

Chapter 5: Handling Events

In the world of programming, events are actions that take place within an application. Normally these events are triggered by the end user and can include things like mouse clicks, mouse drags, keyboard actions, and others, but can also include the sending and receiving of data, component modification, and others.

Not all web browsers handle events in the same way. No surprise there right? Dojo normalizes these events according to the W3C standard so that you don't have to worry about how different browsers handle various events. The `dojo.connect()` method is used to connect events that happen in your application to the code that will run in response to these events.

The ArcGIS Server API for JavaScript is an asynchronous API that connects event listeners with event handlers. So the obvious question you may be asking now is what is an asynchronous API and what does it mean in the context of my mapping application? Basically an asynchronous API just means that not every method that you call on an object will immediately return results. This is often the case with geoprocessing tasks that take awhile to execute and return results. In this case, when the result has been generated on the server it is returned to the application and executed by a handling function whose specific task is to process the results when they become available.

The diagram below illustrates this process. Listeners are responsible for monitoring the application for these events and then triggering a handler function that responds to the event. Multiple events can be registered to the same listener.

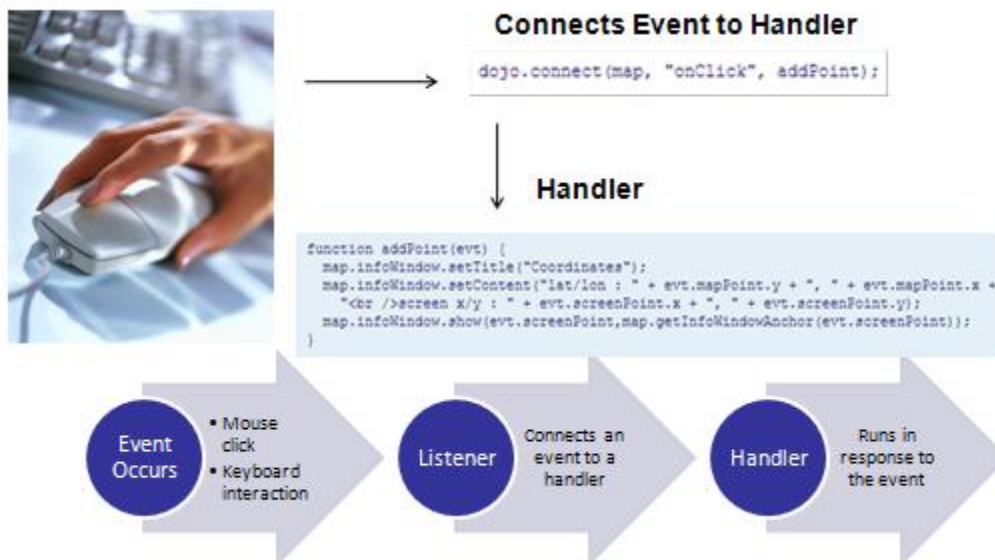


Figure 33: Connecting Events to Handlers

Connecting Events to Handlers

As you'll recall, the ArcGIS Server JavaScript API is built on top of Dojo. With Dojo, events are registered to handlers through the `dojo.connect()` method. This method takes three parameters. Let's take a look at the code below to get a better understanding of how events are registered.

```
dojo.connect (map, "onMouseMove", showCoordinates);
```

Object Event JavaScript function that handles response to the event

We call the `dojo.connect()` method with parameters including 'map', 'onMouseMove', and 'showCoordinates'. The first two parameters indicate the object and the event that we would like to register. In this case that means we are registering the 'onMouseMove' event found on the Map object. This event is fired every time the mouse moves within the confines of the map. The final parameter, 'showCoordinates', indicates the listener that we are registering this event with. Therefore, each time the 'onMouseMove' event on the Map object is fired, it will trigger the `showCoordinates()` function which will run and report the current extent of the map. Although the events and the handlers they are registered to will change depending upon your circumstance, the method of registration is the same.

