Table of Contents

简介

这是一本基于 gitbook 的电子书，在这里详细的列出我们基础班五天所要学习到的内容。对于大家来说，我们可以通过它来完成课前的预期以及课后的复习，但是这里的内容都非大家亲手整理，所以大家在完成每日的复习工作之后需要利用相应的 “脑图” 软件对当天所学内容进行系统的梳理，将分散的知识点在最短的时间内整理出一条主线。通过这种形式来告诉自己今天我们学到了什么，而不是去想老师今天讲了什么。每天积累最终收获！

> 梦想，可以天花乱坠，理想，是我们一步一个脚印踩出来的坎坷道路。

学习目标

1.说出计算机基本定义与特点

2. 说出计算机的硬件系统组成

3. 说出计算机的软件系统组成

4. 了解计算机采用二进制

5. 说出常见的计算机数据计量单位

6. 写出部分常见的**DOS**命令操作

计算机基本介绍

一、为什么需计算机

随着时代的发展每个人需要处理的信息越来越多，如果单纯地依靠人脑来进行处理那么效率是非常低的，因此慢慢的就有了电子计算设备，也就是计算机。所以计算机的出现可以理解为是最大限度的用电脑来模拟和代替人脑的工作。

二、计算机定义

计算机就是一种可以自动高效进行计算操作的电子设备，我们俗称电脑**(PC)**

三、计算机基本特点

1. 计算机可以完成数学和逻辑运算
2. 计算机可以对数据进行记忆和存储
3. 计算机可以在程序指令下自动高效地进行计算

四、计算机组成

1. 计算机由硬件系统+软件系统二大部分组成
2. 硬件：对于计算机来说就是那些我们看得见摸得着的物理设备
3. 软件：软件就是运行在机器中操控硬件、实现指定功能的程序集合

计算机硬件系统

一、计算机为什么需要硬件

所有的计算功能最终都还是要借助于硬件设备来完成

二、计算机硬件介绍

1. 现代电子计算机的硬件部分都会依据冯诺依曼理论将它分为五个部分
2. 输入设备：核心功能就是以不同的形式给计算机提供数据。例如：键盘 鼠标....
3. 输出设备：核心功能就是将计算机处理后的数据展示出来。例如：屏幕 音响....
4. 计算器+控制器： 这二个部分合在一起就是我们平时所说的中央处理器(cpu)，计算器的功能就是完成最终的运算，控制器的功能就是来设置当前数据该如何计算
5. 存储器：对于计算机来说存储器我们分为内存和外存二种，无论是哪一种它们的作用都是对数据进行管理

三、内存与外存相关

1. 内存就是我们平时所说的内存条，大小一般4G-16G不等
2. 外存我们可以理解为是硬盘等，大小一般为500G左右
3. 内存的计算速度相对于外存来说非常的快
4. 内存中的数据断电之后会消失，外存不会
5. 内存我们又分为只读内存（ROM）和随机内存(RAM)

计算机软件系统

一、为什么需要软件系统

如果计算机只有硬件而没有软件那么硬件系统将不知道如何进行工作

二、软件系统基本介绍

1. 软件系统我们分为 系统软件 **(** 操作系统 **)** **+** 应用软件
2. 常见的操作系统软件
    i. 图形化桌面操作系统软件：
        i. windows： 微软件推出，用户量很大
        ii. Linux: 内核是由 李纳斯.杨 推出，目前流行着很多款不同的 linux 分支
        iii. MacOS： 苹果公司推出 可以看做是一款Linux的分支
    ii. 移动设备操作系统：android(底层也是一款小型的Linux) IOS .....
    iii. 服务器操作系统：Linux(开源 稳定) windows server(微软 收费)
3. 应用软件：我们可以认为就是安装在操作系统上的第三方功能软件

二进制基本介绍

一、二进制基本描述

1. 进制：在数学当中我们可以将进制理解为是一种人制定的计数规则。
2. 二进制就是多种规则里的一种，目前被广泛的应用于计算机当中
3. 二进制里只有0 和 1 二个基数，具体的规则就是逢二进1

二、计算机识别二进制原理

现代计算机都是通电进行工作，当电流通过计算机的电子元器件时会产生电压。这时人们就人为的选取一个电压值做为标准，将电压分为二类。比这个值大的称之为高电位(高频)，比这个值小的称之为低电位(低频)。此时就将所有的情况分成了高低二种状态，而刚好在二进制当中只有0 和 1 二个基数，所以就再次人为的将高电位用数字1进行表示，将低电位用数字0 表示。这样就间接的相当于计算机可以读懂0 和 1 二个数字，而0 和 1 可以组成无限的数字可能。

三、计算机采用二进制的优点

1. 技术实现简单，状态稳定
2. 二进制刚好与逻辑运算当中的真假对应。 1就可以代表真(true) 0 就可以代表假(false) [1==true 0==false]
3. 二进制可以方便地转成十进制

常见进制与转换

一、常见的数学进制

1. 二进制(bin):逢二进1 ，它里面只有0 和1 二个基数 【11011】
2. 八进制(oct)：逢8进1，它里面的基数是 0-7
3. 十进制(dec)：逢10进1，它里面基数是0-9
4. 十六进制(hex)：逢16进1，它里的基数就是0-9 A B C D E F

二、常见的进制转换

转换指的就是不同进制的数字之间是可以互相转化的。例如10进制的8 可以转为二进制的 1000，对于人类来说熟悉的是十进制，而计算机采用的是二进制，所以这里我们就讨论其它进制转成二进制和其它进制转十进制。

1. 其它进制转十进制:
    i. 确定当前被转换数字的位数和进制。例如：10010 为五位的2进制数字
    ii. 从右至左开始给每位上的数字进行编号，默认编号从0开始。例如：10011 最右端的1编号为0，倒数第二位的1编号为1

编号	第4位	第3位	第2位	第1位	第0位
数字	1	0	0	1	1

    iii. 将每位上的数字取出，乘以当前进制的 N 次方。其中 N 就等于当前数字所对应的编号
    iv. 最后将每位数字相乘的结果进行相加求和，最终的值就是对应的十进制大小
2. 十进制转二进制
    i. 将十进制的数字除以2 ，得到商和余数
    ii. 判断商是否为0，如果不为0则继续用这个商除去2
    iii. 直到商为0时结束，将这个除法过程中产生的余数反向排列
    iv. 最终的排列结果就是当前这个十进制转成二进制后的数值
3. 8/16进制转二进制
    i. 将八进制或十六制中的所有“基数”都转成二进制。（需要注意的就是八进制转二进制要写成三位，而十六进制要写在四位)
    ii. 有了上述的二进制基数转换之后，当我们拿到一个具体的八进制或者十六进制数值之后我们只需要将其进行拼接就可以

编码基本介绍

一、什么是编码

所谓的编码指的就是将人类可以理解自然语言 “翻译” 成计算机可以理解的机器语言

二、编码的思想

对于计算机来说能读懂的是二进制，而对于人类来说能识别的就是各种自然语言，此时我们就可以人为地将自然语言与二进制数据字进行 一 一 对应，如此一来我们就相当于制做了一张关系对应表。在这张表里将我们人类语言中的字符与二进制当中的数字建立起对应关系，这样一来计算机就可以间接读懂人类的自然语言了。

三、**ascii**码表介绍

依据编码的思想，我们人为地制作自然语言与二进制对应的关系表，当这种对应关系越来越多的时候我们就称之为编码表，其中 ascii 码表是全世界第一张 “单字节” 编码表，在这张表里一共定义了 256 个字符，分成三个部分：不可打印 + 可打印 + 扩展可打印

四、字符集

字符集和编码表是类似的概念，在 ascii 码表当中没有定义汉字编码，所以如果直接使用 ascii 码来处理中文会产生乱码的问题，因此后来就有人和组织基于 ascii 码表又慢慢的制定了其它的编码规则。其中最常见的处理中文的编码标准是 UTF8。

计算机数据计量单位

一、为什么有计量单位

计算机是用来处理数据的，那么当数据达到了一定的量级之后我们肯定要准备合适的单位来进行表达

二、基础的单位

1. 比特（bit 位）： 这是一个人为规定的最小计量单位
2. 节字（ bytes ）:这也是人为规定的单位，其中一字节等于8位

三、常用单位及换算关系

1. **1B = 8b**
2. **1KB = 1024B**
3. **1MB = 1024KB**
4. **1GB = 1024MB**
5. **1TB = 1024GB**
6. **1PB = 1024TB**

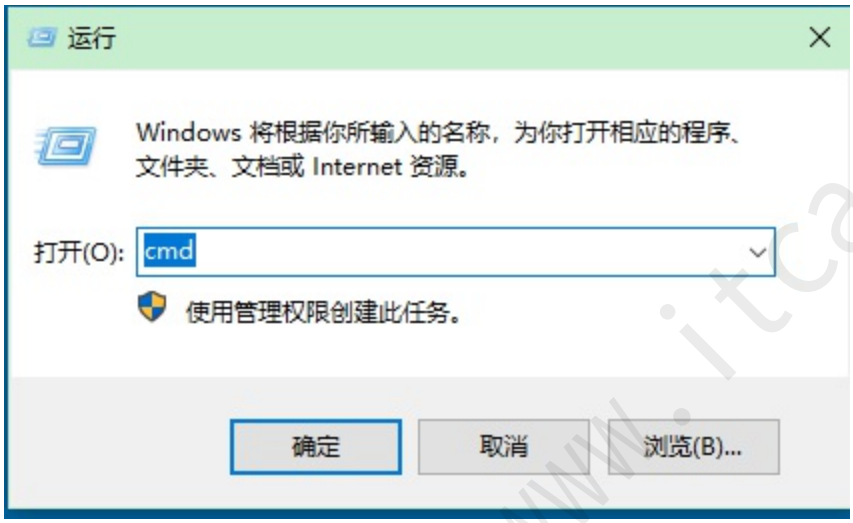
DOS命令使用

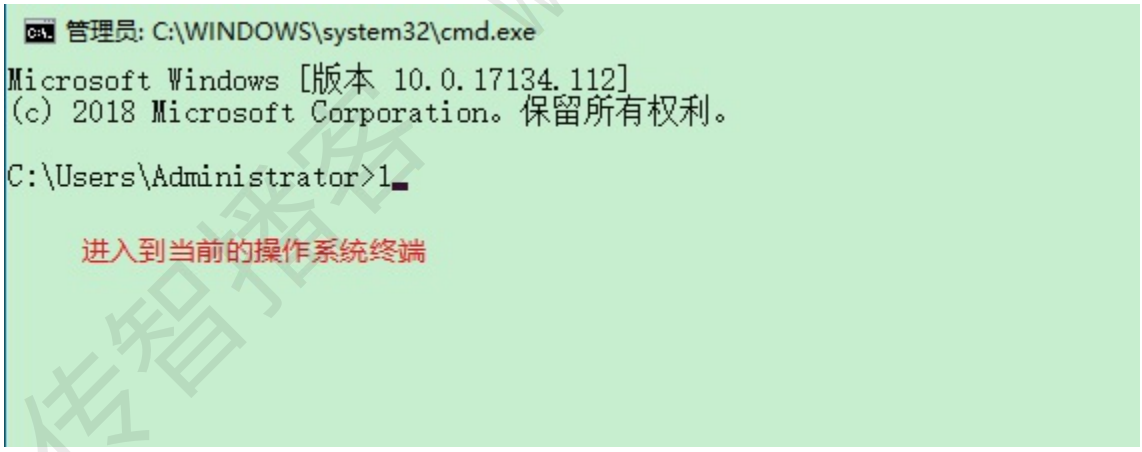
一、DOS基本介绍

所谓的DOS 我们可以理解为是一种操作系统，与我们当下常见的 windows 相比较而言，它里面没有图形化的界面或者鼠标点击操作，主要是通过命令行语句来达到操作计算机。

二、DOS 命令使用基本步骤

1. 通过 win + r 键操作打开运行面板，输入 cmd 进入操作系统终端





2. 在终端中输入 具体的 DOS 命令 然后回车执行查看结果
3. DOS 命令书写一般由 命令+参数组成，参数可以有0 个或者多个，命令名称与参数之间，多个参数之间一般用空格隔开

三、分类DOS 命令

1. 基础**DOS**命令

```
1. time : 显示出本机的当前时间
2. ctrl + c : 终止当前正执行的命令
3. shutdown  /s  /t :设置当前计算机在一定时间之后定时关机
```

```
4. shutdown /a : 取消当前计算机的定时关机任务
5. ping 域名： 向目标域名发送四次请求，同时返回域名的IP 地址
6. ipconfig -all : 可以获取到当前本机的 IP 地址与 mac 地址
7. ↑ ↓ 键： 可以调出之前使用过的历史 DOS 命令
8. cls: 将当前正在编辑的面板清空
9. dir : 将当前路径里最终目录的所有一级子目录展示出来
10. 盘符名称:    切换到某个盘符
11. cd ..  : 在当前的路径基础上向上回退一级
```

2. 切换目录相关命令

```
1. 盘符名称：  在当前路径下切换到另外一个具体的盘符下
2. cd ..  : 在当前路径下向上回退一级
3. cd / :  直接回退到当前路径的根目录
4. 注意  cd 命令是不能用来切换盘符的
```

3. 新建命令

```
1. md: make directory  意思就是新建目录
2. md 目录名称 :  在某一个路径下新建相应的目录 【 只能新建目录 】
3. echo 内容>目标文件： DOS 中可以间接实现新建文件的操作，在相应路径下输出内容到目标文件，同时新建目标文件
```

4. 删除命令

```
1. rd : remove directory  删除目录
2. rd 路径/目录： 删除相应的目录，默认只删除空目录 【 只能删除目录，不能删除文件 】
3. rd /s 目录名 :  可以删除非空目录
4. del 路径/文件名：删除相应的文件
5. del 目录： 询问是否删除某个目录下的所有一级子文件
```

三、**DOS**命令汇总

命令输入	命令作用	注释
time	显示当前时间	需要手动退出才能执行下个命令
ctrl+c	退出当前正在执行的命令	
shutdown /s /t 秒数	设置当前计算机定时关机	/s /t 左右都有空格
shutdown /a	取消自动关机命令	/a 左侧有空格
ping 网址	检查当前网址网络是否连通	可以返回网址对应的ip 和网络状态
ipconfig -all	显示当前计算机所有网卡信息	包含 本机ipv4地址与mac地址
↑↓键	可以翻看最近使用过的命令	帮助我们快速选中某个执行过的命令
盘符名称:	切换到某个盘符	盘符即电脑中的c盘 d盘.....
cd ..	返回上一级目录	.. 与cd 之间存在空格
cd /	返回当前盘符根目录	/ 与cd 之间存在空格
dir	将某个目录的内容以列表形式列出	显示当前目录下的目录与文件
md 路径/目录	可以新建目录	

echo "内容">文件名称	可以间接新建一个带有内容的文件	DOS中间接新建文件的方式
rd 路径/目录	删除目录	默认只能删除空目录
del 文件	删除文件	只能用来删除文件
rd /s	强制删除非空目录	
del 目录	询问是否删除目录里的所有文件	只删除当前目录下下的一级子目录
copy 文件原路径 文件新路径	将文件复制到其它地方	在复制的过程中可以修改名称
move 文件原路径 文件新路径	将文件剪切到其它地方	必须进入被剪切文件所在目录执行

