Slide Note

Press Esc to exit full screen

Modular Kitchen

- Modular kitchen design refers to the approach that involves the organization of distinct parts like kitchen cabinets, drawers, etc., that can be integrated and configured into a single, holistic system.
- It consists of small sections or modules which are accumulated together to form a complete kitchen.
- It helps in easy maintenance and repair. Each kitchen unit can be customized according to the customer's need.
- It is flexible and continually adaptable since it is comprised of independent units.

Slide Note

A similar approach can be implemented in the programming world.

Slide Note

School Progress Report

Consider a scenario where teachers of grades 1 - 9 should prepare their students’ annual progress and achievement reports.

Can we write a JavaScript program that helps teachers accomplish this task?

Menu

Sample Code to Display Grades in School

```
let marksofSubject = 0;
let averageMarks = 0;
let totalMarks = 0;

 //Get the marks for 5 subjects from the user
for(let i = 0; i < 5;i++) {
   //write code to get
marks   totalMarks = totalMarks + marksofSubje
ct;
 }

//calculate average marks
averageMarks = totalMarks/5;
console.log(`The total marks
is ${totalMarks}`);
console.log(`The average marks
are: ${averageMarks}`);
```

```
//categorize the student based on marks
obtained
if(averageMarks > 80) {
      console.log('Grade is A');
   }
else
if(averageMarks <= 79 && averageMarks >= 60)
   {
     console.log('Grade is B');
   }
else
if(averageMarks <= 59&& averageMarks >= 35)
   {
      console.log('Grade is C');
   }
else{
    console.log('Grade is F');
   }
```

Slide Note

Think and Tell

- The above code is used to calculate the grade for a single student.
- A particular grade in school may have 40 students approximately.
- Should we repeat the same task of getting marks from the students, calculating average and allocating grade points for 40 students again?
- Is it possible to break each task into smaller units and reuse the code for each student?

Menu

Slide Note

JavaScript Functions are used to perform tasks independently by breaking the program into smaller chunks.

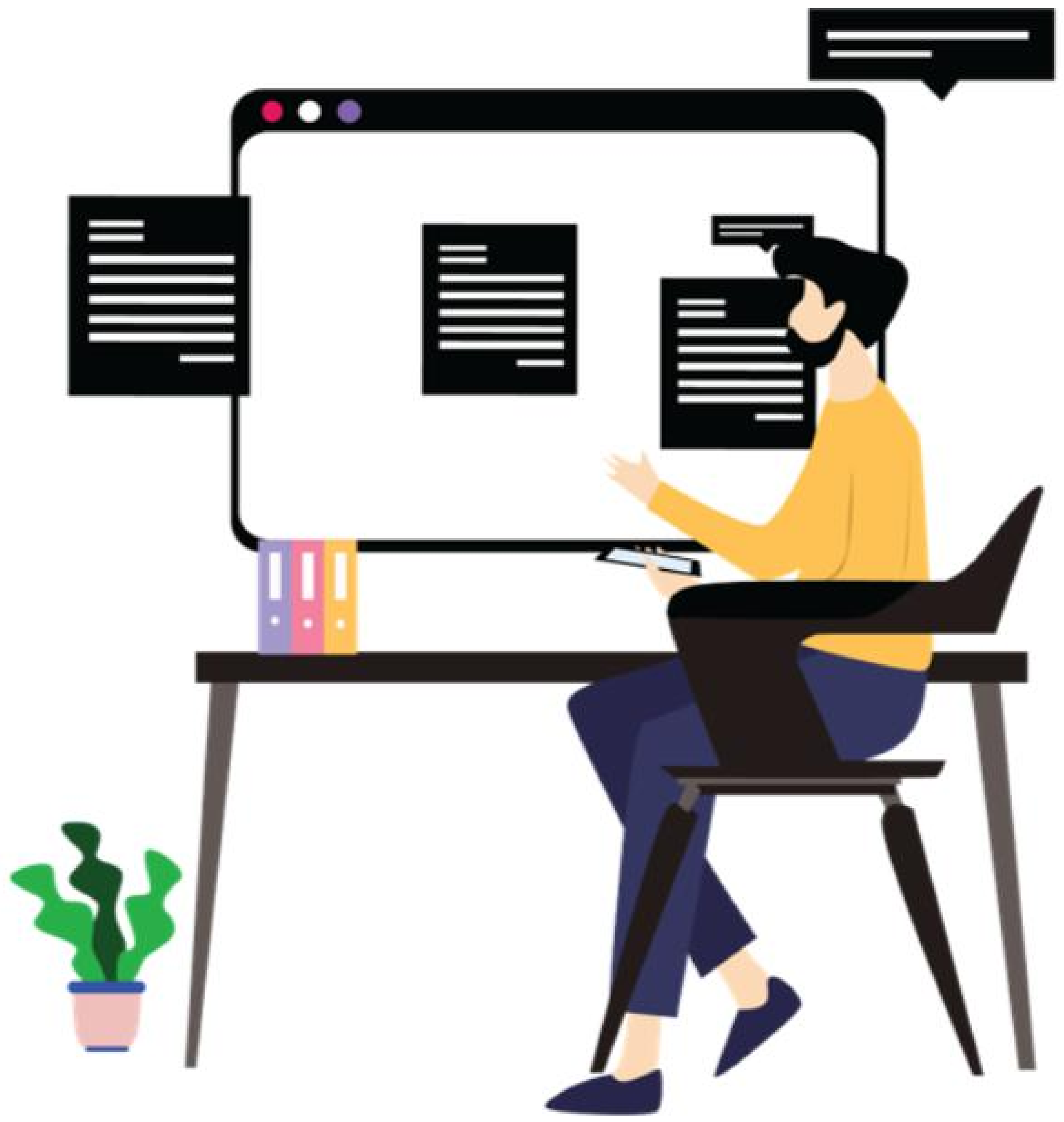
Menu

00:06:00:00:40

06/ 40

Slide Note

Implement Modular Programming Using Functions



Menu

00:07:00:00:40 07/ 40

Slide Note

Learning Objectives

- Define function as an independent unit to perform tasks
- Explain scope of a variable
- Use default parameters in functions
- Explain predefined/built-in functions

