What’s New in TVMLKit

Session 238

Jeremy Foo, tvOS Engineering
Web Inspector Enhancements
Data Binding
Customizing Playback
Web Inspector Enhancements
Web Inspector Enhancement
Toggle Event Listeners
Web Inspector Enhancement
Toggle Event Listeners

- Event Listeners
  - play
    - Node: document
    - Function: handleEvent
    - Bubbling: Yes
    - Enabled: Yes
  - select
    - Node: document
    - Function: handleEvent
    - Bubbling: Yes
    - Enabled: Yes

Accessibility
No Accessibility Information
# Web Inspector Enhancement

## Image Network Resources

![Web Inspector showing network resources](image)

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Type</th>
<th>Transfer Size</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack.xml</td>
<td>localhost</td>
<td>xhr</td>
<td>12.72 KB</td>
<td>17.4ms</td>
</tr>
<tr>
<td>loadingTemplate</td>
<td>—</td>
<td>doc...</td>
<td>(memory)</td>
<td>0ms</td>
</tr>
<tr>
<td>Stack.xml</td>
<td>localhost</td>
<td>doc...</td>
<td>(memory)</td>
<td>0ms</td>
</tr>
<tr>
<td>square_1.jpg</td>
<td>localhost</td>
<td>jpg</td>
<td>73.12 KB</td>
<td>7.53ms</td>
</tr>
<tr>
<td>square_2.jpg</td>
<td>localhost</td>
<td>jpg</td>
<td>75.82 KB</td>
<td>9.32ms</td>
</tr>
<tr>
<td>square_3.jpg</td>
<td>localhost</td>
<td>jpg</td>
<td>67.37 KB</td>
<td>12.4ms</td>
</tr>
<tr>
<td>square_4.jpg</td>
<td>localhost</td>
<td>jpg</td>
<td>41.98 KB</td>
<td>14.2ms</td>
</tr>
<tr>
<td>square_5.jpg</td>
<td>localhost</td>
<td>jpg</td>
<td>66.62 KB</td>
<td>15.8ms</td>
</tr>
<tr>
<td>square_6.jpg</td>
<td>localhost</td>
<td>jpg</td>
<td>85.64 KB</td>
<td>17.5ms</td>
</tr>
</tbody>
</table>
Web Inspector Enhancement
Image Network Resources
Web Inspector Enhancement

Show Focused Element
Web Inspector Enhancement
Show Focused Element
Web Inspector

Latest macOS

Safari technology preview

Using Web Inspector with tvOS Apps
Data Binding
Data Binding

Transformation of data to user interface elements
Data Binding

Transformation of data to user interface elements

Separation of layout and application logic
Data Binding

Transformation of data to user interface elements

Separation of layout and application logic

Reduces JavaScript code
Data Binding

JS Data Fetch

```json
{
  title: "Title",
  description: "Long Description"
}
```
Data Binding

JS Data Fetch → JS Processing

```json
{
  title: "Title",
  description: "Long Description"
}
```

```html
<banner>
  <title>Title</title>
  <description>Long Description</description>
</banner>
```
Data Binding

JS Data Fetch → Data Bindings

```javascript
{
  title: "Title",
  description: "Long Description"
}
```

```html
<banner>
  <title binding="..." />
  <description binding="..." />
</banner>
```

```html
<title>Title</title>
<description>Long Description</description>
</banner>
```
Data Binding

Binding keys

attribute
textContent
items
## Data Binding

### Attribute

<table>
<thead>
<tr>
<th>Data</th>
<th>Template</th>
<th>Final Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
<td></td>
<td></td>
</tr>
<tr>
<td>imageURL: &quot;http://...&quot;</td>
<td>&lt;img src=&quot;http://...&quot; /&gt;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Orange color not standard code highlight. Should it fit one of the existing code highlight styles?
Data Binding

Attribute

Orange color not standard code highlight. Should it fit one of the existing code highlight styles?
Data Binding

Text Content

```json
{
    title: "A nice title"
}
```

Final Document

```
<title>A nice title</title>
```
Orange color not standard code highlight. Should it fit one of the existing code highlight styles?
# Data Binding

