

---

---

# 第三部分

# CSS

李晶  [lijingjing@bupt.edu.cn](mailto:lijingjing@bupt.edu.cn)

# 参考资料

---

- CSS查询手册  
<http://www.w3school.com.cn/cssref/index.asp>
- 标签效果测试  
<http://www.w3school.com.cn/tiy/t.asp>
- 工具：
  - DreamWeaver
  - 记事本

# CSS样式表

- 认识CSS 样式

- CSS带来的好处
- CSS概述
  - CSS样式说明
  - CSS样式规则
  - CSS样式特征
  - CSS样式的分类
  - CSS样式的注释
- 常用的样式属性及值
- CSS框模型
- CSS定位
- CSS浮动
- 页面布局

# 认识CSS样式

```
<html>
<head>
<title>css样式表</title>
</head>

<body>
<p align="center"><font size="5" color="blue">网页内容</font></p>
</body>
</html>
```

在没有CSS的时候，页面内容的显示依靠标签的属性。一个页面的内容和控制内容显示的标签及属性常混在一起，使页面的代码变得混乱。

# 认识CSS样式

```
<p style=“font-family:黑体;font-size:20px;color:blue;  
text-align:center;”>  
网页内容</p>
```

(1)

<p> 网页内容</p>

P {

网页内容和网页显示格  
式分开管理

font-family:黑体;  
font-size:20px;  
color:blue; (2)  
text-align:center;

}

# 认识CSS样式

---

- CSS ( Cascading Style Sheet层叠样式表)的缩写
- 单纯使用HTML在排版和界面效果上具有局限性
- CSS解决了网页界面排版的难题
- HTML的标签：定义网页的内容
- CSS：网页内容如何显示

# CSS样式表

---

- 认识CSS 样式
- **CSS带来的好处**
- CSS概述
  - CSS样式说明
  - CSS样式规则
  - CSS样式特征
  - CSS样式的分类
  - CSS样式的注释
- 常用的样式属性及值
- CSS框模型
- CSS定位
- CSS浮动
- 页面布局

# CSS带来的好处

---

- **页面内容和显示样式分离**
  - 建立定义样式的CSS文件，并且让所有的HTML文件都来使用CSS文件所定义的样式
- **好处：**
  - **代码简洁易读**
    - 使得呈现内容的HTML页面以及样式文件的代码独立，可读性高；
  - **结构清晰优化SEO ( Search Engine Optimization )**
    - 清晰的代码结构，对搜索引擎表现出友好的结构；
  - **网站维护效率高代价低**
    - 不同的HTML文件可以使用同一个CSS样式文件，要想改变这些页面的显示方式，只改一个CSS样式文件即可。

# CSS带来的好处

---

- 重用样式表
  - 不同的HTML文件可以使用同一个CSS样式文件
- 好处：
  - 网站流量费用降低
    - 不同的HTML文件使用同一个CSS样式文件，只需下载一次CSS样式文件即可；
  - 页面载入更快
    - 样式的重用及下载流量的减少

# CSS样式表

---

- 认识CSS 样式
- CSS带来的好处
- CSS概述
  - CSS样式说明
  - CSS样式规则
  - CSS样式特征
  - CSS样式的分类
  - CSS样式的注释
- 常用的样式属性及值
- CSS框模型
- CSS定位
- CSS浮动
- 页面布局

# 样式说明

---

- **样式**
- 即是格式，网页内容的显示方式。
  - 网页的文字、图片、段落、列表等以什么样的方式显示出来。
- **层叠**
- 当HTML文件引用多个CSS样式时，如果CSS的定义发生冲突，浏览器将依据层次的先后顺序来应用样式。
- 如不考虑样式的优先级时，一般会遵循“最近优选原则”。

# CSS样式表

---

- 认识CSS 样式
- CSS带来的好处
- CSS概述
  - CSS样式说明
  - CSS样式规则
  - CSS样式特征
  - CSS样式表的分类
  - CSS样式表的注释
- 常用的样式属性及值
- CSS框模型
- CSS定位
- CSS浮动
- 页面布局

# 样式规则

---

- 样式表由**样式规则**组成
- 样式规则
  - 定义文档的样式，告诉浏览器如何显示文档。
- 样式表的每个样式规则都有两个主要部分：
  - **选择器**（ selector ）：决定哪些因素要受到影响。
  - **声明**（ declaration ）：决定了选择器所选择的对象将受到什么样的影响。
    - 由一个或多个**属性:值**对组成。

# 样式规则

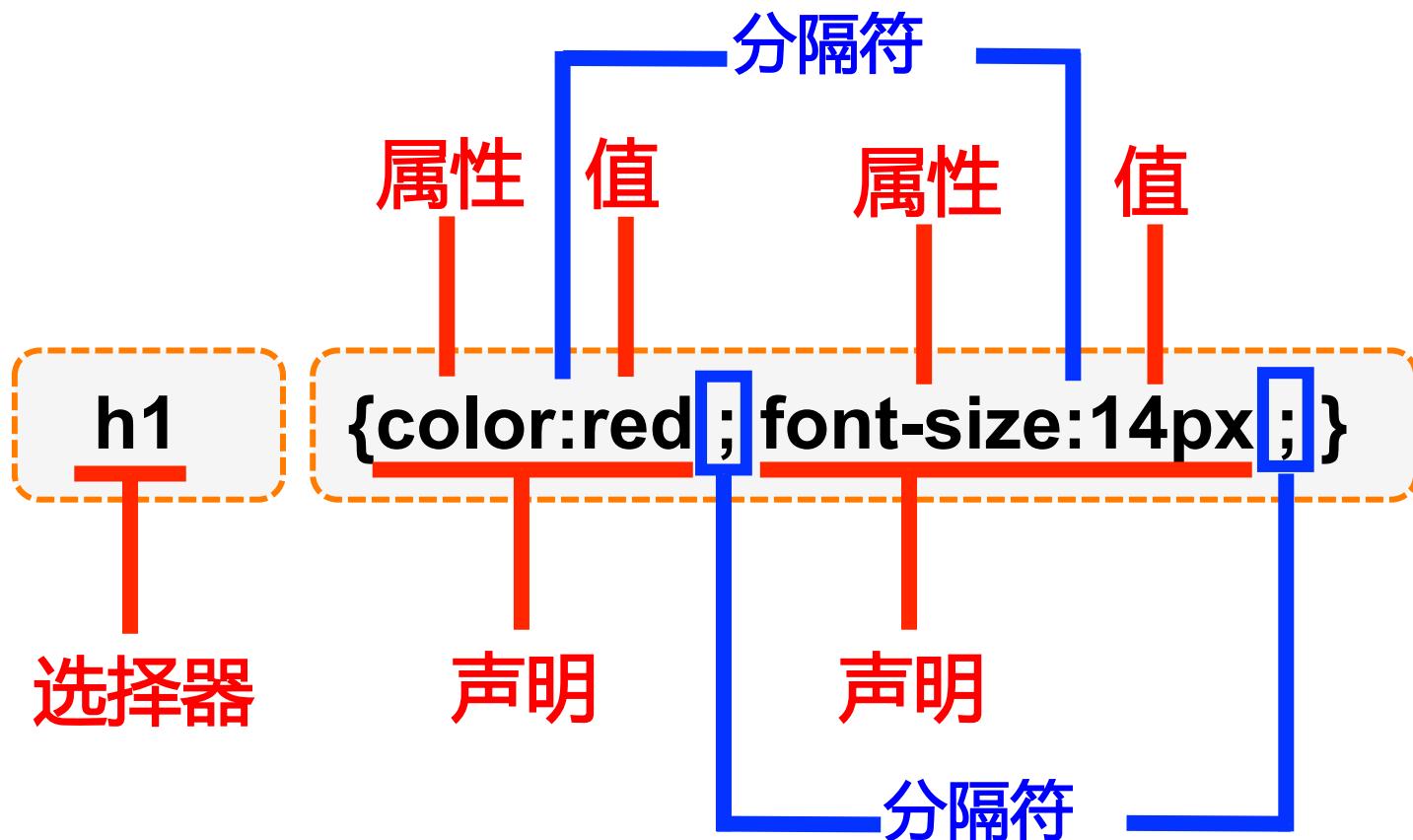
- 基本语法：

```
selector {  
    property1: value;  
    property2: value;  
    .....  
    propertyN: value;  
}
```

- 语法说明：

- selector是选择器，表示希望进行格式化的元素，最普通的选择器就是HTML标签的名称。
- 声明部分包括在选择器后的大括号中；用“属性:属性值”描述要应用的格式化操作；
  - property1、property2和propertyN即为属性名；
  - value是为对应属性名指定的值；
  - 每对属性名/属性值后一般要跟一个分号（括号内只有一对名/值的情况除外）

# 样式规则实例



# CSS样式表

---

- 认识CSS 样式
- CSS带来的好处
- CSS概述
  - CSS样式说明
  - CSS样式规则
  - CSS样式特征
  - CSS样式的分类
  - CSS样式的注释
- 常用的样式属性及值
- CSS框模型
- CSS定位
- CSS浮动
- 页面布局

# 样式特征

- 继承性：发生在有嵌套关系的元素中。
- 许多CSS属性不但影响选择器所定义的元素，而且会被这些元素的后代继承。
  - 例如一个body定义了字符的颜色，这个颜色也会应用到段落的文本中。

✧ 除非定义，否则标签从包含它的父标签的CSS属性中继承。  
✧ 不是所有的属性都具备继承性

# 样式特征

---

- 层叠性
- 在权重（优先级）相同的情况下，同一个标签的样式发生冲突，最后定义的样式会将前面定义的样式覆盖。

与定义样式的顺序有关，与调用顺序无关。

# 样式特征

- <html>
- <head>
- <style type="text/css">
- div {  
    width: 300px;  
    height:300px;  
    line-height:300px;  
    margin:0;  
    padding:0;  
    text-align:center;  
    background-color:black;  
  }  
•   •   •   </style>
- </head>
- <body>
- <div class="div2 div1">HELLO WORLD!</div>
- </body>
- </html>

HELLO WORLD!

定义顺序

调用顺序

of BUPT

# 样式特征

---

- 优先级
- 继承 < 通配符选择器 < 标签选择器 < 类选择器 < ID选择器 < 行内样式 < !Important
- 权重叠加 针对复合选择器在权重（优先级）相同的情况下，同一个标签的样式发生冲突，最后定义的样式会将前面定义的样式覆盖。

# 样式特征

```
• <html>
•   <head>
•     <style type="text/css">
•       div.title{
•         color: red;
•       }
•       .title{
•         color:blue;
•       }
•     </style>
•   </head>
• 
•   <body>
•     <div class="title"> HELLO WORLD!</div>
•   </body>
• 
• </html>
```

- ✧ 假设标签选择器的权重是10，类选择器的权重是100
- ✧ div.title 的权重：  
 $10 + 100 = 110$
- ✧ .title的权重：100
- ✧ div.title的权重大于 .title 所以 class = "title" 的标签字体颜色为**红色**

# CSS样式表

---

- 认识CSS 样式
- CSS带来的好处
- CSS概述
  - CSS样式说明
  - CSS样式规则
  - CSS样式特征
  - CSS样式的分类
  - CSS样式的注释
- 常用的样式属性及值
- CSS框模型
- CSS定位
- CSS浮动
- 页面布局

# CSS样式表的分类

---

- 根据选择器不同可分为：
  - 基本选择器
  - 复合选择器

# CSS样式表的分类

---

- 根据选择器不同可分为：

- 基本选择器

- HTML标记选择器
    - CLASS 类选择器(.className)
    - ID 选择器 (#id号)
    - 通配符选择器
    - 属性选择器
    - 伪类选择器

- 复合选择器

# CSS样式表的分类

---

- **HTML标记选择器**

- 也称为类型选择器；
- 使用HTML标签作为选择器，指定页面中此类元素的样式；
- 任何HTML标签都可以作为选择器。

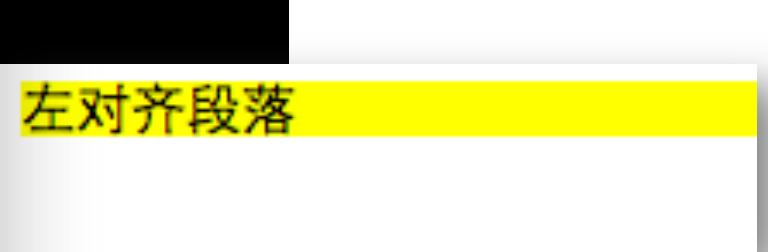
```
P {text-indent: 3em}
```

<p>标签的样式

```
Table {border: 1px solid red}
```

<table>标签的样式

# HTML标记选择器实例

- <html>
- <head>
- <title>HTML标记选择器</title>
- <style type="text/css">  
        p{text-align:left;background-color:yellow}  
  
→
- </style>
- </head>
- <body>
- <p>左对齐段落</p>
- </body>
- </html>

- css中的选择器是p，那么引用该样式的网页中所有p标签都按照这个样子显示

# CSS样式表的分类

---

- **CLASS 类选择器(.className)**

- 使用类选择器，可以把相同的元素分类定义为不同的样式。
- className用于指定选择器的区分类名。

```
selector.classname{property1: value;...}
```

- **提示**：只有适当地标记文档后，才能使用选择器，所以使用选择器通常需要先做一些构想和计划，哪一类内容该显示为什么样式。

# CLASS类选择器实例

- <html>
- <head>
- <title>类选择器</title>
- <style type="text/css">
- → p.left{text-align:left;background-color:yellow}
- p.center{text-align:center}
- </style>
- </head>
- <body>
- → <p class="left">左对齐段落</p>
- → <p class="center">居中段落</p>
- </body>
- </html>

左对齐段落

居中段落

- 使用样式时，只需要在标签内部指定class的值即可引用样式。

# CLASS类选择器实例

- <html>
- <head>
- <title>类选择器</title>
- <style type="text/css">
- .left{text-align:left;background-color:yellow}
- .center{text-align:center}
- </style>
- </head>
- <body>
- <h1 class="center">居中标题</h1>
- <p class="left">左对齐段落</p>
- </body>
- </html>

居中标题

左对齐段落

- 可以不指定选择器，直接用类名，此时称为**多类选择器**。
- 使不同的选择器共享同样的样式，提高代码的灵活度和复用度。

# CLASS类选择器实例

---

- 提示：
  - 类名不允许为以空格分隔的词列表；
  - 不允许出现空格
  - 因为空格是一种特殊的结合符
  - 例如：“class name”

# CSS样式表的分类

---

- **ID选择器(#id号)**

- 通过ID选择器为某个单一元素定义单独的样式。
- IDName指定ID选择器的名称。

```
#IDName{ property1: value;...}
```

# ID选择器实例

- <html>
- <head>
- <title>ID选择器</title>
- <style type="text/css">
  - #note{text-align:left;background-color:yellow}
- </style>
- </head>
- <body>
- <p id="note">左对齐段落</p>
- </body>
- </html>



- 定义名为note的ID选择器
- 在<p>标签中引用该选择器即可。

# ID选择器实例

---

- 提示：
  - ID选择器只能在文档中使用一次；
  - ID选择器不能以组合的方式使用；
  - ID 属性不允许有以空格分隔的词列表；

# CSS样式表的分类

---

- **通配符选择器**
  - universal selector , 显示为一个星号 ( \* )。该选择器可以与任何元素匹配 , 就像是一个通配符。

# 通配符选择器实例

- <html>
- <head>
- <style type="text/css">
- \* {color:red;}
- </style>
- </head>
- <body>
- <h1>这是 heading 1</h1>
- <h2>这是 heading 2</h2>
- <p>段落文本</p>
- </body>
- </html>

=

```
h1{color:red;}\n\nh2{color:red;}\n\np{color:red;}
```

这是 heading 1

这是 heading 2

这是 heading 3

这是 heading 4

这是一段普通的段落文本。

等价于列出文档中所有元素的一个分组选择器。  
利用通配选择器，只需敲一次键（仅一个星号）  
就能使文档中所有元素的 color 属性值指定为  
red。

# CSS样式表的分类

- **属性选择器**

- 根据元素的属性及属性值来选择元素。



**Html 标签 [属性]{property1:value1;.....}**

**Html 标签 [表达式]{property1:value1;.....}**

# CSS样式表的分类

---

## • 属性选择器

选择器

→ [attribute]

[attribute=value]

[attribute~=value]

[attribute|=value]

[attribute^=value]

[attribute\$=value]

[attribute\*=value]

描述

选取带有指定属性的元素。

选取带有指定属性和值的元素。

选取属性值中包含指定词汇的元素。

选取带有以指定值开头的属性值的元素，该值必须是整个单词。

匹配属性值以指定值开头的每个元素。

匹配属性值以指定值结尾的每个元素。

匹配属性值中包含指定值的每个元素。

# 属性选择器实例

- <html>
  - <head>
  - <style>
  - `a[name]{background-color:yellow;}`
  - </style>
  - </head>
  - <body>
  - <p>带有name属性的链接会得到黄色背景 : </p>
  - `<a href="http://www.w3school.com.cn">w3school.com.cn</a>`
  - `<a href="http://www.disney.com" name="disney">disney.com</a>`
  - `<a href="http://www.wikipedia.org" >wikipedia.org</a>`
  - </body>
  - </html>
- 带有name属性的链接会得到黄色背景 :
- [w3school.com.cn](http://www.w3school.com.cn) [disney.com](http://www.disney.com) [wikipedia.org](http://www.wikipedia.org)

# CSS样式表的分类

---

## • 属性选择器

---

选择器	描述
[attribute]	选取带有指定属性的元素。
→ [attribute=value]	<b>选取带有指定属性和值的元素。</b>
[attribute~=value]	选取属性值中包含指定词汇的元素。
[attribute =value]	选取带有以指定值开头的属性值的元素，该值必须是整个单词。
[attribute^=value]	匹配属性值以指定值开头的每个元素。
[attribute\$=value]	匹配属性值以指定值结尾的每个元素。
[attribute*=value]	匹配属性值中包含指定值的每个元素。

---

# 属性选择器实例

- <html>
- <head>
- <style>
- a[name=wiki]{background-color:yellow;}
- </style>
- </head>
- <body>
- <p>name=wiki的链接会得到黄色背景 : </p>
- { <a href="http://www.w3school.com.cn">w3school.com.cn</a>  
  <a href="http://www.disney.com" name="disney">disney.com</a>  
  <a href="http://www.wikipedia.org" name="wiki">wikipedia.org</a>
- </body>
- </html>

name=wiki的链接会得到黄色背景 :

[w3school.com.cn](http://www.w3school.com.cn) [disney.com](http://www.disney.com) [wikipedia.org](http://www.wikipedia.org)

# CSS样式表的分类

---

## • 属性选择器

---

选择器	描述
[attribute]	选取带有指定属性的元素。
[attribute=value]	选取带有指定属性和值的元素。
<b>[attribute~=value]</b>	<b>选取属性值中包含指定词汇的元素。</b>
[attribute =value]	选取带有以指定值开头的属性值的元素，该值必须是整个单词。
[attribute^=value]	匹配属性值以指定值开头的每个元素。
[attribute\$=value]	匹配属性值以指定值结尾的每个元素。
[attribute*=value]	匹配属性值中包含指定值的每个元素。

---

# 属性选择器实例

- <html>
- <head>
- <style>
- [name~=disney]{background-color:yellow;}
- </style>
- </head>
- <body>
- <p>name 属性中包含单词 "disney" 的链接会获得黄色背景。</p>
- { <a href="http://www.w3school.com.cn">w3school.com.cn</a>
- { <a href="http://www.disney.com" name="to disney">disney.com</a>
- { <a href="http://www.wikipedia.org" name="wiki">wikipedia.org</a>
- </body>
- </html>

name 属性中包含单词 "disney" 的链接会获得黄色背景。

[w3school.com.cn](http://www.w3school.com.cn) [disney.com](http://www.disney.com) [wikipedia.org](http://www.wikipedia.org)

# CSS样式表的分类

---

## • 属性选择器

选择器	描述
[attribute]	选取带有指定属性的元素。
[attribute=value]	选取带有指定属性和值的元素。
[attribute~=value]	选取属性值中包含指定词汇的元素。
→ [attribute =value]	选取带有以指定值开头的属性值的元素，该值必须是整个单词。
[attribute^=value]	匹配属性值以指定值开头的每个元素。
[attribute\$=value]	匹配属性值以指定值结尾的每个元素。
[attribute*=value]	匹配属性值中包含指定值的每个元素。

# 属性选择器实例

- <html>
  - <head>
  - <style>
  - [name=dis]{background-color:yellow;}
  - </style>
  - </head>
  - <body>
  - <p>name 属性中以 "disney" 打头的链接会获得黄色背景。</p>
  - { <a href="http://www.w3school.com.cn">w3school.com.cn</a>
  - { <a href="http://www.disney.com" name="dis-island">disney.com</a>
  - { <a href="http://www.wikipedia.org" name="wiki">wikipedia.org</a>
  - </body>
  - </html>
- name 属性中以 "dis" 打头的链接会获得黄色背景。  
[w3school.com.cn](http://www.w3school.com.cn) [disney.com](http://www.disney.com) [wikipedia.org](http://www.wikipedia.org)
- 样式表中的name值必须是整个单词，比如 name="dis"，或者后面跟着连字符，比如 name="dis-island"
- 属性中的name值必须是整个单词，比如 name="dis"，或者后面跟着连字符，比如 name="dis-island" 不可以出现空格

# CSS样式表的分类

---

## • 属性选择器

---

选择器	描述
[attribute]	选取带有指定属性的元素。
[attribute=value]	选取带有指定属性和值的元素。
[attribute~=value]	选取属性值中包含指定词汇的元素。
[attribute =value]	选取带有以指定值开头的属性值的元素，该值必须是整个单词。
→ [attribute^=value]	匹配属性值以指定值开头的每个元素。
[attribute\$=value]	匹配属性值以指定值结尾的每个元素。
[attribute*=value]	匹配属性值中包含指定值的每个元素。

---

# 属性选择器实例

- <html>
- <head>
- <style>
- [name^=dis]{background-color:yellow;}
- </style>
- </head>
- <body>
- <p>name 属性中以 "disney" 打头的链接会获得黄色背景。</p>
- <a href="http://www.w3school.com.cn">w3school.com.cn</a>
- <a href="http://www.disney.com" name="disney-island">disney.com</a>
- <a href="http://www.wikipedia.org" name="disney wiki">wikipedia.org</a>
- </body>
- </html>

name 属性中以 "dis" 打头的链接会获得黄色背景。

[w3school.com.cn](http://www.w3school.com.cn) [disney.com](http://www.disney.com) [wikipedia.org](http://www.wikipedia.org)

值不再要求是整个单词，可以存在空格或者其它字符

# CSS样式表的分类

---

## • 属性选择器

---

选择器	描述
[attribute]	选取带有指定属性的元素。
[attribute=value]	选取带有指定属性和值的元素。
[attribute~=value]	选取属性值中包含指定词汇的元素。
[attribute =value]	选取带有以指定值开头的属性值的元素，该值必须是整个单词。
[attribute^=value]	匹配属性值以指定值开头的每个元素。
<b>[attribute\$=value]</b>	<b>匹配属性值以指定值结尾的每个元素。</b>
[attribute*=value]	匹配属性值中包含指定值的每个元素。

---

# 属性选择器实例

- <html>
  - <head>
  - <style>
  - [name\$=island]{background-color:yellow;}
  - </style>
  - </head>
  - <body>
  - <p>name 属性中以 "island" 结尾的链接会获得黄色背景。 </p>
  - <a href="http://www.w3school.com.cn">w3school.com.cn</a>
  - <a href="http://www.disney.com" name="disney island">disney.com</a>
  - <a href="http://www.wikipedia.org" name="disney wiki">wikipedia.org</a>
  - </body>
  - </html>
- name 属性中以 "island" 结尾的链接会获得黄色背景。
- w3school.com.cn disney.com wikipedia.org
- 值不要求是整个单词，可以存在空格或者其它字符

# CSS样式表的分类

---

## • 属性选择器

---

选择器	描述
[attribute]	选取带有指定属性的元素。
[attribute=value]	选取带有指定属性和值的元素。
[attribute~=value]	选取属性值中包含指定词汇的元素。
[attribute =value]	选取带有以指定值开头的属性值的元素，该值必须是整个单词。
[attribute^=value]	匹配属性值以指定值开头的每个元素。
[attribute\$=value]	匹配属性值以指定值结尾的每个元素。
<b>[attribute*=value]</b>	<b>匹配属性值中包含指定值的每个元素。</b>

---

# 属性选择器实例

- <html>
  - <head>
  - <style>
  - [href\*=www]{background-color:yellow;}
  - </style>
  - </head>
  - <body>
  - <p>href属性中包含 "www" 的链接会获得黄色背景。 </p>
  - {<a href=".index.html">w3school.com.cn</a>
  - {<a href="http://www.disney.com">disney.com</a>
  - {<a href="http://www.wikipedia.org">wikipedia.org</a>
  - </body>
  - </html>
- href属性中包含 "www" 的链接会获得黄色背景。  
w3school.com.cn disney.com wikipedia.org
- 值不再强调关键词出现的位置。

# CSS样式表的分类

---

- **伪类选择器**

- 类用于向某些选择器添加特殊的效果。
- 在选择器后加上伪类名（pseudo-class）

```
selector : pseudo-class {property: value}
```

```
selector.class : pseudo-class {property: value}
```

# 伪类选择器实例

- 锚标签的伪类
  - 锚标签有四种状态的伪类，分别为未被访问时、鼠标悬停时、被用户激活时（鼠标已点击但未释放）、被访问过后。

伪类名	说明
link	设置对象在未被访问前的样式
hover	设置对象在其鼠标悬停时的样式
active	设置对象在被用户激活（在鼠标点击与释放之间）时的样式
visited	设置对象在其链接地址已被访问过时的样式

# 伪类选择器实例

- <html>
- <head>
- <style type="text/css">
- a:link {color: #FF0000} /\*未访问的链接\*/ 这是一个链接。
- a:visited {color: #00FF00} /\*已访问的链接\*/ 这是一个链接。
- a:hover {color: #FF00FF} /\*鼠标悬停在链接上\*/ 这是一个链接。
- a:active {color: #0000FF} /\*选定的链接\*/ 这是一个链接。
- </style>
- </head>
  
- <body>
- → <p><a href="/index.html" target="\_blank">这是一个链  
接</a></p> • 在 CSS 定义中，a:hover 必须被置于 a:link 和 a:visited 之后，才是有效的。  
• 在 CSS 定义中，a:active 必须被置于 a:hover 之后，才是有效的。
- </body>
- </html>

锚标签的伪类实例

# CSS样式表的分类

- **伪元素选择器**

- 用于向某些选择器设置特殊效果。
- 在选择器后加上伪元素名 ( pseudo-element )

```
selector:pseudo-element {property:value;}
```

```
selector.class:pseudo-element {property:value;}
```

# CSS样式表的分类

- 伪元素选择器

伪元素名	说明
:first-letter	向文本的第一个字母添加特殊样式
:first-line	向文本的首行添加特殊样式
:before	在元素之前添加内容
:after	在元素之后添加内容



# CSS样式表的分类

---

- **:first-letter伪元素**

- 用于指定一个元素的第一个字母的特殊样式。

- 注：

- 所有前导标点符号应当与第一个字母一同应用该样式。

- 某些语言有一些要处理为单个字符的字母组合，如果是这样，用户代理可能会把首字母同时应用到这个字母组合。

# CSS样式表的分类

---

- **:first-letter伪元素**
  - 可以应用到首字母的属性是有限的。

<b>font</b>	<b>text-decoration</b>
<b>color</b>	<b>vertical-align</b> (仅当 'float' 为 'none' 时)
<b>background</b>	<b>text-transform</b>
<b>margin</b>	<b>line-height</b>
<b>padding</b>	<b>float</b>
<b>border</b>	<b>clear</b>

# 伪元素实例

- <html>
- <head>
- <style type="text/css">
- p:first-letter {  
color: #ff0000;  
font-size:xx-large}
- </style>
- </head>
- <body>
- > <p>
- The only thing that is changed is everything.
- </p>
- </body>
- </html>

The only thing that is changed is everything.

# CSS样式表的分类

- 伪元素选择器

伪元素名	说明
:first-letter	向文本的第一个字母添加特殊样式
:first-line	向文本的首行添加特殊样式
:before	在元素之前添加内容
:after	在元素之后添加内容



# CSS样式表的分类

---

- **:first-line伪元素**
  - 用于设置元素中的第一行文本的样式，而不论该行出现多少单词。
  - 行的结束由浏览器自动决定。
  - 注：
    - 只能与块级元素关联。(块级元素后面会讲到)

# CSS样式表的分类

---

- **:first-line伪元素**
  - 可以应用首行伪元素的属性是有限的。

<b>font</b>	<b>text-decoration</b>
<b>color</b>	<b>vertical-align</b>
<b>background</b>	<b>text-transform</b>
<b>word-spacing</b>	<b>line-height</b>
<b>letter-spacing</b>	<b>clear</b>

# CSS样式表的分类

---

- 伪元素选择器

伪元素名	说明
:first-letter	向文本的第一个字母添加特殊样式
:first-line	向文本的首行添加特殊样式
→ :before	在元素之前添加内容
→ :after	在元素之后添加内容

# CSS样式表的分类

---

- **:before伪元素**
  - 在元素之前添加内容。
- **:after伪元素**
  - 在元素之后添加内容。
- **语法：**
  - selector : before { content: “...” ;  
  property1: value1; ...}
  - selector : after { content: “...” ; property1:  
  value1; ...}

# CSS样式表的分类

---

CHATOYANT

INEFFABLE

EFFIC



WAFTURE

SUMPTUOUS



# CSS样式表的分类

---

- 根据选择器不同可分为：
- 基本选择器
- 复合选择器
  - 后代选择器
  - 子元素选择器
  - 相邻兄弟选择器
  - 组合选择器

# CSS样式表的分类

---

- **后代选择器**
  - 也称为包含选择器，可以选择作为某元素后代的元素。
- **语法解释**
  - 规则左边的选择器一端包括两个或多个用空格分隔的选择器。
  - 选择器之间的**空格**是一种结合符（combinator）。
  - 每个空格结合符可以解释为“... 作为 ... 的一部分”、“... 作为 ... 的后代”。
  - 要求必须从右向左读选择器。

```
selector1 selector2 ...{property1: value;...}
```

# 后代选择器实例

- <html>
- <head>
- <style type="text/css">
- h1 em {color:red;}
- </style>
- </head>
- <body>
- <h1>一个<em>重要</em> 标题</h1>
- <p>一个<em>重要</em> 段落</p>
- </body>
- </html>

一个**重要** 标题  
一个**重要** 段落

- 可以定义后代选择器来创建一些规则，使这些规则在某些文档结构中起作用，而在另外一些结构中不起作用。
- 只对 h1 元素中的 em 元素应用样式。

# 后代选择器实例

ul em {color:red; font-weight:bold;}

- <ul>
- <ol>
  - <li><em>List item 1-1</em></li>
  - <li>List item 1-2
    - <ol>
    - <li>List item 1-2-1</li>
  - <li>List item <em>1-2-2</em></li>
  - <li>List item 1-2-3</li>
- </ol>
- </li>
- </ul>

- 1. *List item 1-1*
- 2. List item 1-2
  - 1. List item 1-2-1
  - 2. List item **1-2-2**
  - 3. List item 1-2-3

- 选择从 ul 元素继承的所有 em 元素，而不论 em 的嵌套层次多深。
- 因此，ul em 将会选择以下标记中的所有 em 元素。

# CSS样式表的分类

---

- 子元素选择器
  - 只能选择作为某元素子元素的元素。
- 语法解释
  - 子选择器使用大于号作为子结合符。
  - 子结合符两边可以有空白符。
  - 可以解释为“选择 标签1 元素直接子元素的 标签2 元素”。

```
html 标签1 > html 标签2 {property1:value1;.....}
```

