

Saving Data in iOS

Using XML

NSXMLParser

- ⚙ It is the default XML parser included in iOS
- ⚙ It is a **SAX** (Simple API for XML) based XML parser.
- ⚙ It is fast and must be tuned for the XML feed that you are parsing.
- ⚙ Uses a delegate to handle events as they occur while parsing an XML document.

NSXMLParser Cons

- ⚙ NSXMLParser does not perform any validation on your XML document.
- ⚙ It is not meant to modify the XML document and write it to disk.
- ⚙ To construct your own tree representation of the XML document, you will have to do it by hand.
- ⚙ Can be a bit unwieldy.



Using NSXMLParser

```
<?xml version="1.0"?>
<video_games>
  <video_game>
    <name>Minecraft</name>
    <rating ranking="five-star">5</rating>
    <synopsis><![CDATA[ Build your own voxel ...]]></synopsis>
  </video_game>
</video_games>
```

- ⚙ Your class must implement the NSXMLParserDelegate protocol.

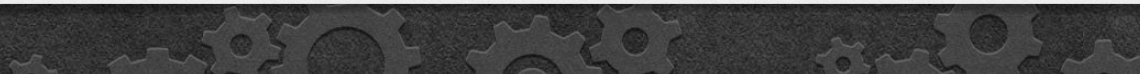
```
self.xmlParser = [[NSXMLParser alloc] initWithData:xmlData];
self.xmlParser.delegate = self;
[self.xmlParser parse];
```


Responding to Events

- ⚙ parser:didStartElement:elementName:namespace:qualifiedName:attributes
- ⚙ parser:foundCharacters:
- ⚙ parser:didEndElement:elementName:namespace:qualifiedName



Demo



DOM Based Parser

- ⚙ DOM based parsing builds an entire tree of the XML inside of your app.
- ⚙ Allows you to run XPath queries on the XML tree.
- ⚙ Some libraries give you the ability to change the actual XML data and write it back to disk.



RaptureXML

```
<?xml version="1.0"?>
<video_games>
  <video_game>
    <name>Minecraft</name>
    <rating ranking="five-star">5</rating>
    <synopsis><![CDATA[ Build your own voxel ...]]></synopsis>
  </video_game>
</video_games>
```

```
RXMLElement * rootXML = [RXMLElement elementFromXMLFile:@"video_games.xml"];
```

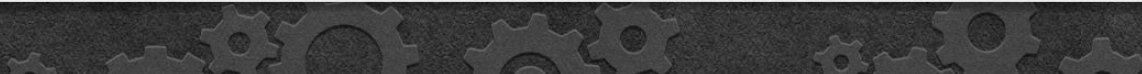
```
NSArray * video_games = [rootXML children:@"video_game"];
```


RaptureXML (cont'd)

```
[rootXML iterate:@"video_game" usingBlock:^(RXMLElement *element) {  
    NSLog(@"Game Name: %@", [element child:@"name"].text);  
    RXMLElement * rating = [element child:@"rating"];  
    NSLog(@"Rating: %@", rating.text);  
    NSLog(@"Rating System: %@", [rating attribute:@"ranking"]);  
}];
```

```
[rootXML iterateWithRootXPath:@"//video_game" usingBlock:^(RXMLElement *element) {  
    NSLog(@"Game Name: %@", [element child:@"name"].text);  
    RXMLElement * rating = [element child:@"rating"];  
    NSLog(@"Rating: %@", rating.text);  
    NSLog(@"Rating System: %@", [rating attribute:@"ranking"]);  
}];
```

Demo



Creating a New XML Document

- ⚙ A new XML document is typically handled by using `NSXMLDocument`
- ⚙ Unfortunately, `NSXMLDocument` is currently not available on iOS.
- ⚙ You can either write the new XML in an `NSString` or use a third party library.



XSWI

```
XMLWriter * writer = [[XMLWriter alloc] init];  
[writer writeStartDocumentWithEncodingAndVersion:@"UTF-8" version:@"1.0"];
```

```
[writer writeStartElement:@"game"];
```

```
[writer writeStartElement:@"name"];
```

```
[writer writeCharacters:@"Minecraft"];
```

```
[writer writeEndElement];
```

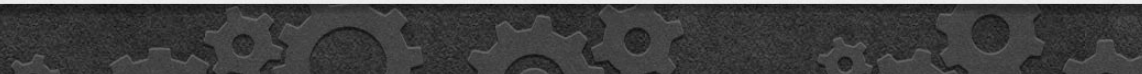
```
[writer writeCDATA:@"desc..."];
```

```
[writer writeEndDocument];
```

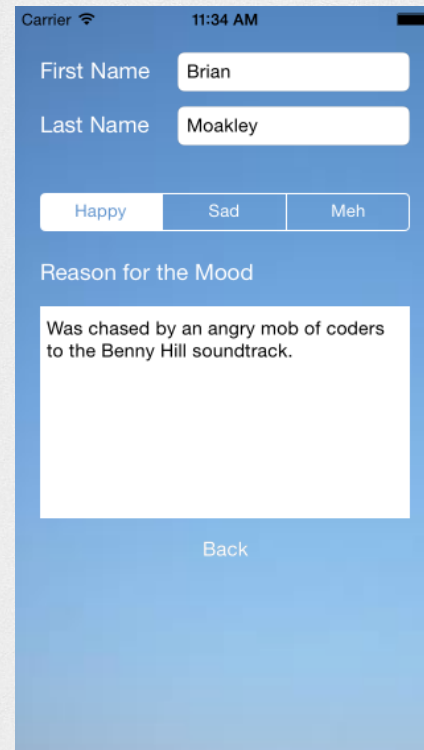
```
myString = [writer toString];
```

```
<?xml version="1.0"?>  
<game>  
    <name>Minecraft</name>  
    <sum><![CDATA[ desc...]]></sum>  
</game>
```


Demo



Challenge



Carrier 11:34 AM

First Name Brian

Last Name Moakley

Happy Sad Meh

Reason for the Mood

Was chased by an angry mob of coders to the Benny Hill soundtrack.

Back