###1. 为什么要了解一个技术的历史?

\* 更好的理解技术本身

\* 专业素养的体现

###2. HTML的发展历史

![](./001-01.png)

\* IETF(Internet Engineering Task Force)就是"互联网工程任务组",负责定义并管理互联网技术。如:IP协议、DNS、SMTP等

\* W3C(World Wide Web Consortium),万维网联盟,W3C是全球互联网最具权威的技术标准化组织,1994年10月在麻省理工学院计算机科学实验室成立。创建者是万维网的发明者Tim Berners-Lee,W3C主要负责web方面标准的制定，像HTML、XHTML、CSS、XML的标准就是由W3C来定制的。

\* WHATWG(Web Hypertext Application Technology Working Group),网页超文本应用技术工作小组,在2004年，由Opera、Mozilla基金会和苹果这些浏览器厂商组成。主要用来推动HTML 5 标准为目的而成立的组织

\* XHTML失败的原因是他违背了伯斯塔尔法则:发送时要保守,接收时要开放

>接收时要开放是WEB发展的动力,浏览器必须能够接受不够标准的东西,做为专业人士,在发送时要严谨一些

##HTML的基本结构

```html

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Document</title>

</head>

<body>

</body>

</html>

```

\* <!DOCTYPE html>是DTD文档声明,作用是告诉浏览器, 这个网页是用哪一个版本的HTML规范来编写的. 浏览器就能够按照这个规范来正确的编译/解析/渲染我们的网页

> \* 声明必须是 HTML 文档的第一行

> \* 声明不是 HTML 标签,所以没有结束标签

> \* 浏览器并不是完全依赖于这个声明, 浏览器有一套自己的默认的处理机制,不写也能运行,但要求大家都写

> \* HTML5之前有2大种规范(HTML和XHTML), 每种规范中又有3种小规范(严格的,过渡的,带有框架的),HTML5之前的DTD文档声明只做了解,例如:

```html

HTML4.01:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

XHTML 1.0

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

```

\* html标签的作用是告诉浏览器我是一个HTML文档

> \* 所有的标签都必须写在html标签里面

\* head标签的作用是给网站添加一些配置信息,如:指定网站的标题,引用外部的css/js文件

> \* head标签内部的内容都不会显示给用户查看

\* meta标签,元数据标签,主要用来添加网站的SEO相关的信息,浏览器适配相关的内容

> \* meta标签内容都不会显示给用户查看

> \* meta标签中指定的字符集可以理解为浏览器中的的一个翻译,负责把页面上的字符正确的显示出来

>> \* 和现实中的翻译一样,有的通晓各国语言,有的就差一点,所以字符集也有所不同的

>> \* GBK(GB2312) 仅仅会翻译汉字和一些常用外文,但他体积比较下,翻译速度快

>>

>> \* UTF-8 可以翻译世界上所有的文字,但他体积比较大,翻译速度慢

>>

>> \* 如果网站仅仅包含中文, 那么推荐使用GB2312,如果除了中文以外, 还包含了一些其它国家的语言 , 那么推荐使用UTF-8,如果懒得记就直接用UTF-8

>>

>> \* 在指定字符集时,HTML文件中指定的字符集必须和保存这个文件的字符集一致,否则会出现乱码

\* title标签,指定网站的标题

> \* title标签必须写在head标签里面

\* body标签,定义显示给用户查看的内容(文字/图片/音频/视频)

> \* body标签就相当于浏览器窗口

> \* 虽然说有时候你可能将内容写到了别的地方在网页中也能看到, 但是千万不要这么干, 一定要将需要显示的内容写在body中

> \* 一对html标签中只能有一对body标签

##HTML的语法总结:

> \* 所有标签都在<\>中

> \* 大多数标签有起始标签<\>和结束标签</\>,我们称为双标签,也有只有起始标签没有结束标签的,我们称为单标签

> \* 起始标签可以包括属性,属性分为名称和值,名称和值用等号连接,值一般包含在引号中,多个属性用空格

##HTML标签的分类

> \* 兄弟关系(并列/平级)

```html

<meta charset="UTF-8">

<title>Document</title>

```

> \* 父子关系(嵌套/上下级)

```html

<head>

<meta charset="UTF-8">

<title>Document</title>

</head>

```

##HTML5的特性

###1.兼容性

> 对于过去技术的过渡和兼容,一旦浏览器不支持H5的某项功能,备选就会悄悄的运行

###2.合理性

> 新增的标签都是从实际中提炼出来的经典,如header/footer...

###3.效率

> 用户即上帝

###4.分离

> 结构行为样式的分离,老版本中HTML的样式表现部分渐渐不被支持

###5.简化

>简化DOCTYPE,简化字符集....

###6.无插件

>canvas/video/audio....

###1. 注释

\* 语法：

```

<!--被注释的内容-->

```

\* 作用：注解代码，方便程序员之间的交流和自己今后阅读

\* 注意：注释不会在浏览器中显示，注释不能嵌套

###2. 标题标签 h1-h6(Heading)

\* 作用：指定标题

\* 注意：标题标签只有h1到h6,h1的语义最强，h6的最弱，一般一个页面只有一个h1标签,标签是用来给文字添加标题语义的,而不是改变文字的样式的

###3. 段落标签 p(Paragraph)

\* 作用：指定段落

\* 注意：会独占一行

###4. 水平分割线签 hr(horizontal rules)

\* 作用：画一条水平分割线

\* 注意：会独占一行，是单标签,另外，hr标签是用来修改样式的,所以不推荐使用.今后开发中添加水平线一般都使用CSS盒子来做

###5. 换行标签 br(break)

\* 作用：内容换行

\* 注意：不另起一个段落进行换行, 而网页中99.99%需要换行时都是因为另起了一个段落,所以应该用p来做

###b(bold)

>\* 加粗文本

###u(underline)

>\* 给文本添加下划线

###i(italic)

>\* 将文本倾斜

###s(strikethourgh)

>\* 给文本添加删除线

注意,以上标签没有任何语义,已经被HTML5废弃掉,如果一定要使用, 一般情况下都是用来作为CSS的钩子来使用

HTML新增了以下标签来代替

---

###strong

>\* 定义重要性强调的文字

###ins(inseted)

>\* 定义插入的文字

###em(emphasized)

>\* 定义强调的文字

###del(deleted)

>\* 定义被删除的文字

###新旧标签对应关系

```

strong == b

ins == u

em == i

del == s

```

###sub(subscript)

>\* 定义下标文本

###sup(superscript)

>\* 定义上标文本

## 1.为什么用sublime?

1. 简单好用

2. 多点编辑

3. 强大的包管理

4. 深度可定制

5. 快速切换文件 Cmd+p

6. 好用的命令执行面板 Shift+Cmd+p

7. 社区活跃

## 2.安装

- 参考 <https://www.sublimetext.com/>

- Windows用户注意操作系统类型(64位还是32位)

## 3.常用操作

1. sublime的打开

- 直接双击图标

- 通过Spotlight呼出

- 通过命令行调出

2. 查看快捷键

- 菜单栏查看

- 在命令面板(Shift+Cmd+p)中模糊输入命令查看

- 在配置文件中查看

3. 文件相关操作

- Cmd+o 打开文件

- Cmd+n 新建文件

- Cmd+s 保存文件

- Cmd+p 模糊输入文件名打开文件,支持以下几种形式

- 文件名:行号

- 文件名@标志(函数名之类的)

- 文件名#搜索关键字

- 目录名 文件名

- Ctrl+- 跳转到打开文件前所在的文件

- Shift+Ctrl+- 文件跳转过了回跳

- Ctrl+Tab 打开的多个文件之间跳转

4. 文件搜索

- Cmd+f 搜索当前文件 Enter查找下一个 Shift+Enter查找上一个

- Alt+Cmd+f 搜索替换

- Shift+Cmd+f 文件夹中查找替换

- 鼠标悬停会在 类/函数/方法的上会出现其定义的链接

5. 多点选择编辑

- 多点选择,按住 Cmd 键,点击鼠标左键

- 列选择，按住 Alt 键，用鼠标左键拖动

- Cmd+d 选中当前单词，继续敲可以选中多个

6. 移动选择

- 方向键,一次移动一个字符

- Alt+方向键,一次移动一个单词

- Cmd+方向键,一次移动一行

- Shift+移动,移动的同时进行选择

7. 编辑

- Cmd+/ 注释掉/取消注释

- Ctrl+ ]/[ 缩进

- Ctrl+l 选择当前行

- Shift+Cmd+d 复制当前行

- Ctrl+z 撤销

## 4.用户数据目录

1. 用户的数据都放在一个固定的目录,操作系统不同,目录的路径也不同,一般情况下参考以下目录

- OS X: ~/Library/Application Support/Sublime Text 3

- Windows: %APPDATA%\Roaming\Sublime Text 3

2. 以我自己的Mac为例,所有的用户数据都存放在/Users/air/Library/Application Support/Sublime Text 3/目录中,以下简称用户数据目录

3. 用户自定义的相关文件存放在 用户数据目录/Packages/User/ 目录下,以下简称用户自定义目录

4. 自定义的内容一般分为Default和User

- Default是编辑器自带的的默认配置,一般情况下不允许更改

- User是用户自定义的文件,优先级高于Default,用户可以根据爱好配置

5. 如果想把自己的sublime恢复到初始状态,可以删除整个用户数据目录,建议删除前备份该目录

## 5.自定义快捷键

1. 查找精确的命令名

- Ctrl + \`(反引号)打开控制台

- 在控制台中输入 sublime.log\_commands(True)

- 再次执行操作,在控制台中就可以看到精确的命令名

2. 打开Key Bindings中的User配置(对应 用户自定义目录 下的Default (OSX).sublime-keymap 文件)

3. 复制一条配置,更改键和对应的命令名,如果是首次打开,可以从Default配置中拷贝一条配置然后修改

> 默认浏览器打开设置

```

方式一：

{ "keys": ["ctrl+b"], "command": "open\_in\_browser" },

方式二：

{ "keys": ["command+b"], "command": "side\_bar\_files\_open\_with",

"args": {

"paths": [],

"application": "/Applications/Google Chrome.app",

"extensions":".\*"

}

},

```

## 6.自定义配置

1. 打开Settings中的User配置(对应 用户自定义目录 下的Preferences.sublime-settings 文件)

2. 添加或者修改相关的配置,比如,Sublime默认是不显示行号的,如果想显示行号,可以添加

"highlight\_line": true,

## 7.包管理

1. 在Mac系统中,Sublime自带默认的包安装在 /Applications/Sublime Text.app/Contents/MacOS/Packages 中

2. 可以通过安装PackageControl来安装和管理包,安装方法参考 <https://packagecontrol.io/>

3. 安装完成后,可以在命令面板(Shift+Cmd+p)中输入Install Package来打开包安装面板

4. 在包安装面板中直接输入需要安装的包名就可以安装对应的包了

5. 安装完成后

- 会在 用户自定义目录 下生成一个PackageControl.sublime-settings文件,这个文件纪录了所有用户自己安装的包信息

- 会在 用户数据目录/Installed Packages/ 目录下生成这个包的文件

- 有的包还会在 用户数据目录/Packages/ 下或者 用户自定义目录 下生成相关的文件

6. 如果想删除某个包,在命令面板中输入Remove Package打开卸载面板直接输入需要删除的包名就可以了

## 8.常用包介绍

1. emmet html/css代码补全

- 参考 <https://docs.emmet.io/cheat-sheet/>

2. SyncedSideBar 在侧边栏中同步展开当前打开的文件

3. Markdown Preview

> 快捷键设置:{ "keys": ["alt+m"], "command": "markdown\_preview", "args": {"target": "browser", "parser":"markdown"} },

## 9.自定义代码段 Snippets

1. Tools->Developer->New Snippet 新建一个Snippet

2. 修改snippet 内容(content)/触发词(tabTrigger)/作用范围(scope)

3. 打开相关的文件,用快捷键 Alt+Cmd+p 可以查看该文件所在的作用范围

4. 保存文件到 用户自定义目录 下, 以 .sublime-snippet结尾

## 10.管理自己的配置

1. 将 用户自定义目录 下的配置文件文件推送到github上

2. 如果新安装Sublime,可以从github上把文件clone下来放到 用户自定义目录 下

3. 安装PackageControl,这样所有的包会自动安装上,所有的配置也会生效

## 1.为什么用sublime?

1. 简单好用

2. 多点编辑

3. 强大的包管理

4. 深度可定制

5. 快速切换文件 Cmd+p

6. 好用的命令执行面板 Shift+Cmd+p

7. 社区活跃

## 2.安装

- 参考 <https://www.sublimetext.com/>

- Windows用户注意操作系统类型(64位还是32位)

## 3.常用操作

1. sublime的打开

- 直接双击图标

- 通过Spotlight呼出

- 通过命令行调出

2. 查看快捷键

- 菜单栏查看

- 在命令面板(Shift+Cmd+p)中模糊输入命令查看

- 在配置文件中查看

3. 文件相关操作

- Cmd+o 打开文件

- Cmd+n 新建文件

- Cmd+s 保存文件

- Cmd+p 模糊输入文件名打开文件,支持以下几种形式

- 文件名:行号

- 文件名@标志(函数名之类的)

- 文件名#搜索关键字

- 目录名 文件名

- Ctrl+- 跳转到打开文件前所在的文件

- Shift+Ctrl+- 文件跳转过了回跳

- Ctrl+Tab 打开的多个文件之间跳转

4. 文件搜索

- Cmd+f 搜索当前文件 Enter查找下一个 Shift+Enter查找上一个

- Alt+Cmd+f 搜索替换

- Shift+Cmd+f 文件夹中查找替换

- 鼠标悬停会在 类/函数/方法的上会出现其定义的链接

5. 多点选择编辑

- 多点选择,按住 Cmd 键,点击鼠标左键

- 列选择，按住 Alt 键，用鼠标左键拖动

- Cmd+d 选中当前单词，继续敲可以选中多个

6. 移动选择

- 方向键,一次移动一个字符

- Alt+方向键,一次移动一个单词

- Cmd+方向键,一次移动一行

- Shift+移动,移动的同时进行选择

7. 编辑

- Cmd+/ 注释掉/取消注释

- Ctrl+ ]/[ 缩进

- Ctrl+l 选择当前行

- Shift+Cmd+d 复制当前行

- Ctrl+z 撤销

## 4.用户数据目录

1. 用户的数据都放在一个固定的目录,操作系统不同,目录的路径也不同,一般情况下参考以下目录

- OS X: ~/Library/Application Support/Sublime Text 3

- Windows: %APPDATA%\Roaming\Sublime Text 3

2. 以我自己的Mac为例,所有的用户数据都存放在/Users/air/Library/Application Support/Sublime Text 3/目录中,以下简称用户数据目录

3. 用户自定义的相关文件存放在 用户数据目录/Packages/User/ 目录下,以下简称用户自定义目录

4. 自定义的内容一般分为Default和User

- Default是编辑器自带的的默认配置,一般情况下不允许更改

- User是用户自定义的文件,优先级高于Default,用户可以根据爱好配置

5. 如果想把自己的sublime恢复到初始状态,可以删除整个用户数据目录,建议删除前备份该目录

## 5.自定义快捷键

1. 查找精确的命令名

- Ctrl + \`(反引号)打开控制台

- 在控制台中输入 sublime.log\_commands(True)

- 再次执行操作,在控制台中就可以看到精确的命令名

2. 打开Key Bindings中的User配置(对应 用户自定义目录 下的Default (OSX).sublime-keymap 文件)

3. 复制一条配置,更改键和对应的命令名,如果是首次打开,可以从Default配置中拷贝一条配置然后修改

> 默认浏览器打开设置

```

方式一：

{ "keys": ["ctrl+b"], "command": "open\_in\_browser" },

方式二：

{ "keys": ["command+b"], "command": "side\_bar\_files\_open\_with",

"args": {

"paths": [],

"application": "/Applications/Google Chrome.app",

"extensions":".\*"

}

},

```

## 6.自定义配置

1. 打开Settings中的User配置(对应 用户自定义目录 下的Preferences.sublime-settings 文件)

2. 添加或者修改相关的配置,比如,Sublime默认是不显示行号的,如果想显示行号,可以添加

"highlight\_line": true,

## 7.包管理

1. 在Mac系统中,Sublime自带默认的包安装在 /Applications/Sublime Text.app/Contents/MacOS/Packages 中

2. 可以通过安装PackageControl来安装和管理包,安装方法参考 <https://packagecontrol.io/>

3. 安装完成后,可以在命令面板(Shift+Cmd+p)中输入Install Package来打开包安装面板

4. 在包安装面板中直接输入需要安装的包名就可以安装对应的包了

5. 安装完成后

- 会在 用户自定义目录 下生成一个PackageControl.sublime-settings文件,这个文件纪录了所有用户自己安装的包信息

- 会在 用户数据目录/Installed Packages/ 目录下生成这个包的文件

- 有的包还会在 用户数据目录/Packages/ 下或者 用户自定义目录 下生成相关的文件

6. 如果想删除某个包,在命令面板中输入Remove Package打开卸载面板直接输入需要删除的包名就可以了

## 8.常用包介绍

1. emmet html/css代码补全

- 参考 <https://docs.emmet.io/cheat-sheet/>

2. SyncedSideBar 在侧边栏中同步展开当前打开的文件

3. Markdown Preview

> 快捷键设置:{ "keys": ["alt+m"], "command": "markdown\_preview", "args": {"target": "browser", "parser":"markdown"} },

## 9.自定义代码段 Snippets

1. Tools->Developer->New Snippet 新建一个Snippet

2. 修改snippet 内容(content)/触发词(tabTrigger)/作用范围(scope)

3. 打开相关的文件,用快捷键 Alt+Cmd+p 可以查看该文件所在的作用范围

4. 保存文件到 用户自定义目录 下, 以 .sublime-snippet结尾

## 10.管理自己的配置

1. 将 用户自定义目录 下的配置文件文件推送到github上

2. 如果新安装Sublime,可以从github上把文件clone下来放到 用户自定义目录 下

3. 安装PackageControl,这样所有的包会自动安装上,所有的配置也会生效

###1.表格标签 table tr(table row) td(table defines)

\* 作用：给内容添加表格语义，以表格的形式来展现数据

\* 注意：

> 1.默认情况下表格的边框为0，可以通过border属性来改变

>

> 2.表格标签和列表标签一样, 它是一个组合标签, 所以table/tr/td要组合出现

>

> 3.表格曾经是网页的主角,例如:[2004年新浪网](http://2004.sina.com.cn/)

###2.表格标签的属性

\* 宽度(width)和高度(height)

> 设置对象：table td

>

> 属性值:数字+长度单位(px)

>

> 默认值:宽度是由内容自动计算出来的

>

> 如果给td标签设置widht/height属性, 会修改当前单元格的宽度和高度, 不会影响整个表格的宽度和高度

>

\* 水平对齐(align）

> 设置对象：table tr td

>

> 属性值:left/center/right

>

\* 垂直对齐(valign）

> 设置对象：tr td

>

> 属性值:top/center/bottom

>

\* 外边距(cellspacing)

> 设置对象: table

>

> 属性值:数字+长度单位(px)

>

> 外边距是单元格和单元格之间的距离,默认有2个像素的距离

\* 内边距(cellpadding)

> 设置对象: table

>

> 属性值:数字+长度单位(px)

>

> 内边距是单元格的边框和文字之间的距离,默认有1个像素的距离

>

\* 背景颜色bgcolor(background color)

> 设置对象: table/tr/td

注意：以上属性只做了解，样式用CSS来控制

###3.细线表表格

\* 通过指定外边距为0来实现细线表格是不靠谱的, 其实它是将2条线合并为了一条线

\* 细线表格的制作方式:

> 1.给table标签设置bgcolor

>

> 2.给tr标签设置bgcolor

>

> 3.给table标签设置cellspacing = "1px"

>

```html

<table bgcolor="black" cellspacing="1px">

<tr bgcolor="white">

<td bgcolor="red">1.1</td>

<td>1.2</td>

</tr>

<tr bgcolor="white">

<td>2.1</td>

<td>2.2</td>

</tr>

</table>

```

###4.完整的表格结构

\* 表头:caption

> 给整个表格设置标题,一定要嵌套在talbe标签内部才有效

\* 标题:thead tr th

> th标签会给每一列设置标题,内容会自动加粗，居中

\* 主体:tbody tr td

> tbody用来存放页面中的主体数据, 如果不写会自动加上

\* 表尾:tfoot tr td

> tfoot用来存放表格的页脚

###5.单元格合并

\* 水平方向上的单元格合并：可以给td标签添加一个colspan属性, 来指定把某一个单元格当做多个单元格来看待

\* 垂直方向上的单元格合并：可以给td标签设置一个rowspan属性, 来指定把某一个单元格当做多个单元格来看待

\* 单元格合并只能向后或者是向下合并

###6.完成案例

\* 汽车排行榜

\* 个人信息表

###1.表单的作用

收集用户信息

> 浏览器中所有的表单标签都有特殊的外观和默认的功能

###2.常用表单标签

####1.form标签

> \* 作用:创建表单用于向服务器传输数据

>

> \* 注意点:

>> \* 所有的表单内容，都要写在form标签里面

>> \* 表单的action属性用来指定提交地址,method属性用来指定提交方法,这些只有在向服务器发送数据时会用到

####2.input标签

> \* 作用:提供输入框让用户输入

>

> \* 注意点:

>> \* name属性用来指定向服务器传输数据时的名字,服务器端的程序可以根据这个名字取到用户输入的数据

>> \* type属性有很多类型的取值, 取值的不同就决定了input标签的功能和外观不同

>>> \* type="text":明文输入框,value属性用来指定默认值,value属性的值是传给服务器的值

>>> \* type="password":暗文输入框,value属性用来指定默认值,value属性的值是传给服务器的值

>>> \* type="radio":单选框

>>>> \* 单选按钮，天生是不互斥的，如果想互斥，必须要有相同的name属性

>>>> \* 可以给input标签添加一个checked属性,值也是checked,让单选框默认选中某一个

>>>> \* 如果属性的取值和属性的名称一样, 可以只写属性名称

>>>> \* value属性的值是传给服务器的值

>>> \* type='checkbox':多选框

>>>> \* 多选框可以一次选择多个值

>>>> \* 在同一个表单中用相同的name属性可以把多选框的值合并发给服务器

>>>> \* 默认值的设置和单选框一样

>>>> \* value属性的值是传给服务器的值

>>> \* type="button":普通按钮,value属性用来指定按钮的内容,配合JS完成一些操作

>>> \* type="image":定义图片按钮,src属性用来指定图片的位置,配合JS完成一些操作

>>> \* type="reset":清空表单的数据,value属性用来指定按钮的内容

>>> \* type="submit":提交表单的数据到服务器

>>>> 数据提交到远程服务器的条件

>>>>

>>>> \* form表单有action属性

>>>> \* 表单元素有name属性

>>> \* type="hidden":隐藏数据,配合提交按钮将一些数据偷偷的提交到服务器

>>> \* type="email":自动校验邮箱的合法性

>>> \* type="url":自动校验url的合法性

>>> \* type="number":只能输入数字

>>> \* type="date":通过日历来选择时间

>>> \* type="color":通过取色板来选择颜色

####3.label标签

> \* 作用:把文字和输入框进行绑定,点击文字时让对应的输入框得到焦点

>

> \* 绑定步骤:

>> \* 将文字用label标签包裹起来

>> \* 给输入框添加一个id属性

>> \* 在label标签中把for属性的值设置成输入框中的id的值就可以进行绑定了

####4.datalist标签

> \* 作用:把待选项和输入框进行绑定,在输入框中输入文字时如果匹配到待选项的内容会有提示

>

> \* 绑定步骤:

>> \* 新建datalist列表并且添加一个id

>>

>> \* 给输入框添加一个list属性,将datalist的id的值赋值给list属性即可

```html

<input type="text" list="cars">

<datalist id='cars'>

<option>待选项内容1</option>

<option>待选项内容2</option>

</datalist>

```

####5.select

> \* 作用:定义下拉列表

>

> \* 格式:

```html

<select>

<optgroup label="分组名称">

<option value='1'>列表数据1</option>

<option value='2' selected>列表数据2</option>

</optgroup>

</select>

```

> \* 注意点:

>> \* 下拉列表的内容只能选择不能输入

>> \* 给option标签添加一个selected属性来指定列表的默认值

>> \* 可以用optgroup标签把option标签包裹起来就行分组,给optgroup添加label属性来指定分组的名称

>> \* option中的value属性的值是传给服务器的值

####6.textarea标签

> \* 作用:定义一个多行输入框

> \* 注意点:

>> \* 默认情况下输入框可以无限换行

>> \* 默认情况下输入框有自己的宽度和高度

>> \* 可以通过cols和rows来指定输入框的宽度和高度, 但是虽然指定了列数和行数, 但是还是可以无限往下输入

>> \* 默认情况下输入框是可以手动拉伸的,如果要禁止手动拉伸,需要设置css为: resize: none;

####7.fieldset和legend标签

> \* fieldset标签表单添加一个边框

> \* legend标签给边框指定一个标题

> \* fieldset标签在form标签中,把需要加边框的元素包裹在其中

> \* legend标签在fieldset标签中

```html

<form action="https://www.kuazhu.com">

<fieldset>

<legend>注册界面</legend>

<p>

用户名: <input type="text" name="username">

</p>

</fieldset>

</form>

```

###图像标签img(image)

> \* 作用:告诉浏览器显示一张图片

> \* 注意点:

>> \* img标签中的src(source)属性是指定图片的地址的

>> \* img标签不会独占一行

>> \* 可以手动指定img标签的宽度(width)属性和高度(height)属性,但有可能会变形

>> \* 想指定宽度和高度,又不想让图片变形,可以只指定宽度和高度其中的一个值,系统会自动计算出另外一个的值, 并且都是等比拉伸的

>> \* 如果img标签没有指定宽高, 那么系统会按照图片默认的宽高来显示

>> \* title属性是当鼠标悬停在图片上时, 显示的提示内容

>> \* alt(alternate)属性是用于告诉浏览器当图片找不到时显示什么内容

###视频标签video

> \* 作用:播放视频

> \* 格式1:

```html

<video src="">

</video>

```

> \* 格式2:

```html

<video>

<source src="" type=""></source>

<source src="" type=""></source>

</video>

```

> \* video标签的属性介绍:

>> \* src: 需要播放的视频地址

>> \* autoplay: 是否需要自动播放视频

>> \* controls: 是否需要显示控制条

>> \* poster: 视频没有播放之前显示的占位图片

>> \* loop: 一般用于做广告视频, 视频播放完毕之后是否需要循环播放

>> \* preload: 预加载视频, 但是需要注意preload和autoplay相冲, 如果设置了autoplay属性, 那么preload属性就会失效

>> \* muted:静音

>> \* width/height: 和img标签中的一模一样

> \* 格式2说明

>> \* 格式2是为了解决浏览器适配问题,因为没有一种视频格式是所有浏览器都支持的

>> \* video支持三种视频格式, 我们可以把这三种格式都通过source标签指定给video标签, 那么以后当浏览器播放视频时它就会从这三种中选择一种自己支持的格式来播放

>> \* source标签的type属性来指定视频格式,取值有:video/webm,video/mp4,video/ogg

>> \* source标签的src属性来指定视频某一个视频的地址

>> \* 不管是格式1还是格式2都需要浏览器支持HTML5,如果不支持,需要引入相应的框架,如:[html5media](https://html5media.info/)

> \* 视频格式和浏览器支持情况

![](./008-01.png)

>> \* Ogg = 带有 Theora 视频编码和 Vorbis 音频编码的 Ogg 文件

>> \* MPEG4 = 带有 H.264 视频编码和 AAC 音频编码的 MPEG 4 文件

>> \* WebM = 带有 VP8 视频编码和 Vorbis 音频编码的 WebM 文件

###音频标签audio

> \* 作用: 播放音频

> \* 格式1:

```html

<audio src="">

</audio>

```

> \* 格式2:

```html

<audio>

<source src="" type="">

</audio>

```

> \* 注意点:

>> \* audio标签的使用和video标签的使用基本一样,video中能够使用的属性在audio标签中大部分都能够使用, 并且功能都一样,只不过有3个属性不能用, height/width/poster

>> \* 格式2中的source标签的type属性来指定视频格式,取值有:audio/ogg,audio/mpeg

> \* 音频格式的浏览器支持情况

![](./008-02.png)

###路径问题

> \* 相对路径:从执行文件(.html)所在的文件夹开始查找

>> \* 同级：资源文件和执行文件在同一个文件夹下

>>> 用法：直接写资源文件的名字 或者 ./资源文件的名字,./表示当前文件夹

>> \* 下级：资源文件存储在执行文件所在文件夹的子文件夹中

>>> 用法：从执行文件所在文件夹开始一层一层的写

>> \* 上级：资源文件存储在执行文件所在文件夹的父文件夹中

>>> 用法：../代表找到当前文件夹的上一级文件夹

> \* 绝对路径:windows从指定的盘符开始查找，linux从根目录开始查找

> \* 注意点:

>> \* 路径中不要出现中文, 否则可能出现未知问题

>> \* 如果是通过"相对路径"来指定资源文件, 不能跨盘符

>> \* 尽量使用相对路径,这样易于移植项目

> \* 路径练习,在test.html中显示所有的照片

```

└── 007-pathTest

├── 5.jpg

├── a

│   └── a1

│   └── 1.jpeg

└── b

├── 2.jpeg

└── b1

├── 3.jpg

├── b11

│   └── 4.jpg

└── test.html

```

###div(division)

>\* 无实际语义,主要用来布局的

###span

>\* 无实际语义,主要用来布局的

---

以下标签是H5中新增的标签

###header

>\* 代表页面头部内容

###nav(navigator)

>\* 页面的导航部分

###section

>\* 代表一个区域,里面可以有其他的标签,如h标签...

###article

>\* 页面当中的核心内容文章

###aside

>\* 页面文章的辅助信息,有时也代表页面的侧边栏

###footer

>\* 代表页面底部内容

\*\*\*\*

###meter

>\* 特定范围的数值

>\* 注意点

>> \* max属性,定义最大值。默认值是 1。

>> \* min属性,定义最小值。默认值是 0。

>> \* value属性,当前取值

###time

>\* 添加时间语义

>\* 注意点

>> \* datetime属性,规定日期 / 时间。否则，由元素的内容给定日期 / 时间。

>> \* pubdate属性,指定时间为发布日期

###progress

>\* 表示进度条

>\* 注意点

>> \* max属性,总共完成的值

>> \* value属性,当前值

注意:以上标签主要用来网页的布局,需要配合css的使用

###什么是字符实体?

