HTML

权重

可以把样式的应用方式分为几个等级，按照等级来计算权重

1. !important　权重值为10000
   . . 内联样式　权重值为1000
   . . ID选择器　权重值为100
   . . 类，伪类和属性选择器　　权重值为10
   . . 标签选择器和伪元素选择器　　权重值为1
2. 其他权重为0

文件英文名

1. 文 件 夹： index.html 首页 favicon.ico ico图标 image 固定使用的图片素材 uploads 非固定使用图片、
   . css 文 件： base.css 基础公共样式 common 多个页面相同模块 index.css首页css文件
   . 页面结构： 容 器:container 页头：header 内容：content/container 页面主体：main
   尾：footer 导航：nav 侧栏：sidebar 栏目：column
   页面外围控制整体布局宽度：wrapper 左右中：left right center 上下： prev next
   菜单：menu 子导航：subnav 头部导航:topnav 快捷导航：shortcut
   横幅：banner

SEO三大优化

```
   <!-- meta:desc 网页描述标签 -->
    <meta name="description" content="京东JD.COM-专业的综合网上购物商城，为您提供正品低价的购物选择、优质便捷的服务体验。商品来自全球数十万品牌商家，囊括家电、手机、电脑、服装、居家、母婴、美妆、个护、食品、生鲜等丰富品类，满足各种购物需求。">
    <!-- meta:kw 网页关键词标签 -->
    <meta name="keywords" content="网上购物,网上商城,家电,手机,电脑,服装,居家,母婴,美妆,个护,食品,生鲜,京东">
    <!-- 网页标题标签 -->
    <title>京东(JD.COM)-正品低价、品质保障、配送及时、轻松购物！</title>
    <!-- link:favicon : 浏览器标题栏图标 -->
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
```

HTML骨架

```
<!-- 声明html -->
<!DOCTYPE html>
<!-- 中文网站 zh-CN 简体中文 / en 英文 搜索引擎归类 + 浏览器翻译 -->
<html lang="zh-CN">
```

```
<head>
    <!--charset="UTF-8" 规定网页的字符编码
        1. UTF-8：万国码，国际化的字符编码，收录了全球语言的文字
        2. GB2312：6000+ 汉字
        3. GBK：20000+ 汉字 -->
    <meta charset="UTF-8">
    <!-- ie(兼容性差) / edge -->
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <!-- 宽度 = 设备宽度 ： 移动端网页的时候要用 -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- 还有SEO三大标签 -->
    <!-- 网页标题 -->
    <title>Document</title>
</head>
<body></body>
</html>
```

精灵图

使用精灵图的步骤是什么?

1. 创建一个盒子
2. 设置盒子大小为小图片大小
3. 设置精灵图为盒子的背景图片 background-image:精灵图地址
4. 将小图片左上角坐标 取负值，设置给盒子的background-position：x y;

建议遵循以下顺序：

1. 布局定位属性：display / position / float / clear / visibility / overflow（建议 display 第一个写，毕竟关系到模式）
2. 自身属性：width / height / margin / padding / border / background
3. 文本属性：color / font / text-decoration / text-align / vertical-align / white- space / break-word
4. 其他属性（CSS3）：content / cursor / border-radius / box-shadow / text-shadow / background:linear-gradient

清楚浮动：

1.直接设置父元素高度

2.额外标签法 在父元素内容的最后添加一个块级元素

```
.clearfix {
            /* 清除左右两侧浮动的影响 */
            clear: both;
        }
```

3.单伪元素清除法

```
/* 单伪元素清除浮动 和 额外标签法原理是一样的 */
        .clearfix::after {
            content: '';

            /* 伪元素添加的标签是行内，要求块 */
            display: block;
            clear: both;

            /* 为了兼容性 */
            height: 0;
            visibility: hidden;
        }
```

4.双为元素清楚法

```
        /* 清除浮动 */
        .clearfix::before,
        .clearfix::after {
            content: '';
            display: table;
        }
        /* 真正清除浮动的标签 */
        .clearfix::after {
            /* content: '';
            display: table; */
            clear: both;
        }
```

5.给父元素设置 `overflow : hidden`

字体图标

icomoon 字库 http://icomoon.io

阿里 iconfont 字库 http://www.iconfont.cn/

1. 将字体图标的font文件夹放入与html同一个文件 将css 中 @font-face复制到html中的style里
2. 使用标签改字体

```
span{
     font-family: '字体图标名字';
}
<span>□</span>
```

3.使用：： before/after 伪元素 添加

```
div::after {
           font-family: '字体图标名字';
           content: '□';   ①
          content: '\e91e';  ②
          }
```

什么网页

网页是图片、链接、文字、声音、视频等元素组成, 其实就是一个html文件(后缀名为html)

网页生成

网页生成制作: 有前端人员书写 HTML 文件, 然后浏览器打开,就能看到了网页.

HTML是什么

HTML: 超文本标记语言, 用来制作网页的一门语言. 有标签组成的. 比如 图片标签 链接标签 视频标签等...

什么是WEB标准

Web 标准是由 W3C 组织和其他标准化组织制定的一系列标准的集合。W3C（万维网联盟）是国际最著名的标准化组织

WEB结构

结构写到 HTML 文件中， 表现写到 CSS 文件中， 行为写到 JavaScript 文件中。

基本语法概述

1. HTML 标签是由尖括号包围的关键词，例如 < html>。
2. HTML 标签通常是成对出现的，例如 < html> 和 < /html> ，我们称为双标签。标签对中的第一个标签是 开始标签，第二个标签是结束标签。
3. 有些特殊的标签必须是单个标签（极少情况），例如 < br />，我们称为单标签。

常用标签

标题标签 < h1> - < h6>（重要）

段落 < p> < /p> 换行标签 < br />、水平线 < hr>

文本格式化标签 后者语义强 b strong 加粗 双标记 i em 倾斜 双标 u ins 下划线 双标 s del 删除线 双标

图片标签

- < img src="图片路径" alt="替换文本" title="提示文本" width="宽度" height="高度" border="边框">

  align属性 top上 middle 中 下 bottom left 左 right 右 相对路径：以引用文件所在位置为参考基础，而建立出的目录路径。 绝对路径：是指目录下的绝对位置，直接到达目标位置，通常是从盘符开始的路径。 同级目录：直接写：目标文件名字！ 下级目录：直接写：文件夹名/目标文件名字！ 上级目录：直接下：../目标文件名字！

超链接： < a href="跳转目标" target="目标窗口的弹出方式"> 文本或图像 < /a> _self默认 _ blank为新窗口打开

链接分类：

- 外部链接: 例如 < a href="http:// www.baidu.com "> 百度 < /a>。
- 内部链接:网站内部页面之间的相互链接. 直接链接内部页面名称即可，例如 < a href="index.html"> 首页 < /a>。
- 空链接: 如果当时没有确定链接目标时， < a href="#"> 首页 < /a> 。
- 下载链接: 如果 href 里面地址是一个文件或者压缩包，会下载这个文件。
- 网页元素链接: 在网页中的各种网页元素，如文本、图像、表格、音频、视频等都可以添加超链接.
- 锚点链接: 点我们点击链接,可以快速定位到页面中的某个位置.
- 超链接标签 (重点)
- 在链接文本的 href 属性中，设置属性值为 #名字 的形式，
- 如 < a href="#two"> 第2集 < /a>)
- 找到目标位置标签，里面添加一个 id 属性 = 刚才的名字 ，
- 如： < h3 id="two">第2集介绍 < /h3>

HTML中的注释以“ < !--”开头，以“ -->”结束。

特殊字符：空格& nbsp; 小于号& lt; 大于号& gt; 版权& copy; 上标&sup ; 下标&sub ; 度& deg;

音频标签

- src：地址 controls：控制播放控件 autoplay：自动播放(部分浏览器不支持) loop：循环播放 source可以让浏览器自己选择播放的格式 mp3和ogg

```
< audio controls="controls">

    < source src="音频地址" type="audio/mp3">

    < source src="音频地址" type="audio/audio/ogg">

     -你的浏览器不支持audio

< /audio>

```

视频标签

- src：地址 controls：控制播放控件 autoplay：自动播放(谷歌需要配合muted实现静音播放) loop：循环播放 poster 显示图片
- source可以让浏览器自己选择播放的格式 mp4和ogg

```
< video controls="controls">

< source src="音频地址.ogg" type="avideo/ogg">

< source src="音频地址.mp4" type="video/mp4">

你的浏览器不支持video

< /video>

