# Using App Widget

# App widget and badge

The Amazon Home API set allows HTML5 web apps to access native home screen functionality on the Fire phone. These APIs provide ways for web apps to manipulate app widget on the home screen. Additionally, you can update icon badge numbers and respond to events that are created from interacting with the app widget.

## Terminology

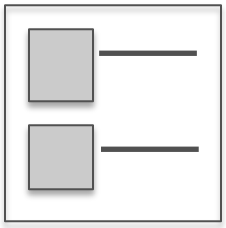
This section provides an introduction to the common terms used in implementing the Amazon Home API with Amazon Web View. All of the Home API's are accessed through the *amazonHomeManager* object.

### App widget

The app widget is displayed under the large application icon in the Amazon home screen carousel. This widget can be either a Grid View or a List View. The views can contain touchable content, which opens the app with an Android Intent that contains information about what was touched. This widget is accessed through the setHeroWidget function as referred to later in the document.

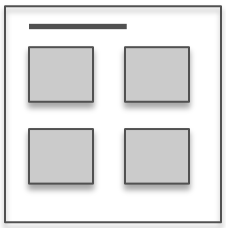
### List View

An app widget in List View format contains objects in a list. The objects can contain both images and text. List Views have several different visual styles available.



### Grid View

An app widget in Grid View format contains objects in a grid. The objects can contain only image thumbnails, not text. You can optionally add a "play button" with duration to the icons. For example, you can provide thumbnails of videos, with a video duration and play icon, so that a user can play a video.



### Badge Number

The badge number is a numerical icon displayed on the home screen anywhere the application icon is displayed. This is useful for displaying if the application needs to notify the user of new information or events waiting for them. The Home APIs allow you to update the badge number of the application.

## Setup

For instructions on setting up your environment please refer to the *"Web Application Developer's Guide"*.

## API Reference

### amazonHomeManager

The `amazonHomeManager` object manages several areas of functionality:

- app widget manipulation
- badge icon manipulation

Before using the Home API, check to ensure that the `amazonHomeManager` object exists, in case you are on a platform that does not support this functionality. Only use the Home API after handling the `amazonPlatformReady` event in web applications.

The following code snippet checks for the existence of the `amazonHomeManager` in web applications:

```
document.addEventListener('amazonPlatformReady', function () {
      if (window.amazonHomeManager){
            //run API code here
      }
});
```

## Public Methods

### updateNumericBadge

This method sets the numerical badge of the application.

`window.amazonHomeManager.updateNumericBadge(number)`

| Argument | Description | Possible values |
|---|---|---|
| number | Integer to display on the badge. | Integer. Setting this to 0 or below removes the badge completely.<br>If you set the badge to a number above 99, the badge displays “99+” |

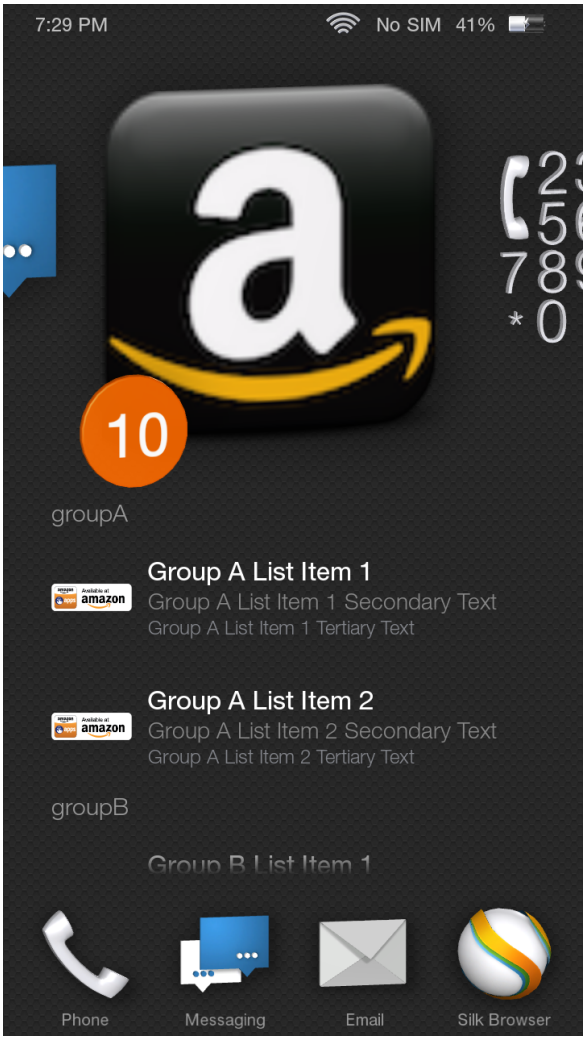
The following snippet sets the badge to 10, 99+, and removes the badge:

```
//Set the badge to 10
window.amazonHomeManager.updateNumericBadge(10);

//set the badge to 99+
window.amazonHomeManager.updateNumericBadge(150);

//remove the badge
window.amazonHomeManager.updateNumericBadge(0);
```

