画页面，先设计结构，再画样式。结构永远比css和js重要

技术栈：html + css

**一、html**

结构

结构(html) 样式(css)相分离

**二、css**

层叠样式表

作用：设置版面的布局和外观显示样式

1.css三种引入方式

行内样式 标签的style 属性设置

内部样式 页面中style 标签中设置

外部样式 通过link 标签引入

**注释：**实现结构(html)与样式(css)相分离

外部样式实现多个页面样式共享

2.css 样式规则

选择器{属性:值} 属性与值键值对

3.css 选择器

作用：用于选择标签，找到特性的html 页面元素，

**注释：**css 主要干两件事，找到元素，设置样式

根据需求不同，重点分为两大组，即基础选择器、复合选择器

适配所有的某一类元素

4.基础选择器

标签选择器 span{}

类选择器 .header{}

id 选择器 #name{}

通配符选择器 \*{} 会匹配页面所有元素，降低页面响应速度

**注释：**类选择器特殊用法，多类名 <span class=”blue font”></span> font 中放通用样式

5.font字体 (css字体样式属性)

字号、字体、unicode 字体、字体粗细、字体样式

**注释：**不同浏览器默认大小不同。谷歌浏览器默认显示字号16px，一般默认给body 指定一个固定大小，统一在不同浏览器显示的字号大小

字体设置多个字体，浏览器不支持第一个字体，适配第二个..都没有则以电脑默认字体为准，尽量使用浏览器默认

微软雅黑 Unicode 字体 “\9ED1\4F53”，有的浏览器不支持中文，为适配多数浏览器

字体 700 等于bold 400 等于normal

字体风格 正常、倾斜的(em 标签实现，或设置字体风格设为normal)

**简写：**font:italic 700 20px “微软黑体”。 顺序不能颠倒，字号和字体不能省略

6.css 文本外观样式 (css字体样式属性)

颜色、文本水平对齐方式、行间距、首行缩进、文本的装饰

**注释：**颜色写法

预定义 red

十六进制颜色 #FF0000 红、绿、蓝

RGB代码 rgb(255,0,0) 或 rgb(100%, 0%, 0%)

水平对齐：左中右

行高一般用来处理文本垂直方向居中对齐

首行缩进：替换&nbsp;&nbsp; text-indent:2em; em是倍数关系

文本的装饰：text-decoration:none; 取消下划线…

7.复合选择器

目的：为了可以选择更精细的目标元素标签

组成：由两个或多个基础选择器，通过不同的方式组合而成

\*后代选择器 .nav ul li{} 又叫包含选择器

\*子元素选择器 .nav>ul{}

\*交集选择器 p.red{} 即是p标签，又有类名为red 的元素，中间没有空格

\*并集选择器 p , span{} 中间是逗号，是和的意思, 通常用于集体声明

\*链接伪类选择器 :link{}

a:link /\*未访问的链接\*/

a:visited /\*已访问的链接\*/

a:hover /\*鼠标移动到链接上\*/

a:active /\*选定的链接，按下鼠标还没离开\*/

**注释：**存在多个时，按照从上到下的顺序写，否则有可能出现失效的情况

因为a链接浏览器有默认样式，所以工作中需要给链接单独指定样式

如 .nav{color:red;} 包含的a标签没有效果，需要单独设置 .nav a{color:red;}

8.标签显示模式

分类：块级元素、行内元素、行内块元素

\*块元素

容器类盒子

常见：div、h1~h6、p、ul、ol、li等

特点：独占一行、高宽内外边距可设置、宽度默认容器100%、里面可放块元素或行内元素

**注意**：文字类标签，如p、h1、dt标签内不能放块级元素

\*行内元素

常见：a、strong、b、em、i、del、s、ins、u、span

特点：一行可显示多个、高宽设置无效、宽度默认为内容本身宽度、里面只能放同类的和文字

**注意：**链接里面不能再放链接，特殊情况a里面可以放块级元素，但是给a转换一下块级模式最安全

\*行内块元素

常见：img、input、td

特点：一行可显示多个、宽高内外边距可设置、宽高默认为本身内容宽高

**注意：**即有行内元素特点，一行可以显示多个，又有块级元素特点，可以设置宽度和高度

\*标签显示模式转换

块转行内：display:inline;

行内转块：display:block;

块、行内转行内块元素：display:linine-bloc;

9.行高

作用：行高等于高度，让单行文本垂直居中

10.css背景

背景色 background-color:transparent; 默认透明

背景图片 background-image:url(); 小细节：必须加url()；地址提倡不加引号“”

**注释：**背景图片和插入图片差别

背景图片：背景图片是在盒子的底下，不是往里面加东西

插入图片：在盒子内部放图片

背景图片在背景颜色上面

背景图片是否平铺 background-repeat: 默认平铺repeat，不平铺no-repeat，横向平铺repeat-x，纵向平铺repeat-y

背景图片位置 background-position 写法：x坐标 y坐标；方位名词 右上角 right top；顺序随意

**注意：**方位名词只指定一个词，另一个默认为居中对齐

如果有精确数值定位，第一个必定是x方向，第二个必定为y方向

如果只写一个，第一个一定是x方向

超大图片为什么做那么大：小屏幕分辨率看不全，能照顾的电脑分辨率越多，小的看中间主要的内容

**注释：**background-position:center top; // 超大背景图片的做法，背景定位

背景附着： background-attachment; 解释背景是固定的还是滚动的，默认是滚动的

**简写：**background:颜色 url 平铺 滚动 位置； 不像font字体，简写顺序没有固定写法，这是开发人员较喜欢的写法

11.背景透明

background:rgba(0,0,0, 0.3) 最后一个参数是alpha 透明度，取值范围0~1，

**注释：**背景半透明是指盒子背景半透明，盒子里面的内容不受影响

因为是css3，所以低于ie9的版本不支持

12.css 三大特性

分类：层叠性、继承性、优先级

\*层叠性：css 样式的叠加，样式设置冲突(重复)时，只会使用一个，其它的会被层叠掉。样式不冲突，属性不会被层叠

\*继承性：子元素会继承父元素的样式

**注释：**继承可简化代码，降低css 样式复杂性，父元素设置一些样式，子元素会继承拥有

并不是所有的样式子元素都能够继承

子元素可以继承父元素的样式text-， font-，line- 这些元素开头的可以继承，以及color属性

\*优先级：

**注释：**选择器相同，执行层叠性就近原则。选择器不同根据优先级决定显示什么样式。

先看该标签是否被选中，再比较权重

**权重计算公式**：

**标签选择器**  **计算权重公式**

继承或者\* 0,0,0,0

标签选择器 0,0,0,1

类选择器 0,0,1,0

id选择器 0,1,0,0

行内样式表 1,0,0,0

!important 无穷大

**权重叠加：**.nav ul li 0,0,1,0 + 0,0,0,1 + 0,0,0,1 = 0,0,1,2

13.css 盒子模型 Box Model

网页布局的本质：利用css设置好盒子的大小，然后摆放盒子的位置，最后把网页元素如文字、图片等放入盒子里面

盒子模型：把html 页面中的布局元素看做一个矩形盒子，也就是一个盛装内容的容器。

盒子模型组成：内容、内边距、边框、外边距

边框组成：宽度 样式 颜色

border:none;

border-bottom:1px solid pink;

**注释：**合并单元格时，重叠问题 border-collapse:collapse; 合并相邻的单元格

内边距：添加内边框，盒子会变大(撑开盒子) 盒子宽度= 内容 + 边框 + 内边距； 没有外边距

新浪导航栏字数不一样多，要中间等宽，可以用padding

**注释：**特殊情况，如果盒子，没有宽度，则padding不会撑开盒子

margin 控制盒子与盒子之间的距离，不控制盒子大小

让块级盒子居中对齐：必须有宽度，左右外边距设置为auto，简写为margin:auto; 上下不用管

**扩展：**文字水平居中与盒子元素水平居中

text-align:center; 文字 行内元素 行内块元素水平居中

margin:auto; 块级盒子水平居中

**扩展：**插入图片与背景图片

插入图片：产品展示类，移动位置只能靠盒模型 padding margin

背景图片：一般用户小图标背景，只能通过background-position 移动位置

**扩展：**清除元素默认内外边距：\*{margin:0;padding:0;}

行内元素：尽量只设置左右内外边距，不用设置上下内外边距，因为没有效果

**垂直外边距的合并现象**：上下两个盒子设置外边距，中间只会取两个盒子间外边距的较大值，

左右没有这个问题，只要设置一个就行了。

**嵌套关系垂直外边距合并(塌陷)：**

方案一：为父元素定义上边框 transparent 透明 border-top:1px solid transparent;

方案二：可以为父级指定一个上padding padding-top:1px;

方案三：可以为父元素添加overflow:hidden;

取消li 列表样式：list-style:none;

14.圆角边框 与 盒子阴影

css3 属性，不会影响页面布局

border-radius:5px; 或50%；

box-shadow:水平阴影 垂直阴影 模糊距离(虚实) 阴影尺寸(影子大小) 阴影颜色 内/外阴影；

box-shadow:2px 2px 2px 2px rgba(0, 0, 0, .3)

15.浮动(float)

网页布局的核心—用css来摆放盒子

css 提供三种机制来设置盒子的摆放位置：普通流(标准流)、浮动、定位

普通流：

块级元素独占一行，从上向下顺序排列；常用div hr p h1~h6 ul ol dl form table

行内元素从左向右顺序排列，碰到父元素边缘则自动换行；常用span a i em等

浮动：让盒子从普通流中浮起来，主要作用让多个块级盒子一行显示

定位：将盒子定在浏览器的某一个位置 – css 离不开定位，特别是后面的js特效

为什么用浮动？

答：将三个块级元素同行显示，转成行内元素，中间会有空白缝隙

不能实现盒子左右对齐

因为一些网页布局要求，标准流不能满足我们的需求，因此需要浮动来完成网页布局

浮动特点：

设置浮动的元素会脱离标准流的控制

设置浮动的元素可以移动到指定的位置

浮动会改变元素display属性

任何元素都可以浮动，浮动元素会生成一个块级框，不论它本身是何种元素，生成的块级框和我们前面的行内框及其相似

浮动的元素之间没有空隙

如果父级装不下几个浮动元素，会多出的盒子会另起一行

浮动和标准流的父盒子搭配，最大化减少对其他标准流的影响

一个完整的网页，是**标准流** + **浮动** + 我们后面要讲的定位一起完成的

**注意：**导航栏列表，最好用li包含a标签，如果直接用a，搜索引擎容易辨别为有堆砌关键字嫌疑

故意堆砌关键字容易被搜索引擎有降权的风险，从而影响网站排名

更有条理性

**浮动与父盒子的关系**：子盒子的浮动参照父盒子对齐，可以在盒子内左右移动

不会与父盒子的边框重叠，也不会超出父盒子的内边距

**浮动与兄弟盒子的关系**：第一个不浮动，第二个盒子浮动，还是按照正常显示，因为第一个是标准流

浮动只会影响后面的盒子，不会影响前面的盒子元素

清除浮动描述：由于浮动元素不再占用源文档流的位置，所以它会对后面的元素排版产生影响

准确地说，并不是清除浮动，而是要清除浮动后造成的影响

1.父级没高度，2.子盒子浮动了，3.影响下面布局了，我们就需要清除浮动了。

**清除浮动：**清除浮动主要为了解决父级元素因为子级浮动引起内部高度为0的问题。清除浮动后，

父级就会根据浮动的子元素自动检测高度。父级有了高度，就不会影响下面的标准流了。

**清除方法：**选择器{clear:属性值} left / right/both;

方法一：额外标签法。 在浮动后面加个空标签<div style=”clear:both”>

方法二：父级元素。父级添加overflow:hidden; 就可以清除浮动

方法三：父级使用after伪元素清除浮动 :after 方式为空元素额外标签法的升级版，好处是不用单独添加标签了

.clearfix:after{content:””;display:block;height:0;clear:both;visibility:hidden;}

.clearfix{\*zoom:1;} /\*IE6、7专有\*/

有点：符合闭合浮动思想 结构语义化正确

缺点：由于IE6~7不支持：after，使用zoom:1 触发hasLayout

方法四：父级双伪元素清除浮动

.clearfix:before, .clearfix:after{content:””; display:table;}

.clearfix:after{clear:both;}

.clearfix{\*zoom:1;}

16.PS 切图工具

一般美工提供的图片都是psd格式的图片，方便我们切图

一般透明背景的选择png图片

切片工具：选中图片、导出选择切片

切图插件:Cutterman

17.css 属性书写顺序

建议遵循以下顺序

1.布局定位属性：display/position/float/clear/visibility/overflow(建议display第一个写，毕竟关系到模式)

2.自身属性：width / height / margin / padding / border / background

3.文本属性：color / font / text-decoration / text-align / vertical-align / white-space / break-word

4.其他属性(css3)：content / cursor / border-radius / box-shadow / text-shadow / background:linear-gradient …

**扩展：**子盒子margin-top ,如果加浮动，就不会有塌陷问题了。

**扩展：**父盒子的三个子盒子都要垂直居中，就利用继承性，给父盒子垂直居中的设置就可以了。Font- text- line- color

**清除浮动：**父级没有高度，子盒子浮动了，影响了下面布局了，我们就需要清除浮动

18.定位 position

将盒子定位在某一个位置 自由的漂浮在其他盒子(包括标准流和浮动)的上面

**注释：**三种布局机制的上下顺序

标准流在最底层(海底) 浮动的盒子在中间层(海面) 定位的盒子在最上层(天空)

定位 = 定位模式(position) + 边偏移(上右下左)

**注释：**定位的盒子有了边偏移才有价值。一般情况下，凡是有定位的地方必定有变偏移

**定位分类**：static静态定位、relative相对定位、absolute绝对定位、fixed固定定位

静态定位：没有定位，没有边偏移，有也没有效果

相对定位：relative。

特点一：相对于自己原来在标准流中位置来移动的；

特点二：原来在标准流的区域继续占有，后面的盒子仍然以标准流的方式对待他

绝对定位：absolute 元素以带有定位的父级元素来移动位置

特点一：完全脱标 – 完全不占位置

特点二：父元素没有定位，则以浏览器为准定位

父盒子有定位(除了静态定位外什么定位都可以)，则以祖先盒子为准 子->父->爷爷

**扩展：**子绝父相

**注释：**相对定位保留位置，绝对定位不保留位置

固定定位：fixed 固定定位是绝对定位的一种特殊形式

特点一：完全脱标 – 完全不占位置

特点二：只认浏览器的可视窗口 -- 浏览器可视窗口 + 边偏移属性 来设置元素的位置

跟父元素没有任何关系；单独使用

不随滚动条滚动

**注释：**标准流的子盒子总是以父级为准移动位置，在父元素内

**扩展：绝对定位水平居中**

答：margin:auto 无效，只能计算居中

先left :50%；再向左移动(margin-left)自己的宽度一半。 搭配使用

**扩展：绝对定位垂直居中**

**答：**先top:50%；再向上移动(margin-top)自己的高度一半。搭配设置

**扩展：堆叠顺序(z-index)**

答：使用定位布局时，可能会出现盒子重叠的情况

加了定位的盒子，默认后来者居上，后面的盒子会压住前面的盒子

应用z-index层叠等级属性可以调整盒子的堆叠顺序

整数。数值越大越在上面

z-index：只能应用于相对定位、绝对定位和固定定位的元素，其他标准流、浮动和静态定位无效。

行内块元素不给宽度，默认宽度就是内容的宽度。

**扩展：影响display 属性转换**

答：绝对定位和固定定位也和浮动类似，默认转换的特性，转换成类似行内块的元素

总结：一个行内的盒子，如果加了浮动、固定定位和绝对定位，不用转换，就可以给这个盒子直接设置宽度和高度。

**扩展：浮动解决外边距合并问题**

浮动元素、绝对定位(固定定位)元素的都不会触发外边距合并的问题(我们以前用padding border overflow 解决的)

也就是说，我们给盒子改为浮动或者定位，就不会有垂直外边距合并的问题了。

19.网页布局总结

一个完整的页面，有标准流、浮动、定位一起完成布局。每个都有自己的专门用法

1）标准流

可以让盒子上下排列 或者 左右排列的

2）浮动

可以让多个块级元素一行显示 或者左右对齐盒子 浮动的盒子就是按照顺序左右排列

3）定位

定位最大的特点是有层叠的概念，就是可以让多个盒子前后叠压来显示。但是每个盒子需要测量数值。

20.css 高级技巧

1.元素的显示与隐藏

①display 显示、visibility 可见性、overflow 溢出

display:none; 先隐藏元素，不再保留位置 ->显示 display:block; 除了转为块级元素外，同时还有显示、隐藏元素的意思

②visibility:hidden 可见性，隐藏，保留位置 ->显示 visibility:visible;

③overflow 溢出 visible hidden scroll auto

visible 默认是visible

overflow

hidden 超出盒子，隐藏。

scroll 超出显示滚动条,总是显示滚动条

auto:超出显示滚动条，不超出不显示

2.css 用户界面样式

为了提高用户体验

①鼠标样式cursor

defualt 默认箭头

pointer 小手

move 移动 十字架样式，放大镜效果

text 输入

not-allowed 禁止符号

②轮廓线 用于绘制元素周围一条线，位于边框边缘的外围，可起到突出元素的作用

写法：outline:颜色 样式 宽度;

因为在不同浏览器显示不同效果，一般设置不要。常在样式初始化时取消轮廓线outline:0 或outline:none;

③防止拖拽文本域

文本域也有轮廓线，通常 outline:none;在通过边框改变颜色 border：1px solid #036;

拖拽会影响布局，设置resize:none; 防止用户拖拽文本域

3.vertical-align 垂直居中

只针对行内元素和行内块元素有效，对块级元素无效

baseline 基线对齐

middle 居中对齐

top 顶线对齐

图片和文字默认是基线对齐

**扩展：**头像图片和文本居中对齐，给图片设置居中对齐样式

**扩展：**去除图片和底部的空隙，设置图片样式居中或顶线对齐；方法二：将图片元素设置为块级元素，不会有空隙了

4.溢出的文字省略号显示

三部曲：强制一行显示，超出部分隐藏，文字溢出，用省略号替代

①white-space:normal 默认文字显示不开时，自动换行

nowap:强制在一行显示，不换行，直到遇到br 或 文本结束才换行

②overflow:hidden

③text-overflow: clip; 简单的裁切，不显示省略标记

text-overflow:ellipsis; 当对象文本溢出时，显示省略号标记 …

5.css 精灵技术 sprite

①为什么需要

为了有效减少服务器接受和发送请求的的次数，提高页面的加载速度

出现了css精灵技术 例如一个页面的背景图片加载（多个背景图片请求多次）

②精灵技术讲解

background-image

background-repeat

background-position 属性进行背景定位

③核心总结

精准测量，每个小图标大小和位置

给盒子指定小背景图片时，背景定位基本都是负值

④制作精灵图

精灵图都是网页美工做的

精灵图上方的都是小的装饰性质的背景图片，插入图片不能往上放

可以横向摆放，也可以纵向摆放，但是每个图片之间留有适当空隙

精灵图最低端，留有一片空隙，方便我们以后添加其他精灵图

5.滑动门

a 设置背景左侧，padding撑开合适宽度

span设置背景右侧，padding 撑开合适宽度剩下由文字撑开宽度

之所以a包含span 就是因为整个导航都是可以点击的

6.扩展

①margin负值之美

②css 三角形

查看块盒子边框组成

div{

width:0;

height:0;

border-top:10px solid red;

border-top:10px solid green;

border-top:10px solid blue;

border-top:10px solid pink;

}

模拟三角效果步骤：通过边框来实现的

高度宽度为0、

4个边框都要写，只保留需要的边框颜色，其余的不能省略，都改写成transparent透明就好了

为了照顾低版本兼容性，加上font-size:0;line-height:0;

p{

width:0;

height:0;

border-style:solid;

border-width:10px;

border-color:transparent transparent transparent red;

font-size:0;

line-height:0;

}

21.扩展

版心

**1.网站的ico图标**

将图标放到根目录下，不要放到图片文件下

制作图标：先切成png图片。使用网上第三方转换网站转成.ico文件

给浏览器使用的

**2.网站优化三大标签**

SEO 搜索引擎优化，如百度，google 等搜索引擎

符合搜索引擎规则，搜索疫情优先显示

title：网站标题

description：网站说明

keywords：关键词

总结：做网站，符合搜索引擎的要求，可以让搜索引擎优先显示我们的网站。

我们需要知道，根据优化人员给我们的内容，往这三个标签里添加

能写出这三个标签

**3.字体图标**

看上去像图片，其实是文字

图片缺点：增加总文件大小，增加额外的http请求，不能很好的缩放

字体图标优点：可作出图片一眼高效果，改变透明度、旋转度等

本质是文字，可改变颜色，产生阴影、透明效果

几乎支持所有浏览器

移动端必备良药

图标库网站：阿里云图标库

IcoMoon网站

四个文件都是字体文件，为了兼容不同浏览器的

追加字体图标

生成的图标文件中有个selection.json 很重要，不能删除， import Icons 按钮，可以添加图标

**4.logo 优化**

Logo 里面首先放一个h1标签，目的是为了提权，告诉搜索引擎，这个地方很重要

H1 里面放一个连接，可以返回首页，给链接一个大小和logo的背景图片

链接里面要放文字(网站名称)，为了搜索引擎收录我们，但是文字不要显示

要用text-indent 移到盒子外面(text-indent:-9999px) 然后overflow:hidden 淘宝的做法

直接给font-size:0; 就看不到文字了，京东的做法

最后给链接一个title 这样鼠标放到logo上，就可以看到提示文字了

**5.过渡效果**

Css3 属性，IE6下无效

transition:要过渡的属性 花费时间 运动曲线 何时开始;

如果有多组属性，用逗号隔开

变色技巧：transition:background-color 1s ease 0s;

所有属性都有变化效果：transition:all 1s ease 0s;

