HTML5相关

script标签

```
<!-- 配置：crossorigin="anonymous"  在引入指定文件时，不向引入资源的服务器发送cookie内容
-->
<script crossorigin="anonymous"  href=""></script>
```

网页标题图标

- 写在head标签内，且`favicon.ico`放到网站的根目录下，浏览器会自动将该图标显示在地址栏和收藏夹中
- 建议设置`ico`格式的图片，并将图片命名为`favicon.ico`，像素大小16*16，颜色不超过16色
- 图片地址：本地图片 / 网络图片 均可。

`<link rel="icon shortcut" href="图片地址" type="image/x-icon" />`

清除浮动

1. `clear:both;`
2. 给父元素添加高度
3. 开启BFC（块级格式化上下文）
    - (BFC可以解决：margin合并 margin塌陷 浮动无法撑开父盒子)
    - (**BFC的子元素**不会对外面的元素产生影响)
    - 触发条件：
        - float 值不为none
        - overflow 值 不为visible
        - `display：table || table-cell || table-block`（不等于block、inline、none）
        - position:absolute || fixed（不为 relative）
        - 根元素html

SEO优化

语义化标签

```
<header>顶部的部分</header>
<nav>导航</nav>
<section>重要的内容</section>
<main>网站的一部分</main>
<article>网站的一部分</article>
<aside>与本网站相关的一些链接或内容区域</aside>
<footer>底部区域</footer>
```

表单增强

- type类型
  - tel 手机号
  - number：只允许输入数字
  - data 日期选择：会弹出日期选择框，value值格式为：`2022-04-21`
  - time 时间选择：value值格式为：`12:35`
  - 其他日期相关：datetime-local年月日时分、month年月、week第几周
  - color 颜色选择：value值格式为十六进制颜色值
  - range 滑块输入，属性：min最小值，max最大值，step分为几步

```
<input type='range' max='100' min='0' step='10'/>
```

- 新增input属性：

```
<!--
pattern：值为正则表达式，在表单提交时进行表单验证
title：值为字符串，当不符合正则时的错误提示
required：有该属性表示此项为必填项
autofocus：有该属性则在页面加载后自动被选中
placeholder：用户未输入时的提示文本，值为字符串

注意：必须使用from标签包裹，且只在点击提交按钮后进行验证
-->
<input type='tel' pattern='1[3-8][0-9]{9}' required title="请输入正确的手机号">
```

- 其他标签

```
<!--
  progress：进度条,与滑块输入相比，该进度条用户不能直接操作
  meter：标尺，好像没啥用
-->
<progress value='30' max='100'></progress>
<meter value='3' min='0' max='100'></meter>
```

- 其他属性

```
<!-- contenteditable='true'  该属性，可以让普通的元素变成一个可编辑的文本编辑器 -->
<p contenteditable='true'>这里的内容是可以进行编辑的</p>
```

CSS预处理语言

常见的css预处理器：Sass、Less、Stylus

Less 快速入门

- 使用：
  - 文件命名：`.less`
- 特点：
  - 引入变量、Mixin(混入)、运算、函数。。。
  - 简化css的编写、降低css的维护成本，用更少的代码做更多的事

Less变量

```
// 定义变量语法：    @变量名:值;
@backgroundColor:red;

// 使用变量：
body{
   background-color:@backgroundColor;
}
```

- 变量分类：
  - 作为普通属性值使用：直接使用`@+变量名`
  - 作为选择器和属性名：`#@{选择器的值}`的形式
  - 作为url：`@{url}`
  -
- 变量命名规范：
  - 必须有@作为前缀
  - 不能包含特殊字符
  - 不能以数字开头
  - 大小写敏感
- 特点：当修改定义时变量的内容时，可以自动修改引用它的地方的内容

Less编译

- less需要转化为css，才能够被使用
- Vscode中借助 easy less插件进行自动编译：保存less文件时，直接生成对应的css

less注释

- `//` 开头的注释，**不会**被编译到css文件中
- `/* */` 包裹的注释**会**被编译到css文件中

```
// 块注释和行注释都可以使用:

/* 一个块注释    会被编译到css中
 * style comment! */

// 一个行注释    不会被编译到css中    @var: white;
```

less嵌套

- 子元素的样式可以直接写在父元素的样式里面
- **伪类、交集选择器、伪元素选择器** 的使用需要在伪类前加`&`符号，依旧写在该元素内部
    - 没有 &符号的选择器，默认被解析为父元素的后代选择器

```
.header {
   width:100%;
   height:100%;
   //less 嵌套，子元素的样式可以直接写在父元素里
   a {
      color:red;
        // 为a添加 :hover 伪类
        &:hover{
         color:pink;
        }
   }
   // 为.header添加 :hover 伪类
   &:hover{
      color:yellow;
   }
}
```

less运算

- 任何数字、颜色、变量 都可以参与运算 + - * /
- 注意点:
    - 运算符左右两侧必须有一个空格隔开
    - 颜色值也可以进行运算，
    - **单位问题:**
        - 如果只有一个单位，则以这个单位为准
        - 当多个数都有单位且不相同时，则以第一个单位为准

```
//less 运算

@border: 5px + 5;
body{
   width: 200px - 50;
   height: 200px * 2;
   border: @border solid red;
   img {
      width: 80 / 50 rem;
      height: (@border + 2) * 20;
   }
}
```

less作用域

- 以括号{}为基础进行作用域划分，在括号内部的内容只属于该作用域，无法在父括号外使用；
- 最外层的定义的内容可以在任意位置使用，但存在命名冲突的内容时，以局部就近原则为准；

变量的延时加载

- 在less中，作用域中内容会先执行完，再回头为使用变量的内容赋值。

```
@var: 0;
.class{
   @var: 1;
   .brass{
      @var: 2;
      three: @var;
      @var: 3;
   }
   one: @var;
}

// 预测 three 和 one 的值
// 结果：three：3   one：1
```

less混合(mixin)

混合就是将一系列属性从一个规则集中引入到另一个规则集的方式

- 普通混合：会将定义的混合随编译放到css中
- 不带输出的混合：定义的混合只保存在less中，不出现在编译后的css中
- 带参数的混合：可以且必须在调用混合时，携带对应的参数；
  - 当不存在默认值时：实参与形参必须对应，且不能忽略
  - 当存在默认值时，可以忽略，或只指定传入某些参数和值

```
// 定义普通混合
.juzhong1{
   position:absolute;
   left:0;
```

```
    right:0;
    top:0;
    bottom:0;
    margin:auto;
}
// 不带输出的混合,就加了个小括号()
.juzhong2(){
    position:absolute;
    left:0;
    right:0;
    top:0;
    bottom:0;
    margin:auto;
}
// 带参数的混合,括号中写参数，类似于函数封装()
// 参数:值  定义混合时，可以设置默认值
.juzhong3(@w:10px,@h,@bgc){
    position:absolute;
    left:0;
    top:0;
    width:@w;
    height:@h;
    background:@bgc;
}


#wrap{
    position:relative;
    left:0;
    bottom: 0;
    .inner1{
        //使用 普通/不带输出的 混合
        .juzhong1;
        .juzhong2;
    }
    .inner2{
        // 使用带参数的混合，但没有默认值
        .juzhong3(100px,200px,red);
        // 使用带参数的混合，有默认值可以只指定某个参数
        .juzhong3(@h:20px;@bgc:hotpink);
    }
}
```

less更多

- [Less 快速入门 | Less.js 中文文档 - Less 中文网 (bootcss.com)]

拖放Drag,Drop

拖拽指的是鼠标点击源对象后一直移动对象不松手，一但松手即释放了（拖拽：Drag，释放：Drop）

使用步骤：

1. 设置拖拽元素为可拖放（为结构标签增加属性）
    - draggable=true 表示可拖拽 (a链接和img图片默认可拖拽)
    - draggable=false 表示不可拖拽
2. 绑定拖拽相关事件

    被拖动的源对象可以触发的事件：

    - ondragstart：源对象开始被拖动
    - ondrag：源对象被拖动过程中(鼠标可能在移动也可能未移动)
    - ondragend：源对象被拖动结束

