Smart Mapping and Symbology with Arcade

Tom Schwartzman, Esri
Session Goals

• Overview of Arcade
  - What is Arcade
  - Why use Arcade

• Arcade Language
  - Variables, Functions, Loops, Conditional Statements

• Arcade examples
  - ArcGIS Pro
  - ArcGIS Online
What is Arcade?

- A new expression language for the ArcGIS Platform
  - ArcGIS Pro
  - Runtime SDKs
  - JavaScript API
  - Web apps

- Primarily designed for ArcGIS web maps and web scenes
Arcade Purpose

- Arcade is NOT intended to be a Full Programming / Scripting Language
  - Goal: Lightweight and Simple
  - Equivalent to a Spreadsheet Cell Calculation

- Embedded Expressions
  - Labeling, Rendering, Symbol Variation, ….
  - Sharing

- Not a replacement for Python for geoprocessing and Automation

Input Values  Arcade Expression  Result Value
Why do I need another language?
ArcGIS: Make, Share, Anywhere
Arcade Goals

- **Portable**
  - Write an Expression in Pro, have it run in a Browser, or on a Device

- **Secure**
  - Ensure that Expressions or Scripting do not compromise Security

- **Lightweight**
  - Most requirements for Scripts and Expressions are Lightweight. The language should be small, and fast

- **Geospatial**
  - Scripts should treat Geospatial as First Class Members
What can I do with it?

```javascript
var tea = new tea();
if (tea.empty()) {
  tea.refill();
} else {
  tea.drink();
}
```
Language Overview
Arcade : Language Features

• Designed for Simple Expressions
  - Single Line
  - Implicit Returns
  - Case Insensitive Language
  - Dictionary Syntax for Field Access

• Has Language Features to Support more Complex Scripts
  - Type System
  - Implicit and Explicit Type Casting
  - Flow Control Statements: IF, FOR
  - Variable Declaration and Assignments
  - User Defined Functions

• Arcade Scripts run and return a value. They are self contained and cannot alter their environment.
Implicit Returns

- Last executable statement that is a value, will automatically be considered the return value.
- Enables very simple Single Line Expressions

\[(10 + 100) * 30\]

Equivalent to:

```java
return (10 + 100) * 30;
```
Case Insensitive

UpPeR(“Hello”)  
$FeAtUrE.FiElD1

• Language is Case Insensitive
  - Quicker to Author
  - Simpler for working with Data, and Field Names
# Functions

## Data Functions
- Dictionary
- DomainCode
- DomainName
- Feature
- First
- HasKey
- IndexOf
- Reverse
- Sort
- TypeOf

## Date Functions
- Date
- DateAdd
- DateDiff
- Millisecond
- Minute
- Hour
- Month
- Weekday
- Year
- Day
- Now
- Today

## Geometry Functions
- Geometry
- Multipoint
- Extent
- Polyline
- Polygon

## Logical Functions
- IsEmpty
- DefaultValue
- When
- Decode
- Iif
- Boolean

## Mathematical Functions
- Abs
- Acos
- Asin
- Atan
- Atan2
- Average
- Ceil
- Cos
- Count
- Exp
- Floor
- Log
- Min
- Max
- Number
- Pow
- Random
- Round
- Sin
- Sqrt
- Stdev
- Sum
- Tan
- Variance

## Text Functions
- Concatenate
- Console
- Find
- Lower
- Left
- Mid
- Proper
- Replace
- Right
- Split
- Text
- Trim
- Upper

“Functions are the Power of Arcade”
Global Variables and Profiles

Rendering
- Use Expressions to provide values required for different types of Renderers
  - Global Variables
    - $feature
    - $view

Labeling
- Use Expressions to provide values required for different types of Renderers
  - Global Variables
    - $feature
    - $view

Future Profiles

Profiles provide the purpose for executing an Arcade Expression
Exploring Profiles and Functions in the Playground
Type System

