**一. HTML**

**1.基础**

1.1<!DOCTYPE html> 这句声明, 是告诉浏览器,请使用HTML5的标准来解析这个页面

DTD 文档类型的定义

1.2 双标签

<meta>内容描述 <h><p> <title> <strong>加粗 <b>加粗 <em>倾斜 <i>倾斜

<sub>下角标 <sup>上角标

单标签: <br/> 换行 <hr/>水平线

<small></small>

<ins></ins> 插入字

<del></del> 删除字

<q></q> 短引用, 效果该段文字 会被加上引号

*<p>WWF 的目标是：<q>构建人与自然和谐共存的世界。</q></p>*

<blockquote></blockquote>长引用, 效果整段文字会被缩进

<abbr></abbr>用于缩略词

*<p><abbr title="World Health Organization">WHO</abbr> 成立于1948 年</p>*

<dfn></dfn> 用于定义 与abbr类似

*<p><dfn title="World Health Organization">WHO</dfn> 成立于 1948 年。</p>*

<address></address>用于联系信息

*<address>*

*Written by Donald Duck.<br>*

*Visit us at:<br>*

*Example.com<br>*

*Box 564, Disneyland<br>*

*USA*

*</address>*

<cite></cite>用于著作标题

*<p><cite>The Scream</cite> by Edward Munch. Painted in 1893.</p>*

<bdo></bdo>双向重写

*<bdo dir="rtl">This text will be written from right to left</bdo>*

<kbd></kbd>键盘输入

*<p><kbd>File | Open...</kbd></p>*

<samp></samp>计算机输出示例

*<samp>*

*demo.example.com login: Apr 12 09:10:17*

*Linux 2.6.10-grsec+gg3+e+fhs6b+nfs+gr0501+++p3+c4a+gr2b-reslog-v6.189*

*</samp>*

<code></code>编程代码示例, 元素不保留多余的空格和折行

*<code>*

*var person = { firstName:"Bill", lastName:"Gates", age:50, eyeColor:"blue" }*

*</code>*

<pre></pre>其中包围代码

*<code>*

*<pre>*

*var person = {*

*firstName:"Bill",*

*lastName:"Gates",*

*age:50,*

*eyeColor:"blue"*

*}*

*</pre>*

*</code>*

<var></var>定义数学变量

<p><var>E = m c<sup>2</sup></var></p>

<!--注释标签-->

<!--[if IE 8]> .... <![endif]--> 条件注释

<base></base>标签

<meta></meta>标签

Html实体 ©&..... <https://www.w3school.com.cn/tags/html_ref_entities.html>

Html符号 <https://www.w3school.com.cn/tags/html_ref_symbols.html>

Html编码 <https://www.w3school.com.cn/tags/html_ref_urlencode.html>

Html语言代码 <https://www.w3school.com.cn/tags/html_ref_language_codes.asp>



1.3 图片

图片格式: jpg gif png

<img src=”图像地址”alt=”加载失败显示的东西” title=”鼠标悬停提示文字”

width=”” height=””/>

1.4链接

<a target=”\_blank”或者 target=”\_self”></a>

<a name=”mao3”></a>

<a href=”#mao3”></a>

1.5列表

无序列表<ul type=”crile或者dics或者square”> 若干li标签 </ul>

有序列表<ol type=”1或者A/a或者I/i”> 若干li标签 </ol>

自定义列表<dl>

<dt>列表标题</dt>

<dd>列表内容1</dd>

<dd>列表内容2</dd>

<dd>列表内容3</dd>

<dd>列表内容4</dd>

</dl>

列表的css样式: list-style: none url(图) inside

list-style-type: none

list-style-image: url()

list-style-position: outside或者 inside

1.6表格

<table>

<tfoot> 定义表格的页脚 </tfoot>

<thead>

<tr>

<th>列的名称</th>

<th>列的名称</th>

</tr>

</thead>

<tbody>

<tr>

<td>列的数据</td>

<td>列的数据</td>

</tr>

</tbody>

</table>

<caption></caption>定义表格的标题

<col></col>定义用于表格列的属性

<colgroup>定义表格列的组

<tbody></tbody>定义表格的主体

rowspan 合行 colspan合列

align 水平对齐: left center right

valign垂直对齐: top middle bottom baseline

border 边框粗细

cellpadding 内容到边框

cellspacing 单元格与单元格之间距离

table的css样式:

caption-side: top bottom left right 标题的位置

border-spacing: 单元格与单元格之间的距离 与cellspacing一样

border-collapse: separate边框分开 collapse边框合并

empty-cells: 空边框是否隐藏 show hidden

table-layout: auto fixed

1.7表单

<from name=”register” action=”/from”要提交到的网址 method=”post”>

文本框<input type=”text”value=”要提交的内容”placeholder=”默认显示”/>

密码框<input type=”password”/>

单选按钮<input type=”radio”name=”r1”id=””value=”money”/>

单选按钮<input type=”radio”name=”r1”id=””value=”world”/>

单选按钮<input type=”radio”name=”r1”id=””value=”China”/>

单选按钮name要相同

复选框 <input type=”checkbox”name=”c1” value=”m”disable禁用/>

复选框 <input type=”checkbox”name=”c2”id=””value=”c”/>

复选框 <input type=”checkbox”name=”c3”id=””value=”h”/>

按钮 <input type=”button”name=””id=””value=”按钮”active=””/>

下拉菜单<select>

<option>北京</option>

<option>上海</option>

<option>深圳</option>

</select>

文本域<textarea cols=”20”rows=”5”>多行文本 </textarea>

日期框<input type=”date”value=””/>

时间框<input type=”datatime” value=””/>

颜色框<input type=”color”value=””/>

邮件框<input type=”email” value=””/>

电话框<input type=”tel” value=””/>

数字框<input type=”number” value=””/>

<fieldset>相当于一个分组的方框

<legend>基本信息</legend> 必须放在fieldset的第一行,它是方框的标题

</fieldset>

上传文件的域<input type=”file” value=”选择文件”/>

上传多个文件<input type=”file” value=”选择文件”multiple=”multiple”/>

图像域 <input type=”image”width=””height=””

border=”solid, 2,#000”src=””/>

聚焦鼠标<label for=”me”>姓名:</label>

<input type=”text”id=”me”placeholder=”请输入姓名”/>

重置按钮<input type=”reset”name=””id=””value=”重置1”/>

提交按钮<input type=”submit”name=””id=””value=”提交1”/>

</from>

放在表单外边的按钮<button>按钮22</button>

表单属性值=

name = “值” 设定表单的名称

action = “值”提交表单URL

method = ”get/post” 提交方式

enctype =“值”规定发表单之前 先编码

target = “\_black \_self \_parent \_top” 何处打开表单

H5新增的属性:

Autocomplete=“on off” 是否自动验证表单

novalidate=”novalidate” 不验证表单

**2. H5新特性**

2.1新增的标签

html新标签 <https://www.w3school.com.cn/tags/index.asp>

结构性标签: <article> <aside> <footer> <header> <nav> <section>

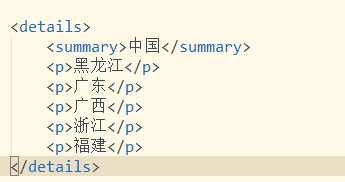
非结构性标签:

<audio> <video> <canvas> <command>

<datalist>:



<details>:



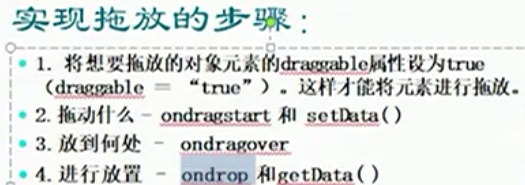
<source> <time> <mark>: 加强语气着重显示

<progress>:<progress max="100" value="10"></progress>

2.2表单的新特性



2.3拖拽





**二. CSS**

**1.css样式** cascading style sheets 层叠样式表

1.1web标准

结构层: xhtml html

表现层: css

行为层: DOM JS

1.2样式分类

内部样式: <style type=”text/css”>

css语句

</style>

外部样式: <link rel=””type=””href=””/> 或 [@import url(“”)]

内联样式: 写在行内 <标签 style=””> </标签>

注: 内联样式优先级最高. 内部样式与外部样式后写的优先级高

多个内部样式后写的有优先级高. 多个外部样式后写的有优先级高.

1.3标签分类

块级元素:

独占一行, 可设置宽高,不设置宽度 默认与父级元素同高

div,h,p,from, dt,dl,dd,ol,ul li, hr, tr,td,table 等

内联元素: 也叫行内元素

不会独占一行, 不能设置宽高

a, span, em, strone

内联块元素:

不会独占一行, 可以设置宽高,

img input select textarea

元素类型转换:

display: block块级 inline内联 inline-block 行内块元素

1.4选择器

标签选择器: div{ }

id选择器: #id{ }

class选择器: .class{ }

\* 通配符选择器: \*{ }

交集选择器: #id.class{ } #iddiv{ } 同时具有两个属性

(群组)并集选择器: #id,div,.class{}

后代选择器: #id div { } 用空格分开,儿孙都变

子选择器: #id > div{ } 儿 一级变, 孙代不变

伪类选择器: a:link{ }初始

a:visited{ }访问过

a:hover{ }悬停

a:active{ }按下

新增的选择器: 属性选择器, 结构伪类选择器, UI选择器,

其他选择器(兄弟元素, 否定伪类, 目标伪类选择器)

选择器权重:

标签: 0001

id:0100

class:0010

并集:0000

子选择器:0000

伪类选择器:0010

伪元素选择器:0010

属性选择器:0010

继承样式:0000

行内样式:1000

值相同后定义的优先级高

1.5背景

background-

-image: url( )

-corlor:

-repeat: no-repeat repeat repeat-x repeat-y

-attachment: scroll 背景固定 fixed 背景滚动

-position: 背景位置

css新增属性

background-size: cover按窗口大小铺满溢出的部分隐藏 100%100%自由拉伸

display:none 隐藏不占位置

visibility: hidder 隐藏占位

border\_radius: 100px 半径为100的圆

1.6文本的css

font

font-size: 大小

font-family: 字体

font-weight: 900粗度 normal =400常规

font-style: italic倾斜 font可简写,几个属性写在一起

color: 文字颜色 #fff 或者 pink或者 rgba(1,255,99,0.2)

line-height: 行高

text

text-align: 水平对齐

vertical-align: 垂直居中

text-decoration: under-line下划线, none, over-line上划线,

line-through删除 blink 闪烁

text-indent: 缩进 40px; -40px;往前溢出

letter-spacing:10px 单词与单词之间的间距

word-spacing:20px 字母与字母之间的间距

text-transform: capitalize;每个单词首字母大写

uppercase所有字母大写 lowercase 所有字母小写

text-shadow: 水平10px 垂直20px 模糊距离blur 阴影颜色color

word-break: normal默认的换行 break-all单词内部换行 keep-all 标点符号换行

word-wrap: normal默认, 只允许断字换行 break-word在长单词内部换行

white-space: 空白处理normal nowrap pre 等等

**2.css盒模型**

2.1标准盒模型

margin

|

border

|

padding

|

margin--border--padding--**content**--padding--border--margin

|

padding

|

border

|

margin

2.2盒内容的溢出

overflow

visible: 默认值, 内容过多会溢出

hidden: 隐藏

scroll: 滚动条

auto: 自动按情况加滚动条, 横向溢出横向加滚动条,纵向溢出纵向加滚动条

overflow-x: 横向溢出

overflow-y: 纵向溢出处理方式

text-overflow: 文本溢出

clip: 溢出隐藏

ellipsis:显示省略号 必须和hidden, white-space:nowrap 一起使用

string: 指定的修饰符代替文字

2.3盒模型样式

border: 线型(solid/dashed/dotted/double) 粗细(10px;) 颜色()

border-style: 仅用于线型

padding: 四个值 上 右 下 左

三个值 上 左右 下

两个值 上下 左右

一个值 上下左右都为同一个值

margin: 四个值 上 右 下 左

三个值 上 左右 下

两个值 上下 左右

一个值 上下左右都为同一个值

注:上下两个盒子的距离,等于margin较大的, 左右两个盒子的距离,等于两margin之和

2.4浮动

页面布局的方式: 文档流 浮动流

.浮动元素不在标准文档流中, 所以浮动后面紧跟着的元素占据了浮动元素原先的位置

.如果浮动元素前面的元素没有浮动属性, 则会另起一行贴着此元素下边浮动

.当元素浮动之后,其下方承载文字的元素会占据浮动元素原来的位置,但是其中文字会一

直围绕在浮动元素的周围,而不被浮动元素覆盖.

浮动产生的副作用 216集

背景不能显示: 如对父级设置了背景或者背景图片,而父级不能被撑开,导致背景不能显示

边框不能被撑开:

margin padding不能正确显示, 特别是上下边的margin和padding

清除浮动:

clear: left right both none

其他清除浮动的方式:

1

2

3

4

2.5定位

position属性:

absolute:设置绝对定位的元素,相对于具有”相对定位的父级元素”的偏移

fixed: 固定, 相对于屏幕的位置固定不变, 在文档流里不再占空间,

层级高于普通元素. 若行内元素使用fixed 会自动转为块元素

relative: 相对于自己原先的位置 偏移

static: 默认值,没有定位,使用的文档流

z-index属性:

默认值为0, 值越大越在上边.

2.6浏览器兼容的问题: 218集

2.7CSS文档统筹:

1. **C3新特性**

3.1属性选择器

p[class \*= “tem”] 星等于 查找带有的

p[class ^= “tem”] 尖号等于 查找开头的

p[class $= “tem”] 结尾的

3.2结构性伪类选择器

3.2.1 a:link a:visited a:hover a:active

#box:first-line 第一行文本

#box:first-letter 首字母

before 某些元素之前插入 p:before{ content: ”\*”; }

after 某些元素之后插入 p:after{ content:”!!”; color:red}

3.2.2

root 选择根元素 :root{ background-color: red; }

not 排除某结构下的某个子元素 p:not(.item2){background-color:yellow;} p标签都是黄色 但是除了类名为 itme2的p标签

empty 指定内容为空白的元素

target 当前的



3.2.3

first-child #box1 p:first-child{ backgroundcolor: red; }

last-child 父元素下最后一个子元素

nth-child(3) 父元素指定某个子元素

nth-last-child(2) 父元素下倒数某个元素

first-of-type 在其父元素下 找第一个指定的元素

last-of-type 在其父元素下 找最后一个指定的元素

nth-of-type(3) 在其父元素下 找指定元素的第n个

nth-last-of-type(2) 在其父元素下 找倒数第n个

only-child 在其父元素下 仅有一个并且为指定的元素

nth-of-type(odd) 奇数

nth-of-type(even) 偶数

3.3 UI状态的伪类选择器

#box1 input:focus{ background:red; } 聚焦的时候才变红色

#box1 input:disabled{ background:red; } 禁用时候才显示的颜色

#box1 input:enabled{ background:red; } 可用时候才显示的颜色

#box1 input:readonly{ background:red; } 可读时候才显示的颜色

#box1 input:readwrite{ background:red; } 可读可写时候才显示的颜色

#box1 input:checked + label{ background:red; } 选中状态下 后边才显示的颜色

#box1 input:indeterminate

.itme1 ~ p 其后所有

.itme1 + p 其后的一个

3.4盒子

3.4.1盒子阴影 box-shadow: 水平 竖直 模糊半径 展开半径 颜色;

3.4.2 border-radius: 一个值 两个值 三个值 四个值

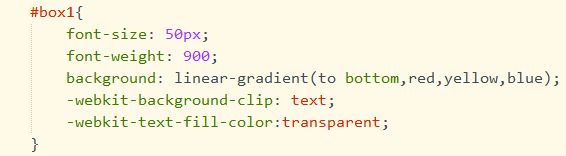
3.4.3 background-size: 背景图片的大小

3.5线性渐变

3.5.1 background: linear-gradient(to right, red,yellow,bule)渐变方向,颜色,颜色,颜色

3.5.2 background: linear-gradient(45deg, red,yellow,bule)

3.5.3字渐变



background-clip: 规定背景的绘制区域

border-box 背景被裁剪到边框盒

padding-box背景被裁剪到内容边框

content-box 背景被裁剪到内容框

text 背景绘制文字部分

3.5.4径向渐变

background: radial-gradient(circle,red,yellow) 圆形渐变

background: radial-gradient(ellipse,red,yellow) 椭圆渐变

background: radial-gradient(circle,50px,red,yellow) 设置大小

background: radial-gradient(80%,20%,red,yellow) 横轴渐变80% 纵轴渐变20%

background: radial-gradient(at left, 50px, red, yellow) 设置大小

background: radial-gradient(形状,大小, at位置,颜色, 颜色2, 颜色3, 颜色4)

3.5.5重复渐变

background: repeating-radial-gradient( red 5%, yellow 10%, bule 30%)

**3.6响应式布局 (弹性盒模型)**

3.6.1 针对容器的属性

display: flex:

flex-direction: 排列方向 row column row-reverse逆序 column-reverse逆序

flex-wrap: nowrap(默认) 这是不换行, wrap 换行, wrap-reverse 从后往前排

默认情况下,项目都排在一条线上, flex-wrap如果一行排不下,可换行

flex-flow: 相当于flex-direction和flex-wrap的简写, 默认值为row nowrap

justify-content: 定义了项目在主轴上的对齐方式, flex-start左对齐 flex-end右对齐

center 居中 space-between排列 space-around也是排列

align-items: 一根交叉轴, 如何对齐

flex-end flex-start center baseline与第一个同行 stretch垂直占满

align-content: 若多根交叉轴, 对齐方式就用此

flex-end flex-start center stretch默认的, space-around space-between

3.6.2针对容器里边,每一个项目的属性进行设置

order: 数值越小排列越靠前

flex-grow: 定义项目的放大比例 默认为0,

如果所有项目的flex-grow属性都为1,则它们将等分剩余的空间(如果有的话),

如果一个项目的flex-grow属性为2, 其他都为1, 则前者占据的剩余空间比其他多一倍

flex-shrink:定义了缩小的比例 默认为1. 如果一个为0,其他为1, 则者不变, 其他等比缩小

flex-basis: 定义了在分配多余空间之前,项目占据的主轴空间.

浏览器根据这个属性计算主轴是否有剩余空间. 它的默认值为auto,即项目的本来大小

flex: 是flex-grow flex-shrink flex-basis 的缩写 默认值 0 1 auto

align-self: 允许单个项目与其他项目不一样的对齐方式,, 可覆盖align-items

默认auto

3.7 css动画

3.7.1 形状上的功能transform

transform: scale(水平缩放, 垂直缩放)

transform: skew(45deg,50deg)绕自身某点倾斜

transform-origin:(水平,垂直) 设置基准点

transform: translate(水平移动距离)

transform: translate(水平,垂直)

transform: rotateX( 45deg);

transform: rotateY( 45deg);

transform: rotateZ( 45deg);

transform: rotateX(45deg)rotateY(45deg)rotateZ(45deg);连写

translate: rotate3d(1,1,1,45deg); 共同设置

transfrom: scaleX(0.5)

transfrom: scaleY(1.5)

transfrom: scaleZ(2.5)

也可连写

共同设置

transfrom: skewX(0.5)

transfrom: skewY(1.5)

transfrom: skewZ(2.5)

也可连写

共同设置

transfrom: translateX(0.5)

transfrom: translateY(1.5)

transfrom: translateZ(2.5)

也可连写

共同设置

3.7.2 动画上的功能

Transition功能支持从一个属性值平滑过渡到另一个属性值

transition: property(对属性操作all所有属性) duration(多长时间完成属性值的平滑过度)

timing-function(linear匀速 ease慢快慢 ease-in慢速开始

ease-out慢速结束 ease-in-out慢速开始慢速结束)

delay(延迟多长时间开始执行)

Animations 功能支持通过关键帧的指定来在页面上产生更复杂的动画效果

**@keyframes yyy** { // ‘yyy’这里设置动画名字  
 **0%**{**background-color**: **darkseagreen**}  
 **25%**{**background-color**: **coral**}  
 **50%**{**background-color**: **blue**}  
 **75%**{**background-color**: **palevioletred**}  
 **100%**{**background-color**: **antiquewhite**}  
 }  
 **div**:**hover**{  
 **animation**: **yyy** 5**s linear** 0**s**;//动画名,时间,方式,停止时间,次数,方向  
 }

如果只有两种状态:

@keyframes yyy{

from{

left: 0px;

}

to{

left: 200px;

}

}

**三. JavaScript**

JavaScript是弱类型语言

作用: 网页特效 表单验证 响应事件

特点: 基于对象 跨平台 改善用户体验 动态性 解释性 单线程

包括:JavaScript( EcmAScript, BOM, DOM)

主流浏览器:

IE trident

Chrome webkit/blink

firefox Gecko

Opera presto

Safari webkit

**1.EcmAScript基础**

1.1

导入.js文件: (导入相当于执行)

<script src=”js/test.js” type=”text/javascrip”charset=”utf-8”>

Console.log(“hello w”)此处代码失效 导入外部的了 此处就不执行了

</script>

在js中输入

console.log(“<h1> sunck</h1>”) 控制台原文输出

alert(“<h1> sunck</h1>”) 提示信息原文输出

document.write(“<h1> sunck</h1>”) 页面,不同, 被渲染

<noscript></noscript>提供无法使用脚本时的替代内容

1.2 JS中的数据类型 (存在栈里, new出新的内存空间)

基本数据类型(原始值): number(包括浮点与整数), string, Boolean, null, Undefined

访问不存在的变量返回undefined或者 已经定义但未赋值也返回undefined

对象数据类型: (存在堆里,只复制引用) 非基本数据类型 array Object function data.......

1.3标识符: 区分大小写, 不能是关键字和保留字

1.4js错误分两种

低级错误(语法解析错误)

逻辑错误(标准错误)

语法错误会引起后续代码终止, 但不会影响其他js代码块

1.5console.log(“sum1”+sum) 后边的整型转为字符串

与console.log(“sum1”+sum.tostring) 效果相同

1.6 Infinity数值上超过js能表示的范围

Infinity+- Infinity=NaN

Infinity+- Int= Infinity

isFinite()判断是否为Infinity

NaN表示不是具体数字, 但值是数字类型, 只要表达式有NaN 计算结果就为NaN

NaN==NaN 返回false

isNaN(NaN) 返回true

1.7 falsy值是假: 0 0.0 “ ” NaN Undefined false null && || ! 短路语句

例如, var ev = e || window.event; 这种方法写兼容

truthy值

1.8类型转换

隐式类型转换

换var num= 1 +“1” 结果为2

isNaN() a++ a-- +a -a 隐式调用Number

这个a+b只要有一个是字符串,则隐式调用string

+ - \* / % 隐士调用Number

&& || ! < > <= >= == !=

显式类型转换

转成number型:

字符串 parseInt(a) parseFloat(a) 从第一个非空白字符起到第一个非数字为止

parseInt(a, radix) 这个radix是把a当作多少进制, 转成10进制

Null Number(null)

Undefined Number(undefined)

Boolean Number(true)

只有数字的串 Number(“+123”)

空字符串 Number(“ ”)

含有非数字的串 Number(“23+324”)得NaN Number(“dsd”)得NaN

转成字符串:

Null “sunck”+null

Undefined “sunck”+Undefined

Boolean true.toString(radix)

Number 234.toString(radix) 原数当成10进制, radix是转成多少进制

还可以用 String(123)

转成布尔型:

“ ” Boolean(“ ”)

Null Boolean(null)

Undefined Boolean(Undefined)

0 Boolean(0)

NaN Boolean(NaN)

false Boolean(false)

0.0 Boolean(0.0)

1.9输入框 var num= parseInt(prompt(“请输入数字”))

typeof 返回值的类型是字符串, for i in trr 其中i值的类型是字符串

没定义的变量, 其typeof返回undefined

null 的类型是object

1.10条件语句 和 循环语句

if, if else 与Java的相同

while, do while, switch case, break,continue 与Java的相同

表达式? 式子1: 式子2 真1 假2

注: console.log(1==”1”) 返回true

console.log(1===”1”)返回false

1.11 for语句

与Java的一样. for(; ;){ } 死循环

for(var i in arr){ } 与Python的一样

1.12数组

var arr1=[10 , 8, ’ab’]

var arr2=new Array(10, 8, ab)

var arr3=new Array(10)

注: for i in arr 中的i是string类型

typeof(arr)输出object,,,,, 因为arr为对象类型

对象? 包含属性和方法 \_\_proto\_\_当成指针看待,无形使用

增加元素 arr2 [12]=6可以超越下标赋值,, arr2[16]=4 可以隔着赋值

数组更新修改arr2 [2]=8,, arr2.length=13 相当于截了

数组删除元素 delete arr2[4] 数据删掉位置还在

数组遍历: for for in 实参arr2.forEach(function(形参item){console.log(item)})

数组常用方法:

push(6) 插后边 Unshift(7)放前边

pop() 删后边 shift() 删前边

join(“#”) 用#连接得字符串

reverse() 倒置数组

slice(1,3) 包前不包后

splice(1,3, 10,20) 从下标为1开始,,替换3个,,替换的内容为10,20

splice(1,0, 10,20,30,40) 从下标为1开始,,替换0个,,替换的内容为 10,20,30,40

concat( [2,3,6,7]) 原数组后加上2,3,6,7,, 得到新数组

var t=arr.indexof(4) 通过数据找到下标

var t=arr.lastIndex(4) 从后往前找

数组排序sort() 冒泡排序

按字符串长短排序 arr.sort( function(x,y){return x.length<y.length} ) 满足条件交换

1.13对象类型

var my\_obj = {

lastName:”Zhang”, 前边是属性名(即key) 后边是属性值(即value)

age: “40”,

}

取值: deng.lastName

赋值:deng.lastName = “lao zhang”

1.14函数

function 函数名(参数列表){

语句

return 表达式 不写return则返回Undefined

}

Js中可以多传几个实参,但是只接受前俩

arguments 接受所有实参,类似于数组的对象, 但不是数组 函数名.length得形参长度

全局变量 在局部可被修改

变量提升, 定义变量的语句先执行, 只升定义 不升赋值

局部变量var i=30 在全局不能使用

在函数中不使用var定义变量,默认为全局变量

function myfun(){ p=30}

myfunc()

console.log(p) 输出30, 但是必须先调用myfun()

函数也是一种数据 a=func() a( ) 也是调用函数

function mysum(num1,num2){return num1+num2}

function myfunc(s,a,b){ return s(a,b)}

console.log(myfunc(mysum,5,6))

匿名函数:

var f=function (a,b){return a+b}

console.log( f(3,4) )

function sum( function(){}, 3, 4) 匿名函数一般当参数传

一次性函数,即时函数 (function(a,b,c){}) (3,4,5)

相当于f(3,4,5)

1.15 js执行过程

语法分析: 执行之前系统先通篇扫描一遍看看语法有没有低级的语法错误, 但不执行

预编译: 变量的声明提升, 函数声明整体提声

imply global 暗示全局变量

a=10 未经声明的变量直接赋值,就归window所有 即 window.a=10

var b = 12 全局上的任何变量,声明了, 也规window所有

window就是全局的域

预编译四部曲:

(1)创建AO对象 Activation Object 执行期上下文,作用是 作用域

(2)找形参和变量声明, 将变量和形参名作为AO的属性名, 值为undefined

(3)将实参值和形参值统一

(4)在函数体里面找函数声明, 值赋予函数体

解释一行执行一行:

1.16字符串

var str1=”hello” typeof(str1)返回string 无属性和方法

var str2= new String(“hello”) typeof(str2)返回object 有属性和方法

字符串一旦创建不能变,str存在只读数据区

字符串常用方法:

charAt(3)通过下标找字母

charcodeAt(4) 下标处对应字母的ASCII码值

String.fromCharCode(97) ASCII--->字母

你replace(“旧”,”新”) 替换

你toLowercase() 大--->小

你toUppercase() 小--->

== === console.log(str1>str2)

str3.locacompare(str4) 右大+1 左大-1 相等0

indexof(“sunck”) 返回该数据对应的下标, 无则返-1

lastindexof() 从后往前找

substring(11,13) 截取从11到13

substr(11,4) 截取从11开始 长度为4

split( ) “sunck is a ”分割

join(“#”) 以“#”拼接

1.17 Math

round四舍五入 ceil向上取整 floor向下取整 pow(2,3)次方 sqrt(25)开方

max min abs Math.random()\*(y-x+1)+x y到x的随机数

1.18 Date

var date1 = Date(“1352-09-11”)

不论是否带参数,返回的都是当前时间 typeof(date1) 返回string类型

var date2 = new Date();

返回当前时间,构造函数不传参数 typeof(date2)返回object类型

var date3 = new Date(“2016/09/18”) 不加8小时

var date4 = new Date(“2016-09-18”) 加8小时

var date5 = new Date(“2016-9-18”) 不加8小时

var date6 = new Date(“2015,5,1, 2,3,4,5”) 年月日 时分秒,毫秒

var date7 = new Date(2000) 2000毫秒

get...... set..... 第231集

var date8 = new Date() 时间转字符串

d.toLocaleString

d.toLocaleTimeString

d.toLocaleDateString

两个时间差: 为两个日期的距离

1.19面向对象

当一个变量没声明就访问会报错, 当对象的属性没定义就访问会返回undefined

对象的创建方法:

1. var obj = { } plainObject 对象字面量/对象

(2) 构造函数

系统自带的构造函数 Object()

自定义

var obj = new Object();

var obj = { } 这两方法来创建对象

var a = new Array();

var a = [ ] 这两种方法来创建数组

自己定义的构造函数:

function Person() { }

构造函数内部原理

(1)在函数体最前面隐式的加上this = { }

(2)执行 this.xxx = xxx

(3) 隐式的返回this,,,, 最后一句隐式的return this

注: 可以显示 return 一个对象, 则return this不会执行,, 用来捣乱构造函数

但是如果显示的 return 一个原始值, 则return this 会被执行, 不会被捣乱

有new 了 就不可能返回原始值

与Java相同

1.20 ES6新特性

var 全局变量

let局部变量

const 常量

模板字符串

箭头函数,

// ES5

var add = function (a, b) {

return a + b;

};

// 使用箭头函数var add = (a, b) => a + b;

// ES5

[1,2,3].map((function(x){

return x + 1;

}).bind(this));

// 使用箭头函数

[1,2,3].map(x => x + 1);

省略function 继承当前上下文的this关键字

当函数只有一个参数可以省略 ( )

当函数只有一个表达式可以省略 { } 和return

对象和数组解构

// 对象

const student = {

name: 'Sam',

age: 22,

sex: '男' }

// 数组

// const student = ['Sam', 22, '男'];

// ES5；

const name = student.name;

const age = student.age;

const sex = student.sex;

console.log(name + ' --- ' + age + ' --- ' + sex);

// ES6

const { name, age, sex } = student;

console.log(name + ' --- ' + age + ' --- ' + sex);

函数默认值

二进制和 八进制字面量 var a = 0o33; let b=0b111;

Spread/Rest 操作符

对象超类 ES6 允许在对象中使用super方法

for in 遍历 for of 遍历 当遍历对象时, for in 一般拿属性名 for of拿属性值

forEach() 不会响应break continue return 这些中断语句

类

1.21 ES7新特性

幂运算 let a= 2 \*\* 3 【这是2的3次方, Math.pow(2, 3) 】

Include [1, 2, 3].includes(3,2) 返回true 【查找3, 从下标为2开始】

1.22 ES8新特性

异步函数 Async functions

Object.entries()和 Object.values()

字符串填充: padStart 和padEnd

Object.getOwnProentyDesctiptors() 指定属性描述符

**2.BOM** browser object model 浏览器对象模型

2.1 window为全局对象

console.log(window.document); 页面文档对象

console.log(window.frames); 浏览器框架集合

console.log(window.navigator); 浏览器机器功能信息的对象

console.log(window.screen); 屏幕信息

console.log(window.location);

console.log(window.history); 历史记录

location:

href 控制地址栏

reload 刷新页面, reload(true) 不带缓存刷新

assign() 加载新页面, 在历史记录有痕迹

replace() 加载新页面, 在历史记录不留痕迹

history:

windows.history.length 历史记录的长度

windows.back()上一页

window.forward()下一页

window.go(num) 跳到第10页

window方法:

window.open(“r.html”,”blank”,”width=90px,height=90px,left=9,top=9”)

window.close() 关闭

2.2 window事件

加载事件 window.onload=function(){alert(“页面加载成功”)}

只有一个有效,后写的被覆盖,,,当页面完全加载成功出发此页面

window.onunload 确认关闭

滚动事件 window.onscroll 发生滚动时执行

滚动高度var a=document.documentElement.scrollTop||document.body.scrollTop

窗口变化事件:

window.onresize=function(){w=document.documentElement.clientWidth

||document.body.clientWidth || window.innerWidth,

h=............ }

2.3定时器

间歇定时器:

vartime time=window.setInterval(function(){console.log(“hello”)},2000)

function func1(){window.clearInterval(time)} 清除定时器

延时定时器:

var time= window.setTimeout(function(){console.log(“world”)},3000)

**3.DOM** Document Object Model 文档对象模型

3.1 DOM是访问HTML和操作HTML的标准

Core DOM 是核心DOM针对任何结构化文档的标准模型

XML DOM 是针对XML文档的标准模型

HTML DOM 是针对HTML文档的标准模型

3.2节点分类

文档节点:整个页面html代码是个文档

标签节点:也叫元素节点

属性节点:不是css 是标签的属性

文本节点:网页里边的文字

注释节点:注释

3.3DOM节点层级关系(DOM树)

父节点: parent node可以拥有任意数量的子节点

子节点: child node

兄弟节点: sibling 拥有相同父节点的节点,叫做兄弟节点

根节点: root node 一个html文档一般只有一个根节点,它是没父节点的

祖先节点: 父节点也是祖先节点

后代节点: 子节点也是后代节点

3.4通过值 获取标签节点的引用

通过id: document.getElementById(“idDiv”)

通过class: document.getElementsByClassName(“classDiv”)

通过name: document.getElementsByName(“inputText”)

通过div标签: document.getElementsByTagName(“div”)

通过p标签: document.getElementsByTagName(“p”)

3.5通过标签节点的引用获取和设置 属性值 文本值 css的值

获取属性节点的值:

官方定义的属性: 标签节点.属性名 直接赋值就是修改

自定义的属性: 标签节点.getAttribute(“属性名”)

标签节点.setAttribute(“名”,”值”) 修改属性值,若无则增加

移除某个属性: 标签节点.removeAttribute(“属性名”)

获取文本节点的值:

标签节点.innerHTML 获取开始标签到结束标签 里边的内容

标签节点.outerHTML 包括开始 结束标签

标签节点.innerText (获取开始标签到结束标签 里边内容的文字) 只获取文字

设置css样式的值:

标签节点.style.backgroundColor=”#fff”

标签节点.style[“height”]=”300px”

以上这两种可以给行间样式,内部,外部样式 赋值. 若获取只能获取到行间样式的值

获取css样式的值：

window.getComputedStyle(标签引用, 伪类). width

window.getComputedStyle(标签引用, 伪类)[“height”]

以上两种能获取行内样式 内部样式 外部样式, 不能赋值

标签节点.offsetTop 是整型的 前边得到的css样式是字符串型的

3.6获取节点的引用

节点类型 nodeName nodeType nodeValue

标签节点 标签名称 1 null

属性节点 属性名称 2 属性值

文本节点 #text 3 文本内容不包括HTML代码

注释节点 #comment 8 注释内容

获取子节点:

标签节点.childNodes

第一个子节点: 标签节点.firstChild

最后的子节点: 标签节点.lastChild

获取该节点的html文档 根节点:

标签节点. ownerDocument

获取当前节点的父节点:

标签节点.ParnetNode

获取同级节点:

前一个: 标签节点.PreviousSibling

后一个: 标签节点.nextSibling

获取该节点的所有属性节点:

标签节点.attributes

3.7对标签的操作

创建标签 var a=document.createElement(“p”)

标签里添加文本 a.innerHTML=”hello”

标签里添加样式 a.style.backgroundColor=”red”

var jsDiv = document.gerElementById(“box1”)

标签里插入子节点 jsDiv.appendChild(a) 插在最后

插入到某某之前 jsDiv.insertBefore(a,jsDiv.lastChild)

创建文本节点 var tt=document.createTextNode(“你好”)

替换子节点 jsDiv.replaceChild(a, jsDiv.lastChild)

复制节点:var newDiv1= jsDiv.cloneNode( ) 只复制当前节点

var newDiv2=jsDiv.cloneNode(true) 复制当前节点和子节点

删除节点:

jsDiv.removeChild(jsDiv.lastChild) 删当前节点的最后一个子节点

jsDiv.parentNode.removeChild(jsDiv)删除当前节点的兄弟节点

返回有定位的祖先节点:

var temp=jsDiv.childNodes[1].offsetParent;

console.log(temp) 返回距离最近的,,如果都没定位,返回body

**4.事件**  (240集 到246集)

4.1添加事件的方式

行间添加 <div id=”box1” onclick=”alert(‘你好’)”> </div>

函数添加 <div id=”box2” onclick=”func()”> </div>

function func(){ alert(“你好”)}

标签节点添加<div id=”box3”> </div>

var jsdiv3=document.getElementById(“box3”)

jsdiv3.onclick = function(){ alert(“中”)}

jsdiv3.onclick = function(){ alert(“中国”)}这个事件把前个事件覆盖

jsdiv3.onclick = null 事件就清除了

监听事件添加<div id=”box4”> </div>

var jsdiv4=document.getElementById(“box4”)

function ff(){ alert(“世界”) }

Jsdiv4.addEventListener(“click”,ff ,false)

Jsdiv4.addEventListener(“click”,ff ,false)不会覆盖上一个

jsdiv4.removeEventListener(“click”,ff ,false)

用this添加事件: this大多指向window

但是写在标签里则指向该标签, 在事件函数里指向调用这个函数的那个标签

4.2事件种类

聚焦事件focus 离焦事件blur

单击事件click 双击事件dblclick

进入mouseover 离开mouseout 按下mousedown 抬起mouseup 移动mousemove

键盘按下keydown 键盘抬起keyup (非功能按键才会触发)

Document.addEventListener(“keydown”,function(e){

var event3= e || window.event

console.log(event3) },false)

4.3事件流

事件捕获true 事件冒泡false

w3c事件流分为三个阶段: 事件捕获阶段, 处于目标阶段, 事件冒泡阶段

var event3= e|| window.event event3.stopPropagation()阻止冒泡出去

a标签: var info = window.confirm(“您浏览的页面存在风险,是否继续?”)

if (info==false){

event3.preventDefault 返回原页不会继续跳转}

4.4 ev.clientX, ev.clientY 浏览器可视窗口坐标

ev.pageX, ev.pageY 页面坐标

ev.screenX, ev.screenY 电脑屏幕坐标

ev.offsetX, ev.offsetY 相对于带有定位的父盒子的x，y坐标

ev.button 0左键 1滚轮键 2右键

4.5 HTML获取屏幕 浏览器 页面 宽高的问题

4.5.1 介绍: 一个页面的展示, 从外到内的容器为 屏幕 浏览器 以及页面本身.

HTML元素展示在页面内, 页面展现在浏览器内 而浏览器展现在屏幕内

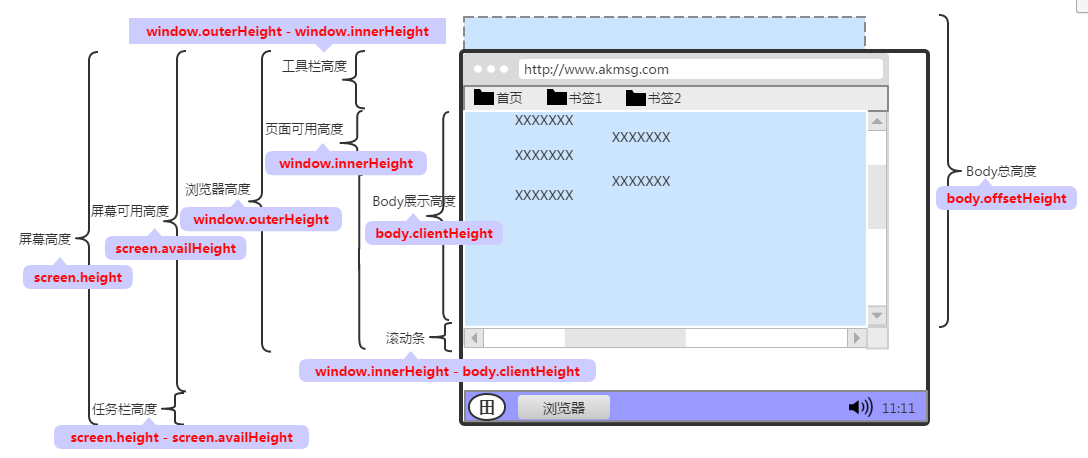
通过js的一些对象可以获取这些容器的高度 宽度

物理尺寸和分辨率: 容器的尺寸是指当前分辨率下的高度 宽度, 而不是物理高度 宽度

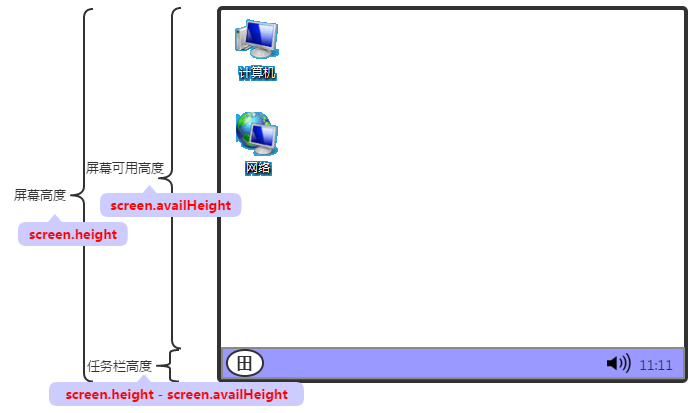
如22寸的显示器, 屏幕分辨率为1366\*768, 那么获取到的屏幕

高度为1366px, 宽为768px.

展示图如下:



4.5.2屏幕信息



**screen.height ：**屏幕高度。

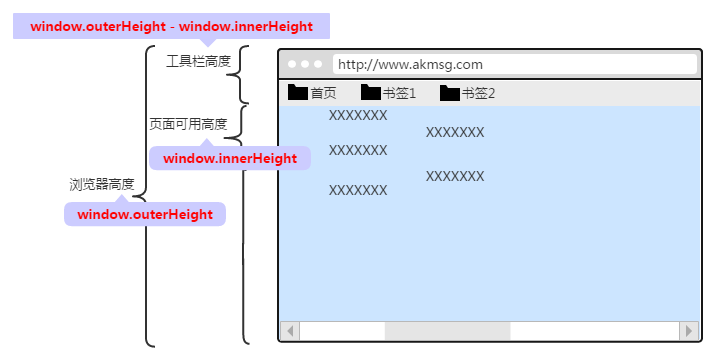
**screen.width ：**屏幕宽度。

**screen.availHeight ：**屏幕可用高度。即屏幕高度减去上下任务栏后的高度，可表示为软件最大化时的高度。

**screen.availWidth ：**屏幕可用宽度。即屏幕宽度减去左右任务栏后的宽度，可表示为软件最大化时的宽度。

**任务栏高/宽度 ：**可通过屏幕高/宽度 减去 屏幕可用高/宽度得出。如：任务栏高度 = screen.height - screen.availHeight 。

4.5.3 浏览器信息



**window.outerHeight ：**浏览器高度。

**window.outerWidth ：**浏览器宽度。

**window.innerHeight ：**浏览器内页面可用高度；此高度包含了水平滚动条的高度(若存在)。可表示为浏览器当前高度去除浏览器边框、工具条后的高度。

**window.innerWidth ：**浏览器内页面可用宽度；此宽度包含了垂直滚动条的宽度(若存在)。可表示为浏览器当前宽度去除浏览器边框后的宽度。

**工具栏高/宽度 ：**包含了地址栏、书签栏、浏览器边框等范围。如：高度，可通过浏览器高度 - 页面可用高度得出，即：window.outerHeight - window.innerHeight。

4.5.4 页面信息



**body.offsetHeight ：**body总高度。

**body.offsetWidth ：**body总宽度。

**body.clientHeight ：**body展示的高度；表示body在浏览器内显示的区域高度。

**body.clientWidth ：**body展示的宽度；表示body在浏览器内显示的区域宽度。

**滚动条高度/宽度 ：**如高度，可通过浏览器内页面可用高度 - body展示高度得出，即window.innerHeight - body.clientHeight。

4.6轮播图 (243---245)

4.7飞机大战

**四. jQuery**

**1.基础**

1.1 jQuery和onload

<script type="text/javascript">

$(document).ready(function(){

alert("你好")

})

window.onload=function(){

alert("中国")

}

</script>

在所有的页面和DOM加载完毕再执行,,相当于onload

不过这个不会相互覆盖, 速度比onload快, 在onload之后再执行.

1.2 jQuery和DOM

$(document).ready( function(){

//DOM对象

var jsDiv50 = document.getElementById("box5")

//jQuery对象

var $jqDiv = $("#box5")

$jqDiv.html("世界")

//jQuery转成DOM对象

var jsDiv51 = $jqDiv[0]

var jsDiv52 = $jqDiv.get(0)

//DOM转jQuery对象

var $jqDiv2 = $(jsDiv50)

})

**2.选择器**

2.1 基本选择器

ID选择器 var $jqDiv = $("#box5")

标签名先择器 var $jqDiv = $("div")

类名选择器 var $jqDiv = $(".class")

并集选择器 var $jqDiv = $(".class, p")

通配符选择器 var $jqDiv = $("\*")

2.2 层次选择器

ancestor descendant选择器 var $jqDiv = $("#box p")

parent > child选择器 var $jqDiv = $("#box >p")

prev+next选择器(同级其后紧挨着) var $jqDiv = $("#box1 + div")

prev~siblings(同级其后的所有) var $jqDiv = $("#box1~div")

**3.过滤器**

3.1简单过滤器

var $jqDiv = $(".pp:first")

var $jqDiv = $(".pp:last")

var $jqDiv = $(".pp:even")偶数行的变 .addClass(“aap”)添加类 .removeClass(“aap”)

var $jqDiv = $(".pp:odd")奇数行的变

var $jqDiv = $(".pp:eq(3)") 下标等于3的

var $jqDiv = $(".pp:gt(3)") 下标大于3的

var $jqDiv = $(".pp:lt(3)") 下标小于3的

var $jqDiv = $(":header")匹配标题

var $jqDiv = $("p:not(.pp)")匹配所有p标签中, class名不是pp的

var $jqDiv = $(".pp:animated")正在执行动画效果的元素

3.2内容过滤器(通过DOM元素包含的文本内容以及是否含有匹配的元素进行筛选)

var $jqDiv = $(".pp:contains(‘中国’)”) 匹配包含给定文本的标签

var $jqDiv = $(".pp:empty") 匹配内容为空的标签

var $jqDiv = $("p:parent") 匹配含有子元素的标签 或者 含有文本的标签

var $jqDiv = $("div:has(p)") 匹配所有div标签中, 其子元素含有p标签的

3.3可见过滤器

var $jqDiv = $(".pp:visible") 匹配所有可见的标签

var $jqDiv = $(".pp:hidden") 匹配所有不可见的标签 type=”hidden”或display:none

3.4表单对象的属性过滤器

var $jqDiv = $("input:checked") input标签中 找到所有被选中的复选框

var $jqDiv = $("input:disabled") 匹配有disabled属性的,并且值是disabled

var $jqDiv = $("input:enabled")匹配所有可用的标签

var $jqDiv = $("select option:selected") 匹配被选中的option

3.5子元素过滤器

:firs-child 匹配给定标签的第一个 子标签

:last-child 匹配给定标签的最后一个子标签

:only-child 无兄弟标签则匹配,,有兄弟标签则不匹配

:nth-child(3) 匹配第几个 也可为even/odd/equation

**4.属性选择器**

var $jqDiv = $(input[name]") 匹配所有带name属性的input

var $jqDiv = $("input[name=aa]") 匹配属性值为aa的标签

var $jqDiv = $("input[name!=aa]") 不等于的

var $jqDiv = $("input[name\*=a]") name中含有a的

var $jqDiv = $("input[name^=a]") 以a开头

var $jqDiv = $("input[name$=a]") 以a结尾

var $jqDiv = $("div[id][name=aa]") 同时满足多个条件

**5.表单选择器**

var $jqDiv = $(":input") 匹配所有的input

var $jqDiv = $("form :input") 匹配form中的所有input

var $jqDiv = $(":button") 匹配所有按钮

var $jqDiv = $("form :checkbox") 匹配所有复选框

var $jqDiv = $("form :file") 匹配所有文件域

var $jqDiv = $("form :hidden") 匹配所有隐藏域

var $jqDiv = $("form :image") 匹配所有图片域

var $jqDiv = $("form :password") 匹配所有密码域

var $jqDiv = $("form :radio") 匹配所有单选按钮

var $jqDiv = $("form :reset") 匹配所有按钮

var $jqDiv = $("form :submit") 匹配所有提交按钮

var $jqDiv = $("form :text") 匹配所有文本框

**6.选择器注意事项** 255集

注: attr(“名”,”值”) addClass removeClass

**7.标签内容的操作**

获得文本内容: .text()获取第一个 .text(“中国”)设置所有内容

获得html内容: .html .html(“中”)

值的操作: .val() .val(“世界”)

$(“input”).val( [2] )

$(“select”).val([“列表1””列表3”])

val/html 设置全部,取值为第一个匹配元素

**8.节点的操作**

8.1创建节点

创建标签节点: var $jsp=$(“<p></p>”)

创建文本节点: var $jsp = $(“<p> hello world </p>”)

创建属性节点: var $jsp = $(“<p title=”nice”> hello world </p>”)

8.2插入节点

内部插入,成为子元素:

$(“div”).append($jsp)追加 后边插入

$jsp.appendTo( $(“div”)) 将所有匹配的元素添加另一个元素集合中

$(“div”).Prepend($jsp) 前边插入

$jsp.prependTo( $(“div”))前边插入

外部插入,成为兄弟元素:

$(“div”).after($jsp) 在每一个匹配的标签之后插入

$jsp.insertAfter( $(“div”))

$(“div”).before($jsp) 在每一个匹配的标签之前插入

$jsp.insertBefore( $(“div”))

8.3删除节点

remove() var $p1=$(“div p:first”) $p1.remove() 内部的子标签也都删 $p1也删了

detach() 对象不删 $p1还能用

empty() 标签内容的清空

8.4节点的复制

$("#box1 p:eq(1)").bind("click",function(){

$(this).clone().insertAfter($(this))

})

如果clone(true) 有了true这个参数,会把事件处理程序也一并复制

8.5替换节点

$("#box2").replaceWith("<div> 123456</div>") 旧.replaceWith(新)

$("<div>good </div>").replaceAll("#box3") 新.replaceAll(旧)

8.6遍历节点

$("p").each(function(index){

$(this).attr("title","我是第"+(index+1)+"p标签")

})

8.7包裹节点

包裹节点 $("button:eq(0)").click(function(){

$("span").wrap("<p></p>");

});

去除包裹 $("button:eq(1)").click(function(){

$("span").unwrap();

});

整体包裹 $("button:eq(2)").click(function(){

$("span").wrapAll("<p></p>");

});

包裹内部 $("button:eq(3)").click(function(){

$("span").wrapInner("<p></p>");

});

**9.样式的操作**

添加样式

添加样式 addClass(class)

删除样式 removeClass(class)

若存在就删除, 不存在就添加 toggleClass(class)

若switch为true则添加, 为false则删除 toggleClass(class,switch)

修改样式 $("button").click(function(){

$("#box2").css("background-color","yellow");

})

获取样式 $("button").click(function(){

console.log($("#box2").css("backgroundColor"));

})

文件加载完毕再执行jQuery

$(window).load(function(){

})

**10.事件的操作**

10.1绑定事件

bind 所有的p都添加,p多了消耗资源性能低, 而且后创建的标签绑定不上

delegate $(“div”).delegate(“p”,”click”,function(){

console.log($(this).text())

})

这种是给父元素div绑定了一个事件,即使多个p也只绑定一次,

并且新建的标签也能绑定上, 不足是DOM层级过多效率低

on $(“div”).on(”click”,“p”,function(){

console.log($(this).text())

})

one一次性事件 $(“#button”).one(“click”,function(){}

Live

10.2解除绑定

unbind $(“div p”).unbind(“click”)

undelegate $(“div”).undelegate(“p”,”click”)

off $(“div”).off(“click”,”p”)

10.3模拟点击

a.trigger(“click”)会导致浏览器同名的默认行为的执行

a.triggerHandler() return false 表示不会

10.4模拟悬停

$(“#button”).hover(function(){

console.log(“移入”)

}, function(){console.log(“移出”)

})

10.5冒泡和默认行为

阻止冒泡: e.stopPropagation() 或者return false阻止冒泡和默认行为

阻止默认行为:$(”#link”).bind(“click”,function(e){

var d=window.confirm(“网页有病毒,是否继续访问?”)

e.stopPropagation() //阻止点击该链接时冒泡

if (d==false){

e.preventDefault() //阻止默认行为

或者用 return false //同时阻止冒泡和默认行为

})

10.6动画的效果

隐藏document.getElementById("b1").onclick = function(){

document.getElementById("box1").style.display="none";

}

或者用 $("#b1").click(function(){

//2000代表2秒没的,,2000可用”slow normal fast”代替

//slow 600 normal 400 fast 200

$("#box1").hide(2000,function(){

console.log("当动画结束之后调用")

})

})

显示 show

一键切换toggle 是隐藏的让你显示,原来是显示的让你隐藏

淡入fadeIn() 位置没了

淡出fadeOut() 位置没了

淡入淡出fadeToggle(1000) 位置没了

切到某个透明度 fadeTo(2000, 0.1)第一个参数是时间第二个参数是透明度位置还在

滑动隐藏slideUp()

滑动显示slideDown

一键切换滑动 slideToggle()

var self = this

$(“#box1”).slideToggle(2000,function(){

$(self).text(“显/隐”)

})

自定义动画

//先横后竖 delay停止两秒 ({left:”+=200px”},2000)继续走200px

$("#box1").animate({left:"200px"},200).delay(2000).animate({top:"200px"},200);

//沿对角线动 opacity是透明度

var div55 = $("#box1").animate({left:"200px", top:"200px", opacity:"0.3"},2000)

//点击按钮停止

$("#b4").click(function(){

div55.stop(false,false)

})

//判断动画是否正在执行

$("#b5").click(function(){

if(a.is(":animated")){

console.log("动画正在执行")

}

})

**五.Ajax**

1.Directx的安装 Apache的安装263集<https://jingyan.baidu.com/album/29697b912f6539ab20de3cf8.html?picindex=2>

2.URL:统一资源定位符，是互联网上资源的地址

http://www.aspxfans.com:8080/news/index.asp?boardID=5&ID=24618&page=1#name

协议 域名也可以是IP 端口号 目录 文件名 参数 锚

访问过的IP存于C:\Windows\System32\drivers\etc\hosts中

3.同源策略（同源三大要素）

同协议：http https

同域名/同IP： www.Baidu.com 22.19.89.989

同端口：8080 http默认使用80端口

4.ajax 266集

**六. 移动端基础**

**1.移动端基础**

1.1手机浏览器内核( 产业概况: 全球有四大独立的浏览器内核)

微软IE的Trident

网景最初研发后卖给Mozilla基金会并演化成火狐的Gecko

KDE的开源内核webkit (Android原生浏览器, 苹果Safari, 谷歌的Chrome都是基于此)

Opera的 Presto

1.2视口viewport

1.2.1<meta name="viewport" content=" width=device-width, initial-scale=1.0,

maximum-scale=1.0, minimum-scale=1.0, user-scalable=no">

在网页的<head>中增加以上这句话，可以让网页的宽度自动适应手机屏幕的宽度，下面是这些属性的解释：

(1)width=device-width ：表示宽度是设备屏幕的宽度

(2)initial-scale=1.0：表示初始的缩放比例，1.0就是占网页的100%

(3)minimum-scale=1.0：表示最小的缩放比例

(4)maximum-scale=1.0：表示最大的缩放比例

(5)user-scalable=no：表示用户是否可以调整缩放比例

如果要兼容IE6/7/8的话，就要用css3-mediaqueries.js

<!--[if lt IE 9]>

　　　　<script src="http://css3-mediaqueries-js.googlecode.

com/svn/trunk/css3-mediaqueries.js"></script>

<![endif]-->

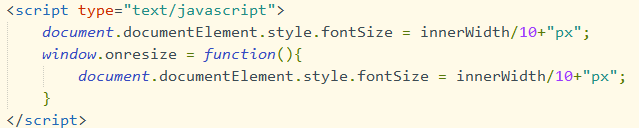
1.2.2宽度不要用绝对宽, 用相对的: width:auto; 或者 width: XX%

1.2.3字体大小是 页面字体默认大小的100%, 即16像素, 不要使用绝对大小”px”,使用”rem”

例: html{font-size: 62.5%} 此段表示字体大小为 默认字体(16px) 的62.5%, 即10px



详情: <https://www.cnblogs.com/eyed/p/7872521.html>



1.2.4 media媒体查询 识别屏幕规格尺寸

1.3像素 (两种像素)

设备像素: 设备屏幕的物理像素, 任何设备的物理像素的 数量都是固定的

CSS像素: 为web开发者创造的, 在CSS中 使用的一个抽象的层

**2.多媒体 (277 278)**

2.1音频audio

2.2视频video

**3.触屏事件 (283)**

3.1

3.2 zepto

**4.swiper 轮播图**

**七. VUE**

1.基础

1.1Vue是什么?

构建数据驱动的web应用开发框架

1.2为什么如此受欢迎?

声明式渲染: 前后端分离是大趋势,让DOM节点跟数据绑定, 只要操作数据就可以更改页面

渐进式框架: 适应各种业务需求以及场景,随着业务改变以及复杂度,可以引入各种库,帮助开发

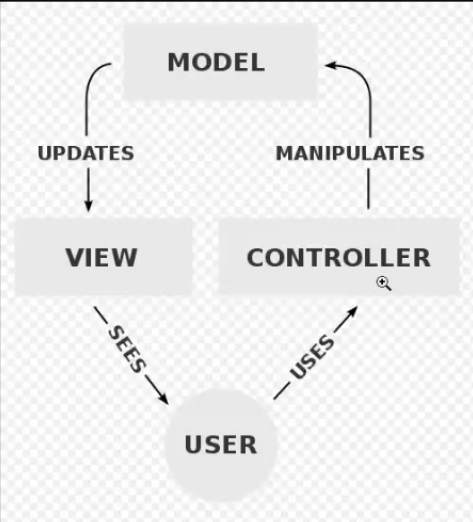
快速交付: 易用灵活高效, 结合第三方UI框架

1.3引用方式?

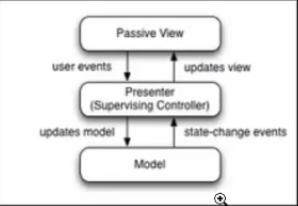
引入CDN,下载js文件或者 搭建脚手架

1.4MVC MVP MVVM架构模型对比?

MVC: controller 薄, View 厚, 业务逻辑大都部署在View

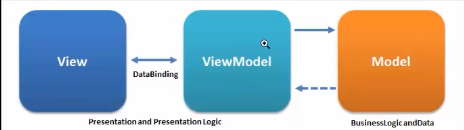


MVP:View 薄, 不部署任何业务逻辑, 称为被动视图, (Passive View) Presenter厚,逻辑都部署在这里



MVVM: 双向数据绑定, View的变动,映射在ViewModel上,反之一样

虽然不是完全的MVVM框架但是受到它的启发



1.5与React Angular对比

相似处:

都使用了虚拟DOM (Virtual DOM)

都提供了响应式(Reactive) 和组件化(Component)的视图组件

都将注意力集中保持在核心库, 而将其他功能 (如 路由 全局状态管理) 交给其他相关库

不同处: vue 虚拟dom 轻量, 学习成本低

vue 有 css作用域

React Native 能够渲染成原生的js, 对应的vuex可以生成原生的js代码

1.6 多页面应用

缺点: 页面切换慢

优点: 首屏时间快, SEO优

单页面应用

优点: 页面切换快

缺点: 首屏时间慢, SEO差

2.语法

2.1 var vm = new Vue({

el:”#app”,

data:{

name:”zhangsan”,

job:”工程师”,

website: “http:www.ddofia.com”,

},

methods:{ }

}) 以上的属性值可以通过 vm.name vm.job 的方式访问到,也可以更改

2.2模板语法

插值: (1)文本 {{ name}} 或用 v-text=”name”

(2)纯HTML v-html=”name” 容易导致xss攻击

(3)表达式

指令: v-bind:href=”website” 绑定属性 引号里边的内筒可以写表达式 三目运算符等等

v-on:click=”age++” 绑定事件,,其中age是属性

v-on:click=”addfun” 绑定事件,, 1其中addfun是方法 若传参需要给()

v-if增删 v-show显隐 v-else, 加一个key值,防止dom复用

v-model

循环: v-for

<div v-for=”(i, index) in list ” :key=”index”> {{ i }}--{{index}}</div>

<div v-for=”(i, index) in list ” :key=”i.id”> {{ i .text}}--{{index}}</div>

数组循环, 当修改数组内容时, 不能通过下标的形式来改变数组,

通过数组的变异方法来修改数组才能够实现,

数据发生变化页面也跟着变,这种响应式的效果.

(1))七个: push pop shift unshift splice截取 sort排序 reserve逆序

(2)改变引用, 直接操作vm.list的引用 让它指向另外一个地址空间的时候

(3)set方法 Vue.set(vm.list, 1, 5)

补充: template 站位符, 临时标签, 可以把循环体放在这里

对象循环: <div v-for=”i in objList”>{{ i }} </div>

<div v-for=”(i,key) in objList”>{{ i }}--{{key}} </div>

<div v-for=”(i,key,index) in objList”>{{ i }}--{{key}}--{{index}} </div>

对象和数组一样动态的用.key 加值不会带来页面的变化,

(1)用改引用的方式 添加数据页面更新 好使

(2)set方法 Vue.set(vm.userInfo, “address”, “beijing”)

3. event.stopPropagation 阻止冒泡事件

或者用事件修饰符 v-on:mousemove.stop=”” 来阻止冒泡事件

事件修饰符: stop once prevent self

4.计算属性

4.1首字母大写 <div> {{ name.substring(0,1).toUpperCase()+ name.substring(1) }}</div>

computed:{

changeword: function(){

return this.name.substring(0,1).toUpperCase()+ this.name.substring(1)

}

}

<p> 我是计算属性 {{ changeword}}</p>

methods:{

changewordMethod: function(){

return this.name.substring(0,1).toUpperCase()+ this.name.substring(1)

}

}

<p> 我是普通方法 {{ changewordMethod()}}</p>

计算属性基于它的依赖可以进行缓存,

计算属性只有它的相关依赖改变时才会重新计算

若不改变 连续调用多遍计算属性, 不会调用changeword方法来重新进行加计算

methods 只要连续调用, 每一次都会重新计算

当页面其他数据更改, 页面会刷新,重新渲染.. 这时methods就会被再一次执行,而计算属性则不会被再一次执行,因为依赖项没有发生更改. 例如 vm.age = 17

4.2 计算属性VS watch

watch中的函数是不需要调用的, 不用return, 一个属性影响多个属性时用

computed函数调用时不需加(), 需要return,多个属性影响一个属性时用

计算属性不适用异步操作, 要用watch

4.3 setter

计算属性默认只有getter, 不过在需要是你也可以提供一个setter

5. 可以实例化多个vue实例, 分别接管不同的div 标签

6.动态绑定css样式

6.1在style标签中,定义activated的样式内容

:class=”{ activated: isact }”

在data中{ isact: false}

methods 中this.isact = !isact

6.2 在style标签中,定义activated的样式内容

:class=”[ ac , bc, cc.....]”

在data中{ ac: “”}

methods中 this.ac = “avtivated”

6.3 用内联样式 用变量

:style=”styleobj”

data中styleobj:{ color:”black”}

methods中 this.styleobj.color = this.styleobj.color ===”black””red”:”balck”

6.4用内联样式 用数组

:style=”[ styleobj, { fontSize:10px }, styleobj3 ]”

data中styleobj:{ color:”black”}

methods中 this.styleobj.color = this.styleobj.color ===”black””red”:”balck”

7. 生命周期函数(钩子)

//生命周期函数就是vue实例在某个时间点会自动执行的函数

new Vue() 创建vue实例的时候

init Events & Lifecycle vue进行基本的初始化

beforeCreate

init injection & reactivity vue会进行外部注入 和 双向绑定相关的内容

created

Has ‘el’option? 是否有el这个选项

有, 走yes这条线, 直接进入下一步

没有, 走no这条线, when vm.$mount(el) is called

Has ‘template’ option? 是否有 template这个选项

有 yes, Complle template into render function 用模板去渲染

没有no, Complle els outerHTML as template 把el外层的标签当成模板

beforeMount 模板和数据相结合的DOM元素,挂载到页面上 之前将会被执行

Create vm $.el and replace‘el’with it

模板结合数据 生成的最终vue实例里边的DOM元素, 会被挂载到页面之上

mounted 挂载完毕会被执行

在Mounted 被挂载的情况下

beforeUpdate 当数据改变 还没重新渲染之前,会执行

Updated 重新渲染之后, 会执行

when vm.$destroy() is called 当destroy方法被调用销毁组件时,

beforeDestroy 在组件销毁之前会调用

Destroyed 在组件销毁之后会调用

activated

deactivated

errorCaptured

10. 脚手架

10.1 安装nodejs [从官网下载]

node -v

npm -v

10.2 安装脚手架

sudo npm install --global vue-cli

[版本三] npm install -g @vue/cli yarn global add @vue/cli

npm uninstall -g @vue/cli yarn global remove @vue/cli

vue --version

10.3 创建项目

vue init webpack “vue-project”

[版本三] vue create vue-project

版本三创建项目 各选项:

Bagel

typeScript

Progressive Web App (PWA) Support

Router

Vuex

CSS pre-processors

Linter / Formatter

Unit Testing

E2E Testing

Use class-style component syntax? (Y/n) 是否使用class风格的组语法

10.4 运行项目 cd到 项目目录下

npm install

npm run dev (或者npm start)

[版本三] npm run serve

10.5 打包

[版本三] npm run build

11.src目录以及项目结构(组件)

11.1Html页面的css样式内容可以单独放到 .css 文件中,然后用href来引入,

HTML页面的script内容也可以单独放到 .js 文件中, 然后用src来引入

同样, js文件中的子组件部分也可以单独放到 .vue 文件中, 然后用import导入

这其中的(“head”,{.......}) 第一个参数为 .vue文件的文件名, 第二参数为文件内容

11.2组件嵌套

子组件名字是什么, (template标签内跟div) class名字就是什么

子组件名字是什么 name名字就是什么

注册成全局组件:**import** My **from './components/My'** Vue.component(**"my66"**,My) *//注册全局组件*

以上操作是在 main.js 中完成的.

注册成局部组件: **import** My **from './components/My'**

**components**:{  
 **HelloWorld**:HelloWorld,  
 **My**:My}

以上操作是在 App.vue中进行的

11.3 组件css作用域

scoped

12. 组件使用的细节

12.1 table下tbody中要用组件, 但是不能直接写, <tr is=’row”></tr> 其中row是子组件

<ul>标签 <select>标签 也是一样要用<option is=”row”></option>

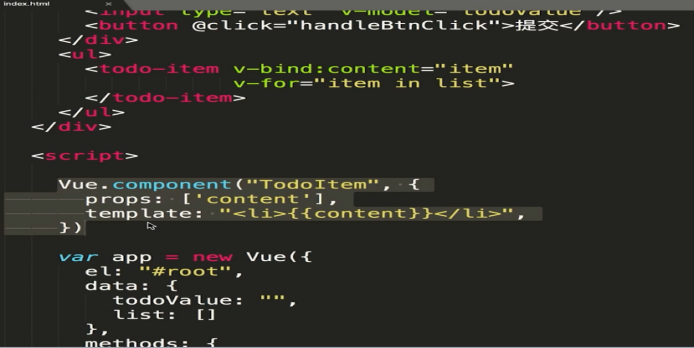
12.2 子组件data用函数定义, 并return一个{ }

12.3 用ref标记某个标签, 用**this**.**$refs**.box1 来获得这个标签节点

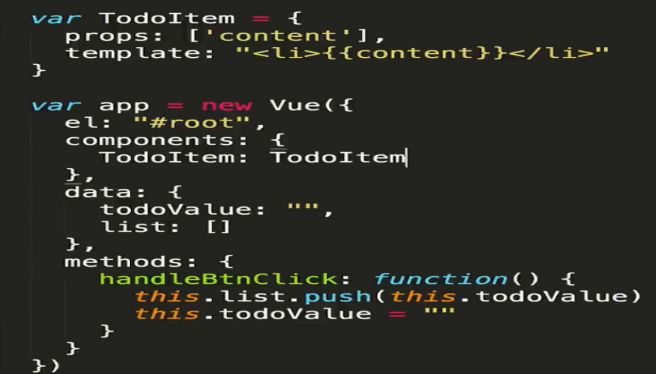
如果ref放于组件当中,获得的是当前组件的引用可以用 .获得子组件属性值

12.4 父子组件

全局子组件:



局部子组件:



父传子:

父组件中: <**my v-bind:ddd="zz"**></**my**> 不加v-bind, zz是字符串,加了 zz就是js表达式

子组件中: **props**:[**"ddd"**] 来接收

不要直接修改父组件传过来的数据, 在子组件中重新定义一个变量:

return { childeee : this.ddd} 这是单向数据流,

子传父:

this.$emit(“自己事件名”, 参数)

传值 string number boolean

传引用 array object

12.5 组件参数校验

父组件向 子组件传递内容, 子组件有权对这些内容进行校验

props: [ content1, content2]

props: { content1: String ,

content2: String}

props: {content1: [String, Number]

content2: String }

props: { content1: {

type:String,

required: true 这个数据毕传不然报错

default: ‘默认值’

validator: function(value){ return value.length > 5 }

}

}

非props特性:

(1)父组件向子组件传递了一个 数据, 但是子组件并没有声明props来接收传递过来的数据,,

子组件就无法使用 父组件传递过来的这个数据,

(2) 父组件这个属性会展示在, 子组件最外层的标签里.

12.6给组件绑定原生事件

@click.native

12.7非父子组件传值 (Bus/总线/发布订阅模式/观察者模式)

<**script**>  
 Vue.**prototype**.**bus** = **new** Vue() //prototype是给类对象增加属性的,   
 Vue.**component**(**'ccc'**,{  
 data:**function**(){  
 **return**{  
 **text**: **this**.**content** }  
 },  
 **props**:{  
 **content**: ***String*** },  
 **template**: **'<div @click="han">{{text}}</div>'**,  
 **methods**: {  
 han: **function** () {  
 **this**.**bus**.$emit(**'ch'**,**this**.**text**)  
 }  
 },  
 mounted: **function**(){  
 **var** mythis = **this  
 this**.**bus**.$on(**'ch'**,**function** (v) {  
 mythis.**text** = v  
 })  
 }  
 })  
  
 **var *vm*** = **new** Vue({  
 **el**: **"#app"**,  
 })  
</**script**>

prototype是给类对象增加属性的,

12.8 vue中的插槽 ( 子组件的一部分内容, 是父组件传递过来的,)

父中 <child> <p>你好</> </child>

子中,child这个组件的template下, <slot></slot>

插槽可以方便的向子组件插入 整块DOM元素

<slot>默认 内容</slot> 如果父组件不传,则显示默认内容

具名插槽: 页面引用处加属性 slot=”zhang”

子组件中 <slot name=”zhang”> </slot>

作用域插槽: 父中<child>

<template slot-scope=”ttt”>

<h1>{{ ttt.i88 }}

</template>

</child>

子中:child组件,template下

<ul>

<slot v-for=”item in datalist :i88=item>

</slot>

</ul>

作用域插槽, 数据来源于子组件, 样式来源于父组件,, 相当于子传给父

12.9 动态组件

<component :is=”子组件名”> </component>

在子组件中 template下 v-once指令 ‘<div v-once> child-one</div>’

13. vue中的动画

14. 路由

安装路由 npm install vue-router --save-dev

路由容器<router-view> </router-view> 显示当前路由地址所对应的内容,

路由重定向: { path:”/”,

redirect:”/home”

重命名路由: {

path: “/home”,

component:Home,

name: “ho”

}

{ path:”/”,

redirect:{name:”ho” }

子路由 children:

跳转:

<router-link to=””> </router-link> 这样的跳转 会渲染成a链接

编程式导航: (来进行跳转)

router.push(“/detail/${index}”)

{ path: “/detail/:myid”

component:Detail }

{{ $route.params.myid }}

mode : history 需要后台配置的支持, 因为我们的应用是单页面客户应用, 如果后台没有正确的配置, 当用户在浏览器直接访问http://xxx.com/home/id, 就会返回404,,要做正真的路由匹配

15. vuex

**八. Sass**

1.Sass的介绍

1.1 Sass是一种CSS的开发工具, 提供了许多便利的写法,大大节省了设计者的时间,使得CSS的开发,变得简单和可维护.

Sass 是一门高于CSS的元语言, 它能用来清晰地 结构化地描述文件样式, 有着比普通CSS更加强大的功能.

Sass是世界上最成熟 稳定 和强大的专业级CSS扩展语言

Sass是Ruby语言写的, 但是两者的语法没有关系.

1.2 Sass的意义

易维护,更方便的定制,比较适合大型的web项目. 可以将css代码模块化, 效率提升.

2. Sass的知识点介绍

变量定义: $

数据类型:( 字符串, 数组, Maps, 颜色)

运算: ( 数字运算, 颜色运算, 字符串运算, 布尔运算, 数组运算)

嵌套

混合

继承扩展

导入

判断语句

数组

循环语句

function

**九. Element-ui**

**1.基础**

1.1安装 npm i element-ui -S

1.2引入 在main.js中

import ElementUI from ‘element-ui’;

import ‘element-ui/lib/theme-chalk/index.css’;

Vue.use(ElementUI);

**十. Echarts**

**1.创建本地库**

github官

**github**

**1.创建本地库**

github官网上注册, 并下载安装git应用程序

点开安装完毕的Git Bash (这个相当与linux的终端命令窗口)

git config --global user.name ”yrvh”

git config --global user.email ”..........”

cd到uu文件夹中, 在该终端窗口执行 git init 来创建本地仓库

**2.在本地仓库中的一些操作**

2.1常用命令

git add 123.txt 添加这个文件到缓存区

git commit -m “这是第一次提交”把缓存区内容提交到本地仓库

git status 查看日志

git diff 查看修改内容

git log 查看曾经提交到本地仓库的日志

git log --pretty=oneline 作用同上, 只显示编号和信息

2.2版本回退

git reset --hard HEAD^ 回退到上一个版本

git reset --hard HEAD^^ 回退到上两个版本

git reset --hard HEAD~100 回退到100之前的版本

git reset --hard 版本号 回退到具体某个版本

git reflog 记录每次执行的命令

2.3撤销某一步

**工作区(代码区)**--------add--------> **缓存区** --------commit--------->**仓库区(版本区)**

git checkout --123.txt 在add之前执行,撤回到与版本库相同, 若已经add到缓存区,又有修改,但还没commit到仓库区,撤回到与缓存区相同

2.4 vim Esc Shift ; q 1 用来退出vim

**3.远程仓库的操作**

3.1终端命令窗口 ssh-keygen -t rsa -C “邮箱”

在这个目录下 /c/users/Administrator/.ssh 复制id\_\_ras和id\_\_rsa.pub文件中的秘钥

3.2GitHub官网, 点击自己头像,再点击setting 复制秘钥于其中

3.3 cd到uu文件夹中,测试秘钥: ssh -T git@github.com

3.4 关联远程库, 网页右侧获得gitHub地址

在uu文件夹下执行 git remote add origin “远程地址”

git remote 测试有没有连接上

3.5拉取远程库内容到本地库 git pull origin master --allow-unrelated-histories

3.6推送本地库内容到远程库 git push origin master

3.7取消关联远程仓库 git remote rm origin

3.8克隆远程仓库 git clone “远程库地址”

3.9选择不上传某些文件

touch .gitignore 在uu文件夹中创建一个文件

将不想提交的文件名写到 .gitignore文件中

但是如果想把.gitignore中的commit到本地仓库 则需加 -f

**4.分支管理**

4.1 git branch 分支名 创建分支

git checkout 分支名 切换分支

git branch 打印出所有的分支

git checkout -b 分支名 创建分支同时切换到该分支

4.2在master主分支下 git merge 分支名 来合并分支

git merge --no-ff把合并的记录添加到gitlog中

git branch -d 分支名 删除本地仓库的分支

在week分支下执行 git push origin --delete week 删除远程分支

在week分支下执行 git push --set-upstream origin week 创建远程分支

在master 下执行 git push origin master 提交到远程主分支

在week分支下执行 git push origin week 提交到远程week分支

git clone “远程库地址” 只能克隆master分支, 在本地再创建week分支,然后在week分支下执行 git pull origin week 把远程week分支的内容拉下来

4.3 解决冲突

master和week 都更改了,并且都add commit了,

在master下 执行git merge week会全部打印出来, 多余的部分手动删除.

git log --graph 查看分支合并图

4.4分支策略

master分支应该是非常稳定的,仅用来发布新版本的,平时不能在上面干活

干活都在dev分支上,也就是说dev分支是不稳定的

你和你的小伙伴都在dev分支上干活, 每个人都有自己的分支,时不时的往dev分支上合并就行了.

4.5 git checkout -b dev origin/dev 创建本地dev分支同时 创建远程分支dev

5.标签管理

git tag 标签名 打标签

git tag 查看所有标签

git tag 标签名 commitID 给特定的commit id打标签

git tag -a 标签名 -m “标签信息” 指定标签信息

git checkout v1.0 切换到指定标签的版本

git show 标签名 查看说明文字

git tag -d 标签名 删除标签

git push origin 标签名 推送标签到远程

git push origin --tags 一次性推送全部尚未推送的标签到远程

删除已经推送到远程的标签 先从本地删除 git tag -d 标签名

再从远程删除 git push origin :refs/tags/标签名

**微信, 支付宝, Threejs, Web安全攻击, Canvans**