Day13笔记

作业！

看视频号中的浮动布局！！！！！！！！！！尽量跟着做。预习flex布局

做tommc上的作业

3.绝对定位

- absolute 绝对定位，元素将脱离文档流，其他元素不为该元素预留空间。

- 它的移动参照为祖先元素的偏移，来确定元素位置。元素会逐层向上寻找自己的参照元素，当找到的最近一层祖先元素具有 position 属性时，该元素就会以这个祖先元素的原点为参照点。

- 可使用 top，right，bottom 和 left 做偏移。

- 当四个方向值发生重合时，以“上 top”和“左 left”为优先。

- 绝对定位的元素可以设置外边距，且不会与其他边距合并。

- 应用场景背景图中的文字和按钮，轮播图中的控制符

文本

描述已自动生成

（1）子元素在父元素中定位

电脑的屏幕

描述已自动生成

（2）子元素在父元素内居中

使用元素定位，让已知高度的子元素在父元素中上下左右居中

|  |
| --- |
| <style>  .baba {  width: 500px;  height: 400px;  background-color: red;  position: relative;  }  .erzi {  width: 150px;  height: 200px;  background-color: blue;  position: absolute;  left:50%;  top:50%;  margin-left: -75px;  margin-top: -100px;  }  </style>  </head>  <body>  <div class="baba">  <div class="erzi"></div>  </div>  </body> |

4.固定定位

fixed 固定定位是元素，决定它的“包含块”是 html，唯一可以限定固定定为元素的就是 html 跟元素

- `position: fixed;`它到父级始终是 html

- 直接在窗口的某个位置固定

- 可使用 top，right，bottom 和 left 做偏移。

(1)居中的弹窗

固定定为，元素在父元素中居中

|  |
| --- |
| \* {  margin: 0;  padding: 0;  }  .alertBox {  width: 300px;  height: 200px;  color: red;  background-color: #aaa;  position: fixed;  top: 50%;  left: 50%;  margin-top: -100px;  margin-left: -150px;  } |

(2)返回顶部元素

|  |
| --- |
| <style>  \* {  margin: 0;  padding: 0;  }  .pic {width: 100%;}  .pic img {  width: 100%;  display: block;  }  .top {  width: 100px;  height: 100px;  border-radius: 5px;  background-color: rgba(0,0,0,0.5);  font-size: 30px;  text-align: center;  line-height: 100px;  position: fixed;  bottom: 100px;  right: 100px;  }  .top a {  color: #fff;  text-decoration: none;  }  </style>  <body>  <div class="pic">  <img src="./img/固定定位长图背景.jpg" alt="">  </div>  <div class="top">  <a href="#">TOP</a>  </div>  </body>  </html> |

5.z-index

z-index 就是 z 轴的顺序，z-index 可以设置字符值，如：auto，多数情况设置整数值，可以为负值。z-index 的最大值是 21,4748,3647 (21 亿多)。跨度尽量 10 以上

