

---

**FEWD**

---

# FINAL PROJECTS

**FINISH UP RESPONSIVENESS**

# AGENDA

---



- Review
- Objects
- Lab — Work on final project

## **LEARNING OBJECTIVES**

- ▶ Describe the concept of "this" as it applies within jQuery functions.
- ▶ Know how to make javascript objects and then to use them

---

**FEWD**

---

# QUICK JS REVIEW

# JQUERY — SELECTING ELEMENTS

---

Selector

```
$('li').addClass('selected');
```

jQuery Function

A diagram illustrating the components of the jQuery selector function. The code snippet is `$('li').addClass('selected');`. A yellow box highlights the selector `'li'` inside the parentheses, with the word "Selector" written above it in yellow. A red box highlights the dollar sign `$` and the opening parenthesis `(`, with the text "jQuery Function" written below it in red.

jQuery Function:

- ▶ Lets us find one or more elements in the page
- ▶ Creates a jQuery object which holds references to those elements

---

## JQUERY OBJECTS — FINDING ELEMENTS: SOME EXAMPLES

---

- ▶ You can use your CSS-style selectors!!!

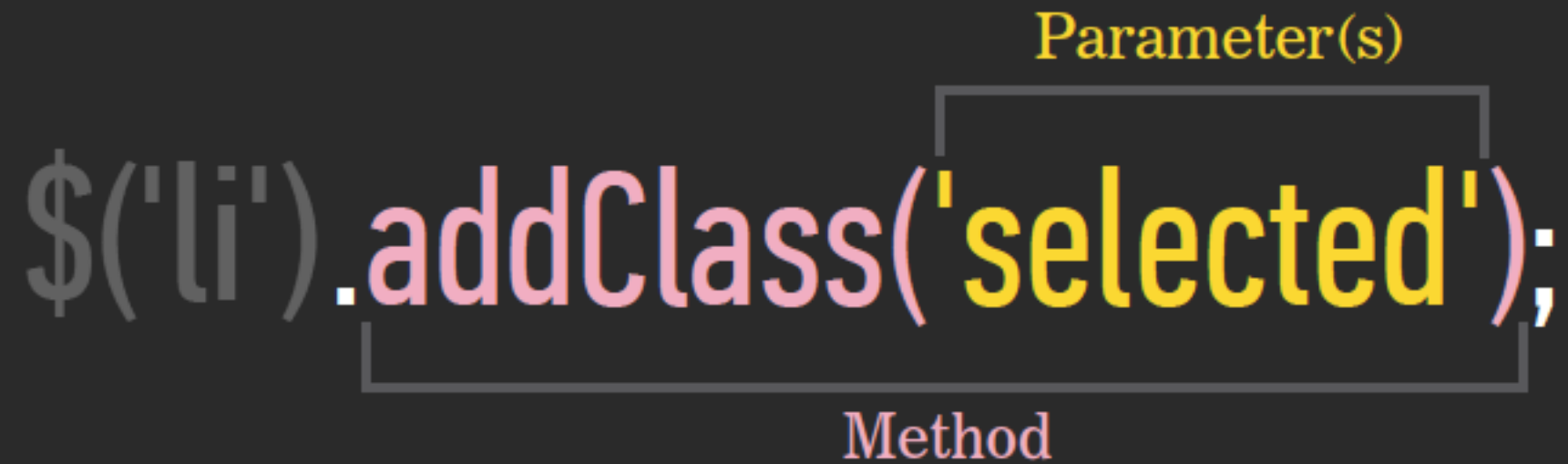
SELECTION:	CSS:		JQUERY:	
	CLASS	.className		\$('.className')
	ID	#idName		\$('#idName')
	MULTIPLE SELECTORS	h1, h2, h3		\$(h1, h2, h3')
	DESCENDANT	li a		\$(li a')

## JQUERY — WORKING WITH THOSE ELEMENTS

Parameter(s)

```
$('li').addClass('selected');
```

Method

A diagram illustrating the components of a jQuery method call. The code snippet is `$('li').addClass('selected');`. The text `$('li')` is in a light gray color. The text `.addClass('selected');` is in a pink color. A bracket above the pink text is labeled "Parameter(s)" in yellow. A bracket below the pink text is labeled "Method" in pink.



