前端

- web1.0 静态页面 用户是信息的消费者
- web2.0 交互效果 用户既是信息的消费者，也是信息的制造者
- web3.0 移动互联网 大前端 基于HTML5开发

vscode

微软开发的 开源 轻量级

1、常用插件

- chinese 设置中文的语言环境
- open in browser 设置浏览器
- view in browser
- 预览Md文件——markdown preview enhanced

2、设置

文件菜单--首选项--设置

- 自动保存 auto save
- 字号大小 font size
- 折行控制 word wrap（on）

3、快捷键

- 打开命令面板 查看--命令面板或 F1|ctrl+shift+p
- 使用默认浏览器打开 alt + b
- 向下复制行 shift + alt + 向下箭头
- 向上复制行 shift + alt + 向上箭头
- 向下移动行 alt + 向下箭头
- 向前移动一个tab shift + tab
- 保存 ctrl + s
- 替换 ctrl + h
- 查找 ctrl + f
- 注释 ctrl + / alt + shift + a

浏览器

浏览器的核心部分——内核或渲染引擎

1）单内核浏览器

- IE
    - 生产商 Microsoft
    - 内核 Trident
- Firefox 火狐
    - 生产商 Mozilla
      网景 1998.11
      1994 第一次浏览器大战 Netscape IE
    - 内核 Gecko
- chrome 谷歌
    - 生产商 谷歌
    - 内核：webkit Blink
- Safari 苹果
    - 生产商 苹果
    - 内核 webkit
- opera 欧朋
    - 生产商 挪威欧普拉软件公司
    - 内核 Presto 2016年之后改用Google内核blink

2）双内核浏览器

```
大部分国产浏览器使用IE内核--trident
随着浏览器的发展现在也出现了一些双内核浏览器--IE内核和chrome内核
如：360浏览器、QQ浏览器
```

HTML介绍

1、前端页面的三个组成部分

```
web标准规定的网页技术的三层分离结构
W3C 万维网联盟 蒂姆 博纳斯 李 专门制定web标准 非盈利
```

- HTML
    - 超文本标记语言
    - 结构层
    - 负责描述页面的语义
- CSS
    - 层叠样式表
    - 表现层
    - 负责描述页面样式
- JavaScript
    - 脚本语言
    - 行为层
    - 负责描述页面的动态效果

2、什么是HTML

- HTML——Hyper Text Markup Language 超文本标记语言
- 是一种使用标记的语言
- 是一种描述网页的语言
- 是一种语法简单的语言
- 是一种结构清晰的语言

3、HTML文档——网页

- 扩展名.html
- 文件名格式

```
主文件名.扩展名
```

.doc .exe

4、标记——标签

- 标签：是由尖括号括起来的关键词<>
- 元素：标签及标签之间的内容的整体

```
单标签：<标签名>  或 <标签名 />
双标签：<标签名>[内容]</标签名>
         开始标签        结束标签
<标签名  属性名1="属性值1"  属性名2="属性值2">[内容]</标签名>
```

- 空元素：严格模式下在开始标签中闭合

> *注意：HTML的属性和属性值对大小写不敏感，推荐小写 abc ABC*

- HTML注释

```
<!-- HTML注释内容 -->  不会显示在浏览器上
```

5、HTML基本结构

```
在.html文档中，输入英文状态的!或html:5，创建基本结构
```

```
<!DOCTYPE html>
<!--
    <!doctype>文件类型定义DTD，作用告诉浏览器这个文档的版本信息，让浏览器解析器知道该使用哪种语法解析页面；不是标签，是一条声明语句；必须写在HTML文档的第一行；
    <!DOCTYPE html> 用HTML5的语法来解析HTML文档
 -->
<html lang="en"><!-- 网页的整理文件，根元素 -->
    <head><!-- 网页的头部信息，不会显示在网页中 -->
        <meta charset="UTF-8"><!-- 网页的元信息
            charset属性：字符集
```

```
                UTF-8 万国码，国际通用编码
                ASCII：0-9数字、大写和小写字母、一些特殊字符
                GBK：汉字国际扩展码，全部中文字符
                    gb2312  针对简体中文
        -->
        <title>网页标题</title>
    </head>
    <body><!-- 网页的主体内容 -->
        内容
    </body>
</html><!-- 网页结束 -->
```

常用的HTML标签

1、块级标签——div、h1~h6、p、hr

- div标签 搭建网页结构的基本标签
    - 语法

```
<div>[content]</div>
```

    - 特点
        - 宽度默认撑满整个父元素
        - 高度默认由内容撑开
        - 独立成行——垂直布局
- 标题标签 h系列标签
  h1~h6大小依次减小，重要程度依次减弱；h1标签在同一个页面中只能使用1次，其他标签可以重复使用
    - 语法

```
<h1>title1</h1>
<h2>title2</h2>
<h3>title3</h3>
<h4>title4</h4>
<h5>title5</h5>
<h6>title6</h6>
```

    - 特点
        - 宽度默认撑满整个父元素
        - 高度默认由内容撑开
        - 独立成行——垂直布局
        - 自带文字加粗效果
        - 自带间距
- 段落标签 p标签
    - 语法

```
<p>段落1</p>
```

- 特点
    - 宽度默认撑满整个父元素
    - 高度默认由内容撑开
    - 独立成行——垂直布局
    - 自带间距
- 其他标签
    - br标签 强制换行标签 空元素
      专门用来实现换行，不能设置其他样式， 不参与分类；不产生新段落的情况下换行

```
<br>  <br />
```

    - hr标签 水平分割线 空元素
      块级标签，默认自带间距和边框

```
<hr> 或 <hr />
```

2、行级标签——span、b、strong、i、em、sup、sub、del

文本格式化标签

- span标签：万能标签 用于区分样式
- b标签：是一个实体标签，被包围的字符显示粗体效果
- strong标签：是一个语义标签，作用加强语气，表示重要的文本，在文本中显示粗体效果
- i标签：被包围的字符显示斜体效果
- em标签：用来呈现被强调的文本，在文本中显示斜体效果
- sub标签：下标
- sup标签：上标
- del标签：删除线

特点

- 宽度默认由内容撑开
- 高度默认由内容撑开
- 默认横向显示——水平布局，相邻的行级元素在同一行显示，当一行排不下时才会换行
- 换行和空格会被解析

3、行块级标签——img

- img标签 图片标签
    - 语法

```
<img src="图片的路径" title="图片标题" alt="替换文本" width="" height="" />
```

src属性：图片的存储位置
title属性：鼠标悬停时的提示文字
alt属性：如果浏览器中无法正常载入图像，则使用alt属性值替代图像
width属性：宽度
height属性：高度

- 使用
    - 占位，可以撑开父元素
    - 属于网页内容，有实际意义，是必不可少的
    - 不可重复
    - 可以被搜索引擎检索到
- 特点
    - 默认水平布局
    - 换行和空格会被解析
    - 元素可以设置宽度和高度

路径

1、绝对路径

- 带协议的完整路径
- 盘符下的某个路径

```
    <img
src="http://www.ujiuye.com/uploadfile/2020/0903/20200903055422179.j
pg">
    <img src="D:\1.jpg" />
```

2、相对路径

- / 表示根目录
- ./ 表示当前目录，可以省略
- ../ 表示上一级目录
  ../../ 表示上上级目录
- 文件夹名/ 表示下一级目录

特殊字符

- © 版权信息©
- ¥ 人民币¥
- ® 注册®
-   空格
- < 小于 > 大于
- ™ 商标™
- ← ←
- & &

父子关系——嵌套关系

兄弟关系——并列关系

```
<html>
    <head>
        <meta charset="utf-8">
        <title></title>
    </head>
    <body></body>
</html>
```

超链接标签——a标签

1、语法

```
<a href="链接的URL地址" title="鼠标悬停时的提示信息"
target="_blank|_self">链接文字</a>
```

- href属性：超链接的目标——已有网站、本地文件、空链接（#）、锚点链接、伪链接（不跳转）

```
    <a href="http://www.baidu.com"></a>
    <a href="http://www.baidu.com">百度</a>
    <a href="http://www.baidu.com">百度</a>
    <a href="readme.md">readme.md</a>
    <a href="images/comp.jpg">comp.jpg</a>
    <a href="#">占位|跳转到当前页顶部</a>
    <a href="javascript:">内容</a>
    <a href="javascript:alert('a')">内容</a>
```

- target属性
    - target="_blannk" 在新窗口打开
    - target="_self" 默认值，在当前窗口显示

2、特点

- 行级元素
- 宽度默认由内容撑开
- 高度默认由内容撑开
- 默认横向显示——水平布局
- 自带下划线

- 自带文字颜色
- 换行和空格会被解析

3、锚点使用

跳转到长页面的某个位置、跳转到某个文件的特殊位置

- 定义锚点

```
<a name="锚点名称"></a>
<div id="锚点名称">[content]</div>
```

- 引用锚点

```
<a href="#锚点名称">链接文字</a>  页内跳转
<a href="文件路径#锚点名称">链接文字</a>  跳转到某个文件的特殊位置
```

列表——块级元素

1、无序列表

无次序、无级别，列表项之间是并列关系

- 语法

```
<ul><!-- 列表容器 -->
    <li>列表项</li>
    <li>列表项</li>
    <li>列表项</li>
</ul>
```

- 特性
    - 宽度默认撑满整个父元素
    - 高度默认由内容撑开
    - 独立成行——垂直布局
    - 自带间距
    - 自带填充
    - 自带列表符
- css属性
    - list-style-type属性 列表符类型
      none无列表符号 disc实心圆,默认值 circle空心圆 square实心方块
    - list-style-position属性 列表符的放置位置
        - list-style-position: inside; 列表符放在文本内
        - list-style-position: outside; 默认值，列表符放在文本的外侧

- list-style属性
  list-style: list-style-type list-style-position;
  `list-style: none;` 去掉列表符

2、有序列表

有排列次序，各个列表项之间是并列关系

- 语法

```
<ol>
    <li>列表项</li>
    <li>列表项</li>
    <li>列表项</li>
</ol>
<ol start="5" type="a" reversed>
    <li>列表项1</li>
    <li>列表项2</li>
    <li>列表项3</li>
</ol>
start属性：开始值
type属性：列表符类型
reversed属性：倒序
```

- 特性
  与无序列表基本一致

3、自定义列表

- 语法

```
<dl>
    <dt>列表标题  或专业术语</dt>
    <dd>列表项或对专业术语的解释</dd>
    <dt>专业术语</dt>
    <dd>对专业术语的解释</dd>
</dl>
```

> *浏览器显示，dd标签中的内容会缩进显示*

- 特性
  - 宽度默认撑满整个父元素
  - 高度默认由内容撑开
  - 独立成行——垂直布局
  - 自带间距

表格

- 早期 实现页面布局
- 现在 用来显示表格数据
- 表格——将数据有效地组织在一起，并以网格的形式进行显示

1、创建表格

- 创建基本表格
  包含table标签、tr标签、单元格；单元格中可以放置任意内容（文本、标签）

```
    <table border="1"><!-- 表格容器，用来定义表格 -->
        <tr><!-- 表格的行，必须嵌套在table内部 -->
            <th>表头单元格</th><!-- 默认文字加粗，水平、垂直居中显示 -->
            <th>单元格</th>
        </tr>
        <tr>
            <td>单元格</td>
            <td>普通单元格</td><!-- 默认水平居左，垂直居中显示 -->
        </tr>
    </table>
```

- 复杂表格的语法
  可以thead、tbody、tfoot划分表格结构

```
<table border="1">
        <caption>表格标题</caption><!-- 在表格中水平居中显示，只在表格中
有意义 -->
        <thead><!-- 表格的头，放置标题之类的内容，内部必须要有tr标签 -->
            <tr>
                <th>表头单元格</th>
                <th>单元格</th>
            </tr>
        </thead>
        <tbody><!-- 表格的正文，放置表格数据 -->
            <tr>
                <td>单元格</td>
                <td>普通单元格</td>
            </tr>
        </tbody>
        <tfoot><!-- 表格的脚注，放置脚注之类的内容 -->
            <tr>
                <td>单元格</td>
                <td>单元格</td>
            </tr>
        </tfoot>
    </table>
```

> *注意：在一个table标签中只能有一个thead和一个tfoot，但是可以有多个tbody标签，这些标签只能放在table标签中*

2、表格特性

- 有内容，没有设置宽度的单元格，由内容撑开
- 没有内容，没有设置宽度的单元格，默认平分整个表格
- 同一行单元格，高度只会识别一个，取最大值
- 同一列单元格，宽度只会识别一个，取最大值

3、表格属性

- border属性：表格边框
    - border="0" 默认，没有边框
    - border="1"
- width属性：宽度
- height属性：高度
- cellpadding属性：单元格内容和边框之间的距离
- cellspacing属性：单元格之间的距离，默认值2px

```
<table border="1" cellpadding="30" cellspacing="0"></table>
```

- 合并单元格
    - 跨列合并单元格 colspan="合并单元格的数量"
    - 跨行合并单元格 rowspan="合并单元格的数量"
    - 实现步骤
        - 1）确定跨行、跨列合并单元格
        - 2）找到目标单元格，添加合并单元格属性设置合并属性
        - 3）删除多余的单元格
- 表格的css属性
    - border-collapse属性
        - border-collapse: collapse; 边框合并
        - border-collapse: separate; 单元格边框独立，默认值
    - border-spacing属性 当边框独立时，行和单元格在水平方向和垂直方向的间距
        - 如果有两个值，第一个值表示水平方向，第二个值表示垂直方向
        - 如果只有一个值，表示水平和垂直方向的间距

总结

- 表格优点：方便排列一些有规律的、结构均匀的内容或数据
- 表格缺点：产生垃圾代码，影响页面下载速度、时间，灵活性不打，难于修改

HTML语义化

1、什么是HTML语义化？

根据页面内容结构，选择合适的标签。便于开发者阅读和写出更优雅的代码；便于浏览器、搜索引擎解析；便于团队开发和维护；有利于SEO。如：标题使用h1~h6，段落用p标签。

2、语义化的好处有哪些？

- 在没有css的情况下，页面也能呈现出较好的内容结构
- 使代码更具可读性，便于团队开发和维护
- 有利于用户体验
- 有利于SEO搜索引擎优化，和搜索引擎建立良好的沟通，有助于爬虫抓取关键字

标签的合理嵌套

- ul和li,ol和li,dl和dt、dd拥有父子关系的标签，ul和ol的第一级子元素只能有li,dl的第一级子元素只能使dt和dd
- p、dt、h系列标签中不能嵌套块级元素
- a标签不能嵌套a标签
- 行级元素不能嵌套块级元素

CSS——cascading style sheet

层叠样式表或级联样式表。为HTML页面文本内容、图片外形、版本的布局等外观样式的设置。
实现了页面内容和样式的彻底分离，提高工作效率

1、css语法

- 组成；选择器{ 属性名称1: 属性值; 属性名称2: 属性值; }
  找对象（标签）
  属性名称1: 属性值————声明语句，多个声明语句之间用分号隔开，属性名称和属性值之间用冒号隔开
- 注释——ctrl+/ shift + alt +a
  /* css注释 */

2、css特点

- 可以取代之前一部分必须使用专门图像处理软件实现的图片特效功能
- 利于管理样式，方便排版，简化管理成本
- 便于统一风格
- 几乎所有浏览器都支持

3、css样式的引入方式

1）行间样式——内联式

- 位置：放在开始标签内部
- 语法：使用style属性

```
<div style="width: 100px; height: 100px; background-color: red;">
</div>
    width属性：元素的宽度，属性值单位px像素
    height属性：元素的高度，属性值单位px像素
    background-color属性：背景颜色，属性值可以设置关键词 red红色
```

- 应用场景
  适用于单个元素需要有特殊样式
- 优缺点：
  比较直观、相同样式需要重复定义，造成代码冗余、作用范围较小，大量使用不利于后期维护，代码不能复用；结构和表现不分离

2）内部样式——嵌入式

- 位置：放在head标签内部
- 语法：使用style标签对

```
<head>
    <style type="text/css"></style>
</head>
```

- 应用场景
  适用于单个HTML文档需要特殊样式
- 优缺点：
  相对于行间样式，作用范围较大，代码可以复用；结构和表现半分离

3）外部样式——外链式

- 位置：将在外部新建的css文件，在head标签中通过link标签引入
- 语法：

```
<link href="css文件路径" rel="stylesheet" type="text/css">
    rel属性：当前文档和被链接文档之间的关系。只有stylesheet得到所有浏览器支持，表示外部文件类型是css文件
```

- 应用场景
  适用于多个HTML文档拥有相同样式
- 优缺点
  便于修改、维护，使用范围广，代码可以复用。结构和表现彻底分离
- 书写步骤
  - 1）新建一个css文件夹，在文件夹中新建一个后缀名为.css的文件

- 2）在外部的.css文件中设置编码方式：

```
@charset "utf-8";
```

- 3）在html文档中，head标签内部，写link标签引入外部样式

```
<link rel="stylesheet" href="css/*.css">
```

样式引入方式的优先级

- 行间样式优于内部和外部样式
- 内部样式和外部样式就近原则

选择器——基础选择器

1、通配符选择器 *

单独使用，匹配所有元素

2、标签选择器

- 元素选择器
- 以标签名称作为选择器，选取所有对应名称的标签
- 语法

```
结构中：<标签名></标签名>
样式中：标签名{ 声明语句 }
```

3、class选择器

- 类选择器
- 语法

```
结构中：<标签名 class="类名"></标签名>
    <标签名 class="类名1 类名2"></标签名>
样式中：.类名{ 声明语句 }
        .类名1{ }
        .类名2{ }
```

> *多个标签可以使用同一个class名称*
> *一个标签可以有多个class名称，多个名称之间用空格隔开（词列表），写在一个class属性中*

4、id选择器

- 语法

```
结构中：<标签名 id="id名称"></标签名>
样式中：#id名称{ 声明语句 }
```

> *id名称在本页面中只能出现一次*

选择器的命名规范

- 名称要有含义
- 名称建议以英文字母开头，包含英文字母、数字、-、_
- 除了-和_之外的特殊字符不允许使用（空格等）
- 不能使用中文汉字、纯数字，不能以数字开头
- 类名区分大小写

基础选择器的优先级

- 选择器的权值越大，优先级越高
- 基础选择器的权值分别为：通配符选择器（0）、标签选择器（1）、class选择器（10）、id选择器（100）、行间样式（1000）
- 选择器的权值相同时，后写的会覆盖前面写的内容

文本文字属性

- photoshop：窗口菜单--字符，打开字符面板
- Photoshop：编辑菜单--首选项--单位与标尺，打开的窗口中修改单位（文字单位、标尺单位）为像素，单击确定

1、文字属性

- font-size属性：字号，属性值单位像素px
  浏览器默认字号为16像素
- font-family属性：字体
  - 如果字体名称为中文或字体中包含空格，字体名称必须加引号
  - 可以把多个字体名称作为一个“回退”系统来保存。如果浏览器不支持第一个字体，则尝试下一个字体，多个字体之间用逗号隔开
  - 中英文混排的段落文字，英文字体要放在前面
  - Arial 微软雅黑 Times New Roman 宋体
- font-weight属性：文字加粗
  为字体指定了9级加粗度，范围100~900之间
  - font-weight: normal; 正常，默认值（相当于400）
  - font-weight: bold; 加粗（相当于700）
  - font-weight: 900;
- font-style属性：字体样式
  - font-style: normal; 正常，默认值

    - font-style: italic; 斜体
- line-height属性：文字行高
    - 文字在行高范围内垂直居中
      Photoshop中：视图菜单--标尺
      参考线：在标尺上按住鼠标左键不松手，拖拽鼠标
      按住alt键，滚动滚轮实现放大|缩小
      选框工具：m
    - 行高撑不起盒高
    - 如果行高的属性值没有单位，表示字号的倍数
- font属性
  font: [font-style] [font-weight] font-size[/line-height] font-family;

> *只有同时具有font-size属性和font-family属性值时，简写有效*

2、文本属性

- color属性 文字颜色
    - 关键词：transparent透明色、red、green、blue
    - 十六进制色值：由0-9 a-f的六位十六进制数组成
      #ff0000--#f00 #334545 #334555 #334455--#345
      photoshop：英文状态下，i键，右键点击--拷贝颜色的十六进制代码--在代码中ctrl+v粘贴
    - rgb颜色模式： r-red红色，g-green绿色，b-blue蓝色，取值范围0-255之间
      rgb(r,g,b) rgb(255,0,0)
    - rgba颜色模式：r-red红色，g-green绿色，b-blue蓝色，取值范围0-255之间；a-alpha透明度，0(完全透明) 1（完全不透明） 0~1之间表示半透明效果
      rgba(r,g,b,a)
- text-align属性：文本（图片）水平对齐方式，该属性设置给块级元素
    - text-align: left; 左对齐，默认值
    - text-align: center; 居中对齐
    - text-align: right; 右对齐
    - text-align: justify; 两端对齐
- text-decoration属性：文本装饰线
    - text-decoration: none; 去掉文本装饰线
    - text-decoration: underline; 下划线
    - text-decoration: overline; 上划线
    - text-decoration: line-through; 删除线

> *文本装饰线的颜色默认为前景色（color颜色相关）*

- text-indent属性：文本块首行缩进，属性值的单位px、em(字符宽度的倍数)、百分比（相对于父元素的宽度）

```
text-indent: 100px;/* 向右缩进 */
text-indent: -100px;/* 向左缩进 */
```

- text-transform属性：字母大小写转换
    - text-transform: capitalize; 首字母大写
    - text-transform: uppercase; 全大写

    - text-transform: lowercase; 全小写
- letter-spacing属性：字间距
- word-spacing属性：词间距

css长度单位

1、px 像素

- 相对于显示器的屏幕分辨率
- 值是固定
- 计算比较容器

2、em

- 相对长度单位，相对于父元素的font-size属性值而言
- 值不固定
- 浏览器默认字号16像素，未经调整的浏览器符合：1em = 16px

3、rem

- 相对长度单位，相对于根元素的font-size属性值而言
- 值不固定
- 浏览器默认字号16像素，未经调整的浏览器符合：1rem = 16px

> *是css3新增单位，IE8及更早版本的浏览器中不兼容*

4、%

- 相对长度单位

盒模型Box Model

1、什么是盒子模型？

- 框模型，在页面布局中，将页面元素合理、有效地组织在一起，形成一套完整的、行之有效的规则、规范
- 包含：元素内容content、内填充padding、边框border、外间距margin几个要素
- 标准盒模型的计算公式：

2、盒模型相关属性

- 元素内容相关属性
    - width属性：元素的宽度
    - height属性：元素的高度
        - 属性值：auto(默认值)、数值(单位px|em|rem|%——相对于父元素的宽度和高度)、inherit
    - max-width属性：最大宽度　min-width属性：最小宽度
    - max-height属性：最大高　min-height属性：最小高
- 内填充——padding属性
  定义元素边框和元素内容之间的区域
    - 单边内填充
        - padding-top属性 上内填充
        - padding-bottom属性 下内填充
        - padding-left属性 左内填充
        - padding-right属性 右内填充
    - 复合写法
        - padding: npx;
        - padding: npx mpx;
        - padding: npx mpx xpx;
        - padding: npx mpx xpx ypx;

```
  padding: 10px;/* 元素的上、右、下、左各有10像素的内填充 */
  padding: 10px 20px;/* 元素的上、下各有10像素，左、右各有20像素内填充 */
  padding: 10px 20px 30px;/* 元素的上10像素，左右各有20像素，下30px的内填充 */
  padding: 10px 20px 30px 40px;/* 元素的上10px、右20px、下30px、左40像素的内填充 */
```

> *上下内填充和左右内填充百分数值是相对于父元素的width属性计算*

- 边框——border属性
  围绕着内容和内填充的线
    - 边框属性
        - border-width属性：边框的宽度，单位像素
        - border-style属性：边框的线型
            - border-style: solid; 单实线
            - border-style: double; 双实线
            - border-style: dotted; 点状虚线
            - border-style: dashed; 条状虚线
            - border-style: none; 没有边框
        - border-color属性：边框颜色
          如果没有设置边框颜色，默认于文本颜色一致
    - 复合写法
        - border-width: npx;
          border-width: npx mpx;
          border-width: npx mpx xpx;
          border-width: npx mpx xpx ypx;

        - border-style
        - border-color
    - 单边属性
        - border-top属性 上边框
          border-top: border-top-width border-top-style border-top-color;
          border-top: 5px solid red;
        - border-bottom属性
        - border-left属性
        - border-right属性
    - border简写
      border: border-width border-style border-color;
    - 去掉边框
        - border: 0;
        - border: none;
    - 使用——实现向下三角

```
.box4{
    width: 0;
    height: 0;
    font-size: 0;
    line-height: 0;

    border-top: 100px solid red;
    border-left: 100px solid transparent;
    border-right: 100px solid transparent;
}
```

- 外间距——margin
  两个盒子之间的距离，盒外属性
    - 单边外间距
        - margin-top属性 上外间距
        - margin-bottom属性
        - margin-left属性
        - margin-right属性

> *margin值可以设置负值，较少元素的占位。如margin-top: -50px; 将元素在原位置的基础上，向上移动50像素，减少50像素的占位*
>
> - *复合写法*
>   *margin: npx;*
>   *margin: npx mpx;*
>   *margin: npx mpx xpx;*
>   *margin: npx mpx xpx ypx;*
> - *盒子的水平居中*
>   *margin: 0 auto; 固定宽度且居中，必须与width属性配合使用才会有效*

背景

应用于内容、padding、border区域

1、背景颜色属性——background-color

- 关键词 transparent透明色 默认背景颜色 red
- 十六进制色值：#+六位十六进制数
- rgb(r,g,b)
- rgba(r,g,b,a)

> *作用于内容、padding、border区域*

2、背景图片属性——background-image

- 语法

```
background-image: url(图片地址);
background-image: none; 没有背景图片，默认值
```

- 特征
    - 背景图不占位
    - 默认背景图在水平方向和垂直方向重复平铺，并且铺满整个内容区域
    - 背景图是网页的装饰，起美化页面的作用
    - 背景图不会被搜索引擎检索到
    - 写在样式中

3、背景图是否重复属性——background-repeat

```
- background-repeat: repeat; 默认值，沿着水平方向和垂直方向重复
- background-repeat: repeat-x; 沿着水平方向重复
- background-repeat: repeat-y; 沿着垂直方向重复
- background-repeat: no-repeat; 不重复
```

4、背景图的位置属性——background-position

- 语法

```
background-position: x y;
    x--沿着水平方向移动
    y--沿着垂直方向移动
```

- 属性值
    - 关键词：left|center|right top|center|bottom
    - 数值：px 水平方向，向右为正；垂直方向，向下为正
    - 百分比：左上角0% 0% 右下角100% 100%

> *如果只有一个关键词，则另一个关键词默认为center*
> *如果只有一个数值表示水平方向，则另一个数值默认为50%（垂直方向）*

5、简写属性——background

- 语法

```
background: background-color background-image background-repeat
background-position;
background: 背景颜色 背景图片 背景图重复 背景图位置;
```

选择器

1、后代选择器——选择器之间用空格隔开

```
祖辈选择器  后辈选择器{ }
    祖辈选择器范围内的所有的后辈选择器都有效
```

2、子代选择器——选择器之间用>隔开

```
父选择器>子选择器{ }
    父选择器范围内的所有的第一级子选择器有效，孙子元素无效
```

3、群组选择器——选择器之间用逗号隔开

- 分组选择器，当样式表中有具有相同样式的元素

```
选择器1,选择器2{ }
    选择器1和选择器2都具有相同的样式
```

4、交集选择器

由两个及以上选择器组成

```
结构中：<div class="box"></div>
样式中：div.box{ }
```

5、伪类链接选择器

用于添加特殊效果

- 语法

```
    选择器:伪类{ }
```

- 用于设置链接不同状态

```
a:link{ 链接的默认状态 }
a:visited{ 链接访问过后的样式 }
a:hover{ 鼠标悬停时的样式 }
a:active{ 鼠标按下时的样式 }
```

