3.3 运算符

3.3.1 运算符分类

- 算术运算符
- 比较运算符
- 逻辑运算符
- 赋值运算符
- 字符串 + 运算符
- 条件运算符
- typeof 运算符

3.3.2 算术运算符

- 给定 y=5，下面的表格解释了这些算术运算符：

运算符	描述	例子	结果
+	加	x=y+2	x=7
-	减	x=y-2	x=3
*	乘	x=y*2	x=10
/	除	x=y/2	x=2.5
%	求余数 (保留整数)	x=y%2	x=1
++	累加	x=++y	x=6
--	递减	x=--y	x=4

```
<script>
  //运算符我们重点给大家介绍
  //求余 %
  //累加 ++
  //1. 如果 ++ 运算符在变量前面，则先做加法，然后得结果
  //2. 如果 ++ 运算符在变量后面，则先使用，然后在做加法运算
  //递减 --
  //除法操作 / ，在JS中除法操作是含有小数部分

  var a = 10;
  var b = 3;
  console.log("a % b = " + a % b);

  var c = ++a; //原来的变量a做一次加1操作，同时变量a值已经变成11
  console.log("c = " + c);
  console.log("a = " + a);

  var d = a++;
  console.log("d = " + d);
  console.log("a = " + a);

  var e = a--;
  console.log("e  = " + e);
  console.log("a = " + a);

  var f = a / b;
  console.log("f = " + f);
</script>
```

3.3.3 赋值运算符

- 给定 y=5，下面的表格解释了这些算术运算符：

运算符	例子	等价于	结果
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

```
<script>
    var x = 2;
    var y = 5;

    // +=, 相当于 x += y; ==> x = x + y
    console.log("x += y : " + (x += y));

    // -=, 相当于 x -= y; ==> x = x - y
    console.log("x -= y : " + (x -= y));

    // /=, 相当于 x /= y; ==> x = x / y;
    console.log("x /= y : "+ (x /= y));
</script>
```

3.3.4 比较运算符

- 比较运算符又称为关系运算符
- 比较运算符的计算结果只有两种值：true 或 false

运算符	例子	结果
>	1 > 2	false
<	1 < 2	true
>=	1 >= 2	false
<=	1 <= 2	true
!=	1 != 2	true
==	1 == 2	false
===	1 === 2	false

- 注意：== 和 === 之间是有区别的
  - == 比较的两个数的值是否相当
  - === 不仅比较两个数的值，还要比较两个数的类型。所谓的全相等的才能返回 true

```
<script>
  var a = 10;
  var b = 9; // Number类型
  var c = 9; // Number类型
  var d = "9"; //字符串类型

  // 大于 >
  console.log("a > b = " + (a > b));

  // 大于等于 >=
  console.log("a >= b " + (a >= b));

  // 两个等于号 ==
  console.log("b == c " + (b == c));

  // 三个等于号 ===
  // 比较值和数据类型完全相等的时候才返回true
  console.log("b === c " + (b === c));

  console.log("b === d " + (b === d)); // false, 因为b是Number类型, d是字符串类型
  console.log("b == d " + (b == d)); //两个等号它是比较值
</script>
```

3.3.5 逻辑运算符

- 逻辑运算符是对两个布尔值类型做运算，且计算后的结果也是一个布尔值

运算符	例子	结果
&&	true && false	false
||	true || false	true
!	!true	false

```
<script>
  var a = true;
  var b = false;
  var c = true;
  var d = false;

  // &&, 要求布尔值全部为true, 则结果为true
  console.log("a && b = " + (a && b));
  console.log("a && c = " + (a && c));

  // ||, 要求布尔值全部为false, 则结果为false
  console.log("a || b = " + (a || b));
  console.log("b || d = " + (b || d));

  // !, 求布尔值相反的值
  console.log("!a = " + (!a));

  // 先用比较运算 + 逻辑运算
  var e = 10;
  var f = 5;
  var h = 12;
  console.log("e > f && e > h = " + (e > f && e > h));
</script>
```

3.3.6 字符串 + 运算符

- 字符串 + 运算符用于把文本值或字符串变量加起来（连接起来）

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

在以上语句执行后，变量 txt3 包含的值是 "What a verynice day"。

```
<script>
  //字符串+，则这个+就是一个连接作用
  var a = 10;
  var b = "Hello";
  var c = 5;
  var d = true;
  var e = false;

  console.log(a + b);

  console.log(a + b + c);

  console.log(b + a + c); //字符串+和算术运算+级别是同等

  console.log(b + a / c); //除法运算符的优先级高于字符串+

  console.log(b + a % c); //求余数运算符优先级高于字符串+

  console.log(b + ++a); //递增运算符优先级高于字符串+

  console.log(b + (a > c)); //比较运算符和字符串+的运算，需要注意运算符优先级

  console.log(b + (d && e));  //逻辑运算符和字符串+的运算，需要注意运算符优先级
</script>
```

3.3.7 条件运算符

条件运算符是 ECMAScript 中功能最多的运算符，它的形式与 Java 中的相同。

```
variable = boolean_expression ? true_value : false_value;
```

