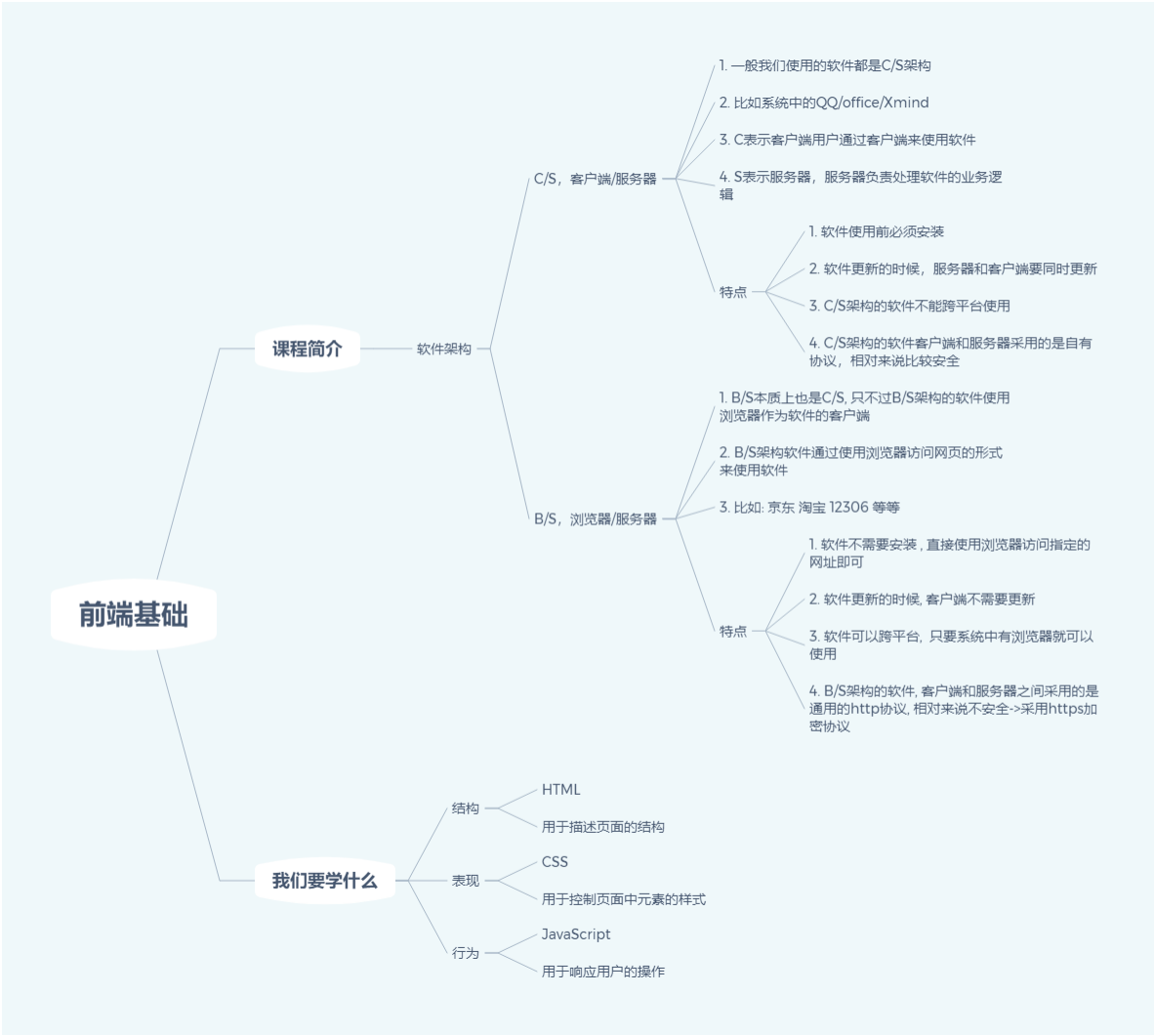

HTML

HTML 超文本标记语言

- 纯文本编辑器中的内容都是纯文本,网页都是用纯文本写的
- 纯文本中只能保存文本内容. 图片, 音频, 视频, 音频等格式化的内容都不能设置
- 超 : 超链接
- 标记 :
    - 使用方式 : <标签名> 内容 </标签名>
        - 一般成对出现
- 负责网页三要素中的**结构**

基本结构:

- 有且仅有有一个根标签 < html > < /html >, 网页的 **所有内容** 都会写在html标签中
- 根标签内有两个子标签 < head > < /head > < body > < /body >
- < head >标签中还有子标签< title >
- < head>中的内容**不会在网页中显示**, 他是写给浏览器看的,浏览器通过判断head中的信息来判断如何解析网页
- < body >表示网页的主体, 页面中所有可见的内容都要写到< body >标签里

- < title >中的内容会显示在网页的标题栏, 搜索引擎在检索页面的时候首先检索title里面的内容, 他是网页中对于搜索引擎来说最重要的内容, 会影响网页在搜索引擎中的排名

即:

```
<html>
   <head>
      <title>
      </title>
   </head>

   <body>
   </body>
</html>
```

注释

基本语法:

```
<!-- 注释内容 -->

<!--
   注释内容
-->
```

- 注释中的内容**不会在页面中显示**, 但是可以在源码中查看

标签属性

字体属性设置 < font > < /font >

属性设置 :

- 可以通过属性来设置标签如何处理标签中的内容
- 只能在**开始标签**中添加
- 实际上就是一个*名值对*的结构
- 属性名 = "属性值"
- 一个标签中可以有多个属性,但在使用的时候应该用**空格**隔开

```
<h1>
<!-- 字体 颜色 = "颜色值" -->
   这是我的<font color = "red">第一个网页</font>
<!--这只是个例子, 在实际开发中不赞成使用, 请用CSS的样式取代他-->
</h1>
```

文档声明

写网页的时候一定把声明写在最上面如果不写, 浏览器解析页面的时候可能进入怪异模式无法正常显示

怪异模式: 为了兼容一些留的页面, 浏览器中设置了两种解析模式 标准模式和怪异模式, 怪异模式在解析网页的时候回产生一些不可预期的行为, 所以我们应该避免怪异模式的出现, 写上doctype

HTML的发展:

- 1993.6 第一个版本, 起初是一个企业内部使用的

- 1995.11 HTML 2.0
- 1997.1 HTML3.2 (W3C推荐)
- 1999.12 HTML4.01 (W3C推荐, 仍在使用)
- 2000年底 XHTML1.0 (W3C推荐, 仍在使用)
- 2014.10 HTML5 (W3C推荐, 仍在使用)

在网页最上面添加一个doctype来告诉浏览器我们是什么版本

HTML5的文档声明

```
<!doctype html>
```

一个最基本的HTML页面

```
<!doctype html>
<html>
   <head>
      <meta charset = "UTF-8" />
      <title> 网页标题 </title>
   </head>
   <body>
   </body>
</html>
```

乱码问题

- 字符集:编码和解码所采用的规则
- 常见的字符集 :
  - ASCII(美国)
  - ISO-8859-1(欧洲)
  - GBK(中国)
  - GB2312(中文系统默认编码)
  - UTF-8(支持地球上所有文字, 也叫Unicode)
  - ANSI (自动以系统语言来保存编码)
- 在中文系统的浏览器中, 默认使用GB2312进行编码

