<Semester II>

</Programming for Interactive Interfaces/>

Week 7 Part 2

Arrays

The Array object, enables storing a collection of multiple items under a single variable name

- JavaScript arrays are resizable and can contain a mix of different data types.
- JavaScript arrays must be accessed using nonnegative integers (or their respective string form) as indexes.
- JavaScript arrays are zero-indexed: the first element of an array is at index 0, the second is at index 1, and so on — and the last element is at the value of the array's length property minus 1.

JavaScript Array Syntax

The Arrays are JavaScript's ordered lists that can store any data types including Strings, Numbers, and Booleans. Each item inside of an array is at a numbered position.

- *An array is represented by square brackets and the content inside.*
- *Elements within an array should be separated by commas.*

```
let colors = ["red", "blue", "green", "yellow"];

const cars = ["Saab", "Volvo", "BMW"];
```

- Arrays can be 1D, 2D or even 3D
- Use arrayName.**length** to find out the number of elements in an array

JavaScript Arrays

JavaScript Array Methods

In JavaScript, **arrays have a number of built-in methods that allow you to manipulate and operate on the elements of an array.** These methods can be used to add, remove, and modify array elements, as well as to perform common operations such as sorting and filtering.

Basic methods: length, indexOf(), push(), pop(), unshift(), shift(), includes(), slice(), splice(), concat(), includes(), split(), join(), toString(), fill(), Array.isArray(), flat()

Iteration methods: forEach(), every(), filter(), find() , map(), some(), reduce(), findIndex()

Iteration methods used to traverse and manipulate collections such as arrays, sets, and maps. Each iteration method has a distinct purpose, offering various functionalities to suit different use cases.

JAVASCRIPT ARRAY METHODS

- ARRAY.MAP()
- ARRAY.FILTER()
- ARRAY.REDUCE()
- ARRAY.REDUCERIGHT()
- ARRAY.FILL()
- ARRAY.FIND()
- ARRAY.INDEXOF()
- ARRAY.LASTINDEXOF()
- ARRAY.FINDINDEX()
- ARRAY.INCLUDES()
- ARRAY.POP()
- ARRAY.PUSH()
- ARRAY.SHIFT()
- ARRAY.UNSHIFT()
- ARRAY.SPLICE()
- ARRAY.SLICE()
- ARRAY.JOIN()
- ARRAY.REVERSE()
- ARRAY.SORT()
- ARRAY.SOME()
- ARRAY.EVERY()
- ARRAY.FROM()
- ARRAY.OF()
- ARRAY.ISARRAY()
- ARRAY.AT()
- ARRAY.COPYWITHIN()
- ARRAY.FLAT()
- ARRAY.FLATMAP()

JS


#30
Using JS
Array Methods
JS

Variable Scope

In JavaScript, **the scope of a variable determines where it can be used/applied within the code.** At its core, scope in JavaScript refers to the context or environment in which variables are declared and can be accessed. It dictates the visibility and lifetime of a variable, determining where in your code a particular variable is valid and accessible.

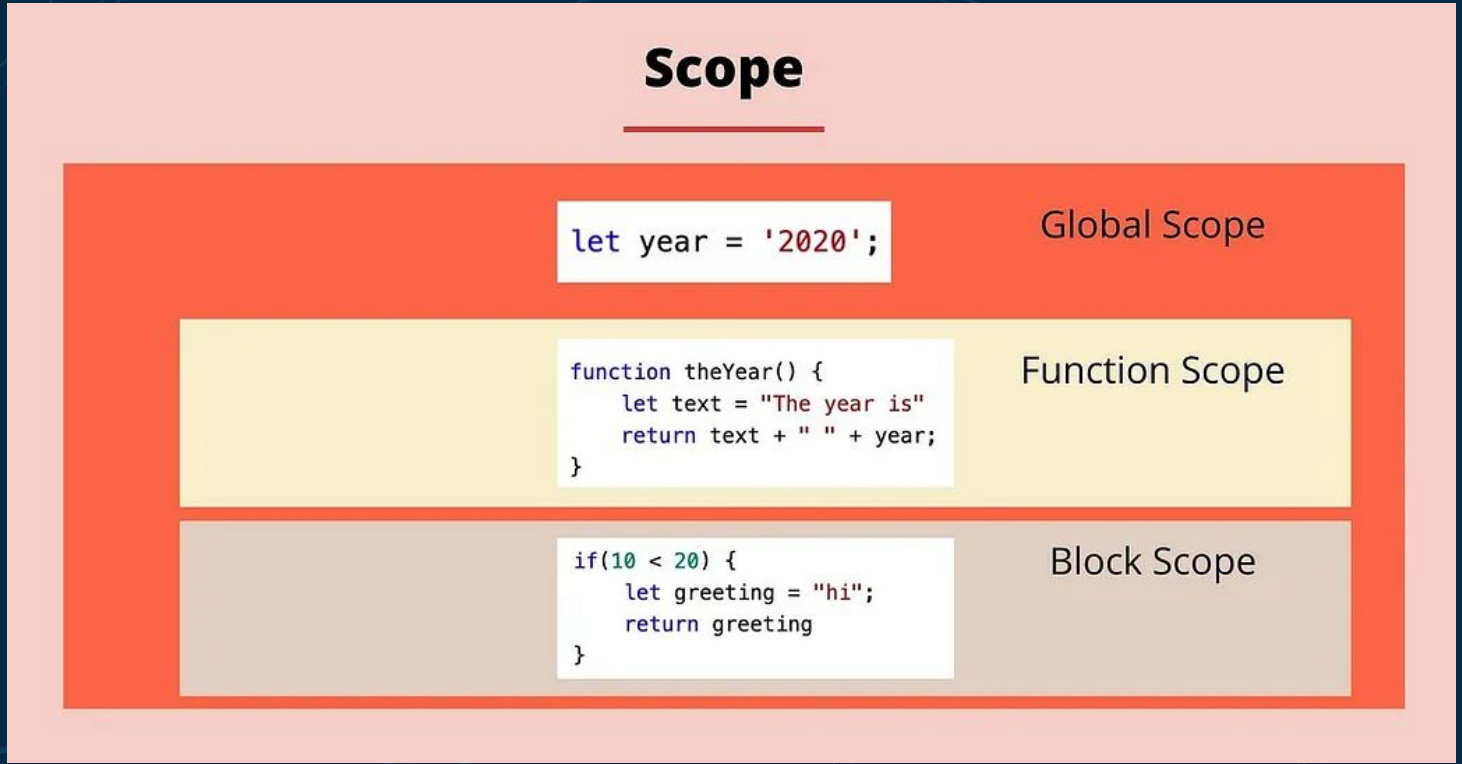
Variables can be declared in different scopes:

- **Global Scope:** In JavaScript, global scope is the widest scope available. Variables declared in global scope are accessible from anywhere in your code, whether it's inside functions, conditional statements, loops, or other blocks of code. Generally they are declared outside of all functions and blocks.

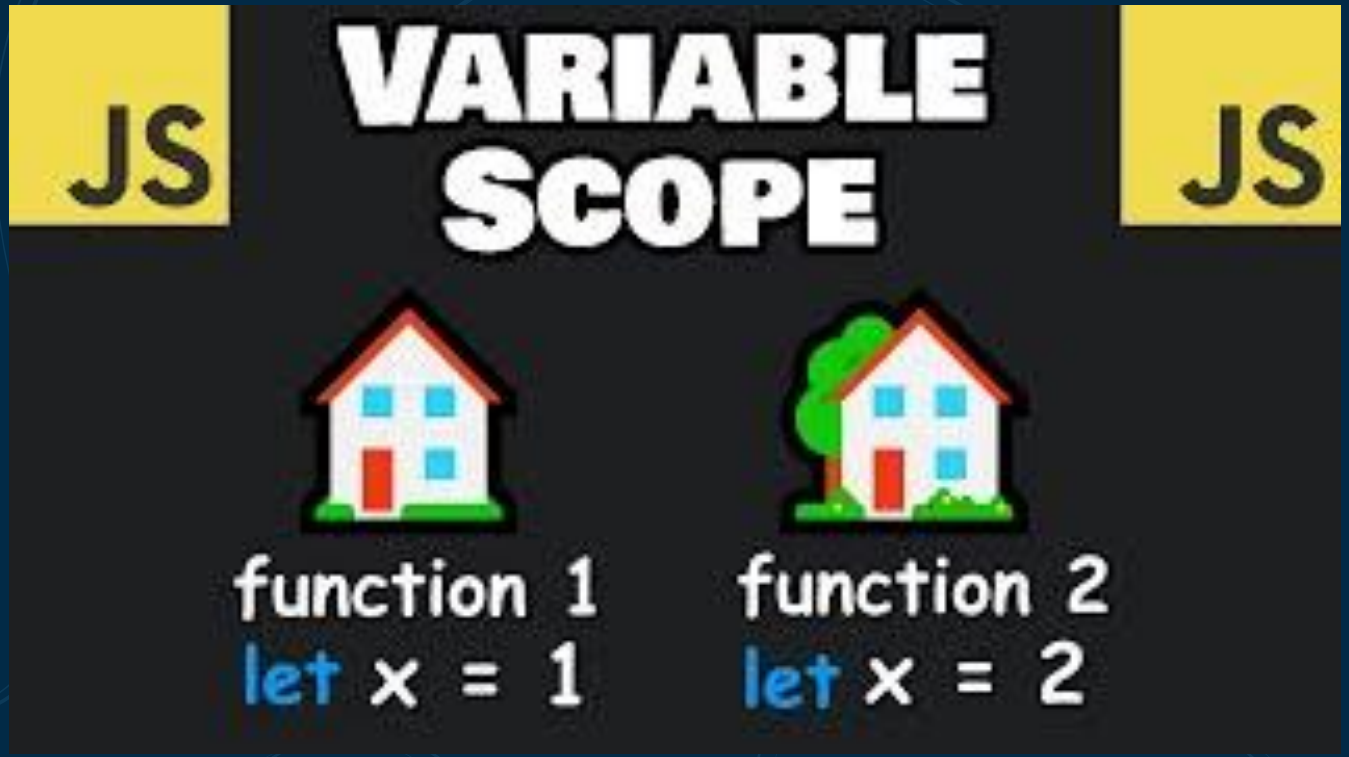
Variable Scope

- **Local (Function) Scope:** JavaScript has function scope: Each function creates a new scope. Variables declared within a JavaScript function, are LOCAL to the function. This means that any variables defined inside a function are not accessible (visible) from outside the function. Variables declared with var, let and const are quite similar when declared inside a function.
- **Block-Level Scope:** Block scope is created within specific code blocks, such as conditional statements (if, else, switch) and loops (for, while). Variables declared in block scope are confined to that block, offering a high degree of isolation.Variables declared with the var keyword can NOT have block scope. Variables declared inside a { } block can be accessed from outside the block.

Variable Scopes



Variable Scopes



Strict Equality Comparison ===

Some Extra Points

- Where the <script> tag is placed inside the HTML file matters. If you need something to happen before the content of the page load, put <script> inside the <head> tag. Generally <script> tag is placed inside the <body> as the last line or after the <body> tag.
- You can check for multiple conditions in an if/else statement.

  Eg.: if (x>10 && y>12){ // if BOTH conditions are met. && = AND

  }

  if (status || c==20){ // if atleast one condition is met. || = OR

  }

  These are called **“Logical Operators”. ! is for NOT**

Homework: Start the HTML + CSS portion of your LCA2 project. If design is still left, complete it and create a working prototype/storyboard.

Learn about and practice while loops.

Use document.getElementsbyClassName() somewhere in your code

W3Schools JavaScript Tutorial

JavaScript for Web | MDN

Reference

1. Array - JavaScript | MDN
2. JavaScript Arrays W3Schools
3. JavaScript Array Methods | WebReference
4. Arrays and Their Properties in JavaScript
5. JavaScript Variable Scope - Programiz
6. Scope in JavaScript. | by devtalib | Medium
7. Scope in JavaScript - Global vs Local vs Block Scope - FreeCodeCamp
8. Logical Operators - JavaScript.info
9. JavaScript while Loop - W3Schools
10. Document: getElementsByClassName() method - Web APIs | MDN

Fonts & colors used

This presentation has been made using the following fonts:

Blinker
(https://fonts.google.com/specimen/Blinker)

Inconsolata
(https://fonts.google.com/specimen/Inconsolata)