(1) 层叠顺序取值

- `z-index`层叠顺序

- 直接设置没有单位的整数，可以为负值，值越大层级越高

图表

描述已自动生成

(2) 层叠领域的准则

- 值大的在上：z-index 的值，在同一个层叠上下文领域，层叠水平值大的那一个覆盖小的。

- 后来的在上：当元素的层叠水平一致的时候，在文档流中处于后面的元素会覆盖前面的元素。

【练习】

>背景图

>浮动布局

电脑游戏的截图

描述已自动生成>定位

十九、渐变

1. 线性渐变

(1) 颜色和角度

- `background-image: linear-gradient(color1,color2,……);`渐变最少两个颜色，才能体现出颜色的变化

- `linear-gradient()`渐变颜色的参数用逗号分开，可以写多种颜色表达方式

- 渐变的方向，按照参数顺序依次向下，可以在颜色前加入角度或关键字控制渐变方向

- `90deg` 角度

- to 终点 关键词，就是终点在哪个位置，top，left，right，bottom，记得加空格 （不建议使用）

【练习】

形状, 矩形

描述已自动生成

> 使用七种颜色作出彩虹渐变效果

> 颜色可以使用任何表示色值的方式

【练习】

图形用户界面, 日程表

描述已自动生成

> 渐变的导航,颜色自定义

> 导航总宽度 1000px 高度 30px

> 需要加入鼠标悬停改变背景颜色的效果

(2) 渐变的比例

- `background-image：linear-gradient(方向，颜色1 起始值 结束值，颜色2 起始值 结束值，颜色n 起始值 结束值……)`

- 多个颜色渐变时，渐变的区域需要调整，可采取控制颜色范围的方法。在函数内颜色的后面加入长度单位（如 px 或%）用空格分隔。

- 【注意】浏览器版本低时，渐变的开始值和结束值无效。需要升级浏览器版本，下载谷歌新版浏览器。

|  |
| --- |
| div {  width: 600px;  height: 300px;  border: 2px solid black;  background-image: linear-gradient(  90deg,  green 0 200px,  yellow 200px 400px,  red 400px 600px  );  } |

背景图案

描述已自动生成2. 径向渐变

(1) 渐变方向和范围

- `background-image: radial-gradient(color1,color2,……);`

- background-image：radial-gradient(颜色 1 起始值 结束值，颜色 2

起始值 结束值，颜色 n 起始值 结束值……)

(2) 半径

- `background-image: radial-gradient(半径,颜色1,颜色4,颜色3);`

- `background-image: radial-gradient(x轴半径 y轴半径,颜色1,颜色4,颜色3);`

- 半径只传入一个参数，则表示该渐变形状为圆

- 半径可以使用 px

- 【注意】真实的渐变控制如果比外层空间小，剩下都是部分会被最后一个颜色填充

图片包含 图形用户界面

描述已自动生成

(3) 圆心起点

- background-image: radial-gradient(x 轴半径 y 轴半径 at x 轴圆心 y 轴圆心,颜色 1 起始值 结束值，颜色 2 起始值 结束值……)

手机屏幕截图

低可信度描述已自动生成- 在圆心后面增加 at 空格 x 轴 y 轴的原点位置，可以使用长度单位或关键词。取值可使用 px，% 或者关键词：left/top/bottom/right

- 注意重点

- 半径只传入一个参数，则表示该渐变形状为圆

- 颜色范围用的百分比,半径变化，百分比参照半径

- 颜色范围用的是 px，则不会因为半径的变化而变化

|  |
| --- |
| d1 {  background-image: radial-gradient(  50px at left top,  yellow 0 50px,  #222 100px  );  }  .d2 {  background-image: radial-gradient(50px at 20% 30%, yellow 0 50px, #222 100px);  } |

3. 重复的渐变

- 注意重复的渐变要求浏览器版本，版本过低的浏览器版本无法查看效果

- `background-image: repeating-linear-gradient()` 重复的线性渐变

- `background-image: repeating-radial-gradient()` 重复的径向渐变

- 当半径过小，页面分辨率显示效果会怪异

|  |
| --- |
| .d1 {  background-image: repeating-linear-gradient(red 0 2px,yellow 4px 6px,green 8px 10px);  }  .d2 {  background-image: repeating-radial-gradient(  red 0 2%,  yellow 2% 4%,  green 4% 6%  );  } |

【练习】

文本, 信件

描述已自动生成

> 信封用重复渐变 background-image: repeating-linear-gradient(-45deg,#f00 0 10px,#fff 10px 20px,#00f 20px 30px,#fff 30px 40px)

