WunschScript

基于PPT实现的程序设计语言

PART 1 数据类型

整数、浮点数

整数:内部存储为64位有符号数,支持十进制、八进制、十六进制字面量。

浮点数:内部存储为双精 度浮点数。 100 0173 0×4f5d

32.54

6.34e - 3

字符串

• 被双引号包围,以UTF-8 编码表示的字符序列。

"hello, world"

布尔量

• 仅可存储"真"和"假"两种状态的类型。

true false

函数

• 接收一定数量的命名参数的语句块。

- 可以返回任意值。
- 若无显式返回语句,默认返回空值nil。

```
(x, y) ⇒ {
    return 2 * x + y - 1;
}
```

列表

- 由中括号包围,逗号分隔的表达式列表。
- 列表中元素的类型不受限制,可以嵌套。

```
[
    12,
    34.6,
    "this is a list",
    [],
    [12, 34, 56]
]
```

字典

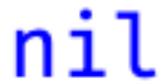
- 由大括号包围,逗号分隔的键值对列表。
- 键和值之间由冒号隔开
- 值的类型不受限制。

```
x:3,
    y:5,
    setX : (x) \Rightarrow \{
         this.x = x;
         return x;
    },
    setY : (y) \Rightarrow \{
         this.y = y;
         return nil;
};
```

空类型

• 不存储任何内容的类型。

• 其字面量为nil。



PART 2 表达式

字面量表达式

• 各种类型数据的字面量。

```
"hello";
    "this is a list",
    [],
    [12, 34, 56]
];
64.64e-3;
(x, y) \Rightarrow \{
    return 2 * x + y - 1;
};
nil;
{
    hello : ["world", "!"]
};
```

变量索引表达式

- 可以用变量名索引一个已经定义的变量。
- 可以作为左值。

元素引表达式

- 点运算符用于根据键获取 字典内的值。
- 方括号运算符可以获取数组、字典中的值。当获取字典元素时,必须传入键对应的字符串。
- 均可作为左值。

```
dictA.x;
dictA["x"];
```

arr1[10];

函数调用表达式

在函数表达式后,传入用 括号包围的表达式列表作 为参数。

```
print("hello, world", "!");
dictA.setX(13);
```

运算符

| 运算符 | 含义 | 优先级 | 结合 性 | 运算符 | 含义 | 优先级 | 结合性 |
|---------|-------|-----|---------|----------------|------|-----|-----|
| ! | 逻辑非 | 11 | 左 | = ≠ | 比较 | 5 | 左 |
| ~ | 按位取反 | 10 | 左 | ક | 按位与 | 4 | 左 |
| + - | 正负 | 9 | 左 | ^ | 按位异或 | 3 | 左 |
| * / % | 乘除、取余 | 8 | 左 | | 按位或 | 2 | 左 |
| + - | 加减 | 7 | 左 | 8 6 | 逻辑与 | 1 | 左 |
| < \ > > | 比较 | 6 | 左 | | 逻辑或 | 0 | 左 |

PART 3 语句

表达式语句

- 表达式作为一条语句。
- 求值结果将被丢弃。
- 需要以分号结尾。

```
(x + 5 * 3) % 6;
somefunc();
```

变量定义

- 用var关键字定义变量。
- 可以赋予初始值。
- 默认初始化为nil。
- 需要以分号结尾。

```
var x = 3.65;
var y; # y = nil
```

变量赋值

- 用等号将一个表达式赋给一个已定义的变量。
- 不要求类型相同。
- 赋值语句不是表达式!
- 需要以分号结尾。

```
var x = 3.65;
x = (n) ⇒ {
    return n * n;
};
```

条件语句

- if后接表达式作为条件。
- 其后的语句块当条件成真时执行。
- 可以接else块。
- 当条件求值后不是布尔量 时,会引发错误。

```
if 3 * 4 == 12 {
    print("Of course");
} else {
    print("Oho!!!")
}
```

while循环语句

- while后接一个表达式作 为条件。
- 其后的语句块当条件成真时不断执行。
- 当条件求值后不是布尔量时,会引发错误。

```
var i = 0;
while i < 4 {
    print("I'm in the loop!");
    i = i + 1;
}</pre>
```

for循环语句

- 用于遍历列表或字典。
- 遍历列表时,迭代列表中的值。
- 遍历字典时,迭代字典中 的键。

```
var arr = [1, 2, 3];
var sum = 0;
for elem in arr {
    sum = sum + elem;
    # `sum` will be 6.
}
var dictA = { x : "y" };
for key in dictA {
    print(key);
    # will print "x"
    print(dictA[key]);
    # will print "y"
}
```

return语句

• 立即中断函数的执行。

• 返回其后的表达式。

```
var add = (x, y) ⇒ {
    return x + y;
};
```

没有啦