在HTML中有的字符是被HTML保留的, 有的HTML字符在HTML中是有特殊含义的, 是不能在浏览器中直接显示出来的, 那么这些东西要想显示出来就必须通过字符实体

> \* 常见字符实体

```

&nbsp;//non-breaking space,不间断空格,有多少个&nbsp;就有多少个空格

&lt;//less than,小于符号 <

&gt;//greater than,大于符号 >

&copy; 版权符号

```

> \* 在HTML中对空格/回车/tab不敏感, 会把多个空格/回车/tab当做一个空格来处理,如果想显示多个空格,就可以写多个&nbsp;

> \* [字符实体参考](<http://www.w3school.com.cn/html/html_entities.asp>)

## 元信息标签meta(metadata)

> \* 作用: 主要用来添加网站的SEO相关的信息,浏览器适配相关的内容和添加服务器发送到浏览器的http头部内容

> \* 注意点

>> \* 标签永远位于 head 元素内部。

>> \* 数据总是以 名称/值 的形式成对出现的。

> \* 属性说明

>> \* name属性,提供 名称/值中的名称,HTML没有指定任何预先定义的名称。通常情况下，开发者可以自由使用有意义的名称,而大多数情况下这个名称有浏览器的开发者指定,常见的名称有:Keywords,Descriiption,Author,Render等

>> \* http-equiv属性,提供 名称/值中的名称。用来告诉服务器在发送实际的文档之前先发送给浏览器包含名称/值的文档头部。通常情况下,服务器会发送很多这种名称/值,但是只有浏览器可以接受并能以适当的方式使用它们时，这些字段才有意义。常见的有Content-Type

>> \* content属性,提供 名称/值中的值,在有http-equiv或name属性的时候，一定要有content属性。该值可以是任何有效的字符串。

## 常见元信息设置举例

###keywords

>\* 作用: 告诉搜索引擎当前网页的关键词, 在SEO中非常重要, 能够提高搜索命中率,让别人更容易找到你,搜索引擎会用这些关键字对文档进行分类。

>\* 格式:

```html

<meta name="keywords" content="技术学习" />

```

###descriiption

> \* 作用: 告诉搜索引擎当前网页的主要内容,在SEO中非常重要,当别人在搜索引擎中搜索到你的网站时会自动作为对你网站的描述信息展示给用户, 能够提高搜索命中率, 让别人更容易找到你

>\* 格式:

```html

<meta name="description" content="这是一个技术学习网站" />

```

###author

> \* 作用: 告诉搜索引擎当前网页的作者

> \* 格式:

```html

<meta name="author" content="Tom chen" />

```

###renderer

> \* 作用: 告诉浏览器，用什么内核进行解析，当然前提是浏览器有相应的内核才可以,一般需要参考浏览器的开发文档,例如[360浏览器内核说明](http://se.360.cn/v6/help/meta.html)

> \* 格式:

```html

<meta name="renderer" content="webkit">

```

###Content-Type

>\* 作用:告诉浏览器准备接受一个什么类型的文档,常见的有text/html表示html文本文档

>\* 格式:

```html

<meta http-equiv="Content-Type" content="text/html">

```

###Refresh

>\* 作用:用http头部内容控制页面重定向

>\* 格式:

```html

<meta http-equiv="Refresh" content="5;url=https://www.kuazhu.com" />

```

CSS

### 1. 行内样式

> \* 用法: 把样式写在标签的style属性里面

> \* 缺点: 重复代码多,维护困难

### 2. 内部样式

> \* 把样式写在style标签里面

> \* 缺点: 只适用于当前页面

> \* 注意:

>> \* style标签必须写在head标签里面

>> \* style标签中的type属性可以不用写, 默认就是type="text/css"

>> \* style标签中的样式设置有固定格式：

![](./002-01.gif)

> \* CSS的语法

>> \* CSS有选择器和声明(一条或者多条)两部分组成

>> \* 花括号前面的是选择器,花括号里面的是声明

>> \* 每条声明由一个属性和一个值组成,属性和值用冒号分开

>> \* 每条声明用分号分开

>> \* 属性不同,值的设置方式也会有所不同

### 3. 外部样式

> \* 把css写在一个外部文件里面，这个外部文件用.css作为扩展名，在需要使用的页面通过link标签引用即可。

> \* 注意:

>> \* link标签要写在head标签内部

>> \* css文件中不用写style标签

>> \* 这种方式也是项目中使用最多的一种方式

### 4. 导入样式

> \* 通过@import将外部样式文件导入进来

```

@import "style.css";

```

> \* 导入样式是css2.1增加的内容，存在兼容性问题，实际项目中并不常用。

###1.文字的样式

> \* font-style: italic;

> \* 取值：italic(倾斜的)/normal(正常的),默认是normal

###2.文字的加粗

> \* font-weight: bold;

> \* 取值：bold(加粗)／bolder(比加粗还要粗)/lighter(细线)/100-900之间整百的数字,默认是lighter

###3.文字的大小

> \* font-size: 30px;

> \* 取值：值＋单位(px)

###4.设置字体

> \* font-family: STSong;

> \* 取值：字体的名字,如:STSong,Arial,STHeiti

> \* 注意:

>> \* 字体可以设置备选,写法font-family: 字体1,字体2,字体3.....

>> \* 设置备选字体时,如果前面的字体找不到,就用后面的字体,如果都找不到就用默认字体

>> \* 设置的字体必须是用户电脑里面已经安装的字体,我们也可以使用自定义字体,自定义字体的方法单独介绍

###5.文字属性的缩写

> \* font:样式 加粗 大小 字体;

> \* 注意点：

>> \* 样式和加粗可以互换位置，并且可以省略其中一个或者都省略

>> \* 大小和字体必须写在一起，并且只能放在最后

### 颜色的取值

#### 1.用英文单词

> \* 英文单词表示颜色主要用来调试,实际开发中用的不多

> \* 17 种标准色是 aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, yellow。

#### 2.用rgb表示

> \* 什么是rgb?

>> rgb就是r(red)红 g(green)绿 b(blue)蓝的缩写，也就是咱们通常说的三原色。可以通过设置三原色的值来确定颜色。

> \* 注意点：

>> \* R,G,B的取值范围0-255,0代表不发光，255代表该颜色最亮

>> \* 三原色的颜色设置一样的情况下可以调出灰色,值越大越灰

#### 3.用rgba表示

> rgb和上述的rgb是一样的，a(alpha)就是透明度，取值从0到1，0表示全透明，1表示不透明

#### 4.十六进制表示方法

> \* 什么是十六进制？逢16位向前进一位,取值从0-9 a-f

> \* 十六进制转十进制

>> 十进制 = 十六进制的第一位\*16 + 十六进制的第二位

>>

>> 如: FF = F\\*16 + F = 15\\*16 + 15 = 240 + 15 = 255

> \* 十进制转十六进制

>> 除16取余，直到商为0为,余数倒着

> \* 格式:#RRGGBB

>> \* 两位表示一个十六进制数，RR就是红色，GG就是绿色，BB就是蓝色

>> \* 当十六进制的每一个颜色的两位是相同的情况下可以简写,例如：

>>

>> \#FF0000 ＝ \#F00;

>>

>> 注意，简写的条件是每一个颜色的两位表示的值要一样

### 1.装饰属性

> \* text-decoration: underline|overline|line-through|none;

> \* 常见的用途就是用于去掉超链接的下划线

### 2.水平对齐方式

> \* text-align: left(默认)|center|right;

### 3.首行缩进

> \* text-indent: 数字＋单位(px|em);

> \* 这个单位可以是px(像素),也可以是em(一个文字的宽度),用的最多的还是em

### 4.文本阴影

> \* text-shadow: 水平偏移距离 垂直偏移距离 模糊距离 阴影颜色;

> \* 取值：水平偏移距离 垂直偏移距离 模糊距离的取值是值加单位，例如10px,

阴影颜色就是的颜色取值

### 5.空白符处理

> \* white-space:normal|nowrap

> \* 取值说明:

>> \* normal,默认值,文字超出容器宽度会自动换行,容器会变高

>> \* nowrap,文字超出容器宽度不会换行，文本会在在同一行上溢出。

>> \* 注意浏览器会丢掉多余的空白符。换行字符（回车）会转换为空格，一行中多个空格的序列也会转换为一个空格。

### 6.文本溢出包含元素时的处理方式

> \* text-overflow:clip(剪贴)|ellipsis(显示省略号...);

> \* 注意:该属性一般需要配合其他属性一起使用,例如

```css

p{

width: 100px;

white-space:nowrap;

overflow: hidden;

text-overflow: clip;

}

```

### 7.长单词换行到下一行

> \* word-wrap: normal|break-word(单词换行);

## 什么是选择器？

选择器其实就是一套规则，规定了选择页面上元素的规则。就上{}前面的部分。

### 1.标签选择器

通过标签名称找到当前页面上所有的元素，然后给他们设置样式

\* 格式：

```

标签名称{

属性: 值;

属性: 值;

...

}

```

\* 注意点：

> \* 不管标签藏的有多深，都能找到

> \* 标签选择器不能单独选中某一个标签

> \* 所有的html标签都可以作为标签选器 p/h/table/tr/td/div/span/ul/ol/li.....

### 2.类选择器

通过给标签添加class属性，然后通过这个属性找到该元素，为它设置样式

\* 格式：

```

.类的名称{

属性: 值;

属性: 值;

...

}

```

\* 注意点：

> \* 必须以英文.开头，中间不能添加空格

> \* 类名是可以重复的，也就是说同一个类名可以添加到不同的标签上

> \* 类名的命名规范

>> \* 只能包括数字(0-9)，下划线(\_)，英文字符(a-z)

>> \* 只能以下划线和英文字符开头，不能以数字开头,一般都是以英文字符开头

>> \* 类名不能是HTML标签的名字

> \* 类(class)是所有标签都有的属性，这个属性是专门用来添加样式的

> \* 一个标签可以添加多个类,每个类中间用空格分开,格式：class="class1 class2 ...."

