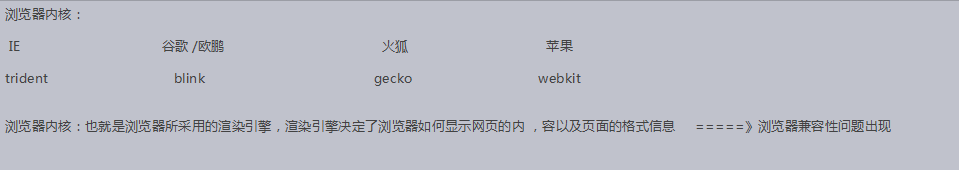
1.注释：快捷键：Ctrl+/ &nbsp; 空格 &lt; < &gt; >

2.!+tab 可以写成html5结构



3.渲染引擎是兼容性问题出现的根本原因

4.浏览器通过http协议向服务器发送请求，服务器上IIS （web服务器）提供网页浏览服务，将网页的内容解析出来，响应报文传至浏览器上。

5.meta 的属性有两种：name和http- equiv

name属性主要用于描述网页，对应于content（网页内容）

<meta name="KEYWords" contect="">向搜索引擎说明你的网页的关键词；

<meta name="DEscription" contect="">告诉搜索引擎你的站点的主要内容；

◆网页重定向

<meta http-equiv="refresh" content="5; **错误！超链接引用无效。**

6.link标签：

◆ 调用外部样式表

<link rel=”stylesheet” type="text/css" href=”1.css”>

◆icon图标

<link rel=”icon” type="text/css" href=”1.ico”>

|  |  |  |  |
| --- | --- | --- | --- |
| **<a>** | 定义一个超链接 | Href=“链接的网址”  title=“提示的文字”  target=“链接在何处打开” | Line |
| **<abbr>** | 定义缩写 | Title=“缩写前的全称” | Line |
| **<area>** | 定义热点，总是嵌套在<map>中实用   1. <img src="images/2s.jpg" usemap="#name1" /> 2. <map name="name1"> 3. <area shape="circle" coords="0,0,50" href="#"> 4. </map> | Alt：定义替换的文字  Cords：热点坐标值  圆形（circle）：（x，y，r）  X,y定义了远点坐标，R半径  方形（rect）：（x1,y1,x2,y2,）  X1,y1:定义了左上角顶点  2x,y2:定义了右下角顶点  (网页图片左上角坐标是：0，0)  Href：目标的链接  Target：何地打开链接  图片的usemap属性要和map的那么属性一样，并且图片的usemap属性值要加井号# |  |
| **Base** | 为页面内所有链接规定默认地址或者默认目标  位于head内部 | Href=“链接地址URL”  Target=“在何地打开链接” |  |
| **Bdo** | 定义文字方向 | Dir=“ltr（左）/rtl（右）” |  |
| **Body** | 定义文档的主题 |  |  |
| **Button** | 定义按钮 | Disabled=“disabled”禁用此按钮  Name：按钮名称  Type=“sibmit/reset/button”  提交/重置/普通按钮  Value=“按钮的初始值” | Line |
| **Caption** | 定义表格标题，必须紧跟table标签，只能对每个标签定义一个标题，通常居中与表格之上 |  |  |
| **Code** | 定义计算机代码文本 |  |  |
| Col | 为一个或多个列设置属性 | Span：规定行横跨的列数  Align水平对其方式  Valign：垂直对其方式 |  |
| **Div** | 定义文档中的分区 |  |  |
| **Dl，dt，dd** | 自定义列表 |  |  |
| **Form** | 表单标签 |  |  |
| **Frame** | 定义一个框架 |  |  |
| **Frameset** | 定义一个框架集 | Col：定义框架的列数  Rew：定义框架水平行 |  |
| **Img** | 定义图像 | Src=“图像路径”  Alt=“替换文本”  Title=“提示文字”  Width、height |  |
| **Input** | 定义输入控件 | Type=“text/button/password/checkbox/radio/submit/reset”  文本/按钮/密码/复选框/单选框/提交按钮/重置按钮  Value=“元素的默认值” |  |
| **Lable** | 为input元素定义标注，  Lable属性的for应与被绑定的元素的id元素相同 |  |  |
| **Link** | 定义文档与外部资源的联系，最常见的用途是链接样式表 | Charset=“链接文档的字符编码方式”  Href=“被链接文档的路径”  Rel=“当前文本与被链接的文本的关系”  Rev=“定义被链接的文本与当前文本的关系” |  |
| **Map** | 定义热点链接 | Id/name |  |
| **Meta** | 定义文档的元数据 | Content=“内容”  http-equiv=“把content属性关联到http头部”  Name=“description/keywords/” |  |
| **Object** | 定义内嵌对象 |  |  |
| **Ol、ul、** | 定义有序/无序列表 |  |  |
| **Select>Option** | 定义下拉列表 | Selected=“selected”定义默认被选中项  Value=“定义被发往服务器的值”  Disabled=”disabled”被禁用的项 |  |
| **Span** | 定义文档中的行内元素 |  | Line |
| **Style** | 定义内嵌式样式表 | Type=“text/css” |  |
| **Sub/sup** | 定义下标/上标文本 |  |  |
| **Table/tr/td** | 定义表格 | Border=“边框” |  |
| **Th** | 定义表头单元格 | <tr><th></th></tr> |  |
| **Textarea** | 定义多行文本输入框 | Cols=“文本可见宽度”  Rows=“文本可见高度”  Readonly=“readonly”规定只读 |  |
| **Tfoot** | 定义表格页脚 |  |  |

## HTML标准属性

|  |  |  |
| --- | --- | --- |
| **属性** | 值 | 说明 |
| **Class** | Classname | 规定元素的类名 |
| **Id** | Idname | 规定元素的唯一ID |
| **Style** |  | 规定元素的行内样式 |
| **Title** | Text | 规定元素的额外信息 |
| **语言属性** |  |  |
| **Dir** | Ltr/lt | 规定文本的方向 |
| **键盘属性** |  |  |
| **Accseekey** |  | 设置访问元素的键盘快捷键 |
| **Tabindex** |  | 设置元素的tab键控制次序 |

## http方法

在客户机和服务器之间进行请求-响应时，两种最常被用到的方法是：GET 和 POST。

### get：从指定的资源请求数据

有关 GET 请求的其他一些注释：

* GET 请求可被缓存
* GET 请求保留在浏览器历史记录中
* GET 请求可被收藏为书签
* GET 请求不应在处理敏感数据时使用
* GET 请求有长度限制
* GET 请求只应当用于取回数据

### Post：向指定的资源提交要被处理的数据

有关 POST 请求的其他一些注释：

* POST 请求不会被缓存
* POST 请求不会保留在浏览器历史记录中
* POST 不能被收藏为书签
* POST 请求对数据长度没有要求

### 比较 GET 与 POST

|  |  |  |
| --- | --- | --- |
|  | GET | POST |
| 后退按钮/刷新 | 无害 | 数据会被重新提交 |
| 书签 | 可收藏为书签 | 不可收藏为书签 |
| 缓存 | 能被缓存 | 不能缓存 |
| 编码类型 | application/x-www-form-urlencoded | application/x-www-form-urlencoded 或 multipart/form-data。为二进制数据使用多重编码。 |
| 历史 | 参数保留在浏览器历史中 | 参数不会保存在浏览器历史中。 |
| 对数据长度的要求 | 是的。当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）。 | 无限制 |
| 对数据类型的限制 | 只允许 ASCII 字符。 | 没有限制。也允许二进制数据。 |
| 安全性 | 与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分。 | POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 web 服务器日志中 |
| 可见性 | 数据在 URL 中对所有人都是可见的。 | 数据不会显示在 URL 中。 |

## 文本格式化

<strong>语气加重、加粗

<b>加粗（尽量不用）

<em>文本倾斜

<i>文本倾斜（尽量不用）

**<del>删除线**

**<s>删除线（尽量不用）**

**<ins>下划线**

**<u>下划线（尽量不用）**

**◆注意：之所以使用<string><em><del><ins>，是因为这些标签更具有语义化**

## img图片标签

<img src=”路径” alt=“**替换图片文本，对SEO优化有好处”** title=“提示文本（鼠标放到图片上显示的文字）”

单独更改img的宽或者高 图片等比缩放 不会变形

**同时设置img的宽高会导致图片变形**

没有定义宽高的时候按照图片尺寸的100%的尺寸显示

## ****路径****

●相对路径：相对于自己出发找其他文件

跳出文件夹就要使用 （ . . / ）

★往下一级目录用（ / ） 往上一级目录用（ . . / ）

●绝对路径：文件完整的路径

## 超链接

href：链接到的地址（必写属性）

target：新链接在哪打开 \_blank 在新窗口打开

title：提示文本

<base target=”\_blank”>写在head之中，作用是让所以的超链接都在新窗口打开

Base 作用是使页面内所有超链接指向同一目标或者同一个链接

## 列表

ul 无序列表

★type属性 如list-style:”square”

■ square 实心方块

● disc 实心圆圈

○ circle 空心圆圈

ol 有序列表

★type属性

1、A、a、I、i

★start 从哪个数开始

dl 自定义列表

dl定义自定义列表

dt 定义标题

dd 定义内容

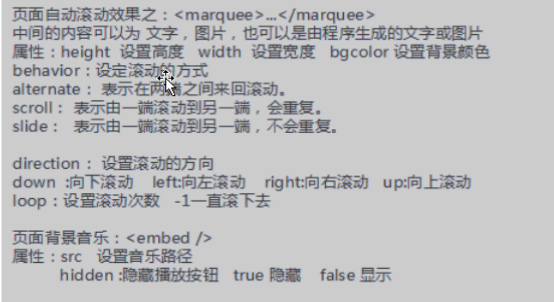
## 音乐标签

Embed

Hidden=“true” 隐藏播放器

## 滚动 marquee

## HTML头部标签



|  |  |
| --- | --- |
| **<head>** | 定义关于文档的信息 |
| **<title>** | 定义文档标题 |
| **<base>** | 定义页面上所有链接的默认地址或默认目标 |
| **<link>** | 定义文档与外部资源的关系 |
| **<meta>** | 定义HTML文档的元数据 |
| **<script>** | 定义客户端脚本 |
| **<style>** | 定义文档的样式信息 |
|  |  |

## Base元素

1. <head>
2. <base href="http://www.w3school.com.cn/images/" />
3. <base target="\_blank" />
4. </head>

Base标签为页面所有链接都指向默认地址或者默认目标

## Link元素

1. <head>
2. <link rel="stylesheet" type="text/css" href="mystyle.css" /> //外链样式
3. <link rel=”shortcut icon（只用icon也可以）” type=“image/x-icon” href=”图标路径”> //网页快捷图标
4. </head>

Link：标签定义文档与外部资源之间的关系

* 1. Rel：属性定义该标签关联的样式表标签
  2. Href：定义该便签所链接的外部资源的URL地址，可相对地址，可以是绝对地址
  3. Type：定义文档类型
  4. Media：设置属性调用的资源主要针对什么设备而使用
     1. Screen：针对计算器机显示器的样式
     2. All：所以的设备
        1. <style type=”text/css”>
        2. @import URL(“css/base.css”);
        3. </style>//不建议使用

## Meta元数据

元数据metadata是关于数据的信息，元数据不会显示在页面上，对与机器是可读的。

Meta元素被用来规定页面的描述、关键词、文档的作者、最后的修改时间以及其他元数据。

Meta标签始终位于head元素中。

元数据可用于浏览器（如何显示内容或重新加载页面），搜索引擎（关键词），或者其他web服务。

◆ 一些搜索引擎会利用meta元素的name和content属性来索引你的页面

下面的 meta 元素定义页面的描述：

1. <meta name="description" content="Free Web tutorials on HTML, CSS, XML" />

下面的 meta 元素定义页面的关键词：

1. <meta name="keywords" content="HTML, CSS, XML" />

下面的http-equiv=“refresh”，定义了网页的重新定向，自动跳转到新网址

1. <meta http-equiv=”refresh” content=”5;www.baidu.com”>

## Script元素

<script>用来定义客户端脚本，典型的JavaScript

## HTML脚本

JavaScript是HTML文档具有更强的动态性和交互性

JavaScript最常用于图片操作、表单验证、动态内容更新

<noscript> 标签提供无法使用脚本时的替代内容，比方在浏览器禁用脚本时，或浏览器不支持客户端脚本时。

noscript 元素可包含普通 HTML 页面的 body 元素中能够找到的所有元素。

只有在浏览器不支持脚本或者禁用脚本时，才会显示 noscript 元素中的内容：

1. <script type="text/javascript">
2. document.write("Hello World!")
3. </script>
4. <noscript>Your browser does not support JavaScript!</noscript>

## HTML字符实体（特殊符号）

在 HTML 中，某些字符是预留的

在 HTML 中不能使用小于号（<）和大于号（>），这是因为浏览器会误认为它们是标签。

在HTML文件中的多个空格会被解释成只有一个空格



## HTML统一资源定位器URL

URL（Unifrom Resource Locator）也被称为网址

统一资源定位器（URL）用于定位万维网上的文档（或其他数据）。

网址，比如 [http://www.w3school.com.cn/html/index.asp](mk:@MSITStore:C:\\Users\\YZZ\\Desktop\\HTML手册.chm::/www.w3school.com.cn/html/index.asp)，遵守以下的语法规则：

scheme://host.domain:port/path/filename

◆解释：

scheme - 定义因特网服务的类型。最常见的类型是 http

host - 定义域主机（http 的默认主机是 www）

domain - 定义因特网域名，比如 w3school.com.cn

:port - 定义主机上的端口号（http 的默认端口号是 80）

path - 定义服务器上的路径（如果省略，则文档必须位于网站的根目录中）。

filename - 定义文档/资源的名称

|  |  |  |
| --- | --- | --- |
| Scheme（服务类型） | 访问 | 用于… |
| http | 超文本传输协议 | 以http://开头的普通网页，不加密 |
| https | 安全超文本传输协议 | 安全网页，加密所有信息交换 |
| ftp | 文件传输协议 | 用于将文件上传或者下载 |
| File |  | 您计算机上的文件 |

## URL编码

URL只能使用ASCII字符集通过因特网进行发送

由于 URL 常常会包含 ASCII 集合之外的字符，URL 必须转换为有效的 ASCII 格式。

URL 编码使用 "%" 其后跟随两位的十六进制数来替换非 ASCII 字符。

URL 不能包含空格。URL 编码通常使用 + 来替换空格。

## HTML WEB Server

**ISP**：因特网服务提供商

大多数小公司会把网站存放到由 ISP 提供的服务器上

## HTML多媒体

Web 上的多媒体指的是音效、音乐、视频和动画。

HTML5多媒体标签

<embed src=“路径” width=“” height=“”></embed>

# XHTML

## 什么是XHTML？

* XHTML 指的是可扩展超文本标记语言
* XHTML 与 HTML 4.01 几乎是相同的
* XHTML 是更严格更纯净的 HTML 版本
* XHTML 得到所有主流浏览器的支持

## 为什么使用 XHTML？

XHTML 是一种必须正确标记且格式良好的标记语言

## 与 HTML 相比最重要的区别

### 文档结构

* XHTML DOCTYPE 是强制性的
* <html> 中的 XML namespace 属性是强制性的
* <html>、<head>、<title> 以及 <body> 也是强制性的

### 元素语法

* XHTML 元素必须正确嵌套
* XHTML 元素必须始终关闭
* XHTML 元素必须小写
* XHTML 文档必须有一个根元素

### 属性语法

* XHTML 属性必须使用小写
* XHTML 属性值必须用引号包围
* XHTML 属性最小化也是禁止的

XHTML 文档必须进行 XHTML 文档类型声明

（XHTML DOCTYPE declaration）。

## 如何从 HTML 转换到 XHTML

1. 向每张页面的第一行添加 XHTML <!DOCTYPE>
2. 向每张页面的 html 元素添加 xmlns 属性
3. 把所有元素名改为小写
4. 关闭所有空元素
5. 把所有属性名改为小写
6. 为所有属性值加引号

# Sublime编辑器

|  |  |
| --- | --- |
| **Html：xt** | **调出HTML基本结构代码** |
| **Ctrl+shift+D** | **快速复制一行** |
| **Ctrl+shift+K** | **快速删除一行** |
| **Ctrl+鼠标左键** | **选择集体输入** |
| **Ctrl+H** | **查找替换** |
| **Ctrl+F** | **查找** |
| **Ctrl+?** | **注释** |
| **Ctrl+L** | **快速选择一行** |
| **Ctrl+shift+↑↓** | **快速上下移动一行** |
| **Alt+shift+12345789** | **界面布局** |
| **F11** | **全屏显示界面** |
| **Ctrl+K+B** | **显示/隐藏侧边栏** |

# HTML第二天 2016.07.29

### !+tab html5文档类型

### Charset 编码格式:和计算机交流，翻译人的语言给计算机听

* <meta charset=”utf-8”>
* ASCII码、Ansi码、Unicode、Gbk 汉字、Gb2315 汉字、Big5繁体字符
* Utf-8 :最大的字符集，包含一百多个国家的文字

### Keywords，关键字,给搜索引擎看的

* <meta name=”keywords” content=”关键字”/>

### 网页描述 description

* <meta name=”description” content=”描述内容”>

### 网页重定向

* http-equiv=”refresh” content=”5;(加分号，跳转时间,单位秒) [www.baidu.com](http://www.baidu.com)(重新定向的网页)”

### link：链接外部资源

* 链接外部CSS文件(stylesheet)、JS文件(script)、icon图标(icon)

### Table表格

* 展示数据
* 对网页重构（css+div）的一个有益补充
* Cellspacing：单元格之间的距离，会撑大表格大小
* Width、height、
* Cellpadding：内容到单元格边框的距离，会撑大单元格大小
* border-collapse:collapse；：合并单元格边框
* Align：left、center、right；表格对齐方式
* 给table加align是确定表格的对齐方式
* 给tr、td加align是确定行、单元格的对齐
* Bgcolor：背景颜色
* Caption：定义表格标题，紧跟table标签，居中显示
* Colspan：合并单元格\_行，不可以和表头表尾合并
* Rowspan：合并单元格\_列，不可以和表头表尾合并
* Valign：top、middle、center 垂直对齐方式
* HTML 表单中有两种类型的单元格：
* 表头单元格 - 包含表头信息（由 th 元素创建）
* 标准单元格 - 包含数据（由 td 元素创建）
* th 元素内部的文本通常会呈现为**居中的粗体文本**，
* td 元素内的文本通常是左对齐的普通文本。
* bordercolor=“red”：改变表格边框颜色

1. <table> //表格标准结构
2. <thead> //表格头部
3. <tr>
4. <td><td>
5. </tr>
6. </thead>
7. <tbody> //表格主题
8. <tr>
9. <td><td>
10. </tr>
11. </tbody>
12. <tfoot> //表格尾部
13. <tr>
14. <td><td>
15. </tr>
16. <tfoot>
17. </table>

### 表单form

* 作用:收集用户输入信息
* **表单的组成**
* 1.提示信息
* 2.表单控件
* 3.表单域：输入内容的区域

**<form action=”提交到地址” method=”get/post”>**

* Action:处理信息
* Method：提交方法
* Get：通过地址栏提交信息，安全性差
* Post：通过文件提交信息，安全性高
* **Input标签：**
* password：密码输入框
* Text：文本输入框

maxlength：最大输入字符

readonly：只读（不能编辑）

disabled：输入框未激活，不可用

name：命名输入框，提交值的时候区分每个输入框

value：初始值，默认显示值，将默认值传给处理文件

* Radio：单选框

Checked=checked：默认选中项

必须使用相同的name，否则会使两项都可以被选中

**Submit/reset：提交/重置 按钮**

* Select>option：下拉列表

Selected：selected默认选项

Multiple：multiple 可多选

Size：定义列表显示的项

Optgroup：对option进行分组

**Hidden=”hidden”：隐藏控件**

1. <select multiple=“multiple” size=“2”>
2. <optgroup liable=”组名”>
3. <option></option>
4. <option></option>
5. <option></option>
6. </optgroup>
7. </select>

* Checkbox：多选框

Checked：checked；默认选项

* **Textarea：多行文本框 单独标签，不需要input**

Cols：每行可输入的长度

Rows：可输入的行数

1. <textarea colw=”30” rows=”10”>

* file文件上传控件
* 按钮

Submit：提交按钮

Button：普通按钮，value=“按钮上的文字”

Image：图片按钮可实现提交功能

1. <input type=”image” src=”路径”>

Reset：重置按钮，将信息重置到默认状态

* Fieldset：表单信息分组

Legend：表单信息分组名称

* **HTML5表单标签**

<Input type=“url”>：网址提交；

<Input type=“date”>：显示日期控件

<Input type=“time”>：时间控件

<input type=”emial”>：邮件控件

<input type=”number” step=“5”(默认1，值是几每次跳几)>：数字控件

<input type=”range” step=“5”>：滑块控件

Step：默认1，值是几每次跳几

* **标签语义化**

根据内容的结构化（内容语义化），选择合适的标签（代码语义化）

好的语义化的网站**标准是去掉样式表后，结构依然很清晰**

* 标签语义化意义
  1. 网页结构合理
  2. 有利于seo和搜索引擎建立良好沟通，有了良好的结构和语义你的网页内容自然容易被搜索引擎抓取。
  3. 方便其他设备解析（如屏幕阅读器、盲人阅读器、移动设备）
  4. 便于团队开发和维护
* 标签语义化（注意事项）
  1. 尽可能少的使用无语义的标签div和span；
  2. 在语义不明显时，既可以使用div和p时，尽量用p，因为p在默认情况下有上下间距，对兼容特殊终端有利。
  3. 不要实用纯样式标签，如：b / font / u等，改用CSS设置
  4. 需要强调的文本，可以包含在strong或者em标签中，strong默认样式是加粗（不要b），em是斜体（不用i）。

# CSS第一天 2016.07.30

**inline-block 常见问题**

1.**inline-block 之间空隙**

inline-block 之间有空格、Tab、换行符。

给父元素设置 font-size: 0，在 inline-block 元素上重新设置 font-size。

2.**inline-block 导致盒子不对齐**（默认以盒子内部文字的基线为参照）

给 inline-block 元素设置 vertical-align: top

通用解决办法 不要设置 inline-block，**使用 float 或 flex。**

**文本溢出加...**

<p class="ellipsis">

唐僧：你想要啊？你要是想要的话你就说话嘛，你不说我怎么知道你想要呢，虽然你很有诚意地看着我，可是你还是要跟我说你想要的。

难道你真的想要吗？你想要的话我会给你的，你想要我怎么可能不给你呢？不可能你想我不给你，你不想要我却偏给你的。

大家讲道理嘛！现在我数三下，你要说清楚你要不要……你真的想要吗？那你就拿去吧！你不是真的想要吧？

</p>

**.ellipsis {**

**overflow: hidden;**

**white-space: nowrap;**

**text-overflow: ellipsis;**

**}**

**多行文本如何居中（flex；table-cell）**

①.ellipsis{

background:green;

display:table-cell;

vertical-align:middle;

height:200px;

}

②.cell{

background:green;

display:flex;

align-items:center;

height:200px;

}

③子绝父相（子元素的尺寸已知）

.ellipsis {

width: 300px;

height: 200px;

padding: 20px;

position: absolute;

top: 50%; left: 50%;

**margin-left: -170px; /\* (width + padding)/2 \*/**

**margin-top: -120px; /\* (height + padding)/2 \*/**

}

④ 上一种情况的变形，记得兼容性

.ellipsis {

width: 140px;

Height:170px;

position: absolute;

top: 50%; left: 50%;

-webkit-transform: translate(-50%,-50%);

-ms-transform: translate(-50%,-50%);

transform: translate(-50%,-50%);

}

**如何用css写出一个三角形**

### <div class="triangle"></div>//对面的三角形去掉

### .triangle {

### width: 0;

### height: 0;

### border-top: 100px solid deepskyblue;

### border-left: 55px solid transparent;

### border-right: 55px solid transparent;}

几种常见图片利弊

**PNG：支持透明，无损压缩。缺点：色彩支持少PNG8/24/32，ie6不支持**

**JPEG：色彩多，有损压缩**

**GIF：动态图，支持透明，体积小。处理256种颜色，不支持半透明，锯齿状效果处理不好.**

**SVG：矢量图形，无损压缩，绘制路径xml语言。缺点浏览器支持不好，主要用于移动端font-face**

**相对定位relative**

元素使用绝对定位，默认以**浏览器为基准，除非它有个父元素是定位元素（relative，absolute，fixed）**

### CSS概念

* CSS（Cascading Style Sheets）：层叠样式表（级联样式表）
* CSS使得结构与样式分离
* 用来美化HTML标签，相当于给页面化妆

|  |  |
| --- | --- |
| **属性** | 解释 |
| **Width：20px；** | 宽 |
| **Height：20px；** | 高 |
| **Background-color：red、#f00、#ff0000** | 背景颜色 |
| **Font-size：20px；** | 文字大小 |
| **Text-align：center、left、right** | 内容水平对齐方式 |
| **Text-indent：2em** | 首行缩进 |
| **Color：red** | 文本颜色 |

### 选择器

1. 选择器{属性：值；}

* **外链样式表**

1. <link rel=”stylesheet” href=”路径” type=“text/css”>

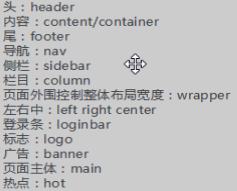
* **内部样式表**

1. <style type=”text/css” >
2. 选择器{属性：值；}
3. </style>

* **内联样式**

1. <p style=“属性一：值一”>行内样式</p>

* 基础选择器
  1. 标签选择器
     1. 标签｛属性：值；｝
     2. 选择所有相同标签名字的标签
     3. 所有被选中标签都会执行相同的样式
  2. ***类选择器***
     1. .自定义类名{属性：值；…… 属性：值；}
     2. 一个标签可以调用多个类选择器
     3. 多个标签可以共用一个选择器
     4. 类命名规则
        1. **不能用纯数字，不能以数字开头**
        2. 不能实用特殊符号，和特殊符号开头，**除 “\_” 外**
        3. 不建议使用汉字来定义类名



不推荐使用属性和属性值来做类名

* 1. ID选择器
     1. #自定义ID名｛属性：值；…… 属性：值；｝
     2. 一个ID选择器在一个页面只能使用一次，如果使用大于一次，不符合W3C规范，JS调用会出问题
     3. 一个标签只能调用一个ID选择器，只能有一个ID
     4. 类选择器和ID选择器不冲突

良好的代码：重用、有序、精简

* 1. 通配符选择器
     1. \*｛属性：值；······ 属性：值；｝
     2. 选择页面所有标签元素。给所有标签相同的样式
     3. 不推荐使用
* 复合选择器：两个或者两个以上的基础选择器通过不同方式连接在一起
  1. 交集选择器
     1. 标签+类（id）选择器｛属性：值；｝

div.box{}

* + 1. 既要满足标签选择器，又要满足类（id）选择器
  1. 后代选择器（重点）
     1. 选择器+空格+选择器+···+选择器+空格+选择器｛属性：值｝
     2. 无限制隔代，选择器可自用组合
  2. 子代选择器
     1. 选择器>选择器｛属性：值；｝
     2. 选择直接子代，不选择子代以后的
  3. 并集选择器
     1. 选择器，选择器，选择器…｛属性：值；｝

### 颜色的显示方式

* 1. 直接写颜色名称
  2. 十六进制显示颜色
     + 0-9 a-f
     + #FFFFFF：红（red）绿（green）蓝（blue）
     + #FFFFFF：白色 #000000：黑色
  3. RGB显示模式color：rgb（255，255，255）；
  4. **RGBA显示模式 color：rgba（255，255，255，0.5）；**
     + A是alpha不透明度，可选值为0-1；
     + 0是完全透明，1是完全不透明

### 文本元素

* Font
* Font-size:文字大小
* Font-weight：值100-900，**700约等于bold（bold不推荐用**），文字粗细
* Font-family：字体，最常用字体（微软雅黑、宋体）
* Font-style：字体风格italic/斜体、normal/正常
* Line-height：行高



font: font-style font-weight font-size/line-height font-family;

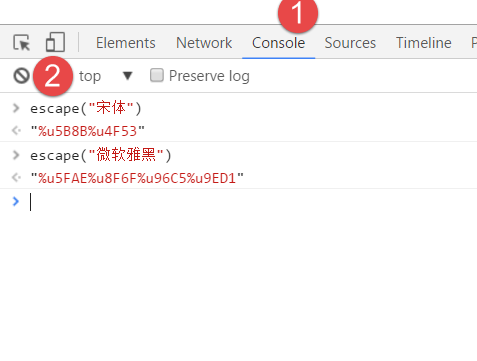
◆：注意：font:后边写属性的值。一定按照书写顺序。

**文本属性连写文字大小和字体为必写项**。

**Font:italic 700 16px/40px 微软雅黑;**

* 文字的表达方式

1. 直接写文字
2. 写字体的英文名称
3. 写字体的Unicode
4. 微软雅黑：Microsoft yahei
5. 宋体：SimSun



# CSS第二天 2016.08.01

### CSS的书写位置

* **内联样式：写在head内部**
* 只能作用当前文件
* 没有真正实现结构表现分离

1. <style type=”text/css”>
2. 选择器｛属性：属性值；｝
3. </style>

* **外链样式：独立的CSS文件，link标签导入**
* 实现表现结构分离
* 作用范围是当前站点，范围广

1. <link rel=”stylesheet” **href**=”CSS文件路径”>

* **italic：斜体**
* **行内样式表：嵌入在行内的**
* 作用范围仅限于当前标签
* 结构表现混在一起
* 不推荐使用

1. <div style=”属性：属性值”></div>

### HTML标签分类（显示方式）：line/block

* **Block：块元素**
* Div/p/h1-h6/li/ul：典型块元素
* 特点：独占一行
* **可定义宽高**，不定义宽度，默认宽度是父级元素的宽度
* 上下左右设置margin、padding都生效
* 不定义高度且无内容，会不显示
* **文字默认大小16px**
* **Line：行内元素**
* a/span/font/strong/：在一行内显示
* **不能直接定义宽高**
* **元素的宽高就是内容的宽高**
* **设置上下margin、padding不生效，只有左右margin、padding生效**
* **Line-block:行内块元素**
* Imput/image：**在一行显示，可设置宽高**

### 块/行内元素之间的转换

* **块元素转换为行内元素**
* Display：inline
* 具备所有行内元素的特点
* **行内元素转换为块元素**
* Display：block；
* 具备所有快元素的特点
* ***块/行内元素转行内块***
* Display：inline-block；
* 空格、回车都会被解释成一个空格

### CSS特性

* **层叠性**
* 当多个样式作用到同一个标签上，样式发生冲突。
* 按照浏览器从上往下的解释顺序，***后来居上***
* 后面的代码覆盖（层叠）前面的代码
* **继承性**
* **需要是包含关系**
* 文字颜色color、文字大小font-size、文字字体font-family、字体粗细font-weight、文字风格font-style 。
* ***文字的所以属性都可以继承***
* **h系列不能继承文字大小**，因为h1-h6有浏览器定义的倍率（em）大小
* a标签不能继承颜色
* **CSS选择器优先级**
* 默认样式<标签选择器<类样式<ID选择器<行内样式<!important

0 1 10 100 1000 1000+

* **继承的属性优先级（权重为0）小于本身的属性**
* 自己为定义属性的时候继承父级属性，本身定义了属性就显示自己的属性
* **权重可以叠加（相加）**

### 链接伪类

##### A:link｛属性：属性值｝；超链接默认状态样式（未访问时）

##### a:visited｛属性：属性值｝；链接访问之后的状态

##### a:hover｛属性：属性值｝；鼠标放到链接之上的状态

##### a:active｛属性：属性值｝；鼠标激活时的链接状态

##### ：focus｛属性：属性值｝；获取焦点

**如果要给所有这些状态都添加样式，请遵循 **LVHFA** 顺序法则。**

6.文本修饰:text-decoration:none/underline/overline/line-through

### 背景属性：background

1. **Background-color：背景颜色**
2. **Background-image：背景图片，同时设置背景图片和背景颜色，图片覆盖颜色**
   1. Background-image：url（“图片路径”）；
3. **Background-repeat：背景平铺（默认平铺：repeat）**
   1. Repeat：平铺
   2. No-repeat：不平铺
   3. Repeat-X/Y：在X/Y轴上平铺
4. **Background-position：背景定位**
   1. 属性值：Left/right/center/top/bottom
   2. 特点：
      1. **当你只写一个值的时候另一个值默认是centr**
      2. 写两个值的时候，值的顺序无要求
      3. 写两个具体值的时候，第一个值代表水平方向，第二个值代表垂直方向
5. **Background-attcchment：背景滚动**
   1. Scroll：滚动，随滚动条而滚动
   2. Fixed：固定，固定位置，不随滚动条滚动

**fixed会使背景定位以浏览器当前页面为参考，scroll则是父元素为参考**

**6.简写：Background：color image repeat position attachment；**

***（url（“路径”））图片必须写***，别的都可以不写

**顺序无要求**

**购物车案例**



**<style>**

**a{**

**display: inline-block;**

**width: 67px;**

**height: 32px;**

**background: url("110.png") ;**

**}**

**a:hover{**

**background: url("110.png") 67px 32px;**

**}**

**</style>**

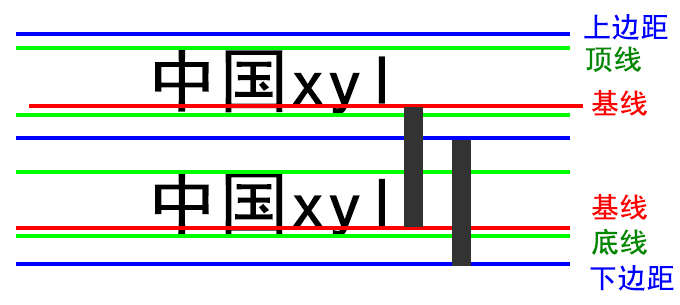
**<body>**

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

**</body>**

# CSS第三天 2016.08.02

### 行高：line-height



1. **行高是文字基线与基线之间的距离，下图两条红色横线就是基线**
2. **行高=文字高度+上下边距**
3. **文字大小等于底线到顶线的距**
4. **一行文字行高等于父元素高度的时候，文字垂直居中显示**
5. 浏览器默认字体大小16px
6. **行高的单位**
   1. Px：
   2. Em：
   3. %：
   4. 数值：

|  |  |  |  |
| --- | --- | --- | --- |
| 单位 | 文字大小 | 行高 | 总大小 |
| Px | 20px | 20px | 20px |
| Em | 20px | 2em | 40px |
| % | 20px | 150% | 30px |
| 数值 | 20px | 2 | 40px |

* 1. 除了以像素为单位，行高都是与文字的乘积

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 单位 | 行高 | 父元素 | | 子元素 | 最终子元素行高 |
|  |  | 文字大小 | 行高大小 | 文字大小 |  |
| Px | **40px** | 20px | 40px | 30px | 40px |
| Em | 2em | 20px | 2em | 30px | 40px |
| % | 150% | 20px | 150% | 30px | 30px |
| 数值 | **2** | 20px | 2 | 30px | 60px |

* 1. **子元素没有设置行高的时候**
     1. **当行高数值由em、%为单位是，子元素的行高是继承父元素的行高乘字体大小的结果**
     2. **当行高数值是px为单位时，子元素直接继承父元素的行高（推荐使用px）**
     3. **当行高无单位直接是数值的时候，行高是和子元素的文字相乘**
  2. **子元素设置行高的时候使用子元素的行高，不继成**

### 盒子模型

1. **盒子的宽度=定义的宽度+边框宽度+左右内边距**
2. **盒子模型：**
   1. **内容：content**，主题内容
   2. **内边距：padding**，内容到边框的距离
      1. **设置内边距会撑大其盒子**
      2. **设置边框会影响其盒子的宽度**
      3. **继承的盒子一般不会被撑大**
         1. **包含（嵌套） 的盒子，如果子盒子没有定义宽度（宽度从父盒子继承来,高度不会继承），给子盒子设置水平内边距（小于父盒子宽度的情况下），不会被撑大，若水平内边距大于父盒子宽度，则会撑大**

**例题.**father{

width: 500px;

height: 300px;

background: #ccc;

}

.son{

height: 100px;

padding-left: 100px;

background: green;

}//子盒子的宽度还是500px

* + - 1. **垂直方向设置内边距撑开其盒子**

**例题.**father{

width: 500px;

height: 300px;

background: #ccc;

}

.son{

height: 100px;

padding-top: 100px;

background: green;

}//子盒子的高度为200px

**例题 一个盒子宽度200px,文字距离盒子内左边距离40px,边框为3px,当前盒子最后的宽度是多大？ 200+40+6**

* 1. **边框：border**，边框的宽度
     1. Border - top/bottom/left/right – width/style/color
     2. 复合属性：宽度 样式 颜色**（没有顺序限制，样式是必写属性）**
     3. Dotted点线 / dashed虚线 solid实线
     4. 如果边框是虚线，虚线部分也显示背景颜色
     5. **盒子大小要加上边框的宽度**
     6. 边框合并：**border-collapse：collapse**
     7. **Td的边框覆盖table的边框**
  2. **外边距：margin**，与另外一个盒子的距离
     1. margin-left/bottom/right/top；
     2. **垂直方向上的外边距合并**
        1. **两个盒子一个设置上外边距，一个设置下外边距，取设置相对较大的值**
     3. **嵌套的盒子，垂直方向外边距塌陷（给子盒子设置上外边距后，父盒子会随着子盒子同时向下移动）**
        1. **解决方法**
           1. **给父盒子设置边框border**
           2. **给父盒子设置：overflow：hidden**

触动BFC：格式化上下文

* + 1. margin：上 右 下 左
    2. 写两个值，第一个代表上下，第二个代表左右
    3. 写三个值，第一个代表上，第二三个代表左右，第三个代表下

#### FireWorks 快捷键

新建文件 ctrl+n

打开文件 ctrl+o

调出和隐藏标尺 ctrl+r

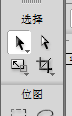
清除辅助线： 视图---辅助线----清除辅助线

放大镜 z 放大镜状态下alt+鼠标左键 缩小

抓手 快捷键 空格

测量距离 ★先拉出2根辅助线

★切换到指针工具



★将光标放到2根辅助线之间，按住shift键

# CSS第四天 2016/08/04

1. **字体大小要放在行高前面**
2. **文档流布局**
   1. 元素自上而下，从左往右，
   2. 块元素独占一行，每个控件后有一个换行符
   3. 行内元素在行内显示，达到碰到父元素边框换行

### 浮动：float

* 1. 让盒子（div）在一行显示
  2. 标签浮动之后脱离了文档流，**不占据原来的位置**
  3. **行内元素设置浮动后变为行内块元素（不推荐使用，因为浮动之后会脱离标准流，最好使用display：inline-block）**
  4. **如果不设置width，那么盒子自动收缩为文字的宽度**
  5. **浮动的作用**
     1. **文本环绕图片**
        1. 文字和浮动元素没有层叠关系
        2. 文字不参与浮动
     2. **制作导航栏**
     3. 网页布局

### 清除浮动：清楚浮动产生的不利影响，

* 1. 当父元素没有定义高度，嵌套的盒子浮动之后，下边的元素位置发生错误。
  2. **清除浮动的方法**
     + 1. **额外标签法；Clear：left/right/both**
          1. 工作中用的最多的是clear：both
          2. 在最后一个标签添加一个标签（用的不多，影响服务器响应速度）

<div class="main">

<div class="content"></div>

<div class="sidebar"></div>

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

</div>

* + - 1. **给浮动父元素使用overflow：hidden**
         1. **如果有内容出了盒子，不能使用该方法**
         2. BFC快元素格式化上下文
      2. **.clearfix：after 为元素清楚浮动**
      3. .clearfix：after｛
      4. Content：“”；
      5. Display：block；
      6. Height：0；
      7. Line-height：0；
      8. Visibility：hidden；
      9. overflow:hidden

Clear：both；

* + - 1. ｝
      2. .clearfix｛
      3. \*Zoom：1；/\*兼容IE浏览器\*/
      4. ｝

### CSS初始化：为了更好的用户体验

* 1. 为了考虑浏览器的兼容性，不同浏览器标签默认值不同，如果没有CSS对页面进行初始化，会出现在不同浏览器页面显示效果不同
  2. 火狐默认字体：微软雅黑
  3. Chrome默认字体：宋体

### Overflow：处理溢出的内容

* 1. Hidden：**溢出的内容将被修剪**，不显示
  2. Auto：溢出时出现滚动条，不溢出时正常显示，不显示滚动条
  3. Visible：默认值，不处理溢出内容，正常显示
  4. Scroll：滚动条，右边和下边不管溢出与否都会出现滚动条

### 定位：position

1. **定位的方向：**left、right、top、bottom
2. **静态定位：**static，默认值，按文档流默认位置定位设置
3. **绝对定位：**absolute
   1. 使用之后**不占据原来的位置**（**脱标**）
   2. **元素使用绝对定位，默认以浏览器为基准，除非它有个父元素是定位元素（relative，absolute，fixed）**
   3. 嵌套的盒子，父元素没有使用定位，子元素使用定位，子元素**定位从浏览器出发**
   4. 嵌套的盒子，父元素使用绝对定位，子元素使用绝对定位，子元素定位**从父元素开始**
      1. **给行内元素使用绝对定位（不推荐使用，推荐使用display：inline-block）后，转换为行内块元素，脱标，可以设置宽高**
      2. **Z-index：调节标签层叠顺序（0-999）值越大，元素越在上面**
4. **相对定位：relative**
   1. 相当于自己的初始坐标移动
   2. 仍然占据原来的位置，不脱标（标准流）
   3. **子绝父相（子元素绝对定位，父元素相对定位）**
   4. 对行内元素使用相对定位不能转换为行内块
5. **固定定位：fixed**
   1. **定位后不占据原来位置，脱标**
   2. 定位从浏览器出发
   3. 行内元素使用固定定位后，会转换为行内块元素（**不推荐**，**推荐使用display：inline-block**）

