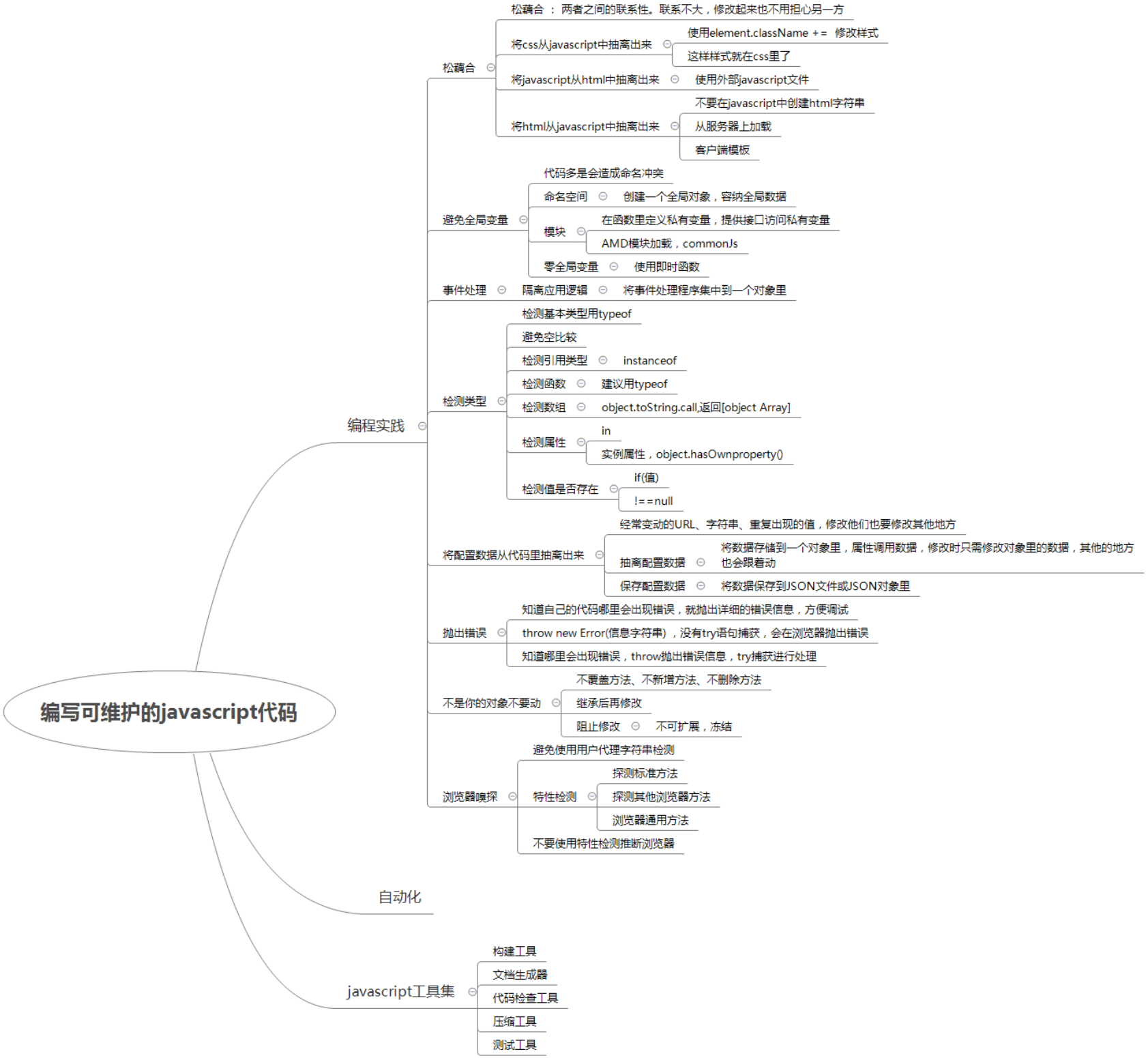
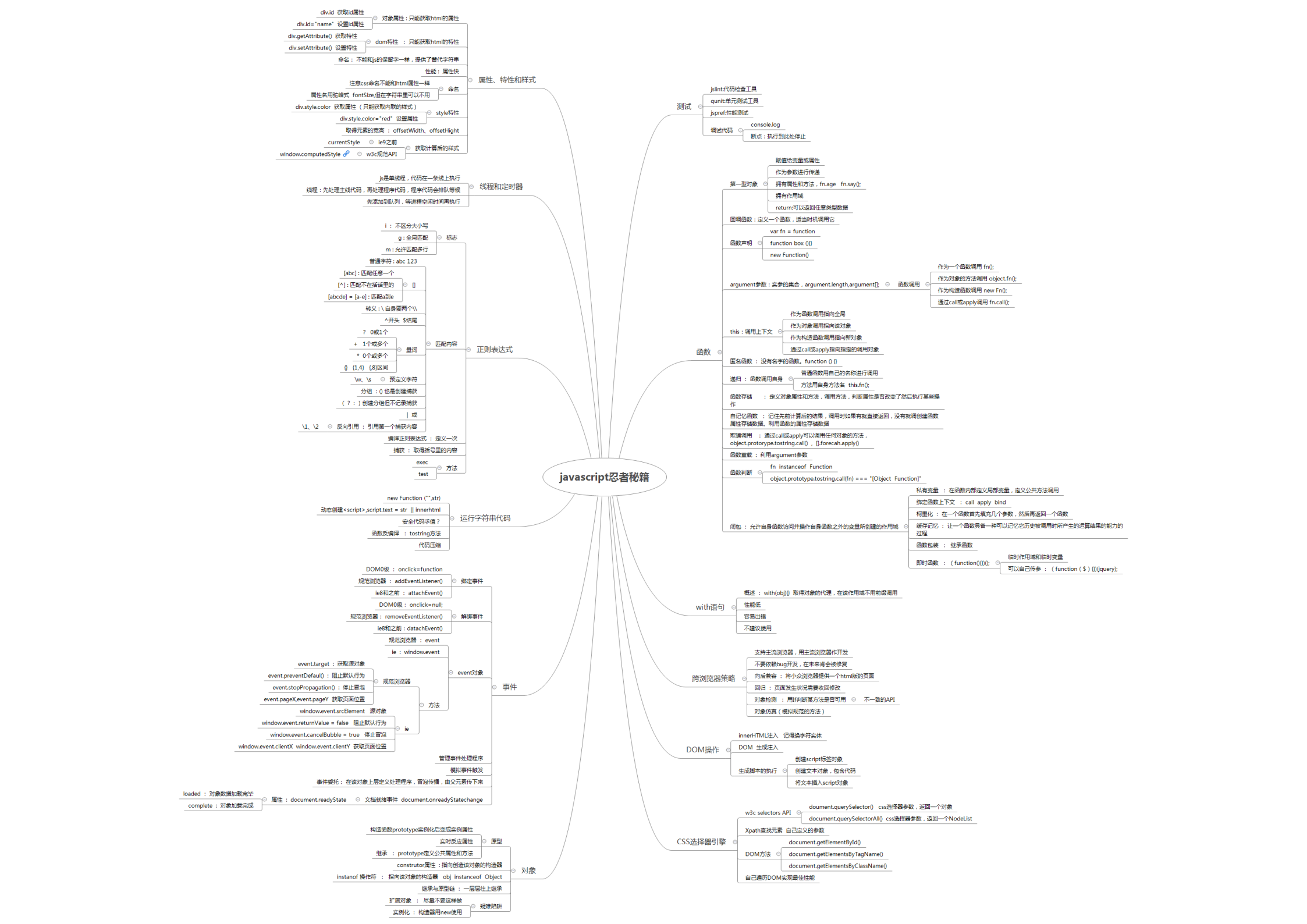
