DAY08 MATH 和 DATE

2023年4月27日 18:38

Math对象(比较特殊)的属性和方法

数学对象也叫内置对象，这些对象在程序执行之前就已经存在了，所以不必显式地实例化对象

接下来的属性和方法都是绑定在Math类下面

静态方法或者属性

学习方式就是记住它的功能和用法

1.Math.round(number):将数值四舍五入为最接近的整数

```
// console.log(Math.round(3.335));//3
// console.log(Math.round(3.535));//4
```

2.Math.random():返回一个随机数,介于0-1之间，理论上包含0一定不包含1.

console.log(Math.random());

利用Math.random配合数学运算符产生需要的随机数

```
// 案例2:
// 封装函数，实现随机区间数    ranNum(4,9) 和 ranNum(9,4) 随机产生4, 5, 6, 7, 8, 9
// function ranNum(min, max) {
//   // 如果参1大于参2，交换位置
//   if (min > max) {
//     var temp = min;
//     min = max;
//     max = temp;
//   }
//   // 产生区间数,思路控制最小的值和最大的值
//   return Math.round(Math.random() * (max - min)) + min;
// }
// console.log(ranNum(4, 9));
// console.log(ranNum(0, 255));
```

3.Math.ceil(number):向上取整

```
// console.log(Math.ceil(3.0000001));//4
// console.log(Math.ceil(-3.999999));//-3
```

4.Math.floor(number)：向下取整

```
// console.log(Math.floor(3.999999));//3
```

5.Math.min()和 Math.max()方法用于确定一组数值中的最小值和最大值，可以接收任意多个数值参数

...扩展运算符，将数组转换成单个的值

```
// console.log(Math.max(12, 345, 56));//345
// console.log(Math.min(12, 345, 56));//12
// var arr = [12, 345, 56, 7, 7777, 8];
// console.log(Math.max(arr));//NaN 不支持数组做参数
// console.log(Math.max(...arr));//7777    ...扩展运算符，将数组转换成单个的值
```

6.Math.abs()：求绝对值

```
// console.log(Math.abs(-5));//5
// console.log(Math.abs(5));//5
```

7.Math.pow()：求幂，第一个参数是底数，第二个参数是指数。

** 新版本语法里面求幂的符号

```
// console.log(Math.pow(2, 3));
// console.log(2 ** 3);//** 新版本语法里面求幂的符号
```

8.Math.sqrt()：开根号

```
// console.log(Math.sqrt(4));
// console.log(Math.sqrt(3));
// console.log(Math.sqrt(2));
```

9.Math.PI：表示π

```
// console.log(Math.PI);//3.141592653589793
```

核心的方法

Math.random
Math.round
Math.ceil
Math.floor

Math.min
Math.max
Math.abs

日期对象的概述

1.构造函数创建日期对象

```
// var d = new Date();    // d创建的日期实例对象     Date构造函数
// console.log(d); // Thu Apr 27 2023 10:04:43 GMT+0800 (中国标准时间 东八区)
// console.log(d.toLocaleString());// 2023/4/27 10:11:07
```

```
//locale:本地  localStorage:本地存储  localhost:本地主机
```

2.时间戳 - 重点

Date 类型使用自 UTC（国际协调时间）1970 年 1 月 1 日午夜（零时）开始经过 的毫秒数来保存日期

时间戳是指格林威治时间1970年01月01日00时00分00秒起至现在的总的毫秒数。

1970年1月1日作为UNIX系统的纪元时间（开始时间）

北京时间1970年01月01日08时00分00秒

3.获取当前时间戳的方式

- 获取1970年01月01日00时00分00秒至现在的总毫秒数
- getTime方法获取时间戳
- Date.parse(new Date())

```
// var timestamp = new Date().getTime()
// var timestamp = Date.parse(new Date());
// var d = new Date();
// console.log(d.getTime());  //这是一个时刻都在改变的数字，以后需要这样的数字可以采用时间戳。
// console.log(Date.parse(d));
```

日期下面的方法

```
var d = new Date();
```

1.获取四位的年份

getFullYear()

```
console.log(d.getFullYear());//2023
```

2.获取月份:老外的月份使用数字0-11分别表示1-12月 -- 特殊情况

getMonth()

```
console.log(d.getMonth() + 1);//4
```

3.获取几天是几号

getDate()

```
console.log(d.getDate());//27
```

4.获取星期:老外的星期使用数字0-6分别表示星期日-星期六 -- 特殊情况

getMonth()

```
console.log(d.getMonth());//4
```

5.获取时分秒

getHours()
getMinutes()
getSeconds()

```
console.log(d.getHours() + ':' + d.getMinutes() + ':' + d.getSeconds());
console.log(`今天是${d.getFullYear()}年${d.getMonth() + 1}月${d.getDate()}日`);
```