# 子元素选择器实例

- <html>
  - <head>
  - <style type="text/css">
  - h1 > strong {color:red;}
  - </style>
  - </head>
  - <body>
- <h1>This is <strong>very</strong> <strong>very</strong> important.</h1>
- <h1>This is <em>really <strong>very</strong></em> important.</h1>
  - </body>
  - </html>
- This is **very very important.**  
This is *really very important.*
- 把第一个 h1 下面的两个 strong 元素变为红色
  - 第二个 h1 中的 strong 不受影响，因为非直接后继。

# CSS样式表的分类

---

- 相邻兄弟选择器
  - 可选择紧接在另一元素后的元素，且二者有相同父元素。
- 语法解释
  - 相邻兄弟选择器使用加号（相邻兄弟结合符）。
  - 相邻兄弟结合符旁边可以有空白符。
  - 可以解释为“选择紧接在 标签1 元素后出现的 标签2，标签1 和 标签2 元素拥有共同的父元素。”

```
html 标签1 + html 标签2 {property1:value1;.....}
```

# 相邻兄弟选择器实例

li + li {color:red;}

- <div>
- <ul>
- <li>List item 1</li>
- { <li>List item 2</li>
- { <li>List item 3</li>
- </ul>
- <ol>
- <li>List item 1</li>
- { <li>List item 2</li>
- { <li>List item 3</li>
- </ol>
- </div>

- List item 1
  - List item 2
  - List item 3
- 
- 1. List item 1
  - 2. List item 2
  - 3. List item 3

- 选择紧接在li标签后出现的li标签
- 并且二者具有同样的父亲结点。

# CSS样式表的分类

- **组合选择器**

- 当有多个选择器需要设置相同的属性和属性值时，可以用逗号将选择器隔开进行组合定义，即一次性设置多个选择器的属性和属性值，这样可以减少样式重复定义。

```
selector1, selector2,...{property1: value;...}
```

- 通过组合，可以将某些类型的样式“压缩”在一起，这样就可以得到更简洁的样式表。

# 组合选择器实例

- <html>
- <head>
- <style type="text/css">
- **h1,p{color:red;}**
- </style>
- </head>
- <body>
- <h1>标题</h1>
- <p>段落</p>
- </body>
- </html>

=

标题

段落

**h1{color:red;}**  
**p{color:red;}**

# CSS样式表的分类

```
h1 {color:silver; background:white;}  
h2 {color:silver; background:gray;}  
h3 {color:white; background:gray;}  
h4 {color:silver; background:white;}  
b {color:gray; background:white;}
```



```
h1, h4 {color:silver; background:white;}  
h2 {color:silver; background:gray;}  
h3 {color:white; background:gray;}  
b {color:gray; background:white;}
```

选择器的分组

# CSS样式表的分类

```
h1, h4 {color:silver; background:white;}  
h2 {color:silver; background:gray;}  
h3 {color:white; background:gray;}  
b {color:gray; background:white;}
```



- h1, h2, h4 {color:silver;}
- h2, h3 {background:gray;}
- h1, h4, b {background:white;}
- h3 {color:white;}
- b {color:gray;}

选择器的分组

# CSS样式表的分类

---

- 定义的位置不同
  - 行内样式(Inline Style)
  - 内部样式(Internal Style sheet)
  - 外部样式(External Style sheet)

# CSS样式的分类

---

- 行内样式

- 行内样式是指将CSS语句混合在HTML标签中使用的方式
- CSS语句只对其所在的标签有效
- 行内样式通过所在标签的style属性声明。

```
<p style= "font-family:黑体;font-size:20px;color:blue;text-align:center" >内容</p>
```

# CSS样式表的分类

- 内部样式

- 内部样式表是指在HTML的<style>标签中声明样式的方式。
- 内部样式表通过<style>标签声明，只对所在的网页有效。
- Style标签位于head标签内部

```
<head>
<STYLE TYPE="text/css">
    p {
        color:red;
        font-size: 24px;
    }
</STYLE>
</head>
<body>
    <p>内容</p>
</body>
```

# CSS样式表的分类

- **外部样式**

- 外部样式表是指将CSS样式表保存成一个独立的文件，然后将该文件引用到网页中的方式。样式表文件名采用后缀“css”。

```
p {  
    color:red;  
    font-size: 24px;  
}
```

保存文件，命名为  
**example.css**

```
<head>  
<link href=".//example.css"  
rel="stylesheet" type="text/css"/>  
</head>  
<body>  
<p>内容</p>  
</body>
```

**Href:**被引用样式文件的URL，相对地址

**Rel:**指定链接文件的类型，**stylesheet**表示外部文件的类型为**CSS**

**Type:**链接文件的内容类型，遵守**MIME**规范

# CSS样式表的分类

---

- 行内样式表、内部样式表、外部样式表各有优势，实际开发中常常需要混合使用：
  - 有关整个网站统一风格的样式代码，采用外部样式表，放置在独立的样式文件\*.css。
  - 某些样式不同的页面，除了链接外部样式文件，还需定义内部样式。
  - 某个网页内，部分内容“与众不同”，采用行内样式。

# CSS样式表的使用优先级

---

- 对于某个HTML标记：
  - 当有多种位置不同的样式时，如果规定的样式没有冲突，则叠加；如果有冲突，按照如下规则叠加一个新的虚拟样式表：
    - 行内样式（在 HTML 元素内部）
    - 内部样式表（位于 `<head>` 标签内部）
    - 外部样式表
    - 浏览器缺省设置



# CSS样式表

---

- 认识CSS 样式
- CSS带来的好处
- CSS概述
  - CSS样式说明
  - CSS样式规则
  - CSS样式特征
  - CSS样式的分类
  - CSS样式的注释
- 常用的样式属性及值
- CSS框模型
- CSS定位
- CSS浮动
- 页面布局

# 样式的注释

---

- 在样式表中添加注释有助于记住复杂的样式规则的作用，应用的范围等，便于维护和应用。
- 例如，下面是一个添加注释的样例。
- `/*此标记应用在文档中*/`  
`h1{color:red;background:yellow;}`

# CSS样式表

---

- 认识CSS 样式
- CSS带来的好处
- CSS概述
  - CSS样式说明
  - CSS样式规则
  - CSS样式特征
  - CSS样式的分类
  - CSS样式的注释
- 常用的样式属性及值
- CSS框模型
- CSS定位
- CSS浮动
- 页面布局

# CSS文本样式属性

---

- **文本属性**
- 定义文本的外观。
- 通过文本属性，可以改变文本的颜色、字符间距，对齐文本，装饰文本，对文本进行缩进等。

# CSS文本样式属性

文本属性	功能	取值方式
text-indent	实现文本的缩进	长度 ( length ) : 绝对单位 ( px ) 或者相对单位 ( 百分比 ) : 相对父标签宽度的百分比。可取负值。
text-align	设置文本的对齐方式	left : 左对齐 ; center : 居中对齐 ; right : 右对齐 ; justify : 两端对齐
line-height	设置行高	数字或百分比 , 可参考文本缩进的取值方式。
word-spacing	词间隔 , 修改段落中词之间的距离	缺省值为 0 。当值为正数时 , 词之间的间隔会增大 , 反之 , 为负数时 , 词间距会减少。 <b>对中文无效。</b>
letter-spacing	字母间隔 , 控制字母或字符之间的间隔	取值同文字间隔类似
text-transform	文本转换 , 主要是对字母大小写的转换	uppercase : 将整个文本变为大写 ; lowercase : 将整个文本变为小写 ; capitalize : 将整个文本的每个文字的首字母大写
text-decoration	文本修饰 , 修饰强调段落中一些主要文字	none 、 underline ( 下划线 ) 、 overline ( 上划线 ) 、 line-through ( 删除线 ) 和 blink ( 闪烁 )

# CSS文字样式属性

---

- 文字属性
  - CSS中通过一系列的文字属性来设置网页中文字的显示效果，主要包括文字字体、文字加粗、字号、文字样式。

# CSS文字样式属性

文字属性	功能	取值方式
font-family	设置文字字体	文字字体取值可以为：宋体、ncursive、fantasy、serif等多种字体
font-weight	文字加粗	normal : 正常字体； bold : 粗体； bolder : 特粗体； lighter : 细体
font-size	文字字号	absolute-size : 根据对象字体进行调节； relative-size : 相对于父对像中字体尺寸进行相对调节； length : 百分比。由浮点数字和单位标识符组成的长度值，不可为负值。其百分比取值是基于父标签中字体的尺寸
font-style	文字样式	normal : 正常的字体； italic : 斜体； oblique : 倾斜的字体

# CSS文字样式属性

---

- **font 属性**：简写属性，在一个声明中一次设置两个或更多的字体属性。
- **说明：**
  - 可以按顺序设置如下属性：
    - font-style
    - font-variant
    - font-weight
    - font-size/line-height
    - font-family
  - 可以不设置其中的某个值，比如 font:100% verdana; 也是允许的。未设置的属性会使用其默认值。

# CSS背景样式属性

背景属性	功能	取值方式
background-color	设置对象的背景颜色	属性的值为有效的色彩数值
background-image	设置背景图片	可以通过为url指定值来设定绝对或相对路径指定网页的背景图像
background-repeat	背景重复，设置指定的背景图象如何被重复	repeat : 背景图像平铺（有横向和纵向两种取值：repeat-x : 图象横向重复；repeat-y : 图象纵向重复）；norepeat : 背景图像不平铺
background-attachment	背景附加，设置指定的背景图像是跟随内容滚动，还是固定不动	scroll : 背景图像是随内容滚动；fixed : 背景图像固定即内容滚动图像不动
background-position	背景位置，确定背景的水平和垂直位置(x y)	左对齐(left)、右对齐(right)、顶部(top)、底部(bottom)和值(自定义背景的起点位置，可对背景的位置做出精确的控制)
background	复合属性，用于设定对象的背景样式	该属性的取值实际上对应上面几个具体属性顺序取值

# CSS背景样式属性

---

- **说明：**
- background-position:x y;
  - x值:left right center 值(单位为象素)/百分比
  - y值:top bottom center 值(单位为象素)/百分比
- background : 按顺序一次性设置所有背景属性。
  - background-color background-image background-repeat background-attachment background-position
  - 默认值为 : transparent none repeat scroll 0 0.

# CSS列表样式属性

---

- **列表属性**
  - CSS 列表属性允许放置、改变列表项标志，或者将图像作为列表项标志。

# CSS列表样式属性

列表属性	功能	取值方式
list-style-type	设置列表项标志的类型。	None(无标记)、 disc(默认，实心圆)、 circle ( 空心圆)、 square(实心方块)、 decimal(数字)、 lower-roman(小写罗马数字)、 upper-roman(大写罗马数字)等。
list-style-position	设置列表中列表项标志的位置。	Inside(列表项标记放在文本内)、 outside(默认值。保持标记于文本的左侧)、
list-style-image	将图象设置为列表项标志	URL(图像的路径)、 none(默认。无图形被显示)
list-style	复合属性。把所有列表的属性设置于一个声明中。	上面几个属性顺序取值

# CSS表格样式属性

列表属性	功能	取值方式
border-collapse	设置是否把表格边框合并为单一的边框。	Separate(默认值,边框会被分开。不会忽略 border-spacing 和 empty-cells 属性。) collapse(如果可能，边框会合并为一个单一的边框。会忽略 border-spacing 和 empty-cells 属性。)
border-spacing	设置分隔单元格边框的距离。	规定相邻单元边框之间的距离(px、cm 等)。不可用负值。length (水平和垂直间距)。length length(第一个水平间距，第二个垂直间距)。
caption-side	设置表格标题的位置。	top(默认，标题在表格之上)、bottom(标题在表格之下)
empty-cells	设置是否显示表格中的空单元格。	Hide(不在空单元格周围绘制边框)、show(在空单元格周围绘制边框。默认。)
table-layout	设置显示单元、行和列的方法。	Automatic(默认,列宽度由单元格内容设定)、fixed(列宽由表格宽度和列宽度设定)

# CSS表格样式实例

- <html>
- <head>
- <style type="text/css">
- table {border-collapse:collapse;} ←
- table, td, th {border:1px solid black;}
- </style>
- </head>
- <body>
- <table>
- <tr><th>Firstname</th><th>Lastname</th></tr>
- <tr><td>Bill</td><td>Gates</td></tr>
- </table>
- </body>
- </html>

Firstname	Lastname
Bill	Gates

由于给表格设置了边框 ( border ) , 该表格应该有边框 , 但是border-collapse属性设置为collapse , 边框便合并为单一的边框。如果将该属性值改为separate , 那么表格将显示为如下样式 , 该样式也是html的默认样式。

Firstname	Lastname
Bill	Gates

# CSS样式表

---

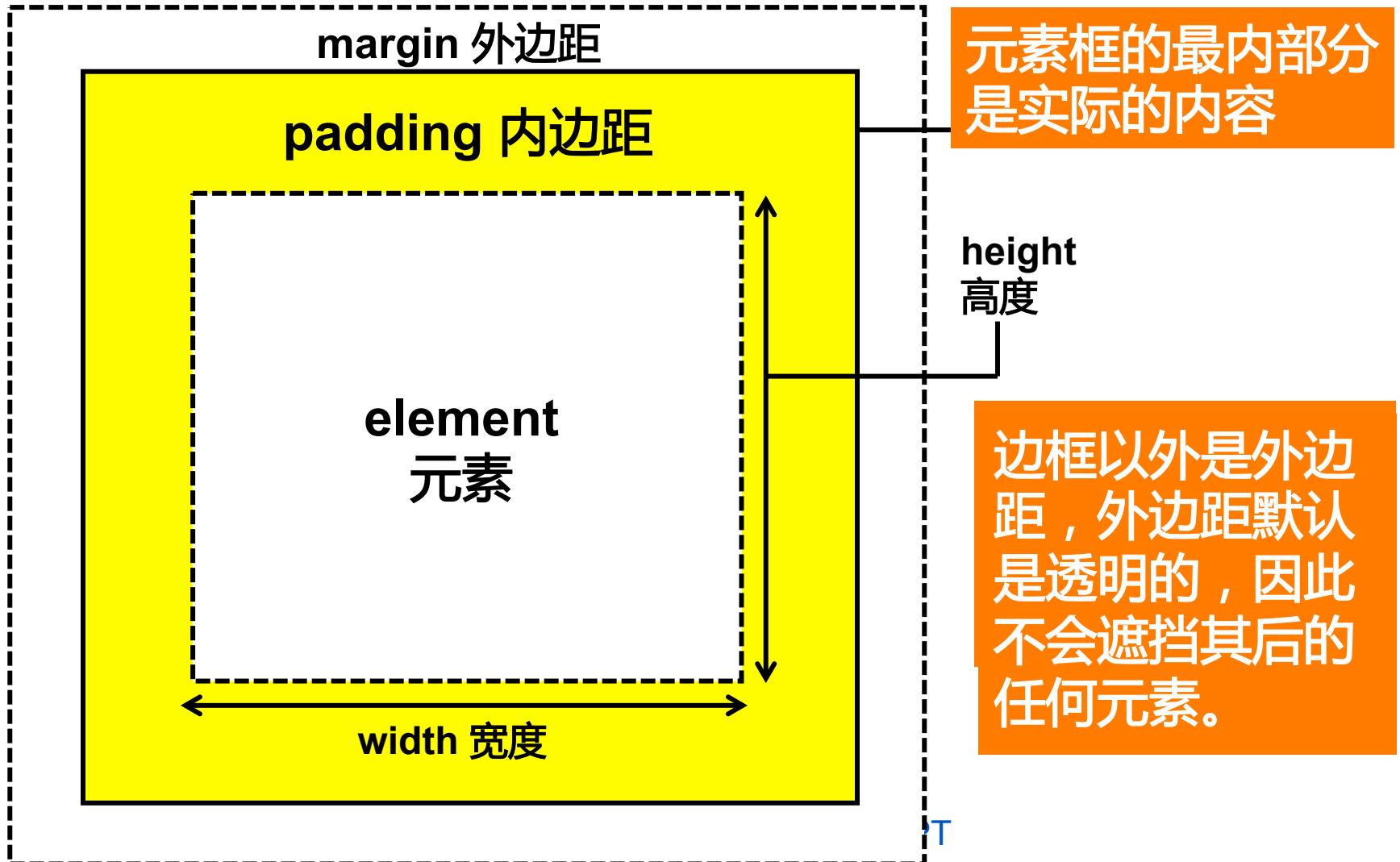
- 认识CSS 样式
- CSS带来的好处
- CSS概述
  - CSS样式说明
  - CSS样式规则
  - CSS样式特征
  - CSS样式的分类
  - CSS样式的注释
- 常用的样式属性及值
- CSS框模型
- CSS定位
- CSS浮动
- 页面布局

