Controlling Program Flow with Selection Statements

Alex Wolf
Software Developer

www.crywolfcode.com

Selection Statements

Statements that execute different branches of code based on specific conditions.

The Types of Selection Statements

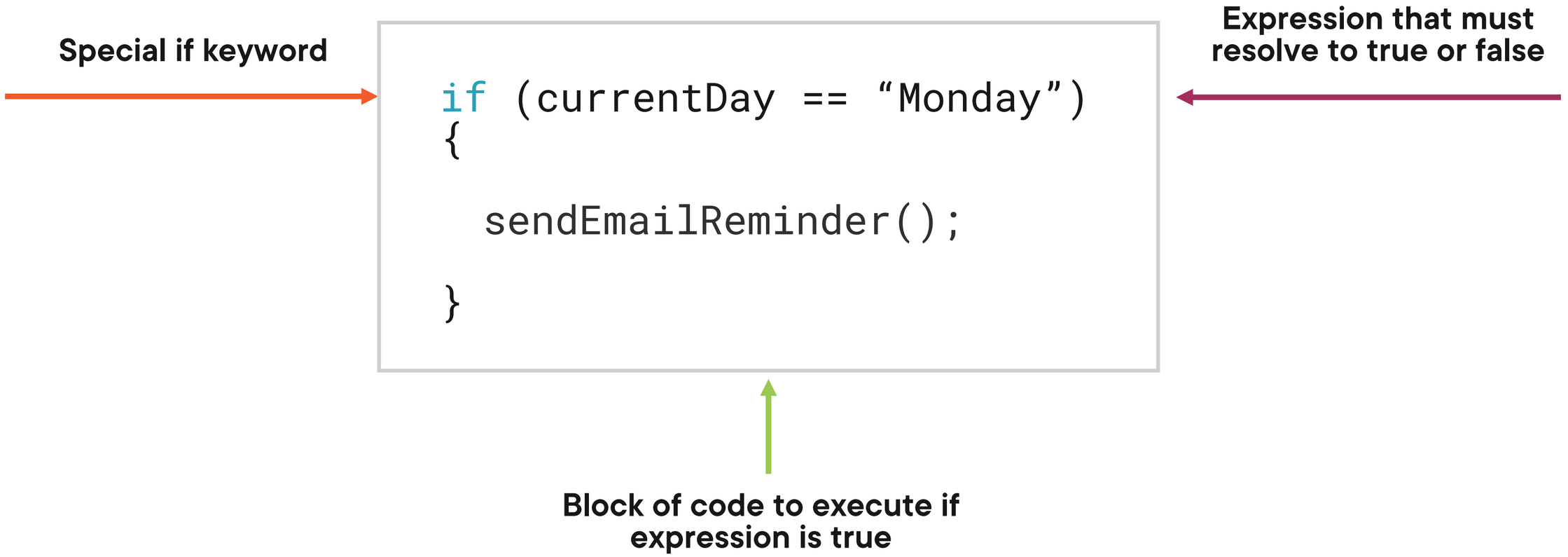
if-else statements

Select blocks of code to run based on boolean logic

switch statements

Select a set of code to run based on pattern matching

Understanding If Statements



```
if (priceScore > serviceScore)
{
    // Do something
}
```

◀ Execute code based on a comparison

```
if (productCategory == “Coffee”)
{
    // Do something
}
```

◀ Execute code based on an equality check

```
if ((responseCount / surveyCount) > .5)
{
    // Do something
}
```

◀ Execute code based on a calculated value and comparison

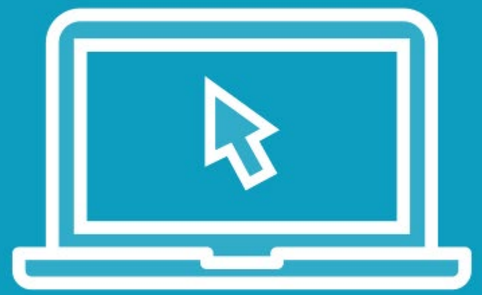
```
if (productCategory == “Coffee”)
{
    // Do something
}
else
{
    // Otherwise do something else
}
```

- Only execute this block of code if the product category is “Coffee”
- Otherwise just execute this block instead

```
if (productCategory == “Coffee”)
{
    // Do something
}
else if (productCategory == “Food”)
{
    // Do something
}
else
{
    // Do something else
}
```

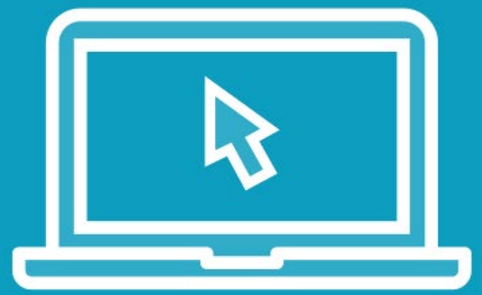
- If the product category is coffee, execute this block of code
- If the product category is not coffee, but it is Food, then execute this block of code
- If the product category is not coffee or food, then execute this block of code

Demo



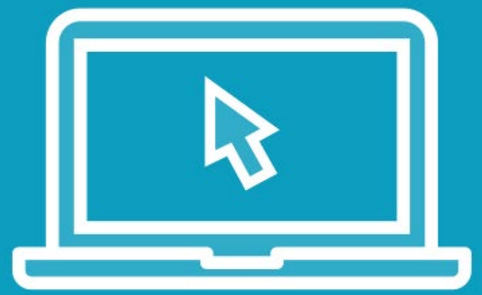
Creating a Simple Selection Statement

Demo



Creating a Branching Selection Statement

Demo



Complex Decisions using Selection Statements

Exploring Switch Statements

Switch Statements

A selection statement that selects a block of code to execute based on pattern matching.

```
switch (productCategory)
{
      case “Coffee”:
      Console.Writeline(“The Coffee category”);
      // Do other stuff
      break;

      case “Food”:
      Console.Writeline(“The Food category”);
      break;

      case “Merchandise”:
      Console.Writeline(“The Merch category”);
      break;
}
```

- The switch keyword and an expression to match against

- One of several cases to check
- The code to execute if category is “Food”
- Breaks out of the switch statement

Switch Statements vs If-Else Statements

Program.cs

```
switch (leastFavoriteProduct)
{
    case “Granola”:
    // Your logic
    break;
    case “Fruit”:
    // Your logic
    break;
}
```

Program.cs

```
if (leastFavoriteProduct == “Granola”)
{
    // Your logic
}
else if (leastFavoriteProduct == “Fruit”)
{
    // Your logic
}
```

```
switch (dayOfWeek)
{
      case “Monday”:
      // Run the Monday report
      break;

      case “Tuesday”:
      // Run the Tuesday report
      break;

      case “Wednesday”:
       // Run the Wednesday report
      break;
      // And so on….
}
```

- Case for every day of the week
- Simpler syntax than splitting into many if-else
- …the cases go on for the rest of the week

```
switch (dayOfWeek)
{
    case “Monday”:
    // Run the Monday report
    break;
}
```

Understanding Case Labels

Define simple patterns that are matched against the top level Switch Expression

Patterns are usually simple constants such as ints, booleans and strings

A note about type patterns.

```
switch (productCategory)
{
      case “Coffee”:
      Console.Writeline(“The Coffee category”);
      // Do other stuff
      break;

      case “Food”:
      Console.Writeline(“The Food category”);
      break;

      default:
      Console.Writeline(“Some other category”);
      break;
}
```

◀ A value or expression to match against

◀ Executes if no other case is matched

```
switch (priceScore)
{
    case 1.0:
    case 2.0:
    Console.Writeline(“Not good.”);
    break;

    case 3.0:
    Console.Writeline(“Average.”);
    break;

    case 4.0:
    case 5.0:
    Console.Writeline(“Goood job!”);
    break;
}
```

◀ A value or expression to match against

◀ Executes if priceScore is 1.0 or 2.0

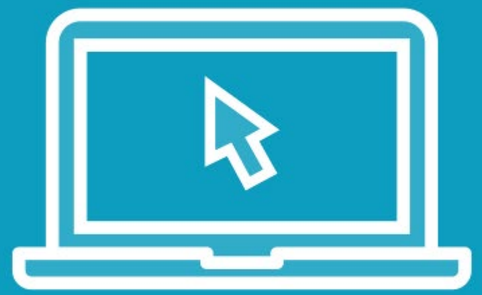
◀ Executes if priceScore is 3.0

◀ Executes if priceScore is 4.0 or 5.0

If-Else Statements vs Switch Statements

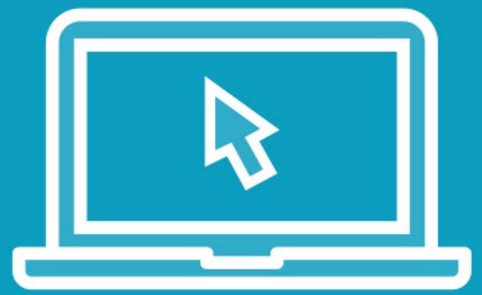
if-else	switch
Select code based on boolean logic	Select code based on value matching
Supports varying, branching logical comparisons	Matches simple patterns against one top level expression
Often verbose with complex paths	Often simpler, easier to read

Demo



Working with Switch Statements

Demo



Organizing the Application Code

Overview/ Summary

- Selection statements directly control Program flow by selecting code blocks to run
- If-else statements select code to run based on logical boolean comparisons
- Allow for branching conditions using if, else, else-if
- Switch statements are an alternative to if-else statements that use pattern matching
- Switch statements can match many different cases against a top level switch expression