



## Welcome to this session: Functional Programming

**The session will start shortly...**

Questions? Drop them in the chat.  
We'll have dedicated moderators  
answering questions.



# Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:



Ian Wyles  
Designated Safeguarding  
Lead



Simone Botes



Rafiq Manan



Charlotte Witcher



Nurhaan Snyman



Ronald Munodawafa



Tevin Pitts

Scan to report a  
safeguarding concern



or email the Designated  
Safeguarding Lead:  
Ian Wyles

[safeguarding@hyperiondev.com](mailto:safeguarding@hyperiondev.com)

# Skills Bootcamp Cloud Web Development

---

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

# Skills Bootcamp Cloud Web Development

---

- For all **non-academic questions**, please submit a query:  
**[www.hyperiondev.com/support](http://www.hyperiondev.com/support)**
- **Report a safeguarding incident:** **[www.hyperiondev.com/safeguardreporting](http://www.hyperiondev.com/safeguardreporting)**
- We would love your feedback on lectures: Feedback on Lectures
- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

# ***Stay Safe Series:***

Mastering Online Safety One week at a Time

---

While the digital world can be a wonderful place to make education and learning accessible to all, it is unfortunately also a space where harmful threats like online radicalization, extremist propaganda, phishing scams, online blackmail and hackers can flourish.

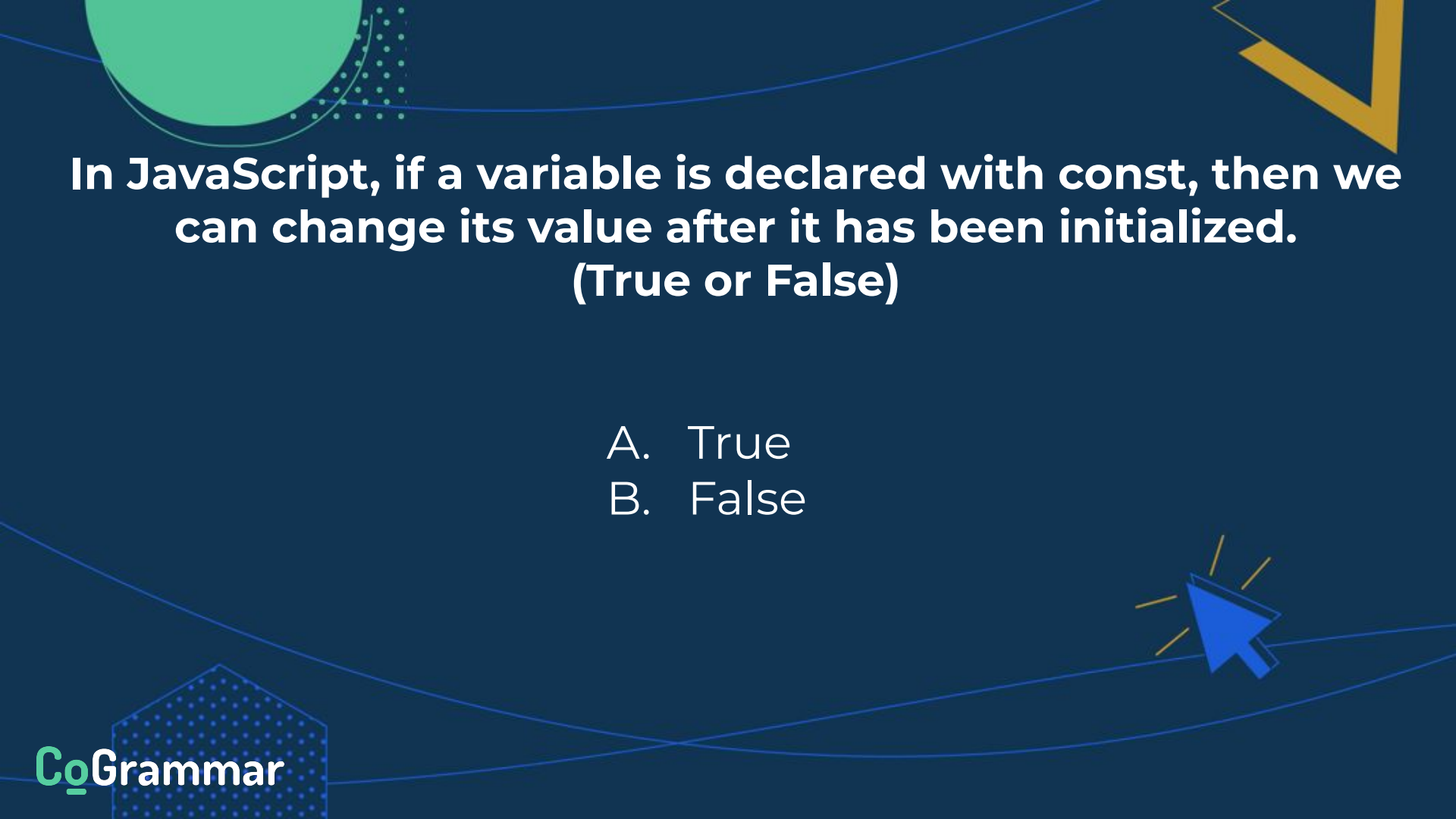
As a component of this BootCamp the ***Stay Safe Series*** will guide you through essential measures in order to protect yourself & your community from online dangers, whether they target your privacy, personal information or even attempt to manipulate your beliefs.