学习目标

1. 了解 **web** 的基本含义

2. 了解**HTML** 的发展历史

3. 说出**HTML**的基本定义

4. 说出**HTML**网页常见的骨架标签

5. 写出常见的 标签、段落、文字标签

6. 写出图片标签及常见图片标签属性

7. 写出超链接标签

一、**web**相关名词介绍

在正式学习HTML 内容之前我们需要先了解一些 WEB 相关的名词解释，这样可以让我们更加明确 HTML 是什么，以及 HTML 的作用是什么。

1. **www** ：world wide web 的简称，我们称之为全球广域网，也叫万维网。对于我们来说就理解为是网站服务
2. 网站：由多个网页组合在一起的页面集合，本质就是一种基于 HTTP 协议和超文本标记语言的跨平台分布式图形信息系统。
3. 网页：通过浏览器展示 包含图片、文字、链接、声音.....等内容的一种HTML文件。

> 总结：
>
> 1. web 就是我们当前的互联网 网站服务，每个站点都会有相应的网页组合提供服务
> 2. 网页里面包含了基础文字 和 文字之外的内容，我们通过 HTML 语言来编写网页
> 3. 网页由前端开发使用 HTML 进行编写，而用户最终看到的是页面，所以这个中间的过程就由浏览器来负责将代码解析成图形界面

一、**HTML**基本定义

HTML 是单词 HyperText Markup Language 的缩写，中文称之为超文本标记语言。其中的超指的就是超链接， 在没有HTML之前，互联网是用来传播文字信息，而有了 HTML之后就 可以传输除了纯文字之外的 图片、声音、视频等内容。

二、**HTML** 语言发展历史

1. 在互联网最初的时候是没有 HTML 的，我们只能通过网络传输最简单的文字内容
2. 随着用户的要求越来越多，同时技术也在不断发展，就出现了一种可以表达文字内容之外的语言 HTML
3. HTML最初的版本是 HTML1.0，目前流行使用的版本是 HTML5。 在这个过程中还存在多个版本。其中有一个W3C组织负责制定 HTML 语言的标准

> 总结：
>
> 1. HTML 是叫超文本标记语言，主要用来书写网页
> 2. 在目前市面上主推的HTML 版本是 HTML5.0
> 3. W3C是一个负责制定 HTML 语法规则的组织
> 4. 浏览器生产厂商生产不同浏览器，用来解析展示网页

一、HTML 网页骨架定义

网页的本质就是一个 HTML 文件，经过浏览器解析之后展示给用户。而所有的网页都会有一部分相同且固定的结构，这部分内容我们就称之为网页骨架

二、HTML 网页骨架内容

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

> 解释：
>
> 1. DocTYPE html 就是 document type html 用来申明当前的文档类型是 html
> 2. html 是网页当中最大的标签，我们称之为根标签
> 3. head 称之为网页的头部，它里面的内容主要用来定义网页标签 及 给浏览器查看的一些信息
> 4. `<meta charset=utf-8>` 用来定义网页的编码标准
> 5. title 称之为网页标题标签 ，它里的内容会显示在浏览器的标签页上
> 6. body 称之为网页主体标签，它里面的内容都会显示在浏览器的白色窗口区域【网页里展示的内容都会写在body标签里】

一、Hbuilder 工具介绍

Hbuilder 是由 Dcloud 公司推的一款支持 HTML5 的 web 开发 IDE ， 本质上就是一款用来书写HTML 的代码编辑器，需要注意的是 能够用来编写 HTML 代码的编辑器有很多种，Hbuilder 只是其中的一款

二、Hbuilder 安装

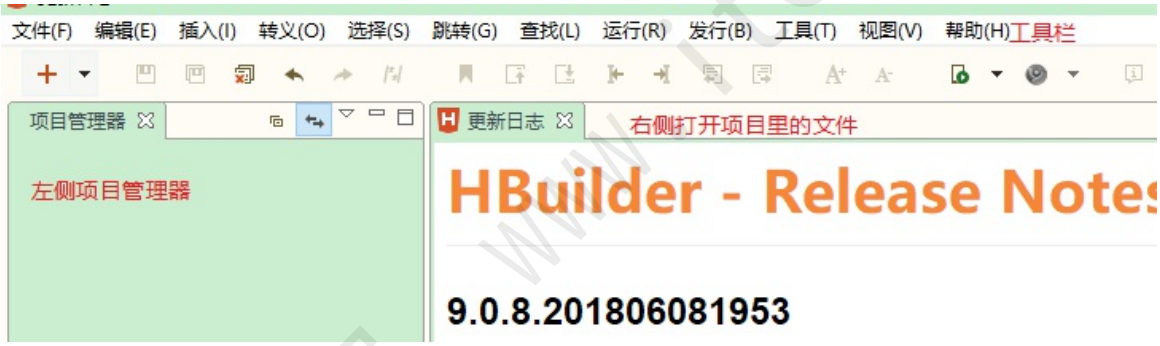
1. Hbuilder 是一款免安装软件，可以在官网上下载相应的版本
2. 将压缩包解压后 直接放置到自己习惯的软件安装目录下
3. 将启动项发送到桌面生成快捷方式

三、使用步骤

1. 点击启动项，打开登录面板，可以不注册使用

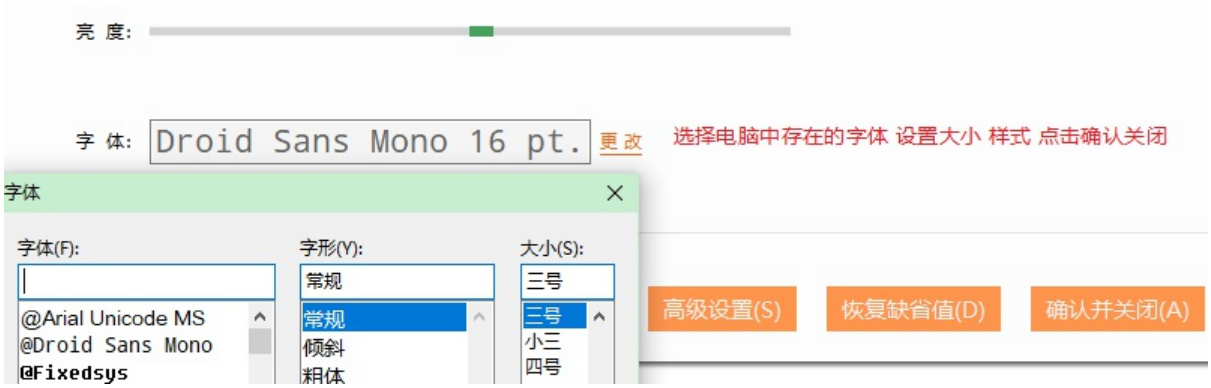
2. 第一次进入之后可以首先会看到配色选项设置，可以选择一个合适自己的灰色

3. 进入hbuilder 之后，上部是工具栏，下面部分为左右二边，左侧为默认hbuilder 项目，右侧面板默认为工具使用说明书，二者默认都可以删除



4. 最后可以设置 Hbuilder 的配置与显示字体，可以在工具栏选项中的 工具选项----视觉主题视置

5. 进入设置面板之后可以选择合适的字体，并设置具体样式后点击确认关闭



> 关于 Hbuilder 的设置可以根据个人爱好进行配置，但不设置同样可以使用

一、HTML 标签和语法

1. HTML 是超文本标记语言，当前主要被用来制作网页。在这个过程中我们可以将 网页比做我们认知当中的 “文章”，而 “文章”都是我们通过汉语当中的 汉字 来书写的，所以在 HTML 这门标记语言中也会存在着一些类似于 汉语中 文字的符号，这些符号我们就称之为标签，即我们要学习的 **HTML** 标签

2. HTML 做为一门语言，那么就和汉语一样 需要遵循一定的规则才能正确使用。而这种规则我们就称之为 HTML 语法，在我们使用 HTML 标签书写网页的时候必须遵循这些语法

二、HTML 标签分类及书写规则

1. HTML 标签被人为定义分为 单标签 和 双标签
2. 单标签书写规则：`<单标签名称 />`
3. 双标签书写规则: `<双标签名称></双标签名称>`
4. 严格的HTML 语法要求，无论是 单标签还是双标签都必须要正确关闭，单标签通过 / 关闭，双标签通过 /标签名 关闭
5. HTML 标签可以嵌套使用，但是不能出现交叉嵌套现象

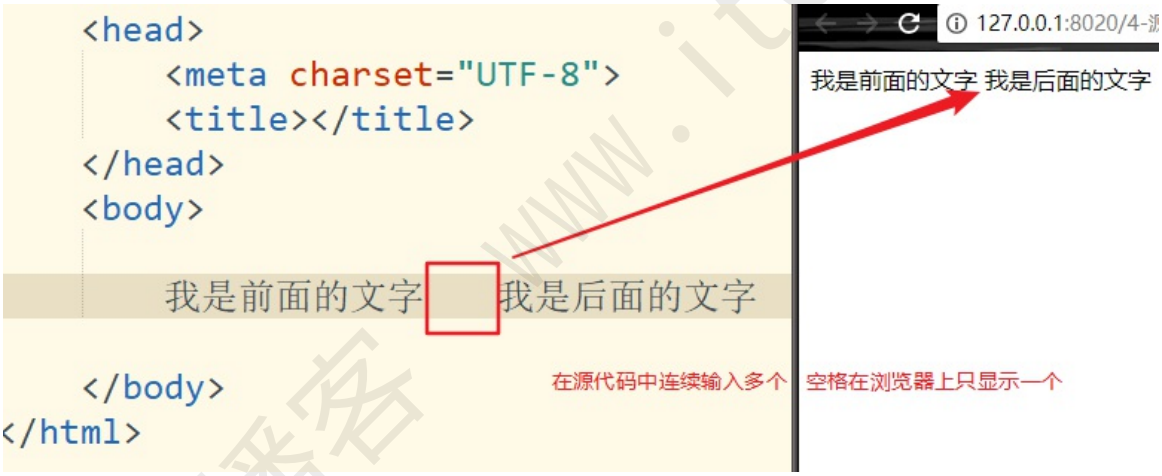
```
正确语法：
    <双标签名1>
        <双标签名2></双标签名2>
    </双标签名1>
```

```
错误语法：
    <双标签名1><双标签名2></双标签名1></双标签名2>
```

三、HTML 内容换行及空格

1. 在 HTML 源代码编写格式中，手动输入换行 在浏览器上是不会有效果的。

2. 在 HTML 源代码中手动多个空格 在浏览器上只会显示一个空格的效果



3. 在 HTML 中想要实现换行可以通过定义单标签 br .语法是 `<br />` ，想要添加空格可以通过实体符号 ` ` 如果采用的是 utf8 编码，那么三个 ` ` 相当于一个汉字

4. 通过上述的现象我们可以看出在 HTML 中如果想要实现某个效果，就必须依赖 HTML 当中的一些标签定义

一、标题标签

在 HTML 中定义了一组专门用表示网页标题的标签，从h1-h6 共计 六个 标签

```
<h1>标题1</h1>
<h2>标题2</h2>
<h3>标题3</h3>
<h4>标题4</h4>
<h5>标题5</h5>
<h6>标题6</h6>
```

二、段落标签

和标题不同 HTML 中定义了六种标题，而段落只有一个标签，标签名是 p，这是双标签

```
<p>段落内容</p>
```

三、文字标签

文字标签有很多，但是在书写网页的时候关于文字的外在样式我们一般通过 CSS 来进行设置，下面会分二组列出一些文字标签

```
弱语义：
<b>文字加粗</b>
<s>删除线</s>
<i>倾斜</i>
```

```
强语义：
<strong>文字加粗</strong>
<del>删除线</del>
<em>倾斜文字</em>
```

四、常用布局标签

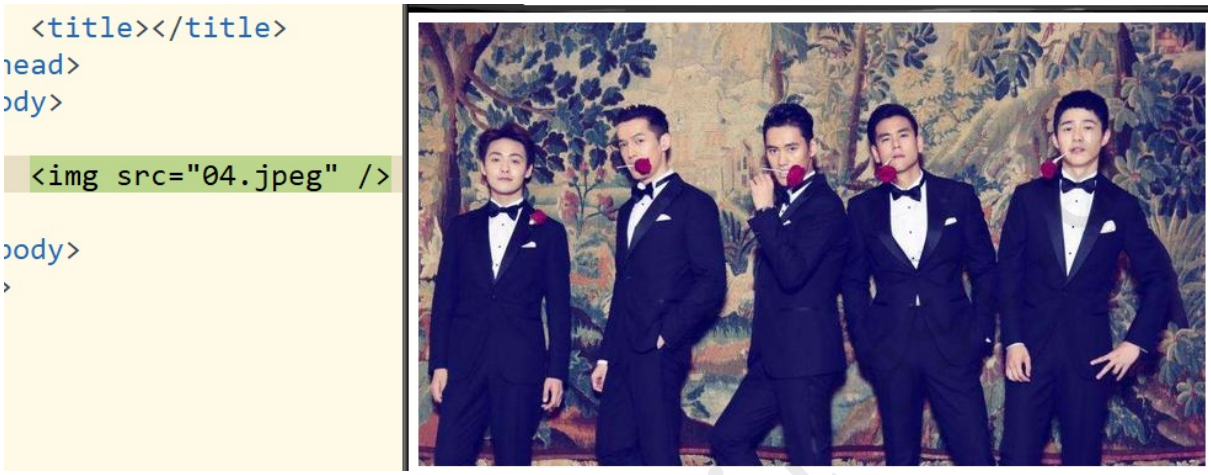
```
<div></div>
<span></span>
```

> 注：
>
> 1 div 和 span 是 HTML 当中很常用的二个标签。对于 div 来说我们可以认为它就表示一个大盒子。在它里面可以嵌套很多其它的小盒子
>
> 2 span 在语义上就是一个小盒子，一般用来存放一些文字内容等

一、图片标签基本介绍

在 HTML 当中定义了一个专门用于像网页中插入图片的标签，名称是 img 它是一个单标签。可以通过 img 标签身的 src 属性来插入图片。

```
<img src="04.jpeg" />
```



二、标签属性介绍

1. 标签可以具有属性，属性名与属性值之间用 单等号 进行连接。 `属性名=属性值` ，这个格式我们也称之为 键值对，属性值就是键名 ，属性值就是键值
2. 键值对与标签名之间需要用空格隔开，例如 src 与 img 之间的空格
3. 一个标签也可以具有多个属性键值对，多个属性键值对之间也需要用空格隔开 `<img 属性名1="值1" 属性名2="值2" />`
4. 属性值一般需要放在引号里，单引或者双引都可以，如果属性值是数字，那么也可以省略引号

三、图片标签常见属性

```
<img src="路径" title="图片标题" alt="图片提示文字" width="宽度" height="高度" />
```

注：

- src 用来指定当前想要引入的图片位置，这个属性值我们也称之为 路径
- title 属性用来定义图片的标题，它里面的内容会在鼠标悬停到图片上方时显示
- alt 属性定义图片的提示文字，当图片由于某些原因无法正常加载显示的时候 ，就会显示 alt 里的文字
- width 用来定义图片的宽度，只需要定义数值大小，默认单位是 px
- height 用来定义图片的高度，规则和width 一样
- 对于图片来说，如果只设置 width 或者 高度中的某一个值，那么另外的一边就会按着原图宽高比自动缩放

一、路径基本介绍

1. 定义：在代码中我们可以将路径看做是查找某个资源所 ”走过“ 的路(一段定位地址)。
2. 路径分类：

   一、绝对路径

   一般是以盘符为起点来查找某个资源，或者是一个绝对的网络资源地址，但是不推荐使用，假如以盘符为起点查找，我们不能保证用户电脑的盘符关系与我们的电脑保持一致，如果是绝对的网络地址我们则不能保证该资源永远存且不发生变化

   二、相对路径

   相对路径我们往往以当前代码所在文件为起点去查找某个资源

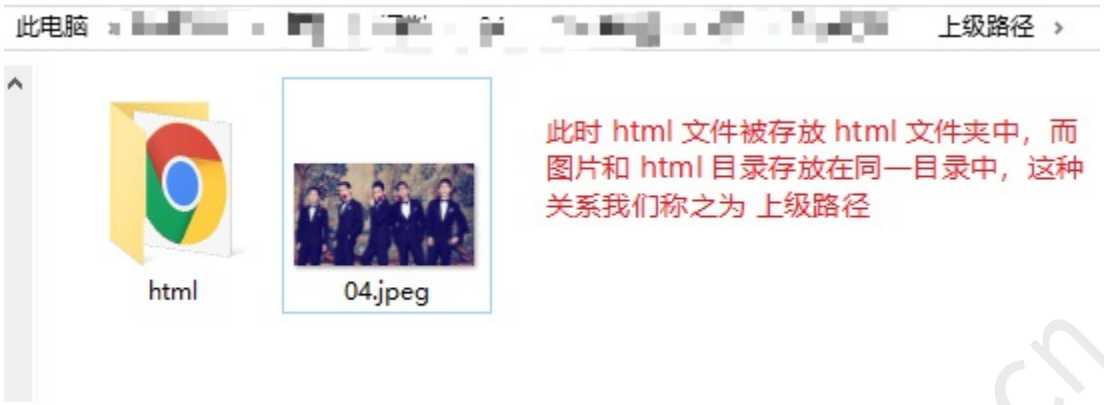
二、相对路径分类及使用

- 同级路径：HTML 文件与被查找的图片存放在同一级目录下，二者是 ”兄弟“关系，此时我们只需要在 src 中写入图片名称即可

- 下级路径：HTML 文件与图片不在同级目录，此时要查找的图片被放在了HTML 文件的下级目录中，此时需要通过 / 符号来向下进行查找。


```
<img src="img/**.jpg" />  下级路径写法，通过 / 来向下进行查找
```

- 上级路径：仍然以HTML文件做为起点，此时 图片被存放在了 HTML 文件上级目录，这种情况下我们需要使用 ../ 符号来向上进行回退查找

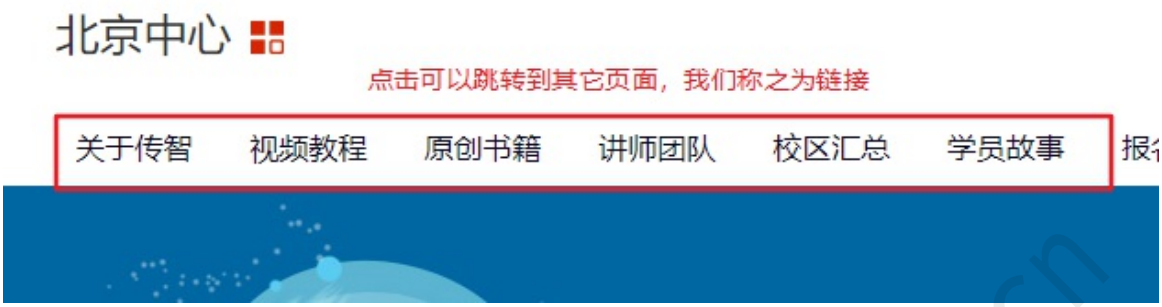



```
<img src="../**.jpg" />
```

> 注意：
>
> 1 上述的相对路径关系是以图片标签做为举例
>
> 2 图片的路径关系适用于所有存在路径的场景当中
>
> 3 图片的格式可以存在多种

一、超链接标签

1. 超链接就是我们平时在网页上见到的点击可以发生页面跳转的模块



2. 在 HTML 当中定义了专门用来实现 链接的标签，名称叫 a ，是一个双标签

```
<a href="目标页面地址">传智播客</a>
```

> 注：
>
> 1 默认情况下 a 标签中的文字内容显示会带有 下划线和颜色
>
> 2 href 属性与 img 标签的 src 属性作用类似，用来指定当前链接点击时要跳转到哪里

二、链接相关使用

1. 空链接：作用就是定义一个点击时不会发生跳转的链接，一般出现在生产阶段。因为此时还不能确定目标地址所在，具体作用可以将 href 的值设置为 # 或 javascript:；

```
<a href="#">空链接</a>
```

2. 新窗口打开页面：在点击链接跳转到新页面的时候会在一个新的窗口打开此页面，通过 target 属性来完成

```
<a href="http://itcast.cn" target="_blank">新窗口打开页面</a>
```

学习目标

1. 了解表单基本概念及表单的作用

2. 说出常见的表单数据提交方法

3. 说出常见的表单元素及不同的 **type** 类型

4. 说出**CSS**基本定义

5. 说出类名选择器使用方法

6. 说出**ID**选择器使用方法

7. 说出**CSS**特性

一、表单基本介绍

表单就是在web网页上用来收集用户数据，并且将数据提交到后台的一个 HTML 模块，在 HTML 里专门定义了一个叫 form 的标签，我们称之为表单域

```
<form action="" method="">
    表单域里的内容
</form>
```

二、表单域属性介绍

1. HTML 当中通过 form 标签来定义 表单域，它只相当于一张 "白纸"，用户在它上面填写数据(通过其它表单元素实现)。

2. 当用户填写完数据之后点击提交按钮，数据就会自动的提交到目标地址，这个地址就是写在 action 属性值里

3. method 属性值里填写的就是数据提交的方法，最常见的有 get post 二种方法

> 注：
>
> 1 action 里可以为空 或者 # 来用于数据提交测试
>
> 2 get 方法提交的数据会显示在 url 地址的后面 ，有数据大小的限制
>
> 3 post 方法提交的数据会写在 http协议请求体当中，没有数据大小限制【后期可以抓包查看】

三、表单标签基本演示

```
<form action="#" method="get">
    用户名：<input type="text" name="username" />
    <br />
    <input type="submit" value="点击提交" />
</form>
```

一、表单元素介绍

form 标签只负责定义表单模块，默认在页面上并不做任何显示，就相当于一张 "白纸"。而如果想要收集并提交用户填写的具体数据 那么就需要在这张白纸上 添加供用户输入数据的模块，此时 HTML 中又定义了很多的 表单元素，其中最常见的就是 input 标签，它是一个单标签。通过设置不同的 type 类型它可以代表不同的表单选项。

二、**input** 标签介绍

```
<input type="text" value="" name="" />
```

> 注：
>
> 1 input 是一个单标签，是 HTML 中规定好的一种表单元素，它默认存在多个标签属性
>
> 2 type 属性表示类型， 在 HTML 中很多的表单选项标签名都叫 input ， 这个时候我们就可以通过 type 属性值来完成区分，例如属性值 text 就表示文本输入框
>
> 3 value 属性来定义 input 框默认展示的文字内容(常见于可供用户手动输入的input 框有效，不须死记)
>
> 4 name 属性用来给 input 框起名字，因为很多的表单选项名都叫 input ,例如 用户名，密码，性别....等，这些表单项HTML名称都叫 input ,那么当用户将填写的数据提交到后台以后。对于后台来说，他们看到的数据都是从input 当中来的。这个时候就无法区分，哪个数据是 用户名input ， 哪个数据是 密码 input .所以为了解决这个问题我们在书写的时候就可以给不同的 input 定义不同的 name ,这样数据到达之后就会具有自己的名称。

三、常见的表单元素

```
<form action="#" method="get">

    文本输入框: <input type="text" value="我是默认文本" name="xxxx" />
    <br /><br />

    密码框：<input type="password" name="" value="123456" />
    <br /><br />

    单选框：
    <input type="radio" name="" value="111" />男
    <input type="radio" name="" value="111" />女
    <br /><br />

    复选框：
    <input type="checkbox" name="" value="1" /> 篮球
    <input type="checkbox" name="" value="1" /> 测试
    <input type="checkbox" name="" value="1" /> 看书
    <br /><br />

    下拉框：
    <select name="">
        <option value="选项1">选项1</option>
        <option value="选项2">选项2</option>
        <option value="选项3">选项3</option>
    </select>
    <br /><br />

    文本域：<textarea name="" cols="30" rows="10"></textarea>
    <br /><br />
```

```
    普通按钮：<input type="button" value="普通按钮"/>
    <br /><br />

    重置按钮：<input type="reset" value="重置"/>
    <br /><br />

    提交按钮：<input type="submit" value="点击提交" />
    <br /><br />

</form>
```

效果图

四、常见表单元素细节

- 一组单选框要达到只能选中其中一个的效果，需要给多个选项设置相同的 name 属性值

```
<input type="radio" name="gender" value="111" />男
<input type="radio" name="gender" value="111" />女
```

- 单选框默认选中某个选项可以设置 `checked=checked` 属性

```
<input type="radio" name="gender" checked="checked" />男
```

- 复选框默认选中某个选项同样可以设置 checked=checked

```
<input type="checkbox" checked="checked" />篮球
```

- 下拉框默认选中某个下拉选项可以设置 selected=selected

```
<select name="">
    <option value="选项1">选项1</option>
    <option value="选项2" selected="selected">选项2</option>
    <option value="选项3">选项3</option>
</select>
```

- 上述的 checked 与 selected 属性在设置的时候 属性值与属性名 是一样的。所以对于这类属性在设置的时候可以只写属性即可

```
例如：<input type="checkbox" checked />篮球
```

一、web 标准

web标准是用来衡量当前网页书写是否标准的一种规则，由 W3C 组织制定。它要求标准的网页写法要达到 结构、样式、行为 三者相分离

1. 结构：指的就是网页架构，即我们使用 HTML 语言搭建的网页结构
2. 样式：指的是网页具体外在展示，我们通过 CSS 来定义
3. 行为：指用户与网页之间可以进行的交互行为，通过 javascript 来实现

> 总结：如果把网页比做一套房子，结构就相当于 毛坯，用HTML 来搭建，样式相当于装修美化毛坯，使用 CSS 来完成，行为相当于让房子与住户产生智能交互，通过 javascript 来完成

二、CSS 基本介绍

CSS 是Cascading Style Sheets 的首字母缩写，我们称之为 级联样式、层叠样式、样式表 ，主要的作用就是用来修饰美化网页内容的展示效果，可以将其看做是一门语言，所以要想使用它修饰网页，就必须先学习语言中的内容

三、CSS 书写基本步骤

1. CSS 是一门语言，HTML 也是一门语言，二者本身互相独立，如果想用 CSS 修饰 HTML 那么就需要先找到书写 CSS 的位置。我们可以将CSS写在 HTML 文件中，二者可以共存。一般可以先将CSS 代码写在 HTML 网页的head 标签里,title 标签下

```
<head>
    <meta charset="UTF-8">
    <title>CSS书写位置</title>

    <style type="text/css">
        /* head里 title 下 */
    </style>

</head>
<body>
```

我们可以将 CSS 代码写在 HTML 文件中的这个位置

2. 因为 CSS 本身也是一种语言，当我们将其写在 HTML 文件中时，为了不让浏览器在解析网页内容时出现语法错误，所以 html 中定义了一个 style 标签来负责存放 CSS 代码，它是一个双标签

```
<style type="text/css">
    /* type属性值中的 text/css 就是申明当前文档类型是 CSS  */
</style>
```

3. 当我们确定书写位置，和书写环境之后就可以按着 CSS 的语法在 style 标签对之间定义相应的元素 CSS 样式

四、常见的CSS 样式

```
<style type="text/css">
    div{
        width:100px;
        height:100px;
```

```
        background-color:seagreen;
    }
</style>
```

> 注：
>
> 1 CSS 书写的基本语法规则是 ：选择器{ } 在大括号里定义具体的 CSS 样式【此处不必在意语法，后面会讲】
>
> 2 width 属性定义元素的宽度，单位是 px
>
> 3 height 属性定义元素高度，单位是 px
>
> 4 background-color 属性定义元素背景色，可以设置一个颜色值

一、CSS 使用介绍

网页是利用 HTML 来书写的，而 CSS 负责美化和修饰 HTML，所以我们可以理解为是 CSS 对 HTML 有一种控制的效果，基于此 CSS 使用的核心思想就是 **找到相应的元素，然后给它设置想要的样式**，其中找元素用的就是 CSS 选择器，设置样式就是按着 CSS 语法来书写 CSS 代码

二、CSS选择器介绍

我们习惯将 CSS 选择器分为简单选择器与复合选择器二大类，每类里面又会包含不同的具体选择器，这里首先从简单选择器说起

简单选择器

1. 标签名选择器：通过标签名称来选中元素
2. 类名选择器：通过标签的类名选中元素
3. id 名选择器：通过 标签的 id 名选中元素

三、选择器语法规则

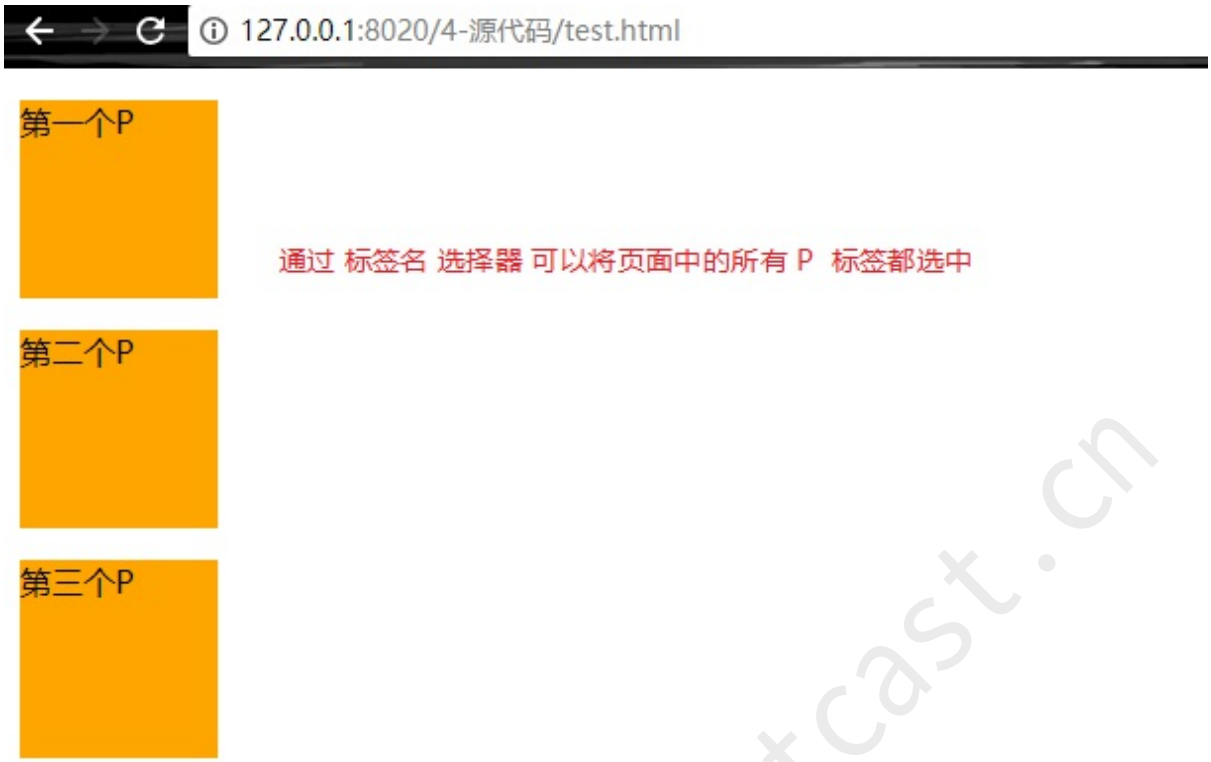
```
<style type="text/css">
    选择器{

        具体的 CSS 属性代码
    }
</style>
```

> 注：
>
> 1 CSS 代码写在 style 标签对之间【 后面会讲CSS也可以写在其它地方 】
>
> 2 CSS 代码块的规是就是通过 选择器找到要操作的元素【CSS选择器类型很多】
>
> 3 通过选择器找到具体的元素之后可以在 { } 中设置具体的 CSS 样式
>
> 4 建议大家为了代码整洁美观 在开发阶段，可以将不同的CSS 样式都书写在一行，每行的结尾用 ; 结束，且层级之间使用一个 Tab 键进行缩进【上线时可以进行代码压缩】

四、标签名选择器

标签名选择器就是通过 HTML 标签名称来选中要操作元素，使用该选择器会将网页中所有叫该名称的 HTML 元素都选中

```
p{
    width:100px;
    height:100px;
    background-color:orange;
}
```



五、类名选择器

一、类名选择器介绍

先从标签名选择器说起，当我们使用标签名选择器的时候可以将页面中所有叫该名字的标签全部都选中，但是有些时候我们只是想从这些标签中选出一部分我们需要的，所以这个时候CSS 里就定了一种类名选择器可以用来选中一部分需要设置相同样式的元素

二、类名选择器使用步骤

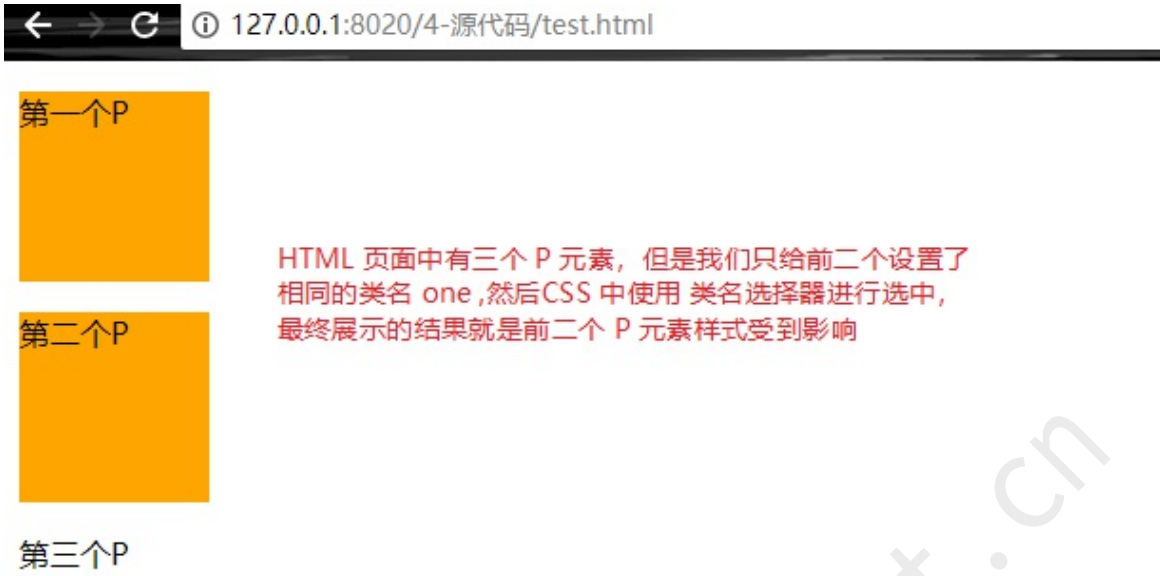
1. 在需要被同时选中的元素身上设置 class 属性，同时给它们定义相同的 class 属性值【类名】

```
<p class="one">class属性值可以自定义，我们称之为类名</p>
<p class="one">class属性值可以自定义，我们称之为类名</p>
<p>第三个P</p>
```

2. 在 HTML 中定义好具体的类名之后就可以在 CSS 中按着固定的语法来进行使用

```
<style type="text/css">
    .one{
        width:100px;
        height:100px;
        background-color:orange;
    }
</style>
```

3. 此时就可以选中多个具有相同类名的元素进行样式设置



注：

1 CSS 语法要求 class 属性值可以自定义，但需要遵循一定的规则（ 后续总结 ）

2 在 CSS 中如果想要使用 类名选择器，必须通过 `.类名{}` 的语法形式来调用

一、id 选择器

id 选择器介绍

标签名选择器会选中所有叫该名称的元素，类名选择器会选中所有具有该类名的元素，而如果只想从一堆元素当中选择其中的一个。那么上述的二种选择器就不太好实现，此时 CSS 中就定义了 ID 选择器，它只会选中具有某个具体 id 值的元素

id 选择器使用步骤

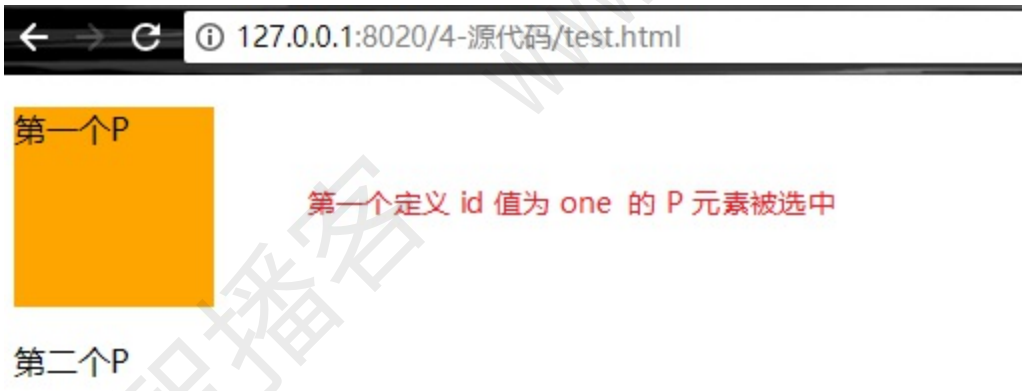
1. 在需要选中的标签身上设置 id 属性，并定义属性值【我们称之为 id 名】

```
<p id="one">第一个 P 标签</p>
<p>第二个 P 标签</p>
```

2. 在 HTML 中给要选择的元素定义好 id 名称之后就可以直接在 CSS 通过固定的语法来使用

```
<style type="text/css">
    #one{
        width:100px;
        height:100px;
        background-color:orange;
    }
</style>
```

3. 通过 `#id名称` 的固定语法选中某个元素之后就可以直接设置样式



> 注：
>
> 1 id 选择器再调用的时候必须要使用 # 号来调用
>
> 2 语法要求某个 ID 名称在当前 HTML 页面中只能出现一次。【虽然语法不报错，但不允许出现同名的 id 值】

一、简单选择器分类

1. 标签名选择器：直接通过标签名选择元素，一次性可以选中当前网页中所有叫该名称的标签。
2. 类选择器：在想要选中的元素身上定义class 属性，并设置类名，在CSS 中通过 `.类名` 的语法调用，一次性可以选中所有具有该类名的元素。
3. id 选择器：在目标元素身上设置 id 属性及属值，在 CSS 中通过 `#id名称` 语法调用，一次性只能选中一个具有该id名称的元素
4. CSS 语法允许一个元素身上同时具有类名和 ID 名，且二者的名称可以相同。

二、**class** 与 **id** 命名规则

下述的规则同时适用于 类名与ID名

- 不能以数字开头，或者纯数字做为 class 或 id 名
- 不能使用中文做为 class 或 id 名【虽然有效，但是不建议使用】
- 不以特殊字符开头，或者包含特殊字符【- _ 二者除外】
- 命名建议做到见名知意，不采用保留字(只需了解概念即可)

一、CSS 特性

所谓CSS 特性指的是 CSS 在使用过程中存在的一些默认现象或者需要遵循的一些细节，为了方便记忆我们人为的分成三点

1. **CSS** 继承性：子元素可以继承父元素的CSS 样式【可以继承不代表一定继承或者全部继承】
2. 覆盖性：同一个元素的相同 CSS 属性，如果存在多个属性值 ，那么后写的会覆盖先写的【权重相同的前提下】
3. **CSS** 优先级：同一个元素若同时被多个不同类型的选择器操作，那么会存在操作能力强弱的现象

二、继承性

一个完整的网页会有很多的 HTML 标签组合嵌套而成，所以在这个过程中肯定会存在多个元素互为父子级的现象。而继承性往往就出现在 子元素的身上

```
<div>
    <p>div下的p元素</p>
    <span>div下的span元素</span>
</div>
```

> 注：此时 div 是父元素，它的下面有二个子元素 分别是 p 和 span 。

我们给 div 设置相应的样式，然后故意不给 p 元素 及 span 元素设置样式，从而查看 p 和 span 的默认展示。从而来验证继承性的存在，同时说明继承性的特点

```
<style type="text/css">
    div{
        width:200px;
        height: 200px;
        background-color:orange;
    }
    p{
        background-color: green;
    }
    span{
        background-color: lightblue;
    }
</style>
```

具体的展示效果

> 注：
>
> 1 继承性一般出现在块元素的身上。
>
> 2 不是所有的属性都会继承(不需要记忆哪些属性可以继承，不继承的时候单独设置即可)

三、覆盖性

由于实际工作中同一个页面可能会有多个人配合完成，所以同一个元素可能会被不同的人多次设置 CSS 样式。此时在权重相同的情况下就会出现后写的覆盖先写的特性

```
<div>
    <p>div下的p元素</p>
</div>
```

此时连续的给 P 标签的同一个 CSS 属性设置样式。

```
<style type="text/css">
    div{
        width:200px;
        height: 200px;
        background-color:orange;
    }
    p{
        width:150px;
           background-color: green;
    }
    p{
        width:100px;
        height:100px;
        background-color: lightblue;
    }
</style>
```

> 注：
>
> 1 上述代码中 连续给 P 元素设置 width 和 background-color 属性，此时权重相同。后写的会覆盖先写的
>
> 2 在 p 标签第一次样式书写中没有设置 height 样式，第二次设置了 height 属性，这种不叫覆盖，覆盖的含义指的是连续给一个元素相同属性设置多次CSS 样式。

四、优先级

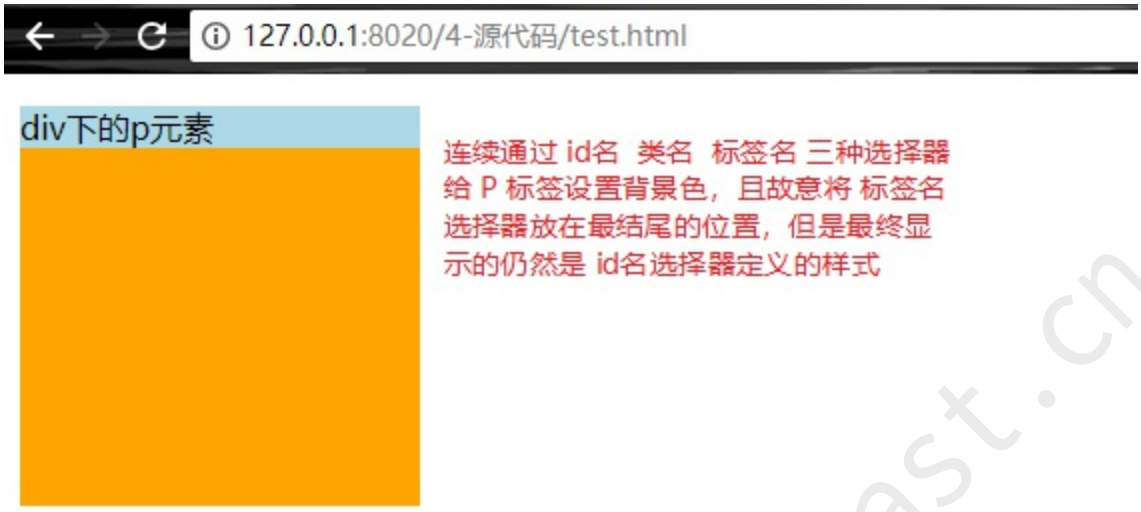
优先级指的是同一个元素被多个不同类型的选择器选中，然后分别设置 CSS 样式 时出现因为选择器不同而出现最终生效样式不同的现象

对于 `简单选择器` 来说 id 选择器的操作能力 > 类名选择器 > 标签名选择器

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>简单选择器权重</title>

        <style type="text/css">
            div{
                width:200px;
                height: 200px;
                background-color:orange;
            }
            #one{
                background-color: lightblue;
            }
            .one{
                background-color:seagreen
            }
            p{
                background-color: salmon;
            }
        </style>

    </head>
    <body>

        <div>
            <p class="one" id="one">div下的p元素</p>
        </div>
```

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

最终展示效果



学习目标

1 应用常见的复合选择器设置元素 **CSS** 样式

2 理解简单选择器 和 复合选择器的权重规律

3 说出**CSS** 文件存放位置

4 说出 **HTML5** 音频标签 及 视频标签

5 了解 **CSS3** 中的文字阴影属性

6 了解**CSS3** 中过渡属性

7 了解 **CSS** 缩放属性

一、复合选择器介绍

之前我们学习过简单选择器，那么为什么还会分成一种复合选择器呢？原因很多，例如我们的网页结构是非常见复杂的，在编写的过程中往往会将同一个类型的标签重复使用，然后在通过设置 CSS 样式来变成我们想要的展示结果。这个时候如果只通过 标签名、类名、id 名就会显示异常 "麻烦"，因为我们需要不停的去思考该如何来给这些元素起名字。而有了复合选择器之后，我们就可以在很大的程度上降低这个过程的难度。因此就出现了多种复合选择器，这里我们只讲解其中的二种：**后代选择器、并列选择器**

二、后代选择器

后代选择器指的就是以某个元素做为起点，我们可以称其为祖先元素。然后通过该元素不停的去查找它下面出现的子元素，这里的子并不仅仅只是子级，还可以是 儿子的儿子，意思是说可以进行穿透查找

后代选择器语法

```
祖先元素 后代元素1 后代元素2.....{
    具体的 CSS 样式
    中间使用空格 隔开
}
```

> 注：
>
> 1 以某个无素为起点，这个元素可以是通过任意类型的简单选择器选中
>
> 2 祖先元素与 后代元素之间需要使用 `空格` 隔开,表示向下级查找的意思。
>
> 3 被查找的元素必须存在于祖先元素的 "树状"分支上【意思要求必须是该祖先的后代，否则找不到】
>
> 4 组成后代选择器的每个位置上都可以由任意简单选择器充当

后代选择器使用

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>后代选择器</title>

        <style type="text/css">
            div{
                width:200px;
                height: 200px;
                background-color:orange;
            }
            div p{
                background-color: orange;
            }
            div .one{
                background-color: seagreen;
            }
            .box p{
                background-color: saddlebrown;
            }
        </style>

    </head>
    <body>

        <div class="box">
```

```
            <p class="one" id="one">div下的p元素</p>
        </div>

    </body>
</html>
```

展示效果

三、并列选择器

在我们书写CSS样式的时候往往会遇到 不同模块中的元素 有可能会具有相同 CSS 样式的情况，所以这个时候如果采用之前的方法就是分别将这些元素选中，然后将相同的CSS 样式分别定义二遍。这样就会有些重复，因此 在 CSS 中就定义了并列 选择器来简化这种问题

