H5基础与新增内容

1.基础语法

- 文件类型, 文件扩展名仍然为 **html** 或 **htm**
- DOCTYPE 声明不区分大小写
- 指定使用 **UTF-8** 为字符编码集
- 一些元素可以省略标记
  不允许写结束标记: **<hr>
 <link> <img> <meta> <input> <col> <embed>**
  可以省略结束标记: ** <dt> <dd> <p> <option> <colgroup> <thead> <tbody> <tfoot> <th> <tr> <td>**
  可以省略全部标记: **<tbody> <colgroup> <html> <body> <head>**
- 可以省略引号, 标签属性值可以使用单引号, 也可以使用双引号

2.H5新增内容

- 一系列语义化标签
- 增强型的表单
- 允许嵌入网页的 **音频 <audio> 与视频 <video>**
- **canvas** 绘图
- **webstorage** 与 **sessionstorage** 存储
- 新的 API

2.1新增语义化标签

- **<header>** : 表示段落的头部
- **<main>** : 表示段落的主体区域
- **<footer>** : 表示段落的底部
- **<section>** : 表示一个独立的主体区域
- **<article>** : 表示一个博客、文章的内容区域
- **<nav>** : 表示导航栏区域
- **<aside>** : 表示侧边栏区域
- **<figure>** : 表示一个独立区域, 可以使用 **<figurecaption> 设置独立区域标题**

> 以上的新增标签均是**双标签**, 并且都是**块级元素**, 用法类似普通的 ** ** 标签

- **<canvas>** : 表示一个绘图区域
- **<hgroup>** : 为文章标题分组
- **<mark>** : 为文本显示高亮

> 以上新增标签并未作为经常使用的标签

- **<audio>** : 为页面添加音频

> 在 <audio> 标签中, 有以下一些标签属性(**可以使用属性代替属性值**):
>
> > **controls** : 用来实现对音频的控制(**必须属性**)
> > **loop** : 让音频实现重复播放
> > **muted** : 设置禁音
> > **autoplay** : 在不同的浏览器中支持不一, chrome 浏览器不支持, firefox 浏览器支持

- **<video>** : 为页面添加视频

> 在 <video> 标签中, 有以下一些标签属性(**可以使用属性代替属性值**):
>
> > **controls** : 用来实现对视频的控制
> > **loop** : 让视频实现重复播放
> > **muted** : 让视频禁音
> > **autoplay** : 让视频实现自动播放, 在设置该属性时, 需要使用 **muted** 属性配合实现
> > **poster** : 让视频实现加载效果, **该属性必须跟同名属性值**

2.2增强型表单

增强型表单仍然使用 **<input>** 标签, 但是需要注意标签属性 **type** 的取值
在这些表单中, 可以使用 **required** 属性进行**强制填写**

- **type = "color"** : 表示一个取色板控件
- **type = "search"** : 表示一个输入搜索框, 当输入有内容时, 右侧会出现清除按钮
- **type = "number"** : 表示一个数字输入框, 右侧会有上下选择箭头. 特别地, 该类型输入框中, 有额外标签属性
  min : 设置一个最小值
  max : 设置一个最大值
  value : 设置一个默认值
  step : 设置步长

- **type = "range"** : 表示一个数值滑块. 特别地, 该类型输入框中额外标签属性与 **type = "number"** 的额外标签属性一致
- **type = "date"** : 表示一个日历选择控件, 具体到日
- **type = "month"** : 表示一个日历选择控件, 具体到月
- **type = "week"** : 表示一个日期选择控件, 具体到周
- **type = "time"** : 表示一个时间选择控件, 具体到分
- **type = "datetime-local"** : 表示一个本地时间日期选择控件
- **type = "tel"** : 表示一个拨号盘控件
- **type = "email"** : 表示一个**带有电子邮件格式验证的文本框**, 要实现验证功能, 需要配合 **<form>** 标签使用. 可以用下面的方法设置一次性上传多个邮箱地址:
  multiple : 设置该属性允许一次性上传多个电子邮箱, 并且**以半角逗号标点隔开**
- **type = "url"** : 表示一个**带有网址格式验证的文本框**, 要实现验证功能, 需要配合 **<form>** 标签使用

2.2.1增强型表单的标签属性k

- **min** : 最小值, 用于 **type = "number"** 与 **type = "range"** 中
- **max** : 最大值, 用于 **type = "number"** 与 **type = "range"** 中
- **step** : 步长值, 用于 **type = "number"** 与 **type = "range"** 中
- **multiple** : 用于 **type = "email"** 中, 实现一次性提交多个邮箱, 邮箱用半角逗号分隔
- **value** : 表示值的意思, **实际占位文本**
  用于 **type = "text / number / range"等普通输入框, 表示输入框中的默认文本**
  用于 **type = "submit / button", 表示显示在按钮上的提示文本**
  用于 **type = "radio / checkbox"**, 方便 js 操作**
  用于 **<option>** 标签中, 方便 js 操作**
- **placeholder** : **虚拟占位文本**
- **required** : 强制输入必填项
- **disabled** : 禁止输入文本
- **readonly** : 只读文本
- **pattern** : 输入文本验证, 该属性的值是一个正则表达式
- **autofocus** : 自动获取焦点
- **autocomplete** : 自动填充文本, **需要应用在对应的 <form> 标签中**
  on : 开启自动填充文本
  off : 关闭自动填充文本
- **list** : 用于模糊搜索, 属性值应当为 **<datalist>** 标签的 **id** 值
  <datalist> 标签, 用来表示 **list** 属性中的值, 通常使用一个 **id** 属性值, 让 **list** 属性的取值为该 **id** 值, 它的内容由多个 **<option>** 标签组成