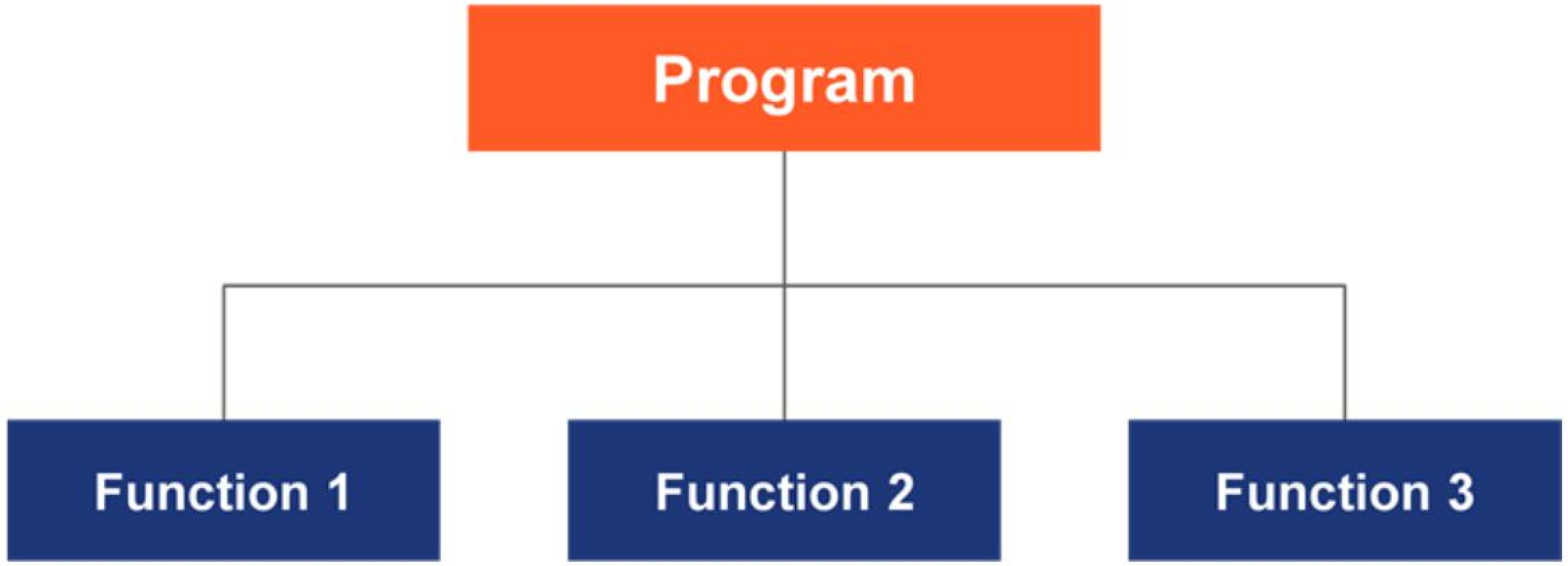
Menu

00:08:00:00:40 08/ 40

Slide Note

What is Modular Programming?

- Modular programming is a software technique that divides the program into small and independent modules to perform some specific functionality, instead of writing a program as one large block of code.

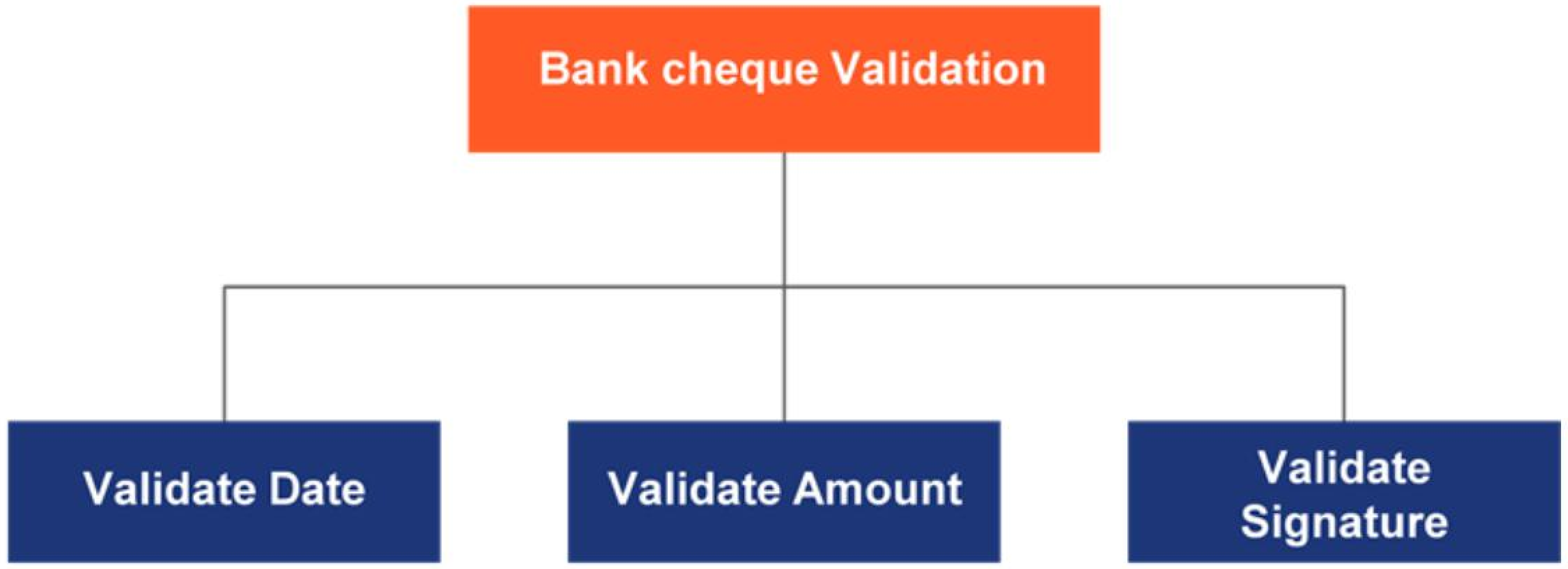


Menu

Slide Note

Modular Programming: Real Example

- To validate a bank cheque leaf, you can divide the main task into simpler tasks as below:
  - Validate Date
  - Validate Amount
  - Valid signature

