JavaScript for Absolute Beginner

By Du Wun Aung

Acknowledgements

I would like to extend my heartfelt appreciation to several individuals who have been instrumental in the creation of this book.

I also want to express my sincere gratitude to my friends, Dr. PSH, Dr. Henry, Dr. Aung Ye Kyaw, Ko Htet Arkar Soe, and Ma May Phyoe Thu, for providing me with opportunities and inspiration. Your friendship and camaraderie have enriched my life, and I am grateful for the creative energy and enthusiasm you bring to every interaction.

Additionally, I want to acknowledge MYEO and all team members for their support and encouragement throughout this journey. Your presence has been a source of motivation, and I am grateful for your unwavering belief in my endeavors.

Last but not least, I want to thank all the readers who have chosen to embark on this journey with me. Your interest and enthusiasm for this book are deeply appreciated, and I hope that the insights and knowledge shared within its pages will inspire and empower you on your own journeys.

With heartfelt gratitude,
Du Wun Aung

© 2024 Du Wun Aung. All rights reserved

Disclaimer

I want to be transparent with you, the reader, that this book underwent grammar and spelling checking with the assistance of ChatGPT. While ChatGPT was used solely for these purposes and not for content generation, it's important to acknowledge its role in the editing process. Additionally, English is not my primary language, so while ChatGPT's assistance was valuable, there may still be nuances and intricacies in the language that could be improved.

Important Note: The Concept and Course Design of this book is the exclusive property of Du Wun Aung. All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without prior written permission from the copyright owner, except for brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.

© 2024 Du Wun Aung. All rights reserved

© 2024 Du Wun Aung. All rights reserved

© 2024 Du Wun Aung. All rights reserved

Introduction

Course Description:

This course is tailored for absolute beginners who are eager to learn JavaScript for web frontend development. JavaScript is a crucial language for creating interactive and dynamic web content. Throughout this course, you will embark on a journey from understanding basic JavaScript concepts to applying them in a practical project. By the end of the course, you'll have the knowledge and skills to start building your own simple web applications.

Course Objectives:

- Provide a comprehensive introduction to JavaScript fundamentals.
- Introduce practical projects to reinforce learning and skills application.
- Cover essential concepts such as variables, loops, functions, and DOM manipulation.
- Foster a hands-on learning environment through coding exercises and project development.
- Prepare students to continue learning and exploring more advanced topics in web development.

Course Outline:

Module 1: Introduction to JavaScript

- What is JavaScript and why is it important?
- Setting up your development environment (text editor, browser)
- Writing your first JavaScript code: alerts, console.log, comments

Module 2: Variables and Data Types

- Understanding variables and their role in programming
- Declaring variables with var, let, and const
- Primitive data types: strings, numbers, booleans
- Working with variables: assigning values, variable naming conventions

Module 3: Operators and Expressions

- Arithmetic operators: +, -, *, /, %
- Comparison operators: ==, ===, !=, !==, >, <, >=, <=

© 2024 Du Wun Aung. All rights reserved

- Logical operators: &&, ||, !
- Using operators in JavaScript expressions

Module 4: Control Flow and Loops

- Conditional statements: if, else if, else
- Switch statement for multi-way branching
- Introduction to loops: for loop, while loop
- Looping through arrays and objects

Module 5: Functions

- Introduction to functions: what they are and why they're useful
- Declaring functions and invoking them
- Function parameters and return values
- Scope in JavaScript: global scope vs. local scope

Module 6: Manipulating the Document Object Model (DOM)

- Understanding the DOM and its structure
- Accessing DOM elements using document.getElementById, document.querySelector, etc.
- Modifying DOM elements: changing text, styles, attributes
- Handling user events: click events, input events

Module 7: Practical Project: Interactive To-Do List

- Applying JavaScript concepts learned in the course to build an interactive to-do list web application
- Creating a user interface with HTML and CSS
- Implementing JavaScript functionality: adding tasks, marking tasks as completed, deleting tasks
- Styling and enhancing the user experience

Prerequisites:

- No prior programming experience required
- Basic familiarity with HTML and CSS is beneficial but not necessary

This course is designed to provide a gentle introduction to JavaScript for absolute beginners, culminating in the development of a practical web application. Through a combination of theoretical explanations, coding exercises, and project

© 2024 Du Wun Aung. All rights reserved

work, students will gain a solid understanding of JavaScript fundamentals and the ability to apply them in real-world scenarios.

