Walk-in Slide: AU 2014 Social Media Feed

1. Click on the link below, this will open your web browser

http://aucache.autodesk.com/social/visualization.html

2. Use “Extended Display” to project the website on screen if you plan to work on your computer. Use “Duplicate” to display same image on screen and computer.
AutoCAD, JavaScript and the Cloud

Fernando Malard
Chief Technology Officer – ofcdesk, llc.
@fpmalard
This class will present the new JavaScript API for AutoCAD software, and attendees will learn how to use this technology to create web- and cloud-based plug-ins. We will connect web capabilities to the JavaScript API technology, demonstrating how users can quickly create the AutoCAD Thin Client software plug-ins, utilizing amazing JavaScript libraries and obtaining a unique user experience with professional results.
Key learning objectives

At the end of this class, you will be able to:

- Learn how to create basic JavaScript code
- Learn how to use the AutoCAD software JavaScript API
- Learn how to integrate the AutoCAD software JavaScript API with existing JavaScript Libraries
- Learn how to integrate the AutoCAD software JavaScript plug-ins with the cloud
Introduction
Introduction

- Nothing to do with Java language
- Created by Brendan Eich
- Script language weakly typed
- Supports most of C/C++ syntax
- Based on functions as first-class objects
```
var a = 2; // Initialize the a variable with number 2
var b = {}; // Initialize the b variable with an object
// Recursive function example
function factorial(n) {
    if (n == 0) {
        return 1;
    }
    return n * factorial(n - 1);
}
```
<html>
<head>
<script>
  function myFunctionHead() {
    document.getElementById("demo").innerHTML = "Hi, I'm at the HEAD."
  }
</script></head>

<body>
<script src="myScript.js"></script>
<script>
  function myFunctionBody() {
    document.getElementById("demo").innerHTML = "Hi, I'm at the BODY."
  }
</script>

<h1>AU Web Page</h1>
<p id="demo">This is the demo placeholder.</p>
<button type="button" onclick="myFunctionHead()">Call Head</button>
<button type="button" onclick="myFunctionBody()">Call Body</button>
</body>
</html>
JavaScript companion technologies
JavaScript companion technologies
AJAX (Asynchronous JavaScript XML)

- AJAX is a group of web technologies
- Implements background communication with the web page server
- Avoid page reloads
- Composed by: HTML, XML, CSS, DOM and XMLHttpRequest
JavaScript companion technologies
DOM (Document Object Model)

- DOM is a cross-platform convention to interact with HTML, XHTML and XML documents

- Useful "nodes" to access the document:
  - The document itself (document node)
  - All HTML elements (element nodes)
  - All HTML attributes (attribute nodes)
  - Text inside HTML elements (text nodes)
  - Comments (comment nodes)
JavaScript companion technologies
CSS (Cascading Style Sheets)

- Used to display and format HTML elements (styles)
- Color, font, size, margin, alignment, border, etc.

**Selectors:**
- element
- id
- class

```
#para1 {
  text-align: center;
  color: red;
}
```
```
p {
  text-align: center;
  color: red;
}
```
```
.center {
  text-align: center;
  color: red;
}
```
JavaScript companion technologies

JSON (JavaScript Object Notation)