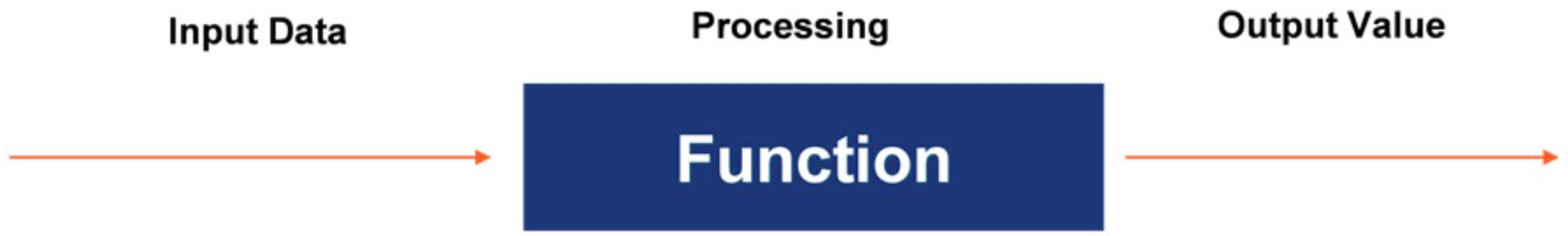


Menu

00:10 00:00:40 10/ 40

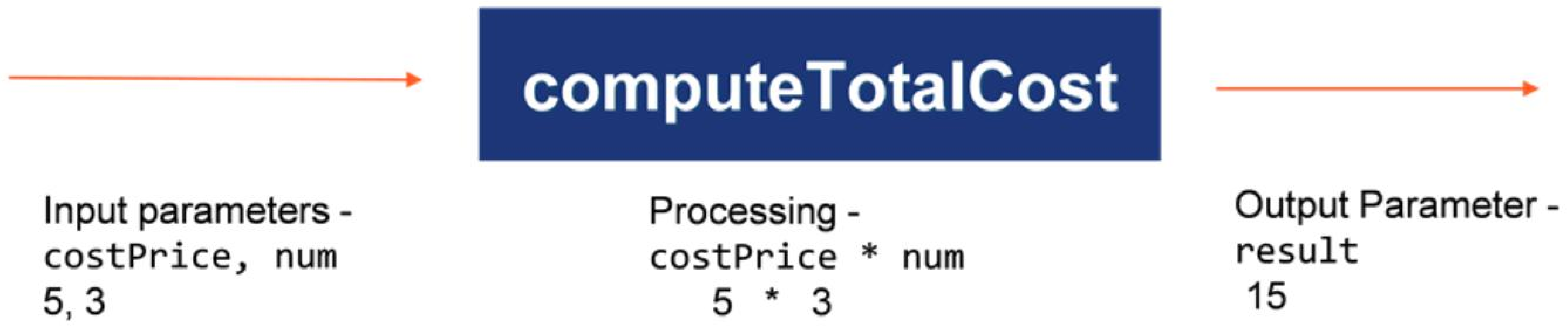
What is a Function?

- A function is a block of organized, reusable code that is used to perform a specific action.
- Actions can be simple tasks like:
  - finding the sum of two numbers
  - finding the maximum score
  - displaying a set of values to the user
  - validating data



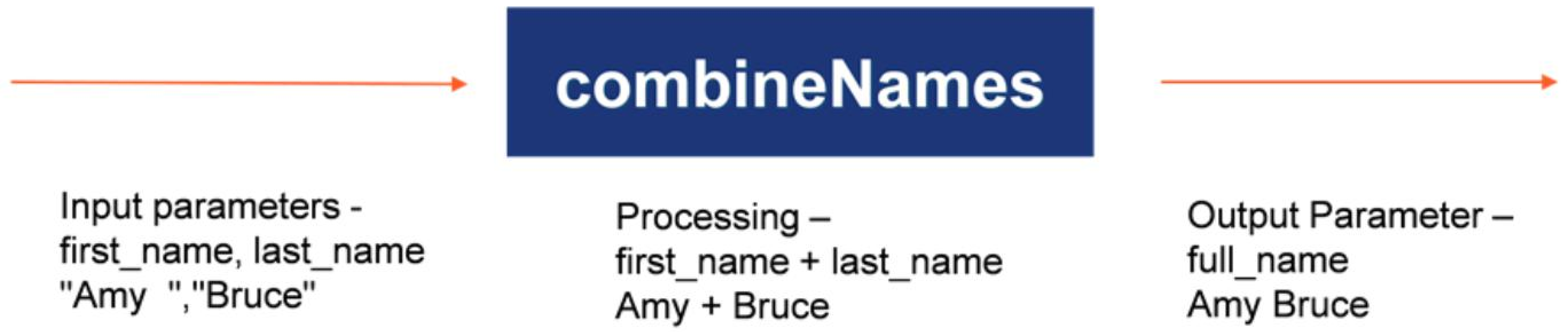
How Does Function Process?

- Adam wants to know the final cost of milk after he bought 3 gallons of milk, each costing $5.
- Multiply the cost (`costPrice`) of each gallon with the number (`num`) of gallons and display the total cost.
- The input parameters of the function are `costPrice` and `num`.
- The function processes the input parameters by multiplying them.
- The result is the output parameter.



How Does Function Process? (cont’d)

- Amy wants her full name which has both first name and last name to be displayed on the college scoreboard.
- Combine two strings first_name and last_name and display the final full_name as a result.
- The input parameters of the function are first_name and last_name.
- Function process the input parameters by concatenating them.
- The full_name is the output parameter which concatenates the two string values.


combineNames
Input parameters - first_name, last_name "Amy ","Bruce"
Processing – first_name + last_name Amy + Bruce
Output Parameter – full_name Amy Bruce

Slide Note

The function cube takes one parameter, called number.

The function body consists of two statements.

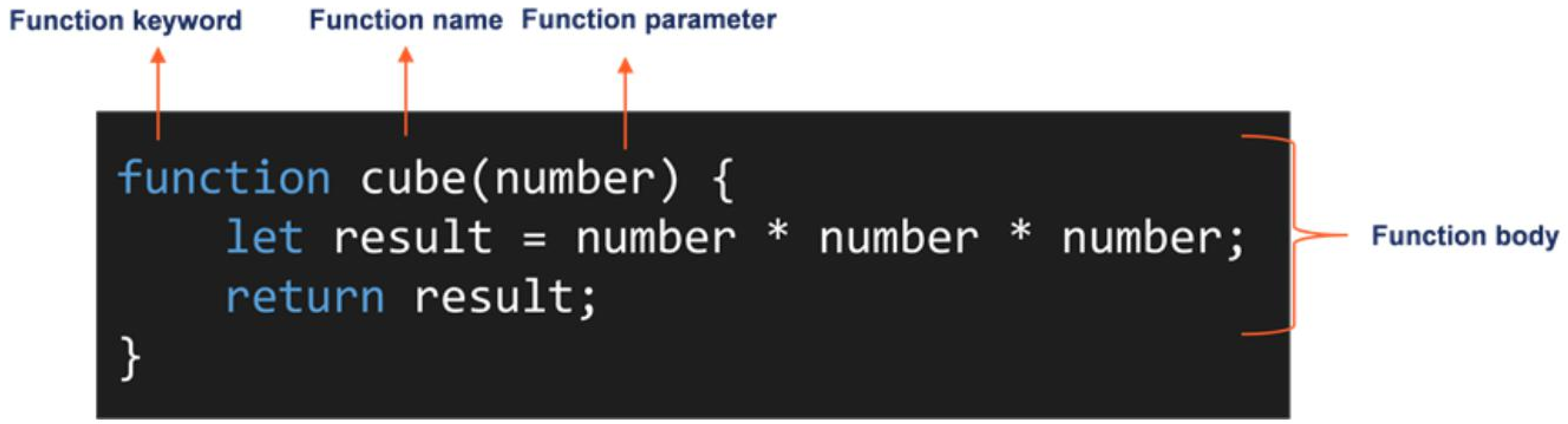
First statement computes the cube of the given number.

The second statement displays the computed cube value on to the console.

Function Definition is also called as function declaration or function statement.

