JavaScript 第二天

流程控制

学习目标

Learning Objectives

1. 掌握算术、比较、逻辑运算符，为程序“能思考”做准备
2. 掌握分支语句，让程序具备判断能力
3. 掌握循环语句，让程序具备重复执行能力

目录

Contents

运算符

- 算术运算符
- 赋值运算符
- 一元运算符
- 比较运算符
- 逻辑运算符
- 运算符优先级

1.1 算术运算符

目标：掌握算术运算符，能写出一些具备运算能力的小程序

数学运算符也叫算术运算符，主要包括加、减、乘、除、取余（求模）。

- +：求和
- -：求差
- *：求积
- /：求商
- %：取模（取余数）
  - 开发中经常作为某个数字是否被整除

1.1 算术运算符

目标：能说出JavaScript算术运算符执行的优先级顺序

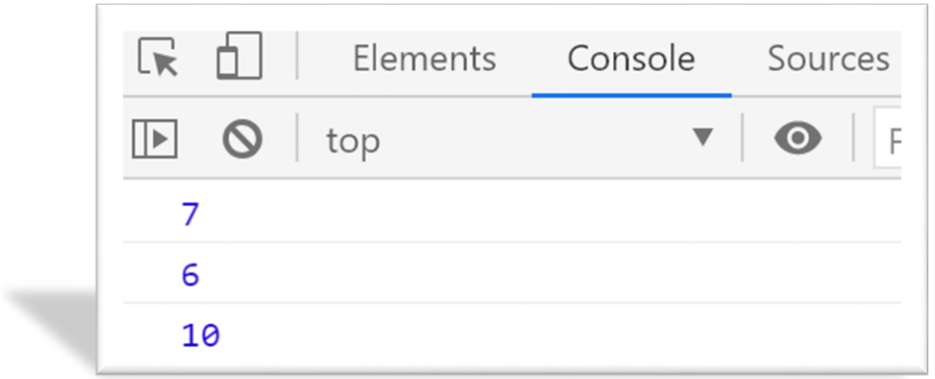
同时使用多个运算符编写程序时，会按着某种顺序先后执行，我们称为优先级。

JavaScript中 优先级越高越先被执行，优先级相同时以书从左向右执行。

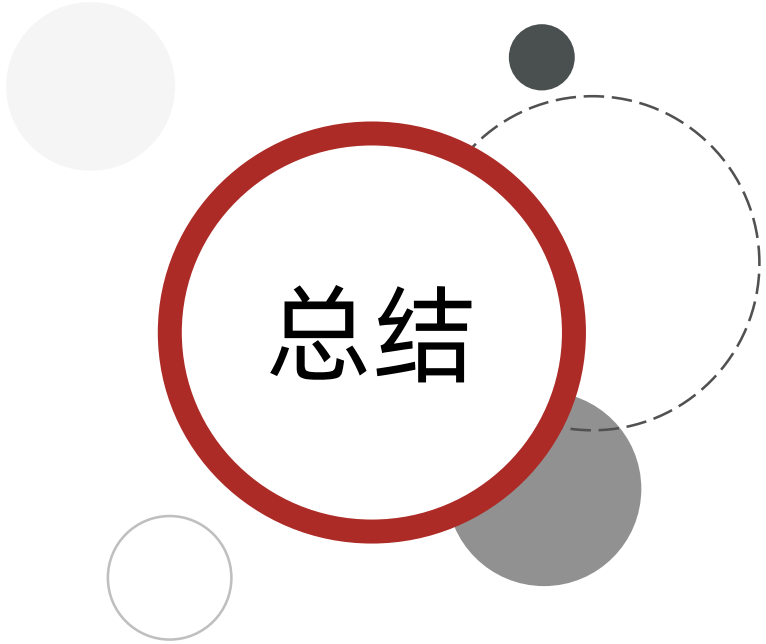
- 乘、除、取余优先级相同
- 加、减优先级相同
- 乘、除、取余优先级大于加、减
- 使用 () 可以提升优先级
- 总结： 先乘除后加减，有括号先算括号里面的~~~

提问：

```
console.log(1 + 2 * 3)
console.log(10 - 8 / 2)
console.log(2 % 5 + 4 * 2)
```



算术运算符小结



1. 算术运算符有那几个常见的？
   - \+ - * / %
2. 算术运算符优先级怎么记忆？
   - 先乘除取余，后加减，有小括号先算小括号里面的
3. 取余运算符开发中的使用场景是？
   - 来判断某个数字是否能被整除

案例

计算圆的面积

需求：对话框中输入圆的半径，算出圆的面积并显示到页面

分析：

①：面积的数学公式： $\pi*r^2$

②：转换为JavaScript写法： 变量 * r * r

运算符

- 算术运算符
- 赋值运算符
- 一元运算符
- 比较运算符
- 逻辑运算符
- 运算符优先级

赋值运算符

- 赋值运算符：对变量进行赋值的运算符
  - 已经学过的赋值运算符：= 将等号右边的值赋予给左边，要求左边必须是一个容器
  - 其他赋值运算符：
    - +=
    - -=
    - *=
    - /=
    - %=
- 使用这些运算符可以在对变量赋值时进行快速操作

