JS

Pull from upstream!

Commit any changes first!

Slides created in conjunction with: Ilchul Yoon, Nelson Padua Perez

Agenda

- Office Hours Updates
- Exercise 1 Questions
- Quick Warm Up
- Intro to JS
- JS Logistics

Office Hours

My Office hours tomorrow are cancelled, all other office hours are going on.

Logistics

Exercise 2 will be posted Wednesday/Thursday

Exercise 1 grades will be posted by the end of the week!

Warm Up!

Write the JavaScript to do the following:

- Create a variable using var
- Create a different variable using let
- Display variable1 * variable2 in a pop up

Functions

- Functions are Objects
- Name of a function is a reference value

Classic way to create a function:

```
function name (params){
    statements
}
```

Logistical Items:

- Functions are invoked by using the () operator
- Don’t use var for parameters (e.g. function print(x, y))
- Parameters are passed by value
- There is no mandatory main function
- Returning values via *return*

How can I create a function?

1. With a function declaration

2. Function Expression

3. Using a function constructor

Arrow Functions

- Alternative to anonymous functions
  - “Lambda Expressions”
- Rely on the => operator
- Format
  - Parameters => code
  - Parenthesis for parameters are only required if the function has no parameters or 2 or more parameters. Function with one parameter do not require parenthesis surrounding the parameters
  - If code is a single expression no curly braces nor return statement are required

Practice

Write a function to take a word, w, and a number n,, then return a string with the word, n number of times.

Fun Functions Facts

- Functions can be passed and returned from other functions

- Default Parameters
  - Created formally in ES6
  - Named parameters will have default values if no parameter is passed

Rest Operator

- Collects remaining items of iterable into an array
- Uses a triple dot prefix (…x)
- Added in ES6

- Rest parameters
  - Rest operator appears at the end of the parameters list; it will receive all remaining parameters
  - Stores the remaining parameters as an array

Spread Operator

- Also, added in ES6

- Opposite of rest operator
  - Converts items of an iterable (e.g., array) into arguments (for a function call) or into elements of array
  - Uses triple dot (exactly like rest operator)
  - Can appear anywhere (not just at the end)
  - Can be used inside array literals

Arrays (One Dimensional)

- Collection of values that can be treated as a unit or individually
  - a special type of objects
  - var a = new Array(4);
- As usual, access elements using []

Arrays can be of any type, and can even contain mixed type elements.

Creating Arrays

Example: arraysOneDim.html, arraysLengthProp.html

- Literal form

- Using the Array Constructor

Two Dimensional Arrays

Example: arraysTwoDim.html

- JavaScript does not support actual two-dimensional arrays
  - Can simulate two-dimensional arrays by using an array of arrays

- About two-dimensional arrays
  - Can be passed to or returned from functions like one-dim arrays
  - Any modifications in the function will be permanent
  - You can have ragged arrays

Destructuring

- Destructuring
  - A destructuring assignment allow us to unpack values from arrays, or properties from objects, into distinct variables
  - Very similar to spread (in a way)

Example: destructuring.html

String Methods

- Comparison based on < and >
- concat
  - returns a new string representing concatenation of strings
- includes
  - determines whether one string is found within another
- startsWith
  - whether string begins with characters from another string
- endsWith
  - whether string ends with characters of another string
- indexOf
  - index of first character in string (or -1 if not found)
-

More String Methods

- repeat
  - returns string repeated n times
- splice
  - extracts section of string
- split
  - splits a string into array of strings
- lastIndexOf
  - index of last occurrence of character in the string (or -1 if not found)
- toLowerCase
- toUpperCase
- trim – trims whitespaces

Getting Characters from Strings

- The function charAt or [] allows us to retrieve the character associated with a particular index position in a string.
  - Access is similar to array indexing (first character at 0).

Example: charAt.html

Array Methods

- **fill** - fill elements of an array
- **concat** - returns copy of joined arrays
- **indexOf** - returns position of element in array
- **join** - returns string with all elements in the array
- **pop** - removes & returns last element
- **push** - adds to the end (returns length)
- **reverse** - reverses the array
- **shift** - removes & returns first element
- **unshift** - adds new element to the beginning

Array Methods

- **slice** – selects elements in an array, as a new array
- **splice** – adds and/or removes elements from an array
- **forEach**
  - Calls a provided callback function once for each element in an array in ascending order.
  - Not invoked for index properties that have been deleted or are uninitialized

Example: arrayMethods.html, arraySlice.html, sorting.html

Example: instanceOf.html

Typeof vs instanceof

- **typeof**
  - Returns “object” for all reference types

- **instanceof** operator
  - Returns true if a value is an instance of the specified type and false otherwise
  - instanceof can identify inherited types

Every Object is an instance of an Object!

for -- of

- Works on objects that have a method that returns an iterator

- Creates a loop iterating over iterable objects, including:
  - built-in String
  - Array
  - Array-like objects
  - TypedArray
  - Map
  - Set, and
  - user-defined iterables.

Template Literals

- String literals that allow embedded expressions.
- Can replace placeholders in text with vars or exprs
- Defined using the backtick character
  - `embedded string expression`
  - Placeholders identified with ${varName}
  - To escape a back-tick in a template literal, use backslash before the back-tick.
- Simpler for multi-line strings
  - Spaces now matter!

Null vs Undefined

- Null
  - a value indicating no value (nothing)
  - Type: “object”
- Undefined
  - Value associated with uninitialized variables
  - Type: “undefined”
  - Example:
    - var x; // in a function
  - Value returned by function when no explicit value is returned (IMPORTANT case)
  - Value associated with object properties that do not exist

Truthy vs Falsy

- A falsy value is:
  - A value that is considered as false in a Boolean context
  - Falsy values are:
    - False
    - 0
    - “”
    - Null
    - Undefined
    - NaN
- A truthy value is:
  - A value that is considered as true in a Boolean context
  - All values are truthy unless they are defined as falsy

Other Notable Methods

- Math.random()- Random number
- isNaN()- Attempts to convert to a number
- Number.isNaN()- Better version, tests before converting
- isFinite()- What makes this false?
- Number.isFinite()- Similar to Number.isNaN() in execution

Debugging

- Select Inspect after loading the script, and Sources. This will open the debugger.
- Click on a source line to set a breakpoint.

- Alternatively, you can add in your code the statement

```
debugger;
```

which will invoke the debugger when you run the script

WTWAW

After today make sure you know:

- How to Create functions in 4 ways
- How to create arrays
- How to use methods to manipulate a String
- Be able to embed data using String literals