Default and Named Arguments

Lesson Objectives

- After completing this lesson, you should be able to:
  - Utilize default argument values in Scala class constructors and methods
  - Leverage named arguments to only pass certain values

Default Argumets

- Allows the developer to specify a value to use for a constructor or method when none is passed by the caller, and omit values that are frequently the same
- Reduces boilerplate in application source code because you don’t have to “overload” methods with different signatures

Default Arguments

```
def name(first: String =  "", last: String = ""): String =
  first + " " + last

scala> name("Martin")
res0: String = Martin
```

Typesafe

Best Practice

- If you have a mixture of default arguments and those that do not have a default value, put the arguments without defaults first

```
def name(first: String,
         last: String,
         middle: String = ""): String =
  first + " " +  middle + " " + last
```

Typesafe

Named Argumets

- Leading arguments can be omitted if they have default values
- You can specify only the values you want to pass

Named Arguments

```
scala> name(last = "Odersky")
res0: String = Odersky

scala> name(first = "Martin", last = "Odersky")
res0: String = Martin Odersky
```

Typesafe

Lesson Summary

- Having completing this lesson, you should be able to:
  - Utilize default argument values in Scala class constructors and methods
  - Leverage named arguments to only pass certain values