```

列表

无序列表

- 无序列表由几个标签组成？分别表示什么？
- 　　ul标签：表示无序列表的整体
- 　　li标签：表示无序列表的每一项
- 无序列表标签的嵌套规范是什么？
- type属性：circle 圆 | square 方 | disc点（默认）
- 　　ul标签中只允许嵌套li标签
- 　　li标签中可以嵌套任意内容

有序列表

- 有序列表由几个标签组成？分别表示什么？
- type=排列方式 | start 默认从几开始 | reversed 倒序
- 　　ol标签：表示无序列表的整体
- 　　li标签：表示无序列表的每一项
- 有序列表标签的嵌套规范是什么？
- 　　ol标签中只允许嵌套li标签
- 　　li标签中可以嵌套任意内容

自定义列表（dl）

结构：dl>dt（名词）>dd（解释名词）用于内容解释

表格标签

- 完成一个简单的表格，需要由几个标签组成？分别表示什么？
- table标签：表格整体
- tr标签：表格每行
- td标签：对于主题的每一项内容
- 表格基本标签的嵌套规范是什么？
- table > tr > td

表格属性

- border属性：表格边框
- width属性：表格整体的宽度
- height属性：表格整体的高度
- 表格整体大标题：caption标签·书写在table标签内部
- 表头单元格：th标签·书写在tr标签内部（用于替换td标签）
- 表格的结构标签分别有哪些？表示什么含义？
- thead：表格头部
- tbody：表格主体
- tfoot：表格底部
- 表格结构标签书写在什么位置？
- 表格结构标签写在table标签内部
- 表格标签内部用于包裹tr标签

合并单元格

- • rowspan：跨行合并→垂直方向合并
- • colspan：跨列合并→水平方向合并
- cellspacing 单元格与单元格边框的边距
- cellpadding 单元格内容与单元格边框的边距

表单 form

表单域：action服务器地址 method发送表单数据时使用的网络请求方法：get/post name 用来区分多个表单

input标签属性

- text:文本框;password：密码; radio：单选框; checkbox：多选框;
- file:文件;submit:提交按钮;reset:重置按钮 button：普通按钮

htm5表单新增

- type=email 邮箱 url 地址 date 日期 time 时间 month 月 week 周 number 数字 tel手机号 search搜索框 color颜色 autofocus 自动获取焦点 autocomplete 显示之前搜索的值 默认off /打开on

```
<!-- 我们验证的时候必须添加form表单域 -->
<form action="">
    <ul>
        <li>邮箱：<input type="email" /></li>
        <li>网址：<input type="url" /></li>
        <li>日期：<input type="date" /></li>
        <li>时间：<input type="time" /></li>
        <li>数量：<input type="number" /></li>
        <li>手机号码：<input type="tel" /></li>
        <li>搜索：<input type="search" /></li>
        <li>颜色：<input type="color" /></li>
        <!-- 当我们点击提交按钮就可以验证表单了 -->
        <li> <input type="submit" value="提交"></li>
    </ul>
</form>
```

常用属性

- placeholder 提示用户输入内容
- value属性：用户输入的内容，提交之后会发送给后端服务器
- name属性：当前控件的含义，提交之后可以告诉后端发送过去的数据是什么含义
- 后端接收到数据的格式是：name的属性值 = value的属性值
  • name属性对于单选框有分组功能 • 有相同name属性值的单选框为一组，一组中只能同时有一个被选中
- checkbox：默认选中
- multiple：上传多个文件

注意点：

- • 如果需要实现以上按钮功能，需要配合form标签使用
- • form使用方法：用form标签把表单标签一起包裹起来即可

- ## button标签

- type属性值（同input的按钮系列）submit reset button(配合js使用)

select标签

- select标签：下拉菜单的整体
- option标签：下拉菜单的每一项
- 常见属性：
- selected：下拉菜单的默认选中

textarea标签

- 场景：在网页中提供可输入多行文本的表单控件
- 标签名：textarea
- 常见属性：
- cols：规定了文本域内可见宽度
- rows：规定了文本域内可见行数

label标签

- 场景：常用于绑定内容与表单标签的关系
- 标签名：label
- 使用方法①：
- 使用label标签把内容（如：文本）包裹起来
- 在表单标签上添加id属性
- 在label标签的for属性中设置对应的id属性值
- 使用方法②：
- 直接使用label标签把内容（如：文本）和表单标签一起包裹起来
- 需要把label标签的for属性删除即可

补充

- readonly="readonly"：设置文本输入框为只读（不能编辑）
- disabled="disabled"：控件属于非激活状态

- required 必须填写
- pattern 定义规则

语义化标签

无语义标签

- 常用于布局的无语义标签有哪两个？各自的特点有哪些?
- div：独占一行
- span：一行中可以显示多

有语义标签

- • header：网页头部
- • nav：网页导航
- • footer：网页底部
- • aside：网页侧边栏
- • section：网页区块
- • article：网页文章

```
<div class=“header”> </div>

<div class=“nav”> </div>

<div class=“content”> </div>

<div class=“footer”> </div>

```

css初识

- css中文名层叠样式表，作用是给页面中的html标签设置样式
- css语法 选择器{属性名：属性值}
- 三种引入：内嵌式 在style标签里,外联式：写在css文件里 用link引入，行内式：在标签里的style属性里

选择器

1. 标签选择器：标签名 {css属性名：属性值}
2. 类名选择器：.类名 {css属性名：属性值} 标签通过class使用
3. id选择器：#id名 {css属性名：属性值} 标签通过id使用 一般配合js使用
4. 通配符名选择器：*{css属性名：属性值} 所有标签选中

字体样式

- 字体大小： font-size：数字+px
- 字体粗细：font-weight正常：normal 或 400 • 加粗：bold 或 700
- ➢ 字体样式：font-style 正常：normal• 倾斜：italic
- ➢ 字体系列：font-family 具体字体1,具体字体2,具体字体3,具体字体4,...,字体系列
- ➢ 字体连写：font : style weight size family;

文本样式

- 文本缩进：text-indent：数字+px 数字+em（推荐：1em = 当前标签的font-size的大小）
- 文本水平对齐方式：text-align：letf center right
- 文本修饰：text-decoration：underline下划线，line-through 删除线， overline上划线，none无线 开发中会使用 text-decoration : none ; 清除a标签默认的下划线
- text-align : center 能让哪些元素水平居中?

  1. 文本
  2. span标签、a标签
  3. input标签、img标签
  4. 注意点：

  如果需要让以上元素垂直平居中 需要给以上元素的 父元素 设置行高：line-height

  让单行文本垂直居中可以设置 line-height : 文字父元素高度

  网页精准布局时，会设置 line-height : 1 可以取消上下间距

  font : style weight size/line-height family ;

补充

- 字母之间的间距letter-spacing
- 单词之间间距word-spacing
- 文本的大小写text-transform
- 文本的装饰text-decoration
- 自动换行word-wrap

颜色

文字：color

背景background-color

方法rgb rgba(a为透明度0~1) 十六进制

复合选择器

后代选择器：空格

选择器语法：选择器1 选择器2 { css }

子代选择器：>

选择器语法：选择器1 > 选择器2 { css }

并集选择器：，

选择器语法：选择器1 ， 选择器2 { css }

交集选择器：紧挨着

选择器语法：选择器1选择器2 { css }

伪类选择器：

- 选择器：hover 鼠标悬停 显示内容
- ： link 未访问链
- : visited 已经访问
- : active活动链接

焦点伪类选择器：

input：focus{}

结构伪类选择器

- E:first-child/last-child/nth-child(n)/nth-last-child(){}
- n为：0、1、2/偶数2n even/奇数 2n+1/-1 odd/找到前五个 -n+5/后五个n+5
- E:nth-of-type(n){}
- •:nth-child→ 直接在所有孩子中数个数（旗下所有孩子男女都有）
- •:nth-of-type→ 先通过该 类型 找到符合的一堆子元素，然后在这一堆子元素中数个数（只有旗下的男孩）

```
 <style>
    ul li:first-child{
      background-color: red;
    }
  </style>

  <ul>
    <li>列表项一</li> 选中他
    <li>列表项二</li>
    <li>列表项三</li>
    <li>列表项四</li>
  </ul>

<style>
    ul li:nth-child(2){     选中 ul 的第二个孩子 必须为li
      /* 字体变成红色 */
        color: red;
    }

    ul li:nth-of-type(2){       选中ul 中第二个孩子 li
      /* 背景变成绿色 */
      background-color: green;
    }
```

```
  </style>


  <ul>
    <li>列表项一</li>
    <p>乱来的p标签</p> 没被选中
    <li>列表项二</li>绿色
    <li>列表项三</li>
    <li>列表项四</li>
  </ul>
```

