JS基础

2020年9月17日 8:31

/*变量

js变量的命名规范

变量只能是数字 字母 下划线 $

数字不开头

变量名命名规范

推荐使用驼峰式

userName

dataOfDb

python推荐使用下划线方式

user_name

不能使用关键字做变量名

js代码的书写位置

可以单独开设js文件

在浏览器中的console界面书写

注意只有关闭标签或者书信页面才能实际清楚所写代码

数据类型

数值类型(number)

vara=11

vara=11.11

查看当前的数据类型(typef)

typefa;

NaN是数值类型，表示的意思是'不是一个数字'

类型转换

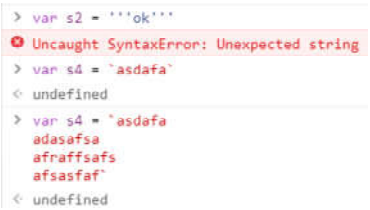
parseInt

parseFloat

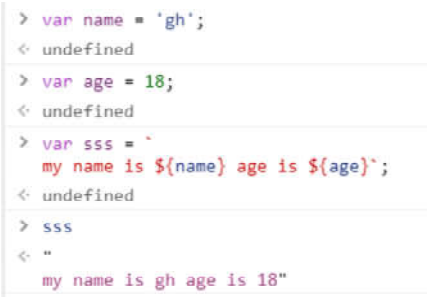
parse

字符串(string)

```
> var s = 'jason'
<- undefined
> var s1 = "jason"
<- undefined
> typeof s
<- "string"
> typeof s1
<- "string"
```



字符拼接



pyhton中不推荐使用 +，推荐使用 join

js中推荐使用+

字符串的常用方法

方法	说明	
.length	返回长度	len()
.trim()	移除空白	strip()
.trimLeft()	移除左边的空白	lstrip()
.trimRight()	移除右边的空白	rstrip()
.charAt(n)	返回第n个字符	
.concat(value, ...)	拼接	join()
.indexOf(substring, start)	子序列位置	
.substring(from, to)	根据索引获取子序列	[]索引取值
.slice(start, end)	切片	
.toLowerCase()	小写	lower()
.toUpperCase()	大写	unper()
.split(delimiter, limit)	分割	split()

```
> var name = 'egondsb'
< undefined
> name.length
< 7
> var name1 = '  egonDSB  '
< undefined
> name1
<   egonDSB  
> name1.trim()
< "egonDSB"
> name1.trimLeft()
< "egonDSB  "
> name1.trimRight()
<   egonDSB"
> var name2 = '$$jason$$'
< undefined
> name2.trim('$')
< "$$jason$$"
> name2.charAt(0)
< "$"
> name2.indexOf('as')
< 3
> name2.substring(0,5)
< "$$jas"
> name2.slice(0,5)
< "$$jas"
> name2.substring(0,-1)
< ""
> name2.slice(0,-1)
< "$$jason$"
> var name3 = 'eGoNDsb123666HahA'
< undefined
> name3.toLocaleLowerCase()
< "egondsb123666haha"
> name3.toUpperCase()
< "EGONDSB123666HAHA"
> var name = 'tank|hecha|liaomei|mengsao|...'
< undefined
> name.split('|')
< ▸ (5) ["tank", "hecha", "liaomei", "mengsao", "..."]
> name.split('|',2)
< ▸ (2) ["tank", "hecha"]
> name.split('|',10)
< ▸ (5) ["tank", "hecha", "liaomei", "mengsao", "..."]
```

```
name.split('|')
(5) ["tank", "hecha", "liaomei", "mengsao", "..."]
name.split('|',2)
(2) ["tank", "hecha"]0: "tank"1: "hecha"length: 2__proto__: Array(0)
name.split('|',10)  # 第二个参数不是限制切割字符的个数还是获取切割之后元素的个数
(5) ["tank", "hecha", "liaomei", "mengsao", "..."]
```

布尔值（boolean）

true，false

false：0　null　undefined　NaN　空字符串

对象

数组 [ ]

```
var l = [11,22,33,44,55]

typeof l
"object"
var ll = [11,'sdasd',11.11,true]

ll[1]
"sdasd"
ll[-1]  # 不支持负数索引
```