# CSS第五天 2016/08/05

1. **已定位的盒子（div）居中显示**
   1. Text-align是内容居中
      1. **Margin：0 auto；是盒子居于父元素中间**
         1. **不能对已定位和盒子使用（因为已定位的盒子脱标了）**
         2. **只能让标准流的盒子居中显示**
   2. **已定位的盒子定位居中方式**

**先向右走父盒子的一半50%,然后再向左移动盒子的一半（margin-left负值）会随着屏幕的大小改变而改变**

Position：absolute；（子绝父相）

Left：50%；

Margin-left：子元素盒子的一半（负值）

1. **使列表横向显示需注意的问题**
   1. 用display：block-inline；方法使列表li横向显示，每个li后面的回车会产生一个空格，
   2. 使用浮动float：left；的办法就不会产生此问题
2. **定位以最近的已定位的父元素为移动基准**
3. **标签包含规范**
   1. DIV可包含所有元素
   2. **P不能包含DIV、H1，P元素一般只能包含行内元素**
   3. H系列系列可以包含DIV、P，但是一般不这么用
   4. 行内元素不推荐包含行内元素，一般都这么用
   5. 行内元素不能包含块元素
   6. Li里面可以放置任何元素
4. **规避脱标流**
   1. 尽量使用标准流（文档流）
   2. 当标准流解决不了的时候，使用浮动
   3. 浮动解决不了的时候，使用定位
   4. **Margin-left：aoto；内容靠左显示**
   5. **Margin-right：auto；内容靠右显示,**可以将盒子冲到父盒子的右边
5. **图片和文字垂直居中**
   1. 图片的下边默认和文字的基线对齐
   2. 对行内块inline-block使用
   3. Vertical-align：垂直对齐
      1. Vertical：baseline，基线对齐
      2. **Vertical：middle，图片与文字垂直居中对齐(img中设置)**
      3. Vertical：sub/sup，对齐文字的下标/上标
6. **CSS可见性**
   1. Overflow：hidden：溢出隐藏

Tab栏切换

* + 1. 设置一个父盒子溢出隐藏
    2. 设置三个和父元素相同大小、颜色不同的子元素
    3. 锚点链接到每个子元素
    4. 点击链接就链接到不同颜色的子元素
  1. **Visibility：hidden**；隐藏标签
     1. 隐藏后仍旧**占据**原来的位置
  2. **Display：none；**隐藏标签
     1. 隐藏后**不占据原**来的位置
     2. 常配合JS使用
  3. Display：block；元素可见

1. **CSS里面内容移除（网页优化）**

点击搜狐图片就能回到主页

* 1. LOGO一般放在H1标题里面，这样做权重高，有益于SEO
  2. 方法一：用text-indent:-5000em；移除文字，**一定要设置inline-block**

<style type="text/css">

.logo{

width: 143px;

height: 76px;

background: url("logo.png");}

a{

**display: inline-block;**

text-indent:-5000em;}

</style>

<body>

<div class="logo">

<h1>

<a href="#">搜狐</a>

</h1>

</div>

* 1. 用DIV的padding移除文字
     1. 设置DIV高度为0
     2. 此时DIV中的内容在DIV外面
     3. 给div一个上内边距
     4. 设置DIV的overflow：hidden；

.box{

width: 300px;

height: 0;

**padding-top: 100px;**

overflow: hidden;

background: red;

}

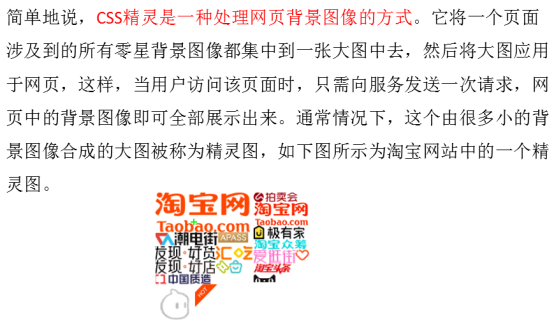
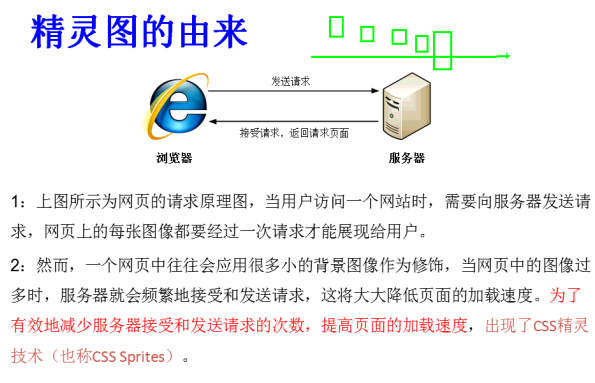
</style>

</head>

<body>

<div class="box">14期威武</div>//14期威武看不见

1. **精灵图**
2. **CSS精灵图技术（也成CSS Sprites）**把许多小图标放在同一张png图片上，要想精确定位到精灵图中的某个小图，就需要使用CSS的background-image、background-repeat和background-position属性进行背景定位，其中**最关键的是使用background-position属性精确地定位**。



* 1. 背景默认居左居顶（左上角）显示
  2. 背景图片往左移动，left：负值
  3. 背景图片往上移动，top：负值
  4. 精灵图的制作
     1. 尽量靠一边，方便取坐标值
     2. 尽量减少图片尺寸（内存）大小
     3. **一定要PNG（透明）格式的**

1. **属性选择器**
   1. 标签[属性=“属性值”] [属性=“属性值”]…｛属性：属性值；｝
2. **Emment快捷键**
   1. ！+tab 生成H5结构
   2. Ul > li.abc$\*4

<ul>

<li class="abc1"></li>

<li class="abc2"></li>

<li class="abc3"></li>

</ul>

C) width:30px==>w30+tab

Height:30px==>h30+tab

Margin:30px==>mg30+tab

Padding:30px==> pd30+tab

Line-height:12px==>lh12px+tab

Background==>bg+tab

### **D)生成带有内容的标签：**ul>li>a{item}\*5

**E）bd+： border: 1px solid #000;**

F).search-logo+.search-input+.search-car.+.searchmoreAlink加tab

# CSS第六天 2016/08/07

### 滑动门

1.简单的滑动门：背景颜色会随着文字的变化而变化

.nav{



height: 100px;

}

.left,.cen,.right{

float: left;

}

.cen{

height: 100px;

background: url("../../03-练习/m.png");

line-height: 100px;

}

<body>

<div class="nav">

<div class="left"><img src="../../03-练习/l.png" alt=""></div>

<div class="cen">传智播客官网传智播客官网传智播客官网</div>

<div class="right"><img src="../../03-练习/r.png" alt=""></div>

</div>

1. 复杂的滑动门



1. 背景默认从左上角开始显示
   1. 给背景位置设置right可使背景从右显示
   2. a span：hover｛属性：属性值｝，给A里面的span添加伪类

li{

list-style: none;

float:left;

}

li a{ height: 35px;

background: url("bg\_r1\_c1.png") no-repeat;

padding-left: 7px;

display: inline-block;

color: white;

}

li span{

height: 35px;

background: url("bg\_r1\_c2.png") no-repeat right;

display: inline-block;

padding-right:22px;

line-height: 35px;

}

**a span:hover{**

**background: url("blue1.png") right;**

**}**

**a:hover{**

**background: url("blue2.png");**

**}**

</style>

<ul>

<li><a href="#"><span>百度一下</span></a></li>

<li><a href="#"><span>百度</span></a></li>

<li><a href="#"><span>中国新闻网</span></a></li>

</ul>

1. **内容自动撑大盒子，盒子大小自适应宽高度**
   1. 是对象成为bfc（块级元素格式化上下文）对象，
      1. 创建bfc对象方法
         1. Overflow：hidden/scroll/auto（除visible之外的属性值）；
         2. Float：left/right（除none之外的值）；
         3. Display：inline-block、table-cells、table-cpation。
         4. Position：绝对定位absolute或者固定定位fixed。
2. Vertical：top/middle/bottom垂直对齐方式（对行内块最敏感，规避了脱标流）
3. Div里有图片，尽量定义宽高



1. **块元素的高不能继承**
2. 标签调用的样式越少越好

# CSS第七天 2016/08/08

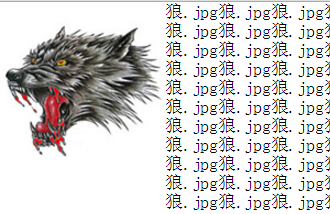
1. **梅兰商贸案例**
2. 不给盒子宽高会造成文字环绕
3. 图文排版
   1. 给img标签浮动，设置P的属性overflow：hidden
   2. 此时文字和图片顶端对齐
   3. 可用外边距调整位置



* 1. 不给P标签overflow：hidden：是文字环绕图片



* 1. 给P标签overflow：hidden；（设置浮动也是如此，把P标签创建为一个bfc对象）属性后是



1. **一个元素层叠在另一个元素上的时候用定位**
2. **大的布局定位的时候用浮动，小的元素用定位**
3. 内在盒子中容水平居中，text-align：center；
4. 行高之可以一行文字在水平居中

# 就业班第一天（京东网页）

## 搭载项目环境

1. HTML 核心文件 index.html
2. CSS：base基本样式 global 全局样式
3. Images：图片文件夹
4. JS 音频视频...
5. 站点=项目=项目文件夹=根目录

Webstrom 快捷键

ctrl+x 剪切一行或删除一行

**ctrl+d 复制本行到下一行**

ctrl+z 退回之前操作

ctrl+shift+z 已经做好的退回之后的下一步

ctrl+a 全选

**ctrl+/ 注释行**

ctrl+shift+/ 注释选定区域

ctrl+g 查找行号

**ctrl+shift+up/down 上下移动代码**

table 补齐代码

ctrl+f 查找文本

1. **www.网址/favicon.ico 下载网页快捷图标**
2. <link rel =”shortcut icon” href = “路径”> （shoutcut可加可不加）
3. Bitbug.net 比特虫图标在线制作
4. Vertical-align：middle； 去掉图片默认的间距

用fireworks采集颜色和图片宽高，快捷键如下：

**V 切换黑鼠标**

**U 画图**

**K 切片 框选后→切换到指针→Alt，shift移动**

**I 吸色**

**Z 放大镜**

**空格 移动**

1. 代码未写，注释先行
2. 谷歌不支持12号以下字体
3. 浮动的盒子不设置宽，容易掉盒子
4. Font字号，字体名字必须写
5. 行高可以继承

## 文字居中

* 1. Text-align:center：//不能精准定位
  2. 尽量使用padding-left/padding-right：可自适应大小
  3. 定位和margin

1. 驼峰命名：除第一个单词外其他的单词首字母大写
2. **先写普通标签 –> 特别标签 ✍ 更特别标签**
3. 尽量避免代码污染
4. Padding：不可设置负值
5. CSS下面的代码样式层叠上面的
6. 选择器尽量不要超过三个
   1. 多了会造成层叠属性困难
7. 浮动、固定定位和绝对定位都可以使行内元素转换为行内块元素
8. **能浮动不定位**

## 盒子的稳定性

* 1. 只给宽高的盒子（高度剩余法）
  2. 给padding的盒子（padding里面不能有内容，宽度不确定的时候使用，可填充背景颜色）
  3. 给margin的盒子（**容易出现塌陷，不能继承背景色**）

## **做logo**

* + - 1. 文字移除：text-indent：-5000em
         1. 给a链接七位数的em/八位数的px会出现（bug）链接点不到
      2. 加链接（a标签）
      3. Logo图片做a标签背景

.search-logo a{

width: 270px;

height: 60px;

display: inline-block;

background: url("../images/logo.png") no-repeat;

text-indent:-5000em;

margin: 20px 0;

}

## 模拟鼠标：

* 1. **cursor：pointer；小手**
  2. **cursor：text；变成输入框光标**
  3. **cursor：move；变成十字箭头**
  4. **cursor：default；默认箭头**

# 就业班第二天（京东网页）

## 知识点

1. 同一个父级元素的盒子要么都设置浮动，要么都不设置
2. 行高：line-height；也可以撑开盒子
3. **文字不设置行高，默认是包含文字的盒子的高度**

* **设置了定位不给left/top值会出现bug**

1. 行内元素有三像素的距离
   1. a标签之间有3像素的距离
   2. 图片之间也有3像素的距离
2. **给html文档相邻的同级盒子设置浮动，会顶对齐**
3. 文字不参与浮动，标准流中的文字不会被浮动的盒子遮挡
4. **一个父盒子中的子盒子要么都浮动，要么都不浮动**
5. 浮动的盒子相互受影响，不论是否在同一个父盒子中

## 矩形圆角

1. Border-radius：宽高的一半
2. Border-radius：百分百（50%）
3. Border-radius：小数（0.5）
4. Border-radius：左上 右上 右下 左下

## 清除浮动

1. 清除浮动的原因：
   1. **父盒子高度为0，子盒子全部浮动、定位，子盒子不会撑开父盒子，下面的元素会到子盒子的下面**
2. **清除浮动的方法**
   1. **Clear：both**
   2. **Overflow：hidden([BFC](file:///E:\\传智播客\\笔记\\Xmind思维导图\\BFC：块级格式化上下文.png))**
   3. **加空标签**
   4. **单伪元素/双伪元素**

//双伪元素

.clearfix:before , .clearfix:after {

Content:””;

Display:table;

}

.clearfix:after {

Clear:both;

}

.clearfix{

\*zoom:1;

}

## 什么时候用margin和padding（不考虑宽高）

1. 需要使用背景图的时候必须使用padding
2. 会出现边距合并或者margin塌陷的时候用padding
3. A标签要用margin，否则鼠标经过a标签附近空白部分会出现 a：hover的属性

4.行内元素上下只能设置padding，不能设置margin。（行内高16px）

5.看需求

## 隐藏盒子

1. Display：none；隐藏盒子，并且不占位置
2. **Overflow：hidden；隐藏超出部分**
3. Visibility：hidden；隐藏盒子，占位置
4. **Opacity：0；隐藏盒子，占位置，可设置 div 元素的不透明级别，比如opacity：0.5，里面的内容也会半透明;另外**属性会被子代继承，子盒子无法通过设置opacity更改子代的属性
5. Position-left/top：-999em；隐藏盒子，占位置

## 电脑组成

1. CPU：中央处理器，电脑的大脑，用来计算收发指令
   1. 英特尔i3/i5/i7；低端的性价比高
   2. amd
   3. 国产龙芯，最高2s版本，没有实现商业化，军队在使用
2. 服务器：特殊的电脑
3. 系统：麒麟系统（中标麒麟）
4. 硬盘：存储数据（永久性存储），读写机制慢
5. 显卡：图像显示，存储量、带宽、架构
6. 内存：存储数据（暂时性存储），CPU要执行的数据冲内存中取，内存中的数据冲硬盘中取。（堆和栈）
7. 驱动：**让硬件和系统的兼容性更好**

## 盒子撑开和撑破

1. **浮动的盒子千万不要让他超出父盒子**。
2. 超出父盒子的部分会影响下面盒子中的浮动的子盒子。

## 课堂反馈

1. 浮动遮挡和顶部对齐
   1. 文字不参与浮动
   2. 同一级别的两个盒子都设置浮动，会顶部对齐
   3. 浮动遮挡
2. 清除浮动
   1. 单伪元素

.clearfix:after{

Content:”.”;

Display:block;

Height:0;

Line-height:0;

Visibility: hidden;

Clear: both;

}

.clearfix{

Zoom:1；

}

* 1. 双伪元素

.clearfix:before{

Content：“.”；

Display:table;

｝

.clearfix:after{

Clear：both；

}

.clearfix{

\*zoom:1;

}

1. 盒子的撑开与撑破
   1. 子盒子设置的高度比父盒子设置的高度高，会撑破盒子
   2. 父盒子不设置高度，子盒子设置高度会撑开盒子

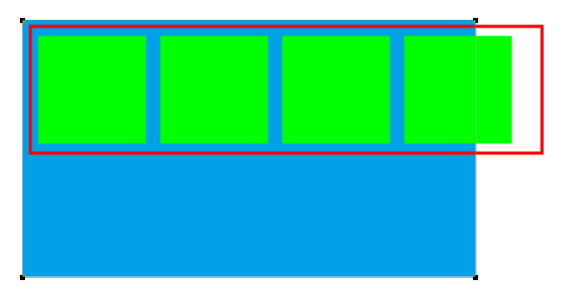
# 就业班第三天（京东网页）

## 知识点

1. 定位的盒子比浮动的盒子层级高
2. 浮动的盒子遮挡不住标准流的文字
3. 块元素的宽可以继承
4. A标签的文字属性不能继承
5. 谷歌不支持小于12号的字体
6. **Div：hover a｛｝；鼠标经过div区域后对a的操作**
7. 建议所有浮动的盒子都给宽高
8. **子盒子不占位置（设置了浮动、绝对/固定定位），父盒子没有高度，下面的会顶上来**
9. 几个盒子浮动，最好给宽高、有一定的间距
10. 要注重标签语义化
11. **关闭a 的链接：**
    1. **Href=“javascript：void(0)”；关闭a链接跳转，既不刷新也不回到顶部。或者写成Href=“javascript:;”**
    2. **A的href = “”：效果是刷新页面**
    3. **Href = “#”：跳到页面顶部，没有刷新**
12. Left的权重比right高，同时设置right的属性不起作用，top比bottom权重高
13. Rgba颜色模式
    1. r红g绿b蓝a透明度（0-1）（1完全不透明）
14. **dotted点线 dashed虚线**
15. **行高可以继承 父盒子给了行高，子盒子就不要用font设置行高**
16. Position：relative/static；设置之后也不能给行内元素设置宽高，只能是浮动、absolute
17. 高和行高都可以撑开盒子，背景图不可以撑开盒子

## 浮动宽度不够的问题

**父盒子宽度不够，又要子盒子浮动不掉下去**



如图，正常情况下，给绿色的盒子右浮动之后，由于父盒子的宽度不够，最后一个绿色盒子会被排列在第二行，

但是如果想要绿色的盒子在一行内显示

解决方法是，在给绿色加一个父盒子，也就是红色边框的盒子，只给红色边框盒子一个宽度属性(宽度大小4至5盒子宽)，这样，绿色的盒子就不会因为蓝色盒子的宽度不够而掉下去了；然后超出的部分可以给父盒子（蓝色盒子）设置overflow：hidden；隐藏超出的部分

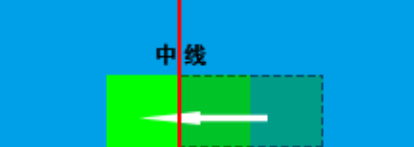
## 层级

* 1. 浮动/标准的盒子代码位置越往下，层级越高
  2. 层级一样，代码在后面的盒子会压住前面的盒子
  3. 高低和占不占位置无关，relative占位置，也比不占位置的浮动层级高。**定位比浮动高，浮动比标准高**
  4. **层级为0的盒子，也比标准量和浮动的高**
  5. **层级为负数的盒子，比标准量和浮动低**
  6. 层级不取小数
  7. Z-idex会导致有外边距的盒子压住另外的盒子



## 子盒子在父盒子居中

**让子盒子在父盒子水平居中，先绝对定位，left50%，然后margin – left：负子盒子的一半宽度；（垂直方向也适用）**



## 课堂反馈

1. **层级**
   1. **有外边距的盒子层级变高会压住其他相邻的盒子**
   2. **浮动的盒子尽量给宽高**

**2015版新增内容：**

**1.判断是否要加引号：只有变量不加引号**

**2. var num = "0";**

**num++;**

**console.log(num);//1**

**var num = "0";**

**num = num +1;**

**console.log(num)//01**

**3.innerHTML ：**

**更换盒子里面的内容，包括文字和标签都换.**

**Input.value ：表单更换内容**

**4.自动获得焦点&鼠标经过选择表单**

**window.onload = function(){**

**var txt = document.getElementById("txt");**

**var sele = document.getElementById("select");**

**txt.focus(); //自动获得焦点**

**sele.onmouseover = function(){**

**this.select(); //选择表单**

**}**

**}**

**自动获得焦点:**

**<input type="text" id="txt"/>**

**鼠标经过选择表单:**

**<input type="text" id="select"/>**

1. **隔行变色**

**.current{**

**background-color: #aaa!important;**

**//行内式的优先级高于class**

**}**

**window.onload = function(){**

**var lis = document.getElementsByTagName("li");**

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

**{**

**if(i % 2 == 0)**

**{**

**lis[i].style.backgroundColor = "#eee";**

**}**

**else**

**{**

**lis[i].style.backgroundColor = "#ddd";**

**}**

**lis[i].onmouseover = function(){**

**this.className = "current";**

**}**

**lis[i].onmouseout = function(){**

**this.className = "";**

**}**

**}**

1. **求平均值**

var arr = [10,20,30,34,67];

function avg(array){ // 封装求平均值函数

var len = array.length; // 数组的长度

var sum = 0;

for(var i=0; i<len; i++)

{

sum += array[i];

}

return sum / len;

}

console.log(avg(arr));

7.遍历京东背景图片

var iis = document.getElementById("iconsprite").getElementsByTagName("i");

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

{

iis[i].style.backgroundPosition=**"0 "+(-i\*25)+"px";**} 必须都是双引号

1. **+ 数值相加 字符相连**

“你好” + 2 //“你好2”

“0” + 10 //“010”

10 - “2” // 8

**所有的input取过来的值是字符型,想要数值相加，必须用Number和parseInt强制转换**

1. **全选和反选**



btns[0].onclick = function(){

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

{

inputs[i].checked = true; // 选中表单

}

}

btns[1].onclick = function(){

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

{

inputs[i].checked = false;

}

}

btns[2].onclick = function(){

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

{

if(inputs[i].checked == true)

{

inputs[i].checked = false;

}

else

{

inputs[i].checked = true;

}

}

或者用函数封装

function all(flag){

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

{

inputs[i].checked = flag;

}

}

btns[0].onclick = function(){

all(true);

};

btns[1].onclick = function(){

all(false);

};

btns[2].onclick = function(){

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

{

inputs[i].checked == true ? inputs[i].checked = false : inputs[i].checked = true;

}

}

}//**btns[0].onclick = all(true)**;还没点击就已经执行完成

1. **变量与属性**

var index = 10; //变量，谁都可以使用

var arr = []; // 数组

arr.index = 20; // 自定义属性，他只能在arr 才能使用

alert(arr.index);

1. **多个tab栏封装**

# 就业班第四天（JS）

## 知识点

1. 作用
   1. 页面特效（PC端网页效果）
   2. 移动端（移动web和app）
   3. 异步和服务器交互（Ajax）
   4. 服务端开发（nodejs）
2. **JS是客户端脚本语言--不需要编译，直接运行时边解析边执行的语言**
   1. 编译执行：把代码编译成CPU认识的语言，然后整体的执行
   2. 解释执行：一行一行解析，解析一行执行一行
3. 弱类型语言，用一个变量声明可以声明很多类型的变量
   1. Var a ；（a可以是很多类型）
   2. 强类型语言：一个声明只能声明一种类型
4. 设计原则：结构、样式、行为-----分离

## JS组成

1. ECMAScript：标准，语法标准
2. Dom：操作网页上元素的API
3. Bom：操作浏览器部分功能的API

## 引用方式

1. 内嵌式：把JS代码写在body的script标签内
   1. 写在html下在执行的时候也会被提到body下
   2. 还可以写在head内
   3. 最好放在body的最下面
2. 外链式(**必须连在一起**)
   1. <script src = “路径.js” type = “text/javascript”></script>
3. 行内式，和事件关联

<button onclick="alert('你好吗')">点击我</button>

我们一般采取的是 外双内单的格式。

<a href=”javascript:;”></a>

<a href=”javascript:void(0);”></a>

### 输出语句

1. Console.log （“控制台输出”）；
2. Console.error（“控制台输出错误内容”）；
3. Console.warn（“输出警告”）；
4. Document.write（“输出到页面”）；
   1. 可输出标签，用引号引起来
5. Prompt（“输入框：”）；
   1. 输入内容被返回
   2. 不输入内容返回null
6. Confirm（“输出带有是否的弹出框”）
   1. 点击确定返回true
   2. 点击取消返回false
7. **注释/\*\* +tab**

## 变量

### 命名规则

1. 驼峰：
   1. 全大写，首字母小写：aRRbCC
   2. 小写：ArrBcc
2. **$和\_ 只有这两个特殊符号可以用作名字**
3. 最长255字符，变量名区分大小写
4. 不能以数值开头
5. 不能用关键字，保留字
6. 可以用alert（）；弹出一下，有问题的都别用

### 赋值

1. 可以一次定义多个变量
2. 定义的同时赋值
3. 不建议跨类型赋值：定义的时候是什么类型就赋值什么类型 Var a = “张三”；

## 数据类型

1. **简单类型（值类型）**：
   1. Number string boolean undefined null
   2. 判断简单数据类型typeof（）；
   3. Number：2进制，8进制（0），16进制（0x）
2. **复杂数据类型：**
   1. **object，function，array，date，regexp，error**
3. **字面量**；字面上看到的值
4. JS小数计算容易出现精度丢失
5. 不要比较浮点数
6. **Infinity：无穷大**
7. **NaN：Not a Number，不是一个数字**
   1. **NaN和任何数字不相等，包括自身**
   2. IsNaN（）;符不符合不是一个数值的标准，判断一个值是不是数字
      1. 是数字返回true
      2. 不是数值返回false
8. ***Undefined和任何数值计算结果都是NaN* console.log(undefined+10)//NaN**
9. **任何值和null都可以看作0来运算 比如console.log(null+10)//10**
10. **字符串不可修改**，给字符串重复赋值的时候，会在栈中重新开辟一个控件，原来的空间会被回收。
11. **Boolean：有两个字面量：true和false，并且区分大小写！（大写不对）**
    1. True 非0数值 字符串 object
    2. **False 0 undefined null 空字符串**
    3. If判断的时候会把（）内的值强制转换为boolean类型进行判断

## 数据类型转换

1. **任何类型转换为string（对于字符串来说，+是连接符）**
   1. 变量+“abc”
   2. **S**tring（变量）；
   3. 变量.toString ( ); **（undefined和null不可以）**
2. **任何类型转换为number**
   1. **减乘除一个数字字符串可转换为number**
   2. Number（变量）；不会取整数，变量里有非数字部分就会返回NaN
   3. parseInt（变量，进制）：从头取整，不取小数部分和非数字部分，截取变量的整数部分

**console.log(parseInt("25px"));//25**

**console.log(parseInt(10,2))//2 二进制10转换成十进制**

var num3 = parseInt(""); //空字符串返回NaN，Number("")返回0

* 1. parseFloat（变量）：从头取数字部分，包括小数部分
  2. null值为0
  3. undefined无数值类型

1. **转换为boolean**
   1. Boolean（变量）
   2. ！！变量

var date=new Date();

console.log(!!date)//true

## 操作符

1. 算数运算符
   1. 一元运算符 ++ --
   2. 二元运算符 + - \* / 需要两个数值才可以运算
   3. 三元运算符 判断？数值1：数值2；
2. 逻辑运算符|| && ！
3. 比较运算符 > < >= <= === !== == !=
4. 赋值运算符+= \*= /= -=
5. 运算符优先级
   1. 括号最高
   2. 赋值运算最低

var num = 1;

**console.log(num++); 先运算再自增**

## &&和||运算

**1.|| / && 找到true / false就返回，找不到就返回第二个**

### &&链接两个boolean类型，有一个是false结果就是false。

链接值不是布尔类型时，按照成布尔类型计算，结果本身不变。（非布尔）

例子： 1 = 2&&1； 0 = 0 && 1； 都是true取后面，都是false取前面。

### ||链接两个boolean类型，有一个是true结果就是true。

链接值不是布尔类型时，按照成布尔类型计算，结果本身不变。（非布尔）

例子： 2= 2||1； 1 = 0 || 1； 都是true取前面，都是false取后面。

**Var a=3&&0||2** //2先算&&

## 选择结构

1. **If语句用法有三种**
2. **if(条件1){程序1}**
3. **if(条件1){程序1}else{程序2}**
4. **if(条件1){程序1}else if(条件2){程序2}...else{程序n}**

二.Switch（）

* 1. ｛case ： ；break；… default：；｝
  2. 括号内实用===比较，类型和数值都得相等
  3. 要加break；

## 循环结构

1. 循环用途：重复执行某段代码
   1. 遍历数组的时候首选for循环,简单的循环用for
   2. While循环强调，不记循环次数，不知道循环次数
   3. **无论怎样都循环一次**选用do…while
2. For（var 变量**；**判断条件；变量变化）｛循环体｝
   1. For循环内部定义的i是全局变量

B)For括号内的两个分号不可省略三个表达式均为可选，但是必须写分号！！！  **for(;;){程序} 死循环;**

1. While
   1. While（判断条件）｛循环体；｝
2. Do…wihle
   1. Do｛…循环体…｝while（判断条件；）

# 就业班第五天（JS）

## 知识点

1. NaN是number类型
2. null是object类型

## 循环结构

### Break和continue

* 1. Break：立即退出当前循环
  2. **Continue：退出本层循环**
  3. 两者之后的代码都不会执行

## 数组

### 知识点

1. 数组的数据类型是object
2. 未赋值的项是undefined

### 定义方式

var arr = [1,3,5,7,9];

var arr = new Array(1,3,5);

1. 字面量定义：var array = [ 1 ,2 ,3];
2. **对象定义： var array = new Array（）；//空数组**
   1. **Var array = new Array （数据1，数据2，…）**
   2. **Var array = new Array（3）； 设置数组长度为3**

### 数组遍历

**遍历就是获取组的每一项**

for （var i =0; i < arr.length; i++）{

console.log( arr[ i ] );

}

### 冒泡排序

1. 双重for循环
2. 指定比较次数和轮数
   1. 比较轮数，数组长度-1
   2. 比较次数，数组长度-1
3. 判断是否符合标准
4. 交换比较的值

## 作业

1. 1乘到100 product 乘积
2. 1-100所有不能被7整除的数字之和
3. 计算出1-100之间所有不能被3整除的数字之和大于（或等于）2000的那个数字
4. 1-100所有不能被7整除的数字之和（用continue）
5. 1-100之间所有不能被3整除的数的和（continue）
6. 计算出1-100之间所有不能被3整除的数字第一个大于2000的和
7. 200-300所有的奇数（odd）和（偶数：even）
8. 200-300之间第一个能被7整除的数
9. While和do…while求和（1-100）
10. 死循环求和
11. 1-100所有奇数反到数组中
12. 1-100之间能被3整除的数值放到数组中
13. 求一个数组中的和和平均值
14. 求数组中最大值和最小值和其位置
15. 将字符串数组用|和其他符号分隔
16. 将数组中值为0的去掉，剩下的存到新数组
17. 反转数组
18. 冒泡排序，从小到大

# 就业班第六天（JS）

## 知识点

1. 函数：就是可以重复执行的代码块
2. **组成：参数，功能，返回值**
3. 为什么要用函数，因为一部分代码使用次数会很多，所以封装起来，需要的时候调用
4. **函数不调用，不执行**
5. 同名函数会覆盖，后面的覆盖前面的(**函数名不能一样)**
6. **加载函数的时候只加载函数名，不加载函数体**
7. **形式上参与运算的变量，无实际值，为实参占位置，就像一个躯壳一样。（可以理解为函数的内部变量外部无法访问）**
8. 参数相当于局部变量
9. **预解析：函数在解释文档的时候会被整体提到文档的最前面，和加载不一样**

10.var temp = 函数名() {（功能）+ 该函数的返回值;}**函数程序运行后的结果外部需要使用的时候，需要通过return返回。函数的返回值必须要执行函数才能得到。**函数执行后剩下结果就是返回值。

## 函数的定义

1. **第一种：解析的时候会被提前，可在任何地方使用和定义**
2. function fun ( a , b ) { 函数体；return ; }
3. **第二种：函数不会被提前，使用必须在定义之后**
4. var fn2 = function fn( )｛函数体；｝
5. //函数名可有可无，无函数名叫匿名函数
6. **第三种：函数不会被提前，使用必须在定义之后，如果函数体不加引号，不调用也会执行**

**3.var fn3 = new Function（“函数体；”）；**

## 函数的参数

1. 设置参数可解决修改值不修改规则这个问题。
2. JS没有方法重载
3. 规则中有几个变化的数（操作数），就定义几个参数
4. 函数可嵌套
5. Math.sqrt( ); 开平方

### 形参

1. 形式上参与运算的值，作用是为实参占位置

### 实参

1. 实际上参与运算的值
2. 实参要进行运算，必须要有形参占位置

### 形参的个数和实参的个数不一样

1. 相等，正常执行
2. 实参个数大于形参，正常执行，多去的舍弃不用
3. 形参大于实参，看程序是否出错
   1. 计算会得到NaN
   2. 实参值是undefined 为给定实参的形参是undefined
   3. 任何值和undefined计算都的NaN

## 返回值

1. 函数程序运行后的结果外边需要使用的时候，我们不能直接给予，需要通过return返回
2. 作用：函数执行完成以后，结果就是返回值
3. 把函数内部的值赋值给外部
4. **如果没有return或者只有return没有值**，**函数默认返回undefined**
5. 函数执行完return后会立即结束，**return后面的代码不会执行**
6. 函数的返回值必须要执行才可以得到
7. return可以截断循环
8. **例题 function $(id){**

**return document.getElementById(id);}**

1. **Math.round（）四舍五入**
2. **Math.pow( a , n); a的n次幂**
3. **Math.floor();地板函数**
4. **var s = Math.floor( Math.randow() \* (max-min+1) + min ); //生成min-max之间的随机数**
5. **函数内用的必须是形参的数组**
6. **bubble 冒泡**
7. 冒泡排序极限后会出现问题
8. var arr = [4,3,2,1];  
   //var arr = [3,1,5,7];  
   function sortBubble1 (array){  
   // var array = [];  
   // for (var i = 0; i<array1.length; i++){  
   // array[i] = array1[i]; //把原数组赋值给新数组  
   // }  
   for (var i = 0; i < array.**length**-1; i++) {  
    var bool = true;  
    for (var j = 0;j <array.**length**-1; j++) {  
    if (array[j] > array[j+1]){  
    var temp = array[j];  
    array[j] = array[j+1];  
    array[j+1] = temp;  
    bool = false;  
    }  
    }if (bool){  
    return array; //这样不能返回正确的值，会返回undefined  
   // break;//这样才可以  
    }  
    }  
   // return array; //这样才可以返回正确值

## 函数名、函数体和函数加载问题

1. JS加载的时候，只加载函数名，不加载函数体,如果想使用内部成员变量，需要调用函数

2. //执行函数，就等于函数的功能+返回值；

//**打印函数名，就等于打印整个函数**。

console.log(fn);

//**打印执行函数，就等于打印函数的返回值**。

console.log(fn()); //函数中包函数，先执行里面，后执行外面

## 变量作用域

### 局部变量

1. 只有局部能够访问的变量
2. 函数内部用var定义的变量
3. 函数执行完毕后，局部变量被回收

### 全局变量（成员变量）

1. 在哪里都可以访问的变量
2. **在函数内部，直接定义变量，不加var为全局变量**，函数必须得执行后才可以使用

## 隐式全局变量

1. 一般存在函数之中
   1. **var a = b = c = 1;** //这种情况，b和c都是隐式全局变量
   2. **var a = 1; b = 2; c = 1;** //这种情况，b和c都是隐式全集变量，**分号相当于换行**
   3. **var a = 1, b = 2, c = 1;** //这种情况，**b和c不是隐式全局变量**

## 预解析：JS解析器

1. 查看语法错误
2. 变量声明提升（**只提升变量名，不提升变量值**），
3. 把内容加载到内存中
   1. 对象加载到堆中
   2. 简单类型加载到栈中
4. **用function定义的函数是整体提升**
   1. 变量提升在函数内照样适用
   2. 就近原则，
5. **预解析（整体提升函数）和函数加载（只加载函数名）的区别**
6. **预解析:检查script标签内的语法，提升函数声明**
7. **函数加载:把函数加载到内存中**
8. **解析完再加载**
9. **两个平级的函数中的变量不会相互影响（可以使用同样的形参名）**

例题1.var num = 10;

fun();

function fun(){

//变量声明提升 值提升变量名，不提升变量值。var num;

console.log(num);//undefined

var num = 20;}

例题2.console.log(cc);

console.log(bb);

console.log(aa);

f2();

function f2(){

var aa = bb = cc = 9;

console.log(aa);

console.log(bb);

console.log(cc);

}

//cc is not defined f2函数只有调用了，cc才有声明

## 作业

1. 圆的周长面积
2. 2个数中的最大值
3. 3个数中的最大值
4. 一组数中的最大值
5. 一组数中的最小值
6. 反转数组（返回一个新数组）
7. 对数组排序，从小到大
8. 求阶乘
9. 求1!+2!+3!+4!+...+n!
10. 判断素数（prime）

# 就业班第七天（JS）

## 知识点

1. null是object（对象）类型
2. 函数也是object（对象）类型
3. 数组也是object（对象）类型
4. parseInt （数值，进制数）；把任意进制转换层十进制
5. num（要转换的数值）.toString（进制数）；十进制的数转换为其他进制数

## 函数高级

### 匿名函数

1. 没有名字的函数
2. 函数整体加小括号不报错
3. 函数调用
   1. **直接调用**
4. （function （）｛函数体；｝）（）；
   1. 事件绑定document.onclick = function（）｛函数体；｝点击后执行
   2. 定时器调用

setInterval(function () {

console.log(444);

},1000)

### 递归：其实质就是函数自己调用自己

**递归注意点：递归必须有跳出条件，否则是死循环。**

function fn(){

alert("从前有座山，山里有座庙...");

i++;

alert(i);

//加条件 if(i<3){

fn();

}

}

1. alert（sum（100））；
2. function sum（n）｛
3. if （n < 1）return 0;
4. return n + sum ( n-1 );
5. ｝//retrurn 100 + sum (99)+…

### **回调函数-- 函数作为参数进行传递**

**什么时候用回调函数？**

回调函数一般是定义一些规则来使用的，规则的传递只能通过函数实现，通过变量无法实现。

1. 函数形参后加小括号
   1. 函数作为参数进行传递和使用
   2. **（function（）｛｝）（）等价于fn（）**
2. 执行函数就等于函数名+（）；也等价于整个函数+（）；

例题 fn(test);

//回调函数:函数作为参数进行传递和使用。

function fn(demo){

demo();

}

function test(){

console.log("我是被测试的函数！");

}

1. function fn （num1,num2,demo）{
2. return demo(num1,num2);
3. }
4. function text1 (a,b){return a+b;}
5. Fn(a,b,text1);//回调函数，传递规则时使用

## 对象

1. **封装信息**
2. 对象具有特征和行为
3. **生活中的对象：具有唯一性的事和物**
   * 1. 一类事物：保时捷汽车
     2. **对象：具有唯一性**，我的那一辆保时捷汽车
4. 程序中 对象
   1. var hero = new Object();
      1. hero:就是一个对象
      2. Object：是一个事物，可以new很多的对象
5. 面向对象：c/c++，java
   1. 可以创建自定义对象，很好的支持继承和多态
   2. **特征：封装、继承、多态**
6. **基于对象：javascript**
   1. **无法创建自定义的类型（构造函数并不能创建新类型）**
   2. 不能很好的支持继承和多态，
   3. JS基于对象，只能new出Object对象
7. **属性和方法：状态（名词）和行为（动词）**
   1. 属性就是人物信息，行为就是施放技能
   2. 对象中的函数叫方法

对象名 . 状态 = 匿名函数（）｛函数体；｝：

* 1. 对象中的变量叫属性

对象名 . 属性 = 属性值；

* 1. 调用对象的方法和属性
     1. **调用自己的属性用this . 属性名；**
     2. **调用属性**

对象名.属性名；

* + 1. **调用方法**

**对象名.方法名（）**；

1. 对象缺点，
   1. 创建单个对象的时候可以，创建多个对象变得非常繁琐，
   2. 创建多个对象，for循环不能修改名字的值，**用函数可以创建多个函数**，缺点代码冗余，方式low。
   3. **用构造函数自定义对象。**

**如何创建单个对象**

例子：var student = new Object();

student.name = "张三";

student.sayHi = function () {

console.log(this.name+"说：大家好！");

}

console.log(student.name);

student.sayHi();

**如何创建多个对象**

var stu1 = createSudent("张三");

var stu2 = createSudent("李四");

console.log(stu1);

**stu1.sayHi();**

console.log(stu2);

**stu2.sayHi();**

//创建一个函数

function createSudent(name){

var student = new Object();

student.name = name;

student.sayHi = function () {

console.log(this.name+"说：大家好！");

}

return student;}

**定义一个构造函数，创建自定义对象**

var stu1 = new Student("王五");

var stu2 = new Student("赵六");

console.log(stu1);

stu1.sayHi();

console.log(stu2);

stu2.sayHi();

//创建一个构造函数

function Student(name){

//构造函数中的对象指的是this。

this.name = name;

this.sayHi = function () {

console.log(this.name+"说：大家好！");

}

}

1. this
   1. **this值出现在函数中**
   2. **谁调用函数，this就指向谁**
   3. new People（）；**People中的this代指的是被创建的对象实例**
2. new
   1. 开辟内存空间，存储新创建的对象
   2. 把this设置为当前对象
   3. 执行内部代码设置对象的属性和方法
   4. **创建并返回新的对象(不需要return)**

**例子var aaa = new stu();**

**console.log(aaa);**

**function stu(){**

**} //stu{}**

**var aaa = stu();**

**console.log(aaa);**

**function stu(){**

**}//undefined**

## 构造函数

1. **new后面调用函数，我们称为构造函数。其本质就是一个函数，只不过构造函数的目的是为了创建新对象，为新对象进行初始化(设置对象的属性)**
2. **首字母要大写new Object();**
3. **构造函数内部不需写对象名字，可用this代替**

var stu = new **S**tudent("张三");  
function Student(name){  
 this.**name** = name;  
 this.sayHi = function(){  
 **console**.log (this.**name**+ "说，大家好!");  
 }}

## 属性绑定

1. **对象名 . 属性 = 属性值**
2. **对象名[变量名] = 对象值；**

## 进制转换

1. 任意进制转换为十进制；**一个参数取整，两个参数，进制转换**
   1. var num = parseInt（其他进制数，数值是几进制的）；

var shiNum = parseInt(111,2);

console.log(shiNum);//7

1. 将十进制转换为其他进制;**无参转换成字符串。带参进制转换。**
   1. var num = num.toString（几进制）；

var num = 10;

var renyi = num.toString(16);

console.log(renyi);//a

# 就业班第八天（JS）

## 知识点

1. 构造函数和new一同使用，**构造函数首字母大写**，构造函数要的是构造实例

## 对象和面向对象

### 对象字面量

1. 对象的字面量就是一个{};而里面的属性和方法是以：形式对应表现的。（键值对）
   1. var obj = ｛key：value，（键值对）｝；
   2. key：键 value：值
2. **对象，数组也可以作为键值**
   1. key可加引号，可不加
   2. 键值对之间用都好隔开，不是分号

例 var obj2 = {**aaa:obj1**,name:"张三","age":18,sayHi: function () {

console.log(1);

}};

### JSON

1. **JavaScript Object Notation**对象字面量的一种表现形式
2. **json和对象（对象字面量）的区别仅仅在于，json的key键值对中的键必须带有“”；**

**例var json = {"name":"拴住","age":18,"arr":[1,2,3]};**

**3.对象本身没有length,所以不能用for循环遍历**

console.log(json.length);//undefined

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

console.log(json[i]);

}

4.为什么使用JSON？

对于 AJAX 应用程序来说，JSON 比 XML 更快更易使用

5.JSON的值可以是

1. 数字（整数或浮点数）
2. 字符串（在双引号中）
3. 逻辑值（true 或 false）
4. 数组（在方括号中）
5. 对象（在花括号中）
6. null

### for…in..遍历对象

1. 由于对象本身没有length属性，所以不能用for循环遍历

var json = { //创建一个对象  
 **"name"** : "张三",  
 **"sex"** : "男",  
 "sayHi" : function (){  
 **console**.log(json.**name** + "，你好！");  
 }  
}  
for (var key in json){  
 **console.log(key);**//打印键名 name sex sayhi

**console.log(json[key]);**//打印键值 张三 男 function(){}

console.log(json.key)//undefined}

### 伪代码：类

1. 强类型语言是用class定义对象
2. JS的ES6提到了class

class Customer {

public String Name;

public String Age;

public String Money;

public void Buy (String id,int num,double price) {

self.money -= num \* price;

}

}

var aaa = new Customer("name",18....);

## 传值和传址

1. **简单类型传值，复杂类型传址**
2. 简单数据类型保存在栈中
   1. 变量保存的是数据本身
   2. 函数调用传递的是值不是地址
   3. 压栈，先进后出
   4. 地址值：哈希值
3. 复杂数据类型保存在堆中
   1. 变量保存的是数据的地址

var num = 1;  
fun(num);  
**console**.log('+++' + num);  
function fun(a) {  
 //num传进来的是 1 这个数值，而不是num的地址，  
 //函数内修改的内容不会影响到函数外的变量  
 a = a \* 10;  
 **console**.log(a);  
}//10 +++1  
var arr = [1, 2, 3];  
fun1(arr);  
**console**.log(arr);  
function fun1(array) {  
 //arr和array指向的是用一个内存地址， // 函数修改array地址内的内容，arr的内容同时也被修改  
 array[0] = 2;  
 **console**.log(array);  
}// [2,2,3] [2,2,3]

例题一 var num = 1;

fn1(num);

console.log(num);

function fn1(aaa){

aaa = aaa\*10;

}//1

var obj = new Object();

obj.name = "张三";

fn2(obj);

console.log(obj);

function fn2(object){

object.name = "李四";

}// object.name = "李四";

例题二.

var a = 1;

b = a;

b = 0;

alert(a);

var obj1 = new Object();

obj1.name = "111";

obj2 = obj1;

obj2.name = "222";

alert(obj1.name);

## 内置对象

1. JS中已经定义好的对象。创建之后可直接使用
2. Date、Array、Math、Regexp（正则表达式）、Error、String、Number
3. 学习内置对象，主要学习API，H5的方法
   1. **调用者**：有无调用者
   2. **参数**：有无，几个
   3. **功能**：干什么用的
   4. **返回值**：有无，什么类型
4. 学习方法：手册，MDN
5. API：应用程序接口**(方法)**

### 数组高级API

1. **检测数组**
   1. instanceof（）；检测是否是数组对象

**console**.log(arr instanceof Array);

* 1. Arrey.isArray（），H5新方法

1. **转换数组**
   1. arr.toString（）：把数组变换成字符串
      * 1. var str = arr.toString();
        2. 需要一个新变量接受返回值
        3. 不改变原数组
        4. 数组中的逗号也会被转换成新元素的值
   2. **valueOf（）：返回数组本身**
2. **join是把数组元素用特殊方式链接成字符串(参数决定用什么链接，无参默认用逗号链接)**
   1. arr = arr.join（）；
   2. 参数决用什么连接
   3. 无参数默认用逗号链接
   4. 用空字符串可以**无缝链接**每个元素

E)空字符串有空格就是用空格隔开

var arr = ["关羽","张飞","刘备"];

var str1 = arr.join();//"关羽,张飞,刘备"

var str2 = arr.join(" ");//如果用空格的话，那么元素之间会有一个空格"关羽 张飞 刘备"

var str3 = arr.join("");//用空字符串，链接元素，无缝连接

"关羽张飞刘备"

var str4 = arr.join("&");"关羽&张飞&刘备"

1. **arguments、伪数组**
   1. **只在函数中使用，实参的数组**
   2. 伪数组，不能修改数组长度
   3. **允许修改值**，长短不可修改

例题 fn(1,2);

fn(1,2,3);

fn(1,2,3,4,5);

function fn(a,b){ console.log(arguments);}

//[1,2] [1,2,3] [1,2,3,4,5]  
**console**.log(fn.**length**);//形参个数  
**console**.log(arguments.**length**);//实参个数  
**console**.log(arguments.**callee**);//打印整个函数

1. **栈操作（先进后出）**
   1. arr = arr.**push（）；**
      * 1. 在最后面插入，
        2. 返回数组的新长度，
        3. **可以插入数组**，
        4. 可以一次加入多个元素
   2. arr = arr.**pop（）；**
      * 1. 去除最后一项
        2. 返回减去的这个元素
2. **队列操作（先进先出）**
   1. arr = arr.**shift（）；**
      * 1. 删除第一个数组元素
        2. 返回被删除的元素
   2. arr = arr.**unshift（）；**
      * 1. 最前面插入元素
        2. 返回新数组的长度
3. 排序、反转方法
   1. arr = arr.reverse（）：反转数组
      * 1. 返回新数组
   2. arr = arr.sort（）；从小到大排序
      * 1. **按照第一个字符排列，按照首字母的unicode值排列**
        2. 返回新数组
        3. 回调函数可以实现按你设置的规则排列，
        4. var arr = [1,21,15,2,3,4,5];  
           **console.log(arr.sort(function(a,b){  
            return a-b;** //设置规则升序  
           }));
        5. 打印unicode编码.charCodeAt（）；A--65
4. 数组拼接concat
   1. arr3 = arr1.aoncat（arr2）；把arr2链接到arr1后面，得到一个新的数组arr3
5. 数组的截取slice
   1. arr4 = arr3.slice（开始截取的位置，结束的位置）**包左不包右，不包括结束的索引**
   2. 只写一个值，从开始截取的位置，截取到最后
   3. 给予负值是从尾部开始截取，截取数值的绝对值个元素

var arr1 = ["a","b","c"];

var arr2 = [1,2,3];

var arr3 = arr1.concat(arr2);

//slice数组的截取

var arr4 = arr3.slice(2)//["c",1,2,3]

var arr4 = arr3.slice(-2)//[2,3]

var arr4 = arr3.slice(4,2);//[]

var arr4 = arr3.slice(2,4);//["c", 1]索引值包括坐标的不包括右边的。

1. **操作和截取原数组splice**
   1. arr4 = arr3.splice（**开始截取位置，截取几个，弥补的值）**；
   2. 弥补的值可以放一个数组，可以放元素
   3. 如果是数组，会把数组整体放在原数组中，形成一个二维数组
   4. 如果只截取，则原数组不变，放入修改的值后，直接修改原数组，返回的是截取后的新数组

var arr1 = ["a","b","c",1,2,3];

var arr4 = arr1.splice(0,3,"关羽","关羽","关羽");

console.log(arr1);//["关羽", "关羽", "关羽", 1, 2, 3]**原数组发生改变**

console.log(arr4);//["a","b","c"]**返回截取后的值**

1. 给元素查看索引indexOf（）
   1. arr.indexOf（“要查看的元素”）；后前往后
   2. arr.lastIndexOf（“要查看的元素”）；从后往前，
   3. 返回索引位置，找到立即返回，一次查找一个，从前往后
   4. 找不到返回-1
2. 数组遍历方法

**arr.every（function(item，index，arr)｛｝）；**

* + - 1. **返回一个boolean值，参数是回调函数**
      2. 对数组每一项进行如下函数，如果都返回true，every返回true，

3.如果有一项返回false则every返回false

4.返回false后函数停止运行

5.callback 被调用时传入三个参数：**元素值，元素的索引，原数组**。

例题 function isBigEnough(element, index, array) {

return (element >= 10);

}

var passed = [12, 15, 28, 130, 44].every(isBigEnough);

console.log(passed)//true

arr.some（function（element，index，array））对数组进行函数运算，有一项返回true则返回true,返回true函数停止运行

**arr.filter（function（element，index，array）**｛函数体；｝） filter返回值是一个新数组。**return为true的数组**

例题function isBigEnough(value) {

return value >= 10;}

var filtered = [12, 5, 8, 130, 44].filter(isBigEnough);

// filtered is [12, 130, 44]

**arr.forEach（function（element，index，array）**｛函数体；｝）遍历数组（**无返回值，纯操作数组中的元素**）

例题 let a = ['a', 'b', 'c'];

a.forEach(function(element) {

console.log(element);});//a b c

arr.map（function（element，index，array）｛函数体｝）对数组中的元素进行函数运算，把每个值返回的值组成一个新的数组

var numbers = [1, 4, 9];

var doubles = numbers.map(function(num) {

return num \* 2;

});// doubles is now [2, 8, 18]

// numbers is still [1, 4, 9]

### 清空数组

1. **方法一：arr.splice（0）**；
2. 方法二：arr.length = 0；（伪数组不可用）
3. 方法三：arr = []; //重新给数组赋值为空

相当于arr= new Array();

### 练习

1.将一个字符串数组输出为|分割的形式，比如“刘备|张飞|关羽”。使用两种方式实现

2.将一个字符串数组的元素的顺序进行反转。["a","b","c","d"] ["d","c","b","a"]。使用两种种方式实现。提示：第i个和第length-i-1个进行交换

3.找到数组中每个字母出现的次数["c","a","z","a","a"]

4.工资的数组[1500,1200,2000,2100,1800],把工资超过2000的删除

# 就业班第九天（DOM）

## 知识点

1. DOM：文档对象模型
2. BOM：浏览器对象模型
3. JS是以事件驱动为核心的一门语言
4. cursor：pointer；鼠标变成小手
5. document.getElementById()
   1. 找不到获取的元素返回null
   2. 找到返回该元素
6. **.innerHTML：双闭合标签里面的内容（识别标签）**
7. **.innerText：双闭合标签里面的内容（不识别标签）**
8. 老版本火狐不支持innerText，支持textContent

## 事件

1. 事件三要素
   1. 事件源
      * + 执行者
        + 谁引发的事件，引发后续事件的标签、动作
   2. 事件
      * + 执行的事情
        + 已经定义好的
   3. 事件驱动程序（处理元素）
      * + 执行的结果
        + 对样式和HTML标签的操作
2. 执行事件步骤

事件源(引发后续事件的标签)、事件（js已经定义好）、事件驱动程序（对样式和HTML的操作）（DOM）。

三句话：获取事件源、绑定事件、书写事件驱动程序。

**let content = document.getElementById('content');**

// 假设有 id="content" 的元素

**let ps = content.getElementsByTagName('p');**

//content 内的所有 <p>

* 1. 绑定事件
     + 1. 事件源.事件 = function（）｛事件处理函数｝
       2. **函数不要有返回值，调用的时候不要加括号**

事件源.事件 = fn；**不能写返回值***（用函数名绑定）*

function fn（）｛

事件；｝

* + - 1. 行内绑定
         1. 外部定义好，行内直接调用
  1. 书写事件驱动程序
     + 1. 操作标签的样式和样式
       2. **事件源.style.样式属性 = ‘属性值’**

属性值要用引号括起来



1. windows.onload：
   1. **window**.**onload** = **function** () {  
        **var aaa = document.getElementById("box");**//**页面加载（文本和图片）完毕的时候触动这个事件；整个页面上所有元素加载完毕再执行js内容（此时js文件不必放在body后面）；window.onload可以预防使用标签在定义标签之前。（**<**div id="box"**></**div**>放在script之后情况**）** aaa.**style**.**width** = **"100px"**;  
      }

## DOM概述

### DOM

#### 什么是DOM和节点

1. DOM由节点组成，**目的其实就是为了能让js操作html元素而制定的一个规范**。
2. 树状模型
3. 节点：object类型，是内置对象
   1. 每个HTML标签都是一个元素节点
   2. 标签中的文字是文字节点
   3. 标签的属性是属性节点
4. DOM操作就是操作节点
5. 是一套操作html和xml文档的API

#### 获取方法

* + 1. **直接获取**
       - 1. 通过ID：document.getElementsById()
         2. 通过标签名

document.getElementsByTagName()返回一个标签伪数组,所以后面得加数组索引

* + - * 1. 通过类名

document.getElementsByClassName() 返回一个标签伪数组，如果没有获取到，返回空数组

**他们两个的返回值是标签数组，习惯性是遍历之后再使用**。

let elements = document.getElementsByTagName('\*');   // 获取所有元素

elements[0]             // 第一个元素

elements**[elements.length - 1]**   // 最后一个元素

**element.querySelector()** 方法用于获取匹配指定选择器的**第一个**元素。没有则返回 null。document.querySelector('.product.featured')

**element.querySelectorAll()** 方法用于获取匹配指定选择器的**静态**元素集合。没有则返回 []。

let warriors=document.querySelectorAll('.warriors .warrior')

**ES6 新增的 for..of** 可用于遍历 HTMLCollection 和 NodeList。

for (let warrior of document.querySelectorAll('.warrior')) console.log(warrior)

**通过 Array.from() 可以将集合转成数组**。

Array.from(document.body.childNodes)

* + 1. **访问关系获取**

**var *box2*** = **document**.getElementsByClassName(**"box2"**)[0];  
**console**.log(***box2***.**parentNode**);

* + - 1. 节点访问关系是以属性的方式存在的
      2. .parentNode：获取父节点
      3. .nextElementSibling或.nextSibling
      4. .previousElementSibing或.previousSibing

**访问单个子节点**

firstElementChild:在火狐谷歌IE9都指的第一个元素节点

**console**.log(box2.**parentNode**.**firstElementChild**);

lastElementChild：在火狐谷歌IE9都指的最后一个元素节点。

**访问所有子节点**

子节点数组 = 父节点.childNodes; 获取所有节点。childNodes：它是标准属性，它返回指定元素的子元素集合，包括HTML节点，所有属性，文本节点

子节点数组 = 父节点.children; **用的最多。children：非标准属性，它返回指定元素的子元素集合。**但它只返回HTML节点，甚至不返回文本节点

**获取指定的兄弟节点**

**节点自己.parentNode.Children[index]**

**获取所有的兄弟节点（返回值是个数组）**

Function fn(ele）{

Var newArr=[];//定义一个新数组，用来装所有元素，将来返回

Var arr=ele.parentNode.children;

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

If(ele!==arr[i]){**newArr.push(arr[i]**)//如果不是传过来的兄弟元素本身，那么添加到新数组中}

}

Return newArr;

}

### 节点的操作

* 1. 节点的创建、添加、删除
  2. 节点的访问关系都是属性，节点的操作都是函数或者方法
  3. **创建节点**

var新标签 = document.createElement（“标签名”）

* 1. **插入节点**

1.父节点.removeChild（子节点）必须制定要删除的子节点

2.父节点.appendChild（新节点） 父节点的最后插入一个新节点

3.父节点.insertBefore（新节点，已知节点）

**E)删除节点**

* + - 1. 父节点.removeChild（’删除的元素（先获取）’）
      2. 父元素.parentNode.removeChild（’要删除的元素’）

**F)复制节点**

* + - 1. 新节点 = 要赋值的节点.cloneNode（true/false）；

true:**表示深复制（复制节点及其所有子节点）;**false:只复制当前节点,里面的内容不复制

## 节点属性

* **node.nodeType 节点类型 - 1 代表元素节点，3 代表文本节点，9 代表 document**
* node.nodeName 节点名称 - BODY DIV #text #comment #document 等
* node.nodeValue node.data 获取/设置节点文本内容
* node.innerText node.textContent 获取/设置节点及其后代的文本内容
* elem.tagName 标签名称-只有元素节点才具有该属性
* elem.innerHTML 以字符串形式获取/**设置**元素内部 HTML
* elem.outerHTML 以字符串形式获取/**替换**元素整体 HTML

## 元素样式

* **elem.style.\* 获取/设置行内样式 document.body.background = '#CDF5DC'**
* **elem.className 获取/设置 class 属性**
* **elem.classList.add() 添加CSS 类 el.classList.add('active') el.classList.add('product', 'discount')**
* **elem.classList.remove() 移除 CSS 类 el.classList.remove('player', 'active')**
* **elem.classList.toggle() 切换 CSS 类 el.classList.toggle('active', navigator.onLine)**
* **elem.classList.contains() 是否包含一个/多个 CSS 类 el.classList.contains('active')**
* **window.getComputedStyle() 获取元素经过计算的所有属性值 getComputedStyle(document.body)**

## 返回顶部链接

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

a.href = '#';

a.innerText = '返回顶部';

a.style.right = '10px';

a.style.bottom = '20px';

a.style.position = 'fixed';

document.body.appendChild(a);//

parent.appendChild(node) 将 node 添加到 parent 结尾

## HTML 属性:一般可通过 . 访问和设置。

Id/ href/ value/Hidden

#### 节点属性操作

* + - 1. 获取属性

a.元素节点.属性**或者**元素节点[属性]//操作DOM对象本身

**var** eleNode = **document**.getElementsByTagName(**"img"**)[0];  
**console**.log(eleNode.**src**);  
**console**.log(eleNode.**tagName**);  
**console**.log(eleNode[**"title"**]);  
**console**.log(eleNode[**"class"**]);  
**console**.log(eleNode[**"alt"**]);  
b.元素节点.方法();//操作标签  
**console**.log(eleNode.getAttribute(**"id"**));

节点名.setAttribute（“id”，“aa”）： 添加一id=“aa”的属性

节点名. removeAttribute（”id”）；删除ID属性

* + - 1. 修改class属性要写成className：：因为class是关键字
      2. .innerHTML修改标签中间的文字

## 作业

1. **京东广告栏**

**topBanner.className += " hide";//保留原类名，添加新类名**

**topBanner.className = "hide";//替换旧类名**

**topBanner.style.display = "none";**

1. **京东狗**
2. **切换图片（a连接+图片）**禁止a链接跳转  
   <a href="http://www.baidu.com" >切换</a><br>  
   <img id="img" src="image2/1.jpg" alt="美女1">  
   <script>  
    var a=document.getElementsByTagName("a")[0]  
    var img=document.getElementById("img")  
    a.onclick=function () {  
    img.src="image2/2.jpg"  
    return false;//其目的禁止a链接连续跳转  
    }  
   </script>

**4.显示和隐藏盒子（if语句判断+innerHTML修改文字）**

.show{  
 display: block;  
 }  
 .hide{  
 display: none;  
 }  
 </style>  
<body>  
<button id="btn">隐藏</button>  
<div id="div">js高级程序设计</div>  
<script>  
 var btn=document.getElementById("btn");  
 var div=document.getElementById("div");  
 btn.onclick=function () {  
  **if(this.innerHTML==="隐藏"){  
 btn.innerHTML="显示";//改文字**  **div.className="hide";**  
 }else{  
 btn.innerHTML="隐藏";  
 div.className="show";  
 }  
 }  
</script>

**5.美女相册（placehold.it制作400\*250的图）**  
<h2>美女画廊</h2>  
<a href="#">注册</a>  
<ul id="imagegallery">  
 <li><a href="image2/1.jpg"title="美女A"><img src="image2/1-small.jpg" alt="美女1" width="100px" ></a></li>  
 <li><a href="image2/2.jpg"title="美女B"><img src="image2/2-small.jpg" alt="美女2" width="100px"></a></li>  
 <li><a href="image2/3.jpg"title="美女C"><img src="image2/3-small.jpg" alt="美女3" width="100px"></a></li>  
 <li><a href="image2/4.jpg"title="美女D"><img src="image2/4-small.jpg" alt="美女4" width="100px" ></a></li>  
</ul>  
<div> </div>  
<img src="image2/placeholder.png" width="450px" id="image">  
<p id="des">选择一个图片</p>  
<script>  
  **//点击小图片，改变下面的大图片的src为a链接的href  
 //让p标签的innerHTML的属性值，变成a标签的title属性值**  
**var ul= document.getElementById("imagegallery");  
var aArr= ul.getElementsByTagName("a");**  
var img= document.getElementById("image");  
var des= document.getElementById("des");  
//以前一个个绑定事件，现在是一个数组，for循环遍历  
 for(var i=0;i<aArr.length;i++){  
 aArr[i].onclick=function () {  
 img.src=this.href;  
 des.innerHTML=this.title;//**写成arr[i].title出错**  
 return false;//不能少这句。**函数加载时，只加载函数名，不加载函数体；onclick延迟执行，只有点击才会去执行** }  
 }  
</script>

**例题1.为什么以下程序不能正确运行？**

(function() {

// 按照 tasks 的顺序和等待时间 wait 执行任务

var tasks = [

{ name: '出场 , wait: 2 },

{ name: '隐身 , wait: 3 },

{ name: '放大 , wait: 5 }

];

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

setTimeout(function() {

console.log('执行任务 ' + tasks[i].name); // 程序报错 Cannot read property 'name' of undefined

}, tasks[i].wait \* 1000);

}

console.log(i); // i 在整个函数作用域内可见

})();

**如何修改?**

**1.把var i改为let i**

2.(function() {

// 按照 tasks 的顺序和等待时间 wait 执行任务

var tasks = [

{ name: '出场 ', wait: 2 },

{ name: '隐身 ', wait: 3 },

{ name: '放大 ', wait: 5 }

];

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

**(function (index) {**

**setTimeout(function() {**

**console.log('执行任务 ' + tasks[i].name);**

**}, tasks[index].wait \* 1000);**

**})(i);**

**}**

console.log(i); // i 在整个函数作用域内可见

})();

**3.使用 Array.prototype.forEach() 遍历任务。**

[ { name: '出场', wait: 2 },

{ name: '隐身', wait: 3 },

{ name: '放大 😎', wait: 5 }

].forEach(function(task) {

setTimeout(function() {

console.log('执行任务 ' + task.name);

}, task.wait \* 1000);

});

例题2.for (var i = 0; i < 5; i++) {

setTimeout(function() {

console.log(i);

}, 1000 \* i);}

**//开始输出一个 5，然后每隔一秒再输出一个 5，一共 5 个 5。**

for (var i = 0; i < 5; i++) {

(function(i) {

setTimeout(function() {

console.log(i);

}, i \* 1000);

})(i);}

/**/输出 0 到 4**

for (var i = 0; i < 5; i++) {

(function() {

setTimeout(function() {

console.log(i);

}, i \* 1000);

})(i);

//**输出 5**

for (var i = 0; i < 5; i++) {

setTimeout((function(i) {

console.log(i);

})(i), i \* 1000);}

//**立马输出 0 到 4**

setTimeout(function() {

console.log(1)}, 0);new Promise(function executor(resolve) {

console.log(2);

for( var i=0 ; i<10000 ; i++ ) {

i == 9999 && resolve();

}

console.log(3);}).then(function() {

console.log(4);});console.log(5);

//首先先碰到一个 setTimeout，于是会先设置一个定时，在定时结束后将传递这个函数放到任务队列里面，因此开始肯定不会输出 1 。然后是一个 Promise，里面的函数是直接执行的，因此应该直接输出 2 3 。然后，Promise 的 then 应当会放到当前 tick 的最后，但是还是在当前 tick 中。因此，应当先输出 5，然后再输出 4 。

最后在到下一个 tick，就是 1 。//“2 3 5 4 1”

# 就业班第十天（DOM）

## 关闭和显示二维码

1. 加类名（hide）隐藏show显示

div.**className** = div.**className**.replace(**"hide"**,**"show"**);

## 表单属性

1. **显示二维码**

要求：***鼠标放到a链接上，显示二维码（添加show类名，去掉hide类名） 鼠标移开a链接，隐藏二维码（添加hide类名，去掉show类名）***  
 .**nodeSmall** {  
 **width**: 50**px**;  
 **height**: 50**px**;  
 **background**: **url**(**images/bgs.png**) **no-repeat** -159**px** -51**px**;  
 **position**: **fixed**;  
 **right**: 10**px**;  
 **top**: 40%;  
 }  
 .**erweima** {  
 **position**: **absolute**;  
 **top**: 0;  
 **left**: -150**px**;  
 }  
 .**nodeSmall a** {  
 **display**: **block**;  
 **width**: 50**px**;  
 **height**: 50**px**;  
 }  
 .**hide** {  
 **display**: **none**;  
 }  
 .**show**{  
 **display**: **block**;  
 }  
<**div class = "nodeSmall" id="node\_small"**>  
 <**a href="#"**></**a**>  
 <**div class = "erweima hide" id="er"**>  
 <**img src="images/456.png" alt=""**/>  
 </**div**>  
<**script**>  
 **window**.onload = **function** () {  
 **var** a = **document**.getElementsByTagName(**"a"**)[0];  
 **var** er = **document**.getElementById(**"er"**);  
 a.onmouseover = **function** () {  
 er.**className** = **"show"**;  
 }  
 a.onmouseout = **function** () {  
 er.**className** = **"hide"**;  
 }  
 }  
</**script**>  
 或 <**style**>  
 .**nodeSmall** {  
 **width**: 50**px**;  
 **height**: 50**px**;  
 **background**: **url**(**images/bgs.png**) **no-repeat** -159**px** -51**px**;  
 **position**: **fixed**;  
 }  
 .**nodeSmall a** {  
 **display**: **block**;  
 **width**: 100**px**;  
 **height**: 100**px**;  
 }  
 .**erweima**{  
 **display**: **none**;  
 }  
  **a:hover + .erweima {  
 display: block;**  
 }   
 </**style**>

**2.禁用文本框**

* 1. **disabled 禁用文本框----disabled：‘true’类型的值都可以**
  2. **Btn1.onclick=function () {**

**inp.disabled = "no"; }//禁用**

**Btn2.onclick=function () {**

**inp.disabled = false; }//解禁**

3.<lable for = “跳转到相应的ID”>

**4.输入框获取失去焦点事件**

* 1. 京东、淘宝搜索框

<**style**>  
 **label** {  
 **position**: **absolute**;  
 **top**: 32**px**;  
 **left**: 48**px**;  
 **font-size**: 12**px**;  
 **cursor**: **text**;  
 **color**: **#CCC**;  
 }  
 .**hide** {  
 **display**: **none**;  
 }  
 .**show** {  
 **display**: **block**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
京东 <**input id="jingdong"type="text"value="我是京东"**><**br**>  
淘宝 <**label for="taobao"**>我是淘宝</**label**><**input type="text"id="taobao"**><**br**>  
placeholder: <**input type="text"placeholder="我是placeholder"**>  
</**body**>  
<**script**>*//要求：京东的input按钮获取了插入条光标立刻删除内容。失去插入条光标显示文字。  
 //要求：在input中输入文字，label标签隐藏，里面的文字变成空字符串，label显示。* **var *jingdong*** = **document**.getElementById(**"jingdong"**);  
 ***jingdong***.**onfocus** = **function** () {  
 **if**(***jingdong***.**value** === **"我是京东"**){  
 ***jingdong***.**value** = **""**; }  
 }  
 ***jingdong***.onblur = **function** () {  
 **if**(***jingdong***.**value** === **""**){  
 ***jingdong***.**value** = **"我是京东"**; }  
 }  
 **var *taobao*** = **document**.getElementById(**"taobao"**);  
 **var *lab*** = **document**.getElementsByTagName(**"label"**)[0];  
 ***taobao***.oninput = **function** () {  
 **if**(***taobao***.**value** === **""**){  
 ***lab***.**className** = **"show"**;  
 }**else** {  
 ***lab***.**className** = **"hide"**;  
 }

**5.注册验证**

**需求：失去焦点的时候判断input按钮中的值，如果账号或密码在6-12个字符之间通过，否则报错。**

<**style**>  
 .**wrong** {  
 **border**: 2**px solid red**;  
 }  
 .**right** {  
 **border**: 2**px solid #91B81D**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 账号：<**input type="text" onblur="***fn*(**this**)**"**/><**br**><**br**>  
 密码：<**input type="password" onblur="***fn*(**this**)**"**/>  
 <**script**> *//3.书写事件驱动程序* **function** *fn*(aaa){  
 *//****html中的input标签行内调用****function的时候,是先通过window调用的function，所以打印this等于打印window*

*console.log(this)指的是window*

*console.log(aaa)指的是*<**input type="text" onblur="***fn*(**this)>** *//只有传递的this才指的是标签本身* **if**(aaa.**value**.**length** < 6 || aaa.**value**.**length**>12){  
 aaa.**className** = **"wrong"**;  
 }**else**{  
 aaa.**className** = **"right"**;  
 }  
 }  
 </**script**>

**6.点亮盒子** 需求：鼠标放到哪个button上，改button变成黄色背景（添加类）

<**style**>  
 **button** {  
 **margin**: 10**px**;  
 **width**: 100**px**;  
 **height**: 40**px**;  
 **cursor**: **pointer**;}  
 .**current** {  
 **background-color**: **yellow**;}  
 </**style**>  
<**body**>  
 <**button**>按钮1</**button**>  
 <**button**>按钮2</**button**>  
 <**button**>按钮3</**button**>  
 <**button**>按钮4</**button**>  
 <**button**>按钮5</**button**>  
 <**script**>**var *btnArr*** = **document**.getElementsByTagName(**"button"**);  
 *//2.绑定事件* **for**(**var** i=0;i<***btnArr***.**length**;i++){

*//每次循环绑定一个属性，属性值为该盒子的索引值  
// btnArr[i].setAttribute("index",i);  
 // 或 btnArr[i].index = i;*  
 ***btnArr***[i].onmouseover = **function** () {  
 *//排他思想（干掉所有人，剩下我一个）  
 //排他思想是和for循环连用* **for**(**var** j=0;j<***btnArr***.**length**;j++){  
 ***btnArr***[j].**className** = **""**;  
 }  
 **this**.**className** = **"current"**;

*// alert(this.getAttribute("index"));  
 // alert(this.index);*  
 }  
 }  
 </**script**>

**7.tab栏切换（js）**

***/需求：鼠标放到上面的li上，li本身变色(添加类)，对应的span也显示出来(添加类);  
//思路：1.点亮盒子。 2.利用索引值显示盒子。***

<**style**>  
 **\*** {  
 **padding**: 0;  
 **margin**: 0;  
 }  
 .**box** {  
 **width**: 500**px**;  
 **height**: 400**px**;  
 **border**: 1**px solid #ccc**;  
 **margin**: 100**px auto**;  
 **overflow: hidden;**  
 }  
 **ul** {  
 **width**: 600**px**;  
 **height**: 40**px**;  
 **margin-left**: -1**px**;  
 **list-style**: **none**;  
 }  
 **li** {  
 **float**: **left**;  
 **width**: 101**px**;  
 **height**: 40**px**;  
 **text-align**: **center**;  
 **font**: 600 18**px**/40**px "simsun"**;  
 **background-color**: **pink**;  
 **cursor**: **pointer**;  
 }  
 **span** {  
 **display**: **none**;  
 **width**: 500**px**;

**background-color**: **yellow**;  
 **text-align**: **center**;  
 **font**: 700 150**px**/360**px "simsun"**;  
 }  
 .**show** {  
 **display**: **block**;  
 }  
 .**current** {  
 **background-color**: **yellow**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div class="box"**>  
 <**ul**>  
 <**li class="current"**>鞋子</**li**>  
 <**li**>袜子</**li**>  
 <**li**>帽子</**li**>  
 <**li**>裤子</**li**>  
 <**li**>裙子</**li**>  
 </**ul**>  
 <**span class="show"**>鞋子</**span**>  
 <**span**>袜子</**span**>  
 <**span**>帽子</**span**>  
 <**span**>裤子</**span**>  
 <**span**>裙子</**span**>  
 </**div**>  
 <**script**>  
 **var *liarr*** = **document**.getElementsByTagName(**"li"**);  
 **var *spanarr*** = **document**.getElementsByTagName(**"span"**);  
 **for**(**var *i***=0;***i***<***liarr***.**length**;***i***++){  
 ***liarr***[***i***].**index** = ***i***;  
 ***liarr***[***i***].onmouseover = **function** () {  
 **for**(**var** j=0;j<***liarr***.**length**;j++){  
 ***liarr***[j].**className** = **""**;  
 ***spanarr***[j].**className** = **""**;  
 }  
 **this**.**className** = **"current"**;  
 ***spanarr*[this.index].className = "show";** }}// 或者***spanarr*[this.index].style.display=“block”;**  
 </**script**>  
</**body**>

## 作业

1.显示二维码

1. 禁用输入框
2. 输入框获取失去焦点事件
   1. 京东、淘宝搜索框
3. 注册验证
4. 设置下拉列表
5. 给文本框赋值，获取文本框的值
   1. for循环赋值和取值

7. 全选和反选

A）点击全选，再点击全不选

* 1. 让下面的按钮等于全选按钮的属性

8.点亮盒子

* 1. 鼠标放到哪那个亮，其他的都不亮
  2. 弹出自己的索引值

9.tab栏切换（js）

10.tab栏切换（精简版）

11.tab栏切换（复杂版）

12.隔行变色

# 就业班第十一天（DOM）

function getEle(id){

return document.getElementById(id);

}

\* 功能：给定元素查找他的第一个元素子节点，并返回

\* @param ele

\* @returns {Element|\*|Node}

\*/

function getFirstNode(ele){

var node = ele.firstElementChild || ele.firstChild;

return node;

}

\* 功能：给定元素查找他的最后一个元素子节点，并返回

\* @param ele

\* @returns {Element|\*|Node}

\*/

function getLastNode(ele){

return ele.lastElementChild || ele.lastChild;

}

\* 功能：给定元素查找他的下一个元素兄弟节点，并返回

\* @param ele

\* @returns {Element|\*|Node}

\*/

function getNextNode(ele){

return ele.nextElementSibling || ele.nextSibling;

}

\* 功能：给定元素查找他的上一个兄弟元素节点，并返回

\* @param ele

\* @returns {Element|\*|Node}

\*/

function getPrevNode(ele){

return ele.previousElementSibling || ele.previousSibling;

}

\* 功能：给定元素和索引值查找指定索引值的兄弟元素节点，并返回

\* @param ele 元素节点

\* @param index 索引值

\* @returns {\*|HTMLElement}

\*/

function getEleOfIndex(ele,index){

return ele.parentNode.children[index];

}

\* 功能：给定元素查找他的所有兄弟元素，并返回数组

\* @param ele

\* @returns {Array}

\*/

function getAllSiblings(ele){

//定义一个新数组，装所有的兄弟元素，将来返回

var newArr = [];

var arr = ele.parentNode.children;

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

//判断，如果不是传递过来的元素本身，那么添加到新数组中。

if(arr[i]!==ele){

newArr.push(arr[i]);

}

}

return newArr;

}

例题1 //获取box改为red

**var box = getEle("box")**;

box.style.backgroundColor = "red"

//获取第一个和最后一个子节点

var parent = box.parentNode;

getFirstNode(parent).style.backgroundColor = "yellow";

getLastNode(parent).style.backgroundColor = "yellow"

//获取所有的兄弟节点（返回值是数组，所以用for循环操作）

**var arr = getAllSiblings(box)**;

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

arr[i].style.backgroundColor = "black";

}

**例题2** <div id="menu">

<ul id="list">

<li class="current"><a href="#">首页</a></li>

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

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

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

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

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

</ul>

</div>

var ul = getEle("list");

var aArr = ul.getElementsByTagName("a");

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

aArr[i].onclick = function () {

//**点击哪个a链接，让他的父亲添加类**

this.parentNode.className = "current";

//**设置他的父元素的其他所有兄弟节点类名为空**

var otherArr = getAllSiblings(this.parentNode);

for(var j=0;j<otherArr.length;j++){

otherArr[j].className = "";}

}

}

### Style的属性

1.样式少的时候使用

console.log(box.style.backgroundColor);

2.**style是对象**

console.log(box.style[1]);

console.log(typeof box.style);

3.**值是字符串**，没有设置值是“”；

console.log(box.style.border);

4.命名规则，驼峰命名。和css不一样

console.log(box.style.backgroundColor);

5.设置了类样式不能获取。（**只和行内式交互，**和内嵌和外链无关）**<div class="box" style="width: 100px;height: 100px;">我爱你中国</div>**

6.box.style.cssText = “字符串形式的样式”；

console.log(box.style.cssText);

box.style.cssText = "width: 200px; height: 200px; line-height:200px;text-align:center;"

## 创建dom元素的三种方式

1. 第一种创建方式：document.write();

document.write("<li>我是document.write创建的</li>")写在那就创建在那;会被覆盖,不常用

1. **第二种：直接利用元素的innerHTNL方法。**（innerText方法不识别标签）

ul.innerHTML += "<li id='li1'>我是innerHTML创建的</li>"绑定属性和内容比较方便，较常用

1. 第三种：利用dom的api创建元素

document.createElement;也是比较多的，**指定数量的时候一般用它**

**var newLi = document.createElement("li");**

newLi.innerHTML = "我是createElement创建的";

var li1 = document.getElementById("li1");

ul.insertBefore(newLi,li1)

//父节点.insertBefore(新节点，参照节点);

或在父元素的最后面添加元素ul.appendChild(newLi);

**例题** **创建，添加，删除，替换**

<ul>

<li id="li1">我是参照标签</li>

</ul>

var ul = document.getElementsByTagName("ul")[0];

var li1 = document.getElementById("li1");

var li2 = document.createElement("li");

li2.innerText = "我是createElement创建的标签，用的是appendChild的方法添加的";

ul.append**Child**(**li2**);

//用insertBefore添加

var li3 = document.createElement("li");

li3.innerText = "我是createElement创建的标签，用的是insertBefore的方法添加的"

//父节点.insertBefore(新节点，参照节点);

ul.insertBefore(li3,li1);

//删除，替换

ul.removeChild(li3);

//父节点.replaceChild(newNode,oldNode);

ul.replaceChild(li3,li2);

//**克隆node.cloneNode(true);深层复制。**

for(var i=0;i<=3;i++){

var newLi = li3.cloneNode(true);

ul.appendChild(newLi);}

## 作业

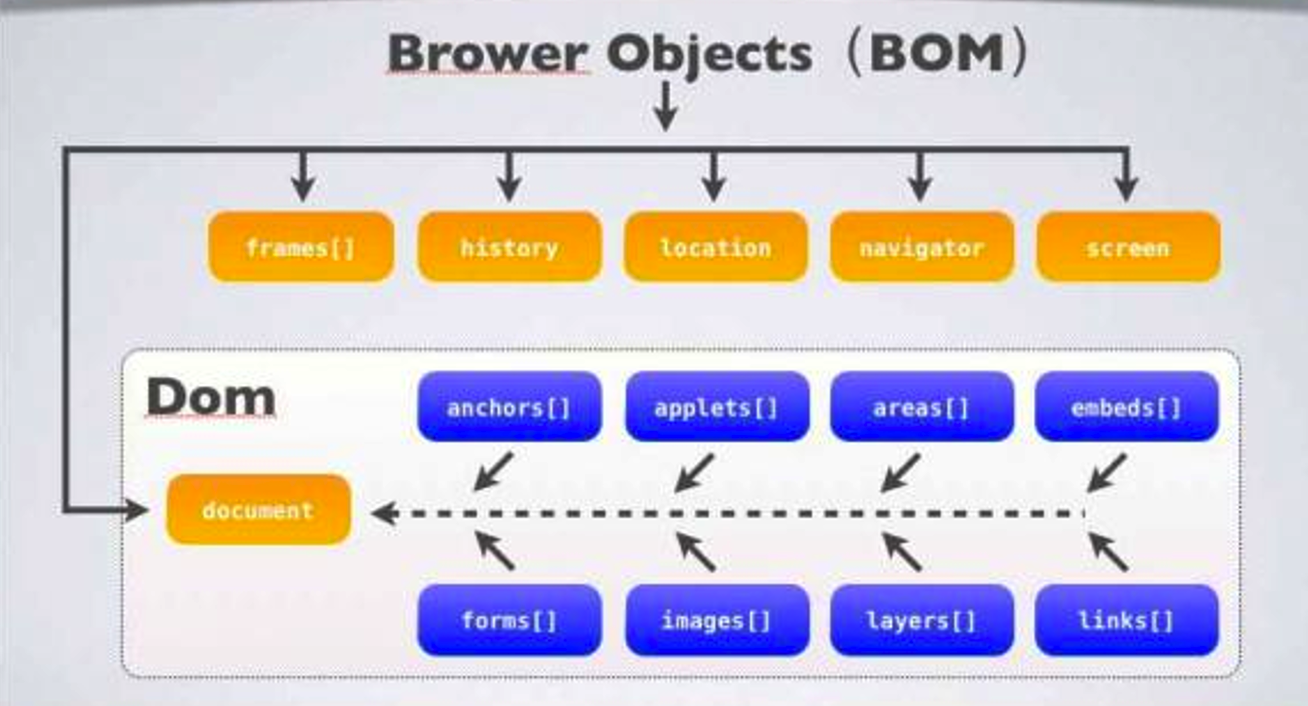
1. 改变盒子的大小透明度
2. 当前输入的文本框高亮显示
3. 高级各行变色
4. 百度皮肤
5. 定位和层级
6. 隐藏盒子
7. **动态创建列表，高亮显示**
8. **练习：祝愿墙**
9. **选择水果**