> *四个伪类状态都有效：L-v-H-a*

- :hover 不仅可以用于链接的悬停，而且可以用于其他标签

```
/* 鼠标悬停到.box盒子上，让.box盒子背景颜色变为绿色 */
    .box:hover{
        background: lightgreen;
    }
/* 鼠标悬停到.box2盒子上，让.box2中的子盒p背景颜色变为red */
    .box2:hover p{
        background: red;
    }
```

css三大特性

1、层叠性

- 浏览器处理样式冲突的一种能力
- 多种css样式 不同的选择器 作用在同一个元素上
    - 样式不冲突：样式叠加，并同时作用在元素上
    - 样式冲突：存在优先级问题，优先级高的显示；不存在优先级问题，后写的会把先写的覆盖掉

2、继承性

- 子元素继承父元素的样式
- 不是所有的属性都能被继承（width、height、background-color、margin、border、padding、text-decoration）
- 可以被继承的属性（目前）
    - 字体系列属性
      font-size属性、font-family属性、font-weight属性、font-style属性、line-height属性、font属性
    - 文本系列属性
      text-align属性、text-indent属性、color属性、letter-spacing属性、word-spacing属性
    - list-style属性

> *a标签的文字颜色需要选中a标签之后单独设置才能修改*

3、优先级

- 样式引入方式优先级
  行间样式优于内部和外部样式；内部样式和外部样式就近原则
- 选择器优先级
  - 选择器优先级与权值相关
  - 基础选择器的权值：通配符选择器（0）、标签选择器（1）、class选择器（10）、id选择器（100）
    复合选择器的权值是所有的单一选择器权值的累加
  - 权值越大，优先级越高；权值相同，后写的会覆盖先写的
  - 继承样式的优先级为0，子元素设置样式会覆盖继承样式
  - 行间样式权值为1000，优于id选择器
  - !important优于行间样式

```
div{
    background: orange !important;
}
```

> *伪类选择器、属性选择器（10）*
> *伪元素选择器（1）*

标签分类与转换

1、标签分类及特征

1）块级标记 block level

- 特征：
  - 宽度默认撑满整个父元素
  - 高度默认由内容撑开
  - 默认独立成行——垂直布局
  - 具有盒模型特征——默认可以设置宽度、高度、padding、border、margin
- 常用块级标记
  div、h1、h2、h3、h4、h5、h6、p、ul、ol、li、dl、dt、dd等

2）行级标记 inline level——内联

- 特征：
  - 宽度默认由内容撑开
  - 高度默认由内容撑开
  - 默认横向显示——水平布局，相邻的行级标记会在同一行显示，直到一行排不下折行
  - 换行和空格会被解析
  - 具有部分盒模型特征——默认没有width、height属性，边框保留，可以设置padding-left、padding-right、margin-left、margin-right

- 常用行级标记
  span、b、strong、i、em、sup、sub、del、a等

3）行块级标记 inline-block

- 特征
    - 具有行级标记的特征：默认横向显示——水平布局，相邻的行级标记会在同一行显示，直到一行排不下折行；换行和空格会被解析
    - 具有块级标记的特征：具有盒模型特征
- 常用行块级标记
  img

2、标签类型转换——display属性

```
- display: block; 转为块级标记
- display: inline; 转为行级标记
- display: inline-block; 转为行块级标记
- display: none; 隐藏元素，隐藏之后不占位
```

外边距（间距）塌陷问题

1、并列关系的外间距的塌陷

- 现象：
  元素并列关系，垂直方向相邻的margin值相遇，会出现叠加现象——两个值相同，取当前值；两个值不同，取较大值
- 原因：
  并列关系的两个元素共用一个外间距
- 解决方案：
    - 为这两个并列关系的元素分别套一个父元素，并为父元素设置overflow:hidden;属性
    - 可以为并列关系的元素分别触发BFC

2、嵌套关系的外间距塌陷

- 现象：
  元素嵌套关系，子元素的margin-top属性值会叠加给父元素；如果父元素有margin-top属性值，会与子元素的margin-top属性值合并，取较大值
- 原因
  父元素和子元素共用一个上外间距区域
- 解决方案：
    - 为父元素设置上边框或上内填充
    - 为父元素设置overflow:hidden; 触发BFC，改变父元素的渲染规则
    - 转换思路，巧用padding，规避margin

float浮动

- css的定义机制：标准流、浮动、定位
- 标准流——文档流：文档中可以显示的对象在排列时所占的位置

1、浮动

- 使元素脱离正常文档流，按照指定方向发生移动，直到碰到父元素的边界或另外一个浮动元素的边界为止

> *浮动让元素水平方向移动，不能上下移动*

2、浮动属性

- `float: left;` 左浮动
- `float: right;` 右浮动
- `float: none;` 不浮动，默认值

3、浮动特性

- 元素脱离正常文档流
- 可以提升层级（半层）
- 使没有设置宽度的块级元素宽度由内容撑开；如果浮动元素的宽度之和大于父元素，则浮动盒会被挤到下一行显示
- 使行级元素支持宽高
- 浮动元素不占位，父级盒高度为零

4、浮动产生的问题

元素浮动之后，脱离正常文档流，导致父元素高度塌陷，会影响与浮动元素父级盒同级的后续元素的正常布局

5、清浮动的方法

- 给浮动元素的父级盒设置一个固定的高度
    - 不够灵活；适用于高度固定的布局中
- 给浮动元素的父级盒设置浮动
    - 会产生新的浮动问题
- 给浮动元素的父级盒添加overflow属性，属性值可以是hidden、scroll、auto
    - 可能会导致内容显示不完全、代码简洁
- 在浮动元素之后，与浮动元素呈现并列关系的位置，加一个空div(div元素本身不浮动，没有尺寸)，在空div上加属性clear: both;
    - 代码冗余；通俗易懂，书写方便
- 推荐方式：给浮动元素的父级盒加类名.clearfix，并在.clearfix中添加样式：

```
.clearfix{
  *zoom: 1;
}
.clearfix::after{
```

```
    content: "";
    display: block;
    clear: both;
}
```

- 不会在结构上产生冗余代码；可以重复使用；结构语义化正确

overflow属性

内容溢出处理方式，包含水平方向和垂直方向
- overflow: visible; 默认值，溢出显示
- overflow: hidden;/* 溢出隐藏 /
- *overflow: auto;/* 自动，内容溢出时显示滚动条 /
- *overflow: scroll;/* 溢出显示滚动条 */
- overflow: inherit; 从父元素继承overflow属性

- overflow-x属性：只包含水平方向的内容溢出处理方式
- overflow-y属性：只包含垂直方向的内容溢出处理方式

clear属性

清除浮动带来的影响
- clear: left; 清除左浮动
- clear: right; 清除右浮动
- clear: both; 清除两侧浮动

伪元素选择器

1、伪元素

> 用css语言创造出来的标签

2、创建伪元素

- element::before{ content:"伪元素的文本内容"; 属性名:属性值; }
  element元素内部，内容之前，添加"伪元素的文本内容"
- element::after{ content:"伪元素的文本内容"; 属性名:属性值; }
  element元素内部，内容之后，添加"伪元素的文本内容"

vertical-align属性 垂直对齐方式

- 常用属性

        - vertical-align: top; 顶端对齐
        - vertical-align: bottom; 底端对齐
        - vertical-align: middle; 中部对齐
        - vertical-align: baseline; 基线对齐，默认值
        - vertical-align: super; 上标
        - vertical-align: sub; 下标
    - 使用
      图片下方间隙问题、文本与图片对齐方式处理、文本输入框和按钮之间的对齐方式处理
        - 图片下方间隙问题
            - 将img标签转成块级元素
              img{ display: block; }
            - 为img标签设置垂直对齐方式
              img{ vertical-align: top|bottom|middle; }
            - 为img标签的父元素设置font-size:0;或line-height:0;
            - 为img标签的父元素设置高度，添加overflow:hidden;

定位

- css的定义机制：标准流、浮动、定位

1、定位原理

- float属性让元素水平移动
- margin属性让元素相对与本身位置移动
- css中定位属性允许元素相对于元素本身的位置，相对于父元素、浏览器窗口位置调整
- 网页中出现覆盖关系，优先考虑使用定位技术
- 定位偏移属性
    - top属性
    - bottom属性
    - left属性
    - right属性
        - 属性值：auto|inherit|length|百分比

> *定位偏移属性不能单独使用，必须与定位属性配合使用才会有效*

2、定位属性

- 1）position: static; 默认值
    - 静态定位，相当于没有定位；元素出现在正常文档流中
    - 不会受到top|bottom、left|right属性影响
- 2）position: relative; 相对定位
    - 相对于元素本身的位置调整；占位依然在原位置
    - 使用
      元素微调；做绝对定位元素的参照元素

    - 特性
        - 不影响元素本身的特性
        - 不会使元素脱离正常文档流——占位
        - 如果没有设置定位偏移属性，对元素本身没有任何影响；如果有定位偏移属性，相对于元素原来的位置偏移；
        - 提升层级
- 3）position: absolute; 绝对定位
    - 相对于最近的定位父元素（定位父级）定位
    - 特性：
        - 元素脱离正常文档流，不占位
        - 有定位父级，相对于定位父级发生位移偏移；如果没有定位父级，相对于整个文档发生偏移
        - 使没有设置宽高的块级元素宽度自适应，使行级元素支持宽、高
        - 提升层级
    - 绝对定位使用步骤
        - 为要做特殊定位的盒子（定位盒），添加：position: absolute; 绝对定位，设置定位偏移属性：
          top: 0; | bottom:0;
          left:0; | right:0;
        - 为定位盒的父级盒（定位父级），添加position: relative; 相对定位
        - 回到定位盒，通过top|bottom、left|right属性进行精确位置的调整
- position: fixed; 固定定位
    - 相对与浏览器窗口的四个角的位置定位
    - 特性
        - 元素脱离正常文档流，不占位
        - 始终相对于浏览器窗口的四个角为原点进行定位；不会随页面滚动而滚动
        - 使没有设置宽度的块级元素宽度自适应；使行级元素支持宽高
        - 提升层级
    - 使用：固定定位的绝对居中

```
width:;
height:;

position: fixed;
left: 0;
right: 0;
top: 0;
bottom: 0;
margin: auto;
```

> *relative相对定位占位；absolute绝对定位、fixed固定定位不占位*

3）定位技巧—— 绝对居中（水平、垂直居中）

- 使用margin:auto; 实现有width属性和height属性的绝对定位元素的居中

```
.wrap1 div{
    width: 100px;
    height: 100px;

    position: absolute;
    left: 0;
    right: 0;
    top: 0;
    bottom: 0;
    margin: auto;
}
```

- 使用margin负间距实现具有width属性和height属性的绝对定位元素的居中

```
.wrap2 div{
    width: 200px;
    height: 100px;

    position: absolute;
    left: 50%;/* 定位父元素宽度的一半 */
    top: 50%;/* 定位父元素高度的一半 */
    margin-left: -元素本身宽度的一半;
    margin-top: -元素本身高度的一半;
}
```

> *margin: auto;实现盒子的绝对居中有兼容问题*

4）定位中存在的问题——层级的问题

- 元素添加了定位属性（相对定位、绝对定位、固定定位）之后，可以覆盖在页面的其他元素上
- 后面加载的定位元素会覆盖先加载的定位元素
- z-index属性：设置定位元素的叠放次序
    - 属性值没有单位，取值可以是正整数、0、负整数
    - 属性值越大叠放次序越高
    - 属性值相同按照结构中书写顺序，后来者居上
    - 正值向上调整层级；负值向下调整层级

> *z-index属性需要于position: relative|absolute|fixed;配合使用才会有效*
> *如果同时有margin:0 auto; float: left|right; position: absolute|fixed; 则：position属性有效*
> *如果同时有margin:0 auto; float: left|right; position: relative; 则float属性有效*

浮动和定位对比

css2中能够脱离正常文档流的属性

- float:left|right
    - 脱离正常文档流，但不脱离文本流
- position:absolute|fixed;
    - 即脱离正常文档流，又脱离文本流

> *注意：所有元素都能使用以上的属性*
> *元素脱离正常文档流之后，不再区分块级和行级元素，都具有相同的属性；没有设置宽度，宽度由内容撑开；可以设置盒模型属性*

透明度

1、rgba(r,g,b,a)颜色模式——颜色透明

- 兼容性：IE6、7、8下不兼容,IE9+支持
- 使用
  background-color属性、color属性、border-color属性等设置颜色透明
- 语法

```
    rgba(r,g,b,a)
        r-red 红色 取值范围0-255
        g-green 绿色 取值范围0-255
        b-blue 蓝色 取值范围0-255
        a-alpha 透明度 取值范围0~1之间表示半透明，0表示完全透明，1表示完全不透明
```

> *超出范围的值将被截至最近的极限值*

2、opacity属性 透明度——元素透明

- 兼容性：IE6、7、8下不兼容，IE9+支持
- 使用：
  将内容（及所有的后代）、背景一起透明
- 取值
  范围0~1之间表示半透明，0表示完全透明，1表示完全不透明

3、filter属性——元素透明

- 兼容性：仅仅支持IE6、7、8、9，IE10及以上被废除
- 使用：
  IE浏览器专属
- 语法

```
filter: Alpha(opacity=n);
   n取值范围0~100之间表示半透明，0表示完全透明，100表示完全不透明
```

```
    .box23{
        opacity: .5;
        filter: Alpha(opacity=50);
    }
```