# CSS框模型概述

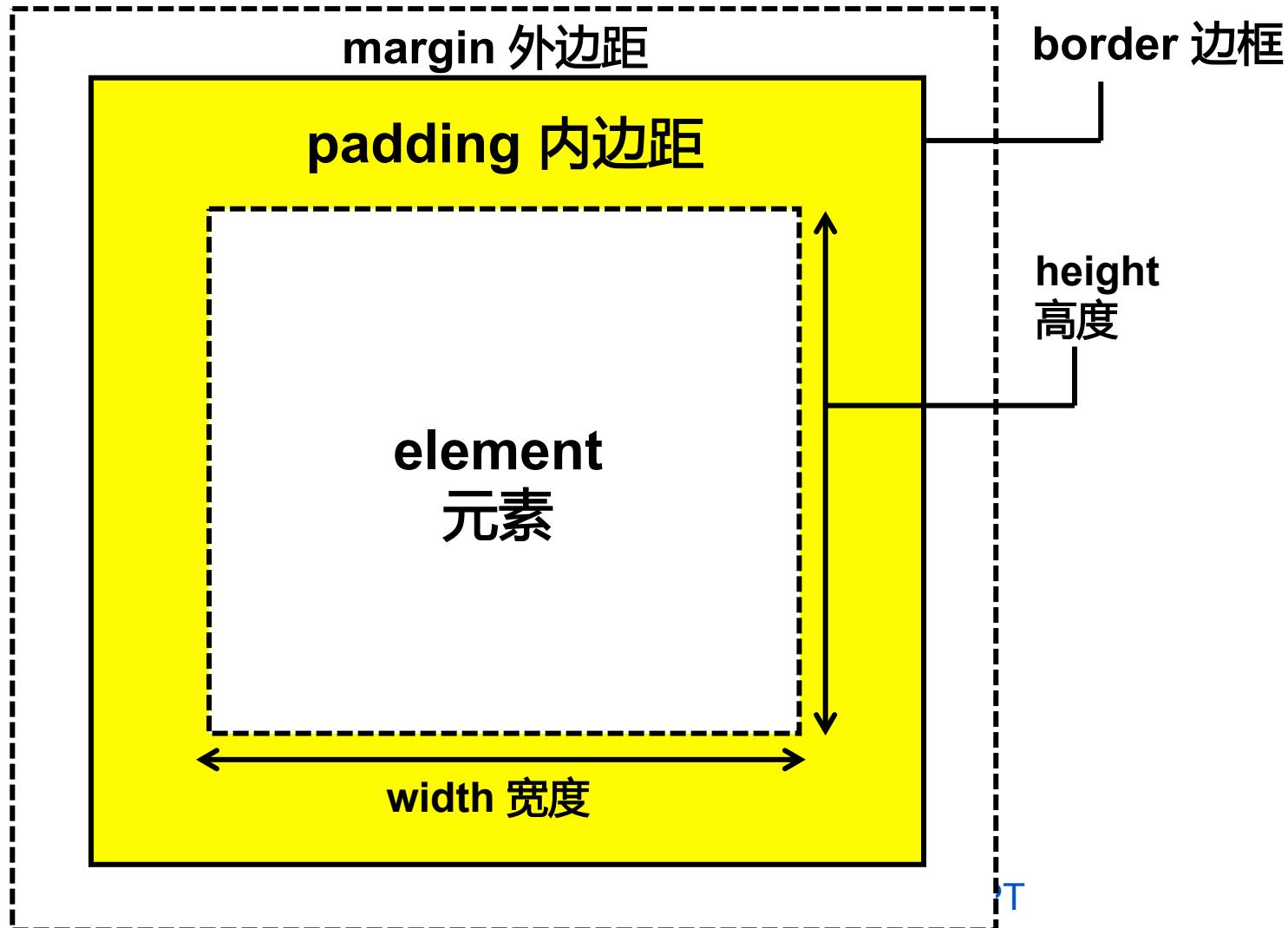
---

- CSS 框模型 (Box Model)
- 规定了元素框处理元素内容、内边距、边框和外边距的方式。

# CSS框模型概述



# 内边距



# CSS内边距

---

- 元素的内边距在边框和内容区之间。控制该区域最简单的属性是 **padding** 属性。
- CSS padding 属性定义元素边框与元素内容之间的空白区域。
- padding 属性接受长度值或百分比值，但不允许使用负值。

# CSS内边距

- CSS内边距属性

列表属性	功能	取值方式
padding-top	上内边距	length : 规定以具体单位计的固定的上内边距值，比如像素、厘米等。默认值是 0px。% : 定义基于父元素 <b>宽度</b> 的百分比上内边距。
padding-right	右内边距	同上
padding-bottom	下内边距	同上
padding-left	左内边距	同上
padding	简写属性	作用是在一个声明中按上述顺序设置元素的所有内边距属性。

# CSS 内边距

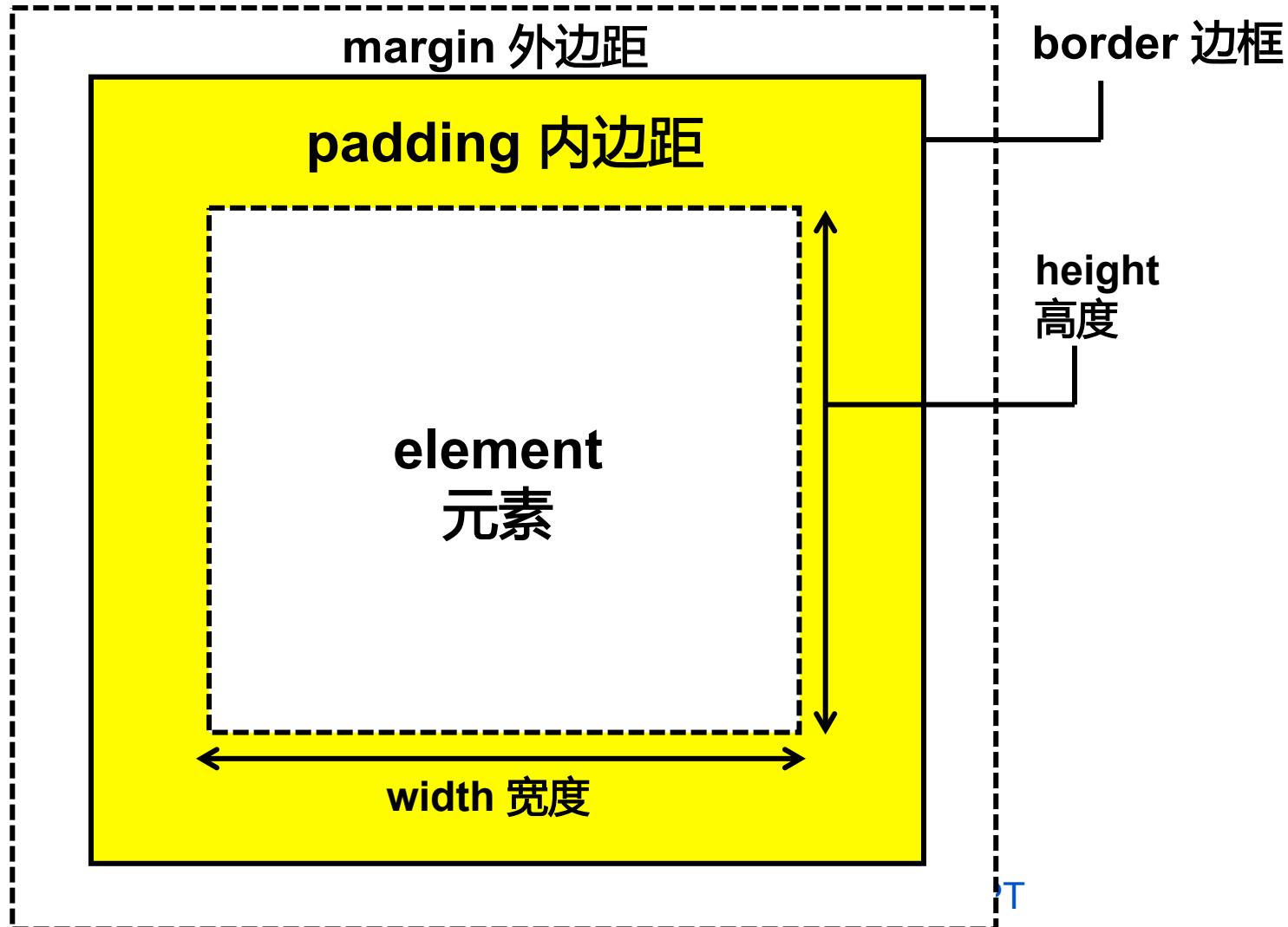
---

- **内边距**

- 内边距复合属性padding的设置有4个参数，根据赋值个数的不同，会有以下几种情况：

- 如果在设定内边距时提供四个参数，将按上→右→下→左的顺时针顺序作用于四个边距。
    - 如果只提供一个参数，则应用于四个边距。
    - 如果提供两个参数，第一个用于上、下边距，第二个用于左、右边距。
    - 如果提供三个参数，第一个用于上边距，第二个用于左、右边距，第三个用于下边距。

# 边框



# CSS边框

---

- 元素的边框 (**border**) 是围绕元素内容和内边距的一条或多条线。
- 边框属性用来设置对象边框的颜色、样式和宽度。
- 边框与背景**
  - CSS 规范指出，边框绘制在 “元素的背景之上”。这很重要，因为有些边框是 “中断的”（例如，点线边框或虚线框），元素的背景应当出现在边框的可见部分之间。
  - CSS2.1 指出：元素的背景是内容、内边距和边框区的背景。大多数浏览器都遵循 CSS2.1 定义，不过一些较老的浏览器可能会有不同的表现。

# CSS边框

---

- **边框颜色**

- 默认的边框颜色是元素本身的**前景色**。
- 如果没有为边框声明颜色，它将与元素的文本颜色相同。
- 如果元素没有任何文本，假设它是一个表格，其中只包含图像，那么该表的边框颜色就是其父元素的文本颜色（因为 color 可以继承）。这个父元素很可能就是 body、div 或另一个 table。

# CSS边框

- CSS边框颜色属性

列表属性	功能	取值方式
border-top-color	上边框	color_name : 颜色值为颜色名称 ( red )。 hex_number : 颜色值为十六进制值 ( #ff0000 )。 rgb_number : 颜色值为 rgb 代码 ( rgb(255,0,0) )。 Transparent : 默认值。透明。
border-right-color	右边框	同上
border-bottom-color	下边框	同上
border-left-color	左边框	同上
border-color	简写属性	作用是在一个声明中按上述顺序设置元素的所有边框颜色属性。

# CSS边框

---

- **边框颜色**

- 复合属性border-color的设置有4个参数，根据赋值个数的不同，会有以下几种情况：

- 如果在设定颜色时提供四个颜色参数，将按上→右→下→左的顺序作用于四个边框。
    - 如果只提供一个颜色参数，则应用于四个边框。
    - 如果提供两个参数，第一个用于上、下边框，第二个用于左、右边框。
    - 如果提供三个参数，第一个用于上边框，第二个用于左、右边框，第三个用于下边框。

# CSS边框

---

- **边框样式**

- 用于设定边框的样式 ( border-style ) 。边框样式同样有4个参数，赋值方式与边框颜色相同。
  - 同样可以使用单边边框样式属性来设置某一个边框的样式
    - Border-top-style
    - Border-bottom-style
    - Border-left-style
    - Border-right-style

# CSS边框

- 边框样式

边框样式值	说明	边框样式值	说明
none	无边框	Grove	根据border-color的值画3D凹槽
hidden	隐藏边框	Ridge	根据border-color的值画菱形边框
dotted	点线边框	Inset	根据border-color的值画3D凹边
dashed	虚线边框	Outset	根据border-color的值画3D凸边
Solid	实线边框		
Double	双线边框		

# CSS边框

---

- **边框宽度**

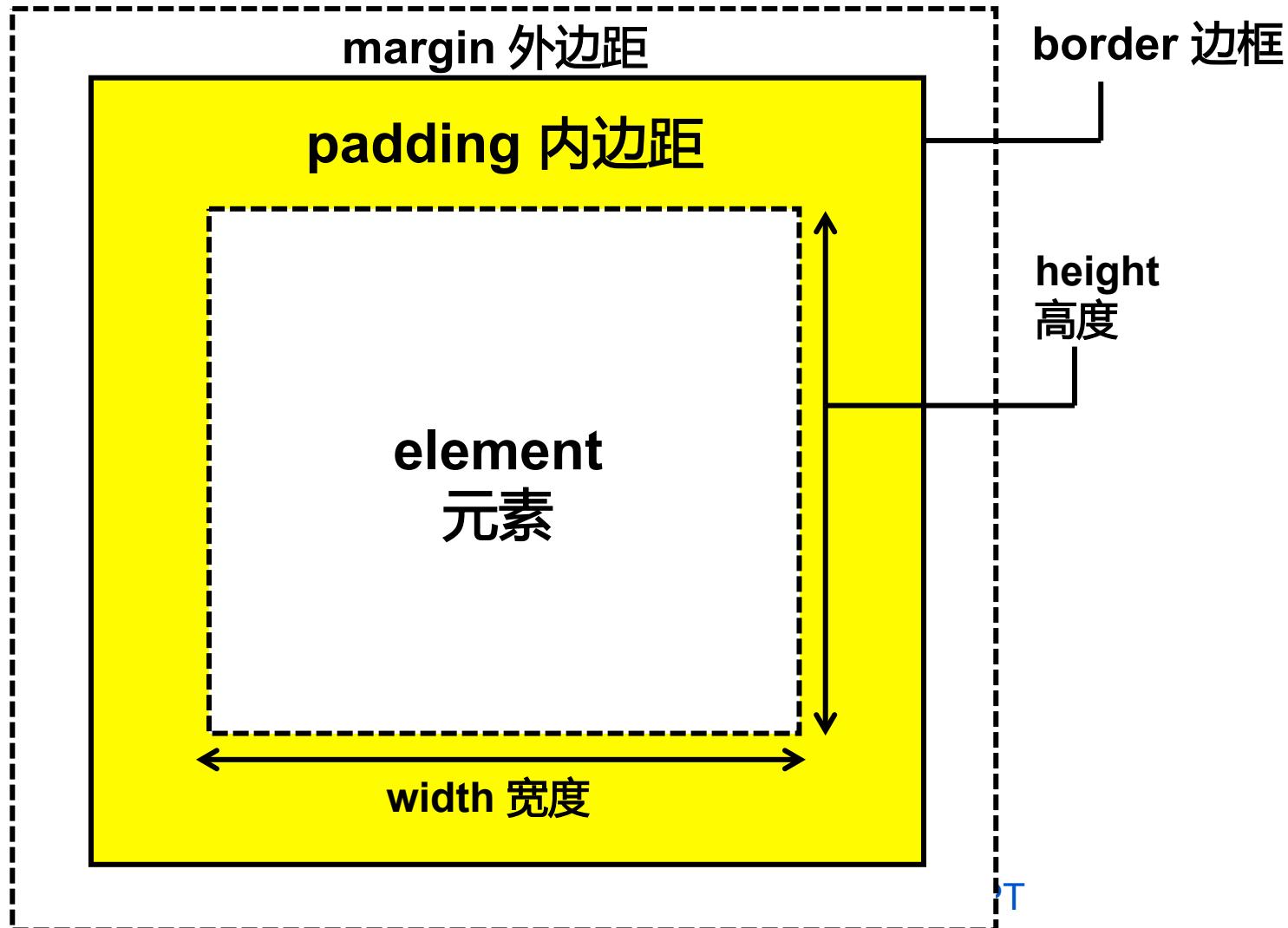
- 用于设定边框的宽度（ border-width ），宽度的取值为关键字或自定义的数值。
    - Medium : 默认宽度
    - Thin : 小于默认宽度
    - Thick : 大于默认宽度
  - 边框宽度同样有四个参数需要赋值，赋值方式与边框颜色相同。