### 3.ID选择器

根据页面元素的ID找到该元素然后设置样式

\* 格式：

```

#ID的名称{

属性: 值;

属性: 值;

...

}

```

\* 注意点：

> \* 必须以#号开头，中间不能添加空格

> \* ID是唯一的，也就是说当前的页面上所有的ID不能重复

> \* ID的命名规范

>> \* 只能包括数字(0-9)，下划线(\_)，英文字符(a-z)

>> \* 只能以下划线和英文字符开头，不能以数字开头,一般都是以英文字符开头

>> \* ID不能是HTML标签的名字

> \* ID是所有标签都有的属性

> \* ID不是用来作为选择器添加样式的，它的主要作用是给JS使用

\* ID选择器和类选择器的区别

> \* ID是唯一的，一次只能选中一个标签，类不唯一，一次可以选中一组标签

> \* ID是以#号开头的,类是以.号开头的

> \* ID是给JS使用的,类是给CSS使用的，除非特殊情况,不要使用id去设置样式

> \* 一般情况下可以把共用的样式抽取到一个类选择器中, 然后在需要使用这个共用样式的标签上添加这个类

### 4.后代选择器

找到指定选择器的所有后代，然后给他们设置样式

\* 格式：

```

选择器1 选择器2{

属性: 值;

属性: 值;

...

}

```

> 意思就是先找到选择器1选中的元素，然后再找选择器1选中元素里面所有的选择器2选中的元素

\* 注意点

>> \* 选择器和选择器之间至少有一个空格，一般就打一个空格

>> \* 除了标签之外，还可以写其他的选择器，比如类，ID

>> \* 后代选择器可以一直写下去

>> \* 后代选择器不仅仅可以找到它的直接后代(儿子)，同时还可以找到它的孙子，重孙子。。。。。

### 5.子选择器

找到指定选择器的儿子元素，然后给他们设置样式

\* 格式：

```

选择器1>选择器2{

属性: 值;

属性: 值;

...

}

```

> 意思就是先找到选择器1选中的元素，然后再找选择器1选中的元素里面所有的儿子中被选中器2选中的元素

\* 注意点

> \* 选择器和选择器之间用>分隔

> \* 除了标签之外，还可以写其他的选择器，比如类，ID

> \* 子选择器可以一直写下去

> \* 子选择器只能找到他的儿子

\* 后代选择器和子选择器的区别

> \* 后代选择器是用空格来分隔,子选择器是用>号来分隔

> \* 后代选择器选中特定选择器的所有的后代，包括儿子，孙子...,子选择器只能选中特定选择器的儿子

\* 后代选择器和子选择器的共同点

> \* 都可以使用标签，类，id选择器

> \* 都可以根据自己的分隔符一直延续下去

\* 如何选择后代选择器和子选择器

根据应用场景，如果想选择特定选择器下面所有的后代的话就用后代选择器,如果只是想选中特定选择器下面的儿子元素的话就用子选择器

### 6.交集选择器

选中所有满足条件的集合，然后设置样式

\* 格式：

```

选择器1选择器2...{

属性1：值1；

属性2：值2；

...

}

```

\* 注意点：

> \* 每个选择器中间不能添加任何字符

> \* 选择器可以是标签，ID，类，但是标签必须放在第一个

### 7.并集选择器

找到每个选择器选中的元素，然后相加，再设置样式

\* 格式：

```

选择器1,选择器2...{

属性1：值1；

属性2：值2；

...

}

```

\* 注意点：

> \* 各个选择器之间用,号分隔

### 8.兄弟选择器

选择兄弟关系中的元素

\* 相邻兄弟选择器

```

选择器1+选择器2{

属性1：值1；

属性2：值2；

...

}

```

>\* 相邻兄弟选择器注意点

>> \* 只能选择紧邻的那个元素

>> \* 如果中间有隔开元素，就不会被选中

>> \* 各个选择器用＋号隔开

>> \* 选择器可以是标签，ID，类

\* 通用兄弟选择器

```

选择器1~选择器2{

属性1：值1；

属性2：值2；

...

}

```

>\* 通用兄弟选择器注意点

>> \* 可以选中所有的满足条件元素

>> \* 不管中间有没有隔开元素，都会被选中

>> \* 各个选择器用～号隔开

>> \* 选择器可以是标签，ID，类

### 9.序选择器

就是根据元素所在的顺序位置来选择，然后给他们添加样式

\* 同级别不区分类型

> \* 选择器:first-child 找到同级别的第一个元素，不区分类型

> \* 选择器:last-child 找到同级别的最后一个元素，不区分类型

> \* 选择器:nth-child(n) 找到同级别的第n个元素，不区分类型

>> 括号里的n除了可以写具体的数字之外，还可以写：

>>> odd 奇数

>>>

>>> even 偶数

>>>

>>> xn+y x,y是用户可以自定义，n的取值是从0到最大元素个数递增，每次增一,例如: p:nth-child(3n+0)可以选中3的倍数

> \* 选择器:nth-last-child(2) 找到同级别的倒数第n个元素，不区分类型

> \* 选择器:only-child 找到父元素的唯一子元素，不区分类型

\* 同级别同类型

> \* 选择器:first-of-type 找到同级别的第一个元素，区分类型

> \* 选择器:last-of-type 找到同级别的最后一个元素，区分类型

> \* 选择器:nth-of-type(n) 找到同级别的第n个元素，区分类型

>> 括号里的n除了可以写具体的数字之外，还可以写：

>>> odd 奇数

>>>

>>> even 偶数

>>>

>>> xn+y x,y是用户可以自定义，n的取值是从0到最大元素个数递增，每次增一,例如: p:nth-child(3n+0)可以选中3的倍数

> \* 选择器:nth-last-of-type(2) 找到同级别的倒数第n个元素，区分类型

> \* 选择器:only-of-type 找到父元素的唯一子元素，区分类型

\* 注意点

> \* 选择器可以是标签也可以是类

> \* 选择器可以是id,但是写id没有意义,因为id本身就是唯一的

> \* 选择器和冒号之间不能有空格,可以理解为这是一个交集选择器的写法

> \* 不区分类型是指所有的兄弟元素都算数,区分类型是指只有被选择器选中的兄弟元素才算数

### 10.属性选择器

根据标签设置的属性来找到需要的标签，给他们设置样式

\* 直接通过属性名称

```

[属性名称]{

属性1：值1；

属性2：值2；

...

}

```

> 找到所有具有属性名称的元素

\* 属性＝值的方式

```

[属性名称=属性值]{

属性1：值1；

属性2：值2；

...

}

```

> 找到所有具有属性名称并且值为属性值的元素,如果属性的值有.的话，需要添加引号

\* 属性的值以什么开始

```

[属性名称|=属性值]{

属性1：值1；

属性2：值2；

...

}

```

> 只能选中用“-”分隔的和独立的,是CSS2中的选择器

```

[属性名称^=属性值]{

属性1：值1；

属性2：值2；

...

}

```

> 所有以属性值开始的都可以选中,是CSS3中的选择器

\* 属性的值以什么结束

```

[属性名称$=属性值]{

属性1：值1；

属性2：值2；

...

}

```

> 所有以属性值结尾的都可以选中,是CSS3中的选择器

\* 属性的值包括什么

```

[属性名称~=属性值]{

属性1：值1；

属性2：值2；

...

}

```

> 只能选中包括用空格分隔的和独立的,是CSS2中的选择器

```

[属性名称\*=属性值]{

属性1：值1；

属性2：值2；

...

}

```

> 所有包括该属性值的都可以选中,是CSS3中的选择器

###11.通用选择器

可以选择页面上所有的元素

\* 格式：

```

\*{

属性1：值1；

属性2：值2；

...

}

```

\* 注意点：

> \* 通用选择器只做为了解，真实项目中用的不多，因为它要选择页面上所有的元素，效率比较低

### 12.伪类选择器

针对标签的状态进行选择，然后添加样式

\* a链接的4种状态

> \* 没有任何操作，默认状态

> \* 鼠标悬停在a链接上

> \* 鼠标点击时

> \* 被访问过

\* 设置方式

> \* a:link //默认状态

> \* a:visited //被访问过

> \* a:hover //鼠标悬停时

> \* a:active //鼠标点击时

\* 注意点

> \* 四种状态可以同时都写，也可以单独写

> \* 如果全写的情况下，必须遵循一定的顺序，特别是active要写到hover后面,否则就会被覆盖,因为点击的同时鼠标肯定是悬停在内容上的

> \* 一般情况下遵循爱恨原则love hate＝ lvha

> \* 默认的样式可以简写，a = a:link

> \* hover对其他元素也起作用

\* 伪类选择器补充

> \* :focus //获取焦点时添加的样式

> \* :after //指定元素的内容后面添加内容

> \* :before //指定元素的内容前面添加内容

### 1. 层叠性

当多个选择器选中同一个元素，并且设置了相同的属性时，CSS解决这种冲突的能力就叫做CSS的层叠性

### 2. 继承性

子元素具备父级元素的样式

\* 注意点

> \* 不是所有的属性都可以被继承，常见的可以被继承的属性有 color/font-/text-/line-

> \* a标签不能继承来自父元素的颜色和下划线属性

> \* h标签不能继承来自父元素的的字体大小和加粗属性,但是字体的大小会随着父级元素大小的改变而改变

> \* 一般情况下可以通过给body元素设置一些通用的样式来保持页面风格的统一

### 3. 优先级

当多个选择器同时选中某个元素时，并且设置了相同的样式，CSS会根据优先级来判断那个样式有效

\* 通过继承间接选中

> 优先级根据就近原则，就是说谁离被选中的元素近听谁的

\* 直接选中

> 当多个选择器同时选中同一个元素，并且设置相同样式时，选择器越精确优先级越高,具体根据以下优先级来判断。

>

> ID选择器>类选择器>标签选择器>通用选择器>继承间接选中>浏览器默认的样式

\* 改变优先级 !important

> \* !important必须写在需要改变优先级属性的分号;前面

> \* !不能省略

> \* !important只能提高某个属性的优先级，不能提高该选择器中其他属性的优先级

> \* 由于!important会改变选择器的优先规则,所以在实际项目中谨慎使用

\* 选择器混合使用时直接选中的优先级判断

> \* 看谁的ID选择器多，就听谁的

> \* 如果ID选择器相同，看类选器，谁的类选择器多听谁的

> \* 如果ID选择器和类选择器一样多，看标签选择器，谁的标签选择器多听谁的

> \* 如果ID选择器，类选择器和标签选择器一样多的情况下，就不继续往下找了，谁的样式写在后面听谁(就近原则)

