ITEC 136

Business Programming Concepts

Week 09, Part 01

Overview

FRANKLIN UNIVERSITY

FOUNDED 1902

Week 9 Overview

- Week 8 review
  - Forms and Form Processing
    - Tags
      - `<form>`
      - `<fieldset>`
      - `<input>`
      - `<textarea>`
      - `<select>` & `<option>`

FRANKLIN UNIVERSITY

Week 9 Overview

- Week 8 review
  - Forms and Form Processing
    - Form processing
      - Event handlers (`onclick`, etc.)
      - `document.getElementById().value`
      - Validation functions
      - Regular expressions (brief)

Week 9 Overview

- Outcomes
  - Use the `Math`, `Date`, and `String` functions and objects to solve problems.
  - Describe the properties and uses of arrays.
  - Instantiate, initialize, and use one-dimensional arrays.

ITEC 136

Business Programming Concepts

Week 09, Part 02

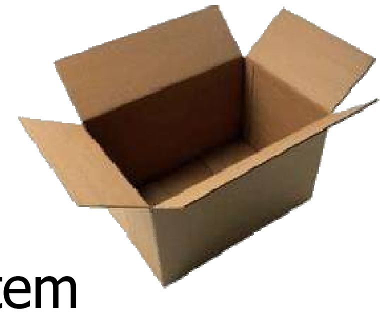
Using Objects

FRANKLIN UNIVERSITY

FOUNDED 1902

Using Objects

- Review
  - Variables are like boxes:
    - Some variables are empty
    - Some variables hold one item
    - Variable contents can be replaced
    - Variables can hold more than one thing (an array)

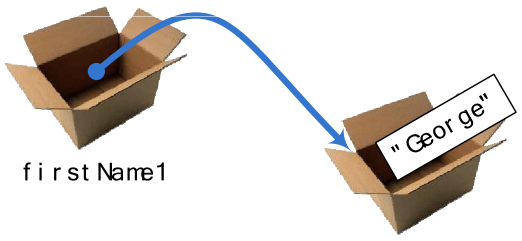


FRANKLIN UNIVERSITY

Variables and Data Types

- A small lie…
  - Actually two boxes involved: the "reference" and the object itself.

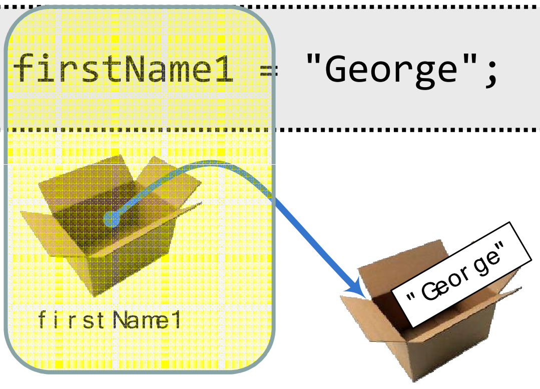
```
var firstName1 = "George";
```



Using Objects

- A small lie…
  - Actually two boxes involved: the "reference" and the object itself.

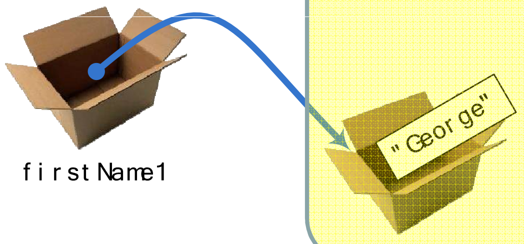
```
var firstName1 = "George";
```



Using Objects

- A small lie…
  - Actually two boxes involved: the "reference" and the object itself.