## Items

<table>
<thead>
<tr>
<th>Data</th>
<th>Template</th>
<th>Final Document</th>
</tr>
</thead>
</table>
| `{  
  items: [  
    { /* list item 1 */ },  
    { /* list item 2 */ },  
    { /* list item 3 */ }  
  ]  
}` | `<section>
  <listItemLockup />
  <listItemLockup />
  <listItemLockup />
</section>` |
Data Binding

Items

```json
{
  items: [
    /* list item 1 */ ,
    /* list item 2 */ ,
    /* list item 3 */
  ]
}
```

```html
<section binding="items:{items}" />
<prototypes>
  <listItemLockup />
</prototypes>
</menuBar>

```
```
Data Binding

Binding keys

attribute
textContent
items
Data Binding

Binding keys

children
Data Binding

Binding keys
- children

Special elements
- fragment
- rules
Data Binding
Children

Generic form of items binding
Data Binding

Children

Generic form of `items` binding

Generates children of any element
Data Binding

Children

Generic form of \texttt{items} binding

Generates children of any element

Works like \texttt{items} binding
Data Binding

Children

Data

```json
{
    items: [
        { /* menu item 1 */ },
        { /* menu item 2 */ },
        { /* menu item 3 */ }
    ]
}
```
Data Binding
Children

```json
{
    items: [
        { /* menu item 1 */ },
        { /* menu item 2 */ },
        { /* menu item 3 */ }
    ]
}
```

Final Document

```html
<menuBar>
    <menuItem />
    <menuItem />
    <menuItem />
</menuBar>
```
Data Binding

Children

```
{
    items: [
        { /* menu item 1 */ },
        { /* menu item 2 */ },
        { /* menu item 3 */ }
    ]
}
```

```
<menuBar binding="children:{items}" />
<prototypes>
    <menuBarItem />
    <menuBarItem />
    <menuBarItem />
</prototypes>
</menuBar>
```
Data Binding
Fragment

Invisible element to help compartmentalize DOM
Data Binding

Fragment

Invisible element to help compartmentalize DOM

Children of a fragment are visible
Data Binding
Fragment

Invisible element to help compartmentalize DOM

Children of a fragment are visible

Works with children binding
Data Binding
Fragment

```json
{
  items: [
    { /* menu item 1 */ },
    { /* menu item 2 */ },
    { /* menu item 3 */ }
  ]
}
```

<menuBar>
  <nowPlayingMenuItem />
  <fragment>
    <menuBarItem />
    <menuBarItem />
    <menuBarItem />
  </fragment>
</menuBar>
Data Binding
Fragment

```
{  
  items: [  
    { /* menu item 1 */ },  
    { /* menu item 2 */ },  
    { /* menu item 3 */ }  
  ]  
}
```

```
<menuBar>
  <nowPlayingMenuItem />  
  <fragment binding="children:{items}">  
    <prototypes>  
      <menuBarItem />  
      <menuBarItem />  
      <menuBarItem />  
    </prototypes>  
  </fragment>  
</menuBar>
```
progress = 0%
{  
    img: "...",
    title: "...",
    progress: 0.60
}
{  
  img: "...",  
  title: "...",  
  progress: 0.60  
}
Data Binding

Data

```json
{
  img: "...",
  title: "...",
  progress: 0.0
}
```

Final Document

```xml
<lockup>
  <img src="..." />
  <title>...</title>
</lockup>

<lockup>
  <img src="..." />
  <title>...</title>
  <overlay>
    <progressBar value="0.60" />
  </overlay>
</lockup>
```
Data Binding

Data