显示和隐藏元素的方法

- display属性
    - display: none; 元素隐藏，不占物理空间
    - display: block; 元素显示
- opacity属性
    - opacity: 0; 元素透明，占位空间仍然存在
    - opacity: 1; 元素不透明
- visibility属性
    - visibility: hidden; 元素隐藏，占位空间仍然存在
    - visibility: visible; 元素显示
- 设置元素的位置让其消失
  使用position属性，用z-index属性遮盖
- overflow:hidden;属性，将要隐藏的元素移动到父元素之外
- 将元素的font-size、line-height、width、height属性的属性值都设置为0

border-radius属性

- border-radius: npx; 元素的四个角都有npx的圆角
- border-radius: npx mpx; 元素的左上角、右下角各有npx；右上角和左下角各有mpx的圆角
- border-radius: npx mpx xpx;元素的左上角npx，右上角和左下角各有mpx，右下角有xpx的圆角
- border-radius: npx mpx xpx ypx;元素的左上角npx，右上角mpx，右下角有xpx的圆角,左下角ypx的圆角
- border-radius: border-top-left-radius border-top-right-radius border-bottom-right-radius border-bottom-left-radius;
  border-radius: 左上角 右上角 右下角 左下角

photoshop

测图、切图、修图

1、文件操作

- 新建文件
  “文件”菜单--“新建” 或 ctrl + n

> *工作区中 灰白格子表示背景透明*

- 打开文件
  “文件”菜单--“打开” 或 Ctrl + o

2、ps单位设置

“编辑”菜单-- “首选项” -- “单位与标尺”，将标尺和文字的单位改为像素

3、标尺

- 显示|隐藏
  “视图”菜单--“标尺” 或 ctrl + r
- 参考线
  - 在标尺上按住鼠标左键拖拽，拉出参考线
  - 水平参考线和垂直参考线的临时切换：按住alt键
  - 删除参考线
    - “视图”菜单 -- “清除参考线”
    - 在移动工具状态下，将参考线拖回到标尺，实现删除
    - 在其他工具状态下，按住Ctrl键，临时切换工具将参考线拖回标尺

4、面板

- 字符面板：“窗口”菜单 -- “字符” 或 “文字”菜单--“面板”--“字符面板”
- 信息面板：“窗口”菜单 -- “信息” 或 F8

5、工具的使用

- 移动工具 v
  属性栏选择"自动选择" --后面选择“图层”
- 矩形选框工具 m
  - 取消选择区 ctrl + d
  - 选中选择区： 按住Ctrl键，单击图层缩略图
  - 在选中某个区域的情况下，按下Ctrl + t键自由变换，拖拽选区
- 复制某个图层，复制某个图层的某个区域，必须要选中这个图层
- 吸管工具 i
- 文字工具 t
- 图片缩放 z
  - 在缩放工具状态下，向外拖表示放大，向内拖表示缩小

        - 按住alt键，滚动鼠标滚轮，向外滚表示放大，向内滚表示缩小
        - ctrl + 表示放大　Ctrl- 表示缩小
    - 抓手工具
        - 按住空格键，临时切换为抓手工具
    - 切片工具
        - 裁剪工具 c　工具组内切换 shift+c　在切片工具状态下，按住ctrl键，临时切换为切片选择区工具
        - 保存切片
          “文件”菜单--“导出”-- “存储为web所用格式” 或 ctrl + alt + shift + s -- 预设中设置图片格式，修改品质，选中切片，单击存储 -- 选择切片保存位置，修改文件名称，图像（仅限图像），切片（选中的切片）-- 保存
    - 撤销
        - Ctrl + z
        - ctrl + alt + z 撤销多步
        - “历史记录”返回

psd格式：Photoshop的源文件格式，一个分层文件

常见的图片格式及特点——网页中实际用到的图片格式

1、jpg格式（jpeg）——网页中的大图、高清图（体积较大）

- 不支持透明
- 优点：色彩丰富、文件较小
- 缺点：有损压缩，反复保存图片质量下降

2、gif格式——色彩单一的图片、动画图片

- 支持透明、不透明，支持动画
- 优点：文件小，支持动画，没有兼容问题
- 缺点：色彩简单，只支持256种颜色

3、png格式——层次较多的透明图片，色彩丰富的图标，细节要求较高的高清大图等

- 支持透明、不透明、半透明
- 优点：无损压缩、简单图片尺寸较小
- 缺点：色彩丰富图片尺寸较大
- 分类：
    - png8 基本透明、色彩简单
    - png24 色彩丰富，对透明层的支持丰富、细节显示较好（IE6不支持）

4、webp格式——适用于支持webp格式的APP或webview

- 优点：文件小，但是能达到jpg格式图片相同的图片质量，支持有损压缩和无损压缩；支持动画、支持透明
- 缺点：浏览器兼容性不好

css sprites

1、什么是css sprites?

- 叫精灵技术或雪碧技术，可以被解释为“css图像拼合”或“css贴图定位”
- 将一堆小图标整合在一张大图上，通过css属性“background-image” “background-repeat”，“background-position”将图片显示出来，并通过背景图定位精确显示，减少服务器对图片的请求数量

2、优缺点

- 优点：
    - 减少网页的http请求，提高页面性能
    - 减少在图片命名上的困扰
    - 更换风格方便
- 缺点：
    - 必须要限制容器的大小，背景图位置需要计算
    - 维护比较麻烦

3、使用步骤

```
- 1）制作一张具有多种状态的拼合图片，处理图片要有一定规律
- 2）给要显示背景图片的盒子设置一个固定的尺寸，以背景的方式让图片局部显示
- 3）通过背景图定位属性控制不同的显示状态
```

页面的TDK

- T-title网页的标题
  尽量将重要关键词放在前面，关键词不要重复。可以作为默认快捷方式或收藏夹名称；标题与页面内容相关，尽量简洁
- D-description网页的描述信息

```
<meta name="description" content="描述信息内容">
```

```
描述信息不要过长，否则搜索引擎会以...省略
```

- K-keywords 网页关键词

```
<meta name="keywords" content="关键词1,关键词2">
```

```
关键词之间用英文逗号隔开
```

- 网页标题栏或收藏夹图标

```
<link rel="icon" href="*.ico">
```

css reset（样式重置）作用

```
网页在显示的过程中，由于浏览器的内核不同、版本不同，对标签自带样式属性值的解析也不同。为了在不同内核、不同版本的浏览器中，都能解析到相同的属性值，需要将所有元素的自带样式去掉，然后重置。
```

- outline属性 轮廓
  位于边框边缘的外侧，语法与border类似

```
    outline: 5px solid red;
    outline: none; 去掉元素的轮廓
```

- outline-offset属性 对轮廓偏移

```
    outline-offset: -5px;
```

拆分原则

1像素原则
按照内容去拆
先上下后左右，从外向里
先整体后局部，再到细节

BFC

1、什么是BFC

- BFC--block formatting context 块级格式化上下文，W3C css2.1规范中的概念
- 页面中的一个渲染区域
- web页面可视化css视觉渲染的一部分，用于决定块级盒子的布局及浮动相互影响范围的一个区域
- 本质：一个封闭的盒子

2、怎样生成BFC?

- 根元素
- float: left|right;
- position: absolute|fixed;
- display: inline-block;
- overflow: hidden|auto|scroll;

3、BFC的特性

- 内部元素会在垂直方向上一个接一个放置
- 垂直方向上的外间距由margin值决定，属于同一个BFC的两个相邻元素的margin值会重叠
- 每个元素的左外间距和包含块的左边界相接触，浮动元素也是如此
- BFC区域不会与浮动元素区域重叠
- 计算BFC高度时，浮动元素也参与计算
- BFC是页面上的一个独立的渲染区域，容器中子元素不会影响到外面的元素，反之亦然

4、BFC解决的问题

- 1）外间距重叠问题
- 2）清浮动
- 3）自适应两栏或三栏布局
    - 自适应两栏布局：左侧宽度固定，右侧不设宽，则右侧宽度为自适应，随浏览器窗口大小变化而变化
    - 自适应三栏布局：左右两侧宽度固定，中间不设宽，则中间宽度自适应，随浏览器窗口大小变化而变化
- 4）防止文字环绕

CSS小箭头

1、原理

- 使用css绘制两个三角形
- 通过绝对定位让两个三角形不完全重叠
- 让处于上层的三角形比处于下层的三角形偏移属性少1px，就可以生成空心箭头

2、兼容

在IE6及更早版本的浏览器中添加font-size: 0; line-height: 0; 目的是为了让三角的height:0; 有效

3、实现

```
结构中：向下箭头
    <div class="arrowDown">
        <i class="arr"></i>
        <i></i>
    </div>
样式中：
    .arrowDown i{/* 向下的两个三角 */
        width: 0;
        height: 0;

        font-size: 0;
        line-height: 0;

        border-top: 5px solid #f5f5f5;
        border-left: 5px solid transparent;
        border-right: 5px solid transparent;

        position: absolute;
        left: 0;
        top: -1px;
    }

    .arrowDown .arr{ /*两个三角错位*/
        border-top-color:#d4d4d4;
        top: 0;
    }
```

iconfont 字体图标

阿里矢量图标库

- 1、通过检索界面选择需要的图标，并添加到购物车
- 2、打开购物车--选择“下载代码”
- 3、将下载好的文件夹中的字体文件（.eot|.svg|.ttf|.woff|.woff2）放入项目的fonts文件夹下
- 4、将下载好的iconfont.css文件，放入css文件夹，并修改css文件中的字体文件路径（../fonts/iconfont.eot）
- 5、将修改好的iconfont.css文件链接到HTML文档中
- 6、在HTML文档中添加指定类名iconfont和图标的名称（□）或在标签中添加类名iconfont和图标对应的类名icon-XX

```
    <p class="txt1"><span class="iconfont">&#xe63e;</span></p>
    <p class="txt2"><span class="iconfont icon-sousuo"></span></p>
```

> *需要通过文字样式修改字体图标的大小和颜色*

文本溢出处理

1、单行文本溢出显示省略号

- 使用的属性
    - text-overflow: clip; 文字溢出后直接被裁切
    - text-overflow: ellipsis; 文本溢出后用省略号表示被裁切的文本
    - text-overflow: string; 文本溢出后用给定的字符串表示被裁切的文本，仅在火狐中生效
- 实现

```
.box{
    width: 200px;
    white-space: nowrap;  /*强制不换行*/
    overflow: hidden;
    text-overflow: ellipsis;/*文本溢出后用省略号表示*/
}
```

2、多行文本溢出显示省略号

- 使用webkit的css扩展属性——只在-webkit-内核的浏览器中有效

```
    .box2{
        width: 200px;
        overflow: hidden;
        text-overflow: ellipsis;

        display: -webkit-box;/* 将元素转换为弹性伸缩盒子 */
        -webkit-line-clamp: 2;/* 显示的文本内容的行数 */
        -webkit-box-orient: vertical;/* 设置弹性伸缩盒中子元素的排列方式
*/

    }
```

- 使用伪元素模拟溢出显示省略号——兼容性较好

```
    .box3{
        position: relative;
        width: 200px;
        height: 40px;/* height是line-height属性值的倍数*/
        overflow: hidden;
```

```
        line-height: 20px;
    }
    .box3::after{/* 模拟省略号 */
        content: "...";
        position: absolute;
        right: 6px;
        bottom: 0;
        background: #fff;
        padding: 0 3px;
    }
```

HTML表单

- 用于采集用户输入数据，发送给服务器，实现用户和服务器之间的数据交互
- 一个完整的表单通常包含表单元素、提示信息、表单域等3个部分

1、表单标签

- 用于申明表单，定义数据采集范围
- 一个页面中可以有多个或一个表单标签，标签之间相互独立，不能嵌套
- 用户向服务器提交数据一个只能提交一个表单中的数据；如果要提交多个表单，需要使用JavaScript中的异步交互方式
- 语法

```
<form action="提交表单时向何处发送表单数据URL" method="get|post"
name="表单名称">表单控件</form>
```

- 属性
    - method属性：提交表单使用的HTTP方法；默认get方式
        - get方式：将数据作为URL地址的一部分发送服务器；安全性较低；请求数据可以被缓存，能保留在浏览器的历史记录中，可以作为书签被收藏；有长度限制。
          https://www.baidu.com/?username1=123&username2=ABC
          https://www.baidu.com/?参数名1=参数值1&参数名2=参数值2
        - post方式：将数据隐藏在HTTP数据流中进行传输；安全性比get方式高；请求数据不会被缓存，不能保留在浏览器的历史记录中，不会作为书签被收藏；对数据没有长度限制
    - name属性
      如果表单元素中不设置name属性，输入框中的内容无法随表单一起提交到后台

2、表单控件

- input标签

```
<input type="" name="" value="" />
    type属性：用来设置不同的控件类型
```

-
- 单行文本输入框
`<input type="text" name="" value="" placeholder="提示信息文本">`
- 密码框（默认掩码处理）
`<input type="password" name="" placeholder="提示信息文本">`

```
- 按钮--提交按钮
```
<input type="submit" value="">
```

> value属性：用于修改按钮上的文字
```

css滑动门

核心技术：css sprites(背景图位置)、padding属性撑开盒子宽度，能够适应不同字数的导航栏

多栏布局解决方案

- 什么是自适应
  让同一个页面自动适应不同大小的设备，解决为不同设备提供不同版本的页面问题
- 自适应布局
  大部分自适应布局指的是宽度自适应布局，解决的是在不同大小的设备上呈现相同网页的问题
  页面元素位置发生变化，元素不随着窗口大小的调整而变化

1、两列自适应布局——左侧宽度固定，右侧宽度自适应

- 1）左右两个盒子，左侧宽度固定，右侧宽度设置100%
- 2）左侧盒子设置绝对定位position: absolute;
- 3）右侧盒子中添加子盒，并为子盒设置padding-left属性，值为左侧盒子的宽度