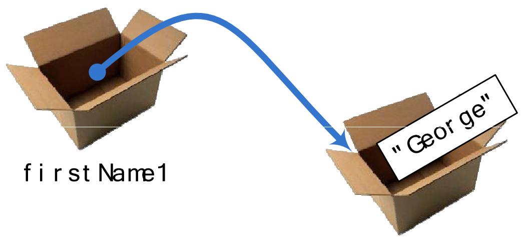
```
var firstName1 = "George";
```



Using Objects

- Assignment
  - Copies a *reference* not the data

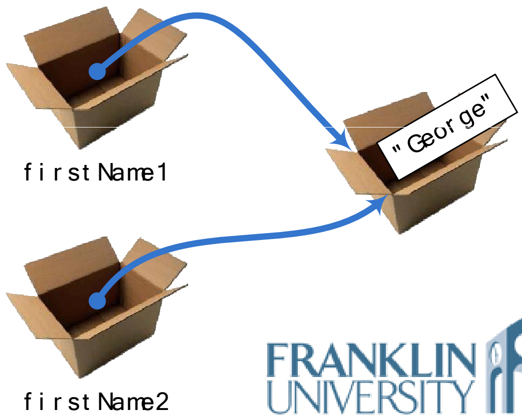
```
var firstName1 = "George";
```



Using Objects

- Assignment
  - Copies a *reference* not the data

```
var firstName1 = "George";
var firstName2 = firstName1;
```

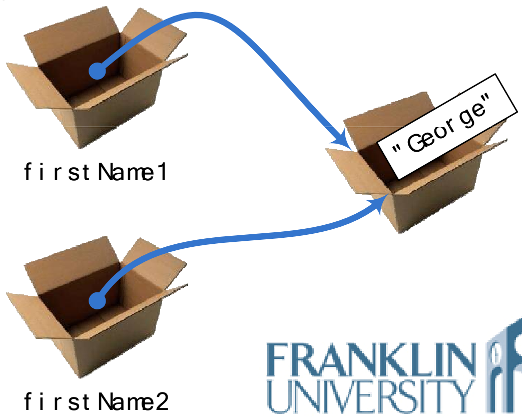


FRANKLIN UNIVERSITY

Using Objects

- Assignment
  - Copies a *reference* not the data
  - Any operation applied to one also takes place on the other

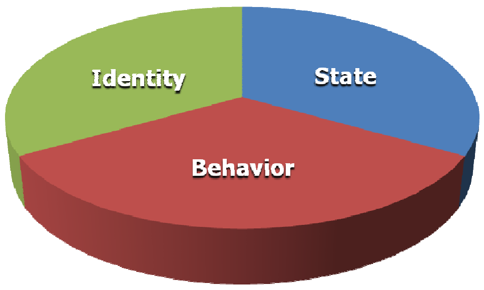
```
var firstName1 = "George";
var firstName2 = firstName1;
```



FRANKLIN UNIVERSITY

Using Objects

- What is an object?
  - An object has *identity*
    - It exists in memory.
  - An object has *state*
    - Data associated with the entity.
  - An object has *behavior*
    - Functions associated with the entity.
    - Act on the data kept in the object.



Using Objects

- Creating new objects
  - Syntax:

```
var objRef = new SomeObject();
```

The object reference. Any identifier works.

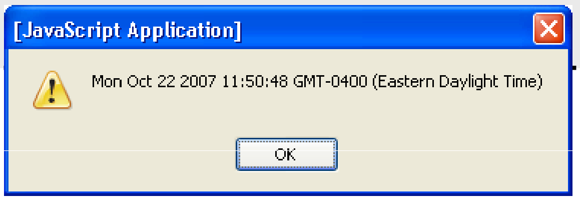
Keyword for object creation.

The "constructor" function which will initialize the object.

