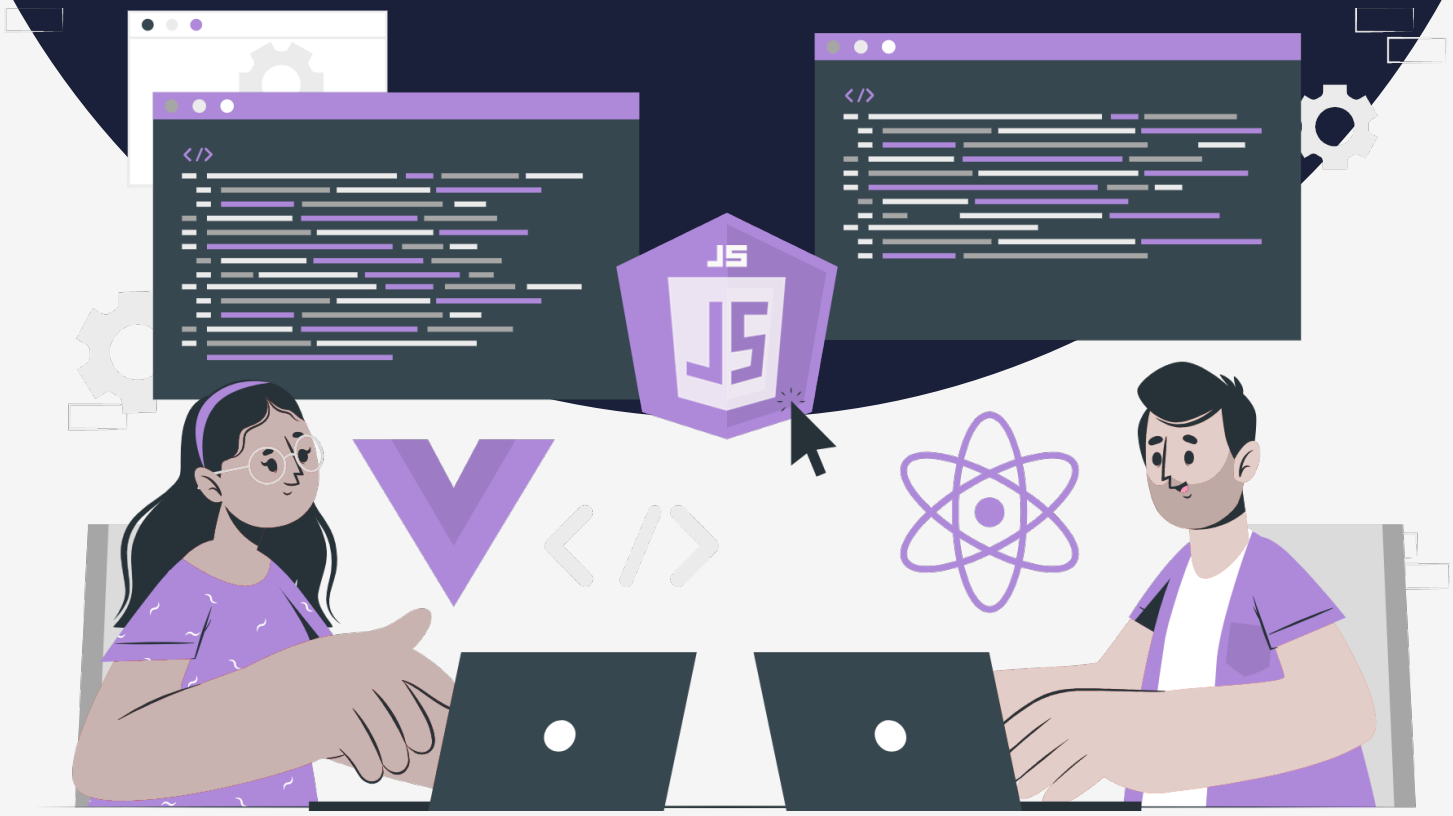
Lesson:

Global Space and Global Scope



PW SKILLS

Topics Covered:

1. Introduction.
2. Global space.
3. Global scope.
4. Difference between global space and global scope.

JavaScript is one of the most widely used programming languages today, powering everything from simple web applications to complex enterprise systems. As JavaScript applications grow in size and complexity, it becomes increasingly important to understand the core concepts that underpin the language. Two such concepts are the global scope and global space, which play a crucial role in determining the behavior and functionality of JavaScript programs.

Global space in JavaScript refers to the area of a program where variables and functions are defined outside of any function or block. These variables and functions are attached to the global object, which is called "window" in the browser environment and "global" in the Node.js environment. Variables and functions defined in the global space are accessible from any part of the program.

Global scope on the other hand refers to the range or visibility of variables and functions within a program. Variables and functions defined in the global scope can be accessed and modified from any part of the program. While global scope can be useful in some cases, it can also lead to naming collisions and make code harder to understand and maintain. Therefore, it's recommended to use local scope wherever possible.

In this lecture we will be discussing these essential concepts in detail, discussing how they work, their advantages and disadvantages, and best practices for using them effectively.

Global Space.

