2018.3.5

1. 安装gitbash
   1. 下载

<https://share.weiyun.com/851ab18164c2c42f8a6bda71ba33cf77>

* 1. 设置用户名和邮箱



$git config –list可以检查是否配置成功

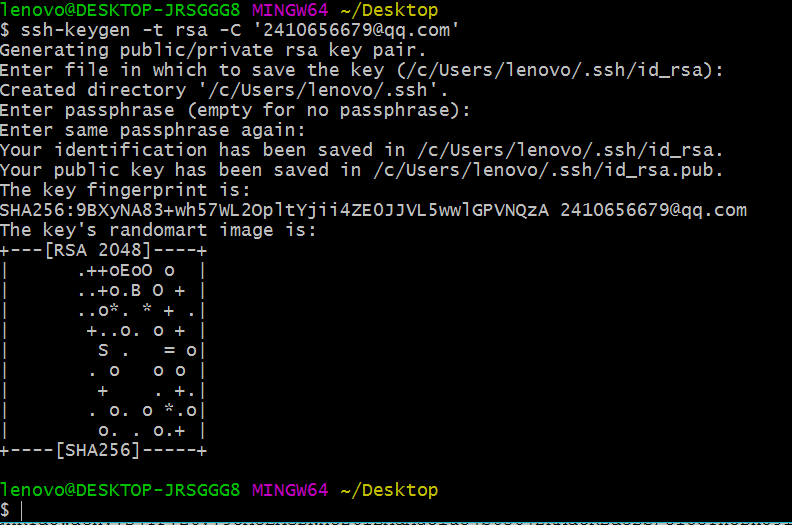


* 1. 使用ssh密钥

Git支持https和ssh两种传输协议，https每次传输都要输密码，ssh不用。

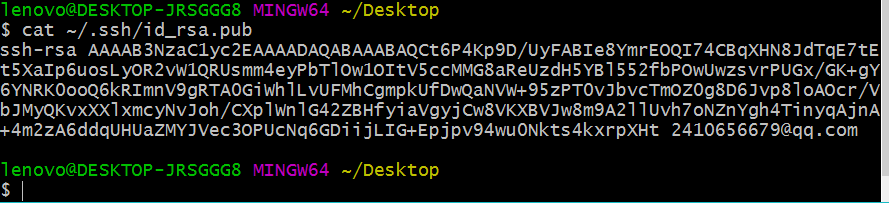
使用ssh协议：

生成密钥对

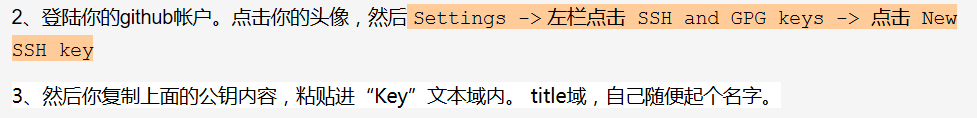


添加公钥到远程仓库

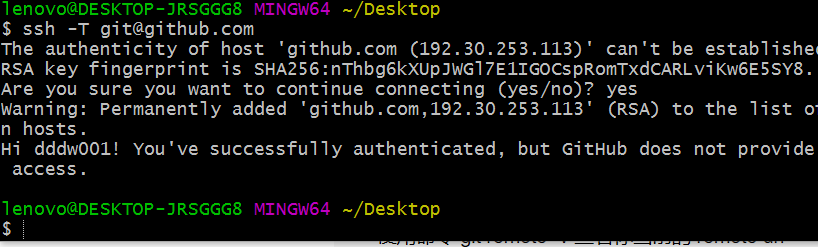
Ⅰ查看生成的公钥



复制粘贴

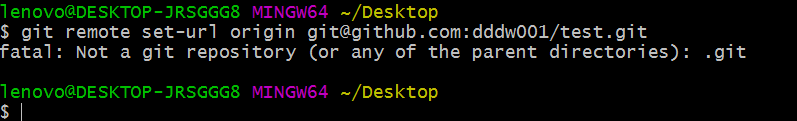


验证



修改gitd remote url

使用$git remote –v可以查看当前的remote url (协议)



$git remote set-url origin + github官网某仓库的clone ssh key 可以把协议变成ssh

图中报错是因为没有本地仓库

* 1. 从本地仓库向远程上传

Ⅰ新建文件夹

Ⅱ初始化为仓库

$git init

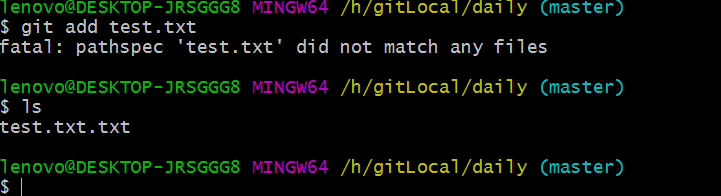
Ⅲ上传

$git add filename/\*

$git commit –m ‘…’

…

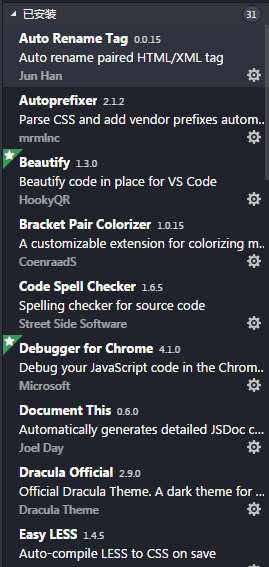
注: .txt文件会自动添加后缀，命名时不需要添加.txt



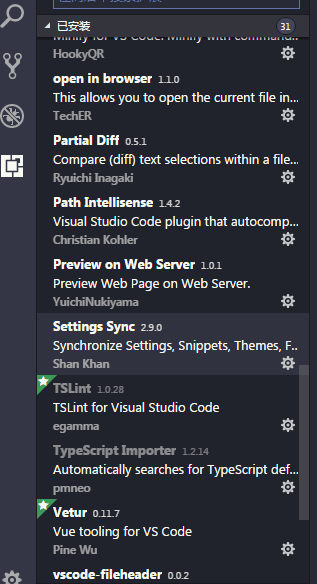
1. 蓝牙键盘按键错乱

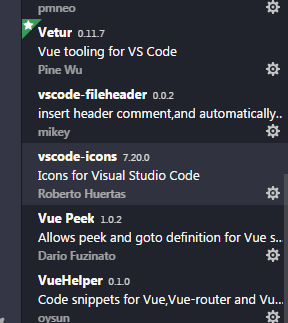
点击 NumLock

1. 安装vscode扩展插件









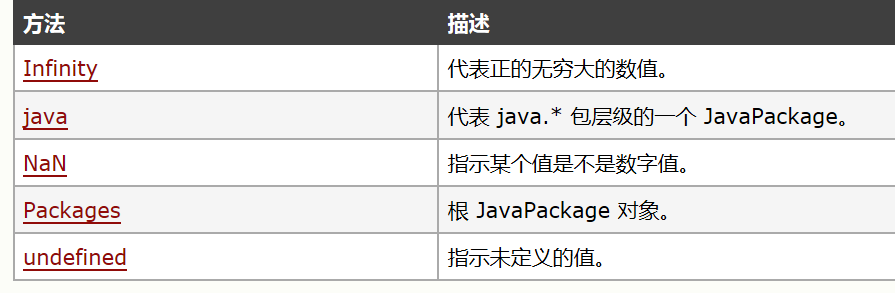
2018.3.6

1. vscode一个窗口只能开一个文件夹下的项目，如果想开另一个工作区，方法是ctrl+shift+n打开一个新窗口。
2. 全局对象

ECMAScript的全局对象是window，nodeJS的全局对象是global。

window对象可以访问全局属性和全局方法

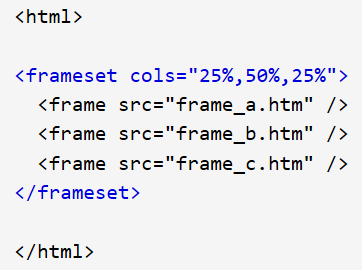




1. frameset

框架集

不能与<body>一起使用



h5不支持<frameset>

属性：cols rows class id style title

事件：onload onunload

1. 修改vscode的默认浏览器

文件—首选项—设置—搜索‘browser’

1. 数组去重

indexOf

对象的键

数组下标

new Set

1. var的重复声明

var a=100;

function fn(){

alert(a);//undefined

var a=200;

alert(a);//200

}

fn();

alert(a);//100

var a;

alert(a);//100

var a=300;

alert(a);//300

1. var重复声明

var a=2;

var a;

var a=3;

引擎负责整个代码的编译以及运行，编译器则负责词法分析、语法分析、代码生成等工作而作用域则如我们熟知的一样，负责维护所有的标识符（变量）。

新变量分配一块儿内存，命名为a，并赋值为2，但在运行的时候编译器与引擎还会进行两项额外的操作：判断变量是否已经声明

1.首先编译器对代码进行分析拆解，从左至右遇见var a，则编译器会询问作用域是否已经存在叫a的变量了，如果不存在，则招呼作用域声明一个新的变量a，若已经存在，则忽略var 继续向下编译，这时a = 2被编译成可执行的代码供引擎使用。

        2.引擎遇见a=2时同样会询问在当前的作用域下是否有变量a，若存在，则将a赋值为2（由于第一步编译器忽略了重复声明的var，且作用域中已经有a，所以重复声明会发生值得覆盖而并不会报错）。若不存在，则顺着作用域链向上查找，若最终找到了变量a则将其赋值2，若没有找到，则招呼作用域声明一个变量a并赋值为2

1. 判断2个对象是否相等
   1. JSON.stringify(obj1)===JSON.stringify(obj2)

如果属性的顺序不同就会不相等

* 1. 遍历属性

function isObjectValueEqual(a, b) {

     var aProps = Object.getOwnPropertyNames(a);

     var bProps = Object.getOwnPropertyNames(b);

     // If number of properties is different,

// objects are not equivalent

if (aProps.length != bProps.length) {

return false;

}

for (var i = 0; i < aProps.length; i++) {

 var propName = aProps[i];

// If values of same property are not equal,

// objects are not equivalent

if (a[propName] !== b[propName]) {

return false;

}

}

     // If we made it this far, objects

     // are considered equivalent

     return true;

}

var obj1 = {

     name: "Benjamin",

     sex : "male"

};

var obj2 = {

     name: "Benjamin",

     sex : "male"

};

//Outputs: true

console.log(isObjectValueEqual(obj1, obj2));

3种情况不能处理：

属性值之一是对象

属性值是NaN

属性值是undefined

* 1. \_.isEqual()

Underscore和Lo-Dash有一个名为\_.isEqual()方法

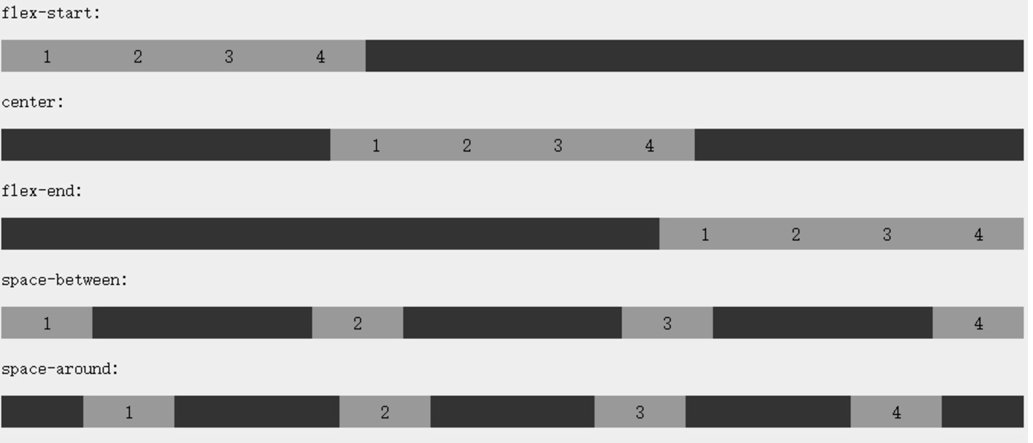
1. 去掉滚动条

<https://segmentfault.com/q/1010000004637082>

1. flex-box

给盒子设置display:flex;display:-webkit-flex;使其成为弹性盒子

justify-content把弹性项目沿着主轴线对齐



flex-wrap:nowrap|wrap|wrap-reverse|initial|inherit;子元素换行方式

order整数 定义排列顺序 可为负 小的在前面

2018.3.7

1. 红宝书的DOM1
2. 避免js中的变量冲突

避免全局变量的泛滥

合理使用命名空间

为代码添加必要的注释

使用命名空间法

var GLOBAL={};

GLOBAL.namespace=function(str){

var arr=str.split('.');

var o=GLOBAL;

var start=0;

if(arr[0]=='GLOBAL'){

start=1;

}

else{

start=0;

}

for(var i=start;i<arr.length;i++){

o[arr[i]]=o[arr[i]]||{};

o=o[arr[i]];

}

}

/\*

\*@method 功能A：实现加法运算

\*@author 程序员甲

\*/

void (function(){

var a=1;

var b=2;

GLOBAL.namespace('A.CAT');

GLOBAL.namespace('A.DOG');

GLOBAL.A.CAT.name='mimi';

GLOBAL.A.DOG.name='xiaobai';

GLOBAL.A.CAT.move=function(){};

GLOBAL.A.str1=a;

GLOBAL.A.str2=b;

console.log(a+b);//3

})();

/\*

\*@method 功能C：实现乘法运算

\*@author 程序员丙

\*/

void (function(){

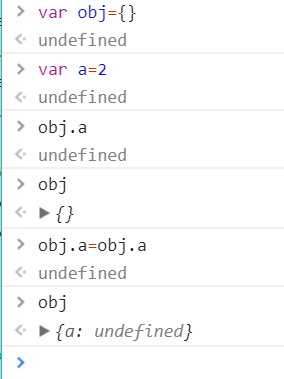
var a=GLOBAL.A.str1;

var b=GLOBAL.A.str2;

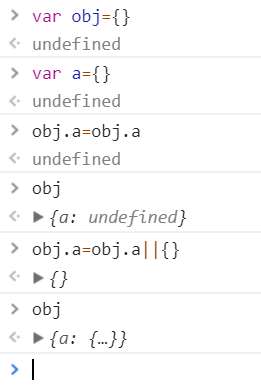
console.log(a\*b);//2

})();

1. obj.a=obj.a||{}即使obj.a不存在在这个等式后obj.a存在了

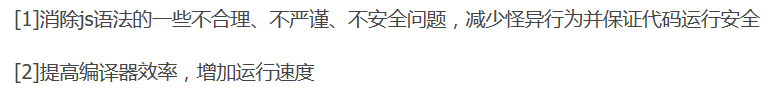


||{} 去空 把undefioned变成{}

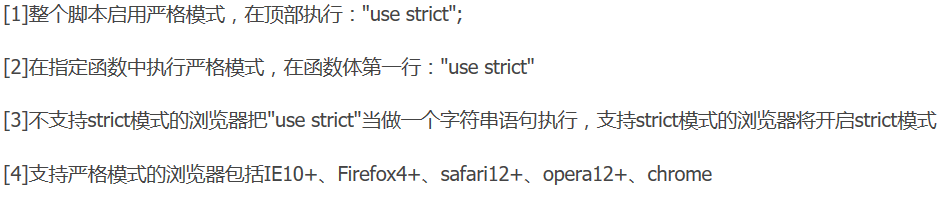


1. js严格模式下的8点规则

严格模式的作用



如何使用严格模式



1. 变量

不允许意外创建全局变量

不能对变量调用delete操作符

"use strict"

str="hello";//Uncaught ReferenceError: str is not defined

var color="red";

delete color;//Uncaught SyntaxError: Delete of an unqualified identifier in strict mode.

1. 对象

不能为只读属性赋值

不能为不可配置的属性使用delete操作符

"use strict"

var person={name:'cook'};

Object.defineProperty(person,'name',{writable:false});

person.name='a';//Uncaught TypeError: Cannot assign to read only property 'name' of object '#<Object>'

Object.defineProperty(person,'name',{configurable:false});

delete person.name;//obj.html:16 Uncaught TypeError: Cannot delete property 'name' of #<Object>

1. 函数

参数必须唯一

修改形参不会影响argument

不允许使用arguments.callee和arguments.caller

"use strict"

function fn(n1,n2){

console.log(n1,n2);//1 1

n1=3;

console.log(arguments[0]);//1

}

fn(1,1);

注：

1. 不允许eval()在包含上下文中创建变量或函数

"use strict"

function fn(){

eval("var x=1");

alert(x);

}

fn();//Uncaught ReferenceError: x is not defined

1. 不运行使用eval和arguments作标识符，也不允许读写他们的值
2. 不允许this的值为null或undefined
3. 不允许使用with语句
4. 不允许使用八进制字面量
5. let

js没有块级作用域，但let声明的变量只在块级内有效

1. for…of

var arr=['a','b','c'];

//for...in循环读取键名

for(let x in arr){

console.log(typeof x);//string

console.log(x);//0 1 2

}

//for...of循环读取键值

for(let y of arr){

console.log(y);//a b c

}

//es6提供了3个新方法用来遍历数组

//keys()-键名 values()-键值

for(let index of arr.keys()){

console.log(typeof index);//number

console.log(index);//0 1 2

}

// for(let elem of arr.values()){

// console.log(elem);//error

// }

for(let elem of Object.values(arr)){

console.log(typeof elem);//string

console.log(elem);//a b c

}

for(let [index,elem] of arr.entries()){

console.log(typeof elem);//string

console.log(index,elem);//0 "a" 1 "b" 2 "c"

}

Array.from可以将类似数组的对象转换为数组

对于普通的对象，for…of不能直接使用，必须部署Iterator接口，但for…in可以遍历键名

let obj={

name:'x',

sex:'woman'

}

for(var key of Object.keys(obj)){

console.log(key,obj[key]);//name x sex woman

}

除了上述方法，还可以使用Generator函数将对象重新包装一下

1. 几种循环的比较

for()麻烦

forEach()不能中途跳出循环

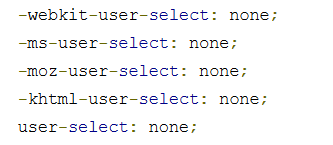
for…in遍历键名 数组键名是数字，甚至遍历手动添加的其他键甚至原型链上的键，以任意顺序遍历键名 主要为遍历对象设计

for…of 没有for…in的缺点，可以跳出循环，提供了遍历所有数据结构的统一接口

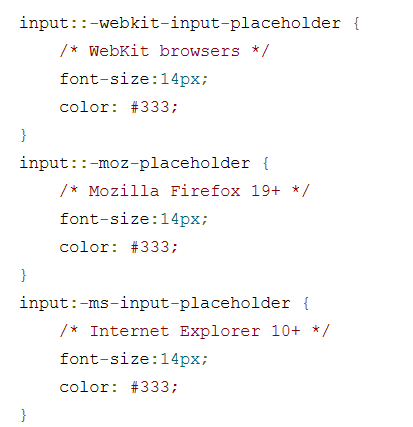
1. 消除transition闪屏



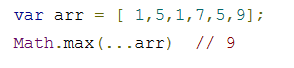
1. 让图文不可复制



1. 改变placeholder的字体颜色大小 只在pc端可以



1. 求数组最大值 最简单



1. 数组去重 最简单



2018.3.8

1. 红宝书 dom扩展
2. Promise
   1. Promise对象代表一个异步操作，有3种状态：Pending进行中、Fulfilled已成功、Rejected已失败。

Promise的状态改变只有2种可能：从进行中到成功、从进行中到失败。一旦状态改变就不会再变。

无法取消Promise，一旦新建就会立即执行。

* 1. 例子1

/\*

Promise对象是一个构造函数，用来生成Promise实例

Promise实例生成后，可以用then 方法指定Resolved 状态和Rejected状态的回调函数

\*/

function timeout(ms){

return new Promise((resolve,reject)=>{

setTimeout(resolve,ms,'done');

})

}

timeout(100).then(value=>{

console.log(value);//done

})

* 1. 例2

/\*

Promise新建后就会立即执行，

then方法指定的回调函数将在当前脚本中的所有同步任务执行完成后才会执行

\*/

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

console.log('Promise');

resolve();

})

promise.then(function(){

console.log('Resolved');

})

console.log('hi');

//结果：Promise hi Resolved

* 1. 例3 异步加载图片

function loadImageAsync(url){

return new Promise(function(resolve,reject){

var image=new Image();

image.onload=function(){

resolve(image);

}

image.onerror=function(){

reject(new Error('Could not load image at '+url));

}

image.src=url;

})

}

* 1. Promise.prototype.then()

Promise实例有then方法，即then方法是定义在原型对象Promise.prototype上的。

作用是为Promise实例添加状态改变时的回调函数。

then返回的是一个新对象，可以用链式写法

* 1. Promise.prototype.catch()

Promise.prototype.catch()是.then(null,rejection)的别名，用于指定发生错误时的回调函数。

最好用catch，不要用then中的reject的回调函数

1. 让input不能获取焦点

onfouce=”this.blur()”

1. readOnly和disabled

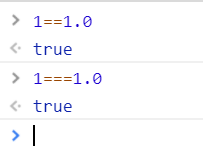
readOnly只对input和textarea有效

表单元素使用disabled时，表单提交时不能被传递出去

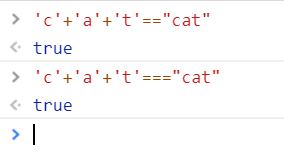
disabled时，不能获取焦点，用户的所有操作都无效

2018.3.9

1. 红宝书第12章DOM1和DOM2
2. 数字类型在内部被表示为64位浮点数



1. js中所有字符都是16位的



1. 文字在div底部

方法一：line-height设置为div的接近2倍

如果是p元素要将p元素的margin-top margin-bottom设置为0px

方法二：position:absolute

1. hr边缘不光滑

.hr2{width: 100%;height: 1px;border: none;border-top: 1px solid red;}



用after代替

p::after{content: "";width: 100%;height: 1px;

border-top: 1px solid red;display: block;}

1. 下划线且首字母大一点

p::first-letter{

font-size: 2em;

}

p::after{content: "";width: 100px;height: 1px;

border-top: 1px solid black;display: block;}

1. 首字母变大

单行变大

p::first-letter{

font-size: 2em;

}

多行变大：

p::first-letter{

font-size: 2.5em;

float: left;

margin-top: -10px;

}

p{word-break: break-all;}

1. 私有变量和特权方法
2. 定义

私有变量：在函数内定义的变量，不能在函数外访问到。包括参数，局部变量和函数内部定义的其他函数

特权方法指能访问私有变量的公有方法，比如闭包

1. 创建方式

构造函数： 每个实例都会创建同一组方法

var privateVar=100;

function myObject(){

//私有变量和私有函数

var privateVar=10;

function privateFun(){

alert(this.privateVar);//100

alert(privateVar);//11

return true;

}

//特权方法

this.publicMethod=function(){

privateVar++;

return privateFun();

}

}

new myObject().publicMethod();

静态私有变量

var privateVar=100;

(function(){

//私有变量和私有函数

var privateVar=10;

function privateFun(){

alert(privateVar);//11

alert(this.privateVar);//100

return false;

}

myObject=function(){};

myObject.prototype.publicMethod=function(){

privateVar++;

return privateFun();

}

})();

new myObject().publicMethod();



1. 作用

利用私有和特权成员，隐藏不应该被直接修改的数据

function Person(name){

this.getName=function(){

return name;

}

this.setName=function(value){

name=value;

}

}

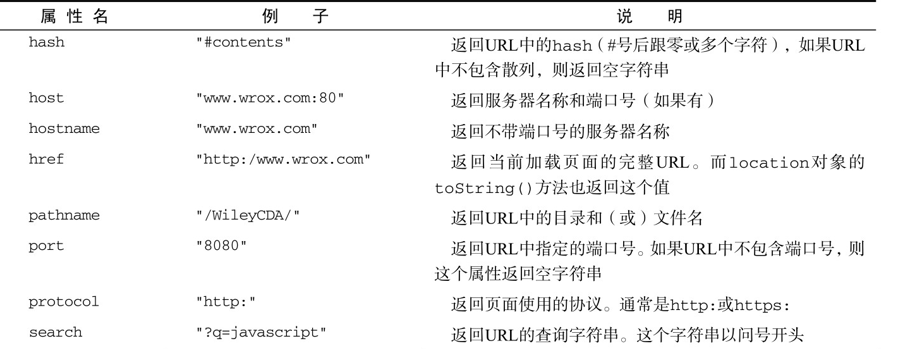
var person=new Person("name1");

console.log(person.getName());//name1

person.setName("name2");

console.log(person.getName());//name2

1. BOM的location对象
   1. location既是window对象的属性，也是document对象的属性，还可以单独使用
   2. 属性



var URL = 'http://localhost:8080/index.html#text?q=test';

console.log(location.hash);//#text?q=test

console.log(location.host);//localhost:8080

console.log(location.hostname);//localhost

console.log(location.href);//http://localhost:8080/index.html#text?q=test

console.log(location.pathname);///index.html

console.log(location.port);//8080

console.log(location.protocol);//http:

console.log(location.search);''

location.search只能取到?后面和#前面的内容

* 1. 解析查询字符串

function getQueryStringArgs(){

//取得查询字符串并去掉开头的问号

var qs = location.search.length > 0 ? location.search.substring(1) : "";

//保存数据的对象

var args = {};

//取得每一项

var items = qs.length ? qs.split("&") : [];

var item,name,value;

var len = items.length;

//逐个将每一项添加到args对象中

for(var i = 0; i < len; i++){

item = items[i].split("=");

name = decodeURIComponent(item[0]);

value = decodeURIComponent(item[1]);

if(name.length){

args[name] = value;

}

}

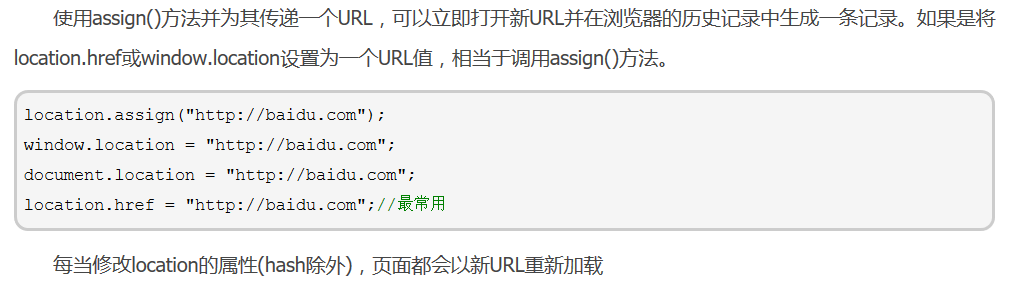
return args;

}

location.search = "?name=abc&password=123&callback=fn";

console.log(getQueryStringArgs());//[name: "abc", password: "123", callback: "fn"]

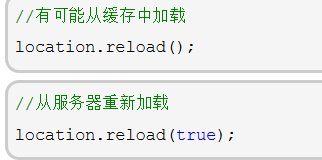
* 1. assign



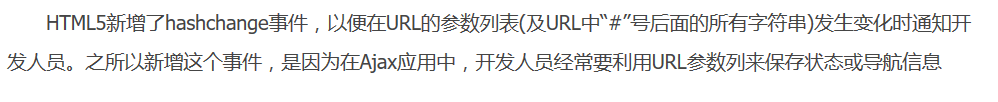
* 1. replace 位置改变但不生成记录



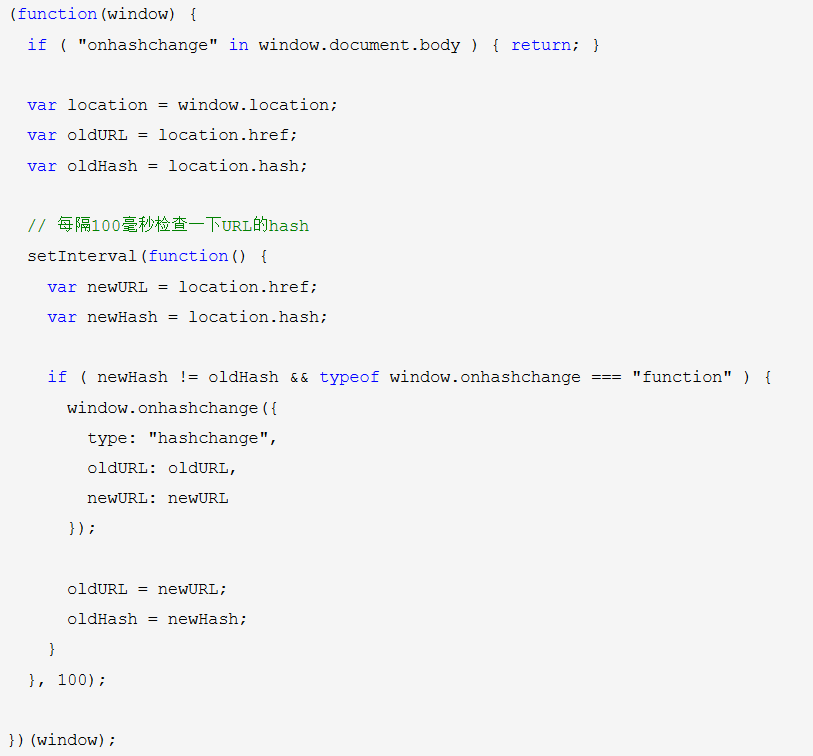
* 1. reload重新加载页面 reload()之后的代码可能不会执行



* 1. hasChange



兼容性写法



1. calc()

.col-md-1{width: calc(8.333% - 20px);}

符号左右必须有空格

2018.3.10

1. table

属性和方法：

<table>:tBodies

<tbody>:rows deleteRow(pos) insertRow(pos)

<tr>:cells deleteCell(pos) insertCell(pos)

创建一个2行2列的表格

var table=document.createElement("table");

table.border=1;

table.width="100%";

var tBody=document.createElement("tBody");

table.appendChild(tBody);

tBody.insertRow(0);

tBody.rows[0].insertCell(0);

tBody.rows[0].cells[0].appendChild(document.createTextNode("cell 1-1"));

tBody.rows[0].insertCell(1);

tBody.rows[0].cells[1].appendChild(document.createTextNode("cell 1-2"));

tBody.insertRow(1);

tBody.rows[1].insertCell(0);

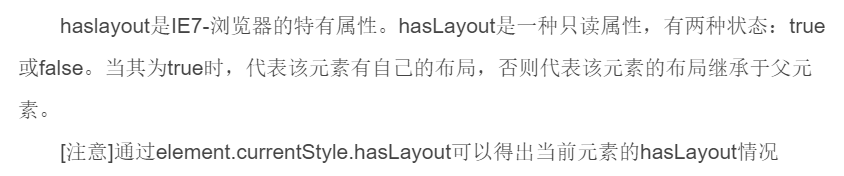
tBody.rows[1].cells[0].appendChild(document.createTextNode("cell -1"));

tBody.rows[1].insertCell(1);

tBody.rows[1].cells[1].appendChild(document.createTextNode("cell 2-2"));

document.body.appendChild(table);

1. hasLayout



1. margin
   1. 没有继承性
   2. ie6下双边距问题

margin双边距布局产生原因：block元素+浮动+margin



* 1. margin重叠问题

水平边距不会重合

边距宽度是相邻边距中较大的值

如果出现负边距，则是最大正边距减去绝对值最大的负边距

如果没有正边距，则从零中减去绝对值最大的负边距

* 1. margin不起作用的情况

多出现在firefox

浮动影响/margin重叠/padding与margin重叠

* 1. margin负值的应用
     1. 页面上实现css sprite背景定位效果

css sprite css精灵图/雪碧图

把页面上所有的图片集中到一张大图上

* + 1. 流动性布局
    2. 选项卡边框线

.title{border-right: 1px solid red;padding: 5px 15px;}

.menu-content li{display: none;margin-top: -1px;border-top: 1px solid black;}

* + 1. 左右边框

.box div{font-size: 14px;display: inline-block;border-left: 1px solid red;border-right: 1px solid red;margin-left: -1px;}

<div style="font-size:0px;" class="box">

<div>div1</div>

<div>div2</div>

<div>div3</div>

<div>div4</div>

</div>



* + 1. 图片与文字对齐问题

图片与文字默认是居底对齐

使用vertical-align:top对ie的兼容性不够好

可以使用margin

img{width: 50px;margin: 0 0 -40px 0;}

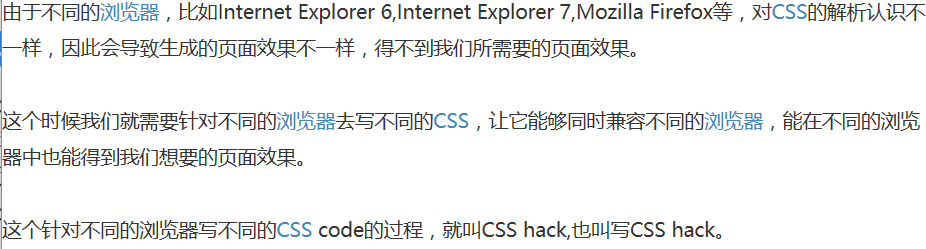
<div class="box">

aaaa

<img src="./img.PNG" alt="">

</div>

1. hack



1. 去除display:inline-block产生间隙的办法
   1. 在编辑器中去掉空格
   2. 不闭合标签
   3. html注释掉空格
   4. 给父元素设置font-size:0;

html{-webkit-text-size-adjust:none;/\* 使用webkit的私有属性，让字体大小不受设备终端的调整，可定义字体大小小于12px \*/}

div{font-size: 0px;\*word-spacing:-1px;/\* 使用word-spacing 修复 IE6、7 中始终存在的 1px 空隙，减少单词间的空白（即字间隔） \*/}

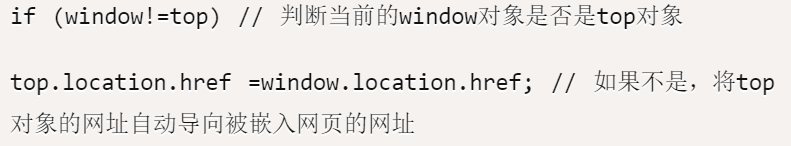
div span{font-size: 14px;letter-spacing: normal;/\* 设置字母、字间距为0 \*/ word-spacing: normal; /\* 设置单词、字段间距为0 \*/}

1. a标签

<a href="#">链接1</a><!--点击时url后面会加个#-->

<a href="javascript:void(0)">链接2</a><!--推荐这种-->

1. 防止网页被嵌入框架的代码



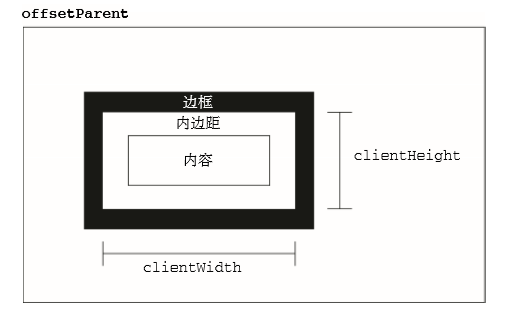
1. 闭包
2. 用js获取页面元素的位置
   1. 网页大小和浏览器窗口大小

一张网页的全面积就是它的大小

浏览器窗口的大小，指的是浏览器窗口中看到的那部分的面积，又叫viewport(视口)

* 1. 获取网页大小-clientWidth clientHeight

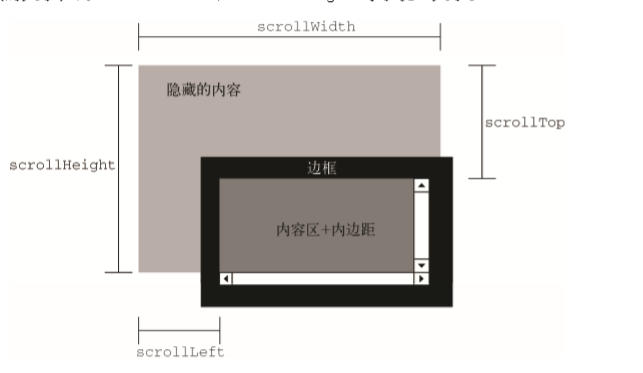
clientWidth和clientHeight不包括border和滚动条



var width=document.body.clientWidth||document.documentElement.clientWidth;

var height=document.body.clientHeight||document.documentElement.clientHeight;

* 1. 获取网页大小-scrollHeight scrollWidth



function getPageArea(){

　　　　if (document.compatMode == "BackCompat"){

　　　　　　return {

　　　　　　　　width: Math.max(document.body.scrollWidth,

　　　　　　　　　　　　　　　　document.body.clientWidth),

　　　　　　　　height: Math.max(document.body.scrollHeight,

　　　　　　　　　　　　　　　　document.body.clientHeight)

　　　　　　}

　　　　} else {

　　　　　　return {

　　　　　　　　width: Math.max(document.documentElement.scrollWidth,

　　　　　　　　　　　　　　　　document.documentElement.clientWidth),

　　　　　　　　height: Math.max(document.documentElement.scrollHeight,

　　　　　　　　　　　　　　　　document.documentElement.clientHeight)

　　　　　　}

　　　　}

　　}

console.log(getPageArea());//Object{width:100px,height:100px}

* 1. 获取网页元素的绝对位置

该元素的左上角相对于整个网页左上角的坐标

function getElementLeft(element){

　　　　var actualLeft = element.offsetLeft;

　　　　var current = element.offsetParent;

　　　　while (current !== null){

　　　　　　actualLeft += current.offsetLeft;

　　　　　　current = current.offsetParent;

　　　　}

　　　　return actualLeft;

　　}

　　function getElementTop(element){

　　　　var actualTop = element.offsetTop;

　　　　var current = element.offsetParent;

　　　　while (current !== null){

　　　　　　actualTop += current.offsetTop;

　　　　　　current = current.offsetParent;

　　　　}

　　　　return actualTop;

　　}

* 1. 获取网页元素的相对位置

该元素相对于浏览器窗口左上角的坐标

function getElementViewLeft(element){

　　　　var actualLeft = element.offsetLeft;

　　　　var current = element.offsetParent;

　　　　while (current !== null){

　　　　　　actualLeft += current.offsetLeft;

　　　　　　current = current.offsetParent;

　　　　}

　　　　if (document.compatMode == "BackCompat"){

　　　　　　var elementScrollLeft=document.body.scrollLeft;

　　　　} else {

　　　　　　var elementScrollLeft=document.documentElement.scrollLeft;

　　　　}

　　　　return actualLeft-elementScrollLeft;

　　}

　　function getElementViewTop(element){

　　　　var actualTop = element.offsetTop;

　　　　var current = element.offsetParent;

　　　　while (current !== null){

　　　　　　actualTop += current. offsetTop;

　　　　　　current = current.offsetParent;

　　　　}

　　　　 if (document.compatMode == "BackCompat"){

　　　　　　var elementScrollTop=document.body.scrollTop;

　　　　} else {

　　　　　　var elementScrollTop=document.documentElement.scrollTop;

　　　　}

　　　　return actualTop-elementScrollTop;

　　}

2018.3.11

1. 元素获取焦点出现蓝色框

<input type="text" />

<input type="text" onfocus="this.blur()"/>

这种办法会使元素无法被选中

1. link和@important

最好不要使用@important

<https://www.cnblogs.com/my--sunshine/p/6872224.html>

1. 复选框单选框与文字对齐问题

当文字为12px时，文字在垂直方向上与单/复选框不对齐

默认对齐方式是vertical-align:baseline

body{font-size: 12px;}

/\*推荐3/5\*/

.div1 input{vertical-align: bottom;margin-bottom: 2px;margin-bottom: -2px\9;}

.div2 input{height: 13px;vertical-align: text-top;margin-top: 0;}

.div3 input{height: 15px;vertical-align: bottom;margin-bottom: 3px;margin-top: -1px;}

.div4 input{height: 14px;vertical-align: top;}

.div5 input{vertical-align: middle;margin-top: -2px;margin-bottom: 1px;}

1. 多行文本垂直居中

div{display: table-cell;width: 500px;height: 1.14em;padding: 0 0.1em;border: 4px solid black;font-size: 10em;vertical-align: middle;}

div span{display: inline-block;font-size: 0.1em;vertical-align: middle;}

1. 大小不固定的图片水平垂直居中
   1. 透明gif图片+背景定位

使用一个透明的gif图片做覆盖层，高宽拉伸至所需要的大小，然后给这个gif图片一个background-position:center声明



* 1. table-cell

.box{display: table-cell;width: 200px;height: 200px;border: 1px solid red;text-align: center;vertical-align: middle;box-sizing: border-box;/\*图片宽高上限128\*/}

.box img{vertical-align: middle;border: 1px solid red;}

* 1. a

a{display: inline-block;width: 100px;height: 200px;text-align: center;vertical-align: middle;border: 1px solid red;line-height: 200px;}

img{vertical-align: middle;border: 1px solid red;}

* 1. 透明图片拉伸对齐



2018.3.12

1. 模板字符串替换

String.prototype.render=function(context){

return this.replace(/{{(.\*?)}}/g,(match,key)=>context[key.trim()]);

}

var str="{{name}}很厉害";

str=str.render({name:'js'});

console.log(str);//js很厉害

1. 静态作用域和动态作用域

var x=10;

function foo() {

var y=x+5;

console.log(y);

}

function bar(){

var x=2;

return foo();

}

function main(){

foo();//static scope:15 dynamic scope:15

bar();//static scope:15 dynamic:7

return 0;

}

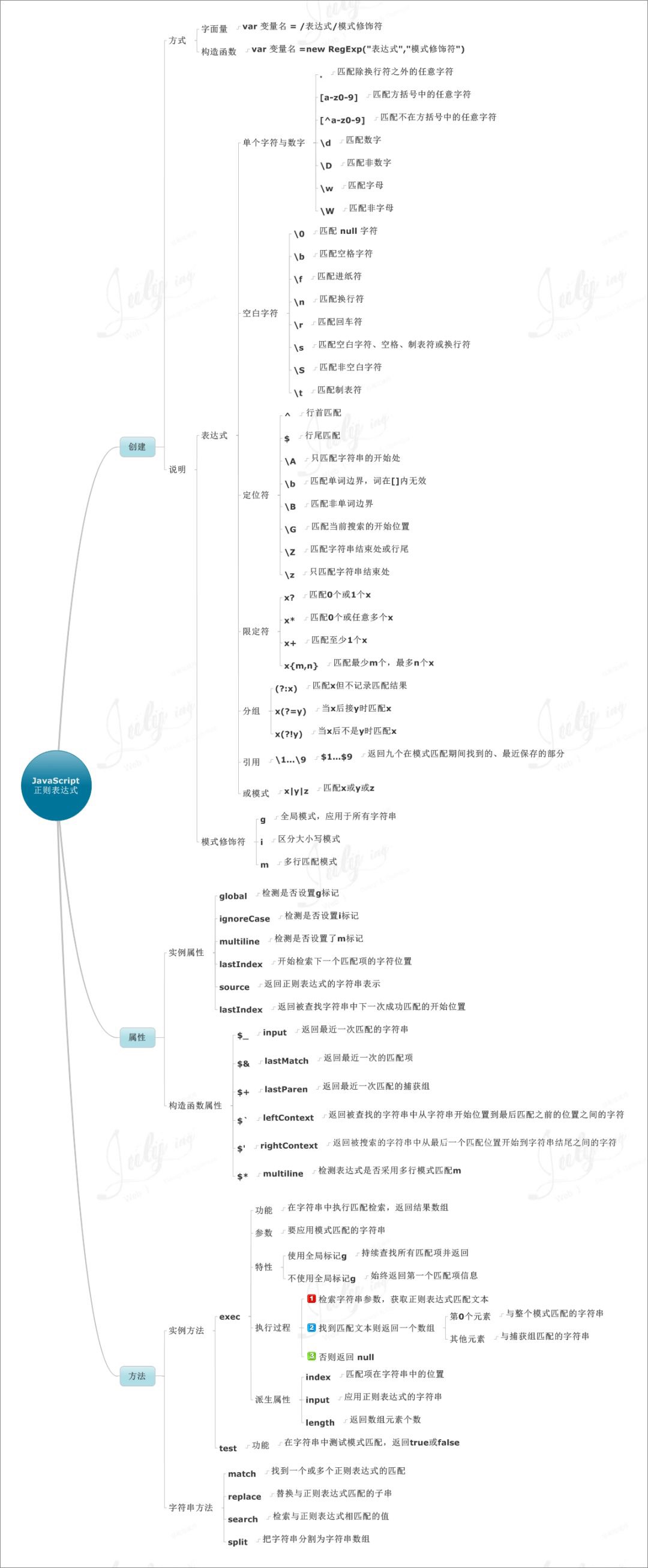
main();

//在静态作用域中，bar 的返回值是基于函数foo 创建时x的值

//在动态作用域中，x的值取决于在当前作用域中的实际定义

js采用词法作用域，即静态作用域

1. 闭包
   1. 在任意时间点，只能有一个执行上下文
   2. 闭包的一个重要用途是保留外部作用域对一个变量的私有引用
2. 正则



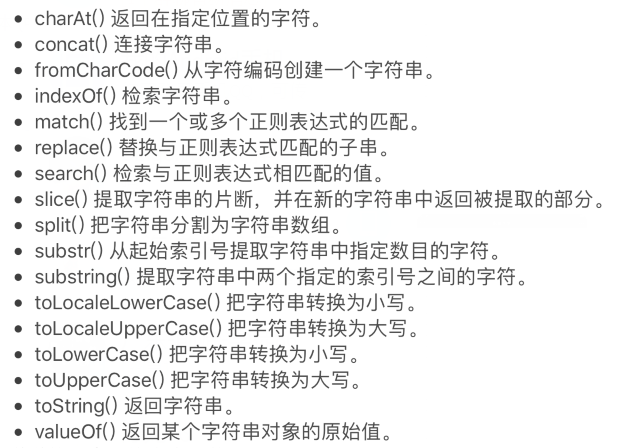
1. a标签的点击事件里，this指向window

this指调用函数的那个对象

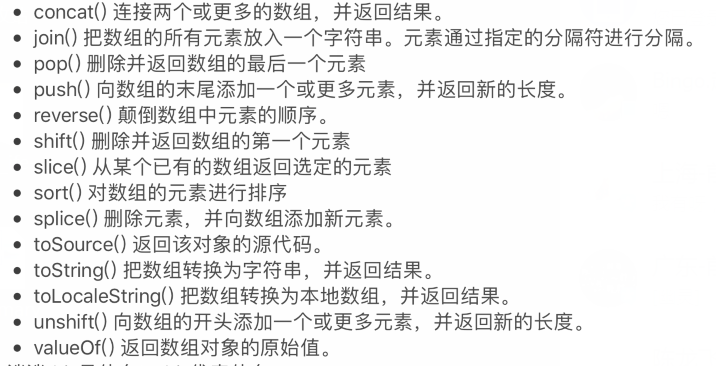
1. 改变this指向的方法

call apply bind

1. 绑定事件方法
   1. 嵌入dom
   2. 直接绑定 onclick
   3. 事件监听 addEventListener/attachEvent
2. 字符串常用函数



1. 数组常用函数



1. 对象-封装

构造函数模式

function Cat(name,color){

this.name=name;

this.color=color;

}

var cat1=new Cat("大毛","黄色");

var cat2=new Cat("二毛","黑色");

alert(cat1.name);//大毛

console.log(cat1.constructor==Cat);//true

console.log(cat2.constructor==Cat);//true

console.log(cat1 instanceof Cat);//true

缺点是浪费内存，如果有一样的内容，每生成一个实例都必须重复

Prototype模式

每个构造函数都有一个prototype属性，指向另一个对象，这个对象的所有属性和方法，都会被构造函数的实例继承。

function Cat(name,color){

this.name=name;

this.color=color;

}

Cat.prototype.type="猫科动物";

var cat1=new Cat("大毛","黄色");

var cat2=new Cat("二毛","黑色");

console.log(cat1.type);//猫科动物

console.log(cat1.type==cat2.type);//true

console.log(Cat.prototype.isPrototypeOf(cat1));//true

console.log(cat1.hasOwnPrototype("name"));//true

console.log(cat1.hasOwnPrototype("type"));//false

console.log("name" in cat1);//true

console.log("type" in cat1);//true

1. 对象-继承

构造函数模式

function Animal(){

this.species="动物";

}

function Cat(name,color){

Animal.apply(this,arguments);

this.name=name;

this.color=color;

}

var cat1=new Cat("大毛","黄色");

console.log(cat1.species);//动物

prototype模式 最常用

function Animal(){

this.species="动物";

}

function Cat(name,color){

this.name=name;

this.color=color;

}

Cat.prototype=new Animal();

Cat.prototype.constructor=Cat;

var cat1=new Cat("大毛","黄色");

console.log(cat1.species);//动物

Cat.prototype=new Animal();

将Cat的prototype对象指向一个Animal的实例。

任何一个prototype对象都有一个constructor属性，执行它的构造函数，如果没有这一行，Cat.prototype.constructor是指向Cat的，加了之后指向Animal，所以要手动纠正

Cat.prototype.constructor=Cat;

直接继承prototype

function Animal(){}

Animal.prototype.species="动物";

function Cat(name,color){

this.name=name;

this.color=color;

}

Cat.prototype=Animal.prototype;

Cat.prototype.constructor=Cat;

var cat1=new Cat("大毛","黄色");

console.log(cat1.species);//动物

缺点：Cat.prototype的修改会影响Animal.prototype

利用空对象作中介

function extend(Child,Parent){

var F=function(){};

F.prototype=Parent.prototype;

Child.prototype=new F();

Child.prototype.constructor=Child;

Child.uber=Parent.prototype;

}

function Animal(){}

Animal.prototype.species="动物";

function Cat(name,color){

this.name=name;

this.color=color;

}

extend(Cat,Animal);

var cat1=new Cat("大毛","黄色");

console.log(cat1.species);//动物

拷贝继承

function Animal(){}

Animal.prototype.species="动物";

function Cat(name,color){

this.name=name;

this.color=color;

}

function extend2(Child,Parent){

var p=Parent.prototype;

var c=Child.prototype;

for(var i in p){

c[i]=p[i];

}

c.uber=p;

}

extend2(Cat,Animal);

var cat1=new Cat("大毛","黄色");

console.log(cat1.species);//动物

1. value

<input type="button" value="123" onclick="alert(value)"/>

input的value属性，默认是””，一直存在，不会沿作用域链向上找

<form action="#">

<input type="text" name="username" value="123" />

<input type="button" value="btn" onclick="alert(username.value)" />

</form>

表单元素的作用域链：this->this.form->document

2018.3.13

1. DOM一致性检测

DOM分3个级别：DOM0、DOM1、DOM2、 DOM3，也分多个部分，因此需要检测浏览器实现了DOM的哪些部分。

document.implementation属性就是这些提供相应信息和功能的对象。

DOM1级只为document.implementation规定了一个方法，即hasFeature()。这个方法接受两个参数:要检测的DOM功能的名称及版本号。如果浏览器支持给定名称和版本的功能，则该方法返回true。

var hasCoreDom1 = document.implementation.hasFeature('Core','1.0');

1. 图片预加载

<http://www.cnblogs.com/xiaohuochai/p/4859485.html>

使用页面无用元素的背景图片预加载

通过new Image()或document.createElement('img')创建<img>标签，然后通过<img>src赋值语句来加载图片

利用图像的onload事件可以确切地知道图片是否被真正加载，并可能在后续执行一系列对图片的操作功能，如获取当前图片的实际宽高及索引等

图片的src赋值语句必须放在图片的onload事件后面。

1. 表单禁用

readOnly只对input(text/password)和textarea有效，disabled对所有表单元素有效。

2018.3.14

1. 红宝书中的第14章表单脚本
2. 获取表单

var form=document.getElementById("form1");

var firstForm=document.forms[0];

var myForm=document.forms["myForm"];

1. 提交表单

<!--通用提交按钮-->

<input type="submit" value="Submit Form"/>

<!--自定义提交按钮-->

<button type="submit">Submit Form</button>

<!--图像按钮-->

<input type="image" src="pic.png"/>

阻止submit事件的默认行为就会取消表单提交

EventUtil.addHandler(oForm,"submit",function(event){

event=EventUtil.getEvent(event);

EventUtil.preventDefault(event);

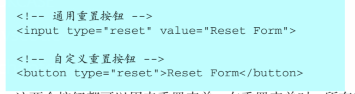
})

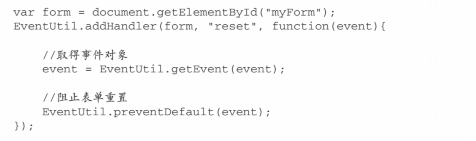
oForm.submit();

在以调用submit()的形式提交表单时，不会触发submit事件

避免重复提交表单方法：在第一次提交表单后禁用按钮，利用onSubmit事件处理程序取消后续的表单操作

1. 重置表单







1. 表单元素

//取得表单中的第一个字段

var field1=oForm.elements[0];

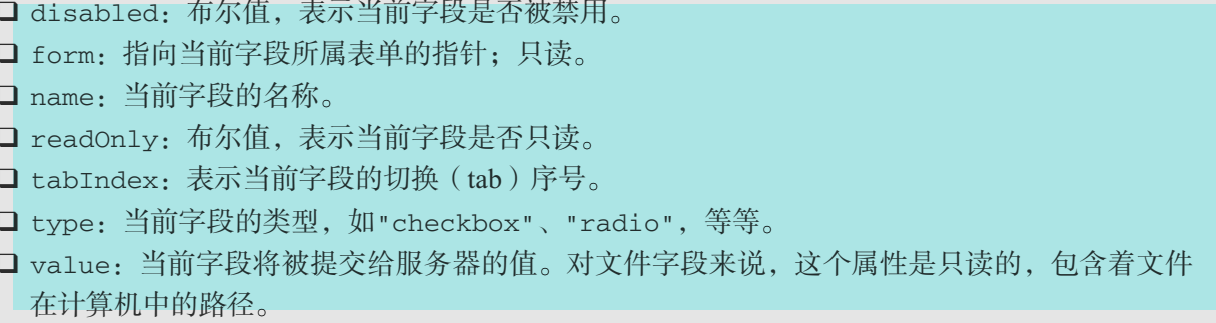
//取得名为color的字段

var aColor=oForm.elements["color"];

//取得表单中包含字段的数量

var iFieldCount=oForm.elements.length;

1. 表单字段属性



var oForm=document.getElementById("myForm");

var field=form.elements[0];

field.value="another value";

console.log(field.form===oForm);

field.focus();

field.disabled=true;

field.type="checkbox";//不推荐

避免多次提交-submit事件中禁用提交按钮

EventUtil.addHandler(oForm,"submit",function(event){

event=EventUtil.getEvent(event);

var target=EventUtil.getTarget(event);

var btn=target.elements["submitBtn"];

btn.disabled=true;

})

focus()获取焦点 不能为不可见的元素使用这个方法

blur()移除焦点

<input type="text" autofocus />

自动获取焦点

1. change事件

对于input和textarea元素，在它们失去焦点且value值改变时触发，对于select元素，在其选项改变时触发

1. 文本框脚本

单行文本：

<!--显示25个字符 输入不能超过50个-->

<input type="text" size="25" maxlength="50" value="initial value">

多行：

<textarea id="" cols="30" rows="10">initial value</textarea>

1. 选择文本

select() 选中所有文本

select事件-选择文本框的文本时就会触发

EventUtil.addHandler(oText,"select",function(event){

console.log(oText.value);

})

在调用select()方法时也会触发select事件

获得选取的文本

function getSelectedText(textBox){

if(typeof textBox.selectionStart=="number"){

return textBox.value.substring(textBox.selectionStart,textBox.selectionEnd);

}

else if(document.selection){

return document.selection.createRange().text;

}

}

选取文本

function selectText(textBox,startIndex,endIndex){

if(textBox.setSelectionRange){

textBox.setSelectionRange(startIndex,endIndex);

}

else if(textBox.createTextRange){

var range=textBox.createTextRange();

range.collapse(true);

range.moveStart("character",startIndex);

range.moveEnd("character",endIndex-startIndex);

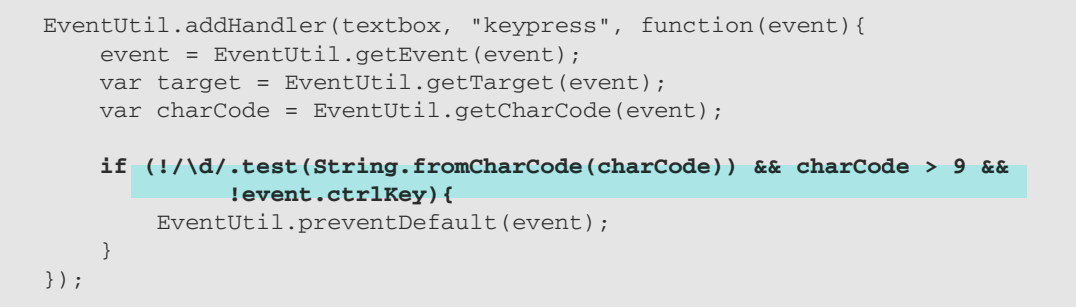
range.select();

}

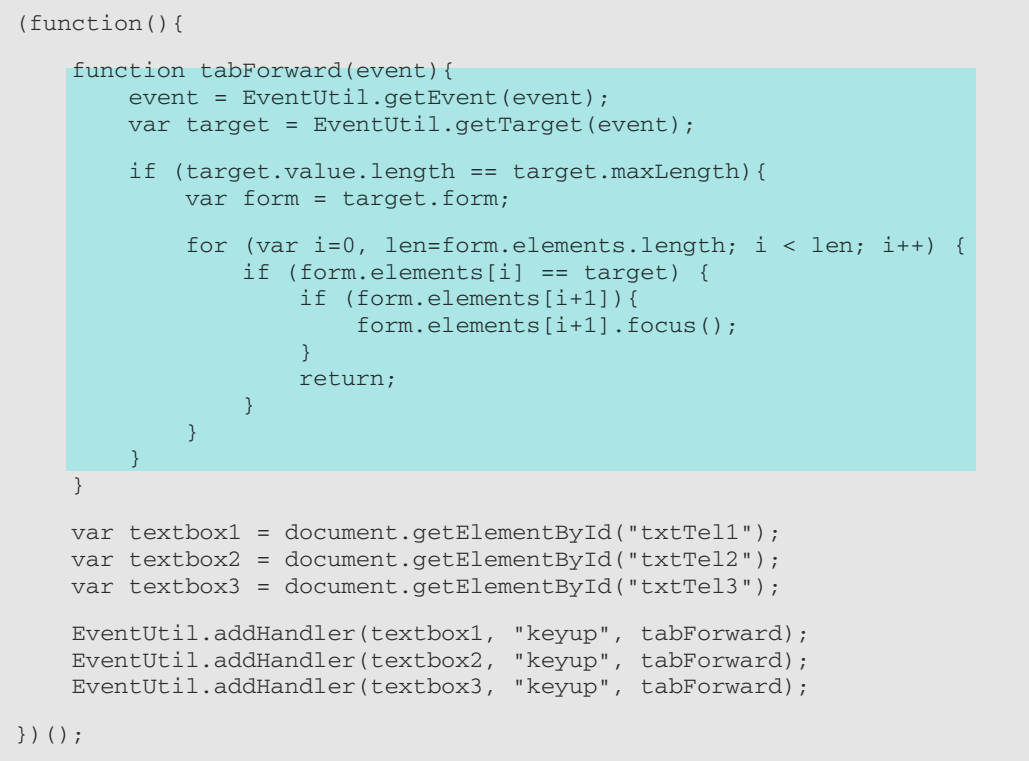
textBox.focus();

}

过滤输入



自动切换焦点



1. 约束api

必填字段

<input type="text" required>

输入类型：email url

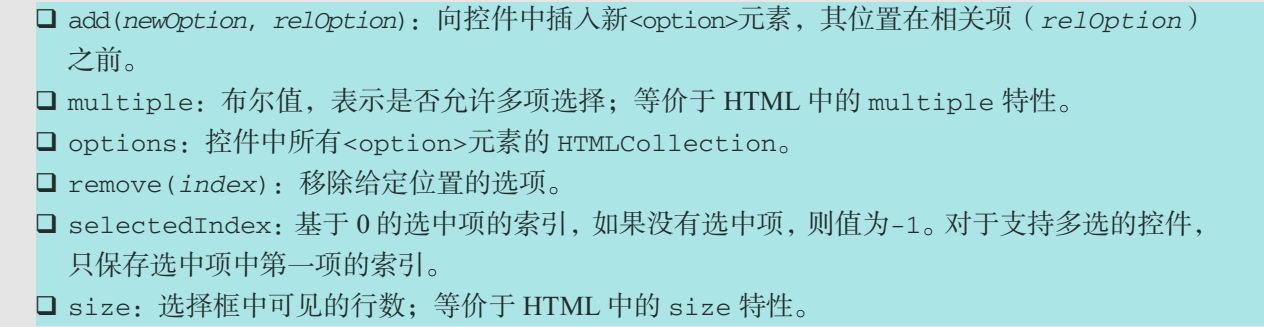
输入模式-pattern

检测有效性-checkValidity() validity

禁用验证-novalidate

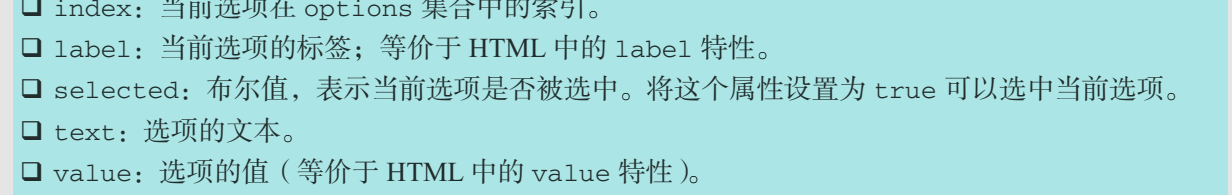
1. 选择框脚本

HTMLSelectElement的属性和方法



选择框的type属性，不是”select-one”，就是”select-multiple”

HTMLOptionElement的属性和方法



<select name="location" id="select1">

<option value="option1">choose1</option>

<option value="">choose2</option>

<option>choose3</option>

</select>

<script>

var oSelect=document.getElementById("select1");

console.log(oSelect[0].value);//option1

console.log(oSelect[1].value);//""

console.log(oSelect[2].value);//choose3

</script>

var text=oSelect.options[0].text;

var value=oSelect.options[0].value;

selectedIndex被选中元素的index

selected布尔值 元素是否被选中

getSelectedOptions(oSelect) 获取被选中的数组

添加选项

var oSelect=document.getElementById("select1");

var newOption=document.createElement("option");

newOption.appendChild(document.createTextNode("new choose"));

newOption.setAttribute("value","new select");

oSelect.appendChild(newOption);

var oSelect=document.getElementById("select1");

var newOption=new Option("new choose","new select");

oSelect.appendChild(newOption);//ie8及以前有问题

添加到其他位置用insertBefore

移除选项

oSelect.removeChild(oSelect.options[0]);

oSelect.remove(0);

oSelect.options[0]=null;

表单序列化

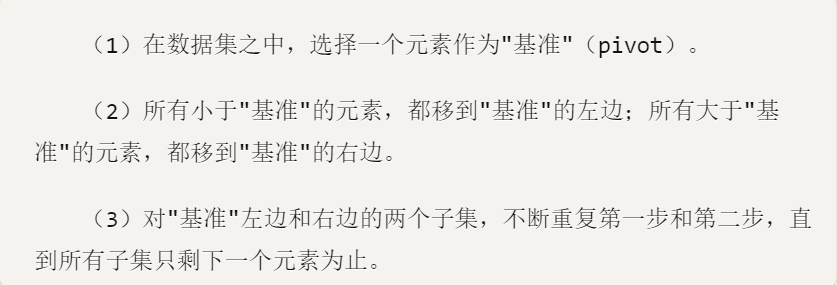
富文本编辑

1. vscode
2. 整段移动

选中按TAB右移，按SHIFT+TAB左移

1. 博客

1.快速排序



function quickSort(arr){

if(arr.length<=1){

return arr;

}

var pivotIndex=Math.floor(arr.length/2);//基准数的index

var pivot=arr.splice(pivotIndex,1)[0];

var left=[];

var right=[];

for(var i=0;i<arr.length;i++){

if(arr[i]<pivot){

left.push(arr[i]);

}

else{

right.push(arr[i]);

}

}

return quickSort(left).concat(pivot,quickSort(right));

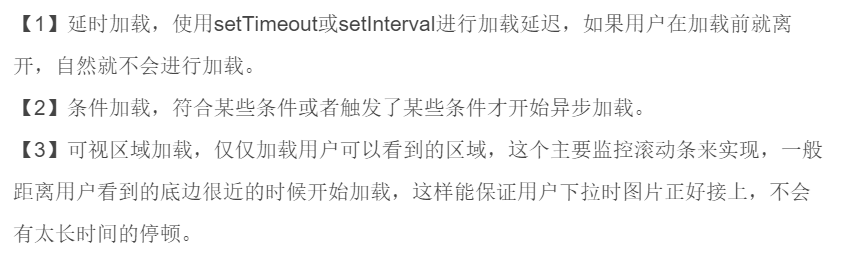
}

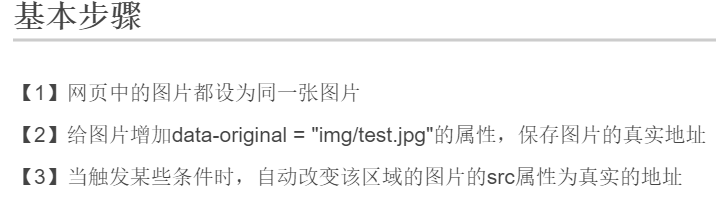
1. 字符串替换

str.replace(a,b);

1. 图片懒加载

图片延迟加载也称为懒加载，延迟加载图片或符合某些条件时才加载某些图片，通常用于图片比较多的网页。可以减少请求数或者延迟请求数，优化性能。





点击按钮显示图片

<button>加载图片</button>

<img src="#" alt="测试" data-original = "img/test.png">

<script>

var oBtn = document.getElementsByTagName('button')[0];

var oImg = document.images[0];

oBtn.onclick = function(){

oImg.src = "img/loading.gif";

if(oImg.dataset){

aftLoadImg(oImg,oImg.dataset.original);

}else{

aftLoadImg(oImg,oImg.getAttribute("data-original"));

}

}

function aftLoadImg(obj,url){

var oImg = new Image();

oImg.onload = function(){

obj.src = oImg.src;

}

oImg.src = url;

}

可视区显示图片

var oBtn = document.getElementsByTagName('button')[0];

var aImages = document.images;

loadImg(aImages);

window.onscroll = function(){

loadImg(aImages);

};

function loadImg(arr){

for( var i = 0,len = arr.length; i < len; i++){

if(arr[i].getBoundingClientRect().top < document.documentElement.clientHeight && !arr[i].isLoad){

arr[i].isLoad = true;

arr[i].style.cssText = "transition: ''; opacity: 0;"

if(arr[i].dataset){

aftLoadImg(arr[i],arr[i].dataset.original);

}else{

aftLoadImg(arr[i],arr[i].getAttribute("data-original"));

}

(function(i){

setTimeout(function(){

arr[i].style.cssText = "transition: 1s; opacity: 1;"

},16)

})(i);

}

}

}

function aftLoadImg(obj,url){

var oImg = new Image();

oImg.onload = function(){

obj.src = oImg.src;

}

oImg.src = url;

}

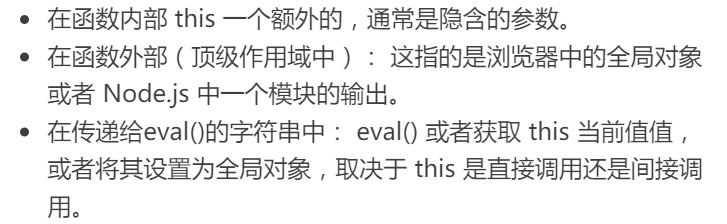
1. css选择器

CSS的一个核心特性是能向文档中的一组元素类型应用某些规则

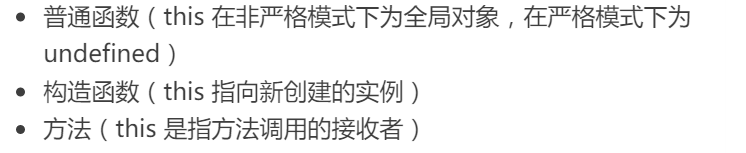
<http://www.cnblogs.com/xiaohuochai/p/4979514.html>

1. this

分类



* 1. 在函数中

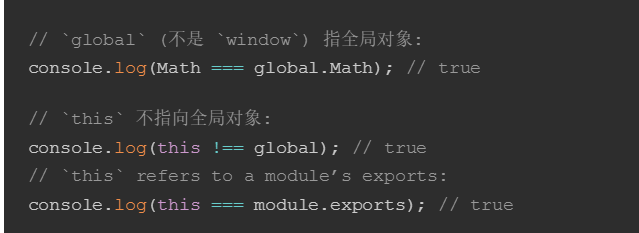


在普通函数中，可以通过call或apply进行函数调用，明确指定this的值

* 1. 在顶级作用域中

在浏览器环境下，this===window

在node环境下



* 1. 在eval中

间接调用eval()，this指向全局对象

直接调用，this与eval中的环境保持一致

2018.3.15

1. 红宝书第15章-使用Canvas绘图

<canvas id="myCanvas" width="200" height="200">A drawing of something</canvas>

<script>

var oCanvas=document.getElementById("myCanvas");

//确定浏览器支持canvas元素

if(oCanvas.getContext){

var context=oCanvas.getContext("2d");

//更多代码

}

</script>

1. 填充和描边

if(oCanvas.getContext){

var context=oCanvas.getContext("2d");

context.strokeStyle="red";

context.fillStyle="#0000ff";

}

fillStyle填充 strokeStyle描边

这2个属性的值可以是字符串、渐变对象、模式对象

1. 绘制矩形

填充矩形：

//绘制红色矩形

context.fillStyle="red";

//矩形的x坐标，矩形的y坐标，矩形的宽度，矩形的高度

context.fillRect(10,10,50,50);

//绘制半透明的蓝色矩形

context.fillStyle="rgba(0,0,255,0.5)";

context.fillRect(30,30,50,50);

描边矩形

//绘制红色描边矩形

context.strokeStyle="red";

//矩形的x坐标，矩形的y坐标，矩形的宽度，矩形的高度

context.strokeRect(10,10,50,50);

//绘制半透明的蓝色描边矩形

context.strokeStyle="rgba(0,0,255,0.5)";

context.strokeRect(30,30,50,50);

清除一个矩形区域

//绘制红色矩形

context.fillStyle="red";

//矩形的x坐标，矩形的y坐标，矩形的宽度，矩形的高度

context.fillRect(10,10,50,50);

//绘制半透明的蓝色矩形

context.fillStyle="rgba(0,0,255,0.5)";

context.fillRect(30,30,50,50);

//清除一个小矩形

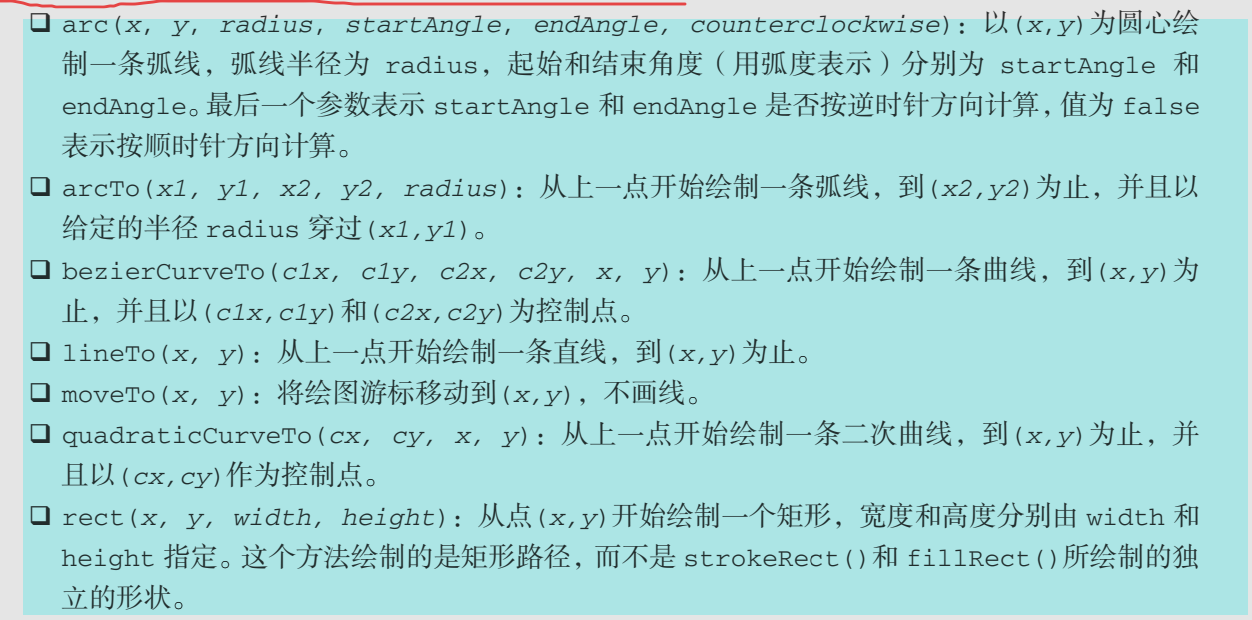
context.clearRect(40,40,10,10);

1. 绘制路径

首先，调用beginPath()

然后，绘制路径

最后，closePath()/fill()/stroke()/clip()



//开始路径

context.beginPath();

//绘制外圆

context.arc(100,100,99,0,2\*Math.PI,false);

//绘制内圆

context.moveTo(194,100);

context.arc(100,100,94,0,2\*Math.PI,false);

//绘制分针

context.moveTo(100,100);

context.lineTo(100,15);

//绘制时针

context.moveTo(100,100);

context.lineTo(35,100);

//描边路径

context.stroke();

context.isPointInPath(x,y);//用于在路径关闭之前确定画布的某一点是否位于路径上

1. 绘制文本

fillText() strokeText()

属性：

font 文本样式、大小及字体

textAlign 文本对齐方式 start center end

textBaseline 文本的基线 top middle bottom等

//正常

context.font="bold 14px Arial";

context.textAlign="center";

context.textBaseline="middle";

context.fillText("12",100,10);

//起点对齐

context.textAlign="start";

context.fillText("12",100,20);

//终点对齐

context.textAlign="end";

context.fillText("12",100,30);

var fontSize=100;

context.font=fontSize+"px Arial";

//在一个140px宽的矩形中绘制文本

while(context.measureText("Hello World").width>140){

fontSize--;

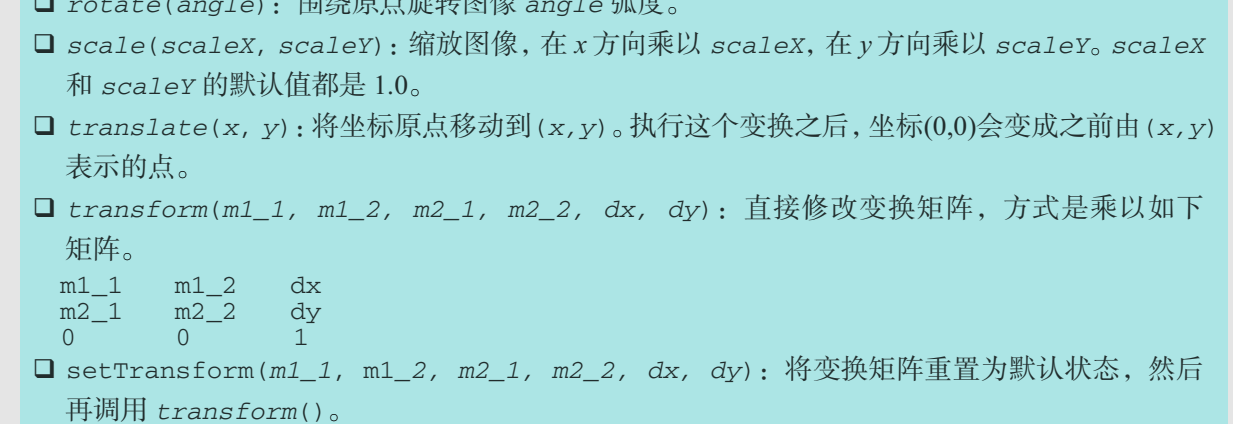
context.font=fontSize+"px Arial";

}

context.fillText("Hello World",0,30);

context.fillText("fontsize:"+fontSize+"px",0,50);

1. 变换



save()保存当前绘图上下文

restore()回到上一个保存的

1. 绘制图像

var image=document.images[0];

//(img,x,y)

context.drawImage(image,10,10);

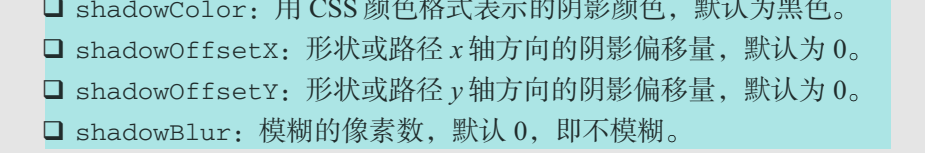
//(img,x,y,width,height)

context.drawImage(image,50,10,20,30);

//(img,x1,y1,width1,height1,x2,y2,width2,height2)

context.drawImage(image,0,10,50,50,0,100,40,60);

1. 阴影



1. 渐变

线性渐变

var gradient=context.createLinearGradient(30,30,70,70);

gradient.addColorStop(0,"white");

gradient.addColorStop(1,"black");

//绘制红色矩形

context.fillStyle="red";

context.fillRect(10,10,50,50);

//绘制渐变矩形

context.fillStyle=gradient;

context.fillRect(30,30,50,50);

径向渐变

//(x1,y1,r1,x2,y2,r2);

var gradient=context.createRadialGradient(55,55,10,55,55,30);

gradient.addColorStop(0,"white");

gradient.addColorStop(1,"black");

//绘制红色矩形

context.fillStyle="red";

context.fillRect(10,10,50,50);

//绘制渐变矩形

context.fillStyle=gradient;

context.fillRect(30,30,50,50);

1. 模式

pattern=context.createPattern(image,”repeat”);

1. 使用图像数据

getImageData(x,y,width,height)取得图像数据

每个imageData对象有3个属性：width height data data是一个数组：红绿蓝透明度

应用：彩色图像变灰白

1. 合成

globalAlpha全局透明度

globlCompositionOperation后绘制的图形怎么与先绘制的图像结合

1. 博客
2. css层叠

优先级：内联样式-id-class/属性选择/伪类-元素/伪元素-结合符/通配选择器

1. 重要声明

!important优先级最高

1. 继承样式

a标签不会继承父元素的文本样式

1. 手风琴效果

每个li有一个初始宽度 25%

hover某li时，此li变宽，其他变窄

离开时，恢复原来的宽度

2018.3.16

1. 红宝书的第16章-h5脚本编程
2. 跨文档消息传递

简称XDM，指的是在来自不同域的页面间传递消息

XDM的核心是postMessage()方法，2个参数-消息和表示消息来自哪个域的字符串

接收到XDM消息时，会触发window对象的message事件

message事件的事件对象event包含3个属性：data传入的数据、origin发送消息的文档所在的域、source发送消息的文档的window代理(只能调用postMessgae)

1. 原生拖放
   1. 文本只有在选中的时候可以拖动，图像和链接在任何时候都可以拖动。
   2. h5新增了一个属性-draggable true可以拖动 false不能拖动
   3. 拖动某元素时，依次触发dragstart drag dragend 。当某个元素被拖到一个有效的放置目标时，依次触发dargenter dragover dragleave/drop
   4. 取消默认行为

有些元素默认不能放置

EventUtil.addHandler(oDiv2,"dragover",function(event){

EventUtil.preventDefault(event);

})

EventUtil.addHandler(oDiv2,"dragenter",function(event){

EventUtil.preventDefault(event);

})

上面的代码改变了光标，要手动放置

在firefox3.5+中，放置事件的默认行为是打开被放置元素的url

EventUtil.addHandler(oDiv1,"drop",function(event){

EventUtil.preventDefault(event);

})

上面代码取消了默认事件

* 1. 在拖放时实现数据交换

dataTransfer对象有2个方法：getData(类型) setData(类型，数据)

保存在dataTransfer对象中的数据只能在drop事件处理程序中获取

* 1. 确定被拖动的元素以及作为放置目标的元素能够接收什么操作

dropEffect

effectAllowed

1. 媒体元素

video视频 audio音频

<video id="video1">

<source src="" id="">

<source src="" id="">

Video player not available

</video>

1. 历史状态管理

2018.3.18

1. 红宝书第20章—JSON
2. json是一种数据格式，不是一种编程语言。
3. json不支持变量、函数或对象实例。
4. json可以有3种类型的值：简单值、对象、数组

var json1="hello,world";

var json2={

"name":"ha",

"sex":"woman"

}

var json3=[

{

"name":"ha",

"sex":"woman"

},

{

"name":"hei",

"sex":"man"

}

]

JSON中对象的属性必须要用双引号

没有声明变量，没有末尾分号

1. stringify变成json字符串 parse变成原生js值

var book={

title:"my title",

authors:"cdx",

year:2018

}

var jsonText=JSON.stringify(book);

console.log(typeof jsonText);//string

console.log(jsonText);//{"title":"my title","authors":"cdx","year":2018}

var book={

title:"my title",

authors:"cdx",

year:2018

}

var jsonText=JSON.stringify(book);

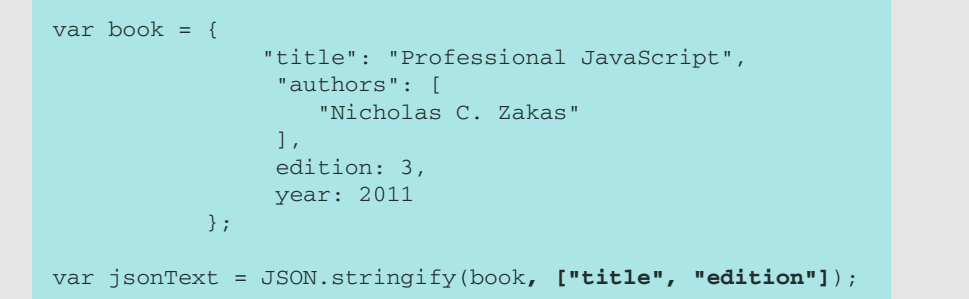
var str=JSON.parse(jsonText);

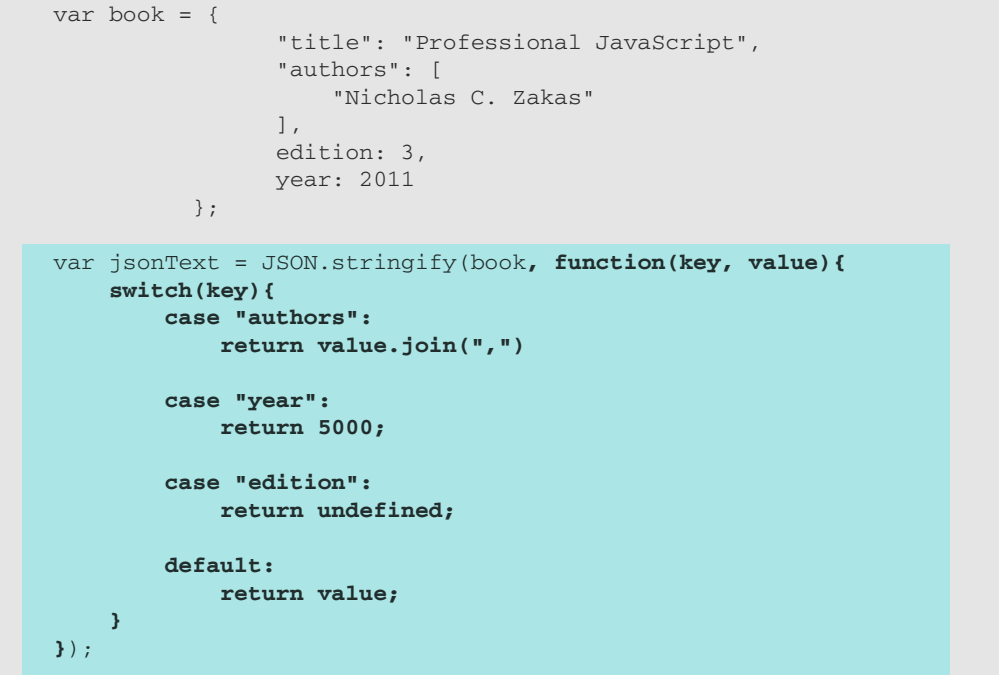
console.log(typeof str);//Object

console.log(str);