1.2 赋值运算符

赋值运算符

我们以 += 赋值运算符来举例

1. **以前**我们让一个变量加 1 如何做的？

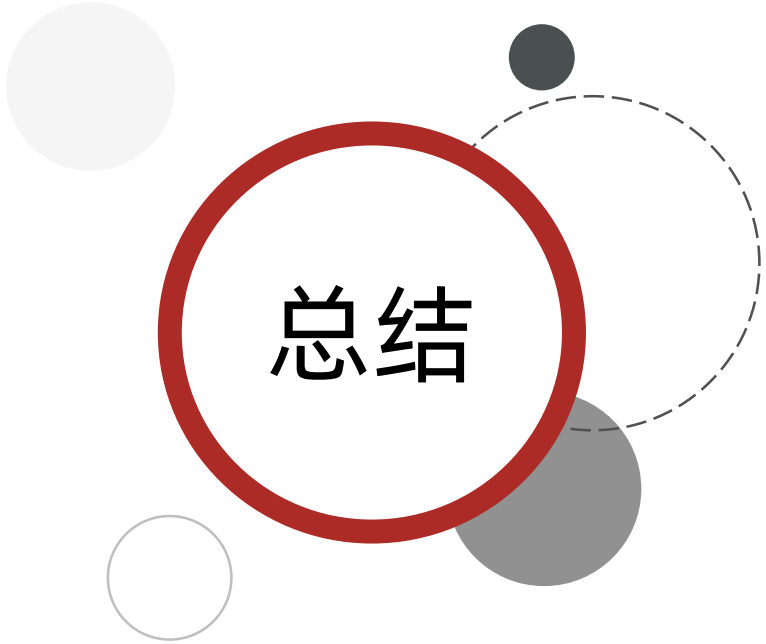
```
<script>
    let num = 1
    num = num + 1
    console.log(num)  // 结果是 2
</script>
```

2. **现在**我们有一个简单的写法啦~~~

```
<script>
    let num = 1
    num += 1
    console.log(num)  // 结果是 2
</script>
```

提问：想变量加3怎么写？

赋值运算符总结



1. = 赋值运算符执行过程？

   - 将等号右边的值赋予给左边, 要求左边必须是一个容器

2. += *= 出现是为了简化代码, 比如让 let num = 10 ， num 加5 怎么写呢？

   - num += 5

运算符

- 算术运算符
- 赋值运算符
- 一元运算符
- 比较运算符
- 逻辑运算符
- 运算符优先级

1.3 一元运算符

目标： 能够使用一元运算符做自增运算

众多的 JavaScript 的运算符可以根据所需表达式的个数，分为一元运算符、二元运算符、三元运算符

- 二元运算符：
  - 例：

```
let num = 10 + 20
```

- 一元运算符：
  - 例： 正负号

问题： 我们以前让一个变量每次+1 ，以前我们做的呢？

```
let num = 1
num = num + 1
```

```
let num = 1
num += 1
```

1.3 一元运算符

目标： 能够使用一元运算符做自增运算

- 我们可以有更简便的写法了~~~
- 自增：
  - 符号：++
  - 作用：让变量的值 +1
- 自减：
  - 符号：--
  - 作用：让变量的值 -1
- 使用场景：

  经常用于计数来使用。 比如进行10次操作，用它来计算进行了多少次了

1.3 一元运算符

目标： 能够使用一元运算符做自增运算

自增运算符的用法：

◆ **前置自增：**

```
let num = 1
++num       //让num的值加 1变 2
```

- 每执行1次，当前变量数值加1
- 其作用相当于 num += 1

◆ **后置自增：**

```
let num = 1
num++       //让num的值加 1变 2
```

- 每执行1次，当前变量数值加1
- 其作用相当于 num += 1

前置自增和后置自增单独使用没有区别

1.3 一元运算符

目标： 能够说出自增/减运算符前置或后置的差异

自增运算符的用法：

前置自增和后置自增如果参与运算就有区别: (难点)

◆ **前置自增：**

➢ 前置自增：先自加再使用（记忆口诀：++在前 先加）

```
let i = 1
console.log(++i + 2)  //结果是 4
//  注意: i是 2
//  i先自加 1,变成2之后，在和后面的2相加
```

◆ **后置自增：**

➢ 后置自增：先使用再自加（记忆口诀：++在后 后加）

```
let i = 1
console.log(i++ + 2)  //结果是 3
// 注意:   此时的 i是 1
// 先和2相加,先运算输出完毕后，i再自加是2
```

