TODAY

• Multi-Threading
• UITableView
MULTI-THREADING

- What for?
  - handling long-running activities
  - blocking activities
  - computationally expensive
- How? Grand Central Dispatch
  - most used are the queues, but also
  - lock-based synchronization
  - barriers
  - etc.

DISPATCH QUEUES

- Main Queue (serial)
  - all UI activity here
  - no long-running, non-UI activity here
  - want UI to be responsive
    ```swift
    let mainQ = DispatchQueue.main
    ```
- Global Queues (concurrent)
  ```swift
  let backgroundQ = DispatchQueue.global(qos: .userInteractive).
  .userInitiated // high priority, longer
  .background    // not user-created, slow
  .utility      // system maintenance
  ```
- Adding to queues
  ```swift
  backgroundQ.async { <#code#> }     // most common
  backgroundQ.sync   { <#code#> }
  ```
CUSTOM QUEUES

- Custom Queues
  - can be serial (single thread)
    ```swift
    let serialQ = DispatchQueue(label: "MySerial")
    ```
  - or concurrent (multiple threads)
    ```swift
    let concurrentQ = DispatchQueue(label: "MyConcurrent",
                                     attributes: .concurrent)
    ```

SESSIONS AND URLS

- Also: URLSession
  - Lets you fetch the contents of an http URL into a Data off the main queue!
  - To repeat: not on the main queue
  - Means we must dispatch back to the main queue for UI stuff
TABLE/COLLECTION VIEWS

- UIScrollView subclasses
  - unbounded amounts of information
  - TV in a long, possibly sectioned list
  - CV in a “flowing” 2D format
  - Similar APIs
TABLE VIEWS

• Can be simple

• Or sectioned
TABLE VIEWS

- Can be simple
- Or sectioned
- With extra information

Left detail style
TABLE VIEWS

- Can be simple
- Or sectioned
- With extra information
- Left detail style
- Right detail style
- Complicated
TABLE/COLLECTION VIEWS

- **How**
  - programmatically
  - draw out from IB library
  - or draw out from IB library with view controller
- **Data**
  - remember that views don’t own their data
  - comes from tableView property called `dataSource`
    - works like a delegate
    - both views also have a `delegate`, but just for how they look
Almost always set to the main controller

```swift
var dataSource: UITableView(Collection)ViewDataSource
var delegate: UITableView(Collection)ViewDelegate
```

Many, many functions, but main ones are:

```swift
func numberOfSections(in tableView: UITableView) -> Int
func tableView(_ tv: UITableView, numberOfRowsInSection section: Int) -> Int
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
```

`indexPath.section` and `indexPath.row/item` gives section and row (for table) or item (collection).

```swift
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let prototype = "ScoreCell"
    let cell = tv.dequeueReusableCell(withIdentifier: prototype, for: indexPath)
    if let myCell = cell as? ScoreCell {
        myCell.rank = score(at: indexPath).rank
        myCell.points = score(at: indexPath).points
        myCell.when = score(at: indexPath).when
        return myCell
    } else {
        return cell
    }
}
```
TABLE/COLLECTION VIEWS

• First cells come from **prototypes**
  • In table view attr inspector, increment “Prototype Cells”
  • New cells cloned from prototypes

• Cell Reuse
  • TableView might have thousands of rows
  • Just keeps views currently showing.
  • `dequeueReusableCell(withIdentifier:)` grabs one from reuse pool
How to access prototype fields?

- In TableView, default cells have only a few things:
  - .textLabel?
  - .detailTextLabel?
  - .imageView?
- But what about custom cells?
TABLE/COLLECTION VIEWS

• Static table views
  • just for laying out UI elements
  • iOS settings
  • set cells to “Static”

TABLE VIEW SEGUES

• prepare() -ing to segue
  • “sender” is UITableViewCell
  • switch on the identifier
  • cast sender to custom cell, if necessary
  • set it up
Your data is often not static, must **notify the view**:

- Change it all:
  ```swift
  func reloadData()
  ```

- Change some:
  ```swift
  func reloadData(at indexPaths: [IndexPath],
                withAnimation: UITableViewRowAnimation)
  ```

- And many others

Each section can have a header

- custom view
- simple string through overriding:
  ```swift
  func tableView(_ tv: UITV, titleForHeaderInSection sec: Int) -> String?
  ```

Can also control the height

- automatic (can help by setting `UITableView.estimatedRowHeight`)
- fixed: `UITableView.rowHeight`
- delegate method:
  ```swift
  func collectionView(_ collectionView: UICollectionView,
                      layout collectionViewLayout: UICollectionViewLayout,
                      sizeForItemAt indexPath: IndexPath
  ) -> CGSize
  ```
GROUP PROJECT

- Randomly assigned 3-person teams (after the break)
- You pick/elaborate on a set of 10-12 pre-defined ideas, such as:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Bike route planner / tracker</td>
<td>2. Weasley magic clock (inspired by Harry Potter)</td>
</tr>
<tr>
<td>a. track route with line on map</td>
<td>a. clock laid out with pre-specified locations (home, work, gym...)</td>
</tr>
<tr>
<td>b. export to a .fit</td>
<td>b. arm for each family member / friend, pointing out where they are</td>
</tr>
<tr>
<td>c. upload to strava</td>
<td>c. intelligent interpolation for intermediate states</td>
</tr>
<tr>
<td>d. plan from current location (this would require google map sdk)</td>
<td></td>
</tr>
<tr>
<td>3. Asteroids-equivalent shooter game</td>
<td></td>
</tr>
<tr>
<td>a. spritekit</td>
<td></td>
</tr>
<tr>
<td>b. sound</td>
<td></td>
</tr>
<tr>
<td>c. artwork</td>
<td></td>
</tr>
<tr>
<td>4. Augmented reality: Real-time flagging of problematic ingredients</td>
<td></td>
</tr>
<tr>
<td>a. aim camera at an ingredient list</td>
<td></td>
</tr>
<tr>
<td>b. real-time bold-face and in red selected ingredients or their equivalents, e.g.</td>
<td></td>
</tr>
<tr>
<td>i. MSG</td>
<td>i. nuts</td>
</tr>
<tr>
<td>ii. nuts</td>
<td>ii. gluten</td>
</tr>
</tbody>
</table>

- Send ideas to me!