2、圣杯布局

左右两列宽度固定，中间部分自适应的三列布局

- 1）HTML结构中——先主体内容后侧边栏(先左后右)
- 2）两侧宽度固定，中间宽度设置100%
- 3）主体内容和左右两侧内容分别加浮动 float:left;
- 4）将左侧盒子拉到最左边（margin-left:-100%）；将右侧盒子拉到最右边（margin-left: -右侧盒子的宽度）
- 5）通过左、右、中盒子的父级盒将中间内容露出来（在父级盒上设置padding: 0 右侧盒子宽度 0 左侧盒子宽度）
- 6）分别还原左侧盒和右侧盒子（分别为左侧和右侧盒子设置position:relative; 左侧盒子设置left属性，属性值为负左侧盒子宽度，右侧盒子设置right属性，属性值为负右侧盒子宽度）

3、双飞翼布局

左右两列宽度固定，中间部分自适应的三列布局

- 1)HTML结构中——先主体结构（主体结构中要有主体内容盒）后侧边栏（先左后右）
- 2）两侧盒子设置固定宽度，中间主体结构盒子宽度为100%
- 3）主体结构盒、左侧盒子、右侧盒子设置浮动float: left;
- 4）将左侧盒子拉到最左边（margin-left: -100%),将右侧盒子拉到最右边（margin-left: -右侧盒子的宽度）
- 5）在主体内容盒子中设置margin值将中间内容露出来（margin: 0 右侧盒子的宽度 0 左侧盒子的宽度）

4、等高布局

实现等高的视觉效果

1）利用内外间距相抵消，父元素设置overflow:hidden; 实现

- 实现：
  每一列使用padding-bottom属性撑开背景颜色；将每一列用padding值撑开的多余的占位让margin负值抵消；父元素设置溢出隐藏
- 优点：
  结构简单，兼容所有浏览器
- 缺点：
  伪等高，扩展性较差，需要合理控制margin和padding值

2）利用内容撑开父元素的特点，实现等高布局

嵌套元素内容撑开外部所有的父元素的高度

- 实现
  三个嵌套div负责背景，三列放在最内部的div盒子中；
  通过相对定位，分配三列背景的位置；
  通过margin负值，将内容移动到对应的背景位置；
  父元素设置溢出隐藏

- 特点：
  结构嵌套复杂，扩展行较好，可以实现自适应

阴影

1、盒子阴影

- 语法

```
box-shadow: h-shadow v-shadow blur spread color inset;
box-shadow: x轴偏移量 y轴偏移量 阴影模糊值 [阴影大小] 阴影颜色 [内阴影inset]|外阴影（默认）
```

> *多个阴影之间用逗号隔开*

2、文字阴影

- 语法：

```
text-shadow: h-shadow v-shadow blur color;
text-shadow: x轴偏移量 y轴偏移量 阴影模糊值 阴影颜色;
```

> *多个阴影之间用逗号隔开*

HTML5基础

1、什么是HTML5？

- web 1.0静态页面 --> web 2.0 交互 --> HTML5时代
  HTML4 XHTML1.0
- HTML的升级版本，不仅用来表示web内容，而且将web带入一个更加成熟的平台。在HTML5平台上，音频、视频、图像、动画、交互等都被标准化
  2004年 提出
  2007年 被W3C接纳 成立HTML工作组
  2008年 第一份正式草案
  2014年 W3C HTML工作组 发布第一份官方推荐标准
  2020年 完成最终测试
  2022年 正式发布

2、HTML5中的新特性

- 新增的语义化标签
- 新增的表单元素和表单属性
- 新增的在网页上绘制图像的canvas元素
- 新增了多媒体相关的video、audio元素
- 对本地离线存储的更好支持
    - 本地存储：提供两种客户端存储数据的方法——localStorage、sessionStorage
    - 离线应用

3、HTML5新增标签

1）新增结构标签

- 网页的头部区域或某个模块的头部
    - 一种具有引导和导航作用的结构元素，定义文档或某个区域的头部信息。通常包含整个页面或一个内容块的标题、搜索框、LOGO等

```
<header></header>
```

- 网页的底部区域或某个模块的底部
    - 定义文档或文档中某个区域的底部信息，通常包含网页中的版权信息、相关链接、文档的作者信息、使用条款链接等等

```
<footer></footer>
```

- 导航
    - 定义导航链接的部分，通常包含页面中的主要的导航链接（传统导航、侧边栏导航、页内导航、翻页操作等）

```
<nav></nav>
```

- 页面中独立、完整的内容
    - 可以包含header标签，一篇有自身含义的文章。通常包含一篇博文、论坛帖子、报刊中的文章、一段用户评论或独立的插件

```
<article></article>
```

- 页面中内容的章节
    - 通常一个section由标题和内容组成

```
<section></section>
```

- 侧边栏

- 当前页面或文章的附属信息，通常包含与主要内容相关的引用、侧边栏、广告、链接组等

```
<aside></aside>
```

- 标题组
    - 可以作为标题或子标题的分组，通常与h1~h6组合使用

```
<hgroup></hgroup>
```

- 文章中的联系信息
    - 通常包含文档作者或文档的编辑者的名字、电子邮箱、电话号码、地址等

```
<address></address>
```

> *块级标记、文字自带倾斜效果*

2）新增其他标签

- figure标签
    - figcaption标签，在一个figure标签中只能有一个figcaption标签

```
<figure>
    被主体内容引用的内容，相对独立的内容块，如：图片、代码块、图表等
    <figcaption>定义figure标签的标题</figcaption>
</figure>
```

> *自带间距、块级元素*

- 带有标记的文本
    - 页面中突出显示的，高亮的部分
    - 行级标记、自带背景颜色（黄色）、自带文字颜色（黑色），颜色可修改

```
<mark></mark>
```

- 日期时间标签
    - 公历日期时间
    - 行级标记

```
<time>2020-9-18</time>
<time pubdate="pubdate" datetime="2020-9-18"></time>
    pubdate属性：当前内容的发布时间
    datetime属性：日期时间；如果没有定义该属性，则必须在time标签中设置日期时间
```

- 进度条
    - 行块级元素

```
<progress max="" value=""></progress>
    max属性：最大值
```

4、HTML5兼容

- 最新版本的chrome、Firefox、Opera、Safari、IE10+支持HTML5
- IE9支持部分HTML5特性
- IE8及更早版本的浏览器中对HTML5新增元素有兼容问题

1）使用JavaScript新增元素的方法解决

```
结构中：<div id="box"></div>
javascript：
    <script>
        //1、新建一个header元素
        var ele = document.createElement("header");//新建header元素放在ele容器中

        //2、找到#box的盒子
        var oBox = document.getElementById('box');//找到#box盒子，放在oBox容器中

        //3、在#box盒子(oBox)中追加新建的header的元素(ele)
        oBox.appendChild(ele);
    </script>
样式中：
    将新建的header标签转为块级元素
```

2）使用谷歌提供的html5shiv.js插件解决

```
用于解决IE9及以下版本的浏览器对HTML5新增元素的不兼容问题
> 必须写在网页的头部（head标签内）
```
<!--[if lte IE 9]>
    <script src="js/html5shiv.min.js"></script>
<![endif]-->
```

## 5、HTML5中已经移除的标签

- acronym标签：首字母缩写
- applet标签：嵌入applet
- font标签：定位字体、颜色、字号
- basefont标签：默认字体、颜色、字号
- big标签：大号文本
- center标签：文本居中
- s标签：删除线文本
```

- strike标签： 删除线文本
- u标签：下划线文本
- dir标签：目录
- frame标签：框架或窗口
- frameset标签：框架集
- noframes：不支持框架的用户替代内容

...

6、新增多媒体标签

1）audio标签

- HTML5中新增的音频的标准方法
- IE8及更早版本的浏览器中不支持
- HTML5中支持的音频格式
    - ogg audio/ogg 支持浏览器：chrome、Firefox、Opera10+
    - mp3 audio/mpeg 支持浏览器：chrome、Firefox、Opera10+、IE9+、Safari
    - wav audio/wav 支持浏览器：chrome、firefox、Opera、Safari
- 语法：

```
<audio src="音频文件路径">您的浏览器不支持audio元素，播放音频文件</audio>
```

- 常用属性
    - src属性：音频文件的路径
    - controls属性：浏览器为音频提供的播放控件
    - loop属性：循环播放
    - muted属性：静音

source标签

可以链接不同格式的文件（音频、视频），浏览器使用第一个可以被识别的格式

```
<audio controls>
    <source src="videoAudio/nada1.wav">
    <source src="videoAudio/biubiubiu2.ogg">
    <source src="videoAudio/hanmai.mp3">
    您的浏览器不支持audio元素，播放音频文件
</audio>
```

2）video标签

- HTML5中视频标准方法
- IE8及更早版本浏览器中不支持video标签
- HTML5支持的视频格式：
    - ogg 支持浏览器：chrome、Firefox、Opera
    - mp4--MPEG4 支持浏览器：chrome、Firefox、Safari、IE9+
    - webM 支持浏览器：chrome、Firefox、Opera

- 语法

```
<video src="视频文件的路径">您的浏览器不支持video元素，播放视频文件
</video>
```

- 常用属性
    - src属性：视频文件路径
    - controls属性：播放控件
    - loop属性：循环播放
    - muted属性
    - width属性、height属性：视频播放器的宽度和高度
    - poster属性：预览图片，视频正在下载时显示的图像

7、新增表单元素及属性

1）新增表单元素

- 包含访问协议的完整的路径的输入域，在提交表单时，自动验证url域的内容

```
<input type="url" placeholder="请输入地址" name="userUrl">
```

- 包含e-mail地址的输入域，在提交表单时，自动验证email域的值

```
<input type="email" placeholder="请输入邮箱地址" name="e-mail">
```

- 用于搜索关键字的文本输入域，一般用于手机客户端

```
<input type="search" placeholder="请输入要检索的内容">
```

- 用于输入电话号码的文本输入域，用于触屏网页开发，在电脑网页中无效，在触屏网页中点击输入域，弹出虚拟电话键盘（0-9、*、#）

```
<input type="tel" placeholder="请输入电话号码">
```

- 用于包含数值的输入域

```
<input type="number" max="10" min="0" value="0" step="2">
    max属性：最大值
    min属性：最小值
    step属性：步长，合法的数字间隔，默认为1
```

> *当用户输入的数值不在制定范围内时，会弹出相关提示信息，并阻止表单提交*

- 用于生成一个数字滑动条，跟number类型相似，区别在于外观样式不同，默认值不同。

```
<input type="range" max="10" min="0" value="0" step="2">
```

> *range类型的min值默认0，max属性默认为100*

- 用于生成一个颜色选择器，值为十六进制色值

```
<input type="color">
<input type="color" value="#ff0000">
```

input的日期时间类型

- 从一个日期选择器中选择一个日期，包含年、月、日

```
<input type="date">
```

- 手动输入一个日期时间（UTC时间）
  1884年 华盛顿 国际经度会议 划分24个时区 英国格林尼治天文台为中时区， 中国UTC+8 东8区 UTC-10

```
<input type="datetime">
```

- datetime-local类型 ，从一个日期时间选择器中选择一个日期时间，包含年、月、日、时、分

```
<input type="datetime-local">
```

- month类型，从一个日期选择器中选择一个月份，包含年、月

```
<input type="month">
```

- time类型，从一个时间选择器中选择一个时间，包含时、分

```
<input type="time">
```

- week类型，从一个日期选择器中选择周，包含年、周（全年的第几周）

```
<input type="week">
```

- datalist标签：可选下拉列表，需要与input标签配合使用，通过input标签的list属性和datalist标签的id属性关联

```
<input list="con">
<datalist id="con">
    <option value="abcd">abcd</option>
    <option value="123">123</option>
</datalist>
```

2）新增表单属性

- placeholder属性
  设置文本域的提示信息，当用户开始输入内容，提示信息消失
- min、max、step属性
  最小值、最大值、合法的数字间隔
  用于包含数字的input类型，规定数值范围，如：input的numbet、range类型等
- list属性
  实现数据列表的下拉效果
  用于input标签和datalist标签关联，实现可选下拉列表
- autocomplete属性
  设置表单是否可以启用自动完成功能，可以加在form标签、input标签上
    - autocomplete="on" 启用自动完成功能
    - autocomplete="off" 不启用自动完成功能

    > *必须在input标签中添加name属性*

```
    <form action="#" autocomplete="off">
        <div><input type="text" placeholder="请输入用户名"
name="user"></div>

        <div><input type="text" placeholder="请输入昵称"
name="userName" autocomplete="on"></div>

        <hr>
        <input type="submit">
    </form>
```

- autofocus属性
  页面加载时自动获取焦点，一个页面中只能有一个表单元素具有这个属性

```
<input type="text" placeholder="请输入昵称" name="userName"
autofocus>
```

- required属性
  提交表单时元素不能为空

    > *要让属性有效，则不能添加value值*

```
<input type="text" required>
```

- pattern属性
  验证输入内容——正则表达式

    ```
    <input type="text" name="content" placeholder="请输入5位字母
    数字" pattern="[0-9a-zA-Z]{5}">
    ```

- multiple属性
  可以选择多个值
  用于上传域file类型和email类型的文本输入域

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

- form属性
  设置输入域所属的表单

```
<form action="#" id="wrap">
    <div><input type="text" placeholder="请输入用户名"
name="user"></div>
</form>
<div><input type="submit" form="wrap"></div>
<div><input type="text" name="user2" form="wrap"></div>
```


css3基础

1、css3简介

1）什么是CSS3？

- 是css的升级版本，在css2的基础上，新增了一些新特性，弥补css2的不足，让web页面开发变得更加高效、便捷
- css3按照模块进行设计，比较重要的模块：
  选择器、框模型、背景和边框、文本效果、2D|3D转换、动画、多列布局、用户界面
- css3现状：
  - 部分属性在浏览器中需要添加兼容性前缀
  - 移动端支持优于PC端
  - 不断更新中
  - 应用相对广泛
- css3向后兼容
- css2和css3的区别
  - css3能使代码更加简介、页面结构更加清晰、合理，性能和效果得到兼顾；css2要请求服务器的次数要高于css3，性能和访问相对于css3要差
  - css3能实现动画效果；css2更加偏向于表现
  - css3数据更加精简使用，许多在css2中需要使用图片实现的效果，在css3中可以通过代码实现
  - 兼容性问题，css2中所有的属性都能在浏览器中呈现；css3中部分属性在主流浏览器的最新版本中仍然需要添加兼容性前缀