3.CSS3

CSS3 是朝着模块化发展的一门语言, 在新的 CSS3 中, 有 **盒子模型、列表模块、超链接方式、语言模块、背景和边框、文字特效、多栏布局等**

优点 : 向后兼容, 有新的特性

3.1渐进增强与优雅降级

渐进增强: 针对低版本浏览器构建页面, 保证基本布局, 再针对高版本浏览器添加额外的特效、交互等效果.
优雅降级: 先构架完整功能的页面, 再对低版本浏览器进行兼容.

> **区别: 优雅降级是从复杂的页面开始, 逐渐降低用户的体验供给; 渐进增强是指先构建一个最基础的、能够起作用的页面, 再不断进行扩充.**

3.2CSS3新增选择器

属性选择器(多用在表单制作时).

> **[propName]** : 中括号中间是一个属性名, 表示选择带有该属性名的所有元素
> **[propName = "value"]** : 中括号中间是一个属性名值对, 表示选择带有该属性名值对的所有元素
> **ElementName[propName]** : 中括号左边是一个元素名, 中括号中间是一个属性名, 表示选择带有该属性名的所有对应元素名的元素
> **ElementName[propName = "value"]** : 中括号左边是一个元素名, 中括号中间是一个属性名值对, 表示选择带有该属性名值对的所有对应元素名的元素
> **[propName ^= "keyword"]** : keyword 代表一个关键词, 表示选择属性名是**以该 keyword 开头**的所有元素
> **[propName $= "keyword"]** : keyword 代表一个关键词, 表示选择属性名是**以该 keyword 结尾**的所有元素
> **[propName *= "keyword"]** : keyword 代表一个关键词, 表示选择属性名中**包含该 keyword**的所有元素
> **[propName ~= "keyword"]** : keyword 代表一个关键词(通常是一个完整的属性值), **当属性值是多个值时(class属性)**, 该选择器选择这些之中**包含有该 keyword** 的所有元素
> **[propName |= "keyword"]** : keyword 代表一个关键词(通常是一个完整的属性值), **当属性值是以该 keyword 或者 keyword- 开头时**, 选择器选中所有这些元素

结构伪类选择器.

> **:first-child** : 对于有共同父元素的一组子元素, 选择子元素中的第一个元素. 等同于 **:nth-child(1)**
> **:last-child** : 对于有共同父元素的一组子元素, 选择子元素中的最后一个元素. 等同于 **:nth-last-child(1)**
> **:nth-child()** : 对于有共同父元素的子元素, **当括号内为数字时, 选择对应顺序的元素**; **当括号内为表达式时(2n,2n+1,3n,4n等), 选择对应计算位置的元素**. 特别地, 在小括号内可以使用关键词 **odd(表示奇数, 等同于 2n+1 或者 2n-1)、even(表示偶数, 等同于 2n)**
> **:nth-last-child()** : 效果与 **:nth-child()** 类似, 从父元素末尾起算
> **:empty** : 表示选择一个空标签, 即标签中**没有任何文本(包括空格与换行)**
> **:only-child** : 表示选择一个**对应的子元素, 并且该子元素的父元素仅有这唯一一个子元素**
> **:only-of-type** : 表示选择一个**对应的子元素, 并且该子元素是父元素中的唯一的元素类型(父元素可以有多个其他元素)**
> **:first-of-type** : 表示选择父元素下**同类型元素中的第一个元素**, 等同于 **:nth-of-type(1)**, **注意与 :first-child** 的区别
> **:last-of-type** : 表示选择父元素下**同类型元素中的最后一个元素**, 等同于 **:nth-last-of-type(1)**, **注意与 :last-child** 的区别
> **:nth-of-type()** : 表示选择父元素下**同类型元素中的第几个元素**
> **:nth-last-of-type()** : 表示选择父元素下**同类型元素中的倒数第几个元素**
> **:root** : 表示当前文档的根元素(<html>)

UI状态伪类选择器.

> **input:enabled** : 在表单中, 选中未设置 **disabled** 属性的元素(**radio** 类型与 **checked** 类型无法实现, 有自己的默认样式)
> **input:disabled** : 在表单中, 选中设置了禁用属性 **disabled** 的元素
> **input:checked** : 在表单中, 修改元素的选中样式
> **::selection** : 修改文本默认的选中样式, 只允许修改 **background-color** 与 **color**
> 在表单中, 设置 **appearance** 为 **none**, 用来清除元素的默认样式

否定伪类选择器.

> **:not()** : 选择除了小括号中的元素

目标伪类选择器.

> **:target** : 选择页面中当前活动的目标元素, 一般地, 配合超链接元素 <a> 使用, 当用户点击超链接, 跳转至当前页面的对应位置时, 对应的目标元素会被该伪类选中

关系选择器.