Function Definition

- Function definition consists of the **function** keyword followed by:
    - the name of the function
    - parameter list (optional)
    - function body which has the JavaScript statements enclosed in curly brackets, {...}
- Function body usually returns some value using the return keyword.
- Alternatively, it can create side effects like displaying values in the console.



```
function cube(number) {
    let result = number * number * number;
    return result;
}
```

Menu

```
//Function without parameter
function greet() {
    console.log("Happy Learning");
}

//Function with one parameter
function square(number) {
    let result = number * number ;
    console.log(`Result
is ${result}`);
}

//Function with two parameters
function add(num1, num2) {
    let result = num1 + num2 ;
    console.log(`Addition Result
is ${result}`);
}
```

Function Parameters

- Function Parameters are the names listed in the function definition.
- Function parameters are just values you supply to the function so that the function can do something utilising those values.
- JavaScript functions can have zero or more number of parameters. Maximum number varies for different browser engines.
- The function definition does not specify data types for parameters.
- The browser's JavaScript engine can determine the datatypes of the variables based on the defined values.
- Function definitions can be created with multiple parameters as shown in the code.

Slide Note

Function With Return Statements

- A return is a value that a function returns to the calling script or function when it completes its task.
- By default, functions return "undefined".
- To return any other value, the function must have a return statement that specifies the value to return.
- The **return** statement ends function execution and specifies a value to be returned to the function caller.
- Syntax is:

```
return expression;//expression whose value is to be returned
```

```
function cube(number) {
return number * number * number;
}
```

Menu

Slide Note

JavaScript Simple Functions

Write a program with a display method displaying 'Hello JavaScript Learners'. Then call the display method to execute it.

Write a program that has an add method that takes two parameters. The function adds the two parameters passed and returns the sum. Call this method to display the result in the console.

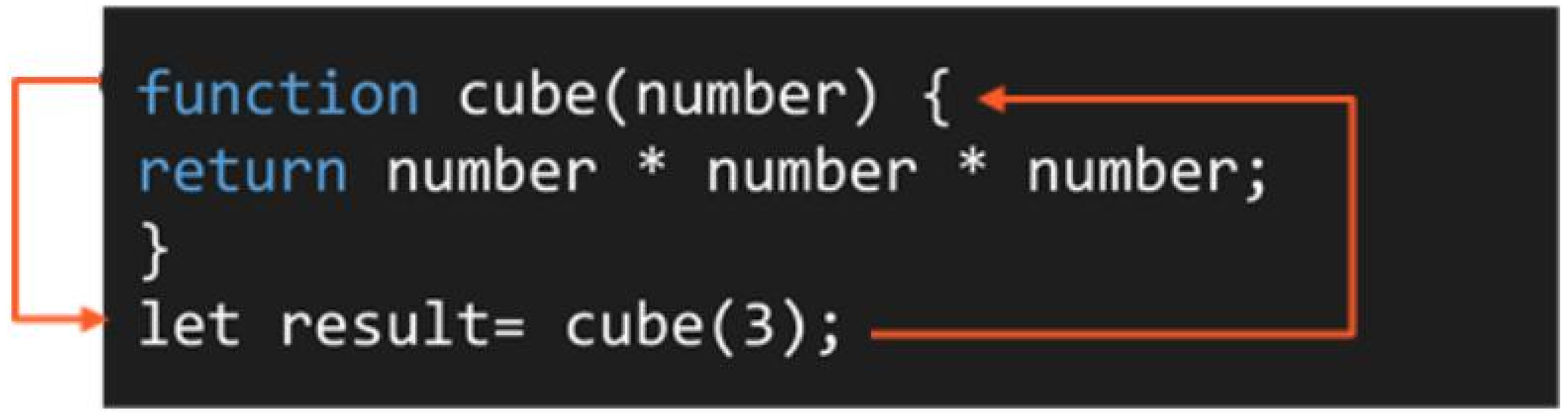
Follow this link for the demo code.

DEMO

Menu

Calling Functions

- Only defining a function does not execute it.
- Defining assigns a name to a function and specifies what the function does when it is called.
- The code inside the function is executed when invoked.

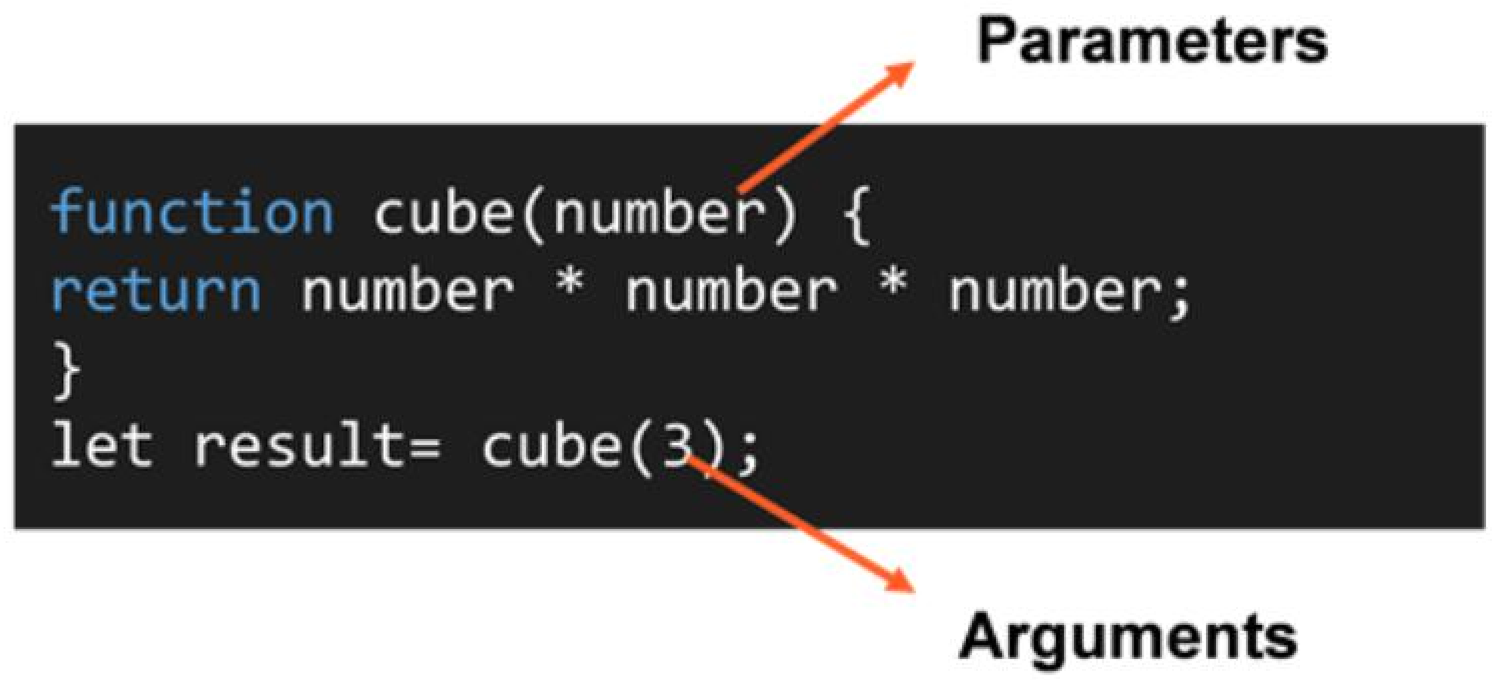


```
function cube(number) {
return number * number * number;
}
let result= cube(3);
```

- The above-highlighted statement calls the function with an argument of 3.
- The function executes its statements and returns the value 27 stored in a variable called result.