4. 对渐变的兼容

渐变效果不支持低版本浏览器，之前关于 css3 的浏览器兼容，厂商的私有前缀

浏览器厂商制作的浏览器，不同时期对 css 的兼容不同，每个浏览器都有自己的内核

查看学子商城渐变的写法，对应每一个内核都要写一遍，要写四遍。但最终只会按照当前浏览器的内核执行对应的代码

（1）浏览器私有前缀

- `-webkit-` 浏览器：chrome/safari/edge

- `-moz- ` 浏览器：firefox

- `-o- ` 浏览器：opera

- `-ms-` 浏览器：IE

（2）渐变的方向

- 不写私有前缀，要写终点的方式，如 to bottom

- 没有私有前缀正确写法:`background-image: linear-gradient(to bottom,red,yellow);`

- 没有私有前缀错误写法:`background-image: linear-gradient(bottom,red,yellow);`

- 写私有前缀要写起点的方式 如 top

- 有私有前缀正确的写法:`background-image: -webkit-linear-gradient(top,red,yellow);`

- 有私有前缀错误的写法:`background-image: -webkit-linear-gradient(to top,red,yellow);`

二十、css3 高级选择器

1.属性选择器

属性选择器可以使用`[]`将元素的属性名称放入其中，从而通过查找具有该属性的元素来设置其样式。

元素的属性包括：

预定义属性：如`id、class、style、title、src、href`等

自定义属性：随意指定的属性名如 abc、w123 或 更标准的自定义属性写法 `data-\* `

(1)简单属性选择器

- 具有某个属性的元素，如 id、class ： `[属性名]{样式声明} `

- 特指某个元素具有某个属性，如`p[class]：元素名[属性名]{样式声明}`

- 具备多个属性的元素，如`[id][class] : [属性名1][属性名2]{样式声明}`

- 特指某元素具备多个属性：`元素名[属性名1][属性名2]{样式声明}`

(2)筛选属性选择器

- 元素的属性名的值为什么，如 id=mydiv ：`[属性名=值]{样式声明} `

- 元素的属性名以什么值开头：`[属性名^=开头的字段]{样式声明}`

- 元素的属性名以什么值结尾：`[属性名$=结尾的字段]{样式声明}`

- 元素的属性名包括某字段的值：`[属性名\*=包含的字段]{样式声明}`

- 元素的属性名包括某独立的字段值：`[属性名~=包含的字段]{样式声明}`

【练习】

> 动图和静态图一起排列

> 给动态图增加动图的标签

图形用户界面, 网站

描述已自动生成

2.目标伪类选择器

- 对应锚点被激活时，匹配的样式,使用 a 标签的 href 属性连接元素锚点

- `选择器:target{样式声明}`

|  |
| --- |
| <style>  div {  width: 200px;  height: 200px;  font-size: 60px;  text-align: center;  border: 1px solid black;  }  div:target {  background-color: yellow;  }  /\*可以做成类似tab切换效果\*/  div {  display: none;  }  div:target {  display: block;  }  </style>  <a href="#div1">选择1</a>  <a href="#div2">选择2</a>  <a href="#div3">选择3</a>  <div id="div1">div1</div>  <div id="div2">div2</div>  <div id="div3">div3</div> |

3.结构性伪类选择器

- 匹配父元素的第一个孩子 ：`父元素>:first-child {样式声明}`

- 匹配父元素的最后一个孩子 ：`父元素>:last-child {样式声明}`

- 匹配父元素的第 n 个孩子： `父元素>:nth-child(n) {样式声明}`

- 匹配内部没有任何内容（包含文本）的标签 ：`父元素 >:empty{ 样式声明}`

- 匹配父元素的唯一子元素：`父元素 >:only-child{样式声明}`

- 使用`not()`参数为选择器，否定该括号内选择器的其他所有选择器，类似于取反：`父元素 >:not(不想要的选择器){样式声明}`

- 【注意】`div:nth-child(2)` 与 `div>:nth-child(2)` 有区别。

|  |
| --- |
| .box1 {  background-color: red;  height: 100px;  }  .box1 > div {  height: 100px;  width: 100px;  float: left;  }  .box1 > :first-child {  background-color: yellow;  }  .box1 > :last-child {  background-color: plum;  }  .box1 > :nth-child(2) {  background-color: purple;  }  .box1 > :empty {  background-color: saddlebrown;  }  ul > :only-child {  background-color: red;  }  /\* 除了谁之外的所有人 \*/  ul > :not(.cqbhj) {  color: green;  } |