2）渐进增强和优雅降级

- 渐进增强：针对低版本浏览器进行页面构建，保证最基本的功能，然后再最对高版本的浏览器进行交互、效果之类的改进，追加一些功能，达到更高的用户体验——普通到华丽
- 优雅降级：一开始构建一个完整的功能，然后再针对低版本浏览器进行兼容——华丽到普通

区别

```
- 渐进增强：从基础到复杂，并能适应未来的环境
- 优雅降级：从复杂到基础，减少用户体验
```

2、兼容性前缀

- 浏览器生产商对新属性或规则的提前支持，暗示这个属性或规则没有成为W3C标准的一般部分
- 书写规则：先写私有css属性或规则，最后写标准css属性
- 浏览器、内核、私有前缀（兼容性前缀）

浏览器	内核	私有前缀
chrome、safari	webkit	-webkit-
opera	Presto	-o-
firefox	Gecko	-moz
IE	Trident	-ms-

- 当一个属性或规则成为标准，并被主流浏览器的最新版本普遍兼容的时候，可以去掉兼容性前缀
- autofix
    - 在VScode中安装autoprefixer插件
    - 在外部的css文件中书写相关的属性和属性值
    - 按下F1，选择autoprefixer:run,之后会在css文件中添加|删除私有前缀

3、选择器

1）属性选择器

- css2中属性选择器
    - element[attr]{ } 指定了属性名，但是没有指定属性值的element元素
    - element[attr=value]{ } 指定了属性名，并且指定属性值的element元素
    - element[attr~=value]{ } 指定了属性名，属性值可以是词列表，在词列表中包含指定value值的element元素
- css3中新增的属性选择器
    - element[attr^=value]{ } 指定了属性名attr,并且属性值为value开头的element元素
    - element[attr$=value]{ } 指定了属性名attr,并且属性值为value结尾的element元素
    - element[attr*=value]{ } 指定了属性名attr,并且属性值包含value的element元素

2)结构伪类选择器

- :root{} 匹配根元素
- ele:first-child{ } 选择一组相同元素中的第一个元素——css2
- ele:last-child{ } 选择一组相同元素中的最后一个元素
- ele:nth-child(n){ } 选择一组相同元素中的第n个元素。n取值可以是数值、关键词、表达式
    - 数值
    - 关键词：odd奇数 even偶数
    - 表达式：2n+1奇数 2n表示偶数
- ele:nth-last-child(n){ } 选择一组相同元素中的倒数第n个元素。n取值可以是数值、关键词、表达式
- ele:nth-of-type(n){ } 选择一组元素中特定类型的第n个ele元素。n取值可以是数值、关键词、表达式
- ele:nth-last-of-type(n){ } 选择一组元素中特定类型的倒数第n个ele元素。n取值可以是数值、关键词、表达式

3)状态伪类选择器

- ele:checked{ } 选择处于选中状态的ele元素
- ele:disabled{ } 选择处于禁用状态的ele元素
- ele:enabled{ } 选择处于可用状态的ele元素

4、背景

1）多背景

```
background-image:url(图片路径), url(图片路径);
```

- 多个背景图片之间用逗号隔开
- 显示越靠前，书写顺序越靠前

2）背景图片尺寸设置

- background-size属性：数值
    - background-size: 300px;/* 按照宽度300像素，高度自适应 */
    - background-size: 200px 100px;/* 按照给定的尺寸设置背景图片大小 */
    - background-size: 100% 100%;/* 背景图片撑满整个盒子 */
- background-size属性：关键词
    - background-size: cover;覆盖，等比例缩放背景图片到铺满整个盒子，但是背景图片可能会无法完整的显示在盒子中
    - background-size: contain;包含，等比例缩放背景图片到完整显示在盒子中，但是背景图片可能在区域内铺不满

```
background:url(images/3.jpg) no-repeat;
background-size: cover;
简写：
background: url(images/3.jpg) no-repeat left top/cover;
```

3）背景图片定位区域——backgorund-origin属性

```
- background-origin: content-box; 背景图片相对于内容区域来定位
- background-origin: padding-box; 默认值，背景图片相对内填充区域来定位
- background-origin: border-box; 背景图片相对于边框区域来定位
```

4）背景颜色绘制区域——background-clip属性

```
- background-clip: content-box; 背景颜色仅在内容区域显示
- background-clip: padding-box;背景颜色在padding区域和内容区域显示
- background-clip: border-box; 默认值，背景颜色在padding区域、内容区域、border区域显示
```

5、渐变

从一种颜色到另外一种颜色的过渡（两种及以上）

1）线性渐变 linear-gradient

- 语法

```
background-image: linear-gradient(方向, 颜色1 范围1, 颜色2 范围2, 颜色3 范围3, ...);
    方向：数值（单位deg角度）、关键词（left|right top|bottom）
    颜色：关键词、十六进制色织、rgb()、rgba()
    范围：每个颜色结点显示的范围
```

- 重复线性渐变

```
background-image: repeating-linear-gradient(方向, 颜色1 范围1, 颜色2 范围2, 颜色3 范围3, ...);
background-image: -webkit-repeating-linear-gradient(方向, 颜色1 范围1, 颜色2 范围2, 颜色3 范围3, ...);
```

2）径向渐变 radial-gradient

- 语法

```
background-image: radial-gradient(中心位置，渐变形状，颜色1 范围1, 颜色2 范围2, ...);
    中心位置： left|center|right   top|center|bottom
    渐变形状：椭圆(默认值 ellipse)、圆形（circle）
```

- 重复径向渐变

```
background-image: repeating-radial-gradient(中心位置，渐变形状，颜色1 范围1，颜色2 范围2，...);
```

> *如果不设置径向渐变的中心位置，默认位置为center*

6、用户界面

1）resize属性：用户是否可以对元素尺寸进行调整

- resize:none; 不允许用户手动调整元素的尺寸
- resize:both; 用户可以手动调整元素的宽度和高度
- resize: vertical; 用户可以手动调整元素的高度
- resize: horizontal; 用户可以手动调整元素的宽度

> *resize属性要生效，必须与overflow属性配合使用，属性值可以是hidden|auto|scroll*

2) box-sizing属性

- box-sizing: content-box;默认值，在width属性和height属性之外绘制元素的内填充和边框
- box-sizing: border-box; 通过已有的设置好的width属性和height属性中分别减去内填充和边框，得到内容的宽度和高度

7、多列布局

- 元素被分割的列数——column-count属性
    - column-count：n; 被划分为n列，没有单位
    - column-count: auto; 由其他属性决定列数
- 列宽——column-width属性
    - column-width: npx; 每一列宽度设置
    - column-width: auto; 由其他属性决定列宽
- 列间距——column-gap属性
    - column-gap: npx; 两列之间的间隔
    - column-gap: normal; 两列之间为常规间隔，W3C建议1em
- 列间分割线——column-rule属性
    - column-rule: column-rule-width column-rule-style column-rule-color;

    > *border、outline*

- 跨列合并——column-span属性
    - column-span: all; 横跨所有列合并

过渡效果属性——transition属性

语法

- transition-property属性：参与过渡的属性 all表示所有属性都有过渡效果
- transition-duration属性：过渡效果执行的时间，默认执行时间为0，属性值单位s|ms
- transition-timing-function属性：速度曲线
    - ease 默认值 平滑过渡
    - ease-in 加速
    - ease-out 减速
    - ease-in-out 先加速后减速
    - linear 匀速
- transition-delay属性：延迟时间，默认时间为0，属性值单位s|ms
- 简写

```
transition: transition-property transition-duration  transition-timing-function  transition-delay;
```

> *参与的属性可以写多组，多个属性名称之间用逗号隔开*

```
transition-property: width,background;
```

2D|3D转换——transform属性

2D转换

- transform: translate(); 位移
    - transform: translate(x,y); 沿着x轴方向和y轴方向位移
    - transform: translateX(); 沿着x轴方向移动
    - transform: translateY(); 沿着y轴方向移动

    > *属性值：向右、向下为正*
    > *transform: translate(100px);/* 沿着水平方向位移 */*

- transform: rotate(); 旋转
    - transform: rotate(ndeg);

    > *属性值：单位deg角度*
    > *正值沿着顺时针方向旋转，负值沿着逆时针方向旋转*

- transform: scale()缩放
    - transform: scale(x,y) 沿着x轴方向和y轴方向缩放
    - transform: scaleX(x) 沿着x轴方向缩放
    - transform: scaleY(y) 沿着y轴方向缩放

> *属性值：没有单位，0~1之间表示缩小，1表示正常大小，1以上表示放大*
> *transform: scale(2);/* 等比例缩放 /*
> *transform: scale(-2);/ 先翻转后缩放 */*

- transform: skew() 倾斜
    - transform: skew(x,y) 沿着x轴和y轴倾斜
    - transform: skewX(x) 沿着x轴倾斜
    - transform: skewY(y) 沿着y轴倾斜

> *属性值：单位deg角度*
> *transform: skew(30deg);/* 沿着水平方向倾斜 */*

transform-origin属性：改变基点位置

- 属性值
    - 关键词：left|center|right top|center|bottom
    - 数值，单位可以是px|em|rem|%

3D变形

- 必须要设置的属性
    - transform-style属性
        - transform-style: preserve-3d;创建一个3d空间
    - perspective属性：景深，单位像素

> *需要设置在父元素上*

- 位移
    - transform: translateX(); 沿着x轴方向位移
    - transform: translateY(); 沿着y轴方向位移
    - transform: translateZ(); 沿着z轴方向位移
    - transform: translate3d(x,y,z)
- 旋转
    - transform: rotateX(); 沿着x轴方向旋转
    - transform: rotateY(); 沿着Y轴方向旋转
    - transform: rotateZ(); 沿着z轴方向旋转
    - transform: rotate3d(x,y,z,ndeg)
- 缩放
    - transform: scaleX() 沿着x轴方向缩放
    - transform: scaleY() 沿着y轴方向缩放
    - transform: scaleZ() 沿着z轴方向缩放
    - transform: scale3d(x,y,z)

> *实现元素既缩放又旋转*

```
transform: scale(0)  rotate(360deg);
```

动画

1、帧动画

- 第一步：定义帧动画

```
@keyframes 动画名称（英文）{
    0%{ 动画开始 }
    50%{}
    100%{ 动画结束 }
}
@keyframes 动画名称{
    from{ 动画开始 }
    to{ 动画结束 }
}
```

- 第二步：引用关键帧

```
animation: animation-name animation-duration [animation-timing-function] [animation-delay] [animation-iteration-count] [animation-direction] [animation-fill-mode];
animation: 动画的名称 动画执行时间（s|ms） 动画的速度曲线（ease|ease-in|ease-out|ease-in-out|linear） 延迟时间 动画执行的次数 动画如何播放 动画结束之后显示的状态;
    动画执行的次数：默认为1，infinite无限循环
    动画如何播放：
        normal 正常播放
        alternate 正向、反向轮流播放
        reverse 反向播放
        alternate-reverse 反向、正向轮流播放
    动画结束之后显示状态：
        both 动画结束或开始时的状态
        forwards 动画结束时的状态
        backwards 动画开始时的状态
```

2、animate.css动画库

```
css3动画库，预设了闪烁（flash)、弹跳（bounce）、翻转（flip)、旋转（rotateIn|rotateOut）、淡入淡出（fadeIn|fadeOut）、抖动（shake）等动画效果。
```

- 使用方法
    - 1）引入文件：
    - 2）使用：

    内容

    animate**animated**为基本类名，必须要添加
    animatebounceIn为指定动画样式名称

css3中过渡和动画的区别和各自适用场景？

区别：语法、触发、执行次数、复杂度

calc()函数

```
css3中新增功能，用于动态计算长度值
流体布局：屏幕分辨率变化时，页面中元素的大小会变化，但是布局不变
```

- 语法

```
calc(表达式)
    可以用来实现加、减、乘、除四则运算
```

> *运算符的前后需要添加空格*

```
    width: calc(100px +100px);/* 语法错误 */
    width: calc(100px+100px);/* 语法错误 */
```

> *运算规则：有括号先算括号里面的，先算乘除后算加减*

```
width: calc(1000px - (100px + 100px) * 2);/*600px*/
```

- 兼容
  在IE9+、firefox、chrome、Safari可以正常呈现
- 利用calc函数实现三列自适应布局

弹性盒子

1、什么是弹性盒子？

- 弹性盒子 弹性布局、flex布局
- 传统布局
    - 兼容性好
    - 布局繁琐
    - 局限性，不会在移动端布局中有较好的呈现
    - PC端页面中更多的使用传统布局
- flex布局
    - css3新增布局方式——一种当页面需要使用不同屏幕大小、不同设备类型时，能够保证元素拥有恰当的布局方式
    - 操作方便，布局简单，移动端使用广泛
    - PC端浏览器支持情况较差
    - IE11及更低版本的浏览器中，不支持或部分支持

        - 在盒模型中较为灵活
    - 弹性盒模型的内容包含：弹性容器、弹性子元素（项目）
    - 引入弹性盒模型的目的：用一种更加有效的方法对容器中的子元素进行排列、对齐、分配空白空间，即使是弹性子元素的尺寸发生改变，弹性盒模型也可以正常工作
    - 原理：为父元素设置flex属性，控制子元素的位置及排列方式

