1.4 CSS

1.4.1 CSS简介

1.4.1.1 CSS引用

引入 CSS 的三种方式：

1.外部样式表

2.内部样式表

3.内嵌样式

1.4.1.2 语法

1. **模版：**
   选择器｛
   属性申明1
   属性申明2
   .
   .
   .
   ｝
2. 浏览器私有属性：
   - **chrome，safari**
     -webkit-

- **firefox**
  -moz-
- **IE**
  -ms-
- **opera**
  -o-

3. 属性值语法
   margin: [<length> | <percentage> | auto] {1,4}
   一般是由 **基本元素** **组合符号** **数量符号** 组成
   - 基本元素：
     - 关键字：(auto,solid,bold...)
     - 类型：
       - 基本类型 (<length>, <percentage>, <color>...)
       - 其他类型 (<'padding-width'>, <color-stop>...)
     - 符号：(/ ,)
     - inherit, initial
   - 组合符号：
     1. 空格
        必须出现，要按顺序出现，例：
        `<'font-size'> <'font-family'>`
        则合法值如下：
        `- 12px arial`
        不合法值如下：
        `- 2em`
        `- arial 14px`
     2. &&
        必须出现，顺序无所谓，例：
        `<length>&&<color>`
        则合法值如下：
        `- green 2px` `- 1em blue`
        不合法值如下：
        `- blue`
     3. ||
        至少出现一个，顺序无所谓，例：
        `underline||overline||line-through||blink`
        则合法值如下：
        `- underline`
        `- overline underline`
     4. |
        只能出现一个，例：
        `<color>|transparent`
        则合法值如下：
        `- orange` `- transparent`
        不合法值如下：
        `- blue transparent`

5. []

   分组作用，括号内可当做一个整体，例：

   `bold [ thin||<length> ]`

   则合法值如下：

   `- bold thin` `- bold 2em`

- 数量符号：

1. 无

   如果只是

   `&lt;length>`

   则合法值如下：

   `- 1px`

   `- 10em`

   不合法值如下：

   `- 1px 2px`

2. +

   可以出现一次或者多次，例：

   `<color-stop>[,<color-stop>]+`

   则合法值如下：

   `- #fff, red`

   `- blue, green 50%, gray` 不合法值如下： `- red`

3. ?

   表示可以出现也可以不出现，例：

   `inset?&& <color>`

   则合法值如下：

   `- inset blue`

   `- pink`

4. {}

   表示元素出现次数，例：

   `<length>{2,4}`

   则合法值如下：

   `- 1px, 2px`

   `- 1px, 2px, 3px, 9px`

   不合法值如下：

   `- 1px`

5. *

   可以出现零次、一次或者多次，例：

   `<time> [,<time>]*`

   则合法值如下：

   `- 1s`

   `- 1s, 4ms`

6. #

   可以出现一次或者多次,中间用**逗号**隔开，例：

   `<time> #`

   则合法值如下：

   `- 2s, 4s`

不合法值如下：

```
- 1ms 3ms
```

4. @规则语法

模式：

@ 标识符 xxx;

@ 标识符 xxx {};

常用的有：

- @media 用做响应式布局，媒体查询条件
- @keyframes 主要用来描述CSS动画的中间步骤
- @font-face 主要用来引入外部字体

1.4.2 CSS选择器

1.4.2.1 简单选择器

1. 简单选择器
   - 标签选择器 tag
   - 类选择器 .
   - ID选择器 #
   - 通配符选择器 *
   - 属性选择器1 [att]
   - 属性选择器2 [att=val]
     - 选中值为val的属性，我们的id选择器其实就是一种特殊的属性选择器2：
       - `#nav{} == [id=nav]{}`
   - 属性选择器3 [att~=val]
     - 选中包含val值的所有属性，我们的class选择器就是一种特殊的属性选择器3：
       - `.nav{} == [class~=nav]{}`
   - 属性选择器4 [att|=val]
     - 选中值为val或val加中划线开头，用的不多一般用在选择en和en-**的时候
   - 属性选择器5 [att^=val]
     - 以val开头的
   - 属性选择器6 [att$=val]
     - 以val结尾的
   - 属性选择器7 [att*=val]
     - 包含val的
   - 伪类选择器
     - :link
     - :visited
     - :hover
     - :active
     - :enabled
     - :disabled
     - :checked
     - :first-child

