Johannes Egger (HTL Vöcklabruck), Eva Weigl (HTL St. Pölten)

Verwenden von JavaScript in Vuforia Studio – Sammlung nützlicher Codebeispiele

Inhaltsverzeichnis

1 Einleitung

Die meisten hier gezeigten Beispiele verwenden eine zusätzliche JavaScript-Datei „fn.js“, die folgendermaßen in ein Projekt eingebunden werden kann:

1. Datei „fn.js“ herunterladen (z.B. von https://github.com/johannesegger/VuforiaStudioExtensions)
2. „fn.js“ als Ressource zum Projekt hinzufügen

3. „fn.js“ von einer .js-Datei (z.B. Home.js) laden

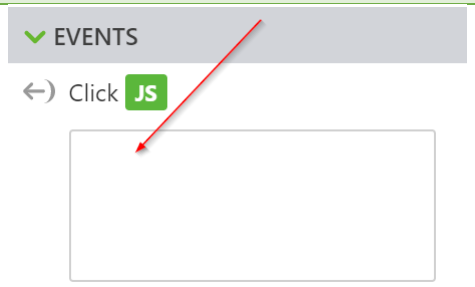
```
// $scope, $element, …

$scope.app.fn.loadResourceScript("Uploaded/fn.js")
```

1.1 Formatierung

```
JavaScript-Anweisungen in Home.js
```

```
AngularJS-Ausdrücke in Textfelder, die mit „JS“ markiert sind
```



Zurück nach oben

2 Ein Widget ausblenden oder wieder einblenden

```
fadeOut(me)
```
Ausblenden („me“ ist das aktuelle Widget)

```
fadeIn("schraube")
```
Einblenden

```
fadeOut("schraube", 3000)
```
Ausblenden über 3 Sekunden

Zurück nach oben

3 Sichtbarkeit eines Elementes steuern

```
show(me)
```
Anzeigen

```
hide("schraube")
```
Verstecken

```
toggleVisibility(me)
```
Sichtbarkeit umkehren

Zurück nach oben

4 Ein Widget blinken lassen

```
flash("screw")
```
Standardmäßig blinken

```
flash("screw", 4)
```
4 x blinken

```
highlight("screw")
```
Hervorheben durch Farbänderung

```
highlight("screw", 5)
```
Hervorheben durch 5 x Farbänderung

```
highlight("screw", 5, 40)
```
Hervorheben durch 5 x Farbänderung mit Hue-Farbwert von 40 (s. z.B. https://www.w3schools.com/colors/colors_hsl.asp)

Zurück nach oben

5 Ändern des Maßstabs

```
me.scale = 2
```
Sofortiges Ändern

```
animateFromTo(me, "scale", 1, 2, 2000, 50)
```
Animiertes Ändern von 1x auf 2x in 2 Sekunden in 50ms-Intervallen

Zurück nach oben

6 Ein Widget einmalig in x/y/z-Richtung bewegen

Diese Bewegung erfolgt nicht kontinuierlich (=animiert), man sieht nur das Ergebnis und NICHT die Animation, wie das Widget zum Ergebnis kommt.

```
changeProperty("screw", "x", 0.05)
```
Verschieben entlang der x-Achse um 0.05m

Zurück nach oben

7 Ein Widget einmalig um die x/y/z-Achse drehen

Diese Bewegung erfolgt nicht kontinuierlich (=animiert), man sieht nur das Ergebnis und NICHT die Animation, wie das Widget zum Ergebnis kommt.

```
rotate("screw", "x", 30)
```
Widget um 30° um die x-Achse rotieren

Zurück nach oben

8 Ein Widget animieren

Diese Bewegung erfolgt kontinuierlich (=animiert), man sieht die Animation, wie das Widget zum Ergebnis bewegt/gedreht wird.

ACHTUNG: Es darf mehrere gleichzeitige Bewegungen pro Widget geben, aber **nicht für dieselbe Widget-Eigenschaft**.

```
animateFromTo("schraube", "y", 1.2, 0.046)
```
Bewegung auf y-Achse von 1,2m nach 0,046m

```
animateBy("schraube", "y", 0.01, 50)
```
Endlose Bewegung auf der y-Achse um 0.01m pro 50ms

```
stopAnimation("schraube", "y")
```
Animation der Bewegung auf der y-Achse stoppen

```
toggleAnimateFromTo("schraube", "y", 1.2, 0.046)
```
Bewegung starten/stoppen

```
toggleAnimateBy("schraube", "y", 0.01, 50)
```
Endlose Bewegung starten/stoppen

```
animateRotation("schraube", "y", 1, 50)
```
Endlose Rotation auf der y-Achse um 1° pro 50ms

```
stopRotation("schraube", "y")
```
Rotation stoppen

```
toggleRotation("schraube", "y", 1, 50)
```
Endlose Rotation starten/stoppen