Difference Between Parameters and Arguments

- Parameters: These are the names listed in the function's definition..
- Arguments: These are the actual values passed to the function while calling.
- Parameters are initialized to the values of the arguments supplied.
- In the sample code shown below, the value 3 passed as an argument is assigned to the parameter 'number'.



```
function cube(number) {
return number * number * number;
}
let result= cube(3);
```

Slide Note

Scope of a Variable

- Scope determines the accessibility or visibility of variables.
- The lifespan of a variable in JavaScript depends upon the scope of that variable. It starts when it declares the variable.
- A local variable lives until the function is under execution. The moment a function completes, the local variable gets deleted.
- A Global variable lives until the user closes the program or until the web browser is closed.
- JavaScript provides 3 types of scope:
  - Block scope
  - Function scope
  - Global scope
- Block scope was introduced from ES6 (ECMA Script 6) JavaScript version.

Menu

Slide Note

JavaScript Scope

Check the understanding of different scopes available in JavaScript using the following demo code.

DEMO

Menu

00:21:00:00:40 21/ 40

Slide Note

```
function compute() {
   let num1 = 100;
   //Variable x is in block scope
   for(x = 1; x <= num1; x++){
     if( x % 3 == 0)
       continue;
     else
        console.log(`${x}`);
    }
    // x cannot be used here
}

compute();
```

Block Scope

- Keywords "let" and "const" provide block scope in JavaScript.
- Variables declared inside { } block cannot be accessed outside it.
- Having variables in block scope is less error-prone since you can declare variables inside block without worrying about overwriting some previously defined variables.
- Variables declared with the "var" keyword cannot have block scope, i.e. they can be accessed outside the block also.
- "var" keyword was predominantly used in earlier versions. It has been replaced by "let" and "const" keywords in JavaScript.
- The variable "x" inside the for loop is in block scope as shown in the code.
- Block scope was introduced from ES6 (ECMA Script 6) JavaScript version.

Slide Note

Function Scope

- Variables declared within a JavaScript function become local to the function.
- Local variables have function scope and they can be accessed only within the function.
- Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.
- Variables defined inside a function are not accessible from outside the function.

```
function compute() {
  //Variable num1 is in function scope
   let num1 = 100;
   for(x = 0; x <= num1; x++){
     if( x % 3 == 0)
       continue;
     else
       console.log(`${x}`);
      }
   console.log(`loop executed ${num1}
times`);
 }
// num1 cannot be used here

compute();
```

Menu

00:23:00:00:40 23/ 40

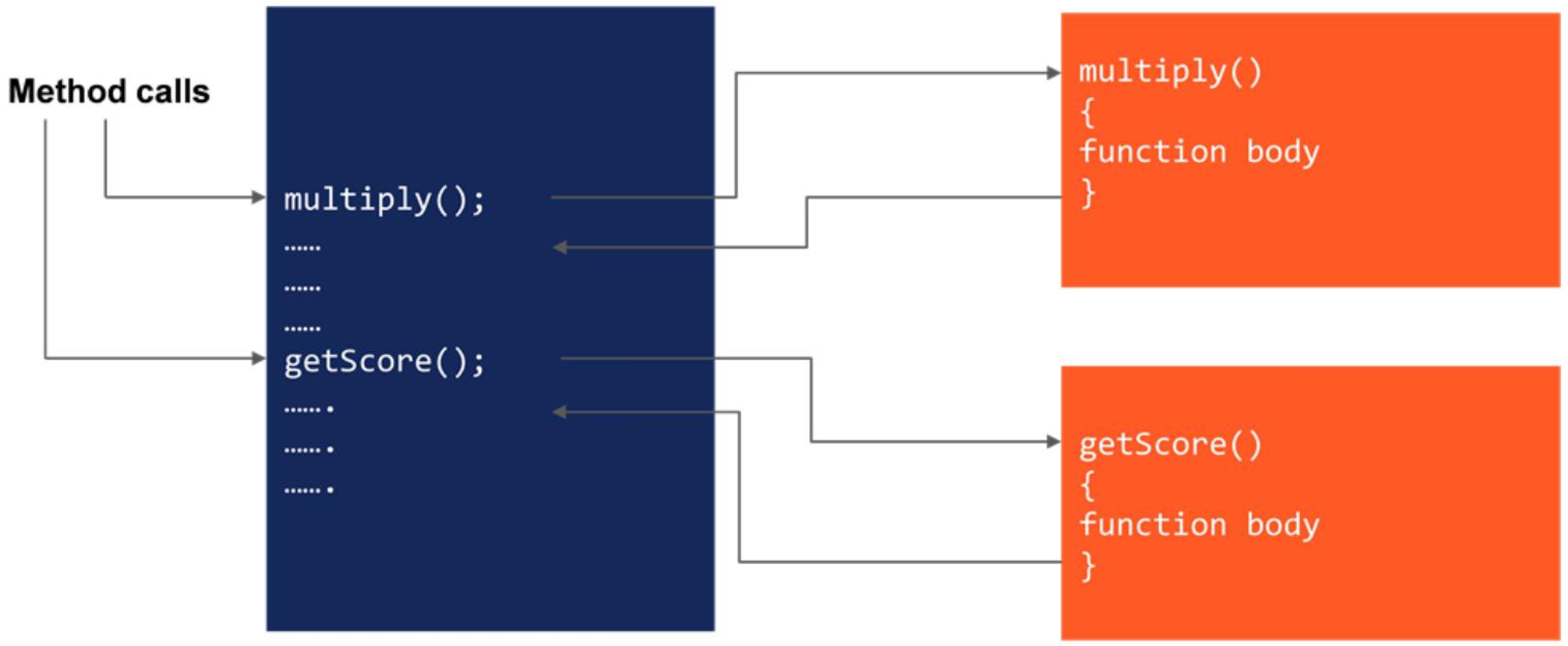
```
// variables are in the global scope
var num1 = 30, num2 = 5, name = 'joe';

//  function is in the global scope
function multiply() {
   return num1 * num2;
}

multiply(); // Returns 150

// Another function in global scope
function getScore() {
   var num1 = 30, num2 = 50;
   return name +'scored ' + (num1+num2);
 }

getScore(); // Returns "Joe scored 80"
```

Global Scope

- Variables declared globally outside any function have **global** Scope.
- Global variables can be accessed from anywhere in a JavaScript program.