> **后代选择器 " "** : 形如 **选择器1 选择器2**
> **子代选择器 ">"** : 形如 **选择器1 > 选择器2**

> **相邻兄弟选择器 "+"** : 形如 **选择器1 + 选择器2**, 选择选择器1后面**相邻的**符合选择器2的兄弟元素
> **普通兄弟选择器 "~"** : 形如 **选择器1 ~ 选择器2**, 选择选择器1后面**所有的**符合选择器2的兄弟元素

3.3浏览器属性前缀

浏览器内核.

> **Trident内核** : IE、Maxthon、Tencent、TheWorld、360. **-ms-**
> **Gecko内核** : Mozilla Firefox. **-moz-**
> **Webkit内核** : Chrome、Safari. **-webkit-**
> **Presto内核** : Opera. **-o-**
> **Blink内核** : Webkit 的分支

3.4文本阴影与元素盒子阴影

文本阴影 : text-shadow.

> **text-shadow : h-shadow v-shadow blur color**. 若要同时添加多个阴影, 使用 **逗号 ","** 分隔
> **h-shadow** : 表示阴影的水平方向偏移量. 正数为右, 负数为左
> **v-shadow** : 表示阴影的垂直方向偏移量. 正数为下, 负数为上
> **blur** : 表示阴影的模糊距离
> **color** : 表示阴影的颜色

元素盒子阴影.

> **box-shadow : h-shadow v-shadow blur size color inset**. 若要同时添加多个阴影, 使用 **逗号 ","** 分隔
> **h-shadow** : 表示阴影的水平方向偏移量. 正数为右, 负数为左
> **v-shadow** : 表示阴影的垂直方向偏移量. 正数为下, 负数为上
> **blur** : 表示阴影的模糊距离
> **size** : 表示阴影的延展半径
> **color** : 表示阴影的颜色
> **inset** : 添加该属性表示内阴影, 默认情况下是外阴影

3.5大小写转换与小型大写字符

英文字母大小写转换.

> **text-transform** : 利用该属性对英文字母进行大小写转换. 可以从 **capitalize、uppercase、lowercase、none** 中选择取值

小型大写字符.

> **font-variant** : 利用该属性值 **small-caps** 实现对小写字母转变成小型的大写字母

3.6圆角属性

复合属性.

> **border-radius** : 一次性设置四个角的圆角参数
> **border-radius : vpx** : v 值代表四个角的圆角参数
> **border-radius : v1px v2px** : v1 值代表左上角与右下角的圆角参数, v2 值代表右上角与左下角的圆角参数
> **border-radius : v1px v2px v3px** : v1 值代表左上角的圆角参数, v2 值代表右上角与左下角的圆角参数, v3 值代表右下角的圆角参数
> **border-radius : v1px v2px v3px v4px** :四个值分别代表左上、右上、右下与左下的值
> 在设置圆角参数时, 可以指定水平半径与垂直半径:
>
> > **border-radius : x / y** : 表示一次性设置四个角的水平半径与垂直半径
> > **border-radius : x1 x2 x3 x4 / y1 y2 y3 y4** : 斜杠左侧分别设置左上、右上、右下与左下的水平圆角参数， 斜杠右侧分别设置左上、右上、右下与左下的垂直圆角参数

单角设置.

> **border-top-left-radius** : 设置左上角的圆角参数
> **border-top-right-radius** : 设置右上角的圆角参数
> **border-bottom-left-radius** : 设置左下角的圆角参数
> **border-bottom-right-radius** : 设置右下角的圆角参数

实际应用.

> 当对一个正方形应用 **border-radius** 属性时, 若取值为其边长(包括内容区域长度与边框长度)的一半时, 会变成圆形. 或者直接使用 50% 取值也可以变成圆形
> 若要在长方形中实现类似跑道样式的形状, 需要将 **border-radius** 属性值设置为较短边(包含内容与边框)的一半即可

3.7外部字体与字体图标

@font-face.

> 使用该属性可以引入外部字体, 通常将字体文件放在网站文件中

> 使用该属性需要遵循以下格式:
> **@font-face : { font-family:customedName ; src:url (filePath) }**

使用字体图标.

> 操作字体图标能够像操作文字一样, 使用字体图标也需要将字体文件放在网站文件中
> 一般地, 使用字体图标需要遵循图标文件中的参考格式

3.8线性渐变

background : linear-gradient(to direction1 direction2, colorBegin, ... , colorEnd). 利用该方法实现线性渐变

> **to direction1 direction2** 两个**方向词**是可选的, 方向词可以从 **left、right、top、bottom** 中进行选择
>
> > 当不添加方向词时, 默认地, 是从上到下进行渐变
> > 当添加一个方向词时, 实现**水平方向**或者**垂直方向**的渐变
> > 当添加两个方向词时, 不区分顺序, 实现**斜向渐变**
>
> **direction** 也可以换作**角度值(deg)**
>
> > 默认地, 当仅仅使用颜色参数时, 不指定角度与方向时, 角度值 deg=180
>
> 其中的 **colorBegin** 指的是开始的颜色, **colorEnd** 指的是结束的颜色, 中间可以添加任意数量的颜色

background : repeating-linear-gradient(to direction1 direction2, color1 num%, color2 num%, color3 num%, ...). 利用该方法实现重复线性渐变