# CSS边框

---

- border属性是复合属性，设置对象的边框样式。
- 语法：border : border-width border-style border-color
- 默认值为：medium none border-color 的默认值将采用文本颜色。
- 注：
  - 指定了边框颜色(border-color)和边框粗细( border-width )时，必须同时指定边框样式( border-style )，否则边框不会被呈现。
  - 如使用该复合属性定义其单个参数，则其他参数的默认值将无条件覆盖各自对应的单个属性设置。  
例如：设置 border : thin 等于设置 border : thin none ，而 border-color 的默认值将采用文本颜色。因此此前的任何 border-color 和 border-width 设置都会被清除。

# 外边距



# CSS外边距

---

- 围绕元素边框的空白区域是外边距。设置外边距会在元素外创建额外的“空白”。
- 设置外边距的最简单的方法就是使用 `margin` 属性，这个属性接受任何长度单位、百分数值甚至负值。
- 外边距同样有四个参数需要赋值，赋值方式与边框颜色相同。

# CSS外边距

- CSS外边距属性

列表属性	功能	取值方式
margin-top	上边距	Auto : 浏览器计算下外边距。Length : 以具体单位计的下外边距值，如像素、厘米等。默认值是 0px。 % : 基于父元素的宽度的百分比的下外边距。
margin-right	右边距	同上
margin-bottom	下边距	同上
margin-left	左边距	同上
margin	简写属性	作用是在一个声明中按上述顺序设置元素的所有外边距属性。

# CSS外边距

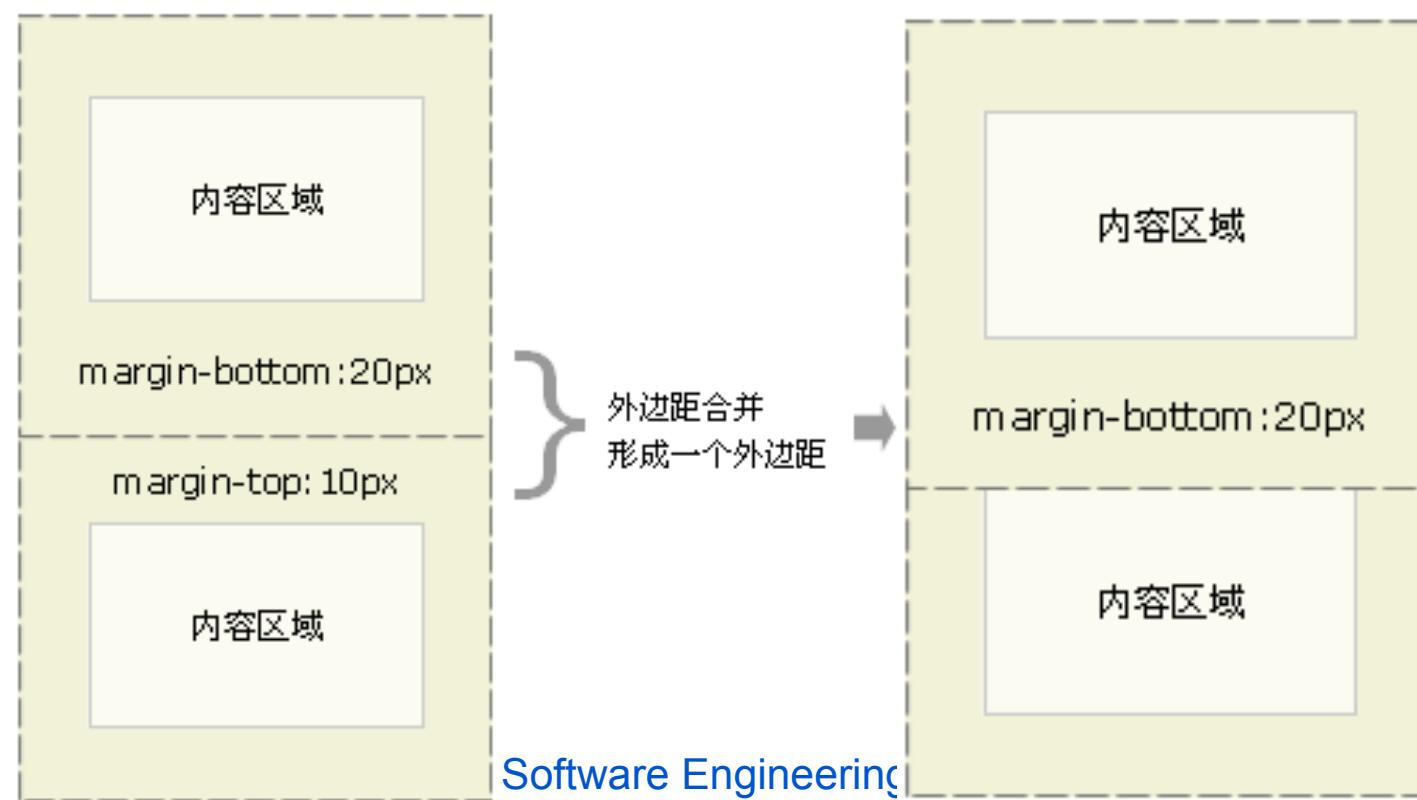
---

- **CSS外边距合并：**
- 外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。
- 合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

# CSS外边距

## • CSS外边距合并实例

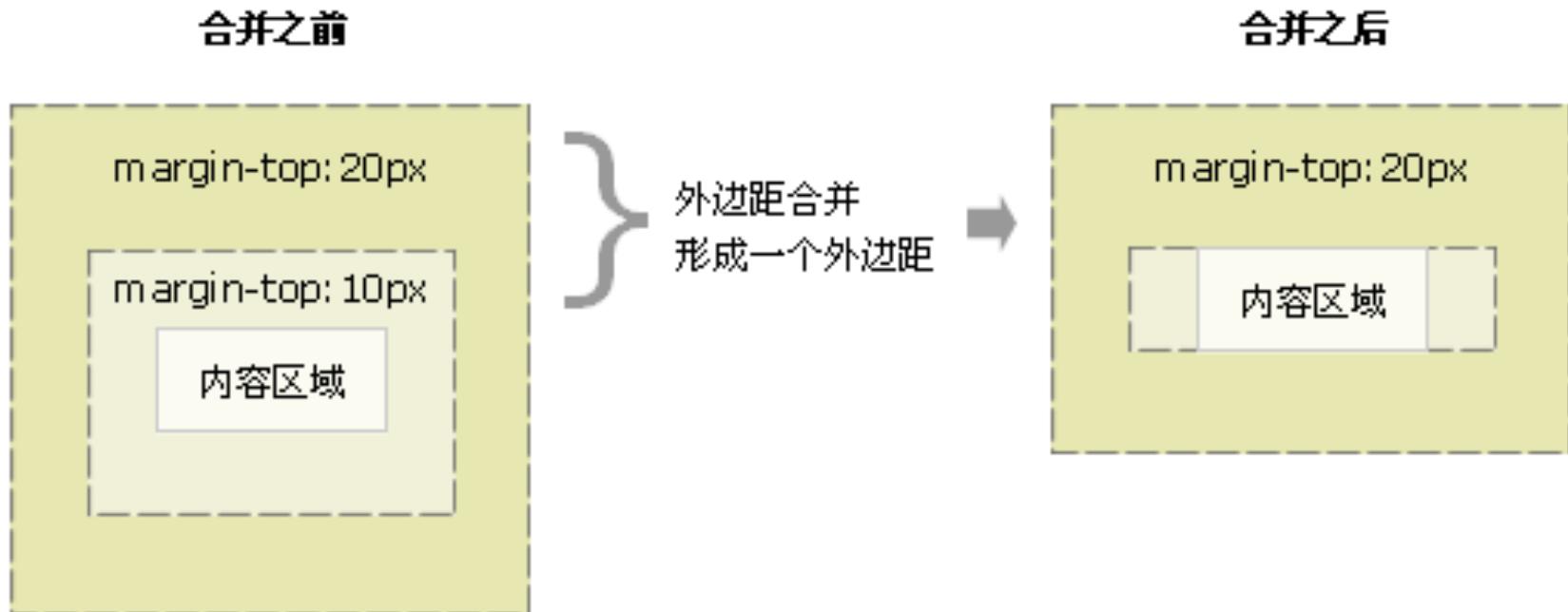
当一个元素出现在另一个元素上面时，第一个元素的下外边距与第二个元素的上外边距会发生合并。



# CSS外边距

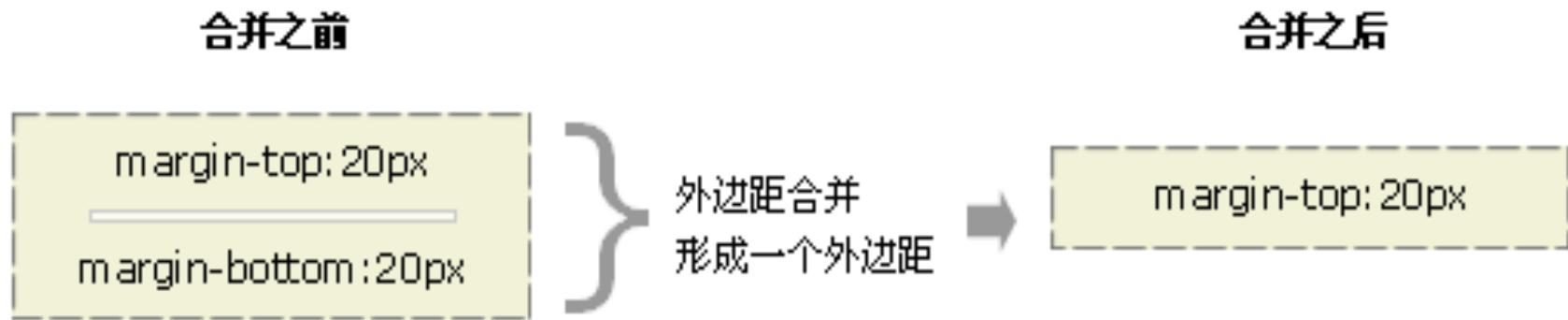
- CSS外边距合并实例

当一个元素包含在另一个元素中时，它们的上和/或下边距也会发生合并。



# CSS外边距

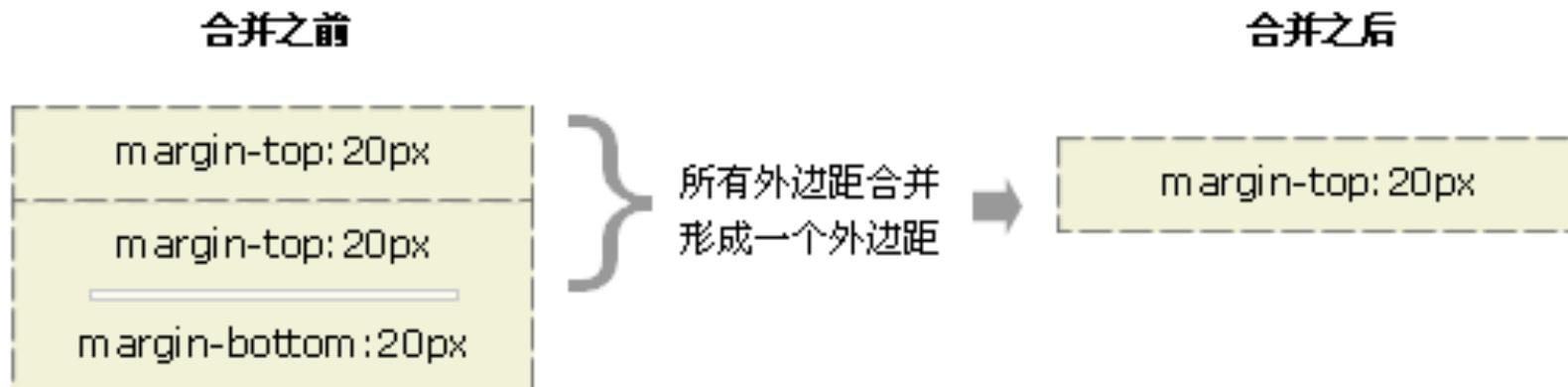
- CSS外边距合并实例



假设有一个空元素，它有外边距，但是没有边框或填充。在这种情况下，上外边距与下外边距就碰到了一起，它们会发生合并。

# CSS外边距

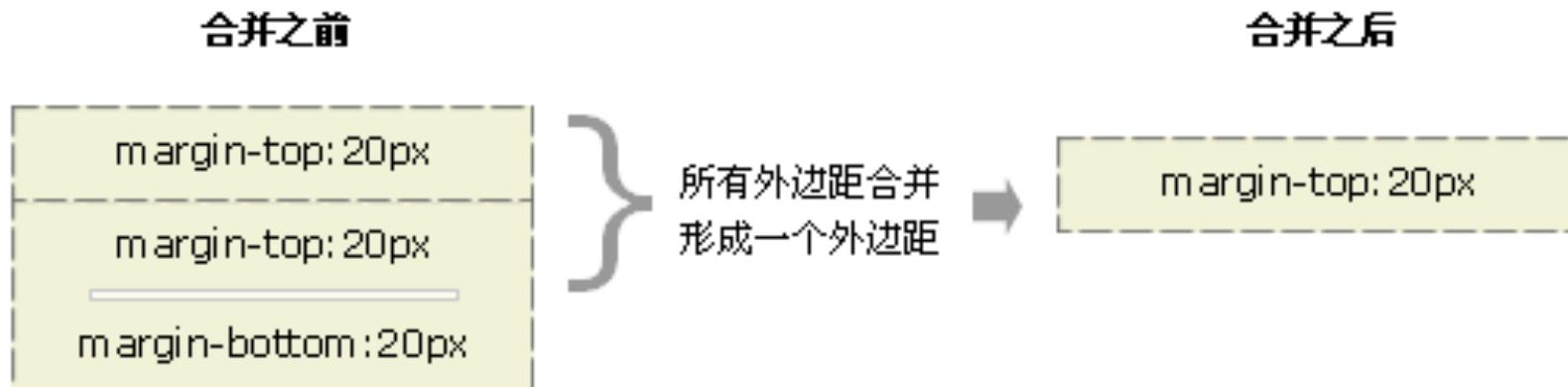
- CSS外边距合并实例



如果这个外边距遇到另一个元素的外边距，它还会发生合并。

# CSS外边距

- CSS外边距合并实例



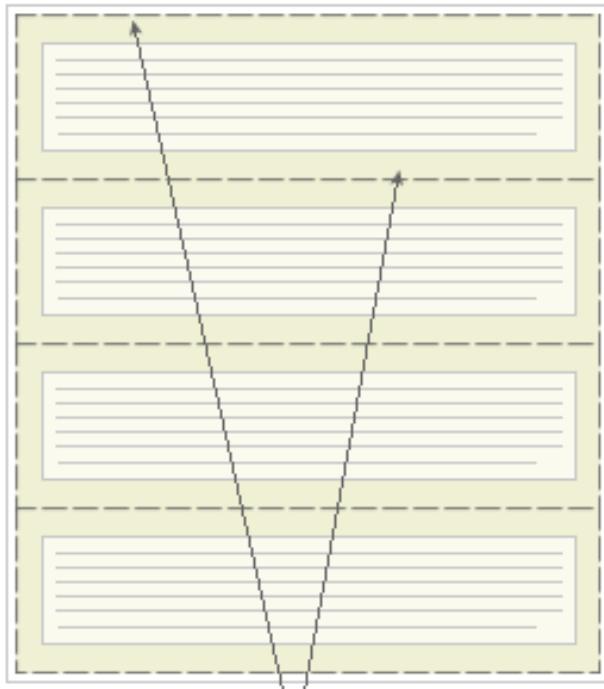
如果这个外边距遇到另一个元素的外边距，它还会发生合并。

这就是一系列的段落元素占用空间非常小的原因，因为它们的所有外边距都合并到一起，形成了一个小的外边距。

# CSS外边距

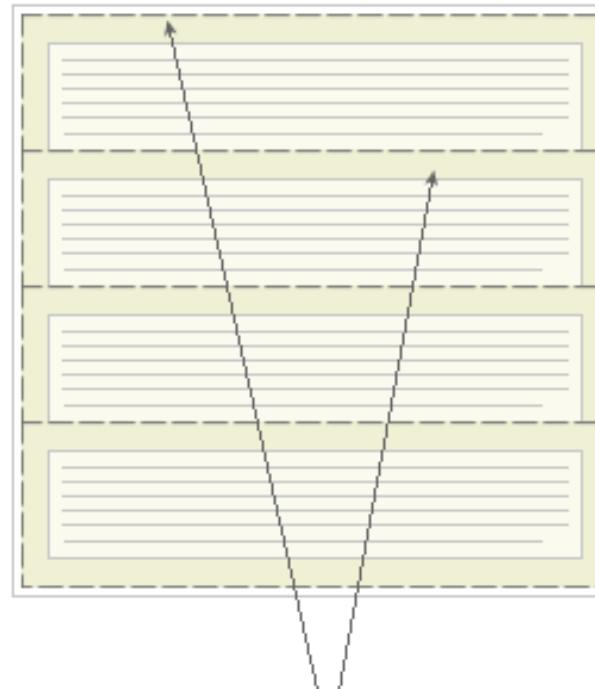
- CSS外边距合并实例

没有外边距合并



段落之间的外边距是上外边距的两倍

有外边距合并



段落之间的外边距与上外边距相同

注：只有普通文档流中块框的垂直外边距才会发生外边距合并。行内框、浮动框或绝对定位之间的外边距不会合并。

# CSS样式表

---

- 认识CSS 样式
- CSS带来的好处
- CSS概述
  - CSS样式说明
  - CSS样式规则
  - CSS样式特征
  - CSS样式的分类
  - CSS样式的注释
- 常用的样式属性及值
- CSS框模型
- CSS定位
- CSS浮动
- 页面布局

# CSS定位

---

- CSS允许对元素进行定位
  - CSS 为定位和浮动提供了一些属性，利用这些属性，可以建立列式布局，将布局的一部分与另一部分重叠。
  - 定位的基本思想：允许定义元素框相对于其正常位置应该出现的位置，或者相对于父元素、另一个元素甚至浏览器窗口本身的位置。

# CSS定位

---

- **HTML标签的分类：**

- 块级元素 ( Block-level Elements )

- 块级元素在显示时独占一行，即使是同级的块级元素也只能新起一行进行显示。
    - 一般是其它元素的容器。
    - 特点：总是另起一行开始；高度，行高以及顶、底边距都可控制；宽度缺省是它所在容器的100%，除非设定一个宽度。
    - 例如：div、h1~h6、p、table等。

- 内联元素 ( Inline-level Elements )，也称行内元素

- 行级元素在显示时不独占一行，同级的内联元素在一行内接着显示。
    - 特点：和其它元素都在一行上；高度，行高以及顶、底边距不可改变；宽度就是它所容纳的文字或图片的宽度，不可改变。
    - 例如：a、b、br、input等。

# CSS定位

## 块元素实例：

- <html>
- <head>
- </head>
- <body>

**第一块** • <h1>This is a Heading</h1>

- <div>
- 
- </div>

**第二块** • <p>一些文本。一些文本。一些文本。一些文本。一些文本。</p>

- </body>
- </html>

This is a Heading



一些文本。一些文本。一些文本。一些文本。

# CSS定位

---

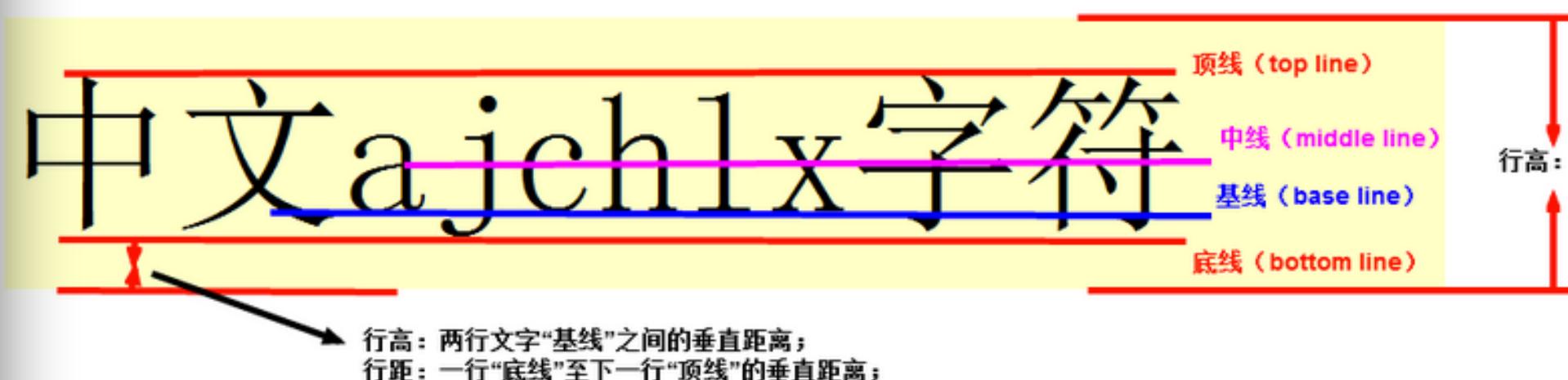
- **一切皆为框**
  - 块级元素，即“块框”。
  - 行内元素或内联元素，即“行内框”。

# CSS定位

---

- **CSS 定位机制：**
  - 三种基本的定位机制：普通流、浮动和绝对定位。
  - 除非专门指定，否则所有框都在普通流中定位。即，元素的位置由元素在 HTML 中的位置决定。
  - 块级元素从上到下一个接一个地排列，元素之间的垂直距离是由元素的垂直外边距计算出来。
  - 行内元素在一行中水平布置。可以使用水平内边距、边框和外边距调整它们的间距。但是，垂直内边距、边框和外边距不影响行内框的高度。
    - 由一行形成的水平框称为行框（Line Box），行框的高度总是足以容纳它包含的所有行内元素。不过，设置行高可以增加这个框的高度。

# CSS定位



行高 line-height

行高50px(`line-height:50px;`)。行高(`line-height`)指的是文本  
行的基线(`base-line`)间的距离。

行距

行高line-height

<http://www.ddcat.net>

# CSS定位

---

- 定位属性

- 定位属性主要从定位方式、层叠顺序、与父标签的相对位置等三个方面来设置。

# CSS定位

定位属性	功能	取值方式
position	定位方式，设置对象是否定位以及定位的方式	<ul style="list-style-type: none"><li>static : 无特殊定位；</li><li>relative : 对象不可层叠，但将依据left , right , top , bottom等属性在正常文档流中偏移位置；</li><li>absolute : 将对象从文档流中拖出，使用left , right , top , bottom等属性进行绝对定位；</li><li>fixed : 固定定位，使用left , right , top , bottom等属性进行固定定位。</li></ul>
z-index	设置对象的层叠顺序	<ul style="list-style-type: none"><li>auto : 遵循其父对象的定位；</li><li>自定义数值 : 无单位的整数值，可为负值</li></ul>
top/right/ bottom/left	相对父对象的位置	<ul style="list-style-type: none"><li>auto : 无特殊定位；</li><li>自定义数值 : 由浮点数字和单位标识符组成的长度值，或者百分数。</li></ul> <p>必须定义position属性值为absolute、fixed或者relative此取值方可生效</p>

# CSS定位

---

- 相对定位 (position:relative)

- 设置为相对定位的元素框会偏移某个距离。元素仍然保持其未定位前的形状，它原本所占的空间仍保留。
  - 如果对一个元素进行相对定位，它将出现在它所在的位置上。然后，可以通过设置垂直或水平位置，让这个元素相对于它的起点进行移动。

# CSS定位实例

- <html>
- <head>
- <style type="text/css">
- h2.pos\_left{position:relative;left:-20px}
- h2.pos\_right{position:relative;left:20px}
- </style>
- </head>
- <body>
- <h2>正常位置的标题</h2>
- <h2 class="pos\_left">相对于正常位置向左移动</h2>
- <h2 class="pos\_right">相对于正常位置向右移动</h2>
- </body>
- </html>

正常位置的标题

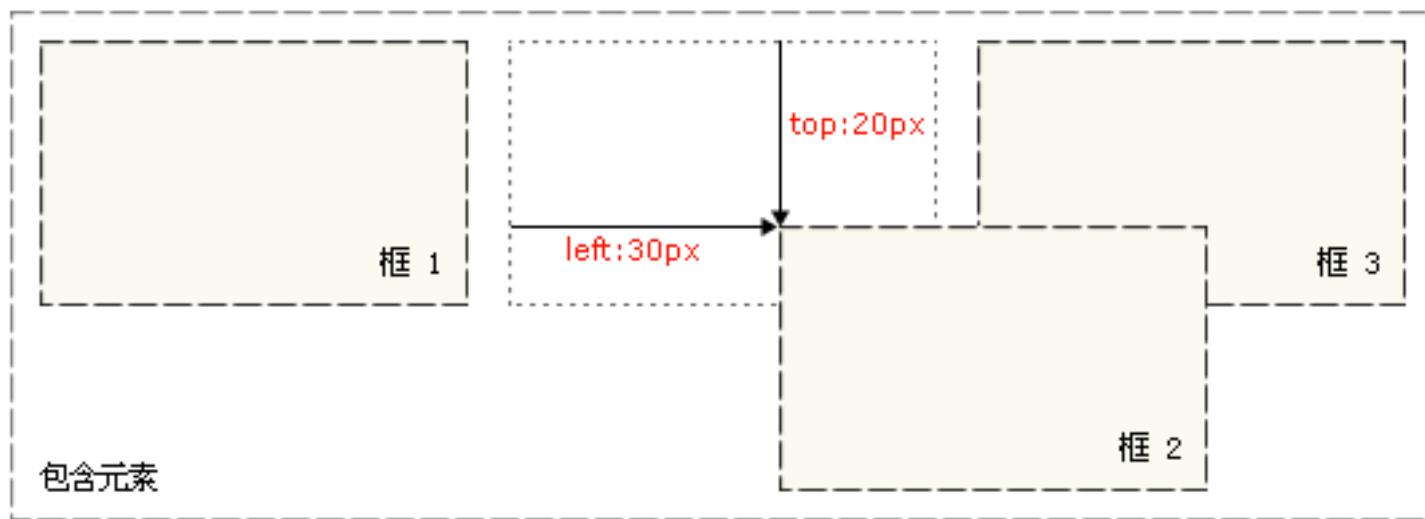
相对于正常位置向左移动

相对于正常位置向右移动

- 相对定位会按照元素的原始位置对该元素进行移动。
- 样式 "left:-20px" 从元素的原始左侧位置减去 20 像素。
- 样式 "left:20px" 向元素的原始左侧位置增加 20 像素。

# CSS定位

- `#box_relative {`
- `position: relative;`
- `left: 30px;`
- `top: 20px;`
- `}`



注意，在使用相对定位时，无论是否进行移动，元素仍然占据原来的空间。因此，移动元素会导致它覆盖其它框。

# CSS定位实例

- <html>
- <head>
- </head>
- <body>
- <h1>This is a Heading</h1>
- <div>
- <b>div开始</b>
- 
- <b>div结束</b>
- </div>
- <p>一些文本。一些文本。一些文本。一些文本。</p>
- </body>
- </html>

This is a Heading



div开始

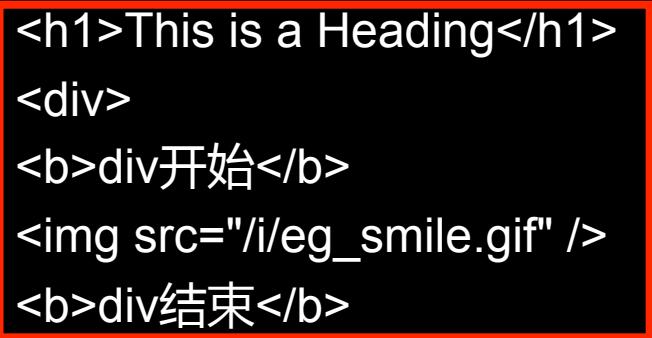
div结束

一些文本。一些文本。一些文本。一些文本。

- Div内的三个内联元素排列在同一行上显示。
- 注意看“div结束”这三个字对应的位置（与下面的文本的相对位置）

# CSS定位实例

- <html>
- <head>
- <style type="text/css">
-  

```
img{position:relative;left:-20px}
```
- </style>
- </head>
- <body>
- <h1>This is a Heading</h1>
- <div>
- <b>div开始</b>
-  

```

```
- <b>div结束</b>
- </div>
- <p>一些文本。一些文本。一些文本。一些文本。</p>
- </body>
- </html>

This is a Heading



div开...

div结束

一些文本。一些文本。一些文本。一些文本。

- Div内的第二个内联元素img的position属性设置为relative，因此，该元素应该相对于它本该出现的位置向左偏移20px。
- 注意：虽然该元素产生了偏移，但它在原本的文档流中所占的空间仍然保留。（注意看“div结束”这三个字对应的位置，没发生向左偏移的情况。）

# CSS定位

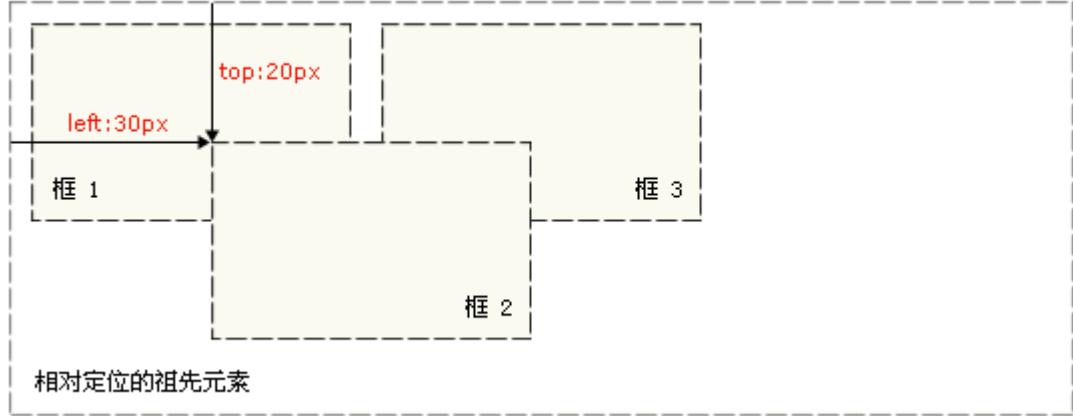
---

- **绝对定位 (position:absolute)**

- 设置为绝对定位的元素框从文档流完全删除，并相对于其包含块定位。
- 包含块可能是文档中的另一个元素或者是初始包含块。
- 元素原先在正常文档流中所占的空间会关闭，就好像该元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框。

# CSS定位

- `#box_absolute {`
- `position: absolute;`
- `left: 30px;`
- `top: 20px;`
- `}`



- 注意，绝对定位的元素的位置相对于最近的已定位祖先元素，如果元素没有已定位的祖先元素，那么它的位置相对于最初的包含块。
- 根据用户代理的不同，最初的包含块可能是画布或 HTML 元素。
- 因为绝对定位的框与文档流无关，所以它们可以覆盖页面上的其它元素。可以通过设置 `z-index` 属性来控制这些框的堆放次序。

# CSS定位实例

- <html>
- 通过绝对定位(**absolute**)，元素可以放置到页面上的任何位置。
- <head>
- <style type="text/css">
- h2.pos\_abs{position:absolute;left:100px;top:150px}
- </style>
- </head>
- <body>
- <h2 class="pos\_abs">这是带有绝对定位的标题</h2>
- <p>下面的标题距离页面左侧 100px，距离页面顶部 150px。</p>
- </body>
- </html>

下面的标题距离页面左侧 100px，距离页面顶部 150px。

# CSS定位

---

- **固定定位 (position:fixed)**
  - 元素框的表现类似于将 position 设置为 absolute, 不过其包含块是视窗本身。

# CSS定位实例

- <html>
- 通过固定定位(fixed) , 元素可以放置到页面上的任何位置。
- <head>
- <style type="text/css">
- p.one{position:fixed;left:5px;top:5px;}
- p.two{position:fixed;top:30px;right:5px;}
- </style>
- </head>
- <body>
- <p class="one">第一行文本</p>
- <p class="two">第二行文本</p>
- </body>
- </html>

第一行文本

第二行文本

# CSS定位

---

- **absolute与fixed的区别：**
- **参照位置不同**
- **absolute**：参照位置是离当前元素最近的并且定位方式为fixed,absolute,relative的祖先元素的左上角。
- **fixed**：参照位置是浏览器窗口的左上角，即坐标点为(0px, 0px)

# CSS定位实例

```
• <html>
•   <head>
•     <style type="text/css">
•       img{position:fixed;top:20px}
•     </style>
•   </head>
•   <body>
•     <h1>This is a Heading</h1>
•     <div>
•       
•     </div>
•     <p>一些文本。一些文本。一些文本。一些文本。</p>
•   </body>
• </html>
```



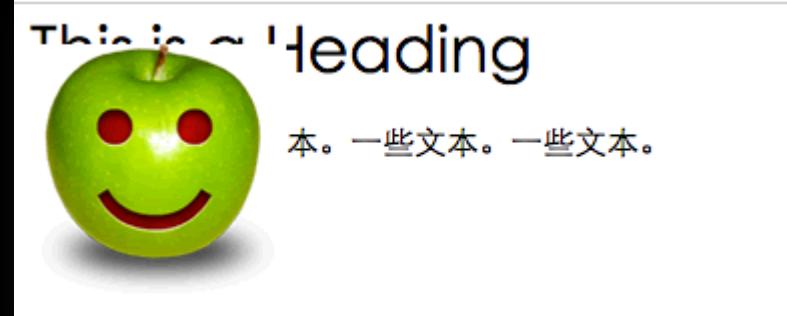
- 由于给标签指定了position为fixed，因此，该元素将相对于浏览器左上角定位，这里距离上边距20px。
- 同时该元素在原来的正常文档流中所占空间会被关闭，就好像元素不存在一样。

# CSS定位实例

- <html>
- <head>
- <style type="text/css">

```
img{position:absolute;top:20px}
```
- </style>
- </head>
- <body>

```
<h1>This is a Heading</h1>
<div>
  
</div>
<p>一些文本。一些文本。一些文本。一些文本。</p>
```
- </body>
- </html>
- 由于给img标签指定了position为absolute，因此，该元素将相对于其父元素进行偏移。
- 但是离img元素最近的父元素div并没有设置position属性，即，该div的position属性值为默认值static，不满足相对定位条件，img会继续向上级父标签寻找符合条件的标签。
- 再上级是body，这时已经是相对于浏览器的位置了。



# CSS定位实例

- <html>
  - <head>
  - <style type="text/css">

```
img{position:absolute;top:25px}
div{position:fixed;top:110px}
```
  - </style>
  - </head>
  - <body>
  - <h1>This is a Heading</h1>
  - <div>

```
<p>div开始</p>

</div>
```
  - <p>一些文本。一些文本。一些文本。一些文本。一些文本。</p>
  - </body>
  - </html>
- This is a Heading
- 一些文本。一些文本。一些文本。一些文本。
- A green apple with a smiling face, positioned at the top of the page.
- 离img元素最近的父元素div设置position属性为fixed，满足相对定位条件
  - img会将该div标签的位置作为其相对位置，上边距偏移25px。

# CSS定位

---

- **Z-index属性**：设置元素的堆叠顺序。拥有更高堆叠顺序的元素总是会处于堆叠顺序较低的元素的前面。
- **说明：**
  - 元素可拥有负的 z-index 属性值。
  - Z-index 仅能在定位元素上奏效(如 position:absolute)
  - 该属性设置一个定位元素沿 z 轴的位置，z 轴定义为垂直延伸到显示区的轴。如果为正数，则离用户更近，为负数则表示离用户更远。

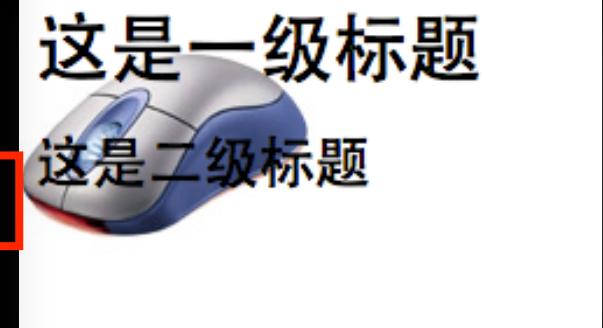
# CSS定位

---

- **Z-index属性取值：**
  - auto 默认。堆叠顺序与父元素相等。
  - number 设置元素的堆叠顺序。
  - inherit 规定应该从父元素继承 z-index 属性的值。

# CSS定位实例

- <html>
- <head>
- <style type="text/css">
- 
- </style>
- </head>
- <body>
- <h1>这是一级标题</h1>
- <h2>这是二级标题</h2>
-  style="border: 2px solid red; width: 100%; height: 100%;"/>
- </body>
- </html>



- **Img**标签设置了相对位置，这里满足条件的标签就是 **body**，因此，图片从坐标 ( 0,0 ) 开始渲染。
- 由于**z-index**属性设置了-1，因此比使用默认值0的其它元素优先级更低，处于它们的Z轴离用户更远的下方。

# CSS样式表

---

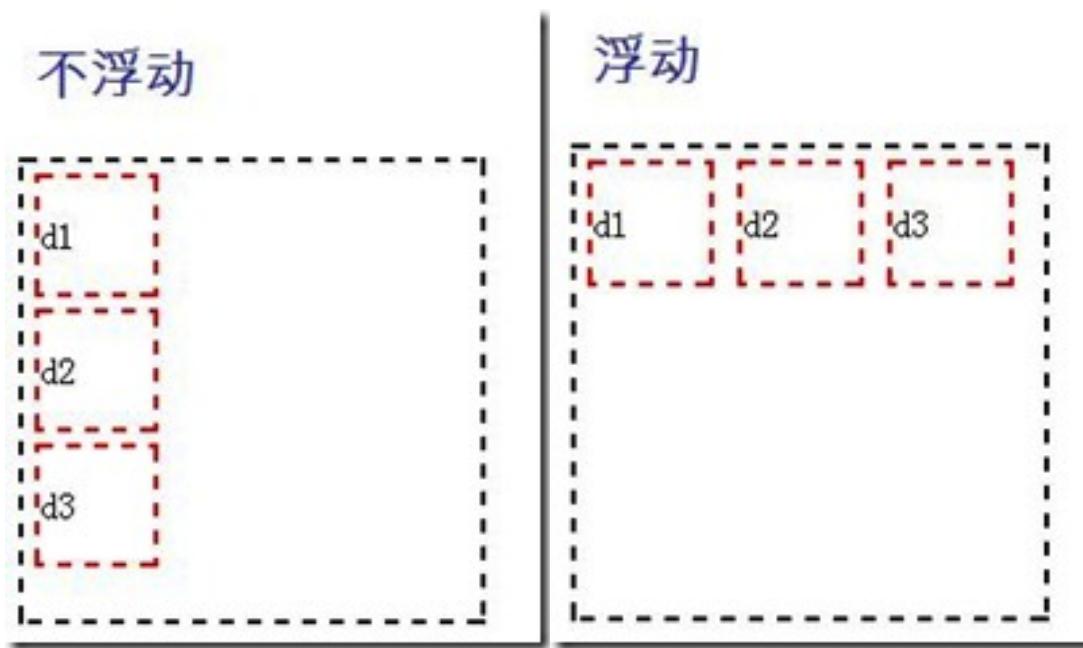
- 认识CSS 样式
- CSS带来的好处
- CSS概述
  - CSS样式说明
  - CSS样式规则
  - CSS样式特征
  - CSS样式的分类
  - CSS样式的注释
- 常用的样式属性及值
- CSS框模型
- CSS定位
- CSS浮动
- 页面布局

# CSS浮动

---

- 浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。
- 由于浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。
- float取值：
  - left：对象浮在左边
  - right：对象浮在右边
  - none：对象不浮动

# CSS浮动

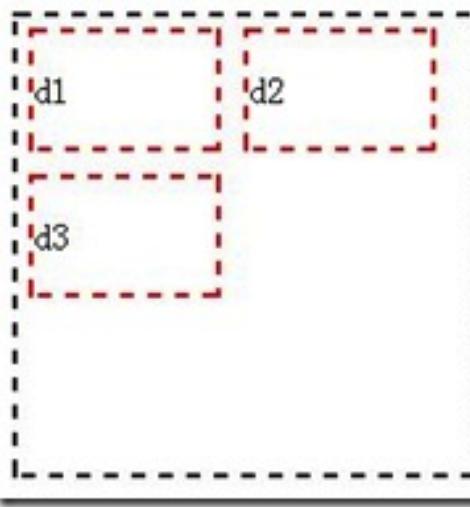


- 浮动与不浮动主要的区别是不浮动的框是由上往下各自独占一行的，而浮动的框是紧靠着排列的。
- 右图中d1、d2、d3都是向左浮动的，所以先把d1放在紧靠着父框摆放，然后把d2紧靠着d1摆放，最后d3紧靠着d2摆放。向右浮动亦与之类似。

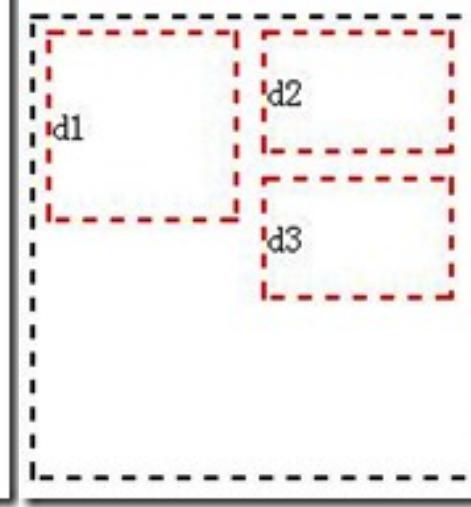
# CSS浮动

- 调整：

下移



卡住

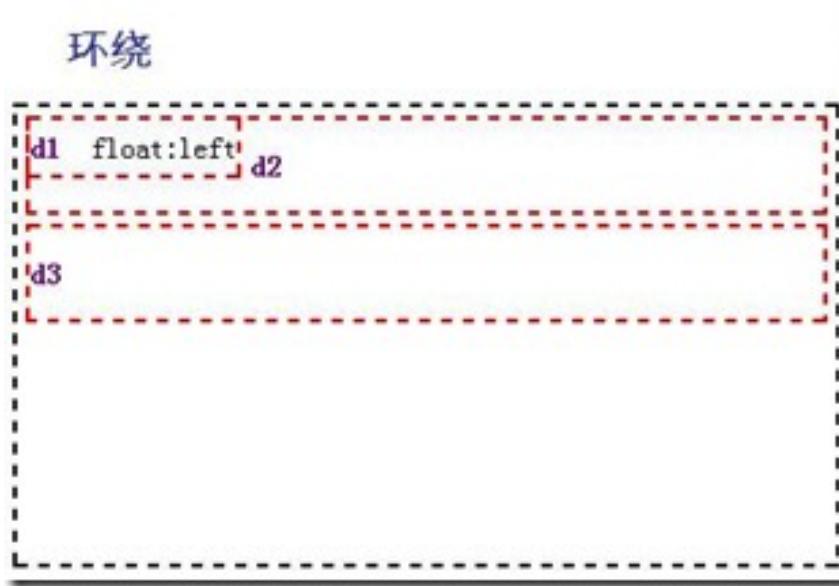


- d1、d2、d3均向左浮动。
- 左图，如果包含框太窄，无法容纳水平排列的三个浮动元素，那么其它浮动块向下移动，直到有足够的空间。
- 右图，如果浮动元素的高度不同，那么当它们向下移动时可能被其它浮动元素“卡住”。

# CSS浮动

- 环绕与清除：

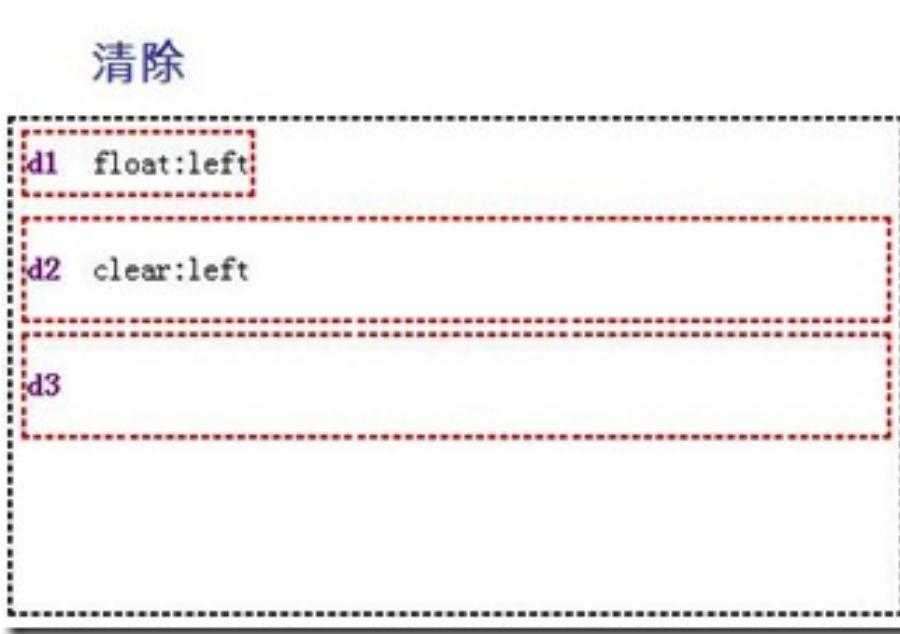
环绕



- d1向左浮动，d2、d3都不浮动。
- 浮动的框的是脱离普通流的，即d1浮在上面，下面的d2、d3感觉不到d1的存在。所以d2在父框中处于顶端。而且d2中的内容不会覆盖d1中的内容，
- 可以利用这一点做出环绕效果。比如d1中放的是一张图片，d2中是图片的解释。

# CSS浮动

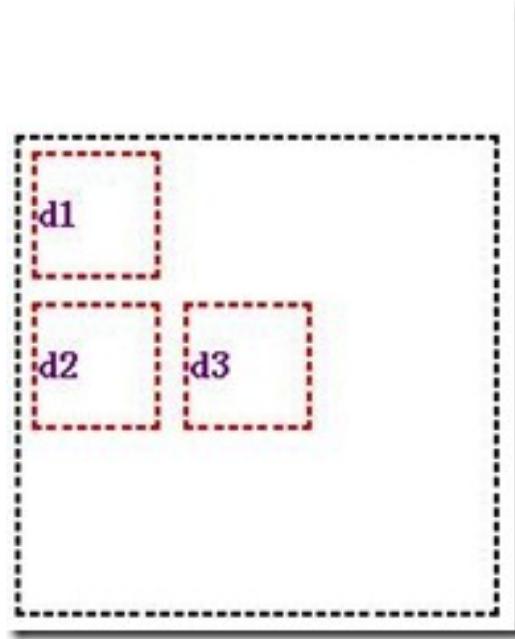
- 环绕与清除：



- d1向左浮动，d2、d3不浮动，且d2的clear属性为left。
- 如果d1与d2是不相关的，就想要d2另起一行。那就要使用另一个CSS属性clear。**clear属性用于设置框的左边或右边或两边不挨着浮动框。**
- Clear取值：none：允许两边都可以有浮动对象；both：不允许有浮动对象；left：不允许左边有浮动对象；right：不允许右边有浮动对象。