2、设置弹性盒子——display属性

- display: flex; 将块级元素设置为弹性盒
- display: inline-flex; 将盒子设置为弹性盒

> *为父元素设置flex属性，子元素中的float、clear、vertical-align属性都会失效*

3、基本概念

- flex容器　项目——弹性子元素
- 默认在容器中有两根轴线
    - 默认主轴方向——水平方向，水平向右（左侧为主轴的起点，右侧主轴的终点）
    - 默认交叉轴方向——垂直方向，垂直向下（上方是交叉轴的起点，下方是交叉轴终点）

4、容器属性——添加在弹性容器上

- flex-direction属性：设置主轴的方向，子元素的排列方向
    - flex-direction: row; 默认值，主轴方向为水平方向，起点在左侧，排列次序与文档顺序一致
    - flex-direction: row-reverse; 主轴方向为水平方向，排列次序与文档顺序相反
    - flex-direciton: column; 主轴方向为垂直方向，起点在上方，排列次序与文档顺序一致
    - flex-direction: column-reverse; 主轴方向为垂直方向，排列次序与文档顺序相反
- justify-content属性：弹性子元素在主轴方向上的对齐方式
    - justify-content: flex-start; 默认值，子元素位于弹性容器的开头
      -justify-content: flex-end; 子元素位于弹性容器的结尾
      -justify-content: center; 子元素位于弹性容器的中间
    - justify-content: space-between; 子元素和子元素之间有空白空间
    - justify-content: space-around; 子元素之前、之间、之后都留有空白空间
- align-items属性：弹性子元素在交叉轴方向上的对齐方式
    - align-items: stretch; 默认值，如果弹性子元素没有高度或高度为auto，将占满整个容器的高度
    - align-items: flex-start; 子元素位于交叉轴的起点
    - align-items: flex-end; 子元素位于交叉轴的终点
    - align-items: center; 子元素位于交叉轴的中间
    - align-items: baseline; 子元素在第一行文字的基线对齐
- flex-wrap属性：弹性子元素在一根轴线上放不下时的换行方式
    - flex-wrap: nowrap; 默认值，不换行

        - flex-wrap: wrap; 换行，第一行在上方显示
        - flex-wrap: wrap-reverse; 换行，第一行在下方显示
    - align-content属性：多根轴线的对齐方式，如果只有一根轴线，属性失效
        - align-content: flex-start; 与交叉轴的起点对齐
        - align-content:flex-end; 与交叉轴的终点对齐
        - align-content:center; 与交叉轴的中间对齐
        - align-content:space-between; 轴线之间的间距
        - align-content:space-around; 轴线两侧都有间距
        - align-content: stretch; 默认值，轴线占满整个交叉轴

5、项目属性——写在弹性子元素上

- order属性：子元素的排列次序
    - 属性值：数值，没有单位，默认数值为0
    - 数值越小，排列越靠前
- flex-grow属性：子元素的放大比例
    - 属性值：数值，没有单位
    - 默认值为0，表示不放大
- flex-shrink属性：子元素的缩小比例
    - 属性值：数值，没有单位
    - 默认值为1，表示当空间不足时，等比例缩小
    - 属性值为0，表示当空间不足，不缩小
- flex属性：
    - flex-grow,属性值为0，不放大，flex-shrink属性值为0 不缩小
    - flex: flex-grow flex-shrink;
- align-self属性：弹性容器中，被选中的弹性子元素的对齐方式
    - align-self: auto; 默认值，元素继承了父容器的align-items属性，如果父容器没有设置则属性值为stretch
    - align-self: stretch; 占满整个容器的高度
    - align-self: flex-start; 交叉轴的起点对齐
    - align-self: flex-end; 交叉轴的终点对齐
    - align-self: center; 交叉轴的中点对齐
    - align-self: baseline; 子元素的第一行文字的基线对齐

预处理

1、预处理

- 一种新的语言
- 基本思想：用一种专门的编程语言，为css增加一些变成特性，将css文件作为目标生成文件，然后程序员使用预处理语言进行编码工作

2、Less

- 包含一套自定义的语法及一个解析器，用户根据语法定义自己的样式规则，这些规则最终通过解析器编译生成对应css文件，只有被编译后的文件才能被浏览器识别
- 扩展css语言，增加了一些新的功能，如：定义变量、混合、函数、运算等
- 好处：
  结构清晰，便于扩展；可以方便地屏蔽浏览器私有语法差异；可以实现多重继承；完全兼容css代码，可以方便地应用到老项目中

3、预处理编译工具Koala

4、less基本语法

1）新建less文件

```
- 新建后缀名为.less的文件
- 新建一个后缀名为.less的文件，并保存在css文件夹下；打开Koala，将项目文件夹拖入目录列表中；右击--设置输出路径，选择正确的输出路径，输入文件名.css，单击确定；执行编译；在HTML文档中将编译好的css文件通过link标签链入
```

2）注释

```
- //单行注释，只在less源文件中保留，编译后被省略
- /* 标准css注释，会保留在编译后的文件中 */
```

3）import导入样式

```
可以导入css文件、less文件
@import必须写在样式表的头部，后面必须添加分号
```

- 语法：

```
@import  "*.css";
@import  url("*.css");
@import  url(*.css);
```

- link和@import的区别：
  - link是HTML标签；@import是css定义的，只能加载样式文件
  - link在页面载入时同时载入；@import在页面加载完成后载入
  - link没有兼容问题；@import在css2.1以下浏览器中不支持
  - link支持使用JavaScript修改样式；@import不支持JavaScript改变样式

4）变量

在全局样式中可以使用

- 定义语法：

```
@变量名: 值;

@col: red;
@wi: width;
@bo: box;
```

- 使用变量
    - 作为属性值

```
.box{ color: @col; }
```

    - 作为属性名

```
.box{ @{wi}: 200px; }
```

    - 作为选择器名称

```
.@{bo}{ @{wi}: 300px; }
```

5）混入

将一种或一系列的属性从一个规则集引入到另一个规则集

- 混入类名：
    - 定义通用属性：

```
.fo{ color: blue; } 在解析后的css文件中可以看到
.fo2(){ font-size: 30px; } 在解析后的css文件中看不到
```

    - 调用定义好的类：

```
.box2{ .fo; }
```

- 混入参数
    - 混入参数没有设置默认值，调用的时候必须传入参数

```
.fo3(@radius){ border-radius: @radius; }
.box{ .fo3(20px);}
```

    - 混入参数并设置了默认值，调用时如果不传入参数，使用默认参数设置；如果传入参数，使用传入参数显示

```
.fo4(@bol:bold){ font-weight: @bol;}
.box{ .fo4; .fo4(400);  }
```

- 使用@arguments 来引用所有传入参数

```
.bor(@bw,@bs,@bc){ border: @arguments; }
.box{ .bor(10px, solid,#000)}

.bor2((@bwi:5px, @bst: solid,@bcl:red){ border: @arguments; }
.box{
    .bor2;
    .bor2(10px);
    .bor2(10px,dotted);
    .bor2(10px,dotted, #f0f);
}
```

6）嵌套

- 模仿HTML结构——选择器嵌套
- 在嵌套代码中，可以使用&表示引用父元素

7）继承

```
extend伪类实现样式的继承
```
.fontSt{ font-size: 30px; }
.box{
    p{
        &:extend(.fontSt);
    }
}
.txt:extend(.fontSt){
    font-style: italic;
}
```
```

8）运算

```
数值、变量、颜色都可以运算
```

- 数值运算
  运算符前后用空格隔开；不需要为每个值都加单位
- 颜色运算
  将颜色色值转为rgb颜色模式，运算，再转回十六进制色值并返回
- 在calc函数中运算

```
width: calc(~"100% - 20px");
```

盒子的绝对居中

- marign负间距实现有width属性和height属性的绝对定位元素的居中
- margin:auto; 实现有width属性和height属性的绝对定位元素的居中
- transform: translate();实现绝对定位元素的居中
- 使用弹性布局实现元素的居中

移动端浏览器及内核

```
UC浏览器、QQ浏览器、百度浏览器、夸克浏览器、360浏览器...
safari、chrome...
- 内核：webkit
```

viewport

- 视口
    - PC端视口：浏览器显示内容的屏幕区域
    - 移动端视口：分类：布局视口、视觉视口、理想视口
- 分类：
  布局视口：移动端设备默认的viewport(宽度768~1024px)
  视觉视口：可以观看区域，屏幕宽度
  理想视口：为了网站的移动端能更好的浏览和阅读
- meta标签
    - 目的：布局视口和理想视口的宽度保持一致。设备有多宽，布局视口就有多宽

```
<meta name="viewport" content="width=device-width,
initial-scale=1.0, user-scalable=no, maximum-scale=1.0,
minimum-scale=1.0">
    创建一个虚拟窗口，窗口的宽度为设备宽，初始缩放为1倍，同时不允许用户手动缩放
```

    - content属性
        - width 虚拟窗口的宽度　　device-width 设备宽
        - initial-scale 初始缩放比
        - user-scalable 是否允许用户手动缩放
            - user-scalable=yes 允许用户手动缩放 默认值
            - user-scalable=no 不允许用户手动缩放
        - maximum-scale 最大缩放比
        - minimum-scale 最小缩放比

移动端web页面适配方案

1、固定布局

设置viewport，布局与PC端一致，假设整个网页的宽度为320px居中，超出部分留白。

- 优点：思路沿用PC端，上手容易
- 缺点：大屏幕手机及手机横屏的时候，左右两侧留有大片留白，并且大屏幕手机下显示内容看起来较小，操作按钮较小，用户体验较差

2、流式布局

宽度使用百分比代替固定宽度px，高度大多用px固定，因此在大屏幕手机下显示效果会变成页面元素宽度被拉长，高度和原来保持一致

- 优点：可以很好解决自适应问题
- 缺点：实际显示效果不协调；大量使用百分比布局，会出现兼容问题；设计有局限性

3、响应式布局

一个网站适配所有终端，实现不同屏幕分辨率下的终端网页的不同布局；使用媒体查询针对不同宽度的设备进行布局和样式设置，从而达到适配不同设备的目的

- 优点：一站适配所有终端、减少工作量；缩短开发周期；对搜索引擎表现更友好；在每个设备中都能得到较好的设计
- 缺点：在老版本的浏览器兼容性不好；兼容各种设备工作量大，效率较低；加载更多的样式和脚本文件；设计比较难于精确定位和控制；是一种折中性质的设计方案，在一定程度上改变网页布局结构，会出现用户混淆情况；维护困难

4、rem布局

rem是css3新增单位，相对于根元素的字号大小而言，如：`html{ font-size: 100px; }` 一个div标签的宽度为1rem,div的宽度是100px

- rem布局的实现
    - 1）设置页面的viewport
    - 2）动态计算并设置html的font-size属性值
    - 3）按照PC端布局方式正常布局，将px单位改为rem单位；如果值较小（1px），不用转换
- 优点：适用于偏APP类型的移动端页面；有利于手机端各种机型的适配；减少代码重复性；有统一的参考值
- 缺点：使用具有局限性，所有的图片都需要设置一个准确值，才能保证在不同机型的适配中正常显示；必须通过js动态控制根元素的大小

5、vw布局

使用纯css实现动态改变font-size属性值，不需要引用js文件

- vw viewport's width 是css3规范中的宽度视口单位，将视口宽度平均分成100份 720px 7.2
- vh viewport's height 高度视口单位，将视口高度平均分成100份
- 1vw = 屏幕宽度的1%
- vmax 相对于视口的宽度或高度中较大的那个值，将最大的那个值平均分成100单位的vmax
- vmin 相对于视口的宽度或高度中较小的那个值，将最小的那个值平均分成100单位的vmin
    - iphone 414px 1vw = 4.14 24.1545 个vw是100px
    - 375px 1vw = 3.75 26.6666个vw是100px
- 原理：确定基准值以常见的750像素宽度的设计稿为例，根据vw单位的原理 100vw = 750px，即1vw = 7.5px，根据设计稿中的px值，转换成对应的vw值进行布局，也可以直接写px，后期借助插件完成自动计算得到需要的vw
- 优点：页面元素随着页面宽度变化
- 缺点：兼容性没有ren好

vw + rem布局

确定基准值以常见的750px宽度的设计稿为例，根据vw单位和rem单位原理实现在750px设计稿下，如果使用vw单位换算，100vw = 750px，1px = 0.133333vw(100/750)；如果使用rem单位换算，预设1rem = 100px，则100px = 13.3333vw

```
html{
    font-size: 13.3333vw;
}
```

移动端样式重置

1、normalize.css 代替样式重置文件

没有重置所有的样式风格，提供了一套合理的默认样式值

2、reset.css样式重置

- 禁止用户选中文字 安卓无效

```
-webkit-user-select: none;
```

- 禁止长按页面显示系统菜单

```
img,a{
    -webkit-touch-callout: none;
}
```

- 修改placeholder的样式

```
inuput::-webkit-input-placeholder{
    color: #333; /*默认样式*/
}
input:focus::-webkit-input-placeholder{
    color: #f00; /*获取焦点后的样式*/
}
```

移动端布局注意：

- 页面布局文字是否能够随着屏幕大小变化而变化
- 流式布局和flex布局主要针对宽度设置布局
- 当屏幕发生改变时，元素的宽度和高度如何等比例缩放

移动特殊处理

1、超小字号处理（小于12px）

```
transform: scale(.7);
```

2、动画定义3d启动硬件加速，提升网站动画渲染性能

```
GPU 图形处理器，处理计算机中跟图形计算有关的工作，将数据更好地呈现在显示器中
CPU 中央处理器，控制计算机运行
```

```
transform: translate3d(0,0,0)
```

3、圆角BUG

```
background-clip:padding-box;
```

4、input的placeholder属性会出现文本在偏上位置显示的问题

```
在保证input输入文本垂直居中的条件下，给placeholder设置padding-top,不设置行高
```

