JavaScript

一、 目录

二、 基础语法

2.1、输出语句

页面输出	document.write("~")
控制台输出	console.log/info/warn/eror
弹出窗口输出	alert("~~ ~~")

2.2、数据类型

2.2.1、typeof运算符

（检测变量的数据类型）

> 返回值为$String$类型

转义字符	含义
\n	换行
\t	制表
\b	空格
\r	回车
\ \	斜杠
\ '	单引号
\ "	双引号

2.2.2、Number类型

> 整数和浮点数

正无穷	Infinity
负无穷	-Infinity
非法数字	NaN

return

2.3、强制类型转换

2.3.1、转换为String类型

toString()
String()
+" "

2.3.2、转换为Number类型

Number（）	任意
parseInt()	整数
parseFloat()	浮点数

2.3.3、转换为布尔型

Boolean（）

return

2.4、对象

> (object)

2.4.1、创建对象

```
var person = new Object();
person.name = "aaa";
person.age = 18;
```

```
var person = {
    name: "aaa",
    age: 18
};
```

2.4.2、访问方法

```
//第一种
person.属性名

//第二种
person['属性名']
```

2.4.3、删除属性

```
delete person.属性名
```

2.4.4、遍历对象

```
var person = {
 name: "zhangsan",
 age: 18
}

for (var personKey in person) {
 var personVal = person[personKey];
 console.log(personKey + ":" + personVal);
}
```

return

2.5、函数

2.5.1、函数创建

```
//第一种
function 函数名([形参1, 形参2]) {
    ......
}

//第二种
var 函数名 = function([形参1, 形参2]) {
    ......
}
```

2.5.2、函数调用

```
/函数名([......])
```

2.5.3、嵌套函数

嵌套函数只能在当前函数访问

```
function fu() {
 function zi() {
     console.log("我是儿子")
 }f
 zi();
}
fu();
/*
嵌套函数只能在当前函数访问
*/
```

2.5.4、匿名函数

```
(function(){
         //立即执行
         ....
     })
```

2.5.5、匿名函数立即执行

```
(function(){
        //不立即执行
        ....
})()
```

2.5.6、对象中的函数

```
var person = {
 name: "zhangsan",
 age: 18,
 sayHello: function () {
     console.log(name + " hello")
 }
}

person.sayHello();
```

2.5.7、构造函数

```
// 使用构造函数来创建对象
function Person(name, age) {
 // 设置对象的属性
 this.name = name;
 this.age = age;
 // 设置对象的方法
 this.sayName = function () {
     console.log(this.name);
 };
}

var person1 = new Person("孙悟空", 18);
var person2 = new Person("猪八戒", 19);
var person3 = new Person("沙和尚", 20);
```

> 判断一个对象是不是另一个对象的子类
>
> *instanceof*

2.5.8、原型

```
// 使用构造函数来创建对象
function Person(name, age) {
 // 设置对象的属性
 this.name = name;
 this.age = age;
}

// 在Person类的原型对象中添加方法
Person.prototype.sayName = function() {
 console.log(this.name);
};

var person1 = new Person("孙悟空", 18);
var person2 = new Person("猪八戒", 19);
var person3 = new Person("沙和尚", 20);
```

```
person1.sayName();
person2.sayName();
person3.sayName();
```

[return]

2.5.9、toString()方法

类型	行为描述
String	返回String
Number	返回Number
Boolean	返回布尔值
Object	返回对象名称
Array	数组逗号隔开
Date	日期
Error	错误信息的字符串
Function	返回函数名称

2.5.10、hasOwnProperty方法

判断是否含有属性或者方法，使用in运算符检查

hasOwnProperty

```
var mc=new MyClass();
console.log("age" in mc);
```

2.5.11、继承

1，原型链继承

```
// 定义父类型构造函数
function SupperType() {
 this.supProp = 'Supper property';
}

// 给父类型的原型添加方法
SupperType.prototype.showSupperProp = function () {
 console.log(this.supProp);
};

// 定义子类型的构造函数
function SubType() {
 this.subProp = 'Sub property';
}

// 创建父类型的对象赋值给子类型的原型
SubType.prototype = new SupperType();

// 将子类型原型的构造属性设置为子类型
```

```
SubType.prototype.constructor = SubType;

// 给子类型原型添加方法
SubType.prototype.showSubProp = function () {
 console.log(this.subProp)
};

// 创建子类型的对象: 可以调用父类型的方法
var subType = new SubType();
subType.showSupperProp();
subType.showSubProp();
```

2，构造函数继承

```
// 定义父类型构造函数
function SuperType(name) {
 this.name = name;
 this.showSupperName = function () {
     console.log(this.name);
 };
}

// 定义子类型的构造函数
function SubType(name, age) {
 // 在子类型中调用call方法继承自SuperType
 SuperType.call(this, name);
 this.age = age;
}

// 给子类型的原型添加方法
SubType.prototype.showSubName = function () {
 console.log(this.name);
};

// 创建子类型的对象然后调用
var subType = new SubType("孙悟空", 20);
subType.showSupperName();
subType.showSubName();
console.log(subType.name);
console.log(subType.age);
```

3，组合继承

```
function Person(name, age) {
 this.name = name;
 this.age = age;
}

Person.prototype.setName = function (name) {
 this.name = name;
};

function Student(name, age, price) {
 Person.call(this, name, age); // 为了得到父类型的实例属性和方法
 this.price = price; // 添加子类型私有的属性
}
```

```
Student.prototype = new Person(); // 为了得到父类型的原型属性和方法
Student.prototype.constructor = Student; // 修正constructor属性指向
Student.prototype.setPrice = function (price) { // 添加子类型私有的方法
 this.price = price;
};

var s = new Student("孙悟空", 24, 15000);
console.log(s.name, s.age, s.price);
s.setName("猪八戒");
s.setPrice(16000);
console.log(s.name, s.age, s.price);
```

4，垃圾回收

```
new person();
person=null//制空
```

return

2.6、作用域

1. 全局作用域
2. 函数作用域

2.7、对象遍历

```
let obj={
    name:"maxin",
    age:21,
    sayHello:function(){
        alert(1)
    }
}
for (let k in obj){
    console.log(k)  //变量属性名
    console.log(obj[k]) //变量值
 // obj[k]();
}
```

三、内置对象

1、Math 对象

- Math.PI
- Math.abs()
- ……
- 随机数

  [0,1)

  ```
  Math.random()
  ```

2、Date 对象