> **to direction1 direction2** 使用方法与上述一致
> **color num%** 表示在该相对位置处的颜色, 相对位置根据渐变的方向确认

3.9径向渐变

background : radial-gradient(positionX positionY, shape size, colorBegin, ... , colorEnd) 径向渐变指的是以某个坐标为中心, 以圆形的形式向外层渐变, **注意 shape 属性与 size 属性之间用空格分隔**

> **position** 指的是渐变中心起始坐标, 默认地, 如果省略该属性, 则以中心点为渐变中心, **注意需要考虑游览器的兼容性, 添加浏览器厂商前缀**, 单位可以为百分比
> **shape** 指的是渐变的形状, 默认地, 如果省略该属性, 则默认为椭圆 **ellipse**, 可以自行设置为 **circle**
> **size** 指的是渐变的结束位置, 一般地, 要指定渐变的结束位置, 首先需要自行设置渐变中心

> **closest-side** : 指到最近的边结束渐变
> **closest-corner** : 指到最近的角结束渐变
> **farthest-side** : 指到最远的边结束渐变
> **farthest-corner** : 指到最远的角结束渐变

colorBegin, ..., colorEnd : 指的是渐变的颜色, **colorBegin** 指的是渐变开始的颜色, **colorEnd** 指的是渐变结束的颜色, 中间可以接任意数量的颜色

background : repeating-radial-gradient(position, shape size, colorBegin num%, ... , colorEnd num%). 利用该方法实现重复径向渐变

> **position、shape、size** 的用法与上述一致
> **color num%** 表示在该相对位置处的颜色, 相对位置根据径向渐变的 **size** 属性确定

3.10怪异盒子、弹性盒及多列布局

标准盒子模型.

> 标准盒子模型的 **box-sizing** 属性取值为 **content-box**, 默认地, 元素是标准盒子模型
> 在标准盒子模型中, **增加内边距 padding 、边框 border 以及外边距 margin 会使得盒子的 width 与 height 增大, 盒子的 padding 与 border 会在指定的 width 与 height 外添加, 向外扩展**

怪异盒子模型.

> 在怪异盒子模型中, **box-sizing** 属性取值为 **border-box**
> 在怪异盒子模型中, **增加内边距 padding 与边框 border 不会使得盒子的 width 与 height 增大, 但是增加盒子的外边距 margin 会使得盒子的 width 与 height 增大. 盒子的 padding 与 border 会在指定的 width 与 height 内添加, 向内扩展**
> 怪异盒子模型多用于移动端布局中

弹性盒.
当页面需要在不同设备上显示时, 保证页面中的元素拥有恰当的布局方式的一种方法. **在弹性盒中, 元素是按照轴线进行排列的**

> **弹性盒的基础用法.**
> 要实现弹性盒布局, 需要**给父元素添加 display : flex** 属性
> **1.添加弹性盒之后, 父元素中的子元素默认会横向排列**
> **2.父元素中的行内元素会被更改为块级元素**
> **3.当父元素中只存在一个子元素时, 为该子元素添加 margin : auto 属性时, 子元素会实现垂直居中显示**

> **弹性盒的基础概念.**
> **flex容器** : 即设定了 **display : flex** 的父元素
> **flex项目** : 即设定了 **display : flex** 的父元素中的子元素
> **横轴** : 即水平方向
> **竖轴** : 即垂直方向
> **主轴** : 即 flex 项目的排列方向
> **侧轴** : 即与主轴对应的另外一个

flex 容器属性.

> **display : flex** : 必须属性, 将父元素设置为 flex 容器
> **flex-direction** : 设定 flex 容器的主轴方向, **row** 表示横向, **column** 表示纵向, **row-reverse** 表示横向反转, **column-reverse** 表示纵向反转
> **justify-content** : 设定**主轴上** flex 项目之间的排列方式(列间距)
>
> > **flex-start** : 让 flex 项目整体从主轴头部开始排列
> > **flex-end** : 让 flex 项目整体从主轴尾部开始排列
> > **center** : 让 flex 项目整体在主轴居中
> > **space-around** : 让 flex 项目实现分散对齐, 即各个 flex 项目之间的左间距与右间距相等
> > **space-between** : 让 flex 项目实现两端对齐, 即两端的项目紧贴主轴边侧, 内部项目平分间距
> > **space-evenly** : 让 flex 项目实现多余空间均分
>
> **align-items** : 设定**侧轴上** flex 项目的对齐方式(未折行, **单行侧轴对齐方式**)
>
> > **flex-start** : 让 flex 项目在侧轴头部排列
> > **flex-end** : 让 flex 项目在侧轴尾部排列
> > **center** : 让 flex 项目在侧轴居中排列
> > **stretch** : 让 flex 项目在侧轴拉伸
> > **baseline** : 让 flex 项目在侧轴基线位置处排列, 默认与 **flex-start** 效果一致
>
> **flex-wrap** : 当 flex 项目宽度超出了 flex 容器之后, 设定 flex 项目是否折行显示. 默认地不进行折行(**nowrap**), 并且项目的宽度会被挤压. 当添加了折行属性后(**wrap**), 超出宽度的 flex 项目会折行显示, **并且每一项 flex 项目的高度会被均分**
> **align-content** : 设定**侧轴上** flex 项目的行间距(有折行, **多行侧轴对齐方式**)
>
> > **flex-start** : 让 flex 项目在侧轴头部排列
> > **flex-end** : 让 flex 项目在侧轴尾部排列
> > **center** : 让 flex 项目在侧居中排列
> > **space-around** : 让 flex 项目在侧轴分散对齐排列
> > **space-between** : 让 flex 项目在侧轴两端对齐排列

flex 项目属性.