在head标签中告诉浏览器用utf-8解析

meta标签: 用来设置网页的一些元数据, 比如网页的字符集, 关键字, 简介
meta是个自结束标签, 编写自结束标签的时候, 可以在开始标签中添加一个 /

```
<html>
   <head>
      <meta charset = "UTF-8" />
   </head>
</html>
```

实体

在HTMl中, 一些如 < > 之类的特殊字符是不能直接使用的, 需要使用一些特殊的符号来表示这些特殊字符, 这些特殊符号叫实体(转义字符), 浏览器解析到实体时, 会将实体自动转换为对应的字符

语法: &名字;

如:

- < <小于
- > >大于
-   空格
- © 版权符号
- 更多内容查看 : W3school离线手册

```
        <p>
            床前   明月光<hr />
        </p>
```

一些常用的标签

标题标签

- 在HTML中一共有六级标签 h1 ~ h6
- 在显示效果上, h1最大, h2最小, 但是我们**并不关心**文字的大小
- 我们关心的是标签的语义, 我们使用的标签都是语义化标签
- 六级标题中, h1最重要, 表示一个网页中的主要内容 h2 ~ h6 重要性依次降低
- 对于搜索引擎来说, h1 的 重要性仅次于 title, 搜索引擎检查完title, 会立即查看 h1
- h1 非常重要, 他会影响页面在搜索引擎中的排名
- 页面中只能写一个 h1
- 一般的页面标题中只有 h1, h2, h3 , 以后的基本不使用

```
<body>
    <h1>一级标题</h1>
    <h2>二级标题</h2>
    ......
    <h6>六级标题</h6>
</body>
```

段落标签

- 用来表示内容中的一个**自然段**
- p标签表示一个段落
- p标签中的内容会**独占一行**
- p元素中不可以包含其他任何块元素

```
<body>
        <p>
            内容
        </p>
</body>

```

换行标签

- 在html中字符之间写再多的空格或者换行, 浏览器也会当成一个空格
- 可以使用br / 自结束标签来换行

```
        <p>
            床前明月光<br />
            疑是地上霜<br />
            夜来风雨声<br />
            花落知多少<br />
        </p>
```

直线标签

- hr标签也是一个自结束标签

```
        <p>
            床前明月光<hr />
            疑是地上霜<hr />
            夜来风雨声<hr />
            花落知多少<hr />
        </p>
```

居中标签

不推荐使用 属于CSS的内容
center标签中的内容会自动在页面上居中显示

```
    <body>
        <center>床前明月光</center>
        <center>化作相思泪</center>
    </body>
```

图片标签

使用img标签来向网页引入外部图片

img标签也是一个自结束标签

属性:

- src : 设置一个外部图片的路径
    - 相对路径 : 相对于本资源所在位置的路径
    - 绝对路径 : 资源本身在资源管理器中的位置
- alt : 设置图片描述 , 只有在图片不能显示的时候才能显示
    - 主要作用 : 搜索引擎可以通过alt属性来识别不同的图片
    - 如果不写alt属性, 则搜索引擎不会对img中的图片进行收录
- width : 可以用来修改图片的宽度, 一般以px作为单位
- height : 可以用来修改图片的高度. 单位也是px
    - 如果调整了一个边的长度, 则另一个边的长度也按比例收缩
    - 除非指定了两个, 则按照指定的来
    - 一般开发中, 除了自适应页面, **不建议**设置 width 和 height

图片的格式:

- JPEG(JPG)

    - 支持的颜色比较多, 可以压缩, 但是不支持透明
    - 一般使用JPEG保存内容丰富的图片
- GIF
    - 支持的颜色少, 只支持简单的直线透明, 支持动态图
    - 图片颜色单一或者动态图的时候使用GIF
- PNG
    - PNG支持的颜色多, 且支持复杂的透明
    - 可以用来显示颜色更复杂的图片
    - 实际开发中用的比较多

图片的使用原则:

- 效果不一致, 使用效果好的
- 效果一致, 使用小的

```
    <body>
        <img src = "D:\VS2019.C++code\VS 背景图\ia_10016.jpeg" alt = "This is a wallpaper!" width="960px" height="540px"/>
    </body>

```

meta标签

搜索引擎在见多页面的时候, 会同时检索页面中的关键词和描述, 但是这两个值不会影响页面在搜索引擎中的排名

使用meta标签设置网页关键字

```
    <meta name = "keywords" content = "HTML5, JavaScript" />

```

可以用来指定网页的描述

```
    <meta name = "description" content = "some info about html"

```

可以用来做请求的重定向

```
    <meta http-equiv = "refresh" content = "5;url = "https://www.baidu.com"">
    <!--meta 这一个记住就行             引号里 "间隔时间 ; 进入的页面" -->

```

还有一系列功能可以去离线手册去找

a标签

使用超链接让我从一个页面跳转到另一个页面

a标签创建超链接

a可以包含任何元素 , 除了它本身

属性

- href : 指向超链接跳转的目标地址

    - 如果将链接地址设置为 # 则点击以后自动跳转到顶部
    - 如果在#后面加上标签的id, 则点击以后自动跳转到此id对应的标签处
- target : 用来指定开链接的位置
    - _self : 在当前窗口中打开(默认值)
    - _blank : 另起一个标签页打开
    - 可以设置为一个内联框架的name值, 将其在内联框架中打开

html中有一个属性 id, 不能以数字开头, 在同一个页面中只能有一个

```
        <a href="https://www.baidu.com">点击跳转到百度</a>

```

文本标签

em 标签 和 strong 标签

em标签 语气上的强调 显示为斜体字

strong标签 内容上的强调 显示为粗体字

```
        <em>Hello World</em>
        <strong>Hello World</strong>

```

i 标签 和 b 标签

i 标签 单纯的将字体变为斜体 没有任何语义

b 标签 单纯的将字体加粗, 没有任何语义

```
        <i>Hello World</i>
        <b>Hello World</b>

```

small 标签和 big 标签

small标签

比父元素中的内容小一些, 语义 : 表示一些细则之类的内容, 如 合同中的小子, 网站的版权声明

big 标签, 没有语义, 单纯的大一点点

```
        <small>small</small>
        <big>Hello World</big>

```

cite 标签

cite标签

网页中所有的加上书名号的东西都能用cite标签, 书名, 歌名, 话剧名, 舞蹈名...

```
        <cite><论语></cite>

```

q 标签和 quote 标签

q标签

表示短引用(行内引用), 引用的内容浏览器自动加上引号

```
        <q>学而时习之</q>

```

blockquote 标签

表示长引用, 块引用, 独占一行

```
        <blockquote>这是一个块引用</blockquote>

```

sup 标签 和 sub 标签

sup 上标(次方)(可以加上链接)

sub 下标(角标)

```
        X <sub>2</sub>
        X <sup>2</sup>

```

ins 标签和 del 标签

ins 标签表示插入的内容, 通常会加上下划线

del 标签表示一个删除的内容, 通常会加上删除线

```
        <ins>insert</ins>
        <del>delete</del>

```

pre 标签和code 标签

code 标签的语义是表示一段代码, 但是并不会加上缩进

pre标签会把其中的文本原格式保留下来(包括缩进)

我们一般结合使用pre和code标签

```
    <pre>
        <code>
            #include<iostream>
                using namespace std;
                int main(){
                    cout << "Hello World!" << endl;
                    return 0;
                }
        </code>
    </pre>