# 就业班第十二天（DOM）

## 知识点

1. indexOf（）：查询字符的索引
   1. indexOf（‘abcde’）
      1. indexOf（‘a’）的索引是0
      2. indexOf（‘abc’）的索引是0，按照第一个字符的索引
      3. indexOf（‘ac’）的索引是-1，查找不到ac连续的字符
      4. indexOf（‘f’）的索引是-1，查询不到返回-1

## BOM：浏览器对象模型



1. **window是javascript的顶级对象**，通常情况下可以省略window.alert(1)； prompt(); confirm();
2. 成员变量也是window的一个属性而已

var aaa = 1;

alert(aaa);

alert(aaa === window.aaa);//true

1. 自定义函数也是window的一个方法
2. window.open（url，target，param）
   1. url：地址
   2. target：打开新窗口的方式，\_blank/
3. window.close()关闭窗口
4. location.href = “跳转到的新网址”

**用div模拟a链接**  <div>点击我跳转到百度</div>

var div = document.getElementsByTagName("div")[0];

div.onclick = function () {

location.href = "http://www.baidu.com"; window.open("http://www.baidu.com","\_blank")}

**setTimeout**(function () {

location.href = "http://www.sina.com.cn";},5000);

炸弹定时器 ，只执行一次执行完报废

1. history 历史记录管理
   1. **Back（）：回退 或** go（-1）回退；0 是刷新
   2. **前进history.forward()或history.go(1)**

8.window.navigator 的一些属性可以获取客户端的一些信息

console.log(navigator.userAgent) 系统，浏览器

console.log(navigator.platform);浏览器支持的系统，win/mac/linux

## 定时器

1. **用途：setInterval()循环定时器；周而复始的执行（循环执行）**
2. **用途：setTimeout() 炸弹定时器；用完以后立刻报废（只执行一次）**
3. **//定义方法1(匿名函数)**

setInterval(function () {

console.log(1);

},1000);

1. **//定义方法2**

setInterval(fn,1000);

function fn(){

console.log(2); }

1. **//定义方法3**

setInterval(**"fn(3)",**1000);

function fn(aaa){

console.log(aaa);}//一定要带引号

1. 清除定时器 clearInterval(timer)

//setInterval他的返回值就是定时器的名字

var timer = setInterval(function () {

console.log(num);

num++

if(num===10){

//如何停止定时器呢？？？

clearInterval(timer); }

},500);

1. script中只能获取行内式样式，内联和内的无法获取

## 作业

1. **左右select切换水果**
   1. 点击按钮，所有子元素都移动到另一个select里面
   2. 点击移动一个的所有被选定的移动到另一个select里面
2. **在线会员列表**
   1. 数组元素用josn
   2. 先获取数组元素，再获取数组元素的josn
3. **祝愿墙**
   1. 关闭的时候要循环绑定事件
4. **模拟百度，搜索文本框**

# 就业班第十三天（DOM）

## 知识点

1. **内置对象**
   1. 语言自带的对象
   2. 提供了常用的、基本的功能
2. 打印数组和字符串不用for…in
3. 打印json的时候才用for…in

## Date

**复杂(引用)数据类型：Object、Array、Date、函数**

1. 获取当前时间（两种办法）
   1. var date = new Date（）；
   2. var date2 = new Date（‘2016/09/06 00：00：00’）；**定义一个指定日期的时间对象**
2. getDate（）：获取日期（1-31）

var date1 = new Date();

console.log(date1.getDate())

1. **getDay（）：获取星期（0-6）（0代表周日）**
   1. var a = date.getDay();  
      if (date.getDay() === 0){  
       a+=7;  
      }  
      **console**.log(a);
2. **getMonth（）：获取月份（0-11：0代表一月）**
   1. var a = date.getDay();  
      if (date.getMonth() === 0){  
       a+=12;  
      }  
      **console**.log(a);
3. getFullYear（）：获取完整年份
4. getHours（）：获取小时0-23
5. getMinutes（）：获取分钟0-59
6. getSeconds（）：获取秒0-59
7. getMilliseconds（）：获取毫秒（1s=1000ms）
8. **getTime（）：获取累计毫秒数（从1970/1/1午夜开始）**
9. **获取当前时间距离1970/01/01的毫秒值四种办法**

var date11 = Date.now();

var date22 = +new Date();

**var date33 = new Date().getTime()**;

var date44 = new Date().valueOf();

例题：**模拟日历**

//1.创建一个当前日期的日期对象

var date = new Date();

//2.然后获取其中的年月日和星期

var year = date.getFullYear();

var month = date.getMonth();

var hao = date.getDate();

var week = date.getDay();

// console.log(year+" "+month+" "+hao+" "+week);

//3.赋值给div

var arr = ["星期日","星期一","星期二","星期三","星期四","星期五","星期六"];

var div = document.getElementsByTagName("div")[0];

div.innerText = "今天是："+year+"年"+(month+1)+"月"+hao+"日 "+arr[week];

## **String**

1. **简单数据类型无法绑定属性和方法**：

var str = "abc";

console.log(str.aaa);//undefined

console.log(str.length);//3

console.log(str.indexOf("c"));//2

1. **String 内置对象，可绑定属性和方法**
   1. var strObj = new String（‘abc’）；
2. **string简单数据类型底层有数据转换**，当简单类型的string 调用了内置对象String的方法后，执行完方法后再转换回去
3. **给索引查字符：charAt/charCodeAt**
   1. charAt（字符索引值）；返回索引位置的字符

var str = new String("abcd");

console.log(str);

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

console.log(**str.charAt(i)**);}

* 1. charCodeAt（字符索引值 ）；返回索引位置字符的Unicode（万国码）编码
  2. **sort();底层用到了charCodeAt();**
     + 1. UTF-8
       2. UTF-16
       3. 都包含ASCII码ї
  3. 字符串长度和字符串所占字节数不同
     + 1. **一个汉字两个字节**
       2. 一个字母一个字节

例题**求一个字符串占有几个字符位**

var str = "I love my country!我你爱中国！";

//技术点：判断该字符是否在0-127之间。（在的话是英文，不在是非英文）

alert(getZFWlength(str));

alert(str.length);

function getZFWlength(string){

//定义一个计数器

var count = 0;

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

//对每一位字符串进行判断，如果Unicode编码在0-127，计数器+1；否则+2

**if(string.charCodeAt(i)<128&& string.charCodeAt(i)>=0 )**{

count++;

}else{

count+=2;}

}

return count; }

1. **给字符查索引：indexOf/lastIndexOf**
   1. 如果两个查相同的字符并且字符串里有两个相同的字符，返回的索引值不一样
   2. 内存的索引值在加载的时候已经固定，不会改变
2. url编码和解码
   1. **数据传递的时候经常需要通过编码后在传递，接收后还需要反编译回来。**
   2. encodeURLComponent（）：编码
   3. decodeURLComponent（）：解码
3. 字符串链接concat（）**连接两个字符串返回一个新字符串，并且不会被修改**
   1. var str = oldStr.concat(newStr);
4. 字符串截取
   1. var str = str1.slice（开始索引，结束索引）
   2. 只给开始索引，从索引截取到最后
   3. 索引值为负值，从后往前截
   4. 如果结束索引比开始索引小，则返回空字符串
   5. **包左不包右**
5. **substr根据长度和索引值截取字符串**
   1. var str = oldStr.substr（开始索引，截取长度）

var str = "I love my family!";

console.log(str.substr(2));//从索引截取到最后

console.log(str.substr(2,6));//**从索引截,长度个字符串 →love m**

console.log(str.substr(-3));//后几个

1. **Substring（慎用，错了差不出）同slice**
   1. **给负值就截取全部字符**
   2. **开始索引比结束索引大后们只能调换位置**

console.log(str.substring(2)); //从索引截取到最后

console.log(str.substring(2,5));//从索引值到索引值

**console.log(str.substring(-1)); //全部截取**

**console.log(str.substring(5,2));//只能调换**

1. **trim 去除字符串前后的空格**
   1. IE6/7/8不支持

var str1 = " a b c ";

console.log(str1);//" a b c "

console.log(str1.trim());//"a b c"

1. replace替换

**var str = oldStr.replace(匹配的字符，替换的字符)**

var str2 = "Today is fine day,today is fine day a!!!"

console.log(str2.replace(**/today/gi**,"tomorrow")) 忽略大小写，全局搜索//全局替换，否则就替换第一个

1. split 字符串变数组

**无参，是把字符串作为一个元素添加进数组中。**

空字符串，分隔字符串中每一个字符，分别添加进入数组中

**指定字符分隔数组：特殊符号将不会出现在数组的任意一个元素中**

var str3 = "关羽|张飞|刘备";

console.log(str3.split());//["关羽|张飞|刘备"]

console.log(str3.split(""));// ["关", "羽", "|", "张", "飞", "|", "刘", "备"]

console.log(str3.split("|"));//["关羽", "张飞", "刘备"]

1. str.toUpperCase（）：转换大写
2. str.toLowerCase（）：转换小写
3. **一个方法，包含三种获取元素节点的功能。**

通过传递参数的不同，来决定使用哪种获取方法。

**简单版**function getEle(str){

var str1 = str.charAt(0);

if(str1==="#"){

//返回获取的元素。（因为传递过来的值带有#，所以我们要从第二项开始截取并搜索）

return document.getElementById(str.slice(1));

}else if(str1 === "."){

return document.getElementsByClassName(str.slice(1);

}else{

return document.getElementsByTagName(str);}

}

**验证**  var divArr = getEle("div");

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

divArr[i].style.border = "2px solid #000";}

getEle("#box").style.backgroundColor = "red";

**最终版 function $(str){**

**var str1 = str.charAt(0);**

**if(str1==="#"){**

**return document.getElementById(str.slice(1));**

**}else if(str1 === "."){**

**return document.getElementsByClassName(str.slice(1));**

**}else{**

**return document.getElementsByTagName(str);**

**}**

**}**

**验证 var divArr = $("div");**

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

**divArr[i].style.border = "2px solid #000";}**

**$("#box").style.backgroundColor = "red";**

## Math

var num = -0.6

console.log(Math.abs(num)); 取绝对值

console.log(Math.floor(num)); 向下取整 向小取

console.log(**Math.ceil(num)**); //向上取整 向大取

console.log(**Math.round(num)**); //四舍五入取整 **正数四舍五入，负数五舍六入**

console.log(Math.random());//随机数0-1

Math.PI

Math.power(num,power) num的power次幂

Math.sqrt(num) num开平方

## 事件绑定（两种办法）

1. **addEventListener事件监听器**
   1. 不会被重叠，可给同一事件添加多个事件监听器
   2. **调用者.addEventListener("触发事件"，执行函数)**

btn.onclick = function () {

console.log("九尺龙泉万卷书，上天生我意何如。");

}

btn.onclick = function () {

console.log("不能报国平天下，枉为男儿大丈夫。");}

调用者.触发事件 = function（）｛｝会把前面的层叠掉

//**第二种事件绑定的方法不会出现层叠。（更适合团队开发）**

btn.addEventListener("click",fn1);

btn.addEventListener("click",fn2);

function fn1(){

console.log("九尺龙泉万卷书，上天生我意何如。");

}

function fn2(){

console.log("不能报国平天下，枉为男儿大丈夫。"); }

调用这是事件源，**参数1事件名(不带on)** ，参数2事件名(执行函数) **参数3事件名(捕获或者冒泡)**

1. 要想传参数到事件监听器中，可以使用匿名函数
   1. el.addEventListener("click", function(){fn("four")}, false);
2. 事件添加的兼容写法

**火狐谷歌IE9+支持addEventListener**

btn.addEventListener("click",fn1);

btn.addEventListener("click",fn2);

**IE678支持attachEvent**

btn.attachEvent("onclick",fn1);

btn.attachEvent("onclick",fn2);

1. **解绑事件**

第一种

btn.onclick = function () {

alert(1);}

**btn.onclick = null;**

第二种

btn.addEventListener("click",fn);

btn.attachEvent("onclick",fn);

// **btn.removeEventListener("click",fn);**

//  **btn.detachEvent("onclick",fn);**

1. **绑定与解绑兼容写法**

var btn = document.getElementsByTagName("button")[0];

EventListen = {

addEvent: function (ele,fn,str) {

//通过判断调用的方式兼容IE678

//判断浏览器是否支持该方法，如果支持那么调用，如果不支持换其他方法

if(ele.addEventListener){

ele.addEventListener(str,fn);

}else if(ele.attachEvent){

ele.attachEvent("on"+str,fn);

}else{

//在addEventListener和attachEvent都不存在的情况下，用此代码

ele["on"+str] = fn;}

},

removeEvent: function (ele,fn,str) {

if(ele.removeEventListener){

ele.removeEventListener(str,fn);

}else if(ele.detachEvent){

ele.detachEvent("on"+str,fn);

}else{

ele["on"+str] = null;}

}

}

EventListen.addEvent(btn,fn,"click");

EventListen.removeEvent(btn,fn,"click");

function fn(){

alert(1);}

1. this
   1. 在addEventListener：事件处理程序会在当前对象的作用域运行，this会指向被调用的对象，
   2. 在attachEvent：事件处理程序是在全局作用域下运行，**this会指向window**

## 作业

1. 模拟日历
   1. 打开获取当前时间
2. 苹果发布会倒计时
3. 索引查自负
4. 校验文件上传格式
   1. 找到value值

上传后获取上传文件的路径（value）

* 1. 转换为小写
  2. 找到最后一个点，lastIndexOf

# 就业班第十四天（特效）

## 三大家族offset/scroll/client和事件对象（event）)

1. event：事件被触动时，鼠标和键盘的状态，通过属性控制

### offset：偏移，检测

js中有一套方便的获取元素尺寸的办法就是offset家族；

offsetWidth和offsetHight 以及offsetLeft和offsetTop以及offsetParent共同组成了offset家族。

1. 获取元素尺寸
2. 检测盒子的宽高

offsetHeight和offsetWidth: 可以检测盒子的宽高。

**包括宽高本身，padding，border。不包括margin**

* 1. 事件源.offsetWidth
  2. 事件源.offsetHeight
  3. 返回的数值没有单位，number类型

3.offsetLeft和offsetTop （**检测距离父盒子有定位的左/上面的距离）**

* 1. 返回最近距离上级盒子，（以定位）的左上位置
  2. 父盒子的边框不算
  3. **从父盒子的padding开始计算**
  4. 事件源.style.left ；与offsetleft区别是**前者带PX**

1. offsetParent
   1. 检测父盒子中带有定位的父盒子节点
   2. 返回上级有定位盒子的标签
   3. 如果父系盒子都没有定位，返回body，
   4. 如果有定位（**除static之外的定位**）返回最近的
2. **offsetLeft和.style.left的区别**

**一、最大区别在于offsetLeft可以返回没有定位盒子的距离左侧的位置。**

**如果父系盒子中都没有定位，以body为准。**

**style.left只能获取行内式，如果没有返回“”;**

**二、offsetTop 返回的是数字，而 style.top 返回的是字符串，除了数字外还带有单位：px。**

**//div.offsetLeft = 100; div.style.left = "100px";**

**三、offsetTop 只读，而 style.top 可读写。（只读是获取值，可写是赋值）**

**//style.left和style.top可以赋值（没有style.bottom和style.right）**

**//offsetLeft和offsetTop只能获取值**

**四、如果没有给 HTML 元素指定过 top 样式，则style.top 返回的是空字符串。**

**//style.left只能获取行内式，如果没有返回“”;**

## 动画：animate

1. 闪动/匀速/缓动（缓慢停止）

闪动（一开始位置left为零）

btn.onclick = function () {

div.style.left = "500px"; }

**匀速运动**

btn.onclick = function () {

setInterval(function () {

//console.log(parseInt(div.style.left));//NaN不能用

//div.style.left = //parseInt(div.style.left)+10+"px"; //NaN不能用

//动画原理： 盒子未来的位置 = 盒子现在的位置 + 步长；

//**style.left赋值，用offsetLeft获取值。**

//style.left获取值不方便容易出现NaN；

//offsetLeft获取值特别方便，而且是**现成number方便计算**。因为他是只读的不能赋值。

div.style.left = div.offsetLeft + 10 + "px";

},300);

}

动画封装（简单版到最终版）

## 焦点图（静态部分、清除浮动、获取元素、点亮盒子和匀速运动封装）

1.获取事件源和相关元素

2.绑定事件和书写事件驱动程序(for循环绑定)

3.**点亮盒子**

4.**移动ul**(ul宽度设为500%，向左移动，给定目标为-值，放到第n个小方块上向左移动n-1个图片的宽)box>inner>ul+square

var inner = document.getElementById("inner");

var ul = **inner.children[0]**;

var spanArr = **inner.children[1].children**;

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

spanArr[i].index = i;

spanArr[i].onmouseover = function () {

for (var j = 0; j < spanArr.length; j++) {

spanArr[j].className = "";

}

this.className = "current";

animate(ul, -this.index \* 490);

}

function animate(ele, target) {

//要用定时器，先清除定时器

//一个盒子只能有一个定时器，这样儿的话，不会和其他盒子出现定时器冲突

//而定时器本身讲成为盒子的一个属性

clearInterval(ele.timer);

//我们要求盒子既能向前又能向后，那么我们的步长就得有正有负

//目标值如果大于当前值取正，目标值如果小于当前值取负

var speed = target > ele.offsetLeft ? 10 : -10;

ele.timer = setInterval(function () {

//在执行之前就获取当前值和目标值之差

var val = target - ele.offsetLeft;

ele.style.left = ele.offsetLeft + speed + "px";

//目标值和当前值只差如果小于步长，那么就不能在前进了

//因为步长有正有负，所有转换成绝对值来比较

if (Math.abs(val) < Math.abs(speed)) {

ele.style.left = target + "px";

clearInterval(ele.timer);

}

}, 30)

}

## 切换图

1. 鼠标放到图片(大盒子)上显示左右箭头
2. 获取两个按钮，点击左侧按钮移动ul向右走(每次只走一张)(计数器从0开始)。
3. 如何移动盒子。利用计数器模拟index值，点右侧自增，点左侧自减。

<div id="box" class="all">

<div class="ad">

<ul id="imgs">

<li><img src="images/1.jpg" /></li>....

</ul>

</div>

<div id="arr">

<span id="left"><</span>

<span id="right">></span>

</div>

</div>

<script>

//需求：鼠标放到盒子上，显示左右切换的图片。点击左则按钮图片(ul)向右移动，相反的按钮向左移动。

//思路：获取两个按钮，点击左侧按钮移动ul向右走(每次只走一张)(计数器从0开始)。

//如何移动盒子。利用计数器模拟index值，点右侧自增，点左侧自减。

//步骤：

//1.鼠标放上去显示移开以藏

//2.点击右侧盒子图片向做移动并用计数器模拟index值。

//3，点击左侧盒子，同理。

var box = document.getElementById("box");

var imgWidth = box.children[0].offsetWidth;

var ul = box.children[0].children[0];

var boxLeftRight = box.children[1];

var btnArr = boxLeftRight.children;

//鼠标放上去显示，移开隐藏

box.onmouseover = function () {

boxLeftRight.style.display = "block";}

box.onmouseout = function () {

boxLeftRight.style.display = "none";

}

//2.点击右侧盒子图片向做移动并用计数器模拟index值。

//定义计数器

var index = 0;

btnArr[1].onclick = function () {

index++;

//我们要对index的值进行约束。**index的值必须在[0,4]**

if(**index>ul.children.length-1**){

index = ul.children.length-1;

alert("到头了！");}

//点击盒子以后移动图片（ul，目标位置）

animate(ul,-index\*imgWidth);}

//3，点击左侧盒子，同理。

btnArr[0].onclick = function () {

index--;

if(index<0){

index = 0;

alert("第一张！");

}

animate(ul,-index\*imgWidth);

}

function animate(ele,target){ },10)

}

</script>

## 无缝滚动

1. 思路
   1. 复制第一张图片放到ul最后
   2. 当图片切换到第五张的时候，直接切换第六张（复制的那张）
   3. 再次从第一张切换到第二张的时候，先瞬间转到第一张图片，然后再滑动到第二张
2. 步骤
   1. 获取事件源以及相关元素
   2. 复制第一张图片所在的li添加到ul最后
   3. 给ol中添加li，ul个数见一个，并点亮第一个
   4. 鼠标放到ol的li上切换图片
   5. 添加定时器
   6. 左右切换图片

# 就业班第十五天（特效）

## 缓动动画

**开始特快越走越慢，步长越来越小**如类似刹车，电梯停止，压缩弹簧...

1. 动画原理 = 盒子位置 + 步长（步长越来越小）
2. 盒子位置 = 盒子本身位置 + （目标位置-本身位置）/n（最好为10）
3. 最后10像素容易出现问题所以要向上取整

**JS实际运算时会四舍五入取整**

* 1. 400-396 = 4 4/10 = 0.4 四舍五入 = 0
  2. 400-396 = 4 4/10 = 0.4 四舍五入 = 0

1. **要用定时器 先清除定时器**
2. 差值大于0的时候向上取整,差值小于0的时候向下取整

var　btn = document.getElementsByTagName("button")[0];

var　div = document.getElementsByTagName("div")[0];

var timer = null;

btn.onclick = function () {

//要用定时器，先清定时器

clearInterval(timer);

timer = setInterval(function () {

var target = 400;

// 步长用目标位置和盒子自身位置的十分之一

//最后10像素的时候都是1像素1像素的向目标位置移动，就能够到达指定位置。

var step = (target - div.offsetLeft)/10;

//**每次获取步长都向上取整**，这种情况就包含step<0.4的情况

//拓展：差值大于0的时候，向上取整，小于0的时候向下取整。

step = Math.ceil(step);

//动画原理：盒子未来的位置 = 盒子当前的位置+步长

div.style.left = div.offsetLeft + step + "px";

//跳出条件：目标位置-当前位置的绝对值，小于步长

if(Math.abs(400-div.offsetLeft)<Math.abs(step)){

div.style.left = 400+"px";

clearInterval(timer);

}

},30);

}

## 筋斗云

1. 导航栏，鼠标移动时筋斗云跟着移动，移开筋斗云回去
2. 鼠标移开，筋斗云跟着移动， 点击后固定

## 第二家族scroll

1. ScrollWidth宽度
   1. 获取宽
   2. **不包括border和margin**
   3. 包括padding
2. scrollHeight高度
   1. 如果内容没有溢出是盒子本身的高度
   2. **如果溢出是超出内容的高度总高度**
   3. IE7（包括IE7）以下是为盒子内容的高度
   4. **包括padding不包括border和margin**
3. scrollLeft
   1. 网页被卷去的左侧，调用者是document.body

写成document.body.scrollLeft

1. scrollTop
   1. 网页被卷曲的头部
   2. **存在兼容性问题**
2. onscroll 滚动事件window.onscroll
   1. 屏幕每滚动一次，哪怕只有1像素都会触发这个事件
   2. 只能使用一次，多了以最后一个为准
3. DTD，旧版本的浏览器遵循一些新的规则
   1. 未声明DTD，谷歌只识别DTD（ie9+认识他）

document.body.scrollTop

* 1. 已经声明DTD，IE6/7/8 只认识他，（IE9+任何时候）

document.documentElement.scrollTop

* 1. **火狐/谷歌/ie9+以上支持的（不管DTD）**

**window.pageYOffset**

1. documengt.title 访文挡标题
2. documengt.head访问文档头部
3. documengt.body访问文档主题

**10.document.document.Element 获取文档的HTML标签；基本结构当中的html标签并不是通过document.html 去访问的，而是document.documentElement**

## json

1. 有自定属性和属性值（键值对）的对象
2. 用for…in 遍历
3. var json = {"aaa":111,"bbb":222};

//遍历方法**for...in..**. (一般数组和dom对象不使用本方法)

for(var k in json){

**console.log(k)**; //属性

**console.log(json[k])**; //值：第二种属性绑定和获取值的方法 }

## 利用json封装scroll()

## 作业

1. 固定导航栏
2. 两侧小广告
3. 返回顶部小火箭
4. 跟随广告
5. **筋斗云**

# 就业班第十六天（特效）

## scroll家族

### 楼层跳跃

## 事件

1. event
2. mouseEvent和鼠标相关的事件
3. document.onclick：点击页面任何部位
4. event.type：事件类型
5. event. target：该事件被传递的对象
6. **pageY/X: 鼠标当前位置到页面顶部/左边距离**
7. **screenY/X：鼠标当前位置到屏幕顶部/左边距离**
8. **clientY/X：鼠标当前位置到浏览器（可视区域）顶部/左边距离**
9. 兼容IE67做page的封装
10. 兼容事件对象
11. 获取鼠标位置
12. 定时器

## 鼠标在盒子中的左上距离

1. 先获取鼠标距离页面的左上距离，
2. 然后获取盒子距离页面的左上距离
3. 大的减小的.
4. onmousemove：鼠标移动

## 放大镜

1. onmouseenter:鼠标进入
2. onmouseleave：鼠标离开
3. 事件源 = null 事件解绑
4. 清除被选文字getselected

## 作业

1. 鼠标跟随
2. 显示鼠标坐标
3. 放大镜
4. 拖拽
5. 模拟滚动条

# 就业班第十七天（特效）

## 知识点

1. **模拟滚动条的解除事件问题**
2. event内置对象，包含了大量  
   事件
3. page兼容性
   1. pageX || clientX + scool（）.top（页面卷去的距离）
4. if（true===a）true最好写前面
5. window.screen.width/height：检测屏幕分辨率
6. 屏幕宽高
   1. **window**.**screen**.**width**
   2. **window**.**screen**.**height**
7. 可视区域宽高
   1. function fn(){  
       if (**window**.**innerHeight** !== **undefined**){  
       return {  
       **'width'**:**window**.**innerWidth**,  
       **'height'**:**window**.**innerHeight** }  
       }else if (**window**.compatMode === "CSS1Compt"){  
       return{  
       **'width'**:**document**.**documentElement**.**clientWidth**,  
       **'height'**:**document**.**documentElement**.**clientHeigt** }  
       }else{  
       return {  
       **'width'**:**window**.body.**clientWidth**,  
       **'height'**:**window**.body.**clientHeight** }  
       }  
      }

## client对象（可视区）

1. clentWidth/Height:网页可是区域的宽/高
   1. **不包括margin和border**
   2. **padding+width**
   3. **clientTop/Left：获取border-top/Left的宽度**
2. 盒子调用指盒子本身
3. body/html调用，可视区域的大小
   1. function fn(){  
       if (**window**.**innerHeight** !== **undefined**){  
       return {  
       **'width'**:**window**.**innerWidth**,  
       **'height'**:**window**.**innerHeight** }  
       }else if (**window**.compatMode === "CSS1Compt"){  
       return{

**'width'**:**document**.**documentElement**.**clientWidth**,  
 **'height'**:**document**.**documentElement**.**clientHeight** }  
 }else{  
 return {  
 **'width'**:**window**.body.**clientWidth**,  
 **'height'**:**window**.body.**clientHeight** }  
 }  
}

1. 浏览器大小变化事件，window.onresize

## 三大家族的区别

1. **width/height**
   1. clientWidth = width+padding
   2. offsetWidth = width +padding + border
   3. scrollWidth 内容高度（不包含border）
2. **Top/Left**
   1. offsetTop/Left
      1. 距离父系盒子中带有定位的距离
   2. scrollTop/Left
      1. 浏览区被卷去的部分
   3. clientTop/Left
      1. 盒子的上/左border宽度
3. X/Y
   1. scrollX/Y
   2. clientX/Y
      1. 鼠标距离可视区域的距离（左上）

## 冒泡

1. 当一个元素上的事件被触发后。**同样的事件**在所有父元素上都被触发
   1. 从原始元素一直冒泡到DOM树最上层
2. 捕获冒泡目标阶段
   1. 捕获：从上往下顺序（DOM树结构）执行事件
   2. 冒泡：从下往上（DOM树结构）顺序执行事件
3. **不会冒泡的事件blur、 focus、 load、 unload、onmoiseenter、、onmouseleave**
4. event.bubbles事件是否冒泡
5. 取消冒泡
   1. IE9+/火狐/谷歌：event.stopPropagetion
   2. IE678：cancel.bubbles
   3. if （event&& event.stopPropagetion）｛
   4. event.stopPropagetion（）；
   5. ｝else｛event.cancleBubble｝

## 事件委托

1. 普通的事件绑定无法为新创建的元素绑定时间

## 捕获addEventListener

function fn (参数1，参数2，参数三)｛

if（参数3）｛

fn（参数1，参数3）；｝

｝

# 就业班第十八天（特效）

## 知识点

1. 缓动函数中opacity 写百分值的值
2. JS一般不用小数运算，会照成精度丢失
3. 元素的默\*认透明度是
4. 层级一次性赋值，不缓动
5. 利用for…in为同一个父元素的子元素绑定属性

## 函数截流

## 正则表达式

1. 定义方法
   1. var reg = new RegExpp(/abc/);  
      var reg = /abc/;
2. reg.test（‘表达式’）；检测是否表达式符合正则表达式，返回布尔值
   1. /abc/.test（‘表达式’）
3. 验证
4. 正则内部类
   1. 预定义类
      * 1. /./：除了换行和回车其他任意支付 \n \r “”空字符串不行
        2. /\d/:匹配数字
        3. /\D/:匹配非数字
        4. /\s/：空白
        5. /\S/：非空白
        6. /\w/：字母（包括下滑线不包括$）
        7. /\W/:非字母
   2. 简单类
      * 1. /sttt/.text（“sttt”） ： 什么都可以写，**必须是完整的**，只要是有sttt就能被匹配（有他就行）
        2. /[stasd]/.test（“t”）：只要包含stasd的**任意一项都可以**
   3. 负向类
      * 1. /[^abc]/.test（ds）：不能含有abc任意一个字符
        2. **不够和正好返回false**，**多了返回true，**
           1. /[^abc]/.test（“aiiiii”）返回true
           2. /[^abc]/.test（“a”）返回false
   4. 范围类
      * 1. /[A-Z]/
   5. 组合类
      * 1. /[a-z1-9A-Z]/
5. 正则边界
   1. ^$精确匹配：/^abc$/以abc为开头，以abc为结尾
6. 量词
   1. \*贪婪：/^a\*$/a的数量大于等于0
   2. +懒惰：/^a+$/a的数量大于等于1
   3. ？占有：/^a?$/a的数量为0或者1
   4. ｛｝：重复多少次
      * 1. ｛6｝：重复6次
        2. ｛n｝：重复n次
        3. ｛n，｝重复n或更多(>=n)
        4. ｛n，m｝，重复次数比n多比m少(包括n和m，n<=x<=m)
   5. | 或
   6. （）：提高运算优先级

# 就业班第十九天（jQuery）

**主要学习如何来使用jQuery操作DOM，其实就是学习jQuery封装好的那些功能方法，这些方法叫做API**

## 知识点

1. this指的是原生js的DOM对象
2. .css（“”）：只写一个值是取值，写俩值是赋值
3. window.onload === $(document).ready();
4. $(“”):获取元素 标签名、.类名、#id
5. 事件源.方法(function(){  
    //事件驱动函数  
   })

## jQuery特点

1. 链式编程
   1. jq.shou（3000）.html（内容）
      * 1. 相当于
           1. jq.shou（3000）
           2. jq.gtml（内容）
2. 隐式迭代
   1. 隐式实用for循环、迭代

## 如何实用jQuery

1. 引包
   1. 一定要在使用之前
   2. <script src = ‘’><>
2. 入口函数
   1. $（document）.ready（function（｛｝））
3. 绑定时间
   1. 事件源.事件（function（）｛事件驱动函数｝）

## jQuery入口函数

1. **文档加载完毕，图片未加载**的时候执行函数
   1. $（document）.ready（function（｛｝））
   2. $(function(){})
2. **文档和图片都加载完毕**执行函数
   1. $(window).reday(function(){})
3. JS入口只能有一次，多次会覆盖
4. jqery可有多个入口函数
5. **$ === jQuery**

## jQuery和DOM对象的区别

1. jQuery对象是一个数组，包含原生jsDOM对象
2. 原生js对象是一个原生对象
3. **jQuery对象名 = $（js对象名）**
4. **js对象转换为jQuery对象**
   1. jquery对象名[索引值]
   2. jquery对象名.get(wwm索引值）
5. 不同的对象有不同的方法
6. JQuery对象方法包含原生js对象方法.
7. jQuery对象.length的返回值是元素个数

## jQuery选择器

1. 基本：#id、.类、标签名、
   1. \*通配符选择器
2. 层级选择器
   1. 子代选择器 >
   2. 后代选择器 空格隔开
   3. 基本过滤选择器
      * 1. ： eq（index）遍历元素数组
        2. ：odd 获取偶数位索引的元素
        3. ：even获取奇数位索引值的元素
3. 筛选选择器
   1. find（selector）：找到父元素的素所有子元素
      * 1. **父元素.find（‘要查找的子元素’）**
        2. **查找所有后代子元素**
      1. children：亲儿子
         1. **父元素.children（‘要查找的子元素’）**
         2. **仅查找子元素**
   2. ：eq（index）
   3. next（）；下一个兄弟节点
   4. parent（）：获取该元素的父元素
   5. siblings：所有的兄弟节点（不包括本身）

# 就业班第二十天（jQuery）

## 样式属性操作.css

1. 操作方法
   1. $("div").css({  
       **'width'**:100,  
       **'height'**:100,  
       **'background'**:'red'  
       }

都写CSS样式

* 1. $("div").css("background","pink")
  2. $("div").css("background")
     1. 获取的是第一个样式

## 类操作

1. **.addClass添加类**
   1. $(“div”).addClass(“class”)
2. **.removeClass删除类**
   1. $(“div”).removeClass(“class”)
3. **.hasClass判断类**，判读是否有class这个类名
   1. $(“div”).hasClass(“class”)
   2. $(“div”).hasClass(“”)
   3. 只要有类名就商贩会true
   4. 一个类名都没有才返回false
4. **.toggleClass切换类**
   1. $(“div”).toggleClass（“class”）
   2. 有class类就删除 没有就添加class类

### 动画

1. 隐藏显示动画
   1. **.show/hide/toggle（）：什么都不写**
      1. **通过display：**block/none来实现隐藏和显示
   2. **.show/hide/toggle（毫秒值）：固定时间**
      * 1. 通过控制宽、高。透明度、display控制
        2. 各种属性只有在变换的期间存在，显示完毕后，宽高和opacity属性清除
   3. **.show/hide/toggle（fast/normer/slow）：字符串**
      * 1. slow（慢600ms）
        2. normer（正常400毫秒）
        3. fast/（宽200毫秒）
        4. 其他字符串默认按照normee执行
   4. **.show/hide/toggle（2000，function（）｛｝）：回调函数**
      * 1. 显示完毕之后执行回调函数
2. 滑入滑出动画
   1. .slidDown显示
      * 1. 无参数/毫秒数/字符串/回调函数
   2. .slidUp 隐藏
      * 1. 无参数/毫秒数/字符串/回调函数
   3. .slidToggle：切换
   4. 以定位的方向为准
      * 1. 比如buttom：0； 就会以下面的边为基准
3. 淡入淡出动画：控制透明度
   1. 无参数/毫秒数/字符串/回调函数
   2. fadeIn:显示
   3. fadeOut：隐藏
   4. fadeToggle：切换
   5. **fadeTo（毫秒值，透明度，function（）｛｝）：改到固定透明度**
      1. **0 全透 1 全不透**
4. 自定义动画
   1. .animeate（｛json串｝，时间，回调函数）
   2. 不支持百背景色
   3. 可以去w3c看都支持哪些属性
5. 停止动画
   1. .stop（false，false）停止执行当前正在执行的动画，执行下一个动画
   2. 都不写，默认都是false
   3. .stop（true，false）
      * 1. 清空队列，后续动画不执行
        2. 不立即完成当前动画，只是停止
   4. .stop（false，false）
      * 1. 不清空队列，后续动画执行
        2. 停止完成当前动画，执行下一个动画
   5. .stop（true，true）
      * 1. 清空动画队列，不执行以后的动画
        2. 立即完成当前的动画
   6. .stop（false，true）
      * 1. 不清空动画队列，后续动画执行
        2. 立即完成当前动画后，执行下一个动画
   7. 第一个参数判断是否清空队列。可以立即结束动画。
      * 1. true 清空
        2. false 不清空
   8. 第二个参数让当前正在执行的动画立即完成，并且重设show和hide的原始样式，调用回调函数等
      * 1. true 立即完成
        2. false 不立即完成

### 节点操作

1. **创建jQuery对象**
   1. $name = $(“<li></li>”)
   2. 创建一个li标签
   3. $(“标签名”).html(“内容”)
   4. html（“”）识别标签
2. **添加对象**
   1. $(“添加到哪”)append（“要添加的元素”）：在盒子最后添加一个元素
   2. newNode.appendTo（$(‘添加到哪’)）
   3. prepend（）在盒子最前面添加
   4. newNode.prepentTo（$(‘添加到哪’)）
   5. $（“”）.after（newNode）:兄弟之后
      * 1. 成倍添加
   6. $（“”）.before（newNode）：兄弟之前
3. **清空节点**
   1. 清空
      * 1. $(“node”).html(“空字符”)
        2. $(“node”).empty()
   2. 删除指定的元素
      * 1. $(“node”).remove();删除自身
4. **复制节点**
   1. var newNode = $(要复制的元素).clone（）；
   2. 默认深层复制

## 作业

1. 动态创建表格
2. <http://f2er.club/>
3. <https://www.awesomes.cn/>
4. [https://code.jquery.com/jquery/#jquery-all-1.x](https://code.jquery.com/jquery/" \l "jquery-all-1.x)

# 就业班第二十一天（jQuery）

## 知识点

1. 取消外边距合并 给父盒子加边框
2. prevAll
3. nextAll
4. each（function（index，js对象））｛
5. $(js对象）
6. ｝
7. $noConflict:放弃版本高的$的使用权,
   1. 版本高的写在下面
8. var new = $.noConflict(true)
   1. 放弃两种符号的使用权
9. asd

## 操作form表单

1. jqipt.attr(“属性名”，属性值)
   1. 获取 jqipt.attr（“属性名”）
   2. 添加类，jqipt.attr（“class”，“类名”）
   3. 移除属性 jqipt.removeAttr（“属性名”）
   4. 添加checked，selected，display用prpo（“属性名”，“属性值”）
2. val（）方法
   1. 获取标签中的value的属性
   2. $(“input”).val():获取
   3. $(“input”).val(“赋值内容”)
3. text（）方法获取文本值，不包含标签
   1. $(“div”).text()；获取内容
      * 1. innerText():不识别标签
        2. innerHTML（）：识别标签
        3. .html（）；识别标签
   2. $(“div”).text()；

## 尺寸操作

1. 获取宽高
   1. jq对象.height/width（）：只是内容尺寸，不包括padding和margin和border
2. 设置宽高
   1. jq对象.height/width（“200px”）：
   2. px可加可不加，不加不需要写双引号
3. 45

## 坐标值操作

1. offset（）.left/top:距离页面顶端或左边的距离
2. position().left/top：距离最近的以定位的父系盒子的距离
   1. 获取的值和padding、margin无关
3. **scrollLeft/top()：**被卷曲的头部
   1. 无参数的时候获取
   2. 有参数的时候设置

## 事件绑定

1. 事件源.bind（“事件1 事件二”，function（）｛｝）
   1. 可为多个触发事件绑定同一个方法
2. delegate（“事件源”，）
3. .on（“事件1 事件2”，“事件源”，｛（json串）“name”：111（传参数）｝，function（）｛｝）
   1. .on（“事件1 事件2”，“事件源”，｛（json串）“name”：111（传参数）｝，function（event）｛alert（event.data.name）｝）
      * 1. 获取name的属性

## 事件解绑

## 事件触发

浏览器触发事件

## 链式编程和隐式迭代

## JQuery复习

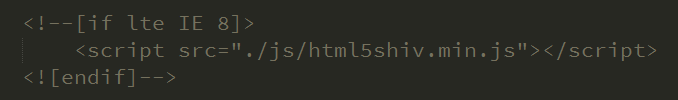
1. 版本从低到高引，先引jquery在引插件
2. JQuery对象和JS对象互相转换
3. 样式操作
4. 类操作
5. 动画
6. 节点操作
7. form表单
8. 属性操作
9. 位置操作
10. 事件对象
11. 事件解绑和绑定
12. 插件

# 就业班第二十二天（HTML5）

## 知识点

1. lang = “en” 所用语言是英文
2. 文档结构更简洁
3. IE8以下不支持h5c3

通过检测IE浏览器的版本来加载三方的一个JS库来解决兼容问题



1. 书写更宽松
2. 标签语义化：在合适的地方使用合适的标签
   1. 对seo优化有益
3. 网页经典布局
   1. 页头、导航、主题（左右）、页尾
4. max-wifth、min-width:最大最小宽度

