第一部分：JavaScript基础知识

查缺补漏

第一章 什么是JavaScript

1.1、完整的 JavaScript 实现的部分组成：

- 核心（ECMAScript）；
- 文档对象模型（DOM，Document Object Model）：提供访问和操作网页内容的方法和接口，用于在HTML(HyperText Markup Language, 超文本标记语言)中使用扩展的XML(可扩展标记语言)；
- 浏览器对象模型（BOM，Browser Object Model）：提供与浏览器交互的方法和接口；

（注：（1）JavaScript编程语言， Html和Css是标记语言。ECMAScript(ES)是JavaScript的标准，JavaScript是ECMAScript的实现；（2）DOM 4新增的内容包括替代Mutation Events的Mutation Observers，鉴于前者浏览器兼容问题和同步执行监听DOM树结构变化的性能问题。故提出Mutation Observers，当使用observer监听多个DOM变化时，并且这若干个DOM发生了变化，那么observer会将变化记录到变化数组中，等待一起都结束了，然后一次性的从变化数组中执行其对应的回调函数）

1.2、for of 和for in和 forEach（）

```
const digits = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];
for (const digit of digits) { console.log(digit);}
for (const index in digits) {console.log(digits[index]);}
```

- for...of 循环将只循环访问对象中的值，不用担心向对象中添加新的属性。for of可以与 break、continue和return 配合使用；只要有 iterator 接口的数据结构，都可以使用 for of循环。如：数组Array、Map、Set、String、arguments对象、Nodelist对象， 就是获取的dom列表集合。
- for...in 循环访问所有可枚举的属性（包括原型属性），意味着如果向数组的原型中添加任何其他属性，这些属性也会出现在循环中。可以用hasOwnProperty（） 避免遍历到原型属性上；
- forEach() 方法用于调用数组的每个元素，并将元素传递给回调函数。forEach() 本身是不支持的 continue 与 break 语句的，但可以通过 some 和 every 来实现。

```
var arr = [1, 2, 3, 4, 5];
//实现continue（不显示3）
arr.forEach(function (item) {
    if (item === 3) {
        return;
```

```
    }
    console.log(item);
});
//实现continue（不显示2）
arr.some(function (item) {
        if (item === 2) {
                return;  // 不能为 return false
        }
        console.log(item);
});
//实现break（只显示1 2 3）
arr.every(function (item) {
        console.log(item);
        return item !== 3;
});
```

```
Object.prototype.objCustom = function() {};
Array.prototype.arrCustom = function() {};
let iterable = [3, 5, 7];
iterable.foo = 'hello';

for (let i in iterable) {
  console.log(i); // logs 0, 1, 2, "foo", "arrCustom", "objCustom"
}

for (let i in iterable) {
  if (iterable.hasOwnProperty(i)) {
    console.log(i); // logs 0, 1, 2, "foo"
  }
}

for (let i of iterable) {
  console.log(i); // logs 3, 5, 7
}
```

第二章 HTML中的JavaScript

2.1、`<script>`元素

```
<script type="text/javascript">  //type属性
 //内容不可以直接出现</script>，需要转义字符转换：
 alert("<\/script>"); //转义字符“\”
</script>
```

2.2、带有 src 属性的`<script>`元素不应该在其`<script>`和`</script>`标签之间再包含额外的 JavaScript 代码。如果包含了嵌入的代码，则只会下载并执行外部脚本文件，嵌入的代码会被忽略。

2.3、一般都把全部 JavaScript 引用放在`<body>`元素中页面内容的后面，在解析包含的 JavaScript 代码之前，页面的内容将完全呈现在浏览器中。浏览器会按照`<script>`在页面出现的顺序解释，前提未使用**async="async"** 和 **defer="defer"**

- async属性表示立即开始下载脚本，不需要等待其他脚本，同时也不阻塞文档渲染，即异步加载，异步脚本不能保证按照他们在页面中出现的次序执行；
- defer表示在文档解析和显示完成后在**执行**脚本是没有问题的；
- 二者均只对外部脚本文件有效

第三章 语言基础

3.1、标识符：可以自主命名的都可以称之为 标识符。例如：变量名、函数名、属性名等。（*标识符中可以含有字母，数字，下划线_，$*；不能以数字开头；**关键字、保留字、true、false、null不能作为标识符**）

3.2、严格模式：ES5引入（"use strict"; 用在脚本开头或者函数内部开头）

3.3、var 介绍：

（1）var message; //未初始化的变量会保存一个特殊的值**undefined**

（2）var：该变量可以用来保存任何值。var message = "hi"; **message = 100;** //有效但是不推荐；

（3）如果变量在函数中定义时没有用var，则为全局变量，在函数外部也可以使用；

var、let、const区别：

- 使用let声明的变量可以重新赋值,但是不能在同一作用域内重新声明；
- **var声明提升，let声明的变量不会在作用域中提升**；
- 使用const声明的变量必须赋值初始化,但是不能在同一作用域类重新声明也无法重新赋值；
- 使用let在全局作用域中声明的变量不会成为window对象的属性

```
var name="mike";console.log(window.name)//'mike'
let age=26;console.log(window.age)//undefined
```

3.4 typeof和instanceof

```
typeof undefined              // undefined
typeof null                   // object
null === undefined            // false
null == undefined             // true
//null可以被理解为是一个特殊的值，undefined可以被认为是声明了一个变量未赋初值
```

3.5 数据类型

ES 能够表示的最小数值保存在 **Number.MIN_VALUE** 中——在大多数浏览器中，这个值是 5e-324；能够表示的最大数值保存在**Number.MAX_VALUE** 中；

3.6 NaN 即非数值（Not a Number，表示本来要返回数值的结果失败了）。有两个特点：（1）任何涉及 NaN 的操作（例如 NaN/10）都会返回 NaN（2）NaN 与任何值都不相等，包括 NaN 本身。

```
alert(isNaN(NaN)); //true
alert(isNaN(10)); //false（10 是一个数值）
alert(isNaN("10")); //false（可以被转换成数值 10）
alert(isNaN("blue")); //true（不能转换成数值）
alert(isNaN(true)); //false（可以被转换成数值 1）
```

3.7 模板自变量

（1）其最常用的特性就是支持字符串插值；在插值内也可以调用函数；

```
var a=2;var b=3;let res=`${a}+${b}=${a+b}`

function capitalize(word){return `${word[0]}.toUpperCase()}${word.slice(1)}`};
console.log(`${capitalize('hello')},${capitalize('world')}!`)
```

(2)用`String.raw`获取原始模板字面量内容

```
String.raw`first line\nsecond line`;  //first line\nsecond line
```

3.8 Symbol类型：符号实例唯一不可变；Symbol（）函数不能用作构造函数，与new一起用；

3.9 操作符

- 位操作符：按位非（~）、按位与（&）、按位或（|）、按位异或（^）、左移(<<)、有符号右移(>>)、无符号右移(>>>)；
- 布尔操作符：逻辑非（！）、逻辑与（&&）、逻辑或（||）；(数值0、null、NaN、undefined经过逻辑非操作后为true)
- 乘性操作符（乘法、除法、取模或取余）、指数操作符（**）
- 相等操作符：

```
null == undefined //true
true == 1;false == 0 //true
"NaN" == NaN ;5 == NaN;NaN == NaN //false 有NaN就是false
undefined == 0 //false
null == 0 //false
"5"==5    //true
```

3.10 语句

标签语句：常用场景是嵌套循环；可以在后面通过break和continue语句引用；

```
outermost: for (let i=0; i<10; i++) {
  for (let j=0; j<10; j++) {
    if (i == 5 && j == 5) {
        console.log(i);
       break outermost;//continue
     }
  }
}
```

3.11 ECMAScript中的函数与其他语言中的函数不同之处：

- 不需要指定函数的返回值，因为任何函数可以在任何时候返回任何值。
- 不指定返回值的函数实际上会返回特殊值undefined。

第四章 变量、作用域和内存

4.1基本数据类型（栈stack中）：Undefined、Null、Boolean、Number 和 String、Symbol；（按值传递）

引用数据类型（堆中）：Array、Object、function（按引用传递）；引用类型的存储需要内存的栈区和堆区共同完成，栈区内存保存变量标识符和指向，堆内存中该对象的指针；

```
var name1 = "Nicholas"; var name2 = new String('Mike');
name1.age = 10; name2.age = 20;
alert(name1.age);  //undefined
alert(name2.age);//20
alert(typeof name1)//string
alert(typeof name2)//object
```

4.2 `instanceof` 运算符用于检测构造函数的 `prototype` 属性是否出现在某个实例对象的原型链上。（确定值的引用类型）

4.3 JavaScript垃圾回收策略有**标记清理**（常用）和**引用计数**（不常用）；

内存泄漏的原因：意外声明全局变量、定时器、使用js的闭包

4.4 块级上下文（作用域）：(由最近的的一对包含花括号{}界定)

- 执行上下文分为全局上下文、函数上下文、块级上下文；
- 代码执行流每进人一个新上下文，都会创建一个作用域链， 用于搜索变量和函数。
- 函数或块的局部上下文不仅可以访问自己作用域内的变量，而且也可以访问任何包含上下文乃至全局上下文中的变量。
- 全局上下文只能访问全局上下文中的变量和函数，不能直接访问局部上下文中的任何数据。
- 变量的执行上下文用于确定任么时候释放内存。

第五章 基本引用类型

5.1 正则表达式

```
let expression=/pattern/flags
```

RegExp实例的主要方法是exec()、text();

exec()方法用于检索字符串中的正则表达式的匹配，若找到匹配项，则返回第一个匹配信息的数组，如果没有找到，则返回null；返回的数组虽然是Array的实例，但是包含两个额外的属性：index（字符串中匹配模式的起始位置）和input（要查找的数字）；

5.2 charAt()和charCodeAt()

```
let mes="abcdefg";
console.log(mes.charAt(2));//c
console.log(mes.charCodeAt(2));//99
```

5.3 单例内置对象：Global、Math

5.4 舍入方法

- Math.ceil()执行向上舍入，即它总是将数值向上舍入为最接近的整数；
- Math.floor()执行向下舍入，即它总是将数值向下舍入为最接近的整数；
- Math.round()执行标准舍入，即它总是将数值四舍五入为最接近的整数；
- Math.fround()返回数值最接近的单精度（32位）浮点值表示；

5.5 字符串操作的几种方法: concat()、slice()、substr()、substring()、、indexOf()、lastIndexOf、、startsWith()、endsWith()、includes()、、trim()、repeat()、padStart()、padEnd()、、toLowerCase()、toLocaleLowerCase()、、exec()、match()、search()、、localeCompare()