> **order** : 设定 flex 项目的顺序, 值越大 樾靠后排列. **默认地, 所有 flex 项目的 order 值为 0**
> **flex** : 设定 flex 项目的宽度与高度比例
>
> > 当 **flex-direction** 取值为 **row** 时(默认), 如果**不给 flex 项目添加高度 height, 高度会被拉伸**. 可以使用 **flex : 1** 实现按比例划分 flex 项目的**宽度**, 当所有 flex 项目使用**相同的 flex 取值时, 父元素宽度被均分**. 如果**其余元素有固定宽度, 给剩余元素添加 flex : 1, 可以让该元素实现占剩余宽度的所有**
> > 当 **flex-direction** 取值为 **column** 时, 如果**不给 flex 项目添加宽度 width, 宽度会被拉伸**. 可以使用 **flex : 1** 实现按比例划分 flex 项目的**高度**, 当所有 flex 项目使用**相同的 flex 取值时, 父元素高度被均分**. 如果**其余元素有固定高度, 给剩余元素添加 flex : 1, 可以让该元素实现占剩余高度的所有**
>
> **flex-shrink** : 设定 flex 项目的宽度超出 flex 容器的宽度时是否折行是否挤压. 当**取值为 1 时**, 默认会产生挤压或者折行; 当**取值为 0 时**, 不会产生挤压与折行
> **align-self** : 设定单个 flex 项目的侧轴排列方式
>
> > **flex-start** : 让 flex 项目在侧轴的头部排列
> > **flex-end** : 让 flex 项目在侧轴的尾部排列
> > **center** : 让 flex 项目在侧轴居中排列
> > **stretch** : 让 flex 项目在侧轴拉伸
> > **baseline** : 让 flex 项目在侧轴基线排列, 效果与 **flex-start** 类似

多列布局.

> 要创建多列布局, 需要有父元素与子元素. 多列布局可以实现不规则图片平铺效果
> **应用在父元素上的属性**
>
> > **column-count** : 设定布局列的数量
> > **column-gap** : 设置列间距
> > **column-rule** : 设置列分隔线, 默认地, 该分隔线贯穿整个父元素高度, 相当于边框线
> > **column-width**: 设定列的宽度, 默认地, **列宽是均分的**, 当设定的列宽比较小时, 不会有任何作用; 当设定的列宽过大, 则会**减少列数**
> > **column-fill** : 设置是否占满整列的高度再换列. 取值为 **balance** 时不占满整列, 取值为 **auto** 时占满整列
>
> **应用在文本元素上的属性.**
>
> > **column-span** : 设定该文本是否横跨所有列. 通常用作显示标题. 取值为 **none** 时不横跨多列, 取值为 **all** 时横跨所有列

> > **break-inside : avoid** : 让子元素不进行折行显示

3.11移动端布局

UI设计师在给设计稿的时候, 均为倍图, 要么是1倍图, 要么是2倍图, 要么是3倍图
可以在浏览器的设备模拟器中, 使用捕获当前屏幕与标注的设备尺寸得出设备像素比(dpr)

> **设备像素比 dpr = 物理像素 / CSS 像素**
> **物理像素** : 即实际设备像素, 是设计稿中的大小
> **CSS像素** : 即虚拟像素, 是需要编写 CSS 代码使用的像素
> 当使用屏幕捕获或者 PS 测量时, 获取的是物理像素, 前端开发人员**编写 CSS 代码的像素应当是: 测量出的像素 / 设备像素比(dpr)**
> 不能直接将 PC 端布局载入移动端布局

理解视口.

> **布局视口** : 编写的页面
> **视觉视口** : 屏幕窗口
> **理想视口** : 理想化的视口, 通过下列代码, 将编写的适应 PC 端的视口应用在移动端的视口中
>
> > **<meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0. maximum-scale=1.0, user-scalable=no"/>**
> > **width=device-width** : 表示页面宽度
> > **initial-scale** : 表示初始缩放比例
> > **minimum-scale** : 表示最小缩放比例
> > **maximum-scale** : 表示最大缩放比例
> > **user-scalable = no** : 禁止使用用户缩放

移动端布局的方法.

> **流式布局**
> **百分比布局**
> **弹性盒布局**
> **弹性盒 + rem 布局**
> **rem 布局**
> **vw 与 vh 布局**
> **flexble.js 布局**
> **响应式布局**

移动端布局的步骤.

> 确定设计稿来自于哪个屏幕
> 计算 dpr, 根据 dpr 确定 CSS 像素
> 添加元数据标签, 引入理想视口
> 引入 icon 图标文字与 CSS 样式表
> 编写 CSS 样式

3.12响应式布局及单位转换

常见的布局方案: **固定布局、弹性布局、混合布局**
固定布局: 指定实际的像素宽度与高度
弹性布局: 利用百分比或者弹性盒子布局
混合布局: 可以在指定部分元素宽度与高度的情况下, 同时使用弹性布局
响应式布局: 针对相同内容对不同的屏幕宽度进行布局设计. 包含**PC优先**与**移动优先**

> 要实现响应式布局, 需要考虑**三个方面**:
>
> > 1. 网站有灵活的网格基础, 即按块划分
> > 2. 网页中的图片必须可伸缩
> > 3. 不用的显示风格
>
> 响应式布局的**分类**
>
> > 1. 挤压-拉伸(基本布局样式不变)
> > 2. 换行-平铺(基本布局样式不变)
> > 3. 删减-增加(基本布局样式不变)
> > 4. 模块位置改变(基本布局样式改变)
> > 5. 隐藏-展开(基本布局样式改变)
>
> 响应式布局的**优劣**
>
> > **优点:** 能够很好地适配不同设备的屏幕分辨率
> > **缺点:**
> >
> > > 兼容多设备, 工作量大
> > > 代码累赘, 隐藏多余的无用元素, 加载时间长
> > > 折中性方案, 多方面影响打不到最佳效果的实现
> > > 会改变原有布局, 造成用户困惑

媒体查询.
使用媒体查询, 需要利用媒体查询语句:
@media screen and (max-width : num px)

> **@media**: 指定义媒体查询
> **screen**: 指定义媒体查询的依据
>
> > **all**: 所有设备
> > **aural**: 听觉设备
> > **braille**: 点字触觉设备
> > **handled**: 便携设备, 如手机, 移动电脑
> > **print**: 打印预览图
> > **screen**: 屏幕(常用)
> > **projection**: 投影设备
> > **tty**: 打字机
> > **tv**: 电视机
> > **embossed**: 盲文打字机
>
> **and**: 关键词, 还包括 **not 、only**
> **max-width**: 指定媒体特性, 其数值称为**断点**, 一般放在小括号中, 满足则执行后面的样式语句. **需要注意的是, 在关键词与媒体特性之间需要使用空格分隔**. 当存在多个断点时, 断点处的样式是在**max-width** 处实现的
> 利用括号中的特性还能实现横屏与竖屏的样式: **(orientation : portrait) 表示竖屏**, **(orientation : landscape) 表示横屏**

布局常用单位.
包括 **px、em、rem、vh、vw、%**

> **px**: 固定单位, 像素宽度
> **em**: 相对单位, 相对于父元素的字体大小, 默认地, **1em = 16px**
> **rem**: 相对单位, 相对于根标签的字体大小, 默认地, **1em = 16px**
> **vh、vw**: 相对单位, 相对于视口大小, **100vh** 表示整个视口高度, **100vw** 表示整个视口宽度
> **%**: 相对单位, 相对于父元素的大小

单位转换.

> **px 与 rem 的转换**
> 在移动端布局中, 使用 **rem** 单位进行布局的时候, 注意是相对于根标签的字体大小, 当确定好设备像素比(dpr)后, 通过设置根标签的字体大小, 使得计算CSS像素更加便捷
> 通常, 将根目录字体大小设置成 **50px**, 当**设备像素比是 2 时**, **用物理像素直接除以 100 即可**
>
> **px 与 vw、vh 的转换**
> 需要了解设备视口的宽度与高度进行换算

3.13rem布局

使用 rem 布局需要**自行设置根标签的字体大小, 一般地, 当设备像素比 dpr=2 时, 设置根标签的字体大小为 50px 方便计算**

3.14flexble.js布局

使用 flexble.js 布局, 需要额外引入 flexble.js 文件
引入文件后, **不需要自行设置根标签的字体大小, 并且不再需要自行设置元数据标签, 在测量好物理像素后, 直接使用该数值并且除以 100 即可转换成为 CSS 像素**

3.15vw布局

使用 vw 布局, 需要使用吃插件 **px-to-vw**, 在测量好物理像素后, 使用 **alt+z** 对目标数值进行转换

3.16过渡/动画/2D与3D

CSS过渡.(需要配合鼠标事件触发动画)

> **transition : all time1 model time2**: 复合属性, 为元素添加过渡效果. 如果希望在触发动画之前或者之后均实现过渡, **直接在初始样式上添加 transition**. tips:display、背景渐变属性不参与过渡.
>
> > **all**: 表示为所有参与过渡的属性
> > **time1**: 表示执行过渡所持续的时间
> > **model**: 表示执行过渡的模式
> > **time2**: 表示延迟执行过渡的时间
>
> **transition-delay**: 设置延迟执行过渡的时间
> **transition-duration**: 设置执行持续过渡的时间
> **transition-timing-function**: 设置执行过渡的模式
>
> > **ease**: 先慢速, 再快速, 再减速
> > **ease-in**: 先慢速, 再快速
> > **ease-out**: 先快速, 再慢速
> > **ease-in-out**: 慢速开始, 再加速, 慢速结束
> > **linear**: 匀速
> > **step(num)**: 以步长进行, 制作逐帧动画
> > **step-start**: 以开始图像作为关键帧, 适用于**定格动画**
> > **step-end**: 以结束图像作为关键帧, 适用于**定格动画**
> > **贝塞尔曲线**

