Accessibility and Companion Objects

Lesson Objectives

- After completing this lesson, you should be able to:
  - Leverage accessibility keywords to manage visibility of methods and fields
  - Describe the role companion objects play in Scala
  - Outline how to use companion objects

Typesafe

Accessibility

- We can use keywords to limit the visibility of methods and fields in class instances
  - **`public`**, the default
  - **`private`**, limiting visibility only to yourself
  - **`protected`**, unimportant for now

Typesafe

Accessibility

```
class Hello {
  private val message: String = "Hello!"
}

class Welcome {
  val message: String = "Hello!"
}
```

Companion Objects

- If a Singleton object and a class share the same name and are located in the same source file, they are called companions
- A companion class can access private fields and methods inside of its companion object

Companion Objects

- This is a great way to separate static members (fields, constants and methods) that are unrelated to a specific instance from those members that are related to a specific instance of that class

Companion Objects

```
object Hello {
  private val defaultMessage = "Hello!"
}

class Hello(message: String = Hello.defaultMessage) {
  println(message)
}
```

Lesson Summary

- Having completing this lesson, you should be able to:
  - Leverage accessibility keywords to manage visibility of methods and fields
  - Describe the role companion objects play in Scala
  - Outline how to use companion objects