Zurück nach oben

9 Anzeigebild eines 3D-Images ändern

```
setImageSource("play-button", "pause.png");
```
„pause.png" muss vorher als Ressource hinzugefügt werden

Zurück nach oben

10 Die Farbe eines Widgets ändern

Eine Farbe wird mittels Rot-, Grün- bzw. Blau-Anteil definiert (jeweils Werte zwischen 0 und 255). Die Verwendung eines Farbauswahltools (z.B. https://colorpicker.me/) erleichtert die Verwendung.

```
setColor("schlitzschraube", 255, 0, 0)
```
Rot einfärben

```
resetColor("schlitzschraube")
```
Farbe zurücksetzen

Zurück nach oben

11 Die Farbe eines Widgets durchschalten

Hier kann man beliebig viele Farben vorab festlegen (Über Farbcode: rgba(0, 0, 255, 1) → rgba(rot, grün, blau, 1) Jeweils 0-255 siehe z.B. http://www.am.uni-duesseldorf.de/de/Links/Tools/farbtabelle.html). Wenn man die Funktion changeColor aufruft, wird nach dem Klick auf die nächste Farbe dieser Liste gewechselt. Der Eintrag "" bewirkt, dass man die Farbe auch wieder zurücksetzen kann.

```
let colors = ["rgba(255, 0, 0, 1)", "rgba(0, 255, 0, 1)", "rgba(0, 0, 255, 1)", ""]
let selectedColor = 0

$scope.changeColor = widgetName =>
{
  let widget = typeof widgetName == "string" ? $scope.view.wdg[widgetName] : widgetName
  widget.color = colors[selectedColor];
  selectedColor = (selectedColor + 1) % colors.length
}
```

```
changeColor("screw")
```

Zur nächsten Farbe wechseln

Zurück nach oben

12 Einzelne Schritte innerhalb einer Sequenz ansprechen

Folgendes Beispiel ändert je nach aktuellem Schritt die Farbe eines Modells. Wenn statt "newStep" "playstarted" verwendet wird, werden die Aktionen ausgeführt sobald ein Schritt abgespielt wird. Wenn z.B. Sound abgespielt werden soll, eignet sich "playstarted" besser, wenn z.B. ein Text eingeblendet werden soll, eignet sich womöglich "newStep" besser.

```
$scope.$on("newStep", () =>
{
    let currentStep = $scope.getCurrentStep("helicopter")
    if (currentStep == 1)
    {
        $scope.setColor("propeller", 255, 0, 0)
    }
    else if (currentStep == 2)
    {
        $scope.setColor("propeller", 0, 255, 0)
    }
    else if (currentStep == 3)
```

```
    {
        $scope.setColor("propeller", 255, 0, 255)
    }
    else
    {
        $scope.resetColor("propeller")
    }
})
```

Zurück nach above

13 Abspielen von Soundelementen

Durch Klick auf einen Button bzw. wenn ein bestimmter Schritt innerhalb einer Sequenz aufgerufen wird

```
let sounds = {}
$scope.playSound = (src, loop) =>
{
    let sound = new Audio(src)
    sound.loop = loop || false
    sound.play()
    sounds[src] = sound
}

$scope.stopSound = src =>
{
    if (sounds[src])
    {
        sounds[src].pause()
        delete sounds[src]
    }
}
```

```
playSound("http://soundbible.com/grab.php?id=1587&type=mp3")
stopSound("http://soundbible.com/grab.php?id=1587&type=mp3")
playSound("app/resources/Uploaded/sound.mp3", true)
stopSound("app/resources/Uploaded/sound.mp3")
```

Musikdatei herunterladen und abspielen (Achtung: Kann zu Verzögerung führen)
Wiedergabe stoppen
Musikdatei, die als Ressource hinzugefügt wurde, abspielen und endlos wiederholen
Wiedergabe stoppen

Zurück nach oben

14 Einbetten von Videos

Einbetten funktioniert denke ich nicht, ev. kann „Öffnen einer Webseite“ als Ersatz verwendet werden.

Zurück nach oben

15 Öffnen einer Webseite

```
window.open("https://www.youtube.com/watch?v=T6rFBN6j_Bc")
```

Zurück nach oben

16 Laden unterschiedlicher Bauteile bzw. Baugruppen in ein Modell

Oder Laden in Abhängigkeit einer anderen Auswahl, d.h. durch die Auswahl einer bestimmten Konfiguration werden unterschiedliche Modelle geladen (Beispiel Dusche: es gibt sie in runder oder eckiger Ausführung)

Achtung: Nach dem Laden muss ev. die richtige Sequenz manuell ausgewählt werden.

```
let models = [
    "grundplatte_illustrate_sequenzen.pvz",
    "htl_rc-car_Raspby_gesamt.pvz"
]
let modelIndex = 0;
$scope.nextModel = widgetId =>
{
    modelIndex = (modelIndex + 1) % models.length;
    $scope.setModel(widgetId, models[modelIndex]);
}
```

`nextModel(me)`	*Nächstes Modell aus Liste anzeigen*

Zurück nach oben

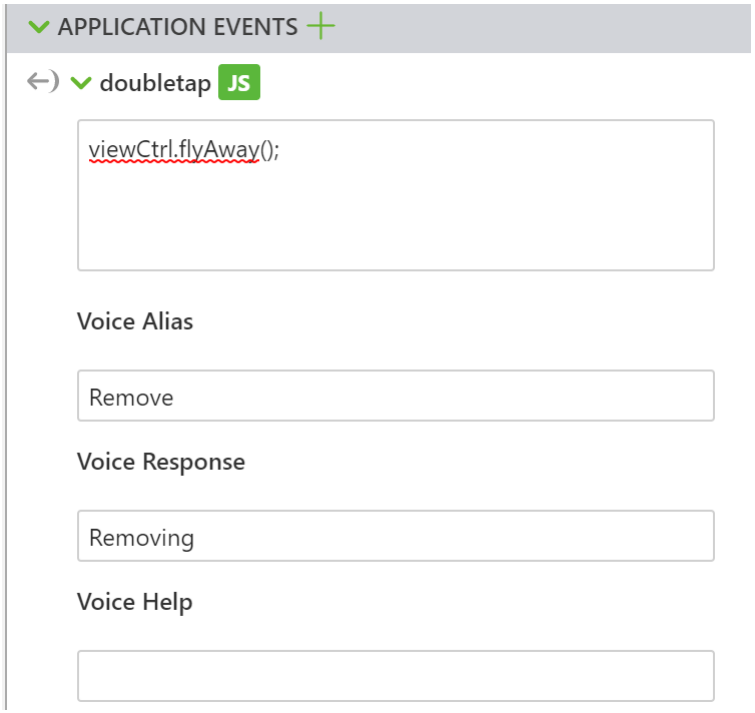
17 Ein Modell mittels Sprachsteuerung/Gesten auseinanderbauen

```
let selectedWidget
$scope.setWidget = widgetId =>
{
    if (selectedWidget == undefined)
    {
        selectedWidget = widgetId
        $scope.setColor(widgetId, 255, 0, 0)
    }
    else
    {
        selectedWidget = undefined
        $scope.resetColor(widgetId)
    }
}
let fly = (axis, offset, offsetFactor) =>
{
    if (selectedWidget === undefined)
    {
        console.error("No model selected")
    }
    else
    {
        axis = axis !== undefined ? axis : "z"
        offset = offset !== undefined ? offset : 1
        let propertyValue = $scope.getProperty(selectedWidget, axis)
        $scope
            .animateFromTo(selectedWidget, axis, propertyValue, propertyValue + offset * offsetFactor)
            .then(() =>
            {
                $scope.resetColor(selectedWidget)
                selectedWidget = undefined
            })
    }
}
$scope.flyAway = (axis, offset) => fly(axis, offset, 1)
$scope.flyBack = (axis, offset) => fly(axis, offset, -1)
```

