

CMSC335

Web Application Development with JavaScript



JavaScript4

Department of Computer Science

University of MD, College Park

Slides material developed by Ilchul Yoon, Nelson Padua-Perez

Debugging (Chrome)

- **Step #1** - Right-click and select **Inspect** after loading the script. Then select **Sources**. This will open the debugger
- **Step #2** - Click on a source line number to set a breakpoint
- **Step #3** - To the right of the code you will see a pane with debugger controls
 - Under “Scope” you will see local and global values
- **Step #4** - Reload script to start execution
- Resume, step over, step into, step out option can be found next to the code
- Use **Watch** to inspect value of expressions
- Call stack - Allows you to change stack frame

Debugging (Chrome) – `debugger;` Statement

- You can add in your code the **`debugger;`** statement which will stop execution at that point
- To use the **`debugger;`** statement
 - Add **`debugger;`** to your code
 - Run the script
 - Right-click and select **Inspect**, and then **Sources**
 - Run the script
 - Execution will stop at the **`debugger;`** statement

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 or return statement are required
- **Example:** ArrowFunc.html

Objects

- **Property** - association between a name and a value
 - When the value is a function the property is referred to as a method
 - Name can be any valid JavaScript string or anything that can be converted to a String (that includes empty string)
 - » Any invalid property name can only be accessed using square bracket notation
- **Object** - Collection of properties
 - You can define your own; browser predefines a set of objects
 - A property can be seen as a variable associated with a value
 - Approaches to access and add properties
 - » Using dot-notation (e.g., obj.name)
 - » Using square brackets (e.g., obj[" name"])

How to Create Objects

- **Using Object constructor (e.g., new Object())**
 - Object constructor creates an object wrapper for the given value
 - » **Example:** `let x = new Object(true);`
 - If the provided value is null or undefined an empty object will be created
- **Using object initializer/literal notation**
 - An initializer is a list of zero or more property names/values in { }
 - **Example:** `let x = {}, y = { radius: 20 }`
- **Using Object.create**
 - Creates a new object, using an existing object as the prototype of the newly created object
- **Example:** `Objects.html`

Objects as Maps

- We can also view an object as an entity that associates values with strings. How? Let's first see how we can use the [] operator to access properties
 - You can use [] operator instead of . (period) operator

`myObj.created → myObj["created"]`

- IMPORTANT:
 - Notice that we have a string on the right side ("created") whereas on the left side it is a property (variable)
- Using [] operator can provide a nice alternative to add properties to an object dynamically (when the program is executing)
- **Example:** AddingProperties.html

Default Parameters

- Standardized in ES6
- Allow named parameters to be initialized with default values if no value or undefined is passed
 - Literal or computed by a function
- **Example:** DefaultParameters.html

Rest Operator (...variable)

- Rest parameters
 - Rest operator(...variable) appears at the end of the parameters list; it will receive all remaining parameters
 - Stores the remaining parameters as an array
- **Example:** RestOperator.html

Spread Operator

- 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
- **Example:** SpreadOperator.html

Destructuring Assignment

- Destructuring
 - A destructuring assignment allow us to unpack values from arrays, or properties from objects, into distinct variables
- **Example:** Destructuring.html

JSON

- JSON - JavaScript Object Notation
- Text data format used to store and send/receive data
- **Example:**

```
{"firstName":"Mary", "lastName":"Smith", "age" : 30}
```
- Popular format used by APIs to return results
- JSON syntax is derived from JavaScript, but code for generating and reading JSON can be done in any language
- JSON objects are written using { }
- JSON data written as name/value pairs where the name must be in quotes (that is not the case for JavaScript objects). The value can be a string, number, boolean, array, object, etc.
- Arrays are written using square brackets ([])
- **Reference:** https://www.w3schools.com/whatis/whatis_json.asp
- **Example:** JSONExample.html
- See JSON resources (e.g., formatters) at
 - <https://www.cs.umd.edu/~nelson/classes/resources/web/>

Array's forEach, find, findIndex Methods

- **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
 - Callback can have one or two parameters
- **find** - Returns the value of the first element in the provided array that satisfies the provided testing function
- **findIndex**
 - Returns the index of the first element in the array that satisfies the provided testing function. Otherwise, it returns -1
 - Tests if at least one element in the array passes the test
- **Example:** ArrayForEachFindFindIndex.html

Array's map, every, some Methods

- **map**
 - **Creates** a new array with the results of calling a provided function on every element in the calling array
- **Example:** ArrayMap.html
- **every**
 - Tests if all elements in the array pass the test implemented by the provided function. Returns a boolean value
- **some**
 - Tests if at least one element in the array passes the test
- **Example:** ArrayEverySome.html