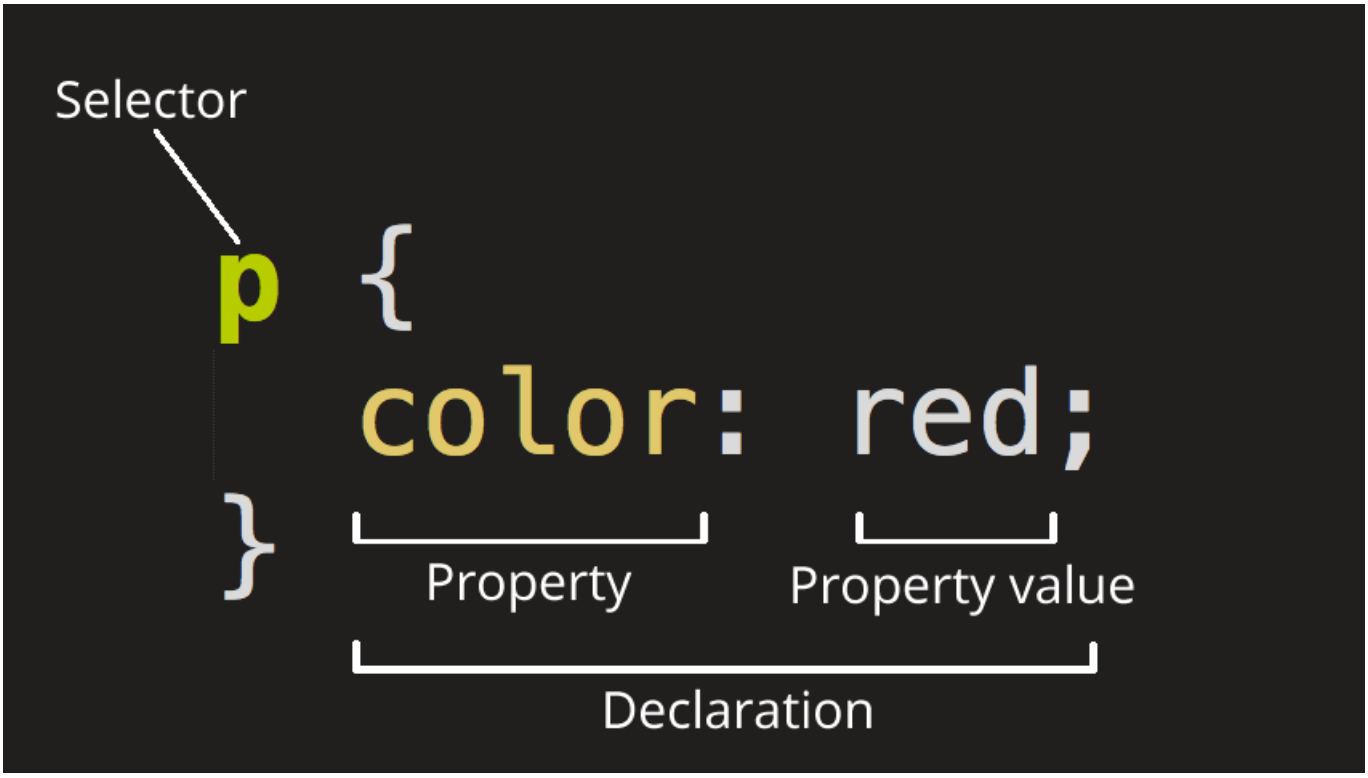
css学习

CSS基础

css规则集详解



选择器（Selector）

HTML 元素的名称位于规则集开始。它选择了一个或多个需要添加样式的元素（在这个例子中就是 `` 元素）。要给不同元素添加样式，只需要更改选择器。

声明（Declaration）

一个单独的规则，如 `color: red;` 用来指定添加样式元素的**属性**。

属性（Properties）

改变 HTML 元素样式的途径（本例中 `color` 就是 `` 元素的属性）。CSS 中，由编写人员决定修改哪个属性以改变规则。

属性的值（Property value）

在属性的右边，冒号后面即**属性的值**，它从指定属性的众多外观中选择一个值（我们除了 `red` 之外还有很多属性值可以用于 `color` ）。

盒子布局

CSS 布局主要是基于*盒子模型*。在你的页面上占用空间的每个盒子都有类似的属性：

- `padding`（内边距）：是指内容周围的空间。在下面的例子中，它是段落文本周围的空间。

- `border`（边框）：是紧接着内边距的线。
- `margin`（外边距）：是围绕元素边界外侧的空间。
- `width`：元素的宽度
- `background-color`：元素内容和内边距底下的颜色
- `color`：元素内容（通常是文本）的颜色
- `text-shadow`：为元素内的文本设置阴影
- `display`：设置元素的显示模式
- `width: 600px;` 强制页面永远保持 600 像素宽。
- font-size：元素内部文字大小
  - px：像素
  - em：相对父元素的字体大小

  ==如果元素没有声明字体大小，则使用父元素的字体大小，如果没有父元素（html），则使用基准字号。
- font-weight：文字粗细

  > strong元素：重要的内容
- font-family：字体

  > 非衬线字体：字体边缘没有修饰的字体
- font-style：字体样式（常用于设置字体倾斜）

  > i元素：技术术语、外文短语或是小说中人物的思想活动等，常用于表示图标(icon)
  >
  > em元素：强调的内容
- letter-spacing：文字间隙
- text-align：元素内部文本水平对齐方式
- text-indent：首行缩进
- text-decoration：文本加线

  > a元素
  >
  > del元素：删除/废弃的内容
- line-height：文本行高，文字在行高内垂直居中
- `margin: 0 auto;` 当你在 `margin` 或 `padding` 这样的属性上设置两个值时，第一个值影响元素的*上下*方向（在这个例子中设置为 `0`）；第二个值影响*左右*方向。(这里，`auto` 是一个特殊的值，它将可用的水平空间平均分配给左和右)

- `background-color: #FF9500;` 指定元素的背景颜色
- `padding: 0 20px 20px 20px;` 给内边距设置了四个值来让内容四周产生一点空间。这一次我们不设置上方的内边距，设置右边，下方，左边的内边距为 20 像素。值以上、右、下、左的顺序排列。
- `border: 5px solid black;` 这是为边框的宽度、样式和颜色设置的值。在本例中，它是一个在主体的所有侧面的 5 像素宽的纯黑色边框。

选择器

- 元素选择器
- id选择器 `#xxx`
- 类选择器 `.xxx`
- 通配符选择器（*选中所有元素）
- 属性选择器（根据属性名和属性值选择元素）
- 伪类选择器（选中某些元素的某种状态）
    - link：超链接未访问过的状态
    - visited：超链接访问过的状态
    - hover：鼠标悬停状态
    - active：激活状态（鼠标按下状态）

  ==爱恨法则：love hate==,四个优先级
- 伪元素选择器
    - before
    - ```
      /* 每一个 <p> 元素的第一行。 */
      p::first-line {
        color: blue;
        text-transform: uppercase;
      }
      ```
    - after

组合选择器

- 交集选择器`#xxx.xxx`
- 并集选择器（逗号）
- 后代选择器（空格）
- 子元素选择器（>）
- 相邻兄弟选择器（+）
- 通用兄弟选择器（~）

层叠（权重计算）
      ```

解决声明冲突（浏览器解决）

一、比较重要性

重要性从高到低：

1. !important样式
2. 普通样式
3. 浏览器默认样式

==重置样式表，如：reset.css, normalize.css...==

二、比较特殊性

选择器选中的范围越窄，就越特殊

> 通过选择器计算出一个四位数（0 0 0 0），四位数越大，特殊性越高
>
> 千位：内联样式记1，否则记0
>
> 百位：等于选择器中所有id选择器的数量
>
> 十位：等于选择器中所有类选择器、属性选择器以及伪类选择器的数量和
>
> 个位：等于选择器中所有元素选择器和伪元素选择器的数量和

三、比较源次序

代码书写靠后的胜出

继承

子元素会继承父元素的**部分**css属性。通常来说，和文字内容相关的属性都能被继承。

css属性值计算过程

浏览器要渲染一个元素的前提：**该元素的所有css属性必须都要有值**

1. 确定声明值：==样式表中没有冲突的声明，直接作为CSS属性值。==
2. 层叠：==样式表中有冲突的声明，使用层叠规则确定css属性值。==
3. 继承：==前两步结束，仍然没有值的属性，如果可以继承，则继承父元素的值。==

   > inherit：强制（手动）继承
   >
   > initial：初始值，默认值

4. 使用默认值：==前三步结束，仍然没有值的属性，使用默认值。==

盒模型

每一个元素都会生成一个矩形区域，这个矩形区域就被认为是一个盒子（box），而盒模型就是用来描述这些盒子的方式。

盒子类型：

- 行盒：display等于inline的元素，水平方向排列，不换行。
- 块盒：display等于block的元素，垂直方向排列，独占一行。

盒子组成：

1. 内容（content），内容盒（content-box）

   width, height

2. 填充（padding）、内边距

   padding-top, padding-bottom, padding-left, padding-right

   简写（速写）属性：padding

   ```
   /* 应用于所有边 */
   padding: 10px;

   /* 上边下边 | 左边右边 */
   padding: 10px 20px;

   /* 上边 | 左边右边 | 下边 */
   padding: 10px 20px 30px;

   /* 上边 | 右边 | 下边 | 左边 */
   padding: 10px 20px 30px 40px;
   ```

   内容区+填充区：填充盒（padding-box）

3. 边框（border）

   border-style, border-width, border-color, border-radius(css3)

   border: 宽度 样式 颜色;

   内容区+填充区+边框：边框盒（border-box）

4. 外边距（margin）

   margin-top, margin-bottom, margin-left, margin-right

   简写属性：margin

盒模型扩展

- 修改宽高范围（box-sizing）
- 修改背景范围（background-clip ）
- 溢出控制（overflow、text-overflow）

行盒

盒子跟着内容延伸，无法直接设置宽高，宽高是由内容决定。

内边距、边框、外边距：水平方向尺寸有效，垂直方向不占空间。

行块盒

display等于inline-block的盒子，水平排列，所有方向的尺寸都有效

> 空白折叠会发生在行盒内部以及行盒之间，包括行块盒
>
> 可替换元素：页面上显示的内容，取决于元素属性，类似于行块盒。如：img，多媒体，文本框，按钮等等
>
> 不可替换元素：页面上显示的内容，取决于元素内容

格式化视觉模型

页面中多个盒子的排列规则，俗称布局规则。

> 包含块：大多数情况下，包含块就是父元素的内容盒，它决定了盒子尺寸和排列位置

普通流

文档流、标准流、常规流、普通文档流、常规文档流。。。

所有元素，默认情况下，都属于普通流布局。

块盒独占一行，垂直排列。行盒不换行，水平排列。

块盒

- 水平方向

  默认情况下，块盒总宽度，等于包含块宽度，因为width默认值是auto（剩余空间全部吃掉）

  常规流里面，固定宽度，左右margin设置为auto，可以使块盒在包含块中居中

- 垂直方向

  height，默认就是auto，表示自适应内容高度

  margin，auto表示0

- 百分比取值

  width, height, padding, margin都可以用百分比取值

  width, padding, margin的百分比，相对的都是包含块的宽度

  height百分比：

  - 如果包含块的高度不是auto：相对于包含块的高度
  - 如果包含块的高度是auto：百分比无效

- margin合并

上下相邻margin会进行合并，取最大值

浮动

不管是行盒还是块盒，浮动之后，都会变成块盒

float：

- 默认值none，普通流
- left：左浮动，元素靠上靠左排列
- right：右浮动，元素靠上靠右排列

auto：

- width、height为auto时，适应内容宽高
- margin为auto时，表示0

==浮动盒子排列时，会避开普通流盒子==

==普通流块盒排列时，会无视浮动盒子==

==行盒盒排列时，会避开浮动盒子。==

高度坍塌问题：

浮动流盒子脱离了文档流，所以普通流在计算高度的时候，不考虑浮动盒子。解决办法如下：

```
/* 给高度坍塌的元素加上这个类名 */
.clearfix::after {
    content: "";
    display: block;
    /* 清除浮动 */
    clear: both;
}
```

文字环绕：

```
img {
    width: 600px;
    float: left;
    /* 计算图片透明度后,根据形状围绕 */
    shape-outside: url(./img/kv-yll_d411792e.webp);
    /* 圆形 */
    /* shape-outside: circle(); */
    /* 椭圆 */
    /* shape-outside: ellipse(); */
    /* 四周从外边距向内嵌入 */
    /* shape-outside: inset(100px 100px 100px 100px); */
    /* 自定义形状 */
    /* shape-outside: polygon(0 6%, 55% 6%, 62% 25%, 75% 30%, 75% 50%, 72% 60%,
```

```
80% 70%, 95% 80%, 100% 90%, 80% 100%); */
}
```

定位

position：用于控制元素在包含块中的精确位置

- static（默认，没有定位）
- relative（相对定位）

  使元素相对于其自身原来的位置进行偏移，不会影响其它元素的位置，不会使元素脱离文档流。
- absolute（绝对定位，**脱离文档流**）

  包含块：包含块是所属定位元素的填充盒，如果它没有已定位的祖先元素，就会相对于初始包含块进行定位。

  auto：宽高适应内容，margin表示0（上下或左右定位不冲突，否则auto会吃掉剩余空间）
- fixed（固定定位， **脱离文档流**）

  包含块：视口（浏览器可视窗口）

  auto：宽高适应内容，margin表示0（（上下或左右定位不冲突，否则auto会吃掉剩余空间）

  > 初始包含块：是和视口大小一样的固定虚拟矩形，它会靠在页面左上方，位置不会动，但大小会跟着视口变化。
- sticky（粘性定位）

  可以理解为相对定位和固定定位的混合，元素在跨越特定阈值前为相对定位，之后为固定定位。

  粘性定位元素的父元素显示，它就会一直显示，父元素消失，它也会消失。

背景图

如果图片是作为网页内容显示的，应该用img元素。 如果图片只是为了美化页面，应该用背景图。

- background-image
- background-size
- background-repeat
- background-position

  精灵图（sprite、雪碧图）

其他元素

video

> 媒体参与度

- controls属性：显示播放控件
- autoplay属性：自动播放
- muted属性：静音
- loop属性：循环播放

```
<video controls autoplay muted loop src="media/hy.mp4" style="width: 1000px;">
</video>
```

audio

> 同video

```
<audio controls src="media/hy.wav"></audio>
```

iframe

内联框架元素

```
<iframe src="https://www.bilibili.com" frameborder="0"></iframe>
```

表单元素

input

- type属性：输入框类型
  - text：单行文本输入框
  - password：密码输入框
  - search：搜索框
  - tel：电话号码输入框
  - number：数字输入框
  - radio：单选框
  - checkbox：多选框
  - date：日期选择框
  - color：颜色选择框
  - file：文件选择框
  - range：滑块选择框
  - button/submit/reset：按钮
- placeholder：占位符
- value：内容

