香港中文大學
The Chinese University of Hong Kong

CSCI2720 - Building Web Applications

Lecture 5: JavaScript Basics

Dr Colin Tsang

Outline

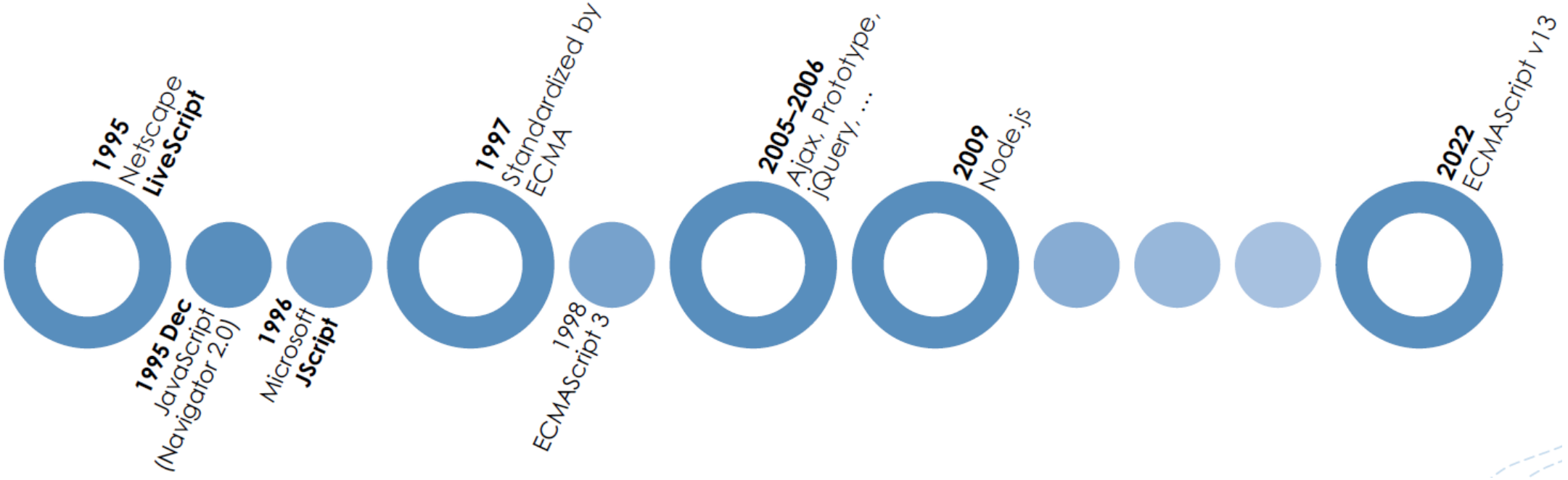
- Why JavaScript?
- Using JavaScript
- Identifiers and variables
- Data types and operators
- Arrays
- Condition and loops
- Functions
- Browser window

Why JavaScript?

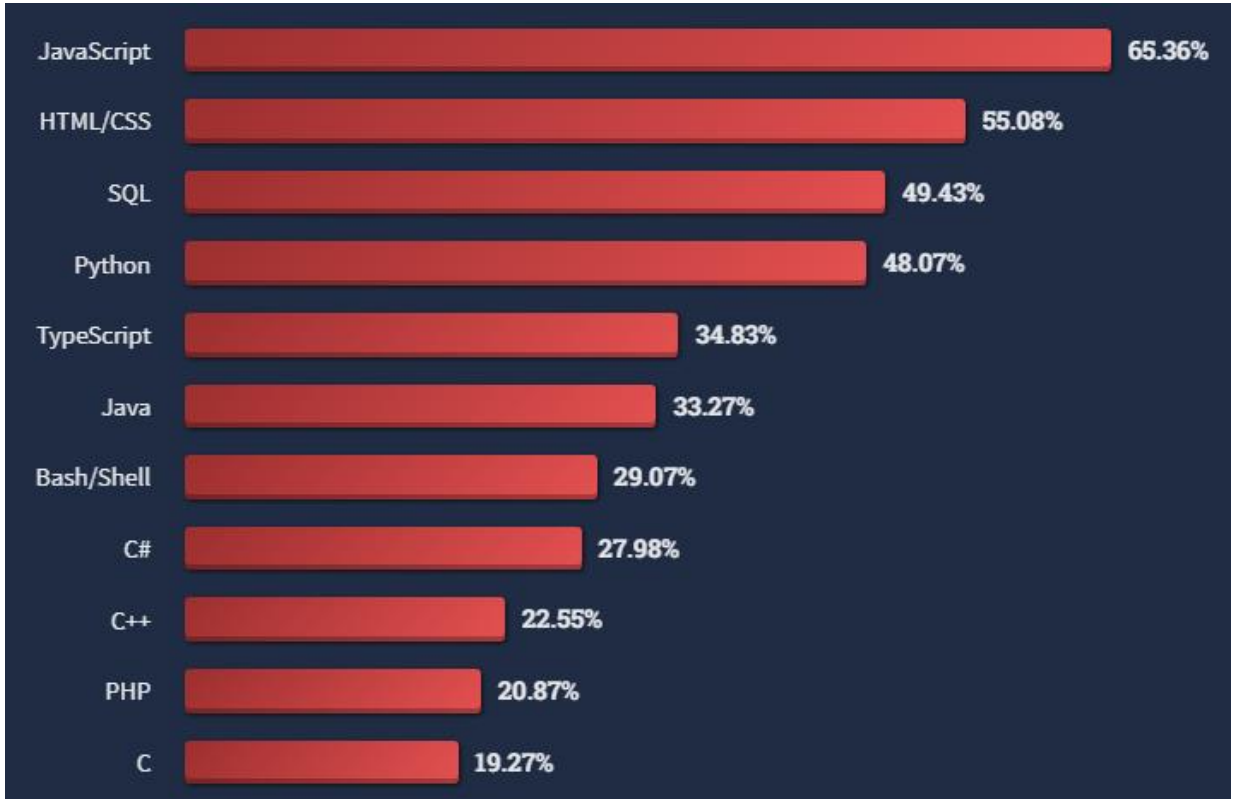
- The programming language of the web
  - Every element being rendered in the browser *can be generated and manipulated* with JavaScript.
  - Beyond the browser, now JS can also be used to set up a (web) server, build mobile apps, or even in platforms outside the web.