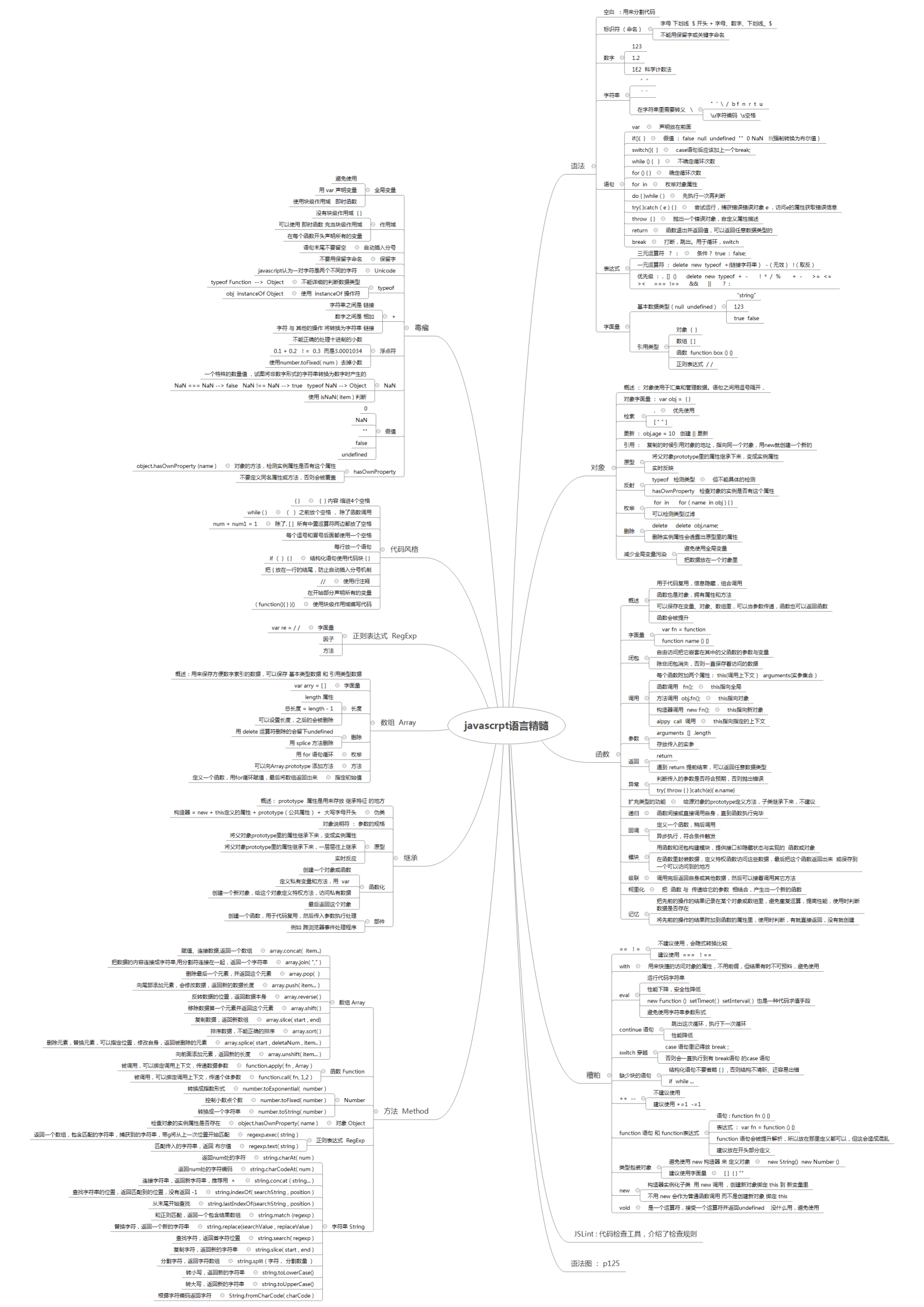