button

type属性：button/submit（默认）/reset

textarea

多行文本输入框、文本域

select

下拉列表选择框

表单状态

- readonly：只读状态
- disabled：禁用状态

form

点击提交按钮的时候，form元素内部的表单内容，会以某种方式提交到服务器。

点击重置按钮的时候，form元素内部的表单内容，会全部重置成初始状态。

其它选择器

伪类选择器

- :first-child：第一个子元素
- :first-of-type：第一个同类型的兄弟元素
- :last-child：最后一个子元素
- :last-of-type：最后一个同类型的兄弟元素
- :nth-child：指定位置的子元素
    - even：等同于2n
    - odd：等同于2n+1
- :nth-of-type：指定位置的同类型兄弟元素

伪元素选择器

- ::first-letter：元素中第一个文字
- ::first-line：元素中第一行文字
- ::selection：用户选中的文字

at-rule

@规则、@语句、@指令、css指令。。。

- import

  导入另外一个css文件：`@import "path";`

- charset

设置css文件使用的字符编码：`@charset "utf-8"`

web字体

```
/* 创建web字体 */
@font-fase {
    font-family: "字体名字";
    src: url("字体文件路径");
}

/* 使用web字体 */
span {
    font-family: "创建的字体名字";
}
```

网格

- 通过把容器 `display: grid` ,来定义一个网格。
- 加三个宽度为`200px`的列。当然，这里可以用任何长度单位，包括百分比。

  cssCopy to Clipboardplay

  ```
  .container {
    display: grid;
    grid-template-columns: 200px 200px 200px;
  }
  ```

- 规则来修改你的网格轨道，创建 3 个宽度为 `1fr` 的列：

  cssCopy to Clipboardplay

  ```
  .container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
  }
  ```

- 使用 `grid-column-gap` 属性来定义列间隙；使用 `grid-row-gap` 来定义行间隙；使用 `grid-gap` 可以同时设定两者。
- 重复构建轨道组

你可以使用`repeat`来重复构建具有某些宽度配置的某些列。举个例子，如果要创建多个等宽轨道，可以用下面的方法。

cssCopy to Clipboard

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-gap: 20px;
}
```

媒体查询

- **CSS 媒体查询**为你提供了一种应用 CSS 的方法，仅在浏览器和设备的环境与你指定的规则相匹配的时候 CSS 才会真的被应用，例如“视口宽于 480 像素”的时候

媒体类型

你可以指定的媒体类型为：

- `all`
- `print`
- `screen`
- `speech`

下面的媒体查询将会在页面被打印的时候把 body 设定为只有 12pt 大小。当页面在浏览器中载入的时候，它将不会生效。

cssCopy to Clipboard

```
@media print {
  body {
    font-size: 12pt;
  }
}
```

`width`（和`height`）媒体特征可以以数值范围使用，于是就有了`min-`或者`max-`的前缀，指示所给的值是最小值还是最大值。例如，要让颜色在视口窄于 400 像素的时候变成蓝色的话，可以用`max-width`：

cssCopy to Clipboard

```
@media screen and (max-width: 400px) {
  body {
    color: blue;
  }
}
```

媒体查询中的“或”逻辑

如果你有一组查询，且要其中的任何一个都可以匹配的话，那么你可以使用逗号分开这些查询。在下面的示例中，文本会在视口至少为 400 像素宽的时候**或者**设备处于横放状态的时候变为蓝色。如果其中的任何一项成

立，那么查询就匹配上了。

cssCopy to Clipboard

```
@media screen and (min-width: 400px), screen and (orientation: landscape) {
  body {
    color: blue;
  }
}
```

媒体查询中的“非”逻辑

你可以用not操作符让整个媒体查询失效。这就直接反转了整个媒体查询的含义。因而在下面的例子中，文本只会在朝向为竖着的时候变成蓝色。

cssCopy to Clipboard

```
@media not all and (orientation: landscape) {
  body {
    color: blue;
  }
}
```