MISSION READY

DARE TO
DEVELOP

Variable scope & window object

Ewan Zhang

Variable scope

What is a Variable?

CONST vs LET vs VAR

Variable recap

	Need Initialize?	Re-Declaration	Re-Assignment
let	✗	✗	✓
const	✓	✗	✗

- Initialize: Create the Variable and give it a value.
- Re-Declaration: Create another Variable with the same name.
- Re-Assignment: Give this Variable another value.

Variable scope

- What is Scope?
- Variable scope allows us to know where and when our declared variables remain valid and can be used.
- For Example

```
function printer() {
    const paper = 'The NY Times';
    console.log(paper); // We can access the variable "paper" in here
}

console.log(paper); // We can't access the variable "paper" in here
```

- If we declare a variable with const or let inside a code block, we can only use the variable within that block. We can call this variable “block scoped”.

Blocks

A code “block” is any code within some curly brackets { }.

- Function `function ( ){ }`
- Loop `for ( ) { }`
- If statement `if (true) { }`

Blocks

- A code “block” is any code within some curly brackets {}.
- If we define a variable with *const* or *let* inside of some block {} then we can only access that variable within those curly brackets
- In this example we defined the variable paper within our printer().
- paper is accessible anywhere within the printer() function.
- scanner() is outside of the block of printer() so it does not have access to paper

```
function printer() {
    const paper = 'The NY Times';
    console.log(paper);
}

function scanner() {
    console.log(paper);
}
```

Blocks continued…

- We have said that any variable declared with *const* or *let* is only accessible within it's block.
- In this example
  - do you think paper is accessible in the if statement?
  - do you think the advert variable is accessible outside of the if statement?

```
function printer() {
    let paper = 'The NY Times';

    if (true) {
        console.log(paper);
        let advert = 'cap for sale';
    }
    console.log(advert);
}
```

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

The troublemaker: Var

- Pre ES6 in 2015 the only way to declare a variable was with the var keyword, but var has some weird scoping rules.
- It is not “block scoped” like in most other programming languages meaning that it can be accessed outside of the block it was declared in, this can lead to accidentally overwriting a variable we weren’t supposed to.

Scope

Global Scope

```
let year = '2020';
```

Function Scope

```
function theYear() {
    let text = "The year is"
    return text + " " + year;
}
```

Block Scope

```
if(10 < 20) {
    let greeting = "hi";
    return greeting
}
```

Global scope

- With JavaScript, the global scope is the JavaScript environment.

- In JavaScript any variable declared outside of any known function is said to be “globally scoped”, meaning it will be accessible by any other function.
- In this example variable “writingInstrument” is globally scoped and is accessible by any function.

```
let writingInstrument = 'pen';

function compose() {
      console.log(writingInstrument)
}
```

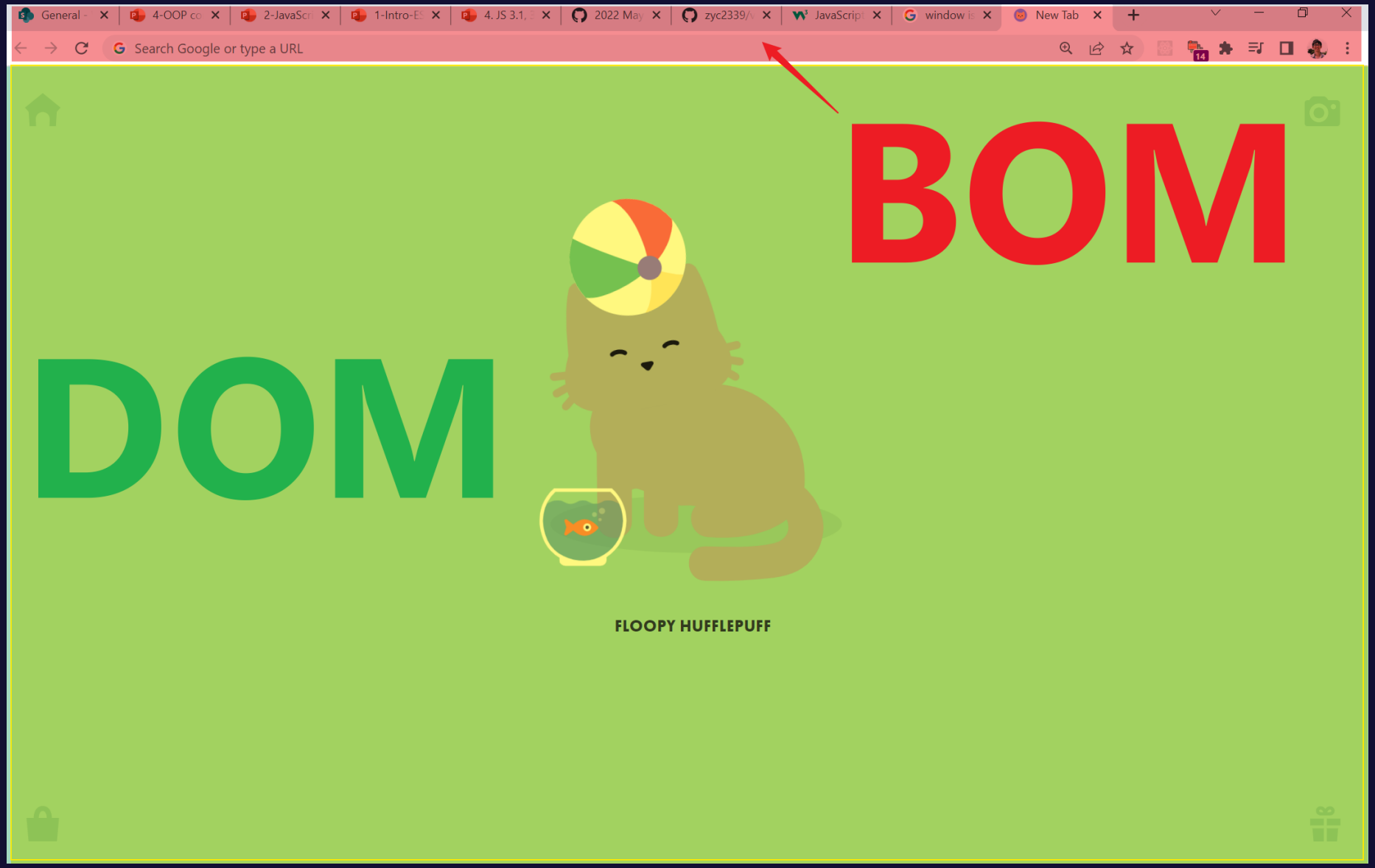
Exercise : What's the result for sayHi() and sayHi2()

```
let myName = "Reuben";

function sayHi() {

  let myName = "Obama";

  console.log(myName + " says good morning.");

  function sayHi2() {

    console.log(myName + " says good morning again");
  }

}
sayHi();

sayHi2();
```

window global object

- What’s in JavaScript?

BOM?



window global object Summary

- The window object is supported by all browsers. It represents the browser's window.
- Window is the main JavaScript object root, aka the global object in a browser.
- The window can be omitted when accessing the window object’s properties. In other words, window.alert() can be written simply as alert().

- All global JavaScript objects, functions, and variables automatically become members of the window object.
- Global variables are properties of the window object. (Use Var)
- Global functions are methods of the window object.
- Even the document object (of the HTML DOM) is a property of the window object.

Best practice for Variable Scope

- Do NOT create global variables unless you intend to.
- Your global variables (or functions) can overwrite window variables (or functions).

- In the block, never name your variables as same as global variables.

```
let x = 'pen’;

function b1 () {
   let x = ‘apple’;
    console.log(x)
}
```

Variable Scope review

- What is scope && Why we need to know about it?
- Scope refers to the visibility and accessibility of variables, functions, and objects in a particular part of your code during runtime. It determines which parts of your code can access or modify specific variables.
- What is Global Scope?
- What is Block Scope?

MISSION READY

www.missionreadyhq.com

DARE TO DEVELOP

Thank you | Ewan Zhang