面内容和栅格系统包裹一个 .container或container-fluid（占据全部视口viewport的容器）容器。

3、栅格系统，Bootstrap 提供了一套最多12列的流式栅格系统，通过 .row表示行 和 .col-xs-4 这种表示宽度的列快速创建栅格布局。

4、Bootstrap 排版、链接样式设置了基本的全局样式。

栅格系统----用于页面布局的

Grid system 固定的格子设计版面布局

Container---固定宽度且居中，把页面放在一个.container（固定宽度且居中）的盒内；

container-fluid ---流式布局，把页面内容放在一个宽度为100%的盒内；

（1）临界值



（2）栅格原理

1) 一个row最多能容纳12个栅格单位，超出会换行;

2) 对于sm、md、lg，视口大于临界值，水平布局，小于临界值会垂直布局；

3) 对于xs，列块始终是水平布局；

（3）混合使用

当xs，sm， md， lg混合使用：

1) 垂直布局临界值以小屏幕的优先

2) 区块宽度以大屏幕优先

（4）列重置

为解决列高度不一致可能导致的问题，可以加入

<div class="row">

<div class="col-md-6"></div>

<div class="col-md-6"></div>

<div class="clearfix visible-xs"></div>

clearfix 清浮动

visible-xs 只有在xs生效尺寸范围内(<768px)显示

（5）列偏移

col-sm-offset-4 向右偏移4个sm单位

col-md-offset-2 向右偏移2个md单位

col-xs-offset-5 向右偏移5个xs单位

列偏移：通过margin实现

（6）列嵌套

子列宽度是相对于父级盒的宽度

（7）列排序

Col-md-push-3 列相对于原来的位置向右偏移3个md单位；

Col-md-pull-9 列相对于原来的位置向左偏移9个md单位；

闭包closure

全局变量：在函数体外定义的变量，每个函数都能使用修改，就会造成全局污染。

局部变量：在函数体内定义的变量，只有当前函数能使用，但是不能在全局重复使用。

闭包优点：技能重复使用，又不会被全局污染；

闭包缺点：外层函数的活动对象不能被释放，消耗资源；

闭包作用：闭包可以在全局函数里面操作另一个作用域的局部变量；

**创建闭包有三步**：

1、外层函数嵌套内层函数

2、内层函数使用外层函数的局部变量

3、把内层函数作为外层函数的返回值

闭包closure：

function fn(){

var num = 1;

function foo(){

num++;

return num;

}

return foo();

}

var fn1 = fn(); //foo()

var num = fn1();//num

**应用：点击每一个li弹出自己的下标**

方法：let、IIFE、闭包closure、自定义属性、array.form(forEach)即for循环。

<ul>

<li>1</li>

<li>1</li>

<li>1</li>

<li>1</li>

</ul>

<script>

//let

var li=document.querySelectorAll("li");

for(let i=0; i<li.length; i++){

li[i].onclick=function(){

alert(i);

}

}

// for循环

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

li[i].index=i;

li[i].onclick=function(){

alert(this.index);

}

}

//闭包

var lis = document.querySelectorAll("li");

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

lis[i].onclick = (function(index){

return function(){

alert(index);

}

})(i)

}

//IIFE

var lis = document.querySelectorAll("li");

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

(function (index){

lis[index].onclick = function(){

alert(index);

}

})(i)

}

</script>

promise（ES6）

用于表示一个异步操作的最终状态（完成或失败），以及其返回的值。它允许你为异步操作的成功和失败分别绑定相应的处理方法。这让异步方法可以像同步方法那样返回值，但并不是立即返回最终执行结果，而是一个能代表未来出现的结果的promise对象。

一个 Promise有以下几种状态:

pending: 初始状态，既不是成功，也不是失败状态。

resolved（fulfilled）: 意味着操作成功完成。

rejected: 意味着操作失败;

then承诺兑现时执行的函数

catch承诺失信时执行的函数

finally都执行

let a = 20;

// 许下了一个承诺，两秒之后a的值大于30

let pro = new Promise((resolve, reject) => {

setTimeout(() => {

if(a < 30){

resolve();

}else{

reject();

}

}, 2000);

});

pro.then(() => {

// 承诺兑现的时候要执行的代码

console.log("yes");