# CSS浮动

- 环绕与清除：



- d1不浮动 , d2 , d3向左浮动
- 虽然不浮动的框感觉到不到浮动框的存在 , 但浮动框却知道不浮动框的存在。如上图 , d1是不浮动的 , d2、d3都是向左浮动的 , d2能够另起一行而不覆盖到d1。 (浮动直到外边缘碰到包含框或者另一个浮动框的边框为止)

# CSS样式表

---

- 认识CSS 样式
- CSS带来的好处
- CSS概述
  - CSS样式说明
  - CSS样式规则
  - CSS样式特征
  - CSS样式的分类
  - CSS样式的注释
- 常用的样式属性及值
- CSS框模型
- CSS定位
- CSS浮动
- **页面布局**

# 页面布局

---

- 一个标准的web页面由结构、外观和行为3部分组成：
  - 结构：用来对网页中的信息进行整理与分类，常用技术HTML
  - 外观：用于对已经被结构化的信息进行外观上的修饰，包括颜色、字体等，常用技术CSS
  - 行为：对整个文档内部的一个模型进行定义及交互行为的编写，常用技术JS
- 网页设计的核心目的：实现网页结构和外观的分离。

# 页面布局

---

- 页面布局是将网页中的各个版块有效组织并放置在合适的位置。
- 页面布局一般分为：
  - 表格布局
  - 框架布局
  - DIV + CSS布局

# 页面布局

---

- **DIV + CSS布局**
- 考虑的首要内容：将页面内容的语义或结构确定下来
- 结构或语义：将所要设计网页中的内容分成块，明确每块内容服务的目的，根据这些内容的目的建立起相应的HTML结构。
- 最大的特色：真正意义上的结构和外观的分离

# 页面布局

整个页面放到一个大的容器内



页面头部包括了：

- LOGO
- MENU
- Banner

会员登录

用户名： 密码：

登录

站点信息

访问：6562 次  
在线：12 人  
会员：9844 人  
留言：3236 个  
注册：2005/03/20

页面标准

风格名称：Red Vision  
设计作者：郑木头  
CSS制作：看云想了  
发布时间：2005/11/04  
展示地址：<http://www.tblog.com.cn>  
风格简介：此风格设计来自网友“郑木头”，由PUBLOG团队  
《爱折腾、爱秀》共同完成，在此感谢“郑木头”  
提供的设计风格。

点击下载该风格

风格名称：Dawn(黎明)  
设计作者：深海红了  
CSS制作：看云想了  
发布时间：2005/09/19  
展示地址：<http://www.tblog.com.cn>  
风格简介：本SKIN适用于PUBLOG2 V2.3版本。  
本SKIN的测试完全兼容IE, Mozilla Firefox浏览器。  
建议在1024X768下浏览。

点击下载该风格

Copyright 2005 Tblog.com.cn All Rights Reserved xhtml.css  
津ICP备06000799号

页面内容可分为左中右三区，或左右两区。

T 页面底部：版权信息

# 页面布局

The screenshot shows a website with the following structure:

- Header 层 (Blue):** Contains the logo "Design From Tblog.com.cn" (featuring a tomato icon), a search bar, and a navigation menu with links to 首页, 像素, 设计, 相册, 论坛, and 关于.
- Sidebar 层 (Red):** Contains a "会员登录" form with fields for 用户名 and 密码, and buttons for 登录 and 取消. It also includes a "站点信息" section with statistics: 会员: 5844 人, 网站: 3856 个, 日期: 2006/03/20, and a "页面标准" section with W3C validation logos.
- MainBody 层 (Orange):** Contains two items, each with a thumbnail image, title, author, CSS version, release date, URL, and a note:
  - 风格名称: Red Vision  
设计作者: 雷木头  
CSS 版本: 基本想了  
发布日期: 2005/11/04  
演示地址: <http://www.tbiqu.com>  
风格简介: 此风格设计来自网友“雷木头”，内部LOGO融入《爱丽丝梦游仙境》兔子先生在地道里“潜水头”强化的设计师。
  - 风格名称: Dawn(黎明)  
设计作者: 雷木头了  
CSS 版本: 基本想了  
发布日期: 2005/09/19  
演示地址: <http://www.tbiqu.com>  
风格简介: 本SKIN适用于PHOTOSHOP V2.3版本。  
本SKIN的源文件全兼容IE, Mozilla Firefox和IE, 支持在IE240x768下浏览。
- PageBody 层 (Pink):** Contains a footer message: Copyright 2006 Tblog.com.cn All Rights Reserved shtml.css  
BKTZ 20060319号

# 页面布局

body{}

#container {}

#Header {}

#PageBody {}

#SideBar {}

#MainBody {}

#Footer {}

```
body {} /*HTML元素*/  
└ #Container {} /*页面层容器*/  
  └ #Header {} /*页面头部*/  
  └ #PageBody {} /*页面主体*/  
    └ #Sidebar {} /*侧边栏*/  
    └ #MainBody {}/*主体内容*/  
  └ #Footer {} /*页面底部*/
```

# 页面布局

---

- **HTML <div>标签**
  - 块级元素：浏览器会在其前后显示折行
  - 元素没有特定的含义
  - 可用作为组合其他 HTML 元素的容器
  - 与 CSS 一同使用，元素可用于对大的内容块设置样式属性。
  - 常见用途：文档布局。取代了表格定义布局的老式方法。

# DIV+CSS布局实例

- <body>
- <div id="container"> ①
- <div id="header"><h1>Main Title of Web Page</h1></div> ②
- <div id="pagebody"> ③
- <div id="menu"><h2>Menu</h2></div>
- <div id="content">Content goes here</div>
- </div>
- <div id="footer">Copyright W3School.com.cn</div> ④
- </div>
- </body>

# DIV+CSS布局实例

Main Title of Web Page

Menu

Content goes here

Copyright W3School.com.cn

# TIPS

---

- 选择器的命名一般遵循四种格式：
  - 组合词：classname
  - 连字符：.class-name
  - 下滑线：.class\_name
  - 驼峰式：.className (有的浏览器对大小写敏感)
- 标识符的命名：
  - 只能包含字符[a-zA-Z0-9]和UCS字符编码U+00A1及以上，再加连字号（-）和下划线（\_）；
  - 不能以数字，或一个连字号后跟数字为开头。
  - 可以包含转义字符加任何UCS字符作为一个数字编码。

# TIPS

---

- 选择器中样式的书写顺序：
  - 位置属性：position , z-index , display , float
  - 大小：width , height , padding , margin
  - 文字：font , color , text-align
  - 背景：background , border
  - 其他：animation, transition

```
.example {  
    color: red;  
    z-index: -1;  
    background-color: #9e0;  
    display: inline-block;  
    font-size: 1.5em;  
}  
  
.example {  
    z-index: -1;  
    display: inline-block;  
    font-size: 1.5em;  
    color: red;  
    background-color: #9e0;  
}
```

X                            ✓

# TIPS

- 在css的属性中，有些是可以缩写的，这样不仅代码少了，而且速度也可以提高。比如margin、padding、background等属性都可以进行缩写

```
.list-box {  
    border-top-style: none;  
    font-family: serif;  
    font-size: 100%;  
    line-height: 1.6;  
    padding-bottom: 2em;  
    padding-left: 1em;  
    padding-right: 1em;  
    padding-top: 0;  
}
```



```
.list-box {  
    border-top: 0;  
    font: 100%/1.6 serif;  
    padding: 0 1em 2em;  
}
```

# TIPS

---

- 简写
  - 为0的值，可以省略px单位；
  - 大于0小于1的小数，可以省略 ‘0’ 。

```
font-size: 0.8em;
```



```
font-size: .8em;
```

# TIPS

---

- 简写class名称的时候一定要让其他人看得明白。



```
.navigation {
    margin: 0 0 1em 2em;
}
.atr {
    color: #93c;
}
```

```
#nav {
    margin: 0 0 1em 2em;
}
.author {
    color: #93c;
}
```

# TIPS

---

- 尽量按照功能为css文件命名。

## CSS样式表文件命名

主要的 master.css  
模块 module.css  
基本共用 base.css  
布局、版面 layout.css  
主题 themes.css  
专栏 columns.css  
文字 font.css  
表单 forms.css  
补丁 mend.css  
打印 print.css

# TIPS

---

# TIPS

## CSS样式命名整理

### 页面结构

容器: container/wrap	整体宽度: wrapper	页头: header	内容: content	页面主体: main
页尾: footer	导航: nav	侧栏: sidebar	栏目: column	中间内容: center

### 导航

导航: nav	导航: mainnav/globalnav	子导航: subnav	顶导航: topnav	边导航: sidebar
左导航: leftsidebar	右导航: rightsidebar	边导航图标: sidebarIcon	菜单: menu	子菜单: submenu

标题: title

### 功能

标志: logo	登陆: login	登录条: loginbar	注册: register	产品: products
产品价格: productsPrices	产品评论: productsReview	编辑评论: editor-review	最新产品: news-release	广告/标语: banner
摘要: summary	生产商: publisher	缩略图: screenshot	常见问题: faqs	关键词: keyword
博客: blog	论坛: forum	搜索: search	搜索输入框: search-input	搜索输出: search-output
搜索结果: search-results	加入我们: joinus	状态: status	按钮: btn	滚动: scroll
标签页: tab	文章列表: list	提示信息: msg/message	当前的: current	小技巧: tips
皮肤: skin	充值: pay	活动: activities	推广: promotion	公告: announcement
排行: ranking	公司简介: companyProfile	公司设备: equipment	公司荣誉: glories	企业文化: culture
企业规模: scaleScale	营销网络: salesNetwork	组织机构: organization	技术力量: technology	分支机构: branches
企业资质:	公司实力: strengthStrength	经营理念: operationPrinciple	经理致辞: manager_oration	发展历程: developmentHistory

# 扩展

---

- **transition 属性**

- 简写属性，用于设置过渡效果；
- 包含四个过渡属性：
  - transition-property
  - transition-duration
  - transition-timing-function
  - transition-delay

- **语法：**

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

# 扩展

---

## 属性

transition-property

transition-duration

transition-timing-function

transition-delay

## 描述

规定设置过渡效果的 CSS 属性的名称。

规定完成过渡效果需要多少秒或毫秒。

规定速度效果的速度曲线。

定义过渡效果何时开始。

---

# 扩展

---

- **transition-property属性：**
  - 规定应用过渡效果的 CSS 属性的名称。
  - 当指定的 CSS 属性改变时，过渡效果将开始。
  - 过渡效果通常在用户将鼠标指针浮动到元素上时发生。
- **语法：**
  - `transition-property: none|all|property`

---

值	描述
<code>none</code>	没有属性会获得过渡效果。
<code>all</code>	所有属性都将获得过渡效果。
<code>property</code>	定义应用过渡效果的 CSS 属性名称列表，列表以逗号分隔。

---

# 扩展

- **transition-timing-function 属性：**
  - 规定过渡效果的速度曲线。
  - 该属性允许过渡效果随着时间来改变其速度。

---

值	描述
linear	相同速度开始至结束（等于 cubic-bezier(0,0,1,1)）。
ease	慢速开始，然后变快，然后慢速结束（cubic-bezier(0.25,0.1,0.25,1)）。
ease-in	慢速开始（等于 cubic-bezier(0.42,0,1,1)）。
ease-out	慢速结束（等于 cubic-bezier(0,0,0.58,1)）。
ease-in-out	慢速开始和结束（等于 cubic-bezier(0.42,0,0.58,1)）。
cubic-bezier(n,n,n,n)	在 cubic-bezier 函数中定义自己的值。可能的值是 0 至 1 之间的数值。

---

# 扩展

- **overflow 属性：**

- 规定当内容溢出元素框时发生的事情。

---

值	描述
visible	默认值。内容不会被修剪，会呈现在元素框之外。
hidden	内容会被修剪，并且其余内容是不可见的。
scroll	内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容。
auto	如果内容被修剪，则浏览器会显示滚动条以便查看其余的内容。

---

# 扩展

- **font属性：**

- 简写属性在一个声明中设置所有字体属性。

---

属性	描述
font-style	规定字体样式。 italic, oblique
font-variant	规定字体异体。 small-caps(小型大写字母的字体)
font-weight	规定字体粗细。 bold, bolder, lighter, 100~900
font-size/line-height	规定字体尺寸和行高。
font-family	规定字体系列。可以把多个字体名称作为一个“回退”系统来保存。如果浏览器不支持第一个字体，则会尝试下一个。

---

# 扩展

---

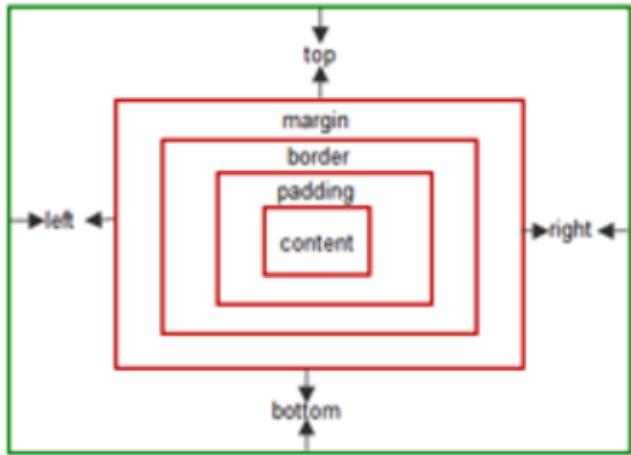
- **nth-child() 选择器**

- 表达形式 : `:nth-child(n)`
  - 语法解释:
    - 选择器匹配属于其父元素的第 N 个子元素，不论元素的类型。
    - n 可以是数字、关键词或公式。

- **last-child选择器**

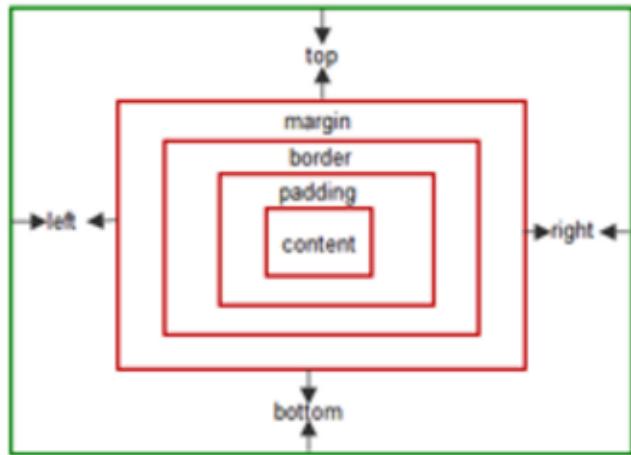
- 表达形式 : `:last-child`
  - 语法解释:
    - 选择器匹配属于其父元素的最后一个子元素的每个元素。
    - 等同于 `p:nth-last-child(1)`。

# 扩展



- **定位元素：**
  - 如果一个元素的 ‘position’ 特性值不是 “static” ( 默认元素不声明position即为static ) , 该元素被称为定位元素。
- **定位框：**
  - 定位的元素生成定位框
  - 定位位置基于四个特性 : 'top' , 'right' , 'bottom' , 'left'。

# 扩展



绝对定位元素



绝对定位元素的包含块边界

- 定位元素：
  - 已定位元素的 left,right,top,bottom 可以不设置初始值；
  - 上述四个属性具有默认值，但默认值并不是 0；
  - 没有设置值的定位元素将使元素的位置和“原来的位置”一样。