```

列表标签

ul 无序列表

- 使用ul标签创建一个无序列表
- 使用li在ul中创建项
- 使用无序的项目符号
- 通过type属性可以修改列表的项目符号[不建议用]
- 去掉项目符号, 修改样式的属性 list-style:none
- 如果镇的需要设置项目符号, 则可以采用为li设置背景图片的方式来设置
- ul 和 li 都是块元素

```
    <style>
        ul{
        /*将列表的项目符号设置为无*/
            list-style: none;
        }
    </style>
...
    <body>
        <ul>
            <li>First</li>
            <li>Second</li>
        </ul>
    </body>

```

有序列表

有序列表就是把 ul 换成 ol

- type属性可以指定序号的类型, 可选值
  - 1, a, A 等阿拉伯数字或字母
  - i I 罗马数字

ol 也是块元素

列表之间可以互相嵌套

```
        <ol>
            <li>1</li>
                <ol>
                    <li style="list-style: none;">1. 1</li>
                    <li style="list-style: none;">1. 2</li>
                </ol>
            <li>2</li>
        </ol>

```

定义列表

使用dl来创建一个定义列表

dl有两个子标签
- dt 名词
- dd 解释

```
        <dl>
            <dt>4</dt>
            <dd>2+2</dd>
            <dt>2</dt>
            <dd>1+1</dd>
        </dl>

```

dl, ul, ol 之间可以互相嵌套

表格标签

- 默认没有边框, 用CSS设置边框
- table标签表示创建一个表格(块元素)
- 用tr表示一行
- 用td创建单元格, 有几个td就有几个单元格
    - td属性cospan横向合并, cospan="n" 合并n个单元格
    - td属性rowspan纵向合并 用法同上

```
        <table>
            <tr>
                <td>1.1</td>
                <td>1.2</td>
            </tr>
            <tr>
                <td>2.1</td>
                <td>2.2</td>
            </tr>
        </table>

```

th特殊的td, 可以表示表头的内容, 用法和td一样, 不同的是有自带的居中和加粗

隔行变色nth-child(even)

长表格

将表格分为三个部分: 表头, 主题, 底部

在HTML中有三个标签:

- thead 表头, 在浏览器中永远在最上面显示
- tbody 表体, 最中间显示
- tfoot 表格底部, 在最底部显示
- 所以以上三个标签可以随意改变顺序

table的子标签, 直接写在table中, tr都需要直接写在这些标签中

如果表格中没有写tbody, 浏览器会自动在表格中添加tbody, 并将所有的tr都放在tbody里, 所以**注意tr并不是table的子元素, 而是tbody的子元素, 通过table > tr 无法选中行, 需要用 tbody > tr**

表单

填写

作用: 向服务器提交信息, 比如:百度的搜索框, 登录等

创建表单用form标签

- action 指向的是服务器的地址, 当我们提交表单的时候, 将会提交到action的地址
- 使用form创建的是一个空白的表单, 我们还应该向form中创建不同的表单项
    - 文本框 input内联元素自结束标签 type属性为text 可以有多个
        - 如果希望提交到服务器中, 还应该指定一个name属性(前端一般不决定)
        - 用户填写的信息会浮在url地址后面以查询字符串的形式发送给服务器 URL地址?查询字符串 格式: 属性名=属性值
        - 使用value设置默认值
    - 提交按钮 使用input内联自结束标签创建按钮 type属性为submit 将表单中的信息提交给服务器
        - value 选项:指定按钮上的文字
    - 密码框 input 创建 属性值为password

```
        <form action="a.html">
            Name<input type="text" /><br />
            Pass<input type="password" /><br />
            <input type="submit" value="Submit">
        </form>

```

单选

- 单选按钮 input创建, type为radio
- 单选按钮通过name分组, name相同分为一组
- 像这种需要用户选择但是不需要用户直接填写的表单想, 还必须指定一个value属性, 这样被指定的value值才会提交给服务器

```
        <form action="a.html">
            性别 <input type="radio" name="gender" value="male">man
                <input type="radio" name="gender" value="famale">woman
            <input type="submit" value="Submit">
        </form>

```

多选

使用input创建, type为checkbox, 也要使用name来分组, 使用value来提交, 但是每个选项的value值不能一样

如果希望在单选按钮或者多选框指定默认选中的选项, 则在分项中加上值为checked的checked属性

下拉列表

使用 select 创建下拉列表

- 在下拉列表中使用option标签来创建列表项
- 下拉列表的name属性要指定给select, 但是value属性要指定给option
- 当给下拉列表添加一个 mutiple="multiple" 则下拉列表变为一个多选的下拉列表
- 下拉列表中的默认选中是添加一个值为selected的selected属性
- 在select中可以用optgroup对选项进行分组

```

```

```
        <form action="a.html">
            Your number
            <select name="name">
                <optgroup label="1开头">
                    <option value="2" selected = "selected">12</option>
                    <option value="3">134</option>
                    <option value="4">1234</option>
                </optgroup>
                <optgroup label="2开头">
                    <option value="2">22</option>
                    <option value="3">234</option>
                    <option value="4">2234</option>
                </optgroup>
            </select>
            <input type="submit" value="Submit">
        </form>

```

文本域

使用textarea创建一个多行文本域

input使用type的值为reset设置一个重置按钮

使用input, type为button可以创建一个单纯的按钮, 没有任何的功能, 只能被点(结合js扩展功能)

也可以用button标签创建一个按钮, 和input类似, 但是成对出现使他更加灵活**(推荐, 更加灵活)**

在html中专门提供了一个标签, 专门用来选中表单中的提示文字的label标签

- 该标签可以指定一个for属性, 该属性的值需要指定一个表单项的id值
- for和id的值要匹配上
- 这样在点击label选择文字的时候回会选上选项

表单项分组

- 使用fieldset来对表单进行分组
- 在fieldset中可以使用legend子标签来指定分组信息

XHTML语法规范

为什么要学Xhtml的语法? 因为比较严格

1. HTML中不区分大小写, 但是一般用小写
2. 注释不能嵌套
3. 标签必须结构完整, 要么成对出现, 要么是自结束标签(即使浏览器会尽力在解析时修正你的代码, 有时候也修正不了)
4. 标签可以嵌套, 但是不能交叉嵌套
5. 标签的属性必须有值, 且值必须加上引号, 单引号双引号都行

内联框架

可以引入一个外部的页面
使用iframe创建一个内联框架

属性:

- src: 指向一个外部页面的路径, 可以使用相对路径

- width
- height等
- name : 可以为内联框架指定一个内容
- 更多内容查看离线手册

```
        <iframe src="https://www.baidu.com" height="500px"></iframe>

```

在现实开发中不推荐使用内联框架,因为内联框架中的内容不会被搜索引擎检索到

框架集

和内联框架类似, 都用于在一个页面中引入其他外部页面, 框架集可以同时引入多个页面, 在h5标准中, 推荐使用框架集, 而不是内联框架

frameset创建框架集 **注意**不能和body出现在同一个页面中, 要使用框架集就不能用body标签

属性:

- rows 指定框架集中的所有框架, 一行一行排列
- cols 指定框架集中的所有框架, 一列一列排列
- 这两个属性**必须选择一个**, 并需要在属性中指定每一部分所占的大小
- frameset中有一个子标签frame自结束标签, src指向一个页面, 用来引入其他的网页
- 引入几个页面就写几个frame

```
    <frameset cols="50%, 50%">
        <frame src="01.html" />
        <frame src="02.html">
    </frameset>