- Simple Types
  - Numbers
  - Booleans
  - Dates
  - Strings
- Object Types
  - Dictionary
  - Feature
  - Array
  - Point
  - Line
  - Polygon
  - Multipoint

```javascript
var myNumber = 10;
var myText = "Hello";
var myDate = Date(2015,1,1);
var myBool = true;
var myDictionary = { "key1": 10 };
var myFeature = { "geometry": {...},
                 "attributes": {"key1": 10 }};
var myArray = [1,2,3];
var myPoint = Point({....});
```

Dates, Dictionary, Feature, Point, Line... all have overloaded constructors.
Implicit and Explicit Type Casting

- **Implicit Casting**
  - For Function Parameters
  - For Expressions

- **Explicit Casting**
  - Functions cast between types
  - Number, Text, Date, Boolean

Implicit Casting
return 10 + "Hello"

Explicit Casting:
return text(10) + "Hello"
If Statement

- IF Statements are supported in the Language

- For simple single line expressions, consider using Functions
  - IIF
  - Decode
  - When

```java
if (myvalue==true) {
    return 1;
}
else {
    return 2;
}
```

Equivalent to:

```
IIF(myvalue==true, 1, 2)
```
For Statement

- For Statements
  - Same syntax as JavaScript

- For In Statements
  - Iterates over indices of an array, or field names of a Dictionary or Feature

- Break, Continue and Return Statements Supported inside Block

```javascript
for (var z=1; z<100; z++) {
    k+=z;
}

for(var k in myArray) {
    n+=myArray[k];
}
```
User Defined Functions

```javascript
function MultiplyNumber(thenumber) {
    return thenumber*10;
}

MultiplyNumber(10);
```

- **Function Keyword used to Declare Functions**
  - Must beDeclared ahead of Use
  - Variables defined inside of Function, have Local Scope
  - Functions cannot be declared inside of Functions.
Variable Assignment

- Variables can have their values reassigned.
  - Objects Types maybe Immutable, if passed into Arcade.
  - Arrays are not Sparse. They must be sequential.
  - Geometry Types are immutable inside. You cannot change the points in a path of a line.

```javascript
++myNumber
i=i+1;
myArray[10]=11;
myFeature.Field1 = “ddd”;
```
Demos

% population without formal education

<table>
<thead>
<tr>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. var noSchool = $feature.EDUCBL_CY;</td>
</tr>
<tr>
<td>2. var total = $feature.EDUCA_BASE;</td>
</tr>
<tr>
<td>3. Round((noSchool/total)*100,2);</td>
</tr>
</tbody>
</table>

- $feature.AVGHH1S2_CY (2014 Average Household Size (MBR))
- $feature.EDUC01_CY (2014 Pop 12+y/EdUC: Without Education (MBR))
- $feature.EDUC02_CY (2014 Pop 12+y/EdUC: Preschool or Kindergarten (MBR))
- $feature.EDUC03_CY (2014 Pop 12+y/EdUC: Incomplete Primary (MBR))
- $feature.EDUC04_CY (2014 Pop 12+y/EdUC: Primary Completed (MBR))
- $feature.EDUC05_CY (2014 Pop 12+y/EdUC: Incomplete Secondary (MBR))
- $feature.EDUC06_CY
Future
More uses of Arcade

- Field Calculators
- Pop-ups
Adding Functions

Identifying Common Operations

- Let us know what you need!

<table>
<thead>
<tr>
<th>Arcade Function Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Functions</td>
</tr>
<tr>
<td>Dictionary</td>
</tr>
<tr>
<td>DomainCode</td>
</tr>
<tr>
<td>DomainName</td>
</tr>
<tr>
<td>Feature</td>
</tr>
<tr>
<td>First</td>
</tr>
<tr>
<td>HasKey</td>
</tr>
<tr>
<td>IndexOf</td>
</tr>
<tr>
<td>Reverse</td>
</tr>
<tr>
<td>Sort</td>
</tr>
</tbody>
</table>
Resources

For Arcade Documentation
https://developers.arcgis.com/arcade

Playground
https://developers.arcgis.com/arcade/playground/

Using Arcade with JS API

Blogs and Posts
https://blogs.esri.com/esri/arcgis/2016/12/19/introducing-arcade/
https://blogs.esri.com/esri/arcgis/2016/12/15/use-arcade-expressions-to-map-your-ideas/
http://odoe.net/blog/hello-arcade/