

# NoiseMaker

## Lesson 7

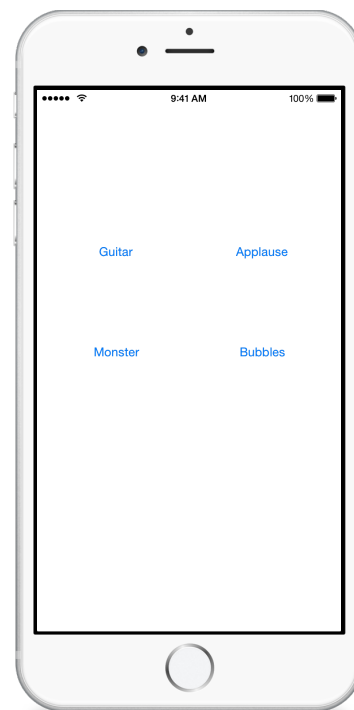


### Description

Replace the individual `AVAudioPlayer` properties with an array of `AVAudioPlayer` objects.

### Learning Outcomes

- Analyze repetitive code and infer opportunities to use data structures.
- Describe the difference between mutability and immutability, and relate these concepts to arrays.
- Practice literal array initialization, and accessing array objects with numeric indices.
- Practice using a `for-in` loop to iterate over an array.
- Describe the use of the `map` function to transform arrays.



### Vocabulary

array	array literal	mutability
immutable	mutable	for-in loop
transformation	map	closure
type annotation	trailing closure	

### Materials

- **NoiseMaker Lesson 7** Xcode project

## Opening

How many `AVAudioPlayer` objects do we have in our model, and what are we doing to create each one?

## Agenda

- Discuss the code smell of repetitive variables with number suffixes (e.g. `url12`), and the numerous similar `AVAudioPlayer` objects as motivation for using a data structure.
- Add a new `[String]` property to the `NoiseMaker` model to store the audio file names.

```
class NoiseMaker {  
  
    let audioFileNames = ["guitar", "applause", "monster", "bubbles"]  
    ...  
}
```

- Discuss the array literal syntax used to provide a default value for the `audioFileNames` property, and how `let` indicates that the array is immutable.
- Replace the four individual `AVAudioPlayer` properties with a single `[AVAudioPlayer]` property to store the `AVAudioPlayer` objects.

```
var players = [AVAudioPlayer]()
```

- Discuss how using `var` indicates that the array is mutable.
- Within the `NoiseMaker` initializer, delete the existing repetitive `NSURL` and `AVAudioPlayer` instantiations.
- Complete the implementation of `init`, using the `audioFileNames` array and a `for-in` loop to create new `AVAudioPlayer` objects.

```
init() {  
    for filename in audioFileNames {  
        let url = NSBundle.mainBundle().URLForResource(filename,  
            withExtension: "wav")  
        players.append(AVAudioPlayer(contentsOfURL: url, error: nil))  
    }  
}
```

- Explain how a Swift `for-in` loop retrieves each `String` in the `audioFileNames` array successively, assigning each `String` to the implicit `filename` constant during each iteration of the loop.
- Discuss the body of the `for-in` loop, and how, for each `filename` retrieved from the `audioFileNames` array, an `NSURL` is created, and a new `AVAudioPlayer` object is appended to the `players` array.

- Refactor each of the "play" methods to use the `players` array instead of specific, named `AVAudioPlayer` properties.

```
func playGuitarSound() {  
    players[0].play()  
}  
...  
func playBubblesSound() {  
    players[3].play()  
}
```

- Run the app (⇧⌘R), and verify that the sounds still play.
- Discuss the reduction of repetitive code in the model, and the decrease in the number of lines of code.
- Discuss how the `NoiseMaker` initializer uses an array of audio filenames to generate an array of `AVAudioPlayer` objects.
- Present the concept of the Swift `map` function.
- Change the `players` property to a constant with a type annotation and no default value.

```
let players: [AVAudioPlayer]
```

- Explain how the initializer will be updated to create and assign an `[AVAudioPlayer]` to the `players` property.
- Replace the `for-in` loop in the initializer with a verbose call to `map`.

```
init() {  
    players = audioFileNames.map( { (filename: String) -> AVAudioPlayer  
in  
    let url = NSBundle.mainBundle().URLForResource(filename,  
        withExtension: "wav")  
    return AVAudioPlayer(contentsOfURL: url, error: nil))  
    })  
}
```

- Explain how the initializer transforms the array of `String` values into an array of `AVAudioPlayer` objects, by calling `map`.
- Explain how `map` receives a closure and invokes it, passing each `String` in the `audioFileNames` array as the `filename` argument; and how it builds an array with each `AVAudioPlayer` object returned by the closure.
- Discuss how Swift trailing closure syntax and inferred data types can increase the brevity of the code.
- Update the call of `map` by using a trailing closure and by omitting the type annotations.

```
players = audioFileNames.map { filename in
    let url = NSBundle.mainBundle().URLForResource(filename,
        withExtension: "wav")
    return AVAudioPlayer(contentsOfURL: url, error: nil))
}
```

- Explain how Swift infers the type of the `filename` parameter based on the `[String]` type of `audioFileNames` array; and infers the return type of the closure based on the `[AVAudioPlayer]` type of the `players` array.
- Discuss how the `NoiseMaker` class no longer relies on a mutable array, and how the initializer is more expressive.
- Run (⌘R) the app, and verify that the sounds still play as expected.

## Closing

We still see repetitive code across both the model and controller "play" methods. Is the repetition related? Can you think of a way we might improve this code even further?

## Modifications and Extensions

- Investigate the concepts of designated and convenience initializers, and create a designated initializer called `initWithFileNames:`. Remove the `audioFileNames` property from the `NoiseMaker` model, and update the `ViewController` to pass a `[String]` of audio file names to the designated initializer.
- Store a collection of audio file names in an external property list file, and load the file names from the property list file instead of using a hard-coded array of explicit file names.

## Resources

The Swift Programming Language: Properties [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/Properties.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Properties.html)

The Swift Programming Language: Collection Types [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/CollectionTypes.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/CollectionTypes.html)

The Swift Programming Language: Initialization [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/Initialization.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Initialization.html)

The Swift Programming Language: Control Flow [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/ControlFlow.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/ControlFlow.html)

Swift Standard Library Reference: Array <https://developer.apple.com/library/ios/documentation/General/Reference/SwiftStandardLibraryReference/Array.html>