该表达式主要是根据 *boolean_expression* 的计算结果有条件地为变量赋值。如果 *Boolean_expression* 为 true，就把 *true_value* 赋给变量；如果它是 false，就把 *false_value* 赋给变量。

例如：

```
var iMax = (iNum1 > iNum2) ? iNum1 : iNum2;
```

在这里例子中，iMax 将被赋予数字中的最大值。表达式声明如果 iNum1 大于 iNum2，则把 iNum1 赋予 iMax。但如果表达式为 false（即 iNum2 大于或等于 iNum1），则把 iNum2 赋予 iMax。

```
<script>
    var a = 10;
    var b = 8;
    var d = 18;
    var c = a > b ? a : b;
    console.log("c = " + c);

    //求出三个数字中最大的数是多少
    c = a > b ? (a > d ? a : d) : (b > d ? b : d);
    console.log("c = " + c);

    //求出三个数字中最小的数是多少
    c = a < b ? (a < d ? a : d) : (b < d ? b : d);
    console.log("c = " + c);
</script>
```

3.3.8 typeof 运算符

- typeof 主要计算某个变量的数据类型是什么
- 语法：typeof 变量

```
<script>
    var a1 ;
    console.log(a1);//undefined
    console.log(typeof a1);//undefined
    console.log(typeof a2);//undefined
    console.log(a2);//Uncaught ReferenceError: a2 is not defined
</script>
```

```
<script>
    //变量的数据类型
    //Number类型, undefined类型, String类型, Boolean类型
    var a = 10;
    var b = "HelloWorld";
    var c = true;
    var d;
    console.log("typeof(a) = " + typeof(a));
    console.log("typeof(b) = " + typeof(b));
    console.log("typeof(c) = " + typeof(c));
    console.log("typeof(d) = " + typeof(d));
</script>
```

3.4 控制语句

3.4.1 控制语句分类

- 条件控制语句，if..else if..else 和 switch
- 循环控制语句，while, do..while, for

3.4.2 if 条件控制语句

通常在写代码时，您总是需要为不同的决定来执行不同的动作。您可以在代码中使用条件语句来完成该任务。

在 JavaScript 中，我们可使用以下条件语句：

- **if 语句** - 只有当指定条件为 true 时，使用该语句来执行代码
- **if...else 语句** - 当条件为 true 时执行代码，当条件为 false 时执行其他代码
- **if...else if....else 语句** - 使用该语句来选择多个代码块之一来执行
- **switch 语句** - 使用该语句来选择多个代码块之一来执行

If 语句

只有当指定条件为 true 时，该语句才会执行代码。

语法

```
if (条件)
  {
  只有当条件为 true 时执行的代码
  }
```

注意：请使用小写的 if。使用大写字母（IF）会生成 JavaScript 错误！

If...else 语句

请使用 if....else 语句在条件为 true 时执行代码，在条件为 false 时执行其他代码。

语法

```
if (条件)
  {
  当条件为 true 时执行的代码
  }
else
  {
  当条件不为 true 时执行的代码
  }
```

If...else if...else 语句

使用 if....else if...else 语句来选择多个代码块之一来执行。

语法

```
if (条件 1)
  {
  当条件 1 为 true 时执行的代码
  }
else if (条件 2)
  {
  当条件 2 为 true 时执行的代码
  }
else
  {
  当条件 1 和 条件 2 都不为 true 时执行的代码
  }
```

```
<script>
    //通过浏览器弹出一个输入框
    //然后我们在输入框中输入一段文字
    //判断文字符合某种条件，则运行不同代码
    var a = prompt("提示","请输入自己名字");
    if (a == "张三") {
        console.log("欢迎你张三");
    } else if (a == "李四") {
        console.log("欢迎你李四");
    } else {
        console.log("欢迎你帅哥和美女");
    }

    //if语句可以单独的存在
    //else if必须在前面有if语句存在
    //必须先有if语句，才有else语句
</script>
```

3.4.3 switch 控制语句

- switch 语句用于基于不同的条件来执行不同的动作
- switch 语句是 if 语句的兄弟语句

语法

```
switch(n)
{
case 1:
  执行代码块 1
  break;
case 2:
  执行代码块 2
  break;
default:
  n 与 case 1 和 case 2 不同时执行的代码
}
```

工作原理：首先设置表达式 n（通常是一个变量）。随后表达式的值会与结构中的每个 case 的值做比较。如果存在匹配，则与该 case 关联的代码块会被执行。请使用 **break** 来阻止代码自动地向下一个 case 运行。

- 简单案例：获取当前时间是星期几

显示今日的周名称。请注意 Sunday=0, Monday=1, Tuesday=2, 等等：

```
var day=new Date().getDay();
switch (day)
{
case 0:
  x="Today it's Sunday";
  break;
case 1:
  x="Today it's Monday";
  break;
case 2:
  x="Today it's Tuesday";
  break;
case 3:
  x="Today it's Wednesday";
  break;
case 4:
  x="Today it's Thursday";
  break;
case 5:
  x="Today it's Friday";
  break;
case 6:
  x="Today it's Saturday";
  break;
}
```