伪元素

- ::before // after
- \1. 必须设置content属性才能生效
- \2. 伪元素默认是行内元素

属性选择器

- ~具有这个属性值的标签--^以什么开头-- $结尾--*含指定字符串--|指定属性值为含他的或以他为开头
- E[arr]选择arr属性的E元素
- E[arr="var"] 选择arr为var的E元素

```
 /* 只选择 type =text 文本框的input 选取出来 */
input[type=text] {
    color: pink;
}
/* 选择首先是div 然后 具有class属性 并且属性值 必须是 icon开头的这些元素 */
div[class^=icon] {
    color: red;
}
/* 选择首先是section 然后 具有class属性 并且属性值 必须是 data结尾的这些元素 */
section[class$=data] {
    color: blue;
}
```

背景（以下重点）

1. 背景颜色：background-color（bgc）
2. 背景图片：background-image（bgi）
3. 背景固定：background-attachment：scroll 随页面滚动 fixed 固定
4. 背景大小：background-size:宽度 高度；contain等比 不超出盒子的最大值 cover覆盖满盒子没有恐怖
5. ==线性渐变：background-image：linear-gradient（方向：0deg，颜色1，颜色2）=
6. 重复线性渐变（repeating-linear-gradient（方向：0deg，颜色1，颜色2））
7. 径向background-image：radial-gradient（渐变形状 圆心位置 at+关键字，颜色1，颜色2）
8. 重复径向渐变（repeating-radial-gradient（渐变形状 圆心位置 at+关键字，颜色1，颜色2
9. 背景平铺：：background-repeat（bgr）：repeat 默认水平和垂直都平铺 no-repeat 不平铺 repeat-x/y 水平/垂直平铺
10. 背景位置：background-position（bgp）：left center right top center bottom 数字+px(x轴 y轴)

11. 背景连写：background：color image repeat position

- 如果需要设置单独的样式和连写

1. 要么把单独的样式写在连写的下面
2. 要么把单独的样式写在连写的里面

元素显示模式

块级元素

- 独占一行（一行只能显示一个）
- 宽度默认是父元素的宽度，高度默认由内容撑开
- 可以设置宽高

行内元素

- 一行可以显示多个
- 宽度和高度默认由内容撑开
- 不可以设置宽高

行内块元素

- 一行可以显示多个
- 可以设置宽高

元素模式转换：

- display：block转换为块级元素 inline-block行内块 inline 行内
- p标签中不要嵌套div、p、h等块级元素
- a标签不能嵌套a标签
- css具有继承性、层叠性、优先级
- 子元素有默认继承父元素样式的特点

优先级排序

- 继承 < 通配符选择器 < 标签选择器 < 类选择器 < id选择器 < 行内样式 < !important
- !important写在属性值的后面，分号的前面！
- !important不能提升继承的优先级，只要是继承优先级最低！
- 实际开发中不建议使用 !important 。
- !important如果不是继承，则权重最高，天下第一！

盒子模型

盒子模型一共有几个部分组成？分别是什么？

1. 内容区域：content
2. .边框区域：border
3. 内边距区域：padding

4. 外边距区域：margin

边框

- 边框：border-width粗细/style样式/color颜色 （solid实线，dashed 虚线 ，点线 dotted）

  属性值：单个取值的连写，取值之间以空格隔开 如：`border : 10px solid red;`
- 属性名：border - 方位名词 属性值：连写的取值
- ➢ 给盒子设置四周 20像素、实线、蓝色的边框，属性应该如何设置？

  `border: 20px solid blue;`

  ➢ 给盒子设置上边框 10像素、虚线、黄色的边框，属性应该如何设置？

  `border-top: 10px dashed yellow;`
- 设置边框图片：`border-image-source: url(images/1.jpg);`
- 图片边框偏移：`border-image-slice: 60;`
- 图片边框宽度：`border-image-width: 30px;`
- 图片边框：`border-image`
- 属性值：source 图片路径 slice 偏移量 width 边框宽度 outset 边框向外延申距离 repeat 平铺方式

内边距

内边距：padding (外边距 margin 方法与其同)

1. 一个值：全部 两个值：上下 左右 三个值：上 左右 下 四个值：上 右 下 左
2. 属性名：padding - 方位名词 属性值：数字 + px

   注意： 水平方向的margin和padding布局中有效 垂直方向的margin和padding布局中无效

盒子阴影

- `box-shadow: h-shadow v-shadow blur spread color inset;`
- h/v必需 水平/垂直 b 模糊度 s尺寸 c颜色 i内部阴影 outset 外部

文字阴影

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

如果不想盒子被撑大?

- ➢ 解决方法 ①： 手动内减
  - 操作：自己计算多余大小，手动在内容中减去
  - 缺点：项目中计算量太大，很麻烦
- ➢ 解决方法 ②： 自动内减
  - 操作：给盒子设置属性 box-sizing : border-box ;即可
  - 优点：浏览器会自动计算多余大小，自动在内容中减去
- 标准流：又称文档流，是浏览器在渲染显示网页内容时默认采用的一套排版规则，规定了应该以何种方式排列元素

- ➢ 常见标准流排版规则： 块级元素：从上往下，垂直布局，独占一行 行内元素 行内块元素：从左往右，水平布局，空间不够自动折行

浮动

`float:left/right`

➢ 浮动元素的特点有哪些?

- 浮动元素会脱标，在标准流中不占位置
- 浮动元素比标准流高出半个级别，可以覆盖标准流中的元素
- 浮动找浮动，下一个浮动元素会在上一个浮动元素后面左右浮动
- 浮动元素有特殊的显示效果：① 一行可以显示多个 ② 可以设置宽高
- 浮动的元素不能通过text-align:center或者margin:0 auto

定位

`position: relative/absolute/fixed`

静态定位

- 静态定位就是之前标准流，不能通过方位属性进行移动
- 之后说的定位不包括静态定位，一般特指后几种：相对、绝对、固定

相对定位

- 需要配合方位属性实现移动
- 相对于自己原来位置进行移动
- 在页面中占位置 → 没有脱标

绝对定位

- 需要配合方位属性实现移动
- 默认相对于浏览器可视区域进行移动
- 在页面中不占位置 → 已经脱标

子绝父相

- 子元素：绝对定位
- 父元素：相对定位
- 父元素是相对定位，则对网页布局影响最小

➢ 使用子绝父相水平垂直居中的操作是什么?

1. 子绝父相 left：50%； top：50%；
2. transform：translate（-50%，-50%）；//位移自己宽度高度的一半

固定定位

- 需要配合方位属性实现移动

- 相对于浏览器可视区域进行移动
- 在页面中不占位置 → 已经脱标

不同布局方式元素的层级关系：

标准流 < 浮动 < 定位

不同定位之间的层级关系：

- 相对、绝对、固定默认层级相同
- 此时HTML中写在下面的元素层级更高，会覆盖上面的元素
- z-index：数字；修改定位元素层级

装饰

垂直对齐：vertical-align

属性值：baseline 默认 基线对齐 top 顶部 middle 中部 bottom底部

- 可以解决的问题
- 文本框和表单按钮无法对齐问题
- input和img无法对齐问题div中的文本框，文本框无法贴顶问题
- div不设高度由img标签撑开，此时img标签下面会存在额外间隙问题
- 使用line-height让img标签垂直居中问题

=光标类型 cursor==

属性值：default 箭头 pointer小手 text 工字型 move 十字光标

防止拖拽文本域 resize

textarea{ resize: none;}

边框圆角：

border-radius 属性值：数字+px 百分比

溢出部分显示效果

属性值：visible 溢出可见 hidden 溢出隐藏 scroll 显示滚动条 auto 自动显示滚动条

元素本身隐藏

visibility：hidden ； 占位置

display：none；不占位置 display：block； （显示）

边框合并

border-collapse：collapse；

过度

```
/* 过渡配合hover使用，谁变化谁加过渡属性 */
    .box {
        width: 200px;
        height: 200px;
        background-color: pink;
        /* 宽度200，悬停的时候宽度600，花费1秒钟 */
        /* transition: width 1s, background-color 2s; */
        /* 如果变化的属性多，直接写all,表示所有 */
        transition: all 1s;
    }
    .box:hover {
        width: 600px;
        background-color: red;
    }*/

```

CSS 三角

```
.box3 {
            /* 宽度 高度为0 */
            width: 0;
            height: 0;
            /*居中*/
            margin: 100px auto;
            /*20像素 实线的透明色*/
            border: 20px solid transparent;
            /*右边框为红色*/
            border-right-color: red;
        }
```

CSS 2D转换

转换属性：transform

- 移动：translate transform:translate(x,y) transform:translateX/Y(n)

  不影响其他元素的位置,百分比是自己宽度高度的多少，行内标签无用 旋转：rotate transform:rotate(0deg) 角度为正 顺时针，负 逆时针; 中心点为元素中心点 360deg时，可以给本身添加过渡transition 缩放：scale transform:scale(x,y) 两个参数：宽和高放大;一个参数：第二个参数和第一个参数一样;0.5是缩小 优势：可以改变中心角transform-origin：方位词 方位词

注意:

- 1.同时使用多个转换，其格式为:transform: translate() rotate() scale() ..等，
- 2.其顺序会影转换的效果。（先旋转会改变坐标轴方向)
- 3.当我们同时有位移和其他属性的时候，记得要将位移放到最前
- 转换transform我们简单理解就是变形有2D和3D之分我们暂且学了三个分别是位移旋转和缩放

- 2D移动translate(x, y)最大的优势是不影响其他盒子，里面参数用%，是相对于自身宽度和高度来计算的可以分开写比如translateX(x)和translateY(y)
- 2D旋转rotate(度数)可以实现旋转元素度数的单位是deg
- 2D缩放sacle(x,y)里面参数是数字不跟单位可以是小数最大的优势不影响其他盒子设置转换中心点transform-origin : x y;参数可以百分比、像素或者是方位名词
- 当我们进行综合写法，同时有位移和其他属性的时候，记得要将位移放到最前

CSS 动画

制作动画分两步

1. 先定义动画

- @keyframes 定义动画
- @keyframes 动画名称{ 0%{} 100%{}}
- 0%是动画的开始，100%是动画的完成。这样的规则就是动画序列。
- 在@keyffames中规定某项CSS样式，就能创建由当前样式逐渐改为新样式的动画效果。
- 动画是使元素从一种样式逐渐变化为另一种样式的效果。您可以改变任意多的样式任意多的次数。请用百分比来规定变化发生的时间，或用关键词"from"和"to”，等同于0%和100%.

2. 在使用(调用)动画

- animation-name：动画名称
- animation-duration：持续时间

动画属性

鼠标经过div 让这个div停止动画，鼠标离开就继续动画! `Ianimation-play-state: paused;`

简写

- animation :动画名称--持续时间--运动曲线--何时开始--播放次数--是否反方向--动画起始或者结束的状态;
- animation: name--duration（数字+s）-- timing-function(linear匀速 ease慢到快 steps步长) --delay（数字+s）-- iteration-count（ iteration重复的 conut次数 (数字) infinite无限）-- direction（*normal（向前）默认值 reverse （向后） alternate 先向前 后向后alternate-reverse 先向后后向前）--fill-mode（none 无 forwards 最后 backwards 最开始 both都保存）；
- 常写： animation: name 1.4s linear infinite
- 简写前两个必须写，属性里面不包含animation-play-state
- 暂停动画: animation-play-state: puased;经常和鼠标经过等其他配合使用
- 想要动画走回来，而不是直接跳回来: animation-direction : alternate
- 盒子动画结束后，停在结束位置: animation-fill-mode : forwards

CSS 3D转换

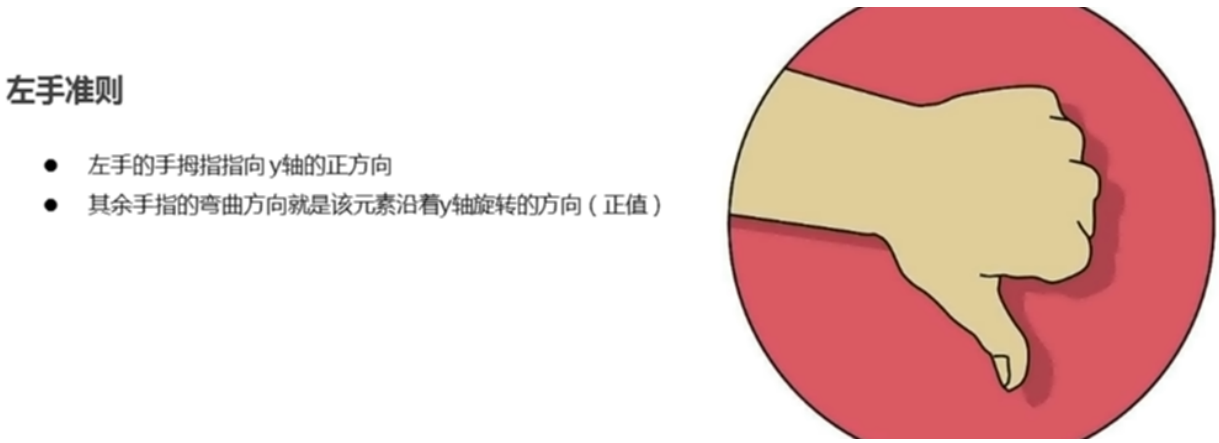
三位坐标

- x轴：水平向右，注意：x右边是正值，左边为负值
- y轴：垂直向下，注意：y下面是正值，y上面为负值
- z轴：垂直屏幕，注意：往外是正值，往里面为负值

主要知识

- 3D位移：translate3d(x,y,z);可以分开写，同2d转换
- 3D旋转：rotate3d(x（1）,y（0）,z（0）,deg（45deg）)deg为自定义轴旋转，可以分开写，同2d转换
- rotate3d(1,1,0,45deg)沿着对角线旋转

左手准则



透视：perspective

用来实现3d效果，透视值越小越大。给父元素添加，位移的Z轴的正值越大物体越大

3D呈现

transfrom-style flat 不开启，preserve-3d 开启

javascript

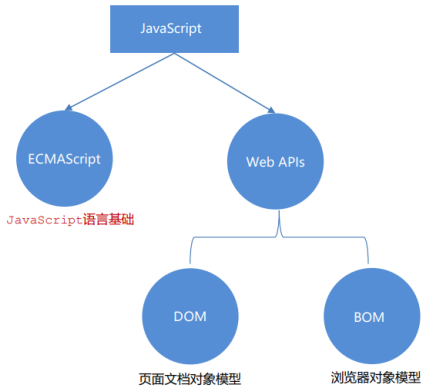
JavaScript是什么?

- JavaScript：运行在客户端（浏览器）的编程语言，实现人机交互效果
- 网页特效 表单验证 数据交互 服务端编程 node.js
- **组成： ECMAScript webAPIs(DOM BOM)**

- ECMAScriptL:规定了js基础语法核心知识
  - DOM 操作文档
  - BOM 操作浏览器，比如页面弹窗

1. 作用和分类

- 作用：就是使用 JS 去操作 html 和浏览器
- 分类：DOM (文档对象模型)、BOM（浏览器对象模型）



书写位置： 内联 内部 外部

内部:

- 规范：**script****标签写在`</body>`上面**
- 拓展： alert('你好，js') 页面弹出警告对话框
- 我们将`< script>`放在HTML文件的底部附近的原因是浏览器会按照代码在文件中的顺序加载HTML。
- 如果先加载的 JavaScript 期望修改其下方的 HTML，那么它可能由于 HTML 尚未被加载而失效。
- 因此，将 JavaScript 代码放在 HTML页面的底部附近通常是最好的策略。

外部:

- 代码写在以.js结尾的文件里 引入方式`<script src="“></script>`
- 1. script标签中间无需写代码，否则会被忽略!
- 2. 外部JavaScript会使代码更加有序，更易于复用，
- 且没有了脚本的混合，HTML 也会更加易读，因此这是个好的习惯。

内联:

代码写在标签内部`< button onclick="alert('你好js')">< /button>`

```
<!-- 内联 -->
    <button onclick="alert('月薪过万')">点击我</button>
    <!-- 内部js -->
    <script>
        // alert('你好，  js')
    </script>
    <!-- 外部js -->
    <script src="./my.js">
        alert(11)
    </script>
```

注释

- JavaScript 注释有那两种方式？单行注释 //　　多行注释 /* */
  JavaScript 结束符注意点 结束符是？　　分号；
  结束符可以省略吗？ 因为 js中换行符（回车）会被识别成结束符　但为了风格统一，要写每句都写，要么每句都不写