该属性写在本身元素上

**6.获取焦点元素**

css3 属性 :focus伪类 选择器用于选取获得焦点的元素，主要针对表单元素

用于获取焦点时改变个颜色啊，字体什么的

outline:none; 什么意思？ 轮廓

**7.表单元素**

label for id 技巧

22.课程总结

1.html 我们学的就是常用标签，就是基本盒子

2.css 就是用来美化布局页面

3.html + css 是没有逻辑可言的，基本就是搭积木摆放盒子的过程，你需要的是耐心

4.现在最困难的是 布局结构，欠缺分析页面布局的能力

5.多看别人写的页面，模仿人家的布局，每次写页面总会有新的收获

6.错误总是在所难免，一定要学会利用chrome调试工具，他们能快速帮我们排查错误，你还需要耐心

7.学好定位，对后面学习JavaScript 有很大的帮助

8.口号：拼死也要克服它 pink

版心

**1.网站的ico图标**

**三、html5 + css3**

1.vs code 软件

开源、比webstorm 更轻；智能提示；自带emmet；插件安装方便；自带强大的调试方式；跨平台

本身是用node 开发的

带自动补全功能

设置主题背景风格：做下角设置按钮里

2.常用插件

Chinese 插件：

Open in browser 插件：右键打开浏览器

JS-CSS-HTML Formatter 插件：每次保存，都会自动格式化js css html 代码，自动对齐

Auto Rename Tag 插件：改动标签名，对应后面的结束标签页自动改正

快速生成html 骨架： ! + Tab 键

放大视图：ctrl 与加号

3.html 5

超文本标记语言第五次调整，具有新的元素，属性和行为。对原有删减和新增

增加的新特性：语义特性；本地存储特性；设备兼容特性，链接特性，网页多媒体特性，三维，图形及特效特性，性能与集成特性，css3 特性。这个集合有时称为html 5

4.标签语义化

以前布局，我们基本用div来做。Div对于搜索引擎来说，是没有语义的

<header> 头部标签

<nav> 导航标签

<article> 内容标签

<section> 块级标签 块级标签，

<aside> 侧边栏标签

<footer> 尾部标签

注意：这种语义化标签主要针对搜索引擎的

这些新标签页面中可以多次使用

在IE9 中，需要把这些标签转换为块级元素，（语义化标签大多作为行内元素使用）

移动端更喜欢使用这些标签，pc端 很少使用，因为大多浏览器版本较低。要考虑兼容性

5. h5 新增多媒体标签

音频：<audio>

视频：<video>

使用他们可以很方便的在页面中嵌入音频和视频，而不再去使用落后的flash 和其它浏览器插件

**1.音频标签**

支持格式：Ogg Vorbis、Mp3、Wav

<audio src=”文件地址” controls=”controls”></audio>

属性：controls 向用户显示控件，比如播放按钮

autoplay 自动播放音频 谷歌浏览器把autoplay 属性给禁用了，其它没有

loop 循环播放

src 音频路径

不同浏览器显示浏览器控件样式不同

因为不同浏览器支持不同格式，我们采取的解决方案使，为这个音频准备多个格式的文件

**兼容性写法，**从上往下支持

<audio controls=”controls”>

<source src=”happ.mp3” type=”audio/mpeg”>

<source src=”happ.mp3” type=”audio/ogg”>

您的浏览器版本比较低，不支持audio 播放

</audio>

**2.视频格式**

支持格式：Ogg、MP4、WebM

<video src=”文件地址” controls=”controls”></ video>

兼容性写法与音频一样，谷歌浏览器把自动播放功能禁用了，有解决方案：给视频添加静音属性

属性：autoplay、controls、width、height、loop、preload、src、poster、muted

**扩展：**从网上扒视频，选中视频，右键，在新标签中打开，可以下载

**3.总结**

音频标签和视频标签使用基本一致

浏览器支持情况不同

谷歌浏览器把音频和视频自动播放禁止了

我们可以给视频标签添加muted 属性可以自定播放视频，音频不可以

视频标签是重点，我们经常设置自动播放，不适用controls空间，循环和设置大小属性

6. 新增input 表单、表单属性

**1. input 表单**

type=”email” 限制用户输入必须为Email类型

type=”url”

type=”date”

type=”time”

type=”month”

type=”week”

type=”number”

type=”tel”

type=”search”

type=”color”

type=”url”

**注释：**必须放在form 域中才有效，移动端用的比较多

**2.表单属性**

required

placeholder 提示文本(占位符)

autofocus 自动聚焦属性，页面加载完成自动聚焦到指定表单

autocomplete off/on 输入过，提示输入过的内容。两点要求：有name属性；成功提交过 才会有效

multiple 可以多选文件提交，针对文件域。如上传文件 <input type=”file” multiple=”multiple”>

7. css3 新增选择器

**1. 属性选择器**

button[disabled]{} 属性选择器的使用方法，不用再创建类进行选择控制了

注意：中间没有空格，选择的即是button，又有disabled 属性的元素

属性选择器的权重高于标签选择器，为10。为10 的还有类选择器，伪类选择器，权重等

E[att] 选择具有att属性的E元素

E[att=”val”] 选择具有att属性且属性值等于val的E元素

E[att^=”val”] 匹配具有att属性、且值以val开头的E元素

E[att$=”val”] 匹配具有att属性、且值以val结尾的E元素

E[att\*=”val”] 匹配具有att属性、且值中含有val的E元素

**2. 结构伪类选择器**

E:first-chid 匹配父元素中的第一个子元素E

E:last-chid

E:nth-chid(n) 匹配父元素中的第n个字元素E

E:first-of-type 指定类型E的第一个

E:last-of-type

E:nth-of-type(n)

注意：结构伪类选择器权重为10

n 可以是数字，关键字和公式

n 可以是关键词 even 是偶数，odd 是奇数

第0个元素或者超出了元素的个数会被忽略

n+5 从第5个开始(包含第五个)到最后

-n+5 前5个(包含第5个)

nth-child(n) 选择父元素里面的第n个孩子，它不管里面的孩子是否同一类型

父元素下是否为同一类型元素决定使用哪个 nth-chid(n) nth-of-type(n)

注意：选择父元素下某一类型第几个元素 div span:first-of-type{} 对应 nth-of-type(n)

**3. 伪元素选择器**

::before 在元素内部的前面插入内容

::after 在元素内部的后面插入内容

注意：before 和 after 必须有content 属性

before 是在元素里前面放内容，after是在元素里后面放内容

before 和 after 创建一个元素，但是属于行内元素

在dom里面看不见创建的元素，所以称为伪元素

伪元素和标签选择器一样，权重为1

扩展：使用图标，常用

8. 转换

转换是css3 中具有颠覆性的特征之一，可以实现元素的位移、旋转、缩放等效果。可以简单理解为变形

**1. 2D转换**

二维坐标系

**移动 translate**

移动盒子的位置：定位、盒子的外边距、2d转换移动

语法：transform:translate(x,y) 在x，y轴上移动位置，中间用逗号隔开

只移动x轴方向：translateX(100px)

只移动y轴方向：translateY(100px)

translateX(50%) 移动距离是盒子自身的宽度50%

translateY(50%) 同理

优点：不会影响其它盒子的位置

常用：鼠标放到图片上，图片上移特效

注意：对行内元素没有效果

盒子垂直居中，可以用定位加translate 做

**旋转 rotate**

让元素在2维平面顺时针旋转或逆时针旋转

语法：transform:rotate(度数)

注意：单位为度数，单位是deg，如rotate(45deg)

度数为正，顺时针，反之…

默认旋转中心是元素的中心点

常用：三角形，用标签做。 div:hover::after{}

**旋转中心点 transform-origin**

设置元素转换的中心点

语法：transform-origin: x y

注意：后面参数是用空格隔开的

x y 默认转换的中心点是元素的中心点 (50% 50%)

还可以给x y 设置像素 或者 方位名词(top botton left right center)

**缩放 scale**

只要给元素添加上这个属性，就能控制它放大还是缩小

语法：transform:scale(x,y)

注意：参数，数字不跟单位，倍数的意思 (1.1,1.1)放大为原来的1.1倍，小于1 代表缩放

scale(2) 同时修改宽度和高度，简单写法

优点：不会影响其它盒子布局，可以设置缩放中心点

**2D 转换综合写法**

语法：transform: translate() rotate() scale() 等…

注意：其顺序会影响转换的效果（先旋转会改变坐标轴方向）

当我们同时有位移和其它属性的时候，记得要将位移放到前面

9. css 动画

动画是CSS3中具有颠覆性的特征之一，可通过设置多个节点来精确控制一个或一组动画，常用来实现复杂的动画效果

相比较过渡，动画可以实现更多变化，更多控制，连续自动播放等效果。

基本使用：1.先定义动画 2.再使用(调用)动画 3.设置持续时间

**定义动画**：用keyframes 定义动画(类似定义类选择器)

@keyframes 动画名称{ // 关键帧

0%{ <==> 等价于 from

//width:100px;

transform:translate(0,0)

}

100%{ <==> 等价于 to

//width:200px;

transform:translate(1000px,0);

}

}

注释：0%是动画的开始，100%是动画的完成。这样的规则就是动画序列

在@keyframes 中规定某项css样式，就能创建由当前样式逐渐改为新样式的动画效果

动画是使元素从一种样式逐渐变化为另一种样式的效果，您可以改变任意多的样式任意多的次数

用百分比来规定变化发生的时间，或用关键词 from 和 to ，等同于 0% 和 100%

**使用动画：** 通过设置样式 animation-name:动画名称，来使用动画

**持续时间：** animation-duration:持续时间；

总结：动画序列， 可以做多个状态的变化 keyframe 关键帧

里面的百分比要是整数

里面的百分比就是 总的时间的划分

**动画常用属性：**

@keyframes 规定动画

animation-name 使用哪个动画

animation-duration 动画一个周期花费的时间，默认为0秒

animation-time-function 动画的运动曲线，默认为ease;

animation-delay 动画延迟几秒才开始。

animation-iteration-count 播放次数，默认是1，还有infinite 无限循环

animation-direction 是否反方向播放，默认走到头，从起头再播放。 默认normal，还有alternate逆方向

animation-fill-mode 动画结束后状态，保持forwards，默认回到起始backwards。前提不是无线循环

animation-play-state 鼠标放上去，是否运行或暂停，默认是running，还有pause。

div:hover{ animation-play-state:pause;} // 鼠标放上去，暂停动画

**简写：**

animation: 动画名称 持续时间 运动曲线 何时开始 播放次数 是否反方向 动画起始或结束的状态；

注意：前两个属性一定要写

简写里不包含animation-play-state 常与鼠标经过等其他配合使用

一个元素可以添加多个动画，用逗号隔开

**案例：**热力图 奔跑的熊

10. 3D 转换

特点：近大远小；物体后遮挡不可见

坐标值：x 右边是正值

y下面是正值

z 往外是正值

**3D位移** translate3d

transform:tanslateX(100px)

transform:tanslateY(100px)

transform:tanslateZ(100px)

简写： transform: translateX(100px) tanslateY(100px); 等

transform3d(100px, 100px, 100px)

注意：tanslateZ 沿着Z轴移动

tanslateZ 单位一般为px

tanslateZ

简写xyz是不能省略的，如果没有就写0

**3D透视** perspective

想要在网页上产生3D效果，需要透视

模拟人的视觉位置，可认为一直眼睛看物体

也称为视距，视距就是人的眼睛到屏幕的距离

距离视觉点越近的在电脑平面成像越大，越远成像越小

透视的单位是像素

透视写在被观察元素的父元素上

D:就是视距，视距就是一个距离人的眼睛到屏幕的距离

Z:就是z轴，物体距离屏幕的距离，z轴越大，可以理解距离眼睛越近，成像越大

注意：透视写在被观察元素的父元素上

透视越小，物体越大

透视的单位是像素

**3D旋转** rotate3d

transform:rotateX(180deg)

transform:rotateY(180deg)

transform:rotateZ(180deg)

旋转方向：左手法则

tansform:rotate3d(x,y,z,deg) 沿着自定义轴旋转 deg为角度

**3D呈现** transform-style

控制子元素是否开启三维立体环境

transform-style:flat;子元素不开启3d立体空间 默认的

transform-style:preserve-3d;子元素开启立体空间

代码写给父级，但是影响的是子盒子

扩展：学习3d 离不开这透视和这个属性

如果有移动或者其他样式，必须先写我们的移动

11. 浏览器私有前缀

浏览器私有前缀是为了兼容老版本的写法，比较新版本的浏览器无须添加

-moz-： 代表firefox 浏览器私有属性

-ms- ： 代表ie浏览器

-webkit- 代表safari、chrome私有属性

-o- 代表Opera 私有属性

注释：提倡先写私有前缀，再写基本的

**四、移动web 开发之流式布局（第一种）**

1.浏览器现状

Pc端：360、谷歌、火狐、qq、百度、搜狗、IE浏览器等

移动端：UC、QQ、欧朋、百度手机、360安全、谷歌、搜狗手机、猎豹以及其他杂牌浏览器

国内大多数手机浏览器都是根据Webkit修改过来的，国内尚无自主研发的内核

**总结**：兼容移动端主流浏览器，处理Webkit内核浏览器即可。

2.移动端调试方法

Chrome DevTools（谷歌浏览器）的模拟手机调试

搭建本地web服务器，手机和服务器是一个局域网内，通过手机访问服务器

使用外网服务器，直接IP或域名访问

3.视口

视口(viewport)就是浏览器显示页面内容的屏幕区域。视口可以分为布局视口、视觉视口和理想视口

常用：理想视口

**布局视口：** layout viewport

一般移动设备的浏览器都默认设置了一个布局视口，用于解决早期pc端页面在手机上显示的问题

Ios，Android基本上都将这个视口设置为980px，所以pc端网页大多都能在手机上显示，只不过元素看上去很小，

一般默认可以通过手动缩放网页。缺点是繁琐，不适合移动端开发

**视觉视口** visual viewport

用户正在看到的网站的区域。注意：是网站的区域

可以通过缩放去操作视觉视口，但不会影响布局视口，布局视口仍保持原来的宽度

**理想视口** ideal viewport

为了使网站在移动端有最理想的浏览和阅读宽度而设定

理想视口，对设备来说，是最理想的视口尺寸

需要手动添加meta视口标签通知浏览器操作

meta视口标签主要目的：布局视口的宽度应该与理想视口的宽度一致，简单理解：设备有多宽，布局视口就多宽

4.meta 视口标签

<meta name=”viewport” content=”width=device-width,user-scalable=no,initial-scale=1.0,maximum-scale=1.0,minimum-scale=1.0”>

width 宽度设置的是viewport 宽度，可以设置device-width特殊值(设备宽度)

initial-scale 初始缩放比，大于0的数字

maximum-scale 最大缩放比，大于0的数字

minimum-scale 最小缩放比，大于0的数字

user-scalable 用户是否可缩放，yes或no(1或0)

注释：不加，一段文字，在手机端很小，看不清了

5.二倍图

**物理像素&物理像素比**

物理像素点指的是屏幕显示的最小颗粒，是物理真实存在的。这是厂商在出厂时就设置好了。如苹果6\7\8是750\*1334

我们开发时候的1px不是一定等于1个物理像素的

pc端页面，1个px等于1个物理像素的，但是移动端就不尽相同

一个px的能显示的物理像素点，称为物理像素比或屏幕像素比

总结：物理像素就是我们说的分辨率

Retina(视网膜屏幕)是一种显示技术，可以将更多的物理像素点压缩至一块屏幕，以达到更高的分辨率，并提高屏幕显示的细腻度。

**多倍图：**图片在手机Retina屏中打开，按照原有物理像素比会方大数倍，造成图片模糊

**背景缩放**

属性规定背景图像的尺寸

语法：background-size：背景图片宽度 背景图片高度;

只写一个参数，如设置宽度，高度不写 会等比例缩放

单位可以写%，相对于父盒子来说的

cover 要完全覆盖div 盒子，可能有部分背景图片显示不全

contain 高度和高度等比例拉伸，当宽度和高度 铺满div 盒子就不再进行拉伸了，可能有部分空白区域

**多背图切图** cutterman

@3X 3倍图

@2X 2倍图

@1X 1倍图

6.移动端开发选择方案

**单独制作移动端页面(主流)**

通常情况下，网址域名前面加上m(mobile)可以打开移动端，通过判断设备，如果移动设备打开，则跳到移动端页面

**响应式页面兼容移动端(其次)**

通过判断截屏宽度开改变样式，以适应不同终端

缺点：制作麻烦，需要花很大精力去调兼容性问题

7.移动端技术解决方案

**移动端浏览器**

移动端浏览器基本以webkit 内核为主，因此我们就考虑webkit兼容性问题，可以放心使用h5和css3样式

浏览器的私有前缀我们只需要考虑添加webkit即可

**css 初始化 normalize.css**

移动端css初始化推荐使用 normalize.css

它保护了有价值的默认值

修复了浏览器bug

是模块化的

有详细的文档

**css3 盒子模型 box-sizing**

传统模式宽度计算：盒子宽度 = css 中设置的width + border +padding

css3 盒子模型： 盒子宽度 = css 中设置的宽度width 里面包含了border 和padding

也就是说，我们css3 中盒子模型，padding 和border 不会撑大盒子了

传统盒子： box-sizing:content-box; 从内容区计算

css3 盒子模型：box-sizing:border-box; 从边框开始计算

注意：css3 盒子模型有兼容性问题，在ie6/7/8 不兼容，在移动端可以放心使用

**特殊样式**

点击高亮我们需要清除 设置为tansparent 完全透明

-webkit-tap-hightlight-color:transparent;

在移动端浏览器默认的外观在ios上加上这个属性才能给按钮和输入框自定义样式

-webkit-appearance:none;

禁止长按页面时的弹出菜单

Img,a{-webkit-touch-callout:none;}

8.移动端常见布局 – 技术选型

**单独制作移动端页面(主流)**

流式布局（百分比布局）

flex 弹性布局（强烈推荐）

less + rem + 媒体查询布局

混合布局

**响应式页面兼容移动端(其次)**

媒体查询

bootstrap

**流式布局 (百分比布局)**

就是百分比布局，也称非固定像素布局

通过盒子的宽度设置成百分比来根据屏幕的宽度来进行伸缩，不受固定像素的限制，内容向两侧填充

流式布局方式是移动web开发使用的比较常见的布局方式

为了保护盒子内容在合理范围之内

max-width:

min-width:

案例开发步骤：

方案：我们采取单独制作移动页面方案

技术：布局采取流式布局 特点，只管宽度，不管高度

结构：搭建相关文件夹结构 img、upload、css、页面 等文件夹

视口、样式：设置视口标签以及引入初始化样式，最大缩放样式和最小缩放样式为1.0，不允许缩放，引入初始化样式等

可以给body 百分百宽度，最小宽度320px;最大宽度640px;margin:0 auto;居中

图片没有垂直居中：verticel-align:

**二倍精灵图做法**

在firework里面吧精灵图等比例缩放为原来的一半

之后根据大小测量坐标

注意代码里面background-size 也要写：精灵图原来宽度的一半

京东的dpg 类型图片

webp 类型图片格式 谷歌开发的图片格式

**五、移动web开发之flex弹性布局（第二种）**

**Flex的布局原理：**用来为盒状模型提供最大的灵活性，任何容器都可以指定为flex布局

采用flex布局的元素，称为flex容器，简称容器。它所有子元素自动成为容器成员，称为flex 项目，简称项目。

**总结：**通过给父盒子添加flex 属性，来控制子盒子的位子和排列方式。

传统布局：兼容性好；布局繁琐；局限性，不能在移动端很好的布局

flex布局：操作方便，布局极为简单，移动端应用广泛；pc端浏览器支持情况较差；IE11或更低版本，不支持或仅支持部分

注意：父元素为flex布局时，子元素的float、clear 和 vertical-align属性将失效，项目有宽度和高度

伸缩布局 = 弹性布局 = 伸缩盒布局 = 弹性盒布局 = flex 布局

**容器属性**：

flex-direction 设置主轴的方向

justify-content 设置主轴上的子元素排列方式

flex-wrap 子元素是否换行

align-content 侧轴上的子元素的排列方式(多行)

align-items 侧轴上的子元素排列方式(单行)

flex-flow 复合属性，相当于同时设置了flex-direction 和 flex-wrap

**主轴和侧轴**；项目是跟着主轴排列的。

flex-direction 设置项目的(主轴)排列方向，

默认 row 从左到右；

row-reverse 从右到做；

column 从上到下

column-reverse

justify-content 设置项目在主轴上的对齐方式，使用这个属性前一定要设置好主轴方向

flex-start 默认值，在主轴方向头部对齐

left-end 尾部对齐

center 水平居中

space-around 平分剩余空间

space-between 先两边贴边，再平分剩余空间(重要)

注释：当子元素装不下时，自动缩小了子元素的大小

flex-wrap 设置子元素是否换行

默认情况下，项目都排在一条线上(主轴)。flex-wrap 属性定义，flex布局中默认是不换行的，如果装不开，会缩小子元素的宽度，放到父元素里面。

nowrap 默认不换行

wrap 换行

align-items 设置侧轴上项目的排列方式(单行)

flex-start 默认、flex-end 、center 居中

stretch 拉伸 但是子盒子不要设置高度。

align-content 设置侧轴上的项目的排列方式（多行）

设置项目在侧轴上的排列方式，并且只能用于子项出现换行的情况(多行)，在单行下是没有效果的

flex-start 、flex-end、center

space-around 平分剩余空间

space-between 先两边贴边，再平分剩余空间(重要)

flex-flow 是 flex-direction 和 flex-wrap 属性的复合属性

flex-flow:column wrap; 设置主轴方向，是否换行。

**项目属性**：

flex 子项占的份数

align-self 控制子项自己在侧轴的排列方式

order 定义子项的排列顺序(前后顺序)

flex 定义子项目分配剩余空间，用flex来表示占多少份 flex 1，占一份

圣杯布局，左右两侧宽度固定，中间占剩余所有空间

align-self 子项自己在侧轴上的排列方式

该属性允许单个项目有与其他项目不一样的对齐方式，可覆盖align-items属性，默认auto,表示继承父元素的align-items 属性，如果没有父元素，则等同于strech

order 属性定义项目的排列顺序

数值越小，排列越靠前，默认为0,

注意：和z-index 不一样

**扩展：**固定的盒子要有宽度，固定定位与父级没有关系，它以屏幕为准

背景线性渐变：background:linear-gradient(起始方向，颜色1，颜色2，…);

背景渐变必须添加浏览器私有前缀

**六、移动web 开发之rem 适配方案（第三种）**

**1.rem 基础**

rem(root em) 是一个相对单位，类似于em，em是父元素字体大小

不同的是rem的基准是相对于html元素的字体大小

比如，根元素(html) 设置font-size=12px;非根元素设置width:2rem; 则换成px表示就是24px;

**rem 优势：**一个页面中只有一个html元素。可以通过修改html里面的字体大小来改变页面中元素的大小。可以整体控制。

rem 适配方案开发，有两种方式，需要了解。

**2.媒体查询 Media Query**

是css3 新语法

使用@media 查询，可以针对不同的媒体类型定义不同的样式

@media 可以针对不同的屏幕尺寸设置不同的样式

当重置浏览器大小的过程中，页面也会根据浏览器的宽度和高度重新渲染页面

**语法：**@media 媒体类型 and|not|only(媒体特性){}

**主要价值：**可以根据不同屏幕尺寸在改变不同的样式

**媒体类型：**不同的终端设备划分成不同的类型，称为媒体类型。

all(所有设备)

print(打印机和打印预览)

screen(电脑屏幕/平板电脑/智能手机) 等

**关键字**：将媒体类型和多个媒体特性连接到一起作为媒体查询的条件

and：将多个媒体特性连接到一起，相当于“且”的意思

not：排除某个媒体类型，相当于“非”

only：指定某个特性的媒体类型

**媒体特性：**每种媒体类型都具体各自不同的特性，根据不同媒体特性设置不同的展示风格。注意他们要加小括号包含

width: 定义输出设备中页面可见区域的宽度

min-width: 最小宽度

max-width: 最大宽度

值是包含的。

注意：为了防止混乱，媒体查询我们要按照从小到大或者从大到小的顺序来写，但是我们最喜欢的还是从小到大来写，这样代码更简洁。样式重叠性，会层叠掉重复的。

注意：screen 和and 是不能重复的，数字的单位不能省略

**3.媒体查询 + rem 实现元素动态大小变化**

rem单位是跟着html来走的，有了rem页面元素可以设置不同大小尺寸

媒体查询可以根据不同设备宽度来修改样式

媒体查询 + rem可以实现不同设备宽度，实现页面元素大小的动态变化

**4.媒体查询 引入资源**

样式比较繁多的时候，可以针对不同的媒体使用不同stylesheets(样式表)

原理：针对不同的屏幕尺寸，调用不同的css文件；就是直接在link中判断设备的尺寸，然后引入不同的css文件

<link rel=”stylesheet” media=”mediatype and|not|only (media feature) ” href=”mystylesheet.css”>

建议：媒体查询最好的方法是从小到大。

**5.less 基础**

css 缺点：css是一门非程序式语言，没有变量、函数、scope(作用域)等概念

css需要书写大量看似没有逻辑的代码，冗余度比较高

不方便维护及扩展，不利于复用

css没有很好的计算能力

非前端开发工程师来讲，往往因缺少css编写经验而很难写出组织良好且便于维护的css代码项目

介绍：less 是css 的一门扩展语言，也称为css预处理器。是在css现有语法上加入程序式语言的特性。

在css语法基础上，引入了变量、mixin(混入)、运算以及函数等功能

总结：less 是一门css预处理器语言，它扩展了css的动态特性。

**6.less 使用基础**

①需要先安装node.js // 验证安装 node -v

②在node环境下安装less。之后，才能使用less npm install -g less // 验证安装 lessc -v

**7.less 使用**

文件后缀名.less，在这个less文件里面书写less语句

变量 @变量名：值 //规范：必须由@为前缀，不能包含特殊字符，不能以数字开头，大小写敏感

嵌套 .header{ a{} } // less 嵌套，

.header{ a{ &:hover{} } }

①子元素的样式直接写到父元素的样式就行了，css不允许，less允许，且自动编译

②如果有伪类、交集选择器、伪类选择器 我们内层选择器前面需要加&符号

运算 变量直接运算，或值直接运算 加减乘除

①乘法\*，除法/  
 ②我们运算符的左右两侧必须敲一个空格隔开  
 ③两个数参与运算 如果只有一个数有单位，则最后的结果就以这个单位为准

④两个数参与运算 如果2个数都有单位，而且不一样，最后的结果以第一个单位为准

**8.less 编译**

本质上，less 包含一套自定义的语法及一个解析器，用户根据这些语法定义自己的样式规则，这些规则最终会通过解析器编译生成对应的css文件。

所以，我们需要把我们的less 文件，编译生成css文件，这样我们的html页面才能使用。

less 转css文件：使用vscode less 插件，(搜索easy less，安装，重新加载一下)，每次保存less文件，在vs中自动生成css文件。这就是这个插件做了less的 编译过程。

**扩展：**一个less 文件，引入到另一个less 文件 @import “common”

@import 导入的意思，可以把一个样式文件导入到另一个样式文件里面(放入到)

link 是把一个样式文件引入到html页面里面

**9.rem 适配方案**

rem 适配方案技术使用(市场主流)

三大步：不同屏幕的媒体查询；确定好当前设备html文字大小；页面元素转成rem 值。然后写内容。

**技术方案1**

less技术 + 媒体查询 + rem

动态设置html标签 字体大小

**技术方案2（推荐）**

flexible.js + rem (淘宝)

总结：两种方案都存在

方案2更简单，现阶段大家无需了解里面的js代码

一般情况下，我们以一套或两套效果图适应大部分的屏幕，放弃极端屏或对其优雅降级，牺牲一些效果，现在基本上以750为准。

做出来的页面漂亮，不同尺寸浏览器，显示大小等都能自动适配。

**flexible.js**

手机淘宝团队出的简洁高效 移动端适配库

不需要再写不同屏幕的媒体查询，因为里面js做了处理

它的原理：把当前设备划分为10等份，但是不同设备下，比例还是一致的。我们要做的就是确定好当前设备的html字体大小就可以了。

**vscode px 转换rem 插件 cssrem**

安装插件。注意：转换时默认html 字体大小基准值是16px。需要修改一下

点击vscode 设置按钮， 打开setting 文件，搜索cssroot，点击左侧编辑，修改大小，重启一下vscode。打开就好使了。

**七、响应式布局**

响应式开发的页面能在pc端，pad端，移动端来显示。一套代码，适配不同设备。

1. 响应式开发原理

就是使用媒体查询针对不同宽度的设备进行布局和样式的设置，从而适配不同设备的目的。

设备划分 尺寸区间

超小屏幕(手机) <768px

小屏设备(平板) >=768px ~ < 992px

中等屏幕(桌面显示器) >=992px ~ <1200px

宽屏设备(大桌面显示器) >=1200px

2. 响应式布局容器

响应式需要一个父级作为布局容器，来配合子集元素来实现变化效果

原理就是在不同屏幕下，通过媒体查询来改变这个布局容器的大小，再改变里面子元素的排列方式和大小，从而实现不同屏幕下，看到不同的页面布局和样式变化。

平时我们的响应式尺寸划分

超小屏幕 设置宽度100%

小屏幕 设置宽度为750px

中等屏幕 宽度设置为970px

大屏幕 宽度设置为1170px

设置宽度比屏幕稍小，为了居中对齐，好看一点。

3. 响应式开发

需求分析：

4. Bootstrap 前端框架

来自Twitter(推特)，是目前最受欢迎的前端框架。Bootstrap 是基于html、css、javascript 的，简洁灵活，使得web开发更加快捷。

**框架：**框架就是一套架构，它有一套比较完整的网页功能解决方案，而且控制权在框架本身，有预制样式库、组件和插件。使用者要按照框架所规定的某种规范进行开发。

**优点：**

标准化的html + css 编码规范

提供了一套简洁、直观、强悍的组件

有自己的生态圈、不断的更新迭代

让开发更简单，提高了开发的效率。

**使用步骤：**1.创建文件夹结构2.创建html骨架结构3.引入相关样式文件4.书写内容

要求当前网页使用IE浏览器最高版本的内核来渲染

<meta http-equiv=”X-UA-Compatible” content=”IE=edge”>

**书写内容**：

直接可以拿框架预先定义好的样式来使用，通过类名设置的。

在类后面添加自定义的类名，修改元素样式，注意权重问题

学好bootstrap 的关键在于知道它定义了哪些样式，以及这些样式能实现什么样的效果。

5. Bootstrap 布局容器

也需要为页面内容和栅格系统包裹一个.container容器，bootstrap 预先定义好了这个类，叫.container 它提供了两个作次用处的类。

1. .container 类

响应式布局的容器 固定宽度

大屏（>=1200px）宽度固定 1170px

中评（>=992px） 宽度固定 970px

小屏（>=768px） 宽度固定 750px

超小屏（100%）

2.container-fluid类

流式布局容器 百分百宽度

占据全部视口（viewport）的容器

适合于单独做移动端开发

6. Bootstrap 栅格系统

container 宽度固定，页面部分自动划分为12等份

页面可以有多个container 组成

<div class=”container”>

<div class=”row”>

<div class=””></div>

</div>

</div>

**栅格选项参数**

栅格系统用于通过一系列行与列的组合来创建页面布局，你的内容可以放入这些创建好的布局中

超小屏幕(手机) 小型设备(平板) 中等屏幕(显示器) 宽屏设备(大桌面显示器)

<768px >=768px >=992px >=1200px

.container 宽度 自动100% 750px 970px 1170px

类前缀 .col-xs- .col-sm- .col-md .col-lg-

**参数：**列数按照不同屏幕划分为1~12等份

行(row) 可以去除父容器作用15px的边距

-xs- 超小 -sm- 小 -md- 中等 -lg-大

列大于12，多余列会作为另一个整体另起一行排列

每一列默认有左右15px像素的padding

可以为一列指定多个类名，以便不同设备显示占据不同列数

**列嵌套：**列嵌套最好加一个row，这样可以取消父元素的padding值，而且高度和父级一样高。

**列偏移：**.col-md-offset-\* 可以将列向右侧偏移，实际上是为当前元素添加了左侧的边距

列排序：了解就行。推和拉操作。

**响应式工具类：**工具类，针对不同设备，元素展示或隐藏

类名 超小屏 小屏 中屏 大屏

.hidden-xs 隐藏 可见

.hidden-sm 可见 隐藏

.hidden-md

.hidden-lg

与之相反，显示 visible-xs…

7. 阿里百秀首页案例

**技术选型**

方案：我们采取响应式页面开发方案

技术：bootstrap 框架

设计图：本设计图采用1280px 设计尺寸

**八、JS**

1. 编程

让计算机为解决某问题而使用某种程序设计语言编写程序代码，并最终得到结果的过程

计算机程序：计算机所执行的一系列的指令集合，人通过计算机执行向计算机发出命令

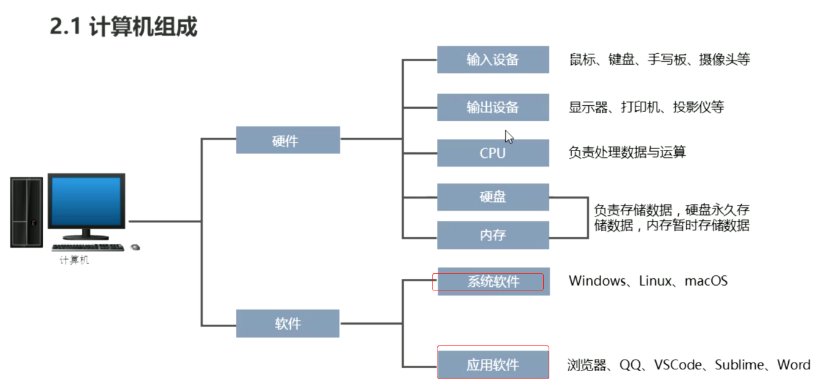
注意：计算机指任何能够执行代码的设备，如智能手机、ATM机、服务器等

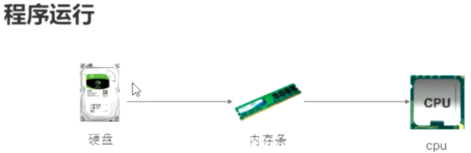
计算机语言、汇编语言、高级语言

翻译器：编程语言 ----->翻译器------> 机器语言

编程语言：有很强的逻辑和行为能力，这是主动的

标记语言(html)：不用于向计算机发出指令，常用于格式化和链接。标记语言的存在是用来被读取的，被动的。





2. JS基础

运行在客户端的脚本语言（script 是脚本的意思）

脚本语言：不需要编译，运行过程中由解释器(js引擎)逐行来进行解释并执行

现在也可以基于node.js 技术进行服务器端编程

**JS作用：**

Js 产生最初目的是表单动态校验（密码强度检测）

网页特效

服务端开发 Node.js

桌面程序(Electron)

App（Cordova）

控制硬件-物联网(Ruff)

游戏开发(cocos2d-js)

**html/css/js 关系**

html 决定网页结构和内容 --描述语言

css 决定呈现给用户的模样 --描述语言

Js 实现业务逻辑和页面控制 --编程语言

**浏览器执行js**

浏览器分为两部分：渲染引擎和JS引擎

渲染引擎：用来解析html和css,俗称内核。比如chrome浏览器的blink，老版本的webkit

JS引擎：也称为JS解析器，用来读取网页中的javascript代码，对其处理后运行，比如chrome浏览器的v8

注意：浏览器本身并不会执行js代码，而是通过内置Javascript引擎（解释器）来执行JS代码。JS引擎执行代码时逐行解释每一句源码(转换为机器语言)，然后由计算机去执行，所以Javascript 语言归为脚本语言，会逐行解释执行。

**JS 组成**：ECMAScript（js语法）、Dom(页面文档对象模型)、Bom（浏览器对象模型）

ECMAscript 即为js。规定了JS编程语法和基础核心知识，是一套语言标准

Dom 通过dom提供的接口可以对页面上的各种元素进行操作(大小、位置、颜色)等

Bom 提供了独立于内容的，可以与浏览器窗口进行互动的对象结构。通过bom可以操作浏览器窗口，比如弹出框、控制浏览器跳转、获取分辨率等

**编写方式**：行内式、内嵌式、引入式

prompt() 浏览器输入框

alert() 浏览器输出框

console.log() 控制台输出

**变量**：存储数据，程序在内存中申请的一块用来存放数据的空间。空间和变量名

注意：只声明，不赋值，结果undefined.

不声明，不赋值 结果报错。 如Tel is not defined

不声明，直接赋值，结果 可以直接使用。变成全局变量

变量严格区分大小写，不能是关键字和保留字。有特殊意义

尽量不要用name作为变量名，在一些浏览器中有特殊意义。

**数据类型：**不同数据占用空间不同。为充分利用存储空间，分为不同数据类型。

js 的变量数据类型是由JS引擎根据=号右侧变量值得类型来判断的。运行完毕之后，变量就确定了数据类型。

Js是动态语言，变量的数据类型是可以变化的。

数据类型分类：简单数据类型、

简单数据类型：number数字型、boolean、string、undefined、null

复杂数据类型：object

其他：NaN 不是数字 用isNaN()。 IsNaN(12); 返回false。IsNaN(‘st’) 返回true

Js 字符串中换行 “\n”

Js 字符串转义符都是以\开头

bool 型，和数值相加，true 代表1，false代表0

undefined 和字符串相加，等于 undefined你好，和1相加，等于NaN

null 和字符串相加，等于null你好，和1相加等于1

**typeOf：获取数据类型**

typeof undefined 等于 undefined

typeof null 等于 object

prompt 取过来的值是字符串类型的。

通过控制台输出的颜色也可以看出是那种类型。（字符串 黑色、数值 蓝色、bool型 ）

转换成字符串：Number.tostring(), String(num)， 拼接

转换成数值型：parseInt(‘3.14’), parseFloat(‘3.14’),Number(‘12’); 隐式转换 （’12’-0）

转换成bool型：Boolean()，代表空、否定的值转换为false,如’’,0,NaN,null,undefined。其余值都转为true。

运算符：加减乘除 取余

浮点数运算，存在精度问题。不要用浮点数计算、做比较。

表达式：由数字、运算符、变量组成的式子，称为表达式。 如：2=1+1.

自增、自减：分前置和后置。如 ++num; num++; 递增和递减运算符必须和变量配合使用。

前置：先加1，后返回值。后置：先返回原值，再加1。

Var e=10;

Var num = e++ + ++e; num = 22;

比较运算符： == 如：18==’18’ 默认转换数据类型，会把字符串类型的数据转换为数字型。值相同就可以。

=== 值和类型完全相同，才返回true

逻辑运算符：&& || ！

赋值运算符：= += -= \*= /= %=

运算符优先级：

**流程控制：顺序结构、分支结构、循环结构** (让代码按照不同的顺序执行)

顺序结构：按从上到下顺序执行

分支结构：根据条件执行 if swich 多分支语句 if(){} else if(){}…

Switch 其中值和类型全等，break不写，不会退出循环，会继续执行下一个case；

循环结构：循环执行 for while do while

三运表达式：处理一些简单扥条件选择。由三元运算符，解决简单的判断 var a = num==1?’true’:’false’;

switch 和 if 区别？

有固定值，用switch；有判断范围的，用if

switch 判断条件后，直接执行到程序的条件语句，效率更高。而if 有几种条件，得判断多次

分支少，用if。分支多，用switch

循环目的：重复的执行某些代码，让代码更简洁、高效

重复执行的代码叫循环体。能否继续执行取决于循环的终止条件

for(初始化变量；条件表达式；操作表达式){}

初始化变量：是一个普通变量，做为计数器

条件表达式：决定是否继续执行

操作表达式：对计数器进行更新

循环执行过程：声明，判断，成功执行语句，更新计数器，判断…不满足，退出循环

循环和多分支结构配合，多次执行不同的操作。

双重for循环

while(判断num){ num++;} 防止死循环

你爱我吗？案例

var msg = prompt(‘你爱我吗？’)

while(msg !==’我爱你’){

Msg = prompt(‘你爱我吗？’)

}

alert(‘我也爱你！’)

do{循环体}while(判断) 先执行一次循环体

do{

var msg = prompt(‘你爱我吗？’)

}while(msg !==’我爱你’)

alert(‘我也爱你！’)

**continue 和 break 区别？**

continue 跳出本次循环，进行下次循环，其他继续执行

break 直接跳出循环，不再执行下次循环

**打断点：**在控制台，source 资源，选中资源文件，在语句打断点

**数组：数据集合，一个变量，存多个值**

利用new 创建数组。Var arr = new Array();创建了一个空数组

用字面量创建数组。Var arr = [1,2]; 其中1,2 为数组元素。元素类型没有限制

用索引取元素值。如 arr[0]， 取值时，索引超出，值为undefined

遍历数组 for(var i=0;i<arr.length;i++){} 遍历：把数组从头到尾为执行一遍

数组加元素：修改length 长度，扩容undefined 元素

Arr[3]=’11’; 3超出索引

算法：观察执行的过程，找出执行规律，用代码写出来

**函数概念：封装一段可以重复执行调用的代码块，目的：就是让大量代码重复执行。封装可以理解为打包。**

函数使用：声明+调用 声明用function，函数名一般为动词，调用时不要忘记小括号。函数不调用，自己不执行。

参数：形参和实参。 声明时，形参。调用时，实参。形参是用来接收实参的。通过参数执行不同的结果或逻辑。

Js中，形参和实参数量可以不匹配。形参可以看成一个不用声明的变量，没有是为undefined

函数返回值，用return实现。函数一般是实现某种功能，结果一般返回给调用者。

return 终止函数，后面有代码不会被执行了。且只能返回一个值。想返回多个值，可以放到一个数组或对象中，返回数组/对象。函数没有return 返回undefined

arguments：当前函数的内置对象，存储了传递过来的所有实参。有length属性，按照索引方式存储，没有数组的一些方法。即为伪数组。可是使用遍历for()。只有函数才有arguments对象。

函数声明方式：1.利用函数关键字自定函数(命名函数) function(){}

2.函数表达式(匿名函数) var name = function(){}

name 是变量名，不是函数名；函数表达式声明方式跟变量差不多，只不过变量里面存的是值，而函数表达式里面存的是函数;函数表达式也可以传递参数

3. 作用域

**作用域**：变量起作用和效果的范围。目的是提高程序可靠性，减少命名冲突。

es6之前分为全局作用域、局部作用域

全局作用域：整个script标签，或一个单独的js文件

局部作用域：又叫函数作用域，变量只在函数内部器作用和效果

变量分类：根据作用域不同，声明的变量分为全局变量和局部变量

注意：在函数内部没有声明接赋值的变量也是全局变量。局部变量在函数外不能直接使用。函数的形参也看做局部变量

从执行效率看，全局变量只有浏览器关闭时才销毁，比较占内存资源。局部变量在函数执行完毕就会销毁，较节约

es6新增一个块级作用域，用{}包含的 如 if(){}

内部函数可以访问外部变量，外部函数不可以访问变量

作用域链：内部函数访问外部函数的变量，采用的是链式查找的方式，一层一层向外查找。这种结构成为作用域链

采用就近原则，如三层函数，最内部调用外部的变量，采用第二层的。

**预解析**：

1.Js引擎运行js 分为两步：预解析 代码执行

预解析： js引擎会把js里面所有的 var 和 function 提升到当前作用域的最前面

代码执行：按照代码书写顺序从上往下执行

2.预解析分为 变量预解析（变量提升） 和函数预解析（函数提升）

变量提升：就是把所有变量声明提升到当前作用域最前面，不提升赋值操作

函数提升：就是把所有的函数声明function \*\*{}提升到当前作用域的最前面，不调用函数

注意：函数表达式，调用必须写在函数表达式的下面，var fun = function(){} fun 是变量

4.Js 对象

对象：对象是一组无序的相关属性和方法的集合，所有具体事物都是对象，例如字符串、数值、数组、函数等。

对象组成：属性和方法，属性是事物的特征(用名词)，方法是事物的行为(用动词)

对象的原因：保存一个值时可用用变量，保存多个值可以用数组，但如何保存一个人的完整信息？

Js对象可以表达结构更清晰、更强大，直观

1.创建对象方式：用字面量、new object()、利用构造函数创建对象

①对象字面量： var obj = {name:’小李’，say:functioin(){}}; 采用键值对形式，方法冒号后面是一个匿名函数

使用属性： 对象名.属性名 或 对象名[‘属性名’]

使用方法： 对象名.方法名();

变量和属性比较？

都是用来存储数据的

变量 单独声明并赋值 使用时直接写变量名 单独存在

属性 在对象里面不需要声明，使用时必须是 对象.属性

函数和方法比较？

都是实现某种功能的

函数是单独声明，并且使用时 函数名() 单独存在

方法是在对象里面，调用的时候 对象.方法()

②利用 new Object() 创建对象

var obj = new Object();

obj.name = ‘aaa’;

利用等号赋值的方式，添加对象的属性和方法

③利用构造函数创建对象

为什么用：创建一个对象，里面很多属性和方法大量相同，我们只能复制；可以用函数的方法封装这些重复的代码

将对象里面一些相同的属性和方法抽象出来封装到函数里面，创建对象。

语法格式：

function 构造函数名(age,name){ // 构造函数

this.age = age；

this.name = name;

this.方法 = function(){}

}

使用：var obj = new 构造函数名(18,’aa’); //对象。调用函数返回的是一个对象

注意：构造函数名首字母要大写

构造函数不需要return 就可以返回对象

必须使用new 创建对象

属性和方法必须使用this

构造函数泛指一类，类似C#中的class。对象 特指一个。通过new 关键字创建对象的过程也称为对象的实例化

new 关键字执行过程（四步）：new 构造函数可以在内存中创建了一个空对象；this就会指向这个对象；执行构造函数里的代码，给这个空对象添加属性和方法；返回这个对象。

④遍历对象

for(变量 in 对象) for(var k in obj){ boj[k] } 输出，k是属性名，obj[k]是属性值。注意：obj[k]不加引号

**JS内置对象**

Js对象分类：自定义对象、内置对象、浏览器对象

前面两种对象是Js基础内容，属于ECMAscript；第三个浏览器对象属于JS独有的，在JS API讲解

自定义对象：自己创建的对象

内置对象：js语言自带的对象，它给开发者提供一些常用的或是最基本而必要的功能(属性或方法)，如Math、Date、Array、String等

浏览器对象：

怎么查内置对象方法：MDN 官网查文档

**Math** 求数值对象：不是构造函数，是静态的，不用new 来调用，直接调用就可以

Math.PI 圆周率属性、Math.max() 求最大值方法、里面有非数字的，返回NaN、如果没有参数，返回-Infinity 负无穷大

Math.abs() 取绝对值、Math.floor() 向下取整、Math.ceil() 向上取整、Math.round() 四舍五入、其他数字都是四舍五入。但是.5特殊，它往大了取值

Math.random() 随机数，返回一个随机小数，范围[0,1)

案例：取两个数之间的随机整数，并且包含这两个整数，再一个随机点名 arr[getRandom(0,arr.length-1)];

function getRange(min,max){

return Math.floor(Math.random()\*(max-min+1))+min;

}

**Date** 日期对象 是一个构造函数，必须使用new 来创建对象

没参数，返回当前系统的当前时间

参数写法，数字型 2019,10,01（10 代表9月） 或者是 字符串型 ‘2021-10-1’

日期格式化 obj.getFullYear() 等 注意：obj.getMonth() 返回的月份小1个月

获取距离1970年1月1号的毫秒数 通过date.valueOf() 或date.getTime() 简单写法 var date = +new Date(); 返回总毫秒数。 H5新增 Date.now(); 别称 时间戳

**数组** 创建：字面量或new Array();

检测是否为数组： instanceof 运算符 可以用来检测是否为数组 arr instanceof Array 或 H5新增 Array.isArray(arr)

在数组末尾添加数组元素 arr.push() 参数直接写新加的元素就可以了，返回值是新数组的长度

在数组开头添加数组元素 arr.unshift()

删除数组最后一个元素 arr.pop() 不跟参数 返回值是被删除的元素，一次只能删除一个元素

删除数组第一个元素 arr.shift()

数组排序

翻转数组 arr.reverse();

数组排序 arr.sort() (冒泡排序)，各位数排序 解决方法 arr.sort(function(a,b){ return a-b; //按照升序的顺序排列 })