编写可维护的javascript代码
编程实践
松藕合
松藕合 ： 两者之间的联系性。联系不大，修改起来也不用担心另一方
将css从javascript中抽离出来
使用element.className += 修改样式
这样样式就在css里了
将javascript从html中抽离出来
使用外部javascript文件
将html从javascript中抽离出来
不要在javascript中创建html字符串
从服务器上加载
客户端模板
避免全局变量
代码多是会造成命名冲突
命名空间
创建一个全局对象，容纳全局数据
模块
在函数里定义私有变量，提供接口访问私有变量
AMD模块加载，commonJs
零全局变量
使用即时函数
事件处理
隔离应用逻辑
将事件处理程序集中到一个对象里
检测类型
检测基本类型用typeof
避免空比较
检测引用类型
instanceof
检测函数
建议用typeof
检测数组
object.toString.call,返回[object Array]
检测属性
in
实例属性，object.hasOwnproperty()
检测值是否存在
if(值)
!==null
将配置数据从代码里抽离出来
经常变动的URL、字符串、重复出现的值，修改他们也要修改其他地方
抽离配置数据
将数据存储到一个对象里，属性调用数据，修改时只需修改对象里的数据，其他的地方也会跟着动
保存配置数据
将数据保存到JSON文件或JSON对象里
抛出错误
知道自己的代码哪里会出现错误，就抛出详细的错误信息，方便调试
throw new Error(信息字符串) ，没有try语句捕获，会在浏览器抛出错误
知道哪里会出现错误，throw抛出错误信息，try捕获进行处理
不是你的对象不要动
不覆盖方法、不新增方法、不删除方法
继承后再修改
阻止修改
不可扩展，冻结
浏览器嗅探
避免使用用户代理字符串检测
特性检测
探测标准方法
探测其他浏览器方法
浏览器通用方法
不要使用特性检测推断浏览器
自动化
javascript工具集
构建工具
文档生成器
代码检查工具
压缩工具
测试工具


javascript忍者秘籍
属性、特性和样式
对象属性：只能获取html的属性
div.id 获取id属性
div.id="name" 设置id属性
dom特性 ：只能获取html的特性
div.getAttribute() 获取特性
div.setAttribute() 设置特性
命名： 不能和js的保留字一样，提供了替代字符串
性能： 属性快
命名
注意css命名不能和html属性一样
属性名用驼峰式 fontSize,但在字符串里可以不用
style特性
div.style.color 获取属性（只能获取内联的样式）
div.style.color="red" 设置属性
取得元素的宽高 ： offsetWidth、offsetHight
获取计算后的样式
currentStyle ie9之前
window.computedStyle w3c规范API
线程和定时器
js是单线程，代码在一条线上执行
线程：先处理主线代码，再处理程序代码，程序代码会排队等候
先添加到队列，等进程空闲时间再执行
正则表达式
匹配内容
标志
i ： 不区分大小写
g：全局匹配
m：允许匹配多行
普通字符：abc 123
[]
[abc]：匹配任意一个
[^]：匹配不在括话里的
[abcde] = [a-e]：匹配a到e
转义：\ 自身要两个\\
^开头 $结尾
量词
? 0或1个
+ 1个或多个
* 0个或多个
{} {1,4} {,8}区间
预定义字符
\w、\s
分组 ：() 也是创建捕获
(？：) 创建分组但不记录捕获
| 或
反向引用 ：引用第一个捕获内容
\1、\2
方法
编译正则表达式 ： 定义一次
捕获 ： 取得括号里的内容
exec
test
运行字符串代码
new Function ("",str)
动态创建<script>,script.text = str || innerhtml
安全代码求值？
函数反编译 ：tostring方法
代码压缩
事件
绑定事件
DOM0级 ： onclick=function
规范浏览器 ： addEventListener()
ie8和之前 ： attachEvent()
解绑事件
DOM0级： onclick=null;
规范浏览器： removeEventListener()
ie8和之前：datachEvent()
event对象
规范浏览器 ： event
ie ： window.event
规范浏览器
event.target ： 获取源对象
event.preventDefaul() ： 阻止默认行为
event.stopPropagation() ： 停止冒泡
event.pageX,event.pageY 获取页面位置
方法
ie
window.event.srcElement 源对象
window.event.returnValue = false 阻止默认行为
window.event.cancelBubble = true 停止冒泡
window.event.clientX window.event.clientY 获取页面位置
管理事件处理程序
模拟事件触发
事件委托： 在该对象上层定义处理程序，冒泡传播，由父元素传下来
属性 ： document.readyState
loaded ： 对象数据加载完毕
complete ： 对象加载完成
文档就绪事件 document.onreadystatechange
对象
原型
构造函数prototype实例化后变成实例属性
实时反应属性
继承 ： prototype定义公共属性和方法
construtor属性 ：指向创造该对象的构造器
instanof 操作符 ： 指向该对象的构造器 obj instanceof Object
继承与原型链 ： 一层层往上继承
疑难陷阱
扩展对象 ： 尽量不要这样做
实例化 ： 构造器用new使用
函数
第一型对象
赋值给变量或属性
作为参数进行传递
拥有属性和方法，fn.age fn.say();
拥有作用域
return:可以返回任意类型数据
回调函数：定义一个函数，适当时机调用它
函数声明
var fn = function
function box ()()
new Function()
argument参数：实参的集合，argument.length,argument[];
函数调用
作为一个函数调用 fn();
作为对象的方法调用 object.fn();
作为构造函数调用 new Fn();
通过call或apply调用 fn.call();
this：调用上下文
作为函数调用指向全局
作为对象调用指向该对象
作为构造函数调用指向新对象
通过call或apply指向指定的调用对象
匿名函数 ： 没有名字的函数。function () ()
递归 ： 函数调用自身
普通函数用自己的名称进行调用
方法用自身方法名 this.fn();
函数存储 ： 定义对象属性和方法，调用方法，判断属性是否改变了然后执行某些操作
自记忆函数 ： 记住先前计算后的结果，调用时如果有就直接返回，没有就调创建函数属性存储数据。利用函数的属性存储数据
靶猸调用 ： 通过call或apply可以调用任何对象的方法，object.protorype.tostring.call() ，[].forecah.apply()
函数重载 ： 利用argument参数
函数判断
fn instanceof Function
object.prototype.tostring.call(fn) === "[Object Function]"
闭包 ： 允许自身函数访问并操作自身函数之外的变量所创建的作用域
私有变量 ： 在函数内部定义局部变量，定义公共方法调用
绑定函数上下文 ： call apply bind
柯里化 ： 在一个函数首先填充几个参数，然后再返回一个函数
缓存记忆 ： 让一个函数具备一种可以记忆它历史被调用时所产生的运算结果的能力的过程
函数包装 ： 继承函数
即时函数 ： (function()())();
临时作用域和临时变量
可以自己传参 ： (function ($) ())(jquery);
with语句
概述 ： with(obj)() 取得对象的代理，在该作用域不用前缀调用
性能低
容易出错
不建议使用
跨浏览器策略
支持主流浏览器，用主流浏览器作开发
不要依赖bug开发，在未来肯会被修复
向后兼容 ： 将小众浏览器提供一个html版的页面
回归 ： 页面发生状况需要收回修改
对象检测 ： 用if判断某方法是否可用 不一致的API
对象仿真（模拟规范的方法）
DOM操作
innerHTML注入 记得换字符实体
DOM 生成注入
生成脚本的执行
创建script标签对象
创建文本对象，包含代码
将文本插入script对象
CSS选择器引擎
w3c selectors API
doument.querySelector() css选择器参数，返回一个对象
document.querySelectorAll() css选择器参数，返回一个NodeList
Xpath查找元素 自己定义的参数
DOM方法
document.getElementById()
document.getElementsByTagName()
document.getElementsByClassName()
自己遍历DOM实现最佳性能
测试
jslint:代码检查工具
qunit:单元测试工具
jspref:性能测试
调试代码
console.log
断点：执行到此处停止


javascrpt语言精髓
语法
空白 ：用来分割代码
标识符（命名）
字母 下划线 $ 开头 + 字母、数字、下划线_ $
不能用保留字或关键字命名
数字
123
1.2
1E2 科学计数法
字符串
" "
' '
在字符串里需要转义 \
" ' \ / b f n r t u
\u字符编码 \s空格
语句
var 声明放在前面
if( ) { } 假值 ： false null undefined "" 0 NaN !!(强制转换为布尔值)
switch( ) { } case语句后应该加上一个break;
while ( ) { } 不确定循环次数
for ( ) { } 确定循环次数
for in 枚举对象属性
do { }while ( ) 先执行一次再判断
try{ }catch (e) { } 尝试运行，捕获错误错误对象 e ，访问e的属性获取错误信息
throw { } 抛出一个错误对象，自定义属性描述
return 函数退出并返回值，可以返回任意数据类型的
break 打断，就出。用于循环，switch
表达式
三元运算符 ? : 条件？true：false;
一元运算符 ：delete new typeof +(链接字符串) -（无效）!（取反）
优先级 ：. [] () delete new typeof + - ! * / % + - >= <= >< === !== && || ? :
字面量
基本数据类型（null undefined）
"string"
123
true false
引用类型
对象 { }
数组 [ ]
函数 function box () {}
正则表达式 / /
对象
概述 ：对象使用于汇集和管理数据。语句之间用逗号隔开，
对象字面量 ：var obj = { }
检索
. 优先使用
[" "]
更新 ： obj.age = 10 创建 || 更新
引用 ： 复制的时候引用对象的地址，指向同一个对象，用new就创建一个新的
原型
将父对象prototype里的属性继承下来，变成实例属性
实时反映
反射
typeof 检测类型 但不能具体的检测
hasOwnProperty 检查对象的实例是否有这个属性
枚举
for in for (name in obj) { }
可以检测类型过滤
删除
delete delete obj.name;
删除实例属性会透露出原型里的属性
减少全局变量污染
避免使用全局变量
把数据放在一个对象里
函数
概述
用于代码复用，信息隐藏，组合调用
函数也是对象，拥有属性和方法
可以保存在变量、对象、数组里，可以当参数传递，函数也可以返回函数
函数会被提升
字面量
var fn = function
function name () {}
闭包
自由访问把它嵌套在其中的父函数的参数与变量
除非闭包消失，否则一直保存着访问的数据
调用
每个函数附加两个属性： this(调用上下文) arguments(实参集合)
函数调用 fn(); this指向全局
方法调用 obj.fn(); this指向对象
构造器调用 new Fn(); this指向新对象
alppy call 调用 this指向指定的上下文
参数
arguments [] .length
存放传入的实参
返回
return
遇到 return 提前结束，可以返回任意数据类型
异常
判断传入的参数是否符合预期，否则抛出错误
try{ throw { } }catch(e){ e.name}
扩充类型的功能 给原对象的prototype定义方法，子类继承下来，不建议
递归 函数间接或直接调用自身，直到函数执行完毕
回调
定义一个函数，稍后调用
异步执行，符合条件触发
模块
用函数和闭包构建模块，提供接口却隐藏状态与实现的 函数或对象
在函数里封装数据，定义特权函数访问这些数据，最后把这个函数返回出来 或保存到一个可以访问到的地方
级联 调用完后返回自身或其他数据，然后可以接着调用其它方法
柯里化 把 函数 与 传递给它的参数 相结合，产生出一个新的函数
记忆
把先前的操作的结果记录在某个对象或数组里，避免重复运算，提高性能，使用时判断数据是否存在
将先前的操作的结果附加到函数的属性里，使用时判断，有就直接返回，没有就创建
糟粕
== !=
不建议使用，会隐式转换比较
建议使用 === !==
with 用来快捷的访问对象的属性，不用前缀，但结果有时不可预料，避免使用
eval
运行代码字符串
性能下降，安全性降低
new Function () setTimeout( ) setInterval( ) 也是一种代码求值手段
避免使用字符串参数形式
continue 语句
跳出这次循环，执行下一次循环
性能降低
switch 穿越
case 语句里记得放 break ;
否则会一直执行到有 break语句 的case 语句
缺少块的语句
结构化语句不要省略 { } ，否则结构不清晰、还容易出错
if while ...
++ --
不建议使用
建议使用 +=1 -=1
function 语句 和 function表达式
语句：function fn () {}
表达式 ： var fn = function () {}
function 语句会被提升解析，所以放在那里定义都可以，但这会造成混乱
建议放在开头部分定义
类型包装对象
避免使用 new 构造器 来 定义对象 new String() new Number ()
建议使用字面量 [ ] { } ""
new
构造器实例化子类 用 new 调用 ，创建新对象绑定 this 到 新变量里
不用 new 会作为普通函数调用 而不是创建新对象 绑定 this
void 是一个运算符，接受一个运算符并返回undefined 没什么用，避免使用
JSLint：代码检查工具，介绍了检查规则
语法图 ： p125
毒瘤
全局变量
避免使用
用 var 声明变量
使用块级作用域 即时函数
作用域
没有块级作用域 { }
可以使用 即时函数 充当块级作用域
在每个函数开头声明所有的变量
自动插入分号
语句末尾不要留空
保留字
不要用保留字命名
Unicode
javascript认为一对字符是两个不同的字符
typeof
typeof Function --> Object 不能详细的判断数据类型
obj instanceOf Object 使用 instanceOf 操作符
+
字符串之间是 链接
数字之间是 相加
字符 与 其他的操作 将转换为字符串 链接
浮点符
不能正确的处理十进制的小数
0.1 + 0.2 != 0.3 而是3.0001034
使用number.toFixed(num) 去掉小数
NaN
一个特殊的数量值 ，试图将非数字形式的字符串转换为数字时产生的
NaN === NaN --> false NaN !== NaN --> true typeof NaN --> Object
使用 isNaN(item) 判断
假值
0
NaN
""
false
undefined
hasOwnProperty
object.hasOwnProperty (name) 对象的方法，检测实例属性是否有这个属性
不要定义同名属性或方法，否则会被覆盖
代码风格
{ } { 内容 缩进4个空格
while ( ) ( ) 之前放个空格 ，除了函数调用
num + num1 = 1 除了.[ ] 所有中置运算符两边都放了空格
每个逗号和冒号后面都使用一个空格
每行放一个语句
if ( ) { } 结构化语句使用代码块 { }
把 { 放在一行的结尾，防止自动插入分号机制
// 使用行注释
在开始部分声明所有的变量
(function(){ })() 使用块级作用域编写代码
正则表达式 RegExp
var re = / / 字面量
因子
方法
数组 Array
概述：用来保存方便数字索引的数据，可以保存 基本类型数据 和 引用类型数据
var arry = [ ] 字面量
length 属性
长度
总长度 = length - 1
可以设置长度，之后的会被删除
删除
用 delete 运算符删除的会留下undefined
用 splice 方法删除
枚举
用 for 语句循环
方法
可以向Array.prototype 添加方法
指定初始值
定义一个函数，用for循环赋值，最后将数组返回出来
继承
概述： prototype 属性是用来存放 继承特征 的地方
伪类
构造器 = new + this定义的属性 + prototype（公共属性） + 大写字母开头
对象说明符 ： 参数的规格
原型
将父对象prototype里的属性继承下来，变成实例属性
将父对象prototype里的属性继承下来，一层层往上继承
实时反应
函数化
创建一个对象或函数
定义私有变量和方法，用 var
创建一个新对象，给这个对象定义特权方法，访问私有数据
最后返回这个对象
部件
创建一个函数，用于代码复用，然后传入参数执行处理
例如 跨浏览器事件处理程序
方法 Method
数组 Array
赋值、连接数据,返回一个数组 array.concat(item..)
把数据的内容连接成字符串,用分割符连接在一起，返回一个字符串 array.join(",")
删除最后一个元素，并返回这个元素 array.pop( )
向尾部添加元素，会修改数据，返回新的数据长度 array.push(item...)
反转数据的位置，返回数据本身 array.reverse( )
移除数据第一个元素并返回这个元素 array.shift( )
复制数据，返回新数组 array.slice(start , end)
排序数据，不能正确的排序 array.sort( )
删除元素，替换元素，可以指定位置，修改自身，返回被删除的元素 array.splice(start , deleteNum , item..)
向前面添加元素，返回新的长度 array.unshift(item...)
函数 Function
被调用，可以绑定调用上下文，传递数据参数 function.apply(fn , Array)
被调用，可以绑定调用上下文，传递个体参数 function.call(fn, 1,2)
Number
转换成指数形式 number.toExponential(number)
控制小数点个数 number.toFixed(number)
转换成一个字符串 number.toString(number)
对象 Object
检查对象的实例属性是否存在 object.hasOwnProperty(name)
正则表达式 RegExp
返回一个数组，包含匹配的字符串，捕获到的字符串，带g将从上一次位置开始匹配 regexp.exec(string)
匹配传入的字符串，返回 布尔值 regexp.text(string)
字符串 String
返回num处的字符 string.charAt(num)
返回num处的字符编码 string.charCodeAt(num)
连接字符串，返回新字符串，推荐用 + string.concat (string...)
查找字符串的位置，返回匹配到的位置，没有返回 -1 string.indexOf(searchString , position)
从末尾开始查找 string.lastIndexOf(searchString , position)
和正则匹配，返回一个包含结果数组 string.match (regexp)
替换字符，返回一个新的字符串 string.replace(searchValue , replaceValue)
查找字符，返回首字符位置 string.search(regexp)
复制字符，返回新的字符串 string.slice(start , end)
分割字符，返回字符数组 string.split (字符， 分割数量)
转小写，返回新的字符串 string.toLowerCase()
转大写，返回新的字符串 string.toUpperCase()
根据字符编码返回字符 String.fromCharCode(charCode)

javascript高级程序设计

高级技巧

高级函数

返回对象 类型 ⊖ Object.prototype.toString.call(对象) ⊖ 安全的类型检测

作用域安全的构造函数 ⊖
```
if ( this instanceof Object) {
        this.name : "" ,
}else{
        return new Object ( name ,age );
}
```
如果不是，重新用 new 调用自己

惰性载入函数 ⊖ 在函数内部将可以执行的赋值给他自己，下次就不用判断了

函数绑定 ⊖ fn.bind(对象) ⊖ 绑定调用的作用域

函数柯里化 ⊖ 创建一个 已经填好某些参数 的函数

防篡改对象

对象不可扩展 ⊖ 不可扩展 ⊖ Object.preventExtensions(对象)

判断对象是否可扩展 ⊖ Object.isExtensible(对象)

密封对象 ⊖ 不可扩展，不可删除 ⊖ Object.seal(对象)

判断对象是否是密封对象 ⊖ Object.isSealed(对象)

冻结对象 ⊖ 不可扩展，不可修改，不可删除 ⊖ Object.freeze(对象)

判断 ' ⊖ Object.isFrozen(对象)

高级定时器

执行时机是不能保证的

代码会按照执行顺序添加如队列

javascript中没有任何代码是立刻执行的，但一旦进程空闲则尽快执行

重复定时器 ⊖ 用超时调用

先执行前面的，执行完后再进行超时调用

错误处理与调试

错误处理 ⊖ try{ }catch(e){ } finally{ } ⊖ 错误对象e有name和message信息属性

finally子句 ⊖ 无论如何到最后都会被执行

合理使用，明确知道代码会发生错误时，再用try-catch语句

错误类型

Error ⊖ 基类型，一般开发人员抛出

EvalError ⊖ 调用eval发生错误

RangeError ⊖ 范围错误，一般是数值

ReferenceError ⊖ 找不到对象

SyntaxError ⊖ 语法错误

TypeError ⊖ 类型错误

URIError ⊖ url错误

抛出错误

throw ⊖ throw {name : "错误",message : "数值不合法", }

应该在出现某种特定的已知错误条件，导致函数无法正常执行时抛出错误

建议重点关注函数和可能导致函数执行失败的因素，进行处理，而不是抛出错误

处理错误策略

建议使用 === ！==操作符

用if判断传入的参数是否符合预期

区分致命错误和非致命错误

非致命错误

不影响用户的主要任务

指影响页面的一部分

可以恢复

重复相同的操作可以消除错误

致命错误

应用程序根本无法运行

错误明显影响到用户的主要操作

会导致其他连带错误

调试技术

把消息记录到控制台 ⊖ console. ⊖ error( ) ⊖ 错误消息

info( ) ⊖ 信息性消息

log( ) ⊖ 一般消息

warn( ) ⊖ 警告消息

把消息记录到当前页面 ⊖ 在页面开辟一小块区域

创建元素，将信息插入元素里

然后插入区域里

断点

assert( ) 自定义函数调试

JSON

表示结构化数据

语法

简单值 ⊖ 数字

字符串

布尔值，null

对象 ⊖
```
{
        "name" : "bar",
        "age" : 16,
}
```
属性必须要加引号

数组 ⊖
```
[
        "fn",
        23,
]
```
对象与数组可以互相嵌套

解析与序列化 ⊖ JSON对象 ⊖ JSON.stringify(JSON数据，数组/函数，缩进选项) ⊖ 序列化为JSON字符串

JSON.parse(序列化后的JSON，fn(key,value)) ⊖ 把JSON字符串解析为原生javascript值

Ajax

异步请求数据，核心是XMLHttpRequest对象（XHR）

获取XHR对象 ⊖ var xhr = new XMLHttpRequest()

用法

xhr.open(请求类型，url,true异步) ⊖ 准备发送

xhr.send(发送的数据) ⊖ 开始发送

收到响应后自动填充XHR属性

xhr.responseText ⊖ 作为响应主体被返回的文本

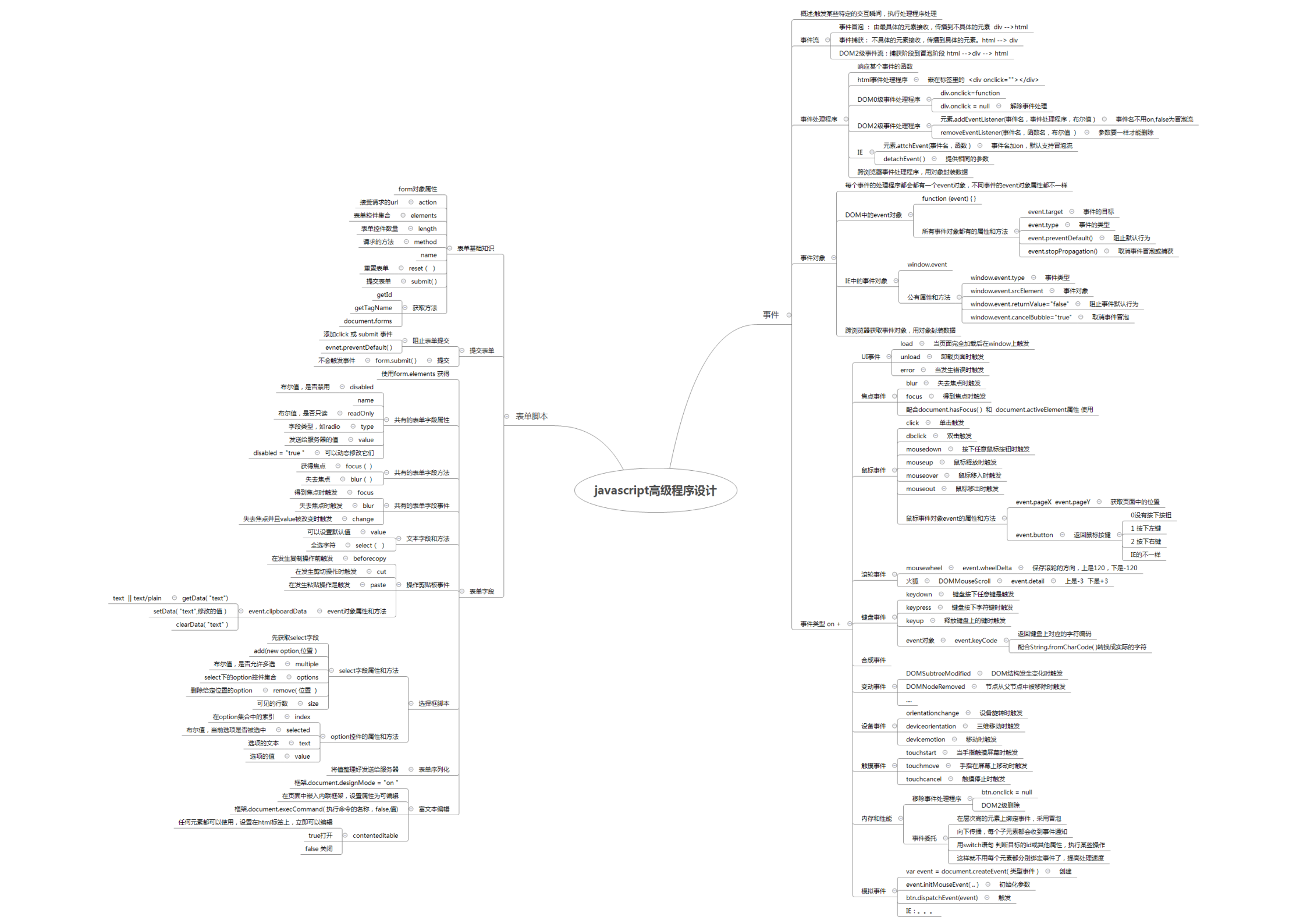
xhr.responseXML ⊖ 如果响应类型是XML，里面就保存着XML文档

xhr.status ⊖ 响应的HTTP状态码

xhr.statusText ⊖ HTTP状态的说明

为xhr绑定onreadystatechange事件 ⊖ 判断xhr.readyState === 4 ⊖ 再判断xhr.status >= 200 && xhr.status < 300 ⊖ 执行某些操作


javascript高级程序设计
函数表达式
匿名函数
没有名字的函数
闭包
概念：指 有权访问另一个函数作用域中的变量 的函数
当函数执行时，会创建一个预先包含 全局变量对象 和 自身作用域的变量 的作用域链
这个作用域链保存到内部[[Scope]]属性里
一般函数执行完毕，其活动对象会被销毁
但涉及但闭包，其活动对象就不会被销毁
因为匿名函数的作用域链仍然在引用这个活动对象，虽然作用域链被销毁了，但活动对象仍然对驻留在内存中，直到匿名函数被销毁
fn = null
内存泄漏
obj = null 用完的变量或其他的引用记得标记清除
模仿块级作用域
立即执行的匿名函数，执行后就会被销毁，外部也访问不了 (function () {})();
私有变量
定义一个函数
里面用 var 定义变量
用this定义公共方法进行操作私有变量,或定义一个对象，添加方法访问私有变量
最后返回这个对象
BOM
浏览器对象模型 window
窗口位置
获取窗口位置 window.screenLeft window.screenTop
火狐 window.screenX window.screenY
接收新的x,y坐标值 window.moveTo(x,y)
在水平或垂直移动的像素数 window.moveBy(x,y)
窗口大小
获取窗口大小
window.innerWidth window.innerHight
document.documentElement.clientWidth clientHeight
document.body.clientWidth clientHeight
调整大小
接收新的宽的和高度 window.resizeTo()
原来宽高差 window.resezeBy( )
窗口
window.open(URL，窗口目标，参数)
关闭窗口 window || self .close()
超时调用
var id = setTimeout(函数，时间)
清除调用 clearTimeout(id)
间歇调用
var = id setInterval(函数，时间)
clearInterva(id)
系统对话框
警告框 alert( )
返回布尔值 确认框 confirm (string)
返回输入的值 输入框 prompt (显示，默认值)
location 对象
导航
返回当前完整URL location.href
转到url location.href = url
刷新页面，true为从服务器刷新 location.reload( )
navigator 对象
浏览器
用户代理字符串，有关浏览器和平台信息 navigator.userAgent
history 对象
历史
跳转，-1为后退一页，1为前进一页 history.go( )
面向对象的程序设计
理解对象
对象数据的默认属性
能否通过delete删除属性 [Configurable]
能否通过for in 枚举属性 [Enumerable]
能否修改属性的值 [Writable]
访问器属性
读取属性时调用函数 [Get]
写入属性时调用函数 [Set]
构造函数
标识符开头大写
内部用this定义属性
用 new 调用，这样this的值就变成了新对象的值
constructor属性，指向对象的构造函数
原型
每个对象都有一个prototype属性，保存共享的属性和方法
obj.prototype.say = "1"
子类将父类prototype里属性继承下来会变成实例属性
方法
判断对象之间是否存在这种关系 对象.isPrototypeOf(对象)
获取对象的原型 Object.getPrototypeOf(对象)
判断给定的属性是否存在于实例中 对象.hasOwnProperty(属性)
检测对象的实力属性和原型，是否有给定的属性 属性 in 对象
简单的原型写法
obj.prototype = {
constructor:obj, //可算
age:10, }
动态性
因为指向的是一个指针
修改父类的原型属性会反映到子类中
重写父类的原型会切断与子类的联系
不建议修改原型
创建对象
构造函数 + 原型
父类的实例属性定义私有属性，用this
原型定义共享属性
稳妥构造函数模式
适用安全的环境
父类定义一个对象
为对象定义属性和方法
最后返回出这个对象
继承
经典继承
定义一个构造函数Fn
定义一个构造函数，在内部用call调用Fn，将其属性拷过来
然后实例化
伪经典继承
定义一个构造函数Fn
定义一个用于被继承的函数Fn2
Fn2里用call方法将Fn在此作用域调用，Fn的属性被拷贝到Fn2里
将Fn2的prototype指向Fn的原型
再将Fn2的prototype.constructor指向Fn2
再定义Fn2自己的原型属性
然后实例化
原型式继承
返回这个对象的副本 Object.create(对象)
引用类型
Object
声明
var obj = new Object( )
obj.age = "10";
var obj = {
age : "10",
}
用来存储和传输数据 无序数据的集合，键值对
Array
用来存储有序的数据，任意类型的数据，适合用数字做索引的
声明
var arry = new Array ( )
var arry = [1,2]
检测数组
instanceof
Array.isArray(数组)
转换方法 toString() 返回以，隔开的数组值字符串
栈方法
push(item..) 把数据添加到数组的末尾
pop() 删除最后一项，并返回这个值
队列方法
shift() 移除第一项并返回该项
unshift(item..) 向数组前面添加数据，返回数组的长度
重排序方法
reverse ( ) 反转数组
sort() 排序数组，比较的是字符串，不准确
操作方法
concat(item...) 连接数组，返回新的数组
slice(num , num) 指定数值复制数组，返回新的数组
splice(删除位置，数值，替换的值)
删除 splice(0,1)
插入 splice(1,0,1)
替换 splice(1,1,2)
位置方法
indexOf(查找的项，位置) 返回项的值
lastIndexOf() 从后面开始查找
迭代方法
每个方法都接受一个函数，函数传三个参数 ：数组的值，项，数组本身
every() 条件。每一项都返回true，函数才返回true
filter() 根据给定的条件，返回符合条件的值所构成的数组
forEach() 没有返回值，利用迭代数组做某些操作
map() 操作值，返回被操作的值构成的数组
some() 条件。只要有一项返回true，函数就返回true
归并方法
reduce(fn(前一个值，当前值，项的索引，数组对象) 从开头开始遍历数组
reduceRight() 从后面开始遍历
Date
时间对象 var now = new Date (可以传日期数值) new Date("2017 7 6 ")
方法
getFullYear() 取得年份
getMonth() 取得月份 ， 0-11
getDate() 取得天数 1-31
getHours() 取得小时 0-23
getMinutes() 取得分钟 0-59
getSeconds() 取得秒数 0-59
RegExp
声明
g全局，匹配所有的，i不区分大小写，m多行
var re = new RegExp( )
var re = / /
方法
exec(要匹配的值)
专门为捕获设计的
返回数组
该数组有两个属性
index 匹配项在字符串的位置
input 应用的字符串
数组的第一项是与整个模式匹配的字符串
后面的是括号的捕获
在模式设置g标志后，匹配整个字符，每次匹配都会在前一个结果后继续匹配
test(要匹配的值) 传入的值是否和正则匹配，返回布尔值
Funtion
函数名实际上是指向函数对象的指针
函数声明会被提升，变量声明也是，但赋值没有被提升
函数内部参数
arguments 实参的集合，类数组对象
this 调用上下文，指向调用的对象
函数属性和方法
length 希望接受命名参数的个数
prototype 原型，保存共享属性和方法
apply(作用域，参数数组) 在特定的作用域中调用函数，实际上是设置函数体内的this对象的值
call(作用域，参数个体)
bind(对象) 创建一个函数的实例
this值被绑定到传入bind函数的对象
基本包装类型
虽然拥有属性和方法，但不能给他们定义属性
Boolean
true false
转换为布尔值 ！！value
Boolean(value)
Number var num = 1
String
var str = "1"
字符方法
charAt(数值) 查找特定位置的字符
charCodeAt() 转换为字符编码
+ 链接字符
操作方法
slice() 截取字符串
substr() 截取字符串
substring(数值，数值+1) 截取字符串，指定开头，结束不含那个值，所以要加1
方法
位置方法
indexOf(查找的字符) 返回字符所在的位置
lastIndexOf() 从后面开始查找
trim() 去除前置和后缀空格，返回副本
大小写转换
转大写 toUpperCase()
转小写 toLowerCase()
模式匹配
match(正则) 返回一个数组，和exec方法一样
replace(正则，替换的字符或函数)
指定g替换全部的匹配到的项
$n，值是捕获到的值，$1是第一个括号的值
String.fromCharCode() 将字符编码转换为字符
Global 全局对象 包含全局的属性，String Object...
Math
数学对象，数值操作
方法
比较数值
Math.min(24,34 ,23) 找到参数中最小的值
Math.max( ) 找到参数中最大的值
舍入方法
Math.floor( ) 向下舍入，不要小数
Math.ceil( ) 向上舍入，增加1
Math.round() 标准舍入，四舍五入
Math.random() 返回0到1的随机数


javascript高级程序设计
表单脚本
表单基础知识
form对象属性
接受请求的url
action
表单控件集合
elements
表单控件数量
length
请求的方法
method
name
重置表单
reset ( )
提交表单
submit( )
获取方法
getId
getTagName
document.forms
提交表单
阻止表单提交
添加click 或 submit 事件
evnet.preventDefault( )
不会触发事件
form.submit( )
提交
表单字段
共有的表单字段属性
使用form.elements 获得
布尔值，是否禁用
disabled
name
布尔值，是否只读
readOnly
字段类型，如radio
type
发送给服务器的值
value
disabled = "true "
可以动态修改它们
共有的表单字段方法
获得焦点
focus ( )
失去焦点
blur ( )
共有的表单字段事件
得到焦点时触发
focus
失去焦点时触发
blur
失去焦点并且value被改变时触发
change
文本字段和方法
可以设置默认值
value
全选字符
select ( )
操作剪贴板事件
在发生复制操作前触发
beforecopy
在发生剪切操作时触发
cut
在发生粘贴操作是触发
paste
event.clipboardData
event对象属性和方法
text || text/plain
getData("text")
setData("text",修改的值)
clearData("text")
选择框脚本
select字段属性和方法
先获取select字段
add(new option,位置)
multiple
布尔值，是否允许多选
options
select下的option控件集合
remove(位置)
删除给定位置的option
size
可见的行数
option控件的属性和方法
index
在option集合中的索引
selected
布尔值，当前选项是否被选中
text
选项的文本
value
选项的值
表单序列化
将值整理好发送给服务器
富文本编辑
designMode = "on "
框架.document.designMode = "on "
在页面中嵌入内联框架，设置属性为可编辑
execCommand(执行命令的名称，false,值)
框架.document.execCommand(执行命令的名称，false,值)
contenteditable
任何元素都可以使用，设置在html标签上，立即可以编辑
true打开
false 关闭
事件
概述:触发某些特定的交互瞬间，执行处理程序处理
事件流
事件冒泡 ： 由最具体的元素接收，传播到不具体的元素 div --> html
事件捕获： 不具体的元素接收，传播到具体的元素。html --> div
DOM2级事件流：捕获阶段到冒泡阶段 html --> div --> html
事件处理程序
响应某个事件的函数
html事件处理程序
嵌在标签里的
DOM0级事件处理程序
div.onclick=function
div.onclick = null
解除事件处理
DOM2级事件处理程序
元素.addEventListener(事件名，事件处理程序，布尔值)
事件名不用on,false为冒泡流
removeEventListener(事件名，函数名，布尔值)
参数要一样才能删除
IE
元素.attchEvent(事件名，函数)
事件名加on，默认支持冒泡流
detachEvent( )
提供相同的参数
跨浏览器事件处理程序，用对象封装数据
事件对象
每个事件的处理程序都会都有一个event对象，不同事件的event对象属性都不一样
DOM中的event对象
function (event) { }
所有事件对象都有的属性和方法
event.target
事件的目标
event.type
事件的类型
event.preventDefault()
阻止默认行为
event.stopPropagation()
取消事件冒泡或捕获
IE中的事件对象
window.event
公有属性和方法
window.event.type
事件类型
window.event.srcElement
事件对象
window.event.returnValue="false"
阻止事件默认行为
window.event.cancelBubble="true"
取消事件冒泡
跨浏览器获取事件对象，用对象封装数据
事件类型 on +
UI事件
load
当页面完全加载后在window上触发
unload
卸载页面时触发
error
当发生错误时触发
焦点事件
blur
失去焦点时触发
focus
得到焦点时触发
配合document.hasFocus( ) 和 document.activeElement属性 使用
鼠标事件
click
单击触发
dbclick
双击触发
mousedown
按下任意鼠标按钮时触发
mouseup
鼠标释放时触发
mouseover
鼠标移入时触发
mouseout
鼠标移出时触发
鼠标事件对象event的属性和方法
event.pageX event.pageY
获取页面中的位置
event.button
返回鼠标按键
0没有按下按钮
1 按下左键
2 按下右键
IE的不一样
滚轮事件
mousewheel
event.wheelDelta
保存滚轮的方向，上是120，下是-120
火狐
DOMMouseScroll
event.detail
上是-3 下是+3
键盘事件
keydown
键盘按下任意键是触发
keypress
键盘按下字符键时触发
keyup
释放键盘上的键时触发
event对象
event.keyCode
返回键盘上对应的字符编码
配合String.fromCharCode( )转换成实际的字符
合成事件
变动事件
DOMSubtreeModified
DOM结构发生变化时触发
DOMNodeRemoved
节点从父节点中被移除时触发
....
设备事件
orientationchange
设备旋转时触发
deviceorientation
三维移动时触发
devicemotion
移动时触发
触摸事件
touchstart
当手指触摸屏幕时触发
touchmove
手指在屏幕上移动时触发
touchcancel
触摸停止时触发
内存和性能
移除事件处理程序
btn.onclick = null
DOM2级删除
事件委托
在层次高的元素上绑定事件，采用冒泡
向下传播，每个子元素都会收到事件通知
用switch语句 判断目标的id或其他属性，执行某些操作
这样就不用每个元素都分别绑定事件了，提高处理速度
模拟事件
var event = document.createEvent(类型事件)
创建
event.initMouseEvent(..)
初始化参数
btn.dispatchEvent(event)
触发
IE：. . .