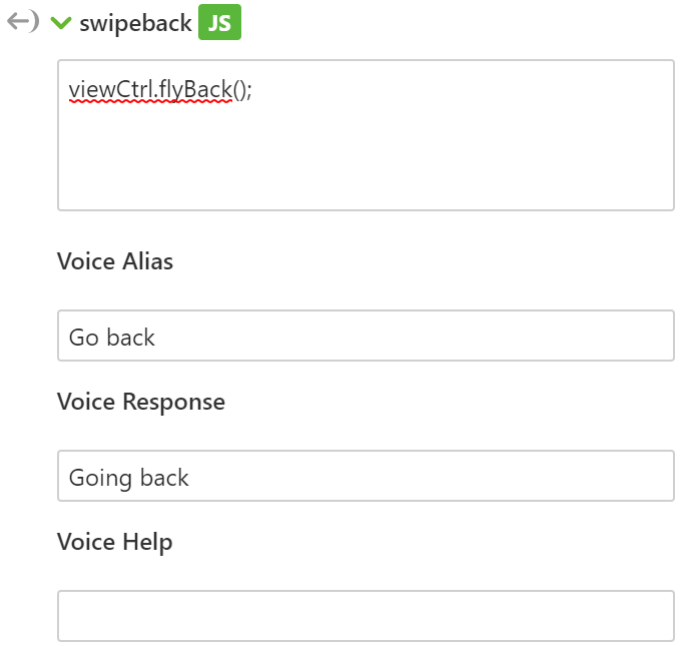
Wichtig: **Jedes** Modell/Modellelement, das entfernt werden soll, muss in dem zum Modell/-element zugehörigen Click JS Feld folgendes stehen haben:

```
setWidget(me)
```

Außerdem muss in einem Application Event (z.B. doubletap) folgendes für das Entfernen eines Modells/-elements festgelegt werden:



Und folgendes für das Zurückführen zur ursprünglichen Position:



Achtung: Das Element muss zuerst ausgewählt werden (es wird rot eingefärbt), bevor eine Aktivität ausgeführt werden kann!

18 Anmerkungen

18.1 Eigenschaften eines Modells:

Eigenschaft	Beschreibung	Wertebereich bzw. Standardwert
translucent	Macht das Element lichtdurchlässig.	True oder false
opacity	Durchsichtigkeit (0 = vollkommen durchsichtig und 1 = vollkommen undurchsichtig)	Zwischen 0.0 und 1.0
occlude	Verdecken: Wenn diese Option auf "true" festgelegt ist, ist die Geometrie eines Widgets unsichtbar, aber gleichzeitig werden alle weiteren Augmentationen ausgeblendet, die in der 3D-Szene dahinter liegen.	True oder false
visible	Sichtbarkeit (Ein-/ausblenden)	True oder false
rx	Rotation um x-Achse	0
ry	Rotation um y-Achse	0
rz	Rotation um z-Achse	0
x	x-Koordinate	0
y	y-Koordinate	0
z	z-Koordinate	0
scale	Skalierung	Zwischen 0.0 und 1.0
src	Pfad zur hinterlegten PVZ Datei	app/resource/uploaded/….pvz
sequence	Welche Sequenz ist hinterlegt	
sequenceList	Welche Sequenzen existieren für dieses Modell	
steps	Anzahl der Schritte in dieser Sequenz	Zahl
currentStep	Welcher Schritt wird gerade abgespielt	Zahl zwischen 0 und steps

Zurück nach oben

18.2 Eigenschaften eines Modellelements:

Eigenschaft	Beschreibung	Wertebereich bzw. Standardwert
color	Farbe des Modellelements	'' oder ""
translucent	Macht das Element lichtdurchlässig.	True oder false
opacity	Durchsichtigkeit (0 = vollkommen durchsichtig und 1 = vollkommen undurchsichtig)	Zwischen 0.0 und 1.0
occlude	Verdecken: Wenn diese Option auf "true" festgelegt ist, ist die Geometrie eines Widgets unsichtbar, aber gleichzeitig werden alle weiteren Augmentationen ausgeblendet, die in der 3D-Szene dahinter liegen.	True oder false
visible	Sichtbarkeit (Ein-/ausblenden)	True oder false
rx	Rotation um x-Achse	0
ry	Rotation um y-Achse	0
rz	Rotation um z-Achse	0
x	x-Koordinate	0
y	y-Koordinate	0
z	z-Koordinate	0
scale	Skalierung	Zwischen 0.0 und 1.0