      拖动源对象可以进入到上方的目标对象可以触发的事件：
    - ondragenter：目标对象被源对象拖动着进入
    - ondragover：目标对象被源对象拖动着悬停在上方
    - ondragleave：源对象拖动着离开了目标对象
    - ondrop：源对象拖动着在目标对象上方释放/松手

注意点：

- 这里常用到事件委托的思想，为更外层的元素绑定拖拽事件，配合`e.target`使用 更便捷、高效

```
// 目标对象被源对象拖动着悬停在上方时
// 阻止对元素的默认处理方式,否则会显示禁止拖动的图标
document.ondragover = function() {
      event.preventDefault();
 }
```

FileReader文件读取

FileReader对象可以读取本地存储的文件，借助表单添加文件的multiple属性，因此Files对象是一个伪数组形式

`var reader = new FileReader; // 实例化一个FileReader对象`

文件读取方法：

- `eader.readAsDataURL(); // DataURL形式读取文件`
    - 该方法将文件读取为一段以 data: 开头的字符串，这段字符串的实质就是 Data URL
    - Data URL是一种将小文件直接嵌入文档的方案。这里的小文件通常是指图像与 html 等格式的文件
    - 可以将读取来的数据直接赋给img标签的src属性
- `readAsBinaryString() // 以二进制编码的形式进行读取`

- `reader.readAsText(file,[encoding]); // 读取文件根据特殊的编码格式转化为内容(字符串形式)`
  - 参数1：文本
  - 参数2：可选值，文本的编码方式，默认值为 UTF-8
- `abort() // 中断读取`

读取处理事件：

- 事件监听：
  - onload 当文件读取完成且成功时调用
  - onabort 中断时触发
  - onerror 出错时触发
  - onloadend 读取完成触发，无论成功或失败
  - onloadstart 读取开始时触发
  - onprogress 读取中
- 属性
  - reader.result 文件读取结果
    - 当文件开始读取时，result就会被填充，
    - 读取失败，则 result 的值为 null ，否则即是读取的结果

> 1.检测浏览器对 FileReader 的支持
>
> ```
> if(window.FileReader){
> var fr = new FileReader();
> }else {
> alert("不支持")
> }
> ```

Canvas绘画与动画

因为canvas的默认宽高为300px*150px，在css中设置canvas的宽高，实际上是把canvas在300px*150px的基础上进行了拉伸，会导致绘制出来的图像会发生变形。

> 因此，如果要动态改变canvas的大小，用以下的方式可以使其不变形的同时放大缩小：
>
> ```
> document.getElementById("canvas").width = 100;
> document.getElementById("canvas").height =100;
> ```

- 正确的设置宽高：在canvas标签中使用 width 和 height 属性设置,默认单位是px
  - 例：`<canvas width=400 height=580 id='myCanvas'>当不支持该标签时显示该文字</canvas>`

建立Canvas绘画环境

- `obj.getContext('2d');` 建立2D绘图对象,obj为Cancas绘图对象

```
<canvas id='myCanvas'>当不支持该标签时显示该文字</canvas>
<!-- <canvas>是html标签，用于建立绘图环境，当浏览器不支持canvas时显示标签中的内容 -->
<!-- <canvas>本身不具有绘图功能，必须用js调用Canvas API进行绘图 -->

<script>
    let obj = document.querySelector('.myCanvas');      //获取绘制对象
    let one = obj.getContext('2d');         //建立2D绘图对象
</script>
```

绘制图形

- `beginPath();` 开始一个新的路径，清除旧的记录，但已经绘制出来的图案不受影响。习惯写在每次绘制线条前
- `stroke();` 该方法是绘制图形的框线，默认黑色；不使用该方法，可能无法看到的绘制的图形。
- `fillStyle="value";` 设置填充的颜色、渐变或模式 (注意fillStyle和fillRect的书写顺序,先设置再填充)
    - `color`：颜色值，默认为黑色
    - `gradient`:使用线性渐变或辐射渐变颜色
    - `pattern`：使用图案样式
- `strokeStyle="value";` 设置线条的颜色，可以应用在线条和矩形线框；value可取值如下
    - `color`：颜色值，默认为黑色
    - `gradient`:使用线性渐变或辐射渐变颜色
    - `pattern`：使用图案样式
- `fill();` 将当前所绘制的路径填满，如果绘制的图案未闭合则自动连接终点到起点形成闭合的图案
- **阴影的相关设置**
    - `shadowColor="value"` 设置阴影的颜色,value为颜色值，配合`shadowBlur`使用
    - `shadowBlur=number;` 设置阴影的模糊级数，直接写数字
    - `shadowOffsetX=number` 设置x轴阴影的偏移量，可以为负值
    - `shadowOffsetY=number` 设置y轴阴影的偏移量，可以为负值
- **色彩渐变效果 (可用于填充矩形、圆形、线条、文本等)**
    - HTML 画布 | 菜鸟教程 (runoob.com)
    - **线性渐变**
        - 建立渐变对象`createLinearGradient(x1,y1,x2,y2);` (x1,y1)(x2,y2)分别代表线性渐变的起点和终点,确定渐变方向
        - 设置区间内渐变的颜色`addColorStop(value,"color");`
            - value是色彩停驻点的位置，取值 0 ~ 1；0是线性渐变的起点，1是线性渐变的终点
            - color是颜色值

```
let obj = document.querySelect('canvas');       //获取canvas对象
let one = obj.getContext('2d');                 //建立2d绘图对象

let one_jianbian.createLinearGradient(20,20,60,20);    //建立渐变对象
one_jianbian
one_jianbian.addColorStop(0,'yellow');
one_jianbian.addColorStop(1,'red');                    //设置完渐变样式

one.fillStyle = one.jianbian;      //设置填充样式
one.fillRect(20,20,160,160);       //绘制填充的矩形
```

- **辐射渐变**
  - 建立渐变对象`createRadialGradient(x1,y1,r1,x2,y2,r2);` r1、r2分别是起点和终点的半径
  - 设置区间内渐变的颜色`addColorStop(value,"color");`与线性渐变用法一致

绘制矩形

- `rect(x,y,width,height);` 绘制矩形，x、y分别表示相对于绘制区左上角的距离，width、height分别表示绘制的宽高
- `strokeRect(x,y,width,height);` 绘制的同时添加外框线，是`rect()`和`stroke()`方法的集合，与`rect()`用法一致
-
- `fillRect(value);`绘制填充的矩形，与`rect()`使用方法一致
- `clearRect(x,y,width,height);`用来清除区块区间，用法与`rect()`用法一致

绘制线条

- `.moveTo(x,y);` 定义绘制线条的起点
- `.lineTo(x,y);` 定义绘制线条的终点，如果继续多次使用，则接着上一个终点继续绘制
- 绘制完成后也需要用`stroke()`方法进行上色
- `.closePath();` 关闭路径，将目前绘图点与绘图起点连接起来，围成图案。
- `.lineWidth` 使用canvas默认的线条宽度为1px，用lineWidth可以改，单位px；`obj.lineWidth=5;`
- `.lineJoin` 设置线条的交点样式
  - `mitre`：默认值，尖角
  - `round` ： 线的交点为圆弧
  - `fill()`：交点为斜的
  - `bevel`：交点是平的
- `.lineCap="value"` 设置线条端点的样式
  - `butt`：默认值，平的
  - `round`：线的端点是圆弧，会让线条变长一些
  - `square`：线的端点为矩形，会让线条变长一些

绘制圆形或弧线

- `.arc(x,y,r,startangle,endangle,counterclockwise);` 绘制圆形或弧线；
  - (x,y)是圆心坐标，r是圆的半径，后两个分别是起始角度和结束角度。一般为：0 - 2*Math.PI
  - 绘制弧线是，只需要调整起始和结束角度即可
  - 参数`counterclockwise`可省略，默认是flase，如果改为true，则逆时针方向绘制弧线
  - 绘制实心圆或者其他半圆则只需闭合弧线，然后改`fullstyle`填充样式即可