输入输出语句

- 输入： prompt()
- 输出： alert()　输出的内容为标签 想body输出内容
- document.write() 页面弹出警告框
- console.log() 控制太语法 程序员调试使用

字面量

- 字面量（literal）是在计算机中描述 事/物
- 如： [] 数组字面量 {} 对象字面量　100 数字字面量

变量

- 变量：装数据的容器

  声明变量有两部分构成：声明关键字、变量名（标识）　Let 变量名/标识符
- let 不允许多次声明一个变量。

能够说出变量的本质是什么

- 内存：计算机中存储数据的地方，相当于一个空间

  变量：是程序在内存中申请的一块用来存放数据的小空间

规则：

1. 不能用关键字
2. 只能用下划线、字母、数字、$组成，且数字不能开头
3. 字母严格区分大小写，如 Age 和 age 是不同的变量

规范：

1. 起名要有意义
2. 遵守小驼峰命名法
3. 第一个单词首字母小写，后面每个单词首字母大写。例：userName

三种变量

全局变量 js标签内 局部变量 函数内 块级变量 {}

采取就近原则的方式来查找变量最终的值

数组

声明: let 数组名 = []

数组是按顺序保存，所以每个数据都有自己的编号

1. 计算机中的编号从0开始
2. 在数组中，数据的编号也叫索引或下标
3. 数组可以存储任意类型的数据

取值 数组名[下标]

计算机程序可以处理大量的数据，为什么要给数据分类?

- 更加充分和高效的利用内存
- 也更加方便程序员的使用数据

数组操作

- 数组长度 a.length
- 查 数组[下标]
- 增 arr.push(新增的内容) arr.unshift(新增的内容)
- 改 数组[下标] = 新值
- 删 arr.pop() arr.shift() arr.splice(操作的下标,删除的个数)
- 数组.push() 方法将一个或多个元素添加到数组的末尾，并返回该数组的新长度 (重点)
- 数组. pop() 方法从数组中删除最后一个元素，并返回该元素的值
- 数组. shift() 方法从数组中删除第一个元素，并返回该元素的值
- 数组. splice(起始位置，删除几个元素) 方法 删除指定元素

```
<script>
        let arr = ['pink', 'hotpink', 'deeppink']
        // 访问 / 查询
        console.log(arr[0])
        // 2. 改
        arr[0] = 'lightpink'
    //增 push   unshift
    arr.push('blue', 'skyblue')
    arr.unshift('pink', 'blue')
    //删  pop   splice
        arr.pop()    // 删除最后一个元素
    arr.splice(1, 1)  从下标为1起  删一个

    </script>
```

基本数据类型

number 数字型 string 字符串型 boolean 布尔型 undefined 未定义型 null 空类型

引用数据类型

object 对象 function 函数 array 数组

注意：

- JavaScript 中的正数、负数、小数等 统一称为 数字类型。
- JS 是弱数据类型，变量到底属于那种类型，只有赋值之后，我们才能确认
- Java是强数据类型 例如 int a = 3 必须是整数

字符串

- 通过单引号（'）、双引号（""）或反引号包裹的数据都叫字符串，单引号和双引号没有本质上的区别，
- 推荐使用单引号。外双内单，或者外单内双
- 拼接 +号　在反引号前前提下使用变量${变量名} 不需要拼接号

```
<script>
        //+ 号拼接
        let uname = prompt('请输入您的名字')
        document.write('我的名字是：' + uname)
        //反引号 字符串
        let age = 81
        document.write(`我今年${age - 20}岁了`)
        document.write(`
        <div>123</div>
        <p>abc</p>
        `)
    </script>
```

布尔数据类型有几个值？true 和 false

什么时候出现未定义数据类型？以后开发场景是？定义变量未给值 如果检测变量是undefined就说明没有值传递

null是什么类型？ 开发场景是？ 空类型 如果一个变量里面确定存放的是对象，如果还没准备好对象，可以放个null

控制台语句经常用于测试结果来使用。数字型和布尔型颜色为蓝色，字符串和undefined颜色为灰色 通过 typeof 检测数据类型

转换数据

Number(数据)

- 如果字符串内容里有非数字，转换失败时结果为NaN（Not a Number）即不是一个数字
- NaN也是number类型的数据，代表非数字

parseInt(数据)

只保留整数

parseFloat(数据)

可以保留小数

转换为字符型:

String(数据) 变量.toString(进制)

运算符

1. 算术运算符：+ - * / % 优先级 先乘除后加减，有括号先算括号里面的
2. 赋值运算符：+= -= *= /= %= = 将等号右边的值赋予给左边,要求左边必须是一个容器变量
3. 一元运算符 -- ++ 、二元运算符 、三元运算符 ？ ：
4. 比较运算符：> < >= <= == === != = 只会得到true或false

   注意：NaN不等于任何值，包括它本身

   = 和 == 和 === 怎么区别？ = 是赋值 == 是判断 只要求值相等，不要求数据类型一样即可返回true === 是全等 要求值和数据类型都一样返回的才是true 开发中，请使用 ===
5. 逻辑运算符：1. 左边为假短路 &&与 一假则假 2. 左边为真则 短路 ||或 一真则真 3.！非 真变假 假变真真

```
let a = 3 > 5 && 2 < 7 && 3 == 4
console.log(a);

let b = 3 <= 4 || 3 > 1 || 3 != 2
console.log(b);

let c = 2 === "2"
console.log(c);

let d = !c || b && a
console.log(d);
```

分支语句

- if分支语句 三元运算符 条件？ 满足条件代码： 不满足代码 switch 语句
- switch case语句一般用于等值判断,不适合于区间判断
- switch case一般需要配合break关键字使用 没有break会造成case穿透
- continue：结束本次循环，继续下次循环 break：跳出所在的循环

```
<script>
     //if语句
      if () { } else if () {} else()
    //switch
    switch (2) {
            case 1:
                alert(1)
                break

            case 2:
```

```
                alert(2)
                break
            case 3:
                alert(3)
                break
            default:
                alert('没有数据')
        }

        // 1. 用户输入数字
        let num = prompt('请您输入一个数字')
        // 2. 判断条件是 小于 10  则数字前面 + '0'   01 否则 不补
        // let t = num >= 10 ? num : '0' + num
        let t = num < 10 ? '0' + num : num
        document.write(t)
    </script>
```

循环语句

for循环和while循环有什么区别呢：

1. 当如果明确了循环的次数的时候推荐使用for循环
2. 当不明确循环的次数的时候推荐使用while循环
3. 　　外层循环一次，内层循环全部

```
<script>
        // 循环必须有3要素
        // 变量的起始值
        let i = 1
        // 终止条件
        while (i <= 3) {
            document.write(`月薪过万 <br>`)
            // 变量变化
            i++
        }
   //for循环
    for (let i = 1; i <= 10; i++) {
            document.write('月薪过万 <br>')
        }
      // 记忆单词案例
        // 分析
        // 1. 外面的循环 记录第n天
        for (let i = 1; i < 4; i++) {
            document.write(`第${i}天 <br>`)
            // 2. 里层的循环记录 几个单词
            for (let j = 1; j < 6; j++) {
                document.write(`记住第${j}个单词<br>`)
            }
        }

    </script>

```

函数

关键词 function

- 格式 function 函数名（）{ } 调用 函数名()
- 形参：声明函数时写在函数名右边小括号里的叫形参（形式上的参数）
- 实参：调用函数时写在函数名右边小括号里的叫实参（实际上的参数）
  尽量保持形参和实参个数一致

return关键词

- return后面不接数据或者函数内不写return，函数的返回值是undefined
  return能立即结束当前函数, 所以 return 后面的数据不要换行写

立即执行函数

- 有什么作用？ 防止变量污染
- 立即执行函数需要调用吗？ 无需调用，立即执行，其实本质已经调用了
- 有什么注意事项呢？ 多个立即执行函数之间用分号隔开

```
<script>
        //函数
        function getMax(x, y) {
    //返回值
            return x > y ? x : y
        }
        // 实参也可以放变量  传参
        let max = getMax(num1, num2)
         // 函数表达式
        let fn = function () {
            console.log(111)
        }
        fn()
     //立即执行函数
       (function () {
            console.log(111)
        })();
    </script>
```

对象

- 声明对象：let 对象名{ 方法 属性}
- 方法名：function(){}
- 属性和值用： 冒号 隔开
- 多个属性用， 逗号隔开
- 访问属性 对象.属性名 调用方法 对象.方法名()
- 改：对象.属性 = 值 对象.方法 = function() {}
- 增： 对象名.新属性名 = 新值