        - :last-child
        - :nth-child(even)
        - :nth-child(odd)
        - :nth-child(3n+1)
        - :nth-last-child(4n+1)
        - :only-child
        - :first-of-type
        - :last-of-type
        - :nth-of-type(3n)
        - :nth-last-of-type(2n+2)
        - :only-of-type
        - :empty
        - :root
        - :not()
        - :target
        - :lang()
    - 组合简单选择器
      如：
        - img[src$=jpg]{}
        - #banner:hover{}

1.4.2.2 其他选择器

1. 简单选择器（上节已说过）
2. 伪元素选择器
   前面是两个冒号（CSS3前只有一个冒号）

    - ::first-letter{}
    - ::first-line{}
    - ::before{content:"abc"}
    - ::after{content:"xyz"}
    - ::selection 选中文字后的效果
3. 组合选择器

    - 后代选择器　　空格
    - 子选择器　　大于号
    - 兄弟选择器　　**都是向后看**
        - 加号　　相邻兄弟选择器
        - 波浪号　通用兄弟选择器
4. 选择器分组 用逗号把选择器分开即可

1.4.2.3 继承、优先级、层叠

1. 部分子元素会默认继承父元素：
   继承属性的：

    - color

- font
- text-align
- list-style
- ...

  非继承属性的：
- background
- border
- position
- ...

2. CSS优先级

   - 计算方法：
     - a = 行内样式
     - b = ID选择器的数量
     - c = 类、伪类和属性选择器的数量
     - d = 标签选择器和伪元素选择器的数量
   - `value = a*1000 + b*100 + c*10 + d` value较大的优先级高
   - 如果优先级相等，后面覆盖前面的
3. CSS层叠

   - 优先级
   - 后面覆盖前面
   - 不同的属性会合并
4. CSS改变优先级

   - 改变先后顺序
   - 提升选择器优先级
   - ！important

1.4.3 CSS文本

1.4.3.1 字体

1. **font-size:** `font-size:<length>|<percentage>|<absolute-size>|<relative-size>`

   - | 表示的是只出现一个
   - 例子：

   文字大小：font-size:12px;

   文字大小：font-size:16px;

   文字大小：font-size:2em;

   文字大小：font-size:200%;
2. **font-family:** `font-family: [ <family-name>|<generic-family> ]#`
   - `generic-family> = serif|sans-serif|cursive|fantasy|monospace

   字体系列：font-family:arial;

字体系列：font-family:arial,Verdana,sans-serif;

字体系列：font-family:Verdana, "microsoft yahei";

字体系列：font-family:“宋体”, serif;

> font-family 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。

1. **font-weight:** `font-weight: normal|bold|bolder|lighter|100|200|300|···|900` 字体粗细：font-weight:normal;

   字体粗细：font-weight: bold;
2. **font-style:** `font-style: normal|italic|oblique`
   - 后两者区别在于 二是基于字体有斜体的前提下，三是字体没有，强行倾斜

   字体风格：font-style:normal;

   字体风格：font-style: italic;
3. **line-height：** `line-height: normal|<number>|<length>|<percentage>`
   - 浏览器一般默认值为1.14左右（normal）

行高：line-height:40px;

行高：line-height:3em;

行高：line-height:300%;

行高：line-height:3;

另一个更直接的例子：

行高：line-height:300%;

行高：line-height:3em;

行高：line-height:300%;

行高：line-height:3em;

总结为：

- line-height中number为直接继承number
- 而percentage需要先计算成length再继承

4. 总结：

    font: [[<font-style>||<font-variant>||<font-weight>||<font-strench>]?<font-size>[/
    <line-height>]?<font-family>]|caption|icon|menu|message-box|small-caption|st
    atus-bar

    如：`font:30px/2 "Consolas",monospace;`
    如：`font:italic bold 20px/1.5 arial,serif;`
    这就错了：`font:100px;`
    至少要有font-size和font-family

5. color：

    - color:red;
    - color:#f00;
    - color:rgb(255,0,0);
    - color:rgba(255,0,0,1);a为透明情况由0到1.