Slide Note

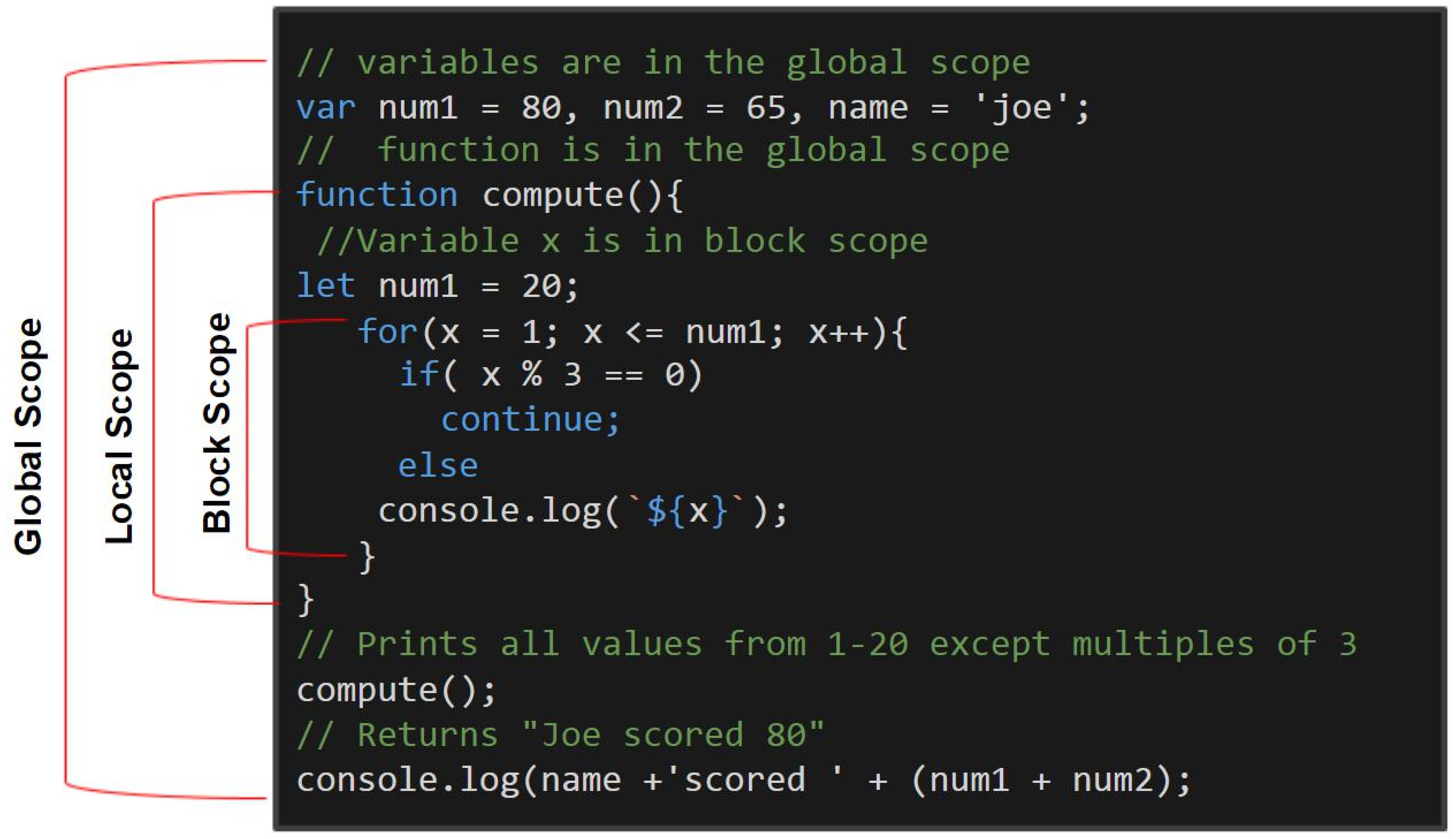
Function Call Sequence



Menu

00:25 00:00:40 25/ 40

Hierarchy of a JavaScript Program



```
// variables are in the global scope
var num1 = 80, num2 = 65, name = 'joe';
//  function is in the global scope
function compute(){
 //Variable x is in block scope
let num1 = 20;
   for(x = 1; x <= num1; x++){
     if( x % 3 == 0)
       continue;
     else
    console.log(`${x}`);
   }
}
// Prints all values from 1-20 except multiples of 3
compute();
// Returns "Joe scored 80"
console.log(name +'scored ' + (num1 + num2);
```

```javascript
function add(x, y = 5) {
  return x + y;
}
add(5); //10
add(5,6); //11
add(5,undefined); //10


function multiply(x = 2, y = 5) {
  return x * y;
}

multiply(); //10
add(5); //25
add(5,3); //15
```

Default Parameters

- Default function parameters allow named parameters to be initialized with default values if no value or undefined is passed.
- When you want to set a different default value other than *undefined,* default parameters are the best option to set values.
- Default parameters are introduced from ES6(ECMA Script) JavaScript version.
- In the past, extra coding is required to test parameter values, and assign a value if it is *undefined.*

Slide Note

Default Parameters

Calculate the maximum of two numbers by passing a default parameter to the function.

Check this link for the demo code.

DEMO

Menu

00:28:00:00:40 28/ 40

Slide Note

Quick Check

Which statement is required if you want to return a value from a function?

A. break

B. continue

C. for

D. return

Menu

00:29 00:00:40 29/ 40

Slide Note

Quick Check: Solution

Which statement is required if you want to return a value from a function?