The following screen capture shows how a badge set to “10” appears on a device:




### updateWidget

This method specifies an app widget for the application.

```
window.amazonHomeManager.updateWidget(appWidget)
```

| Argument | Description | Possible values |
|---|---|---|
| appWidget | Object containing the information for the AppWidget. | A valid app widget object, one of three types: List, Grid, and Empty. |

### setHeroCallback

This method sets the JavaScript handler to be called when the app widget is touched, and sends the relevant callbackData to that function.

```
window.amazonHomeManager.setHeroHandler(widgetHandler)
```

| Argument | Description | Possible values |
|---|---|---|
| widgetHandler | A JavaScript function which takes a string parameter to call back into when the AppWidget item is called and contains the openAppOnTouch Boolean as true. | JavaScript function that takes a string parameter. |

```
//Set the badge to an anonymous callback function
window.amazonHomeManager.setHeroHandler(function(data) {
        console.log("received data in JS: " + data);
});
```

## Objects

### AppGroup objects

If the AppWidget is a "list" or "grid" type, it must have a group property that contains a valid AppWidgetListGroup or AppWidgetGridGroup:

| Property | Description | Possible values |
|---|---|---|
| name | The group name. **(Required)** | Can be any text to be shown on the UI to represent the group. |
| listItems | Array of valid ListItem objects. **(Required if AppWidgetObject is of type "list")** | If the AppObject type is "list" this must be an array of valid ListItem object described below. |
| gridItems | Array of valid GridItem objects. **(Required if AppWidgetObject is of type "grid")** | If the AppObject type is "grid" this must be an array of valid GridItem object described below. |

### AppWidgetObject

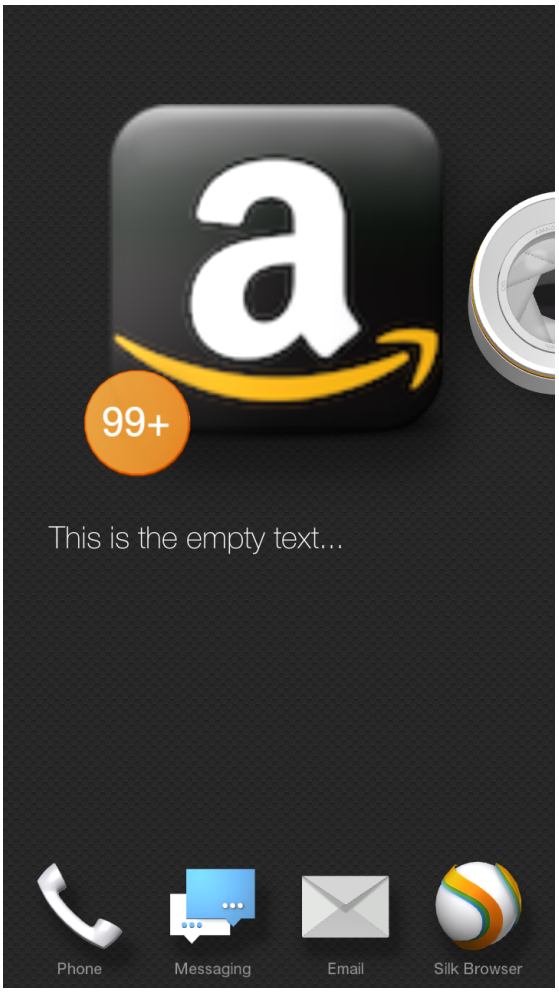
The AppWidgetObject has three valid type values: “list”, “grid”, or “empty”.

| Property | Description | Possible values |
|---|---|---|
| type | The app widget object type.**(Required)** | Valid types are: “list”, “grid”, and “empty” |
| label | The text to be shown in an app widget object of type “empty”.**(Optional)** | Any text value you want shown when the app widget is empty, can be blank. |
| groups | An array of AppWidgetListGroups, or AppWidgetGridGroups. **(Required if object type “list” or “grid”)** | If the AppWidget is of type “grid” or “list” then it must contain a groups property which contains an array of the relevant group objects, described below. |

The following snippet displays an empty AppWidgetObject with a label:

```
//Create an empty AppWidgetObject with a label
var emptyAppObject = {
    type: "empty",
    label: "This is the empty text...",
};
window.amazonHomeManager.updateWidget(emptyAppObject);
```

The following sample code shows how an empty AppWidgetObject appears on a device:




### ListItem Object

If the AppWidgetObject is a “list” type, it must contain a group of ListItem:

| Property | Description | Possible values |
|---|---|---|
| primaryText | Primary text for the list item. **(Required)** | Text shown in the primary position of the list item. Can be truncated depending on visual style. |

| | | |
|---|---|---|
| visualStyle | The visual style of the list item. **(Optional)** | Can be any of the following types:<br>"DEFAULT" (default value if none set)<br>"PEEKABLE"<br>"SHOPPING"<br>"SHOPPING_RETAIL"<br>"SIMPLE"<br>"WRAPPED_TEXT" |
| img | URL of the primary image for the list icon. **(Optional)** | URL for an image. The image can be scaled depending on the visualStyle. Because the URL can be a local file URI, set its permissions to be readable by the application. |
| secondaryImg | URL of the secondary image for the list icon. **(Optional)** | URL for an image. The image can be scaled depending on the visualStyle.<br><br>Because the URL can be a local file URI, set its permissions to be readable by the application. |
| tertiaryImg | URL of the tertiary image for the list icon. **(Optional)** | URL for an image. The image can be scaled depending on the visualStyle. Because the URL can be a local file URI, set its permissions to be readable by the application.. |

| | | |
|---|---|---|
| quaternaryImg | URL of the quaternary image for the list icon. **(Optional)** | URL for an image. The image can be scaled depending on the visualStyle. Because the URL can be a local file URI, set its permissions to be readable by the application.. |
| secondaryText | Secondary text for the list item. **(Optional)** | Text shown in the secondary position of the list item. This text can be truncated depending on visual style. |
| tertiaryText | tertiary text for the list item. **(Optional)** | Text shown in the tertiary position of the list item. This text can be truncated depending on visual style. |
| quaternaryText | Quarternary text for the list item. **(Optional)** | Text shown in the quaternary position of the list item. This text can be truncated depending on visual style. |
| openAppOnTouch | Boolean value of whether or not the list item should open the application on touch **(Optional)** | true (open on touch)<br>false (do not open on touch)<br>Defaults to true, if not specified. |
| callbackData | The string to be provided to the callback function in JavaScript when the list item is touched. **(Optional)** | String provided to the JavaScript callback for the AppWidget. Used by callback function to determine which list item was touched. Defaults to empty string, if not specified. |
| starRating | Float value to indicate a rating for the list item. **(Optional)** | Can be any float between 0 and 5. May only show stars in certain visual styles. |

The following sample code creates a AppWidgetList with multiple styles:

```
//Create a AppWidgetList with multiple visual styles:
var listAppWidgetObject = {
    type: "list",
    groups: [  //group array
        {
            name: "groupA", // group name *required
            listItems: [
                {
                    visualStyle: "DEFAULT",
                    img: "http://test.com/img.png",
                    secondaryImg: "http://test.com/img.png",
                    tertiaryImg: "http://test.com/img.png",
                    quaternaryImg: "http://test.com/img.png",
                    openAppOnTouch: true,
                    primaryText: "Visual Style: DEFAULT",
                    secondaryText: "Secondary Text",
                    tertiaryText: "Tertiary Text",
                    quaternaryText: "Quaternary Text",
                    callbackData: "DEFAULT Visual Callback",
                    starRating: 3.5,
                },
                {
                    visualStyle: "PEEKABLE",
                    img: "http://test.com/img.png",
                    secondaryImg: "http://test.com/img.png",
                    tertiaryImg: "http://test.com/img.png",
                    quaternaryImg: "http://test.com/img.png",
                    openAppOnTouch: true,
                    primaryText: "Visual Style: PEEKABLE",
                    secondaryText: "Secondary Text",
                    tertiaryText: "Tertiary Text",
                    quaternaryText: "Quaternary Text",
                    callbackData: "PEEKABLE Visual Callback",
                    starRating: 3.5,
                },
                {
                    visualStyle: "SHOPPING",
                    img: "http://test.com/img.png",
                    secondaryImg: "http://test.com/img.png",
                    tertiaryImg: "http://test.com/img.png",
                    quaternaryImg: "http://test.com/img.png",
                    openAppOnTouch: true,
                    primaryText: "Visual Style: SHOPPING",
                    secondaryText: "Secondary Text",
                    tertiaryText: "Tertiary Text",
                    quaternaryText: "Quaternary Text",
                    callbackData: "SHOPPING Visual Callback",
                    starRating: 3.5,
                },
```

```
                {
                    visualStyle: "SHOPPING_RETAIL",
                    img: "http://test.com/img.png",
                    secondaryImg: "http://test.com/img.png",
                    tertiaryImg: "http://test.com/img.png",
                    quaternaryImg: "http://test.com/img.png",
                    openAppOnTouch: true,
                    primaryText: "Visual Style: SHOPPING_RETAIL",
                    secondaryText: "Secondary Text",
                    tertiaryText: "Tertiary Text",
                    quaternaryText: "Quaternary Text",
                    callbackData: "SHOPPING_RETAIL Callback",
                    starRating: 3.5,
                },
                {
                    visualStyle: "SIMPLE",
                    img: "http://test.com/img.png",
                    secondaryImg: "http://test.com/img.png",
                    tertiaryImg: "http://test.com/img.png",
                    quaternaryImg: "http://test.com/img.png",
                    openAppOnTouch: true,
                    primaryText: "Visual Style: SIMPLE",
                    secondaryText: "Secondary Text",
                    tertiaryText: "Tertiary Text",
                    quaternaryText: "Quaternary Text",
                    callbackData: "SIMPLE Visual Callback",
                    starRating: 3.5,
                },
                {
                    visualStyle: "WRAPPED_TEXT",
                    img: "http://test.com/img.png",
                    secondaryImg: "http://test.com/img.png",
                    tertiaryImg: "http://test.com/img.png",
                    quaternaryImg: "http://test.com/img.png",
                    openAppOnTouch: true,
                    primaryText: "Visual Style: WRAPPED_TEXT",
                    secondaryText: "Secondary Text",
                    tertiaryText: "Tertiary Text",
                    quaternaryText: "Quaternary Text",
                    callbackData: "WRAPPED_TEXT Visual Callback",
                    starRating: 3.5,
                }
            ],
        },


    ],
}
```

```
window.amazonHomeManager.updateWidget(listAppWidgetObject);
```

The following screen capture shows what the sample code looks like on a device:


### GridItem Object

If the AppWidgetObject is a "grid" type, it must contain a group of GridItem objects:

| Property | Description | Possible values |
|---|---|---|
| img | URL of the thumbnail for the grid object.**(Required)** | The main thumbnail for the grid object. |

| | | |
|---|---|---|
| openAppOnTouch | Boolean value of whether or not the grid item should open the application on touch**(Optional)** | true (open on touch)<br>false (do not open on touch)<br>Defaults to true if not specified. |
| callbackData | The string to be provided to the callback function in JavaScript when the grid item is touched. **(Optional)** | String provided to the JavaScript callback for the AppWidget. Used by callback function to determine which grid item was touched. Defaults to empty string, if not specified. |
| playButton | Boolean if the playbutton should be shown on this grid object.**(Optional)** | true (to display playbutton<br><br>false (to not display playbutton)<br><br>Defaults to false if not specified. |
| playDuration | String of the duration to be shown next to the play button.**(Optional)** | Shown if playButton is set to true. Displays a duration next to the play button, usually this is 5 characters, such as: 23:45 |

The following sample code creates a AppWidgetGrid with Play buttons:

```
//Create a AppWidgetGrid with some play buttons
var gridAppWidgetObject = {
    type: "grid",
    groups: [  //group array
        {
            name: "groupA", // group name *required
            gridItems: [
                {
                    img: "http://test.com/img.png",
                    openAppOnTouch: true,
                    callbackData: "group A grid item 1",
                    playButton: true, //defaults to false
                    playDuration: "23:45",
                },
                {
                    img: "http://test.com/img.png",
                    openAppOnTouch: true,
                    callbackData: "group A grid item 2",
                },
                 {
                    img: "http://test.com/img.png",
                    openAppOnTouch: true,
                    callbackData: "group A grid item 3",
                    playButton: true, //defaults to false
                    playDuration: "23:45 test test test",
                },
                {
                    img: "http://test.com/img.png",
                    openAppOnTouch: true,
                    callbackData: "group A grid item 4",
                    playButton: true, //defaults to false
                },
                {
                    img: "http://test.com/img.png",
                    openAppOnTouch: true,
                    callbackData: "group A grid item 5",
                },
                {
                    img: "http://test.com/img.png",
                    openAppOnTouch: true,
                    callbackData: "group A grid item 6",
                }

            ],
        },
        {
            name: "groupB", // group name *required
            gridItems: [
                {
```

```
                    img: "http://test.com/img.png",
                    openAppOnTouch: true,
                    callbackData: "group B grid item 1",
                },
                {
                    img: "http://test.com/img.png",
                    openAppOnTouch: true,
                    callbackData: "group B grid item 2",
                },
                 {
                    img: "http://test.com/img.png",
                    openAppOnTouch: true,
                    callbackData: "group B grid item 3",
                },
                {
                    img: "http://test.com/img.png",
                    openAppOnTouch: true,
                    callbackData: "group A grid item 4",
                },
                {
                    img: "http://test.com/img.png",
                    openAppOnTouch: true,
                    callbackData: "group B grid item 5",
                },
                {
                    img: "http://test.com/img.png",
                    openAppOnTouch: true,
                    callbackData: "group B grid item 6",
                }

            ],
        },


    ],
}

window.amazonHomeManager.updateWidget(gridAppWidgetObject);
```

The following screen capture shows what the sample code looks like on a device:


## Lifecycle of the setHeroHandler

An HTML5 app's app widget handler function is called asynchronously and can occur at any point after calling the `amazonPlatformReady` event in web applications and `deviceready` in Cordova application. Structure your code to handle the callback gracefully, whenever it occurs in the app's lifecycle.

Interacting with the app widget can either start an app or resume an in-background app. In both situations, the application UI might be shown in a state before calling the app widget handler. Be aware of this scenario so that you can handle changing from the state of the UI when the app is opened to the appropriate state of the UI

after calling the app widget handler function.

To handle this situation, display an intermediate UI state before the `amazonPlatformReady` event, and make sure that the `setHeroHandler` function is set to a callback before making any UI changes to the application. To help facilitate this behavior the `amazonHomeManager`, sets the `window.amazonHomeManager.lastLaunchCallbackData` property which is available after `amazonPlatformReady`:

- If the app was started from a normal touch to the icon, the property will be null
- If the app was started from app widget interaction, the value will be set to the launch data.

The `lastLaunchCallbackData property` will be available as soon as the `amazonPlatformReady` event is called. This variable is only updated on initial app launch and not from a backgrounded state. The possible value for this is either `null` (the app was started from a normal click and not an app widget interaction), or a String containing the `callbackData` of the app item, which was interacted with.

The following code sample shows how to use the `lastLaunchCallbackData property`:

```
document.addEventListener('amazonPlatformReady', function () {

var lastData = window.amazonHomeManager.lastLaunchCallbackData;

if (!lastData) {
  // lastData is null so we were
  // started from normal home click
} else {
  //App was started from AppWidget, callback data in the variable
  console.log("Last data was: " + lastData);
}});
```