方法	说明
.length	数组的大小
.push(ele)	尾部追加元素
.pop()	获取尾部的元素
.unshift(ele)	头部插入元素
.shift()	头部移除元素
.slice(start, end)	切片
.reverse()	反转
.join(seq)	将数组元素连接成字符串
.concat(val, ...)	连接数组
.sort()	排序
.forEach()	将数组的每个元素传递给回调函数
.splice()	删除元素，并向数组添加新元素。
.map()	返回一个数组元素调用函数处理后的值的新数组

python中的列表添加元素的方式有哪些?
1.append 尾部追加
2.extend扩展列表(内部for循环+append)
3.insert()指定位置插入元素

python中的列表删除元素的方式有哪些?
1.pop(index)
2.remove(target)
3.del

python中常用的内置方法
map、zip、filter、reduce

```
var l = [111,222,333,444,555,666]
undefined
l.length
6
l.push(777)
7
l
(7) [111, 222, 333, 444, 555, 666, 777]
l.pop()
777
l
(6) [111, 222, 333, 444, 555, 666]
l.unshift(123)
7
l
(7) [123, 111, 222, 333, 444, 555, 666]
l.shift()
123
l.slice(0,3)
(3) [111, 222, 333]
l.reverse()
(6) [666, 555, 444, 333, 222, 111]
l.join('$')  # 跟python刚好相反
"666$555$444$333$222$111"

l.concat([111,222,333])  # extend
(9) [666, 555, 444, 333, 222, 111, 111, 222, 333]
l.sort()
(6) [111, 222, 333, 444, 555, 666]
```

.

```
> var ll = [111,222,333,444,555,666]
< undefined
> ll.forEach(function(value){console.log(value)},ll)
  111
  222
  333
  444
  555
  666
< undefined
> ll.forEach(function(value,index){console.log(value,index)},ll)
  111 0
  222 1
  333 2
  444 3
  555 4
  666 5
< undefined
> ll.forEach(function(value,index,arr){console.log(value,index,arr)},ll)
  111 0 ▸ (6) [111, 222, 333, 444, 555, 666]
  222 1 ▸ (6) [111, 222, 333, 444, 555, 666]
  333 2 ▸ (6) [111, 222, 333, 444, 555, 666]
  444 3 ▸ (6) [111, 222, 333, 444, 555, 666]
  555 4 ▸ (6) [111, 222, 333, 444, 555, 666]
  666 5 ▸ (6) [111, 222, 333, 444, 555, 666]
< undefined
```

```
ll
(6) [111, 222, 333, 444, 555, 666]
ll.splice(0,3)  # 两个参数 第一个是起始位置 第二个是删除的个数
(3) [111, 222, 333]
ll
(3) [444, 555, 666]
ll.splice(0,1,777)  # 先删除后添加
[444]
ll
(3) [777, 555, 666]
ll.splice(0,1,[111,222,333,444])
[777]
ll
(3) [Array(4), 555, 666]
```

```
var ll = [11,22,33,44,55,66]
undefined
ll.map(function(value){console.log(value)},ll)
VM3115:1 11
VM3115:1 22
VM3115:1 33
VM3115:1 44
VM3115:1 55
VM3115:1 66

ll.map(function(value,index){return value*2},ll)
(6) [22, 44, 66, 88, 110, 132]
ll.map(function(value,index,arr){return value*2},ll)
(6) [22, 44, 66, 88, 110, 132]
```

自定义对象 { }

运算符

```
# 算术运算符
var x = 10;
var res1 = x++;
var res2 = ++x;
res1 10
res2 12
++表示自增1 类似于 +=1
加号在前先加后赋值 加号在后先赋值后加

# 比较运算符
1 == '1'  # 弱等于  内部自动转换成相同的数据类型比较了
true

1 === '1'  # 强等于  内部不做类型转换

1 != '1'
false
1 !== '2'
true
```

```
# 逻辑运算符
  # python中 and or not
  # js中 && || !
5 && '5'
'5'

0 || 1
1

!5 && '5'
5
"""
一定要注意到底什么时候返回的是布尔值 什么是返回的是数据
"""
```

```
# 赋值运算符
= += -= *= ....
```

流程控制

```
# if判断
var age = 28;
# if(条件){条件成立之后指向的代码块}
if (age>18){
  console.log('来啊 来啊')
}
# if-else
if (age>18){
  console.log('来啊 来啊')
}else{
  console.log('没钱 滚蛋')
}
# if-else if else
if (age<18){
  console.log("培养一下")
}else if(age<24){
  console.log('小姐姐你好 我是你的粉丝')
}else{
  console.log('你是个好人')
}
```

```
# switch语法
"""
提前列举好可能出现的条件和解决方式
"""
var num = 2;
switch(num){
  case 0:
    console.log('喝酒');
    break;
  case 1:
    console.log('唱歌');
    break;
  case 2:
    console.log('洗脚');
    break;
  case 3:
    console.log('按摩');
    break;
  case 4:
    console.log('营养快线');
    break;
  case 5:
    console.log('老板慢走 欢迎下次光临');
    break;
}
```

类似C语言

函数

```
# 在python定义函数需要用到关键字def
# 在js中定义函数需要用到关键字function

# 格式
function 函数名(形参1,形参2,形参3...){函数体代码}

# 无参函数
function func1(){
  console.log('hello world')
}
func1()  # 调用 加括调用 跟python是一样的

# 有参函数
function func2(a,b){
  console.log(a,b)
}
func2(1,2)

func2(1,2,3,4,5,6,7,8,9)  # 多了没关系 只要对应的数据
VM3610:2 1 2
undefined

func2(1)  # 少了也没关系
VM3610:2 1 undefined
```

```
# 关键字arguments
function func2(a,b){
  console.log(arguments)  # 能够获取到函数接受到的所有的参数
  console.log(a,b)
}
```

不能解压赋值

```
function func2(a,b){
  if(arguments.length<2){
    console.log('传少了')
  }else if (arguments.length>2){
    console.log('传多了')
  }else{
    console.log('正常执行')
  }
}
```

```
# 函数的返回值  使用的也是关键字return
function index(){
  return 666
}
function index(){
  return 666,777,888,999
}
res = index();
999
res
999  # 只能拿到最后一个
```

```
# 匿名函数  就是没有名字
function(){
  console.log('哈哈哈')
}
var res = function(){
  console.log('哈哈哈')
}

# 箭头函数(要了解一下)
var func1 = v => v;
等价于
var func1 = function(v){
  return v
}
```

```
var func2 = (arg1,arg2) => arg1+arg2
等价于
var func1 = function(arg1,arg2){
  return arg1+arg2
}
```

```
for(let i in d){
  console.log(i,d[i])
}  # 支持for循环 暴露给外界可以直接获取的也是键


"""第二种创建自定义对象的方式  需要使用关键字 new"""
var d2 = new Object()  # {}

d2.name = 'jason'
{name: "jason"}

d2['age'] = 18
{name: "jason", age: 18}
```

Date对象

```
let d3 = new Date()
Fri May 15 2020 14:41:06 GMT+0800 (中国标准时间)

d3.toLocaleString()
"2020/5/15 下午2:41:06"

# 也支持自己手动输入时间
let d4 = new Date('2200/11/11 11:11:11')
d4.toLocaleString()
let d5 = new Date(1111,11,11,11,11,11)
d5.toLocaleString()  # 月份从0开始0-11月
"1111/12/11 上午11:11:11"
```

```
# 时间对象具体方法
let d6 = new Date();
d6.getDate()  获取日
d6.getDay()   获取星期
d6.getMonth()   获取月份(0-11)
d6.getFullYear()    获取完整的年份
d6.getHours()     获取小时
d6.getMinutes()   获取分钟
d6.getSeconds()   获取秒
d6.getMilliseconds()  获取毫秒
d6.getTime()          时间戳
```

JSON对象

```
"""
在python中序列化反序列化
  dumps      序列化
  loads      反序列化

在js中也有序列化反序列化
  JSON.stringify()              dumps
  JSON.parse()                  loads
"""
let d7 = {'name':'jason','age':18}
let res666 = JSON.stringify(d7)
"{"name":"jason","age":18}"

JSON.parse(res666)
{name: "jason", age: 18}
```

RegExp对象

```
"""
在python中如果需要使用正则 需要借助于re模块
在js中需要你创建正则对象
"""
# 第一种 有点麻烦
let reg1 = new RegExp('^[a-zA-Z][a-zA-Z0-9]{5,11}')
# 第二种 个人推荐
let reg2 = /^[a-zA-Z][a-zA-Z0-9]{5,11}/

# 匹配内容
reg1.test('egondsb')
reg2.test('egondsb')

# 题目 获取字符串里面所有的字母s
let sss = 'egondsb dsb dsb'
sss.match(/s/)  # 拿到一个就停止了
sss.match(/s/g) # 全局匹配  g就表示全局模式
```

```
# 全局匹配模式吐槽点
let reg3 = /^[a-zA-Z][a-zA-Z0-9]{5,11}/g
reg2.test('egondsb')

reg3.test('egondsb')  # 全局模式有一个lastIndex属性
true
reg3.test('egondsb')
false
reg3.test('egondsb')
true
reg3.test('egondsb')
false

reg3.lastIndex
0
reg3.test('egondsb')
true
reg3.lastIndex
7
```

```
# 吐槽点二
let reg4 = /^[a-zA-Z][a-zA-Z0-9]{5,11}/
reg4.test()

reg4.test()  # 什么都不传 默认传的是undefined
true
reg4.test()
true

reg4.test(undefined)
true
let reg5 = /undefined/
undefined
reg5.test('jason')
false
reg5.test()
true
```

Math对象

abs(x)	返回数的绝对值。
exp(x)	返回 e 的指数。
floor(x)	对数进行下舍入。
log(x)	返回数的自然对数（底为e）。
max(x,y)	返回 x 和 y 中的最高值。
min(x,y)	返回 x 和 y 中的最低值。
pow(x,y)	返回 x 的 y 次幂。
random()	返回 0 - 1 之间的随机数。
round(x)	把数四舍五入为最接近的整数。
sin(x)	返回数的正弦。
sqrt(x)	返回数的平方根。
tan(x)	返回角的正切。