Using Objects

- Creating new objects
  - Example: creating a Date object

```
var rightNow = new Date();
alert(rightNow);
```

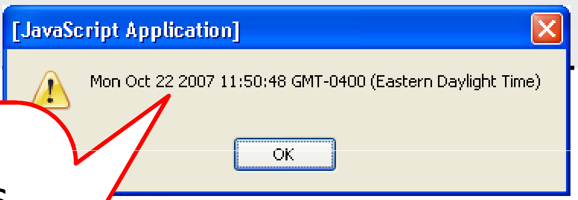


```
Mon Oct 22 2007 11:50:48 GMT-0400 (Eastern Daylight Time)
```

Using Objects

- Creating new objects
  - Example: creating a Date object

```
var rightNow = new Date();
alert(rightNow);
```



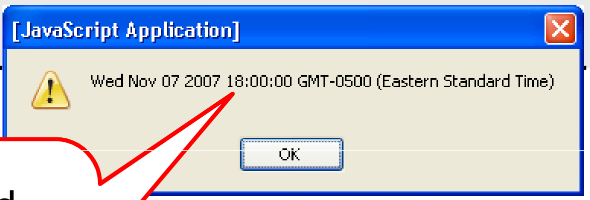
```
Mon Oct 22 2007 11:50:48 GMT-0400 (Eastern Daylight Time)
```

The string used in the alert is created using the Date object's "toString" method. The date and time is set to the current computer clock when sending zero arguments to the constructor.

Using Objects

- Creating new objects
  - Example: using constructor parameters

```
var nextExam = new Date(2007, 10, 7, 18);
alert(nextExam);
```


[JavaScript Application]

Wed Nov 07 2007 18:00:00 GMT-0500 (Eastern Standard Time)

Note the zero-based indexing for the month and the military time for the hour.

FRANKLIN UNIVERSITY

Using Objects

- Calling methods on objects
  - Syntax:

```
objRef.doSomething("foo");
```

The object reference. The identifier of an object created with "new".

The name of the method (function) to be invoked.

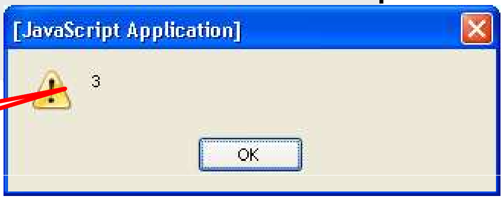
Any parameters needed to carry out the action.

FRANKLIN UNIVERSITY

Using Objects

- Calling methods on objects
  - Example:

```
var nextExam = new Date(2007, 10, 7, 18);
alert(nextExam.getDay());
```


[JavaScript Application]

3

Using zero-based indexing, the number 3 represents Wednesday.

FRANKLIN UNIVERSITY

ITEC 136

Business Programming Concepts

Week 09, Part 03

Math, Date, Number, and String objects

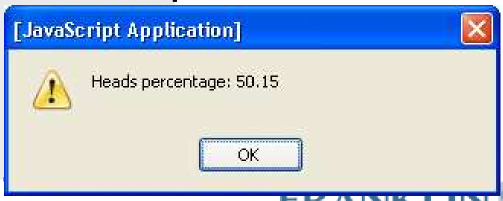
FRANKLIN UNIVERSITY

FOUNDED 1902

Math Functions

- `Math` is not a typical object
  - Don't create a Math object with **`new`**
  - Ex: flipping a coin 10,000 times

```
var heads = 0;
for (var i = 0; i < 10000; ++i)
    if (Math.random() < 0.5)
        ++heads;
alert("Heads percentage: "
    + heads/100);
```



Heads percentage: 50.15

Math Functions

- `Math` is not a typical object
  - A *namespace* to hold functions

```
var MyMath = {
    abs : function(num) {
        return num < 0 ? -num : num;
    },
    // ...more functions defined here
}
```

Math Functions

- Available `Math` functions

abs	acos	asin
atan	atan2	ceil
cos	exp	floor
log	max	min
pow	random	round
sin	sqrt	tan

See the documentation for details!

- Also a number of constants (PI, etc.)