```

frameset中可以再嵌套frameset

为什么不推荐使用？

frameset和iframe一样, 里面的内容都不会被搜索引擎所检索, 使用框架集意味着页面不能有自己的内容, 只能引入其他的页面, 而我们每单独加载一个页面都会重新发送一次请求, 用户的体验比较差, 如果非得用, 建议用frameset而不是iframe

HACK

有些情况, 有一些特殊的代码我们只需要在某些特殊的浏览器中执行，而在其他的浏览器中不需要执行，这时我们就可以使用css hack来解决该问题。

Cssheck实际上是一段特殊的代码，这段代码只在某些特殊浏览器中可以识别，而其他浏览器中不能识别，通过这种方式来为一些浏览器设置特殊代码

CSS

简介

- 层叠样式表 Cascading Style Sheets
- css可用来为网页创建样式表, 通过样式表可以对网页进行装饰
- 所谓层叠, 可以将整个网页想象成是一层一层的结构, 层次高的覆盖层次低的
- css就可以分别为网页的各个层次设置样式

基础知识

像素 px

长度单位 像素px

是在网页中使用最多的一个单位 相当于屏幕中的一个小点 我们的屏幕就是由这些点构成的

不同的显示器, 一个像素的大小也不同, 显示效果越好, 越清晰, 像素就越小, 反之像素就越大

百分比%

也可以将单位设置为百分比的形式, 这样浏览器会根据其父元素的样式来计算该值

好处 : 当父元素的属性值变化的时候, 子元素也会按照比例发生改变

在我们创建一个自适应页面的时候, 经常使用百分比作为单位

em

和百分比类似, 是相对于当前元素字体大小来计算的

1 em = 1 font-size

使用em的时候, 当字体大小发生改变的时候, em也会随之改变

设置字体相关的样式的时候经常会使用em

颜色

可以直接使用颜色的单词来表示不同的颜色

可以用RGB值

- 通过red, blue, green设置, 通过这三种颜色不同的浓度表示不同的颜色
- 方式 : rgb(red值, green值, blue值)
- 数值
    - 可以为 0 -- 255 之间的数字
    - 可以用 00 -- FF 之间的16进制数字, 两位重复的颜色可以用简写, 如 aabbcc 可以表示为 #abc
    - 可以用 0% ~ 100% 之间的百分数

位置

标签中

- 可以将CSS样式写到元素的style属性当中, 叫做内联样式, 只对当前元素的内容起作用, 这样的复用性不太好, **不推荐使用**
- 内联样式属于结构与表现耦合, 在实际开发中不推荐使用
- 也是名-值对结构

```
    <p style="color: red; font-size: 50px;">WARNING</p>
    <!-- 字体颜色; 字体大小 -->
```

head中

- 也可以将CSS样式便写道head中的style标签中
- 将样式表编写到style标签中吗这样可以将样式进一步复用
- 也可以使表现和结构进一步分离
- 是我们推荐的使用方式
- **缺陷**: 但是这种方式只能在同一个页面中复用, 不能在不同页面复用

```
<!doctype html>
<html>
    <head>
        <title>CSS</title>
        <meta charset="UTF-8" />
        <style type="text/css">
    <!-- 中间的type是默认值, 写上是因为有的浏览器不兼容会出问题-->
            p{
            color: red;
            font-size: 50px;
            }
<!-- p表示对下面的所有p标签内容有效,大括号内是对这些东西的设置 -->
        </style>
    </head>
    <body>
        <p>窗前明月光, 疑似地上霜</p>
    </body>
</html>
```

外部CSS文件中

- 开发中最推荐使用
- 还可以将样式表编辑道外部的CSS文件中, 然后用link标签将外部的CSS文件引入到当前页面中
- 完全使得结构和表现分离, 可以使样式表在不同的页面中使用, 最大限度地使样式可以进行复用
- 使用link自结束标签引入, 可以利用浏览器的缓存, 加快用户访问的速度, 提高了用户的体验
- rel="stylesheet" type="text/css"是默认值, 记住就行

```
<!--myStyle.css-->
p{
    color: royalblue;
    font-size: 30px;
}

<!-- test.html -->
<!doctype html>
<html>
    <head>
        <title>CSS</title>
        <meta charset="UTF-8" />
        <link rel="stylesheet" type="text/css" href="myStyle.css"/>
    </head>
    <body>
```

```
        <p>窗前明月光，疑似地上霜</p>
    </body>
</html>
```

CSS语法

- 在style标签中间就不是HTML代码了, 所以用HTML代码的注释就失效了
- 在CSS代码或者文件中应该用 /* */ 表示注释
  CSS语法 : 选择器和声明块
  选择器 :
- 通过选择器可以选中页面中指定的元素, 并且将声明块中的样式应用到选择器对应的元素上

声明块 :

- 紧跟在选择器的后面, 用{}括起来
- 实际上是一组名值对结构, 中间用 : 链接
- 名值对叫做声明
- 一个声明块中可以写多个声明, 多个声明之间用 ; 隔开

内联和块元素

这两个元素在HTML5已经没有了

div :

- 块元素, 独占一行的元素
- 无论内容有多少都会独占一整行
- p, h1, h2 等也是块元素, 都会独占一行
- 没有任何语义(表示一块区域), 就是一个纯粹的块元素, 并且不会为其中的元素设置任何默认样式
- 主要对页面进行布局(分区)

span :

- 内联元素(行内元素)
- 有几个字就占多少地方
- 后面可以跟着其他元素
- 一行容不下的时候会自动换行
- a, img, iframe, span 都是内联元素
- 没有任何语义
- 用来选中自由的文字, 然后为文字设置样式

注意 :

- 一般情况下使用块元素包含内联元素 , 而不是用内联元素包含块元素
- a元素可以包含除了他本身的任何元素
- p不可包含其他任何块元素

常用选择器

元素选择器

作用 : 通过元素选择器可以选择页面中所有指定的元素

```
        <style>
            p{
                color: red;
            }
        </style>
```

id选择器

作用 : 通过元素的id选中唯一的一个元素

语法 : #id

注意 : id不能以数字开头

```
        <style>
            #p123{
                color: black;
            }
        </style>

......

        <p id="p123">123132132123</p>

```

类选择器

我们可以为元素设置class属性

class 属性和 id 类似, 但是可以重复

拥有相同class属性值的元素属于同一个类

语法 : .class的属性值{}

```
        <style>
            .p1{
                color: aqua;
            }
        </style>
    </head>
    <body>
        <p class="p1">123132132123</p>
        <p class="p1">123132132123</p>
        <p class="p1">123132132123</p>
    </body>
```

class 比 id 的功能强大, class 可以设置多个class属性值, 多个值之间用空格隔开

```
        <style>
            .p1{
                color: aqua;
            }
            .p2{
                font-size: 50px;
            }
        </style>
...
    <body>
        <p class="p1 p2">123132132123</p>
        <p class="p1">123132132123</p>
    </body>
```

选择器分组(并集选择器)

通过选择器分组可以同时选中多个选择器对应的元素

语法 : 选择器1, 选择器2, ..., 选择器n{}

```
        <style>
            .p1, .p2{
                color: aqua;
            }
        </style>
...
    <body>
        <p class="p1 p2">123132132123</p>
        <p class="p1">123132132123</p>
    </body>
```

交集选择器

可以同时选择满足多个条件的元素, 如:span标签的某class

语法 : 选择器1选择器2...选择器n{}

```
        <style>
            p.p2{
                color: aqua;
            }
        </style>
...
    <body>
        <p class="p1 p2">123132132123</p>
        <p class="p1">123132132123</p>
    </body>
```

通配选择器

用来选择页面中的所有元素

语法 : *{}

```
        <style>
          *{
                color: aqua;
            }
        </style>
...
    <body>
        <p class="p1 p2">123132132123</p>
        <p class="p1">123132132123</p>
    </body>

```

后代选择器

元素之间的关系

- 父元素 : 直接包含子元素的元素
- 子元素 : 直接被父元素包含的元素
- 祖先元素 : 直接或间接包含后代元素的元素, 父元素也是祖先元素
- 后代元素 : 直接或间接被祖先元素包含的元素
- 兄弟元素 : 拥有相同父元素的元素

后代选择器 :

选中制定元素的后代元素

语法 : 祖先元素 空格 后代元素{}

```
        <style>
            #d1 p{
                /*选中id为d1的父元素中的p标签的元素*/
                color: aqua;
            }

            #d1 span p{
                /*选中id为d1的父元素的span标签中的p子元素*/
                color: red;
            }
        </style>
...
    <body>
        <div id="d1">
            <p class="p1 p2">123132132123</p>
            <span><p class="p4">123132132123</p></span>
        </div>
        <div>
            <p class="p1">123132132123</p>
        </div>
    </body>

```

子元素选择器

作用 : 选中指定父元素的指定子元素

ie6及以下的浏览器不兼容子元素选择器

语法 : 父元素 > 子元素

```
        <style>
            #d1 > p{
                background-color: rgb(8, 204, 83);
            }
        </style>
...
    <body>
        <div id="d1">
            <p class="p1 p2">123132132123</p>
            <span><p class="p4">123132132123</p></span>
        </div>
```

子元素选择器的伪类

- first-child 选择第一个子标签
- last-child 选择最后一个子标签
- nth-child(n) 选择第n个子标签(括号里是变量)
    - 括号里可以传特殊值, 如even 表示偶数位置, odd表示奇数位置
- 和他们的父标签是谁没有关系

```
    <style>
        p:first-child{
            background-color: yellow;
        }
    </style>
...
    <body>
        <p>123123123132</p> <!--变色-->
        <p>123123123132</p>
        <div><p>123123123132</p></div> <!--变色-->
    </body>
```

- first-of-type 某类型的第一个元素
- last-of-type 某类型的第一个元素
- nth-of-type 某类型的第一个元素

```
    <style>
        p:first-of-type{
            background-color: yellow;
            /*p类型的第一个(作为)子标签变色*/
        }
    </style>
...
    <body>
        <p>123123123132</p> <!--变色-->
        <p>123123123132</p>
        <div><p>123123123132</p><!--变色-->
            <p>123123123132</p>
        </div>
    </body>

```

兄弟类型选择器

- 后一个兄弟元素选择器
  - 选中后一个紧挨着的元素
  - 语法 : 前一个 + 后一个
- 后边所有兄弟元素选择器
  - 语法: 前一个 ~ 后面所有

```
    <style>
        /*选择div后面所有的兄弟p元素*/
        div ~ p{
            background-color: yellow;
        }
    </style>
...
    <body>

        <div>
            <p>123123123132</p>
        </div>
            <p>123123123132</p> <!--变色-->
            <p>123123123132</p> <!--变色-->
            <div>123123123123</div>
    </body>
```

伪类选择器

伪类 : 专门用来表示元素的一种特殊的状态

比如 :

- 访问过的超链接
- 普通的超链接
- 获取焦点的文本框

给链接定义样式

- 正常连接 a:link
- 访问过的链接 a:visited
  - 由于涉及到用户的隐私问题, 所以使用visited的时候只能设置字体的颜色
- 鼠标滑过的链接 a:hover
- 正在被点击的链接 a:active
- hover和active不仅可以对a标签用, 还可以对p等其他标签用(IE6除外)

```
    <style>
        a:link{
            color: blue;
        }
        a:visited{
            color: gray;
        }
        a:hover{
            color : red;
        }
```

```
        a:active{
            color: yellow;
        }
    </style>
...
    <body>
        <a href="#">go to top</a>
    </body>
```

- 获取焦点 :focus
- 选中内容 ::selection (两个冒号!)
    - 这个伪类在火狐中要采用另一种方式去写 (-moz-selection{}), 为了都兼容, 可以写两个 selection

```
    <style>
        input:focus{
            /*输入状态的文本框显示蓝色*/
            background-color: aqua;
        }
        p::selection{
            /*选中的文本为绿色*/
            background-color: green;
        }
    </style>
...
    <body>
        <input type="text" />
        <p>this is a test text...</p>
    </body>

```

伪元素选择器

使用伪元素来表示元素中的一些位置

- 首字母 :first-letter
- 首行 : first-line

```
<style>
        p::first-line{
            /*首行字体大小*/
            font-size: 50px;
        }
        p::first-letter{
            /*首个字母颜色*/
            color: red;
        }
</style>
...

        <p>This is a <br />test text...</p>

```

- :before

    - 表示元素最前面的部分
    - 一般的before都需要结合content这个样式一起使用
    - 通过content可以向before或者after的位置添加一些内容
    - 不能被选中
    - 用的不是特别多
- :after
    - 表示元素的最后面
    - 其余和before类似

```
    <style>
        p::before{
            content: "This";
        }
    </style>
...
    <body>
        <p>This is a test text...</p>
    </body>

```

属性选择器

title属性:

- 可以为所有标签指定
- 当鼠标移入到元素上的时候, 元素中的title属性的值会作为提示文字提示

属性选择器

作用 : 可以根据元素的属性或属性值来选取指定元素

语法 :

- 标签[属性名]{} 选取含有指定属性的元素
- 标签[属性名 = ""]{} 选取含有制定元素值的元素
- 标签[属性名 ^= ""]{} 选取属性值以指定值开头的元素
- 标签[属性名 $= ""]{} 选取属性值以指定值结尾的元素
- 标签[属性名 *= ""]{} 选取属性值中包含指定值的元素

```
    <style>
        /*为所有具有title属性并且为te开头的p元素设置背景颜色为黄色*/
        p[title ^= "te"]{
            background-color: yellow;
        }
    </style>
...
    <body>
        <p title="test01">This is a test text...</p>
        <p title="taxs02">This is a test text...</p>
        <p title="test02">This is a test text...</p>
    </body>

```

否定伪类

作用 : 可以从已经选中的元素中剔除某些元素
语法 : :not(选择器)

```
    <style>
        /*为所有p元素设置一个背景颜色为黄色
        除了class 值为a123 的*/
        p:not(.a123){
            background-color: yellow;
        }
    </style>
...
    <body>
            <p class="a123">123123123132</p>
            <p>123123123132</p> <!--变色-->
            <p class="a123">123123123132</p>
    </body>

```

继承性

- 祖先的样式会传给后代
- 不是所有的样式都会传给后代
- 如 : 背景相关的, 边框相关的, 定位相关的

优先级 && 伪类顺序

优先级

当使用不同选择器选中同一个元素的时候, 并且设置了相同的样式, 这时我们的样式之间会产生冲突, 最终会采用哪个样式, 由选择器的优先级(权重)决定

优先级高的优先选中

规则 :

- 内联样式, 优先级 1000
- id选择器 优先级 100
- 类和伪类选择器 优先级 10
- 元素选择器 优先级 1
- 通配选择器 优先级 0
- 继承的样式 没有优先级

注意 :

