Understanding Basic Syntax and Data Types

Brice Wilson

Overview

- Basic types and variable declarations
- Type annotations and type inference
- Creating union types
- Storing data in arrays
- Controlling program flow

Basic TypeScript Types

Boolean

Number

String

Array

Enum

Declarations with *let* and *const*

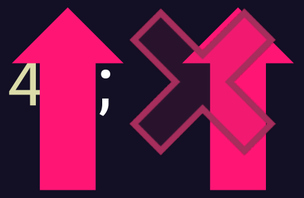
```
console.log(someString);
var someString = 'Hello World';
```

```
let someString = 'Hello World';
console.log(someString);
```

```
const someString = 'Hello World';
console.log(someString);
```

Type Annotations and Type Inference

```
let x: string = 'I will forever be a string.';
x = 4;
```



```
let y = 'I will also forever be a string.';
y = 2;
```

```
let z = GetSomeValue();
let z: number = GetSomeValue();
```

Using *let* and *const* with type annotations

Additional Built-in Types

Void

Null

Undefined

Never

Any

Union Types

```
let someValue: number | string;
```

```
someValue = 42;
```

```
someValue = 'Hello World';
```

```
someValue = true;
```

Using the *--strictNullChecks* Compiler Option

```
let basicString: string;
basicString = null;
basicString = undefined;

let nullableString: string | null;
nullableString = null;
nullableString = undefined;

let mysteryString: string | null | undefined;
mysteryString = null;
mysteryString = undefined;
```







Arrays

Can be declared two different ways

Accessed and used much like JavaScript arrays

Declare as an array of “any” to store any type in the same array

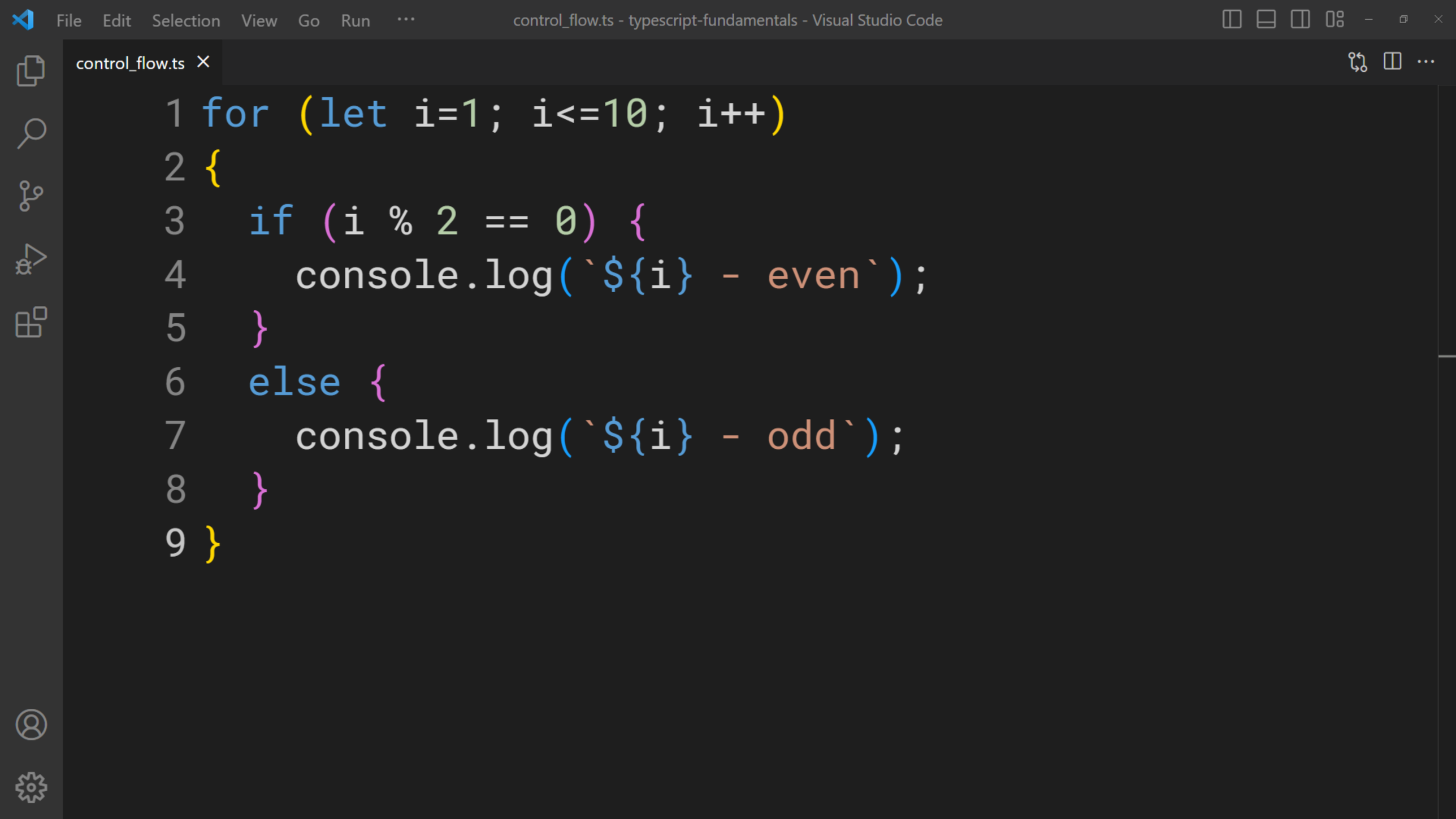
Controlling Program Flow

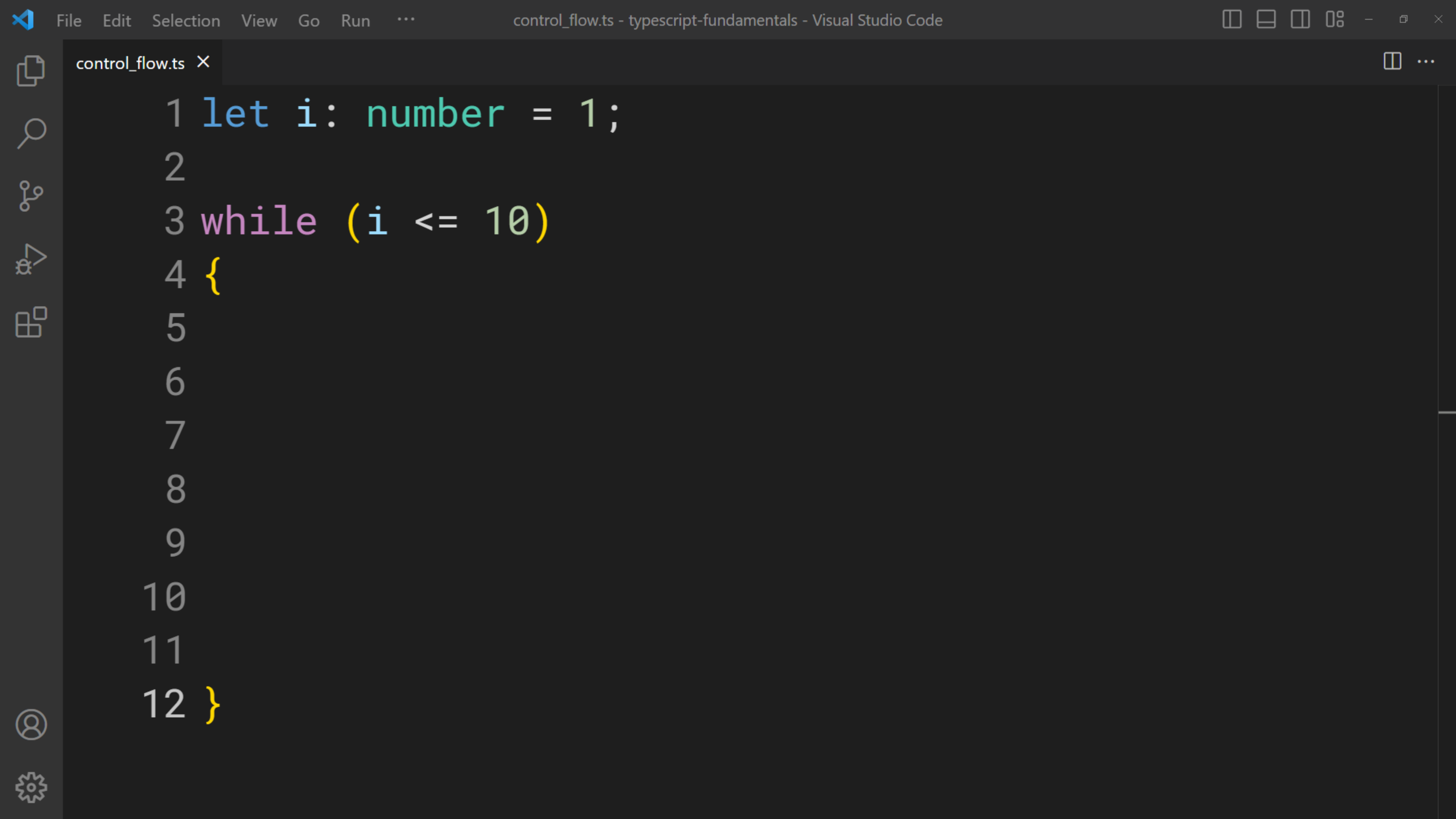
control_flow.ts

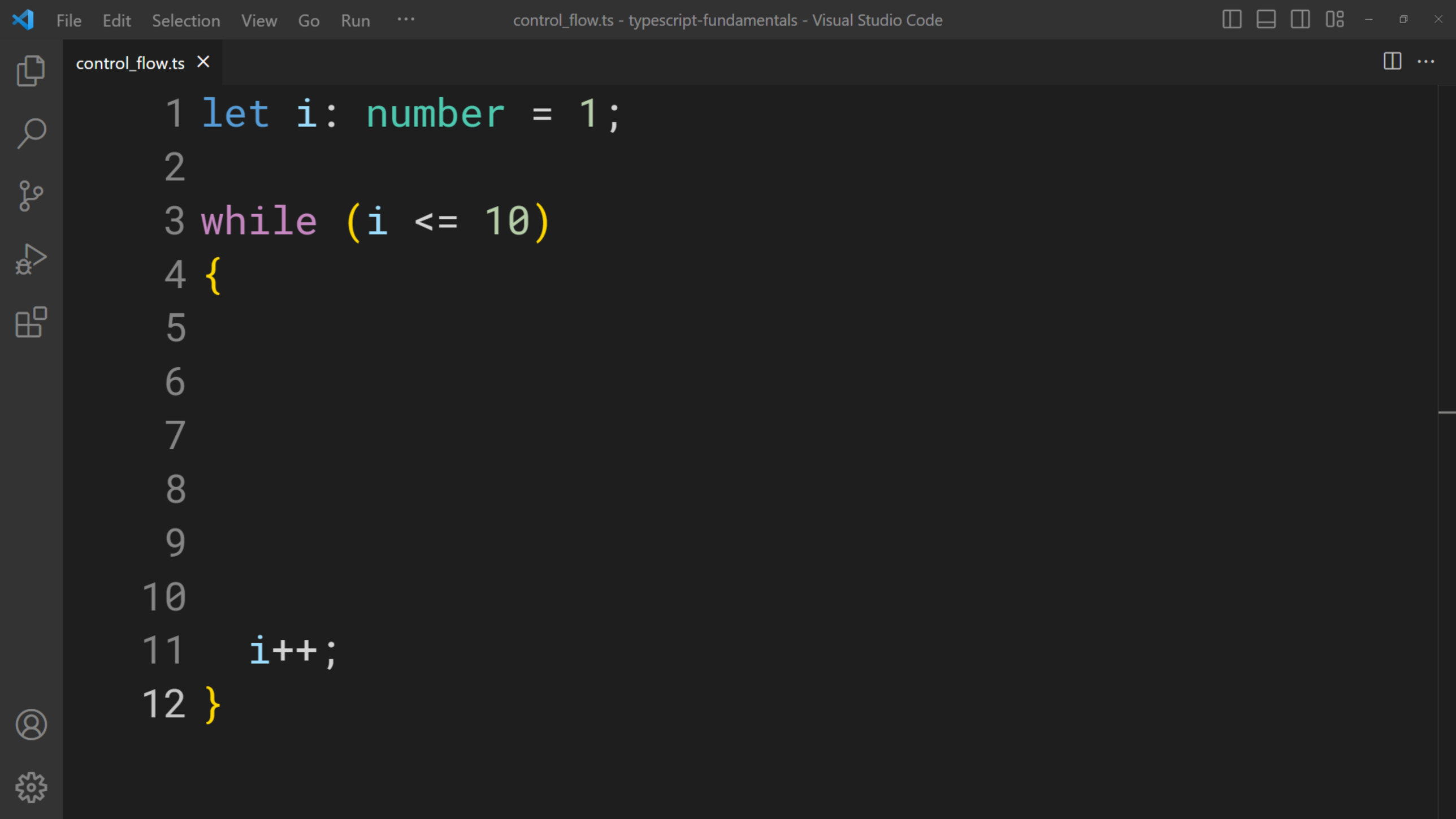
```
1 for (let i=1; i<=10; i++)
2 {
3
4
5
6
7
8
9 }
```

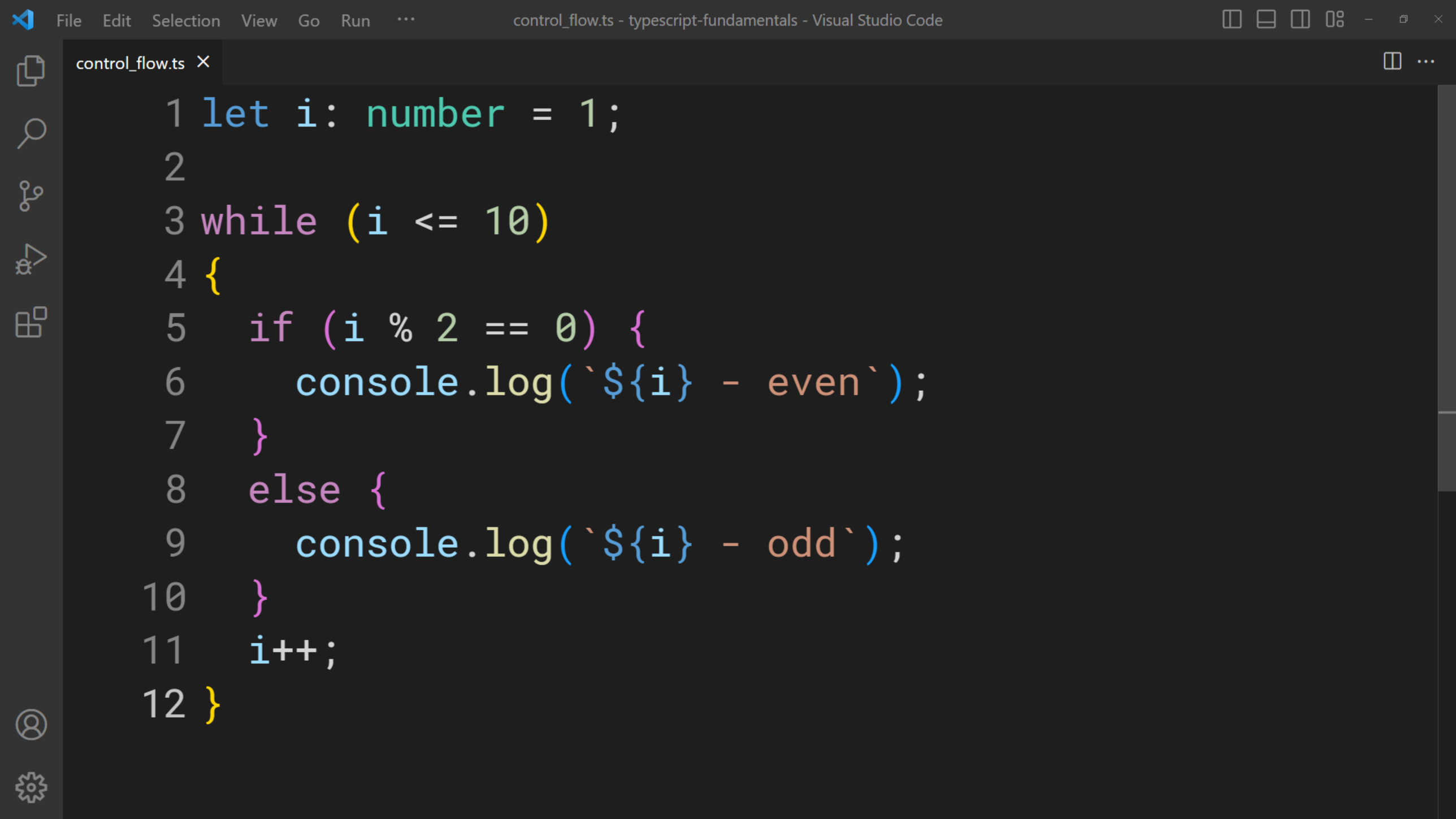
control_flow.ts

```
1 for (let i=1; i<=10; i++)
2 {
3   if (i % 2 == 0) {
4     console.log(`${i} - even`);
5   }
6
7
8
9 }
```

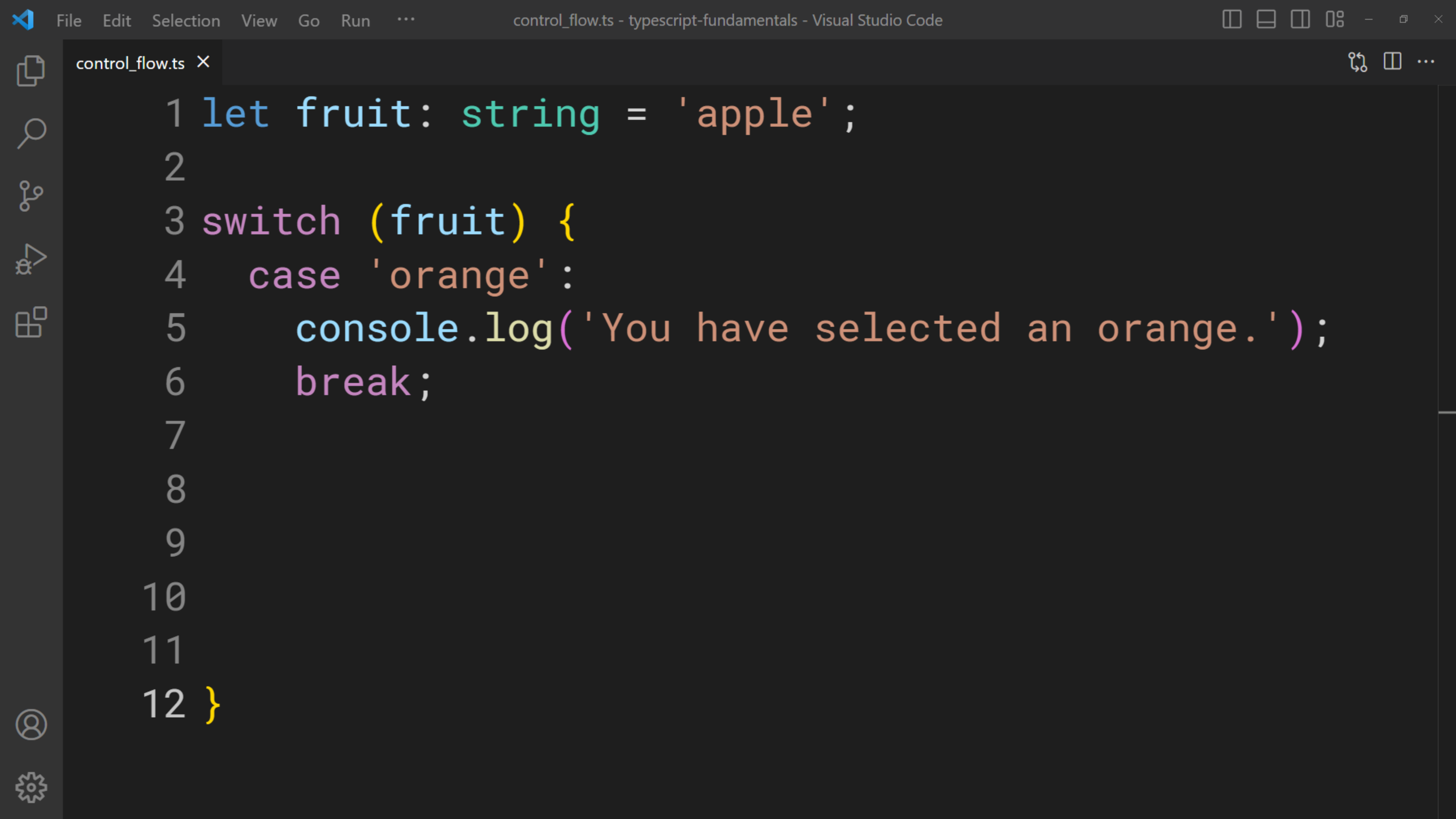

File Edit Selection View Go Run ···
control_flow.ts - typescript-fundamentals - Visual Studio Code
control_flow.ts
1 for (let i=1; i<=10; i++)
2 {
3 if (i % 2 == 0) {
4 console.log(`${i} - even`);
5 }
6 else {
7 console.log(`${i} - odd`);
8 }
9 }

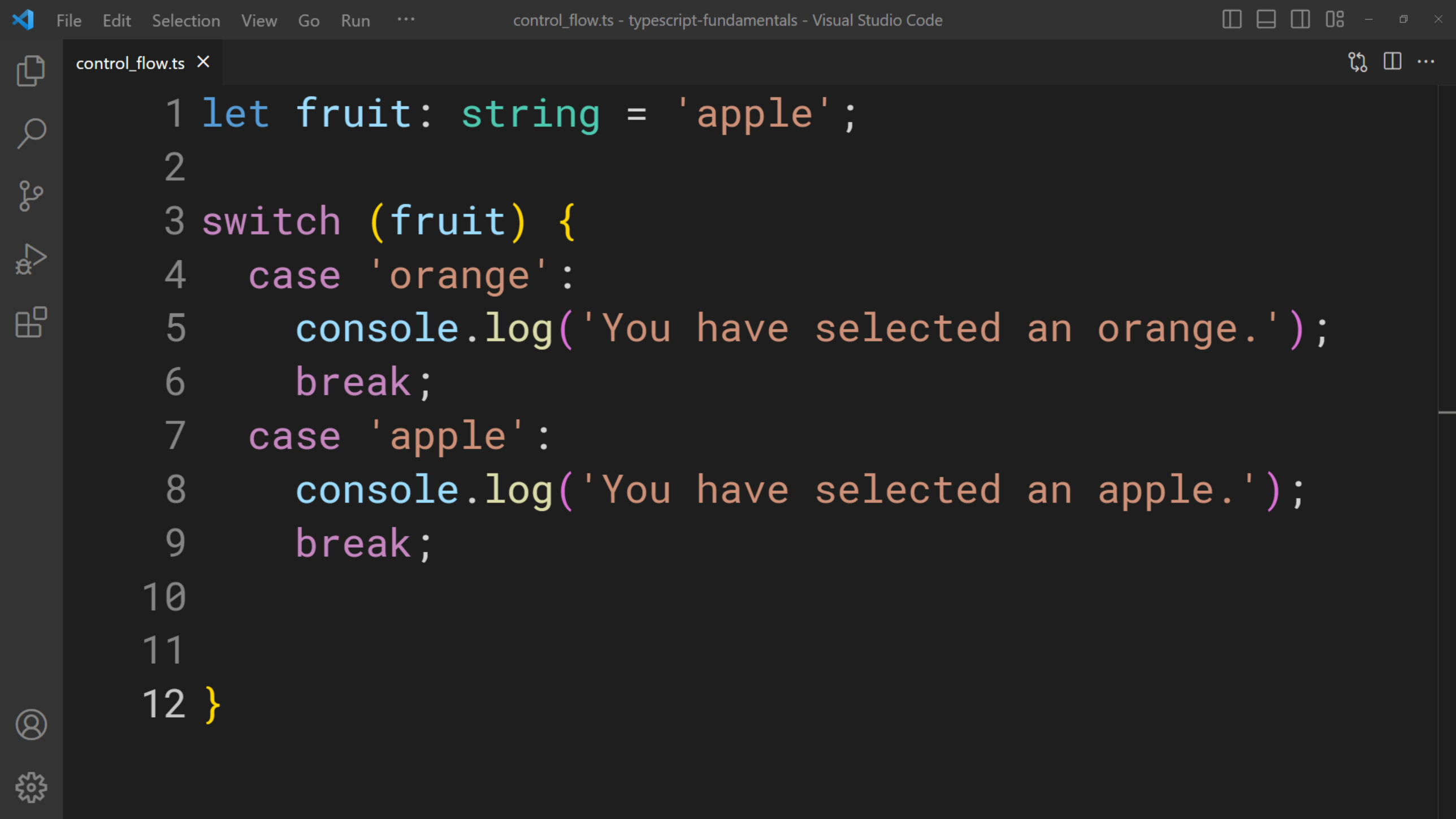

File Edit Selection View Go Run ···
control_flow.ts - typescript-fundamentals - Visual Studio Code
control_flow.ts
let i: number = 1;
while (i <= 10)
{
}


File Edit Selection View Go Run ···
control_flow.ts - typescript-fundamentals - Visual Studio Code
control_flow.ts
let i: number = 1;
while (i <= 10)
{
i++;
}


File Edit Selection View Go Run ···
control_flow.ts - typescript-fundamentals - Visual Studio Code
control_flow.ts
let i: number = 1;
while (i <= 10)
{
if (i % 2 == 0) {
console.log(`${i} - even`);
}
else {
console.log(`${i} - odd`);
}
i++;
}


File Edit Selection View Go Run ···
control_flow.ts - typescript-fundamentals - Visual Studio Code
control_flow.ts
let fruit: string = 'apple';
switch (fruit) {
}


File Edit Selection View Go Run ···
control_flow.ts - typescript-fundamentals - Visual Studio Code
control_flow.ts
let fruit: string = 'apple';
switch (fruit) {
case 'orange':
console.log('You have selected an orange.');
break;
}


File Edit Selection View Go Run ···
control_flow.ts - typescript-fundamentals - Visual Studio Code
control_flow.ts
let fruit: string = 'apple';
switch (fruit) {
case 'orange':
console.log('You have selected an orange.');
break;
case 'apple':
console.log('You have selected an apple.');
break;
}

control_flow.ts

```typescript
let fruit: string = 'apple';

switch (fruit) {
  case 'orange':
    console.log('You have selected an orange.');
    break;
  case 'apple':
    console.log('You have selected an apple.');
    break;
  default:
    console.log('You selected a different fruit');
}
```

Control flow type analysis

Summary

- Arrays
- Basic data types
- Program flow
- Type annotations
- Type inference
- Union types

Up Next:

Creating and Calling Functions