FRESNO STATE
Discovery. Diversity. Distinction.

Web Programming (CSci 130)

Department of Computer Science
College of Science and Mathematics
California State University Fresno
H. Cecotti

Learning outcomes

- In this class,
  - You will learn about the syntax of **JavaScript**
- **Warning:**
  - Csci 41/115 as a pre-requisite -> You know how to use data structures
- JavaScript will be used until the end of the semester
- "There are only two kinds of languages: the ones people complain about and the ones nobody uses."
  - And there are many reasons to complain about JavaScript

Beyond Web Programming

- JS is important for many reasons

  - Many things can be done with JS
    - … some that should be ideally done with something else
  - Jeff Atwood (creator of Stack Overflow) pretty much predicted this phenomenon back in the 2007. He coined Atwood's law:
    - *"Any application that can be written in JavaScript, will eventually be written in JavaScript"*

Introduction

- JavaScript
  - Make webpages alive
  - It is not Java
  - Script – Live script
    - Written in the HTML page, executed automatically when the page loads
    - Origin: script for webpages … Java was popular → JavaScript
      - **Independent from Java**
      - Specifications: ECMAScript (11th Edition – ECMAScript 2020)
        - No relation to Java !!!
  - Script to be executed
    - On the browser
    - On the server
    - Any device where there is **JavaScript engine**

Introduction

- **JavaScript**
  - To understand the script
    - Browser + JavaScript Engine (JavaScript virtual machine)
      - V8: chrome, opera
      - Gecko: Firefox
      - MS Edge Internet explorer: ChakraCore
  - Engine
    - Reads the script
    - Compiles the script to the machine language
      - Optimize the code
    - Run the code
- **Key features**
  - Full integration with HTML/CSS
  - Easy implementation (high level language)
  - Supported by all the main browsers

Introduction

- **JavaScript**
  - Different versions
    - ECMAScript 1 (1997)
    - ECMAScript 5 (2009)
      - Strict mode
    - ECMAScript 2015
    - ECMAScript 2018
    - ECMAScript 2020 (June 2020)
      - → Addition of BigInt primitive for arbitrary-sized integers
  - Each version → addition of new function/functionalities
    - Array.map() in ECMAScript 5 for the functional programming lovers ☺
    - Be careful when you check tutorials, information online
      - Lots of dirty code

JavaScript for what?

- Safe programming language
  - No low-level access to memory or CPU
  - Depends on the environment running JS
    - Node.JS for read/write files…
- JS functionalities in the browser
  - Add new HTML content to the page, modify the style…
  - Answer to user actions, mouse clicks, key presses…
  - Send requests over the network
    - Remote servers
    - Download/upload files (AJAX/COMET)
    - Get/set cookies, show messages
    - Save information on the client side (local storage)

What JS cannot do

- Limitations to JS → for security
  - ➢On a webpage, cannot read/write arbitrary files on the HD
    - Copy, execute programs
    - No access to OS functions
    - Use files via <input>
    - Camera/Microphone → user's permission
  - ➢Each page is independent
    - Cannot access the information from other sites
      - Same origin policy (to not steal information between websites)
        - Special data exchange between pages

JS Code editors

- IDE for JS (editor + special features)
  - IntelliJ
  - Visual Studio (.NET)
  - Eclipse product
  - Netbeans
- Simple editors
  - Notepad++
  - Atom
  - **Visual Studio Code**
  - Emacs…
- On Chrome, F12 → developer tools

JS syntax

- In the HTML page
  - In the body
    - <script> JS code </script>
  - External script
    - <script src="/path/to/script.js"></script>
    - Path to the file from the site root
    - You can give the complete address of the file if you wish: “http:// ….. Myscript.js”
    - You can use several scripts from different files
      - <script src="/jscript/myscript1.js"></script>
      - <script src="/jscript/myscript2.js"></script>
  - Readable code
    - → simple script in the HTML code
    - Big blocks of codes with many functions in files
  - With the script tag: link to the file OR js code !

Code structure

- Comments
  - // It is a comment
  - /* it is another comment */
    - Nested comments like /* */ don't work
- Statements
  - alert('Hello world');
- Semicolons: can be omitted
  - New line → implicit semicolon (most of the time)

Old and new JS

- Evolution of JS
  - → compatibility issues
    - Addition of new features
    - **2009** (old/new features)
- "use strict"; (at the top of the JS code)
  - → Hence, you have to declare variables
  - Modern way
    - → always use it ☺
    - You have to define the variables !

Variables in JS

- Declaration: let x; x="Hi";
  - No specification of the type
  - Specification of the fact it is a variable
- Old scripts: var
  - No block stop
    - Visible through blocks ("global")
  - Processed at the beginning of a function
- Variable names
  - Cannot start with a number
  - Hyphen: not allowed
  - Not: let/class/return/function

```
<!DOCTYPE html lang="en">
<head>
<meta charset="UTF-8" />
<title>My page with Javascript</title>
</head>
<body>
<p>Example of script</p>
<script>
var x = 5;
// x is 5
{
    let x = 8;
    // x is 8
}
// x is 5
</script>
</body>
</html>
```

Constants

- const today='9.18.2017';
  - Cannot be changed
  - today='9.19.2017'; → error
- Use it to define colors, elements that wont change
  - Alias of colors
- Names
  - Pick the names so when you read it you know what it is !!