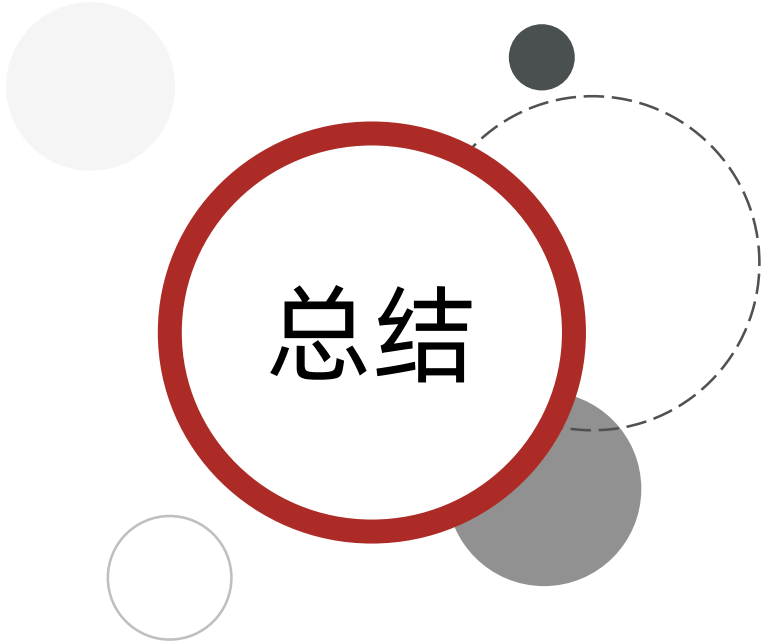
1.3 一元运算符

自增运算符的用法：

but：

1. 前置自增和后置自增独立使用时二者并没有差别！
2. 一般开发中我们都是独立使用
3. 后面 i++ 后置自增会使用相对较多

赋值运算符总结



1.只需要一个表达式就可以运算的运算符叫一元运算符

2. 自增运算符也是为了简化写法，每次自加1，使用场景是什么？

- 经常用于计数来使用。用来计算多少次

2.前后置自增的区别

- 前置：先自增后运算
- 后置：先运算后自增
- 自减同理..
- 开发中，我们一般都是单独使用的，后置++ 使用更多

思考： 面试题

面试题：

```
let i = 1
console.log(i++ + ++i + i)
```

运算符

- 算术运算符
- 赋值运算符
- 一元运算符
- 比较运算符
- 逻辑运算符
- 运算符优先级

1.4 比较运算符

目标：掌握比较运算符，为程序“能思考”做准备

学习路径：

1. 比较运算符的介绍
2. 比较运算符的使用
3. 比较运算符的细节

1.4 比较运算符

1. 比较运算符的介绍

- 比较运算符的介绍
  - 作用：比较两个数据大小、是否相等
  - 实际运用例：

1.4 比较运算符

目标: 能使用常见的比较运算符进行比较运算

2. 比较运算符的使用

- 比较运算符:
  - > : 左边是否大于右边
  - <: 左边是否小于右边
  - >=: 左边是否大于或等于右边
  - <=: 左边是否小于或等于右边
  - ==: 左右两边是否相等
  - ===: 左右两边是否类型和值都相等
  - !==: 左右两边是否不全等
  - 比较结果为boolean类型，即只会得到true或false

1.4 比较运算符

3. 比较运算符的细节

- 字符串比较，是比较的字符对应的ASCII码
  - 从左往右依次比较
  - 如果第一位一样再比较第二位，以此类推
  - 比较的少，了解即可

- NaN不等于任何值，包括它本身

- 尽量不要比较小数，因为小数有精度问题

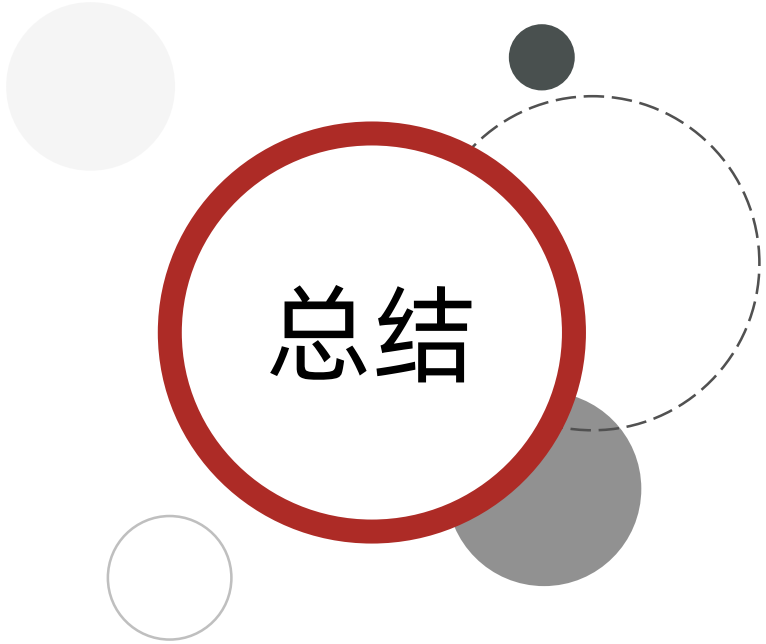
- 不同类型之间比较会发生隐式转换
  - 最终把数据隐式转换转成number类型再比较
  - 所以开发中，如果进行准确的比较我们更喜欢 === 或者 !==

ASCII 字符代码表 一

高四位 / 低四位		ASCII非打印控制字符										ASCII 打印字符													
		0000					0001					0010		0011		0100		0101		0110		0111			
		0					1					2		3		4		5		6		7			
		十进制	字符	ctrl	代码	字符解释	十进制	字符	ctrl	代码	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	ctrl
0000	0	0	BLANK NULL	^@	NUL	空	16	►	^P	DLE	数据链路转意	32		48	0	64	@	80	P	96	`	112	p		
0001	1	1	☺	^A	SOH	头标开始	17	◄	^Q	DC1	设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q		
0010	2	2	☻	^B	STX	正文开始	18	↕	^R	DC2	设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r		
0011	3	3	♥	^C	ETX	正文结束	19	‼	^S	DC3	设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s		
0100	4	4	♦	^D	EOT	传输结束	20	¶	^T	DC4	设备控制 4	36	$	52	4	68	D	84	T	100	d	116	t		
0101	5	5	♣	^E	ENQ	查询	21	§	^U	NAK	反确认	37	%	53	5	69	E	85	U	101	e	117	u		
0110	6	6	♠	^F	ACK	确认	22	▬	^V	SYN	同步空闲	38	&	54	6	70	F	86	V	102	f	118	v		
0111	7	7	•	^G	BEL	震铃	23	↨	^W	ETB	传输块结束	39	'	55	7	71	G	87	W	103	g	119	w		
1000	8	8	◘	^H	BS	退格	24	↑	^X	CAN	取消	40	(	56	8	72	H	88	X	104	h	120	x		
1001	9	9	○	^I	TAB	水平制表符	25	↓	^Y	EM	媒体结束	41	)	57	9	73	I	89	Y	105	i	121	y		
1010	A	10	◙	^J	LF	换行/新行	26	→	^Z	SUB	替换	42	*	58	:	74	J	90	Z	106	j	122	z		
1011	B	11	♂	^K	VT	竖直制表符	27	←	^[	ESC	转意	43	+	59	;	75	K	91	[	107	k	123	{		
1100	C	12	♀	^L	FF	换页/新页	28	∟	^\	FS	文件分隔符	44	,	60	<	76	L	92	\	108	l	124	|		
1101	D	13	♪	^M	CR	回车	29	↔	^]	GS	组分隔符	45	-	61	=	77	M	93	]	109	m	125	}		
1110	E	14	♫	^N	SO	移出	30	▲	^6	RS	记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~		
1111	F	15	☼	^O	SI	移入	31	▼	^-	US	单元分隔符	47	/	63	?	79	O	95	_	111	o	127	Δ	^Back space	

注：表中的ASCII字符可以用:ALT + "小键盘上的数字键" 输入

1.4 比较运算符总结



1. = 和 == 和 === 怎么区别？
   - = 是赋值
   - == 是判断 只要求值相等，不要求数据类型一样即可返回true
   - === 是全等 要求值和数据类型都一样返回的才是true
   - 开发中，请使用 ===
2. 比较运算符返回的结果是什么？
   - 结果只有2个， true 或者 false

运算符

- 算术运算符
- 赋值运算符
- 一元运算符
- 比较运算符
- 逻辑运算符
- 运算符优先级

1.5 逻辑运算符

目标：掌握逻辑运算符，为程序“能思考”做准备

学习路径：

1. 逻辑运算符的介绍
2. 逻辑运算符的使用
3. 逻辑运算符里的短路

1.5 逻辑运算符

1. 逻辑运算符的介绍

- 提问：如果我想判断一个数据大于5且小于10，怎么办？
  - 错误写法： 5 < 数据 < 10

- 逻辑运算符用来解决多重条件判断

2. 逻辑运算符的使用

- 逻辑运算符：

符号	名称	日常读法	特点	口诀
&&	逻辑与	并且	符号两边都为true 结果才为true	一假则假
||	逻辑或	或者	符号两边有一个 true就为true	一真则真
!	逻辑非	取反	true变false false变true	真变假，假变真

1.5 逻辑运算符

目标：能说出短路运算符的运算规则

3. 逻辑运算符里的短路

- 短路：只存在于 && 和 || 中，当满足一定条件会让右边代码不执行

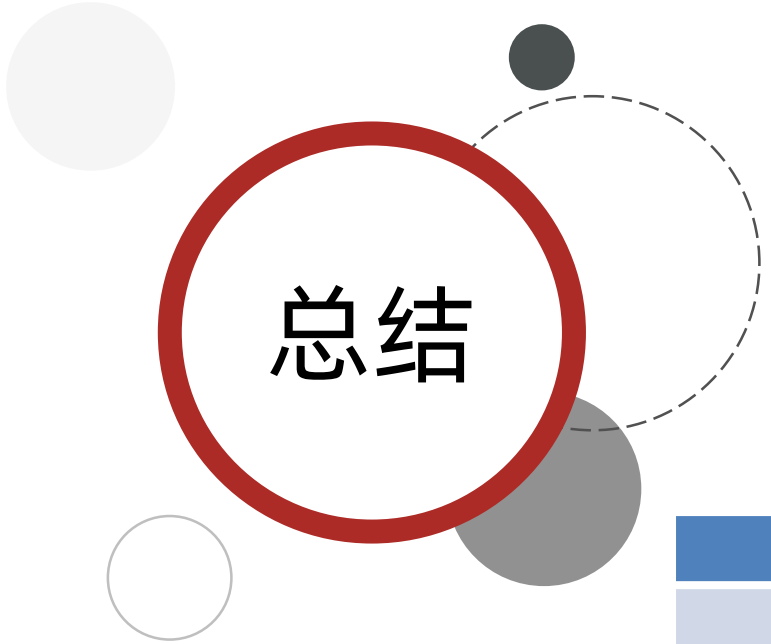
符号	短路条件
&&	左边为false就短路
||	左边为true就短路

- 原因：通过左边能得到整个式子的结果，因此没必要再判断右边
- 运算结果：无论 && 还是 ||，运算结果都是最后被执行的表达式值，一般用在变量赋值

```
console.log(false && 20)  // false
console.log(5 < 3 && 20) // flase
console.log(undefined && 20) // undefined
console.log(null && 20) // null
console.log(0 && 20) // 0
console.log(10 && 20) // 20
```

```
console.log(false || 20)  // 20
console.log(5 < 3 || 20) // 20
console.log(undefined || 20) // 20
console.log(null || 20) // 20
console.log(0 || 20) // 20
console.log(10 || 20) // 10
```

1.5 逻辑运算符总结



1. 逻辑运算符有那三个?
   - 与(&&)　或(||)　非(!)
2. 逻辑运算符短路运算符怎么执行的?
   - 只存在于 && 和 || 中，当满足一定条件会让右边代码不执行

符号	短路条件
&&	左边为false就短路
||	左边为true就短路

练习

判断一个数是4的倍数，且不是100的倍数

需求：用户输入一个，判断这个数能被4整除，但是不能被100整除

分析：

①：用户输入

②：控制台： 是否能被4整除并且100整除

运算符

- 算术运算符
- 赋值运算符
- 一元运算符
- 比较运算符
- 逻辑运算符
- 运算符优先级

1.6 运算符优先级

目标：掌握运算符优先级，能判断运算符执行的顺序

优先级	运算符	顺序
1	小括号	()
2	一元运算符	++ -- !
3	算数运算符	**先 * / % 后 + -**
4	关系运算符	> >= < <=
5	相等运算符	== != === !==
6	逻辑运算符	**先 && 后 ||**
7	赋值运算符	=
8	逗号运算符	,

- 一元运算符里面的逻辑非优先级很高
- 逻辑与比逻辑或优先级高

1.6 运算符优先级

练习

```
let a = 3 > 5 && 2 < 7 && 3 == 4
console.log(a);

let b = 3 <= 4 || 3 > 1 || 3 != 2
console.log(b);

let c = 2 === "2"
console.log(c);

let d = !c || b && a
console.log(d);
```

答案是false，此时发生了逻辑与中断

答案是true，此时发生了逻辑或中断

答案是false 数据类型不匹配

答案是true，此时发生了逻辑或中断

目录

Contents

语句

- 表达式和语句
- 分支语句
- 循环语句

2.1 表达式和语句

目标：能说出表达式和语句的区别

- 表达式：

表达式是一组代码的集合，JavaScript解释器会将其计算出一个结果

```
x = 7
3 + 4
num++
```

2.1 表达式和语句

目标：能说出表达式和语句的区别

- 语句：

js 整句或命令，js 语句是以分号结束（可以省略）

比如： if语句 for 循环语句

2.1 表达式和语句

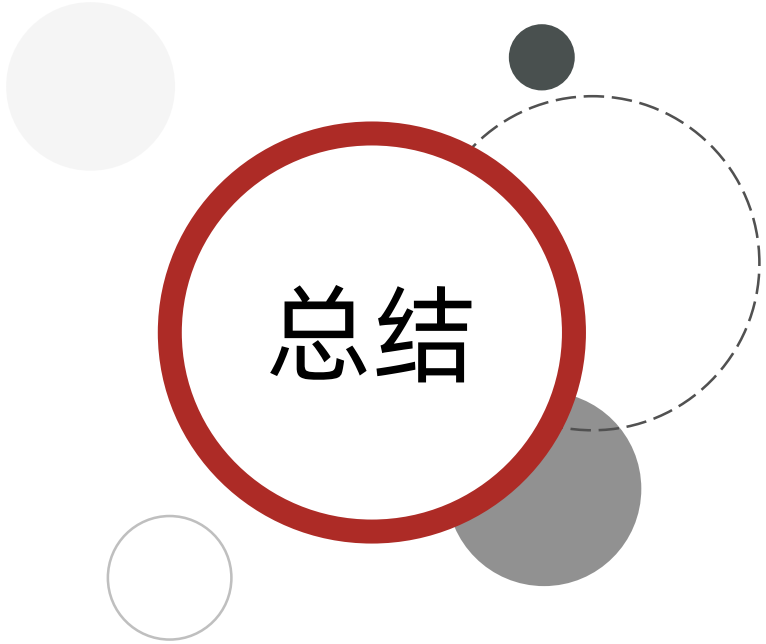
目标：能说出表达式和语句的区别

- 区别：

表达式计算出一个值，但语句用来自行以使某件事发生(做什么事)

- 表达式　　3 + 4
- 语句　　alert() 弹出对话框

其实某些情况，也可以把表达式理解为语句，因为它是在计算结果，也是做事



1. 表达式和语句的区别
   - 表达式计算出一个值 比如 3+5　x = 7
   - 语句用来自行以使某件事发生(做什么事)
     - alert()
     - console.log()
     - 还比如我们接下来学的分支语句..

语句

- 表达式和语句
- 分支语句
- 循环语句

2.2 分支语句

目标：掌握流程控制，写出能“思考”的程序

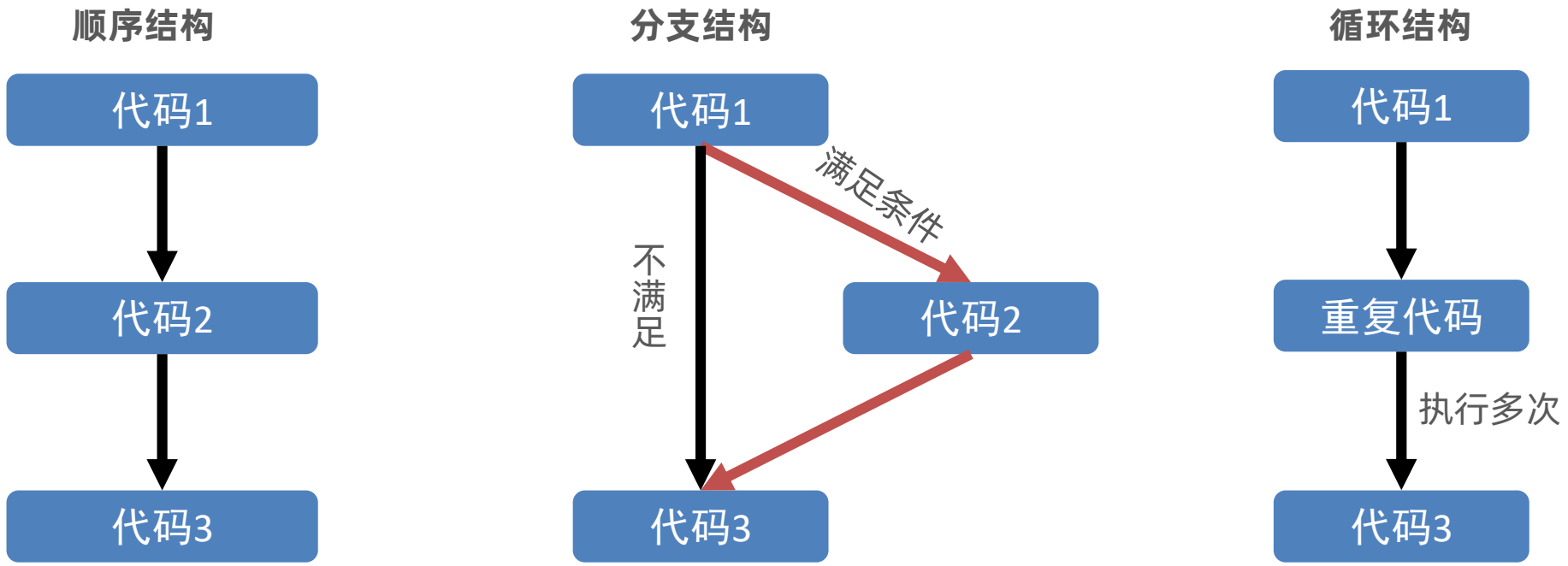
学习路径：

1. 程序三大流程控制语句
2. 分支语句

2.2 分支语句

1. 程序三大流程控制语句

- 以前我们写的代码，写几句就从上往下执行几句，这种叫顺序结构
- 有的时候要根据条件选择执行代码，这种就叫分支结构
- 某段代码被重复执行，就叫循环结构



2.2 分支语句

目标：掌握流程控制，写出能“思考”的程序

学习路径：

1. 程序三大流程控制语句
2. 分支语句

2. 分支语句

- 分支语句可以让我们有选择性的执行想要的代码
- 分支语句包含：
  - If分支语句
  - 三元运算符
  - switch 语句

2.2.1 if语句

目标：能使用if语句执行满足条件的代码

1. if语句

- if语句有三种使用：单分支、双分支、多分支
- 单分支使用语法：

```
if (条件) {
    满足条件要执行的代码
}
```

  - 括号内的条件为true时，进入大括号里执行代码
  - 小括号内的结果若不是布尔类型时，会发生隐式转换转为布尔类型

2.2.1 if语句

目标：能使用if语句执行满足条件的代码

- 单分支课堂案例1：用户输入高考成绩，如果分数大于700，则提示恭喜考入黑马程序员

2.2.1 if语句

目标：能使用if语句执行满足条件的代码

- 双分支if语法：

```
if (条件) {
    满足条件要执行的代码
} else {
    不满足条件执行的代码
}
```

2.2.1 if语句

目标：能使用if语句执行满足条件的代码

- 双分支课堂案例1：用户输入，如果工龄大于1年，年底奖金+2000， 否则年底没奖金

- 双分支课堂案例2：让用户输入年份，判断这一年是闰年还是平年并输出
  - 能被4整除但不能被100整除，或者被400整除的年份是闰年，否则都是平年
  - 需要逻辑运算符

2.2.1 if语句

目标：能使用if语句执行满足条件的代码

- 多分支if语法：

```
if (条件1) {

    代码1
} else if (条件2) {

    代码2
} else if (条件3) {

    代码3
} else {

    代码n
}
```

释义：

- 先判断条件1，若满足条件1就执行代码1，其他不执行
- 若不满足则向下判断条件2，满足条件2执行代码2，其他不执行
- 若依然不满足继续往下判断，依次类推
- 若以上条件都不满足，执行else里的代码n
- 注：可以写N个条件，但这里演示只写2个

2.2.1 if语句

目标：能使用if语句执行满足条件的代码

- 多分支if课堂案例：根据输入不同时间，输出不同的问候语
- 注：
  - 12点以前， 输出上午好
  - 18点以前， 输出下午好
  - 20点以前， 输出晚上好

2. 分支语句

- 分支语句可以让我们有选择性的执行想要的代码
- 分支语句包含：
  - If分支语句
  - 三元运算符
  - switch 语句

2.2.2 三元运算符

目标：能利用三元运算符执行满足条件的语句

- 其实是比 if 双分支 更简单的写法，有时候也叫做三元表达式
- 符号：？与：配合使用
- 语法：

```
条件 ？满足条件执行的代码 ：不满足条件执行的代码
```

- 一般用来取值

案例

判断2个数的最大值

需求：用户输入2个数，控制台输出最大的值

分析：

①：用户输入2个数

②：利用三元运算符输出最大值

案例

数字补0案例

需求：用户输入1个数，如果数字小于10，则前面进行补0， 比如 09　03 等

分析：

①：为后期页面显示时间做铺垫

②：利用三元运算符 补 0 计算

2. 分支语句

- 分支语句可以让我们有选择性的执行想要的代码
- 分支语句包含：
  - If分支语句
  - 三元运算符
  - switch 语句

2.2.3 switch语句

目标：能利用switch执行满足条件的语句

```
switch (数据) {
    case 值1:
        代码1
        break
    case 值2:
        代码2
        break
    default:
        代码n
        break
}
```

释义：

- 找到跟小括号里数据**全等**的case值，并执行里面对应的代码
- 若没有全等 === 的则执行default里的代码
- 例：数据若跟值2全等，则执行代码2

注意事项

1. switch case语句一般用于等值判断,不适合于区间判断
2. switch case一般需要配合break关键字使用 没有break会造成case穿透

案例

简单计算器

需求：用户输入2个数字，然后输入 + - * / 任何一个，可以计算结果

案例

简单计算器

需求：用户输入2个数字，然后输入 + - * / 任何一个，可以计算结果

分析：

①：用户输入数字

②：用户输入不同算术运算符，可以去执行不同的运算 (switch)

2.2 分支语句

目标：掌握流程控制，写出能“思考”的程序

1. 程序三大流程控制
   - 顺序
   - 分支
   - 循环
2. if语句
   - 三种形式
3. switch语句
   - 全等判断
   - break：结束switch语句，防止穿透
4. 三元运算符
   - 也是双分支
   - 一般用来取值

语句

- 表达式和语句
- 分支语句
- 循环语句

2.3 循环结构

目标：掌握循环结构，实现一段代码重复执行

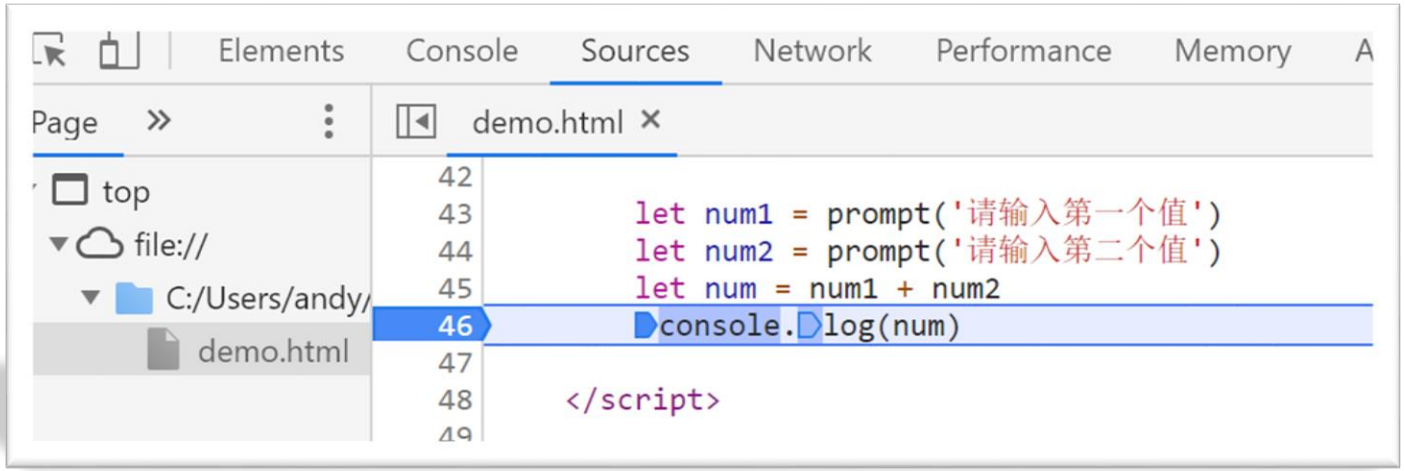
学习路径：

1. 断点调试
2. while循环

2.3.1 断点调试

目标： 掌握断点调试方法，学会通过调试检查代码

- 作用：学习时可以帮助更好的理解代码运行，工作时可以更快找到bug
- 浏览器打开调试界面
  1. 按F12打开开发者工具
  2. 点到sources一栏
  3. 选择代码文件
- 断点：在某句代码上加的标记就叫断点，当程序执行到这句有标记的代码时会暂停下来



2.3 循环结构

目标：掌握循环结构，实现一段代码重复执行

学习路径：

1. 断点调试
2. while循环

2.3.2 while 循环

目标：掌握while循环语法，能重复执行某段代码

循环：重复执行某段代码， 而 while ： 在.... 期间

1. while 循环语法：

```
while (循环条件) {
    要重复执行的代码(循环体)
}
```

释义：

- 跟if语句很像，都要满足小括号里的条件为true才会进入执行代码
- while大括号里代码执行完毕后不会跳出，而是继续回到小括号里判断条件是否满足，若满足又执行大括号里的代码，然后再回到小括号判断条件，直到括号内条件不满足，即跳出

2.3.2 while 循环

目标：掌握while循环语法，能重复执行某段代码

循环：重复执行某段代码， 而 while ： 在.... 期间

2. while 循环注意事项：

循环的本质就是以某个变量为起始值，然后不断产生变化量，慢慢靠近终止条件的过程。

所以，循环需要具备三要素：

1. 变量起始值
2. 终止条件（没有终止条件，循环会一直执行，造成死循环）
3. 变量变化量（用自增或者自减）

```
let i = 1
while (i <= 3) {
    document.write('我会循环三次<br>')
    i++
}
```

2.3.2 while 循环

案例

在页面中打印输出10句“月薪过万”