---

# JQUERY METHODS — GETTING/SETTING CONTENT

---

Get/change content of elements, attributes, text nodes

METHODS	EXAMPLES
<code>.html()</code>	<code>\$('h1').html('Content to insert goes here');</code>
<code>.attr()</code>	<code>\$('img').attr('src', 'images/bike.png');</code>
<code>.css()</code>	<code>\$('#box1').css('color', 'red');</code>
<code>.addClass()</code>	<code>\$('p').addClass('success');</code>
<code>.removeClass()</code>	<code>\$('p').removeClass('my-class-here');</code>
<code>.toggleClass()</code>	<code>\$('p').toggleClass('special');</code>

---

# JQUERY METHODS — EFFECTS/ANIMATION

---

Add effects and animation to parts of the page

METHODS	EXAMPLES
<code>.show()</code>	<code>\$('#h1').show();</code>
<code>.hide()</code>	<code>\$('#ul').hide();</code>
<code>.fadeIn()</code>	<code>\$('#h1').fadeIn(300);</code>
<code>.fadeOut()</code>	<code>\$('#.special').fadeOut('fast');</code>
<code>.slideUp()</code>	<code>\$('#div').slideUp();</code>
<code>.slideDown()</code>	<code>\$('#box1').slideDown('slow');</code>
<code>.slideToggle()</code>	<code>\$('#p').slideToggle(300);</code>

## SYNTAX — DECLARING A FUNCTION

```
function pickADescriptiveName() {  
    // Series of statements to execute  
}
```

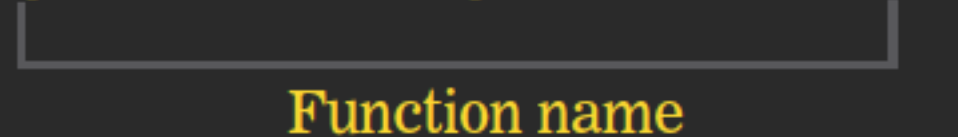
The diagram illustrates the syntax of a function declaration. It shows the code `function pickADescriptiveName() {` on the first line, `// Series of statements to execute` on the second line, and `}` on the third line. A bracket above the word `function` is labeled "Keyword". A bracket above the text `pickADescriptiveName()` is labeled "Name". A large bracket below the entire code block is labeled "Code block".

## SYNTAX — CALLING A FUNCTION

---

- ▶ To run the code in a function, we 'call' the function by using the function name followed by parenthesis.

```
pickADescriptiveName();
```



Function name

## SYNTAX — DECLARING A FUNCTION (WITH PARAMETERS)

Parameters

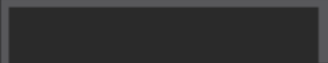
```
function multiply(param1, param2) {  
    return param1 * param2;  
}
```

We can use these parameters like variables from within our function

# SYNTAX — CALLING A FUNCTION (WITH ARGUMENTS)

---

Arguments



```
multiply(350, 140)
```

The diagram illustrates the syntax of a function call. The word "multiply" is in white, and the numbers "350" and "140" are in yellow. A horizontal bracket above the numbers, with vertical lines extending down to each number, is labeled "Arguments" in yellow text above it.

---

# ARRAYS

---

## STORING LISTS OF VALUES

- An array can be used to **store a list of values in a single variable**
- Holds an ordered collection of values
- Can hold numbers, strings, even other arrays!
- Good for things like a grocery list, a list of states, or any other list

---

# DECLARING ARRAYS

---

```
var descriptiveNameHere = [item1, item2, item3];
```



---

# ARRAYS - INDEXING

---

- Each item in an array has an **index**, by which you can access that item.
- The first item has an index of **0**, the second item 1, the third item 2, etc.

0. Milk

1. Eggs

2. Frosted Flakes

3. Salami

4. Juice

# ARRAYS - ACCESSING ITEMS BY INDEX

---

- Each item in an array has an **index**, by which you can access that item.
- The first item has an index of **0**, the second item 1, the third item 2, etc.

```
var myArray = [5, true, 2, 'Hello']
```

0 1 2 3

The diagram consists of four yellow arrows pointing upwards from the numbers 0, 1, 2, and 3 to the corresponding elements in the array: 5, true, 2, and 'Hello'.

# ARRAYS — ACCESSING ITEMS IN AN ARRAY

---

Accessing items in array:

`myArray[1]` → `true`

`myArray[2]` → `2`

`myArray[0]` → `5`

`myArray[3]` → `'Hello'`

```
var myArray = [5, true, 2, 'Hello']
```

---

# ARRAYS - ADDING A VALUE/REPLACING A VALUE

---

## INSERTING A NEW VALUE

- We can insert new values into any space in the array using the positions index.

```
myArray[1] = 'Hello';
```

## UPDATING VALUES

- If there's already an item at that position, it will be replaced with the new value.

```
var myArr = [65, 'hello', true];  
myArr[1] = 'goodbye';  
// myArr[1] now holds 'goodbye' instead of 'hello'
```

---

# ARRAYS - LENGTH

---

- ▶ We can use the `.length` property to find out how many items are in an array

```
var shapes = ['circle', 'triangle', 'square'];
```

```
shapes.length;
```

 => 3

- ▶ Accessing the last element in an array:

```
console.log(shapes[shapes.length-1]);
```

 => Prints 'square'

---

**FEWD**

---


**‘THIS’ KEYWORD**

## THE KEYWORD 'THIS'

---

**this** refers to whatever you *selected* with jQuery

```
$('#p').on('click', function(){  
    $(this).fadeOut(500);  
});
```



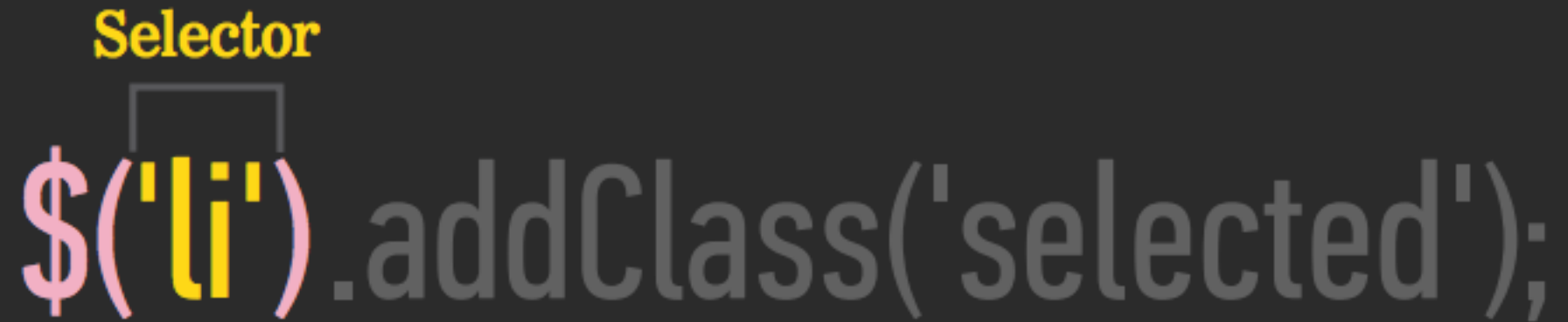
*Notice — no quotes around this!*

## JQUERY — SELECTING ELEMENTS

---

Selector

```
$('li').addClass('selected');
```

A diagram illustrating the selector part of a jQuery code snippet. The word "Selector" is written in yellow above the code. A grey bracket is positioned above the string "'li'" in the code, indicating that this string is the selector used to find the elements to be manipulated.



---

**FEWD**

---

**OBJECTS**

---

# OBJECTS

---

- OOP- Object oriented Programming
- Lets us write reusable code to keep track of data

**Everything is an object!**

# BULLDOG AS AN OBJECT

---

Objects have traits that are common to versions of itself  
These traits are called properties in javascript



## Bulldog Properties

- Legs - 4
- Sound - “Bark”
- Food - “Dog Food”

---

# DECLARING OBJECTS

---

```
function myObject () {  
  
};
```

---

# ASSIGNING PROPERTIES

---

```
function myObject() {  
    this.property = value;  
};
```

---

# MAKING NEW OBJECTS

---

```
var newObject = new object();
```

---

## GETTING OBJECT VALUES

---

```
var newObject = new object();
```

```
newObject.propertyName;
```

---

**FEWD**

---

# **HOMEWORK**



**FINISH RESPONSIVENESS THIS WEEKEND**

---

**FEWD**

---

# EXIT TICKETS