# CMSC335

Web Application Development with JavaScript

## JavaScript III

Department of Computer Science

University of MD, College Park

Slides material developed by Ilchul Yoon, Nelson Padua-Perez

## One-Dimensional Arrays

- **Array:** Collection of values that can be treated as a unit or individually
  - **A special type of object**

```
let a = new Array(4);
```

- **Indexing:** access an element using [ ]
  - First element associated with index 0 (e.g., a[0])
- An element of an array can be of any type, and an array can hold different types of elements
- The **length** property represents the length of the array (e.g., a.length)
- Try printing the contents of an array by using **alert**

## Definition of One-Dim Arrays

- **Using the literal form**
  - Comma separated list of elements within square brackets

  let a = [2, 3, 5];

  let b = []; /* Empty array */
- **Using Array constructor**

  let c = new Array();

  let d = new Array(4); /* Defines array of size 4 */
- **Example:** ArraysOneDim.html, ArraysLengthProp.html

## Two-Dimensional Arrays

- Can be passed to or returned from functions like one-dimensional arrays
- Any modifications to the array in the function will be permanent
- You can have ragged arrays
- **Example:** ArraysTwoDim.html

## for...of Statement

- for…of - Creates a loop iterating over iterable objects. 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
  - Map
  - Set
  - User-defined iterables
- **Example:** ForOf.html

## for...in Statement

- for…in - Allow us to iterate over the properties of an object
- **Example:** ForIn.html

## Getting String Characters

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

```
let x = "Wednesday";
let secondCharacter = x.charAt(1); /* Variable has "e " */
let lengthOfString = x.length;        /* Variable has 9 */
```

- **Example:** CharAt.html

## Template Literals

- String literals that allow embedded expressions
- Can replace placeholders in text with variables or expressions
- Defined using the **backtick** character (character below ~ in keyboard)
  - **`embedded string expression`**
  - Placeholders identified with ${expression}
    - » ${x}, ${x * y}
  - To escape a back-tick in a template literal, use a backslash before the back-tick
- **Simpler for multi-line strings**
  - Space matters
  - **Example:**

```
const string = `Hello
    terps!`
```

- **Example:** TemplateLiteral.html

## NaN

- NaN
  - Generated when arithmetic operations result in undefined or unrepresentable value
  - Generated when attempting to coerce to numeric values non-numeric values for which no primitive numeric value is available
- **Global isNaN function** (i.e., window.isNaN())
  - Determines (returns true or false) whether an argument is not a number
  - **It attempts to coerce the argument to a number (assume it is executing Number() on the expression before evaluating it)**
  - Interesting cases:
    - » isNaN({}), isNaN([]), isNaN([389]), isNaN(true), and isNaN(false)
- **Number.isNaN()**
  - More robust version of isNaN() and the one we should be using
  - **Compares a value to NaN only if the value is a Number–type value**
    - » Return false for all other cases

## NaN

- The following comparisons return false

  NaN == NaN

  NaN === NaN

- **Remember**
  - **! isNaN()** allow us to determine whether an expression is a number
    - » isNaN(20) : false
  - You may want to write a function called **isNumber** that returns
    ! Number.isNaN(x)
- When looking at the following examples, do not think of NaN as “not a number”, but as a special value named **NaN**
- **Example:** NaN.html

## String Methods

- **Comparison based on < and >**
- **concat** - returns a new string representing the concatenation of strings
- **includes** - determines whether one string is found within another
- **startsWith** - determines whether the string begins with characters from another string
- **endsWith** - determines whether the string ends with characters from another string
- **indexOf** - index of the first character in the string (or -1 if not found)
- **lastIndexOf** - index of the last occurrence of a character in the string (or -1 if not found)
- **repeat** - returns string repeated **n** times
- **splice** - extracts a section of a string
- **split -** splits a string into an array of strings
- **toLowerCase/toUpperCase**
- **trim** - trims whitespaces
- **Example:** StringMethods.html
- **Reference**
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String

## Array Methods

- **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 a new element to the beginning
- **splice -** removes elements from an array (modifies original array); returns retrieved elements. When two index arguments, second argument is inclusive
- **slice** - copies (shallow copy) elements from an array (does not modify original array); returns array. When two index arguments, second argument is **not** inclusive

## Array Methods

- **concat** - returns a copy of joined arrays
- **fill** - fill elements of an array
- **Example:** ArrayMethods.html, ArraySlice.html (after opening console, execute the script again to see the array in table format)

## Sorting

- **Example:** Sorting.html

## JavaScript Errors

- You may get a blank page when there is an error
- Use the console to see the error
- Additional debugging information:
- http://www.cs.umd.edu/~nelson/classes/resources/JavaScript/JavaScriptDebugging/
  - **Example:** ErrorHandlerEx.html, ErrorHandler.js

## JavaScript References

- Excellent source of information

  https://developer.mozilla.org/en-US/docs/Web/JavaScript

- Equivalent of Java API

  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

- The previous reference provides excellent examples describing the functionality of methods.  Let's take a look at a couple of methods and the provided examples
- Class web page has a link to the above Reference

  (Resources→JavaScript Reference)