绘制文字

- `.font="value1 value2"` 设置字体
  - 参数1：文字大小
  - 参数2：字体样式
  - 还可以设置的参数还有：字体粗细、行高。。。 `与css的font简写属性类似`
  - 例：`ctx.font="30px Arial";`
- `.fillText(text,x,y);` 绘制实心的文本
  - 参数1：文本内容
  - 参数2：开始的横坐标
  - 参数3：开始的纵坐标
  - 例：`ctx.fillText("Hello World",10,50);`
- `.strokeText(text,x,y);` 绘制空心的文本（使用方法与 fillText(text,x,y); 一致）

绘制图像

- `.drawImage(img,sx,sy,swidth,sheight,x,y,width,height);`
  - `img`：规定要使用的图像、画布或视频
  - `sx/sy`：开始剪切的`x/y`坐标位置,**可选值**
  - `swidth/sheight`：被剪切图像的宽度/高度,**可选值**
  - `x/y`：在画布上放置图像的 x/y 坐标
  - `width/height`：要使用的图像的`宽度/高度`（伸展或缩小图像）,**可选值**

其他方法

- `.toDataURL(type,encoderOptions);` 返回一个包含图片展示的 数据URL
  - `type(类型)`：图片格式，默认为 image/png,可以是其他image/jpeg等
  - `encoderOptions`：0到1之间取值，主要用来选定图片的质量，默认值是0.92，超出范围也会选择默认值。
  - 返回值是一个数据url，是base64组成的图片的源数据、可以直接赋值给图片的src属性。
  - 方法前的内容，应是获取到的画布元素，而不是建立的绘图对象，否则无法下载图片并报错

canvas保存到本地图片方法

- 在canvas区域，右键直接能以png图片形式保存到本地

a标签法：

1. 将 canvas 元素的数据通过原生 api 转换成 base64 编码的图片格式
2. 利用 a 标签设置 download 属性可以将 href 链接元素下载，我们将 a 标签的 href 属性值设置为上一步获得的 base64 格式字符串
3. 构造一个单击事件并通过 api 触发 a 标签的 click 事件完成下载

- 缺点：
  - 无法被异步代码包裹，也就是包含 Ajax 请求的情况下代码不生效。
  - 对于分辨率过高的 canvas, 我们生成的 dataURL 过长，超过浏览器限制，可能会导致无法顺利下载

```
// Converts canvas to an image
function convertCanvasToImage(canvas) {
   var image = new Image();
   image.src = canvas.toDataURL("image/png");
   return image;
}
```

图片转base64

- 使用canvas原生方法实现：

```
function imageBase64(img) {
    //创建canvas画布，并绘制同等大小的canvas
    var canvas = document.createElement("canvas");
    canvas.width = img.width;
    canvas.height = img.height;
    var ctx = canvas.getContext("2d");
    ctx.drawImage(img, 0, 0, img.width, img.height);
    // 将canvas画布转化为png图片格式的 base64编码 并返回
    var dataURL = canvas.toDataURL("image/png");
    return dataURL;
}

function getImgBase64(src){
    var base64="";
    var img = new Image();
    img.src="src";
    //当图片加载完成后，执行函数获取图片banse64编码
    img.onload=function(){
        base64 = imageBase64(img);
        console.log(base64);
    }
}
// 传入图片路径
getImgBase64(src);
```

获取用户的经纬度数据

- GPS：精确的获取地理位置信息，误差小，定位慢、耗电多
- WIFI：可以获取位置信息
- 电话公司基地台：使用三角定位法获取位置信息
- IP地址：由IP地址获取位置信息，特别别是针对固定的IP地址，容易有误差。

判断浏览器是否支持获取位置

```
// navigator.geolocation  判断浏览器是否支持显示位置信息
//各种定位方法与相关属性都是 navigator.geolocation 的子对象，首先应确定它为true才能继续操作。


if(navigator.geolocation){
    语句1;       //如果支持，执行此语句
   }else{
    语句2;  //如果不支持，执行此语句
}
```

