|  |
| --- |
| 笔记张雷 |



# Linux命令符

|  |  |
| --- | --- |
| 命令符 | 注释 |
| mkdir 文件名 | 创建文件 |
| ./ | 当前目录 |
| ../ | 上级目录 |
| cd “相应目录名” | 进目录 |
| Ls [选项] [文件或目录] | 查询目录中内容 |
| ls -l(=ll) | 查看详细信息 |
| ls -a | 查看所有目录（隐藏文件） |
| ls -d | 查看目录属性 |
| ls -h | 人性化显示文件大小（带单位） |
| ls -i | 显示inode |
| . 文件名 | 隐藏文件 |
| rm -rf 文件名 | 强制递归删除目录 |
| cp 源 目标 | 把源文件复制到目标文件 |
|  |  |
|  |  |

文件类型：① - 文件②d 目录③ i 软连接文件（快捷方式）

# 管理员 $普通用户

-rw-r--r--. -文件类型 U所有者 G所属组 O其他人 .acl权限

r读 w写 x执行

目录文件处理

|  |  |
| --- | --- |
| mkdir [目录名] | 创建目录 |
| mkdir -p [目录名] | 递归创建 |
| pwd | 查看当前路径 |
| cd ～或者cd | 进入当前用户家目录 |
| cd - | 进入上次目录 |
| cd .. | 进入上一级目录 |
| cd | 进入当前目录 |
| rm [目录名] | 删除空文件 |
| rm -rf [目录名或文件] | 强制删除 |
| cp [选项] [原文件或目录] [目标目录] | 复制命令(同名复制异名复制) |
| cp -r | 复制目录 |
| cp -p | 连带文件属性复制 |
| cp -d | 原文件是链接文件，则复制链接属性 |
| cp -a | 相当于以上三个-pdr 不改变文件最后更改时间 |
| mv [原文件或目录] [目标目录] | 同目录改名，不同剪切 |
| Touch [文件名] | 创建空文件 |
| Gcc filename | 编译 |
| ./a.out | 执行C语言 |
| Cat 文本文件 | 看文本文件 |
|  |  |

Ctrl+L清屏；Ctrl+v d块删除jk上下动

Table：目录、命令补全

相对路径：参照当前目录，进行查找

绝对目录：写全的目录。一级一级查找

# HTML

1.锚点：a链接到ID。 Eg:<a href=”#id”></a> target：blank新页面

2.ol 的start属性：命令li从何处开始.

3.id 同一页面唯一页面标识符。不可以重复类似IDcard；

4.当你在浏览器输入URL后回车，发生了什么？

①进行dns解析，得到IP

②三次握手，建立TCP链接

③发送HTTP请求（可能是一堆请求）

④服务器响应（根据客户端的缓存情况，返回内容）

⑤

1. table标签

①tr只能嵌套td或th不能添加其他的。

②align写在table令table屏幕居中，写在tr令table行的内容居中。

③cellspcing各个单元格的间距

④cellpadding每个单元格内的间距，如同padding

⑤caption跟在table后作为表格标题

⑥thead、tbody、tfood

⑦rowspan作用于tr和colspan作用于td

1. form标签

①<form>

<fieldset>

<legend>健康信息</legend>

身高：<input type="text" />

体重：<input type="text" />

</fieldset>

</form>

②form属性 action 服务器端URL

method 提交数据方法 get、post

name

③input type属性：text、password、radio、checkbox、(button、image图片 按钮、reset、file)

value 属性 ：改内容、给后端传数据

name ：radio要用一样的

accept：指定file选择类型

multiple：指定file或者select可以多选bool类型，可以不写值

checked：checkbox默认被选中

readonly：只读，不可以编辑

disabled: 禁止操作

maxlength: 最大字符个数

④textarea：多行文本（cols,rows或者width,height）

⑤select: 下拉出菜单（属性size、multiple）

1. doctype:浏览器执行标准

8.<lable for=”inputid”></lable>给对应input点击获得焦点

# CSS

1.嵌套规则：block可以嵌套block和line；

Line可以嵌套line不可以嵌套block

1. 语义化（带有含义的标签）

语义化作用 ①便于搜索引擎优化SEO，（提高网站流量，提高搜索引擎的排名）。

②没有样式的情况下，网页结构组织有序，并且有良好的可读性。

③尽量使用语义化标签。

1. 命名方法：不是驼峰不是下划线。简明有意义。例如：Table-color
2. 类名可以重复、ID名字不可以
3. 交集选择器：父级 元素.classNme ||父级 元素#idname
4. 并集选择器：以“，”隔开。例如：P,DIV,A
5. 伪类选择器： a:link{}位访问初始状态

a:hover{}悬停状态

a:active{}按下时的状态

a:visited{}被访问后状态

1. a标签的字体颜色无法继承；块级元素高度。关于文本的属性基本都可以继承
2. 优先级： id > class > 标签 > 继承 > 浏览器的默认属性
3. Font字体：font-size 、font-family 、font-weight（normal、blod粗、 bloder 特粗、lighter、100的整数倍越大越粗） 、font-style（italic斜体）

font:粗细、字号/行高、字体

1. 文本

①文字颜色color

②字间距letter-spacing（每个字母或者每个汉字，允许负数）

③单词间距word-spacing（汉字使用是要人为区分词组）

④行高line-height

⑤文本装饰text-decoration none 去下划线

underline下划线

overline上划线

line-though删除线

⑥水平对齐方式text-align

⑦垂直对齐vertical-align：middle

⑧首字母缩进text-indent

⑨自动换行white-space

⑩空白符处理white-space normal正常

pre预格式化

nowrap强制不换行

1. table属性：border-collapse:collapse;去掉表格间距
2. magin

①margin-top：上下挤压，（只取最大值）

②margin传递：为父级加border-top: 1px solid red; 或者 padding-top: 1px;

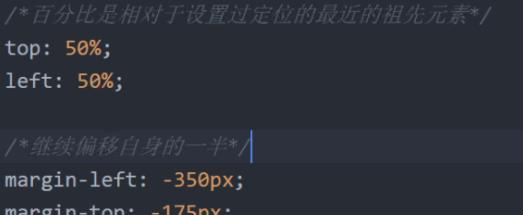
※为保证父级盒子大小不变，无论是加border还是padding都要减去高度。

1. 居中

①margin：0 auto；居中（display：inline-block和float使其失效）

②padding居中，调整left和right值

③定位居中：百分比是相对设置过定位的最近的祖先元素，继续偏移自身一半



④translate居中

1. 排在一行的方法:float 、display(有间隙)、flexbox

float：脱离文档流不占位置。

1. 去掉无序列表点：list-style：none
2. align是行内元素居中

HTML布局：由外向里，从上到下，从左到右

1. 加一些小图片，类似logo，或标记的。用background加图片做然后缩进

如图：

背景图片做出来 使用缩进text-decaration

19.页面所有元素都是左上角出现

20.overflow溢出： hidden超出隐藏、

scroll隐藏一直以滚动条展示、

visible默认、

auto自动添加滚动条，超了有没超没有

21.行内元素不能设置宽高（input、image属于特例）

22.position定位

①relative相对定位：相对自己之前位置定位、一般设置一对（XY轴）、属 于标准流

之后

之前

②absolute据对定位：脱离文档流。

1. .祖先元素无定位，以body为标准
2. 祖先元素有定位（除了static），以最近祖先元素为标准
3. 父级relative子级可以直接absolute相对于父级定位。

③fixed固定定位，例子：导航

23.一旦使用padding撑开内部间距。想着已经改变了盒子大小。但其父级大小不必考虑

24.font有顺序：粗细、倾斜、字号/行高、字体；

1. ；两个img放在一起有间隙，可以通过浮动解决

捕获

1. ①float：一行显示，支持宽高，默认内容撑开宽高，脱离文档流，提高层级

②行内元素：

③块级元素：

1. 清除浮动：

①浮动影响；

1. ：背景不能显示 （2）：边框不能撑开 （3）：margin 设置值不能正确显示(即为居中失效)

②清除浮动

1. 给父元素设置height
2. 父级添加overflow：hidden；
3. 给父级添加伪类选择器

.clearfix::after{

　　　　　　centent:"";//设置内容为空

　　　　　　display:block;//将文本转为块级元素

　　　　　　clear:both//清除浮动

　　　　　}

1. 添加兄弟空标签

.fixfloat{

clear:both

}

（5）在使用float定位时候，margin-bottom无效，它默认是左上

1. 解决input和button之间间隙：代码写一行
2. button上添加icon：在button标签里加i标签
3. before和after必须写content但是可以不写内容。
4. 块元素没设置宽与父级一样宽
5. 使元素消失隐藏的方法

①visibility：hidden、visible占位隐藏和显示（例子：登录错误提示）

②display：none；元素隐藏不占位。

1. 层级z-index:一定要与定位一起用。没有定位时最好用restive，他是标准流。 从0开始无限接近1
2. 在设置border时，边框重合问题margin-left：-边框宽度。
3. cursor设置鼠标样式
4. outline:none;去除textarea、input、select、button默认样式
5. 在使用position定位或者float定位时尽量，使用左上left、 top

39.z-index只能用在absolute

# jQuery

前言

1. 引入jQuery方法：cdn、本地
2. jQuery特点：体积小功能丰富，使用简单，兼容性好

## 一、选择器

基本选择器

1. id、class、元素

2. 子选择器：$(“父级>子级”)

3.相邻下一个兄弟节点选择器：$(“兄弟a + 兄弟b”)

4.符合条件的所有兄弟：$(“兄弟a ~ 兄弟b”)

jQuery基本选择器

1. 从零开始 :even偶数

兄弟之间

:odd奇数

1. 从零开始 :eq(n) 任意选取 只计同类的元素
2. :not(selector) 排除

属性选择器

1. 属性选择器：$(“[key = value]”)
2. 单独value或者value-开头的： $(“[key |= value]”)
3. 以value结尾的：$(“[key $= value]”)
4. 以value开头的：$(“[key ^= value]”)

子元素选择器：与css3伪类选择器相同（从1开始）

1. :first/last-child
2. :nth-child() 计算所有类型的元素，支持表达式，2n、2n-1
3. :nth-of-type() 同类元素选择

## 二、属性

1. css操作：

css({

“width”:”50px”,

“height”:”50px”

})

注意要有大括号

1. addClass(“name”) 添加已经写好的类，可以添加多个类，中间用空格隔开
2. removeClass(“name”) 移除样式
3. toggleClass(“name”) 有类就删除没没有就添加
4. 获取属性值： css(“key”)
5. 获取多个属性：css([“name1”,”name2”,”name3”]); 返回一个json
6. jQuery的链式调用：jQuery中的方法每次都返回符合条件的jQuery对象本身

## 三、DOM操作

1. dom操作 === 节点操作 === 标签操作 === 元素操作

插入：动态添加节点

①添加子节点

1. e.append() 再选中的元素e里面的尾部 添加元素

例：$(＂div＂).append(＂<p></p>＂);将p添加到div里的尾部

1. e.appendTo() 把将要添加的元素e添加到其他元素里

例：$(＂<p></p>＂).appendTo(＂div＂);将p添加到div里的尾部

1. $(＂<p></p>＂)创建元素，或者创建jQuery对象
2. prepend() 在元素的里面的前边添加元素
3. prependTo() 把元素添加到某个元素里面的最前边

②添加兄弟节点

1. e.after() 在选中元素后边添加兄弟节点

insertAfter() 将兄弟元素添加到选中元素后边

1. e.before() 在选中元素前边添加兄弟节点

insertBefore() 将兄弟元素添加到选中元素的前边

③添加元素内容

1. html() 与JS中的innerHTML一样

删除元素

1. e.remove() 删除元素，彻底消失了
2. e.empty() 删除选中元素所有子节点，保留本身
3. e.detach() 保存已经删除的元素，以备日后使用

包裹

1. e.warp() 给符合条件的元素，每一个都添加一个wrap 在元素外部包裹
2. e.warpAll() 给所有符合条件的元素外添加一个包裹 整体包裹
3. e.warpInner() 给元素的内容添加一个包裹 包裹内容

替换

1. e.replaceAll(目标) 选中的元素e被替换成……
2. 目标.replaceWith() 括号里的被E替换掉

## 四、事件(传递参数为function，直接名字)

鼠标事件

1. 鼠标单击 click(function（）{})
2. 鼠标双击 dblclick（） 双击时会先触发单击
3. 鼠标移入 mouseover（）
4. 鼠标移出 mouseout（）
5. 鼠标移入移出合并

hover（function（）{移入}，function（）{移出}）

注意：没有移入时用null占位，只写null就可以。

1. 鼠标右键单击 contextmenu（）
2. 整个页面 $(document)
3. 全局函数 $(document).ready(function(){内容})

键盘事件

1. 键盘按下 keydown() 识别所有
2. 键盘按下 keypress() 识别字母和数字
3. 键盘弹起 keyup() event.which判断哪个键被按下

事件操作

12.已存在或自定义事件绑定 on(＂click＂，function（）{})

13.触发自定义事件 trigger（＂事件名字＂）

14.接触事件绑定 off（＂eventname＂，函数名字） 解除指定函数

注意：①只写eventname解除所有同名事件②不传参数off（）解除所有事件

1. 阻止默认行为event.preventDefault
2. 阻止前辈元素事件event.stopPropagation()

form操作

1. val（）：获得form中input、select、textarea的值
2. 表单选择时候一般用：属性选择器
3. focus（）：获得焦点
4. blur（）：失去焦点

事件委托

1.$(“ul”).on(“click” , “li” , function(){})

2.特性操作：添加attr（key）

设置attr（key，value）

删除removeAttr（key）

3.克隆，jQuery可以克隆事件，clone（true）

## 五、Ajax

1. ①get方式通过URL提交数据，数据在URL上可以看见，post放在 HTTP包的body中

②get浏览器对URL长度有限制，post没有

③服务器取值不一样，例如PHP中get使用$\_get变量值，post 使用$\_post

1. $.ajax( {

method: “” , //请求方法

url: “” , //请求地址,直接填写路由即可

data: “” , //发送到服务器的数据，自动把json或array转成简直对

success: function ( data, textStatus, jqXHR ) {

此函数为ajax是否交互成功、判断status

//执行内容

},

error: function ( jqXHR, textStatus, error ) {

ajax交互失败、判断status

//执行内容

}

} )

3.表单提交，submit（）手动提交事件

# JavaScript

1. 作用域：只以function区分。

注意：undefined和函数同名以函数显示

全局和局部同名，局部使用局部

## （一）ajax

1. 定义：客户端和服务端进行异步数据交互，在无需刷新整个页面的 情况下实现局部数据更新的技术
2. 使用步骤

①创建xhr对象

var xhr = new XMLHttpRuquest()

②准备向服务器发送http请求

open（get, ”1.txt”, true）

参数 method：get、post

URL：请求的服务地址

async:是否异步（默认true异步）

异步：非阻塞模式

同步：阻塞模式

③发送请求send（）

注意：get数据存储在URL中传输

数据格式URL?key1=value1&key2=value2

④用于时刻准备服务端的响应

xhr.onreadystatechange = function(){

if( xhr.readyState === 4 ){

if( ( xhr.status >= 200 ) && (xhr.status < 400) ){

xhr.responseText;

}

}

}

1. 优缺点

①优点：无需刷新页面、异步通信、降低服务器压力

②缺点：浏览器没有历史记录(无法后退)、对搜索引擎不友好

## （二）数字类型装换

4.数字类型装换

1. number：装换成数字
2. parseInt可以设置转换进制。只取整数位
3. parseFloat 保留两位小数输出

## （三）循环

5.跳出循环

1continue

2break

3break out跳除指定循环；out为循环标记。也就是名字

在指定循环前加out：

## （四）运算

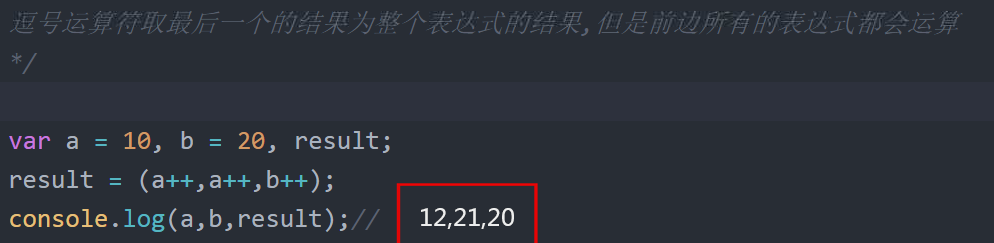
6.逻辑运算

与或非

++d后输出

d++先输出

逗号运算符：取最后一个值，但是所有都计算；



7.关系运算符

数字比较：比大小

数字字符串：比大小

字符串：比asc码

NAN：比较永远是false

## （五）数组

8.数组

⑴push（）在数组末尾添加一个或多个元素。

⑵delete arr[i] 删除元素，但是占位

⑶pop（）删除最后一个，长度减一。

⑷unshift（）添加在数组最前，可以添加一个或多个

⑸shift（）删除数组最前

⑹for（var x in arr）遍历属性,x为遍历结果

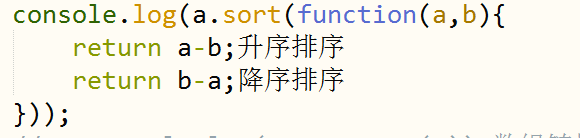
⑺join（）数组拼接，传入参数代表连接符号

⑻reverse() 颠倒数组位置

⑼引用：对象和普通类型

⑽sort()数组排序，默认按照字符编码排序，从小到大

按照数值排序



⑾concat()连接数组，返回新数组。

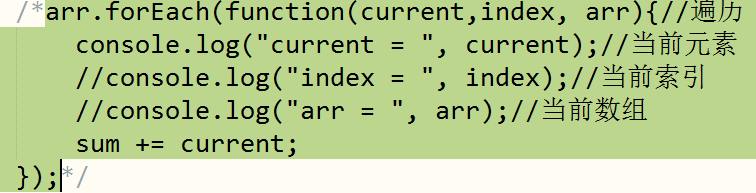
⑿slice(start,end) 对数组进行切割，返回新数组。包含start不包含end对数组内容进行浅复制（引用类型），最后一个为-1

注意这是浅复制,当切割范围有引用类型时,改变得到的新数组,原数组会发生变化;

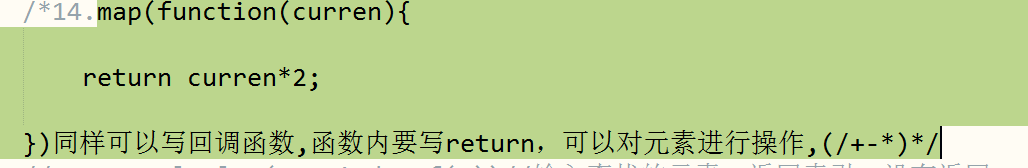
⒀splice（start，delete，add…）增删改，删除的元素组成数组。从哪个位置开始添加。

⒁toString()数组装换成字符串

⒂forEach(callback [,thisArg])遍历数组



⒃map（callback[，thisArg]）



⒄indexof()查找。输入查找的元素，返回索引，没有返回-1

⒅filter(callback) 回调函数，true的留在新数组，不影响原数组

稀疏数组，undefined

⒆every(callback) 所有都是满足条件，返回true，否则false

⒇some(callback) 有满足条件的，返回true。都不满足，返回false

Reduce、find、findindex、includes

## （六）event对象&&属性 事件

9.事件

（1）移除事件：btn.onclick=null；

（2）事件注册（绑定）

①Element.ontype = function(){}

缺点：同一个事件只能注册一个事件处理程序。

移除一个事件程序不方便

②一个事件可以注册多个处理函数，按照注册顺序执行

Firefox、Chrome：

添加事件 Element.addElementListener(“click”,function(){},flase)

删除事件 element.removeElementListenter(“click”,fnname,false)

※注意：移除必须有名字

IE：

添加事件 element.attachedEvent(“onclick”,fnname)

删除事件 element.detachedEvent(“onclick”,fnname)

※注意：有on，执行顺序和绑定顺序相反

10.event对象属性

（1）event兼容处理：var ev = ev||window.event;

（2）在事件处理函数中，this完全等价于event.currentTarget,指向的都是注册该事件处理程序的元素。

（3）event.target || srcElement在处理函数内部，指向冒泡源头

（4）键盘输入：event.code 都是大写，并且字母数字前加key

event.key 键盘按啥输出啥，区分大小写

（5）event.screenX/Y 屏幕左上角开始计算位置

（6）event.clientX/Y 浏览器内容可视区左上角开始计算

（7）event.pageX/Y 可视区左上角计算，包括滚动距离

11.事件event方法

（1）阻止事件传递：event.stopImmediatePropagation()后续事件都不能执行

（2）阻止默认事件：event.preventDefault() ||return false //IE

（3）阻止冒泡：event.stopPropagation() ||cancelBubble = true //IE

12.右键事件contextmenu

13.touch事件(DOM2级事件)

（1）touchstart触摸开始事件

（2）touchend触摸结束事件

（3）touchmove 触摸过程事件

14.onclick在IE低版本中没有。Currenttarget

## （七）DOM操作

1.nodeType:只读，节点类型

①元素：1

②document：9

③文本：3

④属性：2

⑤注释：8

2.获取节点

①document.getElementById();

②document.getElementsByTagName();

③document.getElementsByClassName();

3.节点操作

I获取节点

（1）父节点

①获取父节点：ele.parentNode

（2）兄弟节点

①nextSibling将文本也是别为兄弟

②nextElementSibling 只将标签视为兄弟节点

（3）子节点

①获取任意子节点

childNodes:子节点列表，获取所有子节点，不包括孙子节点，标准浏览器包括文本节点和非法嵌套的节点，IE7以下只包括元素节点不包括非法嵌套。

children:子节点列表集合，只包括元素节点。

II节点操作

1. 创建节点

createElement(tagname)

ele.innerHTML

（2）插入节点

f.appendChild(s)插入到最后

f.insertBefore(one,old)在子元素old前插入one

如果第二个参数为null，功能和appendChild一样

1. 删除节点

f.removeChild(s)

III获取节点

第一个子节点：firstChild firstElementChild

最后一个子节点：lastChild lastElementChild

上一个兄弟节点：previousSibling previousElementSibling

下一个兄弟节点：nextSibling nextElementSibling

IV补充

节点名字：nodeName 他是大写的 可以用做判断

事件代理（事件委托）：高效代码、使用冒泡、当所有子元素都注册相同事件

Onchenge事件：适用input、select、textarea当值改变时触发

4.克隆节点

（1）HTMLCollection是一个接口，列表集合，根据文档结构自动更新。

遍历列表的属性和方法。类似input

1. 被克隆e.cloneNode(true)。克隆所有子孙节点，不可隆事件

e.cloneNode()只克隆当前节点

5.操作属性

（1）获得属性：getAttribute( key )利用自定义属性在标签上存储数据

（2）设置属性：setAttribute(key,value)

（3）删除属性：removeAttribute(key)

6.设置样式

（1）style

（2）cssText

（3）className

※特殊命名cssFloat

（4）window.getComputedStyle（id）获取所有属性集合

获取单个window.getComputedStyle.width;

7.闭包的解决：for循环 内事件传不进去I 不能按索引找到

（1）立即函数：隔离作用域，避免全局变量冲突问题，定义并且立刻调用，并且只调用一次，( function(i){} ) (i);

（2）自定义属性

8.jQuery对象是jQuery对DOM封装的结果

JS和jQuery不能混用。

jQuery 🡺 jsDOM :e.get(0) 从零开始，不是nodeType

jsDOM 🡺 jQuery : $(e)

## （八）字符串操作

（1）创建var str =“” 或 var str = new String（）

（2）str[ i ] 可读不可写。

（3）str.charAt( index )读取字符串中字符

（4）字符串有length，可以遍历

（5）charCodeAt( index ) 获得字符编码

（6）String.fromCharCode() 将编码转为字符

（7）str.indexOf(“iphone”,20) 从20开始找iphone，区分大小写

默认从0开始，返回第一个字符位置，没有则返回-1

（8）slice(start,end) 和数组一样，正负

（9）substr(start,length) 截取 -1最后一个

（10）substring(start,end) 不接受负数，start到end-1

（11）toLowerCase() toUpperCase()大小写转换,全变

（12）str.split(“s”,2) 返回数组,以s切割，在0-2范围内

## （九）定时器

（1）setinterval（callback，delay）间隔定时器

间隔定时器延迟时间，是指多久后去排对。不会立刻执行，

间隔定时器无法保证代码的运行时间。

如果代码的运行时间要大于间隔时间，那么在事件队列中就不断的有该任务去排队，

导致事件队列中会有很多该任务！！！！！

（2）clearInterval(timer)

（3）setTimeout(callback,delay) 只执行一次

（4）所有同步时间执行完，才会执行异步事件（定时器）

（5）使用定时器，先清除var timer = null；clearInterval（timer）

## （十）时间date

（1）创建时间对象：new Date()

（2）获得当前时间

①年：d.getFullYear()

②月：d.getMonth() 从零开始

③日：d.getDate()

④星期：d.getDay()

⑤时：d.getHours()

⑥分：d.getMinutes()

⑦秒：d.getSeconds()

⑧毫秒：d.getMilliseconds()

⑨累计：d.getTime()

（3）设置时间 new Date(“2017-11-11 11:11:11”)

其他都有，年月日时分秒

## （十一）math

①方法

（1）最大值Math.max(num)

（2）最小值Math.min(num)

（3）四舍五入 Math.round()

（4）向下取整 Math.floor()

（5）向上取整 Math.ceil()

（6）随机数 Math.random() 返回 [0,1)随机数

（7）数组最大最小比较 Math.max/min.apply(null,[array])

（8）科学计数法：2.13e31 === 2.13\*10^31

（9）绝对值 abs()

（10）三角函数sin() cos() tan()

（11）反三件函数asin() acos() atan()

（12）幂指数：pow(x,y) x的y次幂

（13）平方根 sqrt()

（14）log() exp()

②属性

1. PI圆周率
2. LN2 LN10
3. LOG2E LOG10E
4. E
5. SQRT1-2 SQRT2

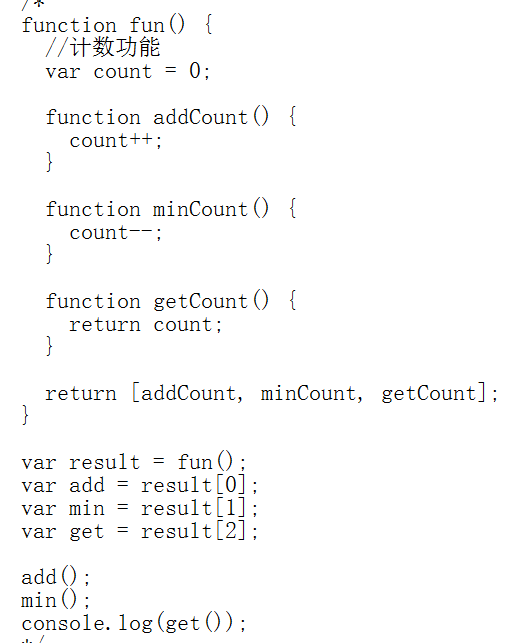
## （十二）闭包

1. 创建闭包：在函数内定义一个函数，并且该函数引用到了当前作用

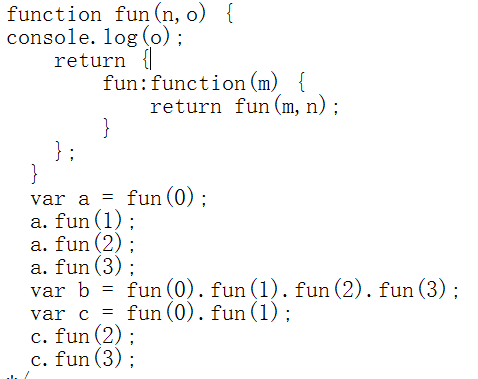
域中的变量，并且要将该函数返回

1. 闭包优点：实现变量私有化，减少全局变量的定义；

动态生成数据（根据参数变化

）

1. 缺点：闭包常量常驻内存，增加内存消耗
2. 经典闭包



## （十三）面向对象

1. 定义：所有操作都是以对象为中心，对象上拥有属性和方法，属性

和方法属于对象的附属。

1. 特点：封装，继承，多态
2. 对象的创建
   1. 使用已有的构造函数

Object构造函数时所有构造函数的祖先，其他都继承了object的属性和方法

* 1. 直接使用字面量来创建对象var obj = {}

对象访问属性可以用.也可以[]

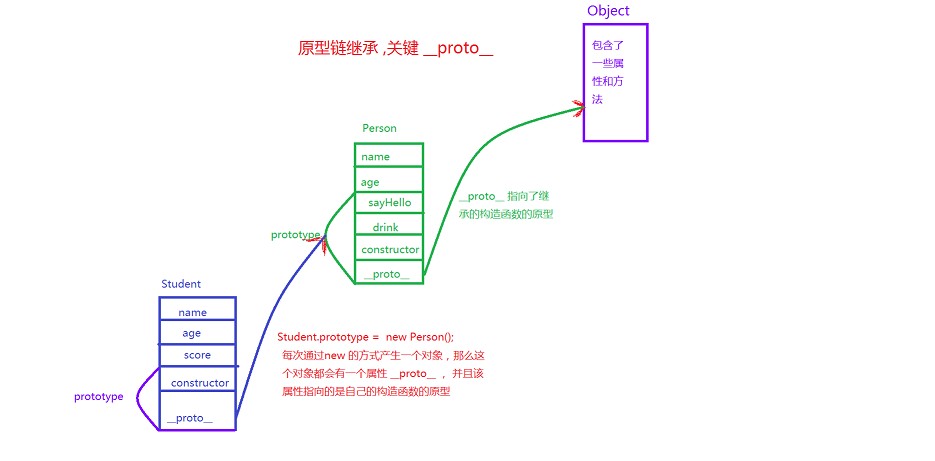
对象可以任意添加属性和方法

* 1. 使用构造函数来创建对象(批量创建)

使用new创建对象时，不要在函数内使用return，构造函数会自动返回对象本身

1. 使用new关键字时发生了什么
   1. This指向新创建的对象
   2. 执行构造函数
   3. 返回创建完成的对象
2. 添加在prototype上的属性和方法，是所有以该构造函数产生的对象所共有的！！！！！
3. constructor可以得到对象是由哪个构造函数产生的。
4. 继承
   1. 原型链继承

New产生的对象，自动有一个属性\_proto\_，该属性指向对象的构造函数。



属性和方法的查找顺序：

从构造中内部开始 -----> 自己的原型 ----> 继承来的 ----> 继承来的 ...... ----> Object

注：全局变量都是window下的

1. 函数调用
   1. 直接调用：fun() this 🡺 window
   2. 间接调用：var obj = { fn: function(){} } 调用obj.fn() this 🡺obj
2. 更改this指向
   1. call( thisObj [,value] ) 更改this指向，第二个参数开始是传递给函数的参数

可以传数组，认为是整体

* 1. apply( thiObj,[] )功能和call一样，只不过参数通过数组或伪数组传递

注：只要使用call或者apply就可以在伪数组上使用真数组方法

Array.prototype.slice.call（arguments，0,3）；

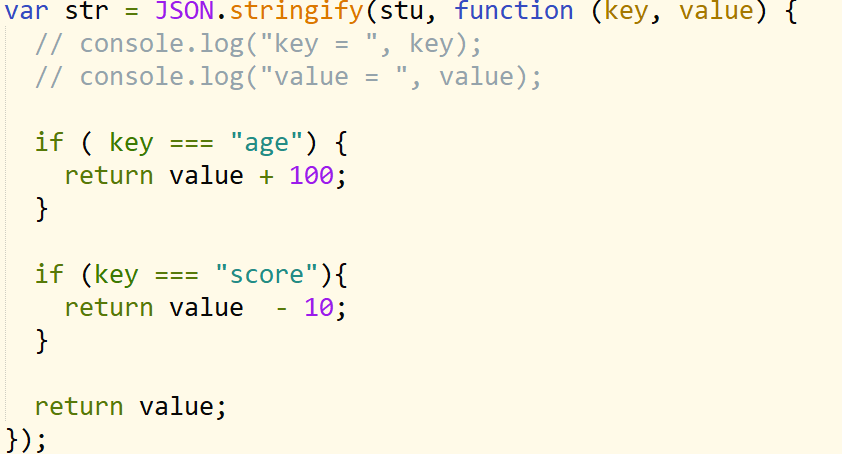
1. bind函数绑定
   1. bind更改指向，提前传参，函数一旦bind，返回的this指向就固定了

var fun = fun1.bind（object，10,20）

* 1. 在调用fun时传参是在20之后，作为额外参数
  2. 可以bind多次

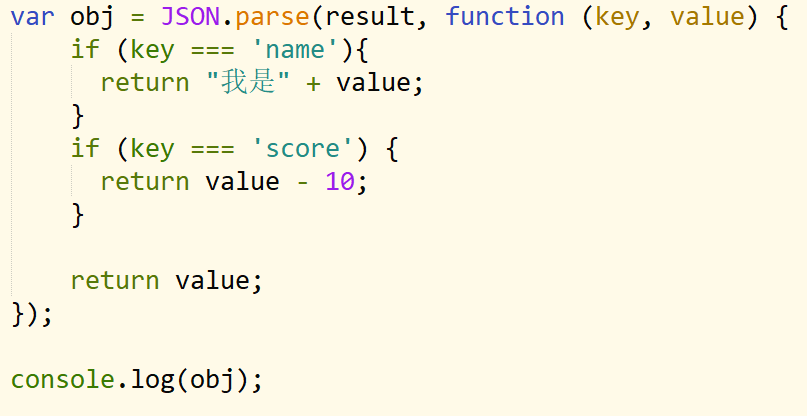
## （十四）JSON

1. 轻量级数据交换格式
2. JSON.stringify(jsonObj [,replacer [,space]]) 序列化，将json转成字符串
   1. replacer可以是函数，函数需要返回value，传入参数key,value



* 1. replacer如果传入数组，则按照数组中key输出
  2. space是空格数量\t进行缩进

1. JSON.parse(string[,fun(){}]) 反序列化，字符串转JSON



1. XML



## （十五）正则表达式

1. 定义
   1. var re = //
   2. var re = new RegExp(//,’g’)
   3. var re = new RegExp(‘’,’g’)
2. re.test(str) 返回Boolean
3. str.match(re) 返回数组，捕获：便于提取匹配项中指定部分

“applegoolechrome”.match(/(apple)(goole)(chrome)/)

* 1. 如果正则表达式不是全局匹配：

元素：

0: 正则表达式的整体匹配项

从下标1开始，全是每个捕获项(就是小括号中的内容),小括号的出现顺序，决定捕获先后

index: 表示匹配项第一次出现的位置

input: 字符串本身

* 1. match进行全局匹配，标志位g

返回值是包含所有匹配项的简单数组，和数组长度

1. re.exec(str) ,只匹配一次，找到返回一个数组，没找到返回null

每次调用接着上一次之后查找



1. str.replace(“”,””)
   1. 第一个参数是字符串，只会替换第一个
   2. 第一个是正则表达式，按照正则表达式来
   3. 第二个参数是function（match,p1,p2…,offset,string）

Match会被返回值替换，他是正则匹配到的

P1…..各个子串

Offset匹配到的第一位所在原来的位置

String原字符串

1. 特殊符号
   1. 点，匹配除\r\n以外字符串
   2. \转义字符，不是本身含义，拥有了特殊含义
   3. \d数字 \s字母(包括\r\n) \w字母数字下划线
   4. \D\S\W都是非的意思
   5. []连续的意思，占一位[a-z]
   6. [^] 当^写在[]才是非
   7. 边界匹配^以什么开头 $以什么结尾
   8. \*前边的字出现任意次（包括零次）

贪婪匹配，最长匹配

* 1. +前边字符最少出现一次
  2. ？前边字符最多一次0,1 最短匹配
  3. {m,n}前边字符出现m-n次
  4. xxx|yyy 左右两边是整体内容，出现XXX或YYY

1. 标志位
   1. g全局匹配
   2. i忽略大小写
   3. m多行匹配

补充：this指向

如果直接将一个函数作为 事件处理函数，那么只是将该函数的代码放到此处而已，和函数原来的this没有关系完全等价于 第一种方式的代码btn.addEventListener('click', obj.handler, false);

## （十六）JavaScript高级用法

1. 函数柯里化：实现参数复用，延时计算，提前返回
2. 函数缓存：如果代码的耗时时间比较长，那么可以使用缓存。如果时间比较短，没有必要了

函数内创建一个对象，用来存储结果。

1. 函数节流：就是不让一个函数连续重复的执行，要打断它的执行

使用定时器实现节流，连续生成使得浏览器卡死。

## （十七）BOM

1. Offset
   1. offsetWidth == width + padding + border 只读
   2. clientWidth == width + padding 只读
   3. offsetLeft [Top] == 最近父元素之间的距离 ——可读可写
2. scrollLeft scrollTop 可读可写，通过改写设置滚动条位置
3. scrollWidth scrollHeight文档宽高
4. 滚动条事件onscroll
5. location对象

(1)

* 1. location.hash
  2. location.hostname
  3. location.pathname
  4. location.port---端口号
  5. location.serch
  6. location.protocol ---http/https

(2)页面跳转

a) location.assign(“url”)

b) location.href = “url”

c) window.location = “url”

d) location.replace(“url”) 没有历史

e) location.reload() 重新加载

6.navigator对象

7.history

# HTML5

## 一、canvas

< canvas ></canvas>

1.获得画布dom对象(getelement)