并列选择器语法

```
选择器1,选择器2...{
    相同的 CSS 样式代码
}
```

> 注：
>
> 1 选择器1 与 选择器2 可以由之前学习过的任意选择器充当，可以是简单，也可以是复合，只要能选中某个元素
>
> 2 ... 省略号表示 后面还可以连接更多的选择器
>
> 3 每个选择器之间需要通过 , (逗号) 隔开，最后一个后面不需要添加

并列选择器演示

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>后代选择器</title>

        <style type="text/css">
            div{
                width:200px;
                height: 200px;
                background-color:orange;
```

```
            }
            p,span{
                    /* 由二个标签名选择器来组合使用 */
                    background-color: seagreen;
                    display: block;
                    width: 100px;
                    height: 40px;
            }
            .one,span{

                /* 一个类名选择器 ， 一个标签名选择器给合使用 */
                background-color: blueviolet;
            }
        </style>

    </head>
    <body>

        <div class="box">
            <p class="one" id="one">div下的p元素</p>
            <span>div下的span标签</span>
        </div>

    </body>
</html>
```

演示效果图

一、选择器权重回顾

CSS 中存在不同类型的选择器来帮助我们选中想要操作的HTML 元素，而这些选择器默认情况下对于元素的控制能力是不一样的，这个控制的能力我们就称之为选择器权重(同个元素不类型选择器之间的控制能力比较)

二、简单选择器权重

`id 选择器 》 类名选择器 》 标签名选择器`

三、复合选择器权重规则描述

复合选择器可以由任意类型的简单选择器组合而成，所以复合选择器的权重比较本质还是比较多个简单选择器组合在一起之后的 "控制力"，此时我们为了方例计算和比较，就人为的给三种简单选择器分别定义了一个数值来表示它们选中元素的 能力，其中 id选择器 100 ,类名选择器10,标签选择器1。有了这三个数值之后，无论我们遇到什么样的复合选择器都可以直接进行数据的累加求和来计算最终的权重大小

复合选择器权重演示

```
.box span {}  权重为 11
#box span {}  权重为 101
#box #span {} 权重为 200
```

> 注：
>
> 1 按着上述人为定义的数据，将复合选择器中出现的权重对照数值进行相加。最后数值越大的权重越高
>
> 2 切记权重的比较指的是不同类型选择器 对同一个素控制力的表达。
>
> 3 权重的比较不应受到继承性的影响（ 详细见案例 ）

四、继承性与权重

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>后代选择器</title>

        <style type="text/css">
            .box{
                color: red; /*给P元素的父级设置文字样式，当前权重为10*/
                width:200px;
                height: 200px;
                background-color:orange;
            }
            p{
                color: blue;/*直接通过标签名选择器选中P ，权重为1*/
            }

        </style>

    </head>
    <body>

        <div class="box">
            <p class="one" id="one">div下的p元素</p>
        </div>
```

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

最终展示

一、CSS 文件存放位置

依据W3C的web标准，我们书写网页追求的就是将结构 样式 行为三者相分离，所以在定义CSS 的时候我们也会有不同的写法。基于常见的存放形式我们人为的分为三种：内嵌CSS、外链CSS、行内CSS

二、内嵌 CSS

内嵌CSS 指的就是将CSS代码写在 style 标签对中，然后再将整个 style 标签放置于 HTML 页面中，可以放在任意的位置，但是基于网页加载性能的考虑，我们一般放在HTML 页面的靠前位置，常见的就是 head标签里，title 标签下

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>后代选择器</title>

        <style type="text/css">
            /* CSS 代码块存放位置 */
        </style>

    </head>
    <body>
        网页的主体内容
    </body>
</html>
```

三、外链 CSS

这种做法比较复合 W3C 组织制的标准，具体的表现形式就是我们将 CSS 代码写在一个外部独立的 CSS 文件当中，然后在需要使用该样式的 HTML 页面中通过 link 标签直接引入

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>后代选择器</title>
        <!-- 通过 link 标签来引入 外部的CSS 文件 -->
        <link href="**.css" rel="stylesheet" type="text/css" />
    </head>
    <body>
        网页的主体内容
    </body>
</html>
```

> 注：
>
> 1 link 是一个单标签，专门用来在 HTML 当中引入一个外部的 CSS 文件
>
> 2 href 属性用来定义被引入 CSS 文件所在地址，它的值就是 ”路径“
>
> 3 rel 属性用来定义当前文件是一个样式表，强烈建议不要省略。否加有些浏览器不能正常加载样式
>
> 4 type 类型定义当前文档是 CSS ，可以省略
>
> 5 外部的 CSS 文件中不需要再书写 style 标签，直接定义需要的 CSS 样式即可

四、行内 CSS

行内 CSS 就是将样式直接写在标签的身上，这种做法常见于修改维护别人的代码，在不能快速获取别人 CSS 文件的时候通过这种方法可以方便的设置自己当前想要的样式，但不推荐使用。具体做法就是将 style 当做标签身上的一个属性，然后具体的样式就变成了它的属性值

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>后代选择器</title>
    </head>
    <body>
        <p style="color:red;background-color:orange"> 行内 CSS </p>
    </body>
</html>
```

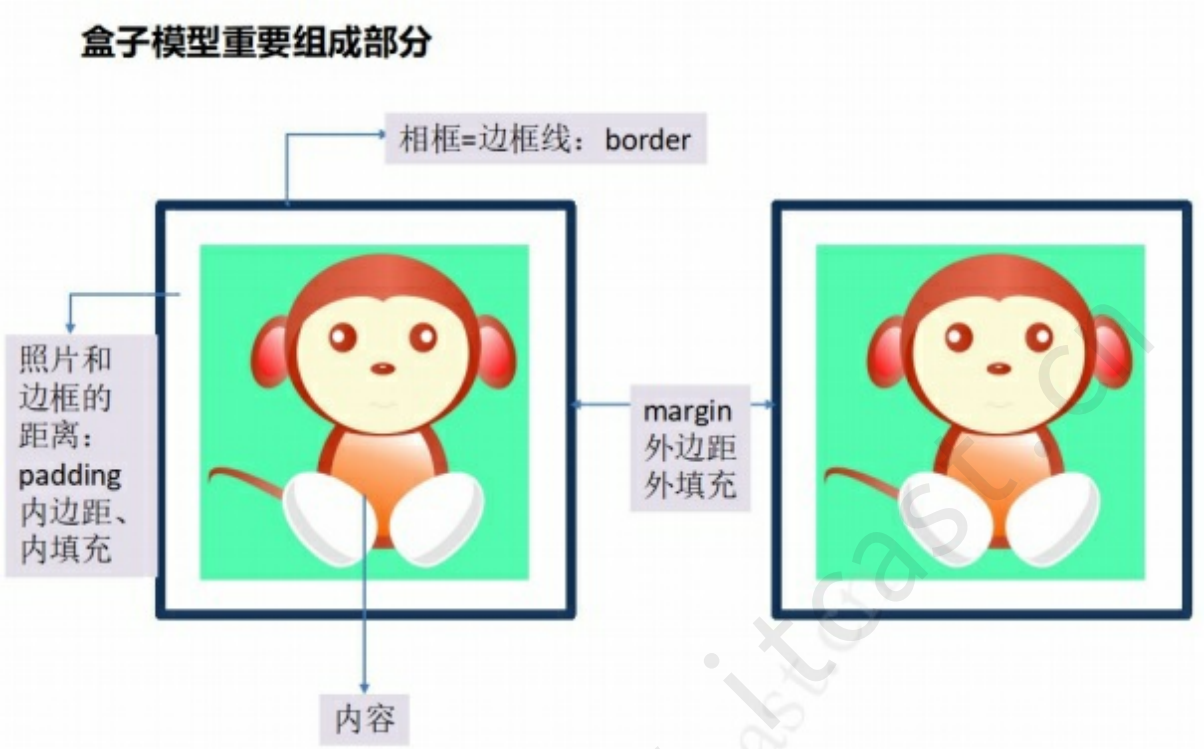
> 注：
>
> 1 style 此时就是 p 标签身上的一个属性
>
> 2 之前如何在 style 标签对中，或者外部 CSS 文件中书写CSS 样式，现在就可以原样书写。

五 、不同位置 CSS 写法的权重描述

1. 默认情况下 行内 CSS 的权重最高，
2. 外链CSS 与内嵌 CSS 默认不存在哪种写法的权重更高。依然遵循之前的比较规
3. 在 CSS 中定义了一个 !important 属性。可以将当前条CSS 属性权重提至最高。

```
<style type="text/css">
    #one{ color:red; }
    p{color:blue!important;} // 此时#one 如果和 p选中的是同一个元素，最终展示的结果会是蓝色。
</style>
```

一、 盒子模型



什么是盒子？

盒子是用来存储物品

思考一下：一个盒子是由哪些部分进行组成！

我们可以将一个盒子理解为一个快递的包裹：有内容+有填充物+纸盒子

我们如何去理解CSS中的盒子呢？ 在CSS中一个盒子的组成部分：内容(content)+内填充(padding)+边框(border)+外边距(margin)

一个盒子中的主要属性：width、height、padding、border、margin

- width：指“宽度”的意思 但是这里的宽度指的盒子里面的内容的宽度 而不是盒子的宽度
- hegiht：指“高度”的意思 但是这里的高度指的盒子里面的内容的高度 而不是盒子的高度
- border：是“外边框”的意思 指的盒子的边框
- margin：是“外边距”的意思 指的是盒子与盒子之间的间距

1.border边框

border：它是“边框”的意思。边框有三个要素：粗细、线型、颜色

语法格式： **border**：粗细 线型 颜色;

说明： 边框的颜色可以省略不写 但是如果不写的话就表示边框的颜色为黑色 其它的两个属性值不能不写 如果不写的话就会不显示边框。

```
<!DOCTYPE html>
<html>
```

```
    <head>
        <meta charset="UTF-8">
        <title>盒子模型</title>
        <style type='text/css'>
            div{
                    width:100px;height:100px;background:pink;
                    /* 最基本的边框写法 四条边一样的效果*/
                    border:1px solid red;
                    /* 单独设置每个方向的边框效果 */
                    border-top:1px solid red;
                    border-bottom:2px dashed blue;
                    border-left:5px solid yellow;
                    border-right:10px dashed green;
                }
        </style>
    </head>
    <body>
        <div>呵呵</div>
    </body>
</html>
```

边框也是有四个方法的：

- border-top:上边框
- border-right:右边框
- border-bottom:下边框
- border-left:左边框

2. padding内填充

padding是“内填充”的意思 指的是盒子中间的内容到边框的这一段距离。padding是有4个方向的 所以我们能够分别描述这4个方向的padding

方法有两种：第一种我们称之为小属性，第二种我们称之为简写属性。

1.小属性：

- padding-top：上内填充
- padding-right：右内填充
- padding-bottom：下内填充
- padding-left：左内填充

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>盒子模型</title>

        <style type="text/css">
            div{
                    width:100px;height:100px;background:pink;
                    /* 最基本的边框写法 四条边一样的效果*/
                    border:1px solid red;
                    /* 单独设置每个方向的边框效果 */
                    border-top:1px solid red;
                    border-bottom:2px dashed blue;
                    border-left:5px solid yellow;
                    border-right:10px dashed green;

                    /* 最基本的内边距、内填充：四个方向间距一致 */
                    padding: 10px;
```

```
                    /* 单独设置每个方向的语法 */
                    padding-left:10px;
                    padding-right:20px;
                    padding-top:30px;
                    padding-bottom:40px;

                    /* 如果是两个值，值1代表上下内边距；值2代表左右内边距 */
                    padding:10px 20px;
                    /* 如果是三个值，值1代表上内边距；值2代表左右内边距； 值3代表下内边距*/
                    padding:10px 20px 30px;
                    /* 如果是三个值，值1代表上内边距；值2代表右内边距； 值3代表下内边距； 值4代表左内边距 */
                    padding: 10px 20px 30px 40px;
                }
        </style>

    </head>
    <body>
        <div>呵呵</div>
    </body>
</html>
```

2.简写属性：

- padding：这个属性是有方向的 可以同时表示四个方向 这个属性的方向是有顺序的 顺序是顺时针方向 也就是：上、右、下、左
- padding:20px 表示上右下左这四个方向的内填充都为20像素
- padding:10px 20px; 表示上下为10像素 左右为20像素
- padding:10px 20px 30px;表示上为10 左右为20 下为30
- padding:10px 20px 30px 40px;表示上为10 左为20 右为30 下为40

3. margin外边距

margin它表示“外边距”的意思 它是指盒子与盒子之间的距离。margin它也是有4个方向的 所以我们也能够通过4个方向对其进行描述。

方法有两种：第一种我们称之为小属性，第二种我们称之为简写属性。

1.小属性：

- margin-top：上外边距
- margin-right：右外边距
- margin-bottom：下外边距
- margin-left：左外边距

2.简写属性：

- margin：简写属性它是有方向的 这里的方向是一个顺时针的方向 它的方向是的顺序是：上、右、下、左
- margin:10px;表示上右下左这四个方向的外边距都是10像素
- margin:10px 20px;表示上下这两个方向的外边距为10像素 左右两个方向的外边距为20像素
- margin:10px 20px 30px;表示上外边距为10像素 左右外边距为20像素 下外边距为30像素
- margin:10px 20px 30px 40p;表示上外边距为10像素 右外边距为20像素 下外边距为30像素 左外边距为40像素

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>盒子模型</title>
```

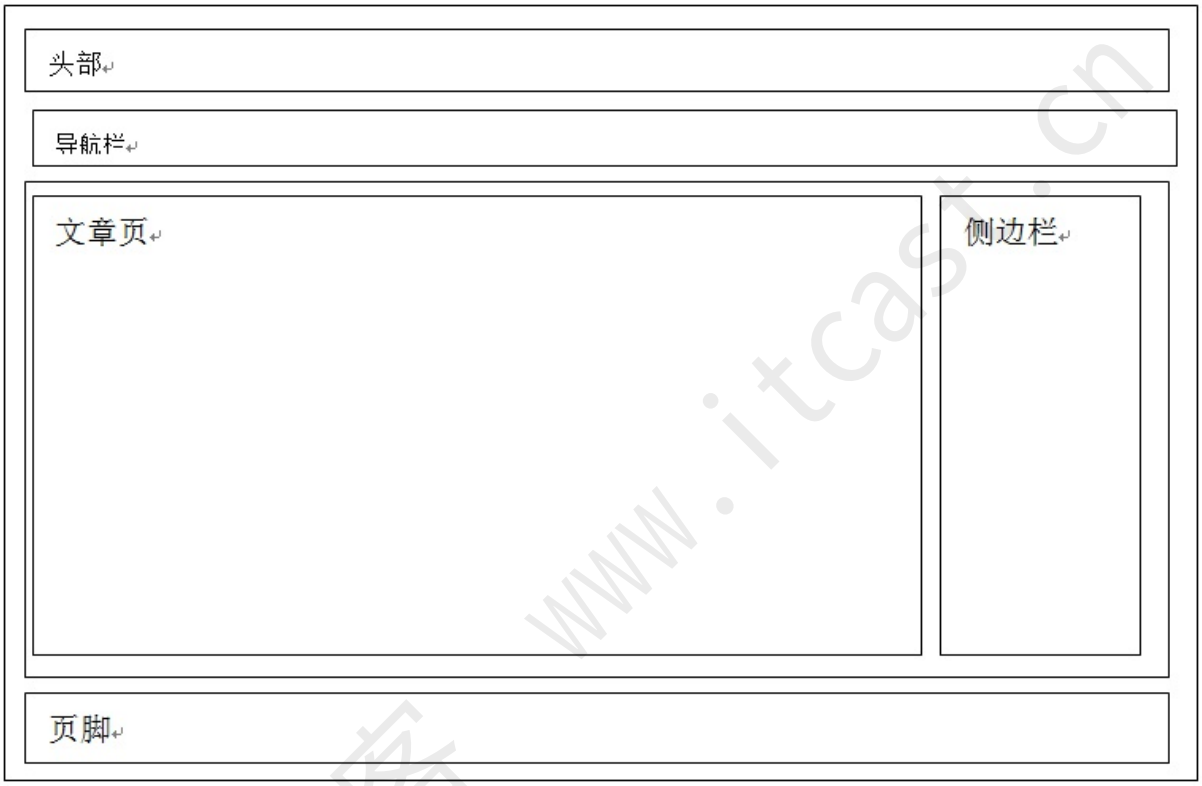
```
        <style type="text/css">
            div{
                    width:100px;height:100px;background:pink;
                    /* 最基本的边框写法 四条边一样的效果*/
                    border:1px solid red;
                    /* 单独设置每个方向的边框效果 */
                    border-top:1px solid red;
                    border-bottom:2px dashed blue;
                    border-left:5px solid yellow;
                    border-right:10px dashed green;

                    /* 最基本的内边距、内填充：四个方向间距一致 */
                    padding: 10px;

                    /* 单独设置每个方向的语法 */
                    padding-left:10px;
                    padding-right:20px;
                    padding-top:30px;
                    padding-bottom:40px;

                    /* 如果是两个值，值1代表上下内边距；值2代表左右内边距 */
                    padding:10px 20px;
                    /* 如果是三个值，值1代表上内边距；值2代表左右内边距； 值3代表下内边距*/
                    padding:10px 20px 30px;
                    /* 如果是三个值，值1代表上内边距；值2代表右内边距； 值3代表下内边距； 值4代表左内边距 */
                    padding: 10px 20px 30px 40px;

                    /* 最基本的外边距：四个方向外边距一致 */
                    margin:10px;

                    /* 单独设置每个方向的语法 */
                    margin-left:10px;
                    margin-right:20px;
                    margin-top:30px;
                    margin-bottom:40px;

                    /* 如果是两个值，值1代表上下外边距；值2代表左右外边距 */
                    margin:10px 20px;
                    /* 如果是三个值，值1代表上外边距；值2代表左右外边距； 值3代表下外边距*/
                    margin:10px 20px 30px;
                    /* 如果是三个值，值1代表上外边距；值2代表右外边距； 值3代表下外边距； 值4代表左外边距 */
                    margin: 10px 20px 30px 40px;
                }
        </style>

    </head>
    <body>
        <div>呵呵</div>
        <div>呵呵</div>
    </body>
</html>
```

一、HTML5 基本介绍

HTML5 可以看做是最新的 HTML 语言标准，在国内的市场环境中 H5 这个词汇 经常出现，我们课上的 HTML5 单指从HTML语言的版本上发展出来的最新HTML版本，而市面上的 H5 指的就是 html5+css+javascript 或者其它脚本集合。指代一整套新的开发流程和模式，我们课上要做的就是了解一些 HTML5 中新增的内容

二、结构标签

用html5来实现常用网页的布局



html5最大的变革，标签具有语义化

- div+css div是容器，是一个盒子，没有语义
- header 页面的头部
- nav 导航栏
- article 文章的内容
- section 一个区块，div相似
- aside 侧边栏
- footer 页脚

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>结构标签</title>
    </head>
    <body>
        <header>头</header>
        <nav>导航</nav>
        <aside>侧边栏</aside>
        <artical>文章块</artical>
```

```
        <footer>底部</footer>
    </body>
</html>
```

三、新增表单标签

html5表单的type新属性值

type属性值	说明
email	限定用户输入的必须是email类型
url	限定用户输入的必须是网络地址类型 包含http://
datetime-local	限定用户输入的必须是时间日期
date	限定用户输入的必须是日期类型 年月日
week	限定用户输入的必须是周类型
time	限定用户输入的必须是时间类型 小时 和分钟
month	限定用户输入的必须是月类型
number	限定用户输入的必须是数值类型
search	搜索框
color	产生一个拾色器
range	产生一个滑块

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>新增表单标签</title>
    </head>
    <body>
        <form action="xx.php" metho="get">
            <!--  type="email"就可以实现邮箱地址的验证 -->
            邮箱：<input type="email" /></br>
            <!--  type="url"就可以验证是否为网址，必须包含http://协议名称 -->
            网址：<input type="url" /></br>
            <!--  type="number"就可以实现让输入框只能输入数字 -->
            数字：<input type="number" /></br>
            <!--  type="search"可以在文本框中的最后位置出现一个X，点击后可以把内容清空 -->
            搜索：<input type="search" /></br>
            <!--  type="datetime-local"让用户可以直接通过此文本框传出的时间日期进行选择 -->
            详细日期时间：<input type="datetime-local" /></br>
            <!--  type="month"可以直接选择月份 -->
            月份：<input type="month" /></br>
            <!--  type="week"可以直接选择星期 -->
            星期：<input type="week" /></br>
            <!--  type="time"可以直接选择时间 -->
            时间：<input type="time" /></br>
            <!--  type="date"可以直接选择日期 -->
            日期：<input type="date" /></br>
            <!--  产生滑块 -->
            滑块：<input type="range" /></br>
            <input type="submit" value="提交">
        </form>
    </body>
</html>
```

三、音频标签

音频标签的作用就是在网页当中插入一段声音资源，这在之前的HTML版本中是不能直接实现的，在HTML5中新增了一个双标签名称叫 audio ，可以通过它来引入一段外部声音，语法和 引入图片类似

```
<body>

    <audio autoplay controls loop>
        对不起，您的浏览器不支持该标签
        <source src="格式1路径" />
        <source src="格式2路径" />
    </audio>

</body>
```

> 注：
>
> 1 audio 是一个双标签，用来定义一个声音资源模块
>
> 2 autoplay 表示设置该声音自动播放【有些浏览器不兼容】
>
> 3 controls 调用当前设备自带的播放控制，可以调节音量等
>
> 4 loop 默认声音只会播放一次，设置该属性之后就可以循环播放
>
> 5 其中 autoplay controls loop 其实是属性值与属性名相同，所以可以只写属性名即可
>
> 6 audio 里的文字内容会在当前浏览器不支持该标签的时候显示出来，是一种出于用户体验的降级处理
>
> 7 source 语法规定它是一个单标签，在它的身上可以设置 src 属性用来引入想要加载的声音资源路径
>
> 8 之所以存在多个路径是因为当下的互联网环境下没有哪一个浏览器可以兼容所有的声音格，也没有哪种声音格式可以支持所有的浏览器。因此出于兼容性考虑，我们一般会引入多个格式。

代码示例

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>音频标签</title>
    </head>
    <body>

        <audio autoplay controls loop>
            对不起，您的浏览器不支持该标签
            <source src="music/yinyue.mp3" />
            <source src="music/yinyue.ogg" />
        </audio>

    </body>
</html>
```

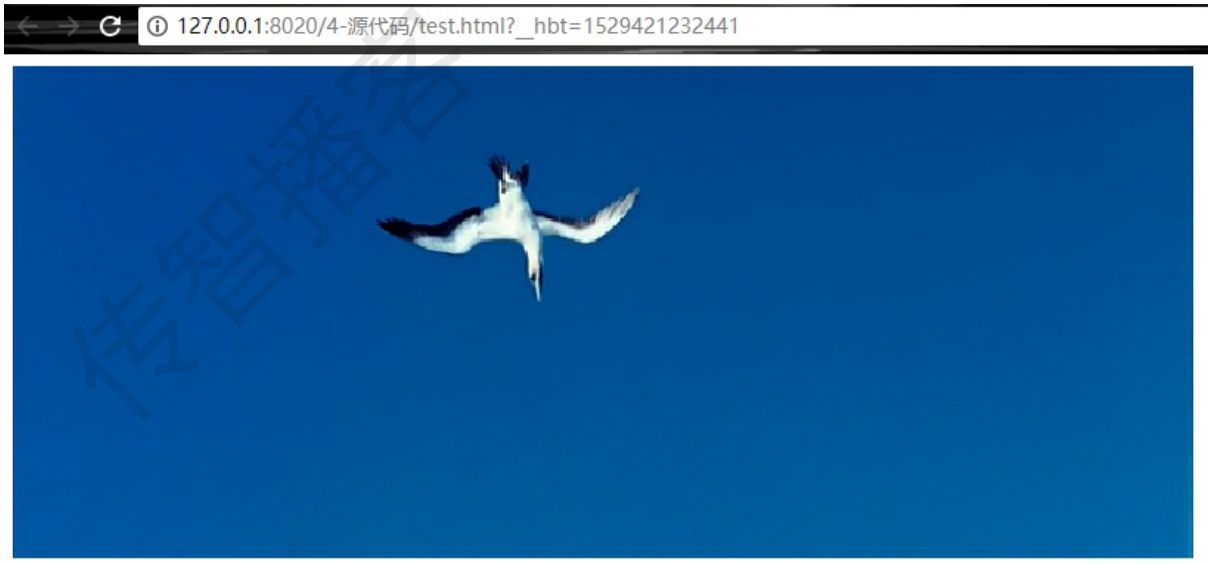
代码效果

chrome 浏览器下展示的效果，因为 HTML5 目前并没有达到所有的浏览器都兼容，所以不同的浏览器可能会出现不同的展示效果

四、视频标签

视频标签的作用就是在网页当中插入一段视频资源，在HTML5没有定义标签之前主要通过自制播放器，或者flash 的形式在网页中插入视频。HTML5当中插入视频的标签名称叫 video ，它的语法及属性和 音频几乎一致

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>视频标签</title>
    </head>
    <body>

        <video width="800" autoplay controls loop>
            <source src="video/movie.mp4" type="video/mp4"></source>
            <source src="video/movie.ogg" type="video/ogg"></source>
            <source src="video/movie.webm" type="video/webm"></source>
            当前浏览器不支持 video直接播放，点击这里下载视频： <a href="myvideo.webm">下载视频</a>
        </video>

    </body>
</html>
```



CSS3新特性

一、 圆角属性

可以使用border-radius设置标签的圆角属性

border-radius:左上 右上 右下 左下；

如果四个值相同

border-radius:数值; 代表左上 右上 右下 左下

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            div{width: 100px; height: 50px; border: 1px solid black; margin: 20px; background: gray; }
            /* border-radius  一个值代表四个方向都一致 */
            .div1{border-radius: 8px;}
            /* border-radius 四个值代表   左上 右上 右下 左下 */
            .div2{border-radius: 8px 0 0 0;}
            /* border-radius两个值 前面的代表 左上和右下  后面的值代表右上和左下
             *  */
            /*.div3{border-radius: 8px 0;}*/
            .div3{border-radius:0 8px;}
            /*.div3{border-radius: 8px 0 8px 0;}*/

            .div4{border-radius: 8px 8px 0 0;}
            .div5{border-radius: 0 0 8px 8px;}
            .div6{border-radius: 10px 20px 30px 40px;}
        </style>
    </head>
    <body>
        <div class="div1"></div>
        <div class="div2"></div>
        <div class="div3"></div>
        <div class="div4"></div>
        <div class="div5"></div>
        <div class="div6"></div>
    </body>
</html>
```

二、渐变色背景

设置背景的渐变色：linear-gradient

语法格式： **linear-gradient(180deg,red 0%,blue 40%,yellow 100%);**

说明：

第一个值：to right 从左向右地渐变

to bottom 从上向下地渐变

90deg：90度的渐变

第二--最后一个值：渐变颜色

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            div{width: 500px; height: 50px; border-radius: 0px 0px 10px 10px; /*background: red;*/
            /*linear-gradient(渐变方向也可以写角度的具体值，色值，色值....)  颜色值后面添加百分比，从0开始到100% 设置颜色渐变的区域
             * 方向值：top上、bottom下、left左、right右
             *
             *  */

            /*background: linear-gradient(to bottom,red,blue,yellow);*/
            /*background: linear-gradient(180deg,red,blue,yellow);*/
            background: linear-gradient(to bottom,red 0%,blue 90%,yellow 100%);

            }
        </style>
    </head>
    <body>
        <div></div>
    </body>
</html>
```

二、盒子阴影

设置某一个盒子区域的阴影、羽化效果

语法格式： **box-shadow: 0px 0px 30px red;**

说明：

第一个值：水平方向的阴影

第二个值：垂直方向的阴影

第三个值：阴影大小

第四个值：阴影颜色

inset：表示内阴影

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            div{width:200px; height: 100px; background: gold; margin:50px; border: 1px solid black; border-radius: 20px;}
            /* box-shadow: 水平方向  垂直方向  阴影大小  阴影颜色 */
            .div1{box-shadow: 0px 0px 30px red;}

            /* inset实现的效果就是内阴影 */
            .div2{box-shadow:0px 0px 30px red inset;}
        </style>
    </head>
    <body>
        <div class="div1"></div>
        <div class="div2"></div>
    </body>
</html>
```

三、透明背景

设置透明背景

语法格式： **background: rgba(62, 109, 245,0.1);**

说明：

第一个值：红色色号

第二个值：绿色色号

第三个值：蓝色色号

第四个值：透明度值

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            body{background: url(bg.jpg);}
            div{width: 300px; height: 300px; border-radius: 10px;
            /* background: rgba（红,绿,蓝,透明度值）;
             * 红绿蓝都是色值号设置的，可以使用浏览器调试工具来把想要的色值拿到;
             * 透明度是的值：0到1之间 ，  0带全透明   1代表全不透明   我们也可以设置0到1之间的小数来代表透明度的百分比
             *  */
              background: rgba(62, 109, 245,0.1);box-shadow: 3px 3px 8px black;}
        </style>
    </head>
    <body>
        <div></div>
    </body>
</html>
```

四、文字阴影

文字阴影是 CSS3当中新增的一个用于给文字添加阴影的属性，属性名称叫 text-shadow

属性语法

```
p{
    text-shadow: x y r color;
}
```

> 注：
>
> 1 CSS3 中的阴影属性名称叫 text-shaow
>
> 2 可以设置四个参数，每个参数之间用空格隔开
>
> 3 前二个参数 分别表示阴影在 X 轴 Y 轴的偏移量，单位 px ，可以设置负值 。这里的正 负不表示大小，表示方向，其中水平向右为正，垂直向下为正
>
> 4 第三个参数表示当前阴影的模糊程度，数值越大 阴影越模糊。
>
> 5 第四个参数表示阴影的 颜色

> 6 CSS3 语法允许给一段文字添加多重阴影，具体的做法就是将四个参数看做一组，然后多层阴影就意味着存在多组，之间用逗号隔开即可。

代码示例

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>文字阴影</title>
        <style type="text/css">
            div{
                color: #fff;
                width: 200px;
                height: 200px;
                margin:100px auto;
                text-align: center;
                background: seagreen;
                font:bold 30px/200px "微软雅黑";
                text-shadow: 0px 0px 10px red;
            }
        </style>
    </head>
    <body>

        <div>我是文字</div>

    </body>
</html>
```

效果展示

一、旋转、缩放、位移属性介绍

旋转、缩放、位移属性也是CSS3的新属性，可以控制某元素进行旋转或实现某个元素缩小或者放大、移动，一般需要配合用户行为和过渡属性同时使用，产生一种交互效果

二、缩放属性语法

```
/* transform: rotate(角度值); */
.div1{ transform: rotate(90deg)}

/* transform: scale(缩放比例)*/
.div2{  transform: scale(1.0,1.0)}

/* transform: translate(x轴位移,y轴位移); */
.div3{transform: translate(100px,100px);}
```

> 注：
>
> 1. transform 属性是 CSS3 当中专门用来定义 转换 的一个属性（只需要使用）
> 2. rotate 表示定义的是控制元素的旋转，小括号填写的是具体的旋转角度
> 3. scale 就表示当前的定义的是 缩放 转换，小括号填写的是具体的缩放系数
> 4. scale 缩放的是元素的 宽 或 高，所以它可以设置二个参数，中间使用 逗号 隔开。分别表示宽度和高度
> 5. scale 参数1.0 表示缩放为原来的 1.0 倍，这个系统可以是整数，也可以是小数或者0。但是负数没有意义。如果只写一个参数那么就表示 宽度和高度都按照此比例进行缩放
> 6. translate 表示元素的位移，小括号填写的x轴和y轴的位移

示例代码

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            body{background: url(bg.jpg);}
            div{width: 300px; height: 100px; background: rgba(204, 65, 65,0.3);
                border-radius: 10px; border:1px solid black;
                margin: 200px;
            }
            /* transform: rotate(角度值); */
            .div1{ transform: rotate(90deg);}
            /* transform: scale(缩放比例)
             *  */
            .div2{transform: scale(0.3);}

            /* transform: translate(x轴位移,y轴位移); */
            .div3{transform: translate(100px,100px);}
        </style>
    </head>
    <body>
        <div class="div1">传智播客</div>
        <div class="div2">传智播客</div>
        <div class="div3">传智播客</div>
    </body>
</html>
```

一、过渡属性介绍

过渡指的就是慢慢变化的一个过程，在网页当中某些元素会有一个默认的展示样式，当鼠标滑过它或者其它的一些用户行为产生之后，这个元素的一些样式就会发生改变。而过渡的作用就是为了让这种样式的改变有一个慢慢变化的过程，看起来类似于动画，更加的流畅，用户体验更好

二、过渡属性语法

```
p{
    transition: '属性名' 过渡时间 运动曲线 延时;
}
```

> 注：
>
> 1 transition 就是CSS3 中新增的过渡属性，通过它设置样式过渡效果，后面可以设置四个参数，每个参数之间用空格隔开
>
> 2 第一个参数表示属性名，用来定义哪些CSS属性要发生过渡，可以是具体的属性名，一般用 all 关键字来表示所有的属性都过渡
>
> 3 第二个参数表示过渡时间，单位是 s 。用来定义当前过渡需要消耗多少时间
>
> 4 第三个参数用来设置 过渡中产生的动画以什么样的规律执行，只需要了解 linear 表示匀速即可
>
> 5 第四个参数用来设置当前的过渡效果是否需要经过多久时间之后才开始产生。

示例代码

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>过渡属性</title>
        <style type="text/css">
            p{
                width: 100px;
                height: 100px;
                background: orange;
            }
            .one{
                transition: all .4s linear 0s;
            }
            .one:hover{
                width:800px;
                background-color: seagreen;
            }
        </style>
    </head>
    <body>

        <p class="one"></p>

    </body>
</html>
```

展示效果

鼠标移上之后 宽度和背景色都发生过渡效果

自定义动画

一、 自定义动画

除了CSS3的过渡、转换外，CSS3有一种自由度更大的自定义动画，开发者甚至可以制作自定义动画，利用纯CSS制作出像Flash一样的效果。在实际使用中不难发现，过渡和转换更适合做元素的交互，而自定义动画除了做交互外还能使到网页具有活力。

二、过渡属性语法

如需在CSS3中创建动画，需要学习@keyframes规则。

@keyframes规则用于创建动画。在 @keyframes 中规定某项CSS样式，就能创建由当前样式逐渐改为新样式的动画效果。

```
@keyframes 动画名称
{
    from {background: red;}
    to {background: yellow;}
}
```

定义好动画操作后，在需要完成动画效果的标签中设置animation属性：

```
    div{animation: keyframes定义好的动画名称 1s alternate infinite;}
```

> 注：
>
> 1 animation 第一个参数表示使用哪个动画
>
> 3 第二个参数动画时间
>
> 4 第三个参数alternate 表示交替执行动画
>
> 5 第四个参数infinite 表示无限执行动画

示例代码

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            div{width: 100px; height: 50px; background: red; border: 1px solid black; border-radius: 10px; animation:
 go 1s alternate infinite;}

            /* alternate 交替执行动画  infinite 无限执行动画 */

            /*
             * 先使用@keyframes 名字    来定义动画的效果
             * from中设置的开始时候的效果
             * to中设置的是结束时候的效果
             *
             * 在想要做动画的标签中设置animation属性 把刚刚设计的动画名字拿过来 写时间就可以是动画效果
             *  */

            @keyframes go{
```

```
                from{background: red; transform: translate(0px,0px);}
                to{background: blue; transform: translate(100px,0px);}
            }
        </style>
    </head>
    <body>
        <div>传智播客</div>
    </body>
</html>
```

JavaScript基础

一、 JavaScript基本介绍

js诞生于1995年，当时的主要目的是验证表单的数据是否合法 科普：Javascript的本来应该叫livescript，但是在发布前夕，想搭上媒体超热java的顺风车，临时把名字改为了 javascript。（也就是说js跟java没有关系，当时的只是想借助java的名气） 作用：控制web前端标准的前两者，结构和样式；

二、JavaScript基础语法

1. js代码写在html中的script标签中
2. alert("弹窗内容");

```
<script>
    alert("弹窗中的显示内容");
</script>
```

三、JavaScript事件

定义：在什么时候做什么事情 作用：捕获用户的行为（单击、双击、鼠标的移入移出。。。） 事件三要素：

1. 事件源：（解释就是这个事件加给谁）
2. 事件类型：（就是指的这个事件是什么时候发生的）
3. 执行的指令：固定写法 function(){ 你的命令写在这里 }

事件源“点”事件名=匿名函数（匿名方法）

> 注意：以下代码中的中括号代表的是选中第几个，0代表第一个；原生js语法会有兼容性问题（ie低版本有些不能 直接使用），需要单独写兼容性语法才能解决。

```
    <div id="id_div">我有id</div>
    <div class="class_div">我有class1</div>
    <div name="name_div">我有name1</div>
    <p>我是个p1</p>
    <div class="class_div">我有class2</div>
    <div name="name_div">我有name2</div>
    <p>我是个p2</p>
```

```
<script>
        document.getElementById('id_div').onclick = function(){
            alert('用id找到的 单击事件被捕获');
        };
        document.getElementsByClassName('class_div')[0].onclick = function(){
            alert('用class找到的 单击事件被捕获');
        };
        document.getElementsByName('name_div')[0].onclick = function(){
            alert('用name找到的 单击事件被捕获');
        };
        document.getElementsByTagName('p')[0].onclick = function(){
            alert('用标签名字找到的 单击事件被捕获');
        };
</script>
```

四、JavaScript书写位置

1. 内嵌js：在html文件中，放在script标签中

```
<script>
 alert("弹窗中的显示内容");
</script>
```

2. 外联js：可以在单独的js文件里，通过script标签中的src属性引用到页面中

```
<script src="js文件的路径">
 此处不要写代码，写什么都不会执行
</script>
```

3. 内行js：写在标签的属性里，这个属性必须是事件属性（任何标签都有事件属性），与行内css一样，不推荐使用。

```
<div onclick=”alert(‘heihei’);” >按钮</div>
```

五、常见的**js**用法

1. **js**定位元素

document代表当前的html文档

基础语法 docment.getElementById(目标标签的id属性值)

示例代码

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <div id="div1">div1</div>
        <div id="div2">div2</div>
    </body>
</html>
<script type="text/javascript">
    //document代表页面文档
    //getElementById 此种写代码的形式叫 驼峰命名法    使用id来获取页面中的标签、标记、元素

    document.getElementById('div2')
</script>
```

其他常见的单个元素查找方法：

```
document.getElementByName(目标标签的name属性值)

document.getElementByClassName(目标标签的class属性值)
```

多元素查找方法：查找到的是页面中的一组元素

```
document.getElementsByClassName(目标标签的class属性值)
```

找到一组元素后如果想访问单个元素需要通过 `[元素下标]` 来实现，下标从0开始：

```
docment.getElementsByClassName(目标标签的class属性值)[1].onclick = function(){alert();}
```

2. js定位元素后设置元素样式

查找到元素后可以对元素进行样式设置，例如设置元素的宽、高、背景色。操作方法是在查找元素的基础上通过英文的点号来选择对应的样式进行设定。

例如：设置元素的宽度

```
    document.getElementById('div2').style.width = '200px';
```

示例代码

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            #div1{width: 100px; height: 100px; background: red;}
        </style>
    </head>
    <body>
        <div id="div1"></div>
        <div id="div2"></div>
    </body>
</html>
<script type="text/javascript">
    //想要设置谁  就必须先要找到它  使用英文状态下的点来进行连接      最后的数值和前面的属性使用等号连接 等号前后
    //使用空格 只是为了看起来舒服
    document.getElementById('div2').style.width = '200px';
    document.getElementById('div2').style.height = '200px';
    document.getElementById('div2').style.background = 'pink';
</script>
```

3. 定义变量

通常可以将查找到的变量使用js中的变量记录下来，方便后续进行使用和样式设置。

js中定义变量的方式如下：

```
var 变量名
```

> 注意： 变量名应该由字母、数字、下划线、$构成，不建议使用中文， 变量名不能以数字开头 变量名不能和js中的关键字（保留字）相同

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            #div1{width: 100px; height: 100px; background: red;}
        </style>
    </head>
```

```
    <body>
        <div id="div1"></div>
        <div id="div2"></div>
    </body>
</html>
<script type="text/javascript">
    // 变量就是给数据请了一个代言人  今后只要出现变量名 就代表等号之后的这堆代码
    var $_dyr = document.getElementById('div2').style;
    // 变量起名字时候的注意事项：可以使用中文、可以使用符号_$、命名不可以使用数字开头、不允许使用js已经占用了的单词（保留字）


    $_dyr.width = '200px';
    $_dyr.height = '200px';
    $_dyr.background = 'pink';

</script>
```

4. js事件

事件：是指在什么情况下作了什么事情

事件三要素：事件源.事件类型=匿名函数

> 说明： 事件源：通常是定位到的某个页面元素
>
> 事件类型：表示在如何操作元素时会发生该事件
>
> 匿名函数：定义事件发生时的处理动作

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            div{width: 100px; height: 100px; background: pink;}
        </style>
    </head>
    <body>
        <div id="box"></div>
        <input type="button" value="变宽" id="btn_w" />
        <input type="button" value="变高" id="btn_h" />
        <input type="button" value="变色" id="btn_c" />
    </body>
</html>
<script type="text/javascript">
    // 事件：在什么情况下 做了什么事情
    // 事件三要素：    事件源.事件类型=匿名函数里面写的就是要执行的命令

    // 先找到页面中的id元素后 习惯性的就把它保存在一个变量中 方便后续的使用
    var btn_w = document.getElementById('btn_w');
    var box = document.getElementById('box');
    var btn_h = document.getElementById('btn_h');
    var btn_c = document.getElementById('btn_c');

    btn_w.onclick = function(){
        // 这里写点击后要执行那些命令
        box.style.width ='200px';
    }
</scirpt>
```

5. js其他事件

除了单击onclick时间外，还有一些常见的其他元素操作事件：

1. ondbclick：鼠标双击事件
2. onmouseover： 鼠标移入事件
3. onmouseout：鼠标移出事件

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            div{width: 200px; height: 100px; background: pink; border: 1px solid black;}
        </style>
    </head>
    <body>
        <div id="div1">单击</div>
        <div id="div2">双击</div>
        <div id="div3">鼠标移入</div>
        <div id="div4">鼠标移出</div>

        <!-- 需要js控制的时候 再添加class或id来准确的找到此元素即可  -->
        <div>我没效果</div>

    </body>
</html>
<script type="text/javascript">
    //  点击div执行弹出对话框操作
    var div1 = document.getElementById('div1');
    div1.onclick = function(){
        alert('单机事件被捕获');
    }

    var div2 = document.getElementById('div2');
    // ondblclick代表双击事件
    div2.ondblclick = function(){
        alert('双击事件被捕获');
    }

    var div3 = document.getElementById('div3');
    //onmouseover 代表鼠标移入事件
    div3.onmouseover = function(){
        alert('鼠标移入事件被捕获');
    }

    var div4 = document.getElementById('div4');
    //onmouseout 代表鼠标移出事件
    div4.onmouseout = function(){
        alert('鼠标移出事件被捕获');
    }

</script>
```

6. js代码搬家

前面我们演示的js代码都放在html标签之外，但通常来说，js代码也应该包含在html标签内。这里我们将调整js代码的位置，让js和css一样，也放置在head标签中。

需要注意的是，js代码通常是元素操作的事件，通常js代码会获取页面元素并进行操作，但由于html文件是由上向下执行的，若js代码放置在了元素定义之前，这会出现问题。此时，我们将在js代码中添加window.onload事件，其含义是 当页面元素都加载完毕后再执行js代码。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            div{width: 100px; height: 100px; background: red;}
        </style>
        <script type="text/javascript">
            // 当页面所有html内容都加载完毕后才执行此处的代码
            window.onload = function(){
                var box = document.getElementById('box');
                box.onclick = function(){
                    alert('点击成功了');
                }
            }
        </script>
    </head>
    <body>
        <div id="box"></div>
    </body>
</html>
```

7. js代码-动态添加、删除

innerHTML 属性设置或返回开始和结束标签之间的 HTML。下面我们将使用元素的innerHTML属性对div标签中的信息进行动态的设置和删除。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            div{width: 200px; height: 200px; background: pink;}
        </style>
        <script type="text/javascript">
            window.onload = function(){
                var box = document.getElementById('box');
                var btn_add = document.getElementById('btn_add');
                var btn_del = document.getElementById('btn_del');

                btn_add.onclick = function(){
                    //点击后设置div中的html显示文字
                    box.innerHTML = '点击后此处文字才会被innerHTML设置成功';
                }

                //点击删除按钮实现  让文字清空
                btn_del.onclick = function(){
                    box.innerHTML = '';
                }
                //  innerHTML作用是设置标签的中间内容
            }


        </script>
    </head>
    <body>
```

```
        <div id="box"></div>
        <input type="button" value="添加" id="btn_add" />
        <input type="button" value="删除" id="btn_del" />
    </body>
</html>
```

8. js代码-自定义函数

可以在js代码中定义命名函数，顾名思义，定义的函数有名称，定义语法如下：

```
<script type="text/javascript">
    function 函数名(){ 自己的js程序 }
</script>
```

定义命名函数的好处就是简化js代码，定义一次函数，可以通过多次执行函数，实现函数功能。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script type="text/javascript">
            window.onload = function(){
                //自定义函数作用：简化代码，实现重复性代码的简单调用
                //function 名字(){ 自己的js程序 } 函数、方法定义
                //如果想要执行（调用）这个函数中的js代码程序，必须要写    函数名（） 才能够执行函数中的命令
                //函数（方法）的定义js程序会在执行的时候优先：函数必须调用才可以执行其中的js代码
                function hi(){
                    alert('弹出');
                }
                hi();
                hi();
                hi();


            }
        </script>
    </head>
    <body>
    </body>
</html>
```

XML简介

一、 用途

XML 称作指可扩展标记语言，用来传输和存储数据。 什么是 XML?

- XML 指可扩展标记语言（EXtensible Markup Language）
- XML 是一种标记语言，很类似 HTML
- XML 的设计宗旨是传输数据，而非显示数据
- XML 标签没有被预定义。您需要自行定义标签。
- XML 被设计为具有自我描述性。
- XML 是 W3C 的推荐标准

XML 与 HTML 的主要差异

- XML 不是 HTML 的替代。
- XML 和 HTML 为不同的目的而设计：
- XML 被设计为传输和存储数据，其焦点是数据的内容。
- HTML 被设计用来显示数据，其焦点是数据的外观。
- HTML 旨在显示信息，而 XML 旨在传输信息。

二、**XML**文件书写形式

1. 由自定义标签构成
2. 标签中填写数据
3. 标签名称区分大小写

```
<ren>
    <shengao>180</shengao>
    <nianling>18</nianling>
    <tizhong>120

        <pang>一般重</pang>
    </tizhong>
</ren>
```

学习目标

1. 了解软件测试行业

2. 说出常见的测试原则

3. 了解常见的软件架构

4. 说出常见的浏览器

5. 了解域名及服务器

一、为什么需要软件测试

对于现在的用户来说我们在使用软件的时候不再仅仅只是关注于当前软件的主体功能是否可用，而且会对软件的外观，易用性，执行效率等方面都有考虑，所以这就需要我们通过大量而全面的测试操作来促使软件变得更加完美

二、为什么选择软件测试

1. 有些人喜欢创造世界所以从事开发工作，我们想让这个世界变重更加完美所以选择了软件测试
2. 当前国内的软件行业对于专业的测试人员需求量非常大

三、为什么不让开发自己做测试

1. 专业度：测试和开发属于软件行业不同的技术方向，每个方向都有自己的技术规范。让专人做专事 更加的合理
2. 思维定式：一个软件的开发需要一定的时间周期，在这个周期内对于开发人员来说绝大多数的时间都是在思考具体的软件功能该如何实现，而不会去从用户的角度来出发，思考何去使用这个功能
3. 测试力度：相对于开发人员来讲，软件就相当于他们自己的孩子，所以“下手”的时候肯定不会那么重

> 注：
>
> 不让开发自己做测试不意味着开发人员不能做测试，当前行业里有很多测试人员之前就是开发

一、软件测试定义

所谓的软件测试指的就是通过**手工或者工具**对被测对象进行测试操作，从而验证实际结果与预期之间是否存在差异

二、软件测试的作用和目的

1. 测试工作可以发现并修复软件中存在的缺陷，从而提高用户对软件的使用信心
2. 测试操作可以记录软件使用过程中产生的一些数据，从而为决策者提供依据
3. 测试操作可以降低同类型软件开发的风险
4. 总结：测试工作的目的就是通过尽可能少的人力 财才 物力来查找并解决软件中存在的缺陷从而降低商业风险等

三、测试原则

1. 测试证明软件存在缺陷：我们的测试工作只能证明当前软件是有缺陷而不能证明它没有缺陷
2. 不能执行穷尽测试：具体的测试操作不可能将所有的情况都一 一逻列出来，所以测试工作肯定有终止的时候
3. 测试应当尽早介入：一般不要在开发完成之后才执行测试，这样不利于缺陷的尽早发现
4. 缺陷存在群集现象：对于一款软件来说核心的功能只占20%，所以在测试的时候我们会花更多的时间去专门测试这些功能，因此它里面缺陷暴露的可能就会更大一些，我们就称之为缺陷群集现象。
5. 某些测试操作依赖于特定的测试环境
6. 杀虫剂现象：不要过多使用同一条测试案例来对软件进行问题查找 ，因为软件会产生“抗性”
7. 不存在缺陷的谬论：任何的软件不可能是完美的

一、常见软件架构

架构基本介绍

架构可以理解为是用来指导软件产品成型的一种思想，当前软件行业最常见的二种架构分别是B/S[浏览器---服务器模型] 和 C/S[客户端----服务器模型] 架构，B指的就是 browser ,C指的就是client, S指的就是server

bs 与 **cs** 架构比较

1. 标准：对于 BS架构来说无论是浏览器还是服务器都有现成的软件供我们去使用，而CS架构中的客户端一般都由开发者自定义完成开发，所以相对来说 BS开发要标准一些。
2. 效率：因为BS架构当中所有的数据处理操作都发生服务器端而CS 的客户端是可以来分担一些服务器数据处理工作的，因此相对来说 CS 的处理效率会高一些
3. 升级：BS架构只需要将服务器进行更新那么前台页面会自动的刷新，而CS架构如果想要升级就必须将二端都重新制作，下载安装后才可以使用。
4. 安全性：相对于CS架构来说 BS 的安全性会低一些。
5. 开发成本：因为浏览器不需要我们开发，所以我们认为 CS的开发成本相对较高

一、浏览器定义

浏览器本身就是一款软件，我们直接将它安装在操作系统上去使用，一般用于浏览网页。目前来说市面上存在五大浏览器生产厂商，分别是： IE 、firefox、chrome、safari、opera。而对于浏览器这款软件来说最核心的技术就是浏览器内核

二、浏览器内核【只需了解】

1. Trident: IE 浏览器1995年推出的一款内核
2. Gecko:目前火狐浏览器使用的内核
3. blink: 由KHTML内核心发展而来，chrome浏览器在使用
4. webkit: 由KHTML内核发展而来，safari 浏览器在使用
5. presto: 这个内核当前已经停止更新，用于oprea(欧鹏浏览器) ,现在这款浏览器已经向chrome 看齐

三、常见的图片类型

1. jpg :颜色信息比较丰富的一种图片格式
2. .png:可以支持透明的一种图片格式
3. .gif ：支持动图，占用体积小
4. .psd: 分层的图片【 常见于PSD设计稿 】

一、域名基本介绍

域名就是为了方便用户去记忆而自己设计的一个名字，一般需要花钱购买。它的组成一般分为三个部分

1. 一级域名：一级域名就是最后一个点号后面的内容 .com .cn .net
2. 二级域名：二级域名就是一级左边，baidu ，一般都是需求方自己设计，一级域名和二级域名连在一起应当是全世界唯一(yyshi.cn yyshi.com)
3. 三级域名：三级域名一般都是用户自定义，最常见的就是www

> 注：域名和服务器一样都需要花钱购买，国内常见的服务商有：万网 新网 西部数据 美橙互联.....

二、服务器与 **URL**

1. 服务器：我们就认为是一台电脑，它的上面可以安装相应的服务器软件，来为我们的用户提供服务操作 。
2. URL：所谓的URL我们就认为是用户写在浏览器地址栏里的一长串，它由协议 + 域名 +端口号 + 路径+ 具体的文件名称组成

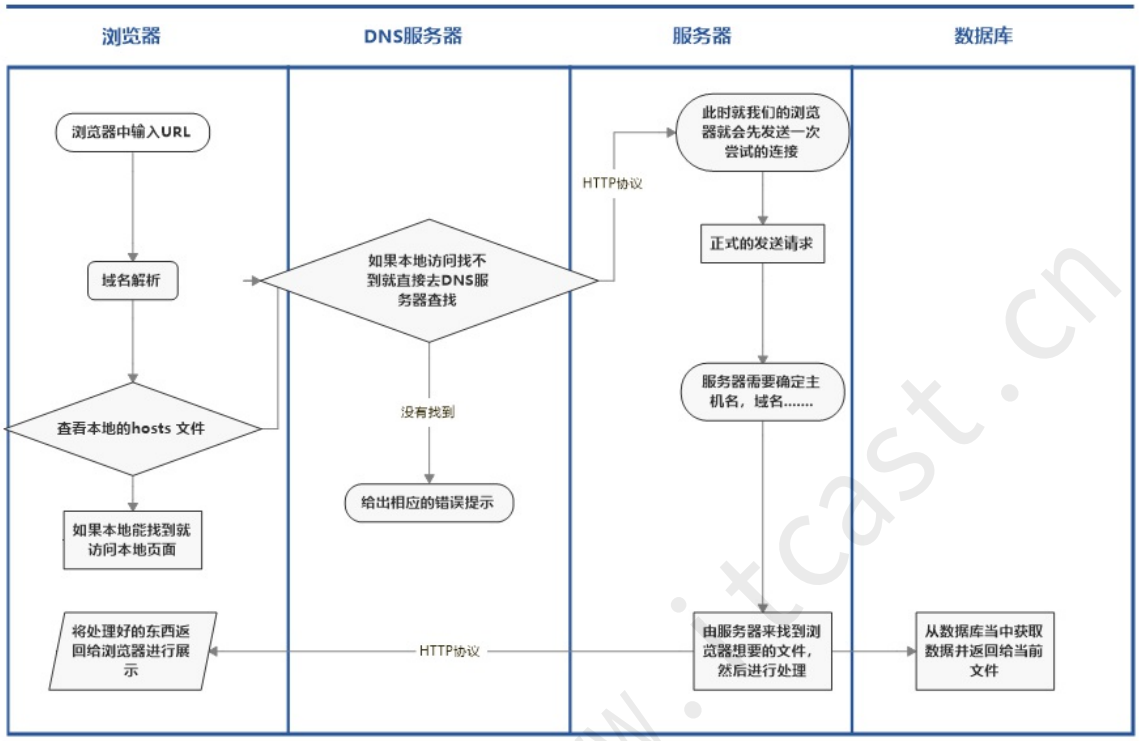
三、网站的访问过程【只需了解】

1. 在浏览器中输入 URL
2. 这个URL需要通过解析 去找到相应的IP
3. 查找IP 首先会从本地的 hosts 文件开始，如果找不到则去 DNS服务器查找
4. 如果DNS帮我们找到了目标的IP ，我们先发送一个测试的请求，通过之后再发送正式请求
5. 服务器接收到正式的请求之后，它还需要进行验证。如果验证通过，则去帮我们找到当次请求想要的文件
6. 服务器开始处理我们的想要的文件，在这个过程中有可能会用到数据库中的数据
7. 1．当服务器将文件处理毕之后再通过 http 协议还给浏览器，【此时浏览器就用自己的渲染引擎来进行渲染展示】

四、网站访问流程图

提示：

网站的访问过程



一、网络编程基本概念

客户端（Client）：移动应用（IOS、Android、Web等应用）

服务器（Server)：为客户端提供服务、提供数据、提供资源等机器

请求（Request）：客户端向服务器索取数据的一种行为

响应（Response）: 服务器对客户端对请求做出的反应，一般指返回数据给客户端

二、HTTP协议

HTTP协议概念：

协议：计算机通信网络中两台计算机之前进行通信所必须共同遵守的规则或规定。

HTTP协议：超文本传输协议，是一种规定了浏览器和服务器之间通信的规则

URL（统一资源定位符）

概念：互联网上资源的地址、位置。每一个资源都有一个唯一的URL。

格式：协议://主机地址/路径

三、HTTP协议之请求内容

HTTP请求组成：请求行、请求头、请求空行和请求数据

请求行

```
请求行：请求方式 主机   协议
GET http://127.0.0.1:8000/ HTTP/1.1
```

请求头

```
Host: 127.0.0.1:8000

保持较短时间连接，反复和服务器通讯，给服务器看的
Connection: keep-alive

客户端信息 给服务器看的
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.3
6 (KHTML, like Gecko) Chrome/62.0.3202.62 Safari/537.36

客户端能够显示的信息  给服务器看的
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/ w
ebp,image/apng,*/*;q=0.8

压缩的方法
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9

## Cookie
Cookie: name1="2|1:0|10:1508218301|5:name1|8:5byg5LiJ|c9d2e2a9c7618a9f
7d9577a0d08465781bc8e88c86cc5652457e0328dfb3d937"; csrftoken=sguSgAQGDpW
lhm7wE7N0yaP8t2vSHNKFfmhZxPPkLIQAG72EMvsMa4Ec055j04jB
```

请求空行

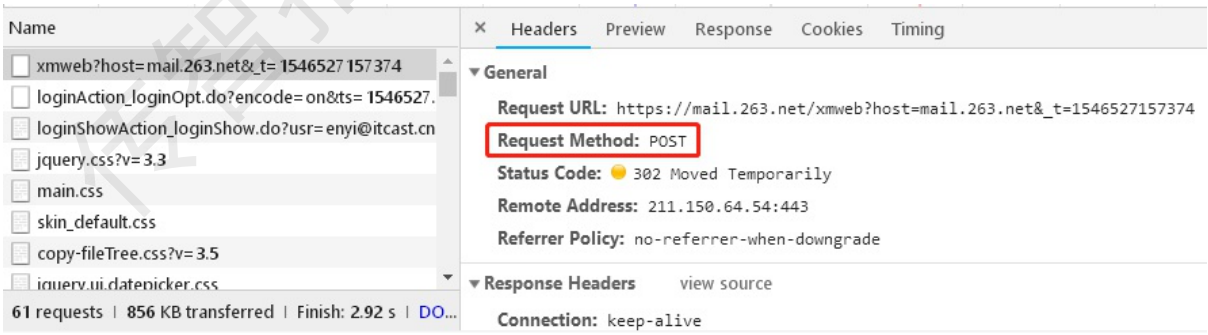
请求体 请求方法为get，请求体没有数据

请求方法为post，请求体有数据

四、GET、POST请求

GET 提交的数据显示在地址栏，不安全；提交的数据量有限制；不重要的数据使用GET

POST 隐式提交数据，更安全；没有数据量大小的限制；重要数据使用POST



五、HTTP协议之响应内容

HTTP响应组成：

响应行、响应头、响应体 响应行

```
响应行:协议 状态码 状态描述
HTTP/1.0 200 OK
```

响应头

```
Date: Fri, 27 Oct 2017 06:48:23 GMT

服务器版本
Server: WSGIServer/0.1 Python/2.7.13

网页中能不能显示iframe:SAMEORIGIN(只能显示自己域名下的iframe)
X-Frame-Options: SAMEORIGIN

文本的类型
Content-Type: text/html; charset=utf-8

响应体长度
Content-Length: 12

响应空行: 真的只是空行呦！(●ˇ∀ˇ●)
```

响应体

```
响应体
我是百度    (自定义网页中的内容)
```

六、**HTTP**协议之常见响应状态码

状态码有三位数字组成，第一位数字定义了响应类型，有5种可能取值。

```
1xx: 指示信息--表示请求已接收，继续处理。
2xx: 成功--表示请求已被成功接收、理解、接受。
3xx: 重定向--要完成请求必须进行更进一步的操作。
4xx: 客户端错误--请求有语法错误或请求无法实现。
5xx: 服务器端错误--服务器未能实现合法的请求。
```

常见状态码

状态代码	状态描述	说明
200	OK	客户端请求成功
400	Bad Request	客户端请求有语法错误，不能被服务器所理解
401	Unauthorized	请求未经授权，这个状态码必须和WWW-Athenticate包头域一起使用
403	Forbidden	服务器收到请求，但拒绝提供服务
404	Not Found	请求资源不存在，url错误
500	Internal Server Error	服务器发生不可预期的错误
503	Server Unavailable	服务器当前不能处理客户端的请求，一段时间后可能回复正常

七、API接口

接口概念：应用程序编程接口（API）：以HTTP协议形式提供，定义了输入、输出、功能描述的服务。

八、接口和客户端功能测试的关系

客户端通过调用服务端提供的接口来获取数据

客户端功能测试过程中需要和接口交互的场景

客户端测试过程发现bug，需要排查是客户端代码问题还是服务端代码问题

客户端测试过程中需要借助接口调用造一些返回数据，辅助客户端功能测试

跳过客户端代码的验证限制直接访问服务端

仿真弱网环境，进行弱网测试