数组索引

indexOf() arr.indexOf(‘blue’); 返回第一个元素的索引号，没有该元素返回-1； 从前往后查找

lastIndexOf() arr.lastIndexOf(‘blue’); 从后往前查

数组去重案例：

数组转为字符串 两种

arr.toString() 默认1,2,3 逗号分隔

arr.join(分隔符) arr.join(‘-’)

concat() 连接两个或多个数组，不影响原数组，返回一个新数组

slice() 数组截取 slice(begin,end) 返回被截取项目的新数组

splice() 数组删除splice(第几个开始，要删除个数) 返回被删除项目的新数组，这个会影响原数组

**字符串对象** 简单数据类型为什么会有length 等属性和方法

基本包装类型，把简单数组类型包装成了复杂数据类型

第一步：把简单数据类型包装为复杂数据类型。第二部：把临时变量的值 给str 第三步：销毁这个临时变量

字符串的不可变性：重新赋值，新建一个内存空间，变量指向这个空间。

str.indexOf(‘春’) str.indexOf(‘要查找的字符’，[起始的位置])

var index = str.indexOf(‘o’);

while(index !==-1){

index = str.indexOf(‘o’,index+1);

}

根据位置返回字符

str.charAt(index)

str.charCodeAt(index) 返回响应索引号的字符ASCII值 目的：判断用户按下了那个键

H5 新增 str[index] 获取指定位置处字符

判断用户是否有该数据 if(o[‘age’])

案例：一个字符串，取出数量最多的字符。思路：遍历字符串，将每个字符赋值给对象o,赋值为1，存在加1，遍历对象属性，取出最大值

concat(‘字符串1’，字符串2…) 拼接字符串，类似加号

substr(‘截取的起始位置’，截取几个字符) 截取字符串

replace(‘被替换的字符’，‘替换为的字符’) 替换字符串，只会替换第一个字符

split(‘分隔符’) 字符串转为数组

**简单数据类型与复杂数据类型**

简单数据类型：又叫做基本数据类型或值类型，存储时变量中存储的是值本身。String number boolean undefined null

复杂数据类型：又叫做引用类型,在存储时变量中存储的仅仅是地址(引用)，因此叫做引用数据类型。通过new 关键字创建的对象(系统对象、自定义对象)，如object、Array、Date等

简单数据类型null 返回的是一个空的对象，object;如果有个变量以后打算存储为对象，暂时没想好放啥，这时候就给null

堆：存储复杂类型(对象)，栈中存放地址，指向堆中的存放内存

栈：简单数据类型存储位置

Js中没有堆栈的概念，通过堆栈的方式，可以让大家更容易理解代码的一些执行方式，便于学习

简单数据类型传递参数：赋值给函数的形参，传递的是值，在函数内重新赋值，对函数外的变量值没影响

复杂数据类型传递参数：赋值给函数的形参，传递的是地址，在函数内重新赋值，函数外的变量值也改变

5.Web APIs

Web APIs 和 JS基础关联性：js基础学的是ECMAScript；Web APIs 学的是Dom 和Bom

JS 组成：ECMAScript js语法(js基础)、Dom 页面文档对象模型、Bom 浏览器对象模型

Js基础只能实现页面效果，做不了重用的网页交互效果

Web APIs 是w3c组织的标准，是js所独有的部分，主要学习页面交互功能，需要js基础课程内容做基础

Js基础学习ECMAScript基础语法为后面做铺垫，Web APIs 是js的应用，大量js基础语法做交互效果

API：是为我们程序员提供的一个接口，帮助我们实现某种功能，会使用即可，不必纠结内部如何实现

Web API：是浏览器提供的一套操作浏览器功能和页面元素的API(BOM和DOM)

**Dom**

是一系列的接口，通过这些接口可以改变网页的内容、结构和样式

dom树:在dom页面，把整个页面划分成一个树形结构，称为dom树。

文档：一个页面就是一个文档，dom中使用document表示

元素：页面中的所有标签都是元素，dom中使用element表示

节点：网页中的所有内容都是节点(标签、属性、文本、注释等)，dom中使用node表示

dom把以上内容都看做是对象，dom是按照一种树型结构给我们看的，在实际开发中主要用来操作元素的

获取元素：getElementById()返回选中的元素对象、getElementsByTagName() 对象集合 伪数组存储、

element.getElementsByTagName() 获取某元素里面某类标签

因为文档页面是从上往下加载，所以先得有标签，所以script 写到标签的下面

必须在文档里面，先找到文档，document.getElementById()

console.dir() 打印我们返回的元素对象，更好的查看里面的属性和方法

h5 新增选择元素的接口： ie6 7 8 等没法使用

document.getElementsByClassName(‘’) 根据类名返回元素对象集合

document.querySelector(‘选择器’) 根据指定选择器返回第一个元素对象

document.querySelectorAll(‘选择器’) 根据指定选择器返回所有元素对象集合

获取元素：

获取body 元素：document.body; 返回body 元素对象

获取html元素：document.documentElement; 获取返回html元素对象

事件基础：js使我们有能力创建动态页面，而事件是可以被js侦测到的行为，简单理解：触发---响应机制

事件组成：事件源、事件类型、事件处理程序 也称为事件三要素

事件元素 事件被触发的对象 谁 按钮

事件类型 如何触发 什么事件，比如鼠标点击 onclick 鼠标经过 还是键盘按下

事件处理程序 通过一个函数赋值的方式 完成

var btn = document.getElementById(‘btn’); btn.onclick = function(){}

执行事件步骤：获取事件源 绑定事件或注册事件 添加事件处理程序

操作元素：js的dom操作可以改变网页内容、结构和样式，我们可以利用dom操作元素来改变元素里面的内容、属性等

element.innerText 不识别标签 非标准 去除空格和换行

element.innerHtml 识别标签 w3c 保留空格和换行

常用元素属性操作

element.src=””

表单元素的属性操作

type value checked selected disabled 禁用，如按钮禁用

btn.onclick = function{

this.disalbed = true; this指向的是事件函数的调用者

}

样式属性操作：

通过js修改元素的大小、颜色、位置等样式

element.style 行内样式操作 js修改style样式操作，产生的是行内样式，css权重比较高

element.className 类名样式操作，元素的类名

焦点事件：onfocus 获取焦点事件 失去焦点事件 onblur text.onfocus = function(){}

案例：京东输入框

如果样式比较少或者功能简单的情况下使用style属性 ，如果修改样式较多，推荐使用类名样式操作(修改元素样式类名)。

向保留元素原来的类名，this.className = ‘firstClass newClass’; 多类名选择器

案例：密码框格式提示错误信息

总结：操作元素是dom核心内容。操作元素内容、操作常见元素属性、操作表单元素属性、操作元素样式属性

排他思想：一组按钮添加事件。先把所有元素背景色样式去掉，再让当前元素背景色样式改为pink；(先排除其他人，再设置自己)

案例：百度换肤 核心算法：把当前图片的src获取出来，再设置body src；注意 图片 url()

案例：表格隔行变色效果 技术栈：鼠标经过事件 onmouseover 鼠标离开事件 onmouseout 核心思路：鼠标经过tr行，当前的行变背景颜色，鼠标离开去掉当前行的背景颜色，标题行不需变色，因此我们获取tbody里的行

案例：全选和反选效果 核心思路：全选和取消权限；然后下方复选按钮，影响上方的全选按钮

获取自定义属性：程序员可以添加自定义属性 如：data-index ….等

获取元素属性值：还可以通过 ele.getAttribute(‘属性’)； 获取属性值

不同：ele.属性 获取内置属性值（元素本身自带的属性）

ele.getAttribute(‘属性’) 主要获取自定义的属性(是一种标准)，我们程序员自定义的属性

设置元素属性值：ele.属性=’值’ 设置内置属性值

ele.setAttribute(‘属性’，’属性值’) 主要针对于自定义属性

移除属性值：ele.removeAttribute(‘属性’)；

案例：tab栏切换 核心思路：排他思想，tab li 和content一一对应，自定义 index，让对应索引号的内容显示出来

H5 自定义属性目的：为了保存并使用数据，有些数据可以保存到页面中而不用保存到数据库中。

自定义属性规范：自定义属性容易引起歧义，不容易判断是元素的内置属性还是自定义属性，h5 规定自定义属性date-开头作为自定义属性名；

H5 新增了一个获取自定义属性的方法ele.dateset.属性 或ele.dateset[‘属性’] 如：(date-index) ele.dateset.index

如果自定义属性里面有多个-链接的单词，我们获取的时候采取 驼峰命名法 如： (date-list-name) ele.dateset.listName

dateset 只能获取date-开头的，而且有兼容性问题

节点操作：目的是获取元素

利用dom提供的方法获取元素，缺点是繁琐，逻辑性不强，ul li需先获取ul，

解决方案是利用节点的层级关系获取元素，利用父子兄关系获取元素，逻辑性强，但是兼容性稍差，有解决办法

节点：页面中所有内容都是节点(标签、属性、文本、注释等)，在dom中，节点用node来表示

dom 树中所有节点均可通过js 获取，创建，修改和删除

节点概述：节点至少拥有nodeType(节点类型)、nodeName(节点名称)和nodeValue(节点值)这三个基本属性

实际开发中，我们操作的主要是元素节点

父级节点 node.parentNode; 得到的是离得最近的父级节点（亲爸爸，不是爷爷）

子节点 node.childNodes (标准) 得到的所有子节点，包含元素节点 文本节点等 不常用

node.children (非标准) 但是最常用，兼容好 获取元素节点，也是开发中最常用的

第一个字节点 node.firstChild 第一个子节点，不管是文本节点还是元素节点

Node.firstElementChid 第一个元素节点 在ie9中有兼容性问题

实际开发的写法：既没有兼容性问题又返回第一个子元素写法

第一个 node.children[0] 最后一个node.children[node.children.leng-1]

案例：下拉菜单 实现思路：li都有鼠标经过的事件，经过li 时，让第二个子元素显示出来

兄弟节点 node.nextsibling 下一个兄弟节点，包含文本节点或文本节点等等

node.nextElementSibling 获取当前元素的下一个兄弟元素节点，找不到返回null 有兼容性问题

node.previousSibling 上一个

node.previousElementSibling 有兼容性问题，没有好的解决办法

只能自己yong nextSibling封装一个兼容性的函数

创建节点：我们想要页面中添加一个新元素：1.创建元素。2.添加元素

案例：发布评论

document.createElement(‘li’) 创建后添加到元素

添加节点 node.appendChild(child) node是父级 child是子级 后面添加元素，类似数组的push

node.insertBefore(new, 指定元素)

案例：发布留言 核心思路：新建一个li 添加到ul里面。获取元素、注册事件，事件中 创建元素，添加元素

删除节点：node.removeChild(child)

案例：删除留言案例 实现思路：添加留言li时，多添加一个删除的链接，点击链接是删除

注意：阻止a链接跳转需要添加javascript:void(0); 或者javascript:;

复制节点 node.cloneNode() 如 ul.chidren[0].cloneNode() 步骤：复制节点，放到哪里

括号为空或者里面是false，是浅拷贝，只复制标签，不复制里面的内容，参数为true，即拷贝标签，有复制里面内容

案例：动态生成表格 实现思路：

三种动态创建元素区别？

document.write() 直接将内容写入页面的内容流，正常诶没问题但是如果文档流执行完毕，则它会导致页面全部重绘

element.innerHTML 拼接字符串的形式，重加内存空间，耗时

document.createElement() 添加新的元素，效率比较高

innerHTML 不拼接字符串，采用数组形式拼接，稍复杂，效率比createElement 更高

页面重绘：重新建一个新的页面

window.onload = function(){} 页面加载完成后，再执行js

Dom重点核心：文档对象模型，是W3C组织推荐的处理可扩展标记语言的标准编程接口。可改变页面内容、结构和样式

对于js，为了能够使js操作html，js就有了一套自己的dom编程接口

对于html，dom使得html形成一个dom树，包括文档、元素、节点

我们获取过来的DOM元素是一个对象，所以称为文档对象模型

关于dom操作，我们主要针对元素的操作。主要有创建、增、删、改、查、属性操作、事件操作

**事件高级：**

注册事件(绑定事件)

注册事件概述：给元素添加事件，称为注册事件或绑定事件。

注册事件有两种方式：传统方式和方法监听注册方式

传统方式：利用on 开头的onclick 特点：注册事件唯一性，后面注册的处理函数会将前面注册的处理函数覆盖掉

方法监听注册方式：w3c标准 推荐方式，addEventListener() 它是一个方法 IE9 之前不支持，可以用attachEvent 代替

特点：同一个元素同一个事件可以注册多个监听器，按注册顺序依次执行

语法：ele.addEventListener(type, listener[,useCapture]) 将指定的监听器注册到目标对象上，当该对象触发指定的事件时，就会执行事件处理函数

type:事件类型字符串，如click、mouseover,注意这里不要带on

listener：事件处理函数，事件发生时，会调用该监听函数

useCapture：可选参数，bool类型，默认false

可以封装一个注册事件的兼容性解决函数，兼容性处理的原则：首先照顾大多数浏览器，再处理特殊浏览器

删除事件(解绑事件)

传统模式：ele.onclick = null;

方法监听注册方式：ele.removeEventListerner(‘类型’, 函数名); ie9 一下，用detachEvent 替代

DOM 事件流

是从页面中接收事件的顺序。事件发生时会在元素节点之间按照特性的顺序传播，这个传播过程即dom事件流

Dom事件流分为3个阶段：捕获阶段、当前目标阶段、冒泡阶段

事件冒泡：IE最早提出，事件开始时由最具体的元素接收，然后逐级向上传播到dom最顶层节点的过程

事件捕获：网景最早提出，由dom最顶层节点开始，然后逐级向下传播到最具体的元素接收的过程

注意：js代码中只能执行捕获或者冒泡其中的一个阶段

onclick 和 attachEvents(ie) 只能得到冒泡阶段

捕获阶段 如果addEventListener 第三个参数是true，那么则处于捕获阶段 document->html->body->fater->son

冒泡阶段 如果addEventListener 第三个参数是false或省略，则处于冒泡阶段 son->fater->body->html->document

实际开发中我们很少使用时间捕获，更关注时间冒泡

有些事件是没有冒泡的，比如 onblur、onfocus、onmouseenter、onmouseleave

事件冒泡有时候会带来麻烦，有时候又会帮助很巧妙的作某些事件，后面讲解

事件对象：ele.onclic=function(event){}

Event 就是一个事件对象，写到我们侦听函数的，小括号里面 当形参来看

事件对象只有有了事件才会存在，它是系统给我们自动创建的，不需要我们传递参数

事件对象是我们事件的一系列相关数据的集合 跟事件相关的，比如鼠标点击里面就包含了鼠标的相关信息，鼠标坐标啊，如果是键盘事件里面就包含的键盘事件的信息 比如 判断用户按下了那个键

事件对象可以自己命名，如e 等 兼容问题，可通过window.event 获取，扩展兼容性 e=e||window.event;

事件对象常见属性和方法

e.target 返回的是触发事件的对象（元素） 函数内如果有this，this返回的是绑定事件的对象(元素) 兼容性问题，e=e||window.event; target = e.target||e.srcElement;

了解 跟this有个非常相似的属性 currentTarget 有兼容性问题

e.type 事件的类型

阻止默认行为(事件)，让链接不跳转或不让提交按钮不提交 e.preventDefault(); //dom标准写法 兼容低版本 e.returnValue;也可以利用 return false；没有兼容性，有个特点，return 后面代码不执行，只限于传统注册方法

e.stopPropagation(); 阻止冒泡事件 兼容处理 e.cancelBubble = true;

事件委托(代理、委派)

事件冒泡本身的特性，会带来的坏处，也会带来好处，需要我们灵活掌握

原理：不是每个子节点单独设置事件监听器，而是事件监听器设置在其父节点上，然后利用冒泡原理影响设置每个子节点

只操作了一次dom，提高了程序的效率

鼠标事件

禁止鼠标右键菜单 contextmenu 主要控制应该何时显示上下文菜单，主要用于程序员取消默认的上下文菜单

document.addEventListener(‘contentmenu’,function(e){

e.preventDefualt();

})

禁止鼠标选中 selectstart

document.addEventListener(‘selectstart’,function(e){

e.preventDefualt();

})

鼠标事件对象 event 对象代表事件的状态，跟事件相关的一系列信息的集合。现阶段我们主要是用鼠标事件对象

MouseEvent和键盘事件对象 KeyboardEvent

e.clientX 鼠标在可视区的x坐标

e.pageX 鼠标在页面文档的x坐标。

e.screenX 在电脑屏幕的x坐标

案例：跟随鼠标的天使 核心思路：鼠标移动事件，获取最新的鼠标坐标，把这个x和y作为图片的left 和right值

键盘事件

onkeyup 按键弹起时触发的事件

onkeydown 按键按下时触发的事件

onkepress 按键按下时触发的事件 不识别功能键，如ctrl shift箭头等

有执行顺序的，先down 然后 press 然后keyup

键盘事件对象 keycode 属性 按键的ASCII码值 keyup 和 keydown 不区分字母大小写的，keypress 区分

案例：模拟京东按键输入内容 核心思路：检测用户是否按了s键，true时，把光标定位到搜索框里

案例：模拟京东快递查询单号 核心思路：输入内容，大号盒子显示 keyup 事件 keydown keypress 在文本落入之前触发，所以用keyup，keypress 不识别功能键，更不能用

**BOM-浏览器对象模型**

它提供独立于内容而与浏览器窗口进行交互的对象，其核心对象是window，可以吧浏览器看做一个对象看待

bom由一系列相关的对象构成，并且每个对象都提供了很多方法与属性，缺少标准，兼容性较差

window：包含document、location、navigation、screen、history

window对象是浏览器的顶级对象，它具有双重角色

它是js访问浏览器窗口的一个接口

它是一个全局对象，定义在全局作用域中的变量、函数都会变成window对象的属性和方法

注意：window下的一个特殊属性window.name;

window 常见事件

window 事件调用时，经常可以省略window

窗口加载事件：window.onload = function(){} 或window.AddEventListerner(“load”,function(){})

Window.onload 是窗口(页面)加载事件，当文档内容完全加载完成会触发该事件(包括图像、脚本文件、css文件等)，就调用的处理函数

只能写一次，如果写多次，以最后一个为准。addEventListerner 就没有限制

相似的：document.addEventListener(‘DOMContentLoaded’,function(){}) DomContentLoaded事件触发时，仅当DOM加载完成，不包括样式表、图片，flash等等 ie9以上才支持。如果页面图片比较多，耗时比较长，则适合用这个

调整窗口大小事件 window.onresize() = function(){} 或 window.AddEventListener(‘resize’,function(){})

是调整窗口大小时加载的事件，当触发时就调用的处理函数，经常用来响应式布局

window.innerWidth 当前屏幕的宽度

两种定时器

setTimeout(调用函数，延时时间) 单位是毫秒，省略是0，立刻执行 只执行一次，就结束这个定时器

页面中可能有很多的定时器，我们经常给定时器加个标识符(名字)

回调函数：需要等待事件，事件到了才去调用这个函数，因此称为回调函数，回头再调用

普通函数：按照顺序依次调用

案例：5秒之后自动关闭广告 核心思路：5秒之后，隐藏掉

停止定时器：clearTimeout(‘定时器的名字’);

setInterval(调用函数，延时时间) 每隔延时时间到了，就去调用函数，一直执行。

案例：京东秒杀倒计时 核心思路：根据时间获取剩余小时、分钟、秒，利用setInerver 放入对应盒子

有个问题，刷新是会有空白，因为定时器1秒钟后才执行，解决方法：先执行一次函数

清除定时器 clearInterval(‘定时器的名字’)

案例：发送短信 核心思路：按钮点击，禁用按钮，通过定时器控制按钮内容，事件到了去除禁用

this 指向问题

全局作用域或者普通函数中this指向全局对象window(注意定时器里面的this指向window)

方法调用中谁调用指向谁

构造函数中this指向构造函数的实例

js执行机制

js是单线程的，也就是说，同一时间只能做一件事。

这是因为这门语言诞生的使命决定的。Js是为处理页面中用户的交互，以及操作dom而诞生的。

单线程就意味着所有任务需要排队，前一个任务结束，才会执行下一个任务。导致的问题：如果js执行事件过长，造成页面的渲染不连贯，导致页面渲染加载阻塞的感觉。

同步和异步：为解决js渲染加载阻塞的问题，利用多核cpu的计算能力，h5提出Web Worker标准，允许js脚本创建多个线程，于是js中出现了同步和异步。

同步：前一个任务结束后再执行后一个任务，程序的执行顺序与任务的排列顺序是一致的、同步的。

异步：通知处理多个任务。现在js允许执行多个任务

同步任务都在主线程上执行，形成一个执行栈

异步任务，js的异步是通过回调函数实现的。异步任务相关回调函数添加到任务队列中（任务队列也称为消息队列）

一般而言，异步任务有一下三种类型：

①普通事件，如click resize等

②资源加载，如load error 等

③定时器，包括setInterval、setTimeout等

执行机制：先执行栈中的同步任务，遇到异步任务(回调函数)提交给 异步进程处理 (如果点击了)放入任务队列中，不点击不会在异步任务队列中出现。一旦主线程执行栈中的所有同步任务执行完毕，系统就会按照次序读取任务队列中的异步任务，于是读取的异步任务结束等待状态，进入执行栈，开始执行。

先执行同步任务，再执行异步任务

事件循环：由于主线程不断的重复获取任务、执行任务、再获取任务、再执行，所以这种机制称为事件循环（eventloop）

location 对象

window对象给我们提供了一个location属性用于获取或设置窗体的url，并且可以用于解析URL。因为这个属性返回的是一个对象，所以我们将这个属性也称为location对象。

url：统一资源定位符，是互联网上标准资源的网址。互联网上每个文件都有一个唯一的url，它包含的信息指出文件的位置以及浏览器应该怎么处理它。

地址组成：protocol：//host [:port]/path/[?query]#fragment

protocol：通信协议 常用的http，ftp，maito 等

host:主机（域名）

port：端口号 可选，省略时使用方案默认端口，如http的默认端口为80

path:路径 一般用来表示主机上的一个目录或文件地址

query:参数 以键值对的形式存在，用&符号分隔开

fragment: 片段 #后面内容，常见于链接 锚点

location.href 获取或设置整个url

location.host 返回主机(域名) 如：www.baidu.com

location.port 返回端口号 如果未写则返回 空字符串

location.pathname 返回路径

location.search 返回参数

location.hash 返回片段 #后面内容，常见于链接 锚点

location.href 赋值，实现跳转页面的效果

案例：5秒后自动跳转到某页面 核心思想：利用定时器，给location.href 赋值

案例：获取url参数在不同页面之间的传递 核心思路：

location对象的方法

location.assign(‘http://www.baiud.com’) 跟href一样，可以跳转页面，也称为重定向页面。记录历史，可以返回

location.replace(‘http://www.baidu.com’) 替换当前页面，不记录历史，不能后退

location.reload(); 重新加载页面，相当于刷新按钮或f5。如果参数为true,强制刷新 相当于ctrl + f5

navigator 对象

包含有关浏览器的信息，它有很多属性，常用的是userAgent，该属性返回由客户机发送服务器的user-agent头部的值。

history对象

与浏览器历史记录进行交互，该对象包含用户(在浏览器窗口中)访问过的url

history.back() 方法 可以后退功能

history.forward() 方法 前进功能

history.go(参数) 方法 前进后退功能，参数为-1，后退1个页面；参数是1，前进1个页面。

6.PC 端网页特效

**offset 系列**

offset 系列的相关属性可以**动态的得到**该元素的位置（偏移）、大小等

获得元素距离带有定位父元素的位置

获得元素自身的大小(宽度高度) 注意：返回的数值 都不带单位

属性：

ele.offsetParent 返回作为该元素带有定位的父级元素 如果父级都没有定位则返回body

ele.offsetTop 返回元素相对带有定位父元素上方的偏移

ele.offsetLeft 返回元素相对带有定位父元素左边框的偏移

ele.offsetWidth 返回自身包括padding、边框、内容区的宽度，返回值不带单位

ele.offsetHeight 返回自身包括padding、边框、内容区的高度，返回数值不带单位

offset 与 style 的区别？

style 只能获得行内样式表中的样式值；offset 可以得到任意样式表中的样式值

style 获得的是电邮单位的字符串； offset 返回的数值没有单位

style 获得不包括padding和border; offsetWidth 包含padding + border + width

style 获得的是可读写属性，可以获取也可以复制；offsetWidth 等属性是只读属性

改写等适合用style offset 适合查看

案例：获取鼠标在盒子中位置的坐标放大镜效果） 核心思路：盒子内点击，得到鼠标距盒子左右距离；再获取鼠标在页面坐标；再得到盒子在页面中距离；盒子距离页面坐标减去盒子在页面中距离。

案例：模态框拖拽

核心思路：点击按钮，模态框和遮挡层显示出来；点击关闭，模态框和遮挡层隐藏;

案例：京东放大镜效果 核心思路：

**元素可视区client 系列**

client 翻译就是客户端，通过client系列相关属性获取元素可视区的相关信息。可以动态获取该元素的边框大小、元素大小等。

ele.clientTop 返回元素上边框的大小

ele.clientLeft 返回元素左边框的大小

ele.clientWidth 返回自身包括padding、内容区的宽度，不含边框，返回数值不带单位

ele.clientHeight 返回自身包括padding、内容区的高度，不含边框，返回数值不带单位

案例：淘宝flexible.js 源码分析

立即执行函数 (function(){}) 主要作用：创建一个独立的作用域，避免了命名冲突的问题

普通函数需要先声明，再调用。立即执行函数：不需要声明，立马就能执行的函数。

写法：(function(){})() 或者 (function(){}()); 可以传递参数进入给函数。多个立即函数，必须用分号分隔

里面的变量都是局部变量，不会有命名参数冲突问题

window 的pageshow 事件，重新加载页面触发的事件

load 与pageshow?

触发load事件情况：a标签的超链接；F5或者刷新按钮(强制刷新)；前进后退按钮

但是在火狐中，有个特点 往返缓存，这个缓存中不仅保存着页面数据，还保存了dom和javascript 的状态；实际上是将整个页面都保存在了内存里。所以此时后退按钮不能刷新页面。

可以使用pageshow 事件触发。这个事件在页面显示时触发，无论页面是否来自缓存。在重新加载页面中，pageshow会在load事件触发后触发；根据事件对象中的persisted 来判断是否缓存中的页面触发的pageshow事件，注意这个事件给window添加。

**元素滚动scroll 系列**

相关属性可以动态得到该元素的大小、滚动距离等。

ele.scrollTop 返回被卷取的上侧距离，返回数据不带单位

ele.scrollLeft 返回被卷取的左侧距离，返回数据不带单位

ele.scrollWidth 返回自身实际的宽度，不含边框，含padding,返回数值不带单位

ele.scrollHeigth 返回自身实际的高度，不含边框，含padding返回数值不带单位

scroll 事件：当滚动条发生变化时会被触发

案例：仿淘宝固定侧边栏

页面被卷去的头部兼容性解决方案：

需注意，页面被卷去的头部，有兼容性问题，因此被卷去的头部通常有如下几种写法：

1.声明了DTD，使用document.documentElement.scrollTop

2.未声明DTD，使用document.body.scrollTop

3.新方法window.pageYOffset 和 window.pageXOffset , IE9开始支持

老师封装了getScroll() 方法，需要的话，查看一下

mouseenter 和 mouseover 的区别？

mouseover 鼠标经过自身盒子会触发，经过子盒子还会触发，有冒泡概念

mouseenter 只会经过自身盒子才会触发，没有冒泡的概念

**动画函数封装：**

动画实现核心原理：通过定时器setInterval() 不断移动盒子位置，注意：元素必须有定位

动画函数给不同元素记录不同定时器（自己专门用自己的定时器），

核心原理：利用js是一门动态语言，可以很方便的给当前对象添加属性。

缓动效果原理：缓动动画就是让元素运动速度有所变化，最常见的是让元素慢慢停下来，思路：让盒子每次移动的距离慢慢变小，速度就会慢慢落下来。核心算法：（目标值-现在的位置）/10 作为每次移动的距离步长。停止条件：盒子位置等于目标位置。步长值需要向上取整。

动画函数添加回调函数：原理：函数可以作为一个参数，将这个参数作为参数传到另一个函数里面，当那个函数执行完之后，再执行传进去的这个函数，这个过程就叫做回调。

案例：京东右侧滑动显示、隐藏

案例：网页轮播图

无缝滚动：

手动调用：手动调用点击事件 ele.click();

节流阀：防止轮播图按钮连续点击造成播放过快。目的：当上一个函数动画内容执行完毕，再去执行下一个函数动画，让事件无法触发。核心思路：利用回调函数，添加一个变量来控制，锁住函数和解锁函数。

callbak && callBack() 写法，短路运算符，只有两者都为true时，才执行函数。

案例：淘宝返回顶部

案例：筋斗云效果

**移动端常见特效：**

触屏事件：移动端浏览器兼容性较好，我们不需要考虑以前js的兼容性问题，可以放心的使用原生JS书写效果，但是移动端也有自己独特的地方。比如触屏事件touch，Android 和IOS都有

touchstart 手指触摸到一个dom元素时触发

touchmove 手指在一个dom元素上滑动是触发

touchend 手指从一个dom元素上移开时触发

触摸事件对象 touchEvent

touches 正在触摸屏幕的所有手指的列表

targetTouches 正在触摸当前dom元素的手指列表 ,如果侦听的是一个dom元素，touches 和targetTouches 是一样的

changedTouches 手指状态发生了改变的列表，从无到有 或从有到无

当我们手指离开屏幕的时候，就没有了touches 和targetTouches 列表，但是会有changedTouches

因为我们一般都是触摸元素，所以重点注意 targetTouches

移动端拖动元素

移动端拖动的原理：手指移动中，计算出手指移动的距离。然后用盒子原来的位置+手指移动的距离

拖动元素三部曲：

①触摸元素touchstart：获取手指初始坐标，同时获得盒子原来的位置

②移动手指touchmove：计算手指的滑动距离，并且移动盒子

③离开手指touchend

注意：手指移动也会触发滚动屏幕所以这里要阻止默认的屏幕滚动e.preventDefault();

移动端常见特效：

案例：移动端轮播图

检测过渡完成事件 transitionend

classList 属性：是h5新增的一个属性，返回元素的类名。但是ie10以上版本支持

该属性用于在元素中添加，移除以及切换css类。有以下方法

添加类：ele.classList.add(‘类名’) 在后面追加类名，不会覆盖以前的类名 注意前面不要加小点 .

移除类：ele.classList.remove(‘类名’)

切换类：ele.classList.toggle(‘类名’) 自动判断有没有这个类，有则去掉，没有则添加上

案例：移动端返回顶部：

移动端click 延时解决方案

移动端click事件会有300ms的延时，原因是移动端屏幕双击会缩放页面

解决方案：1.禁止缩放。浏览器禁用默认双击缩放行为并且去掉300 ms的点击延迟

<meta name=”viewport” content=”user-scalable=no”>

②利用touch事件自己封装这个事件解决300ms延迟。了解看视频

③使用插件，fastclick插件解决300ms延时

插件：js插件是一个js文件，遵循一定规范编写，方便程序展示效果。拥有特定功能且方便调用。如轮播图和瀑布流插件

特点：它一般是为了解决某个问题而专门存在，功能单一，并且比较小。方便快捷

**Swiper 插件的使用：**

1.引用插件相关文件

2.按照语言规范使用插件。 注意不要更改里面的结构和类名。因为js会有一个调用的效果

移动端其他插件插件：superslide 插件、iscroll 插件

插件的使用总结：

①确认插件实现的功能

②去官网查看使用说明

③下载插件

④打开demo实例文件，查看需要引入的相关文件，并且引入

⑤赋值demo实例文件中的结构html，样式css以及js代码

案例：移动端视频插件zy.media.js

h5提供了video标签，但是浏览器的支持情况不同，不同的视频格式文件，我们可以通过source解决。

但是外观样式，还有暂停，播放，全屏等功能我们只能自己写代码解决。这个时候我们可以使用插件方式来制作。

移动端常用框架：

框架，即为一套架构，它基于自身的特点向用户提供一套较为完整的结局方案。框架的控制权在框架本身，使用者要按照框架所规定的某种规范进行开发。

而插件一般是为了结局某个问题而专门存在，其功能单一，并且比较小。

前端常用的框架有Bootstrap、Vue、Angular、React等。技能开发PC端，也能开发移动端

前端常用的移动端插件有swiper、superslide、iscroll等

框架：大而全，一整套解决方案。

插件：小而专一，某个功能的解决方案。

**本地存储**

window.sessionStorage F12 在Application 下可以查看

window.localStorage

本地存储特性

①数据存储在用户浏览器中

②设置、读取方便、甚至页面刷新不丢失数据

③容量较大，sessionStorage约5M、localStorage约20M

④只能存储字符串，可以将对象JSON.stringfy()编码后存储

sessionStorage 生命周期为关闭浏览器窗口；在同一个窗口(页面)下数据可以共享；以键值对的形式存储使用

存储数据 sessionStorage.setItem(key,value);

获取数据 sessionStorage.getItem(key);

删除数据 sessionStorage.removeItem(key,value);

清除所有 sessionStorage.clear();

localStorage 生命周期永久生效，除非手动删除 否则关闭页面也会存在；可以多窗口(页面)共享(同一浏览器可以共享)，以键值对存在。

存储数据 localStorage.setItem(key,value);

获取数据 localStorage.getItem(key);

删除数据 localStorage.removeItem(key,value);

清除所有 localStorage.clear();

案例：记住用户案例

**九、JQuery**

Js库：即library，是一个封装好的特定的集合(方法和函数)。从封装一大堆函数的角度理解库，就是在这个库中，封装了很多预先定义好的函数在里面，比如动画animate、hide、show，比如获取元素等。

简单理解：就是一个js文件，里面对我们原生js代码进行了封装，存放到里面。这样我们就可以高速的使用这些封装好的功能了。

Jq库就是为了方便的操作dom，里面基本都是函数(方法)。

常见js库：jquery、prototype、YUI、Dojo、Ext JS、移动端的zepto

这些库都是对原生js的封装，内部都是用javascript 实现的，我们主要学习JQuery

jQuery 是一个快捷、简洁的javascript库，才几十kb。其设计宗旨是写的更少，做更多的事。

J 是 javascript;Query 是查询；意思是查询js，把js的dom操作做了封装，我们可以快速的查询使用里面的功能。

封装了javascript 常用的功能代码，优化了dom操作、事件处理、动画设计和Ajax交互，学习怎么调用这些函数

Jquery：轻量级，跨浏览器，基本兼容现在主流的浏览器。链式编程、隐式迭代；对事件、样式、动画支持，大大简化了dom操作；支持插件扩展开发。有丰富的第三方差价。如属性菜单、日期控件、轮播图等；免费开源

使用：引入，使用

入口函数：等dom加载完成再指定js代码

$(function(){}); 相当于DOMContentLoaded 主要元素加载完成后

$ 是jquery 简称，是Jquery的顶级对象。相当于原生js中的window。

**jQuery对象和DOM对象？**

用原生js获取过来的元素就是Dom对象

用jQuery获取过来的对象就是jQuery对象。本质是通过$对dom对象包装后产生的对象(伪数组形式存储)

jQuery对象只能使用jQuery方法，DOM对象则使用原生的JavaScript属性和方法。不能混用。

**Dom对象和jQuery 对象之间是可以相互转换的。**

原生js比jQuery更大。原生的一些属性和方法jQuery没有给我们封装，要使用这些方法需把jQuery对象转换为Dom对象才能使用。如视频的play 在原生中有，jquery中没有

Dom对象转jQuery对象：$(DOM对象) 如：$(‘div’)

jQuery对象转为DOM对象(两种方式)：$(‘div’)[index] 或 $(‘div’).get(index) index是索引号

**jQuery 常用API**

jq基础选择器：原生js获取元素方式很多，很杂，而且兼容性情况不一致，因此jq给我们做了封装，使获取元素统一标准。

$(‘选择器’) 里面选择器与css选择器相同，但要加引号

ID选择器 $(“#id”)

全选选择器 $(“\*”)

类选择器 $(“.class”)

标签选择器 $(“span”)

并集选择器 $(“div,span,li”)

交集选择器 $(“li.current”)

子代选择器 $(“ul>li”)

后台选择器 $(“ul li”)

jq 样式 $(“div”).css(“background”,”pink”);

隐式迭代(重要)：遍历内部Dom元素(伪数组形式存储)的过程就叫做隐式迭代

简单理解：给匹配到的所有元素内部进行循环遍历，执行相应的方法，而不用我们再进行循环，简化我们的操作，方便我们调用

**Jq筛选选择器：**

:first $(“li:first”) 获取第一个li元素

:last $(“li:last”) 获取最后一个li元素

:eq(index) $(“li:eq(2)”) 获取到的li元素中，选择索引号为2的元素，索引号index从0开始

:odd $(“li:odd”) 获取到的li元素中，选择索引号为奇数的元素

:even $(“li:even”) 获取到的li元素中，选择索引号为偶数的元素

**jQuery 筛选方法(重点)**

parent() $(“li”).parent() 查找父级，亲爸爸

children(selector) $(“ul”).children(“li”) 相当于$(“ul>li”),最近一级(子代选择器)

find(selector) $(“ul”).find(“li”) 相当于$(“ul li”)，后代选择器

siblings(selector) $(“.first”).siblings(“li”) 查找兄弟节点，不包括自己本身

nextAll([expr]) $(“.first”).nextAll() 查找当前元素之后所有的同辈元素

prevtAll([expr]) $(“.last”).prevAll() 查找当前元素之前所有的同辈元素

hasClass(class) $(‘div’).hasClass(“protected”) 检查当前的元素是否含有某个特定的类，如果有返回true

eq(index) $(‘li’).eq(2) 相当于$(“li:eq(2)”)，index从0开始

鼠标经过时间：$(“.nav>li”).mouseover(function(){}) 注册事件都成为方法的形式。

$(this) 把原生代码的this 包装起来了

**Jq里面的排他思想**

//隐式迭代 绑定点击事件

$(“button”).click(function(){

$(this).css(“background”, “pink”)

$(this).siblings(“button”).css(“background”,””);

})

案例：淘宝服饰精品案例 核心思路：tab栏切换

Jq得到当前元素的索引号 $(this).index();

**操作css 方法**

Jq可以使用css方法来修改简单元素样式；也可以操作类，修改多个样式

1.参数只写属性名，则是返回属性值 $(this).css(‘color’);

2.参数是属性名,属性值。用逗号分隔，则是设置一组样式，属性必须加引号，值如果是数字可以不同跟单位和引号

$(this).css(‘width’,300)

3.参数可以是对象形式，方便设置多组样式。属性名和属性值用冒号隔开，属性可以不用加引号

$(‘div’).css({‘color’:’white’, ‘font-size’:’20px’}); 或 $(‘div’).css({color:’white’, font-size:20,backgroundColor:’red’});

如果是复合属性，则使用驼峰命名法

**设置类样式方法**

作用等同于以前的classList，可以操作类样式，注意操作类里面的参数不要加点

添加类 $(‘div’).addClass(‘current’)

删除类 $(‘div’).removeClass(‘current’)

切换类 $(‘div’).toggleClass(‘current’)

案例：tab栏切换 核心思路：

链式编程操作和隐式迭代 $(this).addClass(‘current’).siblings().removeClass(‘current’);

**类操作与className区别？**

原生js中className 会覆盖元素原先里面的类名

Jq里面类操作只是对指定类进行操作，不影响原先的类名

**Jq 效果**

Jq给我们封装了很多动画效果，最为常见的如下：

显示隐藏：show()、hide()、toggle()

滑动 ：slideDown()、slideUp()、slideToggle()

淡入淡出：fadeIn()、fadeOut()、fadeToggle()、fadeTo()

自定义动画：animate()

show([speed,[easing,[fn]]) 参数都可省略，无动画直接显示。speed，速度。easing,指定切换效果。fn，回调函数

hide([speed,[easing,[fn]])

toggle([speed,[easing,[fn]]) 一般情况下，不加参数，直接显示隐藏就可以了

**事件切换效果**

hover([over,]out)

over：鼠标移到元素上要触发的函数(相当于mouseenter)

out：鼠标移出元素要触发的函数(相当于mouseleave)

$(‘.nav>li’).hover(function(){

$(this).children(‘ul’).slideDown(200); // 移入

},function(){

$(this).children(‘ul’).slideUp(200); // 移出

})

更简洁写法：如果只写一个函数，那么鼠标经过和鼠标离开都会触发这个函数

$(‘.nav>li’).hover(function(){

$(this).children(‘ul’).slideToggle(200);

})

**动画队列及其停止排队方法**

动画或效果队列：动画或者效果一旦触发就会执行，如果多次触发，就造成多个动画或者效果排队执行。

停止排队：stop 方法必须写到动画的前面，这是一个规范。相当于停止结束上一次的动画。

$(this).children(‘ul’).stop().slideToggle(200);

fadeTo() 修改透明度，这个方法速度和透明度必须要写

案例：高亮显示案例 核心思路：鼠标放到元素上，让它兄弟元素透明度降低

动画都会有排队为题，用stop() 处理

**自定义动画animate()**

animate(params,[speed],[easing],[fn])

params：想要更改的样式属性，以对象形式传递，必须写。属性名可以不用带引号，如果是复合属性则需要采取驼峰命名法borderLeft。其余参数都可省略。

案例：王者荣耀的手风琴案例 核心思路：每个小li都有一个连接，两个图片

**jq属性操作**

设置或获取元素固有属性值prop()。所谓固有属性就是元素本身自带的属性，如a标签的href,input标签的type等

获取属性 prop(‘属性’)

设置属性 prop(‘属性名’, ‘属性值’)

自定义属性 通过attr() 获取或设置自定义的属性 如data-index

另一种获取数据方法：数据缓存data() 元素作为变量来看。

data() 方法里面的数据是存放在元素的内存里面，并不会修改dom结构。一旦页面刷新，之前存放的数据都将被移除。

date() 方法获取data-index h5自定义属性，第一个不用谢data-，而且返回的是数字型 如：$(“div”).data(“index”);

案例：购物车全选按钮 核心思路：事件可以用change()方法 :checked 选择器，被选中的复选框 parents(‘’) 祖先级父元素选择器 toFiexd(2) 保留两位小数 parseInt()转成int类型

**jq 内容文本值**

获取元素内容 html()

设置元素内容 html(‘内容’)

设置或设置文本 text()

获取或设置表单值 val()

**jq 元素操作**

主要是遍历、创建、添加、删除元素操作

遍历元素

Jq隐式迭代是对同一类元素做了同样的操作。如果想要给同一类元素做不同操作，就需要用到遍历

语法：$(‘div’).each(function(index, item){}) 回调函数第二个参数是一个dom元素对象，想要调用jq方法，需$(item) 转换成jq对象。

第二种写法：$.each($(‘div’),function(index,item){}) 不同是这个主要用来处理数据的（数组或对象）

创建元素： $(‘<li></li>’) 创建元素，放到某个元素里

$(‘ul’).append(li) 内部添加且放到内容的最后面

$(‘ul’).prepend(li) 内部添加且放到内容的最前面

ele.after(‘内容’) 外部添加且放到目标元素后面

ele.before(‘内容’) 外部添加且放到目标元素前面

内部添加元素，生成之后，他们是父子关系；外部添加元素，生成之后，他们是兄弟关系

删除元素 $(‘ul’).remove() 删除匹配的元素,自杀; $(‘ul’).empty() 删除匹配元素的子节点，删孩子；等同于$(‘ul’).html(‘’)

**jq事件**

单个事件注册：ele.事件(function(){ //事件处理程序 })

其他事件和原生基本一致。

如：mouseover、mouseout、blur、focus、change、keydown、keyup、resize、scroll等

事件处理：on() 绑定事件 on方法在匹配元素上绑定一个或多个事件的事件处理函数

ele.on(events, [selector], fn);

events：一个或多个用空格分隔的事件类型，如click或keydown

selector：元素的子元素选择器，如给li 添加事件

fn：回调函数

$(‘div’).on({mouseenter:function(){ },click:function(){ }})

优势1：可以绑定多个绑定事件

如：$(‘div’).on(‘mouseenter mouseleave’,function(){ //事件处理程序如果是一样的，可以这样写

$(this).toggleClass(‘current’);

})

优势2：可以事件委派操作。事件委派的定义就是，把原来加给子元素身上的事件绑定到父元素身上，就是把事件委派给父元素。$(‘ul’).on(‘click’,’li’,function(){ //事件处理 }) //给ul绑定点击事件，触发者是li，省了隐式迭代，提高工作效率

优势3：（最重要）动态创建的元素，click() 没有办法绑定事件，on()可以给未来动态生成的元素绑定事件。传统绑定事件时还没有子元素，新创建的元素没有绑定事件。$(‘ul’).on(‘click’,’li’,function(){ //事件处理 }) 则解决了这个问题

案例：微博发布 核心思路：on可以给动态生成的元素添加事件绑定事件

事件处理 off() 解绑事件：off() 方法可以移除通过on方法添加的事件处理程序

$(‘div’).off() 解除了所有的绑定事件

$(‘div’).off(‘click’) 解除点击事件

$(‘ul’).off(‘click’,’li’); 解绑事件委托

如果有的事件只想触发一次，可以使用one()来绑定事件

**jq自动触发事件** trigger()

某些事件希望自动触发，比如轮播图自动播放功能跟点击右侧按钮一致。可以利用定时器自动触发右侧按钮点击事件，不必鼠标点击触发。

第一种：元素.事件(); 如$(‘div’).click();

第二种：元素.trigger(‘事件’) $(‘div’).trigger(‘click’);

第三种：元素.triggerHandler(‘事件’) 这个不会触发元素的默认行为。

如input 得到焦点 光标闪烁 $(‘input’).on(‘focus’,function(){})

**jq事件对象**

事件被触发，就会有事件对象的产生

ele.on(events, [selector], function(e){}) e就是事件对象，事件对象可以所很多操作，

如：阻止默认行为：e.preventDefault() 或者return false；阻止冒泡：e.stopPropagation();

**jq 拷贝对象**

如果想要把某个对象拷贝(合并)给另一个对象使用，此时可以使用$.extend()方法

语法：$.extend([deep], target, object1, [objectN] )

deep：如果设为true为深拷贝。默认为浅拷贝

target：要拷贝的目标对象，拷贝给这个对象

object1：待拷贝到第一个对象的对象。

$.extend(targetObj, obj) // 会覆盖targetObj 里面原来的数据

浅拷贝：是把被拷贝的对象复杂数据类型中的地址拷贝给目标对象，修改目标对象会影响被拷贝对象。

深拷贝：是把里面的数据完全复制一份给目标对象，如果里面有不冲突的属性，会合并到一起。

**jq多库共存**

jq使用$符号作为标识符，随着jq的流行，其他js库也会用这个符号作为标示符，这样一起使用会引起冲突。

$ 的简单封装：

$(function(){

function $(ele){

return document.querySelector(ele);

}

$(‘div’)

$.each() // 会报错

})

客观需求：需要一个解决方案，让jq和其他的js库不存在冲突，可以同时存在，这就叫做多库共存。

Jq解决方案：

1.把里面的$符号 统一改为jQuery。比如jQuery(‘div’);

2.让jq释放对$控制权，让用自己决定 var tidai = jQuery.noConflict();

**Jq 插件**

Jq功能比较有限，想要用更复杂的特效效果，可以借助jq插件完成。

注意：这些插件也是依赖于jQuery来完成的，所以必须要先引入jQuery文件，因此也称为jQuery插件

Jq插件常用的网站：

1.jQuery 插件库：https://www.jq22.com/

2.jQuery 之家：http://www.htmleaf.com/ 免费开源的

Jq插件使用步骤：

第一步：引入相关文件 (jq文件和插件文件)

第二步：复制相关html、css、js

案例：瀑布流插件 核心思路：引用插件

**图片懒加载**

图片使用延迟加载可提高网页的下载速度。它也能帮助减轻服务器的负载。当我们页面滑动到可视区域，再显示图片。只显示用户看到区域的图片，其它图片不加载。

借助插件，搜索栏加载。

使用jquery 插件库 EasyLazyload。注意：此时的js引入文件和js调用必须写到dom元素(图片)最后面

全屏滚动 fullpage.js 插件 网址：<https://github.com/alvarotrigo/fullpage.js>

bootstrap.js 插件相关使用

bootstrap 框架也是依赖于jQuery 开发的，因此里面的js插件使用，也必须引入jQuery 文件。

组件：一些常用组件

插件：

显示模态框 两种方式

第一种：data-toggle=”modal” data-target=”#btn” 在模态框里定义 id=“btn”

第二种：通过js 显示 $(‘.myBtn’).on(‘click’, function(){ $(‘#btn’).modal(); });

案例：ToDolist 注意：本地存储只能存储字符串的格式。把数组格式转为字符串格式 JSON.stringfy(obj);

JSON.parse(str) 字符串转为对象 splice(index,1) 数组删除元素

**Jq尺寸、位置操作**

width()/height() 获取/设置元素width和height大小 如$(‘div’).width()

innerWidth()/innerHeight() 获取/设置元素width和height + padding大小

outerWidth()/outerHeight() 获取/设置元素width和height + padding + border大小

outerWidth(true)/outerHeight(true) 获取/设置元素width和height+ padding + border + margin大小

如果以上参数为空，则是获取响应值，返回的是数字型。

如果参数为数字，则是修改相应值。 不用写单位

位置offset()、position()、scrollTop()/scrollLeft()

①offset() 设置或获取元素偏移，距离文档的位置，跟父盒子没有关系 该方法有2个属性。Left 和top

获取偏移 $(‘.son’).offset().top ; 距离头部距离

设置元素的偏移：offset({top:10,left:30});

②position() 获取元素偏移。获取距离带有定位父级位置(偏移)，如果没有带有定位的父级，则以文档为主。该方法只能获取，不能设置

③scrollTop()/scrollLeft() 设置或获取元素被卷去的头部和左侧

如$(window).scroll(function(){

If($(document).scrollTop()>= 100){}

})

案例：带有动画的返回顶部 主要思路：使用animate animate只有元素才能做动画，不能是文档

案例：品优购电梯导航 主要思路：

**十、JS 高级**

编程思想：面向过程、面向对象

面向过程：pop 即分析出解决问题所需要的步骤，然后用函数一步一步实现，使用时再一个一个调用。

面向对象：oop 即把事务分解成一个个对象，然后由对象之间分工与合作。先找出对象，并写出对象有哪些功能，再做为调度者把这些功能实用起来。

面向对象特性：封装性、继承性、多态性

面向对象思维特点：抽像对象公用的属性和行为组织(封装)成一个类(模板)。对类进行实例化，获取类的对象

在js中，对象是一组无序的相关属性和方法的集合，所有的事物都是对象，例如字符串、数值、数组、函数等。

对象由属性(特性)和方法(行为)组成。

类泛指某一类，对象指具体的对象。

**es6创建类：**

class Start{

constructor(){}

sing(){}

}

利用类创建对象 new Start();

类 constructor：是类的构造函数(默认方法)，用于传递参数，返回实例对象。通过new 命令生成对象实例时，自动调用该方法。如果没有显示定义，类内部会自动给我们创建一个constructor();

注意：①类里面所有的函数不用加function ②多个函数之间不要加逗号分隔

**继承**：extends 子类继承父类的属性和方法 class Son extends Father{}

**Super 关键字**：用于访问和调用父类上的构造函数。

①继承中，如果实例化子类输出一个方法，先看子类有没有这个方法，如果有就执行子类的

②如果没有，就去查找父类有没有这个方法，有则执行父类的这个方法(就近原则)

如果想调用父类中的方法，可以在子类方法中用super.say() 调用父类普通方法

注意：super 必须在子类this 之前调用

注意：es6中类没有变量提升，所以必须先定义类，才能通过类实例化对象

类里面的共有的属性和方法一定要加this使用

类里this指向问题。Constructor 里面的this指向实例对象，方法里面的this指向这个方法的调用者。

案例：面向对象的tab 栏 核心思路：

ele.insertAdjacentHTML() 可以直接把字符串格式元素添加到父元素中

input .seelct() 让其中值处于选中状态

**构造函数和原型**

以前有三种方式创建对象

方式一：对象字面量

方式二：new Object()

方式三：自定义构造函数

function Start(uname,age){

this.uname = uname;

this.age = age;

this.sing = function(){}

}

构造函数是一种特殊的函数，主要用来初始化对象，即为对象成员变量赋初始值，它总与new 一起使用。我们可以把对象中一些公共的属性和方法抽取出来，让后封装到这个函数里面。

在js中，使用构造函数时要注意两点：构造函数用于创建某一类对象，其首字母要大写；构造函数要和new 一起使用才有意义。

new 在执行时做四件事：

①在内存中创建一个新的空对象

②让this指向这个新的对象

③执行构造函数里面的代码，给这个新对象添加属性和方法

④返回这个新对象(所以构造函数里面不需要return )

实例成员：就是构造函数内部通过this添加的成员 uname,age 等就是实例成员。实例成员只能通过实例化对象来访问

静态成员：在构造函数本身上添加的成员 Start.sex=’男’ 静态成员只能通过构造函数来访问，不能通过实例化对象访问

构造函数缺陷：如存在sing()函数，多个对象创建多个内存空间，存在浪费内存的问题。

构造函数原型：构造函数通过原型分配的函数是所有对象所共享的，解决构造函数的缺陷问题。 prototype

每个构造函数都有一个prototype，指向一个对象，直接把方法放到这个对象上，所有势力共享这些方法。

每个实例对象都有一个\_\_proto\_\_，指向构造函数的prototype原型对象，两者是等价的

方法查找规则：先查找实例对象上是否有这个方法，若没有，通过\_\_proto\_\_,上构造函数原型对象上prototype上查找该方法

constructor 构造函数：对象原型\_\_proto\_\_ 和 构造函数prototype 原型对象里面都有一个属性constructor属性，称为构造函数，因为它指回构造函数本身。

用处：很多情况下，需要手动利用constructor指回原来的构造函数

原型链：\_\_proto\_\_

**this 指向**

指向函数的调用者

原型对象应用：通过原型对象，扩展内置对象没有的自定义方法 Array.prototype.sum = function(){}

通过 构造函数+原型对象模拟实现继承，称为组合继承

Call() 调用这个函数，并且修改函数运行时的this指向 fun.call(objArr, arg1…) objArr 修改this 指向

**ES5 新增方法**

**数组方法** 迭代(遍历)方法 forEach()、map()、filter()、some()、every()

arr.forEach(function(item,index,arr){})

arr.filter(function(item,index,arr){}) 过滤器，返回一个新的数组，主要用于筛选数组

arr.some(function(item,index,arr){}) 判断，返回一个bool 值，判断是否存在满足条件的元素

arr.map(item=> ‘item.name’) 处理，把数组处理后，返回处理后的数组

arr.every(item=>)

案例：查询商品 核心思路：数据遍历

**字符串方法** trim() 清除字符串两端的空白字符 str.trim()。不影响原字符串本身，返回一个新的字符串

**对方方法**

Object.keys() 用于获取对象自身所有的属性

Object.keys(obj) 效果类似 for…in 返回一个由属性名组成的数组

以前obj.name=’aa’ 新增、修改对象的属性

Object.defineProperty() 定义对象中新属性或修改原有的属性 Object.defineProperty(obj, prop, descriptor)

obj ：修改的对象

prop：添加或修改的属性

descriptor:以对象形式{}书写，决定属性的特性。其中值如下

value:设置属性的值，默认为undefined

writable:值是否可以重写，默认为false

enumerable:目标属性是否可以被枚举。默认为false

configurable:目标属性是否可以删除或是否可以再次修改特性。默认为false

delete obj.name 删除对象中的属性

**函数进阶**

函数定义：函数声明方式function 关键字(命名函数)；函数表达式(匿名函数) var fun = function(){}

第三种方式：var f = new Function(‘console.log(123)’); 利用new Function(‘参数1’, ‘参数2’, ‘函数体’) 函数体以字符串形式。书写麻烦，效率低。所有函数都是Function 的实例(对象) 函数也属于对象

函数的调用：

1.普通函数 fn() 或 fn.call()

2.对象的方法 obj.sing()

3.构造函数 new Star();

4.绑定事件函数 点击按钮就可以调用

5.定时器函数 定时器自动自己调用

6.立即执行函数 自动调用

**函数内This 指向问题** 调用方式的不同决定了this的指向不同，一般指向我们的调用者

普通函数 this指向window

对象的方法 this 指向 的是调用者obj

构造函数 this 指向 如obj 这个实例对象

绑定事件函数 this 指向 函数的调用者 如btn 按钮对象

定时器 this 指向 函数的调用者 window

立即执行函数 this 指向 函数的调用者 window

**改变函数内部的this指向**  常用 bind()、call()、apply()

fun.call(o, 1,2) //将函数内this指向o对象，call还可以调用函数。call 主要作用可以实现继承

fun.apply(o) //可以调用函数，可以改变函数内this的指向。但是它的参数必须是数组(伪数组)。apply 主要应用:比如借助于数学内置对象求最大值 Math.max.apply(null, arr);

fn.bind(o) //不会调用函数，但是能改变函数内部this的指向。返回的是原函数改变this之后新的函数

bind 典型应用：有的函数不需要立即调用，但是又想改变函数内部this指向。如一个按钮，点击了之后，禁用这个按钮，3秒后再开启这个按钮，定时器内部函数的指向

**严格模式** strict mode es5 ie10 之上，消除js 开发不严谨的，为了js 发展 ‘use strict’

为整个脚本(script标签)开启严格模式：

为函数开启严格模式：

变量变化：变量必须先声明再使用。2.不能随意删除已声明的变量3.严格模式下全局作用于函数中this指向undefined

4.严格模式下，如果构造函数不加new调用，this会报错5.定时器函数内，this还是指向window

函数变化：严格模式下，同一函数参数不允许有同名。2.不允许在非函数代码块内声明函数

**高级函数**

高级函数是对其它函数进行操作的函数，它接收函数作为参数或将函数作为返回值输出。

**闭包 closure**

根据作用域不同分为：全局变量和局部变量

①函数内部可以使用全局变量

②函数外部不可以使用局部变量

③当函数执行完毕，本作用域内的局部变量会销毁

闭包：指有权访问另一个函数作用域中变量的函数。 函数内部定义另一个函数，内部函数可以访问外部函数变量

在外部函数外面，访问它的变量，思路：返回内部的函数。

闭包的作用：延伸了变量的作用范围。容易造成内存泄漏。

案例：循环注册点击事件 核心思路：

以前是通过动态添加属性的方式实现。立即执行函数也称为小闭包。

案例：循环中的定时器 3秒后打印小li 核心思路：

案例：计算打车价格 核心思路：

**函数递归**

一个函数在内部可以调用其本身，那么这个函数就是递归函数。自己调用自己。容易造成栈溢出问题。死循环

递归里面必须加return 退出条件

案例：1到n的阶乘 核心思路：注意退出条件

案例：利用递归求斐波那锲数列 核心思路：

案例：根据id返回数据对象 核心思路：

**浅拷贝和深拷贝**

浅拷贝只拷贝一层，更深层次的对象级别只拷贝引用。深拷贝拷贝多层，每一级别的数据都会拷贝

浅拷贝可以用for…in

es6 新增浅拷贝方法 Object.assign(target,…source)

深拷贝：利用函数递归进行深拷贝

**正则表达式 Regular Expression**

是用于匹配字符串中字符组合的模式。js中正则表达式也是对象。灵活性、逻辑性、功能性非常强。

如：验证表单、过滤内容中的一些敏感词、从字符串中获取我们想要的特定部分

创建：var reg = new RegExp(/表达式/)

字面量创建： var rg = /123/ 正则表达式里面不需要加引号，不管是数字型还是字符串型

测试正则表达式：test() 测试是否符合正则表达式规范

regObj.test(str)

正则表达式组成：由元字符组成，是具有特殊意义的专用符号

边界符： ^ 是以什么开头 $ 是以什么结尾

/abc/ 只要包含abc的都是true

/^abc/ 以abc开头的都是true

/abc$/ 以abc结尾的都是true

字符类： [] 只要匹配其中一个就可以了

/^[abc]$/ 只有三个中的一个才是true，多选一，多次的话为false

[-] 范围，如[a-c] a到c中的任意一个

[a-zA-z] 字符组合。

[^a-z] 中括号中的^ 符号表示取反

量词符：用来设定某个模式出现的次数

\* 0次或多次

+ 1次或多次

？ 0次或1次

{n} n次

{n,} n次或更多次

{n,m} n次到m次

案例：验证用户名 核心思路：失去焦点验证

中括号 字符集合，匹配方括号中的任意字符

大括号 量词符，里面表示重复次数

小括号 表示优先级

预定义类：某些常见模式的简写方式

\d 匹配0-9 之间的任一数字 相当于[0-9]

\D 匹配0-9以外的字符 相当于[^0-9]

\w 匹配任意的字母、数字和下划线 相当于[a-zA-Z0-9\_]

\W 除所有字母、数字和下划线以外的字符，相当于[^a-zA-Z0-9\_]

\s 匹配空格(包括换行符、制表符、空格)，相当于[\t\r\n\v\f]

\S 匹配非空格的字符，相当于[^\t\r\n\v\f]

| 或者符号

正则表达式：<https://c.runoob.com/front-end/854/>

案例：验证表单 核心思路：失去焦点，验证，调整提示信息

**正则表达式中的替换**：str.replace(‘替换项’,’替换为’) str.replace(/andy/,’baby’);

第一个参数：被替换的字符串 或者 正则表达式

第二个参数：替换为的字符串

返回值是一个替换完毕的新字符串

正则表达式参数 /表达式/[switch] switch 也称为修饰符，按照什么样的模式来匹配。

有三种值：

g：全局匹配

i：忽略大小写

gi：全局匹配 + 忽略大小写

案例：str.replace(/激情|gay/g, ‘\*\*’);

**ES6** 新的标准化规范

语言更完善，功能加强

如：变量提升特性增加了程序运行的不可测性

语法过于松散，实现相同的功能，不同的人可能会写出不同的代码

**let** 新增关键字，声明的变量只在所处的块级作用域有效

防止循环变量编程全局变量

声明的变量不存在变量提升

暂时性死区，在块级作用域外用var声明同名变量，在let 声明之前，使用会报未声明错误

**const** 声明常量。值不能改变，声明数组，地址不能改变。有块级作用域。声明变量时必须赋值。

var let const 区别？

点：范围 变量提升 值可更改性

**解构赋值**

es6中允许从数组中提取值，按照对应位置，对变量赋值。对象也可以实现解构。

let [a,b,c,d] = [1,2,3]; 解构给变量赋值,d变量为undefined

对象结构：let {name, age, score} = {name:’张三’, age:18}; score 变量没有对应属性，为undefined

let {name:myName,age:myAge} = {name:’张三’, age:18}; 左侧name 用于变量匹配，赋值给myName

箭头函数：()=>{} 用来简化函数定义语法的

const fn = ()=>{}

箭头函数不绑定this关键字，箭头函数中的this，指向的是函数定义位置的上下文this

对象不能产生作用域

**剩余参数**

剩余参数语法允许我们将一个不定数量的参数表示为一个数组 …args

const sum = (…args)=>{

let total = 0;

args.forEach(item=> total+=item;);

return total;

}

let [s1, …s2] = [‘张三’, ‘李四’, ‘王五’]

**Array 扩展运算符**

扩展运算符可以将数组或对象转为用逗号分隔的参数序列

let arr = [1,2,3];

…arr 1,2,3 利用扩展运算符将数组转为以逗号跟个的参数序列

扩展运算符可以应用于合并数组

let arr1 = [1,2,3];

let arr2 = [4,5,6];

let arr = […arr1, …arr2]; 合并后的数组

也可以用 arr1.push(…arr2); 进行合并

将伪数组转为真正的数组也可以用 var arr= Array.from(伪数组, item=>item\*2)。第二个参数可空，代表只转换

find() 查找复合条件的数据，没有返回undefined var ret = arr.find((item,index)=> item.id==2);

findIndex() 查找第一个复合条件的数据索引 没有返回-1 var i = arr.findIndex(item=> item>15);

includes() 判断数组是否包含给定的值，返回布尔值 var b = [1,2,3].includes(‘a’);

**模板字符串**

`这是一个模板${ age }字符串`

模板字符串中可以解析变量；内容可以换行； 模板字符串中可以调用函数，显示函数返回值；

**字符串startsWith() 和 endsWith()** 扩展方法

判断字符串是否以某字符开头或结尾

**字符串repeat** 方法

将原字符串重复n次，返回一个新字符串

const ret = ‘x’.repeat(3) 返回 ‘xxx’

**Set数据结构**

Es6 提供了新的数据结构Set。它类似于数组，但是成员的值都是唯一的，没有重复的值

Set本身是一个构造函数，用来生成set数据结构

例如：搜索关键字的历史记录

const set = new Set(); new Set([1,2,3,4]); set.size 长度

利用set 做数组去重 const set = new Set([…arr]);

添加值 set.add(‘a’).add(‘b’) 可以链式调用

删除值 set.delete(‘a’); 返回布尔值

判断是否存在 set.has(‘a’) 返回布尔值

清空 set.clear()

遍历 set.forEach((item)=>{ })

**十一、Node**

1.基础

node 开发属于服务器开发

能够和后端程序员更加紧密的配合

网站业务逻辑前置，学习前端技术需要后端技术支撑(ajax)

node 使用js语法开发后端应用，前端开发工具大多基于node开发

服务端开发需要：

实现网站的业务逻辑

数据的增删改查

**Node是什么？**

node是一个基于Chrome V8 引擎的js 代码运行环境

浏览器(软件) 能够运行js代码，浏览器就是js代码的运行环境

Node(软件)能够运行js代码，node就是js代码的运行环境

**环境搭建（node）** https://nodejs.org/zh-cn/

LTS 长期支持稳定版 Current 最新 实验版

下载安装包直接安装即可，利用命令行工具进行操作。

node -v 查看安装是否成功

**安装常见错误**

错误代码 2502 、2503 系统账户权限不足

解决办法： ①以管理员身份运行powershell 命令行工具 ②输入运行安装包msiexec/package node安装包位置

执行命令报错 node -v

原因：node 安装目录写入环境变量失败

解决办法：将Node安装目录手动添加到环境变量中，重新打开命令行工具

命令行工具执行命令怎么查找node -v 先去当前工作目录查找，找不到，再到系统环境变量path 目录中查找命令的位置

cd demo 切换到demo 文件夹

./ 当前目录

../ 上一目录

**Node.js 组成**

Js 由ECMAScript、Dom、Bom 组成，ECMAScript 是核心，dom和bom是浏览器环境为其提供的一些API

Node.js 由ECMAScript 及 Node 环境提供的一些附件API组成的。包括文件、网络、路径等等一些更加强大的API

所有ECMAScript 语法在node环境中都可以使用

**快捷操作命令行工具**

在node 文件所在目录 Shift 键 + 鼠标右键 打开命令行工具

Node 01(前几个字符) + tab键 可直接补全该文件

命令行工具中 clear 清除之前

**Node.js 模块化开发**

Js存在问题：文件依赖不明确和命名冲突导致代码覆盖

软件模块化开发：一个功能就是一个模块，多个模块可以组成完整应用，抽离一个模块不会影响其它功能的运行。

node.js 中模块化开发规范：规定一个js文件就是一个模块，模块内部定义的变量和函数默认情况下在外部无法得到。

模块内部可以使用exports对象进行成员导出，使用require方法导入其它模块。模块导入时文件后缀可以省略

exports.version = version;

const a = require(‘./a.js’)

模块成员导入还可以用 module.exports.version = version 导出；

两种导出方式：exports 是 module.exports 的别名(地址引用关系)，导出对象最终以module.exports 为准。

**系统模块**

node运行环境提供的API，这些api都是以模块化的方式进行开发的，所以我们又称node运行环境提供的api为系统模块。

【1】fs文件操作系统模块 file system 文件操作系统

const fs = require(‘fs’); //导入fs模块

①读取文件内容 fs.readFile(‘文件路径/文件名称’[,’文件编码’], callback);

fs.readFile(‘../css/base.css’, ‘utf-8’, (err, doc)=>{

// 如果文件读取发生错误，参数err 的值为错误对象 否则err 的值为null。Doc 参数为文件内容

if(err == null){

console.log(doc);

}

})

②写入文件内容 fs.writeFile(‘文件路径/文件名称’, ‘数据’, callback);

const content = ‘<h3>正在使用fs.writeFile写入文件内容</h3>’

fs.writeFile(‘../index.html’, content, err=>{

if(err != null){

console.log(err);

return;

}

console.log(‘文件写入成功！’);

})

**系统模块 path 路径操作**

为什么要进行路径拼接？ 因为不同操作系统的路径分隔符不统一 /public/uploads/avatar

Windows 上是\ /

Linux 上是/

path.join(‘路径’, ‘路径’, …)

const path = require(‘path’); //导入path 模块

let finialPath = path.join(‘itcast’, ‘a’, ‘b’, ‘c.css’);

**相对路径vs绝对路径**

相对路径：当前命令行工具的当前工作目录

绝对路径：\_\_dirname 文件绝对路径

**第三方模块**

别人写好的、具有特定功能的、我们能直接使用的模块即第三方模块。

由于第三方模块通常都是由多个文件组成并且被放置在一个文件夹中，所以又称为“包”。

第三方模块存在形式：两种

第一种：以js文件的形式存在，提供实现项目具体功能的API接口。

第二种：以命令行工具形式存在，辅助项目开发

获取第三方模块 通过npmjs.com 第三方模块的存储和分发仓库

npm(node package manager):node 的第三方模块管理工具

下载：在命令行工具输入 npm install 模块名称 npm 是命令 install 是下载 默认下载到命令行工具当前工作目录下

卸载：npm uninstall package 模块名称

本地安装和全局安装：一般是按照下方方式进行安装

命令行工具：全局安装

库文件 ：本地安装

**第三方模块 nodemon**

nodemon 是一个命令行工具，用以辅助项目开发

在node.js中，每次修改文件都要在命令行工具中重新执行该文件，非常繁琐，nodemon 帮助监控文件在保存后立即执行它。

使用步骤：

①使用npm install nodemon -g 下载它

②在命令行工具中用nodemon 命令替代node 命令执行文件

ctrl + c 退出当前文件监控

**第三方模块 nrm** 命令行工具

nrm(npm registry manager)：npm下载地址切换工具，提高模块下载速度

npm默认的下载地址在国外，国内下载速度慢

使用步骤：

①使用npm install nrm -g 下载

②查询可用下载地址列表 nrm ls

③切换npm 下载地址 nrm use 下载地址名称

2.Gulp

**第三方模块Gulp**

基于node 平台开发的前端构建工具

将机械化操作编写成任务，想要执行机械化操作时执行一个命令行命令任务就能自动执行了

Gulp 能做什么

项目上线，html、css、js文件压缩合并

语法转换(es6、less..)

公共文件抽离

修改文件浏览器自动刷新

Gulp使用

①使用npm install gulp 下载库文件

②在项目根目录下建立gulpfile.js文件

③重构项目的文件夹结构 src目录放置源文件 dist 目录放置构建后文件

④在gulpfile.js 文件中编写任务

Gulp 中提供的方法

gulp.src() 获取任务要处理的文件

gulp.dest() 输出文件

gulp.task() 建立gulp 任务

gulp.watch() 监控文件的变化

const gulp = require(‘gulp’)

gulp.task(‘first’,()=>{ // 使用gulp.task() 方法建立任务。任务名称参数，任务的回调函数

gulp.src(‘./src/css/base.css’); // 获取要处理的文件

.pipe(gulp.dest(‘./dist/css’)); // 将处理后的文件输出到dist目录

})

在命令行工具中 npm install gulp-cli -g 全局安装

执行gulp first 执行任务

⑤在命令行工具中执行gulp任务

**Gulp插件**

gulp-htmlmin：html文件压缩

gulp-csso：压缩css

gulp-babel：js语法转换

gulp-less：less语法转换

gulp-uglify：压缩混淆javascript

gulp-file-include：公共文件包含

browersync：浏览器实时同步

插件使用：

①下载插件

②引用插件

③编写任务，调用插件

**Gulp node\_modules 文件夹的问题**

①文件夹以及文件过多过碎，当我们将项目整体拷贝给别人的时候，传递速度会很慢很慢

②复杂的模块依赖关系需要被记录，确保模块的版本和当前保持一致，否则会导致当前项目运行报错

解决方法：package.json 文件作用

项目描述文件，记录了当前项目信息，例如项目名称、版本、作者、github地址、当前项目依赖了哪些第三方模块等

在项目根目录下 使用npm init -y 命令快速生成package.json文件。

**项目依赖**

在项目的开发阶段和线上运营阶段，都需要依赖的第三方包，称为项目依赖

使用npm install 包名 命令下载的文件会默认添加到package.json 文件的dependencies 字段中

{

“dependencies”:{

“jquery”:”^3.3.1”

}

}

**开发依赖**

在项目的开发阶段需要依赖，线上运营阶段不需要依赖的第三方包，称为开发依赖

使用npm install 包名 –save-dev 命令将包添加到package.json文件的devDependencies字段中，以此来区分项目依赖和开发依赖。

**package-lock.json 文件作用**

**scripts 作用**

**Node.js 中模块的加载机制**

情况一：模块查找规则-当模块拥有路径但没有后缀时

①require 方法根据模块路径查找模块，如果是完整路径，直接引入模块

②如果模块后缀省略，先找同名js文件再找同名js文件夹

③如果找到了同名文件夹，找文件夹中的index.js

④如果文件夹中没有index.js 就会去当前文件夹中的package.js文件中查找main选项中的入口文件

⑤如果找指定的入口文件不存在或者没有指定入口文件就会报错，模块没有找到

情况二：模块查找规则-当模块没有路径且没有后缀时

node.js 会假设它是系统模块

node.js 会去node\_modules 文件夹中

首先看是否有该名字的js文件

再看是否有该名字的文件夹

如果是文件夹看里面是否有index.js

如果没有index.js 查看该文件夹中的package.json 中的main选项确定入口文件

3.URL

统一资源定位符，是专为表示Internet网上资源位置而设的一种编码方式，我们平时所说的网页地址指的就是url。

**URL的组成**

传输协议：// 服务器IP或域名:端口/资源所在位置表示

http: 超文本传输协议，提供了一种发布和接收html页面的方法

IP地址或域名地址：找到要请求的服务器

端口：在没有输入端口时，浏览器在请求是默认加上80端口

**开发过程中客户端和服务器端说明**

在开发阶段，客户端和服务器端使用同一台电脑，即开发人员电脑

开发人员电脑：客户端(浏览器) 服务器端(Node)

每台电脑都有一个默认的ip和域名 127.0.0.1 localhost

**node创建网站web服务器**

const http = require(‘http’) // 引用系统模块

const app = http.createServer(); // 创建web服务器

app.on(‘request’,(req,res)=>{ // 当客户端发送请求的时候，node 和js一样，是基于事件驱动的语言

res.end(‘<h1>hi, user</h1>’) //响应

});

app.listen(3000); // 监听3000端口

**http协议**

超文本传输协议：规定了如何从网站服务器传输超文本到本地浏览器，它基于客户端服务器架构工作，是客户端(用户)和服务器端（网站）请求和应答的标准。

客户端和服务器端进行沟通的统一规范

报文：在http请求和响应的过程中传递的数据块就叫报文，包括要传送的数据和一些附加信息，并且要遵守规定好的格式

----------------------------------------------------------------------------------------------------------------------------------------------------------------

客户端 =》 请求报文 =》 服务器端

请求方式 post

请求地址 [www.itcast.com](http://www.itcast.com)

Hello,请给我一个消息

-------------------------------------------------------------------------------------------------------------------------------------------------------------

客户端 《= 响应报文

内容类型 text/html 《= 服务器端

内容长度 20

Hi，我就是好消息

----------------------------------------------------------------------------------------------------------------------------------------------------------------

请求方式：Get 请求 post 请求

在服务器端 req.method 为请求的方式

表单中 method：指定当前表单提交的方式

action：指定当前表单提交的地址

http 状态码 200 请求成功

404 请求资源没有找到

500 服务器端错误

400 客户端请求有语法错误

内容类型 text/html

text/css

在服务器端 req.url 为请求地址

req.headers 获取请求报文信息

req.headers[‘accept’]

req.writeHead(400);

req.writeHead(400, { ‘content-type’:’text/plain；charset:utf-8’ }) 返回状态码和内容类型

请求参数：客户端向服务器端发送请求时，有时需要携带一些客户信息，客户信息需要通过请求参数的形式传递到服务器端，比如登录操作。

get请求参数：参数被放置在浏览器地址栏中，如<http://localhost:3000/?name=zhangsan&age=20>

接收 req.url

node 提供了url 内置模块，url.parse(req.url, true) 第一个参数要解析的url地址，第二个参数将查询参数解析成对象形式

let params = url.parse(req.url, true).query; params.name params.age

let {query, pathname} = url.parse(req.url, true); 解析请求路径，请求参数

post 请求参数：参数在请求报文中，post参数是通过事件的方式接收的，使用querystring 系统模块将参数转换为对象格式

接收 data 当请求参数传递的时候发data事件

end 当参数传递完成的时候发end事件

post 参数不是一次就能接收完的，分多次接收。node 提供另一个内置模块处理post请求参数 querystring 模块

let postParams = ‘’;

req.on(‘data’,params=>{

postParams += params;

});

req.on(‘end’,()=>{

console.log(querystring.parse(postParams));

})

**路由**

路由是指客户端请求地址与服务器端程序代码的对应关系。简单的说，就是请求什么就响应什么

其实路由即是一些判断代码

const http = require(‘http’);

const url = require(‘url’);

const app = http.createServer();

app.on(‘request’,(req, res)=>{

const method = req.method.toLowerCase(); // 获取请求方式

const pathname = ur.parse(req.url).pathname; // 获取请求地址

res.writeHead(200, {‘content-type’:’text/html;charset=utf-8’})

if(method ==’get’){

if(pathname ==’/’ || pathname ==’/index’){

res.end(‘首页’)

}else if(pathname==’/list’){

}else{

}

}else if(method==’post’){

}

})

app.listen(3000);

**静态资源**

服务器端不需要处理，可以直接响应给客户端的资源就是静态资源。如css、js、image文件

mime 第三方插件，处理请求文件类型插件

**动态资源**

相同的请求地址不同的响应资源，这种资源就是动态资源

**同步API，异步API(node中)**

同步API:只有当前API执行完成后，才能继续执行下一个API

异步API:当前API的执行不会阻塞后续代码的运行

同步API 和异步API的区别？

①同步api可以从返回值中拿到api执行的结果，但是异步api是不可以的。

只能通过回调函数拿到返回值，自己定义函数让别人调用。

②同步api从上到下依次执行，前面代码会阻塞后面代码的运行。

同步代码执行区 异步代码执行区 回调函数队列 (所有代码执行都是在同步代码执行去执行的)

事件处理函数，也是异步函数

**解决异步编程回调地狱 Promise 构造函数**

let promise = new Promise((resolve, reject)=>{

if(true){

resolve({name:’aa’})

}else {

reject(‘失败了’)

}

}).then(res=>{ // 获取成功结果 }).catch(err=>{ // 获取失败信息 })

resolve函数：将异步api 返回结果传递出去

reject 函数：将失败信息传递出去

有链式编程功能

**异步函数 （基于Promise 封装）**

异步函数是异步编程语法的终极解决方案，将异步代码写成同步的形式，让代码不再有回调函数嵌套，使代码结构清晰明了

const fn = async ()=>{}

async function fn(){}

async 关键字

普通函数定义前面加上async 关键字，普通函数就变成了异步函数

异步函数默认的返回值是promise 对象

异步函数内部使用return关键字进行结果返回，结果会被包裹在promise对象中。return 关键字替代了resolve 方法

异步函数内部使用throw关键字进行抛出程序异常

调用异步函数再链式调用then方法获取异步函数执行结果

await 关键字

只能出现在异步函数中

后面只能写promise对象。

可以暂停异步函数的执行 等待promise对象返回结果后再向下执行函数

Node.js 全局对象global

js中全局对象是window，node全局对象是global

node中全局对象下有以下方法，可以在任何地方使用，global可以省略

console.log() 在控制台中输出

setTimeout() 设置超时定时器

clearTimeout() 清除超时定时器

setInterval() 设置间歇定时器

clearInterval() 清除间歇定时器

4.服务器端开发

**网站组成**

网站应用程序主要分为两大部分：客户端和服务器端

客户端：在浏览器中运行的部分，就是用户看到并与之交互的界面程序。使用html、css、javascript构建

服务器端：在服务器中运行的部分，负责存储数据和处理应用逻辑。

网站：就是客户端与服务器基于请求与响应模型的一种应用结构。

**node 网站服务器**

能够提供网站访问服务的机器就是网站服务器，它能够接受客户端的请求，能够对请求做出响应。

访问网站：通过IP 地址查找服务器

Ip 地址：互联网中设备的唯一标识，代表互联网协议地址。

域名 ：所谓域名就是上网所使用的网址。域名和IP是对应的关系

访问时，最终会将域名转换为ip才能访问到指定的网站服务器

服务器可以有多个服务，为了区分不同的服务，有了端口的概念。

5.数据库及环境搭建

动态网站中的数据都是存储在数据库中的

数据库可以用来持久存储客户端通过表单收集的用户信息

数据库软件本身可以对数据进行高效的管理

**数据库**

即存储数据的仓库，可以将数据进行有序的分门别类的存储。它是独立于语言之外的软件，可以通过API去操作它

node.js 通过数据库提供的API 访问数据库。Mongo DB 数据库开放的API 是js。

需下载数据库mongo db软件和客户端软件compass。Compass 客户端允许通过图形界面的方式操作数据库。

database 数据库， 库

collection 集合 表

document 文档 一行数据

field 字段

**mongoose 第三方包**

node.js 操作mongodb 数据库需要依赖第三方包 mongoose

启动MongoDB 在命令行工具中 net start mongodb

关闭net stop mongodb

数据库连接：使用mongoose 提供的connect 方法即可连接数据库

创建数据库：在mongodb中不需要显示创建数据库，如果正在使用的数据库不存在，mongodb会自动创建

创建集合 ：①创建集合规则 ②创建集合，创建mongoose.Schema 构造函数的实例即可创建集合

创建文档 ：创建文档实际上就是想集合中插入数据。分两步：①创建集合实例②调用实例对象下的save方法将数据保存到数据库中。

mongodb 数据库导入数据：在命令行工具中执行

mongoimport -d 数据库名 -c 集合名称 --file 要导入的数据文件

找到mongodb数据库的安装目录，将安装目录下的bin目录放置在系统环境变量中，才能执行这个命令

查询文档：find()

大于 $gt 小于 $lt 包含 $in 选择要查询的字段 select(‘name age’)

排序 find() 方法后面用 sort(‘age’)进行排序

分页 skip 跳过多少条数据 limit 限制查询数据数量

删除文档：finedOneAndDelete({}) deleteMany({})

更新文档：updateOne()

mongoose 验证：在创建集合规则时，可以设置当前字段的验证规则，验证失败则输入插入失败 required：true 必传字段

集合关联：查询集合关联 使用populate 方法

案例：用户信息增删改查

第一步：搭建网站服务器，实现客户端与服务器端的通信

第二步：链接数据库，创建用户集合，向集合中插入文档

第三步：…

**模板引擎** 是第三方模块 art-template

让代码以更加友好的方式拼接字符串，使项目代码更加清晰、更加易于维护

①下载模板引擎

②引入模板引擎

③告诉模板引擎要拼接的数据和模板在哪 const html = template(‘模板路径’, 数据); 模板的后缀是.art

语法：标准语法和原始语法

输出 标准语法 {{ 数据 }} 原始语法 <%= %>

原文输出：如果数据中携带html标签，默认模板引擎不会解析标签，会将其转义后输出

标准语法 {{ @数据 }} 原始语法 <% -数据 %>

**条件判断**

在模板中可以根据条件来决定显示哪块html代码

标准语法

{{ if 条件 }} … {{/if}}

{{if v2}}…{{else if v2}} … {{/if}}

原始语法

<% if(value) {%>…<%}%>

<% if(v1){ %>…<%} else if(v2){%>…<%}%>

**循环**

标准语法：

{{each 数据}}

{{$index}} {{$item}}

{{/each}}

原始语法：<%for(){%> <%}%>

**子模板**

使用子模板可以将网站公共区块(头部、底部)抽离到单独的文件中，通过include 关键字

标准语法：{{include ‘子模板路径’}}

原始语法：<% include(‘子模板路径’) %>

**模板继承**

使用模板继承可以将网站html骨架抽离到单独的文件中，其他页面模板可以继承骨架文件。

骨架模板可以预留出一些位置，供子模板填充

①填充css内容

②填充js内容

③填充页面主体内容

通过 block 关键字 预留位置， 通过名字来区分不同填充

父模板中：{{block ‘head’}}{{/block}} 预留位置

{{ extend ‘.’ }} 继承模板

**模板配置**

向模板中导入变量 template.defaults.imports.变量名 = 变量值

例如：处理时间 模块 dateformat

设置模板根目录：template.defaults.root = path.join(\_dirname,’views’);

设置模板默认后缀：template.defaults.extname = ‘.art’;

案例：学生档案管理

目标：模板引擎应用，强化node.js 项目制作流程

知识点：http请求响应、数据库、模板引擎、静态资源访问

第三方模块router 实现路由

获取路由对象

调用路由对象提供的方法创建路由

启用路由，使路由生效

第三方模块 serve-static

功能：实现静态资源访问服务

引入模块，获取创建静态资源服务功能的服务

调用方法创建静态资源服务并指定静态资源服务目录

启用静态资源服务

6.Express 框架

是一个基于Node平台的web应用开发框架，它提供了一系列的强大特性，创建各种web应用。npm install express 下载

提供了简洁的路由定义方式

对获取http请求参数进行了简化

对模板引擎的支持程度高，方便渲染动态html页面

提供了中间件机制有效控制http请求

拥有大量第三方中间件对功能进行扩展

**send() 方法**

内部会检测响应内容的类型

会自动设置http状态码

会帮助我们自动设置响应的内容类型及编码

**中间件**

就是一堆方法，可以接收客户端发来的请求、可以对请求作出响应，也可以将请求继续交给下一个中间件继续处理。

组成：中间件方法以及请求处理方法

中间价由express 提供，负责拦截请求，请求处理函数由开发人员提供，负责处理请求。

可以针对一个请求设置多个中间件，对同一个请求多次处理

默认情况下，请求从上到下依次匹配中间件，一旦匹配成功，终止匹配。

可以调用next方法将请求的控制权交给下一个中间件，直到遇到结束请求的中间件。

**app.use中间件用法**

app.use 匹配所有的请求方式，可以直接传入请求处理函数，代表接收所有的请求。

app.use 第一个参数也可以传入请求地址，代表不论什么请求方式，只要是这个请求地址就接收这个请求。

**中间件应用**

路由保护，客户端在访问需要登录的页面时，可先试用中间件判断用户登录状态。用户如果未登录，则拦截请求，直接响应，禁止用户进入需要登录的页面。

网站维护公告，在所有路由的最上方定义接收所有请求的中间件，直接为客户端作出响应，网站正在维护中。

自定义404页面 res.status(404) 为客户端设置状态码

**错误处理中间件**

在程序执行的过程中，不可避免的会出现一些无法预料的错误，比如文件读取失败，数据库连接失败。

错误处理中间件是一个集中处理错误的地方。

只能处理同步异常，异步错误

异步时：

当程序出现错误时，调用next()方法，并且将错我信息通过参数的形式传递给next()方法，即可触发错误处理中间件。

**捕获错误**

在node.js中，异步api的错误信息都是通过回调函数获取的，支持promise对象的异步api发生错误可以通过catch方法捕获。异步函数执行结果发生错误要如何捕获呢？

try catch 可以捕获异步函数以及其他同步代码在执行过程中发生的错误，但是不能捕获其他类型的api发生的错误。

**构建模块化路由**

**get参数获取**

express 框架中使用req.query 即可获取get参数，框架内部会将get参数转换为对象并返回

**post参数获取**

express 框架中接收post请求参数需要借助第三方包 body-parser;

extended:false 方法内部使用querystring 模块处理请求参数的格式

extended:true 方法内部使用第三方模块qs处理请求参数的格式

**express 路由参数**

req.params

**静态资源**

通过express 内置的express.static 可以方便地托管静态文件，例如img、css、js文件等。

app.use(express.static(‘public’))

**模板引擎**

为了使art-template 模板引擎能够更好的和express 框架配合，模板引擎官方在员art-template 模板引擎的基础上封装了express-art-template.

使用npm install art-template express-art-template 命令进行安装

app.locals 对象

将变量设置到app.locals对象下面，这个数据在所有的模板中都可以获取到

**十二、Ajax**

传统网站中存在问题

网速慢时，页面加载时间长，用户只能等待

表单提交后，如果一项内容不合格，需要重新填写所有变淡内容

页面跳转，重新加载页面，造成资源浪费，增加用户等待时间。

1.基础

浏览器提供的一套方法，实现页面无刷新更新数据，提高用户浏览网站应用的体验。可局部更新数据

**应用**

页面上加载更多数据

列表数据无刷新分页

表单项离开焦点数据验证

搜索框提示文字下拉列表

**运行环境：**

ajax技术需要运行在网站环境中才能生效，当前课程会使用node创建的服务器作为网站服务器。

**运行原理**： 浏览器 <==> ajax <==>服务器端 请求/响应

Ajax相当于浏览器发送请求与接收响应的代理人，以实现在不影响用户浏览页面的情况下，局部更新页面数据，从而提高用户体验。开发人员可控。

**ajax实现步骤**

创建ajax对象 var xhr = new XMLHttpRequest(); 实例对象即ajax对象

②告诉ajax请求地址以及请求方式 xhr.open(‘get’,’http://www.example,com’); 请求方式和请求地址

③发送请求 xhr.send();

④获取服务器端给客户端的响应数据 xhr.onload = function(){ console.log(xhr.responseText); } 请求有时限，时长不定

在服务器中执行页面才能发送，xhr.responseText 是字符串

**服务器端响应的数据格式**

大多数情况下会以json对象作为响应数据的格式。

在http请求与响应的过程中，无论是请求参数还是响应内容，如果是对象类型，最终都会被转换为对象字符串进行传输。

JSON.parse(); //将json 字符串转换为json对象进行处理。

**请求参数传递**

①get请求方式：

xhr.open(‘get’,’http://www.example.com?name=张三&age=20’);

F12 在控制台下 NetWork，XHR 监听的就是ajax请求

②post请求方式

xhr.setRequestHeader(‘Content-Type’,’application/x-www-form-urlencoded’);请求参数类型 报文头名称和值。固定写法

xhr.send(‘name=zhangsan&age=20’); 参数要放到send中，会将参数放到请求体中

post 请求必须在请求报文中明确设置请求参数的类型。

请求报文：在http请求和响应的过程中传递的数据块就叫做报文，包括要传送的数据和一些附加信息，这些数据和信息要遵守规定好的格式。分为报文头和报文体。

**请求参数的格式**

①application/x-www-form-urlencoded name=zhangsan&age=20

②application/json {name:’zhangsan’, age:’20’}

在请求头中指定Content-Type属性值为application/json，告诉服务器端当前请求参数的格式为json，也需要将json对象转为json字符串传递。 Json.stringify();

注意：get请求不能提交json对象数据格式的，传统网站的表单提交也是不支持json对象数据格式的。

**获取服务器端的响应的另一种方式** （已过时，但还有人用）

ajax状态码：在创建ajax对象，配置ajax对象，发送请求以及接收完服务器端响应数据，这个过程中的每一个步骤都会对应一个数值，这个数值就是ajax状态码。

0：请求未初始化(还没open)

1：请求已建立，但是还没有发送(还没调用send())

2: 请求以调用

3：请求正在处理中，通常响应中已经有部分数据可以用了

4：响应已经完成，可以获取并使用服务器的响应了

xhr.readyState //获取ajax状态码

ajax 提供了一个事件 onreadystatechange事件，当ajax状态码发生变化时将自动触发该事件

IE低版本不支持onload事件。onreadystateChagne事件会被调用多次，需判断状态码

**ajax错误处理**

情况一：网络畅通，服务器端能接受到请求，服务器端返回的结果不是预期的结果

可以判断服务器端返回的状态码，分别进行处理。xhr.status获取http状态码，判断状态

情况二：网络畅通，服务端没有接收到请求，返回404

检查请求地址是否错误

情况三：网络畅通，服务器端能接收到请求，服务器端返回500

服务端运行出错，后台人员处理

情况四：网络中断，请求无法发送到服务端 (f12 newwork 下有个offline 模拟断网)

会触发xhr对象下面的onerror事件，在onerror事件处理函数中对错误进行处理。

ajax 状态码：表示ajax请求的过程状态，ajax对象返回的

http状态码：表示请求的处理结果，是服务器端返回的。

**低版本IE浏览器的缓存问题**

在低版本的IE浏览器中，ajax请求有严重的缓存问题，即在请求地址不发生变化的情况下，只有第一次请求会真正的发送到服务器端，后续的请求都会从浏览器的缓存中获取结果。即是服务器端的数据更新了，客户端依然拿到的是缓存中的旧数据。

解决方案：在请求地址的后面加请求参数，保证每一次请求中的请求参数的值不同。

**同步异步概述**

同步：一行代码执行完成后，才能执行下一句代码，即代码逐行执行。

异步：异步代码需花费时间去执行，程序不会等待异步代码，而是继续执行下方的代码，当后续代码执行完成后再回头看异步代码是否返回结果，如果已返回结果，再调用时间准备好的回调函数处理异步代码执行的结果。

ajax是异步请求

在js中，回调函数也是异步调用函数的一种能。

**ajax封装**：代码冗余，将请求代码封装到函数中，发请求时调用函数即可。

xhr.getresponseHeader(); 获取响应头中的数据类型

Object.assign(defaults, options);

**模板引擎概述**

作用：使用模板引擎提供的模板语法，可以将数据和html拼接起来。art-template 模板 template-web.js

将模板引擎的库文件引入到当前页面

②准备art-template模板，模板语法。

<script type=”text/html” id=tp1>

<h1>{{name}}</h1>

</script>

③告诉模板引擎将那个数据和哪个模板进行拼接

模板id，数据 对象类型 方法的返回值就是拼接好的html字符串

var html = template(‘tpl’,{username:’张三’， age:’20’});