- 删： delete 对象名.属性名
- 对象如果有这个属性相当于重新赋值 对象如果没有这个属性相当于动态添加一个属性

遍历对象

- for in 循环语句
- 语法
- for (let k in 对象名) {} 重点
- k 变量 k 或者 key value

```
<script>
        // 声明人对象
        let person = {
            uname: '刘德华',
            age: 18,
            sex: '男',
            // 方法名: function(){}
            sayHi: function () {
                console.log('hi~~~')
            },
            mtv: function (s) {
                console.log(s)
            }
        }
        // console.log(uname)
        // 1. 访问属性  得到值   对象.属性名
        console.log(person.uname)
        console.log(person.age)
        // 2. 访问属性  得到值   对象['属性名']
        console.log(person['sex'])
        // 调用方法 对象.方法名()
        person.sayHi()
        person.mtv('无间道')
    //改
        person.uname = '马云'
        person.sayHi =function () {
                console.log('hhh~~~')
            }
        //删
        delete 对象名.属性名
        // document.write()

//遍历对象

        let obj = {
            uname: '小明',
            age: 18,
            sex: '男'
        }
        // for  in  循环语句
        // 语法
        // for (let k in 对象名) {}  重点
        // k 变量  k  或者  key  value
```

```
        for (let k in obj) {
            console.log(k)  // 属性名
            // console.log(obj.k)  // obj.k 意思是 obj里面的k属性
            // console.log(obj['k'])
            console.log(obj[k])  // 属性值

            // 为什么这么写?
            // k  ===  'uname'   === 'age'  === 'sex'
            // // obj.k
            // // obj['uname']
            // obj['sex']  === 18
        }
    </script>
```

内置数学对象

- random：生成0-1之间的随机数（包含0不包括1）
- ceil：向上取整 floor：向下取整 max：找最大数
- min：找最小数 pow：幂运算 abs：绝对值

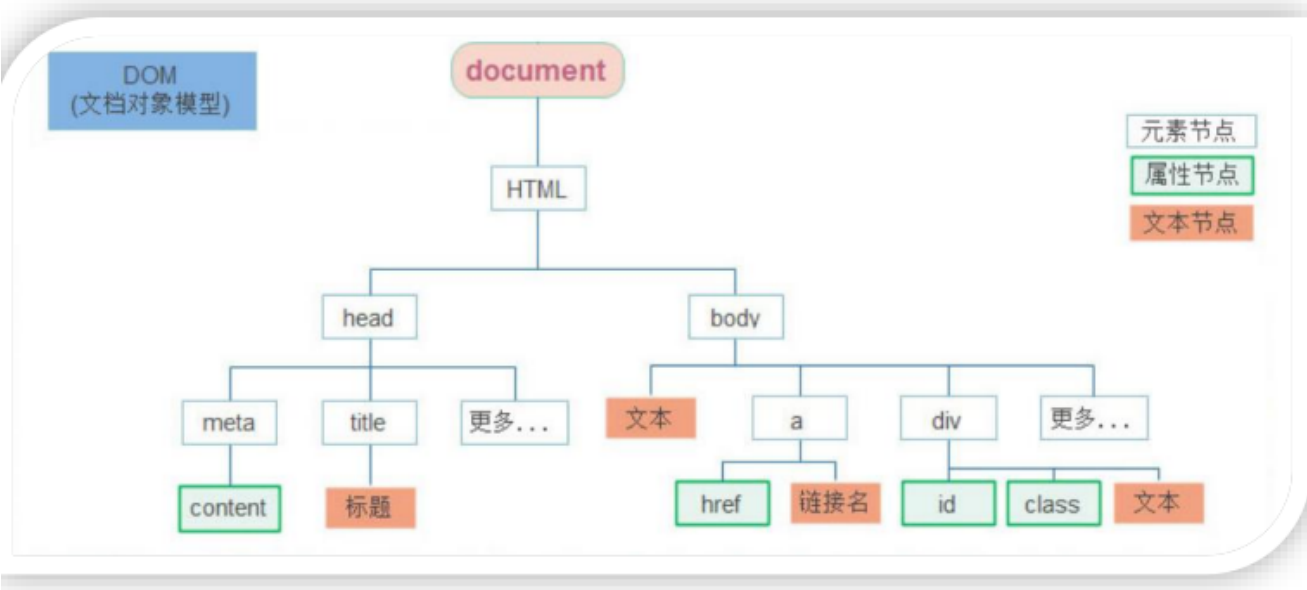
```
<script>
     console.log(Math.PI)  //  圆周率    π
        console.log(Math.random())  //  随机数  随机抽奖  随机点名
        // 返回的是小数  但是能得到 0  得不到 1
        // 向上取整  返回的整数
        console.log(Math.ceil(1.1))  // ceil  2
        console.log(Math.ceil(1.5))  // ceil  2
        console.log(Math.ceil(1.9))  // ceil  2
        // 向下取整  返回的整数  floor
        console.log(Math.floor(1.1))  // floor  1
        console.log(Math.floor(1.5))  // floor  1
        console.log(Math.floor(1.9))  // floor  1
        console.log('------------------------------')
        // round 就近取整( .5往大取证)  返回的整数
        console.log(Math.round(1.1))  // round  1
        console.log(Math.round(1.5))  // round  2
        console.log(Math.round(1.9))  // round  2
        console.log('------------------------------')
        console.log(Math.round(-1.1))  // round  -1
        console.log(Math.round(-1.5))  // round  -1
        console.log(Math.round(-1.9))  // round  -2
function getRandom(min, max) {
    //如何生成N-M之间的随机数
    //0-6   6+1  最大值 (6) 减 最小值 (0) +1 +最小值 (0)
    //1-6   5+1+1 最大值 (6) 减 最小值 (1) +1 +最小值 (1)
            return Math.floor(Math.random() * (max - min + 1)) + min
        }

      </script>
```

Web APIs

DOM :文本对象模型(操作网页内容)，开发网页内容特效和实现用户交互

DOM树

- 含义：将 HTML 文档以树状结构直观的表现出来，我们称之为文档树或 DOM 树
  作用：文档树直观的体现了标签与标签之间的关系



DOM对象

- 浏览器根据html标签生成的 JS对象（DOM对象）
- DOM的核心就是把内容当对象来处理

document 对象 document.write() 是用来访问和操作网页内容的

获取对象

1. document.querySelector('类名') 获取一个第一个
2. document.querySelectorAll('类名') 获取多个 伪数组 通过遍历的方式，获得里面的每一个dom对象（元素）

修该元素

1. 修改标签（元素）内容 box是对象 innerText 属性 innerHTML解析标签

```
<script>
      // 1. 获取标签(元素)
      let box = document.querySelector('div')
      // 2. 修改标签（元素）内容  box是对象   innerText 属性
      // 对象.属性  = 值    不识别标签

      // box.innerText = '有点意思~'
```

```
        // box.innerText = '<strong>有点意思~</strong>'
        // 3. innerHTML解析标签
        box.innerHTML = '<strong>有点意思~</strong>
          box.style.backgroundColor = 'hotpink'
         box.className = 'one active'
        // add是个方法 添加  追加
        box.classList.add('active')
        // remove() 移除 类
        box.classList.remove('one')
        // 切换类
        box.classList.toggle('one')

      input.value = '小米手机'
        input.type = 'password'
    </script>
```

1. 修改元素属性 对象.属性 = 值
2. 修改元素样式属性 对象.style.样式属性 = '值'
    - 注意：
        1. 修改样式通过style属性引出
        2. 如果属性有-连接符，需要转换为小驼峰命名法
        3. 赋值的时候，需要的时候不要忘记加css单位
3. 操作类 className 和 classList
    - className 会覆盖原有的类
    - classList add 添加类 remove 移除类 toggle 切换类（有 删除 无添加）
4. 修改表单元素 按钮关闭 disabled、 复选框默认选择 checked、 下拉默认选择 selected
    - // disabled 不可用 = false 这样可以让按钮启用
    - 其他两个 true 选中 false 未选

间歇函数（定时器）

```
<script>
       function show() {
           console.log('月薪过2万')
       }
      //创建定时器  setInterval(函数名，毫秒数)
    let timer = setInterval(show, 1000)
       // 清除定时器
       clearInterval(timer)
   </script>
```

事件监听

语法 事件源.addEventListener('事件', 事件处理函数) 第三个值 是否捕获了解

1. 事件：click 单击 dblclick双击 mouseenter 鼠标经过 mouseleave鼠标离开 foucus/blur 获得失去焦点 scroll 滚动事件 resize窗口尺寸事件 Keydown 键盘按下触发 Keyup 键盘抬起触发

2. 事件监听三要素是什么?

