### **全局解析规则**

### **函数内部解析规则**

### **变量提升**

定义变量的时候，变量的声明(使用var 申明的变量)会被提升到作用域的前面，变量的赋值不会提升

### **函数提升**

### JavaScript解析器首先会把当前作用域的函数声明(函数表达式提升的是前面的变量)提前到整个作用域的前面

## **3. 对象(重点)**

### **为什么要有对象**

1.对象也是我们存储数据的一种常用方式

2.面向对象编程(js高级部分)

### **什么是对象**

具体的事物，一个具体的事物就会有行为和特征。

### **JavaScript中的对象**

JavaScript中的对象其实就是生活中对象的一个抽象

JavaScript的对象是无序属性的集合。

其属性可以包含基本值、对象或函数。对象就是一组没有顺序的值。我们可以把JavaScript中的对象想象成键值对，其中值可以是数据和函数。

对象的行为和特征

特征---属性 行为---方法

事物的特征在对象中用属性来表示。

事物的行为在对象中用方法来表示。

### **属性和方法**

如果一个变量属于一个对象所有，那么该变量就可以称之为该对象的一个属性，属性一般是名词，用来描述事物的特征

如果一个函数属于一个对象所有，那么该函数就可以称之为该对象的一个方法，方法是动词，描述事物的行为和功能

### **对象字面量** 字面量：11 'abc' true [] {}等

### **对象创建方式**

1. 对象字面量

var o = {

name: 'zs',

age: 18,

sex: true,

sayHi: function () {

console.log(this.name);

}

};

1. new Object()创建对象 （Object对象是所有对象的根对象）

var person = new Object();

person.name = 'lisi';

person.age = 35;

person.job = 'actor';

person.sayHi = function(){

console.log('Hello,everyBody');

}

1. 自定义构造函数

function Person(name,age,job){

this.name = name;

this.age = age;

this.job = job;

this.sayHi = function(){

console.log('Hello,everyBody');

}

}

var p1 = new Person('张三', 22, 'actor');

1. 工厂函数创建对象(工厂模式)

function createPerson(name, age, job) {

var person = new Object();

person.name = name;

person.age = age;

person.job = job;

person.sayHi = function(){

console.log('Hello,everyBody');

}

return person;

}

var p1 = createPerson('张三', 22, 'actor');

### **new关键字**

构造函数 ，是一种特殊的函数。主要用来在创建对象时初始化对象， 即为对象成员变量赋初始值，总与new运算符一起使用在创建对象的语句中。

(1.) 构造函数用于创建一类对象，首字母要大写。

(2.) 构造函数要和new一起使用才有意义。

**new在执行时会做的事情**

new会在内存中创建一个新的空对象

new 会让this指向这个新的对象

执行构造函数 目的：给这个新对象加属性和方法

new会返回这个新对象

### **this详解**

函数在定义的时候this是不确定的，只有在调用的时候才可以确定

2. 一般函数直接执行，内部this指向全局window

3. 函数作为一个对象的方法，被该对象所调用，那么this指向的是该对象

4. 构造函数中的this其实是一个隐式对象，类似一个初始化的模型，所有方法和属性都挂载到了这个隐式对象身上，后续通过new关键字来调用，从而实现实例化

## **对象的使用**

### **获取对象的属性和行为**

1. **获取对象的属性和行为有两种方式**

1.1 点语法 object.property

1.2 中括号语法 object['property']

注意:中括号语法中对象的属性名要加引号包裹起来

### **遍历对象的属性 （**通过for..in语法可以遍历一个对象**）**

### **删除对象的属性**

function fun() {

this.name = 'mm';

}

var obj = new fun();

console.log(obj.name); // mm

delete obj.name;

console.log(obj.name); // undefined

### **4.简单类型和复杂类型的区别**

基本类型又叫做值类型，复杂类型又叫做引用类型

值类型：简单数据类型，基本数据类型，在存储时，变量中存储的是值本身，因此叫做值类型。

引用类型：复杂数据类型，在存储是，变量中存储的仅仅是地址（引用），因此叫做引用数据类型。

堆和栈

堆栈空间分配区别：

　　1、栈（操作系统）：由操作系统自动分配释放 ，存放函数的参数值，局部变量的值等。其操作方式类似于数据结构中的栈；

　　2、堆（操作系统）： 存储复杂类型(对象)，一般由程序员分配释放， 若程序员不释放，由垃圾回收机制回收，分配方式倒是类似于链表。

JS基础 === day06

## **内置对象(重点)**

JavaScript 提供多个内置对象：Math/Array/Number/String/Boolean...

### **Math对象**

Math对象不是构造函数，它具有数学常数和函数的属性和方法

Math.PI // 圆周率

Math.random() // 生成随机数 0-1包含0,不包含1

Math.floor()/Math.ceil() // 向下取整/向上取整

Math.round() // 取整，四舍五入

Math.abs() // 绝对值

Math.max()/Math.min() // 求最大和最小值

Math.pow()/Math.sqrt() // 求指数次幂/求平方根

### **Date对象**

创建 Date 实例用来处理日期和时间。Date 对象基于1970年1月1日（世界标准时间）起的毫秒数。

// 获取当前时间，UTC世界时间，距1970年1月1日（世界标准时间）起的毫秒数

var now = new Date();

console.log(now.valueOf()); // 获取距1970年1月1日（世界标准时间）起的毫秒数

Date构造函数的参数

1. 毫秒数 1498099000356 new Date(1498099000356)

2. 日期格式字符串 '2015-5-1' new Date('2015-5-1')

3. 年、月、日…… new Date(2015, 4, 1) // 月份从0开始

获取日期的毫秒形式

var now = new Date();

// valueOf用于获取对象的原始值

console.log(now.valueOf())

// HTML5中提供的方法，有兼容性问题

var now = Date.now();

// 不支持HTML5的浏览器，可以用下面这种方式

var now = + new Date(); // 调用 Date对象的valueOf()

日期格式化方法

toString() // 转换成字符串

valueOf() // 获取毫秒值

// 下面格式化日期的方法，在不同浏览器可能表现不一致，一般不用

toDateString() //只有日期

toTimeString() //只有时间

toLocaleDateString() //本地日期

toLocaleTimeString()// 本地时间

获取日期指定部分

getTime() // 返回毫秒数和valueOf()结果一样，valueOf()内部调用的getTime()

getSeconds() // 返回0-59

getMinutes() // 返回0-59

getHours() // 返回0-23

getDay() // 返回星期几 0周日 6周6

getDate() // 返回当前月的第几天

getMonth() // 返回月份，\*\*\*从0开始\*\*\* 0 - 11

getFullYear() //返回4位的年份 如 2017

### **Array对象**

1.检测一个对象是否是数组

x instanceof Array 如果x是数组则返回true,否则返回false

Array.isArray(x) 如果x是数组则返回true,否则返回false HTML5中提供的方法，有兼容性问题

**数组常用方法**

（1）**数组转换字符串** array.join() 作用：将数组的值拼接成字符串

var arr = [1,2,3,4,5];

arr.join(); //不传参数，默认按【,】进行拼接

arr.join("-"); //参数是字符串类型 按【-】进行拼接

（2）**数组的增删操作**

array.push ==== 》 将一个或多个元素**添加到数组的结尾**，并**返回新的长度。**

array.pop() ===》 从数组中**删除最后一个元素，并返回该元素的值,改变了数组的长度**

array.unshift() ===> 将一个或多个元素**添加到数组的开头，并返回新的长度。**

array.shift() === >从数组中**删除第一个元素，并返回该元素的值,改变了数组的长度**

（3）**数组的翻转与排序**

array.reverse() ==== 》翻转数组,返回翻转过的数组

array.sort() ==== 》默认排序顺序是根据字符串Unicode码点。

var arr = ["ba","bb","a"]; //arr = [1,2,10,4,40,5];

arr.sort(function(a,b){

return a-b; //按照正序

//return b-a //按照倒序

})

（4）**数组的拼接与截取**

concat： 数组合并，**不会影响原来的数组**，**会返回一个新数组**。

slice: 复制数组的一部分到一个新数组，**并返回这个新数组 原来的数组不受影响，包含头，不包含尾** var newArray = array.slice(begin的索引, end索引);

splice: **以新元素来替换旧元素**，以此来修改数组的内容，**返回被替换的内容,原数组被改变**

//start:开始位置 deleteCount:删除的个数 items:替换的内容

array.splice(start, deleteCount, ‘item1’，‘item2’，‘item3’);

（5）**数组查找元素**

indexOf(要查找的元素，开始查的下标)**方法用来查找数组中某个元素第一次出现的位置，如果找不到，返回-1**

lastIndexOf(要查找的元素，开始查的下标)**从后面开始查找数组中元素出现位置,如果找不到，返回-1**

**（6）操作数组里的元素**

1.filter方法返回一个由符合函数要求的元素组成的新数组

element:表示数组里的元素; //index:表示索引值; //array:表示传入的数组

**var newArr = arr.filter(function(element,index,array){}**

2.forEach() 方法对数组的每个元素执行一次提供的函数,且这个函数没有返回值

**var result = arr.forEach(function (element, index, array) {}**

3.some() 方法测试数组中的某些元素是否通过由提供的函数实现的测试.

**result = arr.some(function (element,index,array) {}**

4.every() 方法测试数组的所有元素是否都通过了指定函数的测试

**result = arr.every(function (element,index,array) {}**

1. **清空数组**

**// 方式1 推荐 ==== 》 arr = [];**

**// 方式2 ======》arr.length = 0;**

**// 方式3 ===== 》arr.splice(0, arr.length);**

### **基本包装对象**

### **String对象**

1. **字符串的不可变**
2. **创建字符串对象 ==== 》var str = new String('Hello World');**
3. **字符串对象的常用方法**

**1 字符方法**

charAt() //获取指定位置处字符

str[0] //HTML5，IE8+支持 和charAt()等效

fromCharCode() 返回一个字符串

String.fromCharCode(101, 102, 103); 把ASCII码转换成字符串

**2 字符串操作方法 (常用)**

**查找指定字符串**

indexOf: 获取某个字符第一次出现的位置，如果没有，返回-1

lastIndexOf:从后面开始查找第一次出现的位置。如果没有，返回-1

**去除空白**

trim() ==== 》 去除字符串两边的空格，内部空格不会去除

**大小写转换**

toUpperCase() === 》全部转换成大写字母

toLowerCase() === 》全部转换成小写字母

**字符串拼接与截取**

**字符串拼接**

**1.可以用concat，**用法与数组一样(返回新的字符串)，但是字符串拼串我们一般都用 + 号

**slice ：**从start开始，end结束，包含头,不包含尾 ,返回一个新的字符串,原字符串不变

**substring ：**从start开始，end结束，包含头,不包含尾 ,返回一个新的字符串,原字符串不变

**substr ：**从start开始，截取length个字符。(推荐)

**-2.字符串切割**

**split:将字符串分割成数组,原字符串不改变（很常用）**

var str = "张三,李四,王五";

var arr = str.split(",");

**- 3.字符串替换**

**replace(searchValue, replaceValue)**

参数：searchValue:需要替换的值 replaceValue:用来替换的值

**//注意,返回一个新的字符串,原来的字符串没有改变**

var str = "abcd";

var newStr = str.replace("d","aaaa")

console.log(str);//abcd

console.log(newStr)//abcaaaa