1. 序列化选项

stringify还可以接收另外2个参数，过滤器-数组或函数，是否在json字符串中保持缩进





字符串缩进，如果参数是一个数字，表示的是每级别缩进的空格数，最大缩进空格数是10。如果参数是一个字符，则被用作缩进字符，月不超过10个

1. toJSON()
2. JSON.parse()可接收另一个参数，函数，该函数参数是key,value。
3. 其他
4. shim和polyfilla

shim是一个库，它将一个新的API引入到另一个旧的环境中，而且仅靠旧环境中已有等等手段实现一个polyfill就是一个在浏览器api上面的shim。先检查当前浏览器是否支持某个api，如果不支持的话就加载对应的polyfill，然后新旧浏览器就都可以使用这个api了。

polyfill是shim的一种。shim是将不同api封装成一种，比如jqury的$.ajax。polyfill特指shim成的api是遵循标准的，其典型的做法是在IE浏览器中增加 window.XMLHttpRequest ，内部实现使用 ActiveXObject。在实际中为了方便做对比，会特指 shim 的 api 不是遵循标准的，而是自己设计的

1. 序列化

将json对象或表单元素等变成字符串

非数组对象的属性不能保证以特定的顺序出现在序列化后的字符串中。

布尔值、数字、字符串的包装对象在序列化过程中会自动转换成对应的原始值。

undefined、任意的函数以及 symbol 值，在序列化过程中会被忽略（出现在非数组对象的属性值中时）或者被转换成 null（出现在数组中时）。

所有以 symbol 为属性键的属性都会被完全忽略掉，即便 replacer 参数中强制指定包含了它们。

不可枚举的属性会被忽略

1. 原生类型(原始数据类型)

String Number Boolean undefined null

js数据类型划分-原始数据类型和引用类型

2018.3.19

1. 红宝书ajax
2. js的10个设计缺陷
3. 为什么有缺陷

设计阶段过于仓促。开发只用了10天，设计初衷只是为了解决简单的网页互动

没有先例。结合了函数式编程和面向对象编程。唯一使用prototype继承模型的语言

过早的标准化。Js推出一年半之后，国际标准就发布了。

1. 10个缺陷
   1. 不适合开发大型程序

没有命名空间，难模块化，允许重复定义，不利于模块加载

* 1. 非常小的标准库

标准库里的函数只能完成一些基本操作

* 1. null和undefined

null属于对象的一种，对象为空。undefined是一种数据类型，表示未定义。

console.log(typeof null);//object

console.log(typeof undefined);//undefined

var foo;

console.log(foo==null);//true

console.log(foo==undefined);//true

console.log(foo===null);//false

console.log(foo===undefined); //true

* 1. 全局变量难以控制。任何一个函数内部都可能产生全局变量
  2. 解释器为语句在行尾自动加分号
  3. +

console.log(1+1);//2

console.log("1"+"10");//110

console.log(1+"10");//110

console.log("10"+1);//101

* 1. NaN

console.log(NaN===NaN);//false

console.log(NaN!==NaN);//true

console.log(1+NaN);//NaN

* 1. 数组和对象的区分

判断一个对象是不是数组：

Array.isArray(arr)

obj instanceof Array

Object.propotype.toString.call(obj)===’[object Array]’

obj.constructor===Array

判断一个对象是否为函数

Object.propotype.toString.call(obj)===’[object Function]’

* 1. ==和===

用==时，会自动转换数据类型

* 1. 基本类型的包装对象

Boolean Number String

1. img

1.<img>并不会在网页上插入图像，而是从网页上链接图像。<img>标签创建的是被引用图像的占位空间

2.alt属性

图像替代文本，供搜索引擎抓取使用

1. MIME

描述消息内容类型的因特网标准

1. 抓包

把发送到和接收网络的信息拦截下来

2018.3.20

1. 红宝书22章
2. 作用域安全的构造函数

function Person(name,age,job){

this.name=name;

this.age=age;

this.job=job;

}

var person=Person("name1",20,"worker");

console.log(window.name);//name1

如果没有用new操作符调用构造函数，由于this对象是在运行时绑定是，this会映射到全局对象window上。

修改：

function Person(name,age,job){

if(this instanceof Person){

this.name=name;

this.age=age;

this.job=job;

}

else{

return new Person(name,age,job);

}

}

var person=Person("name1",20,"worker");

console.log(window.name);//""

console.log(person.name);//"name1"

如果使用构造函数窃取模式继承且不使用原型链，这个继承可能会被破坏。

使用构造函数窃取结合原型链或寄生组合可以解决这个问题。

function Polygon(sides){

if(this instanceof Polygon){

this.sides=sides;

this.getArea=function(){

return 0;

}

}

else{

return new Polygon(sides);

}

}

function Rectangle(width,height){

Polygon.call(this,2);

this.width=width;

this.height=height;

this.getArea=function(){

return this.width\*this.height;

}

}

Rectangle.prototype=new Polygon();

var rect=new Rectangle(5,10);

console.log(rect.sides);//2

1. 惰性载入函数

惰性载入函数表示函数的分支仅会发生一次。

实现方法：在函数被调用时再处理函数，在第一次调用的过程中，该函数会被覆盖为另一个按合适方式执行的函数。在分支里：fn=function(){}在声明时就指定适当的函数。在分支里：return function(){}

1. 函数绑定

只要是将某个函数指针以值的形式进行传递，同时该函数必须在特定环境中执行，被绑定函数的效用就显现出来了。

常常和回调函数和事件处理程序一起使用，以便在将函数作为变量传递的同时保留代码执行环境。

var handler={

message:"Event handled",

handleClick:function(event){

console.log(2);

console.log(this.message+":"+event.type);

}

}

var btn=document.getElementById("btn");

EventUtil.addHandler(btn,"click",handler.handleClick.bind(handler));

1. 函数柯里化

把接受多个参数的函数变换成接收单一参数的函数，并且返回接受余下的参数而且返回结果的新函数。

与函数绑定的共同点：使用闭包返回一个函数。区别：当函数被调用时，返回的函数还需要设置一些传入的参数。

1. 放篡改对象

一旦把对象定义为防篡改，就无法撤销了

* 1. 不可扩展对象

不能添加属性，可以修改和删除已经存在的属性

var person={name:"my name"};

Object.preventExtensions(person);

person.age=20;

console.log(person.age);//undefined

person.name="new name";

console.log(person.name);//new name

console.log(Object.isExtensible(person));//false

* 1. 密封的对象

不能添加和删除属性，可以修改属性值

被密封说明也不能扩展

var person={name:"my name"};

Object.seal(person);

person.age=20;

console.log(person.age);//undefined

person.name="new name";

console.log(person.name);//new name

console.log(Object.isSealed(person));//true

* 1. 冻结对象

不能增加和删除属性，也不能修改属性值

var person={name:"my name"};

Object.freeze(person);

person.age=20;

console.log(person.age);//undefined

person.name="new name";

console.log(person.name);//my name

console.log(Object.isFrozen(person));//true

被冻结说明不能扩展而且密封了

1. 定时器

在js中没有代码是立即执行的，但一旦进入空闲则尽快执行。

指定的时间间隔表示何时将定时器的代码添加到队列，而不是何时实际执行代码。

setTimeout(function(){

//

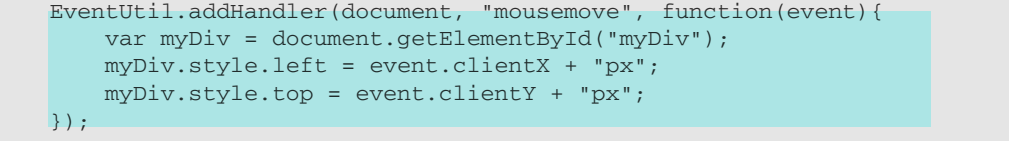
setTimeout(arguments.callee,interval);

},interval)

链式调用setTimeout 模仿setInterval 推荐这种

1. 拖放

鼠标拖尾：



拖放：p622

1. 博客
2. 检测类型

function isArray(obj){

return Object.prototype.toString.call(obj)=="[object Array]";

}

console.log(isArray([1,2,3]));//true

function isFunction(obj){

return Object.prototype.toString.call(obj)=="[object Function]";

}

var fn=function(){};

console.log(isFunction(fn));//true

function isRegExp(obj){

return Object.prototype.toString.call(obj)=="[object RegExp]";

}

console.log(isRegExp(/a/));//true

return window.JSON&&Object.prototype.toString.call(JSON)=="[object JSON]";//是否为原生json

1. css计数器

body,h2,h3,p{margin: 0;}

.show{counter-reset: c2;}

.show h2{counter-reset: c3 cp;counter-increment: c2;}

.show h3{counter-reset: cp;counter-increment: c3;text-indent: 2em;}

.show p{counter-increment: cp;text-indent: 4em;}

.show h2::before{content: counter(c2);}

.show h3::before{content: counter(c2) '.' counter(c3);}

.show p::before{content: counter(c2) '.' counter(c3) '.' counter(cp)}

counter-reset:计数器标识符 重置的数字 默认为none

counter-reset:c1 4 表示将c1的计数器重置为4

counter-reset c1 表示将c1的计数器重置为0

counter-increment 计数器标识符 递增的数字

counter-increment: c1 4 将c1的递增设为4

counter-increment: c1 将c1的递增设为1

还需要结合伪元素、cotent和counter()来使用

div::before{

content:counter(c1,upper-roman)

}

第一个参数是计数器

第二个参数是计数器样式 默认为数字 none disc circle square decimal lower-roman upper-roman lower-alpha upper-alpha lower-greek decimal-leading-zero

1. 伪元素

:first-letter

元素第一个字母

只能与块级元素相连

选择器与左大括号有空格时，ie6才支持

:first-line

元素第一行

只能与块级元素相连

选择器与左大括号有空格时，ie6才支持

:before :after

默认为行内元素

继承可继承的属性

ie7中必须声明!DOCTYPE否则不起作用

::selection

匹配用户选中的部分

firefox浏览器需要加-moz-前缀

只支持双冒号写法

只支持颜色和背景色2个样式

钉子：

.box::before{display: block;content: "钉子";height: 50px;width: 50px;border-radius: 50%;background-color: black;color: white;font-weight: bold;text-align: center;line-height: 50px;}

.box::after{display: block;content: "";width: 0;height: 0;border: 25px solid transparent;border-top: 50px solid black;margin-top: -20px;}

图片叠加效果

body{margin: 0;}

.box{position: relative;margin: 30px auto;width: 300px;}

.box-img{position: absolute;z-index: 1;border: 5px solid gray;width: 100%;}

.box::before,.box::after{content: "";position: absolute;background-color: palevioletred;width: 300px;height: 200px;border: 5px solid gray;}

.box::before{left: -10px;top: 0;transform: rotate(-5deg);}

.box::after{top: 4px;left: 0;transform: rotate(4deg);}

1. js jq css选择器

<https://www.cnblogs.com/ooo0/p/6115324.html>

1. 继承

<https://www.cnblogs.com/humin/p/4556820.html>

1. 原生js常用方法
2. 布尔值

假值：0 “” null undefined false NaN

其余都为真值

1. str+num



如果str里面有字母不能转换成数字，则结果为NaN

1. undefined void 0

undefined是不可靠的

undefined可当作局部变量并赋值，当undefined作为全局变量时不能赋值

var undefined;

function foo2(){

undefined=1;

console.log(undefined);//undefined

}

foo2();

所以应该用void 0 或void(0)代替undefined

1. 字符串也有length属性

不能用length来判断是否为数组，应该用Object.toString.call(obj)===’[object Array]’

1. 创建一个随机数组或者打乱数组

arr.sort(()=>{

return Math.random()-0.5;

})

arr.sort((a,b)=>{

return a>Math.random()\*10;

})

注：a-b返回从小到大排序

1. 去除前、后、前后空格

//type:1-所有空格 2-前后空格 3-前空格 4-后空格

function trim(str,type){

switch(type){

case 1:return str.replace(/\s+/g,"");

case 2:return str.replace(/(^\s\*)|(\s\*$)/g, "");

case 3:return str.replace(/(^\s\*)/g, "");

case 4:return str.replace(/(\s\*$)/g, "");

default:return str;

}

}

1. 首字母大/小写 全部大/小写

text-transform:capitalize每个单词首字母大写 uppercase全部大写 lowercase全小写

1. 重复字符串

console.log("abc".repeat(3));

1. 替换字符串中的一部分

console.log("abcd".replace(/a/,"1"));