   事件源 (谁被触发了)

   事件 (用什么方式触发，点击还是鼠标经过等)

   事件处理程序 （要做什么事情）

```
<script>
        //1. 获取按钮元素
        let btn = document.querySelector('button')
        //2. 事件监听    绑定事件 注册事件 事件侦听
        // 事件源.addEventListener('事件', 事件处理函数)
        btn.addEventListener('click', function () {
            alert('月薪过万')
        })
    </script>
```

高阶函数

函数表达式

let fn = function(){} 函数表达式 为高级函数的一种形式

回调函数

function fn(){} setIntreval(fn,1000) 此时fn为回调函数 回头去调用的函数

环境对象

变量this：他就是个对象【谁调用， this就是谁】是判断this指向的粗略规则

`btn.addEventListener('click',function(){console.log(this)})` this指向btn

思想

排他思想

1. 干掉所有人　使用for循环
2. 复活他自己　通过this或者下标找到自己或者对应的元素

DOM节点

- DOM树里每一个内容都称之为节点
- 元素节点 如div标签
- 属性节点 如class属性
- 文本节点 如 文字

创建节点

创建节点：document.createElement('标签名')

查找结点

- 父节点：子元素.parentNode　　返回最近一级的父节点 找不到返回为null
- 子节点：父元素.children　仅获得所有元素节点 返回的还是一个伪数组
- 兄弟节点：nextElementSibling 下一个previousElementSibling 上一个

删除节点

- 删除节点：父元素.removeChlid(要删除的元素)　如不存在父子关系则删除不成功
- 删除节点和隐藏节点（display:none）有区别的：　隐藏节点还是存在的，但是删除，则从html中删除节点

追加节点：

- 插入到父元素的最后一个子元素　　父元素.appendChild(要插入的元素)
- 插入到父元素的某个子元素前面　　父元素.insertBefore（要插入的元素，在哪个元素前面）
- 注意：获取元素 类名要加点　追加类 类名不加点

克隆节点：

- 元素.cloneNode(布尔值)　cloneNode会克隆出一个跟原标签一样的元素，括号内传入布尔值
- 若为true，则代表克隆时会包含后代节点一起克隆
- 若为false，则代表克隆时不包含后代节点 默认为false

```
<body>
    <ul>
        <li>我是大毛</li>
        <li>我是二毛</li>
    </ul>
    <script>
        //获取节点
        let ul = document.querySelector('ul')
            // 1. 创建新的标签节点
        let li = document.createElement('li')
            // 删除节点
        ul.removeChild(ul.children[0])
        li.innerHTML = '我是xiao ming'
            // 2. 追加节点　父元素.appendChild(子元素) 后面追加
        ul.appendChild(li)
            // 3. 追加节点　父元素.insertBefore(子元素，　放到那个元素的前面)
        ul.insertBefore(li, ul.children[0])
            //4.将克隆的li节点　追加到ul第一个孩子前面
        ul.insertBefore(ul.cloneNode(li), ul.children[0])
    </script>
</body>

```

时间对象

- 获取当前时间let date = new Date()
- 获取指定时间let date = new Date('1949-10-01')

- 本地时间：new Date().toLocaleString()

方法	作用	说明
getFullYear()	获得年份	获取四位年份
getMonth()	获得月份	取值为 0 ~ 11
getDate()	获取月份中的每一天	不同月份取值也不相同
getDay()	获取星期	取值为 0 ~ 6
getHours()	获取小时	取值为 0 ~ 23
getMinutes()	获取分钟	取值为 0 ~ 59
getSeconds()	获取秒	取值为 0 ~ 59

什么是时间戳

是指1970年01月01日00时00分00秒起至现在的毫秒数，它是一种特殊的计量时间的方式

三种方式获取时间戳

1. 使用 getTime() 方法 let date = new Date()console.log(date.getTime())
2. 简写 +new Date() console.log(+newDate())
3. 使用 Date().now() console.log(Date().now())
4. 但是只能得到当前的时间戳， 而前面两种可以返回指定时间的时间戳

注意：

- 通过时间戳得到是毫秒，需要转换为秒在计算
  毫秒数 /1000=秒数
- 转换公式：
  d = parseInt(总秒数/ 60/60 /24); // 计算天数
  h = parseInt(总秒数/ 60/60 %24) // 计算小时
  h = h < 10 ? '0' + h : h
  m = parseInt(总秒数 /60 %60); // 计算分数
  m = m < 10 ? '0' + m : m
  s = parseInt(总秒数%60); // 计算当前秒数
  s = s < 10 ? '0' + s : s

```
<script>
  let hour = document.querySelector('#hour')
  let minutes = document.querySelector('#minutes')
  let scond = document.querySelector('#scond')
  time()
  setInterval(time, 1000)
  function time() {
    // 1. 得到现在的时间戳
    let now = +new Date()
    // 2. 得到指定时间的时间戳
    let last = +new Date('2021-8-29 18:30:00')
```

```
      // 3. （计算剩余的毫秒数） / 1000 === 剩余的秒数
      let count = (last - now) / 1000
      // console.log(count)
      // 4. 转换为时分秒
      // h = parseInt(总秒数 / 60 / 60 % 24)   //   计算小时
      let h = parseInt(count / 60 / 60 % 24)
      h = h < 10 ? '0' + h : h
      // m = parseInt(总秒数 / 60 % 60);     //   计算分数
      let m = parseInt(count / 60 % 60)
      m = m < 10 ? '0' + m : m
      // s = parseInt(总秒数 % 60); //   计算当前秒数
      let s = parseInt(count % 60);
      s = s < 10 ? '0' + s : s
      // console.log(h, m, s)
      hour.innerHTML = h
      minutes.innerHTML = m
      scond.innerHTML = s
    }

  </script>
```

事件对象

事件对象是对象里有事件触发时的相关信息

在事件绑定的回调函数的第一个参数就是事件对象 一般命名为event、ev、e

e的 部分常用属性

- type　获取当前的事件类型
- clientX/clientY 获取光标相对于浏览器可见窗口左上角的位置
- offsetX/offsetY获取光标相对于当前DOM元素左上角的位置
- key用户按下的键盘键的值现在不提倡使用keyCode
- 常用　e.target 真正触发事件的元素　e.target.tagName 事件对象的触发元素的名字

事件流

捕获阶段 从父到子

冒泡阶段 从子到父 事件冒泡是默认存在的

- 当一个元素触发事件后，会依次向上调用所有父级元素的同名事件
- 说明： addEventListener第三个参数传入true代表是捕获阶段触发（很少使用） 若传入false代表冒泡阶段触发，默认就是false 若是用 L0 事件监听，则只有冒泡阶段，没有捕获
- 阻止事件流动 事件对象.stopPropagation() 不光在冒泡阶段有效，捕获阶段也有效
- 鼠标经过事件： mouseover 和 mouseout 会有冒泡效果

  mouseenter 和 mouseleave 没有冒泡效果(推荐)
- 阻止默认行为，比如链接点击不跳转，表单域的跳转 e.prevenDefault()

两种注册事件的区别：

传统on注册（L0）

- 同一个对象,后面注册的事件会覆盖前面注册(同一个事件)
- 直接使用null覆盖偶就可以实现事件的解绑
- 都是冒泡阶段执行的

事件监听注册（L2）

- 语法: addEventListener(事件类型, 事件处理函数, 是否使用捕获)
- 后面注册的事件不会覆盖前面注册的事件(同一个事件)
- 可以通过第三个参数去确定是在冒泡或者捕获阶段执行
- 必须使用removeEventListener(事件类型, 事件处理函数, 获取捕获或者冒泡阶段)
- 匿名函数无法被解绑

```
<script>
       let fa = document.querySelector('.father')
       let son = document.querySelector('.son')
       fa.addEventListener('click', function (e) {
           alert('我是爸爸')
           e.stopPropagation()  //阻止冒泡
       })
       son.addEventListener('click', function (e) {
           alert('我是儿子')
           // 阻止流动 Propagation 传播
           e.stopPropagation()
       })
       document.addEventListener('click', function () {
           alert('我是爷爷')
       })
    let a = document.querySelector('a')
       a.addEventListener('click', function (e) {
           // 阻止默认行为 方法
           e.preventDefault()
       })
   </script>
```