获取经纬度数据

- 用户允许浏览器获取位置信息后，会传回经度(longitude)、纬度(latitude)信息。

```
// getCurrentPosition(onSuccess,onError,option) 方法          获取用户的经纬度信息

//参数2、3可省略,这三个参数都属于函数，名称可以自定义，并通过调用函数的方法使用它们
```

第一个参数

参数 `onSuccess`：是一个回调函数，其中的 `position` 对象包含：`coords` 属性、`timestamp` 属性

`timestamp` 属性 本身是个对象，记录用户的当前 时间戳。

`coords` 属性本身也是一个对象，包含下列信息

- `coords.latitude` 纬度信息
- `coords.longitude` 经度信息
- `coords.accuracy` 位置准确度信息(如果有就返回)，单位：米
- `coords.altitude` 海拔高度信息(如果有就返回)，单位：米
- `coords.altitudeAccuracy` 海拔高度准确度信息，单位：米

- `coords.heading` 设备前进方向，用正北顺时针方向角度表示
- `coords.speed` 目前前进速度

```
if(navigator.geolocation){
    navigator.geolocation.getCurrentPosition(weiZhi);      //如果支持，使用
getCurrentPosition方法
   }else{
    语句2;  //如果不支持，执行此语句
}

function weiZhi(position){
    console.log(position.coords.latitude);     //打印纬度信息
    let timer = new Date(position.timestamp);  //获取时间戳并转化为 data实例对象，获取
具体时间信息
    console.log(timer.toLocaleTimeString());     //利用 date提供的方法对时间进行格式化
}
```

第二个参数

参数：`onError` 是错误处理函数，其中包含的`error对象` 含有`code属性` 用于返回错误信息代码

只有当获取位置出现错误时，这个函数才会被调用。

`code属性的可能值：`

- `PERMISSION_DENIED` 或 `1`：表示使用者拒绝提供信息
- `POSITION_UNAVAILABLE` 或 `2`：表示目前没有信息
- `TIMEOUT` 或者 `3`：请求用户地理位置超时
- `UNKNOWN_ERROR`：不明错误原因，未知错误

第三个参数

提供下列可配置信息

- `enableHighAccuracy`:默认值是 false，若是true，则告诉浏览器要获取高精度位置信息
- `maximumAge`：设定位置缓存时间，以毫秒为单位，如果不设置该值，该值默认为0，超过响应时间后浏览器会再次提供新数据
- `timeOut`：设定响应时间内未能获取位置定位，则执行`errorCallback()`返回code 3。默认为无穷大，如果timeout为负数，则默认timeout为0。

持续返回位置信息

```
// watchPosition() 方法     持续返回浏览器用户最新的位置信息，与getCurrentPosition()方法
用法一致
// clearWatch() 方法       终止 watchPosition()方法
```

百度地图API

- 在现实开发中，通过调用第三方API（如百度地图）来实现地理定位信息，这些API都是基于用户当前位置的，并将用位置位置（经/纬度）当做参数传递，就可以实现相应的功能。

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

  - 1.在搜索引擎中搜索“百度地图生成器”。
  - 2.切换当前城市。
  - 3.查找定位地点。
  - 4.拖动地图，微调定位地点，使定位地点位于地图的中央。
  - 5.也可通过地图上的控件调整地图的等级和位置。
  - 6.根据项目需求设置地图的显示尺寸，要现实的操作控件以及地图的状态。
  - 7.添加标注点，并对标注点进行设置说明，然后保存。
  - 8.预览最终结果，然后获取生成代码。
  - 9.拷贝生成代码，创建html文件，例如，map.html，并粘贴拷贝的代码。
  - 生成代码默认使用的gb2312编码

    修改成utf-8即可。
  - 地图中的图标默认是显示不出来的，需要手动替换图片地址。
  - http://api.map.baidu.com/lbsapi/creatmap/images/us_cursor.gif、http://api.map.baidu.com/img/markers.png

  功能复杂的地图建议：http://lbsyun.baidu.com/ 百度地图api，调用百度API进行地图设置