定时器

一.间隔定时器(反复执行)

1.开启间隔定时器

- 每间隔多少时间就执行一次函数
- 语法：setInterval(要执行的函数，间隔多少时间ms)

```
// var num = 0
// var t = window.setInterval(function () {//函数会每隔1s执行一次，函数内部需要逻辑。    t是定时器的返回值，作用就是关闭定时器时有用
//    num++;
//    console.log(num);
// }, 1000);
```

2.关闭间隔定时器

- 关闭以后，定时器就不会在执行了
- 语法：clearInterval(要关闭的定时器返回值)

```
// btn.onclick = function () {
//    window.clearInterval(t);  //关闭返回值为t的定时器
// }
```

二.延时定时器(规定的时间内执行一次)

1.开启延时定时器

- 延时多少时间以后执行函数
- 语法：setTimeout(要执行的函数，多长时间以后执行)

```
// var t1 = window.setTimeout(function () {
//    console.log('定时器触发');
// }, 3000);
```

2.关闭延时定时器

- 关闭以后，定时器就不会在执行了
- 语法：clearTimeout(要关闭的定时器返回值)

```
// window.clearTimeout(t1);
```

设置日期

注意：不能设置星期

1.利用字符串(约定的格式)设置日期

```
var time = new Date('2023-4-28 16:00:00');
var time = new Date('2023/4/28 16:00:00');
var time = new Date('2023.4.28 16:00:00');
var time = new Date('2023年4月28日 16:00:00');//Invalid Date 无效日期
var time = new Date('2023-14-280 16:00:00');//Invalid Date 无效日期
console.log(time.toLocaleString());
```

2.利用数字设置日期(识别前六位-年月日时分秒)

注意：月份自动+1，允许进位(如果日期超过了限定范围向前进位)

```
var time = new Date(2023, 4, 28, 12, 13, 14, 15, 16, 17);
console.log(time.toLocaleString());
var time = new Date(2023, 14, 28, 12, 13, 14);
console.log(time.toLocaleString());//2024-3-28 12:13:14
```

3.利用ES提供的方法进行设置

可以进位

setFullYear 年

setMonth 月

setDate 日

setHours 时

setMinutes 分

setSeconds 秒

```
var d = new Date();
d.setDate(d.getDate() + 30);//d.setDate(27+30)    进位到下个月
console.log(d.toLocaleString()); //2023/5/27 14:50:14
```

同步和异步

- 首先，javascript不同于其他后端语言，javascript语言是单线程机制，只有一个主线程。
- 所谓单线程就是按顺序执行，执行完一个任务再执行下一个。
- 对于浏览器来说，也就是无法在渲染页面的同时执行其它代码。

总结：浏览器(JS代码的解释器)只有一个主线程，一次只能完成一个任务，然后继续下一个任务。

- 于是，所有任务可以分成两种，一种是同步任务，另一种是异步任务

1.同步任务(阻塞模式)：在主线程上的任务，只用前一个任务执行完毕，后一个任务才能紧接着执行。

```
// console.log('第一个任务');
```

```
// console.log('第二个任务');
// for (; ;) { console.log('假设这个任务要执行30s'); }//第三个任务等到上面的循环完成(30s之后)才能继续
// console.log('第三个任务');
```

注意：到目前位置，除了定时器，所有的代码都是以同步的方式在执行。

2.异步任务(非阻塞模式)：先不进主线程，而进入到任务队列，等待主线程任务(同步)完成，才能进入到主线程然后执行完毕。

定时器是异步任务

注意：异步代码等到同步代码执行完成，才执行。

```
// console.log(1);//同步
// setTimeout(function () {
//   console.log(2);//异步
// }, 0);
// console.log(3);//同步
// 输出的结果：1,3,2

// console.log(1);//同步
// setTimeout(function () {
//   console.log(2);//异步
// }, 3000);
// console.log(3);//同步

// 面试题：
for (var i = 1; i <= 5; i++) {//同步 i:1,2,3,4,5   i=6的时候for循环结束了
  window.setTimeout(function () {
    console.log(i);//输出5次6  循环一次开启一个定时器,但是代码进入了队列(异步),等到同步完成再执行.
  }, 1000);
}
console.log('我是同步代码，我再定时器后面，但我比定时器先执行');
```