## 语义化标签

1. header页头
2. nav导航
3. section：区块
4. aside：侧边栏
5. article文章
6. footer页脚

7.<figure> 表示媒介内容分组 与 ul > li 做个比较

## 智能表单

### 表单

1. 收集用户信息
2. h5自动验证表单
3. <form action="">

<fieldset>表单分组

<legend>表单名</legend>

新增

* 1. email：邮箱
  2. color：取色器
  3. url：网址
  4. search：搜索栏
     + 1. 带有删除符号
       2. 手机端会显示键盘
  5. number：只能输入数值
     + 1. step 步长
  6. tel：电话
     + 1. 没有自动验证
  7. time；时间
  8. week 周
  9. month月份
  10. date 日期
  11. datetime 时间
  12. range 滑动条

**可以设置value=“100”max=“300”min=“0”**

### 数据列表

1. <select >
2. <option>选项1</option>
3. <option>选项1</option>
4. <option>选项1</option>
5. <option>选项1</option>
6. </select>
7. <input type="text" **list="car"** />
8. <!--数据列表标签-->智能下拉菜单
9. <datalist **id="car"**>
10. <option>宝马</option>
11. <option>宝骏</option>
12. <option>奥迪</option>
13. <option>奥迪奥</option>
14. <option>迪奥</option>
15. </datalist>

### 加密

<keygen>：保证数据传输安全

### meter：度量器

low：低于该值后警告

high：高于该值后警告

value：当前值

max：最大值

min：最小值

### Progress进度条

设置value max min

### 表单新增属性

1. ***placeholder***：提示文字（占位符）
   1. 输入文字后，提示文字消失

placeholder="廖建树"

1. ***autofocus***：自动获取焦点
2. ***autocomplete***：自动匹配已经输入过的内容
   1. on（默认开启）autocomplete=“on”
   2. off关闭
3. ***required***：必填项，空值会提示
4. ***multiple***：多选

<input type="file" name="file" multiple>

1. ***novalidate***：关闭表单验证功能（只能加给form）
2. ***pattern***= “正则表达式”：自定义正则验证

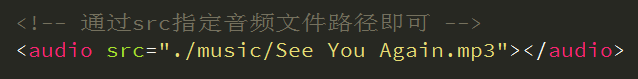
Pattern=“1/d{10}”

### 表单事件

1. oninput：一般用作字数统计
   1. 事件源.oninput
2. oninvalid：验证不通过的时候触发，用来设置验证不通过的时候的提示文字，
   1. 事件源.setCustomValidity（“提示文字”）

## 多媒体

1. 之前在网页上播放多媒体必须依赖第三方插件
   1. Mediaplay,快播,flash插件
2. 现在在H5里面提高了视频、音频标签



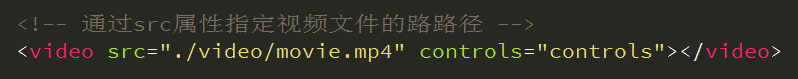
1. ***audio音频标签、video视频标签***
   1. ***controls:播放控制条***
   2. ***autoplay：自动播放***
   3. ***loop：循环播放***
   4. **音频支持：mp3/wav/ogg**
   5. **视频支持：mp4、ogg、webm**
      * 1. ***Vidio audio行内块显示***

***推荐兼容写法：***

* 1. **<audio controls loop>  
      <source src = "music.mp3">  
      <source src = "music.ogg">  
      <source src = "music.wav">**

**抱歉，你的浏览器不支持音频  
</audio>**

1. video视频播放



<video controls autoplay>

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

<source src="video/movie.ogg"/>

<source src="video/movie.webm"/>

抱歉，浏览器要换了！

</video>

## DOM拓展

1. document 文档对象

### 类操作

* + - 1. var new = document.querySelector（“选择器”）
         1. 只会选择符合条件的第一个元素
      2. var newArr = document.querySelectorAll(“选择器”)

var box1= document.querySelector('.box1');

* + - 1. **Box1.classList.add(“类名”)：添加类名**
      2. **Box1.classList.remove(“类名”):删除类名**
      3. **Box1.classList.contains（“类名”）：是否包含某个类名 返回布尔值**
      4. **Box1.classList.toggle(“类名”):切换类名**

**document.querySelector('button').onclick=**

**function(){document.querySelector('.box3').classList.toggle('active');}**

### 自定义属性

* + - 1. **给标签添加自定义属性必须以data开头，比如data-name**
      2. 获取标签属性

A)原有属性:box.属性名

B)自定义属性:box.dataset[‘data后的部分’]

Box.dataset[“name”]

<div class="box" title="盒子" **data-name**="zs" **data-content**="我是一个div">div</div>

<script>

Var box=document.querySelector('.box');

console.log(box.title);

console.log(**box.dataset['content']**);

</script>

如果自定义属性的属性名为**data-my-name**，那么获取的时候要

box.dataset**[“myName”]**

设置自定义属性的值

box.dataset['content']='aaaa';

# 就业班第二十三天（CSS3）

## 知识点

1. 兼容性差
2. **移动端优于pc端**
3. 渐进增强原则
4. 考虑用户群体
5. 遵照产品的方案
6. **.box$\*8**
7. **手册一些常见的符号**

[] 表示全部可选项

|| 表示或者

| 表示多选一

？ 表示0个或者1个

\* 表示0个或者多个

{} 表示范围

## 选择器

* 1. div+p：选择div后第一个p（紧跟div）
  2. div~p：选择div后所有P

### 属性选择器

1. E[attr] 表示存在attr属性即可；**div[class]**

2、E[attr=val] 表示属性值完全等于val **div[class=mydemo]**

3、E[attr\*=val] 表示的属性值里包含val字符并且在“任意”位置；**div[class\*=mydemo]**

4、E[attr^=val] 表示的属性值里包含val字符并且在“开始”位置；**div[class^=mydemo]**

5、**E[attr$=val]** 表示的属性值里包含val字符并且在“结束”位置；div[class$=demos]

### 伪类选择器

1. ：link ：正常状态
2. ：visited 已访问过的链接
3. ：hover 鼠标经过
4. ：action 点击
5. **按照“lvha”顺序**

#### 结构伪类

**:first-child选择某个元素的第一个子元素；**

li:first-child {background: green;}/\*选择li父元素的第1个元素的背景设为绿色\*/

**:nth-of-type(n)选择指定的元素**；它只会匹配相同标签的兄弟元素。也就是在**:nth-child的基础上加了一条限制：标签要一致**。

对于:nth-child，纳入考量的永远是同属同一个父容器下同一级所有的兄弟元素。而对于:nth-of-type来说，则是同一父容器下，同一级所有兄弟元素中标签类型相同的元素

**:nth-child(n)选择某个元素的一个或多个特定的子元素；**

:last-child选择某个元素的最后一个子元素

:nth-last-child()选择某个元素的一个或多个特定的子元素，从这个元素的最后一个子元素开始算

:nth-last-of-type()选择指定的元素，从元素的最后一个开始计算

**:first-of-type选择一个上级元素下的第一个同类子元素；**

:last-of-type选择一个上级元素的最后一个同类子元素；

:only-child表示的是一个元素是它的父元素的唯一一个子元素

:only-of-type选择一个元素是它的上级元素的唯一一个相同类型的子元素

:empty选择的元素里面没有任何内容

****nth-child(n+5)****这个选择器是选择从第五个元素开始选择。  
:nth-child(-n+5) 选择前面五个

例题1

<div class="wrap">

<h1 class="child">h1</h1>

<h2 class="child">h2</h2>

<div class="ct">

<h1 class="child">ct</h1>

</div>

</div>

.child:first-child{

color:yellow;

}//h1 ct文字会变黄色

.child:first-of-type{

color:yellow;

}//h1 h2 ct文字都会变色

#### 状态伪类

##### E:empty 如果盒子是空的就会被选中

##### E:target 目标伪类

配合锚点链接使用，点击锚点后，锚点链接的目标被激活，叠加样式

## 伪元素

1. 伪就是假的，通过css模拟出html效果，必须有**content属性**
2. 双：：
3. ：：before，在之前添加
   1. div::before{  
       content : "在前面";  
      }
4. ：：after，在之后添加
   1. div::after{  
       content : "在后面";  
       color: #f00;  
      }
5. **产生的伪元素相当于div的子元素**
6. 可以使用标签的属性

## 伪元素选择器

在大盒子下选中某一款区域用盒子包裹起来，可以单独设置样式

1. ：：first-letter 选择第一个字母
2. ：：first-line 选择第一行
3. ：：selection 选择区域的样式
   1. 一般只设置背景颜色和字体颜色

4.float：left 文字环绕

## 颜色

1. opacity属性会被子代继承，无法更改子代的属性
2. transparent；完全透明，无法修改属性值
3. ***RGBA模式***
   1. 红绿蓝
   2. alpha（透明度）
4. ***HSLA模式***
   1. H:色相、色调 0-360（红橙黄绿青蓝紫）
   2. S:饱和度 0-100%
   3. L:明度 、亮度 0-100%
   4. A:透明度 0-1 0（完全透明）~1（完全不透明）

## 文本

1. 文本阴影
   1. text-shadow：水平距离 垂直距离 模糊程度 颜色
   2. **浏览器向右向下为正方向**

.tu{

text-shadow: -1px -1px 1px #fff, 1px 1px 1px #000;

}

.ao{

text-shadow: -1px -1px 1px #000, 1px 1px 1px #fff;

}

* 1. 一个文本可以有多个阴影，用逗号隔开



## 盒子模型

1. 传统盒子模型
   1. 盒子的最终宽度 = padding+border+width
2. css3盒子模型
   1. box-sizing：content-box/padding-box/border-box
   2. 设置谁，谁不变
   3. content-box：内容盒子，外加盒子，加大边距和边框会加大总体大小，内容区域大小不变
   4. padding-box：内边距盒子
   5. border-box：边框盒子，内减盒子。加内边距和边框只会减小内容的大小

例题1 div{

width: 200px;

height: 200px;

background-color: red;

margin-top:20px;

}

.box3{

border:25px blue solid;

padding:25px;

/\* 设置的放内容区域大小,盒子宽高各300px\*/

**/\*外加模式\*/**

box-sizing: content-box;}

.box2{

/\* 设置盒子模型中最大盒子大小，盒子宽高还是200px,如果在加padding border 只会减小放内容的区域\*/

**/\*内减模式\*/**

box-sizing: border-box;

border:25px blue solid;

padding:25px;}

例题2.

li{

float: left;

margin-right:20px;

}

li img{

width: 316px;

height:170px;

/\* 设置了图片border-box 的大小\*/

box-sizing: border-box;

}

li:hover img{

border:5px solid #666;

}//鼠标悬浮在图片上面时，盒子并没有加宽

私有化前缀

1. 解决兼容性问题
2. **-webkit- 谷歌、苹果浏览器内核**
3. **-moz- 火狐浏览器内核**
4. **-ms- IE浏览器内核**
5. **-o- 欧鹏浏览器内核**

# 就业班第二十四天（CSS3）

## 知识点

1. ：hover只能经过父元素 改变子元素
2. 不可以兄弟元素之间互相改变 以及无关元素相互改变

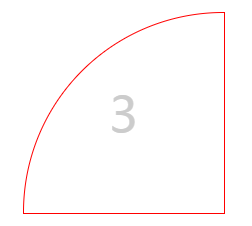
## 边框圆角

1. border-radius
   1. **border-top-left-radius：水平半径 垂直半径**
   2. border-radius：水平半径/垂直半径
   3. 单位：百分数和像素，最好使用%
   4. 赋值顺序 从左上开始，顺时针赋值,**如果当前角没有值，取对角的值**

**border-radius: 20px 60px 100px;//右上左下60px**

border-radius: 20px 60px//左上右下一样值

**扇形**（宽200px,高200px）

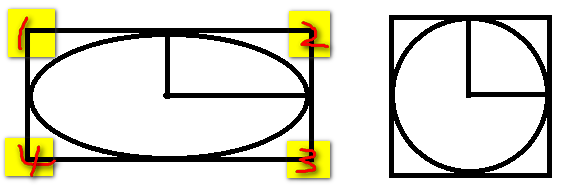


div:nth-child(3){

border-radius: 200px 0 0 0;

}

**椭圆**（宽200px,高200px）



可分别设置长、短半径，以“/”进行分隔，遵循“1，2，3，4”规则，“/”前面的1~4个用来设置横轴半径（分别对应横轴1、2、3、4位置 ），“/”后面1~4个参数用来设置纵轴半径（分别对应纵轴1、2、3、4位置 ）

(1)div:nth-child(4){

height:100px;

border-radius: 50%;

}//横轴比较长

(2)div:nth-child(5){

width: 100px;

border-radius: 50%;

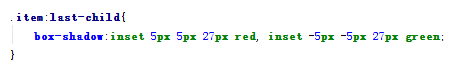
}//纵轴比较长

## 边框阴影

*box-shadow: 5px 5px 27px red, -5px -5px 27px green;*

模糊度是不能为负值；

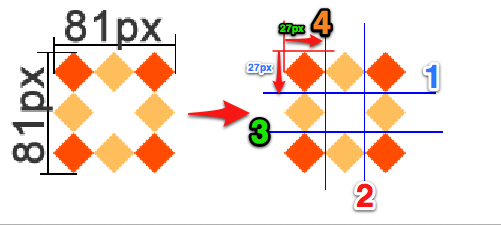
**box-shadow: 水平位移 垂直位移 模糊程度 阴影大小 阴影颜色 外/内阴影(inset) 默认外阴影不必写**



设置边框阴影不会改变盒子的大小，即不会影响其兄弟元素的布局。可以设置多重边框阴影，实现更好的效果，增强立体感。

## 边框图片

**border-image:url("images/border.png") 27/20px round**



* 1. 路径source
  2. 裁剪slice
  3. 宽度border-image-width
  4. 平铺repeat

## **repeat :正常平铺 但是可能会显示不完整**

## **round: 平铺 但是保证 图片完整**

## **stretch: 拉伸显示背景**

作业：用代码实现Android logo

## 背景图片大小

**background-size:width,height 可以设置背景图片的宽度以及高度**

* + - 1. 背景尺寸：数值，百分比，cover，contain

cover会自动调整缩放比例，保证图片始终填充满背景区域，如有溢出部分则会被隐藏。

contain包含:**最大化的等比例显示在盒子中，**不会铺满盒子

* + - 1. Cover，contain都会自适应盒子尺寸
      2. **背景原点background-origin(原点，起点)**设置背景定位的原点：控制背景从什么地方开始显示 background-origin: content-box;
         1. 默认背景从padding开始平铺
         2. 可选值：border-box，content-box
      3. background-clip背景裁剪
         1. **border-box :从 border-box开始裁剪**
         2. **padding-box :从 padding-box开始裁剪**
         3. **content-box :从 content-box开始裁剪**
         4. 属性值要是比原点的范围大，不起作用
      4. 多背景用逗号隔开

给盒子加多个背景，按照背景语法格式书写，多个背景使用逗号隔开

background: url(images/bg1.png) no-repeat left top

,url(images/bg2.png) no-repeat right top

,url(images/bg3.png) no-repeat right bottom

,url(images/bg4.png) no-repeat left bottom

,url(images/bg5.png) no-repeat center;

自适应例子：头顶各有一个背景图片，中间插入背景色

background: url(./images/head.jpg) left top no-repeat,

url(./images/foot.jpg) left bottom no-repeat;

background-color: #4A00AB;

## 渐变

渐变是CSS3当中比较丰富多彩的一个特性，通过渐变我们可以实现许多炫丽的效果，有效的减少图片的使用数量，并且具有很强的适应性和可扩展性。**可分为线性渐变、径向渐变**

### 线形渐变gradient

1. 渐变颜色属于背景图
2. background-image：linear-gradient（方向，起始颜色，终止颜色）；
   1. to right：从左到右，to bottom 向下（**默认从上向下）比如background-image:linear-gradient(yellow ,green)**
   2. **方向可以写角度 deg**

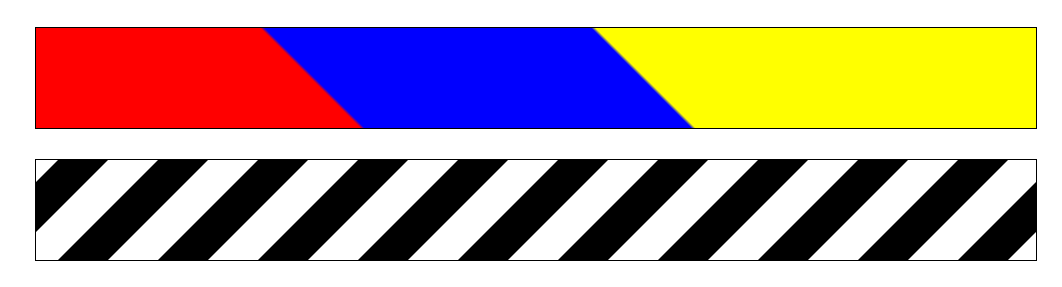
**background-image**: **linear-gradient**(135**deg**,**yellow**,**green**)

c）**多颜色渐变**

**background-image: linear-gradient(to right,red 0%,blue 30%,yellow 60%);**

**D)颜色突变**

background-image: linear-gradient(45deg,red 0%,red 30%,blue 30%,blue 60%,yellow 60%,yellow 100%);



**E)动态变化（用到动画）**

div:nth-child(6){

background-image: linear-gradient(135deg,#000 0%,#000 25%, #fff 25%, #fff 50%,#000 50%, #fff 75%, #fff 75%, #fff 100%);

background-size:100px 100%;

**animation: gun 1s infinite linear;**

}

**@keyframes gun** {

0%{

}

100%{

background-position: 100px 0px;

}

}

### 径向渐变

radial-gradient径向渐变指从**一个中心点开始沿着四周产生渐变效果**

1. **radial-gradient（辐射半径, 中心点位置，起始颜色，终止颜色）**
2. **中心位置参照的是盒子的左上角**，at left，right，top，bottom，center
3. **关于辐射范围：**其半径可以不等，即可以是椭圆。50px（水平距离） 50px（垂直距离）

例如 background-image: radial-gradient(100px 50px at center,yellow ,green);

## 过渡（transition）

过渡是CSS3中具有颠覆性的特征之一，可以实现元素不同状态间的平滑过渡（补间动画），经常用来制作动画效果。

**补间动画**：自动完成从起始状态到终止状态的的过渡。不用管中间的状态。**帧动画**：扑克牌切换.通过一帧一帧的画面按照固定顺序和速度播放。如电影胶片

特点：当前元素只要有“属性”发生变化时，可以平滑的进行过渡。

1. transition：过渡属性 过渡时间 速度曲线（linear：匀速） 延迟时间

比如：transition:width 2s,background-color 2s

* 1. transition-property：过渡属性
  2. transition-duration：过渡持续时间
  3. transition-timimg-function：运动曲线
     + 1. linear：匀速
       2. ease：减速
       3. ease-in：加速
       4. ease-out：减速
       5. ease-in-out：先加速后减速
  4. transition-delay：过渡延迟,超过时间后执行动画.

1. **如果需要所有属性都过渡**
   1. transition-property：all；

例题：.box{ width:200px;

Height:200px;

Background-color:red;

**transition:all 4s linear 1s ;**

**过渡必须加给盒子本身**

}

.box:hover{

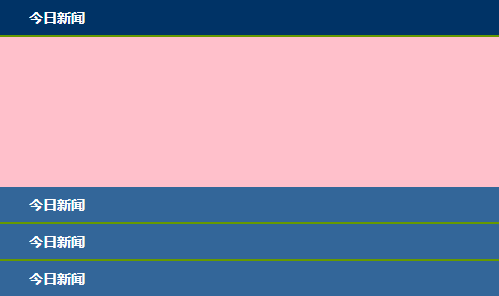
width: 600px;

background-color: blue;

height:400px;

}

例题1手风琴菜单



.list .pictxt {

height: 0px;

background: pink;

**transition: all 0.3s ease 0s;;**

}

.list:hover h3 {

background: #036;

}

.list:hover .pictxt {

height: 150px;

}

<div class="list">

<h3>今日新闻</h3>

<div class="pictxt"></div>

</div>

例题2冒泡



.pop {

width: 300px;

height: 120px;

margin: 30px auto;

**background:url("images/paopao.png")left bottom no-repeat, url("images/paopao.png") right top no-repeat;**

background-color: #036;

transition: all 1s linear;

}

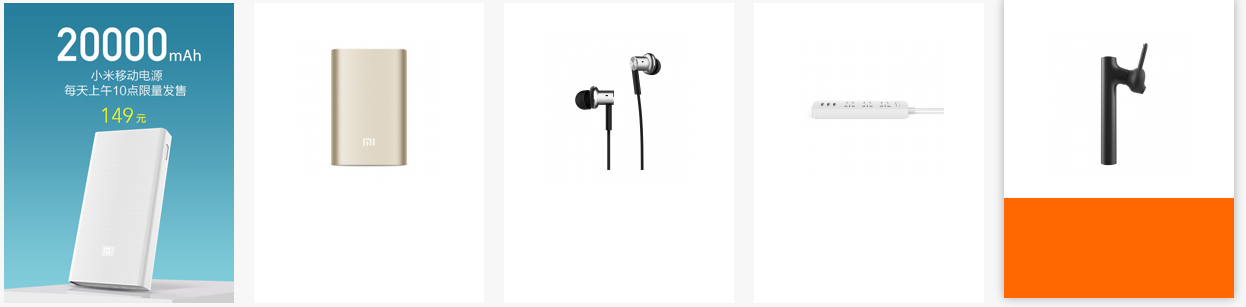
.pop:hover{

background-position:left top, right bottom;

}

<div class="pop"></div>

例题3 仿小米商品列表（阴影+过渡+盒子往上移动3px）



.item{

Position:relative

overflow: hidden;/\*先隐藏起来\*/

**transition: all .5s;}**

.desc {

position: absolute;

bottom: -100px;/\*盒子定位，先隐藏起来\*/

width: 100%;

height: 100px;

background-color: #ff6700;

**transition: all .5s;**

}

.item:hover {

margin-top: -3px;

box-shadow: 0 0 15px #AAA;

}

.item:hover .desc {

bottom: 0;

}

<div class="items">

<div class="item">

<img src="./images/1.jpg" alt="">

</div>

<div class="item">

<div class="pic">

<img src="./images/2.jpg" alt="">

</div>

<div class="desc"></div>

</div>

## 2D变换（transform）

转换是CSS3中具有颠覆性的特征之一，**可以实现元素的位移、旋转、变形、缩放**，甚至支持矩阵方式，配合即将学习的过渡和动画知识，可以取代大量之前只能靠Flash才可以实现的效果。在css3 当中，通过transform(变形) 来实现2d 或者3d 转换,其中**2d 有缩放、移动、旋转**

1. Transform:none；无变化
2. 缩放scale（倍率）
   1. scale(x, y) 可以对元素进行水平和垂直方向的缩放，x、y的取值可为小数，不可为负值，**不可以写百分数**
   2. 大于1是放大，小于1缩小
   3. 写一个值，宽高等比缩放
   4. **内容也会等比缩放**
   5. 写两个值，第一个是水平缩放倍率，第二个是垂直缩放倍率
3. 移动translate（）：可以改变元素的位置，x、y可为负值；
   1. 右移正数，左移负数（右下为正，左下为负）
   2. 第一个值水平移动，第二个值垂直移动距离
   3. **如果只写一个值水平移动，写百分比是相对于自身移动**
   4. **盒子在父盒子水平居中方法（子绝父相）** 
      * 1. **position：absolute；**
        2. **left：50%；**
        3. **transform：translate（-50%）；**
4. 旋转：rotate（角度）可以对元素进行旋转
   1. 正值顺时针，负值逆时针
   2. **旋转中心：transform-origin：水平坐标 垂直坐标**
      * 1. 默认旋转中心：几何中心

5.**skew()** 函数用于指定 X 轴方向 和 Y 轴方向倾斜角度

skewX() 函数用于指定 X 轴方向 倾斜角度 skewX(20deg)

skewY() 函数用于指定 Y 轴方向 倾斜角度 skewY(30deg)

**例题1小火箭升天**.rocket{

width: 50px;

position: absolute;

left:0;

bottom:0;

transform:translate(-100px,100px) rotate(45deg);//初始状态

/\* 过渡\*/

transition:all 1s;

}

body:hover .rocket{

transform:translate(500px,-500px) rotate(45deg);//结束状态 }

**例题2盾牌复原（先把他们打乱）**

.box img{

transition:all 2s;

}

/\* 打乱正常的图片\*/

.box img:nth-child(1){

transform:translate(100px,100px) rotate(45deg);

}

.box:hover img{ transform:translate(0px,0px) rotate(0deg)}//Transform:none;

**例题3 扑克牌案例（先把所有图片定位于一处）**



img{

width: 100%;

transition:all 1s;

position: absolute;

left:0;

top:0;

/\* transform-origin：变换中心 默认是 图片几何中心\*/

/\*transform-origin ：水平坐标 垂直坐标 \*/

/\*transform-origin: 50px 50px;\*/

**transform-origin: center bottom;**

box-shadow: 0 0 3px 0 #666;

}

.box:hover img:nth-child(6){

transform:rotate(-10deg);

}

.box:hover img:nth-child(5){

transform:rotate(-20deg);

}

# 就业班第二十五天（CSS3）

## 3D变换

1. 屏幕坐标系
   1. 水平向右X正方向
   2. 竖直向下Y正方向
   3. 水平指向我们Z正方向
2. 透视:perspective

**perspective有两种写法**

**作为一个属性，设置给父元素，作用于所有3D转换的子元素;作为transform属性的一个值，做用于元素自身**

* 1. 设置的是用户的眼睛距离平面的距离
  2. 只是视觉上的呈现，并不是真正的3d
  3. 属性值越大，立体感越小
  4. transform:rotateX(360deg),perspective加在其父盒子上

1. **transform：translateZ(50px)**

**沿Z轴移动得先给父盒子加透视perspective：100px**

1. **backface-visitility：hidden**

**背面不可见** 应用于百度钱包例子

1. 伪元素获取自定义属性

<span data-text="传">传</span>

content:attr(data-text)（自定义属性名）

1. 所有的3d旋转，在正方向看，都是顺时针旋转

7.**transform-style:preserve-3d 给父盒子添加，让子盒子存在真3d效果**

* 1. 指定子元素定位在3d空间中
  2. flat：默认值，该元素所在元素平面内被扁平化



例题1.音乐盒（先把两张图片定位于一处，后面这张沿着X轴转动）

.music-box::after,  
.music-box::before {  
 content: "";  
 display: block;  
 position: absolute;  
 left: 0;  
 top: 0;  
 width: 100%;  
 height: 100%;  
 border-radius: 50%;  
  **transform-origin: bottom;**  
}  
.music-box::before {  
 background-image: url(images/musicb.jpg);  
}//伪元素用得不习惯

例2 百度钱包（backface属性，一个从-180°到0°，另一个0到180°）

.box::after,.box::before{

content:"";

position: absolute;

width: 100%;

height: 100%;

border-radius: 50%;

background-color: pink;

background: url(images/bg.png) no-repeat left top;

transition:all 1s;

backface-visibility: hidden;/\* 让盒子的背面隐藏\*/ }

.box::before{

transform:rotateY(-180deg);}

.box::after{

background-position: right top;}

.box:hover::before{

transform:rotateY(0deg);}

.box:hover::after{

transform:rotateY(180deg);

例3 反转文字

例4 3D导航

例5 立体导航栏

**例6 立方体（preserve-3d+动画）**

.box{ /\* 让子盒子保持3d效果\*/

**transform-style:preserve-3d;**

/\*transform:rotateX(30deg) rotateY(-30deg);\*/

animation:gun 8s linear infinite;

}

.left{ background-color: rgba(255,0,0,0.3);

/\* 变换中心\*/

transform-origin: left;

/\* 变换\*/

transform:rotateY(90deg) translateX(-125px);

}

@keyframes gun {

0%{

transform:rotateX(0deg) rotateY(0deg);}

100%{

transform:rotateX(360deg) rotateY(360deg);}

## 动画animate

动画是CSS3中具有颠覆性的特征之一，可通过设置多个节点来精确控制一个或一组动画，常用来实现复杂的动画效果

1. /\*先定义再使用\*/
2. **@keyframes 动画名{**
3. **from{开始状态}//**0%{} 25%{} 50%{}定义多组动画
4. **to{结束状态}**
5. **}**
6. **调用：animation：动画名称 持续时间 执行次数（infinite：无限） 是否反向（alternate） 运动曲线（linear）延迟时间；**
7. animation属性
   1. animation-name
   2. animation-duration：持续时间，
   3. animation-iteration-count：重复次数
      * 1. infinite：无数次
   4. animation-direction：执行方向
      * 1. normal：正常
        2. alternate：反向
   5. animation-delay：延迟时间
   6. animation-fill-mode：动画结束后盒子的状态
      * 1. backwards：保持动起来开始前状态
        2. forwards：保持结束后状态
   7. animation-timing-function：ease-in-out/linear

/steps（number）：把动画分为几步执行

例题 **太阳系（难在定位）**

.earth{

width: 350px;

height: 350px;

position: absolute;

left:50%;

top:50%;

/\* 保证居中\*/

transform:translate(-50%,-50%);

border: 1px solid #ccc;

border-radius: 50%;

animation: rot 8s linear infinite; }

.earth::before{

content:"";

width: 40px;

height: 40px;

position: absolute;

left:0;

top:50%;

transform:translate(-50%,-50%);

background-color: dodgerblue;

border-radius: 50%;

@keyframes rot {

0%{

transform:translate(-50%,-50%) rotate(0deg);}

100%{

transform:translate(-50%,-50%) rotate(360deg);}

}

**案例：大海；无缝滚动；钟表；全屏切换；游动的鲨鱼**

## 多列布局

1. column-count：number

number是几就分几列，**注意兼容性，记得加前缀**

1. column-rule：1px dashed red
   1. 分割线
2. column-gap：30px；
   1. 列间间距
3. column-width：设置分栏宽度
4. column-span:all 设置跨列

# 就业班第二十六天（CSS3）

## 伸缩布局flex

CSS3在布局方面做了非常大的改进，使得我们对块级元素的布局排列变得十分灵活，适应性非常强，其强大的伸缩性，在响应式开中可以发挥极大的作用。

1. 安卓手机移动端基本都是webkit内核
2. **display：flex**
   1. 侧轴和主轴垂直
   2. 默认行为:flex 子项等高。
   3. flex 子项撑满 flex 容器高度。
   4. flex 容器中的 flex 子项呈单行排布，不换行。
   5. **flex 容器的布局方向是从左往右**(对于阅读方向是从左往右的语言，如中文)
3. **设置主轴方向**
   1. flex-direction：row（默认属性）

row：从左往右

* 1. flex-direction：row-reverse

row-reverse从右向左排列

* 1. flex-direction：column

竖直方向，从上往下

* 1. flex-direction：column-reverse

竖直反向，从下往上

1. **主轴的对齐方式**
   1. justify-content：flex-start；

从主轴开始的方向对齐

* 1. justify-content：flex-end

从主轴结束的方向对齐

* 1. justify-content：center

居中对齐（水平/竖直）

* 1. justify-content：space-around

平分父盒子空间

* 1. justify-content：space-between

两端对齐，中间平分

1. **侧轴对齐方式**
   1. align-items：flex-start

从侧轴开始的方向对齐

* 1. align-items：flex-end

从侧轴结束的方向对齐

* 1. align-items：center

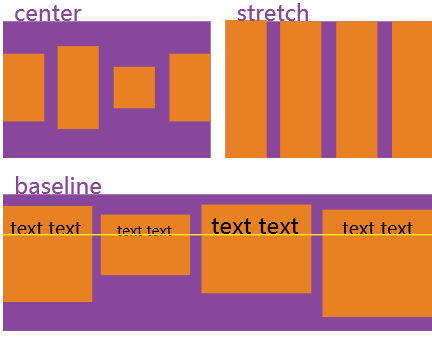
在侧轴方向上居中

* 1. align-items：baseline基线对齐

**如弹性盒子元素的行内轴与侧轴为同一条**，则该值与'flex-start'等效

* 1. align-items：stretch
     + 1. 拉伸，和父盒子高度一样
       2. **有高度不会被拉伸**

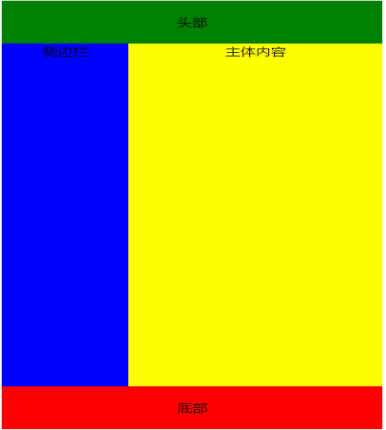
**align-items 各个值的效果示意图,默认情况 flex-direction: row**



1. 伸缩比例
   1. flex：number
   2. flex不设置值就不参与平分

例题1：伸缩式布局

设置主轴方向：竖直方向，先垂直排列



.layout{

width: 800px;

height: 1000px;

border: 1px solid #000;

margin:0 auto;

display: flex;

/\* 主轴方向垂直\*/

flex-direction: column;

}

header{

background-color: pink;

flex:1;

}

main{

background-color: green;

flex:7;

display: flex;

}

.right{

flex: 1;

background-color: red;

}

.left{

flex: 4;

background-color: yellow;

}

footer{

background-color: #000;

flex:1; }

**例题2 携程豪华酒店**



## WEB字体

1. 可以使用自己指定的特殊字体，无需考虑用户是否安装
2. 支持程度较好
3. .eot字体是IE专用字体
4. 使用web字体
   1. 声明字体
      1. **@font-face**{  
         font-family："webfont"; /\*定义的字体名字\*/   
         src:url("字体路径") format();/\*format（）函数给浏览器提示是该字体文件是何种类型\*/  
          }  
          .web-font{  
          font-family:'webfont'; /\*定义字体名字\*/  
          }
5. 伪元素添加图标字体 (给那个标签添加icon类名那个标签就会有这个图标)
   1. .icon::before{  
       content: '\e655';  
       font-family: iconfont;  
      }
   2. **&#xe67c; 在伪元素添加的时候content=“\e67c”**

**360 案例:**

监听滚轮 var i=0;

document.onmousewheel=function(){

console.log(i++);}

//需要处理兼容 （我们是靠监听滚轮的时间来处理）

//我们需要使用到插件.(滚屏插件) 基于jQuery 的一个插件.

# 就业班第二十七天（HTML5）

## 多媒体

**方法**：load() 加载、play() 播放、pause() 暂停

document.querySelector('button:nth-child(4)').onclick=function(){document.querySelector('video').pause();}

**属性**：currentTime 视频播放的当前进度、

duration:视频的总时间,paused:视频播放的状态

**事件**：oncanplay: 事件在用户可以开始播放视频/音频（audio/video）时触发。

ontimeupdate:通过该事件来报告当前的播放进度.

onended:播放完时触发

全屏：video.webkitRequestFullScreen()

1. **<figure></figure>:多媒体标签**
2. **<figcaption> </figcaption>:多媒体标题**
3. video：行内块显示
   1. text-align：center；对行内块适用
4. 案例
   1. 基本样式，进度条、播放按钮、当前时间/总时间、全屏
   2. 设置播放暂停事件，修改按钮
   3. 获取总时间。当前时间，同步时间
   4. 进度条同步播放，当前时间/总时间
   5. 全屏事件
   6. 点击位置播放点击出视频
   7. （缓冲视频进度条）

## 拖拽

1. 默认图片和链接可以拖拽
2. 设置draggable=“true”属性，元素可拖拽
3. 拖拽元素事件

ondrag 应用于拖拽元素，整个拖拽过程都会调用

ondragstart 应用于拖拽元素，当拖拽开始时调用

ondragleave 应用于拖拽元素，当鼠标离开拖拽元素时调用

ondragend 应用于拖拽元素，当拖拽结束时调用目标元素事件

4.目标元素

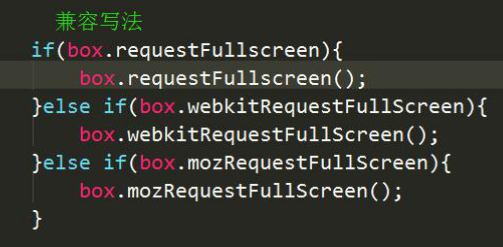
ondragenter 应用于目标元素，当拖拽元素进入时调用

ondragover 应用于目标元素，当停留在目标元素上时调用

ondrop 应用于目标元素，当在目标元素上松开鼠标时调用

ondragleave 应用于目标元素，当鼠标离开目标元素时调用event.preventDefault();阻止默认拖拽事件，允许将元素放置到其他元素中

## 全屏



## web储存

1. document.cookie:传统储存方式，4k大小
   1. 解析复杂
   2. 存储数据量小
2. Storage存储
   1. window.sessionStorage（会话存储，容量5M）
      * 1. 生命周期为关闭浏览器窗口
        2. 在同一个窗口下数据可以共享
   2. window.localStorage（本地存储，容量20M）
      * 1. 永久生效，除非手动删除,清理垃圾，存储在硬盘上
        2. 可以多窗口共享
   3. 设置、读取方便
   4. 只能存储字符串，可以将对象json.stringify（），编码后储存
   5. 可能存储在浏览器内存和硬盘上
3. 方法：
   1. window.sessionStorage关闭窗口数据销毁
      * 1. window.sessionStorage.setItem（key，walue）设置储存内容

再次给同一个key赋值，会覆盖之前的数据。

* + - 1. window.sessionStorage.getItem（key）获取存储数据
      2. window.sessionStorage.removeItem（key）删除key
      3. window.sessionStorage.clear（）清除所有数据
  1. window.loaclStorage数据存储在硬盘上，永久存储
     + 1. window.localStorage.setItem（key，walue）设置储存内容

再次给同一个key赋值，会覆盖之前的数据。

* + - 1. window. localStorage.getItem（key）获取存储数据
      2. window. localStorage.removeItem（key）删除key
      3. window. localStorage.clear（）清除所有数据

1. 打开页自动获取数据：window.onload = function（）｛获取数据｝

## 地理定位

1. 获取用户地理信息
2. 可以开发基于用户位置的互联网应用
3. 基于位置服务**L**ocation **B**ase **S**ervice
4. 定位方式
   1. IP可以一般精确到城市，运算代价大
   2. GPS 非常精确
   3. BeiDoui（BDS）
   4. wifi信号：室内
   5. 手机信号
   6. 用户自定义数据（用户自己输入自己位置）
5. 浏览器会自动以最优化方式获取用户地理信息
6. navigator.geolocation.getCurrentPosition(successCallback,errorCallback)
7. var wd = **position**.coords.**latitude**; //纬度  
   var js = **position**.coords.**longitude**;//经度

## 网络状态

1. window.addEventListener（‘online’，function（）｛｝）用户网络链接时被调用（on和其他事件的on不同，这里代表打开）
2. window.addEventListener（‘offline’，function（）｛｝）用户网络断开的时候调用
   1. 拔掉网线或者禁用以太网
3. window.addEventListener（‘事件（不带on）’）

## 应用程序缓存

1. 离线应用
2. 只需创建（后缀.appcache）一个CACHE MANIFEST文件
   1. CACHE MANIFEST  
        
      #注释开头用#  
        
      CACHE:  
      #要缓存的文件  
        
      NETWORK  
      #指定必须联网才能缓存的文件  
        
      FALLBACK  
      #当前页面无法访问时，返回的一个页面
3. 调用
   1. <html manifest="hc.appcache">

# 就业班第二十八天（AJAX）

## 知识点

1. c/s架构：客户端/服务器
   1. 优点：原生应用，显示效果更炫酷
   2. 缺点:重量级，必须安装客户端
2. b/s架构：浏览器/服务器
   1. 优点：轻量级
   2. 缺点：无法实现非常炫酷的效果
3. 互联网访问你的网站
   1. 服务器
   2. 域名
4. 局域网访问

## PHP基础

1. <?php ?>写在标签内
2. $num：创建一个变量
3. echo：输出

按照utf-8编码：**header**(**"content-type:text/html;charset=utf-8"**);

1. 不用和中文命名
2. php中拼接字符串的方式是点 . 而不是加号
3. php中单双引号不同
   1. 双引号会将变量的值放进去
   2. 单引号就把变量名当作字符串输出
   3. 单双引号混用
4. get、post提交数据
   1. get：提交的数据会显示在url中
   2. post：提交的数据不会显示在url中
5. <form action="get.php" method="GET">
6. $\_GET[‘key’]：超全局对象
7. 都是通过input的name属性获取数据

# 就业班第二十九天（AJAX）

## 知识点

1. 关系数组
   1. array(“key”=>”value”,…)
2. get没有数据大小的限制
3. post上传大小没有限制
4. 二维数组
   1. $oneArr = array(array(“”),array(“”)…);
   2. 输出数组：print\_r(数组名)
5. 不指定上传方式，默认是get
6. 异步对象
   1. 创建异步对象
   2. 设置请求的url等参数
   3. 发送请求
   4. 注册事件
   5. 在注册的事件中获取返回的内容并修改页面显示的内容
7. 布尔类型不能直接用echo输出

## 文件上传

1. enctype = multipart/form-data
2. sleep（时间（单位秒））；延时

## 作业

[www.md5.com](http://www.md5.com)

# 就业班第三十天（AJAX）

## 知识点

1. 背景大小：/100px 100px
   1. /contain：填满
   2. /cover：
2. file\_get\_contents(“路径”)：读取文件

## ajax发送post请求

1. post请求
   1. 创建异步对象
   2. 使用post请求
      * 1. ajax.open（‘post’，‘ajax\_post.php’）
   3. 使用post提交数据要添加http头
      * 1. ajax.setRequestHeader（‘content-type’，‘application/x-www-form-urlencoded’）
   4. 发送请求
      * 1. ajax.send（’ xxx.php?name=jack&psd=123’）；
   5. 注册事件
      * 1. ajax.onreadystatechange = functtion（）｛