Date Object

- A standard JS object
  - Has identity, state, behavior, created with keyword "new"

```
var birthdayStr = prompt("Enter your birthday",
    "April 28, 1975");
var birthday = new Date(birthdayStr);
var today = new Date();
var difference = today - birthday;
alert("You are " +
    Math.floor(difference/1000/60/60/24/365) +
    " years old");
```

Date Object

- A standard JS object
  - Has identity, state, behavior, created with keyword “new

```
var birthdayStr = prompt("
    "April 28, 1975");
var birthday = new Date(bi
var today = new Date();
var difference = today - birth );
alert("You are " +
    Math.floor(difference/1000/60/60/24/365) +
    " years old");
```

Dates are internally represented as milliseconds since the “epoch.” This division converts milliseconds into years.

Date Object

- Some available `Date` functions

```
getDate            getDay            getFullYear
getHours           getMilliseconds   getMinutes
getMonth           getSeconds        getTime
getTimezoneOffset  getYear           Parse
setDate            setFullYear       setHours
setMilliseconds    setMinutes        setMonth
setSeconds         setTime           setYear
```

See the documentation for details!

Number Object

- Also a standard JS object
  - Used primarily to access its constant properties (`MAX_VALUE`, `NaN`, etc.)
  - Rarely need to create one with “new” as all number variables are instances of `Number`.

Number Object

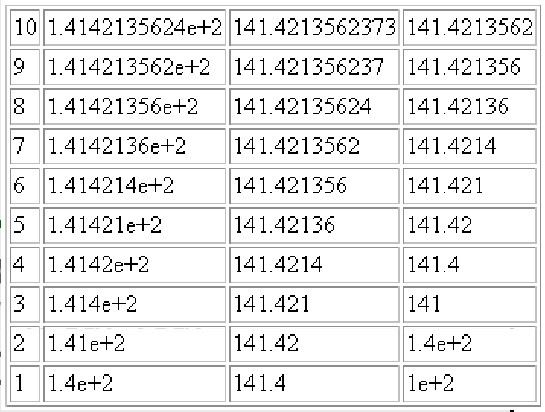
- Ex: differing number formats

```
var sqrt2 = Math.SQRT2*100;
var str = "<table border='1'>"
for (var i = 10; i > 0; --i)
{
    str += "<tr><td>" + i + "</td><td>";
    str += sqrt2.toExponential(i) + "</td><td>"
    str += sqrt2.toFixed(i) + "</td><td>"
    str += sqrt2.toPrecision(i) + "</td></tr>";
}
str += "</table>"
document.writeln(str);
```

Number Object

- Ex: differing number formats

```
var sqrt2 = Math.SQRT2*100;
var str = "<table border='1'>"
for (var i = 10; i > 0; --i)
{
    str += "<tr><td>" + i + "</
    str += sqrt2.toExponential(
    str += sqrt2.toFixed(i) + '
    str += sqrt2.toPrecision(i)
}
str += "</table>"
document.writeln(str);
```



10	1.4142135624e+2	141.4213562373	141.4213562
9	1.414213562e+2	141.421356237	141.421356
8	1.41421356e+2	141.42135624	141.42136
7	1.4142136e+2	141.4213562	141.4214
6	1.414214e+2	141.421356	141.421
5	1.41421e+2	141.42136	141.42
4	1.4142e+2	141.4214	141.4
3	1.414e+2	141.421	141
2	1.41e+2	141.42	1.4e+2
1	1.4e+2	141.4	1e+2

Number Object

- Some available `Number` functions

`toExponential`	`toFixed`	`toPrecision`
`toSource`	`toString`	`valueOf`

- Also, some available constants

`MAX_VALUE`	`MIN_VALUE`
`NEGATIVE_INFINITY`	`POSITIVE_INFINITY`
`NaN`	

See the documentation for details!

