12.16

js

（1）基础知识

书写位置：位于HTML标签之外

行为特性：空格和换行可以区分每条js，但是代码封装会删除所有空格分号分开

（2）var let声明变量

作用域：

var 声明的变量是函数作用域或全局作用域。如果在函数内部使用 var 声明变量，该变量在整个函数内都是可访问的。如果在函数外部声明，则该变量是全局变量。

let 声明的变量是块作用域，即它只在其所在的代码块（如 if 语句、循环等）内有效。

变量提升：

使用 var 声明的变量会被提升到函数或全局作用域的顶部，但在实际赋值之前，该变量的值为 undefined

使用 let 声明的变量也会被提升，但在声明之前，访问该变量会导致 ReferenceError报错

重复声明：

var可以在同一作用域内多次声明同一个变量，后面的声明不会报错

let在同一作用域内不能重复声明同一个变量，重复声明会导致语法错误。

（3）const常量

声明常量

必须在声明时初始化，并且一旦赋值后，值不能被重新赋值。

块作用域

与 let 一样，const 声明的变量具有块作用域。

对于对象和数组

使用 const 声明的变量本身是不可变的，但如果它是一个对象或数组，仍然可以修改其属性或元素。

变量提升

与let相似

（4）弱变量类型语言

js变量类型不看定义，看赋值

（5）number：所有数字，Infinity -Infinity（无穷大、小） NaN

如果使用Number表示的变量超过了Js处理范围的数值，则会返回Infinity

数学方法：

Math.abs(-33); 绝对值

Math.random() [0,1) 包含0 不包含1

Math.ceil() 向上取整

Math.floor(4.9999) 向下取整

Math.round(a) 四舍五入

Math.PI π

随机一个整数 0-8 最大最小 代入算

let b = Math.floor(Math.random() \* 9)

随机3-13

console.log(Math.floor(Math.random() \* 11 + 3));

（6）3种弹出框

alert('警告框');//警告框

//用户输入框，有返回值，直接使用变量接收

// prompt('提示文本', '默认值');

let res = prompt('请输入手机号');

console.log(res);

//所有从用户界面获取过来的都为 字符串

console.log(typeof res);

//用户确认框,有返回值 true/flase

let resss = confirm('确定删除吗');

console.log(resss)

（7）强制转换

1.string转number

parseInt()

将一个string转为一个整数，不四舍五入，向下取整 ，直接截取整数部分

string字符串里面如果除了数字还有其他的部分，那么就截取前面数字部分

如果不能转，那么就返回NaN

parseFloat()

将字符串转为浮点数，尽可能的将一个字符串转为浮点数，浮点数之后如果有乱七八糟的内容，直接舍弃

2.number转string

String()

里面直接放number

数字与空字符串进行连字符运算

123+""

(8)运算符 (++ 并行 -- ) !非 数学 比较 逻辑 (&& ||) 赋值

数学运算符 值为number

包括 + - \* / %(取余)

隐式转换：所有带有字符串的运算都会尽可能的转为数字进行计算

true会转为1；false会转为0；null会转为 0

不纯的字符串（如"8天"）和undefined是不能帮我们进行隐式转换的，结果都是NaN

比较运算符

包括 > < >= <= == != === !== 值永远是boolean

string 和 string 也能够进行关系运算，比较的就是字符编码顺序

字符编码顺序，就是数字 30、大写字母41、小写字母 61

与数字进行关系运算时，纯数字字符串被转为数字，null转换为0，true转换转为1， false转换为0

null不能进行和0的相等判定

NaN不等于自己，不全等于自己

console.log(4 > 5 > 1);这种两个连续的比较运算符的先计算前面一个得到true/false，然后再把布尔值转为数字进行比较

逻辑运算符

&& 逻辑与运算 并且

|| 逻辑或运算 或者

! 逻辑非运算 反向

短路语法

1.负性值/正性值

负性值： false，null， 0， NaN， ""(空字符串)，undefined

正性的：除了上面的，全是正性的

2.&&

a&&b,a是负性的，那么直接扔出a；如果a是正性的，直接扔出b

3.||

a&&b,a是负性的，那么直接扔出b；如果a是正性的，直接扔出a

赋值运算符

包括 = += -= \*= /= %= ++1/1++(自增 后/前) --1/1--(自减 后/前)

赋值运算的参与者，一定是变量

自增/减前后区别

var a = 3;

console.log(a++ + a++); //3+4

console.log(a); //5

var a = 3;

console.log(++a + ++a); // 4+5

console.log(a); //5

（9）条件分支语句

if...else...

基础：

if(条件){

//如果条件为真,执行这个大括号里面的

}else{

//如果条件为假,执行这个大括号里面的

}

多分支：

if......else if ...... else if...... else

跳楼现象，用户会选择一个分支执行（跳楼），选择这个分支了，就暗含了上一个分支没有满足

嵌套

switch case语句

switch(待检测值){

case 值1 :

值1 与 待检测值相同 时做的事情

break;

case 值2 :

值2 与 待检测值相同 时做的事情

break;

case 值3 :

值3 与 待检测值相同 时做的事情

break;

default :

默认要做的

break;

}

底层是===的比较,必须完全相同

多个值一起检测，执行一条语句：

switch(m){

case 1:

case 3:

case 5:

alert('123');

break;

case 2:

case 4:

case 6:

alert('123');

break;

}

条件 (三元) 运算符

条件 ? 条件为真时的值 : 条件为假时的值

12.17

（1）循环遍历

for循环

循环顺序：

for(let i = 1;(1) i <= 99;(2) i++(4)){

console.log(i); (3)

};(5)

do while语句 后置验证语句(一般都是死循环)

do{

①语句

}while(②条件表达式)

程序一开始就会执行一次①，然后验证②是否为真，如果是真，继续执行①

while语句 前置验证语句

while(条件){

语句

};

break关键字

遇见了break语句，那么当前这个循环就会立即终止

只能跳出当前所在的最内层循环,也就是当前循环

想终止所有的循环，那么JS中允许你给循环语句加label：

outer : for(var i = 1 ; i <= 10 ; i++){

for(var j = 1 ; j <= 10 ; j++){

console.log(i,j);

if(j == 5){

break outer; //break label的语法

};

};

};

continue关键字

会立即终止执行后面的语句，然后进入下一次迭代

也只能终端当前最内层的for，外层for也可和break一样去取个名字，然后终止所有循环

（2）函数

1.定义与调用

关键字function 定义函数

(1)函数声明

function 函数名(){

};

(2)函数表达式

let 函数名 = function(){

};

函数名后面加一个 () 进行调用

同名函数，后面的会把前面的覆盖

函数声明可以无条件提升（代码解析时会先看函数），函数表达式不会提升

遇到同名标识符，函数优先

2.参数

function fun(a（形参）){

console.log(a （形参）);

};

fun(88（实参）);

参数可以有无数个，用逗号隔开，一一对应 （少传的参数为NaN，多传的会直接无视）

函数在调用的时候，传进去的实参是什么类型，那么形参接收的就是什么类型

3.返回值

通过return的语句来返回值

遇见了return，将立即返回结果，返回调用它的地方，而不会执行函数内的剩余的语句

4.相互调用和递归

函数可以相互调用，在一个函数体里调用另一个函数名，和在函数体外一样

递归：自己调用自己

5.arguments

arguments对象，类数组对象，有长度和下标

涵盖了所有实参，包括超出形参个数而不显示的

功能：模拟函数的重载，使得同一个函数，根据参数个数的不同，有不同的作用。

6.IIFE 立即执行函数

//降级为表达式

+function foo(a){

console.log('foo' + a);

}(123);

-function foo(a){

console.log('foo' + a);

}(444);

//括号形式

(function foo(a){

console.log('foo22' + a);

})(333);

7.作用域

一个变量如果定义在了一个function里面，那么这个变量就叫做局部变量，只作用于函数范围内

（3）数组

1.定义

//调用内置对象方式

let arr = new Array('aa', 'bb', 'cc');

//字面量方式

let arr = ['aa', 'bb', 'cc'];

使用下标来精确访问数组中的某一个项，下标从0开始

通过下标可以获取项的值，可覆盖设置，直接赋值

2.数据类型

可以是所有数据类型，但一般还是相同类型保存在一起

length 表示这个数组的项的个数

3.数组的遍历

for(let i = 0; i < arr.length; i++){

//就可以对arr[i]来进行操作、判断......

}

4.数组的常见方法

数组的头尾操作push()、pop()、unshift()、shift()

push()方法，在数组的末尾添加项目，可以添加1个，也可以添加多个（arr.push(1,2)用逗号分开）可以添加另一个数组 数字 undefined

pop()方法，删除数组的最后一项，只能删除最后一项，无法删除多项。能够返回被删除的元素

unshift 头插

shfti 头删

splice 增 删 替换

第一个实参数：删除 索引 从下标起(包含)，删除后面所有

arr.splice(1);

第二个参数为 要删除的 个数

arr.splice(1, 2);

从第三个参数起，后面都替换或添加的项值

arr.splice(1, 2, '哈', '长沙', true);(从下标1起删除2个，在原位置添加后面的项值)

arr.splice(1, 0, '哈', '长沙', true);(从下标1起删除0个，在原位置添加后面的项值)

负数

arr1.splice(-2)（第一个参数为负数，表示从数组最后开始的个数. 也就是说删除最后2项）

倒序reverse()

立即让数组倒置：改变原数组

排序sort()

arr2.sort(function(a, b){

return a - b;

return b - a;

});

12.18

（1）字符串

1.定义

调用内置对象

new String()

字面量

let str = ''

2.长度与下标

length 和数组一样

可以遍历，遍历的是字符串里面的每一个字符

3.字符串方法

charAt() 和 indexOf()

charAt() 传下标，得到下标位置的字符

indexOf() 传字符，得到字符在字符串里的下标

includes()

查找是否包含传的字符，返回布尔值

replace()

替换匹配到的字符，返回新的字符串

第二个参数可以为一个函数

let newStr = str.replace('d', function(){

return '哈哈';

});

match()

找到一个或多个匹配的字符，可以搭配正则表达式使用

slice(x,y)

克隆出字符串，从起始下标到结束下标，左闭右开，并返回提取字符

传入负数从后往前的个数

split()

把字符串转为数组

substr() substring()

substr(起始下标，个数) 克隆字符串，从起始下标（包含） 2提取字符串的个数。

substring(开始下标，结束下标) 克隆字符串，提取字符串中两个指定的下标之间的字符。

toLowerCase() 把字符串转换为小写。

toUpperCase() 把字符串转换为大写。

（2）对象

1.定义和添加属性值

调用内置对象

new object()

obj.name = '小明';

字面量

let obj = {

name:'小猫',

}

值的形式：{key:value}键值对

2.对象的方法

对象的属性中有一个值是函数，称为对象有什么方法

let obj ={

name:'',

hello:function(){

alert("hello");

},

}

使用： obj.hello()

3.in 运算符

检测对象内是否有此属性，返回布尔值

let obj = {

name : '小明',

age : 18

};

console.log('name' in obj);

4.删除对象属性

delete 对象名.属性名

5.对象遍历

for(let k in obj){

console.log(k); //打印对象中所有属性

console.log(obj[k]); //打印出每个属性的值

}

（3）基本类型和引用类型区别

1.保存位置：

基本数据类型：直接保存在栈内存中

引用数据类型：保存到堆内存中的

2.保存和引用方式

基本数据类型：值与值之间是独立存在

引用数据类型：相同的对象链接多个不同的引用，引用是变量保存了对象的内存地址，当一个通过一个变量修改属性时，另一个也会受到影响

（4）DOM

1.获取页面元素

getElementById() 根据元素的 ID 选择单个 DOM 元素

getElementsByTagName() 标签名选择 DOM 元素

querySelector() 使用 CSS 选择器选择第一个匹配的 DOM 元素

querySelectorAll() 使用 CSS 选择器选择所有匹配的 DOM 元素，返回一个 NodeList

操作页面对象元素标签的属性：使用点语法操作，有什么属性直接打点什么属性

2.操作css样式

通过style属性去操作元素的行间样式

功能：获取元素属性值和设置元素的属性值

.style只能得到或者设置行间样式，所有写在内部样式表和外部样式表，一律不能得到

设置：（直接赋值）

<div id="oP" style="background:'red',">

let oP = document.getElementById("pp");

oP.style.background = "red";

oP.style.fontSize = "100px";

所有都操作在行间样式的

12.19

（一）自定义属性和值

<div id="box" title="标题" goods="123">box</div>

里面的title，goods就是自定义的属性

getAttribute和setAttribute 获取和设置

获取 参数为标签的属性

console.log(oBox.getAttribute('title'));

设置: 参数为二个：属性 值

oBox.setAttribute('title', '长沙');

（二）排他模型

先把所有设置成一样的，再把需要的一个进行改变

例子： let aP = document.getElementsByTagName('p');

// 四个P, 遍历 //批量监听

for(let i = 0; i < aP.length; i++){

aP[i].onclick = function(){

console.log(i);

//在当前元素变红之前，先把所有变绿 遍历

for(let i = 0; i < aP.length; i++){

aP[i].style.background = 'yellowgreen';

};

//点击谁就把谁变红

aP[i].style.background = 'red';

};

}

（三）定时器 延时器

定时器 每隔一个时间，运行一次 ms

let timer（变量名） = setInterval(函数,重复间隔时间（单位：毫秒）)

延时器 运行一次

let t = setTimeout(函数, 延迟时间（单位：毫秒）);

变量名用于清除定时器/延时器

定时器：clearInterval(timer（变量名）);

延时器：clearTimeout(t);

（四）模板字符串

模板字符串 反引号 `` 模板${}

可以用于传入变量对内容进行改变