# Pause Before You Post:

## Managing Your Digital Presence

---

- Impact on Reputation.
  - Permanent Record.
  - Privacy Concerns.
  - Miscommunication.
  - Influence on Others.
- Professional Implications.
  - Mental Well-being.



**In JavaScript, if a variable is declared with const, then we can change its value after it has been initialized.  
(True or False)**

- A. True
- B. False



## What is the correct syntax for creating a for loop in JavaScript?

- A. `for (let i = 0, i < 10, i++) { }`
- B. `for (const i : 0; i < 10; i++) { }`
- C. `for (int i = 0; i < 10; i++) { }`
- D. `for (let i = 0; i < 10; i++) { }`



## Learning Outcomes

---

- Define and use functions in JavaScript, including understanding local and global scope.

# Tutorial Overview

---

- Basic JavaScript functions
- Scope
- Live coding session (Tutorial)

# Functions

**A block of organised, reusable code that accomplishes a specific task.**

- ❖ A function can be **called repeatedly** throughout your code.
- ❖ Functions can either be **user-defined** or **built-in**.
- ❖ This helps us **minimise repeating lines of code** unnecessarily.
- ❖ The main benefits of using functions are:
  - It improves code **modularity, management** and **maintenance**.
  - It makes our code more **readable**.
  - It **reduces potential errors**.



# BASIC JAVASCRIPT FUNCTIONS

- ❖ Declaring a function in JavaScript involves using the keyword **function**, providing a **function name**, followed by a list of **parameters** enclosed in **parentheses ()**, and the **function body** enclosed within curly braces **{}**.
- ❖ Basic syntax of a function:

```
function functionName(parameter1, parameter2, ...parameterN) {  
    // function body  
    // statements defining what the function does  
}
```

# BASIC JAVASCRIPT FUNCTIONS

- ❖ A JavaScript function has three key components:
  - **Parameters** - These are variables listed as a part of the function definition. They act as placeholders for the values on which the function operates, known as arguments.
  - **Function body** - Enclosed between curly braces {}, the function body consists of statements that define what the function does.

# BASIC JAVASCRIPT FUNCTIONS

- **Return statement** - How a function sends the result of its operations back to the caller. Not all functions have to return a value; those that don't are often used for their side effects, such as modifying the global state or producing an output.

# BASIC JAVASCRIPT FUNCTIONS

- ❖ Example of a function that doesn't return anything:

```
function sayHi() {  
  console.log("Hi");  
}
```

- ❖ Example of a function that returns something:

```
function sayHi() {  
  return "Hi";  
}
```

# CALLING A FUNCTION

- ❖ After a function has been declared, it can be invoked or called anywhere in your code by using its name followed by parentheses ().
- ❖ If the function requires parameters, you'll include **arguments** within the parentheses.
- ❖ Each argument corresponds to the position of the parameter in the function declaration.



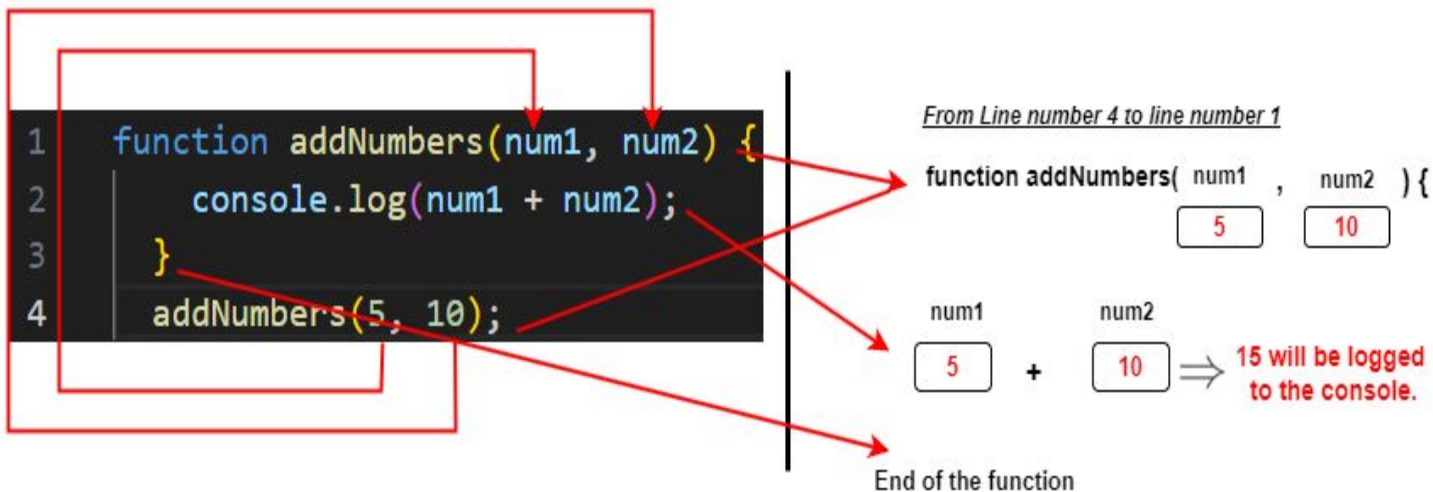
# CALLING A FUNCTION

- ❖ Example of calling a function:

```
function addNumbers(num1, num2) {  
    console.log(num1 + num2); // Log the sum of num1 and num2 to the console.  
}  
  
addNumbers(5, 10); // Calling the addNumbers function with five and ten as arguments
```

# CALLING A FUNCTION

- ❖ Let's trace through this function:



# CALLING A FUNCTION

- ❖ The primary difference between parameters and arguments:
  - **Parameters** - Parameters are used when defining a function. They represent the **'input'** the function needs to do its job, and they act as placeholders for actual data.
  - **Arguments** - Arguments are used when calling a function. They represent the actual **'input'** that will be operated on by the function's code.

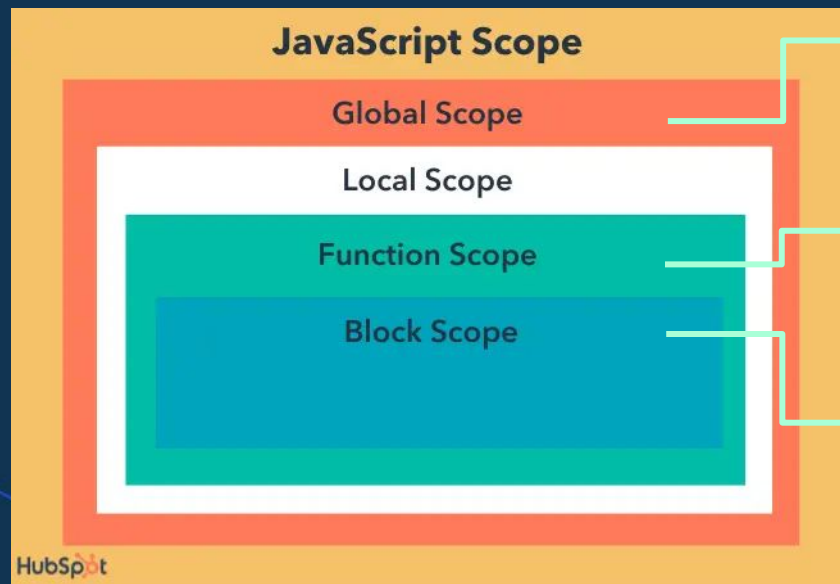
# Scope

The area of visibility and accessibility of a variable in a program.

- ❖ The **scope** of a variable determines **where in the code it can be seen**.
- ❖ JavaScript has **function scope**, meaning variables declared **inside a function** are only **accessible within** that function.
- ❖ Variables declared outside of a function, known as **global variables**, can be accessed anywhere (**hoisting** allows for variables to be accessed before their definition).
- ❖ JavaScript has **three types of scope**:

- Global Scope
- Function Scope
- Block Scope

# Scope



**Global Scope:** variables declared outside all functions or blocks. They can be accessed from any part of the code.

**Function Scope:** variables declared within a function. They are only accessed within their function body.

**Block Scope:** variables declared with the **let** or **const** keyword inside a block. They can only be accessed in their block (does not apply to **var** keyword).

Source: [HubSpot](#)

# SCOPE

## ❖ Global scope:

- When a variable is declared outside all functions or block scopes, its scope is global.
- Global variables can be accessed from any part of the code, whether within a function or outside.

# SCOPE

## ❖ Function scope:

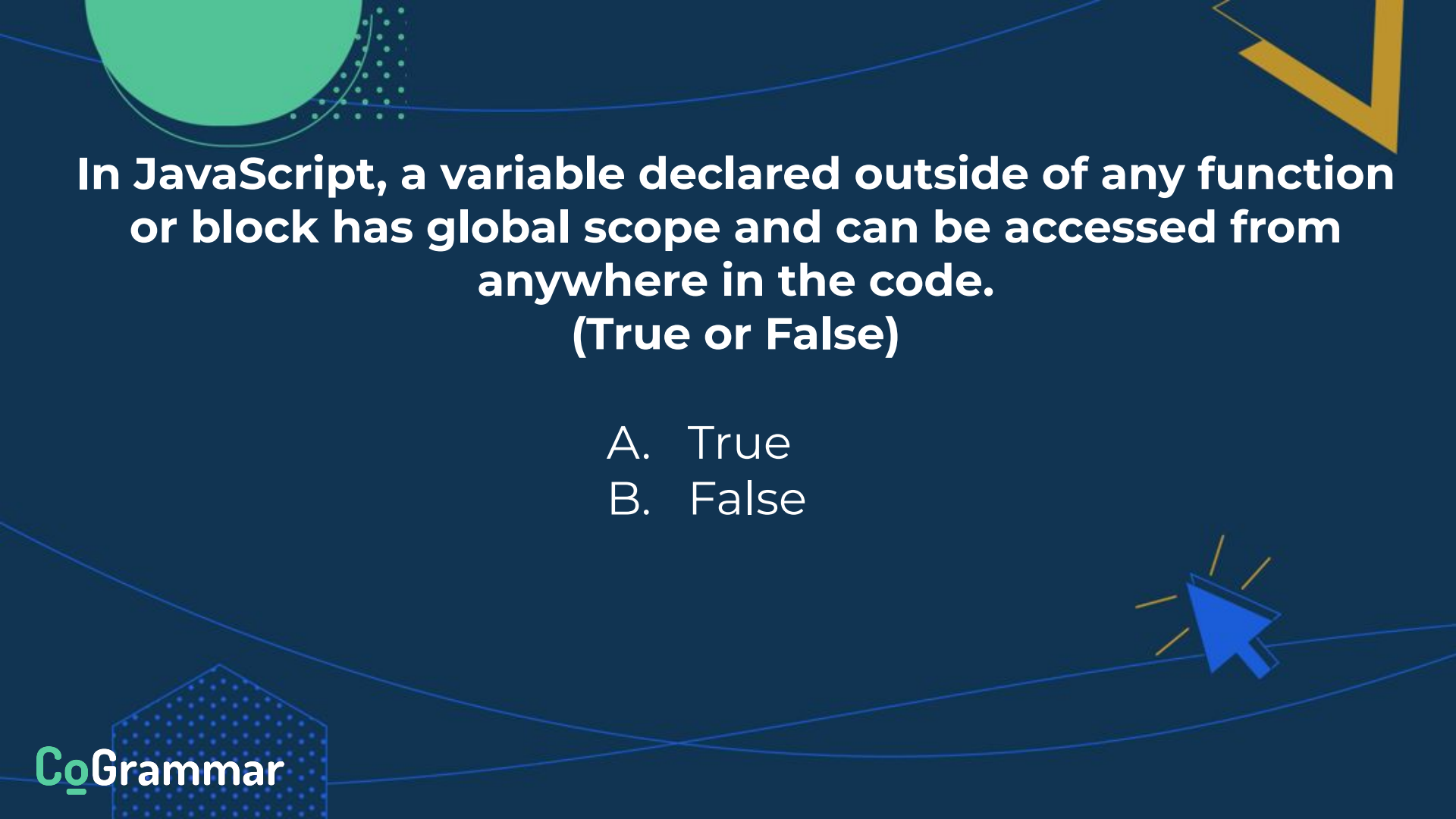
- Variables declared within a function are accessible only within the function body and are said to have the function scope.
- They cannot be accessed outside of the function in which they are declared.
- Attempting to access a function-scoped variable from outside the function will result in a reference error.



# SCOPE



- ❖ Block scope:
  - Variables declared with `let` or `const` are confined to the block in which they are declared.
  - Attempting to access block-scoped variables outside their block results in a reference error, as they are only accessible within the block where they were defined.





**In JavaScript, a variable declared outside of any function or block has global scope and can be accessed from anywhere in the code.  
(True or False)**

- A. True
- B. False



## **What is the purpose of the return statement in a JavaScript function? (Select all that apply)**

- A. To end function execution.
- B. To create a new variable.
- C. To send back a value to where the function was called.
- D. To loop through a function.



Let's take a  
break



# Questions and Answers



# Thank you for attending



**CoGrammar**



  
Department  
for Education