> **transition-property**: 设置进行过渡的属性, 包括 **width、height、background-color、border-radius等**, 使用 **all** 包含全部能够过渡的属性

2D效果.(需要配合鼠标事件触发动画)

> **transform**: 设置元素的 2D 效果, 可以应用以下属性
>
> > **translate(x y)**: 复合属性, 设置元素的平移距离, x 表示水平方向平移距离, y 表示垂直方向平移距离
> > **translateX()**: 设置元素水平方向的平移距离
> > **translateY()**: 设置元素垂直方向的平移距离
> >
> > **transform-origin : x y**: **设置元素旋转中心、缩放中心的位置**, x 表示水平方向的位置, y 表示垂直方向的位置, 可以设置负数. 也可以使用**关键词**来定义中心位置: **left、center、right、top、center、bottom**. **从元素的左上角开始计算位置**. **该属性需要放在原始样式中**
> >
> > **rotate(deg)**: 设置元素绕中心点的旋转度数
> > **rotateX(deg)**: 设置元素绕X轴的旋转度数
> > **rotateY(deg)**: 设置元素绕Y轴的旋转度数
> >
> > 在缩放中, num 的**绝对值**取值为 >1 的数值时, 表示放大; num 的**绝对值** >0 并且 <1 时, 表示缩小; num 的值 =1 时, 表示原大小. **负数表示向相反的方向进行缩放**
> > **scale(num)**: 复合属性, 设置元素中心缩放的效果
> > **scaleX(num)**: 设置元素沿着X轴方向的缩放效果
> > **scaleY(num)**: 设置元素沿着Y轴方向的缩放效果
> >
> > **skew(Xdeg Ydeg)**: 设置元素在X轴与Y轴上的倾斜效果
> > **skewX(deg)**: 设置元素在X轴上的倾斜效果
> > **skewY(deg)**: 设置元素在Y轴上的倾斜效果

transform多样式应用. 使用 transform 属性, 后面可以接多个属性值, 例如 **transform : translate() rotate() skew()**, 多个属性值用**空格**分隔

> 注意, 使用不同的组合方式会造成不同的结果. **旋转会导致X轴与Y轴的方向改变, 所以先平移再旋转与先旋转再平移会造成不同的结果**

关键帧动画.(不需要配合鼠标事件触发动画)

> **@keyframes**: 使用 **@keyframes** 定义动画名称 (name) 以及需要执行的**动画效果**或者**关键帧(第二种)**

```
/* 第一种方法 */
@keyframes name{
  from{
    /* 初始状态代码块 */
    /* 当元素的初始状态与已定义的样式一致时，可以不用编写初始状态的代码块 */
  }
  to{
    /* 结束状态代码块 */
  }
}

/* 第二种方法 */
@keyframes name{
  0%{
    /* 初始状态代码块 */
    /* 当元素的初始状态与已定义的样式一致时，可以不用编写初始状态的代码块 */
  }
  /* 中间状态，可以定义多个中间阶段的过渡样式 */
  100%{
    /* 结束状态代码块 */
  }
}
```

> **animation**: 复合属性.
>
> > 基本格式为: **animation : aniName time1 model time2 number alternate**
> > **aniName**: 使用该属性设置元素应用的关键帧名称, 名称是使用上述 **@keyframes** 定义的 **name**
> > **time1**: 动画执行时间
> > **model**: 动画执行模式
> > **time2**: 动画延迟执行时间
> > **number**: 重复执行动画的次数,默认为 1 次. 设置为一个数值, 或者设置为无穷次重复 **infinite**

> **alternate**: 设置该属性值, 表示开启动画的反向执行. 当添加该属性值后, **反向执行动画也算作执行一次动画效果**, 因此, 需要更改 **number** 属性值为**偶数**或者**infinite**

animation-name: 设置元素应用的关键帧名称
animation-duration: 设置动画的持续时间
animation-timing-function: 设置动画的模式
animation-delay: 设置延迟执行动画的时间
animation-iteration-count: 设置动画的重复次数
animation-direction: 设置动画是否反向执行, 使用 **alternate** 表示反向执行动画
animation-play-state: 设置动画播放状态, **paused** 表示当鼠标悬停时停止播放, **running** 表示持续播放

> **transition** 与 **animation** 的区别?
> 同: 都是随着时间改变元素的属性值
> 异: 前者需要使用鼠标事件才能触发相应动画; 后者可以自动进行触发动画

3D效果.

transform-style: 设置元素的所在空间. 设置为 **flat** 时表示 2D 平面空间(默认值), 设置为 **preserve-3d** 时表示 3D 空间. tips: 该属性需要应用在父元素中
perspective: 设置 3D 景深效果, 一般地, 设置为 900px-1200px, tips: 该属性需要应用在父元素中
transform: 设置元素的 3D 效果, 可以使用以下属性值:

> **translate3d(x,y,z)**: 设置元素在水平方向、垂直方向与空间方向的平移距离
> **translateX(number)**: 设置元素在水平方向上的平移距离
> **translateY(number)**: 设置元素在垂直方向上的平移距离
> **translateZ(number)**: 设置元素在空间方向上的平移距离
>
> **rotate3d(a,b,c,deg)**: 设置元素旋转中心, 以向量形式表示旋转轴. a,b,c 取值为 (0,1)
> **rotateX(deg)**: 设置元素以X轴为旋转轴旋转的角度
> **rotateY(deg)**: 设置元素以Y轴为旋转轴旋转的角度
> **rotateZ(deg)**: 设置元素以Z轴为旋转轴旋转的角度
>
> **scale3d(x,y,z)**: 设置元素分别在 x 轴、y 轴、z 轴上的缩放效果, 单独使用该属性值没有效果, 需要配合其他效果使用
> **scaleZ(num)**: 设置元素在Z轴上的缩放效果, 单独使用该属性值没有效果, 需要配合其他效果使用

3.17Gird网格布局

网格布局: 网格布局, 将父元素划分成若干格子, 使用格子来进行组合布局. 在**父元素**中添加 **display:grid(块状网格)** 或者 **display:inline-grid(行内块)** 触发网格布局. 使用网格布局, **需要为区域划分行和列才能布局, 才能显示网格并且组合**

grid 布局与 flex 布局的异同点.

> 同: 都有父元素容器与子元素项目之分
> 异: flex 布局为轴线性布局, grid 布局为二维布局

grid 布局与表格布局的异同点.

> 同: 都是二维布局, 都存在行与列之分
> 异: 嵌套层级不同

grid 布局基础概念.

> 要产生一个 m 行 n 列的网格布局, 需要使用 **m+1** 个横向线条与 **n+1** 个纵向线条
> **grid 容器**: 触发 grid 布局的父元素
> **grid 内容**: 显示项目的区域
> **grid 项目**: 每个 grid 格子中放置的元素
> **网格线**: grid 布局中横向与纵向的线条
> **间距**: 网格与网格之间的距离
> **单元格**: 网格线交汇的格子

容器属性.

> **display : grid**: 触发块状网格布局
> **display : inline-grid**: 触发行内块网格布局
>
> **grid-template-rows**: 划分行数, 该属性后面**可接多个值**, 用**空格分隔**.
>
> > 当属性值为带有单位的**纯数值时, 按照该值划分每一行的行高; 若某一行的取值为 auto, 则表示该行占剩余高度所有**, 当属性值为**百分比**时, 为**相对于容器的高度**所占的高度; 当属性值为带有 **fr** 的值时, 相当于 **flex 属性**的用法, 按比例划分行高; 当使用带有 **minmax(min,max)** 函数的数值时, 当剩余高度足够容纳最大值时, 以最大值表示行高, 否则, 使用最小值以最大值中间的数值作为行高, 如果最小值也无法容纳, 则该行会被挤出容器
>
> **grid-template-columns**: 划分列数, 该属性后面**可接多个值**, 用**空格分隔**.
>
> > 当属性值为带有单位的**纯数值时, 按照该值划分每一列的列宽; 若某一列的取值为 auto, 则表示该列占剩余宽度所有**; 当属性值为**百分比**时, 为**相对于容器的宽度**所占的宽度; 当属性值为带有

> **fr** 的值时, 相当于 **flex 属性**的用法, 按比例划分列宽; 当使用带有 **minmax(min,max)** 函数的数值时, 当剩余宽度足够容纳最大值时, 以最大值表示列宽, 否则, 使用最小值以最大值中间的数值作为列宽, 如果最小值也无法容纳, 则该列会被挤出容器

当使用上述两个方法划分行列时, 当需要多次划分相同尺寸的行列时, 可以使用 **repeat(time,size)** 函数进行划分. 其中 **第一个参数** 表示重复划分的次数, 即几行或者几列; 可以使用 **auto-fill** 进行自动划分行数或者列数, 当宽度或者高度不再够的时候停止划分; **第二个参数** 表示行列的尺寸

grid-row-gap: 设置网格行间距
grid-column-gap: 设置网格列间距
grid-gap : rowGap colGap: 复合属性, 同时设置行间距与列间距

grid-auto-flow: 设置容器中项目的排列方式. 取值为 **row** 时横向排列(默认值), 取值为 **column** 时纵向排列

justify-items: 设置网格项目**水平方向**的对齐方式. 可以取值为 **start、center、end**
align-items: 设置网格项目**垂直方向**的对齐方式. 可以取值为 **start、center、end**
place-items : align justify: 复合属性, 分别设置网格项目的垂直对齐方式与水平对齐方式

justify-content: 设置网格在容器内的水平对齐方式, 可以从 **start、center、end、space-around、space-between、space-evenly、strench** 中取值
align-content: 设置网格在容器内的垂直对齐方式, 可以从 **start、center、end、space-around、space-between、space-evenly、strench** 中取值
place-content : align justify: 复合属性, 设置网格在容器内的垂直对齐方式与水平对齐方式

项目属性
容器中的元素会自动按照网格布局的方式进行显示, 这些元素被称作**网格项目**

grid-row-start: 设置合并单元格的起始行
grid-row-end: 设置合并单元格的结束行
grid-column-start: 设置合并单元格的起始列
grid-column-end: 设置合并单元格的结束列
grid-row : start/end: 复合属性, 设置合并单元格的起始行与结束行
grid-column : start/end: 复合属性, 设置合并单元格的起始列与结束列