### 盒模型介绍

所有的HMTL标签都是一个盒子，他和咱们现实中的盒子是一样的,盒模型其实是现实生活中盒子的一个比喻

<img src="008-box.png" width="400px">

\* 以戒指盒子为列,通过观察,我们发现戒指盒子有以下特性

> \* 戒指盒子和戒指盒子之间有一定的距离

> \* 戒指盒子自身有一个边框

> \* 戒指和边框之间有一定的距离

> \* 戒指自身有自己高度和宽度

#### 边框

围绕在元素四周的线条,边框有三要素,即宽度,样式,颜色

\* 边框设置

> \* 复合样式1(设置所有)

>> \* border:宽度 样式 颜色

>>> \* 颜色可以省略，默认的是黑色

>>> \* 常见样式有dotted(点状),dashed(虚线),solid(实线)

>>> \* 宽度可以省略，默认是1px

> \* 复合样式2(按属性)

>> \* border-width:上 右 下 左;

>> \* border-style:上 右 下 左;

>> \* border-color:上 右 下 左;

> \* 按照顺时针方向设置。如果省略了一个边，就取对边的值，如果只设置了一个边，那么所有的边都一样

> \* 复合样式3(按方向)

>> \* border-top: 宽度 样式 颜色;

>> \* border-right: 宽度 样式 颜色;

>> \* border-bottom: 宽度 样式 颜色;

>> \* border-left: 宽度 样式 颜色;

> \* 单一样式(一个方向上设置一个属性)

>> \* border-top-width: 宽度;

>> \* border-top-style: 样式;

>> \* border-top-color: 颜色;

>> \* border-right-width: 宽度;

>> \* border-right-style: 样式;

>> \* border-right-color: 颜色;

>> \* border-bottom-width: 宽度;

>> \* border-bottom-style: 样式;

>> \* border-bottom-color: 颜色;

>> \* border-left-width: 宽度;

>> \* border-left-style: 样式;

>> \* border-left-color: 颜色;

\* 利用边框绘制倒三角举例

```css

div{

width: 0px;

border-top: 50px solid black;

border-right: 50px solid white;

border-bottom: 50px solid white;

border-left:50px solid white;

}

```

#### 内边距

内容和边框之间的距离

\* 内边距的设置

> \* 单一样式

>> \* padding-top

>> \* padding-right

>> \* padding-bottom

>> \* padding-left

> \* 复合样式

>> \* padding:上 右 下 左;

>> \* 按照顺时针方向设置。如果省略了一个边，就取对边的值，如果只设置了一个边，那么所有的边都一样

\* 注意点

> \* 设置了padding后，元素的高度和宽度会发生变化

> \* 内边距是有背景色的，他的背景色和内容的背景色一样

#### 外边距

标签和标签之间的距离

\* 外边距的设置

> \* 单一样式

>> \* margin-top:

>> \* margin-right:

>> \* margin-bottom:

>> \* margin-left:

> \* 复合样式

>> \* margin:上 右 下 左;

>> \* 按照顺时针方向设置。如果省略了一个边，就取对边的值，如果只设置了一个边，那么所有的边都一样

\* 注意点

> \* 水平方向和垂直方向上的margin值可以叠加

> \* 普通元素垂直方向上的margin值发生重叠现象，重叠后取较大的值

> \* 普通的父子元素，给子元素设置margin top时候，这个值会传递到父元素,可以通过给父元素添加边框或者是内边距来阻止margin值的传递,还可以给父元素添加overflow:hidden来阻止margin的值传递

> \* 外边距没有背景色

> \* 外边距不会影响元素的宽和高，但是会影响整个元素空间的宽和高

> \* padding主要是用来设置父元素和子元素之间的距离，margin主要是用来设置元素和元素之间的距离

> \* margin可以取负值,表示相反的方向

#### 内容

可以真正使用的部分，由width/height来设置

\* 内容的设置

> width:

>

> height:

\* 注意点

> \* 元素的宽和高决定了真正可以填充内容的空间

> \* 取值可以是具体的像素也可以是百分比,百分比是相对于自己的父元素来说的

#### 盒子模型的计算

<img src="008-boxModel.jpeg" alt="">

\* 内容的宽和高

> \* 内容的宽:通过width属性设置的值

> \* 内容的高:通过height属性设置的值

\* 元素的宽和高

> \* 元素的宽：左边框＋左内边距＋内容的宽度＋右内边距＋右边框

> \* 元素的高：上边框＋上内边距＋内容的高度＋下内边距＋下边框

\* 元素空间的宽和高

> \* 元素空间的宽：左外边距＋左边框＋左内边距＋内容的宽度＋右内边距＋右边框＋右外边距

> \* 元素空间的高：上外边距＋上边框＋上内边距＋内容的高度＋下内边距＋下边框＋下外边距

#### 盒子模型的注意点

\* 盒子新增padding和border之后,如何保证盒子元素的宽度和高度不变

> \* 方法一,通过计算并修改元素的内容的宽和高来实现

> \* 方法二,添加box-sizing:border-box

>> \* box-sizing有两个取值,content-box(默认值),不会自动计算宽度和高度,border-box,自动计算宽度和高度

>> \* 使用border-box属性,元素的宽高 == width/height的宽高

\* 居中

居中分为水平居中和垂直居中，水平居中就是元素的左右边框离父元素的左右边框距离一样，垂直居中同理。

> \* 元素居中的方法

>> \* 方法一,设置父元素的padding值

>> \* 方法二,设置子元素的margin值

>> \* 方法三,通过设置左右边距为auto的方法让元素在父元素水平居中

>>>> \* 使用auto水平居中的前提是元素需要设置宽度

>>>> \* 垂直居中不可以通过auto的方法

> \* text-align:center和margin:0 auto的区别

>> \* text-align:center只能让内容(文本／图片)居中

>> \* margin:0 auto 是让元素自身相对于父元素水平居中

\* 样式重置

> \* 为什么要清空默认的内外边距

>> 因为HTML标签自身有默认的内外边距，这样不利于对盒模型空间的计算，为了解决这个问题，在实际的开发中需要清空默认的内外边距

> \* 样式重置的方法

>> \* 方法一,用通用选择器(不推荐使用)

>>

>> ```

>> \*{

>> margin: 0;

>> padding: 0;

>> }

>> ```

>>

>> \* 方法二,用成熟的框架,如[雅虎CSS重置代码](http://yui.yahooapis.com/3.5.0/build/cssreset/cssreset-min.css)

```css

html{color:#000;background:#FFF}body,div,dl,dt,dd,ul,ol,li,h1,h2,h3,h4,h5,h6,pre,code,form,fieldset,legend,input,textarea,p,blockquote,th,td{margin:0;padding:0}table{border-collapse:collapse;border-spacing:0}fieldset,img{border:0}address,caption,cite,code,dfn,em,strong,th,var{font-style:normal;font-weight:normal}ol,ul{list-style:none}caption,th{text-align:left}h1,h2,h3,h4,h5,h6{font-size:100%;font-weight:normal}q:before,q:after{content:''}abbr,acronym{border:0;font-variant:normal}sup{vertical-align:text-top}sub{vertical-align:text-bottom}input,textarea,select{font-family:inherit;font-size:inherit;font-weight:inherit}input,textarea,select{\*font-size:100%}legend{color:#000}#yui3-css-stamp.cssreset{display:none}

```

> \* 注意样式重置不仅仅是清空默认的边距,还有其他的默认样式

\* 行高

行高指的是每行内容的高度,可以通过设置line-height属性来改变行高

> 注意点:

> \* 文字默认的是在自身的行内垂直居中的，所以可以利用这个属性，将行高和盒子的高度设置成一样来让一行的文字垂直居中

> \* 如果盒子中有多行文字, 就不能使用设置行高等于盒子高来实现让文字垂直居中, 只能通过设置padding来让文字居中

> \* 行高和盒子高不是同一个概念,盒子高指的是元素的高度

\* 盒子阴影

给盒子添加阴影

> box-shadow: 水平距离 垂直距离 模糊距离 阴影大小 阴影颜色 内外阴影;

>

> 注意点:

> \* 背景阴影默认的颜色是盒子内容的颜色一样

> \* 快速添加阴影只需写前三个参数即可

> \* 前两个参数允许取负值

> \* 默认是外阴影(outset),可以把最后一个参数改为inset来设置为内阴影

\* 圆角边框

给边框添加圆角

\* 圆角的设置

> \* 单一样式

>> \* border-top-left-radius:取值1 [取值2];

>> \* border-top-right-radius:取值1 [取值2];

>> \* border-bottom-right-radius:取值1 [取值2];

>> \* border-bottom-left-radius:取值1 [取值2];

> \* 复合样式

>> \* border-radius:取值1(左上 右上 右下 左下)/[取值2(左上 右上 右下 左下)];

>> \* 按照顺时针方向设置。如果省略了一个点，就取对点的值，如果只设置了一个点，那么所有的点都一样

>

> \* 取值说明

>> \* 取值可以是具体的数字加单位也可以是百分比,百分比是相对于元素的宽和高的,即包含边框和内边距

>> \* 取值1代表水平方向半径,取值2代表竖直方向半径

>> \* 取值2省略的情况下，水平方向和竖直方向的半径相等

>> \* 大值特性,当值很大的时候，只会使用能够渲染的圆角大小渲染。

>> \* 圆角具有外半径和内半径之分,当radius半径值小于border的值时，没有内半径，所以内角是直角，当radius半径值大于border的值时，才有内半径，也才有内圆角。内半径的值等于外半径的值减去border的值。

>> \* 圆心是从边框的外沿垂直向内由水平和垂直的半径相交的点

>> \* 只要水平和垂直的半径是相等的，产生的就是四分之一圆。

>> \* 半径为50%,如果宽高一样,就是一个正圆,如果宽高不一样时就是一个椭圆

>> \* 圆角是设置给边框的

> \* 绘制月牙举例

```css

.moon{

width:100px;

height:100px;

border-right:20px solid #F52;

border-top-right-radius: 50%;

border-bottom-right-radius: 50%;

}

```

#### 盒子模型案例-百度新闻

<img src="008-box-demo.png" alt="">

### 设置背景颜色

\* 格式:background-color: 颜色的英文单词|rgb|rgba|十六进制

###设置背景图片

\* 格式:background-image:url(图片地址)