- Language independent and alternative to XML
- Less verbose and based on attribute-value pairs
- Types: String, Number, Boolean, Array, Object and null

```json
{"employees": [
   {"firstName":"John", "lastName":"Doe"},
   {"firstName":"Anna", "lastName":"Smith"},
   {"firstName":"Peter", "lastName":"Jones"}
]}

<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```
AutoCAD JavaScript API
AutoCAD JavaScript API

- Currently in version 2 (AutoCAD 2015)
- Solves the potential **cross-platform** challenges
- Uses **Chromium** component and **V8** JavaScript engine
- Handled by **AcWebBrowser.exe** module
- Runs through **HTML** pages
- Communicates with AutoCAD via IPC encoded in **JSON**

http://app.autocad360.com/jsapi/v2/Autodesk.AutoCAD.js
AutoCAD JavaScript API

Acad.Promise

- Standard for AutoCAD JavaScript API callbacks
- Design pattern to avoid multiple level code indentation
- **Attach** callbacks to a promise
- Can be **reused** to respond to multiple promise calls
- In AutoCAD, implements **Success** and **Error**
- Used via “.then(success,error)” calls
AutoCAD JavaScript API
Acad.Promise (Demo 1)

function onGetDistanceSuccess(jsonPromptResult) {
  if (jsonPromptResult.status == Acad.PromptStatus.OK) {
    alert(jsonPromptResult.value);
  }
  else
    alert(JSON.stringify(jsonPromptResult));
}

function onGetDistanceError(jsonPromptResult) {
  alert(JSON.stringify(jsonPromptResult));
}

function promptGetDistance() {
  var options = new Acad.PromptDistanceOptions('Input Distance: ');
  options.allowZero = false;
  options.allowNegative = false;
  Acad.Editor.getDistance(options).then(
    onGetDistanceSuccess, onGetDistanceError);
}
AutoCAD JavaScript API
Acad.Application / Acad.Prompt… (Demo 2)

- Provide access to dialog like `showModalDialog`
- **Palette** creation
- Access the **active** document
- **Observers** (reactors) via active document
- **Highlight** operations
AutoCAD JavaScript API
Acad.Editor (Demo 1 and 3)

- Access the current **viewport**
- **Zoom**, orbit, pan (through current viewport)
- Register and call **commands**
- **Select** entities
- Get **distances**, **points**, angles, text, etc.
AutoCAD JavaScript API
Acad.SystemVariableCollection (Demo 4)

- Access to the AutoCAD system variables
- Retrieved by its name
- Change their value (except the read-only)
- Attach listeners (reactors) to monitor and receive callbacks when their value changes
AutoCAD JavaScript API
Acad.DrawJig (Demo 5)

- Add **temporary graphics** to the AutoCAD screen
- Users can have better **visual response** to commands
- Drawing is driven by **Transient** and **DrawStream** XSD schemas:
- Work together with `Acad.Editor.drag()` function
AutoCAD JavaScript API
Acad.Oset / Acad.DBEntity (Demo 2)

- Support to **selection sets**
- Work like a **collection** (add, remove, contains, etc.)
- **Input parameter** to several API functions
- Retrieve **entities** and get **properties**
- Get entity’s **extents** information (max/min points)
Debugging JavaScript code inside AutoCAD
Debugging JavaScript code inside AutoCAD

Developer Tools

- Part of Chromium component
- Adds debugging capabilities to HTML inside AutoCAD
- Accessed through F12 key
- Supports web files like CSS, HTML and JavaScript
- Can reload files on the fly
- Works asynchronously with AutoCAD
- Property and variable inspectors
- Break points, step into, step over, run
Debugging JavaScript code inside AutoCAD
Developer Tools
Extending AutoCAD JavaScript API
Extending AutoCAD JavaScript API

- Add **new features** to current JavaScript API
- Supported by ObjectARX in **C++** and **.NET**
- Requires the **module** to be loaded into AutoCAD
- Useful when you need to **hide and protect** some features of your application
- **JSON** are not natively supported but in .NET you can use third-party packages like **Newtonsoft.Json**
Extending AutoCAD JavaScript API

The callback method in .NET

```csharp
[JavaScriptCallback("MyJavaScriptCallback")]
public string MyJavaScriptCallback(string jsonArgs)
{
    var doc = AcadApp.DocumentManager.MdiActiveDocument;
    if (doc == null)
        return ("{"retCode":1}"");

    // Receive
    var jo = JObject.Parse(jsonArgs);
    var myString = jo.Property("message").Value.ToString();
    var myBool = (bool)jo.Property("enabled").Value;

    // Return
    var joRet = new JObject();
    joRet.Add("retCode", 0);
    joRet.Add("result", 99);
    return joRet.ToString();
}
```

- **JavaScriptCallback attribute**
- **Receive** a JSON string as input string
- **Should return** a JSON string as result
- **Use JSON parsing** to access elements as properties
Extending AutoCAD JavaScript API
The JavaScript interop function (Demo 6)

- JavaScript function will call the .NET callback
- The function will send JSON parameters
- Function will receive a JSON return string
- Calls are made via `exec()` or `execAsync()` entry point

```javascript
function MyJavaScriptCallback() {
  var jsonResponse =
    exec(
      JSON.stringify(
        {
          functionName: 'MyJavaScriptCallback',
          invokeAsCommand: false,
          functionParams: { message: 'MyMessage', enabled: false }
        }
      )
    );
  alert(JSON.stringify(jsonResponse));
}
```
Autodesk Cloud Services
Build your own Cloud
Build your own Cloud

- Specific **Business rules**
- Adds support to **mobile/remote access** to complex services
- Can be **integrated** with other Cloud services
- Hosting can be extended with Amazon / Azure high-end servers across the world (**99.99% available**)
- Requires **server-side** development
Build your own Cloud
MyCloud Sample - Demo

- Simple server with a **MySQL** database
- **HTML** web page to input data
- 100% **JavaScript** client inside AutoCAD
- Ideal for remote access via **mobile** applications
- **Low data** transfer via Internet
- Send / Receive data via **JSON**
- Can be **secured** by Internet Protocols
Conclusion

- AutoCAD JavaScript is a new technology and will evolve a lot during the next years
- Perfect solution for mobile and simple plugins
- Probably the most known language around
- Professional skills easy to find and hire
- Vast documentation and libraries
- Cross-platform natively
- Early adopters will work harder
Thank You
Session Feedback

- Via the Survey Stations, email or mobile device
- AU 2014 passes given out each day!
- Best to do it right after the session
- Instructors see results in real-time
Students, educators, and schools now have FREE access to Autodesk design software & apps.

Download at www.autodesk.com/education
Earn your professional Autodesk Certification at AU

Visit the AU Certification Lab