需求：使用while循环，页面中打印，可以添加换行效果

能不能改进，让用户输入打印输出的个数呢？

案例

While 练习

需求：使用while循环，页面中打印，可以添加换行效果

1. 页面输出1-100

- 核心思路： 利用 i ,因为正好和 数字对应

2. 计算从1加到100的总和并输出

- 核心思路：
  - 声明累加和的变量 sum
  - 每次把 i 加到 sum 里面

3. 计算1-100之间的所有偶数和

- 核心思路：
  - 声明累加和的变量 sum
  - 首先利用if语句把 i 里面是偶数筛选出来
  - 把 筛选的 i 加到 sum 里面

2.3 循环退出

目标： 能说出continue和break的区别

- 循环结束：
  - continue：结束本次循环，继续下次循环

2.3 循环退出

目标： 能说出continue和break的区别

- 循环结束：
  - continue：结束本次循环，继续下次循环
  - break：跳出所在的循环

案例

页面弹框

需求：页面弹出对话框，‘你爱我吗’，如果输入‘爱’，则结束，否则一直弹出对话框

分析：

①：循环条件永远为真，一直弹出对话框

②：循环的时候，重新让用户输入

③：如果用户输入的是： 爱，则退出循环 （break）

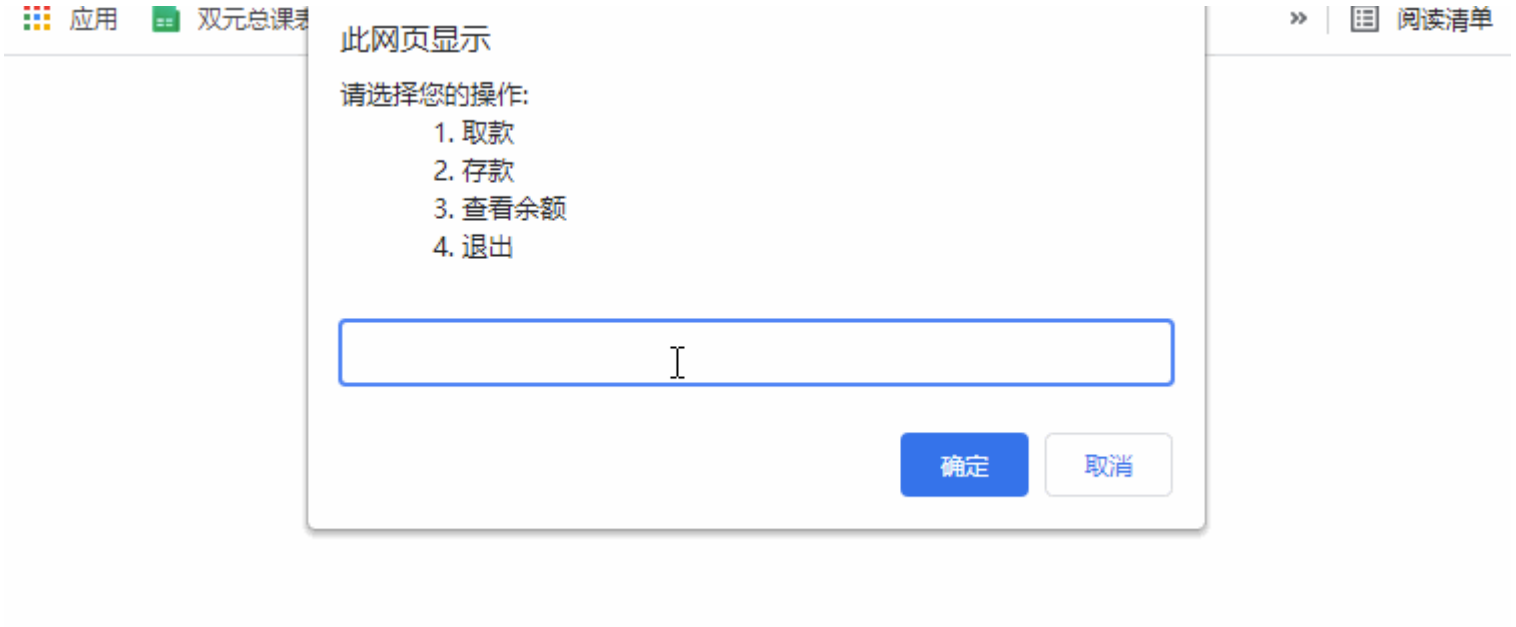
目录

Contents

案例

简易ATM取款机案例

需求：用户可以选择存钱、取钱、查看余额和退出功能



案例

简易ATM取款机案例

需求：用户可以选择存钱、取钱、查看余额和退出功能

分析：

①：循环的时候，需要反复提示输入框，所以提示框写到循环里面

②：退出的条件是用户输入了 4，如果是4，则结束循环，不在弹窗

③：提前准备一个金额预先存储一个数额

④：取钱则是减法操作， 存钱则是加法操作，查看余额则是直接显示金额

⑤：输入不同的值，可以使用switch来执行不同的操作

今日复习路线

1. 晚自习回来每个同学先必须xmind梳理今日知识点 (md 笔记也行)
2. 需要把今天的所有案例，按照书写顺序写一遍。
3. 独立书写今日作业
4. 明天上午复习今天内容， 下午预习后天的内容（for循环 + 数组）
5. 每日一句鼓励自己的话：

1. 战歌响起来~~~　　明月天涯.mp3

纽约时间比加州时间早三个小时，
但加州时间并没有变慢。
有人22岁就毕业了，
但等了五年才找到好的工作；
有人25岁就当上CEO，
却在50岁去世；
也有人迟到50岁才当上CEO，
然后活到90岁。

有人依然单身，
同时也有人已婚。
奥巴马55岁就退休，
川普70岁才开始当总统。
世上每个人本来就有自己的发展时区，
身边有些人看似走在你前面，
也有人看似走在你后面，
但其实每个人在自己的时区有自己的步程。

不用嫉妒或嘲笑他们，
他们都在自己的时区里，
你也是。

生命就是等待正确的行动时机，
所以，放轻松，
你没有落后，你没有领先，
在命运为你安排的属于自己的时区里，
一切都准时。

传智教育旗下高端IT教育品牌