\* 注意点:

> \* 图片默认是在水平和垂直方向上平铺的

> \* 图片的地址可以是本地资源，也可以是网络资源

###背景图片平铺的设置

\* 格式:background-repeat: repeat-x（只在水平方向上平铺）| repeat-y（只在垂直方向上平铺）| no-repeat 水平和垂直方向上都不平铺;

###背景图片固定

\* 格式:background-attachment: scroll(背景图片会随着滚动条的滚动而滚动)|fixed(背景图片固定在某个位置,不随滚动条的滚动而滚动);

###背景图片的位置定位

\* 格式:background-position: 水平移动距离 垂直移动距离;

\* 取值:具体的数值+px|方向单词

> \* 方向单词：水平:right center left 垂直:top center bottom

> \* 移动的参考点是容器的左上点

###背景的复合样式

\* 格式:background:背景颜色 背景图片 平铺设置 背景图片固定 图片定位

\* 注意点:复合样式中的属性可以都写，也可以省略，也可以交换位置，但是图片定位的两个属性要写在一起不能拆开

###CSS精灵图的使用

由于每显示一张图片，浏览器都会请求一次服务器，所以我们可以把多张图片合成一张

，这种合成图片的技术就叫做精灵图

\* 使用CSS精灵图的方法

> \* 给元素设置一个宽高，宽高就是需要显示区域的宽和高

> \* 给元素添加背景图片

> \* 在图片中找到需要显示元素的坐标，然后根据坐标设置图片背景的位置

\* 图片标签和背景图片的区别

> \* 图片标签占用空间，背景图片不占用

> \* 图片标签的语义强，背景图片只是起到装饰作用

###设置背景图片的大小

\* 格式:background-size:宽度 高度 | 单词

\* 注意点:

> \* 宽度和高度的取值有:

>> \* 具体的像素

>> \* 百分比(相对于容器)

>> \* auto(谁设置为auto把谁等比拉伸)

> \* 单词的取值有:

>> \* cover,图片等比拉伸到宽度和高度都填满元素

>> \* contain,图片等比拉伸到宽度或高度都填满元素

###指定背景图片开始显示区域

\* 格式:background-origin:padding-box(从内边距开始显示)|border-box(从边框开始显示)|content-box(从内容区域开始显示)

\* 注意点:该属性用来指定从哪个区域开始显示背景图片,默认是padding-box

###指定背景颜色开始显示区域

\* 格式:background-clip:padding-box(从内边距开始显示)|border-box(从边框开始显示)|content-box(从内容区域开始显示)

\* 注意点:该属性用来指定从哪个区域开始显示背景颜色,默认是border-box

###背景图片练习

<img src="009-bj-demo.png" alt="">

### HTML中标签的分类

\* 容器级

能嵌套其它所有标签的标签

> div header nav section article footer h ul ol dl li dt dd a ...

\* 文本级

只能嵌套文字/图片/视频/音频/超链接的标签

> p span b u i s stong em ins del ...

### CSS中标签的分类

<img src="010-CSS显示模式.png" alt="">

\* 常见的块级元素

> p div header nav section article footer h ul ol dl li dt dd ...

\* 常见的行内元素

> span b u i s stong em ins del ...

\* 常见的行内块级元素

> img input textarea ...

### CSS元素显示模式的转换

\* 给元素添加display属性可以手动的改变元素的显示模式,常见取值有:

> \* block 块级

> \* inline 行内

> \* inline-block 行内块级

### 行内块状元素垂直对齐

\* 什么是基线

其实网页中的文字可以想象为显示在四线各种,而四线格中底部倒数第二条线就是基线

<img src="010-base\_line.jpg" alt="">

> 注意点:如果给元素设置了行高,四线格的高度还是文字的高度

\* 如何设置垂直对齐

> \* 格式:vertical-align:baseline|top|middle|bottom|text-bottom

> \* 注意点

>> \* 只有行内块状元素设置vertical-align属性才有效

>> \* 默认的对齐方式是baseline(基线对齐)

>> \* top是相对于父元素的顶部对齐

>> \* middle是相对于父元素的中部对齐

>> \* bottom是相对于是相对于父元素的底部对齐

>> \* text-bottom是相对于父元素的文字底部对齐

### 1.什么是网页布局？

对网页中各个元素进行排版

### 2.标准流的网页布局

标准流的网页布局是浏览器默认的排版方式

> \* 行内元素、行内块状元素从左到右

> \* 块状元素从上向下

> \* 流可以理解为流式排列

### 浮动流布局

给元素添加float属性,元素就属性浮动流了

> \* float的取值只有左(left)对齐和右(right)对齐，没有中间对齐,对齐是相对于父元素的左右边框的

> \* 浮动流主要是用来水平排版的

> \* 虽然把元素设置为行内块状也可以实现水平排版,但有不足之处,例如左右水平排版,如果父元素宽度不固定的情况下会发生错位,所以在项目中一般都用浮动流来实现水平排版

### 浮动元素脱标

可以想象为在我们看到标准流浏览器的上面还浮动了一层透明的浮动流浏览器,浮动元素脱标就是当给这个元素添加浮动属性后,他就会从标准流浏览器从删除,然后跑到了那层透明的浮动流浏览器中

> \* 浮动元素脱标后对标准流的影响

>> \* 由于脱离标准后,标准流中位置空闲了出来,因此后面标准流中的元素会去顶替脱离标准流的那个元素的位置

> \* 浮动元素脱标后的在浮动流中的特点

>> \* 因为脱离标准流,所以浮动的元素不区分块状，行内，行内块状，他的属性和行内块状一样

>> \* 浮动流的元素不可以用margin: 0 auto;来水平居中

> \* 浮动元素脱标后在浮动流中的排列规则

>> \* 同方向上,先浮动的显示在前面，后浮动的显示在后面,注意这个先后指的结构上的先后而不是css的书写先后

>> \* 不同方向上,左浮动找左浮动元素，右浮动找右浮动元素

>> \* 浮动后元素的位置由浮动前元素在标准流中的位置决定,这个位置主要指的是浮动前在标准流中是第几行,浮动后还是显示在第几行

>> \* 由于浮动元素的位置和浮动前在标准流的位置有光,所以严格意识上说浮动元素是非完全脱标

### 浮动元素贴靠现象

当父元素的宽度不足以撑起所有浮动的子元素的时候，会发生贴靠，即后面的元素会找他前面元素的上一个元素贴靠，如果还不够，再向上找，如果都不够的话，会贴靠父元素

### 浮动元素围绕现象

浮动的元素不能盖住没有浮动元素的文字

### 浮动布局规律

垂直方向上用标准流块状元素,水平方向上用浮动元素

### 浮动的元素无法撑起父级块状元素的高度

标准流中的块状元素的默认高度是由内容撑起来的,但是由于浮动元素脱离了标准流，所以浮动的元素是撑不起来父级块状元素的高度的

### 为什么要清除浮动

由于浮动元素撑不起盒子的高度，会让后面盒子里浮动的元素找前面盒子里浮动的元素

### 清除浮动的方式

> \* 方式一,给前面的盒子添加高度

>> \* 项目中一般高度是有内容撑起来的,尽量不设置高度

> \* 方式二,通过给第二个盒子添加clear属性来清除前面元素浮动带来的影响

>> \* clear属性的取值有:left|right|both|none(默认)

>> \* clear属性的意思是不让自己的子元素找前面浮动的元素

>> \* 通过clear属性清除浮动会导致第一个盒子的margin-bottom和第二个盒子的margin-top失效,第二个盒子的margin-top失效是因为margin的值传递

> \* 方式三,给第一个盒子和第二个盒子之间添加一个空盒子，并且给这个空盒子设置clear:both属性,俗称外墙法

>> \* 第一个盒子的margin-bottom依然不可以用,但是第二个盒子的margin-top可以用

>> \* 可以通过设置空盒子的高度来达到设置margin-bottom/top的效果

> \* 方法四,给第一个盒子所有元素的后面添加一个空盒子，并且给这个空盒子设置clear:both属性,俗称内墙法

>> \* 第一个盒子的margin-bottom可以用,第二个盒子的margin-top也可以用

>> \* 也可以通过设置空盒子的高度来达到设置margin-bottom/top的效果

>> \* 内墙法可以撑起盒子的高度，而外墙法不可以

> \* 方法五,通过伪元素选择器(:after)实现内墙法

>> \* 作用和内墙法一样，但是不用额外添加空元素，可以实现结构和样式分离，在实际项目开发中推荐使用

```css

.clearfix:after{

content: "";

clear: both;

display: block;

height: 0px;

visibility: hidden;

}

.clearfix{

\*zoom:1;

}

```

> \* 方式六,通过给第一个盒子添加overflow: hidden来清除浮动

### 布局练习

\* 布局练习1

<img src="012-float-demo.png" width='980px' alt="">

\* 布局练习2

<img src="012-float-demo2.png" width='980px' alt="">

### 相对定位

相对于自己在标准流当中的位置发生的移动

\* 注意点

> \* 只写position: relative;元素是不会发生任何移动的,需要配合设置四条边的移动距离来一起使用

> \* 设置四条边的移动距离分别用 top|right|bottom|left

> \* 四条边的移动距可以写一个,也可以写两个,写两个的时候要注意在同一个方向的只能设置一个边

> \* 相对定位元素是不脱离标准流的,所以相对定位元素在标准流当中的空间不会释放出来

> \* 相对定位元素是不脱离标准流的,所以相对定位元素区分块状，行内，行内块状

> \* 如果元素有margin/border/padding值,相对定位是在margin/border/padding值生效后起作用

> \* 相对定位元素主要用来对对元素进行微调和设置绝对定位的参考元素

### 绝对定位

相对于父元素发生位置的移动

\* 注意点

> \* 只写position: absolute;元素是不会发生任何移动的,需要配合设置四条边的移动距离来一起使用

> \* 设置四条边的移动距离分别用 top|right|bottom|left

> \* 四条边的移动距可以写一个,也可以写两个,写两个的时候要注意在同一个方向的只能设置一个边

> \* 绝对定位元素是脱离标准流的,所以绝对定位元素在标准流中的空间会被释放出来

> \* 绝对定位元素是脱离标准流的,所以绝对定位元素不区分块状,行内,行内块状

\* 绝对定位元素的参考点

> \* 绝对定位元素会参考离他最近的具有定位属性的祖先元素,如果祖先中没有定位元素就参考body元素,换句话说如果一个元素设置为绝对定位,首先去找他的父元素,看他的父元素有没有position属性,如果没有就找他的父亲的父亲,直到找到第一个具有position属性的祖先,然后相对于该祖先发生位置移动,如果所有祖先都没有position属性,就相对于body发生位置的移动

