# 复习

alert(): 弹出框

console.log(): 在console面板輸出

prompt(提示文本，默认文本): 提供给用户进行输入的方法。 结果数据类型：string

parseInt(): 取整

parseFloat(): 浮点数

字面量：固定的值（数字字面量，字符串字面量，布尔值，undefined, null）

变量：盛放数据的容器

声明变量var a; 声明变量没有赋值结果是undefined

赋值： = （将右侧的数据赋值给左侧的变量）

变量声明提升：预解析，只提升变量声明，不提升赋值语句。

# 运算符

## math

Math是js内置对象

Math.random(): 随机数； 参数：不书写； 结果是0-1之间的浮点数

Math.pow(，): 幂的运算； 参数：一个参数底数 二个参数指数 结果是幂的计算数值结果

Math.sqrt(): 开方运算 参数：一个参数，要开方的数字 结果：开方数值

Math.PI 圆周率π（属性）

## 比较运算符

> 大于

< 小于

>= 大于等于

<= 小于等于

== 等于

=== 全等于

!= 不等于

!== 不全等于

（只要是比较运算符，结果一定是布尔值）

* 两侧都是数字参与
* 布尔值，null参与运算时，隐式转换。true(1) false(0) null(0), 特别的null参与==，===时不是转换0
* 纯数字字符串参与时也是隐式转换。
* ==等于，在判断两侧关系尽可能让两侧相等。只判断数值是否相同，不判断数据类型是否相等。

|  |
| --- |
| 1. console.log("2" == 2); |

* ===全等于，尽可能让两侧关系不等。同时判断数值和数据类型是否相同
* != 不等于，和==是对立面。只要==是true，那么!=一定是false；只要==是false，那么!=一定是true
* !== 不全等于，和===是对立面。只要===是true，那么!==一定是false；只要===是false，那么!==一定是true

计算：3 > 2 > 1 (顺序：从左到右)

|  |
| --- |
| 1. = true > 1 2. = false |

* 两侧都是字符串比较，不是比较字符串长短，而是比较字符的Unicode。编码顺序越靠后，越大

顺序：数字（0-9），大写字母（A-Z），小写字母（a-z）

## 逻辑运算符

&& :逻辑与运算符

|| 逻辑或运算符

! 逻辑非运算符

* 布尔值参与运算，结果一定是布尔值

### 逻辑与真值表

|  |  |  |
| --- | --- | --- |
| a | b | a&&b结果 |
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | false |

总结：都真才真，一假则假

### 逻辑或运算符

|  |  |  |
| --- | --- | --- |
| a | b | a || b结果 |
| true | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

总结：都假才假，一真则真

### 逻辑非运算符

!true 结果false

!false 结果true

非布尔值参与逻辑运算结果不一定是布尔值

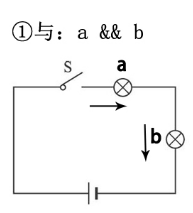
## 短路语法

隐式转换为真：非0数字(Infinity)，非空字符串

隐式转为假：0， “”，undefined, NaN, null

①逻辑与&&

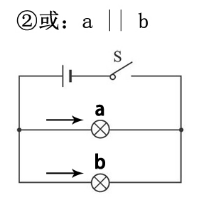
将逻辑与比喻成串联电路，判断过程，想象成电流通过的过程。



电流通过：如果a为真，电流能够通过流通到b，结果就是b；如果a为假，电流不能通过，停留在a，结果为a。

②逻辑或||

将逻辑或比喻成并联电路，判断过程，想象成电流通过的过程。



电流经过时：如果a为真，电流直接从a完成循环，结果为a；如果a为假，电流从a不能经过，流经到b，我们结果是b。

逻辑运算符优先级：先非，再与最后或

|  |
| --- |
| 1. /\* 2. 真：非0数字，非空字符串 3. 假：0， “”， undefined, NaN, null 4. a&&b: 如果a是真，结果是b;a假，结果是a 5. a||b: 如果a是真，结果是a;a假，结果是b 6. 非 与 或 7. \*/ 8. var sum = 4 && "hello" || !false || !true && null; 9. /\* 10. = 4 && "hello" || true || false && null; 11. = "hello" || true || false && null; 12. = "hello" || true || false; 13. = "hello" || false 14. = "hello" 15. \*/ 16. console.log(sum); |

## 1.5 赋值运算符

赋值运算符是赋一个基于它右边操作数值的值给它左边的操作数。最简单的赋值运算符是等于（=），将右边的操作数值赋给左边的操作数。那么 x = y 就是将 y 的值赋给 x。

含义：将右侧的数据**运算**之后，赋值给左边。

+= 加等于

-= 减等于

\*= 乘等于

/= 除等于

%= 取余等于

++ 递加

-- 递减

在变量原基础上，进行右侧的运算，将运算的值再赋值给左边的变量。

|  |
| --- |
| 1. // 声明变量 2. var a = 10, b = 20, c = 30; 3. var sum = (a ++) + (++ b) + (c ++) + (++ a); 4. /\* 5. = 10 + 21 + 30 + (++ a) 第二次出现就是再次使用，再次使用都是自加1后的结果 6. = 61 + 12 7. = 73 8. \*/ 9. console.log(sum); 10. console.log(a); 11. console.log(b); 12. console.log(c); |

## 综合运算

顺序：贴身的（! ++ --）→ 数学 → 比较 → 逻辑 → 赋值

|  |
| --- |
| 1. /\* 2. 1优先级： 贴身，数学，比较(结果一定是布尔值)，逻辑(与，或)，赋值 3. 2a++ :使用变量当前值参与运算，a第二次使用是自加1后的结果 4. 3真：非0数字，非空字符串 5. a&&b: 6. a||b: 7. \*/ 8. var a = 4; 9. var sum = 1 \* (2 + 3) && a++ || 5 > 6 && 7 < 8 || 9; 10. /\* 11. = 1 \* (2 + 3) && 4 || 5 > 6 && 7 < 8 || 9; 12. = 5 && 4 || 5 > 6 && 7 < 8 || 9; 13. = 5 && 4 || false && true || 9 14. = 4 || false && true || 9 15. = 4 || false || 9 16. = 4 || 9 17. = 4 18. \*/ 19. console.log(sum); 20. // 第二次出现a = 5 21. console.log(a); 22. console.log(++ a); |

|  |
| --- |
| 1. /\* 2. 1优先级： 贴身，数学，比较(结果一定是布尔值)，逻辑(与，或)，赋值 3. 2a++ :使用变量当前值参与运算，a第二次使用是自加1后的结果 4. 3真：非0数字，非空字符串 5. a&&b: 6. a||b: 7. 4数学运算：纯数字字符串，true (1),false(0),null(0) 8. \*/ 9. var a = 4; 10. var sum = 1 + 2 && 3 \* a++ % 5 || 6 < 7 == 8 / !false; 11. /\* 12. = 1 + 2 && 3 \* 4 % 5 || 6 < 7 == 8 / true; 13. = 3 && 2 || 6 < 7 == 8 14. = 3 && 2 || true == 8 15. = 3 && 2 || false 16. = 2 || false 17. = 2 18. \*/ 19. console.log(sum); 20. console.log(a); |

# 流程控制语句

js 中执行顺序是从上到下。我们可以通过流程控制语句改变js语句的执行顺序，让某些语句先执行，某语句不执行。

通过条件来控制语句的执行顺序。

条件分支语句分类：if语句，三元表达式，switch语句。

## if语句

语法：if 如果…… else否则

|  |
| --- |
| 1. if (条件表达式) { 2. 当满足条件表达式，执行结构体1; 3. }else { 4. 当不满条件表达式，执行结构体2； 5. } |

1条件表达式：结果一定是布尔值，结果如果是true，满足条件；

结果如果是false,不满足条件

2 结构体：用大括号包裹js语句。

|  |
| --- |
| 1. // 当用户输入分数，告诉用户是否及格 2. // 的到用户输入的分数 3. var score = parseInt(prompt("请输入该次考试分数",100)); 4. // if 如果 else 否则 5. if (score >= 60) { 6. // 及格，该大括号结构体执行的条件 score>=60结果是true 7. console.log("恭喜及格"); 8. console.log("该次你的考试分数是" + score); 9. }else { 10. // 否则，对上面条件否定 11. // 结构体：不满足条件表达式才执行该结构体 12. // 隐含条件 score < 60 13. console.log("加把劲，努力"); 14. } 15. /\* 16. 当用户输入40， 17. 11 score = 40 18. 13 计算score >= 60结果是false不满足条件，执行else后面的结构体 19. 输出"努力" 20. \*/ |

* else 语句可以省略，当不满足条件，直接跳出if语句，执行后面其他的语句。

|  |
| --- |
| 1. // 请输入输入 2. var score = parseInt(prompt("请输入分数")); 3. // if语句 4. if (score >= 60) { 5. console.log("及格"); 6. } 7. // 后面语句 8. console.log("查询结束"); |

* 当结构体里面只有一条语句时，可以省略大括号

|  |
| --- |
| 1. // 请输入天气,晴朗，有风 2. var w = prompt("请输入天气"); 3. // 如果晴朗，出去玩 不晴朗学习 4. if (w == "晴朗") 5. console.log("出去玩耍"); 6. else 7. console.log("踏实学习"); 8. console.log("天气查询完毕"); 9. /\* 10. 1输入晴朗 11. 条件满足，执行if后面的结构体，输出“玩耍”，if语句跳出，执行后面的语句，输出“查询” 12. 2输入有风 13. 不满足条件，执行else后面的结构体，输出“学习”，跳出if语句，执行后面的语句 14. 输出“查询” 15. \*/ |

## 2.2 多分支的if语句

|  |
| --- |
| 1. if (条件1) { 2. 满足条件1，执行结构体1； 3. }else if (条件2) { 4. 不满足条件1，满足条件2执行结构体2 5. }else if (条件3) { 6. 不满足条件1和2，满足条件3执行结构体3； 7. }…… 8. else { 9. 以上条件都不满足，执行该结构体 10. } |

多分支的if语句，结构体只会执行一个，执行完成，立即跳出if语句。

* 多分支的if语句else语句也可以省略，以上条件都不满足，直接跳出if,执行后面其他语句

|  |
| --- |
| 1. var score = parseInt(prompt("请输入分数")); 2. if (score >= 90) { 3. console.log("优秀"); 4. }else if (score >= 80) { 5. console.log("良好"); 6. }else if (score >= 70) { 7. console.log("还行"); 8. }else if (score >= 60) { 9. console.log("及格"); 10. } 11. // 后面语句的语句 12. console.log("查询结束"); |

* “殊途同归”，不管if语句执行哪一个结构体，if语句后面的其他语句都会执行
* “跳楼现象”多分支的if语句只会执行一个结构体，然后立即跳出if语句

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |