

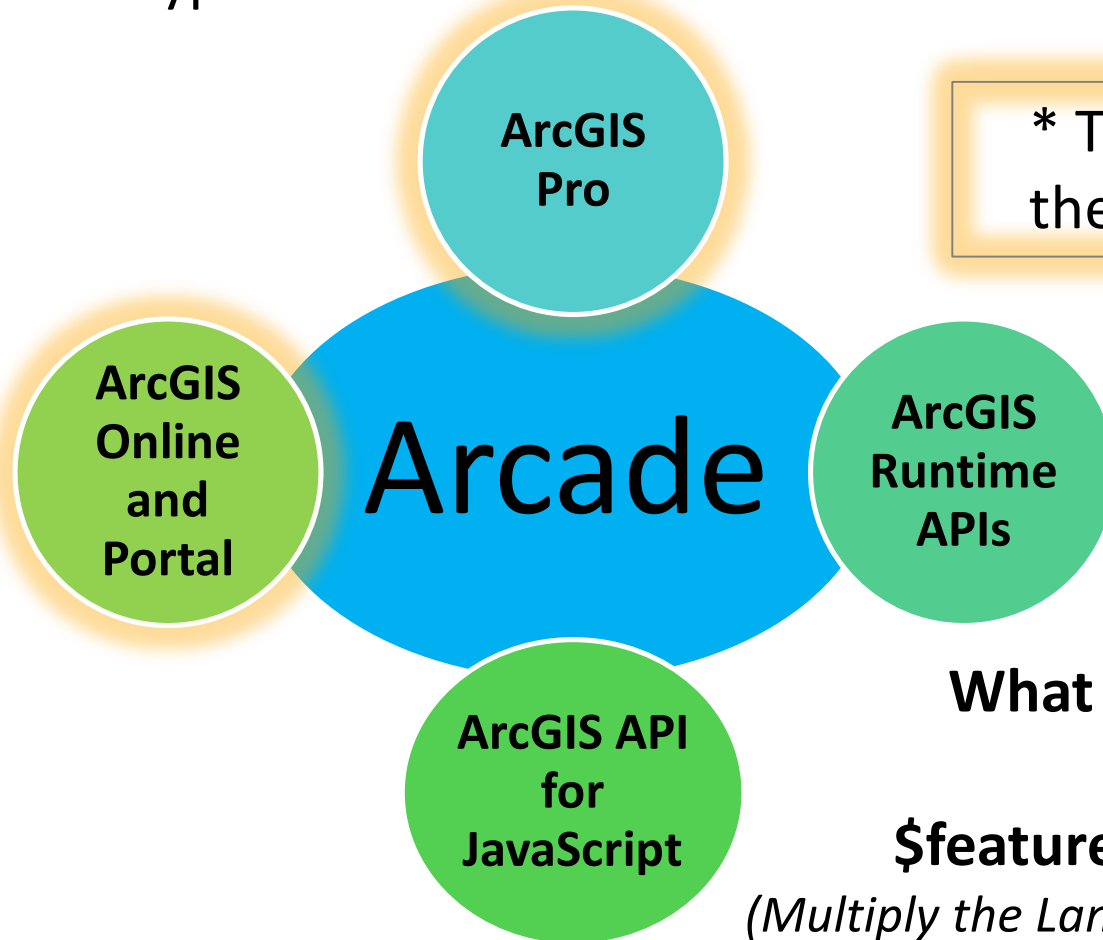
# Arcade Scripting

## The Arcade Scripting Language

- Why ANOTHER Language?
- Basic Syntax
- Arcade Operators and Functions
- Custom Symbology with Arcade
- Conditional Statements
- Arcade for Labeling

# Why ANOTHER Language?

- Arcade is a lightweight, yet powerful, expression language used securely across the entire **ArcGIS platform** and on many different device types



\* This talk will focus on these two uses of Arcade

**What does it look like?**

**`$feature.landAcreage * .01`**

*(Multiply the LandAcreage attribute by .01 and return the result.)*

- Labeling, field calculations, symbology, pop-ups will work the same
- Use across windows, Android devices, and iOS devices
- Small and uncomplicated
- Syntax similar to JavaScript and Python



Label Class Wells - Bottom Hole - Default

Class | Symbol | Position

Language Arcade

Fields	Functions
Rig Kelly Bushing Elevation	Abs()
Well Bottom Hole Measurec	Acos()
Well Bottom Hole True Verti	Asin()

Insert Values

Expression

```
$feature.WBHMD + TextFormatting.newline + $feature.WBHTVD
```

Refresh
  OK
  Cancel

Insert fields with coded value description function  
 Remove extra spaces  
 Remove extra line breaks

[Learn more about label expressions](#)

- 🌐 Arcade manipulates data on the fly for:

## Labeling

```
$feature.TYPE + "\n" + $feature.STATUS;
```

## Symbology

```
if ($feature.OilProduction > 0) {  
  return "Oil Producer"  
}  
else {  
  return "Other"  
}
```

## Pop-Ups

```
$feature.GasProductionMCF \ 5800;
```

## Field Calculations

```
$feature.Shape_Length \ 5280;
```

- 🌐 Profiles – Arcade has slightly different capabilities depending on where in the ArcGIS world you are using them
  - **Field Calculation**
  - **Labeling**
  - **Pop-Ups**
  - **Visualization**
  - **There are others**
- 🌐 Profiles define which parameters are available as global variables
  - In the Arcade interface, you see only those that are used by that profile

## Custom Pop-Ups in ArcGIS Pro

The screenshot displays the ArcGIS Pro interface for configuring pop-ups. The **Pop-ups** panel on the left shows the **Fields** tab selected, with a list of fields including `{NAME1}` and `Fields(25)`. At the bottom of this panel, the **Expressions** button is highlighted with a red arrow.

The **Expression Builder** panel in the center shows the following configuration:

- Language:** Arcade
- Name:** Lessees
- Title:** Custom
- Fields:** FID, SERIAL\_NO, PRICE PER ACRE, CASE\_TYP\_CD
- Functions:** Abs(), Acos(), Asin(), Atan()

The **Fields Options** panel on the right shows a list of fields with checkboxes, including `PERINT_1 (PERINT_1)`, `NAME_2 (NAME_2)`, `INTREL_2 (INTREL_2)`, `PERINT_2 (PERINT_2)`, `UPDATE_DT (UPDATE_DT)`, `CAT_CD (CAT_CD)`, `Shape_Leng (Shape_Leng)`, `Shape_Area (Shape_Area)`, `EXPIRE_YEA (EXPIRE_YEA)`, `TOTALPRICE (TOTALPRICE)`, `GASSALES (GASSALES)`, `OILSALES (OILSALES)`, `PricePAcre (PricePAcre)`, and `Custom (expression/Lessees)`.

### Expression

```
$feature.NAME_1 + "\n" + $feature.NAME_2
```



# An Example: Pop-Ups in Web Maps

Configure Pop-up

Tanks

**Pop-up Contents**

Display: A list of field attributes

These field attributes will display:

OBJECTID {OBJECTID}  
STATUS {STATUS}  
TYPE {TYPE}

Configure Attributes

**Attribute Expressions**

Adding expressions allows you to create new information from existing fields for use in pop-ups.

ADD

No expressions.  
Click 'Add' to add one.

OK CANCEL

Trust Center Contact Esri Report Abuse Contact Us

Custom Edit

Expression

Test

```
1 $feature.STATUS + "\n" + $feature.TYPE
```

Custom Edit

Expression

```
1 $feature.STATUS + "\n" + $feature.TYPE
```

# Basic Syntax

- Similar to Python and to JavaScript

- But NOT case sensitive

```
$feature.API or $FEATURE.api
```

- Arcade scripts execute and return a value. That's all they do. They cannot alter their environment. With multi-line scripts, use `Return` to return the value (but this can be omitted).

```
return $feature.Shape_Length / 5280
```

- Lines should end with a `;` (but this can be omitted on single line code or when it's obviously the end of the line)

```
return left($feature.API,2);
```

- Comments start with `//` or `/*`

```
/*
```

```
This code concatenates lease  
acreage with expiration date
```

```
*/
```

```
//This is a single line comment
```

- Concatenate strings using a +

```
$feature.API + "-" + $feature.Operator;
```

- Arcade makes smart assumptions (implicit type casting) about the data type to determine whether the + should concatenate or do math
- What about the Python "\n" or Visual Basic vbNewLine equivalent?

```
"\n"
```

```
TextFormatting.NewLine
```

It doesn't work in ArcGIS Online yet. It should be available later in 2019. It works in Pro, though.

Other TextFormatting: .BackwardSlash, .DoubleQuote....

- 🌐 ArcGIS Pro Field Calculations
  - 🌐 Calculate the total cost of a lease, given that you have the following columns:
    - Price Per Acre (as a text string)
    - Shape\_Area (in square meters)
1. Convert the price per acre field to a number
  2. Convert the Shape\_Area to acres
  3. Multiply the price per acre times the area, in acres

```
($feature.Shape_Area * 0.000247105)  
  * Number($feature.PRICE)
```

# Arcade Operators and Functions

- Mathematical Operators

+, -, /, \*, % (modulus), ++, many others

- Boolean/Logical Operators (for comparisons)

Description	Operator
Equals	==
Not Equal	!=
AND	&&
OR	



## String Functions:


Upper, Lower, Mid, Left, Right, many others

## Numeric Functions:

Mean, Min, Max, Round, many others (work on feature sets, using \$layer.<field\_name>)

## Date Functions:

Today, DateDiff, Day, Month, Year, many others

 These things allow us to do what we have been doing for years in ArcMap, but there are also Data Functions and Geometry Functions, for more advanced Arcade scripts (not covered in this workshop)

- \$feature is a global variable that represents the feature that is being processed (for example, the feature that you clicked on when you get a pop-up)
- Use \$feature before field names to reference the field values. You can think of \$feature like a pointer to the current record:  

```
$feature.API + "\n" + $feature.Operator;
```
- Use \$map and \$layer to get access to features from multiple layers all at once (to find min/max, for example).
  - There are functions that know how to work with \$map and \$layer
  - Mathematical functions, geometry functions, many other types

## Mathematical Functions

- Abs
- Acos
- Asin
- Atan
- Atan2
- Average
- Ceil
- Constrain
- Cos
- Exp
- Floor
- Log
- Mean
- Min
- Max
- Pow
- Random
- Round
- Sin
- Sqrt
- Stdev
- Sum
- Tan
- Variance

- Area
- AreaGeodetic
- Buffer
- BufferGeodetic
- Centroid
- Clip
- Contains
- Crosses
- Cut
- Difference
- Disjoint
- Distance
- Equals
- Extent
- Geometry
- Intersection
- Intersects
- Length
- LengthGeodetic
- MultiPartToSinglePart
- Multipoint
- Overlaps
- Point
- Polyline
- Polygon
- Relate

## Logical Functions

- IsEmpty
- DefaultValue
- When
- Decode
- If
- Boolean

## Text Functions

- Concatenate
- Find
- Lower
- Left
- Mid
- Proper
- Replace
- Right
- Split
- Trim
- Upper

## Date Functions

- Date
- DateAdd
- DateDiff
- Millisecond
- Second
- Minute
- Hour
- Month
- Weekday
- Year
- Day
- Now
- Today
- Timestamp
- ToLocal
- ToUTC

## Data Functions

- Console
- Count
- Dictionary
- Distinct
- DomainCode
- DomainName
- Feature
- First
- Guid
- HasKey
- IndexOf
- NextSequenceValue
- Number
- Reverse
- Sort
- Text
- Top
- TypeOf

- Consider this pop-up customization that will report the number of wells in a lease that is clicked-on

```
// get the count for all wells in the Lease
```

```
var well_layer = FeatureSetByName($map,"Piceance_Wells")
```