SyntaxError 语法错误

像素

- 是web开发中常用单位，是组成数字图像的基本单元
- 分类：设备像素、设备独立像素、css像素

1、设备像素

- 物理像素：设备固有属性
- 物理像素点：屏幕显示最小的颗粒，是真实存在
- 对应设备分辨率：显示器的宽度和高度分别有多少个物理像素

2、设备独立像素

- 逻辑像素、独立像素：操作系统定义的像素单位
- 普遍规律：屏幕的像素密度越高，需要更多的物理像素来显示一个逻辑像素
- 对应逻辑分辨率：使用屏幕的宽*高来表示

物理像素和逻辑像素之间的关系

```
设备像素比 = 物理像素/逻辑像素
```

3、css像素

- css中使用的像素，当缩放页面时，元素的css像素的数量不变，但是每个像素的大小会变化
  元素width: 100px; 缩放到200%后，宽度是100个css像素，每个css像素的宽、高是原来的两倍

css像素和逻辑像素之间的关系

```
缩放比例 = css像素的边长/逻辑像素的边长
元素的宽度为100个逻辑像素，缩放到200%后，元素的宽度变为200个逻辑像素
在缩放比例为100的情况下，一个css像素 = 一个逻辑像素
```

css像素和物理之间的关系

```
设备像素比：本质是一个css像素表示多少个物理像素
```

视网膜屏Retina

```
将更多的像素点压缩在一块屏幕中，达到更高的分辨率，并调高屏幕显示的细腻程度
普通屏幕 1个css像素对应1个物理像素
Retina屏 1个css像素对应4个物理像素
```

- 1像素边框问题
  border-width: thin; 细边框
  border-width: medium; 中等粗度边框
  border-width: thick; 粗边框

flexible.js + rem

- lib-flexible 手机淘宝团队
- 用来解决HTML5页面终端适配问题
- 开源库
- 实现原理
  设备划分10等份，确定好html标签的font-size属性值
  如：750px html{ font-size: 75px; }，页面中元素的rem值 = 元素的px值/75，剩余部分由flexible.js来完成

1、使用方法

1)在head标签中，插入库文件

- 直接下载文件到本地，通过script标签对链接

```
<script src="js/flexible.min.js"></script>
```

- 通过script标签对，加载CDN文件

```
<script src="http://g.tbcdn.cn/mtb/lib-flexible/0.3.4/??
flexible_css.js,flexible.js"></script>
```

> *dpr 设备像素比*

2）将视觉稿中的px转换成rem

2、flexible的实质

—— 动态改写dpr、根元素的font-size

- 做的工作
  - 动态改写meta标签

    ```
    <meta name="flexible" content="initial-dpr=2">
    initial-dpr将dpr设置为给定的值
    ```

  - 动态改写data-dpr的值

- 给html标签添加font-size属性值，并动态改写font-size的值

3、特点

- 优点：可以很方便的解决HTML5页面的适配问题
- 缺点：由于viewport单位得到众多浏览器的兼容，lib-flexible这个过渡方案已经可以放弃使用，不管是现在的版本还是以前的版本，都存有一定的问题。

css3中过渡和动画的区别和各自的适用场景

区别：

- 1）语法：
    - 过渡：transition: 参与过渡属性名称 过渡的完成时间 速度曲线 延迟时间;
    - 动画：
      @keyframes 动画名称{} 定义关键帧
      animation:动画名称 动画的执行时间 速度曲线 延迟时间 播放次数 播放顺序; 引用关键帧
- 2）触发：
    - 过渡：需要借助伪类、JavaScript、@media触发
    - 动画：自动触发
- 3）执行次数
    - 过渡：执行一后不会执行，但是可以通过transitionEnd事件添加循环
    - 动画：可以通过animation-iteration-count属性设置循环的次数
- 4）复杂度
    - 过渡：简单动画效果
    - 动画：复杂动画效果，可以定义关键帧，控制每一帧的动画效果

适用场景

```
- 过渡：适用于元素从一种样式变换为另一种样式的动画效果
- 动画：可以在网页中取代动画图片、flash及需要灵活定位多个帧及循环的动画中
```

- img标签上title和alt属性的区别？
    - 分析：功能、属性、适用标签
- web标准的理解
    - 分析：三层分离结构、W3C、W3C对web标准提出的编码规范
- 谈谈你对HTML5和css3的理解
- div+css布局较table布局由什么优点？

回顾——响应式布局

一个网站适配所有终端，实现不同屏幕分辨率下的终端网页的不同布局；使用媒体查询针对不同宽度的设备进行布局和样式设置，从而达到适配不同设备的目的

- 优点：一站适配所有终端、减少工作量；缩短开发周期；对搜索引擎表现更友好；在每个设备中都能得到较好的设计
- 缺点：在老版本的浏览器兼容性不好；兼容各种设备工作量大，效率较低；加载更多的样式和脚本文件；设计比较难于精确定位和控制；是一种折中性质的设计方案，在一定程度上改变网页布局结构，会出现用户混淆情况；维护困难

什么是响应式设计？

- 响应式布局、响应式开发
- 解决移动互联网实现不同屏幕分辨率下网页的不同的展示效果，不需要为每个终端定制特性的版本
- 包含：弹性布局、图片、css media query等
- 页面设计与开发，需要根据用户行为、设备环境进行相应的调整
- 通过媒体查询检测不同的设备尺寸，通常在页面头部添加meta标签声明viewport

响应式设计技术要点

1、标签的设置

- meta标签设置
    - 虚拟窗口：创建虚拟窗口，自定义窗口大小和缩放功能，适应移动端设备的屏幕大小
    - 使用高版本的IE内核浏览器或chrome

```
<meta http-equiv="X-UA-Compatible"
content="ie=edge,chrome=1">
```

    - 针对手持设备优化，一些老的不识别viewport的浏览器

```
<meta name="HandHeldFriendly" content="true">
```

- IE8- 不支持媒体查询，使用media-queries.js或respond.js增加IE对媒体查询的支持

```
<script src="https://cdn.bootcdn.net/ajax/libs/livingston-
css3-mediaqueries-js/1.0.0/css3-mediaqueries.min.js">
</script>
<script
src="https://cdn.bootcdn.net/ajax/libs/respond.js/1.4.2/re
spond.min.js"></script>
```

2、使用媒体查询适配对应的样式

- 媒体查询常见的媒体尺寸设置——断点

屏幕	设备	断点
超小屏幕	手机	<768px
小屏幕	平板	>=768px~<992px
中等屏幕	桌面	>=992px~<1200px
大屏幕	桌面	>=1200px

```
常用媒体查询尺寸
    1200px
    1100px 1024px 1000px
    980px
    768px  720px
    640px
    480px
    320px
```

- 移动设备优先：超小设备 手机 768px以下
  @media screen and (min-width: 768px){ 小屏幕 }
  @media screen and (min-width: 992px){ 中等屏幕 }
  @media screen and (min-width: 1200px){ 大屏幕 }
- 大屏幕优先：大屏幕 桌面 1200px及以上
  @media screen and (max-width: 1199px){ 中等屏幕 }
  @media screen and (max-width: 991px){ 小屏幕}
  @media screen and (max-width: 767px){ 超小屏幕}

3、响应式内容设置

- 字体设置
  css3 rem
  html{ font-size: 100%;}
  @media all and (max-width: 1199px){
  html{ font-size: ;}
  }
  px
- 百分比布局
  - 宽度使用百分比，内填充、margin、border
  - 布局：浮动、定位、弹性盒子
  - 图片
    - 前景图：max-width属性控制图片大小，height: auto
    - 背景图：background-size属性

媒体查询

1、媒体查询

- 可以根据设备显示器的特性，设置不同的css样式
- css2 允许设计者根据不同的媒体类型定义不同的css样式
- css3 可以针对设备特性在不改变页面内容情况下，为特定输出设备定制显示效果
- 媒体查询可以检测的内容：viewport宽度和高度、设备的宽度和高度、设备的朝向、分辨率

2、引入方式

1）在样式表中引入——在style标签对引入、在外部的css文件中添加媒体查询并通过link标签引入

```
@media mediaType  and     (media feature){
    选择器{ 属性: 属性值; }
}
```

- mediaType设备类型：
    - all 所有多媒体设备
    - print 打印机或打印预览
    - screen 电脑屏幕、平板电脑、智能手机等
    - speech 屏幕阅读器等发声设备
- media feature 媒体特性表达式
    - width 浏览器宽度 height浏览器高度
    - max-width 最大宽度 min-width 最小宽度
    - device-width 设备宽 device-height 设备高
    - max-device-width 最大设备宽 min-device-width 最小设备宽
    - max-device-height 最大设备高
    - orientation 浏览器朝向(landscape横屏、portrait纵屏)
    - aspect-ratio:1/2 可视区宽度和高度的比率（max、min)
    - device-aspect-ratio:1/2输出设备的屏幕可视区的宽度和高度的比率

2）通过link标签引入——media属性

```
<link rel="stylesheet" media="mediaType and (media Feature)"
href="css文件路径" >
```

```
<link rel="stylesheet" href="css/media2-1.css" media="all and (max-
width: 1000px)">
```

- 如何让img标签在div中上下居中
- 如何居中一个浮动元素
-
- 文字水平、垂直方向上重叠

Grid布局——网格布局

- flex布局：弹性布局，轴线布局，只能在指定的轴线上排列，可以看作一维布局
- grid布局：网格布局，二维布局，水平方向和垂直方向同时存在

1、基本概念

- 容器container：采用网格布局的区域
- 项目item：容器中采用网格定位的子元素
- 行row：容器中的水平区域
- 列column：容器中的垂直区域
- 单元格cell：行列交叉区域，正常情况下m行n列 m*n个单元格
- 网格线grid line：划分网格的线
    - 水平网格线　行
    - 垂直网格线　列
      正常情况下，m行有m+1根水平网格线，n列有n+1根垂直网格线

2、属性

- 容器属性
    - display属性：指定容器使用网格布局
        - display: grid;
        - display: inline-grid;
    - grid-template-rows、grid-template-columns 对网格进行水平方向和垂直方向的划分
        - 属性值：px、百分比、auto、fr(对网格剩余空间比例分配，值的大小大于等于1全部分配，否则会有剩余空间)
        - repeat() 重复，可以设置两个参数，第一个参数表示重复次数，第二参数表示重复的值
        - minmax() 长度范围，两个参数，分别表示最大值和最小值
        - 网格线名称

```
    grid-template-rows: [r1] 100px [r2] 200px [r3] 100px;
    grid-template-columns: [c1] 100px [c2] 200px [c3] 100px;
```

    - grid-row-gap属性、grid-column-gap属性、grid-gap属性 设置网格间隙
        - grid-gap: grid-row-gap grid-column-gap;

```
grid-row-gap: 10px;/* 行间隙 */
grid-column-gap: 20px;/* 列间隙 */
grid-gap: 10px;/* 行间隙和列间隙都为10像素 */
grid-gap: 10px 0; /* 10px指行间隙 */
grid-gap: 0 20px;/* 20px值列间隙 */
```

- grid-auto-flow属性：划分网格后，对容器中子元素的排列位置的调整
    - grid-auto-flow: row; 默认值，先行后列的排序
    - grid-auto-flow: column; 先列后行
- justify-items属性、align-items属性、place-items属性：内容的分布
    - justify-items属性：设置网格元素的水平呈现方式
    - align-items属性：设置网格元素的垂直呈现方式
    - place-items: align-items justify-items;
    - 属性值
        - start 单元格左侧对齐或顶端对齐
        - end 单元格右侧对齐或底端对齐
        - center 单元格中间对齐
        - stretch 占满整个单元格，默认
- justify-content属性、align-content属性、place-content属性 元素分布
    - justify-content属性：水平分布方式
    - align-content属性：垂直分布方式
    - place-content: align-content justify-content;
    - 属性值：
        - start 容器的起始位置 默认值
        - end 容器的结束位置
        - center 容器的中间位置
        - space-between 项目之间留有空白间隙
        - space-around 项目之间、之前、之后都留有空白间隙，并且项目之间间隔比容器和项目之间的间隔大
        - space-evenly 项目之间、之前、之后都留有空白间隙，并且项目之间间隔和容器与项目之间的间隔相等
- grid-template-area属性 定义网格区域

```
.wrap1{
    grid-template-areas: "b1 b1 b2"
                         "b3 b3 b2"
                         "b3 b3 b2";
/*将3行3列的单元格分成三类，用3个项目实现布局*/
}
```

项目属性：grid-area属性：项目处在哪个区域

```
.wrap1 div:nth-child(1){
    grid-area: b1;
}
.wrap1 div:nth-child(2){
    grid-area: b2;
}
.wrap1 div:nth-child(3){
    grid-area: b3;
}
```

- 项目属性
    - 指定项目的位置
        - grid-column-start属性： 项目的左边在哪根网格线上
        - grid-column-end属性： 项目的右边在哪根网格线上
        - grid-row-start属性： 项目的上边在哪根网格线上
        - grid-row-end属性： 项目的下边在哪根网格线上
        - grid-column属性
          grid-column: grid-column-start/grid-column-end;
        - grid-row属性
          grid-row: grid-row-start/grid-row-end;
    - justify-self属性、align-self属性、place-self属性 设置单元格内容对齐方式
        - justify-self属性： 单元格内容水平对齐方式
        - align-self属性：单元格内容垂直对齐方式
        - place-self: align-self justify-self;
        - 属性值
            - start 单元格起始位置
            - end 单元格结束位置
            - center 单元格中间位置
            - stretch 默认值，占满整个单元格

a标签

- 点击之后会直接打开邮件客户端，在收件人栏"111@163.com"

```
<a href="mailto:111@163.com">111@163.com</a>
```

- 为10086和10010发送内容为ABC的信息

```
<a href="sms:10086,10010?body=ABC">发送信息</a>
```

- 点击后可以直接拨打400-400-400电话

```
<a href="tel:400400400">拨打电话400-400-400</a>
```

- 下载图片

```
<a href="images/logo.jpg" download="logo.jpg">下载图片</a>
```