Global space is an area of a program where variables and functions are defined outside of any function or block. In other words, it's the space where variables and functions are defined without being enclosed by a function or a block. Variables and functions defined in the global space are accessible from anywhere in the program.

For example, let's imagine a company with multiple departments and each department has its own workspace or office. There might be certain resources, such as a printer or a scanner, that are needed by employees from all departments. In this case, the printer or scanner can be considered a variable or function in the global space, while the departments' individual workspaces or offices can be compared to functions or blocks within a program.

Just as it's important for employees to coordinate their use of shared resources, it's important for developers to carefully manage the use of global space in their programs to avoid conflicts and ensure efficient use of resources. For example, if multiple functions within a program rely on a particular global variable, it's important to ensure that the variable is named appropriately and that its value is updated consistently across all functions. Similarly, if multiple functions within a program rely on a particular global function, it's important to ensure that the function is defined appropriately and that its behavior is consistent across all functions.

Members of Global Space.

As we know, the global space in JavaScript is the entire environment in which the program is executed. It includes a variety of members that provide a rich set of functionality that can be used in any JavaScript program.

1. **Global Object:** A global object is a special object that represents the global environment in which JavaScript code is executed. It is the top-level object in the JavaScript hierarchy and is available for all globally available properties and methods.
2. **Built-in Objects:** JavaScript provides a number of built-in objects that are available in the global space, such as Math, Date, Array, and String. These objects provide a variety of useful methods and properties that can be used in any JavaScript program.
3. **DOM Objects:** The Document Object Model (DOM). The DOM provides a set of objects and methods that can be used to interact with web pages in a browser. The global space includes a number of DOM objects, such as window, document, and navigator.
4. **Variables and Functions:** Any variable or function defined outside of any function or block is considered to be in the global scope and is therefore a member of the global space. These variables and functions can be accessed from anywhere in the program.

Window Object.

A window object is a global object in web browsers that represents the browser window or tab in which the JavaScript code is running.

Accessing the window object:

```
> console.log(window)
  ▶ Window {window: Window, self: Window, document: document, name: '', location: Location, …}   VM99:1
```

This will log the window object to the console, which will show a number of properties and methods that are available on the window object.

Setting and getting a property on the window object:

```
> window.customProperty = "PW Skills"
< 'PW Skills'
> console.log(window.customProperty)
  PW Skills                                                                                       VM268:1
```

In the above example, we set a new property called customProperty on the window object and then log its value to the console. Because the window object is global, any properties or methods we define on it are also global and can be accessed from anywhere in the script.

It's important to note that the window object is specific to web browsers and is not available in other environments, such as Node.js. In Node.js, the global object is different and does not have the same properties and methods as the window object.

However, the global object serves a similar purpose to the window object in web browsers. It provides access to a range of properties and methods that are available globally within a Node.js application.

Advantages:

1. Reusability: Global variables and functions can be accessed and used from anywhere in the program, making them reusable and reducing code duplication.
2. Sharing data: Global space makes it easy to share data between different parts of the program, simplifying communication between different functions and modules.
3. Simplifies development: Global space can make development easier by reducing the complexity of code and allowing developers to access and modify global variables and functions from anywhere in the program.

Disadvantages:

1. Naming conflicts: Since global variables and functions can be accessed from anywhere in the program, naming conflicts can occur if different parts of the program use the same names for different variables or functions.
2. Security risks: Global space can create security risks if sensitive data or functionality is exposed to unauthorized access, as any part of the program can potentially access and modify global variables and functions.
3. Hard to maintain: Large programs that heavily rely on global space can be difficult to maintain and debug, as changes to one part of the program can have unintended consequences on other parts of the program.

Global Scope.

From the previous lecture, we know that global scope refers to the set of variables and functions that are accessible from anywhere in the program, including inside functions and blocks. Any variable or function defined outside of any function or block is considered to be in the global scope.

Variables and functions defined in the global scope can be accessed and modified from anywhere in the program, making them easy to reuse and share between different parts of the program. However, this also means that care must be taken to avoid naming conflicts and security risks, as any part of the program can potentially access and modify global variables and functions.

Difference between global space and global scope.

Global Space:

1. Refers to the entire environment in which the program is executed.
2. Includes all properties and methods of the global object, such as console, Math, and Date.
3. Includes variables or functions defined in the global scope.
4. Properties and methods of the global object can be accessed from anywhere in the program.
5. Variables or functions defined in the global space can be accessed from anywhere in the program.

Global Scope:

1. Refers to the set of variables and functions that are accessible from anywhere in the program, including inside functions and blocks.
2. Includes variables or functions defined outside of any function or block.
3. Does not include properties and methods of the global object.
4. Variables or functions defined in the global scope can be accessed from anywhere in the program, including inside functions and blocks.
5. Variables or functions defined inside functions or blocks are not part of the global scope.

Global space refers to the entire environment in which the program is executed, including the global object and its properties, while global scope refers to the subset of that environment that can be accessed from anywhere in the program, including inside functions and blocks. Global space and global scope are related concepts, but they are distinct and serve different purposes in JavaScript programming.