事件委托

总结：

- 优点：给父级元素加事件（可以提高性能）
- 原理：事件委托其实是利用事件冒泡的特点
- 实现：事件对象.target 可以获得真正触发事件的元素

```
<body>
    <ul>
        <li>我是第1个小li</li>
        <li>我是第2个小li</li>
        <li>我是第3个小li</li>
        <li>我是第4个小li</li>
```

```
        <li>我是第5个小li</li>
    </ul>
    <script>
        // 不要每个小li注册事件了  而是把事件委托给他的爸爸
        // 事件委托是给父级添加事件 而不是孩子添加事件
        let ul = document.querySelector('ul')
        ul.addEventListener('click', function (e) {
            // 得到当前的元素
            e.target.style.color = 'red'
        })
    </script>
</body>
```

滚动事件

- 事件名：scroll 给 window 或 document 添加 scroll 事件
- 事件名：load 给 window 添加 load 事件 不光可以监听整个页面资源加载完毕，也可以针对某个资源绑定load事件
- 事件名：DOMContentLoaded 给 document 添加 DOMContentLoaded 事件
-

```
<script>
      let div = document.querySelector('div')
          //页面滚动
      window.addEventListener('scroll', function() {
          console.log(111)
      })
      // 盒子滚动
      div.addEventListener('scroll', function() {
          console.log(111)
      })
    //加载事件
     window.addEventListener('load', function () {
          let div = document.querySelector('div')
          console.log(div)
      })
  </script>
```

元素大小和位置

scroll家族

- scrollLeft和scrollTop 获取取元素内容往左、往上滚出去看不到的距离 可以修改
- scrollWidth和scrollHeight 获取元素的内容总宽高（不包含滚动条）返回值不带单位
- document.documentElement HTML 文档返回对象为HTML元素

offset家族

- offsetLeft和offsetTop 注意是只读属性 获取元素距离自己定位父级元素的左、上距离
- offsetWidth和offsetHeight 获取元素的自身宽高、包含元素自身设置的宽高、padding、border

client家族

- clientLeft和clientTop 注意是只读属性 获取左边框和上边框宽度
- clientWidth和clientHeight 获取元素的可见部分宽高（不包含边框，滚动条等）
- resize 窗口尺寸改变的时候触发事件

```
    <script>
        //通过 scrollTop  和  offsetTop  显示隐藏
       let sk = document.querySelector('.sk')
       let header = document.querySelector('.header')
       // 1. 页面滚动事件
       window.addEventListener('scroll', function () {
           // 2. 要检测滚动的距离 >= 秒杀模块的offsetTop 则滑入   sk.offsetTop 是 秒杀模块距离上边的距离
           if (document.documentElement.scrollTop >= sk.offsetTop) {
               header.style.top = '0'
           } else {
               header.style.top = '-80px'
           }
       })
   </script>
```

BOM操作浏览器

window对象

- window对象 是一个全局对象，也可以说是JavaScript中的顶级对象
- 像document、alert()、console.log()这些都是window的属性，基本BOM的属性和方法都是window的。
- 所有通过var/let 定义在全局作用域中的变量、函数都会变成window对象的属性和方法
- window对象下的属性和方法调用的时候可以省略window

```
  <script>
        window.document.querySelector('.box')
         window.setInterval()
           function fun() {
             }
        window.fun()
        addEventListener('scroll', function () {
            console.log(111)
        })
        window.alert()
        window.prompt()
        console.log(window)
    </script>
```

延时器和定时器

延时器器setTimeout	定时器 setinterval
延迟一段时间之后才执行对应的代码	每隔一段时间就执行一次
let timerId = setTimeout(回调函数, 延迟时间)	let timerId = setintervalt(回调函数, 多久执行一次)
清除延时器clearTimeout(timerId)	清除延时器clearinterval(timerId)
延时函数: 执行一次	间歇函数:每隔一段时间就执行一次,除非手动清除

```
<button>解除延时器</button>
    <script>
        let btn = document.querySelector('button')
        let timer = setTimeout(function () {
            console.log(111)
        }, 3000)
        // 仅仅执行一次
        btn.addEventListener('click', function () {
            clearTimeout(timer)
        })
    </script>
 <script>
     //定时器
        function show() {
            console.log('月薪过2万')
        }
       //创建定时器  setInterval(函数名，毫秒数)
     let timer = setInterval(show, 1000)
        // 清除定时器
        clearInterval(timer)
    </script>
```

location对象 主要负责网页的地址栏

- location.href　　跳转页面
- location.reload()　刷新
- locaction.search　?后面的内容
  locaction.hash　　#后面的内容

```
  <a href="#one">第一个</a>
   <a href="#two">第二个</a>
   <script>
   location.href = 'http://www.itcast.cn'  跳转链接
       locaction.search        //?username=w       表单为post 无法使用
       location.hash      // #two one
       location.reload()   //相当于浏览器刷新按钮
   </script>
```

navigator对象　主要用来获取浏览器的信息

navigator.userAgent 在这个字段里面判断是否有Mobile字段. 如果有表示是手机,反之则表示PC

```
<script>
      // 检测 userAgent（浏览器信息）
      !(function () {
          const userAgent = navigator.userAgent
          // 验证是否为Android或iPhone
          const android = userAgent.match(/(Android);?[\s\/]+([\d.]+)?/)
          const iphone = userAgent.match(/(iPhone\sOS)\s([\d_]+)/)
          // 如果是Android或iPhone，则跳转至移动站点
          if (android || iphone) {
              location.href = 'http://m.itcast.cn'
          }
      })()

  </script>
```

histroy对象　管理历史记录

history.forward() 前进 history.back() 后退 history.go(1/-1) 1前进 -1 后退

```
 <button class="forward">前进</button>
  <button class="back">后退</button>
  <script>
      let qianjin = document.querySelector('.forward')
      let houtui = document.querySelector('.back')
      qianjin.addEventListener('click', function () {
          // history.forward()
          history.go(1)
      })
      houtui.addEventListener('click', function () {
          // history.back()
          history.go(-1)
      })
  </script>
```

js执行机制

- js 把任务分为 同步任务和异步任务
  同步任务
  　　let num = 10
  异步任务
  　　定时器　事件 click　　load 加载　　ajax
- 执行执行栈（同步任务）里面的任务，执行完毕再去任务队列（异步任务）里面看看是否有任务，如果有，则得到放入执行栈中执行，再次循环

swiper 插件

插件: 就是别人写好的一些代码,我们只需要复制对应的代码,就可以直接实现对应的效果
https://www.swiper.com.cn/

本地存储

- 作用: 可以将数据永久存储在本地(用户的电脑), 除非手动删除 语法 存 localStorage.setItem('键', '值') 取 localStorage.getItem('键') 删 ocalStorage.removeItem('键')
- 存储复杂数据类型：本地只能存储字符串,无法存储复杂数据类型.需要将复杂数据类型转换成JSON字符串,在存储到本地 转换成JSON字符串的语法
    - JSON.stringify(复杂数据类型)　　将复杂数据转换成JSON字符串
      JSON.parse(JSON字符串)　　将JSON字符串转换成对象

```
<script>
     let obj = {
           uname: '刘德华',
           age: 17,
           address: '黑马程序员'
       }
     localStorage.setItem('obj', JSON.stringify(obj))
   JSON.parse(localStorage.getItem('obj'))
```

自定义属性

- 固有属性: 标签天生自带的属性 比如class id title等, 可以直接使用点语法操作
- 自定义属性: 由程序员自己添加的属性,在DOM对象中找不到, 无法使用点语法操作,必须使用专门的API
  getAttribute('属性名') // 获取自定义属性
  setAttribute('属性名', '属性值') // 设置自定义属性
  removeAttribute('属性名') // 删除自定义属性
- 规范一下 为 data-自定义属性
  传统的自定义属性没有专门的定义规则,开发者随意定值,不够规范,所以在html5中推出来了专门的data-自定义属性
- 在标签上一律以data-开头 在DOM对象上一律以dataset对象方式获取

```
<body>
    <div class="box" data-index="0" data-name="andy"></div>
    <script>
        // 设置自定义属性
        let box = document.querySelector('.box')
        // box.setAttribute('myid', 10)
        // console.log(box.getAttribute('myid'))
        console.log(box.dataset)
        console.log(box.dataset.index)
    </script>
</body JQuery
```