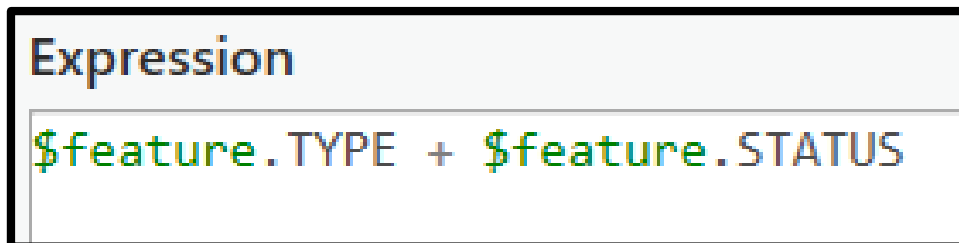
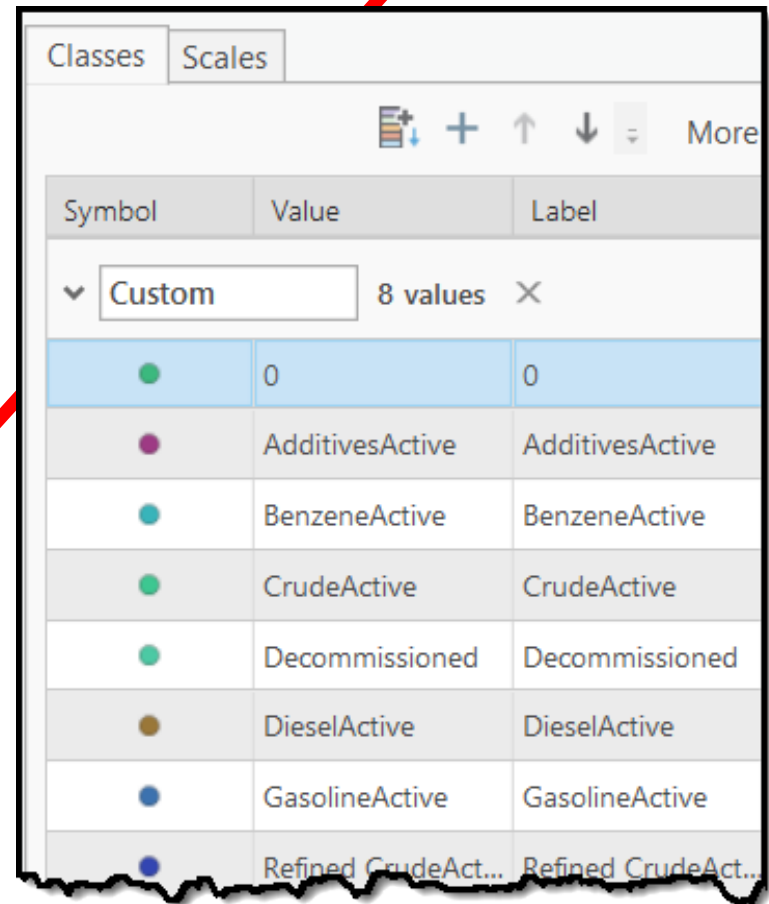
```
var well_count = Count(Intersects(well_layer,$feature))
```

```
return well_count
```

# Custom Symbology with Arcade

- Using Arcade, you can manipulate data on the fly to symbolize on a calculated field
  - You have production in MCF, but you want to symbolize based on OEB
  - You are length in feet, but you want to symbolize based on length in miles
  - You have oil and gas production values, but you want to symbolize based on whether the well is an oil or a gas producer:  
“if the oil production is greater than zero and the gas production is zero, symbolize this as an oil well...”

In ArcPro...



## Symbology in Pro

Symbology - Current Leases

Primary symbology

Graduated Colors

Field: Custom

Normalization: <None>

Method: Natural Breaks (Jenks)

Classes: 5

Color scheme: [Color palette]



Expression Builder

Language: Arcade

Title: Custom

Fields	Functions
ESA Operator Sort Name	Abs()
Lease Expiration Year	Acos()
Owner 1	Asin()
Owner Number 1	Atan()

Insert Values

Expression

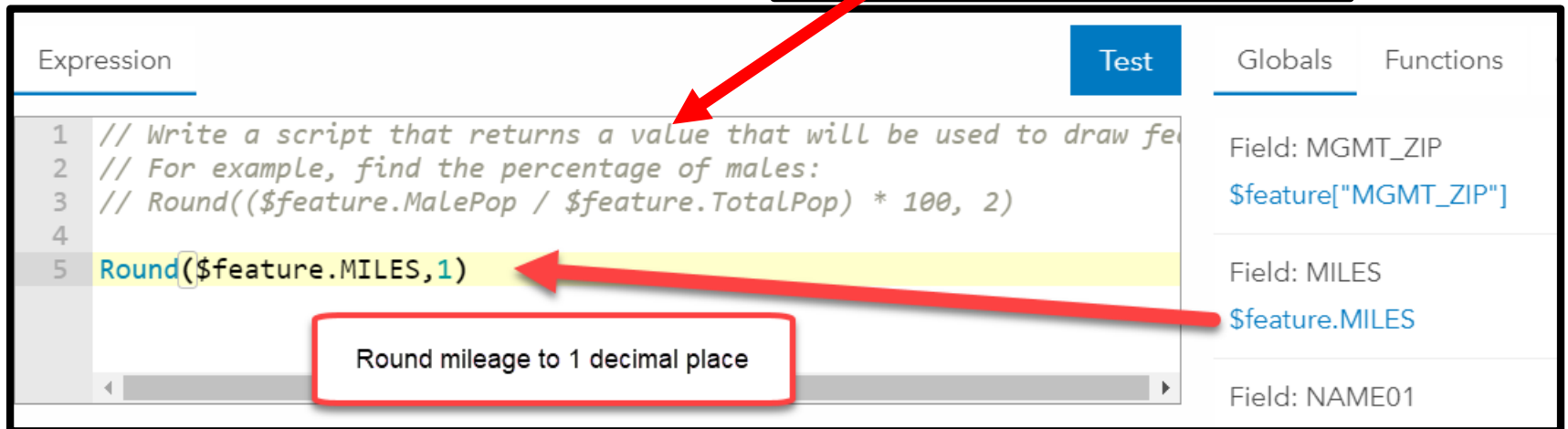
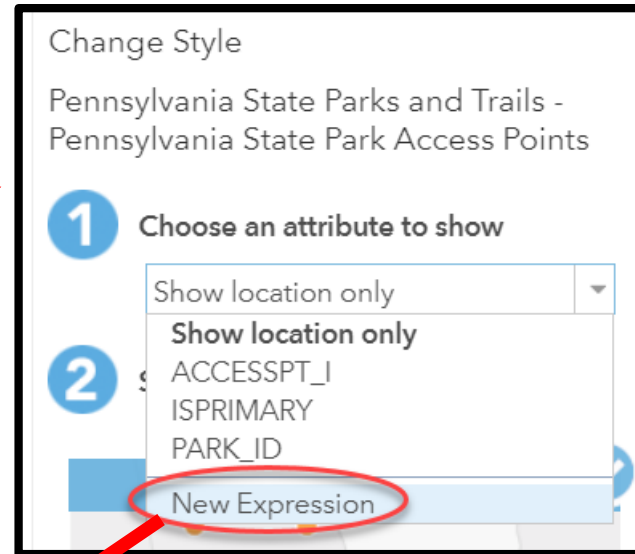
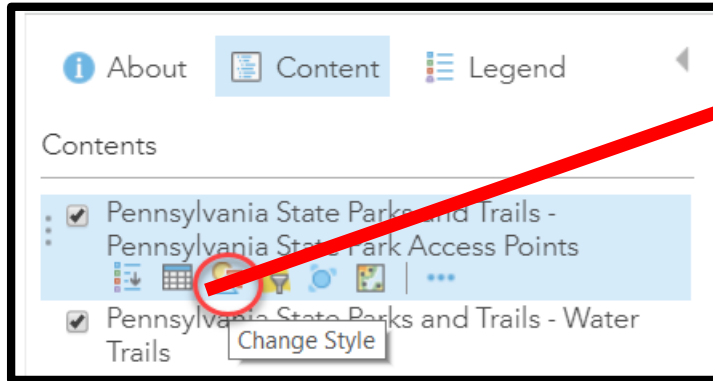
```
$feature.LEASEEXPYR - year(now())
```

✓ Expression is valid

OK Cancel



In ArcGIS Online or Portal...



Change  
Style  
Button

Attribute to  
Show drop-  
down

Scroll to the  
bottom of  
the list

Create New  
Expression

- Suppose you want to symbolize your leases based on how long until they expire. BUT...suppose the only relevant column that you have in your leases table is the Expiration date, and it is stored as a date field.

```
Round(DateDiff(Date($feature.EXPIRE_DT), now(), 'days'))
```

# Conditional Statements

🌐 Conditional statements (if/then/else):

```
if ( left($feature.API, 2) == "17") {
return "Louisiana"}
else {
return "Somewhere without good food"}
```

🌐 Other single-line conditional statements:

IIF, Decode, When

🌐 A reminder about operators:

Description	Operator
Equals	==
Not Equal	!=
AND	&&
OR	

## Create symbology based on a conditional statement

Symbology - Petroleum Permits

Primary symbology




Unique Values

Field 1: Custom

Add field

Color scheme

Classes Scales

Symbol	Value	Label
Custom 2 values X		
	After the Regulati...	After the Regulati...
	Prior to Regulation	Prior to Regulation
<all other values>		
	<all other values>	<all other values>

Expression Builder

Language: Arcade

Title: Custom

Fields: OBJECTID, NUMBER, HOLDER, STATUS, DATE

Functions: Abs(), Acos(), Asin(), Atan(), Atan2()

Insert Values

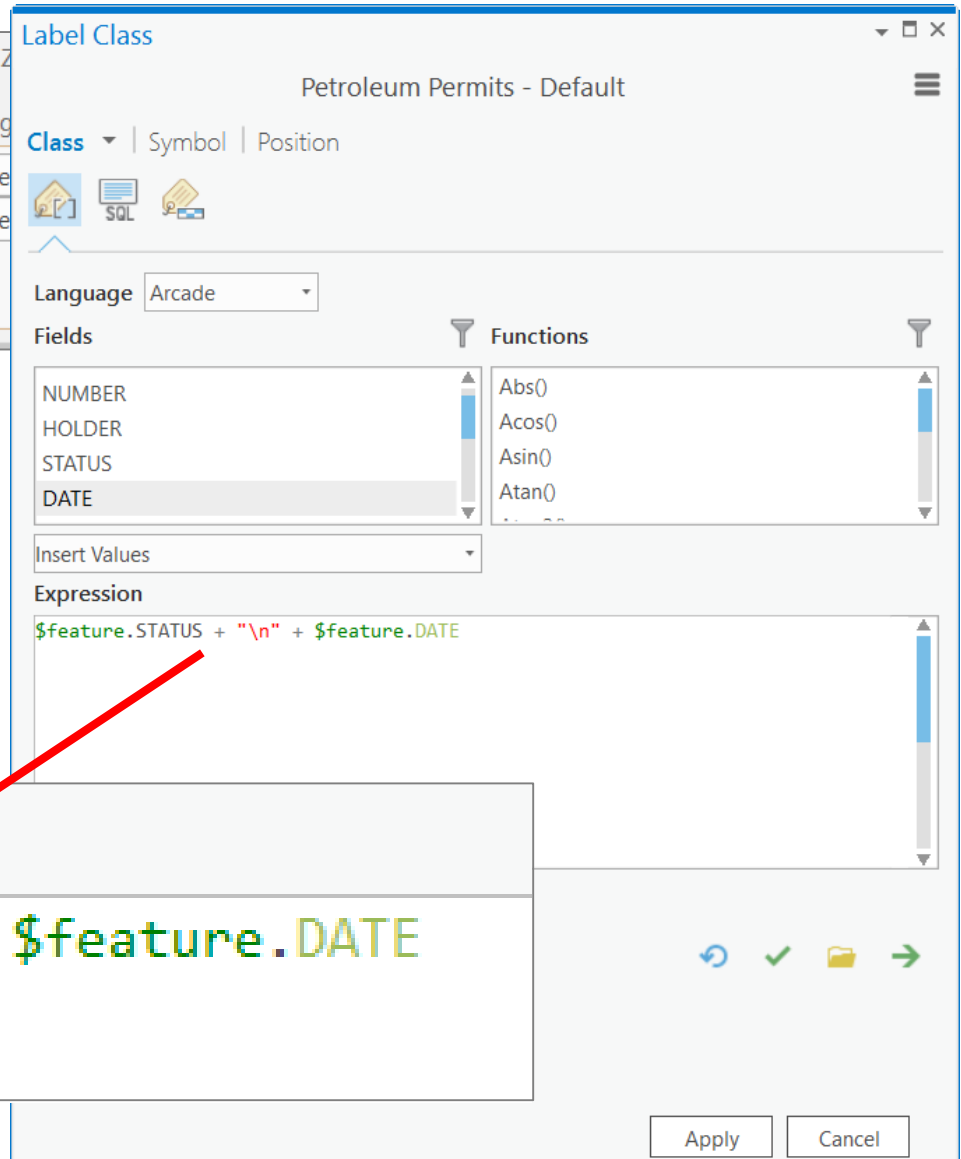
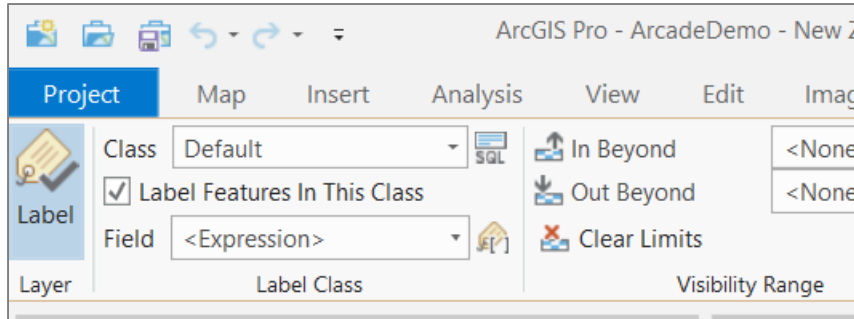
Expression

```
if ($feature.DrillYear < 2000){
    "Prior to Regulations"
}
else {
    "After Regulations"
}
```

OK Cancel

# Arcade for Labeling

## Create Custom Label Expressions

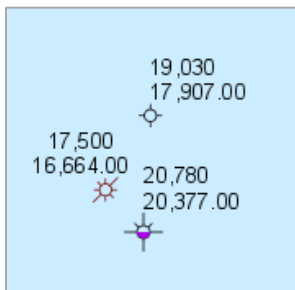


### Expression

`$feature.STATUS + "\n" + $feature.DATE`

Want it to look better?

- Format the output with the Text function
- ##,### (group by thousands)
- ##,###.00 (round to 2 decimal places)
- # is a placeholder for any digit, but it omits leading or trailing zeros
- 0 is a placeholder for any digit (puts a zero if nothing is there)



Label Class

Wells - Bottom Hole - Default

Class | Symbol | Position

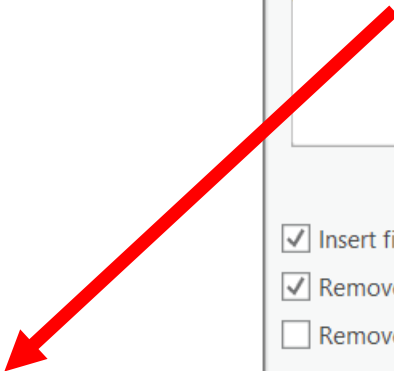
Language: Arcade

Fields: Rig Kelly Bushing Elevation, Well Bottom Hole Measurec, Well Bottom Hole True Verti

Functions: Abs(), Acos(), Asin()

Expression

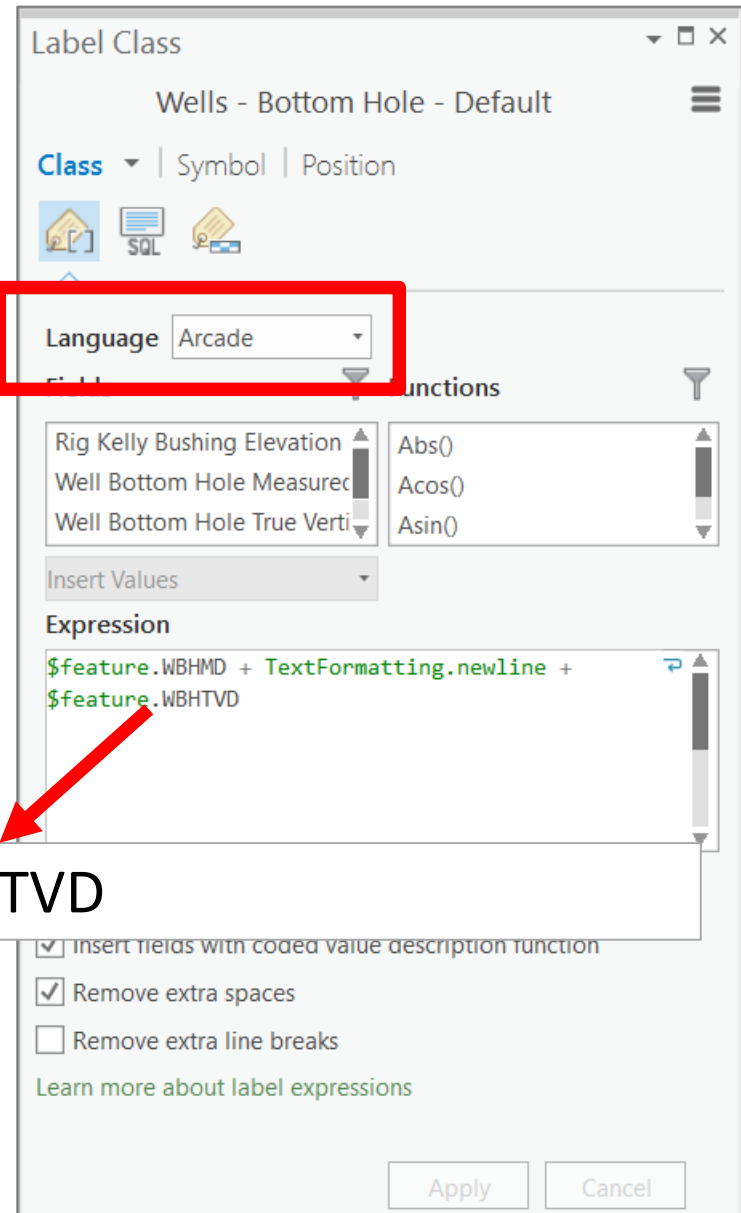
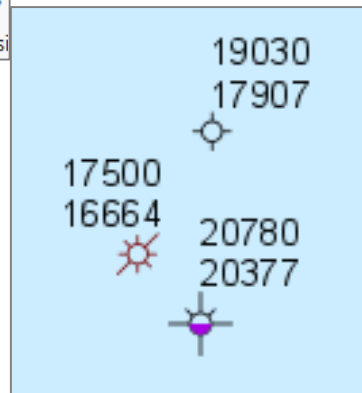
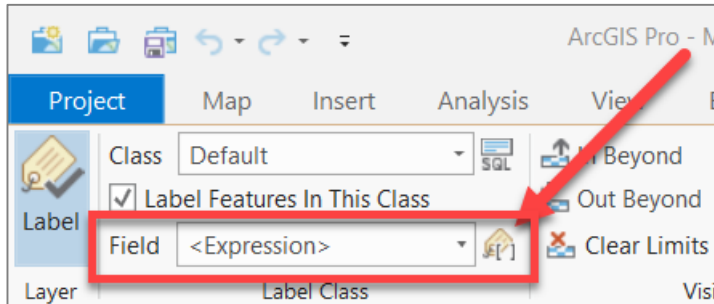
Insert fields with coded value description function  
 Remove extra spaces  
 Remove extra line breaks



```
text($feature.WBHMD,"##,###") + "\n" + text($feature.WBHTVD,"##,###.00")
```



- An example: Labeling in ArcGIS Pro:
- Label wells with MD and TVD



`$feature.WBHMD + "\n" + $feature.WBHTVD`

What would this have returned?

`$feature.WBHMD + $feature.WBHTVD`

## The Arcade Scripting Language

- Why ANOTHER Language?
- Basic Syntax
- Arcade Operators and Functions
- Custom Symbology with Arcade
- Conditional Statements
- Arcade for Labeling



**Jennifer Harrison**

TeachMeGIS

[info@TeachMeGIS.com](mailto:info@TeachMeGIS.com)

713-278-7883