# CSS

一、什么是CSS？

Cascading Style Sheets

层叠式级联样式表

这里CSS又是一种新的语言类型

Java 高级编程语言

HTML 标记语言

CSS 样式表语言

二、CSS入门

应该如何书写CSS代码？

一般来说CSS都是用来修饰HTML页面的，所以总是和HTML关联

在HTML页面上有三种方式写入CSS样式

1、内部样式表



在html的head标签中，加入style标签，css代码可以直接写在style标签里面

<style type=”text/css”>

样式表有多种表现形式，一定要设置当前使用的是文本类型CSS样式，这样html才能正确的读取样式的设置。

span{

color: red;

}

选择器 {

属性名 : 属性值;

属性名 : 属性值; ……

}

tips：内部样式表的影响范围是整个页面，只要页面中标签符合选择器的要求就会受到影响

2、行内样式表



在标签上普遍有style这个属性，这个属性的属性值就是行内样式表

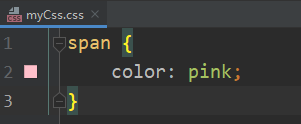
由于这个样式表直接写在一个标签的内部，受影响的范围只在这个标签上

tips：行内样式表的影响范围只有当前标签自己。

3、外部样式表

将CSS代码独立生成一个文件，这个文件的后缀就是.css，然后在需要用到这个样式的html页面上引用.css文件实现样式的引用。

1、先生成一个css的文件，写入需要的样式



2、在要引用的页面上添加link标签，按照图示写入属性



href：要引用的文件的位置，当前使用了相对路径

type：等价于<style>标签中的type，声明引用文件的类型

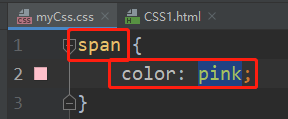
rel: 声明引用文件的用途，这里是当做样式表引入

tips：上面的三个属性必须设置，缺一不可

什么是层叠呢？

下面演示一个效果

外部样式表调颜色



内部样式表调大小



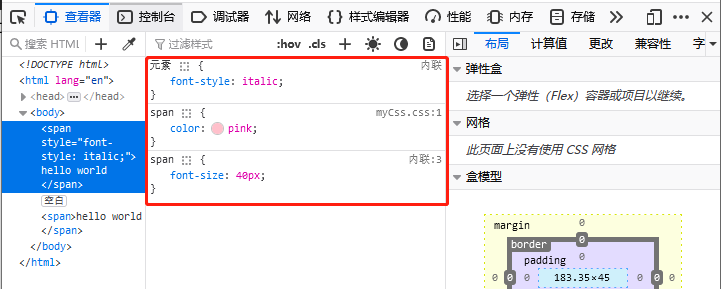
行内样式表调字形



这三个不同来源互不冲突样式最后同时作用于一个标签上，实现了



这样一个效果，三方都成立了，效果叠加了



在F12中可以发现，三方都出现在了样式表中，也就是三方叠加出最后的效果。

层叠可以理解为是多个样式表组合起来实现一个效果，这里就要提另外一个问题，样式冲突。

如果三种写法指向了同一个标签的同一个属性，最后它听谁的？

优先级

**行内 > 外部 = 内部**

外部和内部优先级是相同的，谁后出现谁覆盖其他的。

tips：在讨论优先级的时候有一个普遍的规律

范围越大，优先级越低

范围越小，优先级越高

在当前这个模型中，行内的样式表只能影响一个标签，内部和外部样式表都是影响一整个页面。所以范围最小的行内，优先级是最高的，范围相同的内部和外部优先级一样的。

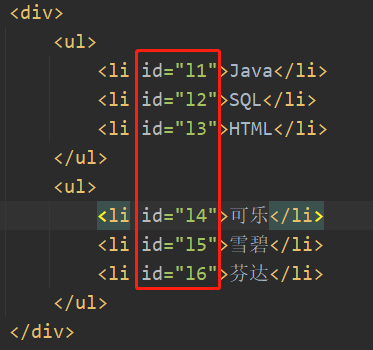
以内部样式表为基础学习CSS的写法

在刚才的开发中发现CSS代码分为两部分1、选择器 2、样式定义

一、选择器

基础选择器

id选择器

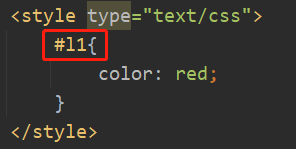


在标签上基本上都有共有属性id，这个选择器就是根据id值进行选择

#id{

}

在id选择器中，需要在开头写上#表示根据id值进行选择



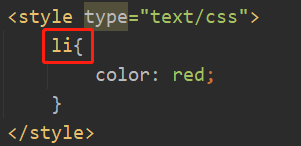
tips：id选择器，由于id在定义的时候不允许出现重复，所以选择器最多选中1个标签，是范围最小的选择器

tips：其实也未必，id写成重复的值在idea中是会提示错误的，但是放到浏览器中执行是可以执行的，也会按照复数个id进行选择

上面的这种情况不是有效的用法，属于轻量级的bug，应该要避免而不是利用。

标签选择器

在选择器部分直接写标签名即可



根据给出的标签名选择当前页面中所有的同类型标签

tips：标签选择器是范围最大的选择器，当前页面所有同类的标签都会被选中

类选择器（最常用的选择器）

id范围太小，每次只能修改一个标签

标签范围太大，总是全部一起修改

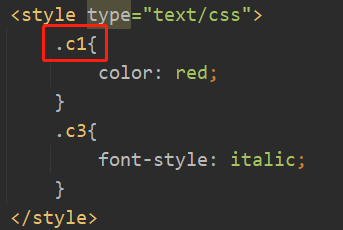
这两个范围都不合适，希望的是受影响的范围可以完全由程序员自定义。

类选择器使用class这个属性的值进行选择，class不是一个关键属性，它也不会直接影响页面的显示，class不存在唯一的定义，多个标签可以使用同一个class属性。一个标签上还可以定义多个class。

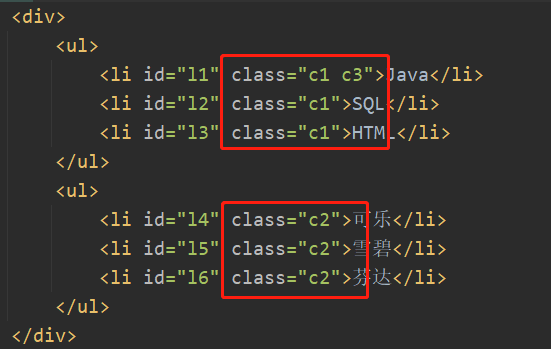
类选择器的写法是

.类名{

}



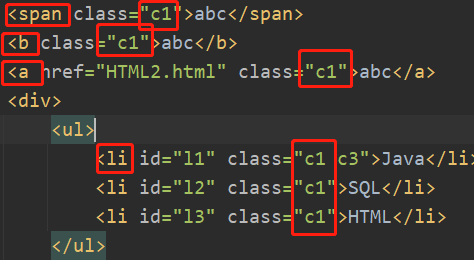
在标签上可以对应的class属性的值，让选择器选中



请关注第一个li标签，他的class属性中写入的是”c1 c3”这就表示这个标签有两个类型，它同时是c1和c3两个类的

在class属性中可以用空格隔开多个类型

tips：类选择器还有一个优点，他可以跨标签类型实现效果



可以通过将不同的类型的标签设置相同的class属性值，让他们有相似的样式

复杂选择器

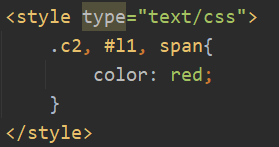
复杂选择器中没有新的选择器要学习，是对简单选择器的组合使用

分组选择器

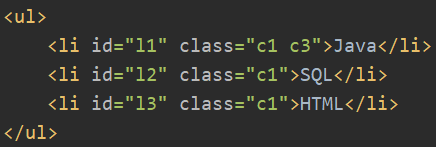
sel1 , sel2 , sel3…{

}