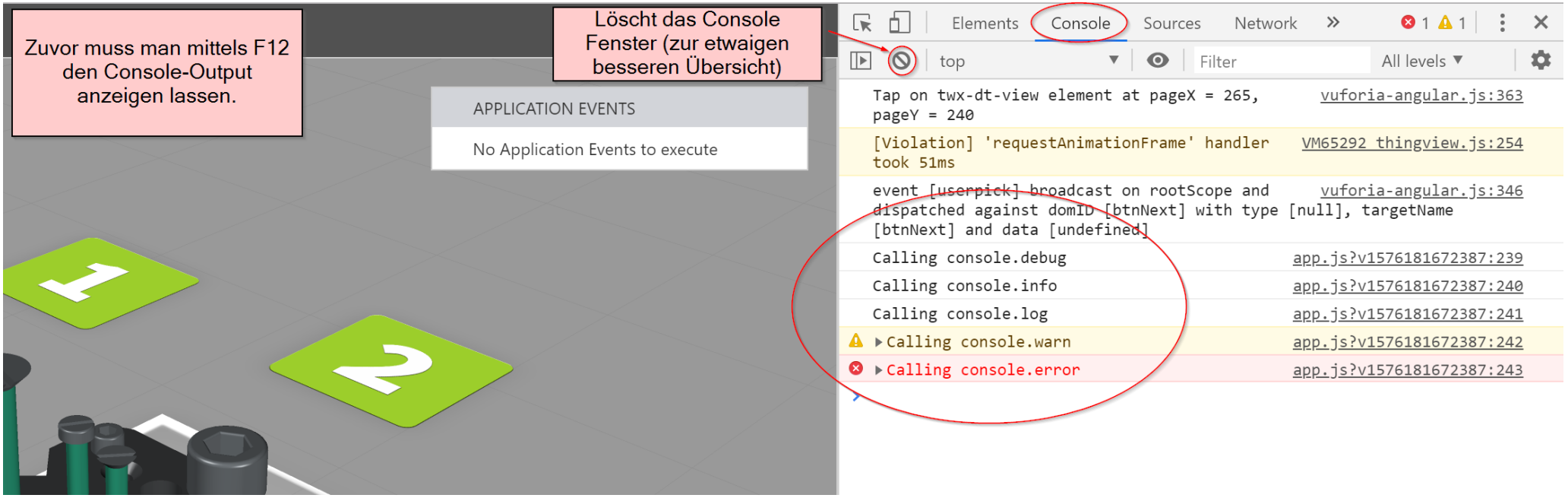
Zurück nach oben

18.3 Debugging-Möglichkeit – Wie erkennt man einen Fehler?

Es gibt mehrere Befehle, um etwas auf der Console ausgeben zu lassen. Diese Befehle kann man im Home.js z.B. in einer Funktion einbauen. Es gibt folgende Varianten:

```
console.debug("Calling console.debug");
console.info("Calling console.info");
console.log("Calling console.log");
console.warn("Calling console.warn");
console.error("Calling console.error");
```

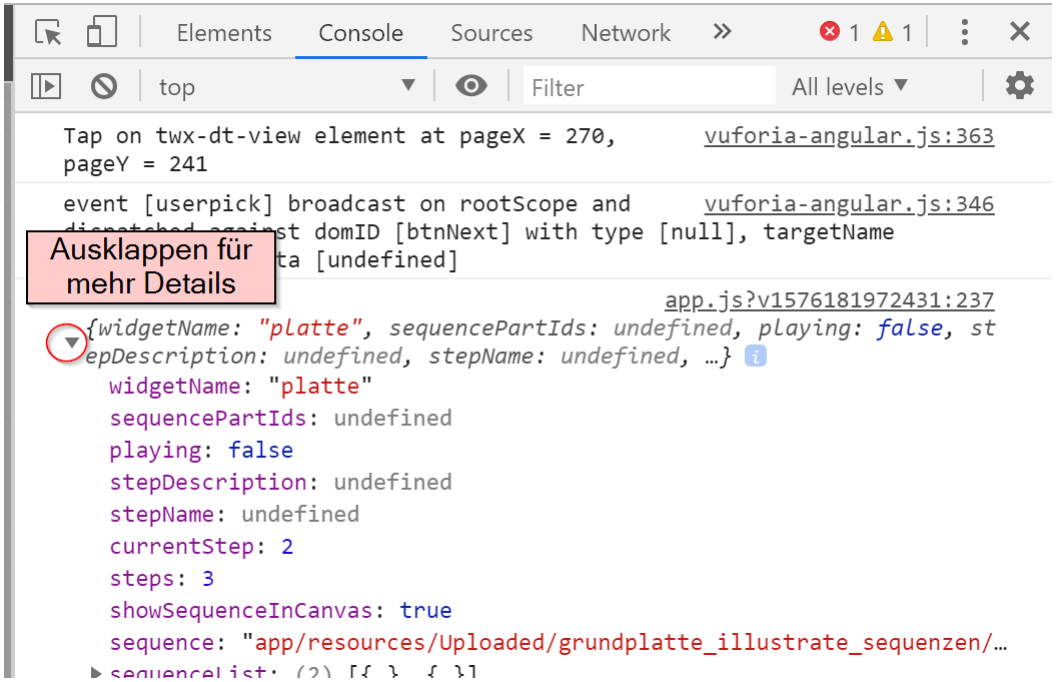
In der Preview-Ansicht der Experience drückt man dann die Taste F12 und erhält den zugehörigen Output:



Man kann aber auch Objekte wie z.B. ein Modell debuggen:

```
console.debug(widget)
```

Das resultiert in folgender Anzeige:



Zurück nach oben

18.4 Funktionsaufruf aus Home.js in Kontrast zum Aufruf aus dem Click Event

Wenn man im Model Event im JS etwas einträgt und damit etwas auslösen möchte, z.B. die Sichtbarkeit von einem Element steuern, dann schaut die Syntax dazu ein wenig anders aus:

In diesem Fenster werden AngularJS Expressions (s. https://docs.angularjs.org/guide/expression) eingefügt, die in einem $scope-Context ausgewertet werden, deshalb muss $scope nicht extra vorangestellt werden – s. https://docs.angularjs.org/api/ng/type/$rootScope.Scope#$eval

Zurück nach oben

18.5 Bei einem Application Event eine Funktion aufrufen (HoloLens!)

Um bei einem Application Event eine Funktion aufzurufen, die im Home.js gespeichert ist:

Die Application Events werden in einem anderen Kontext ausgeführt. Das, was ansonsten `$scope` heißt, heißt in den Application Events `viewCtrl`. In dem Textfeld kann wie bei den Model Events eine AngularJS Expression eingetragen werden.

Zurück nach oben

18.6 Eine Funktion wiederholen

https://docs.angularjs.org/api/ng/service/$interval

```
$interval(fn, delay, [count], [invokeApply], [Pass]);
```

The `fn` function is executed every `delay` milliseconds.

The return value of registering an interval function is a promise.

```
cancel([promise]);
```

Cancels a task associated with the `promise`.

```
Beispiel:
var stopTime = $interval(updateTime, 1000);

$interval.cancel(stopTime);
```

Eventuell auch folgende (Standard-)Funktionen, bei denen aber in den Callbacks mit $scope.$apply o.Ä. verwendet werden muss, falls Properties verändert werden:

```
var stopTime = setInterval(updateTime, 1000);

clearInterval(stopTime);
```

Zurück nach oben

18.7 Verfügbare Services in AngularJS

- $scope - Bindeglied zwischen AngularJS Controller (Home.js) und View
  - Funktionen, die an $scope angehängt werden, können von View aus aufgerufen werden
- $element - Referenz auf das HTML-Element
  - Könnte man ev. verwenden, um zusätzliche 3D-Elemente anzuzeigen. Für 2D Experiences ev. etwas mehr Einsatzmöglichkeiten, aber im Allgemeinen für Vuforia-Studio-Anwendungen uninteressant
- $attrs - HTML-Attribute von $element
  - Für Vuforia-Studio-Anwendungen uninteressant
- $injector - Dependency management
  - https://docs.angularjs.org/api/auto/service/$injector
  - Für Vuforia-Studio-Anwendungen uninteressant
- $sce - Strict Contextual Escaping
  - https://docs.angularjs.org/api/ng/service/$sce
  - Für Vuforia-Studio-Anwendungen uninteressant
- $timeout – Befehle zeitverzögert ausführen
  - https://docs.angularjs.org/api/ng/service/$timeout
- $http – Externe Daten laden bzw. Daten auf externen Server laden
  - https://docs.angularjs.org/api/ng/service/$http
  - Achtung: Asynchrone Funktion, d.h. async/await verwenden und Properties mit $scope.$apply updaten
- $ionicPopup – Message box, die dem Benutzer Nachrichten anzeigen kann bzw. Daten abfragen kann
  - Dokumentation: https://support.ptc.com/images/cs/articles/2018/11/1542884896qGz5/VuforiaAngularJS_1.pdf ab Folie 13
  - Achtung: Asynchrone Funktion, d.h. async/await verwenden und Properties mit $scope.$apply updaten
- $ionicPopover – Ähnlich wie $ionicPopup, z.B. für Anzeige von Details
  - Funktioniert nicht korrekt (zumindest bei 3D Experience)

Weiters sind lt. Quellcode aktuell folgende Services verfügbar: $interval, $stateParams, $location, $rootScope, tml3dRenderer

Zurück nach oben