函数

Ref Javascript

Parameters & return

函数是一种可重复使用的代码块。函数声明为其命名，您可以使用它来调用该函数：

```
function greet(name) {
  return `Hello, ${name}!`;
}

console.log(greet("Mei"));   // Hello, Mei!
console.log(greet("Tom"));   // Hello, Tom!
```

- name 是参数：一个变量，在函数被调用时接收值。
- "Mei" 是实参：传入的实际值。
- return 向调用者返回值并立即结束函数。

普通函数如果没有返回值而执行到末尾，则返回 undefined。因此请**返回**答案，让调用者决定是否打印它：

```
function area(width, height) {
  return width * height;
}

const a = area(4, 3);
console.log(a + 1);          // 13  (a returned value can be used again)
```

Arrow functions

箭头函数是编写函数的简写方式。将其存储在 const 中：

```
const square = (n) => n * n;
const add = (a, b) => a + b;

console.log(square(5));      // 25
console.log(add(2, 3));      // 5
```

- 当 => 后面只有一个表达式时，其值会被直接返回，无需书写 return。
- 使用花括号时，需要再次使用 return：

```
const grade = (mark) => {
  if (mark >= 50) {
    return "pass";
  }
  return "fail";
};
console.log(grade(64));      // pass
```

箭头函数通常作为参数传递给其他函数，这将在主题 7 中的 `map` 和 `filter` 中看到。

Scope

变量的作用域是指能够访问该变量的代码部分。在 `{ }` 中使用 `let` 或 `const` 声明的变量可在该块及其嵌套作用域中访问。即使外层函数调用结束后，嵌套函数仍可保持对该变量的访问权：

```
const level = "global";

function show() {
  const inside = "local";
  console.log(level);        // global  (outer variables are visible)
  console.log(inside);       // local
}

show();
console.log(typeof inside);  // undefined  (inside does not exist out
↪  here)
```

- 在函数内部声明的变量是局部变量；每次调用都会获得一个新的实例。
- 通过参数**传入**值，并通过 `return` **返回**结果，而不是在函数内部修改外部变量。

常见错误

- 忘记 `return`：函数计算出答案后仍会返回 `undefined`。
- 不带括号调用：`greet` 表示函数本身；`greet("Mei")` 用于执行它。
- 在函数内部打印**并**打印其返回值会导致打印两次，或者仅打印 `undefined`。