1801210682 吴欣雨

# JavaScript语法

## 目录

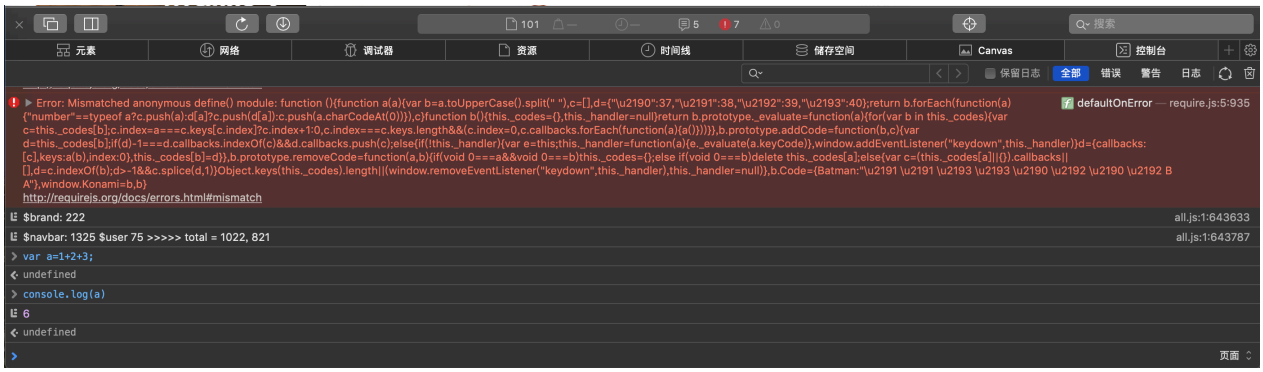
1801210682 吴欣雨

# 一. 嵌入HTML

1. JavaScript代码的嵌入可以有两种形式

- 直接在<head></head>标签中使用<script></script>标签嵌入JavaScript代码
- 将JavaScript代码放到单独的.js文件中，通过在<head></head>中引入文件来执行
  - 引入方式为<script src="/static/js/abc.js"></script>
  - 可以引用多个.js文件，浏览器将顺序执行
  - 在命令行中使用JavaScript abc.js可以直接执行.js文件

2. 可以在浏览器的JavaScript控制台中直接输入JavaScript代码并执行




# 二. 数据类型

1. JavaScript中的Number类型包含整数、浮点数、科学计数法、NaN以及Infinity

- Number类型在使用toString时要先加括号: (123).toString();

2. ==运算符会自动转换数据类型再做比较，===不会转换数据类型，如果数据类型不一致就会返回false（应该始终使用===做比较）

3. NaN与其他所有的值都不相等，因此不可以使用a==NaN，判断NaN只能使用isNaN()

4. null是空值，它与0和空字符串都不相等

5. JavaScript中的数组可以包含任意数据类型

- 可以直接定义： var arr = [1, 2, 3.14, 'Hello', null, true];
- 也可以使用Array函数： **new Array**(1, 2, 3);
- arr.length会返回数组的长度
- push()在数组末尾添加元素，pop （）删除数组末尾的元素

- unshift()在数组头部添加元素，shift（）删除数组头部的元素
- 空数组继续删除不会报错，而是返回undefined
- sort()对数组元素进行排序
  - 可以对sort定义回调函数，自定义排序规则
  - return 1意思为调换位置，return -1意思为不做交换
- splice可以对数组进行操作分别输入起始位置、删除元素个数以及添加的元素内容

6. JavaScript中的对象是无序集合，获取对象属性的方式为对象变量.属性名：

persion.hasCar

```
var person = {
    name: 'Bob',
    age: 20,
    tags: ['js', 'web', 'mobile'],
    city: 'Beijing',
    hasCar: true,
    zipcode: null
};
```

- 最后一个键值对不需要添加逗号
- 如果属性名包含特殊字符，需要用单引号括起来，并且在访问时要使用对象名称['属性名称']
- 使用delete函数删除对象中的属性，如果删除对象不存在则会返回undefined(不会报错)
- 检测对象是否具有某个属性应该使用in操作符（继承得到的属性也会返回true）
- hasOwnProperty返回的是对象自身拥有的属性（不包括继承的属性）

7. 如果变量没有用var声明，则会自动声明为全局变量，不同JavaScript文件中的同名变量可能会相互影响

- 需要在第一行使用“use strict”启动strict模式，在该模式下，变量必须经过var声明才可以使用，否则会报错
- 所有JavaScript代码都应该使用strict模式

8. 模版字符串用于输出变量，会自动将输出字符串中的${name}替换为name变量的值，但是必须使用反引号（``）扩起来

9. 字符串本身不可变，使用字符串操作的方法（toUpperCase、toLowerCase、indexOf、substring）会返回一个新的字符串而不是改变原有字符串

10. Map中的键可以为任意数据类型，定义方式为:varm=new Map();

- 插入新的键值对使用m.set();
- 查询key使用m.has();
- 查询value使用m.get();
- 删除键值对使用m.delete();

11. set中的元素互不相同

- 添加元素使用s.add()，添加已存在元素没有效果

## 三. 循环语句

1. for…in语句可以循环一个变量的所有属性

2. for…of语句可以循环数组，Set和Map的元素（它们都具有iterable类型），如果该对象有属性不会显示

3. forEach方法可以有同样的效果，function为回调函数，每次迭代都会自动调用

```
'use strict';
var a = ['A', 'B', 'C'];
```

```
a.forEach(function (element, index, array) {
    // element: 指向当前元素的值
    // index: 指向当前索引
    // array: 指向Array对象本身
    console.log(element + ', index = ' + index);
});
```

▶ Run

```
A
B
C
```

- 数组的回调参数为element，index和array
- Map的回调参数为value，key，map
- Set的回调参数为element，sameElement，set

## 四. 函数

4. JavaScript中的函数定义有两种形式：

- 直接定义，这种情况下a为函数名称，如果没有return语句，函数执行完毕将返回undefined，此时a()是一个函数对象，函数名a是指向该函数的变量：

function a(x){

}

- 以匿名函数的方式定义，末尾需要添加分号;其中a是变量，这种方式定义的变量为全局变量：

var a=function(x){

};

- arguments指向调用传入的所有参数，如果传入的参数多于定义的参数也会显示，因此可以用来判断传入参数的个数
- rest参数可以用来保存传入的定义参数之外的参数，具体写法为：

```
function foo(a, b, ...rest) {
    console.log('a = ' + a);
    console.log('b = ' + b);
    console.log(rest);
}

foo(1, 2, 3, 4, 5);
// 结果:
// a = 1
// b = 2
// Array [ 3, 4, 5 ]

foo(1);
// 结果:
// a = 1
// b = undefined
// Array []
```

5. 函数中声明的所有变量会默认出现在函数体顶部，对于内部变量和外部变量重名的情况，会使用内部变量

6. 所有的全局变量都默认是全局对象window的一个属性，为了避免不同.js文件中全局变量的冲突，可以将文件中的所有变量和函数都绑定到一个全局对象中，该全局变量也被称为命名空间

```
// 唯一的全局变量MYAPP:
var MYAPP = {};

// 其他变量:
MYAPP.name = 'myapp';
MYAPP.version = 1.0;

// 其他函数:
MYAPP.foo = function () {
    return 'foo';
};
```

7. let可以用来声明作用于函数内局部的变量

8. 解构赋值可以用来给多个变量赋值，或者从对象中取出多个值

- 给多个变量赋值

```
let [x, [y, z]] = ['hello', ['JavaScript', 'ES6']];
```

- 从对象中取出多个值

```
var person = {
    name: '小明',
    age: 20,
    gender: 'male',
    passport: 'G-12345678',
    school: 'No.4 middle school'
};
var {name, age, passport} = person;
```

- var {hostname:domain, pathname:path} = location;可以用来获取当前页面的域名和路径

- 解构赋值可以设置默认值，如果向函数传入一个对象，在定义参数时可以通过解构赋值直接将对象的属性存入变量中：

```
function buildDate({year, month, day, hour=0, minute=0, second=0}) {
    return new Date(year + '-' + month + '-' + day + ' ' + hour + ':' + minute + ':' +
second);
}
```

9. function*可以用来定义generator，它与函数的区别在于generator可以返回多次，并在每一次返回后暂停，在执行.next时继续，直到遇到下一个yield或者return

- .next()方法会有返回值，当返回值为true时表示已经执行完毕
- 可以使用for…of循环迭代generator对象，无需额外判断.next()返回值

## 五. 方法

1. 方法是指写在对象中的函数，其中的this变量始终指向当前对象，想要让this指向对象必须直接以obj.method（）的方式调用

2. 使用var that=this并在方法内部定义的函数中使用that可以避免this指向全局变量的情况

3. 箭头函数x=》x*x是一种简化的匿名函数，并且优化了this使其始终指向外层调用者

4. map（函数名）可以实现对一个数组的每个元素分别实现该函数

- 将一个数组中的所有数字转换为字符串，可以通过arr.map(String)来实现
- 将一个数组中的英文名称改为首字母大写，其他小写的规范名字：

```
function normalize(arr){
        return arr.map(function(s){
            return s[0].toUpperCase()+s.substring(1,s,length).toLowerCase();
        });
}
```

5. reduce（函数名）可以实现对一个数组的每个元素使用该函数进行累计计算

- 计算一个数组的乘积：

```
function product(arr){
        return arr.reduce(function(x,y){
                return x*y;
        });
}
```

• 将字符串转换为数字(s.split('')可以将一个字符串的每个字符分离并返回一个数组):

```
function string2int(s){
        return s.split('').map(function(x){
                return x-'0';
        }).reduce(function(x,y){
                return x*10+y;
        });
}
```

6. filter（函数名）可以实现对一个数组中符合要求的元素进行过滤

• 利用filter接收的回调函数中的参数，实现数组去重（indexOf始终返回的是元素第一次出现的位置）：

```
function distinct（arr）{
        return arr.filter(function(element,index,self){
                return self.indexOf(element)==index;
        });
}
```

7. apply和call都可以实现调用一个对象的一个方法，并用另一个对象替换当前对象

• apply要求将所有参数都以数组的形式传递

• call可以接受多个参数，包括新this对象以及参数列表，这个方法主要用在js对象各方法相互调用的时候，使当前this实例指针保持一致，或者在特殊情况下需要改变this指针。如果没有提供thisObj参数，那么 Global 对象被用作thisObj。

## 六. 闭包的概念

1. 闭包是一个函数和函数所声明的词法环境的结合，特殊情况是在函数嵌套调用时，内部函数访问了外部函数的变量
2. 把一个函数作为返回值返回，可以访问父级作用域
3. 因为全局变量指向了返回的函数，使顶层函数的变量没有随着函数的执行完成而被释放
4. 每次调用同一个闭包都会返回一个新的函数（之前的函数未被释放），即使传入相同的参数也是两个不同的闭包
5. 如果想要引用循环变量，需要使用创建一个匿名函数并立即执行的方法：

```
(function (x) { return x * x }) (3);
```

6. 使用闭包可以把多参数函数变成单参数函数，可以定义一个参数固定的多个函数，在使用时再指定其他参数：

```
function make_power（n）{
        return function(x){
            return Math.pow(x,n);
        }
}
```

## 七. 正则表达式

1. 正则表达式的创建可以有两种方式，一种是通过/正则表达式/写出来，一种是通过new RegExp创建正则表达式对象，如果通过对象创建正则表达式需要注意字符转义问题

```
var re1 = /ABC\-001/;
var re2 = new RegExp('ABC\\-001');
```

2. .test()用于测试输入表达式是否与设置的正则表达式匹配，返回值为true或false
3. .split()中可以加入正则表达式更好地进行切分
4. .exec可以用来提取匹配成功的字符串中的字串，需要在正则表达式中用（）标记需要提取的子串

```
var re = /^(\d{3})-(\d{3,8})$/;
re.exec('010-12345'); // ['010-12345', '010', '12345']
re.exec('010 12345'); // null
```

```
var re = /^(0[0-9]|1[0-9]|2[0-3]|[0-9])\:(0[0-9]|1[0-9]|2[0-9]|3[0-9]|4[0-9]|5[0-9]|[0-9])\:(0[0-9]|1[0-9]|2[0-9]|3[0-9]|4[0-9]|5[0-9]|[0-9])$/;
re.exec('19:05:30'); // ['19:05:30', '19', '05', '30']
```

- 字符操作的优先级高于|操作，因此0[0-9]|1[0-9]判断的是0[0-9]或1[0-9]而不是[0-9]|1

5. /正则表达式/g和new RegExp('正则表达式','g')都支持多次执行exec方法，每次返回当前的index值
6. i表示忽略大小写，m表示多行匹配

## 八. 浏览器对象

1. location.reload()可以重新加载当前页
2. location.assign()可以加载新页面
3. document对象表示当前页面，document.title就是html中的<title>xxx</title>，可以动态改变
4. 使用document对象提供的getElementById()和getElementsByTagName()可以获取到html中某个id或tag的节点，使用.innerHTML可以获得DOM节点中的内容
5. 获取某个节点下的所有子节点:test.children，第一个子节点：test.firstElementChild，最后一个子节点：test.lastElementChild
6. 使用querySelector和querySelectorAll可以根据条件选择节点，querySelector只返回第一个满足条件的节点，如果存在多个相同的id、class，要使用querySelectorAll并加上数组下标来找到所需节点
7. .innerHTML可以用来修改节点的html内容，需要防止XSS攻击

8. 可以在获取节点后使用.style更改CSS样式

```
var p = document.getElementById('p-id');
// 设置CSS:
p.style.color = '#ff0000';
p.style.fontSize = '20px';
p.style.paddingTop = '2em';
```

9. 使用appendChild可以插入DOM节点，待插入的节点会首先从原先的位置被删除，在插入到当前的位置；如果父节点为空，也可以使用innerHTML替换掉原先的节点

10. 可以使用createElement创建新节点，再将创建好的新节点插入DOM树中

11. 使用parentElement.insertBefore(newElement, referenceElement);将新创建的节点插入到referenceElement之前

12. Array.from可以将一个可遍历的类数组转换为真正的数组（可以将一个节点的列表转换为数组便于进行排序或其他处理）

```
// sort list:
var arr=Array.from(document.querySelector('#test-list').children);
arr.sort(function(s1,s2){
return s1.innerText-s2.innerText;});
arr.forEach(i=>document.querySelector('#test-list').appendChild(i));
```

13. 使用deleteChild可以删除子节点，子节点删除后编号可能会发生变化

## 九. 数据传递

1. 对于HTML中的input控件，可以使用.value获取输入值，对于单选框(type=‘radio’)或复选框(type=‘checkbox’)，要使用.checked查看是否被选中

```
// <input type="text" id="email">
var input = document.getElementById('email');
input.value; // '用户输入的值'
```

- 如果type=“radio”，在判断是否被选中的时候要使用

document.getELementById.checked来判断

2. JavaScript在处理表单提交时有两种方法：

- 一种是使用<form id=‘’>，在<button>的onclick中添加javascript，然后在代码中使用document.getElementById获取表单中的值，再使用.submit方法提交到后端
- 另一种是在<form>的onsubmit中添加JavaScript，然后在代码中使用document.getElementById获取表单中的值，最后在函数中return true，返回至html中，使用正常的<button type=“submit”>提交表单至后端

3. 在html中可以使用<input type=“file”>上传文件

- 当表单包含<input type=“file”>时，表单必须指定enctype=multipart/form-data，method=post