- 当出现多种选择器的时候, 需要将所有选择器的优先级相加, 但是 : 选择器的优先级计算不会超过它的最大数量级
- 如果选择器的优先级一样, 则使用靠后的样式
- **并集选择器**的优先级**单独计算**
- 可以在样式的最后添加一个 !important 则样式将会获得最高的优先级, 优先于所有的优先级显示, 甚至超过内联样式, 但是在开发中**尽量避免使用**

伪类顺序

link, visited, hover, active 这四个伪类的优先级是一样的

所以如果两个状态同时存在, 则用下面的状态

所以伪类的顺序:

- link
- visited
- hover
- active

如果伪类设置了背景图片, 则会在第一次切换图片的时候, 发现图片会有个非常快的闪烁, 这个闪烁会产生不好的用户体验

- 原因 : 背景图片是以外部资源的形式加载进网页的, 浏览器每架在一个外部资源会单独的发送一次请求, 但是我们外部资源并不是同时加载, 浏览器会在使用资源的时候才加载, 由于link和hover的状态没被马上触发, 所以并不会预先加载, 在加载的过程中会有一段时间, 导致背景图片无法显示, 才会出现闪烁的情况.
- 解决方法 : 把三个状态弄到同一张图片上(CSS-Sprite图片整合技术), 不同的状态显示不同的位置, 用background-position
- 优点:
    - 浏览器只需要发送一次请求就可以加载多个图片, 提高了访问效率, 提高了用户体验
    - 减小了图片的总大小(颜色表合并), 提高了请求速度, 提高了用户体验

字体

- 前景颜色 color
- 背景颜色 background-color
- 大小 font-size 默认是16px
    - 设置的并不是文字本身的大小
    - 在页面中, 每个文字都是在一个'格子'里, 我们设置的font-size实际上是设置的'格子'的高度
    - 一般情况下, 文字都要比这个值要小一些
    - 也有时候会比格子大
    - 根据字体不同, 显示的效果也不一样
- 字体样式 font-family
    - 如 Arial, 微软雅黑等
    - 如果浏览器支持, 就用指定的, 不支持就用默认字体
    - 该样式可以同时指定多个字体, 不同的字体用逗号隔开, 当采用多个字体的时候, 浏览器会优先使用前面的字体

字体分类

网页中的字体分成了五大类(大的分类)

- serif(衬线字体) (有顿笔)
- sans-serif(非衬线字体) (没有顿笔)
- monospace(等宽字体)
- cursive(草书字体)
- fantasy(虚幻字体)

可以将这些最设置为大分类的字体, 然后浏览器从大分类中的字体中自动选择一个使用

不同浏览器对同一个字体的理解可能不同, 所以在不同浏览器打开的时候可能会不一样

一般会将大分类指定为 font-family 的最后一个字体

```
<span style="font-family: Arial;">你好世界HelloWorld123</span>
<span style="font-family: cursive;">你好世界HelloWorld123</span>

```

其他样式

font-style 可以用来设置文字的斜体

可选值

- normal 默认值, 文字正常显示
- italic 文字会以斜体显示
- oblique 以斜体显示
- 大多数浏览器的这两个倾斜是一样的

```
        <p style="font-style: italic;" >123123</p>

```

font-weight 可以用来设置文本的加粗效果

可选值:

- normal 默认值
- bold 加粗
- 100--900 之间的值表示粗细程度
    - 但是因为用户计算机中没有这么多级别的字体, 所以可能达不到我们的效果
    - 200可能比100粗, 也可能一样

```
<p style="font-weight: bold;">12132</p>

```

font-variant 可以设置小型大写字母

可选值:

- normal 正常
- small-caps 小型大写字母

```
<p style="font-variant: small-caps;">ABCabc</p>

```

还有一个样式叫font, 可以通过这个同时设置字体相关的所有样式

也就是以上的那些样式, 可以同时设置, 不同的值之间使用空格隔开 没有顺序要求 甚至可以写可不写(默认值)

但是文字的大小和字体必须写, 而且字体必须是最后一个样式, 大小必须是倒数第二个样式

```
    <style>
        p{
            font : 100 bold 30px;
        }
    </style>
...
    <body>
        <p>123123</p>
    </body>

```

实际上使用简写属性的性能比较好

行高

CSS中并没有提供直接设置行间距的方式, 我们只能通过设置行高来间接设置行间距

而且文字会默认居中在行中显示

行间距 = 行高 - 字体大小

通过设置 line-height 来调整行高

- 可以直接接收一个大小
- 也可以指定一个百分数:相对于字体的大小
- 可以传入一个数字表示相对于字体的倍数

```
<style type="text/css">
    p{
    line-height:20px;
}
</style>

```

tips:

- 对于单行文本来说, 将行高设置为和父元素高度一致, 则单行文本在父元素中居中
- 在 font 中也可以指定行高
    - 语法 : <span style="font: 30px/50px "微软雅黑">
    - 意即 : 字体大小为30px; 行高为 50px
- 在字体的大小后也可以指定行高, 改制是可以选的, 如果不指定就使用默认值
- 如果先设置了line-height又设置了font, 则font中的会把line-height中的样式覆盖掉

文本大小写

text-transform

可选值:

- none 默认值
- capitalize 每个单词的首字母大写(通过空格识别)
- uppercase 所有都大写
- lowercase 所有都小写

```
    <style>
        p{
            text-transform: capitalize;
        }
    </style>
...
    <body>
        <p>123123</p>
    </body>

```

文本修饰

text-decoration
用来给文本添加各种线或者闪烁文本

可选值

- none 默认值
- underline 文本下划线 (超链接默认为这个)
- hoverline 文本上划线
- line-through 穿过文本的线

```
   <style>
        p{
            text-decoration:underline;
        }
    </style>

```

字符间距

字母间距 letter-spacing 字符之间的间距 中英文都有效
单词间距 word-spacing 单词之间的间距 词与词之间空格的大小(对中文来说没啥意义)

- 这两个属性都可以直接指定一个长度或者百分数, 正数表示增加距离, 负数表示减少距离

```
    <style>
        p{
            letter-spacing: 1px;
            word-spacing: 3px;
        }
    </style>

```

对齐文本

text-aligh
用于设置文本的对齐方式

可选值

- left 左对齐 默认值
- right 右对齐
- center 居中对齐

- justify 两端对齐 调整文本之间的空格的大小达到两端对齐的目的

test-indent
用于设置首行缩进

- 设置值, 对其他行没有影响
- 可以设置像素值, 但是最好设置为**em**, 即几倍的字体大小
- 当指定正值时, 则自动向右侧缩进指定的像素
- 负值向左缩进 (可以将一些不想显示的字隐藏起来)
    - 不希望用户看见, 但希望搜索引擎看见

```
    <style>
        p{
            text-align:center;
            text-indent: 2em;
        }
    </style>
...
    <body>
        <p>i love you</p>

```

盒子模型

一个盒子我们会分成几部分, 盒子的大小由以下这些加起来决定

- 内容区(content)
- 内边距(padding)
- 边框(border)
- 外边距(margin)

边框

- width **内容区**宽度
- height **内容区**高度
- background-color **内容区**背景颜色

要为一个元素设置边框, 必须设置三个样式

- border-width 宽度
    - 可以分别指定四个边框的宽度, 空格之间用空格隔开, 分别表示 *上右下左* 的边框(顺时针)
    - 如果设置 3 个值, 则分别设置给 上 左右 下
    - 两个值的话, 就分别设置为 上下 左右
    - 如果指定一个值, 则四个边都是这个值
    - 注意 : 这个规则不仅适用于边框, 还适用于下面的边框颜色, 样式等
- border-color 颜色
- border-style 边框样式
    - 可选值 : solid 实线
    - dotted 点状
    - dashed 虚线
    - double 双线
    - ...更多的参见W3C手册