Data types

- JS variable
  - Can contain any data
  - Initialized with a string, then later become a number,
    - It is possible!
- Number: let x=5; x=3.1415956;
  - Regular operations with numbers
    - Infinity, NaN
  - Math operations
    - Safe (it will catch errors)
      - Division by 0

Data types

- Strings
  - Between quotes (simple, double, backsticks);
  - Embed variable in a string
    - ${variable name}
  - Evaluate an expression in a string
    - ${5*5}
- Boolean
  - true/false
- Special values
  - Null: ref to nothing (non existing object)
  - Undefined: let i; alert(i);
- Objects and symbols

- **Typeof** operator
  - typeof undefined // "undefined"
  - typeof 0 // "number"
  - typeof true // "boolean"
  - typeof "foo" // "string"
  - typeof Symbol("id") // "symbol"
  - typeof Math // "object" (1)
    - Built in object providing math operations
  - typeof null // "object" (2)
    - Not an object, error in the language… oops
  - typeof alert // "function" (3)
    - Alert function of the JS language
  - Typeof …
    - typeof NaN
      - Number
    - typeof Infinity
      - Number

Conversion

- Conversion
  - ToString
    - Null → “Null”
    - True → “true”
  - ToNumber
    - Empty string → 0
    - Error = NaN
    - True → 1 , False → 0
    - Null → 0
  - ToBoolean
    - Empty string, 0, nan → false
    - Something → true

- **Warning**
  - Examples
    - 5+5
      - 10
    - 5 + '5'
      - ‘55’
    - '5' + 5
      - ‘55’
    - 5 + +'5'
      - 10
    - '5' - - '5'
      - 10

Operators

- Unary
- Binary
  - Concatenation of strings: +
  - Addition: +
- Operand
  - Operators precedence
    - Same as in primary school
  - Assignment: =
    - let x=5;

Double and Triple equals (== vs. ===)

- ===
  - Testing for **strict** equality
  - → both the **type** and the **value** we are comparing have to be the same
    - 5 === 5 // true
    - 77 === '77‘ // false (Number v. String)
    - 'cat' === 'dog‘ // false (Both are Strings, but have different values)
    - false === 0 // false (Different type and different value)
- ==
  - Testing for **loose equality**
  - == performs **type coercion**
    - 77 == '77' // true
    - false == 0 // true

Double and Triple equals

- Special cases
  - false == 0 // true
  - 0 == "" // true
  - "" == false // true
  - null == null // true
  - undefined == undefined // true
  - null == undefined // true
  - NaN == null // false
  - NaN == undefined // false
  - NaN == NaN // false

Math operators

- Remainder % (modulo)
- Exponentiation **
  - 2**3 = 8
  - 4**(1/2) = 2 (square root)
- Increment, decrement
  - Like in C++
  - counter++ / ++counter
    - Before: do it first then assign
    - After: assign then do it

Interaction – basic input/output

- Alert
  - alert(message)
- Console

- Prompt
  - Myresult=prompt(title[,default]);
    - Modal window with a text message
      - + buttons OK/Cancel
    - Title: text shown to the user
    - Default: initial value of the input field
      - let age = prompt('How old are you?', 12);
      - alert(`You are ${age} years old!`); // You are 12
- Confirm
    - let isPresent = confirm("Are you present?");
    - alert(isPresent); // true if OK is pressed

Operators

- Conditional operators
  - IF

    ```
    If (a==10) { alert('something');}
    ```
  - IF ELSE

    ```
    If (a==10) { alert('something');}
    Else if (a==5) { alert('something');}
    ```
  - let x= (a==20) ? 5 : 10;
  - SWITCH

    ```
    switch (x) {
        case 'v1' : code1 [break]
        case 'v2': code2 [break]
        …
        [break]
        }
    ```

- Logical operators
  - Or ||
  - And &&
  - Not !

Loops

- While
  - while (condition) {code }
- Do while
  - do {code} while (condition)
- For
  - For (start;condition;step) { code }
  - break;
  - continue;

Functions

- function nameoffct(list of parameters) { code }
  - Local variables in the function
  - List of parameters
    - function f1(a,b,c) {}
    - Default values f(a,b=0)
      - Not given = undefined
- Readable code
  - The name of the function means something
    - Show, Get, Set, Create, Init, Display …
    - When return a Boolean
      - IsChecked(), IsComplete, IsFilled, …
  - Short functions
- See examples

Array

- See files on Canvas
  - class_javascript_array.html
  - class_javascript_array.html (dynamic array)
  - class_javascript_array_map.html

- Advice
  - Practice & Work on personal project
    - Portfolio to find a job !

What's next?

- How to link the content of the HTML page with JS code
  - How to create dynamic pages through JS
    - Example: creation of tables of size 20 x 20
    - How to fill tables with special values
    - …
  - Solution: **DOM**
    - Document Object Model
    - Next session

Conclusion

- **JavaScript**
  - Primarily Client side
    - Node.**js** and its frameworks → you can use **Javascript** in both **client side** and **server side**
      - to produce dynamic web page content **before** the page is sent to the user's browser
  - Powerful to enable dynamic alive webpages
  - Most widely adopted browser language with complete integration of HTML/CSS
- Debugging
  - **GO STEP BY STEP → HARDER TO DEBUG**
    - Test the code, function by function !!!
      - Hard to debug
    - Think about both the **values** and the **types** of the variables
- Other languages that are transformed into JS
  - CoffeeScript
  - Typescript (MS)

Concluding remarks

- **Warning**
  - ‘+’ sign
    - Arithmetic addition when used with numbers
    - String concatenation when not both operands are numbers
    - “implicit typecasting” (to string)
      - source of confusion in JS
    - Any object may have .toString() method
    - Any value is an object in JS.
    - Strings get converted to numbers automatically due to (arithmetic) context
  - Double equal vs. Triple equal
    - == doesn’t check value type
    - === does check value type