2.canvas元素有width和height属性

3.获得画布上下文对象 cxt = can.getContext(“2d”);

4.开始绘制图形

①cxt.beginPath();

②cxt.moveTo(x,y) 开始位置

③cxt.lineTo(x,y)直线的一个点

1. 描边cxt.stroke();
2. 填充cxt.fill();
3. 样式
   * 1. 颜色cxt.fillStyle = “red”;[strokeStyle]
     2. 线条宽度cxt.lineWidth = 5;

8. ctx.fillRect(x,y,width,height) 绘制填充矩形

9.cxt.strokeRect(x,y,width,height) 绘制描边矩形

10.关闭画布cxt.closePath()

11. 擦除矩形ctx.clearRect(x,y,width,height)

注：重新设置画布的宽高，可将画布清空

12. 绘制圆弧ctx.arc(x,y,r,sAngle,eAngle,counterclockwise),其中(x,y) 用于设置圆弧的圆心坐标， r为圆的半径，sAngle为开始弧度，eAngle为结束弧度，Math.PI == π

counterclockwise为true或者false决定是否逆时针，默认false。

13.绘制文本

* + - * 1. cxt.font = “50px 宋体”
        2. cxt.textAlign = “center”;
        3. cxt.textBaseline = “middle”
        4. cxt.fillText(“我喜欢钱”，200,200) 【strokeTextl镂空】

14.绘制图片

1. 获得图片img = dg
2. img.onload = function(){ //保证图片加载完被绘制

context.drawImage(img,sx,sy,swidth,sheight,x,y,width,height);

对图片裁剪，并在画布上定位被剪切的部分

sx,sy 裁剪的以图片左上角为参考的坐标，

swidth：裁剪图片的高度。

sheight:裁剪的高度

x,y canvas的坐标

width, height: 设置图片最终显示的大小

}

15.cxt.clip();按照画好的图形进行裁剪画布，裁剪后只有该区域可以绘

制图形

16.扇形

要有cxt.moveTo()作为起始点，配合cxt.arc()

17.画布渲染画布

cxt2.drawImage(can1, 67, 71, 110, 50, 67 \*2, 71 \*2, 110\*2, 50\*2);

//从已有的画布中抠图

18.将画布导出为图片

can.toDataURL(type, encoderOptions) //base64编码格式图片

* 1. type默认image/png
  2. ②encoderOptions在指定图片格式为 image/jpeg 或 image/webp的情况下，可以从 0 到 1 的区间内选择图片的质量。如果超出取值范围，将会使用默认值 0.92。其他参数会被忽略。

补充：base64格式图片优缺点

优点：图片转成字符串，不发送HTTP请求，减轻服务器压力

缺点：体积变大

使用场合：小体积图片

19.画布缩放：影响后续的绘制，画布缩放体现在绘制图形缩放

20.cxt.save()保存当前状态

cxt.restore()倒序回复save()

## 二、新属性

1.选择器document.querySelector(“.class/#id/ele”);选择第一个

2.document.queryAllSelector(“同上”);选择所有

3.classList操作

①classList.remove(“class”)删除class

②classList.add(“class1”，“class2”) 添加class，可以添加多个

③classList.toggle(“class”) 有就删除，没有就添加

④classList.contains(“class”);有是true，没有false

4.自定义属性操作方法

注：使用方法必须用data-开头

①如果data后边有多个“-”，会被换成小驼峰命名

②获取值ele.dataset.age

③修改ele.dataset.age = “lisi”;直接赋值

# CSS3

1. 子选择器：nth-child
2. 兄弟选择器
3. 伪类选择器
4. 伪元素
5. 新增属性

## （1）透明

①opcity : 影响后代所有透明度（0~1），可以改图片、文字、颜色

②rgba : 只影响自己，a代表透明度（0~1）

③hsla : 不常用

④边框透明：transparent

## （2）文本

①文本阴影 ：text-shadow:x y m color

注意：x轴偏移量、y轴偏移量、模糊度（不能为负）、颜色

可以设值多种，以逗号分开

凹进去：深色在上，浅色在下

凸出来：深色在下，浅色在上

②边框圆角化 ：border-radius

注意：长方形==>椭圆，正方形==>圆

③target

## （3）盒子

①盒子阴影：box-shadow:x y 模糊度 外延color insert(内阴影)

注意：盒子阴影box-shadow:2px 2px 8px grey

②盒子模型：box-sizing: content-box width设置盒子内容宽

border-box width设置盒子可视区宽

## （4）背景

①background-size: cover(图片填充盒子)\contain(盒子包裹图片)

background-size：Xpx Ypx;设置图片大小

②background-origin：padding-box、border-box、content-box设置原点

默认背景图片以padding开始摆放

③background-clip :背景裁剪

多背景图片逗号分开。

## （5）渐变position下

①线性渐变linear-gradient：

②径向渐变radial-gradient

## （6）过渡transition：过渡属性、过渡时间、过渡样式、延迟时间

过渡样式： ease：快到慢（默认）

linear：匀速

ease-in：加速

ease-out：减速

ease-in-out：先+后-

cubic-bezier

## 6.2D转换transform属性：标准流不脱离文档

①translate：移动（标准流，相对于自身移动）

translate（X偏移，Y偏移）

单位：px、percent、

②scale（x,y）缩放(没有单位)

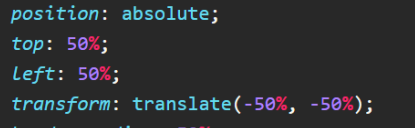
③rotate（deg）旋转

④skew(Xdeg , Ydeg)拉伸

## 7.3D

turn=360。

## 8.同心圆



## 9.animation：step步长

## 10.伸缩盒子flex

①display：flex在父元素上设置，表示子元素变成可伸缩的。在一行

②flex-grow子级扩展比例

③flex-shrink收缩比例

④flex-direction排列方向column纵轴方向

⑤flex-wrap父级设置，多余部分换行

* + - * 1. justify-content

## 11.自适应（三栏布局中间自适应）

①float：布局：左右中

②position absolute：没有布局顺序

③flex中间伸缩flex：1

1. meta标签
2. media

# SASS

1. 定义：css预处理器
2. 变量$name

①命名规则：$开头、中划线、驼峰

②变量==>应用于属性、class、复杂样式==>#{$name}{}

③变量==>正常属性值==>$name



1. 注释 :单行// 多行/\*\*/

注释不可以出现中文

1. 多值变量(list、map)
2. list类似数组、从1开始

①一维：$name:5px 6px 7px;

②二维：$name:5px yellow,6px blue;

$name:(5px yellow) (6px blue)

以“，”或者“（）”两种形式分开

③取值：nth($name,index)、二维嵌套

nth(nth($name,1),1) ==> 取第1-1

1. map

①$name(key1:value1 , key2:value2)

1. 嵌套
2. 选择器嵌套：父子逐级嵌套、&代表父元素

.one{

.one\_1{

.one\_1\_{}

}

&：hover{}；

}

1. 属性嵌套

$name{

border:{

style:solid;

color:yellow;

width:1px;

}

}

1. @at-root跳出最外层（media跳不出去）

6.

# Bootstrap

## 准备

1. 移动设备优先，响应式布局、前端框架。
2. 引入方式：本地和CDN
3. readme、dist
4. hack
5. normalize.css样式重置表
6. 依赖jq
7. 使用方法：引入JQ、bootjs、bootcss
8. 两个hack第一个支持HTML5，第二个支持媒体查询

## 基础学习

1. 容器（class）

①container-fluid：流体布局，没有版心

②container：固定布局、有版心（1170、970、750、auto）

③阈值：1200、992、768

注意：容器都有padding，尽量不要嵌套

④！important提高优先级

1. 栅格系统（class）

①一行row、列col

②一行12、占满换行、没占满留空白

③col-lg-\* 1200 large

col-md-\* 992 middle

col-sm-\* 768 small

col-xs-\* 最小 extra small

④组合模式：BS官网

1. 列偏移（offset）

①1-12、超出、小于无作用

②col-[lg\md\sm\xm]-offset-\*

注意：当没设置宽度时，默认以12-偏移量为宽

1. 排序（pull、push）

①pull向前、push向后

pull-right右浮动

②排序和偏移区别：offset只能向后

偏移当行内有多个元素，偏移量加宽度超多12，将会换行

排序会同行超出。

1. 嵌套（行内嵌套）

①按照父级分配、不是容器

②遇到浮动问题：清浮动<div class=”clearfix”></div>在浮动列上方

## 响应式工具

### （1）可见类

1. 显示visible—[lg\md\sm\xs]—[block\inline\inline-block]
2. 隐藏hidden—[lg\md\sm\xs]—[block\inline\inline-block]
3. 天猫侧边栏
4. row自动处理padding
5. 固定定位affix

### （2）打印类

1.在打印模式下显示出来

2.显示：visible-print-[block\inline\inline-block]

3.隐藏：hidden-print-[block\inline\inline-block]

## 字体图标

1. 有点：减少请求，易于控制
2. 使用方法：class引入，通过font-size color控制
3. 图标自己单独一个标签，内部不要添加内容
4. font-face在css源码修改字体引入路径
5. 图标转web字体：AI画导出SVG格式，导入icomoon，在css中设

置class

## 预定义样式（都是改颜色）

1. 首选项primary 蓝色
2. 成功success 绿色
3. 一般信息 info 浅蓝色
4. 警告warning 橙色
5. 危险danger 红色
6. 文字颜色text-\*
7. 按钮颜色btn btn-\*
8. 段落颜色bg-\*
9. 警告框alert alert-\*
10. 面板 panel panel-\*
11. 登录框实例

## 按钮

1. 使用时必须加btn，全加
2. 样式：默认+预定义
3. 样式：btn-defaul、btn-link
4. 大小：btn-[lg\ sm\ xs]默认md

注意大小不止用在btn、其他都可以

1. 状态：active按下（选中）、disabled禁止

注意：普通hover与active一样、按下与有active移入一样

6.按钮种类：a button input（在这些上添加btn类）

7.块级：自适应父级btn-block

8.按钮组btn-group按钮连在一起

9.btn-group-justifie：适应父级

10.btn-group-vertical竖着排列，父级不加btn-group，否则会有圆角不对称。

11.下拉表单<span class=“caret”></span>。dropup箭头向上

注意：加图标时input有问题

1. github按钮组
2. role Aria辅助属性帮助特殊人群

## 下拉菜单

1. 写法

<div class=”dropdown”>

<button class=”btn btn-default dropdown-toggle” type=”button” data-toggle=”dropdown”>

<span class=”caret”></span>

</button>

<ul class=”dropdown-menu”>

<li><a href=”#”>1</a></li>

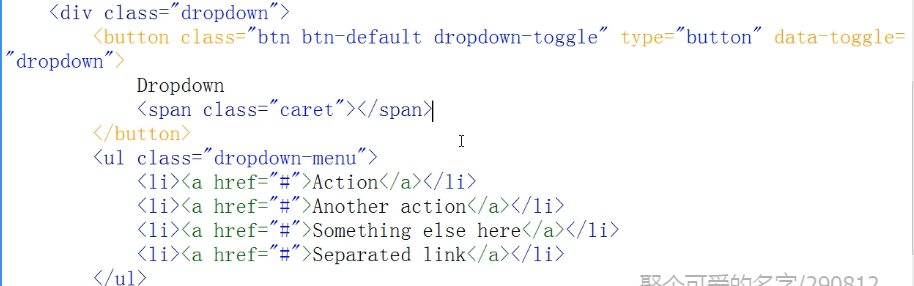
<li><a href=”#”>2</a></li>

<li><a href=”#”>3</a></li>

<li><a href=”#”>4</a></li>

</ul>

</div>



1. open设置下拉展开
2. dropdown-pull-right
3. 下拉菜单默认不占位置，脱离标准流，会覆盖下一行
4. btn-block可以设置ul适应父级
5. ul分部分li class=dropdown-header
6. ul两部分分割线class=divider
7. 选中 active；禁用 disabled
8. 父级dropdown改成dropup下拉菜单朝上
9. 文字居中text-center
10. 应用：运动且下拉菜单
11. data-target=“classname”设置控制的下拉菜单
12. 扩展：自定义属性，js写调用接口（视频12）

## 选项卡

### （1）头部

1. 写法

接口

<ul class=”nav nav-tabs”>

<li class=”active”><a href=”#” data-toggle=”tab”></a></li>

</ul>

1. ul默认自适应父级、可以设置自身width
2. 按钮li nav-justified按照父级等分
3. 小于768 变一列
4. 线自适应li nav-tabs-justified
5. nav-pills设置有颜色按钮 和 nav-tabs一样
6. nav-stacked竖状
7. 可以和下拉菜单结合使用

### （2）内容

1. 写法

<ul class=”tab-content”>

<li class=”tab-pane”><a href=”#” data-toggle=”tab”></a></li>

</ul>

1. 内容里的li要与头部li数量对应
2. 头部控制内容切换：锚点。头部控制内容id
3. 内容淡入淡出方式fade，默认开始隐藏，显示的话fade in
4. 移入移出操作 jQuery调用接口show



## 导航条（navbar）

1.

# NodeJS

## 1.定义：Node.js异步非阻塞编程语言

a)同步：获得数据 = 请求一个数据（），代码顺序执行

b)阻塞：获得数据 = 请求数据（数据有立刻返回，没有阻塞在赋值操作）

c)异步：获得数据 = 请求一个数据（），不必等待

d)非阻塞：获得数据 = 请求数据（数据有立刻返回，没有立刻返回错误）

## 2.npm使用

npm install xxxx

npm install xxx@version

npm uninstall xxxx

npm install -g xxx

npm uninstall -g xxx

npm updata xxx

npm install xxx@latest最新版

npm ls

npm ls -g

npm –help帮助文档

npm init –yes 产生一个package.json文件

npm install jquery –save 将jQuery作为项目依赖填写到package.json

npm install jquery -S 同上

npm install xxx –save-dev 将xxx作为开发者工具类的模块依赖

npm install直接安装package.json中的依赖模块

## 3.创建http服务器

var http = require(“http”);

var server = http.createServer(function(req,res){

res.writeHead(200,{‘Content-Type’:”text/html”});

res.write(“<h1>I am server…</h1>”);

res.end();

});

server.listen(8000,function(){

监听8000端口

})

## 4.require是nodeJS的全局函数，用于引入其他模块

var fs = require(“fs”);

①异步读取

fs.readFile(“./nvm.md”,”utf8”,function(err,data){

回调函数，只有文件读取结束才会执行

})

②同步读取

var data = fs.readFileSync(“./nvm.md”,”utf8”)

## 5.require

a.引入一个事件模块

var EventEmitter = require(‘events’);

b.创建事件发射对象

var EventOne = new EventEmitter();

c.事件注册(on连续触发,且可以绑定多个事件，可以添加多个回调函数)

EventOne.on(‘custom\_event’,function(obj){

移除回调函数

this.removeListener(‘custom\_event1’,handler)

})

d.事件触发

setTimeout(function(){

触发事件并传递参数

EventOne.emit(‘custom\_event’,{value:100});

移除事件所有函数

eventOne.removeAllListener(‘custom\_event1’)

移除所有事件及所有函数

eventOne.removeAllListeners();

},1000);

## 6.once(事件触发一次)

eventOne.once(‘’,function(value){}) 通过once注册事件，只触发一次

7.eventOne.listeners返回一个处理函数组成的数组

8.\_filename文件绝对路径

\_dirname文件所在文件夹绝对路径

9.global全局变量，任何模块都可以使用

10.setImmediate(function( value ){

立即定时器，插队执行，默认执行一次

},10)

11.清除定时器clearImmediate(timer);

## 12.process进程

①process.argv获取命令行参数，是一个数组

第0个，node.exe的绝对路径

第一个，文件绝对路径

第二个，开始是命令行参数

process.argv.forEach(function (value, index){

// console.log("index =" + index + " value = " + value);

//${name} 直接去变量name的值

console.log(`index=${index} value=${value}`);//模板字符串

});

②process.exit()结束当前程序，0表示正常退出

③process.on(‘beforeExit’ , function(code){}) 正常退出会触发，手动调用process.exit()不会触发

process.on(‘exit’ , function(code){}) 只要退出就会触发

④process.pid当前进程编号，当进程结束后，进程号就不是当前进程

⑤process.kill(9840,’SIGKILL’)给一个进程发信号，每个信号都有不同意义，可能是结束一个进程。第一个参数是进程号

⑥process.nextTick(function(){},100) 回调函数在事件循环的下一次循环中执行，插队执行

⑦process.stdin标准输入

process.stdin.setEncoding(“utf8”)设置输入的字符编码格式

process.stdin.write(‘please input >> ’);

process.stdin.on(‘readable’ , function(){有数据输入，数据可读

process.stdin.read()读取数据

})

process.stdin.on(‘data’ , function( chunk ){})读取数据

process.stdin.on(‘end’ , function(){}) 数据读取完毕

process.stdin.pipe(process.stdout) 直接将输入流中数据输出到输出流中

## 13.fs模块—都有同步Sync

（1）读取文件

fs.readFile(“URL”, encode, function(err, data){

没有错误err === null

不写编码格式默认16进制

data为读取到的内容

})

（2）读取链接

fs.readlink(link, encode, function(err, data){})

（3）读取文件夹

fs.readdir(path, function( err, files ){

files当前文件的文件夹下所有文件

})

（4）判断文件类型

fs.stat(path, function( err, stat){

stat是一个对象

stat.isFile()普通文件

stat.isDirectory()目录

})

（5）写文件：有则覆盖，没有创建

fs.writeFile(filename, content, {encoding: “utf8”, mode: 0o444, (err)=>{}})

mode设置读写权限r w x

（6）创建文件夹

fs.mkdir(foldername, (err)=>{})

（7）判断文件是否存在

mode:

fs.constants.F\_OK ——0

fs.constants.R\_OK ——4

fs.constants.W\_OK——2

fs.constants.X\_OK——1

判断文件是否存在

fs.access(path,[mode],callback)

可读可写r+

（8）文件内容追加

fs.appendFile(filename, data, options, callback(err))

data: string buffer

options: encoding mode flag

（9）打开文件

fs.open(path, flags, mode, callback(err,fd) )

flags:r读 r+读写 rs+同步读写，绕过缓存

fd: 文件数据

mode: 0o666

（10）写文件

fs.write(fd, buffer, offset, length, position, callback(err, bytesWritten, buffer))

（11）箭头函数

（arg1, arg2, ……）=>{} 箭头函数的this === window

（12）process.nextTick()添加回调函数在循环返回之前就执行，直接插队到队首

setimmediate() setTimeout()添加回调函数都输在队列尾部，执行顺序不确定。

（13）fs.close关闭打开的文件

（14）fs.unlink(path，callback(err))删除文件夹

## 14.nvm nodejs多版本管理

（1）nvm ls 列出当前系统安装的nodeJS版本

（2）nvm current列出当前的nodeJS版本

（3）nvm ls-remote列出nodeJS的所有版本

（4）nvm use xxx 使用指定的nodeJS版本

（5）nvm install xxx 安装指定的版本

（6）nvm uninstall xxx 卸载指定的版本

15.nodeJS体系架构，nodeJS事件循环机制

<http://www.ruanyifeng.com/blog/2014/10/event-loop.html>

## 16.socket.io

a.搭建socket服务

（1）下载安装socket模块在当前项目下，npm install socket.io

（2）创建node服务，http

var http = require(“http”)

var socketIo = require(“socket.io”);

var server = http.createServer(function(req, res){

进行文件读取操作

})

server.listen(8000, function(){

console.log(“监听8000”)

})

（3）创建socket服务，并且监听httpserver

var io = socketIo(server);

（4）客户端通过socket.io连接服务器时触发，

io.on(“connection”, function(socket){

①socket是本次通过socket链接的客户端

②广播发送信息给客户端，不包括此次当前的客户端

socket.broadcast.emit

③断开连接触发

socket.on(“disconnect”, function(){})

④监听客户端自定义事件

socket.on(“自定义事件名称”, function(data, fn){

data客户端数据

fn();内部函数，返回信息给客户端

})

⑤发送数据到客户端

socket.emit(“string”)

})

b.客户端

（1）通过cdn引入socket.io库

（2）创建socket对象，并连接服务器

var socket = io(“http://127.0.0.1:8000”)

（3）注册一个事件，自定义，用于接收服务端发来的信息

socket.on(“server message”, function(msg){})

（4）发送socket请求到服务端

socket.on(“connection”， function(socket){})

注：前后端通信，都是自定事件，监听同一事件就可以，emit进行信息发送。

## 17.node中的 this

（1）nodeJS代码内部直接调用，即每个模块作用域内

this === module.exports

（2）直接调用函数，函数内部this

this === global

（3）通过call指定，

this === call指定的

（4）普通定时器内部this ==== 定时器对象

箭头形式的定时器内部 this === module.exports

## 18.http模块request，response，url

a).req属性和方法

（1）req.headers头部信息

（2）req.method请求方法

（3）req.httpVersion协议http的版本

（4）req.url请求路径

b).res属性和方法

（1）res.writeHead(200, {“Content-Type”: “text/html;charset=utf-8”})

注：text/plain普通文本、application/json json格式

（2）res.write(data, encode, callback)发送数据

（3）res.end(data)结束本次通信，end也可以发送数据，数据发送

之后结束

（4）res.setHeader()设置http状态码，以及响应头部，其中头部

可以包含自定义的值

c).URL模块var url = require(‘querystring’)

（1）url.parse(req.url, true)

true是将query查询部分处理，转成json对象，默认false

自己转换querystring.parse(url.query)

注1：jpg —>base64; png—>utf-8

注2：str.lastIndexOf(searchvalue,fromindex)，

第一参数：查找的字符；第二：从字符串哪开始找

返回一个指定的字符串值最后出现的位置，在一个字符串中的指定位置从后向前搜索。

## 19.module模块

### （1）exports对象

a) exports === module.exports全局对象，不需要引入

b)用于导入数据

exports对象通过exports.add添加方法

module.exports = { add : add}通过json对象添加方法

c)require引入文件，返回一个exports对象

d)exports对象可以被重写，覆盖，此时要使用module.exports对象

（2）模块内部测试代码，只在本模块内运行，其他require引入不能运行

require.main === module 当前模块直接调用时true，被require引入时false

require.main.filename === \_filename作用同上。

（3）在某一个模块中，重写、覆盖module.exports对象时，在其他模块中不能操作，已经破坏引用关系。