香港中文大學
The Chinese University of Hong Kong

JAVASCRIPT BASICS

CSCI2720 2022-23 Term 1

Building Web Applications

Dr. Chuck-jee Chau
chuckjee@cse.cuhk.edu.hk

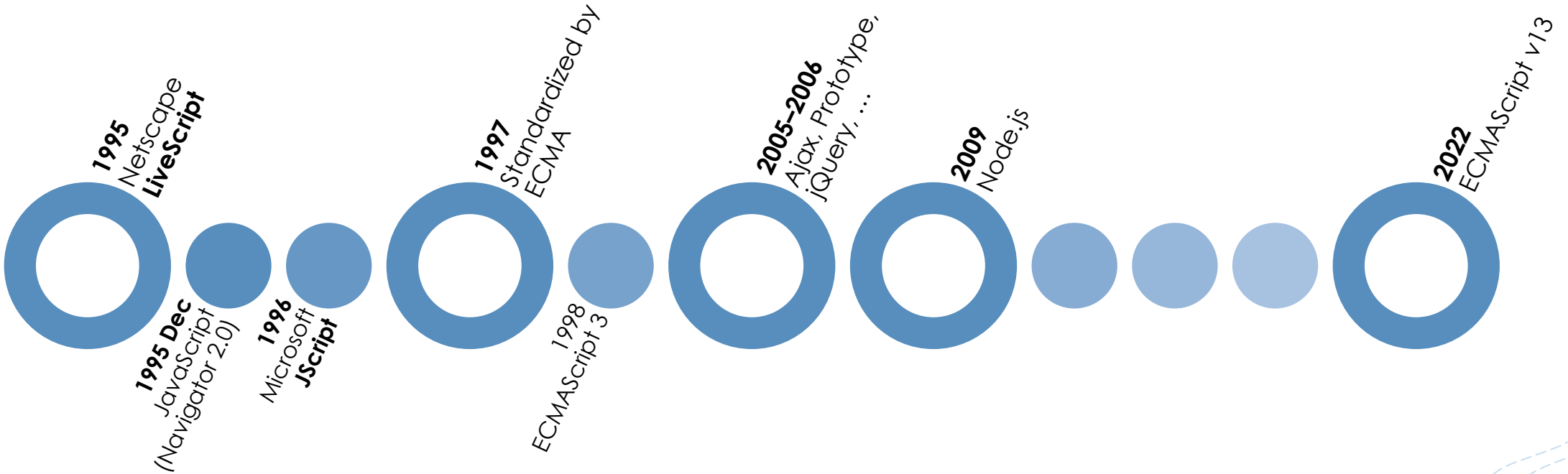
OUTLINE

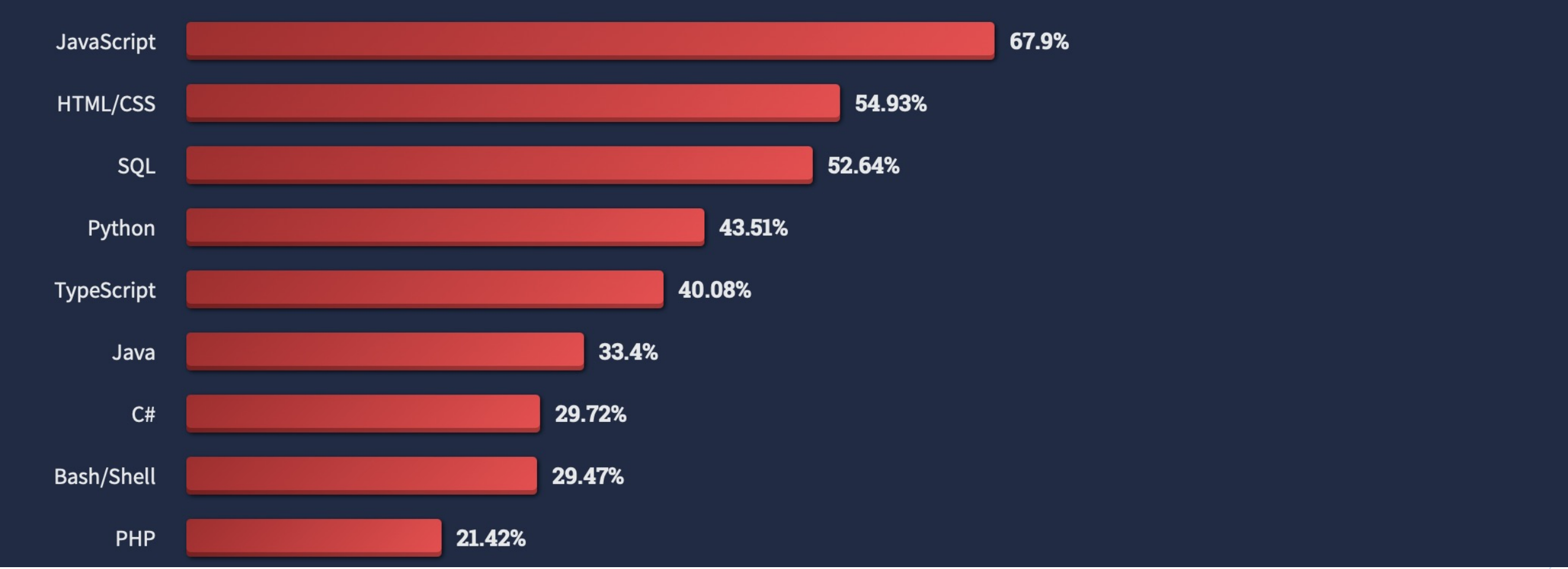
- Why JavaScript?
- Using JavaScript
- Identifiers and variables
- Data types and operators
- Arrays
- Condition and loops
- Functions
- The browser window

WHY JAVASCRIPT?

- The programming language of the web
  - Every element being rendered in the browser can be ***generated and manipulated*** with JavaScript – a *dynamic* page
  - Beyond the browser, now JS can also 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/#most-popular-technologies-language-prof

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 curious developments in modern web design!*

WHAT CANNOT JAVASCRIPT DO?

- JavaScript "lives" in the browser, i.e. it 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
- *Note: We only discuss client-side JavaScript in this lecture*


JAVASCRIPT
ME
Strongly Typed inherently Safe Compiled Programming Languages

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****!*

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 and network loading
  - Embedding code in HTML

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

    - Usually put at the end of HTML body for faster page load
- Code is executed when page is loading

USING JAVASCRIPT

- You can test JavaScript directly in the ***JavaScript Console*** in major browsers
  - Chrome: F12 / CTRL+SHIFT+J (or *View » Developer »* ***Developer Tools***)
- Outputting messages or errors
  - To the console: **`console.log(…)`**
    - This is one of the most important tools for developers, learn it well: *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
  - Somehow *browser technology 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

- Identifier names
  - Case-sensitive
  - Letters, digits, underscore **_**, 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, some surprising characteristics*
    - **`let`**: more preferred, block-scoped and no redeclaration
    - **`const`**: constant with scopes like **`let`**
  - ***Undeclared*** variables are created as *globals*!

See: https://www.freecodecamp.org/news/var-let-and-const-whats-the-difference/

DATA TYPES

- JS *primitive* types – immutable, not an object, no methods
  - *string*: textual data, enclosed by `''` or `""`, first element at index 0
  - *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 nonexistent or invalid object or address
- *See: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures*

DATA TYPES

- The **`typeof`** operator can find out the type of a variable
- JavaScript 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"
> typeof 123n
< "bigint"
```

DATA TYPE CONVERSION

- When adding a number and a string, the number is treated as a string
  - JS evaluates expressions from ***left to right***, so different sequences could result in different results!

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

- Converting a value to a number

```
let numberVar = Number(x);
let numberVar = x - 0;
```

- Converting a value to a string

```
let stringVar = String(x);
let stringVar = x + "";
```

- Converting a value to a boolean

```
let boolVar = Boolean(x);
let boolVar = !!x;
```

STRINGS

- Strings in JS are quoted by `' '` or `" "`
- Usual escape sequences: `\' \" \\ \n` …
- String contents can be compared directly using `< > ==` etc.
- String characters can be accessed like array contents
- Strings can be easily manipulated with RegEx
  - *See: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions*
- Notable string methods (and many more!):
  - `trim()`
  - `split(`*`text`*`)`
  - `slice(`*`start`*`, `*`end`*`)`
  - `indexOf(`*`text`*`)`
- JavaScript encodes text in UTF-16
  - Slightly different from the usual UTF-8
  - But most characters are still well supported, even Emojis
  - *See: https://dmitripavlutin.com/what-every-javascript-developer-should-know-about-unicode/*

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!

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

- *See: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals#string_interpolation*

OPERATORS

- Arithmetic operators
  - `+ - * ** / % ++ --`
- Assignment operators
  - `= += -= *= /= %= **=`
- Comparison operators
  - `== != > < >= <=`
  - `===` equal value ***and type***
  - `!==` unequal value/unequal type

- Logical operators
  - `&& || !`
- Bitwise operators
  - `& | << >>`
  - `~` not
  - `^ ^=` xor
  - `>>>` zero fill shift
    - Discarding least-significant bit
- *See: 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
  - Array index starts at `0`
  - 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 line comments, starting with **//**
  - Block comments, enclosed by **/* */**
- *Try your best to put down comments to explain your code!*

- Some legacy web code put HTML comments and JS code intertwined for compatibility, yet there could be unexpected behaviours
  - *See: 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
```

CONDITIONAL AND AND LOOPING STATEMENTS

- Similar to C and Java
  - Conditional statements
    - **`if`** statement
    - **`if...else`** statement
    - **`? :`** ternary conditional statement
    - **`switch`** statement
  - Looping statements
    - **`for`** loop
    - **`while`** loop
    - **`do...while`** loop
    - **`break`** statement
    - **`continue`** statement

THE FOR…OF AND FOR…IN LOOPS

- The **for…of** loop iterating in Arrays, Strings, Maps, NodeLists, etc.

```
let cars =
["Honda", "Toyota", "Nissan"];
let x;

for (x of cars) {
  document.write(x + "<br >");
}
```

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

```
let person = {fname:"John",
lname:"Doe", age:20};

let text = "";
let x;
for (x in person) {
  text += person[x];
}
```

See: https://www.w3schools.com/js/js_loop_for.asp

FUNCTIONS

- JS functions are declared with the keyword **`function`**

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

- A function can be stored in a variable, then it can be used as a function

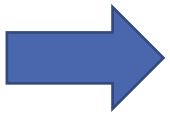
```
let x = function (a, b) {return a * b}; // anonymous function
let z = x(4, 3);
```

- Function arguments are passed by value without type check
- *See: https://www.w3schools.com/js/js_function_definition.asp*

ARROW FUNCTIONS

- Arrow function is a shorthand for declaring functions, with some subtle differences…

```
hello = function() {
  return "Hello World!";
}
```



easier syntax

```
hello = () => {
  return "Hello World!";
}
```

if only return statement

```
hello = () => "Hello World!";
```

- 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 members of **`window`**
  - Variables: **`window`** properties
  - Function: **`window`** methods
- Handy window objects
  - **`window.screen`**: width, height, pixelDepth, etc.
  - **`window.location`**: hostname, protocol, etc.
  - **`window.history`**: back, forward, etc.

POPUP BOXES

- Messages to user can be delivered with JS popup boxes
  - Alert box: `window.alert(`*`message`*`)`
  - Confirm box: `window.confirm(`*`message`*`)`
    - Return is `true` for *OK*, and `false` for *Cancel* or anything else
  - Prompt box: `window.prompt(`*`message`*`)`
    - Return is the string of user input
- These boxes are browser dependent, and *not* CSS skinnable

THE STATUS OF JAVASCRIPT…

- There are new JS features in the new ECMAScript (ES) 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
- ***Wish you good luck!***

READ FURTHER…

w3schools.com JavaScript Tutorial

https://www.w3schools.com/js

MDN JavaScript Tutorial

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