Stanford CS193p

Developing Applications for iOS
Fall 2017-18

Today

- More about Documents
  - `UIDocumentBrowserViewController`
- Demo
  - Use `Codable` to create a JSON representation of our document
  - Store it in the filesystem
  - Think better of that and let `UIDocument` store it
  - Use `UIDocumentBrowserViewController` to choose/create/rename/move our documents

UIDocumentBrowserViewController

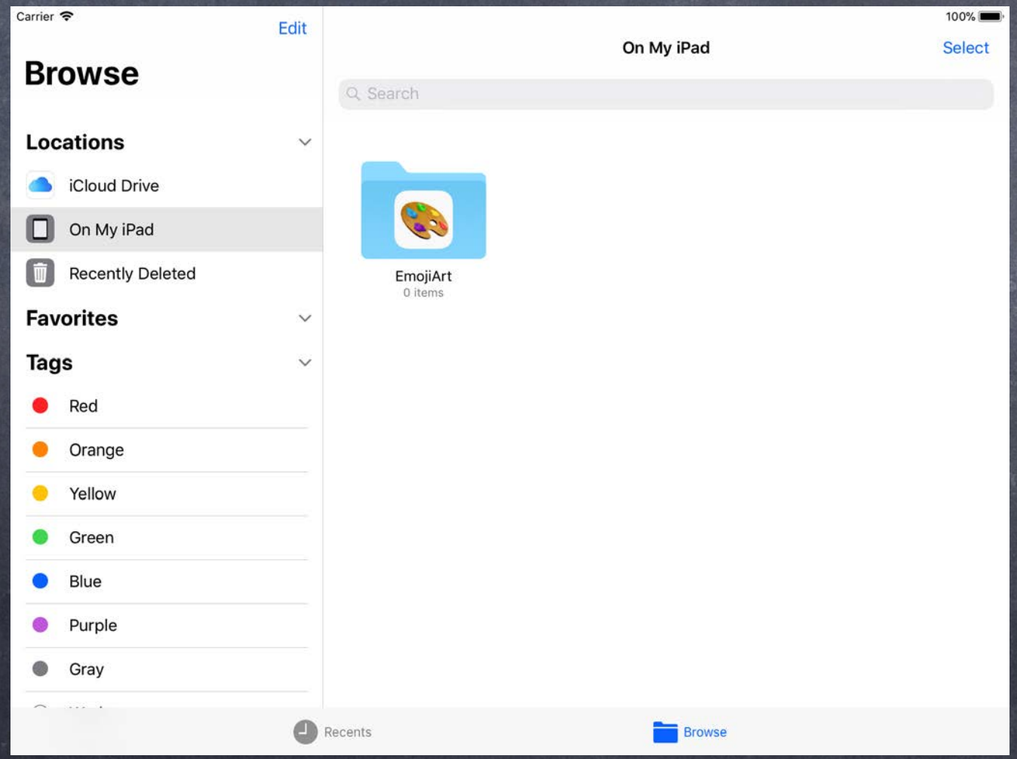
- Managing user documents

  You probably want users to be able to easily manage their documents in a document-based app.

  Choosing files to open, renaming files, moving them, accessing iCloud drive, etc.

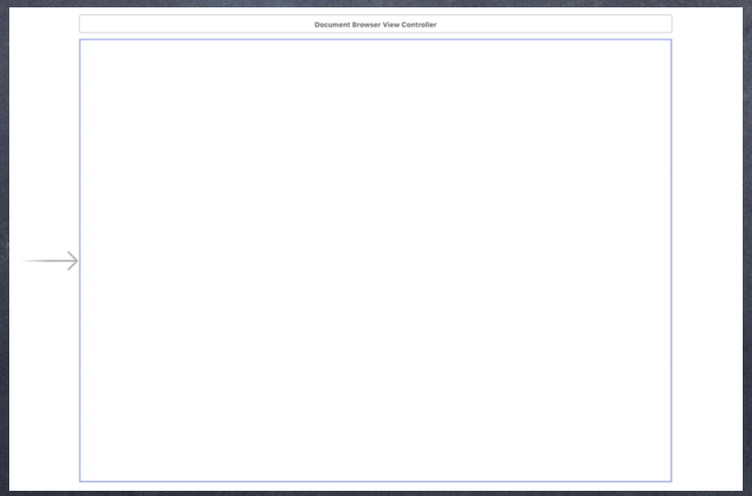
  The `UIDocumentBrowserViewController` (UIDBVC) does all of this for you.

  Using `UIDocument` to store your document makes leveraging this UIDBVC easy.



UIDocumentBrowserViewController

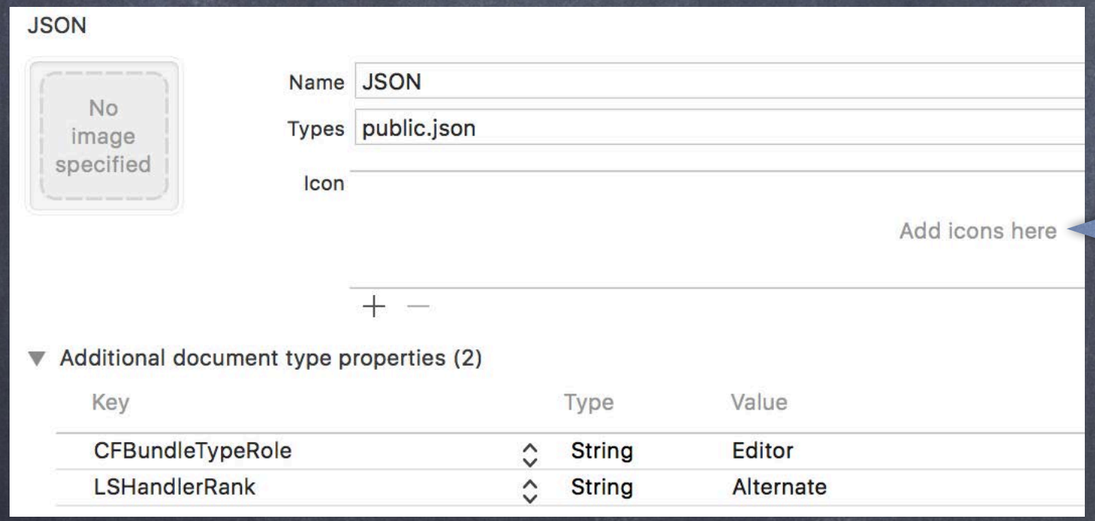
- Managing user documents
  - You probably want users to be able to easily manage their documents in a document-based app.
  - Choosing files to open, renaming files, moving them, accessing iCloud drive, etc.
  - The `UIDocumentBrowserViewController` (UIDBVC) does all of this for you.
  - Using `UIDocument` to store your document makes leveraging this `UIDBVC` easy.
- Using the `UIDocumentBrowserViewController`
  - It has to be the root view controller in your storyboard (i.e. the arrow points to it).
  - Your document-editing MVC will then be presented modally on top of (i.e. takes over the screen).



UIDocumentBrowserViewController

- What document types can you open?

  To use the UIDBVC, you have to register which types your application uses.
  You do this in the Project Settings in the Info tab with your Target selected.
  In the Document Types area, add the types you support.
  Here's what it looks like to support JSON files ...

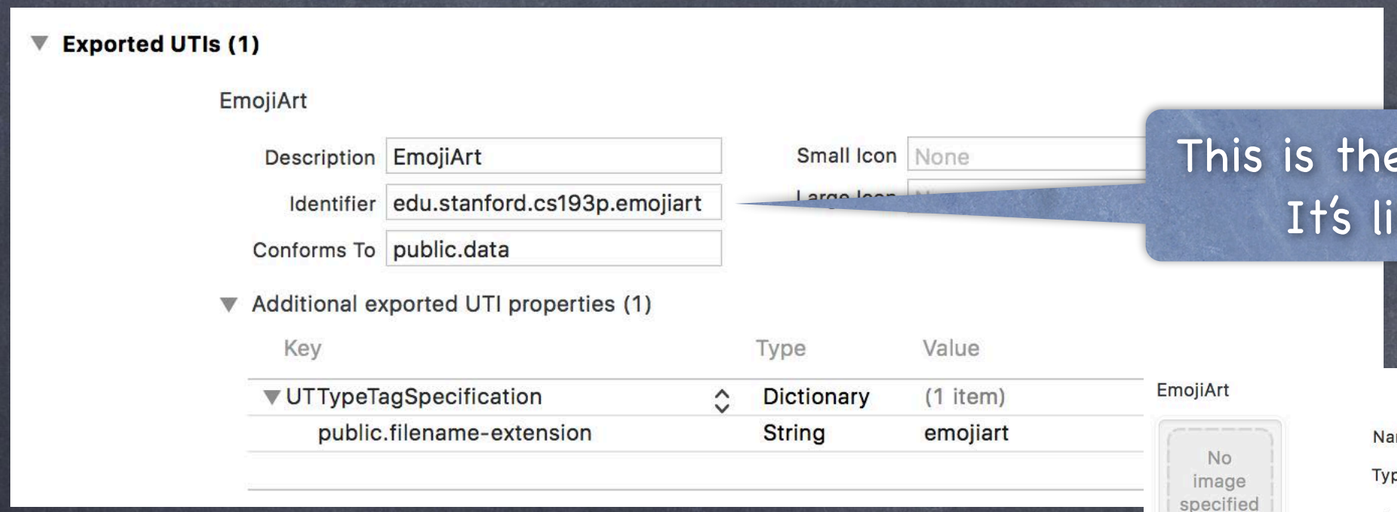


You can add an icon for the file type too.

The Types field is the UTI of the type you want to support (e.g. public.json, public.image).
The CFBundleTypeRole and LSHandlerRank say how you handle this kind of document.
Are you the primary editor and owner of this type or is it just something you can open?

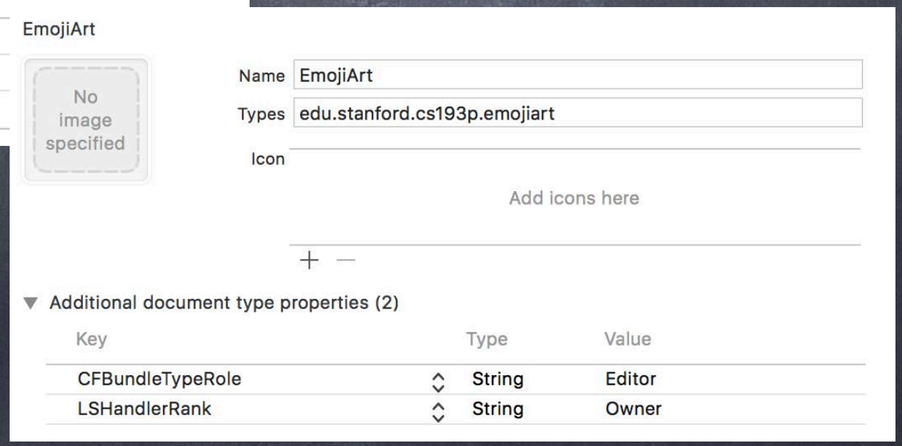
UIDocumentBrowserViewController

- Declaring your own document type
  - You might have a custom document type that your application edits
  - You can add this under Exported UTIs in the same place in Project Settings
  - Here's an example of adding an "emojiart" type of document …



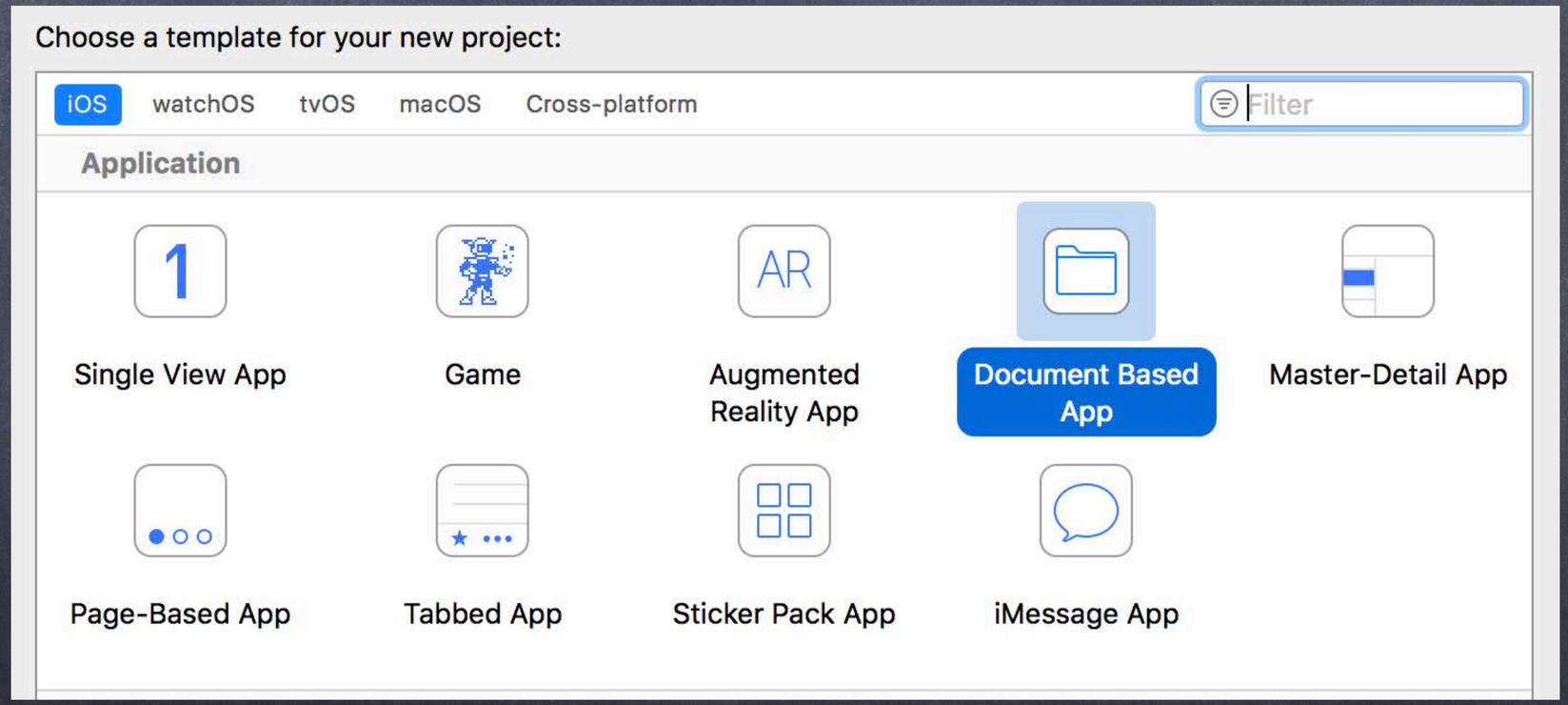
This is the "UTI" that we keep referring to. It's like `public.json` is for JSON.

… and then add it as a supported Document Type



UIDocumentBrowserViewController

- Xcode template
  - Setting up a UIDocumentBrowserViewController-based application requires a bit of setup
  - Mostly an entry in your Info.plist, a little bit of AppDelegate code and some stubbed-out code
  - We don't usually use an Xcode template in this course, but in this case it make sense



UIDocumentBrowserViewController

- What is in the template?
  - A stub for `Document Types` in Project Settings (supports `public.image` file types)
  - An `Info.plist` entry `Supports Document Browser = YES`
  - A bit of code in `AppDelegate` to allow other apps (like Files) to get your app to open a file
  - A stubbed out `UIDocument` subclass (with empty `contents` and `load(fromContents)` methods)
  - A stubbed out MVC to display a document (just calls `UIDocument`'s `open` and `close` methods)
  - A subclass of `UIDocumentBrowserViewController` (with almost everything implemented)
- What you need to do to personalize this template …
  1. Use your `UIDocument` subclass instead of the stubbed out one
  2. Use your document-viewing MVC code (already using `UIDocument`) instead of stub
  3. Add code to `UIDBVC` subclass to …
     a. configure the `UIDBVC` (allow multiple selection? creation of new documents? etc.)
     b. specify the url of a template document to copy to create new documents
     c. present your document-viewing MVC modally given the url of a document
  4. Update the `Document Types` in Project Settings to be your types (instead of `public.image`)

UIDocumentBrowserViewController

- Steps 1 and 2

  As long as you properly implement UIDocument in your MVC, this is no extra work

- Step 3a: Configuring the UIDBVC

  This happens in its viewDidLoad …

```
override func viewDidLoad() {
    super.viewDidLoad()
    delegate = self      // the guts of making UIDBVC work are in its delegate methods
    allowsDocumentCreation = true
    allowsPickingMultipleItems = true
    browserUserInterfaceStyle = .dark
    view.tintColor = .white
}
```

  Set these as you wish.

UIDocumentBrowserViewController

- Steps 3b: Specifying the "new document" template URL

  This happens in this UIDBVC delegate method …

```
func documentBrowser(_ controller: UIDBVC,
    didRequestDocumentCreationWithHandler handler: @escaping (URL?, UIDBVC.ImportMode) -> Void
) {
    let url: URL? = … // where your blank, template document can be found
    importHandler(url, .copy or .move)
}
```

  Usually you would specify `.copy`, but you could create a new template each time and `.move`.

  Likely you would have some code here that creates that blank template (or ship with your app).

UIDocumentBrowserViewController

- Aside: Presenting an MVC without segueing

  We haven't covered how to present MVCs in any other way except by segueing.

  So let's cover it now!

  It's very easy. You present a new MVC from an existing MVC using `present(animated:)` …

```
let newVC: UIViewController = …
existingVC.present(newVC, animated: true) {
    // completion handler called when the presentation completes animating
    // (can be left out entirely if you don't need to do anything upon completion)
}
```

  The real trick is "where do I get `newMVC` from?"

  Answer: you get it from your storyboard using its identifier which you set in Identity Inspector

```
let storyboard = UIStoryboard(name: "Main", bundle: nil) // Main.storyboard
if let newVC = storyboard.instantiateViewController(withIdentifier: "foo") as? MyDocVC {
    // "prepare" newMVC and then present(animated:) it
}
```

UIDocumentBrowserViewController

- Steps 3c: Presenting your document MVC modally

  The Xcode template stubs out a function called `presentDocument(at: URL)` to do this …

```
func presentDocument(at url: URL) {
    let story = UIStoryboard(name: “Main”, bundle: nil)
    if let docvc = story.instantiateViewController(withIdentifier: “DocVC”) as? DocVC {
        docvc.document = MyDocument(fileURL: url)
        present(docvc, animated: true)
    }
}
```

  You can call this function anything you want.

  But the point is that it takes a `URL` to one of your documents and you show it.

  The Xcode template then calls this from the appropriate `delegate` methods in `UIDBVC`.

  That's all you have to do to get `UIDBVC` working.

UIDocumentBrowserViewController

- Step 4: Specifying your types
  - Unless your app opens `public.image` files, you'll need to change that in Project Settings
  - For your homework, for example, you'll probably need to invent a new type for Image Gallery

Demo Code

Download the demo code from today's lecture.