String Object

- One of the most common objects!
  - Many string methods, but only a small subset of them are used.
    - Regular expression based: `match`, `replace`, `search`
    - Substring based: `substr`, `substring`, `slice`, `split`
    - Character based: `charAt`, `indexOf`

String Object

- Ex: Detecting a palindrome string
  - A palindrome is a phrase that is spelled the same both forward and backward. For example:
    - “mom”
    - “Able was I ere I saw Elba.”
    - “A man, a plan, a canal, Panama!”

String Object

- Palindrome algorithm:
  - From both the left and right sides of the string, find the first alphabetic character. Note their indices.
  - Compare the two characters. If they're not the same, it's not a palindrome.
  - Find the next two characters in and repeat the process until the two indices cross in the middle

FRANKLIN UNIVERSITY

String Object

- Palindrome (ctd):
  - Determine if a character is a-z, A-Z

```
function isAlpha(ch) {
  return typeof ch == 'string'
      && ch.length == 1
      && (ch >= 'a' && ch <= 'z' ||
      ch >= 'A' && ch <= 'Z')
}
```

FRANKLIN UNIVERSITY

String Object

```
function isPalindrome(str) {
  var left = 0, right = str.length - 1;
  str = str.toLowerCase();
  do {
    while (left <= right && !isAlpha(str.charAt(left)))
      ++left;
    while (left <= right && !isAlpha(str.charAt(right)))
      --right;
    if (str.charAt(left) != str.charAt(right))
      return false;
    ++left;
    --right;
  } while (left < right)
  return true;
}
```

String Object

- Some common `String` functions

`charAt`	`indexOf`	`lastIndexOf`
`match`	`replace`	`search`
`slice`	`split`	`substr`
`substring`	`toLowerCase`	`toUpperCase`

See the documentation for details!

ITEC 136

Business Programming Concepts

Week 09, Part 04

One-dimensional Arrays

FRANKLIN UNIVERSITY

FOUNDED 1902

One Dimensional Arrays

- What is an array?
  - A single object that holds many other objects within itself.
  - Each object is associated with an index numbered [0, length).
  - Use "square brackets" (i.e. [and]) to access elements at a particular index.

One Dimensional Arrays

- What is an array?
  - A single object tha ther objects within itse
  - Each object is assoc ted with an index numbered [0, length).
  - Use “square brackets” (i.e. [and]) to access elements at a particular index.

Zero based indexing and right-bound not included!

One Dimensional Arrays

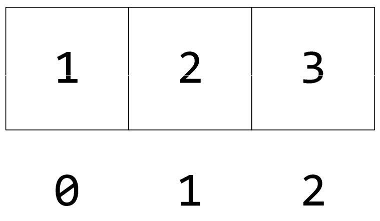
- Creating an array

```
// two ways to create an empty array
var arr1 = new Array();
var arr2 = [];
```

One Dimensional Arrays

- Creating an array with initial data

```
// two ways to create and initialize an array
var arr1 = new Array(1, 2, 3);
var arr2 = [1, 2, 3];
```



One Dimensional Arrays

- Reading and writing elements in an array

```
// reading and writing elements in an array
var arr = [1, 2, 3];     // create the array
var element0 = arr[0];   // puts 1 in element0
arr[1] = arr[2] + 2;     // overwrites 2 with 5
arr[3] = 9;              // adds a new element
```

One Dimensional Arrays

- Length property of an array
  - The number of elements in an array is always available through the property called "`length`"
  - As elements are added, the length property increases.

One Dimensional Arrays

- Fill an array with user input

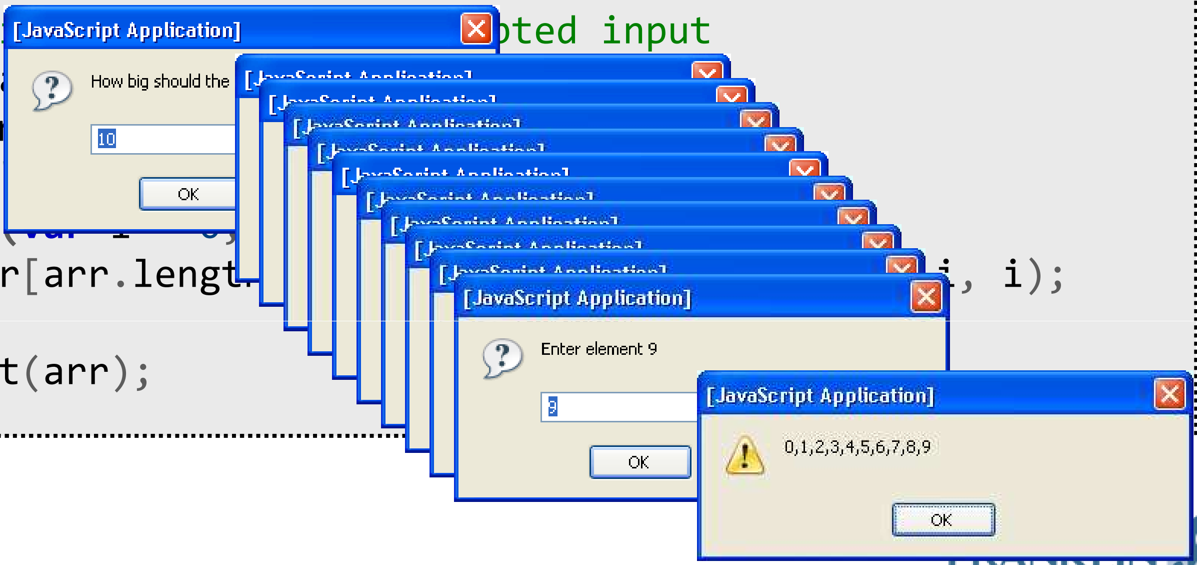
```
// fill an array with prompted input
var arr = new Array();
var max = parseInt(prompt(
    "How big should the array be?", 10));
for (var i = 0; i < max; ++i) {
  arr[arr.length] = prompt("Enter element " + i, i);
}
alert(arr);
```

One Dimensional Arrays

- Fill an array with user input

```
// fi                         pted input
var a
var r

for (var i = 0;
  arr[arr.lengt                                   i, i);
}
alert(arr);
```



One Dimensional Arrays

- Processing arrays
  - Usually using "`for`" loops.

```
// add 5 to each element of an array
var arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
for (var i = 0; i < arr.length; ++i) {
    arr[i] = arr[i] + 5;
}
```

One Dimensional Arrays

- Enhanced for loop

```
for (var index in arrayVariable) {
    var element = arrayVariable[index]
    // do something with element
}
```

One Dimensional Arrays

- Enhanced for loop

```
for (var index in arrayVariable) {
    var element = arrayVariable[index]
    // do something with element
}
```

"index" is assigned a new value each time through the loop

The array from which to pull elements.

One Dimensional Arrays

- Enhanced for loop

Equivalent code using standard for loops:

```
for (var index in arrayVariable) {
    var element = arrayVariable[index]
    // do something with element
}
```

```
for (var i = 0; i < arrayVariable.length; ++i) {
    if (i in arrayVariable) {
        var element = arrayVariable[i];
        // do something with element
    }
}
```

One Dimensional Arrays

- Enhanced for loop

Equivalent code using standard for loops:

```
for (var index in arrayVariable) {
    var element = arrayVariable[index]
    // do something with element
}
```

```
for (var i = 0; i < arrayVariable.length; ++i) {
    if (i in arrayVariable) {
        var element = arrayVariable[i];
        // do something with element
    }
}
```

The keyword "in" tests to see if the index exists in the current array.

One Dimensional Arrays

- Exercise: Array filtering
  - Given an array that contains a set of data, write a function that will return an array containing data that matches a specific criterion.

One Dimensional Arrays

- Exercise: Array filtering
  - Step 1: Write a function that receives three parameters: min, max, and length. The function should create an array of the given length. It should then populate the array with random integers between min and max. Finally, it should return the array.

One Dimensional Arrays

- Exercise: Array filtering
  - Step 2: Write a function that receives the array created in Step 1 as a parameter. This function should walk through the array, copying out those elements that meet a criterion (say, are at least three digits and are evenly divisible by seven) into a second array. Return that array.

One Dimensional Arrays

- Exercise: Array filtering
  - Step 3: Extract the criterion into a separate “predicate” function from that written in Step 2. Call this function to determine if the criterion is met.

One Dimensional Arrays

- Exercise: Array filtering
  - Step 4: Modify the function in Step 2 again to receive the predicate function as a parameter.

One Dimensional Arrays

- Common array operations
  - Searching – next time
  - Sorting – next time
  - Filtering
  - Splicing
  - Enqueue/dequeue
  - Push/pop

One Dimensional Arrays

- Some common `Array` functions

`concat`	`join`	`pop`
`push`	`reverse`	`shift`
`slice`	`slice`	`splice`
`sort`	`unshift`	

See the documentation for details!

FRANKLIN UNIVERSITY

ITEC 136
Business Programming Concepts

Week 09, Part 05

Build Your Own Objects

FRANKLIN UNIVERSITY

FOUNDED 1902

Custom JavaScript Objects

- How can a “custom” object be created?
  - Use the `Object` class!

```
// two ways to create an empty Object
var obj1 = new Object();
var obj2 = { };
```

FRANKLIN UNIVERSITY

Custom JavaScript Objects

- Creating custom object properties

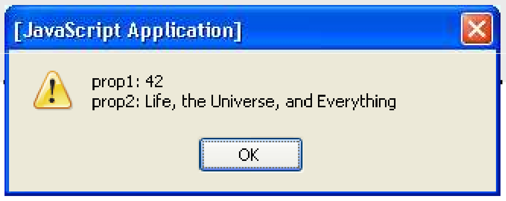
```
// two ways to create Object properties
var obj = new Object();
obj.prop1 = 42;
obj["prop2"] = "Life, the Universe, and Everything";
```

FRANKLIN UNIVERSITY

Custom JavaScript Objects

- Treating custom objects as arrays

```
// Objects treated like arrays
var str = "";
for (var prop in obj) {
    str += prop + ": " + obj[prop] + "\n";
}
alert(str);
```



```
prop1: 42
prop2: Life, the Universe, and Everything
```

FRANKLIN UNIVERSITY

Questions?

FRANKLIN UNIVERSITY

ITEC 136

Business Programming Concepts

Week 09, Part 06

Self Quiz

FRANKLIN UNIVERSITY

FOUNDED 1902

Self Quiz

- List three methods (functions) of the String class and what they do.
- Write a function that receives a string as a parameter and reverses the string (i.e. "foo" -> "oof")
- How are methods different from functions?

FRANKLIN UNIVERSITY

Self Quiz

- Name and define the three properties of every object
- Write a function that takes an array of strings and concatenates them together using a given delimiter (i.e. [“hello”, “cruel”, “world”] -> “hello-cruel-world” when ‘-’ is the delimiter). Return the string.

FRANKLIN UNIVERSITY

Self Quiz

- How is the enhanced for-loop different from a standard for-loop?
- What does the keyword “in” do?
- Where can you find the documentation on each JavaScript built-in object?

FRANKLIN UNIVERSITY

ITEC 136

Business Programming Concepts

Week 09, Part 07

Upcoming deadlines

FRANKLIN UNIVERSITY

FOUNDED 1902

Upcoming Deadlines

- Exam 2 – in class next week 3/9
- Reflection paper 2 – due 3/9
- Lab 3 – due 3/16
- Pre-class exercise 11 – due 3/16
- Homework 8 – due 3/16