}).catch(() => {

// 承诺失信的时候要执行的代码

console.log("no");

}).finally(() => {

console.log("finally");

})

new Promise((resolve, reject) => {

let ajax = new XMLHttpRequest();

ajax.open("get", "data.php", true);

ajax.send(null);

ajax.onreadystatechange = function () {

if(ajax.readyState === 4){

if(ajax.status === 200){

resolve(ajax.responseText);

}else{

reject();

}

}

}

}).then((data) => {

console.log(data);

})

Promise.all(iterable)

-- iterable : 可迭代对象（数组）

-- 返回Promise对象

-- 当数组中所有Promise对象都完成时，回调执行成功的函数，当只要有一个执行失败时，就回调执行失败的函数。

继承

封装、继承、多态（重载和重写）。Es6：class extends/Object create()

继承：子类共享父元素的属性和方法。有以下几种方法：

1 **原型链继承**：子类的原则指向父类的实例；

特点：

非常纯粹的继承关系，实例是子类的实例，也是父类的实例；

父类新增原型方法/原型属性，子类都能访问到；

简单，易于实现；

缺点：

要想为子类新增属性和方法，必须要在new Animal()这样的语句之后执行，不能放到构造器中；

无法实现多继承；

来自原型对象的引用属性是所有实例共享的；

创建子类实例时，无法向构造函数传参；

2 **构造继承**（call、apply继承）；

特点：

解决了1中，子类实例共享父类引用属性的问题；

创建子类实例时，可以向父类传递参数；

可以实现多继承（call多个父类对象）；

缺点：

实例并不是父类的实例，只是子类的实例；

只能继承父类的实例属性和方法，不能继承原型属性/方法；

无法实现函数复用，每个子类都有父类实例函数的副本，影响性能；

3 **组合继承**（原型链继承和构造继承的结合）

特点：

弥补了方式2的缺陷，可以继承实例属性/方法，也可以继承原型属性/方法

既是子类的实例，也是父类的实例

不存在引用属性共享问题

可传参

函数可复用

缺点：

调用了两次父类构造函数，生成了两份实例（子类实例将子类原型上的那份屏蔽了）

4 **拷贝继承**

特点：

支持多继承

缺点：

效率较低，内存占用高（因为要拷贝父类的属性）

无法获取父类不可枚举的方法（不可枚举方法，不能使用for in 访问到）

5 **寄生组合继承**

放在自运行函数里

特点：堪称完美

缺点：实现较为复杂

**继承语法糖**

class cat extends Animal{

constructor(){

Supper();

}

....... //重写 (可以写自己的方法)

}

设计模式

**设计模式**：代码经验的总结。

**工厂模式**：工厂函数就是做一个对象创建的封装，并将创建的对象return出去。

function newObj (name, age) {

var obj = new Object();

obj.name = name;

obj.age = age;

return obj;

}

console.log(newObj("zhangsan", 18));

**单例模式**：只允许存在一个实例的模式（闭包）

// 外层函数自调用，得到返回值为内层函数，所以getInstance就是内层函数

var getInstance = (function() {

var date = new Date();

function instance () {

return date;

}

return instance;

})();

// 反复调用getInstance得到的是外层函数活动对象里的同一个date

// 闭包

getInstance();

**观察者模式**：（事件监听）

**策略模式**：定义一系列的算法，把它们一个个封装起来，并且使它们可以相互替换，从而避免很多if语句，曾经学过最简单的策略模式雏形就是使用数组的方式解决传入数字得到对应星期几问题的算法。

**代理模式**：

**设计模式的六大原则**

开闭原则、

里氏代换原则、

依赖倒转原则、

接口隔离原则、

迪米特法则，又称最少知道原则、

合成复用原则

模块化

函数封装-----对象-----立即执行函数-----模块化规范

是一种项目的构架模式， 这种构架模式让JS代码重用性变得非常高，让项目构架的一些复杂问题全部得以解决。即多个script标签就不会出现了。

**模块化的规范**：AMD规范、CMD规范、common规范（基于node.js）。

CMD：就近依赖；

AMD：依赖前置；

这些规范目标一致都是为项目实现模块化，期中有遵循AMD规范的 require.js 和遵循CMD规范的sea.js 。

规范：模块的定义和使用方式；

1.首先对于依赖：

AMD提前执行(不管什么时候用，先加载过来)

特点：前期消耗网络资源大，但是后期运行效率高.

CMD需求执行(有用我再去加载)

特点：整个自选消耗曲线比较平缓。

2.CMD推崇依赖就近，AMD推崇依赖前置。

3.AMD的API默认是一个当多个用，CMD的API严格区分，推崇职责单一。

**引入**：requery([ ],function( ){ }) 可以有多个模块，不加后缀名，形参：引入模块的返回值。

**定义**：define(function( ){ } return)

**配置**：requery.config({baseUrl:” 绝对路径 ”,paths：{“ 文件名 ”：” 路径 ”}})

gulp

Node.js 是基于chrome V8引擎的JavaScript运行环境。

Node.js使用了一个事件驱动、非阻塞式I/O的模型，使其轻量与高效。

gulp安装：

LTS安装稳定版；

Current安装最新版。

gulp是基于Nodejs的自动任务运行器，她能自动化地完成 javascript/coffee/sass/less/html/image

/css 等文件的的测试、检查、合并、压缩、格式化、浏览器自动刷新、部署文件生成，并监听文件在改动后重复指定的这些步骤。

gulp：自动化构建工具。Gulp基于node.js。

使用流程：

安装nodejs -> 全局安装gulp -> 项目安装gulp以及gulp插件 -> 配置gulpfile.js -> 运行任务

使用：<http://www.ydcss.com/archives/18>

操作流程：

**创建package.json文件**

1 在当前文件目录下git bash here；

2 初始化git init；

3 依次根据需求修改参数；

或者 npm init -y（全部yes）。

**创建package-lock.json，生成 node\_modules 文件夹**

在文件下创建gulp 即npm i gulp@3

**在根目录创建gulpfile.js文件**

通过requery引入，const来申明； 即const gulp = require('gulp'),

**制定任务**

gulp task( ‘css’ , () =>{

//src取源文件

// pipe管道（文件传输的过程，可以在过程中对文件做处理

// dest destination 目的地，管道里处理完成以后放到目标文件夹里

gulp.src('src/css/\*.scss')

.pipe(gulpSass())

.pipe(minifyCss())

.pipe(gulp.dest('dist/css'))

}) -----------制定css任务

**安装gulp插件**（压缩css. 压缩HTML. 压缩js）

1. 安装 nodejs

node -v

npm -v

使用 npm 安装包：node package manager

npm install <package-name> -g （--save-dev）

-g 全局安装

--save 局部安装并保存到package.json配置中

-dev 存在package.json的devDependencies配置项里，意思是生产环境依赖的模块

文件：package.json(NodeJS项目的配置文件)

选择性安装 cnpm （http://npm.taobao.org/）:

npm install -g cnpm --registry=https://registry.npm.taobao.org

可以使用 cnpm 替代 npm 来安装资源

2. 全局安装 gulp

npm install gulp@3 -g

或：

cnpm install gulp@3 -g

测试：gulp -v

以下步骤在项目目录下操作：

3. 在项目目录下生成 package.json 文件：

npm init或

cnpm init

4. 在项目目录中本地安装 gulp：

cnpm install gulp --save-dev

本地安装成功后，会生成 node\_modules 文件夹

5. 在项目目录中本地安装 gulp 插件 (https://www.npmjs.com)：

压缩CSS：gulp-minify-css

cnpm install gulp-minify-css --save-dev

压缩JS：gulp-uglify

cnpm install --save-dev gulp-uglify

压缩Html：gulp-htmlmin

cnpm install --save-dev gulp-htmlmin

**命令行**

进入方式：

1）win-R CMD

2）powerShell

3）Git

常用命令：

1 查看是否安装成功：node -v

2 进入环境：node

3 退出nide：ctrl-v

4 进入某一目录：cd

5 新建文件：touch

6 新建目录：mkdir

7 删除文件：rm

gulpfile.js

将ES6转换为ES5：gulp-babel

cnpm install --save-dev gulp-babel @babel/core @babel/preset-env

服务器：gulp-connect

cnpm install --save-dev gulp-connect

文件合并：

cnpm install --save-dev gulp-concat

监听文件的的变化执行对应的任务：watch；

给每一个任务加上pipe(connect.reload())；

处理图片：

gulp.task("img", function(){

gulp.src('src/images/\*\*/\*')

.pipe(gulp.dest('dist/images'))

})

gulp.task("default", ["sass", "html", "minijs", "connect", "watch", "img"]);

sass转css：gulp-sass

cnpm install --save-dev gulp-sass

gulpfile.js中新增代码：

var connect = require('gulp-connect');

gulp.task('server', function() {

connect.server({

livereload: true,

port: 2333

});

});

gulpfile.js中修改js任务

var babel = require('gulp-babel');

gulp.task('js', function(){

gulp.src('src/\*\*/\*.js')

.pipe(babel({

presets: ['@babel/env']

}))

.pipe(uglify())

.pipe(gulp.dest('dist/js'))

});

const gulp = require('gulp'),

htmlmin = require('gulp-htmlmin'),

minifyCss = require('gulp-minify-css'),

sass = require('gulp-sass'),

uglify = require('gulp-uglify'),

babel = require('gulp-babel'),

connect = require('gulp-connect');

// 制定一个html任务

gulp.task('html', () => {

// 压缩html

gulp.src('src/\*\*/\*.html')

.pipe(htmlmin({

removeComments: true,//清除HTML注释

collapseWhitespace: true,//压缩HTML

collapseBooleanAttributes: true,//省略布尔属性的值 <input checked="true"/> ==> <input />

removeEmptyAttributes: true,//删除所有空格作属性值 <input id="" /> ==> <input />

removeScriptTypeAttributes: false,//不删除<script>的type="text/javascript"

removeStyleLinkTypeAttributes: true,//删除<style>和<link>的type="text/css"

minifyJS: true,//压缩页面JS

minifyCSS: true//压缩页面CSS

}))

.pipe(gulp.dest('dist'))

.pipe(connect.reload());

})

// 制定css任务

gulp.task('css', () => {

// 编译scss

// 压缩css

// 除了module里面的scss都要编译

gulp.src(['src/css/\*\*/\*.scss', "!src/module/\*.scss" ])

.pipe(sass())

.pipe(minifyCss())

.pipe(gulp.dest('dist/css'))

.pipe(connect.reload());

})

// 制定js任务

gulp.task('js', () => {

// ES6转ES5

// 压缩

gulp.src('src/js/\*\*/\*.js')

.pipe(babel({

presets: ['@babel/env']

}))

.pipe(uglify())

.pipe(gulp.dest('dist/js'))

.pipe(connect.reload());

})

// 制定libs任务

gulp.task('libs', () => {

// 移动

gulp.src('src/lib/\*\*/\*').pipe(gulp.dest('dist/lib'));

})

// 制定images任务

gulp.task('images', () => {

// 移动

gulp.src('src/img/\*\*/\*').pipe(gulp.dest('dist/img'));

})

// 监听

gulp.task('watch', () => {

gulp.watch('src/\*\*/\*.html', ['html']);

gulp.watch('src/css/\*\*/\*.scss', ['css']);

gulp.watch('src/js/\*\*/\*.js', ['js']);

})

// 开服务器

gulp.task('server', () => {

connect.server({

root: 'dist',

port: 1902,

livereload: true

});

})

gulp.task('default', ['html', 'css', 'js', 'libs', 'images', 'server', 'watch']);

art-template模板引擎

在HTML中引入template-web.js文件；

<script type=”text/html”></script>;

第一个参数就是即将要渲染的script的id

第二个参数script里面需要的数据

返回值就是渲染完成的html字符串

轮播图

swiper.js

不基于jQuery AMD规范

垫片

不满足AMD规范的插件又依赖别的插件就需要添加垫片

shim：{“ ” :{}}

放大镜

jQuery.elevatezoom插件

zoom () {

$(".zoom-img").elevateZoom({

gallery:'gal1',

cursor: 'pointer',

galleryActiveClass: 'active',

borderSize:'1',

borderColor:'#888'

});

}

fly

加购物车以抛物线的形式加至购物车：jQuery.fly.js

start/end/onEnd

Node.js

浏览器地址栏输入地址后会发生的：

1 DNS域名解析

2 查找主机IP

3 TCP三次握手

4 http请求

5解析入口文件

6 加载 渲染

7 渲染完成 TCP四次挥手断开连接

jQuery是一个类库

绑事件：$(“bottom”).on(“click”,function(){})

hide()隐藏

show()显示

toggle()切换

conuect.reload重启服务器

Jq中可发起ajax请求的方法有：

$.ajax、$.get、$.post、$.getJSON

Jq中发起添加类名的方法是：addClass;

Jq中实现动画的方法是：$().animate;

Jq中延迟动画执行的方法是：delay；

SVN提交命令是commit

SVN同步服务端仓库的命令是checkout

jq绑定事件的方法有：bind、one、on

Jq选择器的结果是对象

Jq事件委托的绑定：on

Jq中关于dom增删改的操作有：append、appendTo、remove、detach；

Jq中设置样式$().css

**安装**

node安装：下载安装、nvm、Chocolaty

**常用属性**

1.OS系统

2.FS系统文件

writeFile 写文件

​ appendFile 追加文件

​ readFile 读文件（读出的文件是一个16#的流，用buffer 可转成正常的文字）

3.http模块（可以创建一个服务器）

http.createServer((req,resp), =>{

}.listen()

4.path路径

path.resolve()

**server**

node.js模块规范是`commonJS`;

1 创建一个服务器（接受请求require，返回数据响应response）；

//引入核心模块

const http = require("http");

http.createServer = ((req,resp) =>{

console.log(req.url); //路径

resp.write("");

resp.end()

}).listen(8080); //监听 服务器端口号

2 配置路径渲染模块

模块化配置一个router对象；

导出(暴露)模块：module.exports = router

// 路由

const router = {

"/" : "index.html",

"/list": "list.html",

"/detail": "detail.html",

"/index": "../index1.html"

}

// 导出（暴露）模块

module.exports = router

3 配置地址引入

通过path.resolve()配置，上一目录或者引入目录

// 引入需要依赖的核心模块

const http = require('http'),

fs = require('fs'),

path = require('path'),

router = require('./router')

// request response

http.createServer((req, resp) => {

if(req.url === '/favicon.ico'){

resp.end();

return;

}

fs.readFile(path.resolve('./static', router[req.url]), (err, data) => {

if(err) throw err; //抛出异常

else resp.end(data); //渲染数据

})

}).listen(8080)

express

概念：express 是基于Node.js平台，快速、开放、极简的 Web 开发框架(Node.js框架)\*\*

​ express是一个保持最小规模的灵活的Node.js Web应用程序开发框架，为Web和移动应用程序提供一组强大的功能。使用您所选择的各种 HTTP 实用工具和中间件，快速方便地创建强大的 API。

官网：http://www.expressjs.com.cn/

安装：npm install express --save

**Express应用程序生成器**

通过应用生成器工具 express-generator 可以快速创建一个应用的骨架。

新建文件myapp

1 假定你已经安装了 Node.js，接下来为你的应用创建一个目录，然后进入此目录并将其作为当前工作目录。

mkdir myapp

2 进入myApp文件夹

cd myapp

3 通过 npm init 命令为你的应用创建一个 package.json 文件。

npm init -y

4安装express

npm install express --save

5 main.js 通过node main命令，在浏览器输入端口号访问

// 引入express模块

const express = require('express')

// 创建express的实例

const app = express()

// 处理来自根目录的get请求

app.get('/', (req, resp) => resp.send('Hello World!'))

// 监听3000端口

app.listen(3000, () => console.log('Example app listening on port 3000!'))

\*\*新建ExApp文件\*\*

- express-generator 包含了 express 命令行工具。通过如下命令即可安装：

```js

npm install express-generator -g

```

- 帮助参数

```js

express -h

```

- 回上一级（exApp文件夹里）

```js

cd ..

```

- 创建了一个名称为 myapp 的 Express 应用。此应用将在当前目录下的 exApp 目录中（文件名自定）创建,并且设置为使用 ejs模板引擎；

```js

express --view=ejs exApp

```

- 进入目录

```js

cd exApp/

```

- 安装npm install

```js

npm i

```

- 执行express

```js

npm start

```

- 执行前可以自行配置端口号(8090)

```js

port=8090 npm start

```

\*\*app.js文件夹内容的注释\*\*

```js

// 引入http报错模块

var createError = require('http-errors');

// 引入express模块

var express = require('express');

// 引入核心path模块

var path = require('path');

// 引入处理cookie的模块

var cookieParser = require('cookie-parser');

// 引入日志模块

var logger = require('morgan');

// 引入路由文件模块

var indexRouter = require('./routes/index');

var usersRouter = require('./routes/users');

// 创建express实例

var app = express();

// 设置模板引擎， 如果要加载视图文件，就去views目录

// \_\_dirname 当前路径 C:\1902-node\express-project\exApp\views

app.set('views', path.join(\_\_dirname, 'views'));

// 设置模板引擎为 ejs

app.set('view engine', 'ejs');

// app.use 中间件

// 使用日志的中间件

app.use(logger('dev'));

// 使用处理json格式数据的中间件

app.use(express.json());

// 使用解析url的中间件

app.use(express.urlencoded({ extended: false }));

// 使用cookie解析器的中间件

app.use(cookieParser());

// 处理静态文件的中间件（静态资源从public里面获取）

app.use(express.static(path.join(\_\_dirname, 'public')));

// 使用路由

app.use('/', indexRouter);

app.use('/users', usersRouter);

// 处理404错误

app.use(function(req, res, next) {

next(createError(404));

});

// 处理500错误

app.use(function(err, req, res, next) {

// set locals, only providing error in development

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

// render the error page

res.status(err.status || 500);

res.render('error');

});

// 暴露app

module.exports = app;

```

**自动重启**

nodemon

1 全局安装 npm install -g nodemon;

2 在scripts添加自己的nodemon;

```js

{

"scripts": {

"start": "node ./bin/www",

"st": "nodemon ./bin/www"

}

}

```

3 运行代码：npm run st（属性名）.

**ejs 模板引擎**

ejs:类似前后端使用的air-template模板引擎。

官网：https://ejs.bootcss.com/

书写方式：<%= ? %> \*\*`语句中没有等号`\*\*

mongoDB

mongoDB是一个\*\*非关系型数据库\*\*

mySQL是一个\*\*关系型数据库\*\*

mongoDB官网：https://www.mongodb.com/

安装教程：https://www.runoob.com/mongodb/mongodb-window-install.html

下载网址（V3.1.4）：https://www.mongodb.com/download-center/community

\*\*安装完成需要的设置：\*\*

（1）设置环境变量：把bin的目录设置到环境变量中

（右键计算机--高级系统设置--高级--环境变量）

（2）新建自己的服务器运行的地址

（3）开启服务：

​ 1 在bin目录下选择命令行（或者git bash）；

​ 2 命令行中输入：

mongod --dbpath 自己新建的文件路径

```js

mongod --dbpath 自己新建的文件路径

```

\*\*什么是MongoDB？\*\*

- MongoDB是一个文档数据库，具有您需要的可查询性和索引所需的可伸缩性和灵活性。

- 文档模型映射到应用程序代码中的对象，使数据易于使用。

- 即席查询，索引和实时聚合提供了访问和分析数据的强大方法。

- MongoDB是一个分布式数据库，因此内置了高可用性，水平扩展和地理分布，并且易于使用。

##### 可视化工具 robo 3T

还有Navicat MongoDB等其他可视化工具。

官网：https://robomongo.org/

下载：https://robomongo.org/download

\*\*默认端口号port：27017\*\*；

安装可能出现的问题

win 7 下安装robo 3t遇到“无法启动程序，引文计算机丢失api-ms-win-crt-runtime-|1-1-1.dll”;

解决办法：安装VC\_redist;

##### mongoose包

官网：https://mongoosejs.com/

\*\*mongoose：基于node.js的优雅mongodb对象建模.\*\*

##### 使用mongoose

- 接下来使用npm以下命令从命令行安装Mongoose ：

```js

npm install mongoose

```

- 我们需要做的第一件事是在项目中包含mongoose，并test在我们本地运行的MongoDB实例上打开与数据库的连接。

```js

// getting-started.js

// 引入模块

const mongoose = require('mongoose');

// 连接数据库

mongoose.connect('mongodb://localhost/test', {useNewUrlParser: true});