1.4.3.2 对齐方式

1. text-align （水平线上）
    - `text-align:left|right|center|justify`

文本对齐：text-align: left;

文本对齐：text-align: right;

文本对齐：text-align: center;

文本对齐：text-align: justify文本对齐齐齐;

1.4.3.2 对齐方式

1. text-align

    - text-align:left|right|center|justify

文本对齐：text-align:left;qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq文本对齐

文本对齐：text-align:center;qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq文本对齐

文本对齐：text-align:right;qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq文

本对齐

文本对齐：text-align:just;qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq文本对齐

？？？ left 和 justify 的区别在哪儿？？？

2. vertical-align
    - vertical-align: baseline|sub|super|top|text-top|middle|bottom|text-bottom|<percentage>|<length>
        - baseline 文字基线
        - middle 居中
        - <length>以及<percentage> 表示以baseline为基线向上
        - sub 下标
        - super上标
        - text-top 文字最高点
        - top 当前最高点
3. text-indent
    - text-align: <length>|<percentage>
    - 首行缩进:text-indent:2em;首行缩进首行缩进首行缩进首行缩进首行缩进首行缩进首行缩进首行缩进首行缩进

首行缩进:text-indent:2em;首行缩进首行缩进首行缩进首行缩进首行缩进首行缩进首行缩进首行缩进首行缩进

1.4.3.3 格式处理

1. white-space
    - white-space:normal|nowrap|pre|pre-wrap|pre-line

	New Lines	spaces and Tabs	Text Wrapping
'normal'	Collapse	Collapse	Wrap
'nowrap'	Collapse	Collapse	No Wrap
'pre'	Preserve	Preserve	No Wrap
'pre-wrap'	Preserve	Preserve	Wrap
'pre-line'	Preserve	Collapse	Wrap

Preserve 保留
Collapse 合并
Wrap 换行

2. word-wrap
    - word-wrap:normal|break-word

      让过长的部分断开
3. word-break
    - word-break:normal|keep-all|break-all

      让过长的单词断开

1.4.3.4 文字修饰

1. text-shadow 阴影
    - text-shadow: none | [<length> {2,3} && <color> ?] #
2. text-decoration 线
    - text-decoration: none | [underline || overline || line-through]

1.4.3.5 高级设置

1. text-overflow
    - text-overflow:clip|ellipsis
    - 使用...表示未结束需要 `text-overflow:ellipsis;` 配合: `overflow:hidden;` 和 `white-space:nowrap;` 使用
2. cursor 鼠标
    - cursor: [<uri>,] * [auto | default | none | help | pointer | zoom-in | zoom-out | move]
3. inherit 强制继承

1.4.4 盒模型

1.4.4.1~2 盒模型

CSS盒模型

有外到里我们分别定义为 `margin`、`border`、`padding` 和 `content`
其中前三个都有 `top`、`right`、`bottom` 和 `left` 四个部分
第四个有 `width` 和 `height` 属性

margin合并！上下盒子如果都设置了margin，一个bottom一个top，则取较大的为两个盒子之间的距离

1. border-radius
   - border-radius: [<length> | <percentage>] { 1, 4 }[/ [<length> | <percentage>] { 1, 4 }] ?
2. overflow
   - overflow: visible | hidden | scroll | auto
3. box-sizing
   - box-sizing: content-box | border-box | inherit
4. box-shadow
   - box-shadow: none | <shadow> [, &shadow>] * <shadow> : inset ? && <length> { 2, 4 } && <color> ?
5. outline
   - outline: [<outline-width> || <outline-style> || <outline-color>] | inherit
   - outline的轮廓不占大小

1.4.5 背景

1.4.5.1-2 背景

1. background-color
   - `color`
     - transparent;
     - color;
     - #000
     - rgb(0,0,0);
     - rgba(0,0,0,1);
2. background-image
   - `< bg-image > [ , < bg-image > ] *`
   - `< bg-image > = < image > | none`
   - 引入方式用 url()
   - 可以有多个背景图片，靠前的靠上层，background-color 在最底层
3. background-repeat
   - `< repeat-style > [ , <repeat-style ] *`
   - `< repeat-style> = repeat-x | repeat-y | [ repeat | space | round | no-repeat ]{ 1, 2 }`
4. background-attachment
   - `<attachment> [ , < attachment > ] *`
   - `< attachment > = scroll | fixed | local`
   - 默认是scroll（固定）
   - local是跟随

- fixed是相对于整个窗口

5. background-position
   - `< position > [ , < position > ]*`
   - <position > = [left | center | right | <percentage> | <length>] | [left | center | right | <percentage> | <length>] [top | center | bottom | <percentage> | <length>] | [center | [left | right] [<percentage> | <length>] ?] && [center | [top | bottom] [<percentage> | <length>] ?]
     - 这里的 percentage 要注意，两个百分数的时候分别表示的是 x y 轴 image 和 框的百分值的位置，如：background-position：20% 50%表示的是 image 在横纵坐标分别为 20%和 50%处的点和框的横纵坐标的 20%和 50%的点重合。
6. linear-gradient()
   - `[[ < angle > | to < side-or-corner > ],] ? < color-stop >[,< color-stop >] +`
   - `< side-or-corner > = [ left | right ] || [ top | bottom ]`
   - `< color-stop > = < color > [ < percentage > | < length > ] ?`
7. radial-gradient()
   - [[circle || <length>] [at <position>] ?, | [ellipse || [<length> | <percentage>] {2}] [at <position>] ?, | [[circle | ellipse] || <extent-keyword>] [at <posion>] ?, | at <position> ,] ? <color-stop> [, <color-stop>] +
8. repeating-*-gradient * 换成上面的linear 或者 radial
9. background-origin
   - `<box> [ , <box> ] *`
   - `<box> = border-box | padding-box | content-box`
10. background-clip
    - `<box> [ , <box> ] *`
    - `<box> = border-box | padding-box | content-box`
11. background-size
    - `<bg-size> [ ,<bg-size> ]*`
    - <bg-size> = [<length> | <percentage> | auto] { 1,2 } | cover | contain
    - 百分比是相对容器的，cover是充满容器宽度，contain是充满容器高度
12. background
    - `[ <bg-layer , ] * <final-bg-layer>`
    - ``
    - <bg-layer> = <bg-img> || <position> [/<bg-size>] ? || <repeat-style> || <attachment> || <box> || <box>
    - `<final-bg-layer> = <bg-layer> || <'background-color'>`

1.4.6 布局

1.4.6.1 display

1. display

- block 块级元素
  - 默认宽度为父元素宽度
  - 可设置宽高
  - 换行显示
- inline
  - 默认宽度为内容宽度
  - 不可设置宽高
  - 同行显示
- inline-block
  - 默认宽度为内容宽度
  - 可设置宽高
  - 同行显示
  - 整块换行
- none
  - 设置元素不显示（不占空间）
  - 与visibility：hidden区别在于是否 **占空间**

2. 块层元素水平居中：

   - margin: auto;
   - text-align: center;（行级元素）

1.4.6.2 position

1. position 设置定位方式，重点在于参照物

   - A = top, right, bottom, left, z-index
   - static
   - relative
     - 仍在文档流中（保留了原来的位置）
     - 设置A的参照物为元素本身
     - 相对定位最常用的应用场景是给决定定位做参照物
   - absolute
     - 默认宽度为内容宽度
     - 脱离文档流
     - 设置A的参照物为第一个 **相对定位** 祖先 **/ 根元素**
   - fixed
     - 默认宽度为内容宽度
     - 脱离文档流
     - 参照物为视窗

2. 遮罩

   - .class{
     position: fixed;
     top: 0; left: 0;
     height: 100%;width: 100%;
     z-index:999;

```
}
```

- position 让遮罩覆盖于整个视窗
- top，left 指定了位置
- height，width 指定了大小
- z-index 设置为除了要突出显示的demo以外的第二大

1.4.6.3 float

1. float
   - left right
     - 默认宽度为内容宽度
     - 脱离文档流
     - 向指定方向一直移动
   - none
   - inherit
   - float的元素在同一文档流，则全都脱离文档流，但元素之间会按顺序排列并不会重叠
   - **float元素半脱离文档流（对元素，是脱离文档流的，但是对内容，在文档流之中）**
2. clear 清楚浮动很重要！！！
   - both
   - left right
   - none
   - inherit
   - 应用于后续元素
   - 应用于块级元素
   - 方法：
     - 空白元素
     - clearfix 在父元素中加入 class：clearfix即可清楚浮动

```
.clearfix:after{content: ".";display: block;clear: both;height: 0;
  overflow: hidden;visibility:hidden;}
.clearfix{zoom: 1;}IE低版本不支持 :after方法我们用zoom:1代替
```

1.4.6.4 flex

1. display: flex
   - flex item
     - 在文档流中的子元素（absolute绝对定位以及孙元素不是在文档流中的）
   - 方向
     - flex-direction
       - row column
       - row-reverse column-reverse
     - flex-wrap
       - nowrap
       - wrap

        - wrap-reverse
    - flex-flow
        - <'flex-direction'> || <'flex-wrap'>
        - 前两个的缩写，建议用这个而不是分开用上两个
    - order
        - <interger>
        - initial:0
        - order大的向后方
- 弹性
    - flex-grow
        - <number>
        - initial: 0
        - 将剩余空间按照grow的比例分配给元素
        - 最终的宽度为：

          ```
          flex-basis + flow-grow / sum( flow-grow ) * remain
          ```
    - flex-shrink
        - <number>
        - initial: 1
        - 与flex-grow类似也是在分摊剩余空间，只是这里的剩余空间为负值
        - 因为flex弹性布局在内部元素超过父级元素的时候会被压缩，因此此时的剩余空间会是负值，如果要按原式宽度，那么可以设置 flex-shrink: 0
    - flex-basis
        - main-size | <width>
        - 设置flex iterm的初始宽/高
    - flex
        - `flex: <'flex-grow'> || <'flex-shrink'> || <'flex-basis'>`
        - initial: 0 1 main-size
- 对齐
    - justify-content
        - 有剩余空间时的排布
        - `justify-content: flex-start | flex-end | center | space-between | space-around`
        - 分别表示：靠左（上），靠右（下），居中，分散对齐两端无空隙，分散对齐两端有空隙
    - align-items
        - 与justify-content类似，描述的是辅轴上的对齐方式
        - `align-items: flex-start | flex-end | center | baseline | stretch`
        - stretch为上下铺满baseline为基线对齐
    - align-self
        - `align-self: auto | flex-start | flex-end | center | baseline | stretch`
        - 设置单个flex item在cross-axis方向上对齐方式
    - align-content
        - `align-content: flex-start | flex-end | center | baseline | stretch`
        - 内部元素有多行的时候对辅轴上行的对齐方式

1.4.7 变形

1.4.7.1 2D变形

1. transform
   - `transform: none | <transform-function> +`
   - transform-function:
     - rotate() 旋转
       - `rotate( <angle> )`
     - translate() 移动
       - `translate ( <translation-value> [ , <transtlation-value> ] ? )`
     - scale() 缩放
       - `scale ( <number> [ , <number> ] ? )`
     - skew() 倾斜
       - `skew ( <angle> [ , <angle> ] ? )`
     - transform-origin 坐标圆心
       - ```
         transform-origin:
            [left | center | right | top | bottom | <percentage> | <length> ]
            |
            [left | center | right | <percentage> | <length> ]
            [top | center | bottom | <percentage> | <length> ] <length> ?
            |
            [center | [left | right]] && [center | [top | bottom]] <length> ?
         ```

1.4.7.2 3D变形

1. perspective
   - `perspective: none | <length>`
     此处的length表示人眼到图层的距离
   - perspective-origin 人眼位置
     - ```
       perspective-origin:
          [left | center | right | top | bottom | <percentage> | <length> ]
          |
          [left | center | right | <percentage> | <length> ]
          [top | center | bottom | <percentage> | <length> ]
          |
          [center | [left | right]] && [center | [top | bottom]]
       ```
2. translate3d()
   - `translate3d( <translation-value> , <translation-value> , <length> )`
3. scale3d()
   - `scale3d( <number> , <number> , <number> )`
4. rotate3d()
   - `rotate3d( <number> , <number> , <number> , <angle> )`
5. transform-style
   - `transform-style: flat | preserve-3d`
     (让一个transform元素 扁平化 | 保留3D 效果)

6. backface-visibility
    - `backface-visibility: visible | hidden`

1.4.8 动画

1.4.8.1 transition 过渡

1. transition-property （属性设置）
    - `transition-property: none | <single-transition-property> [ ',' <single-transition-property> ] *`
        - <single-transition-property> = all | <IDENT>
2. transition-duration （过渡时间）
    - `transition-duration: <time> [, <time> ] *`
3. transition-timing-function （定义时间函数）
    - `transition-timing-function: <single-transition-timing-function> [ ',' <single-transition-timing-function> ] *`
        - <single-transition-timing-function> = ease | linear | ease-in | ease-out | ease-in-out | cubic-bezier(<number>,<number>,<number>,<number>) | step-start | step-end | steps(<integer> [,[start | end]] ?)
4. transition-delay （延迟）
    - `transition-delay: <time> [, <time> ] *`

transition:

- `transition: <single-transition> [ ',' <single-transition> ] *`
    - <single-transition> = [none | <single-transition-property>] || <time> || <single-transition-timing-function> || <time>

1.4.8.2 animation

1. animation-name （动画名称）
    - `animation-name: <single-animation-name> [ ',' <single-animation-name> ] *`
        - <single-animation-name> = none | <IDENT>
2. animation-duration （过渡时间）
    - `animation-duration: <time> [, <time> ] *`
3. animation-timing-function （定义时间函数）
    - `animation-timing-function: <single-timing-function> [ ',' <single-timing-function> ] *`
        - <single-timing-function> = <single-transition-timing-function>
4. animation-iteration-count （循环次数）
    - `animation-iteration-count: <single-animation-iteration-count> [ ',' <single-animation-iteration-count> ] *`
        - <single-animation-iteration-count> = infinite | <number>
5. animation-direction （动画方向）

- `animation-direction: <single-animation-direction> [ ',' <single-animation-direction> ] *`
  - <single-animation-direction> = normal | reverse | alternate | alternate-reverse

6. animation-play-state （播放状态）
   - `animation-play-state: <single-animation-play-state> [ ',' <single-animation-play-state> ] *`
     - <single-animation-play-state> = running | paused
7. animation-delay （延迟）
   - `animation-delay: <time> [, <time> ] *`
8. animation-fill-mode （开始和结束后是否保持第一帧和最后一帧状态）
   - `animation-fill-mode: <single-animation-fill-mode> [, single-animation-fill-mode> ] *`
     - <single-animation-fill-mode> = none | backwards | forwards | both

animation:

- `animation: <single-animation> [ ',' <single-animation> ] *`
  - <single-animation> =
    <single-animation-name> || <time> || <single-animation-timing-fu nction> || <time> || <single-animation-iteration-count> || <single-animation-direction> || <single-animation-fill-mode> || <single-animation-play-state>

@keyframes

-
-

这样我们就可以使用之前的animation调用这个关键帧：