- Evolution together with various web technologies.

A brief history of JavaScript



Most commonly used programming language



See: https://survey.stackoverflow.co/2022/

What can JavaScript do?

- Form validation before submission
- Interactivity in web pages: changing appearance basing on events
  - e.g., changing page content color on certain actions
- Extra communication with the web server
  - e.g., loading/showing contents on scrolling
- Drawing in the HTML canvas

- And a lot more……

What cannot JavaScript do?

- JavaScript “lives” in the browser.
- JavaScript is *bounded* by the browser runtime environment.
  - No direct file system or memory access
    - Except when user explicitly selects a file to open
  - No access to hardware devices unless explicitly granted
  - Can only communicate over browser ports or protocols, e.g., HTTP/HTTPS
- We only discuss client-side JavaScript in this lecture

Other characteristics of JavaScript

- Interpreted, or just-in-time compiled language
- Single-threaded
- Multi-paradigm: object-oriented, imperative, functional

- How to learn it well? Learn from *examples*!
- Or: https://www.w3schools.com/js/

Using JavaScript

- Two ways to execute JS code:
  - Linking to a `.js` file (usually in the HTML head)
  - `<script src="myscriptfile.js"></script>`
  - Easier separated maintenance

- Embedding code in HTML.
  - The code is executed when page is loading

```
<script>
  document.write("Hello world!");
</script>
```

- Usually put at the end of body (i.e., before </body>). Why?

Using JavaScript

- You can test JavaScript directly in JavaScript Console in major browsers
  - Chrome: *F12 / CTRL+SHITF+J*
    - Or *View > Developer > Developer Tools*

- Outputting message or errors
  - To the console: *console.log(…)*
    - Learn more:
    - https://www.freecodecamp.org/news/javascript-console-log-example-how-to-print-to-the-console-in-js
  - To an alert box: *window.alert(…)*
  - To HTML output: *document.write(…)*

Using JavaScript

- All tabs in the browser have separate execution space
  - Browser dependent
- If you run something in the JS console, it is in the context of the current visited page
- If you are worried that you might mess up the page, you can use a blank page at this address:
  - *about:blank*

Identifiers and variables

- Identifiers names
  - Case-sensitive
  - Letters, digits, underscores, dollar sign
  - Cannot start with a digit
  - A list of reserved words that cannot be used

- Variables
  - They can be declared using:
    - *var*: old-fashioned, can be updated and redeclared
    - *let*: more preferred, can be updated but no redeclaration
    - *const*: cannot be updated or redeclared

can be declared again / changed

- *Undeclared variable* are created as *globals*.
- https://www.freecodecamp.org/news/var-let-and-const-whats-the-difference/

Data types

- JS primitive types:
  - **string**: textual data, enclosed by ‘’ or “”, first element at index 0 (array of characters).
  - **number**: double-precision 64-bit, including *infinity* and *NaN*
  - **bigint**: a new type, allowing arbitrarily large numbers over 2^53-1
  - **boolean**: true or false. Anything not *0, null, false, NaN, undefined, “”* are true.
  - **undefined**: an auto value assigned to variables just declared
  - **symbol**: a special piece of private data created using Symbol()
  - **null**: actually an object, representing a non-existent or invalid object or address

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

Data types

- Use *typeof* to check data type.

- JS is dynamically typed
  - The same variable can be used to store different types of data

```
> let x
< undefined
> typeof x
< "undefined"
> typeof 123
< "number"
> typeof "hello world"
< "string"
```

Data type conversion

- When adding a number and a string, the number is treated as a string.
  - JS evaluates expressions from left to right.

- Converting a value to a number:
  - *Number();*

- Converting a value to a string:
  - *String();*

- Converting a value to a Boolean:
  - *Boolean();*

```
> let x = 1 + 2 + 3
< undefined
> x
< 6
> let y = 1 + '2' + 3
< undefined
> y
< "123"
```

Strings

- Strings in JS are quoted by ‘’ or “”
- Commonly used escape sequences:
  - \’
  - \”
  - \\
  - \n
- Strings contents can be compared directly using <, >, ==, etc.
  - JS compare strings lexicographically by the Unicode value, similar to Python.
- String characters can be accessed like array contents.
  - `let stringVar = "abcde";`
  - `stringVar[0]` is “a”

☆ Index starts from 0.

Strings

- Notable string methods (and many more!):
  - *trim()*
  - *split(text)*

```
> let variableStr = "  abcdefg    "
< undefined
```

Delete space

```
> variableStr.trim()
< 'abcdefg'
```

seperator

```
> variableStr.split('c')
< ▶ (2) ['  ab', 'defg    ']
```

- JavaScript encodes text in UTF-16
  - Slightly different from the usual UTF-8
  - But most character are still well supported, even emojis.
  - https://dmitripavlutin.com/what-every-javascript-developer-should-know-about-unicode/

Strings

- String can be easily manipulated with RegEx.
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions
- `const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;`
  - Create a RegEx between / /
  - `^` asserts the start of the string
  - `[^\s@]+` matches one or more characters that are not whitespace or @ symbol
    - `[^abc]` matches any single characters that is not a, b, or c.
    - `[^abc]+` matches one or more characters that is not a, b, or c.
  - `@` matches the @ symbol
  - `[^\s@]+` again
  - `\.` matches the dot character
  - `[^\s@]+` again
  - `$` asserts the end of the string

String interpolation

- This looks similar to other programming languages:

```
const a = 5;
const b = 10;
console.log("Fifteen is " + (a + b) + " and\nnot " + (2 * a + b) + ".");
// "Fifteen is 15 and
// not 20."
```

- But, you can also do this for an equivalent result using *backtick* ` and *${}*: ⇒ printf()

```
const a = 5;
const b = 10;
console.log(`Fifteen is ${a + b} and
not ${2 * a + b}.`);
```

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

Operators

- Arithmetic operators
  - +, -, *, **, /, %, ++, --
- Assignment operators
  - =, +=, -=, *=, /=, %=, **=
- Comparison operators
  - ==, !=, >, <, >-, <=
  - === : equal value and type
  - !== : unequal value or type
- Logical operators
  - &&, ||, !
- Bitwise operators
  - &, |
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators

Arrays

- An array is a list of items, which can be of mixed types
- Index starts at 0
  - Same as: Python, C, C++, Java, …
  - Unlike: MATLAB, R, Fortran, …
- An array item can be another array
- The array has a type of *object*

```
> let CSE = ['CSCI', 'CENG'];
< undefined
> CSE[2] = 'AIST';
< "AIST"
> CSE
< ["CSCI", "CENG", "AIST"] (3)
> CSE.length;
< 3
```

Comments

- JavaScript supports both
  - Single comments, starting with //
  - Block comments, enclosed by /* */

- You should write comments to your code.

- Some legacy web code put HTML comments and JS code intertwined for compatibility, yet there could be unexpected behaviours
  - https://riptutorial.com/javascript/example/9722/using-html-comments-in-javascript--bad-practice-

```
> typeof NaN
< "number"
> 9999999999999999
< 10000000000000000
> 0.5+0.1==0.6
< true
> 0.1+0.2==0.3
< false
> Math.max()
< -Infinity
> Math.min()
< Infinity
> []+[]
< ""
> []+{}
< "[object Object]"
> {}+[]
< 0
> true+true+true===3
< true
> true-true
< 0
```

```
> true==1
< true
> true===1
< false
> (!+[]+[]+![]).length
< 9
> 9+"1"
< "91"
> 91-"1"
< 90
> []==0
< true
```

Usually used as

Math.max(1,2) => Return 2

If no argument is provided, there is nothing compare to the default minimum: -Infinity