A. break

B. continue

C. for

D. **return**

Menu

00:30:00:00:40 30/ 40

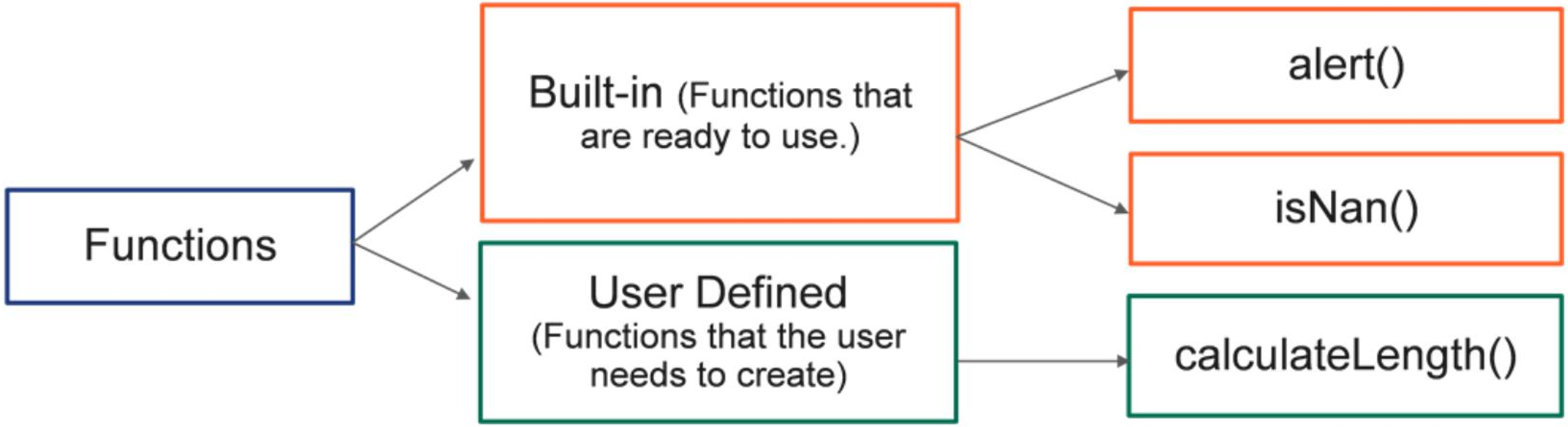
Slide Note

Alert(): An alert box usually gives some information or warning message to the user. Alert() is a predefined function of window object.

isNan(): In JavaScript NaN is short for "Not-a-Number". The isNaN() method returns true if a value is NaN. The isNaN() method converts the value to a number before testing it.

Pre-defined and User-Defined Functions

- Pre-defined Functions: JavaScript provides "top-level" or "global" functions and are called globally.
- User-defined Functions: Functions created by user to perform certain actions.


Functions
Built-in (Functions that are ready to use.)
User Defined (Functions that the user needs to create)
alert()
isNan()
calculateLength()

Slide Note

JavaScript Predefined Functions

Demonstrate some predefined functions like alert, confirm and prompt used in JavaScript.

Write a function to accept a name from the user and display a greeting message to the user if the name is valid using prompt, confirm and alert methods.

Check this link for the demo code.

DEMO

Menu

Slide Note

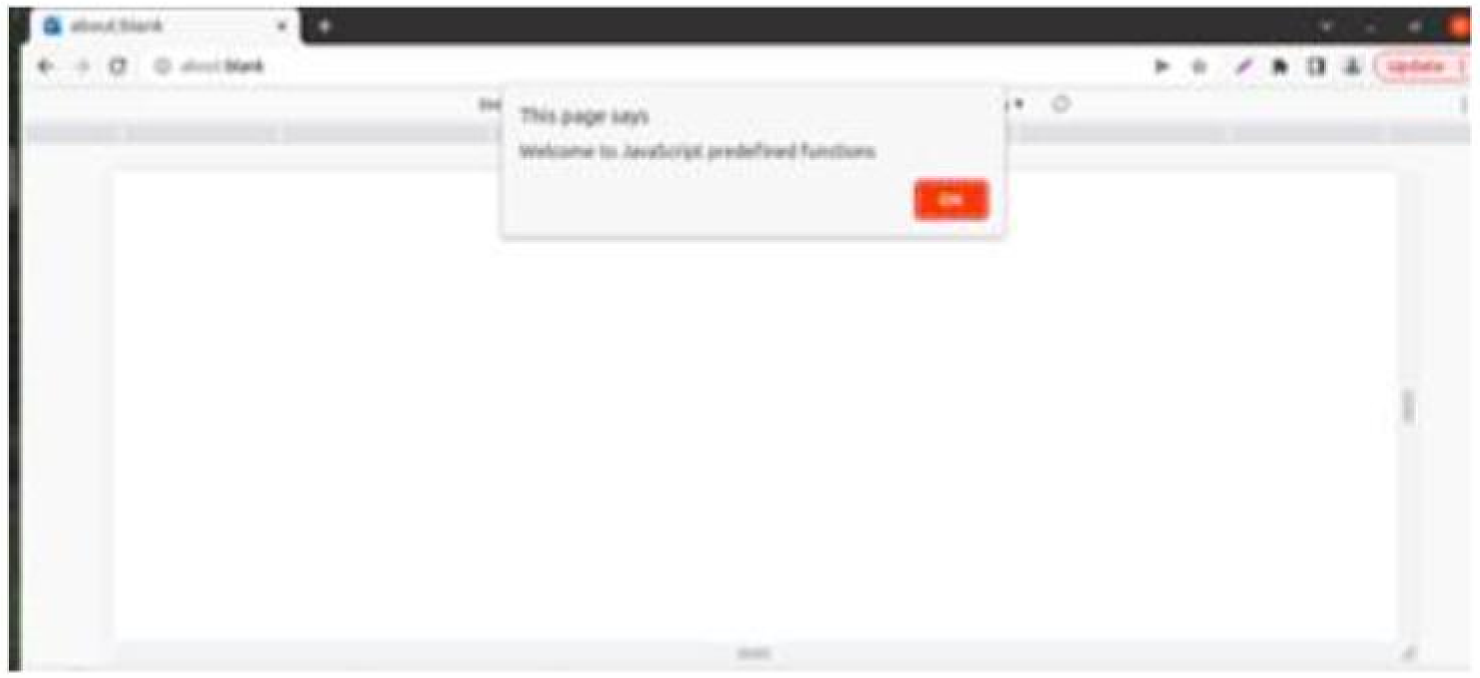
Alert() is a predefined function of window object.

JavaScript predefined window object will be covered later.

JavaScript alert() Function

- Window.alert() method instructs the browser to display a dialog with an optional message and to wait until the user dismisses the dialog.
- An alert box usually gives some information or warning message to the user.
- When the alert dialog box with some specified message (optional) pops up, it takes focus, and the user must click "OK" to proceed.

```
function display() {
  alert('Welcome to predefined
functions');
}

display();
```


This page says

Welcome to JavaScript predefined functions

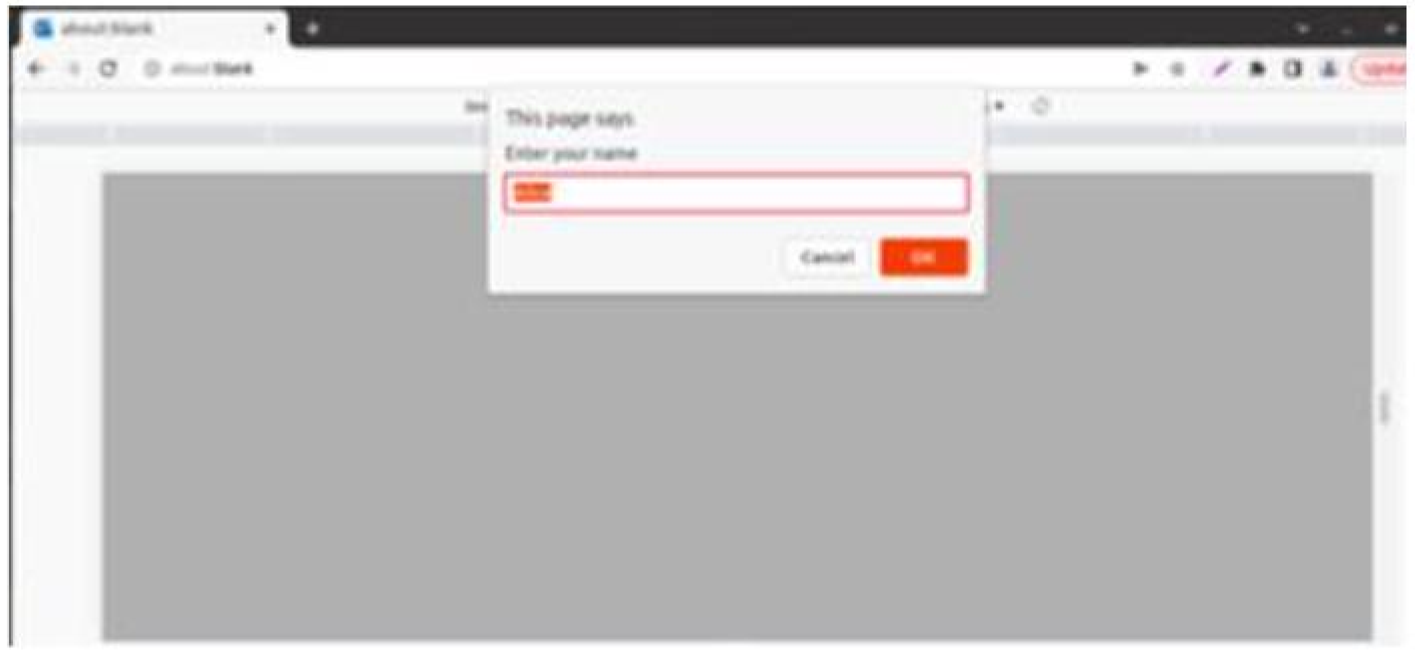
Preview: Alert Dialog

Note: JavaScript predefined window object will be covered later.

JavaScript prompt() Function

- window.prompt() instructs the browser to display a dialog with an optional message prompting the user to input some text, and wait until the user either submits the text or cancels the dialog.
- A prompt box is used if you want the user to input some value.
- prompt() function takes 2 arguments. The first argument (optional) is a string of text to display to the user. The second argument (optional) is the string containing the default value displayed in the text input field.

```
function getValue() {
  prompt('Enter your
name' , 'Alice');
}

getValue();
```



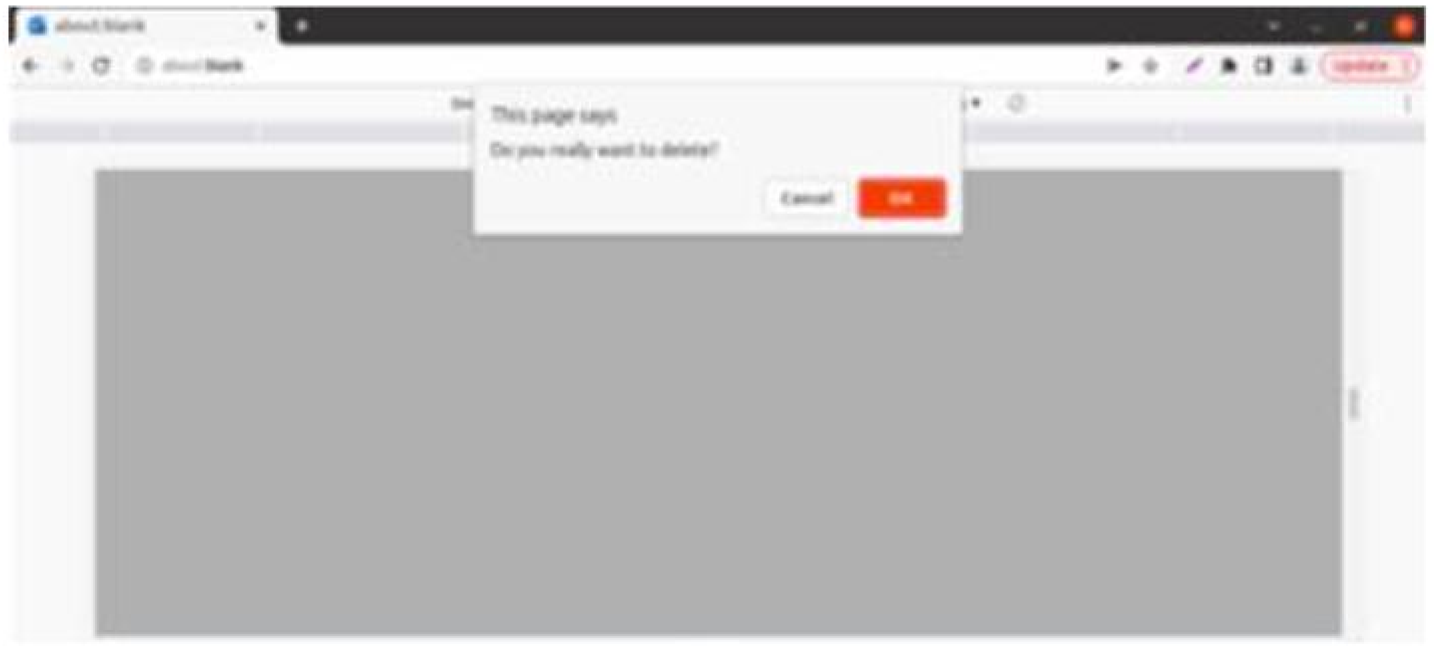
Preview: Prompt Dialog

Slide Note

JavaScript confirm() Function

- window.confirm() instructs the browser to display a dialog with an optional message and wait until the user either confirms or cancels the dialog.
- A confirm box displays a dialog box with a message, an OK button, and Cancel button.
- confirm() function takes an argument which is a string you want to display in the confirmation dialog.
- confirm() function returns the value true if the user clicked "OK", otherwise returns false.

```
function confirmDelete() {
  confirm('Do you really
want to delete');
}

confirmDelete();
```


This page says

Do you really want to delete?

Preview: Confirm Dialog

Slide Note

Grades

In a school, the teachers of grades 1- 9 should prepare the annual performance report of their students. Write a program that helps teachers classify the grades based on the average scores.

Create separate functions for calculating total marks, calculating average scores, and classifying grades based on the average scores.

Use this link for the demo code.

DEMO

Menu

Slide Note

Advantages of Modular Programming Using Functions

- Improves readability as it reduces the length of the programs.
- Makes management of large applications easier.
- Allows reusability of the code.
- Simplifies understanding and debugging of the code.

Menu

Slide Note

Quick Check

A __________ is a group of reusable code that can be called anywhere in your program.

A. exception

B. function

C. loop

D. switch

Menu

Slide Note

Quick Check: Solution

A __________ is a group of reusable code that can be called anywhere in your program.

A. exception

B. **function**

C. loop

D. switch

Menu

00:39/00:00:40

39/ 40