

CMSC388A

Web Application Development with JavaScript



JavaScript Intro

Department of Computer Science

University of MD, College Park

Slides material developed by Ilchul Yoon, Nelson Padua-Perez

JavaScript

- A lightweight, interpreted, or just-in-time compiled language that can function as both a procedural and an object-oriented language
- Appears in a web browser and non-browser environments (e.g., Node.js, Apache CouchDB)
- It allows us to:
 - To create interactive web pages
 - To control a browser application
 - » Open and create new browser windows
 - » Download and display the contents of any URL
 - To interact with the user
 - Ability to interact with HTML forms
 - Access data from databases and other online resources
- **Example:** SqrTable.html

ECMAScript

- **Ecma International** - Organization that creates standards
 - <https://www.ecma-international.org/>
- **Scripting language** - language that acts on a system or an entity
- **ECMAScript** - specification for a general-purpose scripting language
 - Provides rules that a scripting language must observe to be considered ECMAScript compliant
- **ECMAScript specification**
 - <https://www.ecma-international.org/publications-and-standards/standards/ecma-262/>

JavaScript

- **Javascript** - a general-purpose scripting language that conforms to the ECMAScript specification
- JavaScript is based on the ECMAScript specification
- Reference: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

JavaScript Engine

- JavaScript engines process **JavaScript** code
 - **Safari** - JavaScriptCore
 - **Chrome** - V8
 - **Firefox** - Spidermonkey
 - **Edge** - Chakra
- **Client-Side JavaScript**: the result of embedding a JavaScript engine in a web browser
- A JavaScript program can appear:
 - In a file by itself, typically named with the extension .js
 - In HTML files between a <script> and </script> tags
- **Example**: TemplateJS.html
 - Right-click→Inspect→Console to see console.log() output

“use strict” in the template JS

- JavaScript's strict mode, introduced in ES5
- A way to opt-in to a restricted variant of JavaScript, thereby implicitly opting out of "sloppy mode"
- Several changes to normal JavaScript semantics:
 - Makes JavaScript silent errors throw errors
 - Prohibits some syntax likely to be defined in future versions of ECMAScript
- **Examples (not allowed with strict mode):**
 - Declaring a function in a block
 - » `if (a < b) { function f() {} }`
 - **Setting a value to an undeclared variable**

Processing HTML Page with JS

- **DOM - Document Object Model**
 - Structured representation of the HTML page
 - Every HTML element is represented as a node
 - Browser uses HTML to build the DOM and can fix problems with the HTML, so a valid DOM is generated
- **Lifecycle**
 - **Set the user interface**
 - » Parse the HTML and build the DOM
 - » Process (execute) JavaScript code
 - **Enter a loop and wait for events to take place**
- When JavaScript is seen on a page, the DOM construction is halted, and JavaScript code execution is started
- **JS can modify the DOM** (e.g., creating and modifying nodes)
 - One reason why `<script></script>` elements appear at the bottom of a page

Event-Handling

- **Relies on a single-threaded execution model**
- An **event queue** keeps track of events that have taken place but have not been processed (the event-handler function for the event has not been called)
- All generated events (whether user-generated or not) are placed in the event queue in the order they were detected by the browser
 - The browser mechanism that detects events and adds them to the event queue is separate from the thread that is handling the events
- JavaScript periodically checks the event queue, and if any event is found, it executes the appropriate handler (if one was defined)

Browser's Global Objects

- Browsers provide two global objects: **window** and **document**
- **window** object - represents the window in which a page resides
 - Provides access to other global objects (e.g., **document**)
 - Keeps track of the user's global variables
 - Allows JavaScript to access Browser's APIs
- **document** object
 - Property of the **window** object that represents the DOM of the current page
 - **Via this object, you can access & modify the DOM**

Types of JavaScript Code

- **Function Code**
 - Code contained in a function
- **Global Code**
 - Code placed **outside all functions**
 - Automatically executed by JS engine
- As in Java, a stack keeps track of function calls. Each function call generates a **function execution context** (stack frame)
- There is one frame called the **global execution context** created when the JS program starts executing
 - Only one global execution context (at the bottom of the stack)

JavaScript Comments

- **Comments in JavaScript**

- Used to provide information to the programmer
- Used to identify sections in your code
- Ignored by the JavaScript interpreter

- **Two types of comments**

- Inline comment **// This is a comment until the end of the line**
- Block comment

/* The following is a
comment that spans
several lines ***/**

Variable Declarations

- Variable declaration (no type specification)

var x; /* old (avoid)*/
let x; /* for variables*/
const x; /* for constants*/

- Variables names must start with:
 - A letter, underscore, or dollar sign and then
 - Can be followed by any number of letters, underscores, dollar signs, or digits

JavaScript Data Types

- **JavaScript has no class concept**
 - We have functions (which are objects)
 - Using functions and prototypal inheritance, we can implement the concept of classes
 - **Syntax was added to define classes as you do in Java**, but it is just syntactic sugar (no actual classes as in Java)
- **Two kinds of types**
 - **Primitive types** - data that is not an object and has no methods.
All primitives are immutable
 - **Reference types** - references to objects

JavaScript Data Types

- **Seven (7) primitive data types in JavaScript**
 - **null** - has the value null
 - **boolean** - has the value **true** or **false**
 - **number** - numeric data type using a double-precision 64-bit floating point form (IEEE 754)
 - **string** - character sequence delimited by single, double quotes, or ``
 - **undefined** - value automatically assigned to a variable just declared or to parameters that have no corresponding arguments
 - **bigint** - represents integers in arbitrary precision format (precision limited by the host system)
 - **symbol** - represents a unique identifier (guaranteed to be unique)
 - » `let x = Symbol("A"); let y = Symbol("A"); // x === y is false`
- **typeof** operator
 - Returns string indicating the type of data
 - Note: **typeof "house"** will return **string**

JavaScript Data Types

- Reference types represent addresses of objects
- **Object** - a collection of properties
 - **Property** - a string that is associated with a value
 - **Value** - could be a primitive or reference to an object
- **Object creation**

```
let a = new Object(); // first approach
let b = {};           // second approach
let c = {              // third approach
    id: 789,
    name: "Rose Smith"
};
```
- You could add properties: `a.name = "Rose";`
- **JavaScript relies on garbage collection**
 - When an object is no longer needed, set the variable to null