Disconnecting Events and Handlers

You can also assign a variable to the connection as seen in the code example below.

```
var mouseMvHandler = dojo.connect (map, "onMouseMove", showCoordinates);
```

This comes in handy later when you want to release the handler when it is no longer needed in your application or when the application closes. It is important to remove the connections between events and listeners when your application closes. Failure to do so can cause memory leaks. To remove an event listener you can call `dojo.disconnect()` when the `Map.onUnload()` event occurs. The `Map.onUnload()` event is fired when the page is refreshed or closed. A code example below illustrates this process of disconnecting an event from a handler.

```
var mouseMvHandler = dojo.connect (map, "onMouseMove", showCoordinates);

var unloadConnect = dojo.connect (map, "onUnload", unloadHandler);

function unloadHandler () {
    ...
    dojo.disconnect (mouseMvHandler);
}
```

There are many different events that are available on objects in the ArcGIS Server API for JavaScript. However, it is important to keep in mind that you do not have to register every event with a listener. Only those events that are necessary for your application should be registered. When an event occurs that hasn't been registered with a listener the event is simply ignored.

Passing Information to an Event

Information is passed into the events in the form of parameters. For instance, the `Map.onClick(event)` event is passed an Event object which contains information specific to the event that has occurred. These Event objects are dynamic in that they contain different information depending upon the event that was fired. For instance, the Event object passed into `Map.onClick()` contains information about the x,y location of the point that was clicked on the map. Not all events are passed an Event object. For instance, the `Map.onLayerAdd(layer)` is passed a Layer object containing information about the layer that was added to the map.

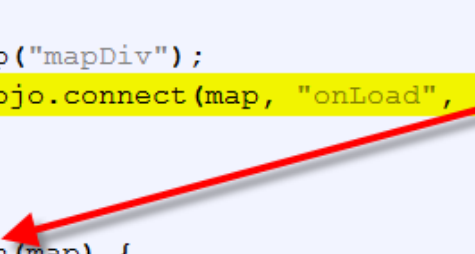
Common ArcGIS Server API for JavaScript Events

Many of the classes in the JavaScript API have events, but certain classes like Map have more commonly used events. For instance, the Map class has all of the events you see on the object model diagram below plus more. Most of these events fall within several groups related to zooming, panning, and layer events. There are also events triggered when the map extent changes, a layer has been added to the map, and when the user has clicked the map. Each of these events can have a handler that responds to having been fired.

Map.onLoad and Map.onUnload

We've already discussed `Map.onUnload()` which is fired anytime the page refreshes or is closed. The `Map.onLoad()` event is fired after an initial layer has been added to the map. The graphics layer which holds graphics for your application is also loaded at this time. At this point you can interact with the map in your code.

```
function init() {  
    var map = new esri.Map("mapDiv");  
    mapOnLoad_connect = dojo.connect(map, "onLoad", configNavigation);  
    map.addLayer(...);  
}  
  
function configNavigation(map) {  
    map.disableMapNavigation();  
}
```



Map Events Related to Clicking the Map

The `Map.onClick()` and `Map.onDbClick()` events are fired when the user clicks or double clicks

the map. An application might elect to display the coordinates where the user clicked the map or perhaps create a graphic at that point. The `Map.onClick()` event can be used to handle these scenarios.

Map Events Related to Layers

A number of events related to layers can be found on the Map object. These include events that fire when a layer is added, removed, and reordered. Events related to layers include `Map.onLayerAdd()`, `Map.onLayerRemove()`, `Map.onLayerReorder()`, `Map.onLayersRemoved()`, and a few others.

Map Events Related to the Mouse

There are a lot of events that can happen related to the mouse and the Map object. Some of the more commonly used mouse events include `onMouseDown`, `onMouseUp`, `onMouseOver`, `onMouseDown`, `onMouseWheel`, and several others. Each corresponds to some action the user has initiated with the mouse.

Zoom and Pan Events

Zoom and pan operations are common place in a mapping application and the API contains several events that fire when these take place. These events include `onExtentChange()`, `onPanStart()`, `onPan()`, `onPanEnd()`, `onZoomStart()`, `onZoom()`, and `onZoomEnd()`.

Note: For a more in-depth treatment of the ArcGIS Server API for JavaScript please refer to our [Building Custom ArcGIS Server Applications with JavaScript](#). This course is offered as both a traditional instructor led course and an instructor guided online course. For more information on this course please visit our website at geospatialtraining.com