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

}

｝

//创建异步对象  
var ajax = XMLHttpRequest();  
//发送报文  
ajax.open("get","demo.php");  
//相应报文  
ajax.send();  
//注册事件  
ajax.onreadystatechange = function(){  
 if (ajax.**readyState** == 4 && ajax.**status** == 200) {  
 **console**.log(ajax.**responseText**);  
 }  
}

1. 不考虑提交文件，post和get作用基本一致，除了写法不同

## XML

1. 标签名字随意写
2. <?xml version=”1.0” encoding=”UTF-8”?>
   1. 必须写第一行
3. ajax获取xml
   1. 通过ajax.resonseXML获取返回的xml值
4. 解析复杂
5. 双标签

## json

1. 是一种格式
2. 是字符串
3. 将json字符串转换为该语言对象的语法
4. 创建一个json对象
   1. var json = '{"name":"jack","sex":"nan"}'
5. 转换为对象
   1. var jsonObj = **JSON**.parse(**ajax**.**responseText**);
6. json路径是相对于html的路径
7. 987

## ajax五步封装函数

// 将 get 跟post 封装到一起  
/\*  
 参数1:url  
 参数2:数据  
 参数3:请求的方法  
 参数4:数据成功获取以后 调用的方法  
 \*/  
function ajax\_tool(url,data,method,success) {  
 // 异步对象  
 var ajax = new XMLHttpRequest();  
  
 // get 跟post 需要分别写不同的代码  
 if (method=='get') {  
 // get请求  
 if (data) {  
 // 如果有值  
 url+='?';  
 url+=data;  
 }else{  
 }  
 // 设置 方法 以及 url  
 ajax.open(method,url);  
  
 // send即可  
 ajax.send();  
 }else{  
 // post请求  
 // post请求 url 是不需要改变  
 ajax.open(method,url);  
  
 // 需要设置请求报文  
 ajax.setRequestHeader("Content-type","application/x-www-form-urlencoded");  
  
 // 判断data send发送数据  
 if (data) {  
 // 如果有值 从send发送  
 ajax.send(data);  
 }else{  
 // 木有值 直接发送即可  
 ajax.send();  
 }  
 }  
  
 // 注册事件  
 ajax.onreadystatechange = function () {  
 // 在事件中 获取数据 并修改界面显示  
 if (ajax.**readyState**==4&&ajax.**status**==200) {  
 // console.log(ajax.responseText);  
  
 // 将 数据 让 外面可以使用  
 // return ajax.responseText;  
  
 // 当 onreadystatechange 调用时 说明 数据回来了  
 // ajax.responseText;  
  
 // 如果说 外面可以传入一个 function 作为参数 success  
 success(ajax.**responseText**);  
 }  
 }

# 就业班第三十一天（AJAX）

## 模版引擎

1. 正则子表达式
2. <script type = ‘text/tmplate’></script>
3. script的type不写或者写成text/javascript会当作js解析，写成其他的不会显示在页面中

## 作业

落网

github.com

# 就业班第三十二天（AJAX）

## 知识点

1. array\_rand(哪个arr,取几个1)随机的index
2. $arr = json\_decode($str) 将字符串转换为php的数组
   1. json\_encode将object转换为json格式字符串-
3. json格式是字符串
4. print\_t（数组）；php返回一个数组
5. PHP获取数组长度：count（数组）；
6. .blur：失去焦点 focus（）获取焦点
7. in\_array(要查询的数组,查询内容)；查询
8. PHP定义数组
   1. $userArr = array(‘值1’， ‘值2’，);
9. fadeIn/fadeOut（时间；淡入淡出）
10. delay（延时）
11. 包装对象

var obj = ｛

items：json数组

｝

1. 最外层是大括号是对象，是中括号是数组
2. jq中有 数组.each（function（）｛｝）循环方法

## jQuery操作AJAX

1. $.get（url，{可直接写对象name：‘jack’，age：‘18’}，回调函数，参数4（从服务器获取的数据类型，可不写））
   1. $.get(url,date,callback,type)
   2. 如果参数4写json内部会JSON.parse（）自动转换
   3. 不要擅自更换参数位置
2. $.post(url,date,function(date){},type)
3. 3.$\_ajax({
4. url:‘’，
5. success：function（date）{},
6. type:‘post/get’，
7. datatype：‘数据类型（json）’，
8. data：｛
9. ‘name’：‘’，
10. ‘kill’：‘’
11. ｝，
12. beforeSend：function（）｛
13. console.log（‘发送之前调用’）；
14. return false；//阻止这次请求
15. 验证用户数据是否填了
16. ｝，
17. error：function（）｛
18. console。log（‘请求失败后调用’）
19. ｝
20. })

## 瀑布流

1. 实现技术
   1. ajax
   2. jq
      * 1. ajax请求
   3. 模版引擎
      * 1. 渲染页面
2. 为jQuery添加插件
   1. $.fn.extenf();
   2. 注册完毕后，使用$(‘调用者’).函数名 调用

$.fn.extend({

函数名：function(){

函数体；

}

var $\_this = this;

//为了可以链式编程

return $\_this;

})

* 1. $.extenf();
  2. 注册完毕后实用$.xxx

1. 987

## 同源以及跨域

1. 同源：域名、协议、端口完全相同
2. iframe嵌入另外一个页面

## 作业

1. www.w3cschool.cn

# 就业班第三十三天（移动web）

## 知识点

1. 对象直接点出来，数组要加索引值
2. 安卓内置谷歌chrome浏览器
3. 苹果内置safari
4. 移动端的浏览器内核只需要考虑webkit
5. lorem 1000 自动出文字信息

## viewport属性

<meta

name="viewport"

content="

width=device-width, //设备的宽度

user-scalable=no, //设置用户缩放，一般禁止缩放no/false

initial-scale=1.0, //初始的默认缩放比例，一般默认是指1.0

maximum-scale=1.0,//最大的缩放值

minimum-scale=1.0"//最小的缩放值

/>

1. name = ‘viewport’；告诉浏览器，这个meta标签设置的是viewport属性
2. 不设置高度让内容撑开

## 移动web样式

1. -webkit-tab-highlight-color：transparent；鼠标点击高亮
2. 移动端设置所有的盒子box-sizing：border-box；保证盒子大小不变
3. input默认样式清楚
   1. -webkit-appearance：none；
4. 宽度使用百分百，高度使用像素单位
5. sans-serif移动端默认字体

## 作业

1. open.baidu.com
2. lbsyun.baidu.com
3. developer.baidu.com

# 就业班第三十四天（移动web）

## 知识点

1. ：nth-child(2n):选择位置为2的倍数的元素
2. li标签不可以用ling-height居中，要用父盒子的padding挤
3. h3:first-of-type:找到第一个这种类型的标签
4. h3:last-of-type:找到最后一个这种类型的标签
5. :nth-of-type(n):找到第几个该类型的标签

## LESS

1. @变量名:属性值;定义变量
2. 如果less文件在名为less文件夹里,考拉编译后会自动生成一个css文件夹保存.css文件,否则会在一个文件夹生成.css文件
3. less可以更好的维护和编写css
4. 混合:可以把一个定义好的class完全引入到另一个class中
   1. .类名(@color:默认值){  
       //样式  
       background-color: @color;  
       border: 1px solid @color;  
       color:@color;  
      }  
      选择器{  
       .类名(@color值);  
      }
5. 嵌套
   1. div{  
       width: 100%;  
       ul{  
       width:100%;  
       li{  
       width: 100%;  
       float: left;  
       a{  
       color:red;  
       }  
       .cla{  
       color:green;  
       }  
       }  
       }  
       .list{  
       width:50%;  
       }  
      }
   2. 相当于
   3. div {  
       width: 100%;  
      }  
      div ul {  
       width: 100%;  
      }  
      div ul li {  
       width: 100%;  
       float: left;  
      }  
      div ul li a {  
       color: red;  
      }  
      div ul li .cla {  
       color: green;  
      }  
      div .list {  
       width: 50%;  
      }
6. 伪元素
   1. 交集选择器要加 &.current
   2. 如果不加,直接写.current会变成并集选择器
   3. 伪元素也要 **&::hover**
7. 1

## 作业

zh.learnlayout.com

# 就业班第三十五天(移动web)

## 知识点

1. opacity:会使子元素也变透明,rgba则不会
2. offsetTop:距离顶部高度
3. offsetHeight:自身的高度
4. onscroll:页面滚动事件
5. windo.document.body.scrollTop;滚动距离
6. .querySelectot/.querySelectorAll(“可以写从css,css3中的选择器”)
7. 移动端一定要加视口属性:meta:vp
8. 时分秒
   1. **hour** = **Math**.floor(totalSecond/3600);  
      **minute** = **Math**.floor((totalSecond%3600)/60);  
      **scon** = totalSecond%3600
9. span.[^icon\_]{}

## 过渡结束事件

ul.addEventListener("webkitTransitionEnd", function () {  
 **console**.log("过渡结束");

}

要加私有化前缀:webkit/o/moz/ms(ie)

## 触摸事件

1. 触摸事件
   1. **touchstart**:开始触摸的时候触发,有触摸点的值
      1. var x = event.touches[***0***].*clientX/Y*;
   2. **touchmove:**手指按下并移动的时候, 有移动的值
      1. var x = event.touches[***0***].*clientX/Y*;
   3. **touchent:**手指抬起来的时候,没有触摸点的值
2. 不能通过dom.ontouchstart的方式绑定
3. 只能通过addEventList(‘touchstart’,function(){})的方式绑定事件
4. 只支持移动设备
5. 封装事件
   1. 左滑,右滑
   2. 捏合
   3. 长按

## 精灵图通用代码用法

[class^="icon\_"]{  
 background: url("../images/sprites.png") no-repeat;  
 background-size: x y ;  
}

## 定时器函数

//定时器函数  
function cutDownTime(){  
 //获取事件源  
 var time = **document**.querySelectorAll('.product:nth-child(1) .title li');  
 var totalSec = 7200;  
 //定义定时器  
 var tim = setInterval(function () {  
 //如果剩余时间为0,停止执行函数  
 if(totalSec<=0){  
 totalSec = 0;  
 return;  
 }  
 totalSec--;  
 //转换时分秒  
 var hour = **Math**.floor(totalSec/3600);  
 var minute = **Math**.floor((totalSec%3600)/60);  
 var sec = totalSec%60;  
  
 //显示到页面中  
 //十位部分显示  
 time[0].**innerHTML** = **Math**.floor(hour/10);  
 //个位部分显示  
 time[1].**innerHTML** = hour%10;  
  
 //十位部分显示  
 time[3].**innerHTML** = **Math**.floor(minute/10);  
 //个位部分显示  
 time[4].**innerHTML** = minute%10;  
  
 //十位部分显示  
 time[6].**innerHTML** = **Math**.floor(sec/10);  
 //个位部分显示  
 time[7].**innerHTML** = sec%10;  
 },1000)  
}

## 排他思想函数

//排他思想函数,干掉所有人,留下我自己  
function exclusive(elementArr){  
 for(var j = 0;j<elementArr.**length**;j++){  
 //循环为每一项绑定点击事件  
 elementArr[j].onclick = function () {  
 //把每一项的类名都去掉  
 for(var i = 0;i<elementArr.**length**;i++){  
 elementArr[i].**className** = '';  
 }  
 //只留下触发当前点击事件的触发者,加上类名  
 this.**className** = 'current';  
 }  
 }  
}

# 就业班第三十六天(移动web)

## 知识点

1. 定时器的时间和过渡的时间相等或者小于过渡的时间,会出问题
   1. 相等:会一直在滑动状态,触发不了过渡结束事件
   2. 小于过渡事件,会跳, 触发不了过渡结束事件
2. 通用代码封装成函数
3. 确定不了的变量封装成函数
4. 代码中不要写死代码
5. click事件在移动端有延迟
6. 自己封装一个tap的方法,手指点击
7. 最牛逼的程序员看颈椎病康复大全
8. Date.now();,获取当前的系统时间,返回的的是毫秒值
9. console.log():十分耗性能,只会出现在测试的时候,项目上线的时候会删除

## 响应式布局

1. 基本概念
   1. 针对u所有设备
   2. 开发一次即可
   3. 会感染疟疾设备的屏幕尺寸改变而改变布局
2. 缺点
   1. 代码较多
   2. 维护复杂
3. 简单响应式布局
   1. 媒体查询

**@media** *screen and* (*min-width*:500*px*){

body{  
 *background-color*: *#f00*;  
 }  
}  
**@media** *screen and* (*max-width*:700*px*){  
 body{  
 *background-color*: *#ff0*;  
 }  
}

## Bootstrap框架

1. 栅格系统
   1. container
      1. row
   2. 默认一行为12行
2. Bootstrap

# 就业班第三十七天(移动web)

## 知识点

1. 响应式布局依托于媒体查询**@media**
2. 伪元素一个冒号和两个冒号都可以
   1. 一个是老版本的写法
   2. 两个是新版本的写法
3. 父元素设置text-align:center;子元素居中显示
4. **lorem**1000:自动生成文本
5. 不要修改bootstrap的源文件
   1. 拷贝bootstrap的原生的文件
   2. 修改拷贝的原生样式的类名
   3. 修改css
   4. 实用组件的地方替换自己的类名
   5. 不会覆盖元素的css
6. bootatrap的a便签有一个默认样式:
   1. 有内边距
   2. display:block;
7. a标签不支持嵌套
8. 移动web

## 栅格系统

1. bootstrap的栅格系统必须写在**.container**中
2. .container-fluid;自适应100%宽度

## :nth-child()选择器

li:*nth-child*(2*n*){}//选择序号为2的倍数的标签

li:*nth-child*(*n*+2){}//选择序号大于2标签(包括2)

# 就业班第三十八天(移动web)

## 知识点

1. bootstrap层次结构复杂,修改样式的时候拷贝css源代码.然后修改根选择器类名
2. hidden-lg\md\ms\xs;在相应的图片尺寸内隐藏
3. bootstrap的js插件里的**标签页插件**里的点击切换和层次关系无关,给标签和内容各加一个父盒子不影响效果
   1. 标签和内容是通过锚点来关联的;href=’#name’;id=’name’
4. 先解析和后解析会造成页面布局不同
5. 给div加box-sizing:border-box;之后,如果内容大小超过固定大小之后还是会撑大父盒子的
6. div:nth-of-type(2n);从第一个div开始计数
   1. 2n:选择索引为2的倍数的标签
   2. n+2:选择索引值大于等于2的标签
   3. 2n+1:从0开始计算(2\*0+1=0),索引为2n+1的标签,会选择到第一个标签
   4. **2n-1:不可以**
7. 媒体查询
   1. **@media** screen and (max-width:1200px){  
       li{  
       width:25px;  
       }  
      }
   2. 小于1200px的时候执行该样式
8. 列表居中方式
   1. 给ul设置display:inline-block;
   2. 给ul的父盒子设置:text-align:center;
9. web

## zepto.js

1. 轻量级,jquery太大
2. 根据需求导入需要用的模块
3. js文件会自动缓存,不修改就不用再次请求
4. 必须先导入event.js事件处理模块
   1. 用$(‘.class’).on(‘click’,function(){})
5. event:事件处理
6. fx:动画
7. touch:触摸事件(单击,滑动,长按...)
8. -支持链式编程
9. jquery改变标签内的文本用.text(‘文本内容’);
10. $(‘div/#box/.box’);选择器
11. $(‘<p>新元素</p>’);创建一个新元素
12. $(‘<p/ >’,{text:’内容’,id:”p1”,css:{color:’red’}})
    1. => <p id=greeting style="color:darkblue">Hello</p>
    2. 创建带有属性的元素

## 作业

1. rem
2. www.gitbook.com
3. web

# 就业班第三十九天(JS高级)

## 知识点

1. 在js里,变量.log+tab ✍ console.log(变量);
2. name是一个window的对象 ✍ window.name
3. object.toString 返回[object object]
4. 函数名优先级大于变量

## JS基础知识

### js包含Bom/dom/ECMAscript

* 1. ECMA:规定了语法规范
  2. Bom:浏览器对象(browser object model)
  3. DOM:文档对象(document object model) :给我们提供了一套完整的操作页面元素的api

### JS基本类型

* 1. string\boolean\number

### 复杂数据类型(引用类型):

* + - 1. Array Date Obeject RegExp String Boolean Number
      2. 核心类型Object function
      3. 获取数据类型typeof()

(typeof是个关键字)

(返回值是字符串)

typeof 变量名;

* + - 1. 首字母大写的是函数
      2. 空类型
      3. null:

变量不可能为null值,除非手动设置

解除对象的占用的时候给对象赋值null

* + - 1. undefined:定义之后未赋值

### ===和==的区别

===:类型和数值必须全等

==:数值相等即可

### in关键字

* + - 1. 用法一

for…in:变量对象的键

for(var 变量;in 对象名)

对象的键名为字符串类型

* + - 1. 用法二

判断属性(数值就是索引值)是否存在对象中

‘关键字’ in 对象名

判断关键字是否在关键字

返回值是boolean类型

* + - 1. 用法三

判断数组是否存在某个键

in关键字判断数组的时候是判断索引是否存在,而不是值

如何判断数组中是否存在指定的值

indexOf

ar.indexOf(要查找的值)

找到返回指定元素的索引值,

没有找到返回-1

* + - * 1. *console*.log(*3* in arr);  
           /\*\*  
            \* arr.indexOf('b')判断arr数组中是否含有'b'这个值  
            \* 3 in arr 判断arr数组中是否有3这个索引值  
            \*/  
           *console*.log(arr.indexOf('b'));

1. 关系运算符:> < >= <= =

### 创建对象使用new对象

* 1. var obj = new Date();
  2. GMT:格林尼治标准时间
  3. UTC:世界协调时间

### 类型

* + - 1. 值类型
         1. 储存的就是数据本身的变量 就是值类型数据
      2. 引用类型Object

储存的是数据在内存中的地址,数据类型在内存中单独储存 就是引用类型数据

* + - 1. 函数调用时,默认将实参赋值给形参
      2. 形参没有具体的数值
      3. 看到对象字面量和{大括号}才会创建新对象
      4. 值类型做函数的参数

函数内部的变量,也就是形参和实参只是简单的赋值操作,两个数据独立储存

在函数内部对形参进行修改,不会影响外面的值

* + - 1. 引用类型做函数的参数

是把实参存储的地址赋值给了形参,在函数内部,形参同样指向该对象

所以在函数内部对该变量进行修改,会修改外部的值

如果在函数内创建对象,修改一个值不会影响另外一个.两个独立存在

* + - * 1. /\*\*  
            \* 值类型和引用类型  
            \* 值类型:  
            \* 储存的是变量本身,可直接修改  
            \* 引用类型  
            \* 变量储存的是对象的地址,把地址赋值给另外一个变量后,另外一个变量修改,可直接影响本来的值  
            \* 对象作为实参传递的时候,传递的是一个地址值  
            \*/  
           var obj = {  
            *num*:*5*};  
           *console*.log(obj); //输出num的值是5  
           function num(number) {  
            number.*num* = *100*;  
            number = {  
            *num*:*15555* //函数内创建的对象和函数外的互不影响  
            }  
            return number;  
           }  
           num(obj);  
           *console*.log(obj);//输出num的值是*100*
      1. 引用类型中除了function都是object类型

### 对象的动态特效

* + - 1. 对象的动态特性是指,在对象创建出来之后,为对象添加新的属性或者方法

obj.satHello = function(){方法;}

obj.xxx = yyy;

如果xxx存在,是进行修改

如果xxx不存在,是添加属性或者方法

* + - 1. obj.[”属性名”]:可以用这种方法取值,也可以新增属性或者方法
      2. obj[0] = ‘oooo’;可以添加成功,因为这里会进行一个隐式转换

[ ]内的值会被隐式为转换字符串类型

把对象{}转换层字符串object object

* + - 1. 动态添加属性/方法

obj.新属性 = 新值

obj[新属性] = 新值

* + - * 1. /\*\*  
            \* 对象的动态特性  
            \* 在创建一个对象之后,可以给对象追加属性和方法  
            \*/  
           var prople = {  
            *name*:'jack',  
            *age*:*18*}  
           *console*.log(prople); //输出 name 和 age 两个属性  
           prople.*sex* = '男'; //追加属性  
           prople.sayHallo = function () { //追加方法  
            *console*.log("我叫jack");  
           }  
           *console*.log(prople); //输出追加后的对象,name age sex 三个属性和sayHallo这个方法  
           prople.sayHallo(); //输出 我叫jack  
             
           prople['height'] = *180*;  
           *console*.log(prople);//输出追加后的对象,name age sex height四个属性和sayHallo这个方法

### 逻辑中断(短路运算)

* + - * 1. true || false //返回true  
           false || true/false //返回true/false

1. 表达式1||表达式二
2. 1为真 返回1
3. 1为假 返回2
4. true && false //返回false 第一个真返回第二个  
   false && true //返回false 第一个假返回第一个
5. 表达式1 && 表达式2
6. 如果1为真,判断2
7. 真 返回2
8. 假 返回2
9. 如果1为假,直接返回1

### delete运算符的作用

1. 用来删除对象的属性,还有删除未使用var声明的变量
2. 不能删除实用var声明的变量
3. 删除一个不存在的属性,也会删除成功
   1. delete 对象名.对象属性名;
4. 返回值是删除是否成功false/true
5. 如果删除的属性存在与原型中,返回true,但是并未删除

### 循环语句

1. for
   * 1. 跳出循环:continue
2. while
3. do…while
4. for…in

### 分支语句

* 1. if…else
  2. swith…case

### 函数

* 1. 函数声明
     1. function name(){}
  2. 函数表达式
     + 1. 在使用函数表达式声明函数的时候,function后面可直接跟函数名(name)
       2. var fn = function name(){}
       3. 但是这个函数名(name),直线在函数内部使用,外部无法访问
       4. var fn = new *Function*(“”)
     1. //解析会被提前  
        function name(){  
         //函数体  
        }  
          
        //解析不会被提前,使用必须在定义之后  
        var fun0 = function () {  
         //函数体  
        }  
          
        //解析不会被提前,使用必须在定义之后  
        //创建一个函数对象  
        var fun = new *Function*("");//函数体放 引号中间  
          
        //创建一个匿名函数  
        var a= (function (){/\*函数体\*/})(); //创建完毕直接调用

### Dom操作

* 1. 增删改查
     + 1. 增
          1. document.createElement
          2. oppendChild
       2. 删
          1. removeChild
       3. 改
          1. style
          2. id
          3. className
          4. innerHTML
          5. innerText
          6. ...
       4. 查
          1. getElementById
          2. getElementsByTagName
          3. getElementsByClassName

### 异常处理

* + - 1. 捕获语句:try catch finally
      2. 语法
         1. try{  
             //可能出现的代码  
             //无法捕获语法错误  
            }  
            catch (e){  
             //出现错误后执行的代码  
            }  
            finally{  
             //不管出不出现错误,这里的代码都会执行  
             //node.js  
             //一般做释放资源的操作  
            }
      3. 异常最大是特征:出现异常之后后面的代码将不会在执行
      4. 手动抛出异常

使用throw关键字

错误代码(编号),错误信息

throw可以在外部定义

* + - 1. /\*异常处理\*/  
         try{  
          a();  
         }catch(e){  
          //console.log(e);  
          // e 是错误信息  
          throw('a()是一个未定义的函数');  
          //throw(e)相当于不写这个异常处理函数  
          //throw()手动抛出异常;括号内写抛出的自定义信息  
         }  
         finally{  
          *console*.log("这里的一定会执行");  
         }

### 面向对象

* + - 1. 面向过程思维方式:把解决问题发关注点放在每一个步骤上
      2. 面向对象思维方式:把解决问题的关键点放在解决问题需要的一系列的对象上
      3. 面向对象是一种思维方式
      4. 万物皆对象(具体到每一个事物)
      5. 对象的特征
      6. 用来描述对象的相关信息
      7. **对应JS的属性**
      8. 对象的行为
         1. 对象可以干的事情
         2. **对应JS的方法**
      9. **在JS中键值对的组合就是对象**
      10. 名词就是对象
      11. createElement(‘div’):创建一个div
      12. 面向对象是对面向过程的封装
          1. 封装的意义就是复用
      13. 编写代码规则DRY
          1. don’t repeat yourself
      14. 使用函数封装带来的问题
      15. 变量名污染
      16. function createEle(box) {  
           var ele = *document*.createElement('div');  
           ele.*style*.*width* = '100px';  
           ele.*style*.*height* = '100px';  
           ele.*style*.*backgroundColor* = '#f00';  
           ele.*style*.*margin* = '0 auto';  
           box.appendChild(ele);  
          }
      17. 使用对象进行封装 对象封装
      18. var obj = {  
           *jQy*: {  
           Id: function (ele) {  
           return *document*.getElementById(ele);  
           },  
           Class: function (ele) {  
           return *document*.getElementsByClassName(ele)[*0*];  
           }  
           }  
          }
      19. jQuery中甭创建对象直接添加即可
      20. $('body').append("<div style='width: 100px;height: 100px;background-color: #f00;margin: 0 auto;'></div>")
      21. 面向对象

## 作业

# 就业班第四十天(JS高级)

## 面向对象的三大特性

1. 继承
   1. js中的继承是指:
      * 1. 一个对象没有一些方法和属性,而另一个对象有
        2. 把另一个个对象的属性和方法,拿过来自己用,这就是继承
   2. 混入式继承 for … in
      * 1. 父类的属性和方法在所有之类
2. 封装
   1. 把一些属性和方法装到一个对象里
3. 多态
   1. JS中没有相应的体现
   2. 在强类型语言比较常见
   3. 实用父类的变量接受子类的对象
   4. 父类的属性方法供所有的子类共享

## 创建对象的方式

1. 创建对象
   1. 四种方法
   2. 字面量创建
      * 1. 用一个创建一个,不能服用,照成代码冗余,资源浪费
   3. 内置构造函数创建对象
      * 1. var p = new Object();
        2. var arr = new Array();
        3. 创建出来的对象都是空对象,要手动添加属性,照成代码成代码重复
   4. 封装简单的工厂函数(不推荐使用)

function createObj(name,age){

var obj = {

boj : name;

obj:age

sayHello : function(){

console.log(‘’);

}

};

return obj;

}

* 1. 自定义构造函数

function Porple(data1,data2){

this.xx = data;

}

* + - 1. 概念:用来实例化对象,并且给初始化对象赋值
      2. 构造函数名首字母大写
      3. 构造函数一般和new关键字一起使用
      4. 构造函数返回值默认为新创建好的对象,如果手动设置返回值
         1. 返回值类型如果不是Object类型的都还返回创建的对象
         2. 如果是Object类型的就返回object
      5. 1

1. 1
   1. 对象字面量{},创建对象
   2. 自定义构造函数,创建对象
      * 1. 构造函数也是函数,通常用来初始化对象
        2. new用来创建对象
        3. 构造函数用来初始化函数
        4. 构造函数名要大写,
   3. 构造函数的执行过程
      * 1. 使用new关键字创建对象
        2. 调用构造函数把新创建出来的对象赋值给构造函数的this关键字
        3. 在构造函数内实用this为新创建出来的对象添加成员
        4. 默认返回新创建的这个对象
        5. 如果return一个空值,或者return一个基本类型数据仍旧是返回新创建的对象
           1. 返回undefined和null都是返回新对象
        6. 如果return的是object类型,将返回return后面的对象
        7. 函数名加括号就是调用函数
        8. 不写括号是把函数整体代码赋值给另一个变量
        9. 如果调用函数的时候不给构造函数实参,构造函数不会创建新对象,里面的this将会指向window,添加的属性也会添加给window
        10. 如果构造函数没有参数,可以不写括号调用
2. 对象成员:属性和方法
3. 创建对象
4. webstrom 先该系统时间,推迟到之前好多年
   1. 再去安装

## 原型

1. 在构造函数创建出来的时候,系统会默认帮构造函数创建并关联一个神秘函数
2. 原型中的属性和方法可以被实用该构造函数创建出来的对象使用
3. 原型默认是一个空的对象
4. 访问构造函数的原型
   1. 构造函数.prototype
5. **构造函数.prototype.新属性 = 属性值**
6. 自己和原型中都有的属性,优先用自己的
7. 创建一个对象叫做实例化对象
   1. 通过构造函数创建对象创建对象的过程
8. 实例
   1. 通过构造函数实例化出来的对象就是该构造函数的一个实例
9. 原型的实用方法:
   1. 利用对象的动态特性给原型对象添加属性
10. 直接替换原型对象

Porple.prototype = {

name : ‘Jack’,

}

* 1. 直接替换原型对象出现的问题
     + 1. 在**替换原型之前创建的对象的原型** 和在**替换原型之后创建的对象的原型** 不是同一个
       2. 替换不是修改
  2. 点语法进行属性赋值的时候,不会去原型查找
  3. **使用点语法赋值的时候,如果对象中不存在该属性,就会对该对象新增属性,不会去原型中查找**
  4. 如果原型总分的属性是引用类型的属性,那么所有的对象共享该属性,并且一个对象修改了该引用类型中的成员,所有对象都会被修改
  5. 一般不会把属性放置到原型中

1. 通过构造函数访问原型
   1. 构造函数.prototype
2. 通过对象访问原型
   1. 对象.\_­\_proto\_\_  **//不推荐使用**
3. constructor:原型对象的属性

# 就业班第是四十一天(JS高级)

## 继承的实现方式

1. 概念L当前对象没有的属性和方法,别人有,拿来给自己用,就是继承

### 混入式继承

* + 1. var a = {};  
       var obj = {  
        *name* : 'jack',  
        *age* : *18*,  
        sayHello : function () {  
        *console*.log('hello');  
        }  
       }  
       //混入式继承,a中将继承obj的所有属性  
       for(var k in obj){  
        a[k] = obj[k];  
       }

### 原型继承

1. 利用原型中的成员可以被其相关对象共享这一特性,可以实现继承
2. 实现步骤
   1. 给原型对象添加新成员(**通过对象的动态特性**),不是严格意义上的继承 ,,,,**实例对象继承了原型**
   2. **直接替换原型对象**
      * 1. 构造函数.prototypr = 新对象
        2. 实例对象继承了原型(新对象)
        3. 如果对象原本有属性和方法,使用替换的方法会覆盖原有的属性和方法
   3. **利用混入的方式给原型对象添加成员**
      * 1. 混入,遍历一个对象的属性和方法,赋值给另一个对象

### 继承的应用 000

1. 拓展内置对象(给内置对象新增成员)
2. 如何安全的拓展内置对象

function MyArray(){

}

var arr = new Array();

MyArr.prototype = arr;

var myArr = new MyArray();

//myArr这个对象是一个MyArray这个构造函数

//MyArray这个构造函数的原型继承自数组对象

//myArr就可以调用Array对象的所有属性和方法

//同时myArr还可以拥有

myArr.push(1,50,6,8); //给myArr加数据

myArr.length;//myArr的长度

myArr

## 原型链

1. 每个构造函数都有原型对象
2. 每个对象都会有原型对象
3. 每个构造函数原型都会是一个对象
4. 那么每个原型对象的构造函数也会有原型对象
5. 这样就形成了一个原型链
6. 原型链

### 属性搜索原则

1. 当访问一个对象的成员的时候,先在自身找有没有,如果有,直接使用
2. 如果没有找到,则去当前对象的原型对象中去找,如果有,直接使用
3. 如果没有找到,则去原型对象的原型对象中去找,如果有,直接使用
4. 指导Object,如果还是没有,则返回null

通过修改原型链继承结构实现的继承就叫原型继承

## Object.prototype的成员

### constructor

### hasOwnProperty

### propertyIsEnumerable

判断属性是否属于对象本身

判断属性师傅可以被遍历

### toString和toLocaleString

toString转换为

### valueOf

获取对象的值

### \_\_proto\_

可以使用该对象去访问原型对象\_

## Function

### 创建函数的的三种方法

#### 直接声明函数

function fn(){ 函数体 }

#### 函数表达式

var fun = function(){ 函数体 }

#### new Function

var fun new Function(“ 函数体 ”);

* + - 1. 一个参数不穿的情况,创建一个空函数
      2. 只穿一个参数的清空,这个参数就是函数体
      3. 穿多个参数的清空,最后一个参数是函数体,前面的参数是该函数的实参

var sum = new *Function* ("a","b","return a+b");  
*console*.log(sum(*2*, *3*));

* + - 1. 实用模版引擎的方式将代码写在模版内,然后获取标签的内容

#### grguments

* + - 1. 函数内部的一个对象 arguments
      2. 当函数调用的时候系统会将所有传入的实参,依次传入这个数组对象,函数对象才可以这样使用

#### eval ( 不推荐使用 )

1. eval可以用来将字符串转换为JS代码执行
2. eval用来解析JSON格式字符串的时候,会将{}解析为代码款而不是对象字面量
   1. 解决办法,在JSON的字符串的前面拼接 ” var a =”
   2. 解决办法而,将JSON格式的字符串用()括起来
3. JSON格式的数据,有兼容性问题

#### 静态成员和实例成员

静态成员:构造函数的属性和方法,通过构造函数访问

把工具方法,作为静态成员

实例成员:实例的属性和方法,通过实例化对象访问

把跟对象相关的方法,作为实例成员

Function.prototype的原型是一个函数

## 作业

### 继承

#### 混入式继承

for .. in

使用for in遍历对象1属性,将所有的属性加到另一个对象2上

这时候所对象2继承了对象1的属性

#### 原型继承

1. 利用对象的动态特性,为原型对象添加成员
2. 直接替换原型对象
   * + - 1. 替换前的对象,在替换之后,所有的成员会丢失
         2. 替换原型的时候,需要手动指定原型对象的construtor属性
         3. construtor原型用来指向构造函数的属性
3. 利用混入给原型对象添加成员

#### 经典继承

var obj1 = Object.create(obj2)

这个时候,obj1继承自obj2

存在兼容性问题

解决方法

A 检测浏览器是否支持create

如果不支持给浏览器添加create方法

B 自定义一个函数,在函数内部判断浏览器是否支持create

支持直接调用

不支持就自己创建一个

function create(oj){

if(Objject.create){

return Object.create(obj);

}else{

function F(){

}

F.prototype = obj;

return new F();

}

}

#### 原型链

每个构造函数都有原型对象,每个对象都有构造函数,

每个构造函数的原型对象都是对象,那么也有构造函数

就形成了一个链式结构,就是原型链

##### 原型继承

通过修改原型链的结构实现继承的方式就是原型继承

##### 对象和原型的成员关系

function Prople(){};

var p = new Prople();

p对象中包含的成员有:Prople.prototype的成员和自身的成员

Perple.prototpe中的成员雨哦

#### Object.protottype的成员

constructor:指向和该原型相关的构造函数

hasOwnProperty方法:判断对象本色是否拥有某个属性

properIsEnumerable方法:

* + - * 1. 判断属性是否属于对象本身
        2. 判断属性是否可以被遍历

toString toLocalString: 将对象转换成字符串 toLocalString转换成字符串的时候应用的本地的设置格式

valeeOf方法: 在对象参与运算的时候，首先调用valueOf方法获取对象的值，如果该值无法参与运算，将会调用toString方法获取对象的值

\_\_proto\_\_:指向当前对象的原型对象

#### Function

##### 三种声明函数的方式

直接声明函数

函数表达式

new Function

##### 可以用它来创建函数

var 函数名 – new Function();创建一个空函数

var 函数名 = new Function(“函数体”);创建一个没有参数的函数

var 函数名 = new Function(“参数一”,”参数二”..”函数体”) 给Function传多个参数的时候麻醉后一个是函数体,前面的是函数实参

Function接受的都是字符串类型

##### arguments 对象

是函数内部的一个对象,在函数调用的时候,系统会默认把传入的参数都转换为数组

不管有没有形参,实参都会被存入该对象

##### eval

可以将字符串转换为js代码并执行

eval解析JSON格式字符串的时候,会将{}解析为代码段

可以在SJON格式字符串前面拼接var 变量名 =

可以在JSON格式字符前后拼接()

#### 静态层成员和实例成员

##### 静态成员

通过构造函数访问的属性和方法

##### 实例成员

通过对象去访问的属性和方法

# 就业班第四十二天(JS高级)

## 复习

### 继承

#### 混入式继承

通过for..in

#### 原型继承

##### 利用对象的动态特效,为原型对象添加成员

##### 直接替换原型对象

##### 利用混入式继承给给原型添加成员

#### 经典继承

##### var 新对象 = Object.create(想要继承的对象)

##### 方便我们进行继承操作

### 原型链

每个构造函数都有一个原型对象,每个对象都有一个构造函数,每个构造函数又有一个原型对象,就形成了一个链式结构

### Object.prototype的成员

##### .constructor属性:获取和原型对象相关联的构造函数

##### hasOwnProperty()方法:判断对象本色是否拥有某个属性

对象.hasOwnPropety(“”)

##### isPrototypeOf():判断对象1是不是对象2的原型

##### 对象1.isPrototypeOf(对象2);判断对象1是否是对象2的原型

##### propertyIsEnumerable()方法:

判断对象属性是否属于对象本身

判断属性是否可以被遍历

与的关系 1真才判断2

##### toString/toLocalstring

toLocalString将字符串转换为本地格式的字符串

##### valueOf

参与运算的时候,会先调用valueOf转换为

##### \_\_proto\_\_:指向当前对象的原型对象

### Function和eavl

##### Function可以用来创建函数

1. Function可以初始化一个构造函数,用来初始化函数对象
2. 创建函数的时候如果使用的new Function();是创建一个空函数
3. newFunction(参数):这个参数为要创建函数的函数体
4. newFunction(参数1,参数2...参数N,函数体):最后一参数是函数体,前面的参数是函数的形参
5. Function这个构造函数接受的所有参数都是字符串类型

##### eval可以将字符串转换为js代码并执行

会将字符串内的{}当作代码块来解析,所以直接解析会报错

a)解决方案一:在JSON格式的字符串前面加”var 变量名= ”

b)解决方案而:在JSON格式的字符串前后加上”()”

### Function和eval的区别

##### 共同点: 都可以将字符串转换为代码

##### 不同点:

* 1. Function创建出来的是函数,并不会直接调用,只有手动调用的时候才会执行
  2. eval 把字符串转换为代码之后,直接执行

### arguments

* + - * 1. 函数内部的一个对象,在函数调用的时候默认的会将传入的实参 存入arguments这个对象,是一个伪数组
        2. arguments.length可以用来表示传入实参的个数
        3. arguments.callee();自调用,指向函数本身

### 静态成员和实例成员

##### 构造函数调用的是静态成员

##### 实例化对象调用的是实例成员

## 知识点

## instanceOf关键字

### 对象 ubstanceOf 构造函数

##### 判断该构造函数的原型是否存在该对象的原型链上

function Peoson(){}  
var p = new Peoson()  
*console*.log(p instanceof Peoson); //返回true  
*console*.log(p instanceof Object); //返回true

## Function的原型链

##### Function的原型对象是一个空的函数,空函数的原型是Object,Object的原型对象指向null

##### Object构造函数是通过Function构造函数实例化出来的

##### Function构造函数也是他Function构造函数实例化出来的

## 面向对象复习

### 面向对象是一种编程思想

把解决问题的重点放在和解决问题相关的一系列对象上

### 面向过程

把解决问题的重点放在解决问题的每一个步骤上

### 面向对象三大特性

##### 继承:自己没有的,拿别人的属性或方法来用

##### 封装:将一些属性和方法封装起来,jQuery

##### 多态:父类的成员可以共享给所有子类,子类可以给父类添加属性

### JS对象

##### 键值对的集合(无需)就是JS对象

### 创建对象的方式

##### 使用对象字面量

##### 使用内置构造函数Object

##### 使用简单工厂函数(不推荐使用)

##### 使用自定义构造函数

### 传统构造函数的问题

##### 把方法定义在构造函数中,没创建一个对象都会新建一个方法,同样的代码会在程序内存在多个

##### 解决方法:

* 1. 把方法提取出来定义在全局中,在构造函数中引用该函数
  2. 使用原型来解决,原型中的所有成员都可以被跟其关联的对象访问

### 原型

##### 原型:在构造函数创建的时候,系统会默认创建一个对象与之关联,这就是原型对象,原型对象默认是空对象,可以被所有通过该函数的实例化对象访问

##### 原型的作用:

a)可以被所有通过该函数的实例化对象访问

##### 原型的使用方式

* + - * 1. 利用对象动态特效给原型对象添加成员
        2. 直接替换原型对象
        3. 通过混入的方式给原型对象添加成员

### 原型链

每一个构造函数都有一个原型对象,每一个对象又都有一个构造函数,这样就形成了一个链式结构就是原型链

### 继承的实现方式

##### for...in变量

##### 原型继承,通过更改原型链结构,实现继承

##### 经典继承,Object.create();有兼容性问题

var 对象名 = Object.create(要继承的对象)

### Object.prototype的成员

##### constructor:属性,指向该原型相关的构造函数

##### hasOwnProperty:方法,判断对象本身是否拥有某个属性

obj.hasOwnProperty(“属性名”)

##### isPrototypeOf(),判断一个对象是不是另一个对象的原型对象

obj1.isPrototypeOf(obj2);判断1是不是2的原型对象

##### propertyIsEnumerable方法:先判断属性是否属于本身,如果是继续判断属性是否可以被遍历,如果不是返回false

##### tostring/toLocaleString():转换成字符串

toLocaleString:转换为本地设置的格式字符串

##### valueOf():对象参与运算的时候,会首先调用valueOf方法获取底薪的值,如不行调toString

##### \_\_proto\_\_:指向对象关联的原型

### Function和eavl

##### 都可以将字符串转换成代码

##### 不同点

* + - * 1. Function创建的是函数,不会自动调用
        2. eval直接可以将字符串转换为代码并且直接执行

### arguments

##### 函数内部的一个对象,用来接受在函数调用的时候,传入的实参,切放到该对象中,是个伪数组

arguments.length:表示接受参数的个数

arguments.callee 指向当前函数(匿名函数,没有名字)

## 案例

增 数组.push()

删 数组.splice(开始删除索引,删除几个)