盒子的可见框的大小由内容区, 内边框, 外边距加起来决定

除了border-width, CSS还提供了四个border-xxx-width, xxx可以使top, left, right, bottom, 可以分别设置指定边的宽度

```
    <style>
        .box1{
            /*宽和高*/
            width: 100px;
            height: 100px;

            /*设置背景颜色*/
            background-color: blue;

            /*边框*/
            border-width: 20px 10px 20px 10px;
            border-color: red orange;
            border-style: solid double; /*实线*/
        }
    </style>
...
    <body>
        <div class="box1"> </div>
    </body>


```

大部分浏览器, 边框宽度和颜色都是有默认值的, 而边框的样式默认值为none

边框简写样式 border

- 通过他可以直接设置四个边框的样式, 宽度, 颜色 而且没有任何的顺序要求
- 但是一指定就是四个边, 四个边不能分别制定
- 但是可以通过 border-top, border-left, border-bottom, border-left 分别对四个边的属性进行指定, 规则同 border

内边距

指的是盒子的内容区与盒子边框之间的距离

一共有四个方向的内边距, 可以通过

- padding-top
- padding-left
- padding-right
- padding-bottom
  来设置四个方向的内边距值

内边距会影响盒子的可见框的大小, **元素的背景会延伸到内边距**

使用padding可以**同时设置**四个边框的样式, 规则和border-width一致

外边距

指的是当前盒子与其他盒子的距离

盒子有四个方向的外边距

- margin-top

- margin-right
- margin-bottom
- margin-left

这个值可以理解为是一个最小值

由于页面中的元素都是靠左靠上拜访的时候, 当我们设置上和左外边距的时候, 会导致盒子的位置发生改变, 如果是设置下和右边距的大小的时候, 会导致其他盒子的位置发生改变, 也就是"挤走"

外边距也可以指定为一个负值, 即向反方向移动

margin还可以设置为auto, aoto一般只设置给**水平**方向的margin

- 如果只指定, 左外边距或者有外边距的margin为auto, 则会将外边距设置为最大值
- 垂直方向如果设置为auto, 则默认就是 0
- 如果left和right同时设置为auto, 则会将两侧的外边距设置为相同的值
- 就可以使元素自动在父元素中居中, 所以我们经常将左右外边距设置为auto, 以使子元素在父元素中居中

外边距也可以使用简写属性margin, 具体方法和内边距一样

垂直外边距的重叠

- 在网页中, **垂直**方向**相邻**外边距会发生外边距的重叠
- 即 兄弟元素之间的相邻外边距会区最大值而不是取和
- 如果父子元素的垂直外边距相邻了, 则子元素的外边距会设置传递给父元素
    - 解决方法:
    - 给父元素设置内边距或者给子元素设置外边距让他俩不相邻
    - 使用空的table标签可以隔离父子元素的外边距, 阻止外边距重叠
- 最终方案:
    - display可以将一个元素设置为表格显示
    - 具体方法见"解决高度塌陷的最终方案"

浏览器为了 在页面中没有样式的时候也有一个比较好的显示效果, 默认设置了一些margin和padding, 而这些默认样式正常情况下我们是不需要使用的, 所以在我们编写样式之前需要将浏览器中的默认margin和padding去掉

```
<!-- 清除浏览器默认样式 -->
*{
    margin:0;
    padding:0;
}

```

内联元素的盒子

内联元素不能设置width和height

可以设置水平内边距, 可以影响页面布局
可以设置垂直内边距, 但是不会影响页面的布局

可以设置边框, 水平边框影响布局, 垂直的不影响

支持水平方向外边距, 不会重叠
支持垂直方向外边距, 会重叠

_display&visiability&overflow

通过display样式可以修改元素的类型

- inline 可以将一个元素作为内联元素显示
- block 可以将一个元素作为块元素显示
- inline-block 将以一个元素转换为行内块元素
  - 既可以设置宽高, 又不独占一行
- none 不显示元素并且元素不会在页面中继续占用位置
  - 使用该方式隐藏的元素不会在页面中显示并且不再占用页面的位置

visibility 可以用来设置元素的隐藏和显示的转台
可选值: visible 和 hidden

使用这个方式隐藏的元素虽然不可见了, 但是还会占用空间

子元素默认在父元素的内容去中, 理论上子元素的最大大小可以等于父元素的内容区的大小如果超过了父元素的大小, 超过的内容会在父元素意外的位置显示, 超过的内容叫溢出的内容
溢出的元素默认在外面显示
通过overflow可以设置如何处置溢出的内容

可选值:

- visible 一处之后在父元素意外的位置显示
- hidden 溢出的内容会被修剪掉
- scroll 显示滚动条, 可以拖动滚动条查看全部内容
  - 缺点 : 不论是否溢出都会显示滚动条
- **aoto** 根据需求自动添加滚动条, 需要水平就水平, 需要垂直就垂直

文档流

文档流处在网页的最底层表示的是一个页面中的位置, 我们所创建的元素都是在文档流中

元素在文档流中的特点

- 块元素
  - 块元素在文档流中会独占一行 块元素**自上向下**排列
  - 块元素在文档流中默认宽度是父元素的100%
  - 当元素的高度或者宽度的值为auto的时候, 此时指定内边距不会影响可见框的大小, 而是会自动修改宽以适应内边距
  - 块元素在文档流中高度默认被内容撑开
- 内联元素
  - 内乱元素在文档流中只占自身大小, 会从左到右排列
  - 如果装不下了就换行之后 从左向右排列
  - 宽和高默认都是被内容撑开

浮动与文档流

如果希望块元素在页面中水平排列, 可以使块元素脱离文档流--使用float来使元素浮动, 从而脱离文档流

float

可选值

- none 默认值, 在文档流中排列
- right 元素会脱离文档流向右浮动

- left 元素会脱离文档流向左浮动

- 当为一个元素设置浮动以后(float非none), 元素会立即脱离文档流, 脱离以后, 在其下的元素会立即向上移动, 浮动以后会尽量向页面的右上或者左上浮动
- 元素浮动以后, 会**尽量**向页面的左上或右上漂浮, 直到遇见**父元素**的边框或者**其他的浮动元素**
- 如果浮动元素上面是一个没有浮动的块元素, 则浮动元素不会超过块元素
- 如果一行中不足以容纳所有浮动元素, 则会自动换行
- 浮动的元素不会超过它的兄弟元素, 最多对齐
- 浮动的元素不会盖住文字, 文字会自动环绕在元素周围, 所以我们可以通过浮动来设置文字坏绕图片效果

- 在文档流中, 子元素, 默认占据父元素的全部
- 块元素脱离文档流以后, 高度和宽度都被内容撑开
- 内联元素脱离文档流以后会变成块元素

布局

高度塌陷

在文档流中, 父元素的元素是默认被子元素撑开的, 也就是子元素多高, 父元素就多高, 但是, 当为子元素设置浮动以后, 子元素会完全脱离文档流, 此时, 将会导致子元素无法撑开父元素的高度, 导致了父元素的高度塌陷

带来的问题 : 由于父元素的高度塌陷, 代指了父元素的下面的所有元素向上移动, 导致了页面布局混乱
在开发中一定要尽力避免高度塌陷的问题

解决问题的办法:

- 我们可以将父元素的高度写死来避免这种问题, 但是一旦写死, 父元素的高度将不能自动适应子元素, 所以**不推荐使用**

办法一:

- 原理 : 根据W3C标准, 页面中的元素都有一个隐含的属性, Block Formatting Context (BFC), 该属性可以设置打开或关闭, 当开启了之后, 元素将会有如下特性
    - 父元素的垂直外边距不会和子元素重叠(子元素的外边距将不会传递给父元素)
    - 开启BFC的元素不会被浮动元素所覆盖
    - 开启BFC的父元素可以包含浮动的子元素(子元素就可以撑开父元素了)
- 如何开启?
    - 设置元素浮动
        - 虽然可以撑开父元素, 但是会导致父元素的宽度丢失
        - 而且使用这种方式也会导致斜边的元素上移动
        - **不能彻底解决问题**
    - 设置元素绝对定位
    - 设置元素为inline-block
        - 可以解决问题, 但是也会导致宽度丢失, 不推荐使用
    - 将元素的overflow设置为一个非visible的值(设置为hidden是最**简单**的方式)
        - 但是在IE6及以下的浏览器中不支持开启BFC, 所以使用这种方式不能兼容IE6
        - 在IE6中虽然没有BFC但是又另一个隐含的属性hasLayout, 该属性的作用和BFC类似, 所以在IE6中可以通过开启hasLayout解决
        - 副作用最小的: 将元素的zoom设置为1即可 zoom:1
        - zoom表示放大, 后面的数值为将元素放大几倍
        - zoom只在IE6中支持

方案二 :

clear样式 清除其他元素对当前元素产生的影响, 清除以后, 其他元素会回到原来的位置
可选值:

- none 不清除浮动
- left 左侧
- right 右侧
- both 两侧(清除对他影响最大的元素的浮动)

可以直接在高度塌陷的父元素的最后添加一个空白的div, 由于这个div并没有浮动, 所以他是可以撑开父元素的高度的, 然后再对其进行清除浮动, 这样可以通这个空白的div来撑开父元素的高度, 基本没有副作用

但是, 只用这种方式虽然可以解决问题, 但会在页面中添加多余的结构.

方案三:

通过after伪类, 选中父类的最后面, 想元素的最后添加空白的块元素, 对其清除浮动, 这样做和添加一个div的原理一样, 可以达到相同的效果, 而且不会在页面中添加多余的div, 这是最推荐使用的方式, **几乎没有副作用**

(IE6不支持, 还IE6是要用hasLayout的)

```
    <style>
        .box1{
            border:1px solid red;
        }

        .box2{
            height: 100px;
            width: 100px;
            background-color: blue;
        }

        .clearfix{
            content: "";
            display: block;
            clear: both;
        }
    </style>
...
    <body>
        <div class="box1 clearfix">
            <div class="box2">
            </div>
        </div>
    </body>

```

最终方案:

经过修改后的clearfix是一个多功能的, 既可以解决高度塌陷, 又可以确保父元素和子元素的垂直外边距不会重叠

```
<style>
    .clearfix:before, .clearfix: after{
        content: "";
        display: table;
        clear: both;
    }
</style>

```

定位

快捷生成标签
标签名.类名$*数量

如

```
div.box$*3

结果为:

<div class="box1"></div>
<div class="box2"></div>
<div class="box3"></div>

```

- 定位指的就是将指定的元素摆放到任意位置
- 通过定位可以任意的摆放元素
- position属性来设置定位, 可选值
    - static 默认值, 元素没有开启定位
    - relative 开启了相对定位
    - absolute 开启了绝对定位
    - fixed 开启了固定定位

相对定位

特点

1. 开启了相对定位(position值relative)而不设置偏移量的时候不产生任何变化
2. 是相对于元素在文档流中原来的位置的定位进行移动
3. 相对定位的元素不会脱离文档流, 原来的位置还被它占着
4. 相对定位会使元素提高一个层级, 即"盖住别的元素"
5. 通常只需要使用两个就够了
6. 开启之后元素的性质不变, 块还是块

开启后, 可以通过left right top bottom 四个属性来设置元素的偏移量

- left 相对于定位位置的左侧的偏移量
- right 右
- top 上
- bottom 下

绝对定位

当position的值为absolute的时候开启了绝对定位

特点

- 开启之后, 元素脱离文档流(与相对定位不同)
- 开启之后, 如果不设置偏移量, 元素的位置不变
- 绝对定位是相对于离他最近的开启了定位的祖先元素进行定位的(一般情况, 开启了子元素的绝对定位的同时会开启父元素的相对定位)
    - 如果所有的祖先元素都没有开启定位, 则相对于浏览器的窗口进行定位
- 绝对定位会使元素提升一个层级
- 绝对定位会改变元素的性质, 内联元素变为块元素, 块元素的宽度和高度默认被内容撑开

固定定位

当元素的position属性设置为fixed的时候开启了固定定位

固定定位也是一种绝对定位, 它的大部分特点都和绝对定位一样

不同的是:

- 永远都会相对于浏览器窗口进行定位
- 固定定位的元素会固定在浏览器窗口的某个位置, 不会随滚动条滚动而被滚上去

IE6不支持固定定位

元素的层级

- 如果定位元素的层级是一样的, 则下面的会盖住上面的
- 通过z-index属性可以用来设置元素的层级
- 可以为z-index指定一个整数值, 该值将会作为当前元素的层级
- 对于没有开启定位的元素不能用z-index
- 父元素的层级再高也不会盖住子元素

设置透明属性opacity

需要一个0-1之间的值, 0表示完全透明, 1表示完全不透明(这个属性在IE8及以下的浏览器中不支持, 需要使用 filter:alpha(opacity=50)相当于0.5的透明度)

背景

使用background-image设置背景图片

格式 :background-image:url(相对路径);

返回上一个路径用 ../ (和cmd一样哎)

- 如果图片大于元素, 默认显示图片的左上角, 一样大就全部显示
- 如果图片小于元素大小, 则将图片**平铺**到元素的大小
    - 如何设置只显示一个 ? background-repeat
    - 默认值 repeat 背景图片会双向重复
    - no-repeat 不重复, 有多大显示多大
    - repeat-x 沿着x轴重复
    - repeat-y 沿着y轴重复
- 可以同时设置背景颜色和背景图片, 这样背景颜色将会在背景图片的下面
- 一般设置背景图片的时候会同时指定一个背景颜色

- 背景图片默认贴着元素的左上角显示
    - 如何改变背景图片的开始位置 ? background-position
    - 可以使用 yop right left botton center 中的两个值指定一个背景图片的位置(如果只指定了一个值, 则另一个默认为center)
    - 可以使用xpos ypos 直接指定两个偏移量, 水平偏移量 垂直偏移量, 正值向右/下
- 如何让背景图片不随着滚动条滚动 ? background-attachment
    - 默认scroll 随着滚动
    - fixed 不随着滚动, 这时候背景图片的定位永远相对于浏览器的窗口
        - 不随窗口滚动的图片我们一般都设置给body
    - inherit 继承父元素的background-attachment设置

背景简写 : background, 没有顺序要求 没有数量要求, 不写的就使用默认值

表格样式

- width 宽度
- margin 外边距
- border 外边框
- border-spacing: table和td边框之间默认有一个距离, 可以设置这个数值(px)来改变这个距离
    - 如果想让边框合并呢 ?
    - border-collapse 的值改为 collapse
    - 如果设置了边框合并, 则border-spacing自动失效