UITextFields and UIPickerViews

UITextField

- An object that displays an editable text area
- Used too gather text-based input from the user using the onscreen keyboard
- Keyboard is configurable for many types of input, such as:
  - plain text
  - emails
  - numbers
  - etc…
- Uses actions and a delegate object to report changes made during the course of editing
- Show the keyboard via invoking
  - becomeFirstResponder()
  - Tapping within the bounds of the object
- Hide the keyboard via invoking
  - resignFirstResponder()

UITextFieldDelegate

- No **required** functions
- Can be set programmatically in the **ViewController**
- Can be set via Storyboard in **InterfaceBuilder**
- Offers options to manage the editing and validation of text in a **UITextField** object

Using The Delegate

```swift
import UIKit

final class ViewController: UIViewController {
    @IBOutlet private weak var textField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        textField.delegate = self // Can also be set in Storyboard
    }
}

/// All of the available `UITextFieldDelegate` functions
extension ViewController: UITextFieldDelegate {
    // return NO to disallow editing.
    func textFieldShouldBeginEditing(_ textField: UITextField) -> Bool

    // became first responder
    func textFieldDidBeginEditing(_ textField: UITextField)

    // return YES to allow editing to stop and to resign first responder status. NO to disallow the editing session to end
    func textFieldShouldEndEditing(_ textField: UITextField) -> Bool

    // may be called if forced even if shouldEndEditing returns NO (e.g. view removed from window) or endEditing:YES called
    func textFieldDidEndEditing(_ textField: UITextField)

    // if implemented, called in place of textFieldDidEndEditing:
    func textFieldDidEndEditing(_ textField: UITextField, reason: UITextField.DidEndEditingReason)

    // return NO to not change text
    func textField(_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool

    @available(iOS 13.0, *)
    func textFieldDidChangeSelection(_ textField: UITextField)

    // called when clear button pressed. return NO to ignore (no notifications)
    func textFieldShouldClear(_ textField: UITextField) -> Bool

    // called when 'return' key pressed. return NO to ignore.
    func textFieldShouldReturn(_ textField: UITextField) -> Bool
}
```

UITextField InputView

```
var inputView: UIView?
```

The custom input view to display when the text field becomes the first responder.

```
var inputAccessoryView: UIView?
```

The custom accessory view to display when the text field becomes the first responder

We will use the inputView attribute of **UITextField** in conjunction with a **UIPickerView** to create a custom keyboard later in this lecture's activity.

UIPickerView

- A view that uses a spinning-wheel or slot-machine metaphor to show one or more sets of values
- Needs two things in order function:
  - delegate (UIPickerViewDelegate)
  - dataSource (UIPickerViewDataSource)

UIPickerViewDelegate

- No **required** functions
- Can be set programmatically in the **ViewController**
- Can be set via Storyboard in **InterfaceBuilder**
- Offers options for managing attributes of a **UIPickerView**, including:
    - Component customization (title/size/view)
    - User input

```
func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) -> String? {
    return AstrologicalSign.allCases[row].displayName
}

func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int) {
    // Implement this method to d something when a row is selected
}
```

Using The Delegate

```swift
import UIKit

final class ViewController: UIViewController {
    @IBOutlet private weak var pickerView: UIPickerView!

    override func viewDidLoad() {
        super.viewDidLoad()
        pickerView.delegate = self // Can also be set in Storyboard
    }
}

extension ViewController: UIPickerViewDelegate {
    func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) -> String? {
        return AstrologicalSign.allCases[row].displayName
    }

    func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int) {
        // Implement this method to d something when a row is selected
    }
}
```

UIPickerViewDataSource

- The bridge between the UIPickerView and what it's displaying
- Can be set programmatically in the **ViewController**
- Can be set via Storyboard in **InterfaceBuilder**
- **Two required functions**:

```
func numberOfComponents(in pickerView: UIPickerView) -> Int {
    // Method used to notify the pickerView of how many components are in its list
    return 1
}

func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {
    // Method used to notify the pickerView of how many items are in its list
    return AstrologicalSign.allCases.count
}
```

Using The Data Source

```swift
import UIKit

final class ViewController: UIViewController {
    @IBOutlet private weak var pickerView: UIPickerView!

    override func viewDidLoad() {
        super.viewDidLoad()
        pickerView.dataSource = self // Can also be set in Storyboard
    }
}

extension ViewController: UIPickerViewDataSource {
    func numberOfComponents(in pickerView: UIPickerView) -> Int {
        return 1
    }

    func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {
        return AstrologicalSign.allCases.count
    }
}
```

UIDatePicker

- A special **UIPickerView** used for inputting date and time values

- We will not implement one in class 😰
  - Documentation available here