[笔记][LIKE-JS][2-JavaScript语法篇-流程控制]

JavaScript

---

023. JS基础-流程控制介绍（掌握）

顺序结构、选择结构、循环结构

几种流程控制语句：

- `if`
- `switch`
- `while`
- `dowhile`
- `for`

---

024. JS基础-流程控制-if语句基本使用(掌握)

```
if(条件表达式1)
{
    语句块1;
}else if(条件表达式2){
    语句块2;
}else if(条件表达式3){
    语句块3;
}else{
    语句块4;
}
```

> `Python` 里面用冒号 `:` 来代替花括号 `{}`。

---

025. JS基础-流程控制-错误纠正(掌握)

---

026. JS基础-流程控制-if语句注意事项(掌握)

注意点

- 如果只有一条语句时if后面的大括号可以省略

```
if(age > 18)
    console.log("成年人");
```

- 开发中尽量不要省略大括号
- 分号 `;` 也是一条语句, 空语句
- `if else` 是一个整体， `else` 匹配 `if` 的时候匹配离它最近的一个 `if`
- 对于非 `Boolean` 类型的值,会先转换为 `Boolean` 类型后再判断
- `if` 语句可以嵌套使用
- `if (0 == a)` 把常量写在前面，如果少写 `=`，会报错，因为如果是赋值，左边一定是变量。

---

027. JS基础-流程控制-if语句-练习-1(掌握)

补充: `prompt` 与 `window.prompt` 有什么区别?

- `alert()` 弹出提示框（确定）
- `confirm()` 弹出确认框（确定、取消）
- `prompt()` 弹出输入框（可输入的文本框）

警告框、确认框、提示框是 `window` 对象的接口方法。
由于 `window` 对象位于对象层次的顶层，因此实际应用中不必使用这些消息的全名（例如 `window.alert()`）。
不过采用全名是一个好主意，这样有助于记住这些消息框属于哪个对象。

```
<script>
    // 从键盘输入一个整数，判断是否为偶数，如果是输出 YES，如果不是输出 NO
    // 1. 定义一个变量
    var tmp;

    // 2. 接收用户输入的整数
    // 2.1 把字符串转成数值
    tmp = +(window.prompt('请输入一个整数'));
```

```
        // 3. 合法性检验
        if (isNaN(tmp)) {
          alert('输入的内容有误');
        } else {
            // 4. 判断用户输入的整数是否是偶数
            tmp % 2 === 0 ? window.alert('YES') : window.aler
t('NO');
        }
</script>
```

---

028. JS基础-流程控制-if语句-练习-2(掌握)

略

---

029. JS基础-流程控制-if语句-练习-3(掌握)

```
console.log(num1 > num2 ? num1 : num2);
```

> 简单的用三目，复杂的用 `if else`

---

030. JS基础-流程控制-if语句-练习-4(掌握)

选择排序

```
<script>
```

```
    var a, b, c;

    a = Number(window.prompt('请输入数字a'))
    b = Number(window.prompt('请输入数字b'))
    c = +(window.prompt('请输入数字c'))

    var tmp;
    if (a > b) {
        tmp = a;
        a = b;
        b = tmp;
    }

    if (a > c) {
        tmp = c;
        a = c;
        c = tmp;
    }

    if (b > c) {
        tmp = b;
        b = c;
        c = tmp;
    }

    console.log(a, b, c)

</script>
```

冒泡排序

```
<script>
    var a, b, c;
    a = +(window.prompt('请输入第一个数'));
    b = +(window.prompt('请输入第一个数'));
    c = +(window.prompt('请输入第一个数'));

    var tmp;
    if (a > b) {
        tmp = a;
```

```
            a = b;
            b = tmp;
        }

        if (b > c) {
            tmp = b;
            b = c;
            c = tmp;
        }

        if (a > b) {
            tmp = a;
            a = b;
            b = tmp;
        }

    </script>
```

031. JS基础-流程控制-if语句-练习-5(掌握)

```
    <script>
        // 剪刀石头布
        // 0-剪刀，1-石头，2-布

        // 1. 定义变量，玩家和电脑
        var player, computer;

        // 2. 电脑出拳
        computer = Math.floor(Math.random() * 3);

        // 3. 玩家出拳
        player = +(window.prompt('请出拳：0.剪刀 1.石头 2.布'));

        // 4. 判断比较
        if (player < 0 || player > 2) {
            window.prompt('请重新出拳');
```

```
    } else {
        if (
            player === 0 && computer === 2 ||
            player === 1 && computer === 0 ||
            player === 2 && computer === 1
        ) {
            alert('恭喜您，赢了！');
        } else if (
            computer === 0 && player === 2 ||
            computer === 1 && player === 0 ||
            computer === 2 && player === 1
        ) {
            alert('很遗憾，你输了！');
        } else {
            alert('平局！');
        }
    }
</script>
```

032. JS基础-流程控制-switch-上(掌握)

格式

```
switch (条件表达式) {
    case 表达式:
        语句1;
        break;
    case 表达式:
        语句2;
        break;
    ...
    default:
        语句n+1;
        break;
}
```

计算条件表达式，结果与 `case` 后面相比较，如果全等，那么进入分支。

注意事项

- `case` 全等问题，即判断 `===`，判断类型和值都相等。
- `case` 后面可以是常量，也可以是变量。
- `case` 后面是表达式，会先计算表达式的值，再进行判断
- `case` 只要匹配一次，其它就会失效，前提是要加上 `break`，如果没有加，则形成穿透。
- `default` 可以省略，可以放到任意位置，但是推荐放到最后

---

033. JS基础-流程控制-if和switch的区别(掌握)

```
<script>
    var score;
    score = +(window.prompt('请输入0-100之间的分数\n'));
    var level;
    switch (level = Math.floor(score / 10)) {
        case 10:
        case 9:
        case 8:
            alert('厉害了');
            break;
        default:
            alert('渣渣');
    }
</script>
```

> 注意上面的 `case` 叠写。

什么时候用 `switch`

- 分支不多，出口固定
- `switch` 效率比 `if` 高

034. JS基础-流程控制-switch练习题(掌握)

略

035. JS基础-流程控制-while(掌握)

格式

```
while (条件表达式) {
    语句块;
}
```

注意

- 如果条件一开始都不成立，那么循环体不会被执行
- 小心死循环
- 任何值都有真假性
- 如果循环体只有一条语句，可以不用大括号（不建议）
- `while (条件);` 直接加分号，那么后面大括号不会被执行。

036. JS基础-流程控制-while练习题(掌握)

略

037. JS基础-流程控制-break-countinue(掌握)

`break`：跳出当前循环

- 用于循环和 `switch`，其它地方没用
- 一个 `break` 只跳出一层
- `break` 后面的语句不会被执行

`continue`：结束本次循环，进入到下一次循环

- 只能用于循环
- `continue` 后面的不会被执行

```
do {
    循环体;
} while (条件表达式);
```

注意后面的分号。

---

038. JS基础-流程控制-for循环(掌握)

```
for(初始化表达式; 循环条件表达式; 循环后操作表达式) {
    循环体;
}
```

1. 初始化表达式（只执行一次）
2. 循环条件表达式
3. 循环体
4. 循环后操作表达式
5. 循环条件表达式

   ...

两种死循环：

- `for(;;){}`
- `while(1){}`

`for` 里面都可以省略，循环条件表达式默认为真

```
<script>
    for (var i = 1; i < 3; i++) {
        console.log('好友列表' + i);
        for (var j = 1; j <= 3; j++) {
            console.log('    好友' + j);
        }
    }
</script>
```

---

039. JS基础-流程控制-for循环-练习1(掌握)

`window.document.write();` 在界面上输出
`window.document.write('<br/>');` 换行

---

040. JS基础-流程控制-for循环-练习2(掌握)

> 小技巧：使用 `Ctrl + Alt + L` 规范代码

完成于 `20190225`