使用**逗号**隔开多个简单选择器，最终会作用在所有的简单选择器结果的并集上。



图中的例子就是三个简单选择器复合使用，分别用了类选择器、id选择器、标签选择器，最终受到影响的是三个选择器结果的并集。



标签之间的嵌套和平行关系在这里被理解为了家庭关系

ul和li

ul是外侧标签，li是它里面的子标签

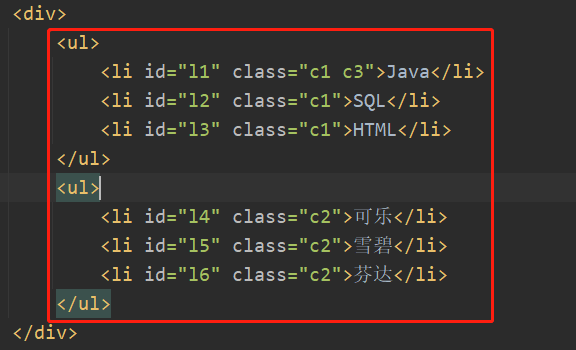
认为ul是父标签，li是子标签

假如li里面还有子标签他和ul是什么关系？爷孙关系如果还有呢？总计为后代

把并列出现的标签称为兄弟标签，在图中的例子里 l1 l2 l3 这三个li标签就是兄弟标签，兄弟的定义只在同一个父标签中生效，跨父标签就不称为兄弟标签了。

后代选择器

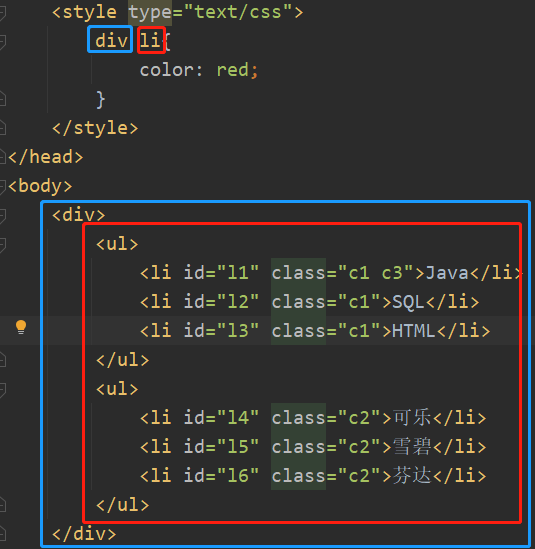
一个标签的所有内部标签都是他的后代（不管嵌套多少层，只要在它内部都算）



sel1 sel2{

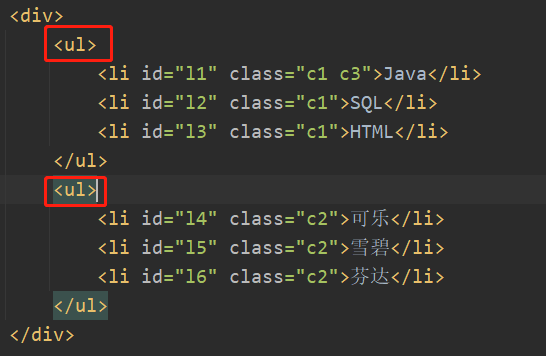
}

用空格隔开两个选择器，第一个选择器选中的是父标签，第二个选择器在这个父标签的后代中选择



子选择器

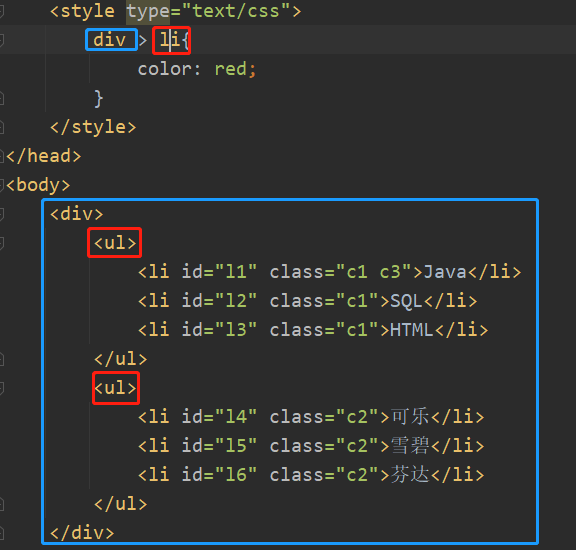
一个标签的所有直接子标签（必须直接存在于它的内部，不包含嵌套关系）



sel1 > sel2{

}

先根据sel1找到所有的父标签，在父标签的子标签中选择符合sel2的结果

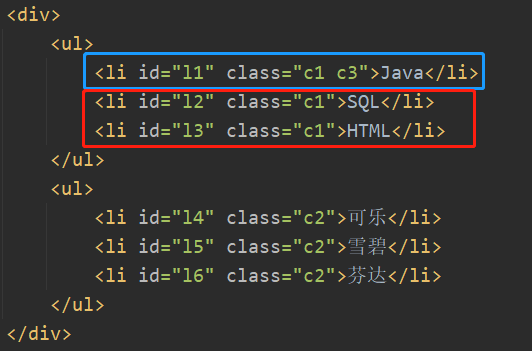


这种写法是一个都不会生效的，在div中直接子标签只有ul，li这个标签选择器选不到合适的内容，页面上不会有效果

所有兄弟选择器

读代码的时候总是从上往下，从左往右读代码

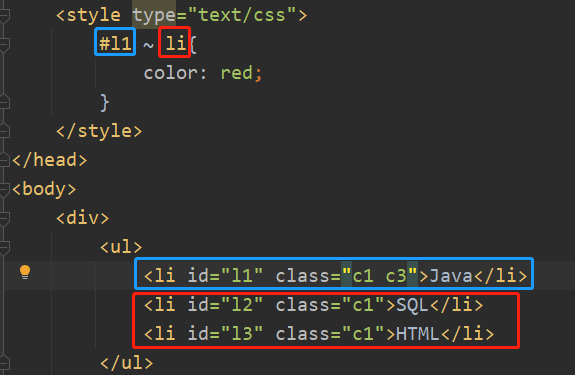
在获取兄弟标签的时候只有在自己下方的标签，且平级的才是自己的兄弟标签，所有兄弟，在自己以下的所有平级标签



sel1 ~ sel2{

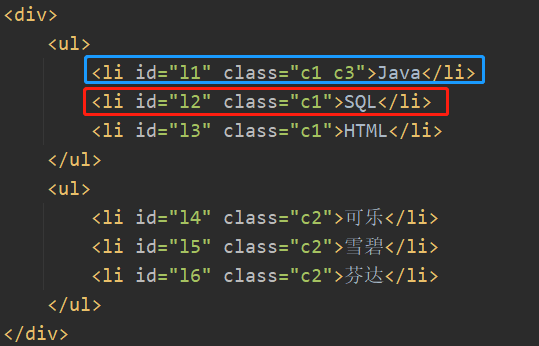
}

先根据sel1找到兄标签，在所有弟标签中找符合sel2的结果



下一个兄弟选择器

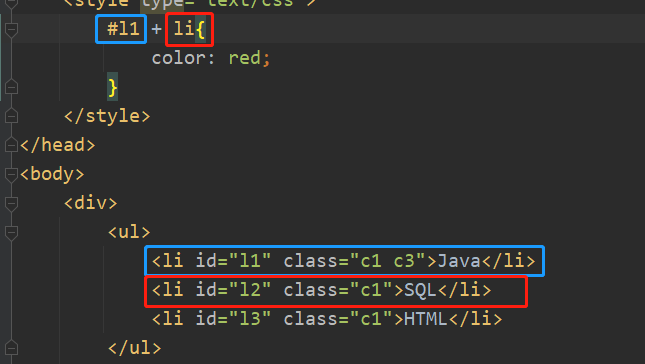
在所有兄弟的基础上只找与自己紧紧相邻的下一个兄弟标签。



sel1 + sel2{

}

先根据sel1找到兄标签，在它的下一个兄弟标签中找符合sel2的标签



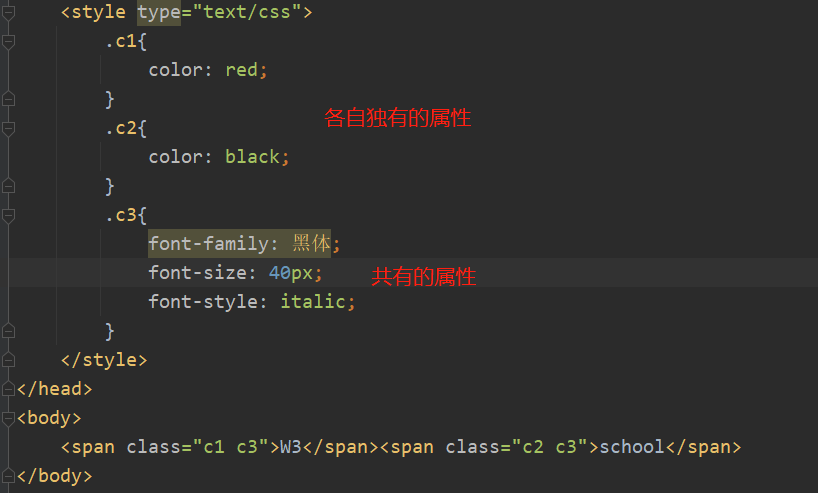
家庭作业

1、请大家使用已经学过的知识尽量仿写一个页面



一个单词两种颜色，整个斜体，字形黑体，文字放大

要求只能用内部样式表实现



2、尝试完成一个table，在table中写入t\_user表中的所有的数据（列+行）

然后实现隔行换色，不考虑列名那一行，从数据第一行开始，奇数行字是黑色的，偶数行字是红色的

3、预习CSS中的常用的样式

4、预习JS相关内容。

昨天的课上已经介绍完了所有的选择器

1、基础选择器：#id选择器 .类选择器 标签选择器

2、复杂选择器：,分组选择器 后代选择器 >子选择器 ~所有兄弟选择器 +下一个兄弟选择器

接下来的内容就是介绍常用的样式

1、字体样式

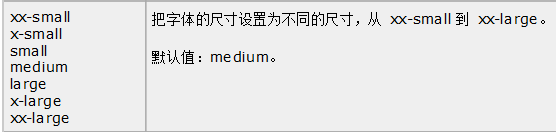
在字体样式中主要修改的文字的样式

font-size: 文字大小

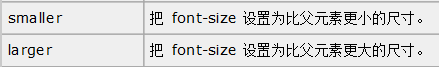
tips：页面上文字的默认大小是16px;

tips：在设置文字大小的时候，单位不一定是px

可以不设置具体的值，使用预设的档位设置



这里还有一个比较值



基于当前字体的大小，变得更大更小

这里就存在一个问题，如何表达原来的这个大小

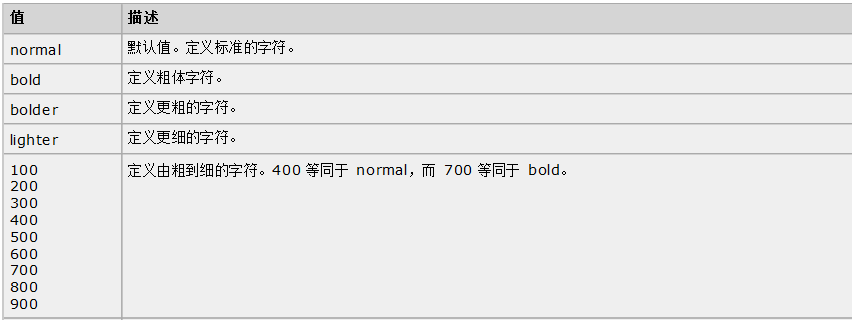
需要引入一个新的单位 em

1em就表示当前字体的大小，2em当前字体大小的两倍

tips：em是可以接收浮点数值的，1.5em也是可接受的数据

font-weight：文字的粗细

tips：为什么是weight在英文语境下文字的粗细的单位是磅（重量单位）。



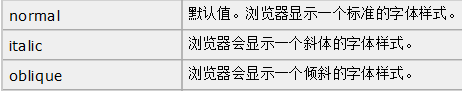
首先是两个固定的档位 normal 正常粗细（默认值）、bold加粗

是两个比较值 bolder在当前粗细下再次加粗 lighter在当前粗细下再变细

最后是9个数字档位 100~900，需要注意虽然范围是100~900，但没有细分这么多个数字，只有整百位有效，只有9个档位

tips：之前反复讲过文字就是图片，所谓的加粗和正常就是一个文字的两张图片，一个图片是正常粗细的，一个图片是加粗的。如果分为9个档位就意味着你的电脑中针对一个字符要保存9张图片。一般的操作系统都不会这么做，没有意义。大部分的操作系统只支持两个档位：正常100~500和加粗600~900。这里的细分其实没有意义，在后续的编程中只要使用normal和bold就可以了