“+” operator can be used for string and number, the data type will automatically change to string

“-“ operator is only used for numeric types

Conditional and looping statements

- Similar to C and Java
  - Conditional statements
    - *if*
    - *if ... else*
    - *? ... : ...*
    - *switch*
  - looping statements
    - *for*
    - *while*
    - *do ... while*
    - *break*
    - *continue*

The for…of loops

- The *for…of* loops iterating in arrays, strings, maps, NodeLists, etc.

```
> let cars = ["honda", "toyota",  "nissan"];
< undefined
> let x;
< undefined
> for (x of cars) {
      console.log(x);
  }
  honda
  toyota
  nissan
< undefined
```

The for…in loops

- The *for…in* loop iterating in object properties with key-value pairs

```
> let person = {fname:"John", lname:"Doe", age:20};
< undefined
> let text = "";
< undefined
> let x;
< undefined
> for (x in person){
      text += person[x];
  }
< 'JohnDoe20'
```

Functions

- JS functions are declared with the keyword function:

```
function myFunction(a, b) {
  return a * b;
}
```

- A function (i.e., anonymous function) can be stored in a variable, then it can be used as a function:

```
let x = function (a, b) {return a * b};
let z = x(4, 3);
```

- See: https://www.w3schools.com/js/js_function_definition.asp

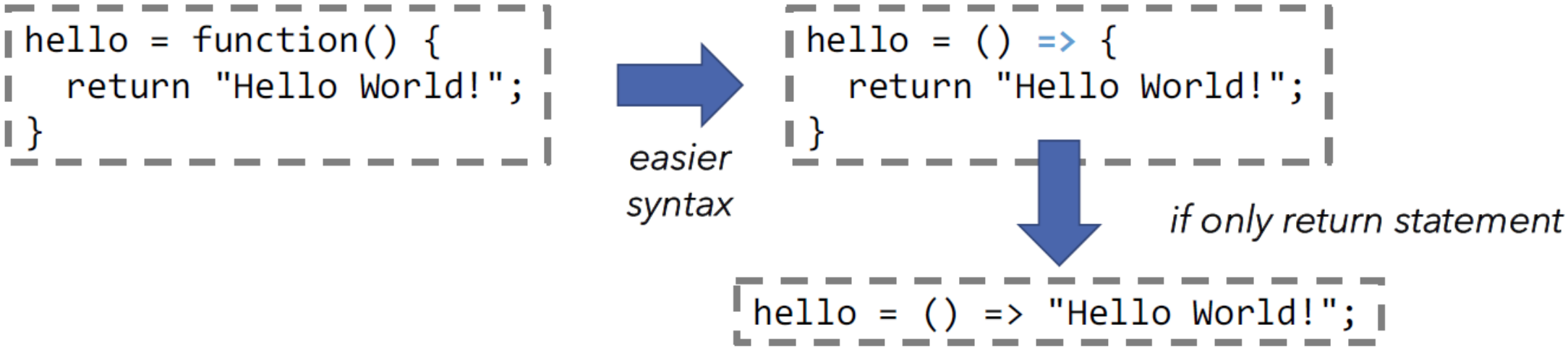
Functions

- Function arguments are passed by value without type check

```
> function ModifyValue(value){
      value = 10;
  }
< undefined
> let number = 5;
< undefined
> ModifyValue(number);
< undefined
> console.log(number);
  5
< undefined
```

Arrow functions

- Arrow function is a shorthand for declaring functions, with some subtle differences
  - Very common in modern code
  - See: https://www.w3schools.com/js/js_arrow_function.asp



The browser window

- In every browser, there is a *window* object representing the browser's window
  - All global JS variables, objects, and functions are member of *window*
  - Variables: *window* properties
  - Functions: *window* methods

- Some window objects:
  - **window.screen**: widthm, heitgh, pixelDepth, etc.
  - **window.location**: hostname, protocol, etc.

Popup boxes

- Messages to user can be delivered with JS popup boxes
  - Alert box: window.alert(*message*)
  - Confirm box: window.confirm(*message*)
    - return true for OK, and false for cancel or anything else
  - Prompt box: window.prompt(*message*)
    - return the string of user input

- These boxes are browser dependent, and not CSS skinnable.

The status of JavaScript

- There are new JS feature in the new ECMAScript version every year
  - See: https://dev.to/brayanarrieta/new-javascript-features-ecmascript-2022-with-examples-4nhg
- The number and diversity of web developers is huge.
- You may see all kinds of old and new techniques in tremendous number of examples and tutorials online

Further readings

- w3schools:
- https://www.w3schools.com/js/

- MDN JS tutorial:
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript