

Drag and Drop

• Item Provider

We are going to see drag and drop code in action in demo today.

The heart of drag and drop is the `NSItemProvider` class.

It facilitates the transfer of data between processes (via drag and drop, for example).



Drag and Drop

• Item Provider

We are going to see drag and drop code in action in demo today.

The heart of drag and drop is the `NSItemProvider` class.

It facilitates the transfer of data between processes (via drag and drop, for example).

We are not really going to go into detail about it, though.

It's a little bit beyond the scope of this course.

There is some code to help with getting data from an `NSItemProvider` in the EmojiArt demo.



Drag and Drop

• Item Provider

We are going to see drag and drop code in action in demo today.

The heart of drag and drop is the `NSItemProvider` class.

It facilitates the transfer of data between processes (via drag and drop, for example).

We are not really going to go into detail about it, though.

It's a little bit beyond the scope of this course.

There is some code to help with getting data from an `NSItemProvider` in the EmojiArt demo.

`NSItemProvider` can facilitate the transfer of a number of data types in iOS, for example:

`NSAttributedString` (a string with formatting) and `NSString`

`NSURL`

`UIImage` and `UIColor`



Drag and Drop

• Item Provider

We are going to see drag and drop code in action in demo today.

The heart of drag and drop is the `NSItemProvider` class.

It facilitates the transfer of data between processes (via drag and drop, for example).

We are not really going to go into detail about it, though.

It's a little bit beyond the scope of this course.

There is some code to help with getting data from an `NSItemProvider` in the EmojiArt demo.

`NSItemProvider` can facilitate the transfer of a number of data types in iOS, for example:

`NSAttributedString` (a string with formatting) and `NSString`

`NSURL`

`UIImage` and `UIColor`

Unfortunately all of these NS things are pre-Swift.

So when we want to use `NSItemProvider`, we end up "bridging" Swift types to these types.

We do this with "as", for example, `String as NSString`.

