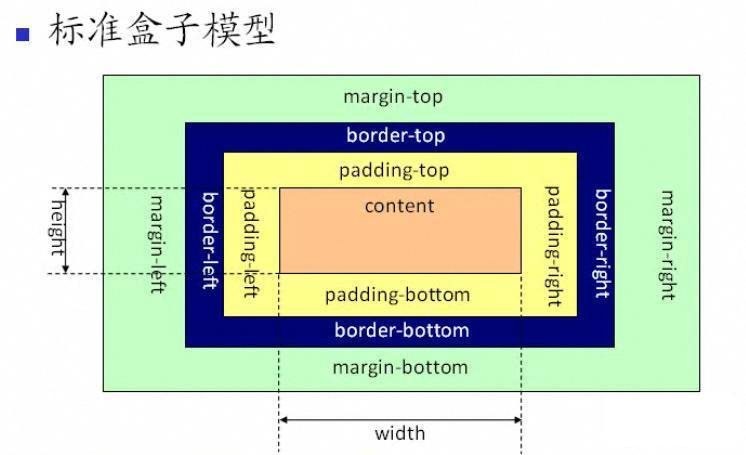
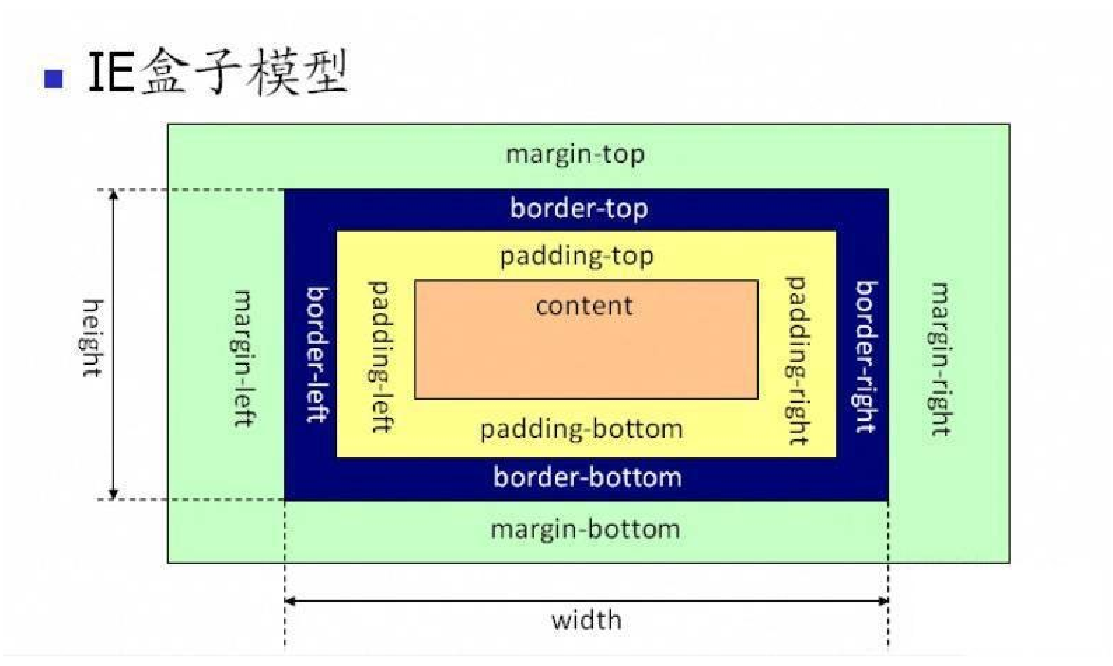
**Lesson04**

# 1 盒模型样式

## 盒模型的类型



可以看到，在标准盒模型下，width 和 height 是内容区域即content 的width 和height。而盒子总宽度为

**在标准模式下，一个块的总宽度= width + margin(左右) + padding(左右) + border(左右)**

而 IE 盒模型或怪异盒模型显而易见的区别就是，width 和height 除了content 区域外，还包含

padding 和border。盒子的总宽度为

**一个块的总宽度= width + margin(左右)（即 width 已经包含了 padding 和 border 值）**

## HTML 元素分类

在讲解CSS 布局之前，我们需要提前知道一些知识，在 CSS 中，html 中的标签元素大体被分为三种不同的类型：块状元素、内联元素和内联块状元素。

|  |  |  |
| --- | --- | --- |
| 元素分类 | 特性 | 标签 |
| 块级元素 | 1、每个块级元素都从新的一行开始，并且其后的元素也另起一行。  2、元素的高度、宽度、行高以及顶和底边距都可设置。  3、元素宽度在不设置的情况下， 是它本身父容器的 100%（和父元素的宽度一致），除非设定一个宽度。 | <div> <p> <h1>~<h6>  <ol> <ul> <dl>  <address>  <blockquote> <form> |
| 行内元素 | 1、和其他元素都在一行上；  2、元素的高度、宽度、行高及顶部和底部边距不可设置；  3、元素的宽度就是它包含的文字或图片的宽度，不可改变。 | <a> <span> <br/> <i>  <em> <strong> <label>  <q> |
| 行内块状元素 | 1、和其他元素都在一行上；  2、元素的高度、宽度、行高以及顶和底边距都可设置。 | <img> <input> <select> <textarea> <button> <iframe> |

**元素分类转换 display**

* block：将元素转换为块级元素
* inline：将元素装换为行级元素
* inline-block：将元素转换为内联块元素
* none: 将元素隐藏

**描边 border**

* border:2px solid #f00;
* 盒子模型的边框就是围绕着内容及补白的线，这条线你可以设置它的粗细、样式和颜色(边框三个属性)。
* 线条的样式：
  + dashed（虚线）| dotted（点线）| solid（实线）。
* css 样式中允许只为一个方向的边框设置样式：
  + 下描边 border-bottom:1px solid red;
  + 上描边 border-top:1px solid red;
  + 右描边 border-right:1px solid red;
  + 左描边 border-left:1px solid red;

**间距 margin**

* div{margin:20px 10px 15px 30px;}
* 元素与其它元素之间的距离可以使用边界（margin）来设置。边界也是可分为上、右、下、左。
* 也可以分开写：

|  |
| --- |
| div{ |
| **margin-top**:20px; |
| **margin-right**:10px; |
| **margin-bottom**:15px; |
| **margin-left**:30px; |
| } |

* 当 margin 给一个值时：指的是四个方向
* 当 margin 有两个值时：指的是上下 左右
* 当 margin 给三个值时：指的是上 左右 下
* 当 margin 给四个值时：值的是上 右 下 左

**内填充 padding**

* padding:10px
* 元素内容与边框之间是可以设置距离的，称之为填充。填充也可分为上、右、下、左。
* 如下代码：div{padding:20px 10px 15px 30px;}
* 顺序一定不要搞混。可以分开写上面代码：

|  |
| --- |
| div{ |
| **padding-top**:20px; |
| **padding-right**:10px; |
| **padding-bottom**:15px; |

|  |
| --- |
| **padding-left**:30px; |
| } |

* 当 padding 给一个值时：指的是四个方向
* 当 padding 有两个值时：指的是上下 左右
* 当 padding 给三个值时：指的是上 左右 下
* 当 padding 给四个值时：值的是上 右 下 左

•

# 2 浮 动 float

## 浮动原理

* 浮动使元素脱离**文档普通流**，漂浮在普通流之上的。
* 浮动元素依然按照其在普通流的位置上出现，然后尽可能的根据设置的浮动方向向左或者向右浮动，直到浮动元素的外边缘遇到包含框或者另一个浮动元素为止，且允许文本和内联元素环绕它。
* 浮动会产生块级框（相当于设置了 display:block），而不管该元素本身是什么。

## 清除浮动带来的影响

* clear 清除浮动：
  + none : 不清除（默认值）。
  + left : 不允许左边有浮动对象
  + right : 不允许右边有浮动对象
  + both : 不允许两边有浮动对象
* 利用伪类实现清除浮动

|  |
| --- |
| .clearFix{ |
| content=""; |
| **display**:block; |
| **width**:0; |
| **height**:0; |
| **clear**:both; |
| } |

# 3 定位 position

定位功能可以让布局变的更加自由。

## 层模型--绝对定位（相对于父类）

如果想为元素设置层模型中的绝对定位，需要设置 position:absolute(绝对定位)， 这条语句的作用将元素从**文档流**中拖出来，然后使用left、right、top、bottom 属性相对于其最接近的一个具有定位属性的父包含块进行绝对定位。如果不存在这样的包含块，则相对于 body 元素，即相对于浏览器窗口。

* position:absolute;

|  |
| --- |
| div{ |
| **width**:200px; |
| **height**:200px; |
| **border**:2px red solid; |
| **position**:absolute; |
| **left**:100px; |
| **top**:50px; |
| } |
| */\*实现 div 元素相对于浏览器窗口向右移动 100px，向下移动 50px。\*/* |

## 层模型--相对定位（相对于原位置）

如果想为元素设置层模型中的相对定位，需要设置 position:relative（相对定位），它通过 left、right、top、bottom 属性确定元素在正常文档流中的偏移位置。相对定位完成的过程是首先按 static(float)方式生成一个元素(并且元素像层一样浮动了起来)，然后相对于原位置移动，移动的方向和幅度由 left、right、top、bottom 属性确定，偏移前的位置保留不动。

* position:relative;

|  |
| --- |
| #div1{ |
| **width**:200px; |
| **height**:200px; |
| **border**:2px red solid; |
| **position**:relative; |
| **left**:100px; |
| **top**:50px; |
| } |
| */\*相对于以前位置向下移动 50px，向右移动 100px;\*/* |

## 层模型--固定定位（相对于网页窗口）

与 absolute 定位类型类似，但它的相对移动的坐标是视图（屏幕内的网页窗口） 本身。由于视图本身是固定的，它不会随浏览器窗口的滚动条滚动而变化，除非你在屏幕中移动浏览器窗口的屏幕位置，或改变浏览器窗口的显示大小，因此固定定位的元素会始终位于浏览器窗口内视图的某个位置，不会受文档流动影响， 这与 background-attachment:fixed 属性功能相同。

* position:fixed