```json
{
  img: "...",
  title: "...",
  progress: 0.0
}
```

Template

```html
<lockup>
  <img src="...
  <title>...
</lockup>
```

Final Document

```html
<lockup>
  <img src="...
  <title>...
</lockup>
```

```html
<lockup>
  <img src="...
  <title>...
  <overlay>
    <progressBar value="0.60"
  </overlay>
</lockup>
```
Data Binding
Rules

Refinement of UI based on data states
Data Binding

Rules

Refinement of UI based on data states

Invisible element
Data Binding

Rules

Refinement of UI based on data states

Invisible element

Operates on sibling elements
We are not allowed to alter your code in any way. Please make adjustments.
We are not allowed to alter your code in any way. Please make adjustments.
Data Binding

Rules

<prototypes>
   <lockup>
      <img binding="@src:{imgURL}" />
      <title binding="textContent:{title}" />
      <placeholder tag="progress" />
      <rules>
         <specialize state="{progress}-greater-than:0">
            <overlay tag="progress">
               <progressBar binding="@value:{progress}" />
            </overlay>
         </specialize>
      </rules>
   </lockup>
</prototypes>

We are not allowed to alter your code in any way. Please make adjustments.
We are not allowed to alter your code in any way. Please make adjustments.
We are not allowed to alter your code in any way. Please make adjustments.
We are not allowed to alter your code in any way. Please make adjustments.
We are not allowed to alter your code in any way. Please make adjustments.
Data Binding

We cannot make code any smaller than it is on slide now without losing legibility.
Customizing Playback
Customizing Playback

Works with all playback experiences
Customizing Playback

Works with all playback experiences

Provide TVPlayer and UIViewController
Customizing Playback

Works with all playback experiences

Provide [TVPlayer] and [UIViewController]

Close analogue to TVMLKit JS APIs

<table>
<thead>
<tr>
<th>JS Player</th>
<th>TVPlayer</th>
</tr>
</thead>
<tbody>
<tr>
<td>JS Playlist</td>
<td>TVPlaylist</td>
</tr>
<tr>
<td>JS MediaItem</td>
<td>TVMediaItem</td>
</tr>
</tbody>
</table>
Customizing Playback

Works with all playback experiences

Provide `TVPlayer` and `UIViewController`

Close analogue to TVMLKit JS APIs

<table>
<thead>
<tr>
<th>JS Player</th>
<th>TVPlayer</th>
</tr>
</thead>
<tbody>
<tr>
<td>JS Playlist</td>
<td>TVPlaylist</td>
</tr>
<tr>
<td>JS MediaItem</td>
<td>TVMediaItem</td>
</tr>
</tbody>
</table>

Limited JavaScript Bridge
Customizing Playback

TVPlayer
Customizing Playback

TVPlayer

Public AVPlayer adaptor to Playback Pipeline
Customizing Playback

TVPlayer

Public AVPlayer adaptor to Playback Pipeline

Dispatch custom events to JavaScript
Customizing Playback

TVPlayer

Public AVPlayer adaptor to Playback Pipeline

Dispatch custom events to JavaScript

KVO properties for changes from JavaScript
Customizing Playback

TVPlayer

Public AVPlayer adaptor to Playback Pipeline

Dispatch custom events to JavaScript

KVO properties for changes from JavaScript

Sequential playlist
Customizing Playback

TVPlayer

Public AVPlayer adaptor to Playback Pipeline

Dispatch custom events to JavaScript

KVO properties for changes from JavaScript

Sequential playlist

```swift
// TVApplicationControllerDelegate

func player(for appController: TVApplicationController) -> TVPlayer? {
    return TVPlayer()
}
```
Customizing Playback

Playback User Interface

Implement your own playback UI
Customizing Playback

Playback User Interface

Implement your own playback UI

```swift
// TVInterfaceCreating

func playerViewController(for player: TVPlayer) -> UIViewController? {
    return MyPlaybackViewController.init(player: player)
}
```
Customizing Playback

Caveats

Handle “should” events yourself

shouldHandleStateChange (pause, fast forward, etc)

shouldChangeToMediaAtIndex
Customizing Playback

Caveats

Handle “should” events yourself

shouldHandleStateChange (pause, fast forward, etc)
shouldChangeToMediaAtIndex

Use AVContentKeySession for secure key loading
Customizing Playback

Caveats

Handle “should” events yourself

shouldHandleStateChange (pause, fast forward, etc)
shouldChangeToMediaAtIndex

Use AVContentKeySession for secure key loading

<table>
<thead>
<tr>
<th>Advances in HTTP Live Streaming</th>
<th>WWDC 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVContentKeySession Best Practices</td>
<td>WWDC 2018</td>
</tr>
</tbody>
</table>
Customizing Playback

Caveats

Handle “should” events yourself

shouldHandleStateChange (pause, fast forward, etc)
shouldChangeToMediaAtIndex

Use AVContentKeySession for secure key loading

Bring your own additional user interface

overlay
interactiveOverlay
Summary

Data Binding is now more powerful

Build your own native playback experience
More Information

https://developer.apple.com/wwdc18/238