在当前对象中调用当前对象的方法中和属性,必须用this调用

## 递归

##### 在函数内调用函数自己,就是递归

##### 函数不调用不占用内存

##### 没有递归结束条件的递归就是死队规

### 递归的两个要素

##### 自己调用自己

##### 要有结束的条件

leetcode

nodeType判断节点类型

节点.nodeType == 1:元素节点/2:属性节点/3:文本节点

concat 返回的是一个新的数组

### 使用递归的方法

### 化归思想:把问题由难化易,化繁为简,有复杂变简单的过程成为化归

## 复习

### 原型相关知识复习

##### 原型链

##### 函数的原型链

* + - * 1. 可以把Function当作一个构造函数,其他构造函数都是Function构造函数的实例化对象,再用对象原型链的方式去分析这个原型关系

##### instanceof:判断一个构造函数的原型是不是存在与该对象的原型链上,JS中所有的对象都会有Object.protype,所以,所有对象 instanceof Object 结构都是true

##### 属性放在构造函数内,方法放在原型对象中

### 面向对象案例

### 递归

化归思想:化繁为简,化未知为一直

两个要素:自己调用自己,有结束条件

获取所有的后代元素

# 就业班第四十三天(JS高级)

## 复习

### instanceof: 对象 instanceof 构造函数

判断这个构造函数是不是在这个对象的原型链上

### 获取节点

childNodes:获取所有类型子代节点

children:获取所有子代元素节点

jQuery有find方法,获取所有后代节点

### 原型链

### 函数的一

函数也是一个对象,这个对象是Function构造函数创建的,Function这个构造函数也会有一个原型对象

对象 ✍ 构造函数 ✍ 原型对象 ✍ ... ✍ Object ✍Object.prototype ✍ null

### instanceof

### 完整原型链的绘制

### 面向对象案例

增删改查

一个有只属性的对象,把方法放进原型对象

### 递归

函数在函数内自己调用自己

必须有结束条件

思路:化归思想,化繁为简

### 递归寻找所有节点

## 知识点

##### JS执行过程,解析和执行a

##### in 判断某个对象是否存在某个属性

##### 只有函数能创建作用域

## 作用域

作用域:起作用的范围

在JS中唯一能产生作用域的东西就是函数

### 块级作用域

JS中没有块级作用域

### 词法作用域

在代码写好的哪一科,变量的作用域已经确定,这个作用域就是词法作用域

##### 规则

1. 内部函数允许访问外部函数的变量,外面的不可以访问里面的
2. 只有函数可以新建作用域
3. 作用域首先使用提升规则分析
4. 如果当前作用域中有该变量,就不使用外面的同名变量

### 动态作用域

### 作用域链

##### 只要有函数就可以创建作用域

##### 函数中又可以再创建函数

##### 函数内部的作用域可以访问函数外边的作用域

##### 如果多个函数嵌套,那么救护构成一个链式访问结构,这就是作用域链

#### 变量访问规则

##### 首先在自己的作用域查找

##### 找到直接使用

##### 找不到,去上级找

##### 指导全局作用域链

##### 再找不到就报错

##### 同级的作用域脸部key相互访问呢

## 变量和函数提升

##### 变量提升,只提升变量名

##### 函数提升,提升整体

##### 提升var声明的变量和以function开头的函数

##### 普通函数的this指向window

##### 构造函数没有形参可以省略括号

##### 必须使用var声明的变量才属于这个作用域

##### 声明函数是的函数名,其实也是一个变量

##### 可以通过这个变量名给其赋值

##### new Foo.getName();先计算Foo.getName,在计算new(无意义)

##### new Foo().getName();先执行new Foo();在执行后面的

### 预解析提升

### 变量和函数同名

在提升的时候变量和函数同名,忽略变量,只提升函数

### 函数同名

会将函数都提升,但是后面的函数会覆盖前面的函数

函数表达式不会被提升

### 变量的提升分作用域

## 闭包

##### 封闭的,对外不公开的包裹结果,或空间

##### 闭包的原谅就是作用域访问规则

##### 上级作用域无法直接访问下级作用域

### 闭包要解决什么问题

##### 怎么让函数外面访问到函数里面的数据

##### 下级作用域可以访问上级作用域的数据

### 闭包基本模式

##### 在外部函数内创建函数

##### 将外部函数的返回值设置为新创建的函数

##### 在外部函数就可以接收到返回值

##### 使用这个函数就可以在外部访问到内部的数据

### 作用

创建一个私有控件,保护数据

外部想要访问数据,只能他用过函数内部提高的方法

在提供的方法中,我们可以设置一些条件,来规定修改的数据

## 复习

### 使用递归获取后代元素

### 作用域

变量起作用的范围

##### 块级作用域, 使用代码块现代的作用域,JS中没有块级作用域,

##### 词法作用域:在代码写好的时候就确定变量的作用域

##### 动态作用域

##### JS中只有函数能创造作用域

### 变量提成

JS代码执行的两个阶段

#### 预解析阶段

##### 变量名和函数提升

1. 将var声明的变量和function开头的函数提升
2. 提升到当前作用域的最上方

#### 执行阶段

#### 注意

##### 变量和函数同名时:只提升函数,忽略变量

##### 函数同名是:都提升,但后面的会覆盖前面的

##### 函数表达式,只会提升变量名,不提升后面的函数

##### 变量提示只会将变量和函数提升到当前作用域的最上方

##### 变量的提示是分块<script>的

##### 写在条件判断语句里的函数能否被提升,取决于浏览器

### 作用域链

只要函数都有作用域,函数内部的作用域可以访问外部的作用域

当多个函数嵌套的时候,就会形参一个作用域链

### 变量访问规则

### 闭包

一个封闭的对外不公开的包裹结构或者空间

JS中的闭包是函数

#### 闭包要解决的问题

##### 在函数外部访问不到翻书内部的数据

##### 要解决的问题就是需要在外部简洁访问函数内部的数据

#### 闭包的基本结构

function foo(){

var name = ‘jsak’;

return function() {

return name;

}

}

复杂结构

function foo(){

var name = ‘jsak’;

return {

getName:function(){

return name;

}

setName:function(value){

name = value;

}

}

}

#### 作用

闭包可以创建一个私有空间,在做这个空间内,外部无法直接访问

创建一个私有控件,保护数据

外部想要访问数据,只能他用过函数内部提高的方法

在提供的方法中,我们可以设置一些条件,来规定修改的数据

# 就业班第四十四天(JS高级)

## 复习

##### 作用域JS没有块级作用域

##### JS没有块级作用域

##### JS的作用域是词法作用域

##### 变量提升

##### 作用域链

##### 绘制作用域链图

##### 变量搜索规则

首先坐在使用变量的作用域进行查找,找到直接使用

找不到去上一级寻找

指导0几作用域

##### 闭包

##### 一个封闭的对外不公开的空间结构

##### 解决函数内部的数据无法在外部访问的问题

##### 在函数内创建函数进行返回

##### 作用域链的作用,函数内部的函数可以访问函数外部的数据

## 知识点

## JS单线程

##### 单线程,一次只能处理一件事情,多线程可以实现多件事情同时做

##### JS中,分了三个任务

##### 渲染任务

##### JS代码执行任务

##### 事件出了处理任务

### 立即执行函数IIFE (自调用函数)

* + - * 1. (function (){})()

## 缓存

1. cache
2. 作用就是将一些常用的数据存储起来
3. 提升性能
4. cdn

## 沙箱

1. 与外界隔绝的一个环境,外界无法修改内部的数据
2. 沙箱基本模型
   1. 沙箱中将所有变量的定义放最上面
   2. 中间放一些逻辑代码

(function(){

var a= 123;

})();

1. 如果需要在外加暴力一些属性或者方法,就可以将这些属性或者方法加到window全局对象上
2. 但是这个window全局对象不可以直接引用,因为引用会破坏沙箱原则
3. 所以我们选择使用传参的形式将window对象传入沙箱内
4. 一般应用在书写第三方框架或者为第三方框架书写插件或者书写功能独立的一些组件
5. 654

#### 沙箱的优势

1. 杀星模式使用的是IIFE,不会在外界报篓任何全局变量,也不会找茬代码污染
2. 沙箱中的所有数据,都和外界隔离的,外界无法对其进行修改

#### 沙箱实现原理

1. 函数可以创建作用域, 上级作用域无法访问下级作用域

## 函数的四种调用模式

看调用语句来确定是那种调用模式

在那个函数内看那个函数的调用模式

### 函数调用模式

##### this指向winsow全局对象

##### fun()

### 方法调用模式

##### this指向调用这个方法的对象

##### 对象.getName()

1. var obj = {  
    *name*:'jack',  
    sayHello: function () {  
    *console*.log('hello');  
    }  
   }  
   obj.sayHello();

### 构造函数调用模式

##### this指向使用new创建出来的对象

##### new Obj()

##### 简单工厂模式的构造函数,创建出来的对象和工厂函数无关

* 1. function Person(name,age) {  
      var o = {  
      *name*: name,  
      *age*: age  
      }  
      **return o;**}  
     var p = Person('jack',*20*);  
     *console*.log(p);//P是一个Object对象,和构造函数Person()无关

##### 寄生式构造函数

* 1. function fun(name, age) {  
      var o = {  
      *name*: name,  
      *age*: age  
      }  
      return o;  
     }  
     var obj = new fun('jack', *20*);

创建出来的是Object对象,和构造函数无关

##### 简单工厂模式的构造函数,实际上调用模式是函数模式调用

### 上下文模式

##### 左值(等号左边) 右值(等号右边)

##### 左值被赋值

##### 右值不能被赋值,

##### 上下文调用模式中,可以修改this的值,也就是可以修改函数调用方法

##### 使用如下方法可以修改上下问,也就是this的值

##### **apply:**函数.apply(**this修改成的那个对象**,**[可传可不传]**)//传数组

##### **call**:函数.call(**this修改成的那个对象**,**[value1], [value2], ...[valueN]**);

##### 定义在Function.prototype里

##### 数组可以点出来任何属性不报错

##### call和apply的区别

##### 第一个参数都是要把this修改成对象

##### 当函数需要参数的时候,那么apply用数组进行传值

##### call用单个参数传值

##### apply和call方法第一个传入参数是null的时候都表示为函数调用模式

##### 也就是将this指向window

##### array.prototype === [];两个书写方式表达的意思相同

##### api文档语句中[],的内容表示可有可无

##### undefined\*undefined = NaN

# 就业班第四十五天(JS高级)

## 复习

### 闭包

### 斐波那契输了性能问题

### jQuery缓存

arr.shifu();删除并且返回数组第一个元素

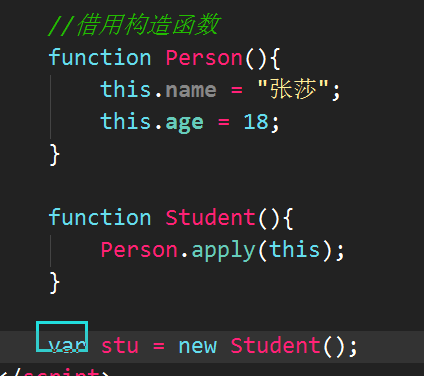
### 沙箱模式

(function(w){  
 //变量定义  
 //逻辑代码  
 //如果需要,使用window向外界暴露借口  
 //window == w  
})(window);

### 函数的四种调用模式

1. 函数调用模式
   1. this指向widow
   2. fun();
2. 方法调用模式
   1. this指向调用方法的对象
   2. obj.fun();
3. 构造函数调用模式
   1. this指向new创建出来的对象
   2. var fun = new Obj();
4. 上下文调用模式
   1. apply(对象,[])
      * 1. 改变函数内部的this指向
   2. call(对象,[],[],[],...,[])
      * 1. 改变函数内部的this指向
   3. fun.call(null) === fun.call(window)
   4. 全局对象window有name这个属性
   5. 在全局访问一个未定义变量会报错
   6. 用对象访问一个未定义的变量返回undifined
   7. arr.push(arr1)会把整个数组arr传进去
   8. arr.push.apply(arr,arr1);会把arr1拆开传进去
   9. 使用call和apply第一个传入的参数为值类型的时候,会将值类型转换为对应的对象(引用类型)

### 借用构造函数



## 注册事件方式

1. ele.on事件 = function
2. **addEventListener(事件类型,事件处理函数,useCaptrue)**
   1. 是元素的方法
   2. IE9之前不支持
   3. useCaptrue:Boolean值
      * 1. true在捕获阶段调用同类事件(从外到内)
           1. 事件执行顺序:捕获 ✍ 目标
        2. false在冒泡阶段调用同类事件(从内到外)
           1. 事件执行顺序:目标 ✍ 冒泡
   4. 事件三个阶段:
      * 1. 捕获(从外向内寻找触发事件的元素),
        2. 目标(找到了触发事件的元素),
        3. 冒泡(事件从内向外冒泡)
   5. 兼容写法
      1. function warp() {  
          if (*window*.addEventListener) {  
          **return function(ele, event, fun) {  
          ele.addEventListener(event, fun);  
          }** } else if (*window*.attachEvent) {  
          **return function(ele, event, fun) {  
          ele.attachEvent('on' + event, function() {  
          ele.apply(ele, window.event)  
          })  
          }** } else {  
          **return function(ele, event, fun) {  
          ele['on' + event] = fun;  
          }** }  
         }  
         var a = warp();  
         a(div, 'click', function() {  
          alert('ad');  
         })
3. attachEvent(事件类型,事件处理函数)
4. 三种方法中的this指向不一样

## JS高级总复习

### JS基础复习

### 面向对象

#### 面向对象的概念,基础理论

1. 把解决问题的重点放在与解决问题相关的对象上
2. 封装/继承/多态
   1. 封装:把属性和方法封装到一个对象里
   2. 继承:自己没有的拿别人的来用
3. 对象:万物节对象
   1. JS对象:键值对的无序集合
   2. 属性:数值类型的,名词
   3. 方法:函数类型的,动作行为
4. 创建对象的方式
   1. 字面量:用完之后不可以复用
   2. 内置对象创建:
      * 1. 缺点:创建的是空对象,需要手动添加属性
        2. 动态特效:创建好对象后,可以给对象添加属性和方法
   3. 工厂函数
      * 1. 定义一个对象,给他动态添加属性,返回
   4. 自定义构造函数
      * 1. 构造函数:函数的一种,一般用来初始化对象
        2. 和new一起出现,new创建一个对象,构造函数初始化对象
        3. 首字母大写
        4. 返回值是new创建的对象
        5. 如果收到写return
           1. return的是基本类型,不影响,仍返回new的对象
           2. 如果是Object类型的,返沪这个object
        6. 执行过程
           1. new出来对象
           2. 把对象赋值给this
           3. 给对象添加成员
           4. 返回
        7. 原型
           1. 在构造函数在创建出来的时候没洗头默认会创建一个对象与之关联,这个对象就是原型对象
           2. 访问

构造函数.prototype

对象.\_\_proto\_\_

* + - * 1. 原型的使用方法

原型继承

直接替换

替换之前和替换之后对象的原型不同

混入式添加属性和方法

经典继承

* + - * 1. 原型链

每个对象都有原型对象,原型对象又有原型对象,

指导object.prototypr

* + - 1. object.prototype的属性
         1. hasOwnProperty
         2. toString/toLocalString

toString直接转换字符串

toLocalString:转换为本地格式的字符串

* + - * 1. valueof:获取当前对象的值

能参与运算,直接用

不能参与运算,调用toString

* + - * 1. .\_\_proto\_\_

#### Function

1. 创建函数
   1. 表达式
   2. 2
   3. 3
2. instanceof:判断对象是否在原型链上
   1. 随便对象 instanceof Object true
3. Function是自己创建出来的

### JS高级知识

#### 递归

##### 求和/阶乘/斐波那契数列/获取所有页面节点

#### 作用域

1. 变量起作用的范围
2. js是词法作用域
3. js没有会计作用域
4. 只有函数可以创建作用域
5. 变量提升
6. js执行阶段
   1. 预解析阶段
   2. 执行阶段
7. 变量同名
   1. 函数和变量:不管变量,只提升函数
   2. 两函数同名:后面覆盖签名的
8. 作用域链
   1. 外部作用域无法访问内部的
   2. 内部的可以访问外部的

#### 闭包

1. 封闭的对外不公开的空间
2. 外面访问不到
3. 外部访问只能通过内部的借口
4. 作用
   1. 保护数据
5. 案例
   1. 返回多个值,设置和获取
   2. 歌曲播放列表
   3. 解决了斐波那契数列问题
   4. 闭包对缓存实现了一个保护作用
6. jQuery实现缓存
   1. 定义一个对象储存数据
   2. 定义一个数组储存键
      * 1. 记录顺序
        2. 记录长度,超容量判断

### 沙箱模式

1. 一个封闭的对外隔绝环境
2. 避免外界印象内部的数据
3. 使用自调用(IIFE)函数实现
4. 限制作用域
5. 自调用函数没有名字
6. 代码结构
   1. 最上面:定义变量
   2. 中间:逻辑代码
   3. 下面:暴漏接口
      * 1. 最后加上window
        2. 给函数一个形参
        3. 形参.
7. 作业
   1. 封装第三方插件
   2. 封装框架

### 函数四种调用模式

1. 函数调用模式
   1. this指向window
2. 方法调用模式
   1. this指向与之关联的对象
3. 构造函数调用模式
   1. this指向new创建的对象
4. 上下文调用模式
   1. call(对象,[],[],..[])
   2. apply(对象,[])
      * 1. 可以用apply实现借用构造函数
        2. 在当前函数内使用
           1. 其他函数.apply(this)

### 注册事件兼容性

1. addEventListeven(事件,处理函数)
2. attachEvent(on事件,处理函数)
   1. 回调函数的this的指向与其他俩不一样
3. .on事件 = 处理函数

### 歌曲管理案例

##### 方法放在原型

##### 属性放在构造函数内

### Tab栏切换案例

## js严格模式

“use strict”;

# 就业班第四十六天(canvas)

## convas

1. 这个图从哪来
   1. 展示绘图效果的标签,行内块元素
   2. 打开canvas,(通过getContext方法打开),然后在上面绘图
2. canvas绘图步骤
   1. **移动到开始位置**
      * 1. .moveTo(x,y)
   2. **开始画线条**
      * 1. .lineTo(x,y)
   3. **描边显示**
      1. .stroke()
3. 画布大小在convas标签设置
   1. <canvas id = ‘cns’ width = 500 height = 300></canvas>
4. **描边颜色**:.strokeStyle = ‘颜色’: 描边之前设置颜色
5. **描边宽度**:.lineWidth = 数值;
6. 解决锯齿:多画一条边(不推荐使用)
7. **闭合路径:**.closePath(),可以省去最后一条边的绘
8. 封装绘制矩形函数
9. **清除路径**.beginPath();为了放置重绘
10. **填充**:.fill() 不定义填充,定义填充颜色也不起作用
11. **填充颜色**.fillStyle = ‘颜色’
12. **如果先填充后描边,描边宽度正常**
13. **先描边,后填充,描边宽度会显示一半**
14. 非零环绕原则:判断哪些区域属于路径内
    1. 在路径包围区域中,早一点没想外发射一条射线
    2. 和所有环绕他的边相交
    3. 开启一个计数器,从0计数
    4. 如果这个设下遇到顺时针环绕,+1
    5. 遇到逆时针环绕-1
    6. 计算结果非0,即在路径内
15. 先填充,后描边,才能显示设置的宽度
16. canvas在绘制线条的时候,会左右偏移线宽的一般
    1. 线宽是奇数,边缘的颜色会变浅一半
    2. 线宽是偶数,无影响
17. **线帽样式**:.lineCap = ‘
    1. butt(默认值)
    2. round:圆头,半径为线宽的一半
    3. square:方头(各突出线宽的一半)’
18. **交点样式**:.lineJoin = ‘
    1. miter:默认尖角
    2. round:圆角
    3. bevel:平角’
19. 画一条矩形路径:.rect(starX,starY,width,height);
    1. 只是路径,要在后面描边stroke();
20. 绘制一个描边矩形:strokeRect(starX,starY,width,height)
21. 绘制一个填充矩形:fillRect(starX,starY,width,height)
22. 清除画布.clearRect(starX,starY,width,height)

## 渐变矩形

1. rgb模式,颜色数值不能为小数

## 虚线

.setLineDash([实线长度,虚线长度])

传一个值表示实线虚线各占多少

设置值的个数为奇数时,系统存储的是设置值的个数\*2

1. .getLineDash()获取如何设置的线条规则
2. .lineDashOffset = 3 偏移量

## 绘制网格

## 会绘制坐标系轴

## 绘制折线图

## forEach

var arr = 数组.forEach(function(value,index,arr){

把数组内的每一项执行一次函数

不会使用返回结果

}[,this])

## map

数组.map(function(value,index,arr){

把数组内的每一项执行一次函数

返回一个新数组

}[,this])

# 就业班第四十七天(canvas)

## 复习

1. 非零环绕原则
2. 1

### API

1. getConText(‘2d’||’webgl’):获取一个绘图环境(工具)
   1. width,height:画布的宽高
2. .moveTo(x,y):获取起点
3. .lineTo(x,y):画一条路径线
4. .stroke():根据当前路径描边
5. .strokeStyle=’color’:描边颜色
6. .fill():填充
7. .fillStyle=’color’:填充颜色
8. .lineWidth=3:描边宽度
9. .closePath():闭合路径
10. .beginPath():清除路径
    1. 只是清除路径,不清除样式
11. .rect();画一个举行路径
12. .fillRest(x,y,width,height):填充一个矩形
13. .strokeRest(x,y,width,height):描边矩形
14. .clearRest(x,y,width,height):
15. .setLineDash([a,b,c,d]):设置虚线规则
16. .getLineDash():获取虚线规则
17. .lineDashOffset = 2;设置偏移量
18. .lineCap = ‘butt/square/round’
19. .lineJoin = ‘miter/bevel/round’

## 画弧

1. .arc(圆心X坐标,圆心Y坐标,半径,起点弧度,结束点弧度,[是否逆时针画])
2. arc会从上次的结束点到弧的开始点之间画条直线

## 文字API

### 描边文字

1. .strikeText(文字,参考X坐标,参考Y坐标[,限制文字最大长度])
2. .font = ‘加粗 字号/行高 字体’:设置字体样式
   1. 单独设置字号不可以,必须加字体样式才会生效
3. .在canvas中绘制文本，不会产生路径吗

### 填充文字

1. .fillText(文字,参考X坐标,参考Y坐标[,限制文字最大长度])
2. .textAlign = ‘left/right/start’对齐方式
3. .textBaseline = ‘top/middle/bottom/默认基线(alphabetic)’

# 就业班第四十八天(canvas)

## 绘制图像.drawImage()

1. .drawImage()
   1. 三种使用方式
      * 1. 三个参数
           1. .drawImage(获取标签,坐标X,坐标Y)
           2. img.onload = function(){}:img加载完毕之后触发
        2. 五个参数:把图片放置到指定位置,并指定宽高
           1. .drawImage(获取标签,坐标X,坐标Y,width,height)
        3. 九个参数:把裁剪后的图片放到指定的位置,并且知道大小
           1. .drawImage(获取标签,裁剪图片的位置X, 裁剪图片的位置Y, 裁剪图片的width, 裁剪图片的height,坐标X,坐标Y,width,height)

## 状态的保存与回滚

1. .save();状态保存
   1. 把当前的状态(绘制环境的所有属性)复制一份保存
2. .restora();状态回滚
   1. 把最近保存的一次状态作为当前状态
   2. 回滚一次会把上次保存的属性删除
      * 1. 再回滚会拿到上次保存的属性
   3. 可多次回滚,依次往回找保存的属性

## 平移/旋转/缩放

1. **平移.translate**(X平移的量,Y轴平移的量)
2. **旋转.rotate**(旋转的弧度)
3. **缩放.scale**(X轴缩放比值,Y轴缩放比值)
4. 以绘制的图像不会手打影响
5. 只会收到之后的图像
6. 角度转弧度:Math.PI / 180 \* deg (deg是要转换的角度)
7. 绘制一个旋转的图像
   1. 现平移坐标轴到图像的中心
   2. 旋转坐标轴
   3. 绘制图像
      * 1. 平移旋转之后,坐标体系变化
        2. 先保存最初的坐标系
        3. 绘制完之后在回滚

# 就业班第四十九天(canvas)

## 复习

1. 定时器的this指向window
2. 状态的保存与路径无关
3. 内部使用的方法一般用 \_ 开头
4. var s = Math.floor( Math.randow() \* (max-min+1) + min ); //生成min-max之间的随机数

## 复习

1. 原型链知识
   1. Array instanceof Arrat 返回false
2. ‘use strict’; 开启严格模式
3. 构造函数return非基本类型的数据,新对象就是该数据

1.fun():函数调用模式,this指向window

2.Obj.fun():方法调用模式,this指向调用fun的对象

3.new Fun():构造函数调用模式:this指向新创建的对象

4.fun.apply/call(this,[]):上下文调用模式:this指向参数设置的内容

# 就业班第五十天(框架封装)

## 学习课程介绍

1. JS原生框架
   1. 模仿jQuery搭建
2. 学习目标
   1. 复习jQuery
   2. 复习原生js方法
   3. 了解jquery的大概实现
   4. 锻炼编码能力
3. 第一天-框架结构
   1. jQuery对象本质
   2. 框架核心结构
   3. 入口函数(构造函数)
4. 第二天-核心方法
   1. 入口函数对函数的处理
   2. 原型上的核心方法
   3. map和each
5. 第三天-Dom操作
   1. 创建dom
   2. 删除dom
   3. 追加dom
6. 第四天-属性样式操作
   1. class属性操作
   2. 公告属性操作
   3. 样式操作
7. 第五天-事件
   1. 事件绑定
   2. 事件解绑
8. 第六天-最后
   1. Ajax
   2. 插件实现原理
   3. 折线图饼图做成框架插件
   4. 刮刮乐插件

## jQuery

1. jquery对外暴露了$和jQuery这两个方法
2. 调用这个方法,可以得到一个jQuery实例对象
3. jquery实例对象是一个伪数组对象
4. jquery和$是一个工厂函数
5. 只有浏览器有window对象
6. **原型简写**
   1. jquery.fn = juqery.prototype = function(){}
7. **jquery插件的实现机制**
   1. 就是给原型添加方法
   2. jquery.prototype.新插件 = function(){}
   3. 也就等于jquery.fn.新插件 = function(){}
8. 任何函数都可以是构造函数,new谁 谁就是构造函数
9. **传入null / undefined / 0 / ‘’ /NaN**
10. **传入的是选择器,会返回获取到的元素**
    1. 按照下标存储起来,同时添加一个length属性代表数量
11. **传入HTML片段,会根据这些片段创建DOM对象**
    1. 然后按照下标储存起来,同时添加一个length熟悉代表数量
    2. document.body.appendChild($(‘<span></sapn>’)[0])
    3. 返回的是一个伪数组
12. **传入一个数组或者(**伪数组**),会把数组中的每一项分别添加到实例身上,同时把length有也加过来**
    1. $([‘a’,’b’])
13. **传入对象,直接把对象按照下标添加到实例身上**
14. **传入基本数据类型,null,ndefined,字符串除外,**
    1. 直接把数据添加到实例身上
    2. 传字符串会被认定为是选择器
15. **传DOM对象,也会添加到实例身上**
16. **传入的是null,undefined,0,NaN,’’**
    1. null,undefined,0,NaN,’’ 转换为boolean类型都会是false
    2. 返回一个this
    3. 构造函数里的this指向新的实例对象
17. **传入的是html片段**
    1. html开头结尾都是<>,并且length>=3,就可认为是一个html片段
       * 1. 创建一个临时的div
         2. 设置这个div的innerHTML为html片段
         3. 这些片段就转变为了div的子元素
         4. 遍历div的子元素分别添加到this身上,记住给实例补充length属性
         5. 可以使用数组的push()来给实例添加,同时可以使用apply简化遍历过程
         6. **实现**
            1. **var tempbox = document.createElement(‘div’)**
            2. **tempbox.innerHTML = selector;**
            3. **[].push.apply(this,tempbox.childNodes)**
18. **传入的是选择器,按照选择器获取页面中的DOM,把DOM添加到实例身上**
    1. 使用querySelectorAll获取所有的元素
    2. 把遍历到的所有元素分别添加到实例身上
19. **传入伪数组和数组**
    1. **排除函数和window,因为函数和window都有length属性**
       * 1. 排除函数 typeof function
            1. if( typeof selector == ‘function’ ){ return false}
         2. window.length属性代表页面中iframe的数量
         3. window.window === window
         4. if(select.window === window)

{return false}

* + - 1. 如果一个数据的window属性是自己,那么就认为它是window
  1. **toString判断是不是真数组**
     + 1. new Array()创建的数组和[]字面量创建的数组
       2. 稀疏数组
          1. 某些数组下标不存在
  2. **判断是不是伪数组**
     + 1. 伪数组特点
          1. 必须有length属性
          2. 如果length属性值是0,那么这个对象没有任何元素,是空数组
          3. 如果length属性不为0,那么这个伪数组一定有(length-1)为下标是属性值
       2. **判断伪数组**

**必须有length属性**

**要么length为0,要么有length-1这个下标**

* + - 1. 判断过程

1. 传入的是选择器
2. charAt()获取数组指定下标元素
3. 借用其他对象方法
   1. 借用数组方法
      * 1. [].push.apply(this,)
        2. [].pop.apply(this,)
4. replace(/^\s+|\s+$/g,’’):替换首位空字符

## 传入jQuery不同参数的不同返回结果

1. 传入null,undefined,0,NaN,’’返回空对象空实例
2. 传入字符串,需要贩毒案是html片段还是其他
   1. 是片段,创建对象的DOM,然后添加到实例身上
   2. 否则按照选择器获取页面中的DOM,然后把获取到的DOM添加到实例身上
3. 如果是数组或者伪数组,那么把每一项分别添加到实例身上
4. 出了上面的数据类型,剩余的统一添加到实例身上

## 对传入的参数进行处理

## toString伪数组

1. 根据内部的this 返回一个类似于这样的字符串
   1. [ object className]
   2. **Object.prototype.toString.call(this的指向)**
   3. Array Data都继承自Function
   4. new Array 是Array类型,Array是Function类型
   5. ({}) === Object.prototype
2. 不能获取用户自定义对象的类型
3. 只能获取内置对象的类型
4. 自定义对象类型都返回[object object]
5. {}:js默认识别为代码块
   1. 代码块就是一对{}
   2. 如果把一对大括号赋值给其他变量,或者参与运算,大括号就变成了自面对新
   3. ():是一个运算符

# 就业班第五十一天(框架封装)

## 把入口函数融合到框架中

1. Function 的length属性是形参的个数
2. window的length属性是ifrim的数量
3. 判断是不是window
   1. w.window === w
4. !!str 把str转换为boolean类型值
5. **静态方法**是加给实例
6. **实例方法**是加给原型
7. 构造函数不能使用自己原型中的方法(Function例外)
8. 实例不能直接使用构造函数身上的静态方法
9. apply可以改变this指向,同时可以把数组或者伪数组怕你平铺传入函数
   1. IE8,apply只能平铺真数组,或者内置伪数组,自定义的伪数组会报错
   2. 需要把自定义伪数组转换为真数组,再使用apply方法
   3. [].push.apply(this,[].slice.call(伪数组))
      * 1. slice()不传参数整体截取

## 传入函数

1. 会在页面解析完成后执行
2. DOMContentLoaded比onload快
3. 如果document.readyState === “complete” 说明dom已经构建完毕
4. 否则判断docment.addEventListeer

如果支持document.addEventListeven(‘DOMContentLoaded’,fn)

否则document.attachEvent(‘onreadyststechange’,function()){

if(document.readyState === ‘compalete’)

fn()

* 1. }

## jQuery主要方法

1. jquery:获取版本号
2. selector
3. length
4. toArray:转换为数组,获取的是原生DOM
   1. return [].slice.call(this)
5. get:获取元素
   1. 不写参数,获取所有
   2. -1:获取最后一个
   3. 传入null或者undefined,转换为数组返回
   4. 如果传入的是证书,获取指定下标的元素返回
   5. 传入的是负数,按照下标倒着(a.length +负数 )获取元素
   6. **实现**
      1. if (*i* == null) /\*null和undefined一样\*/  
         {  
          return this.toArray();  
         }  
         if (*i* > *0*) {  
          return this[*i*]  
         }  
         else {  
          return this[this.*length* - *1*]  
         }
6. **each**
   1. 数组的forEach先获取value再获取index
   2. jQ的each先获取index后获取value
7. **map**
8. slice:
   1. 截取部分实例,构成一个新的jQuery实例返回
   2. 通过数组的slice截取部分实例(slice返回的是数组)
   3. 把截取到的元素按照实例返回
   4. **实现**
      1. var nodes = [].slice.*apply*(this, arguments)  
         return jQuery(nodes)//包装成实例对象
9. first():得到第一个jQuery实例
10. last():得到最后一个jQuery实例
11. eq():
    1. 传入null和undefined,返回一个新的实例
    2. 不穿参数,获取一个新实例
    3. 传入正数,按照指定下标返回元素,再包装成新实例返回
    4. 传参按照指定下标获取元素,获取到的元素是新的jQuery实例
12. push():给实力添加一个新元素
13. sort():给实例排序
    1. .sort(function(a,b){return a.innerHTML>b.innerHTML })
14. splice(开始下标,删除几个,要替换的元素)

# 就业班第五十二天(框架封装)

## 实例方法和静态方法

1. 通过构造函数创建出来的对象,就叫实例
2. 实例和对象的关系
   1. 万物皆对象
   2. 实例就是对象
3. **实例成员**
   1. 提供给实例使用的方法和属性
4. **实例方法**
   1. 提供给实例使用
   2. 构造函数原型中的成员也是实例方法
   3. 提供给所有实例对象使用
   4. **构造函数不能使用实例方法(定义在原型中)**
   5. **实例不能使用静态方法(构造函数.方法)**
5. **静态成员(类成员)**
   1. 提供给构造函数使用的方法和属性
6. **静态方法(类方法)**
   1. 提供给构造函数使用的方法
   2. 构造函数 点 出来的方法
   3. 不创建实例也可以使用
   4. 可以在复用
   5. 实例不能直接使用静态方法,通常会根据静态方法再实现一个实例方法,供实例使用
7. **实例复用静态方法**
   1. 静态方法服务面更广
   2. 实例方法只服务于实例
8. 变量名全大写代表常量
9. 4

## 函数

1. **普通函数**
   1. 直接定义
      * 1. function fn(){}
        2. 会被预解析
   2. 函数声明
      * 1. var fn = function(){}
        2. 不会预解析
2. **方法**
   1. 一个函数有了宿主

var obj = {

fn :　function（）{}

}

1. **构造函数**
   1. 如果一个函数,被通过new用来创建对象
   2. 通过new创建的函数
2. **回调函数**
   1. 被作为参数传递给另一个函数
3. 4

## 常量

1. 常量就是不变的量
   1. 比如:Math.PI
   2. 常量全大写
2. 变量就是可能会变化发量

## 类

1. 是对某一组具有相同特征或者特性的对象的抽象描述

## object.prototype

### Object

#### new Object

#### 字面量 {}

### 所有函数默认的原型

#### 所有函数默认的原型

##### 内置构造函数的原型

* Object.prototype
  + 终点
* Function .prototype
  + 所有函数继承 ^^^
    - 内置函数
      * Object
      * Function
      * Boolean
    - 自定义函数
      * Person
      * XXX
      * new Object()
* Number.prototype
  + 数字对象
  + new Number()
* Array.prototype
  + 所有数组
    - new Array
    - 字面量 []
* String.prototype

##### 自定义构造函数默认的原型

* Person
  + Person所有的实例
    - new Person
* XXX
  + XXX的所有实例

### Math

##### 没有实例继承Math

### prototype:显式原型

##### 原型属性

### \_\_proto\_\_:隐式原型

##### 原型对象:就是实例继承的对象

## 原型链

长度不定,一定有限

## DOM操作

1. **empty**:清空所有元素内容,返回值是实例
   1. **实现**
   2. 遍历
   3. 清除遍历到的每一个元素
      * 1. 遍历到的元素.innerHTML = ‘’;
   4. foreach(this,function(value,index)){}
   5. for(var I = 0,len = arr.length;len<10;i++)
      * 1. 为了优化
2. **remove**:删除所有的元素,**返回值为实例,为了链式编程**
   1. 遍历
   2. 删除遍历到的每一个元素
      * 1. 获取要删除的节点
        2. 获取其父节点,parentNode
        3. 父节点.removeChild(要删除的节点)
3. 对于empty和remove
   1. 只有操作元素自己,并没有涉及到获取其他内容
   2. 所以可以返回实例,继续链式变成
4. 对于html和text
   1. 获取操作,得到的是获取的值
   2. 设置操作,得到的是实例,可以继续链式编程
5. **html**:设置所有元素的内容,获取第一个元素的内容
   1. **传入0个参数,返回结果为第一个元素的内容**
   2. **传一个参数,返回值是实例,能解析标签**
   3. **传参设置,不传参获取**
   4. **实现**
      * 1. 通过arguments判断有没有传入参数
        2. 没有传入参数,获取第一个元素,返回这个元素的innerHTML的内容
        3. 如果传入了参数,遍历
        4. 设置每一个元素的innerHTML内容为传入的参数
6. **text**:设置所有元素文本内容,获取所有元素的文本内容,返回实例,不解析标签
   1. **实现**
      * 1. 通过arguments判断有没有传入参数
        2. 传入一个参数,遍历,把每一个俄元素的innerHTML加一起返回
        3. 如果传入了一个以上参数,遍历设置
        4. 设置每一个元素的innerHTML内容为传入的参数
7. **appendTo**:把所有的元素,添加到指定元素中
8. **append**:给所有的元素,添加新的内容
9. **prependTo**:把所有的元素添加到指定元素的前面
10. **prepend**:给所有的元素的最前面,添加新的元素
11. arguments:存储的是一个代表实参的对象(伪数组)
    1. 存储了所有传进来的实参
    2. 通过有一个length属性代表所有实参的数量
    3. 参数不定的时候用arguments
12. callback(回调函数)
13. 给jq工厂传入一个jq对象,可以人是传入jq对象的close版本
14. 给jq传入一个DOM对象,得到一个新的实例存储这个DOM
15. 给jq传入一个选择器,得到一个新的实例,存储着获取到的DOM元素

10..54

11..30

# 就业班第五十三天(框架封装)9.06

## 知识点

1. insertBefore(要添加的元素,添加到哪)
2. 第一次添加本体,第二次添加得clone(克隆)

## 链式编程

1. 调用一个方法,该方法可以返回一个对象,可以继续使用该对象的方法
2. 调用该对象的方法,可能仍然返回以一个对象,也可能是原来的对象
3. 那么就可以继续调用返回对象的方法,这样一直下去,直到方法没有返回对象为止

## append(),appendTo(),prepend(),prependTo()

1. **$(‘div’).append(‘字符串’):**传入字符串,直接在原来的innerHTML上累加字符串
2. **$(‘div’).append(jq对象):**传入jq对象,把这个对象的每一项分别添加到所有的div中
   1. **传入dom对象和传入jq对象一样,因为东湖村纳入的dom对象先要包装成jq对象**
   2. **只有字符串要独立处理**
3. 返回的是this,谁调用返回谁

## 属性和样式操作

1. **addClass**:添加新的class
2. **removeClass**:删除所以元素指定的class
3. **hasClass**:判断所有元素有没有指定的class,只要是有一个有就返回true
4. **toggleClass**:有删除,没有则添加
5. **attr**:设置或者获取元素的属性节点
   1. **属性节点**:
      * 1. 标签的属性就是属性节点,
        2. 只能属于DOM对象,
        3. 有DOM节点的特性(nodeType,nodeName等)
           1. nodeType节点类型:

元素节点

属性节点

文本节点

8. 注释节点

9. 文档节点

* + - 1. 所有DOM,都有一个attributes属性
      2. 存储了该DOm所有的属性节点
      3. 通过Attributes属性获取所有的属性节点对象,然后按照下标取出,获取每一个属性节点对象,在通过nodeValue属性得到这个节点只
      4. 通过DOM的getAttributes(属性节点名)**获取**
      5. **设置属性节点值**:通过setAttribute(value值)
  1. **属性:**
     + 1. 对象身上的变量,
       2. 可以属于任意节点,
       3. 属性和存储的东西没有共性(任意存储)
       4. 属性可以通过点的方式添加
       5. 通过 点 获取 或者[]获取

1. **prop**:设置或者获取元素的属性值
2. **val**:获取或者设置元素的value属性
   1. 传参数设置.val()
   2. 不传参获取
   3. 原生获取:元素.getAttrobute(‘属性值’)
      * 1. 获取默认的
        2. 最新的值通过.value获取

# 就业班第五十四天(框架封装)

## 事件模块

1. on:事件绑定
2. off:事件移除
3. 事件绑定方式
   1. DOM.on事件名 = function(){}
   2. DOM.addEventListeven(事件名,function(){})(IE8不支持)
   3. IE8:DOM.attachEvent(on事件名,function(){})
4. 事件解除方式
   1. DOM.on事件名 = null
   2. DOM.removeEventListeven(事件名,function(){})(IE8不支持)
   3. IE8:DOM.dettachEvent(on事件名,function(){})
5. DOM元素都有nodeType
6. off传入事件类型,解除该事件类型的所有回调
   1. 传入事件类型和回调,那么解除该事件的指定回调

# 就业班第五十六天(git)

## shell

1. 图形化界面shell
2. 命令行shell
3. 认识bash这个shell
4. bash命令格式：命令[-options][参数]

## 命令

1. cd:切换目录
   1. cd .. 返回上一级
   2. **cd /c/projec**t cd ~/desktop 家目录下的desktop
   3. ./ 当前目录
2. ls:查看目录
   1. **Ls –a;查看所有文件（包括隐藏文件）**
   2. –l:查看详细信息
   3. –a –l == -al
3. pwd:查看当前目录
4. **mkdir:创建文件夹** (**Make Directory**)
5. touch:创建一个文件
   1. touch js/index.js:在js文件夹下创建文件
6. **cat:查看文件所有内容**
   1. cat 文件名
7. more/less:查看文件
   1. **less:分页阅读;q退出**
8. rm:删除文件(**remove**) 删除文件，如 rm index.html
9. **rm -r node\_modules 删除非空文件夹**
10. rmdir:删除文件夹,只能删除空文件夹
11. **mv:移动/重命名文件**
    1. mv 移动的文件 移动到哪 **mv index.html images** 移到images文件夹下
    2. mv 原文件 修改文件名后的文件 可实现改名

**mv index.html images/aaa.html** 移到images下更改名字

Mv index.html abc.html 重命名

1. **cp:复制也可以修改文件名**
   1. cp 原文件 修改文件名后的文件 可实现改名
2. head:查看文件前几行
   1. head -5 文件名
   2. head –n -5 文件名
   3. 两种方式一样
3. tail:查看文件后几行
   1. tail -10 文件名
   2. tail –n -10 文件名
   3. 两张方式相同
4. history:查看历史
5. 重定向 > 覆盖 >> 追加 :修改输出的位置
   1. ls > index.html 把ls的结果输出到index.html这个文件内
6. wget 网址 下载
7. tar 解压缩
8. 命令 –help 查看帮助文档
9. curl 网址 模拟网络请求
10. huami:查看当前用户
11. | 管道符:可以将多个命令连接
    1. 上一个命令的执行结果,会被当作下一个命令的参数
12. grep:正则匹配内容

## VI编辑器

如同Windows下的记事本，vi编辑器是Linux下的标配，通过它我们可以创建、编辑文件。它是一个随系统一起安装的文本编辑软件

打开/创建文件， **vi 文件路径 vi index.html**

1. **命令模式**
   1. dd:删除当前行
   2. yy:复制当前行
   3. p 粘贴内容
   4. u辙销操作，可多次使用
   5. ctrl+f:向下翻页;ctrl+b:向前翻页
   6. **进入编辑模式**
      * 1. i:当前光标处插入
        2. a:当前光标后插入
        3. A:光标移动端行尾
        4. o:光标下面插入行
        5. O:光标上面插入行
2. 底行模式

:w保存，:w filename另存为

:q退出

:wq保存并退出

:e! 撤销更改，返回到上一次保存的状态

:q! 不保存强制退出

:set nu 设置行号

## SSH

SSH是一种网络协议，用于计算机之间的加密登录。

1. 对称性加密:加密和解密使用相同的密钥
2. 非对称性加密:加密和解密用不同的密钥
   1. 公钥
   2. 私钥

## GIT分布式版本控制

1. git管理文件的三中状态
   1. **已提交:committd ✍ git仓库**
   2. **已修改:modified ✍工作目录**
   3. **已暂存:staged ✍ 暂存区域**
2. 把普通目录变为git工作区
   1. **git init**
   2. **git add** 文件名 把文件放到暂存区
      * 1. **git add –A 全部加入暂存区**
        2. **git add \* 全部加入暂存区**
   3. git commit –m ‘标记’
      * 1. 执行一次commit就新建了一个版本/存盘点
        2. 可以还原到任意一个存盘点
   4. git config –global user.name/email 配置
   5. git status 查看仓库状态
      * 1. modified 被修改的
        2. 绿色的代表暂存区的文件
        3. untarched files 未追踪的文件
   6. git checkout 文件名 将暂存区的内容还原到工作区,(撤销更改)
   7. git reset --hard 提交ID

# 就业班第五十七天(git)

## git分支

1. 分支
   1. 至少要创建一次commit
   2. 一个人干多件事情的时候,为了互相不干扰
   3. 存盘点基于当前分支
   4. 新创建的的分支基于当前分支
      * 1. 会继承父分支的所有提交历史
2. git **branch** 分支名称: 创建分支
3. git branch :查看分支
4. git cheackout 分支名称:切换分支
5. git **merge** 要合并的文件名 合并文件
6. git **branch –d** 删除的分支名
   1. 删除失败可能是上次任务没有完全提交
   2. fit branch –D 强制删除
7. 不要有太多的树杈（子分支）
8. 要有一个“稳定分支”，即**master**分支不要轻意被修改
9. 要有一个开发分支（**developer**），保证master分支的稳定性
10. 所有的功能分支（**feature**）从developer创建
11. 所有功能开发完成后新建发布分支（**release**）

## GIT远程仓库

1. git push 远程仓库地址 分支名称 : 将分支共享到远程仓库
2. git pull 远程仓库地址 分支名称 : 获取分支
3. 共享仓库
   1. .git结尾的目录
   2. 必须是裸仓库(没有工作目录的仓库)
      * 1. git init –-bare: 创建裸仓库
   3. git自动忽略空文件夹
4. git remote add 变量名 共享仓库地址路径
5. git clone 远程仓库地址 [自定义名] 把共享仓库克隆至本地

## github

1. 创建密钥
   1. ssh-keygen –t rsa
   2. [www.ruanyifeng.com](http://www.ruanyifeng.com)

## 步骤

* + - 1. git config 配置本地仓库
         1. git config –-global user.name
         2. git config –global user.email
         3. git config –- list查看配置详情
      2. git init 初始化仓库
         1. git init –bare 可以初始化一个共享(裸)仓库
      3. git status 查看仓库状态
      4. git add 将工作区文件添加到暂存区
         1. 可写文件名单独放
         2. –A / \* 全部放到暂存区
      5. git commit –m ‘备注信息’ 将暂存区文件到到本地仓库
      6. git reset –hard id值 还原
      7. git log 查看本地仓库提交历史
      8. git branch 分支名 创建/查看分支
      9. git checkout ‘分支名称’ 切换分支
         1. **git checkout –b ‘分支名’** 创建并切换分支
      10. git merge ‘分支名’ 合并分支
      11. git branch –d/D 删除分支
          1. D强制删除
      12. git clone 仓库地址 克隆仓库
      13. git push 仓库地址 分支名 上传至仓库
      14. git pull 仓库地址 分支名 下载文件
      15. git remote 查看已添加远程主机
      16. git remote show 主机名 查看主机ip
      17. git remote add 变量名 地址 把地址放到变量名中
      18. git push 地址 本地分支:远程分支 将本地分支上传至远程仓库
      19. develop
      20. feature

# 就业班第五十八天(angularJS)

## 知识点

1. 前端MVC框架 / 诸多类库的集合 / 以**数据和逻辑**为核心
2. backbone / knockoutJS / **vue** / react 框架
3. 核心特性:模块化 / 双向数据绑定 / 语义化标签 / 依赖注入

## MVC开发模式

1. / 模型(model): 处理数据
2. / 视图(view): 展示数据
3. / 控制器(controller): 链接模型和视图的桥梁.
4. 静态资源 css,js,images
5. 放在public文件夹下
6. $this - > display () 展示视图
   1. - > 调用方法
7. $this - > assgin(‘赋值给谁’,变量) 赋值

## 模块化

1. 引入angular之后,会有一个angular全局对象,对象下有很多方法
2. **angular.module(‘模块名’,[依赖的模块]) :** 可以创建一个模块
   1. **var App = angular.module(‘模块名字’,[依赖关系])**
      * 1. App就是新创建的模块,这个App又是一个对象
        2. 在此对象下也有很多方法
        3. 实现具体逻辑
   2. App.controller(‘name’,[‘$scope’,function($scope){ }]) : 创建一个控制器
      * 1. $scope是一个空对象,就是module
        2. ng-app:指定模块
        3. ng-controller:指定控制器
        4. 不能嵌套
        5. ng-repeat = ‘ (key,内容) in 数组 ’

## 指令:

1. angularJS自定义的HTML属性或者标签
2. ng-为前缀
3. **内置指令**
   1. **ng-app**:定义应用
   2. **ng-controller**:定义控制器
   3. **ng-repeat**:循环标签
   4. **ng-bind** = ‘属性名’: 绑定属性 ✍ {{属性名}}
   5. **ng-show** = ‘true / flase’ ✍ 显示内容
   6. **ng-hide** = ‘true / false’ ✍ 隐藏内容
   7. **ng-if** = ‘true / false’ ✍ 控制元素是否存在
   8. **ng-src** / href ✍ 增强图片路径 / 地址
   9. **ng-class** = ’{类名 : true / false}’
   10. **ng-include** = “ ‘模版路径’ ” 引入模版
   11. **ng-disabled** = ‘true / false’ 表单禁用
   12. **ng-readyOnly** = ‘true / false’ 只读标签
   13. **ng-checkbox** = ‘true / false’ 选中复选框
   14. **ng-selected** = ‘true / false’ 默认选中
4. 自定义指令:通过directive自定义指令
   * 1. App.directive('name', function () {  
         return {  
         *restrict*: 'EA',   
         // E:element可作为标签   
         // C:class可作为类名   
         // M:mark可作为注释   
         // A:attribute可作为标签使用   
         *template*:'<p>aaaa</p>',  
         *templateUrl*: '路径',  
         *replace*:'替换'  
         }  
        })

## 数据绑定

1. angularJS:以数据驱动的MVC框架
2. 单项绑定:模型(model)上的数据传递到视图(view)
3. 双向绑定:模型的数据传递到视图,视图的数据也可传递到模型
4. 相关指令:
   1. **模型(model)**数据向**视图(view)**模版绑定:{{}} ng-bind
   2. ng-clock:解决闪烁
   3. ng-bind-template = ‘{{数据1}}{{数据2}}’
   4. ng-init : 初始化model
   5. ng-repeat : 循环
   6. ng-switch on = ‘条件’ : 根据表达式显示或者隐藏对应的部分
      * 1. ng-switch-when = ‘匹配条件’
        2. ng-switch-default:默认选中项
5. 视图的数据向模型绑定(双向绑定(视图数据✍✍模型数据))
   1. 视图数据向模型传递要借助表单元素
   2. ng-model(data) –--视图向模型传递数据---✍ $scope.data
   3. ng-sibmit 回车提交事件
   4. from事件
   5. jrap裁切jQuery插件

# 就业班第五十九天(AngularJS)

## 作用域

1. 只有函数可以新建作用域
2. 子级可以访问父级作用域,父级不可以访问子级
3. controller控制器可以创建作用域
4. 视图可以相互嵌套
5. ng-init初始化模型
6. ng-app所在的标签是根作用域

## 过滤器

1. {{val|过滤器:’格式’}}
2. 内置过滤器(内置九个)
   1. date:时间;
      * 1. yyyy – MM – dd EE h:m:s; 年 - 月 - 日 星期 时:分:秒
   2. currency:货币
   3. filter:给的数组中选定符合条件的,返回一个新数组,
      * 1. 条件可以可以是函数\字符串\对象(**对象后面的括号要加一个空格**)
   4. json:转换为json对象
      * 1. IE7以下,对象最后一个元素不要有逗号
        2. json最后一个对象也没有
   5. limitTo:’数值’; 截取数组
   6. lowercase:转换小写
   7. cuuercase:转换大写
   8. num:’保留几位小数’转换为数值:默认保留三位小数四舍五入
      * 1. number()转换字符串为数值,字符串中不能有任何非数值字符
        2. parseInt()转换字符串为数值,遇到第一个非数字字符停止
        3. parseFloat()转换字符串为数值,遇到小数点后第一个非数字字符停止
   9. ordeyBy:’某一个属性(如果排序的是对象)’:true(降序)/false(升序);对数组排序

## 自定义

1. 自定义指令(directive)
   * 1. //自定义指令  
        App.directive('name', function () {  
         return {  
        })
2. 自定义过滤器(filter)
   * 1. //自定义过滤器  
        App.filter('demo', function () {  
          
         return function(input) {  
         //首字母大写  
         return input[*0*].toUpperCase()+input.slice(*1*);  
         };  
        })
3. 自定义控制器(controller)

# 就业班第六十天(angularJS)

## 复习

1. XMLHttpRequest对象,具备向服务器端发送请求和接受想要的一个能力
2. var xhr = new XMLHttpRequest;
3. xhr.open(‘get/post’);
   1. post请求放在请求主体里
   2. get发送请求的时候没有请求主题
   3. 请求主体就是请求的参数的一些数据
   4. get发送可以不设置请求主体
   5. $\_POST
   6. 直接传对象
      * 1. 请求头
   7. angularJS默认支持json格式
4. xhr.send();
5. alert会阻塞代码执行

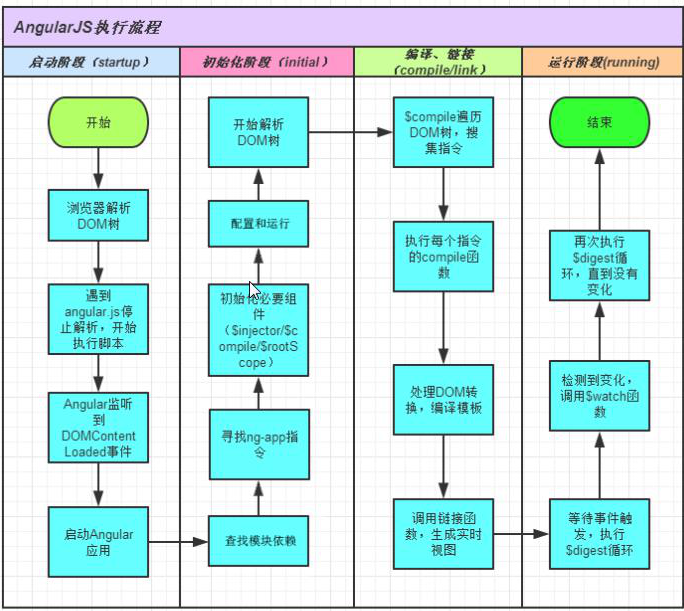
## jsonp:一种跨域的解决方案

1. 标签的src/href属性没有跨域的限制
2. jsonp利用src属性跨域
3. **angularJS的$http**
   * 1. $http({  
         *url*:'', //请求地址  
         *method*:"jsonp", //请求方式  
         *headers*:{}, //请求头  
         *data*:{}, //post传递的数据  
         *parmams*:{  
         *call*:'JSON\_CALLBACK' //回调函数(固定的  
        }, //get请求的数据  
        }).success(function(/\*返回的数据\*/){  
         '对返回的数据处理'  
        })

## 自定义服务

1. 服务的本质是一个对象或者函数
2. **factory**
   1. 一个方法
   2. 0要显式runtun一个对象回去
   3. 新建一个服务有可能依赖其他服务
   4. arr.factory('name',[/\*依赖的服务\*/],function(){  
      //自定义服务,依赖于依赖的服务  
       return function(){  
       return *data* }  
       })
3. servic
   1. app.service('name', ['依赖', function (/\*参数\*/) {  
       //添加具体方法  
       this.test = function () {  
       }  
       /\*相当于 return {  
       text:function(){}  
       }\*/}
4. value
   1. app.value('name','data')
   2. 本质上是一个服务
   3. 表现形式上是一个常量
   4. 常量就是不变的值

## 模块加载



1. js会阻塞页面解析
2. js不要卸载head里面,一般放在后面
3. 一个服务在代码中就是一个模块
4. 配置块从功能说就是配置服务

## config配置服务(配置块)

1. 大部分服务一般都对应一个provider,用来执行与对应服务相同的功能或对其进行配置
2. **App**.*config*(['$logProvider', '$filterProvider', function ($logProvider,$filterProvider) {  
    //允许一次性配置多个  
    //传递一个数组,依赖前面注入的方式  
    $logProvider.debugEnabled(false)//禁用debug  
    //新增一个过滤器  
    $filterProvider.register('name',function(){  
    return function(input){  
    return input[*0*].toUpperCase()+input.slice(*1*);  
    }  
    })  
   }])

## 运行块

1. 直接运行相应的服务模块
2. App.run(['$http', '/\*依赖服务\*/', '$rootScope', function ($http/\*依赖服务\*/, $rootScope) {  
    $rootScope.*name* = '根作用域';  
    //其他模块都可以访问到  
    //执行服务  
   }])

## 雅虎十三条

1. 尽可能的减少 HTTP 的请求数 [content]

2. 使用 CDN（Content Delivery Network） [server]

3. 添加 Expires 头(或者 Cache-control ) [server]

4. Gzip 组件 [server]

5. 将 CSS 样式放在页面的上方 [css]

6. 将脚本移动到底部（包括内联的） [javascript]

7. 避免使用 CSS 中的 Expressions [css]

8. 将 JavaScript 和 CSS 独立成外部文件 [javascript] [css]

9. 减少 DNS 查询 [content]

10. 压缩 JavaScript 和 CSS (包括内联的) [javascript] [css]

11. 避免重定向 [server]

12. 移除重复的脚本 [javascript]

13. 配置实体标签（ETags） [css]

14. 使 AJAX 缓存

## 路由

锚点就是前端的路由

1. SPA但页面应用
   1. 不产生页面跳转
   2. 把若干功能集成到一个页面
   3. 动态生成数据,通过ajax异步获取
   4. 为了增强用户体验
   5. 可以提升性能
   6. 仿制手机APP的交互

### 单页面应用

* + - 1. hashchange监听锚点变化,绑定给weindow对象

*window*.addEventListener('hashchange', function () {  
 var hash = *location*.*hash*; //获取锚点  
 hash = hash.slice(*1*)//去除#号  
 var xhr = new XMLHttpRequest;  
 xhr.open('get', 'test.php?hash=' + hash);  
 //将锚点作为参数传递给php  
 xhr.send();  
 xhr.*onreadystatechange* = function () {  
 if (xhr.*readyState* == *4* && xhr.*status* == *400*) {  
 var resute = xhr.*responseText*;  
 *document*.querySelector('.content').*innerHTML* = resute  
  
 }  
 }  
})

code.angularjs.org

* + - 1. 简单单页面

<body>  
<a href="#/index">index</a>  
<a href="#/include">include</a>  
<a href="#/no">no</a>  
<div class="content">  
 <div ng-view></div>  
</div>  
</body>  
<script type="text/javascript" src='../../angular.min.js'></script>  
<script type="text/javascript" src='../../angular-route.js'></script>  
<script type="text/javascript">  
 var App = *angular*.module('app', ['ngRoute'])  
 App.*config*(['$routeProvider', function ($routeProvider) {  
 $routeProvider.when('/index', {  
 *template*: '<h1>content</h1>'  
 }).when('/include', {  
 *template*: '<h1>include</h1>'  
 }).when('/no', {  
 *template*: '<h1>no</h1>'  
 }).otherwise({  
 *redirectTo*: '/index'  
 })  
 }])

* + - 1. json\_encode(arr);pHp把数组转换为字符串
      2. 前端路由就是锚点
      3. 路由参数: $routeParams:获取参数
         1. url问号传参数
         2. 占位符传参
      4. foreach($list as $key => $val)

## 其他

1. angular.element():把原生DOM对象转换为jQuery对象
2. Angular’s jqLite
3. bower:用来下载插件的
4. npm install bower -g --registry=https://registry.npm.taobao.org
5. 1

## 作业

* jQuery设置请求头

# 就业班第六十一天(Gulp)

1. 常用工具:Grunt,Gulp,F.I.S(百度),webpack
2. 项目构建
   1. 使用Sall或Less等CSS预处理器解析css
   2. 合并css,js代码,压缩HTML,css,js,images,可以加速网页打开速度
   3. 构建工具就是通过简单配置可以帮我们实现合并,压缩,效验
   4. npm install 工具名称

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

* 1. 设置淘宝镜像为默认下载地址:npm config set registry=https://registry.npm.taobao.org
  2. 本地安装: npm install gulp --save-dev

**gulp-less** 编译LESS文件

**gulp-autoprefixer** 添加CSS私有前缀

**gulp-cssmin** 压缩CSS

**gulp-rname**重命名

**gulp-imagemin** 图片压缩

**gulp-uglify** 压缩Javascript

**gulp-concat** 合并

**gulp-htmlmin** 压缩HTML

删除空白符:collapsewhitespace:true

**gulp-rev** 添加版本号

版本号为MD5字符

**gulp-rev-collector** 内容替换

**gulp-useref** 合并文件,不压缩

**gulp-if**

在目标根目录创建一个gulpfile.js配置文件

var gulp = *require*('gulp');  
var less = *require*('less');   
gulp.task('less', function () {//引入依赖包  
 gulp.*src*('路径')

//需要构建资源的路径，字符串或数组（可以正则方式书写）  
 .pipe(less()) //管道，将需要构建的资源“输送”给插件。  
 .pipe(gulp.dest('保存路径'))

// 构建任务完成后资源存放的路径（会自动创建）  
}

[www.npmjs.com](http://www.npmjs.com)

npm install === npm i

package.json

# 就业班第六十二天(补课)

## 单页面

1. 基本目录结构
   1. public()
      * 1. css
        2. js
        3. images(UI图)
   2. views(视图)
   3. scripts(业务逻辑js)
   4. uploads(内容图,上传文件)
   5. api(接口目录)
2. 开发步骤(使用angularJS)
   1. 引入
   2. 指定ng-app
   3. 创建模块var app = angular(‘name’,[依赖模块])
   4. 按模块的思想开发将具有类似功能的程序归类
   5. 将使用控制器归类,所有指令归类
   6. 在使用时将这个独立的模块作为app的依赖使用
   7. 通过块配置和when筛选执行哪一个模块
   8. 创建控制器app.controller

## 移动端

1. em相对于自己当前的字号长度
2. rem相对于跟元素的字号长度html元素
3. 浏览器的默认字号16px
4. 1rem = 16px
5. C3属性,PC端尽量不用,移动端支持多
6. 媒体查询
   1. all 所以设备
   2. print 打印机和打印机预览
   3. screen 屏幕电脑屏幕 手机屏幕 屏幕电脑

@media only screen and (max-widthj:320px){

html{

background:pink

}

}

* 1. 关键字
     1. device-width:屏幕宽度

## css预处理器

##### less

lessc 源文件.less 转换后.css

* + <link rel= ‘stylesheet/less’ herf=’xx.less’>
  + <script src = ‘js/less.js’></script>
  + less.js内部可以发送ajax请求
  + 通过一些正则进行替换操作
  + 然后将替换的结果生成一个<style>标签
  + style = document.createElement(‘style’)
  + style.innerHTHM = 替换的结果
  + less里@import ‘路径’; 引入其他less文件
  + 盒子都要有高度
  + :not(.class): 排除

## git

1. .gitignore:忽略文件
2. git difftool:比较工作区与暂存区的不同
3. git difftool ‘’ 比较与特点提交的差异
4. git difftool ‘’ ‘’ 比较两次的差异
5. git –hard
   1. -–soft
6. git push 地址 – delete 删除分支

# 就业班第六十三天(nodeJS)

## JS文件过多问题

1. 命名冲突
   1. 模块开发
   2. 代码复用
2. 文件依赖

## 如何进行模块化开发

1. 全局函数
   1. 会造成命名冲突问题
2. 命名空间
   1. var name = { }
   2. 通过约定的方式定义的
   3. 可以解决命名宏图问题,不好
   4. 驼峰命名:autoFileName
   5. 匈牙利命名: 类型+变量名;fnQueryNode
   6. 无法做私有封装
3. 划分私有空间,需要约定
   1. var func = (function () {  
       var add = function (a, b) {  
       return a + b;  
       }  
       return {  
       add: add  
       }  
      })(window.func ||　{});  
      func().add()

## SeaJS

1. SeaJS所在的路径为默认根目录
2. condi : 配置
3. cmd规范 : define用来定义模块的
4. require:加载JS
   1. 原理:创建script标签,引入js文件
5. config:配置

seajs.config({

base:'定义根目录',

alias: {

'别名' : '路径' //方便调用

}

})

1. moudle.exports ✍ 相当于return
   1. 可以返回方法,字符串,对象
   2. 不同之处
   3. 两次moudle.exports下面会覆盖上面
   4. moudle.exports的初始化是一个空对象
   5. 用来打破封装性,返回想要的属性和方法

# 就业班第六十四天(NodeJS)

1. node.js V8引擎解析js
2. 事件驱动,无阻塞I/O模型
3. js引擎套一个壳子,写js代码传给引擎,操作硬件
4. js> ✍ nodejs ✍ libuv
5. nvm npm nrm
6. 安装与配置
7. repl交互式运行环境
8. Ctrl + c 退出
9. **fast-init**
10. process.stdout.write(‘log原型’)
11. process全局对象,和现在启动中的node进行交互

## 总结

1. node(一门技术)和JS的关系
2. v8引擎
3. 特点
   1. 无阻塞I/O,事件驱动
   2. 轻量/高效
4. nvm的安装
5. global
6. 断言assert(foo==3(田条件),’输出内容’)

# 就业班第六十五天(NodeJS)

## 知识点

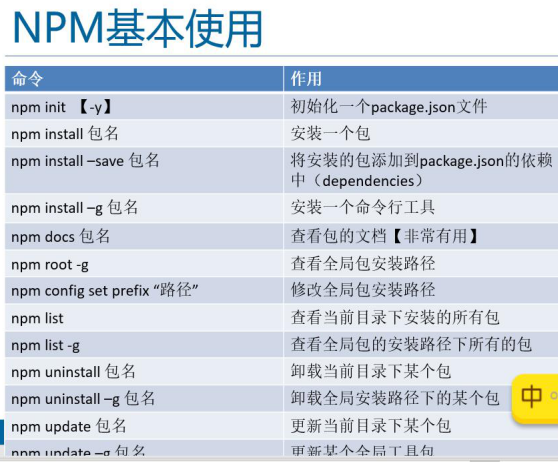
1. node是技术
2. javascript是语言
3. 基于v8引擎,不用考虑兼容性
4. 特点:事件驱动,异步非阻塞i/o模型
5. node优点:轻量高效
6. node.js可运行运行在linux系统上
7. path环境变量
8. REPL运行环境
9. node 运行js文件
10. js文件所在目录
    1. node js文件名
11. \_\_dirname和\_\_filename
    1. 文件名和文件完整路径
12. global
13. cinsole
    1. assert(foo == 1, ‘失败’); 断言
    2. time(‘timer’) ---✍ timeEnd( ‘timer’ )
14. 一个文件就是一个模块
15. 在一个莫款内的变量-函数-对象都属于这个模块,对外封闭
16. require(‘路径’):引入模块(读取js文件)
17. exports
    1. module.exports的一个别梦-引用,
    2. 打破封装性
18. readFile(‘路径’,’编码格式’,回调函数)
    1. 异步读取文件
19. rreadFileSync(‘路径’,’编码’)
    1. 同步读取
20. eavl把字符串转换成js代码
21. node会按照.js.node.json的此讯 依次补足 拓展名,一次尝试
22. 田永强<深入迁出nodejs>

## 包

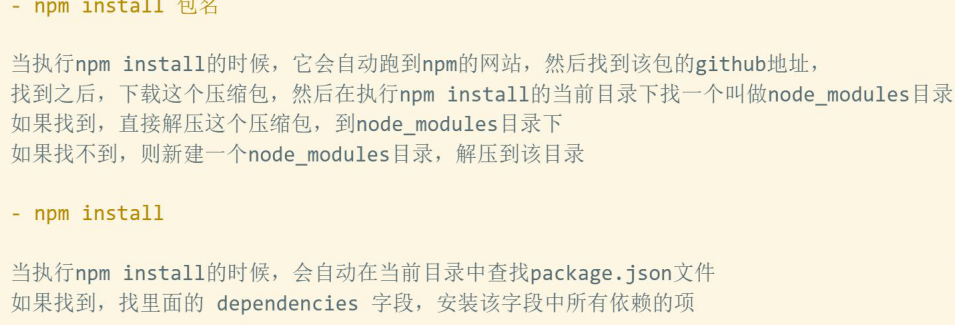
1. 包是一个文件夹
2. package.json : 包的描述文件,说明文件
3. Bin : 存放可执行二进制文件的目录
4. Lib ; 存放javascript代码的目录
5. Doc : 存放文档的目录
6. Test : 存放单元测试用例的代码



1. npm init



1. npm install : 包管理工具



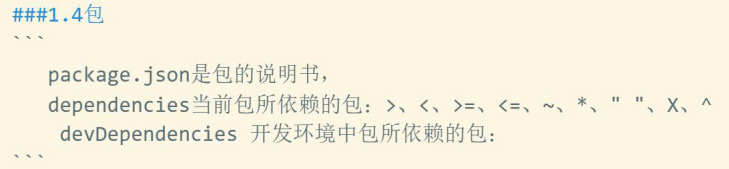
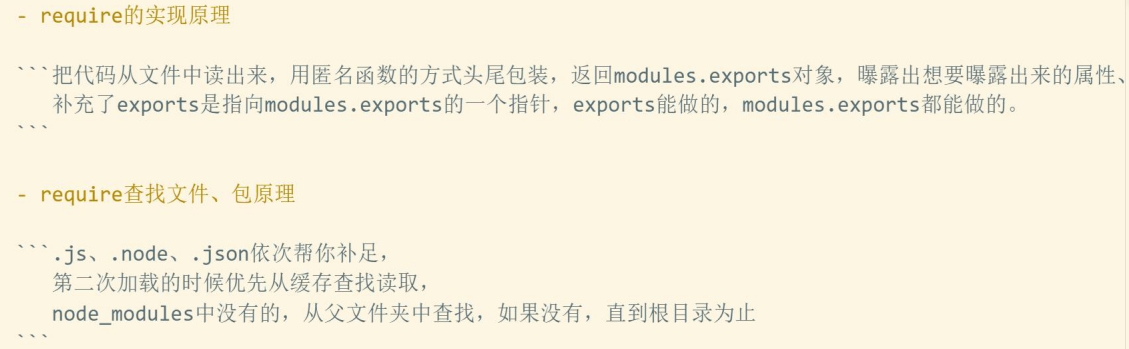
1. Cnpm



1. cnpm
2. 箭头函数
3. ng-bind ;: 可以防止页面闪烁
4. angular底层核心原理,了解胀检查机制

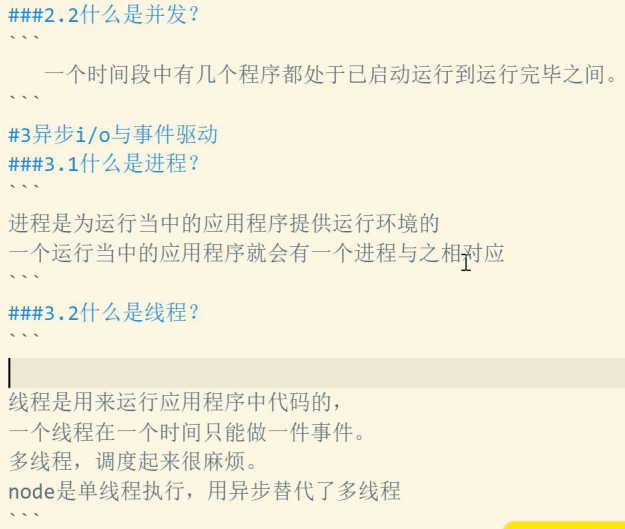
# 就业班第六十六天(NodeJS)

## 复习

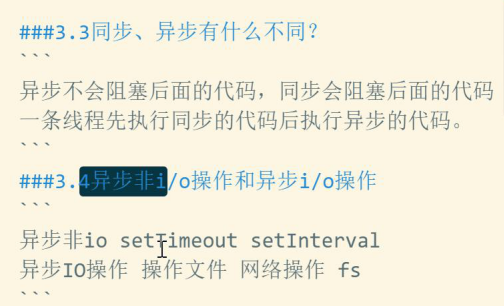


## I/O

1. input / output
2. I / o 服务器上的读写操作
3. **并发:**一段时间段内有几个程序都处于已启动运行到运行完毕之间



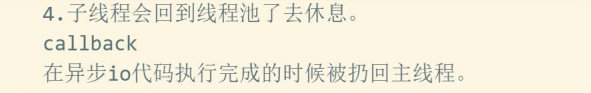
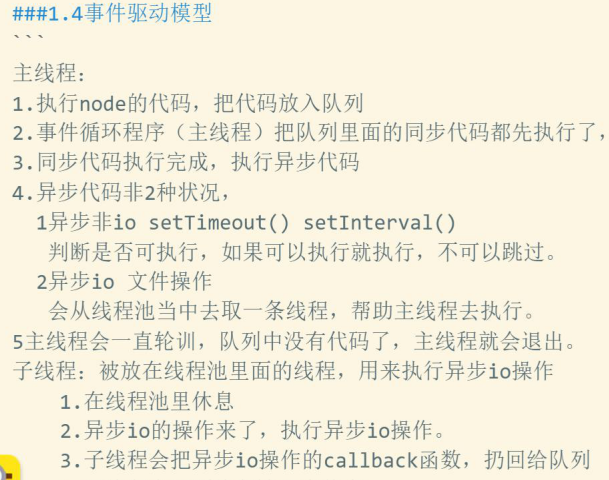
1. 异步:
   1. ajax
   2. setTimeout
   3. setInterval
   4. 先执行同步代码,再执行异步代码
   5. 同步代码会阻塞掉异步和同步代码



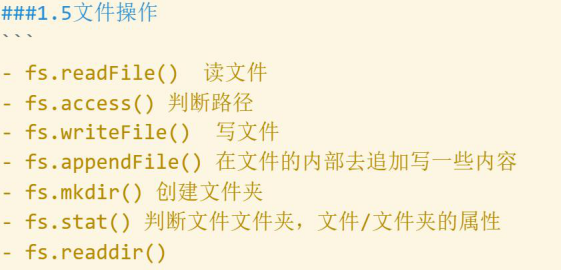
# 就业班第六十七天（NodeJS）

## 复习

1. i/o：输入输出，文件读写操作，网络操作中的请求和应答
2. 并发：同意时间内，很多程序都处在开始执行到结束之间
3. 进城：为程序提供运行环境
4. 线程：执行应用程序中的代码，同意时间内只能干一件事情
5. node中异步就是帮助完成多线程编程
6. 多线擦编程比较复杂
7. 同步异步
   1. 同步会**阻塞代码**，优先执行，主线程执行
   2. 异步不会阻塞代码执行，后执行
8. 事件驱动模型



1. 文件操作



1. 迭代：f(1),f(2),f(3),f(4)...
2. 递归：f(),f(f()),f(f(f())),f(f(f(f())))……….
3. path:
   1. path.join():字符串拼接成路径

$http.({`

url : “./a”,

tethod : ‘get’

}).success(function(data){

data是报文体;

}).then(function(data){

data:是报文,如果取里面的json数据需要data.data

},function(err){

})

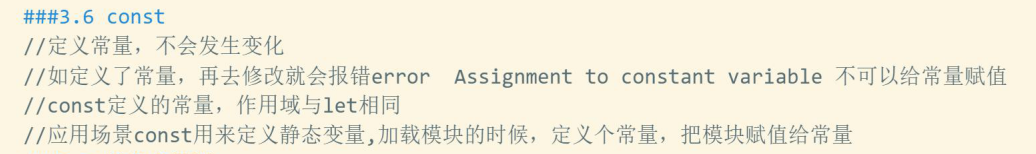
## debug

1. 调试 :
   1. node debug xxx.js
      * 1. n下一步
        2. s步入
        3. o退出
        4. –help查看帮助
   2. node-inspector
   3. VSCode调试
   4. 打开项目文件夹 -> ->



* 1. 点击甲壳虫 ->
  2. 设置调试环境node.js ->
  3. 打开launch.json ->
     + 1. "program": "${workspaceRoot}/xxx.js",

1. IDE:集成开发环境
2. use strict : 严格模式
   1. 消除代码运行的一些不安全的地方,保证代码运行的安全
   2. 提高编译器效率,增加运行速度
   3. 为未来新版本的js做好铺垫
   4. ‘use strict’以下的内容都执行严格模式
3. let 定义变量 : 定义的变量值作用于一个或括号内,一个作用域
4. const : 定义常量,引包的时候使用
   1. 常量不可修改值,内容恒定



1. 字符串拓展
   1. str.includes(str):是否包含str这个字符串
   2. str.startWith(str):是否是以str开头
   3. str.endWith(str):是否以str结尾
   4. str.repeat(次数):循环打印字符串
   5. 链接字符串 `前${tab上面的点}后`
2. promise
   1. 异步流程控制解决方案
   2. 回调地狱
   3. .then()
3. 异步流程 ✍ 回调函数的嵌套
4. Buffer
   1. 一个数组对象,每个元素存放两位十六进制数
   2. 十六进制数字
   3. 文件流:
      * 1. 流对象



1. 'use strict'
2. const fs = require('fs');
3. const rt = fs.createReadStream('./1.wmv');
4. const wt = fs.createWriteStream('./2.wmv');
5. rt.pipe(wt);

## 作业

做一个包，传一个路径，读出来，什么是文件文件夹

# 就业班第六十八天(NodeJS)

## 网络编程

1. cs / bs 架构
2. 编写程序是两台两台联网的计算机可以完成数据交互,完成网络通信
3. 服务器系统:linux的centos , ubuntu
   1. 服务器多远centos
   2. 应用服务器:作为服务器执行共享业务应用程序的底层的系统软件
   3. web 服务器:是一种应用服务器,提供web服务,对内提供web应用程序的运行环境
   4. apache / Nginx / IIS / tomcat : 服务器



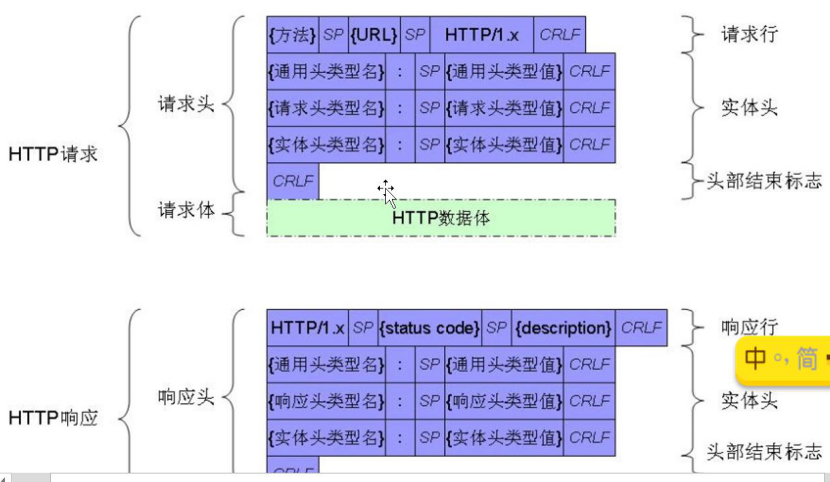
## 跨域

1. 域名不同
2. 二级域名不同
3. 端口不同,协议不同
4. 应用场景
   1. 动静分离
   2. 不适合开接口的情况

## 开发分工

## 协议

1. 协议就是双方约定好的一些数据格式



# 就业班第六十九天(angular)

## 复习

1. 协议:帮助我们简化一些事情(交通规则)
2. http协议:浏览器和服务器之间的关系
3. get/post区别
   1. post有报文体;传值安全
   2. get在地址栏传参数
4. package.json:说明文件
5. 协议:cookie

## 网站优化-✍思想

1. 能压缩的压缩,能合并的合并
2. 减少script标签,减少http请求
3. 少用js去遍历对象或者写for循环
4. 能提前运算好的提前运算好,使用枚举法(把所有可能发生的情况列举出来)
5. CDN节点,从最近的CDN节点请求数据
6. 流式(响应式)图片 ---✍ 图片压缩



## http模块

1. request:有请求发送给服务器后触发该事件
2. 触发事件后调用回调函数function(请求对象, 相应对象){}-
3. 请求报文封装到request对象里
4. 实用repsponse里面的方法构造响应把平稳,调用response里面的方法,将一下相应仍会去
5. .listen(3000):监听3000端口
6. node默认的80
7. java默认8080
8. nodejs默认3000
9. writeHead(200,{‘content-Type’:’text/html;charset:utf-8’})

# 就业班第七十天(NodeJS)

## 复习

### 第一天SeaJS

##### 模块化编程

##### 组件化、模块化

* + 1. 开源精神

##### 命名冲突、文件依赖问题

* + 1. 块级作用域
    2. es6.ruanyifeng.com
    3. es6原生支持模块化，支持文件引入

##### 依赖注入

* + 1. angular的依赖注入是为了防止文件压缩
    2. 要什么，在参数上体现

##### 开闭原则

* + 1. 不修改别人的东西

##### AMD ✍ RequireJS

* + 1. 提前执行依赖模块
    2. 懒加载

##### CMD ✍ SeaJS

* + 1. 延迟执行依赖的模块

##### SeaJS

* + 1. seajs.use（‘引入模块路径’，回调函数）
    2. 启动模块

define（function（require，exports，module）｛

var obj = require（‘./JS路径’）；

exports.add = obj.add；

exports.add2 = obj.add2；

//只暴露一个

module.exports = add；

｝）

* + 1. 写一个js模块
    2. 通过define定义一个模块
    3. 扔出对象exports.add=obj.add

### 第二天nodejs

##### 双数稳定版，单数激进版本

##### npm3.0扁平化包管理

##### npm2.0树形包管理

## MVC

1. module、view、controller
2. 代码优雅 –-> 高内聚，低耦合
3. 低耦合
   1. 独立完成任务
   2. 赋予on高兴
   3. 方便维护和修改
   4. 耦合是格模块之间相互链接的一种度量，耦合强度取决于模块借口的复杂程度，进入活访问一个模块的点以及通过
4. 高内聚
   1. 可毒性
   2. 服用型，
   3. 方便维护和修改
   4. 一个好的内聚模块应当恰好做一件事--》高内聚

## express

1. KOA2
2. express –save-dev 脚手架工具
3. 快速构架年框架模型
4. 惊人的路由能力
5. var app = express();创建一个服务器实例
6. express.static（）将某个目录里面的文件变成一个静态网站
7. app.use使用express.statuc()的方法
8. app.get（‘/getData’，function（）｛
9. res.end（‘’）
10. ｝）