> \* 具有定位属性指的是相对定位,绝对定位,固定定位,但不包括静态定位

> \* 父元素的margin/border的值对绝对定位的子元素的位置没有影响,绝对定位的子元素是从父元素的padding区域开始参考的,但如果相对于body,始终相当于相对于浏览器的首屏的可视区域

\* 相对定位和绝对定位配合使用的方法

> \* 子元素用绝对定位

> \* 需要参考的父级元素用相对定位

\* 绝对定位的元素水平居中

> left:50%

> margin-left: - 元素宽度的一半

### 固定定位

参照body发生位置的移动,固定定位的元素不会随着滚动条的移动而移动

\* 注意点

>\* 只写position: fixed;元素是不会发生任何移动的,需要配合设置四条边的移动距离来一起使用

>\* 设置四条边的移动距离分别用 top|right|bottom|left

>\* 固定定位元素是脱离标准流的,所以固定定位元素在标准流中的空间会被释放出来

>\* 固定定位元素是脱离标准流的,所以固定定位元素不区分块状,行内,行内块状

>\* 固定定位元素不会随着滚动条的移动而移动

>\* 固定定位元素始终参考的都是body

### 静态定位

>\* 默认情况下是静态定位

### 定位流中元素显示的顺序

\* 默认情况下,定位流中的元素会盖住标准流中的元素

\* 默认情况下,定位流中后面的元素会覆盖住前面的元素

\* 可以通过z-index来更改元素的覆盖关系,值越大,显示越靠前,默认值是0

\* 父级元素的z-index属性会影响到子元素的z-index,子元素的z-index属性最终的值是所有父级元素中最顶层父级元素的z-index值

### 案例练习

<img src="013-demo.png" alt="">

### 过渡

让元素的状态逐渐的发生变化

\* 添加过渡的必要条件

> \* 给需要过渡的元素添加transition-property属性来指定那个属性需要过渡,如果有多个属性同时变化,将多个属性用逗号隔开即可

> \* 给需要过渡的元素添加transition-duration属性来指定过渡的执行时间,如果有多个属性同时变化,将过渡的执行时间用逗号隔开即可

> \* 给需要过渡的元素添加一个触发事件(通常是hover),在事件中指定需要过渡的属性的变化值

\* 过渡的其他属性

> \* 延迟过渡的执行, 例如: transition-delay: 2s;

> \* 过渡的执行速度, 例如: transition-timing-function: linear|ease 默认的|ease-in|ease-out|ease-in-out|cubic-bezier(x1,y1,x2,y2)

>> \* 预设的速度都可以用三次贝塞尔曲线函数cubic-bezier(x1,y1,x2,y2)来实现

>> <img src="./014-cubic.png" alt="">

>> \* 分别用A,B,C,D表示这四个点，其中起始点固定值为A(0,0),终止点固定为D(1,1)剩下的中间点B(x1,y1),C(x2,y2)也就是所要动态操控的两个点了,对应cubic-bezier (x1,y1,x2,y2)中的四个参数,通过改变B,C两点的坐标值来动态生成一条贝塞尔曲线表示动画中的速度变化。

\* 过渡复合样式的写法

> \* 一组属性的每个属性用空格隔开

>> transition:需要过渡的属性 执行过渡的时间 执行过渡的速度 延迟时间;

```css

transition:width 3s linear 2s;

```

> \* 多组属性之间用逗号隔开

```css

transition:width 3s linear 2s,background-color 3s linear 2s;

```

> \* 如果有多个属性发生同样的过渡效果的话可以把多个属性简写为all

```css

transition:all 3s;

```

\* 添加过渡的步骤

> \* 布局页面

> \* 触发修改属性

> \* 给修改属性添加过渡效果

### 过渡案例

手风琴效果

### 2D转换

\* 旋转

> \* 格式: transform: rotate(ndeg);

>

> \* 注意点:

>> \* n是一个具体的数字,deg代表度数的单位

\* 平移

> \* 格式: transform: translate(水平方向移动距离,垂直方向移动距离);

>

> \* 注意点:

>> \* 移动距离可以是负数,负数代表向反反向移动

\* 缩放

> \* 格式: transform: scale(水平方向缩放比例,垂直方向缩放比例);

>

> \* 注意点:

>> \* 缩放比例如果是1代表不变,大于1代表放大,小于1代表缩小

>> \* 如果水平和垂直方向上缩放比例一样,可以简写一个

\* 复合样式

> \* 格式: transform: rotate(ndeg) translate(水平方向移动距离,垂直方向移动距离) scale(水平方向缩放比例,垂直方向缩放比例);

>

> \* 注意点:

>> \* 多个转换之间用空格隔开

>> \* 旋转会修改坐标系,因此旋转后再平移(即旋转写在平移前面)是在旋转修改后的坐标系下平移

\* 修改旋转基点

旋转基点就是以哪个点为参考点进行旋转

> \* 格式: transform-origin: 水平方向移动距离 垂直方向移动距离;

>

> \* 注意点:

>> \* 默认情况下所有的元素都是以自己的中心点作为参考来旋转的

>> \* 移动距离可以是具体的像素,百分比,和关键字,关键字在水平方向上有left center right,在垂直方向上有top center bottom

>> \* 移动基点的参考点是元素的左上角顶点

\* 修改旋转轴向

在rotate后面加上轴向可以改变旋转的轴向,即rotateZ是沿着Z轴旋转,rotateX是沿着X轴旋转,rotateY是沿着Y轴旋转

> \* 默认情况下元素是沿着Z轴旋转的,即rotateZ和rotate是等价的

> \* 为了让元素呈现近大远小效果,需要给元素的父元素上面添加透视距离属性perspective

### 案例

\* 倒地照片

\* 相片墙

\* 翻转菜单

### 动画的基本使用

\* 创建动画的步骤

> \* 给需要执行动画的元素指定动画的名称

> \* 创建指定名称的动画

> \* 给需要执行动画的元素指定动画的执行时间

\* 过渡和动画的区别

> 过渡需要人为的触发,而动画不需要

### 动画的其他属性

\* 延迟执行

```css

animation-delay: 2s;

```

\* 执行速度

```css

animation-timing-function: linear|ease 默认的|ease-in|ease-out|ease-in-out|cubic-bezier(x1,y1,x2,y2)

```

> 注意：动画执行的速度和过渡的一样

\* 执行次数

```css

animation-iteration-count: 具体的数字(默认是1)|infinite(无限循环);

```

\* 是否执行往返动画

```css

animation-direction: normal(默认状态不执行往返动画)|alternate(执行往返动画);

```

> 注意如果是执行往返动画,返回执行也算一次动画

\* 改变动画的执行状态

```css

animation-play-state: running(默认状态动画执行)|paused(动画暂停)

```

\* 规定动画等待时和结束时的显示状态

```css

animation-fill-mode:none(不做改变)|backwards(动画等待时显示动画的第一帧)|forwards(动画执行结束保持显示动画的最后一帧)|both(动画执行前显示动画的第一帧并且动画执行结束显示动画的最后一帧)

```

### 动画的复合样式写法

```css

animation:动画的名称 动画的执行时间 动画的速度 延迟时间 动画的执行次数 是否执行往返动画;

```

> 注意：动画名称和执行时间不可以省略

### 创建动画的方式二

用百分比创建动画

> 注意点:

>

> \* 0%代表动画开始,100%代表动画结束

>

> \* 0%和100%推荐都写

### 案例

滚动焦点图

### 添加3D效果

给需要实现3D效果的元素的父元素添加transform-style: preserve-3d;属性

\* 注意点:

> \* 元素默认是2D展示的

> \* 3D是有厚度的,是立体的,相当于长方体,而2D没有厚度,是平面的,相当于长方形

### 案例

\* 正方体

\* 长方体

\* 3D轮播图

### 常见字体文件格式

\* EOF(Embedded Open Type) 嵌入式字体，是微软开发的技术。

\* WOFF(Web Open Font Format) Web开发字体格式,一种专门为了Web而设计的字体格式标准

\* TTF(TrueType Font) Windows和Mac系统最常用的字体格式

\* SVG(Scalable Vector Graphics) 由W3C制定的开放标准的图形格式

### 字体浏览器的支持情况

<img src="img/018-1.png" alt="">

### 自定义字体的方法

> \* 准备自定义字体文件,通常可以从网上下载

> \* 使用@font-face 定义一个新字体

```css

@font-face{

font-family: 自定义字体名称;

src:url(字体文件1),

url(字体文件2);

[font-weight: bold;]

[font-style: italic;]

}

```

> \* 在CSS中使用自定义的字体

```css

font-family: 自定义字体名称;

```

> \* 注意点

>> \* src中的url可以是本地字体文件也可以是网络上的字体文件,如果是网络上的字体文件注意跨域的问题

>> \* 自定义字体默认的font-weight和font-style都是normal

>> \* font-weight和font-style是可选项

>> \* 加粗和倾斜本质上也是对应有一个字体文件

>> \* font-weight需要单独指定一个包含了粗体的字体文件,如果不指定,使用自定义字体时,浏览器会使用默认的加粗效果

>> \* font-style需要单独指定一个包含了斜体的字体文件,如果不指定,使用自定义字体时,浏览器会使用默认的倾斜效果

### 自定义字体实现图标

把图标做成字体文件,在页面中以字体的形式使用这些图标

\* [fontawesome图标库](https://fontawesome.com/)

\* 使用方法

> \* 方法一: CSS+字体

>

```css

<i class="fas fa-camera-retro"></i>

```

>

> \* 方法二: HMTL+字体

```css

<i class="fas">&#xf083;</i>

```

> 注意点:

>> \* 无论是方法一还是方法二,本质都是在网页中插入一个四位的16进制数,然后用特殊的字体把他显示出来

>> \* 网页中插入16进制数,需要在前面面添加&#x

>> \* 无论是方法一还是方法二,都需要在使用前引入相应的css文件

>> \* 需要注意引入的css文件中使用的字体文件的路径