font-style: 文字的样式（斜体）



这里一般只使用前两个属性值，normal正常字体 italic 斜体

斜体也需要独立的存储斜的字的图片，一般支持叠加加粗

tips：合理推测以下，一个文字在操作系统中至少是4张图片 正常 斜体 加粗 斜体并且加粗

font-family: 文字的字形

这里的值一般是直接写希望使用的字形的名称



在word或者WPS等文字编辑软件中，可以直接看到当前操作系统支持的字形，在网页上也可以引用一些在线字形。

color：文字的颜色

和之前学过的font的color属性一样，CSS中的color也可以支持多种参数的传入

颜色名 一般是颜色的英文名称

#rrggbb 使用两位十六进制的数字描述一个颜色的强度

#rgb 使用一位十六进制数字描述一个颜色的强度（精度较差,一般描述纯色）

rgb(r, g, b); 使用三个十进制的数字描述颜色（0~255）

rgba(r, g, b, a) 在上面函数的基础上添加了alpha透明度，可以设置一个半透明的颜色（0全透明 之间半透明 1不透明）

这里对于颜色的传值在整个CSS中是通用的，后续的背景色、边框色等都可以使用上面的五种方式。

tips：在API中可以查看CSS中预设的颜色名

2、段落

不再针对文字的细节进行设置，对文字的布局格式进行设置

word-spacing 单词间距，根据空格区分单词，实际上调整的是单词之间的空格的宽度

letter-spacing 字母间距，根据每个字母控制间距，字和字之间都会应用间距

text-indent首行缩进

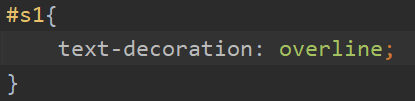
一般写成 text-indent: 2em; 表示每行缩进两个字符的大小

tips：em会按照当前语境进行设置，一般来说网页都是中文语境的，所以缩进是按照中文的两个字符的大小缩进的（一般中文会占两个英文字符的空间）

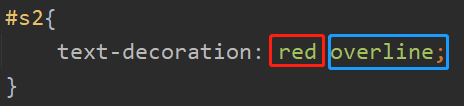
text-decoration 文字的装饰

主要是各种修饰线：上划线、删除线、下划线

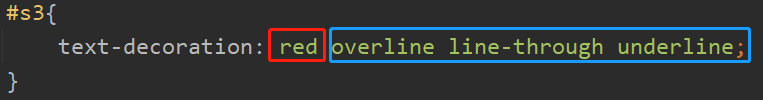
在定义修饰线的时候，可以简单的只定义线



也可以为线添加颜色



还可以同时展示多种修饰线



不管有几条线，只能共用一种颜色

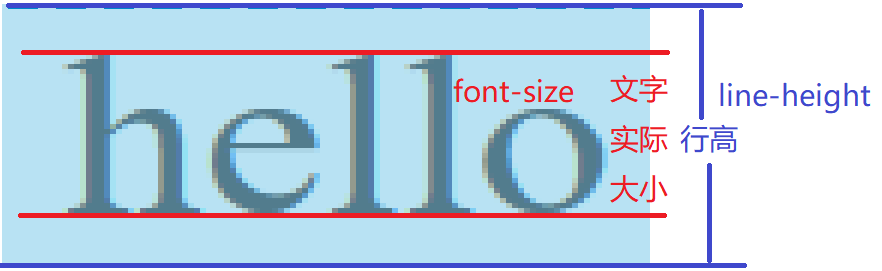
text-align: 对齐方式（水平方向的对齐）

是在当前标签的内部调整对齐的方向，如果标签的大小刚好等于文字，是看不到对齐的变化的。

靠左对齐left、居中center、靠右对齐right

line-height：行高

行高文字所在行的高度，和文字大小没有关系

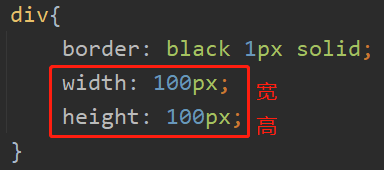


一般来说，行高的使用场景是，实现文字的上下居中（垂直居中）

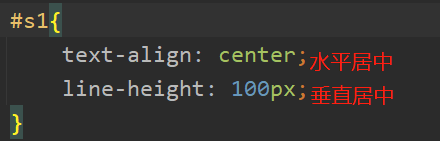
这里有一个规律，文字一定在这一行的正中间，如果将行高设置为容器的高度就可以保证文字也一定在容器的正中间了。

例如：





如果想要文字在div的正中间，需要设置两个内容1、水平居中 2、垂直居中



3、背景

background-color: 纯色背景

这里的颜色赋值还是使用上面的五种赋值方式

background-color: red;

background-image: 图片背景

对图片的赋值要使用url这个函数

background-image: url(“图片路径”);



默认的放置的方式是平铺，如果容器小于图片会显示不全，如果大于图片会重复显示直到铺满容器背景为止

background-repeat: 设置背景图片的平铺逻辑

repeat: 在x轴和y轴上都平铺（重复），默认值

repeat-x: 仅在x轴上重复

repeat-y: 仅在y轴上重复

no-repeat: 只输出一次，不重复

background-position: 设置背景图片首次出现的位置

left左侧 right右侧 top上方 bottom下方

可以使用单一方向进行定位，也可以组合使用（不能矛盾）

background-position: 距离左侧的长度 距离上方的长度;

background-position: 10px 20px;

tips: 讨论定位的时候，一般都会以左上角作为基准点，第一个值就表示距离左侧移动多少，第二个就表示距离上方移动多少。这个规律是普遍有效的。

tips: 背景图片的位移在项目开发中有实际应用场景

在页面上需要使用切好的图片作为按钮或者互动组件的背景，如果每一个按钮都是一张独立的图片，就需要经历多次的图片下载才能展示一张完整的页面。

可以将所有的按钮的图片放在同一张图片上，一般称为精灵图。通过位移只展示这张大图片的一部分，从而显示不同的按钮图片。

4、盒子模型

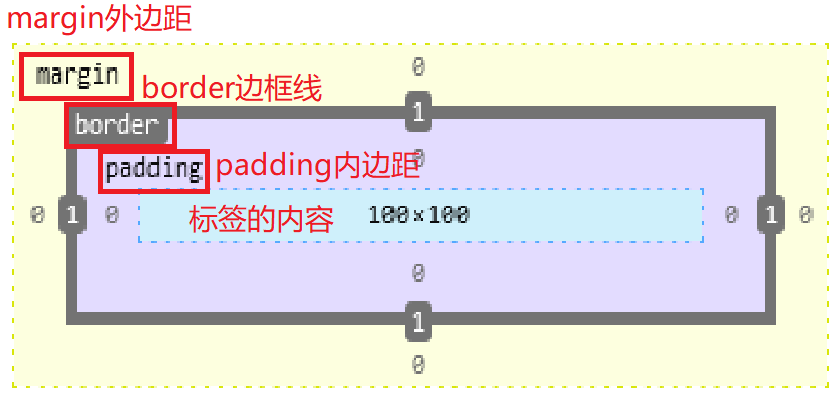
尺寸

width: 设置标签的宽度

height: 设置标签的高度

一般来说，都是用px作为单位，设置标签的固定的大小。

盒子模型



border：边框线

属性是可以整合和拆分的

以border为例，边框是有四个边，三个基本属性，可以根据这两个维度进行拆分

border: black 1px solid; 没有拆分只写了一个总的设置，四个边就使用相同的效果

根据四个边进行拆分

border-top: yellow 5px solid;

border-bottom: green 5px solid;

border-left: purple 5px solid;

border-right: pink 5px solid;

针对每一个边独立赋值，每个边有自己独立的三个属性的赋值

还可以根据三个属性进行拆分

border-color: red;

border-width: 5px;

border-style: dashed;

为每一个属性独立赋值，四个边还是使用相同的属性

还可以同时根据四个边三个属性进行拆分

border-top-color: purple;

border-top-width: 3px;

border-top-style: solid;

border-bottom-color: orange;

border-bottom-width: 5px;

border-bottom-style: double;

border-left-color: green;

border-left-width: 1px;

border-left-style: dotted;

border-right-color: blue;

border-right-width: 7px;

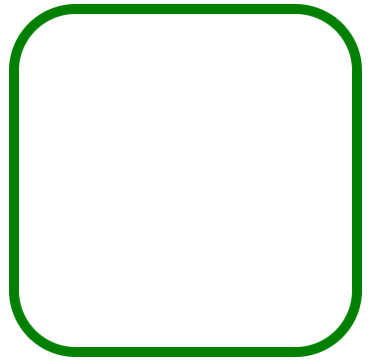
border-right-style: groove;

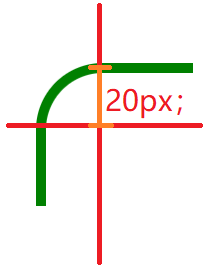
在之前的学习中学到的 font-size font-weight等等这些开头相同的属性其实都可以合并编写。

tips：现在的网页的设计，讲究一个圆润，会尽量避免出现直角的边框，希望边框的角都是圆角的。

border-radius: 圆角的半径

border-radius: 20px; 效果如下图所示。





在盒子模型中，有两个基准点要关注，1、标签的内容本身 2、边框

从内容content到边框border之间的距离就是内边距padding

从边框border到离他最近的另一个标签就是外边距margin

在设置padding和margin的时候，它和border一样可以拆成四个方向来赋值

这里引入一个新的赋值的方法，不做拆分还要四向赋值

以颜色赋值为例

border-color: red; 如果只输入一个值，四个方向是同一个值

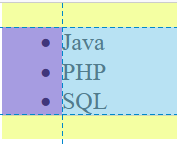
border-color: red orange; 如果输入两个值，第一个值是上下方向，第二个值是左右方向的。

border-color: red orange green purple; 如果输入四个值，会按照顺时针方向依次赋值，顺序应该是 上 右 下 左

tips：如果输入三个值呢？还是按照上 右 下 左的顺序，最后左边没有被赋值沿用右侧的值。

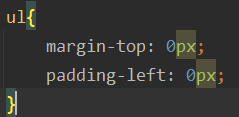
原生的HTML的标签使用的盒子模型

ul/ol/dl



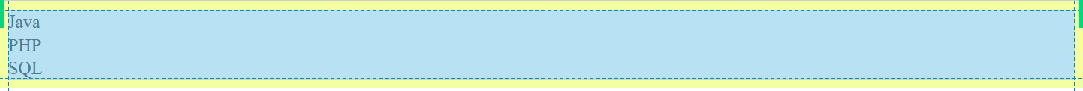
列表有固定的上下方向的外边距，对内容还有左侧内边距

尝试使用代码修改默认的样式



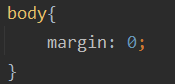
其他的标签的盒子模型

body



margin: 8px;

如果你希望页面没有白边，可以将margin设置成0，就可以有覆盖整个页面的背景



通常商业网页上都会执行这个设置，保证页面展示的完整性。

tips：在使用盒子模型的时候一定要注意。

**块级元素有完整的盒子模型**

**行级元素没有完整的盒子模型**

display：展示的模式

block 块级，当标签的display属性为block的时候他是块级元素

inline 行级，当标签的display属性为inline的时候他是行级元素

inline-block 行内块级元素，希望将多个元素放在一行中，但又要用完整的盒子模型的时候就要使用行内块级这个定义，当被设置为inline-block之后，元素有完整的盒子模型也能多个元素在同一行中。

none 消失，当标签的display属性为none的时候这个标签不展示在页面上

5、定位

position：设置标签的定位方式

如果不设置这个值，页面上的标签是流式布局，块级元素独占一行，行级元素放在一行中，一行放满后换行

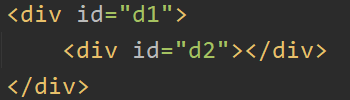
这里修改的就是定位的方式，决定标签出现在页面上的哪里。

absolute 绝对定位

设置完position: absolute之后，元素就离开流式布局，有自己的独立的定位方式，需要额外使用top left bottom right设置元素和边界的距离

tips：虽然提供了四个方向的定义，一般只使用top和left两个方向（左上）

在设置top和left的时候参考系，如果上层元素没有流式布局之外的元素，就以浏览器的边缘作为参考系。如果存在（绝对定位、相对定位的元素），就以这些元素为参考系



以这里的d1和d2为例，如果d2要设置绝对定位，d1如果还是流式布局的，d2就会以浏览器的边缘作为参考系。如果d1已经不在流式布局中，d2就会以不在流式布局中的父标签作为参考系，也就是以d1作为参考系。

层叠

流式布局中的内容总是在层叠的最下层，如果要手动控制层叠效果，至少要被叠加的元素都不在流式布局中

z-index: 1~10

1在最下面 10在最上面

relative 相对定位

在绝对定位中，元素原来在流式布局中的位置会被清除，其他的元素会当做原来的这个不存在再分配自己的位置

相对定位会保留在流式布局中的位置，因为相对定位是相对与原来的位置，它原来的位置就是它的参考系。

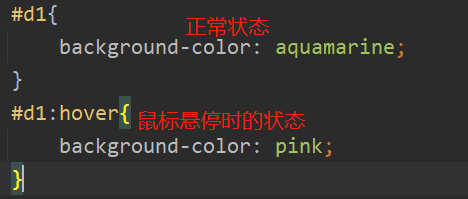
tips：由于相对定位的特性：它不在流式布局中，但是会保留流式布局中的位置，一般用它来做绝对定位的坐标系。

伪类

在选择器上，最终定位的是一个或个标签，伪类就是在这个的基础上定位这个标签的一种特殊状态

hover 鼠标悬停状态

通过这个伪类将一个标签再次分为了两个分支 1、有鼠标悬停的分支 2、没有鼠标悬停的分支

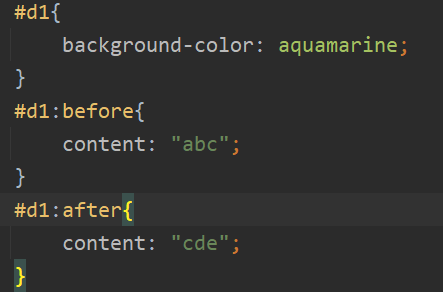


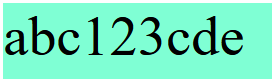
伪对象

伪对象表示标签上的一个特殊的位置

before: 在标签的实际内容之前

after: 在标签的实际内容之后





123是写在div标签中的实际的内容