```

- 我们与在localhost上运行的测试数据库有挂起的连接。如果我们成功连接或发生连接错误，我们现在需要收到通知：

```js

//得到了连接实例

let db = mongoose.connection;

// 处理连接错误输出

db.on('error', console.error.bind(console, 'connection error:'));

//监听一次打开事件

db.once('open', function() {

// we're connected!

});

```

- 使用Mongoose，一切都来自Schema。

Schema：把非关系型数据库转换成关系型结构。

```js

let kittySchema = new mongoose.Schema({

name: String,

...

});

```

- 一个带有一个属性的模式name，它将是一个 String。下一步是将模式编译为模型。

```js

var Kitten = mongoose.model('Kitten', kittySchema);

```

- 模型是用于构造文档的类。

```js

var silence = new Kitten({ name: 'Silence' });

console.log(silence.name); // 'Silence'

```

- 向MongoDB保存内容

```js

// 存数据 异步方法

silence.save(function (err, silence) {

if (err) return console.error(err); // 报错

fluffy.speak();

});

```

- 查找

```js

// model的kitten

Kitten.find(function (err, kittens) {

if (err) return console.error(err);

console.log(kittens);

})

```

##### mongooseDB增删改查

```js

// 引入mongoose模块

const mongoose = require('mongoose');

// 连接本地数据库服务器，打开1902数据库

mongoose.connect('mongodb://localhost/1902', {useNewUrlParser: true});

// 得到连接实例

let db = mongoose.connection;

// 处理连接错误输出

db.on('error', console.error.bind(console, 'connection error:'));

// 监听一次打开事件

db.once('open', function() {

console.log("we're connected!");

});

// schema把非关系型数据库转换成关系型结构

let usersSchema = new mongoose.Schema({

username: String,

password: String

});

// 根据usersSchema得到了一个model,这个model是一个抽象，他是没有值的，只是代表这个collection的抽象的模型

let Users = mongoose.model('Users', usersSchema);

const userModel = {

/\*

\* 插入document

\* @param userInfo <object> {username, password}

\*/

insert: (userInfo) => {

//往模型当中去填值，得到一个实例的document

let user = new Users(userInfo)

return new Promise((resolve, reject) => {

// 存数据

user.save((err, user) => {

if(err) reject(err);

else resolve(user);

})

})

},

/\*

\* @param where 查询条件 <object>

\* @param isOne 是否查一条 默认为false

\*/

find: (where, isOne = false) => {

if(isOne) {

return new Promise((resolve, reject) => {

Users.findOne(where, (err, docs) => {

if(err) reject(err)

else resolve(docs)

})

})

}else{

return new Promise((resolve, reject) => {

Users.find(where, (err, docs) => {

if(err) reject(err)

else resolve(docs)

})

})

}

},

/\*

\* 删除

\* @param where <object> 删除条件

\* @param [isOne] <Boolean> 是否删一条 默认值true

\*/

delete: (where, isOne=true) => {

return new Promise((resolve, reject) => {

if (isOne) {

// 只删除一条

Users.deleteOne(where, err => {

if(err) reject(err);

else resolve();

})

}else {

Users.deleteMany(where, err => {

if(err) reject(err);

else resolve();

})

}

})

},

/\*

\* 更新

\* @param where <object> 更新条件

\* @param updatedData <object> 更新之后的数据

\* @param [isOne] <Boolean> 是否一条 默认值true

\*/

update: (where, updatedData, isOne = true) => {

return new Promise((resolve, reject) => {

if(isOne) {

Users.updateOne(where, updatedData, (err, res) => {

if(err) reject(err);

else resolve(res);

})

}else {

Users.updateMany(where, updatedData, (err, res) => {

if(err) reject(err);

else resolve(res);

})

}

})

}

}

// 暴露出去（导出模块）

module.exports = userModel;

```

**MVC 框架**

模型module 视图 控制器controller

controller 渲染到前端

module 写的逻辑

db 引入mongoose模块，连接服务器

用promise来处理异步( ES7: await / async )