案例：验证邮箱地址唯一性 核心思路：失去焦点事件

案例：搜索框内容自动提示 核心思路：输入事件 oninput，①解决无意义请求。②搜索框无内容，下拉框隐藏

案例：省市区三级联动 核心思路：下拉框 改变事件onchange

2.FormData对象的作用

①模拟html表单，相当于将html表单映射成表单对象，自动将表单对象中的数据拼接成请求参数的格式。

②异步上传二进制文件(如图片文件，视频文件等)。表单一定要有name属性，name作为请求参数

**将html表单转化为formData对象**

var form = document.getElementById(‘form’); form表单

var formData = new FormData(form); // 创建formData实例，即为获取表单实例

提交表单对象 shr.send(formData) 不能用get请求，只有post 方法能实现文件上传

**formData对象的实例方法**

formData.get(‘key’); //获取表单对应属性的值。key 即为name 对应的值

formData.set(‘key’,’value’); //设置表单对应属性的值。 引用场景：对表单数据二次处理。

// 如果对应key 不存在。表示添加属性，设置值

formData.delete(‘key’) // 删除表单对象中属性的值。如确认密码，验证后请求时可以删除。

formData.append(‘key’,’value’) //向表单对象中追加属性值 应用场景：创建formData()时，可以不传dom对象，创建空from对象。追加属性。

注意：set 与 append 区别。在属性名已存在的情况下，set会覆盖已有键名的值，append会保留两个值。

**formData 二进制文件上传**

<input type=”file” id=”file”/> // 默认input file 选择文件也只能选择一个文件

var file = document.getElementById(‘file’);

file.onchange = function(){ // 当用户选择文件的时候

var from = new FormData(); // 创建空formData对象

formData.append(‘attrName’, this.files[0]); // 将用户选择的二进制文件追加到表单对象中

xhr.open(‘post’,’www.example.com’); // 配置ajax对象，请求方式必须为post

xhr.send(formData);

}

**formData文件上传进度展示**

file.onchange = function(){

xhr.upload.onprogress = function(ev){ // 文件上传过程中持续触发onprogress事件

bar.style.width = (ev.loaded/ev.total)\*100 + ‘%’; //当前上传文件大小/文件总大小 再将结果转换为百分比

}

}

upload 上传相关的事件，都在这个对象中。上传文件比较大，不可能一次就上传完成，每次上传都触发一次onprogress事件。

**formData 文件上传图片即时预览**

在我们将图片上传到服务器后，服务器端通常会将图片的地址作为响应数据传递到客户端，客户端可以从响应数据中获取图片地址，然后将图片再显示在页面中。

我们是不能直接访问服务器端硬盘文件地址的。要访问静态资源文件 即项目下的路径。

给img 标签设置src后，加载图片也是需要时间的，加载完成后追加到dom中 img.onload = function();

3.ajax 请求限制

ajax 只能向自己的服务器发送请求。即自己网站的客户端向自己网站的服务器端发送请求。跨网站发送请求是不允许的。

因为ajax有同源政策限制。

**同源**

相同的协议、域名和端口相同，三者相同，属于同源。只要有一个不同，就是不同源。

同源政策的目的：是为了保证用户信息的安全，防止恶意的网站窃取数据。

最初的同源政策指A网站在客户端设置的cookie，b网站是不能访问的。随着互联的发展，同源政策也越来越严格，不同源时，其中有一项规定就是无法向非同源地址发送ajax请求，如果请求，浏览器就会报错。不是发不出去，而是浏览器拒接收响应的数据。

**第一种：使用jsonp解决同源限制问题**

Jsonp 不属于ajax请求，但它可以模拟ajax请求的效果。需要前后端开发人员配合完成

①将不同源的服务器端请求地址写在script 标签的src属性中。

<script src=”www.example.com”></script>

因为浏览器端，并不是所有能发送请求的途径，都受同源政策的影响。如

<script src=”http://cdn.bootcss.com/jquery/3.3.1/jquery.min.js”></script>

②服务器端响应数据必须是一个函数的调用，真正要发送给客户端的数据需要作为函数调用的参数。

③在客户端全局作用域下定义函数fn

function fn(data){}

④在fn函数内部对服务器端返回的数据进行处理

**动态Jsonp 非同源请求**

核心思路：动态添加script 标签，添加请求src 属性指定值。将标签添加append到body。监听请求完成事件onload，完成后删除该标签。

这不是ajax请求，知识模拟

**优化**

客户端需要将函数的名称传递给服务器端。

②将script请求的发送编程动态请求

③封装jsonp函数，方便请求发送。

案例：获取腾讯天气 核心思路：非同源请求天气信息

**第二种：CORS 跨域资源共享**

即跨域资源共享，它允许浏览器向跨域服务器发送ajax请求，克服了ajax只能同源使用的限制。ajax不用调整，在服务器端设置。

cors跨域请求过程：

浏览器端请求，如果浏览器检测到请求是跨域的，会自动在请求头加上origin字段，该字段的值就是当前请求的域名信息。如<http://localhost:3000>。

服务器端会根据这个字段的值来决定是否同意这次请求。不管是否同意这次请求，服务器端都会给客户端一个正常的http响应(响应头 Access-Control-Access-Origin)。

浏览器判断服务器是否同意本次请求。如果同意，服务器会在响应头加上 Access-Control-Access-Origin。如果不同意，就不会加这个字段。

Access-Control-Access-Origin:’http://localthost:3000’ // 允许这个客户端访问请求

Access-Control-Access-Origin:’\*’ // 允许所有客户端访问请求

服务端需设置：

①允许哪些客户端访问我 Access-Control-Access-Origin:’\*’ 允许所有客户端

②允许客户端使用哪些请求方法访问我 Access-Control-Allow-Methods, ‘get,post’ get 请求与post请求

只需要在服务器端设置两个响应头就可以了。

可以在所有拦截请求前进行拦截设置。

**第二种：访问非同源数据 服务器端解决方案**

同源策略是浏览器给与ajax技术的限制，服务器端是不存在同源政策限制的。

----------------------------------------------------------------------------------------------------

A浏览器端 ----请求-----> A服务器端 ----请求----> B服务器

<----响应----- <----响应----

----------------------------------------------------------------------------------------------------

可以通过这种方式绕过浏览器同源限制。

4.cookie

客户器端与服务器端 请求/响应 的模式进行交互。使用的协议是http协议（无状态）。

cookie 是客户端与服务器端交互身份识别的一种技术。

客户端访问服务器，服务器响应的同时给客户端发一个身份证明的cookie，客户端再次请求时，会自动携带这个身份证明。从而建立客户端与服务器端的长久联系。

一旦涉及到跨域，cookie 就不会随着请求被发送到服务器端了。是处于安全的考虑。

**withCredentials 属性**

在ajax 发送跨域请求时，默认情况下不会在请求中携带cookie信息。

withCredentials：指定在涉及到跨域请求时，是否携带cookie信息，默认为false.

同时，需要在服务器端设置 Access-Control-Allow-Credentials：true允许客户端发送请求时携带cookie。

案例：实现跨域登录 核心思路： 发送跨域请求时，设置ajax withCredentials为true。同时后台设置Access-Control-Allow-Credentials

5. $.ajax() 方法概述 与 serialize方法

作用：发送ajax请求

content-Type: 指定参数的格式类型，字符串 json 等

beforeSend:function(){} 请求被发送之前调用 可做发送前验证，不发送return false;

**serialize 方法**

作用：将表单中的数据自动拼接成字符串类型的参数

var params = $(‘#form’).serialize();

将表单中用户输入的内容转换为对象类型

function serializeObject(obj){

var params = obj.serializeArray();

var result = {};

$.each(params, function(index, value){

result[value.name] = value.value;

})

return result;

}

**$.ajax() 发送jsonp 跨域请求**

$.ajax({

url:’’,

dateType:’jsonp’, // 指定当前发送jsonp 请求 即跨域请求

jsonp:’cb’, // 修改callback参数名称，一般不设置，用人家默认生成的

jsoncallback:’fnName’, // 指定函数名称，，， 一般不设置，用人家默认生成的

success:function(res){}

})

$.get() 、$.post() 方法

作用：$.get 方法用于发送get请求 $.post 方法用户发送post请求

$.get(‘http://www.example.com’,{name:’aa’, age:30}, function(res){}) 写法一样

案例：TodoList 一天任务列表 核心思路：将数据放到数组，通过数组更新页面。数据驱动视图思想

删除时，事件添加给ul，其中li触发。事件委托方式

ulObj.on(‘click’,’.desctroy’, function(){

var id = $(this).arrt(‘data-id’);

var index = arr.findIndex(item=>item.id==id);

arr.splice(index,1);

})

ulObj.on(‘change’,’.toggle’,function(){

const stats = $(this).is(‘:checked’);

const id = $(this).siblings(‘buttong’).attr(‘data-id’);

var task = arr.find(item=>item.id==id);

})

6. jQuery中全局事件

只要页面中有ajax请求被调用，对应的全局事件就会被触发

.ajaxStart() // 当请求开始发送时触发

.ajaxComplete() // 当请求完成时触发

jQuery中规定，全局事件一定要绑定到document上

$(documnet).on(‘ajaxStart’, function(){

NProgress.start();

})

$(documnet).on(‘ajaxComplete’, function(){

NProgress.done();

})

**NProgress 进度条插件**

官宣：纳米级进度条，使用逼真的涓流动画来告诉用户正在发生的事情

<link rel=’stylesheet’ href=’nprogress.css’/>

<script src=’nprogress.js’></script>

NProgress.start(); //进度条开始运动

NProgress.done(); //进度条结束运动

7. Restful 风格的API

是一套关于设计请求的规范。

Get 获取数据

Post 添加数据

Put 更新数据

Delete 删除数据

8. XML

可扩展标记语言，它的作用是传输和存储数据。

Html 是展示数据

XML DOM 即xml文档对象模型，是w3c组织定义的一套操作xml文档对象的API。浏览器会将xml文档解析成文档对象模型。

xmlDom.getElementsByTagName(‘title’)[0].innerHTML;

几乎不用了，我没用过

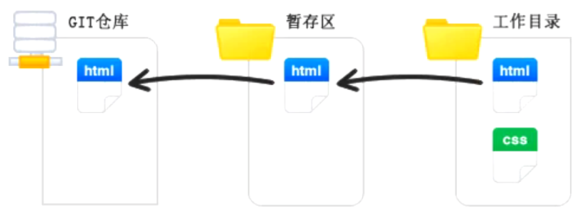
**十三、Git**

是一个软件，版本管理控制系统，可以在任何时间点，将文档的状态作为更新记录保存起来，也可以在任何时间点，将更新记录恢复过来。

1. 基本流程

目的为了管理项目的更新迭代，管理项目的版本。

操作窗口其实是命令行程序，学习git除了学习概念，就是学习它提供的命令。



**git仓库 ：**由git创建的，存放项目状态的文件夹

**工作目录：**被git管理的项目文件夹，被管理的项目文件，写代码的地方。

**暂存区 ：**临时存放被修改文件

在git 中每次存储项目状态时，只存储那些被修改的文件，git 怎么知道被修改过的？

开发者每次将修改过的文件从工作目录提交到暂存区，这就是修改过的文件。再将暂存区中的文件提交到git仓库

2. 全局用户配置

git –verson // 查看git版本

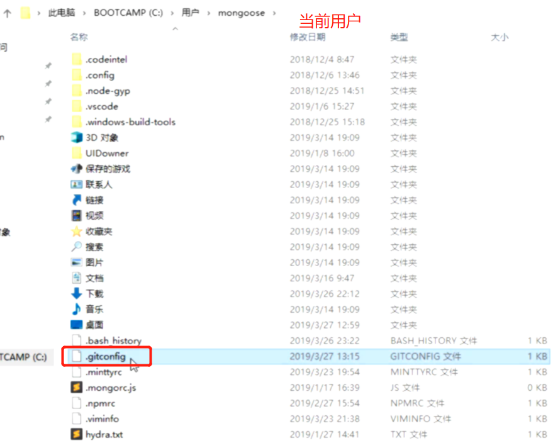
**配置用户：**使用git前，需要告诉git你是谁，向git仓库中提交时需要用到。

git config --global user.name 姓名 // 配置提交人的姓名

git config --global user.email 邮箱名称 // 配置提交人的邮箱

git config –list // 查看git配置信息

注意：如果要对配置信息进行修改，重复上述命令即可。配置只需要执行一次。配置的信息保存在如下位置。



3. 命令

git init // 初始化git仓库

git status // 查看文件状态

git add 文件列表 // 将文件添加到暂存区，跟踪文件

git commit -m 提交信息 // 向仓库中提交代码，提交信息是备注

git log // 查看提交记录

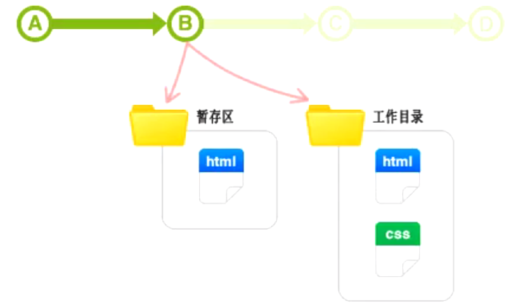
注释：向git仓库中提交项目状态

**撤销**

git checkout 文件名 //用暂存区中的文件覆盖工作目录中的文件

git rm --cached 文件名 //将文件从暂存区中删除

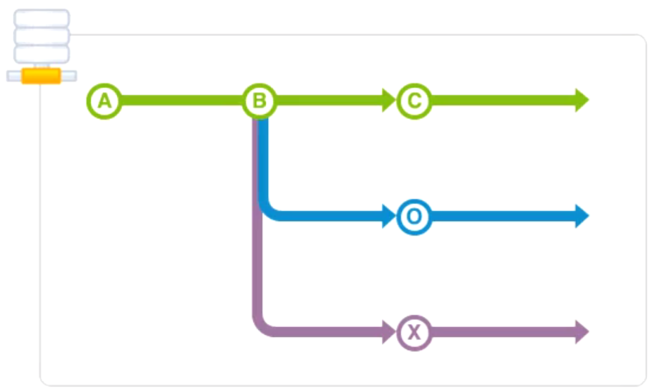
git rest --hard commitID //将git仓库中指定的更新记录恢复出来，并且覆盖暂存区和工作目录



4. 分支

为了便于理解，可以暂时认为分支就是当前工作目录中代码的一份副本。

使用分支，可以让我们从开发主线上分离出来，以免影响开发主线。



**主分支(master)**：第一次向git仓库中提交更新记录时自动产生的一个分支。



注意：一般不在主分支上开发，是为了保证代码的稳定性。

**开发分支(feature)：**作为开发具体功能的分支，基于开发分支创建。开发完成并测试好后，合并到主分支。

允许多件事情同时开发，互不影响。

5. 分支命令

git branch // 查看分支

git branch 分支名称 // 基于所在分支创建分支

git checkout 分支名称 // 切换到某分支

git merge 来源分支名称 // 合并分支

git branch -d 分支名称 // 删除分支

分支与分支之间在开发过程中应该是没有联系的。

**切换注意**：切换分支之前，当前分支中的文件提交到git仓库中，保证当前分支的工作区是干净的状态。否则会出现问题。

**合并注意**：将开发分支合并到主分支。开发分支被合并到主分支上，应该先切换到主分支上，要站在主分支的角度上合并开发分支。开发分支合并到主分支后，开发分支依然存在，可以回到开发分支继续开发。

**删除注意**：如果分支没有被合并，默认分支是不允许删除的。这是git系统对分支的保护机制。防止误删除。

如果确实不想要这个分支。可以强制删除分支，将小d 换成大D，即强删除

如果当前处在要删除的分支，也是不允许删除这个分支的，需要切换到其它分支再删除。

6. 暂时保存更改

git stash // 存储临时改动

git stash pop // 恢复改动

上面讲过，切换分支前需要先提交分支上的工作，否则会出现问题。但实际开发中常出现其它分支的突发情况去处理。就需要临时保存更改操作了。

git中提供了储藏功能，可以把它想象成一个剪贴板，它可以提取分支上所有的改动，并将改动保存起来。你就可以得到一个干净的工作副本，也就是说一个不存在任何改动的工作目录，接下来就可以随意切换分支了。

其它的分支调整完后，再切换回当前分支，可以将刚才的储藏恢复回来，继续工作。

在git中，可以暂时提取分支上所有的改动并储存，让开发人员得到一个干净的工作副本，临时转向其它工作。

使用场景：分支临时切换。

恢复后，剪贴板上的内容就没有了

注意：git提供的储存功能是独立于分支的，也就是说在其它分支上也可以执行这个命令。其它分支上执行这个命令会将内容恢复到其它分支上。所以一定要注意。

7. github

在版本控制系统中，大约90%的操作都是在本地仓库中进行的：暂存，提交，查看状态或者历史记录等等。除此之外，如果仅仅只有你一个人在这个项目里工作，你永远没有机会需要设置一个远程仓库。

需要和开发团队共享数据时，需要设置一个远程仓库。可以把它想象成一个“文件管理服务器”，利用这个服务器可以与开发团队里的其他成员进行数据交换。

8. 多人协作开发流程

git push 远程仓库地址 分支名称 //将本地仓库推送到远程仓库

git remote add origin 远程仓库地址 // 为远程仓库地址起个别名origin

git push origin master // 使用别名推送

git push -u origin master // git 帮助记住远程地址和分支名称，下次提交直接git push

A在自己的计算机中创建本地仓库

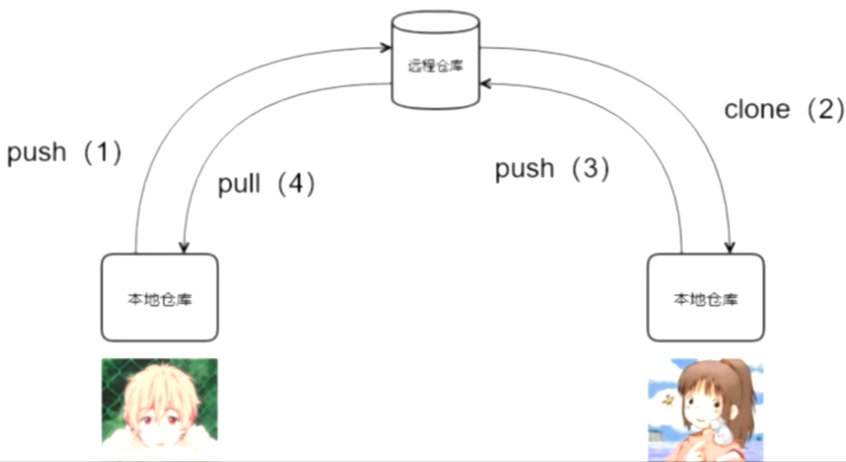
A在github中创建远程仓库

A将本地仓库推送到远程仓库

B克隆远程仓库到本地进行开发

B将本地仓库中开发的内容推送到远程仓库

A将远程仓库中的最新内容拉取到本地



①A工作目录创建好后，打开git命令行工具 git init 初始化一个仓库

②文件下创建一个test.html 文件然后添加到暂存区 git add test.html

③提交到本地仓库 git commit -m 这是第一次提交

④在github 个人中心页面，点击创建项目()即创建一个远程仓库。创建好后会生成一个远程仓库地址。

⑤git push 远程仓库地址 分支名称 //将本地仓库推送到远程仓库 刷新个人中心页面就可以看到了。

推送比较负载，远程仓库地址太长。可以起个别名

⑥git remote add origin 远程仓库地址 // 为远程仓库地址起个别名origin

再次推送时，用git push origin master 就可以了，把本地master 分支推送到远程仓库

注意：第二次提交时，没有输入账号密码，这不是git提供的功能，是win10 系统提供的。保存在【设置】-【控制面板】-大图标显示找到【凭据管理器】-【windows凭据】中

⑦过还嫌提交麻烦，再次修改后提交到本地仓库中，

执行git push -u origin master 。加上-u后git会帮助记住远程仓库地址和分支名称，下次提交时直接写git push 就可以了。

⑧git clone 远程仓库地址 // B克隆远程仓库

⑨git push origin master // B向远程仓库推送代码

程序员B没有权限向远程仓库中提交代码，因为没有权限。需要程序员A邀请B称为项目开发者。

A在github 上【Setting】-【Collaborators】添加合作者。将B的github 账号添加进去(Add collaborator)。A将【Copy invite link】邀请链接发送给B。

B需要在登录github 账号的情况下访问邀请链接，并且接受邀请链接。B就可以向远程仓库中推送了。

⑩然后A再拉取远程仓库的代码

git pull 远程仓库地址 分支名称 //拉取远程仓库中最新的版本

注意：git clone 只需要用一次就可以了，git pull 会与本地仓库代码做比较

如果远程仓库中的版本高于本地仓库，是不允许提交的，需先拉取远程仓库内容。

**解决冲突**

多人开发一个项目是，如果两个人修改了同一个文件的同一个地方，就会发生冲突。冲突需认为解决。（第二个人需要将冲突解决掉，才能提交到远程仓库中）。

B 先将A提交的拉取到本地

修改冲突，再次提交

**跨团队协作**：也支持非合作者提交，但是需要原仓库作者A审核，合并。C【Fork】一下，这个项目地址就是自己的了，克隆项目到本地，修改提交。C点击【Pull reqeusts】 创建一个新的request 请求，跟原仓库作者对话，录入主题，备注发送给原作者。

A 的【Pull requests】 有一条数据，就是C给A的，A还可以与C沟通，在【Files changed】下有C的调整。

A 点【Conversation】-【Merge pull request】合并C提交的代码。

总结：对远程仓库的操作，是使用https协议的仓库地址，这种协议要实现身份验证，用户必须提供用户名和密码。正常个情况下每次向远程仓库中提供内容，都需要提供用户名和密码，win 10 系统帮助我们记住了。如果你的系统不具备这种功能，哪就头疼了。

9. ssh 免登录

另一种仓库通信协议ssh。这种协议在身份认证时是不需要账户和密码的。它可以免登录操作。

身份认证通过秘钥实现，是成对出现的，通过公钥的私钥的配对进行验证。

生成秘钥：ssh-keygen //git 命令行工具中



上面私钥，下面公钥，私钥必须保留在开发者的电脑中，公钥需放在github 服务器中，Github 头像，下拉中【Settings】-【SSH and GPG keys】key 中。使用ssh 对仓库操作，头像下拉中【Your repositories】

10. git 忽略清单

将不需要被git 管理的文件名字添加到文件中，在执行git命令的时候，git就会忽略这些文件。

git忽略清单文件名称：.gitignore

跟目录下文件夹

node\_modules

test

index.html

等

将工作目录中的文件全部添加到暂存区：git add . // 点代表所有文件

**为仓库添加详细的说明**

readme.md

**二十、扩展**

1. 屏幕拾色器

2. 浏览器开发工具

浏览器中点击F12

3. gsk项目图片列表下移

解决：原来怀疑是文档流塌陷，但是分析，文档流塌陷是当前图片脱离文档流上移

真正原因是块元素没有设置固定高度，一个元素宽度增高，将下面的顶下去了

4. iphone 手机视频列表都显示在左侧，宽度很小

解决：原来怀疑是iphone手机适配原因

真正原因是父标签没有设置宽度100%，iphone 手机有可能元素没有继承父元素样式100%。设置宽度后就好了

5. 画图工具

解决：画图工具可以改变图片背景色、合并图片等操作。

6. iphone input 录入框录入不了文字

解决：