前后端跨域

服务器与服务器之间不存在兼容问题，只有浏览器之间才存在兼容问题

**什么是跨域?**

当协议、域名、端口有其中任意一项不一致时则为跨域（非同源）。

**前端解决跨域**

txt

a.CORS

前端：ajax（解决get/post跨域需求）

后端：设置要请求的源 设置相应开头信息： Access-Control-Allow-Origin:\*（PHP）

b.jsonp

利用<script>在引入外部JS时不受同源策略限制的特性，src的开放性原则来实现跨域。

前端：

1. 创建 <script> 元素

2. 设置 src 属性，传递 callback 参数指明全局回调函数的名称

3. 添加到 body 中

4. 创建全局函数，用于处理响应数据

5. 删除 <script> 元素

（在src里面写接口地址）

后端：

由服务器端构建一个字符串：字符串中的内容是能够在 JS 中执行的函数调用的结构

**原生ajax**

// 创建一个ajax原生对象

var ajax = new XMLHttpRequest();

// open方法打开连接（请求方式get\post，请求的地址url，是否异步true代表异步

ajax.open("POST","url",true);

// send方法，发送请求

ajax.send("name=zhangsan&age=18");

// 监听状态的变化

ajax.onreadystatechange = function(){

if(ajax.readyState == 4 && ajax.status == 200){

var json = JSON.parse(ajax.responseText);

fn(json);

}

**postman工具**

官网：https://www.getpostman.com/

作用：

​ 1.检测接口；

​ 2.当返回数据慢时可以通过postman工具检测接口所需要的时间，方便判断是前端渲染数据慢或者是后端的接口问题；

\*\*jquery托管在线地址\*\*： http://code.jquery.com/jquery-3.2.1.min.js

**实现跨域**

- cors

- 安装cors跨域插件

- npm i cors

- 引入相关模块

```js

// 引入模块

var express = require('express');

var path = require('path');

const cors = require('cors');

// 创建express的实例

var app = express();

// 使用中间件 use方法

app.use(cors());

// list.html 需要引入jQuery文件

$.get('/api/goods?subject\_id=5571&page=1&size=3', function (resp) {

console.log(resp);

})

// list.js

router.post('/', function(req, res, next) {

res.json({

"res\_code": 200,

"res\_body": [

{"id":1, title: "商品1"},

...

]

});

});

```

- 代理（proxy）

- 代理跨域

```html

http-proxy-middleware

```

- 安装插件

```html

npm install --save-dev http-proxy-middleware

```

- 引入依赖

- 根据options创建一个代理的实例

- 使用中间件

- 代理配置项

- 设置目标源的主机

- 修改源为true

- 把前端的地址重写为真实接口的地址

```js

// 引入依赖

const express = require('express')

const proxy = require('http-proxy-middleware')

const path = require('path')

// 根据options创建一个代理的实例

var goodsProxy = proxy(options)

// 得到一个express实例

var app = express()

// 设置默认访问静态页面public \_\_dirname为当前目录

app.use(express.static(path.join(\_\_dirname, 'public')));

// 使用goodsProxy代理中间件

app.use('/api', goodsProxy);

app.listen(3000);

// 代理配置项

var options = {

target: 'https://apiv2.pinduoduo.com', // 目标源的host

changeOrigin: true, // 修改源

ws: true, // 常连接

pathRewrite: {

// 把前端的虚拟地址/api/goods重写为真实接口地址/api/fiora/subject/goods

'^/api/goods': '/api/fiora/subject/goods'

},

router: {

// when request.headers.host == 'dev.localhost:3000',

// override target 'http://www.example.org' to 'http://localhost:8000'

'dev.localhost:3000': 'http://localhost:8000'

}

}

// HTML部分

$.get('/api/goods?subject\_id=5571&page=1&size=3', function (resp) {

console.log(resp);

})

})

```

登录

登录流程：

- 浏览器向后台发送post请求提交登录信息

- 后台接收用户名密码，访问数据库验证登录信息

- 登录成功后台将会向前端发送一个随机的字符串" token "，返回前端存在cookie里面

- 前端跳转页面，前端首页会验证是否登录成功，就会带上cookie，发送给后端对比数据库的登陆信息是否匹配，若是匹配则登录成功（可能：则把用户名密码返回到首页）

- 前端首页就会显示登录状态

**cookie**

存cookie

npm install cookie-session

引入模块使用

var cookieSession = require('cookie-session')