© 2024 Du Wun Aung. All rights reserved

Module 1: Introduction to JavaScript

JavaScript is a versatile programming language primarily used for adding interactivity and dynamic behavior to web pages. It is one of the three core technologies of web development, alongside HTML and CSS. In this module, we'll explore what JavaScript is and why it's essential for web development. Additionally, we'll dive into writing your first JavaScript code, covering basic techniques such as alerts, console.log, and comments.

What is JavaScript and Why is it Important?

JavaScript is a high-level, interpreted programming language that was initially developed for use in web browsers. It allows developers to create interactive elements, manipulate content, and respond to user actions on web pages. Unlike HTML (markup language) and CSS (styling language), JavaScript is a full-fledged programming language that adds functionality and behavior to static web content.

Key Points:

- JavaScript enables dynamic content updates without requiring page reloads, providing a smoother user experience.
- It is essential for creating interactive features such as forms validation, animations, and dynamic page elements.
- JavaScript is supported by all modern web browsers, making it a ubiquitous language for web development.
- With the rise of client-side frameworks like React, Angular, and Vue.js, JavaScript's importance has grown even further in building complex web applications.

Write Your First JavaScript Code: Alerts, console.log, and Comments

Now, let's dive into writing some actual JavaScript code. We'll start with simple examples to demonstrate basic JavaScript syntax and features.

© 2024 Du Wun Aung. All rights reserved

a. Alerts:

An alert is a built-in function in JavaScript that displays a message dialog with an optional message and an OK button. It's a quick way to provide information or prompt the user for input.

Example:

```
alert("Hello, world!");
```

This code will display a dialog box with the message "Hello, world!" when executed.

b. console.log:

console.log() is a method used for logging messages to the console. It's commonly used for debugging purposes, as it allows developers to inspect values, variables, and execution flow within their JavaScript code.

Example:

```
console.log("Hello, world!");
```

This code will log the message "Hello, world!" to the browser's console.

c. Comments:

Comments are non-executable lines of text used to annotate code for readability and documentation purposes. JavaScript supports both single-line (//) and multi-line (/* */) comments.

Example:

```
// This is a single-line comment
console.log("Hello, world!"); // Logging a message to the console

/*
 This is a multi-line comment
 It can span multiple lines
*/
```


© 2024 Du Wun Aung. All rights reserved

Using JavaScript in HTML Documents

You can include JavaScript code directly within HTML documents using the `<script>` tag. Here's how to embed JavaScript into an HTML file:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>JavaScript in HTML</title>
</head>
<body>

 <h1>JavaScript in HTML Example</h1>

 <!-- JavaScript code embedded within HTML -->
 <script>
 // JavaScript code goes here
 alert("Welcome to our website!");
 </script>

</body>
</html>
```

In this example, the JavaScript code enclosed within the `<script>` tags will be executed when the browser encounters it while rendering the HTML document. This allows you to seamlessly integrate JavaScript functionality into your web pages.

Exercises

Exercise 1: Alert Message

Create an alert message that welcomes the user to the website.

Answer:

```
alert("Welcome to our website!");
```


© 2024 Du Wun Aung. All rights reserved

Exercise 2: Console.log Messages

Log the following messages to the console:

"Hello, world!"
"Today is a great day!"
"JavaScript is awesome!"

Answer:

```
console.log("Hello, world!");
console.log("Today is a great day!");
console.log("JavaScript is awesome!");
```

Exercise 3: Comments

Add comments to the following code snippet to explain what each line does:

```


console.log("This is a comment exercise.");

console.log(5 + 3);

alert("Thanks for participating!");
```

Answer:

```
// Display a message in the console
console.log("This is a comment exercise.");

// Log the result of a mathematical operation
console.log(5 + 3);

// Alert a message to the user
alert("Thanks for participating!");
```

These exercises should provide you with practice in using alert messages, console.log statements, and comments effectively in JavaScript code.

© 2024 Du Wun Aung. All rights reserved

Conclusion

In this module, we've explored the fundamentals of JavaScript, understanding its significance in web development and writing our first lines of code. As we progress through the course, we'll delve deeper into JavaScript's capabilities and learn how to harness its power to create dynamic and interactive web experiences.

© 2024 Du Wun Aung. All rights reserved

Module 2: Variables and Data Types

In JavaScript, variables are used to store and manipulate data. Understanding variables and data types is fundamental to programming. In this module, we'll explore how to declare variables, understand different data types available in JavaScript, and how to work with them effectively.

Variables in JavaScript:

Variables are containers for storing data values. In JavaScript, variables can be declared using the `var`, `let`, or `const` keywords.

- `var`: Historically used for variable declaration. Variables declared with `var` are function-scoped.
- `let`: Introduced in ES6 (ECMAScript 2015) for block-scoped variables. Use `let` when the variable's value will change.
- `const`: Also introduced in ES6, `const` declares constants, which cannot be reassigned a new value.

Example:

```
var x = 10; // var declaration
let y = 20; // let declaration
const z = 30; // const declaration
```

Data Types in JavaScript

JavaScript is a loosely typed language, meaning variables can hold values of any data type without explicitly specifying the type. Common data types in JavaScript include:

- Primitive Data Types:
    - Number: Represents numeric values. Can be integers, floating-point numbers, or scientific notation.
    - String: Represents textual data enclosed in single (' ') or double (" ") quotes.
    - Boolean: Represents a logical value, `true` or `false`.
    - Undefined: Represents an uninitialized variable.

© 2024 Du Wun Aung. All rights reserved

- Null: Represents the intentional absence of any object value.
- Symbol: Introduced in ES6, represents unique identifiers.

Example:

```
let num = 42; // Number
let str = "Hello, World!"; // String
let isTrue = true; // Boolean
let myVar; // Undefined
let empty = null; // Null
let sym = Symbol('foo'); // Symbol
```

- Non-Primitive (Reference) Data Types:
  - Object: Represents a collection of key-value pairs. Objects can be created using object literals `{}`, constructors, or classes.

Example:

```
let person = { name: "John", age: 30 }; // Object literal
let arr = [1, 2, 3]; // Array (special kind of object)
let today = new Date(); // Date object
```

Working with Variables and Data Types

- Variable Naming Conventions: Variable names should be descriptive, meaningful, and follow camelCase convention. They cannot begin with a number, and certain keywords are reserved.

Example:

```
let myVariable; // Good variable name
let my_variable; // Not recommended (use camelCase)
let 1num; // Invalid variable name (s
```

- Typeof Operator: The `typeof` operator returns the data type of its operand.

Example:

© 2024 Du Wun Aung. All rights reserved

```
console.log(typeof num); // Output: "number"
console.log(typeof str); // Output: "string"
console.log(typeof isTrue); // Output: "boolean"
```

Exercises

Exercise 1: Variable Declaration

Declare variables for the following:

Your age.
Your name.
Your favorite color.

Answer:

```
let age = 25;
let name = "John";
let favoriteColor = "blue";
```

Exercise 2: Arithmetic Operations

Perform arithmetic operations using variables:

Add your age and your friend's age.
Subtract your age from 100.
Multiply your age by 2.

Answer:

```
let yourAge = 25;
let friendsAge = 30;

let sumAges = yourAge + friendsAge; // 55
let subtractFrom100 = 100 - yourAge; // 75
let multiplyBy2 = yourAge * 2; // 50
```

Exercise 3: String Concatenation

Combine variables to create meaningful strings:

Concatenate your name and favorite color to form a greeting message.

© 2024 Du Wun Aung. All rights reserved

Concatenate your age and your friend's age to form a sentence.

Answer:

```
let greetingMessage = "Hello, " + name + "! Your favorite color is " + favoriteColor
+ ".";
let ageSentence = "I am " + yourAge + " years old, and my friend is " + friendsAge +
" years old.";
```

Exercise 4: Boolean Variables

Create boolean variables representing the following:

Whether it's raining today.
Whether you are a student (true/false).

Answer:

```
let isRaining = true;
let isStudent = false;
```

Exercise 5: Null and Undefined

Create variables representing the following:

A variable initialized with a null value.
A variable without initialization (undefined).

Answer:

```
let nullVariable = null;
let undefinedVariable;
```

These exercises should help reinforce your understanding of variables and data types in JavaScript.

Conclusion

Understanding variables and data types is crucial for writing effective JavaScript code. In this module, we've covered variable declaration, common data types in

© 2024 Du Wun Aung. All rights reserved

JavaScript, and how to work with them. Having a solid grasp of variables and data types will lay a strong foundation for your JavaScript programming journey.

© 2024 Du Wun Aung. All rights reserved

Module 3: Operators and Expressions

In JavaScript, operators are symbols used to perform operations on operands, such as variables or values. Expressions are combinations of variables, values, and operators that result in a single value. Understanding operators and expressions is essential for manipulating data and controlling program flow. In this module, we'll explore various types of operators and how to use them effectively in JavaScript.

Arithmetic Operators

Arithmetic operators are used to perform mathematical operations on numeric operands.

- Addition (+): Adds two operands.
- Subtraction (-): Subtracts the right operand from the left operand.
- Multiplication (*): Multiplies two operands.
- Division (/): Divides the left operand by the right operand.
- Modulus (%): Returns the remainder of the division of two operands.

Example:

```
let x = 10;
let y = 5;
let sum = x + y; // 10 + 5 = 15
let difference = x - y; // 10 - 5 = 5
let product = x * y; // 10 * 5 = 50
let quotient = x / y; // 10 / 5 = 2
let remainder = x % y; // 10 % 5 = 0
```

Comparison Operators

Comparison operators are used to compare two values and return a Boolean result (true or false).

- Equal to (==): Returns `true` if the operands are equal.
- Strict equal to (===): Returns `true` if the operands are equal and of the same type.
- Not equal to (!=): Returns `true` if the operands are not equal.

© 2024 Du Wun Aung. All rights reserved

- Strict not equal to (!==): Returns `true` if the operands are not equal or not of the same type.
- Greater than (>): Returns `true` if the left operand is greater than the right operand.
- Less than (<): Returns `true` if the left operand is less than the right operand.
- Greater than or equal to (>=): Returns `true` if the left operand is greater than or equal to the right operand.
- Less than or equal to (<=): Returns `true` if the left operand is less than or equal to the right operand.

Example:

```
let a = 5;
let b = 10;
console.log(a == b); // false
console.log(a !== b); // true
console.log(a < b); // true
```

Logical Operators

Logical operators are used to combine multiple conditions and return a Boolean result.

- Logical AND (&&): Returns `true` if both operands are true.
- Logical OR (||): Returns `true` if at least one operand is true.
- Logical NOT (!): Returns the opposite boolean value of the operand.

Example:

```
let x = 5;
let y = 10;
console.log(x > 3 && y < 20); // true
console.log(x > 7 || y < 5); // false
console.log(!(x > 7)); // true
```

String Concatenation Operator

The string concatenation operator (+) is used to concatenate (join) two or more strings together.

© 2024 Du Wun Aung. All rights reserved

Example:

```
let firstName = "John";
let lastName = "Doe";
let fullName = firstName + " " + lastName; // "John Doe"
```

Exercises

Exercise 1:

Calculate the result of the following arithmetic operations:

15 + 7
30 - 12
5 * 6
20 / 4
10 % 3 (remainder of dividing 10 by 3)

Answers:

15 + 7 = 22
30 - 12 = 18
5 * 6 = 30
20 / 4 = 5
10 % 3 = 1

Exercise 2:

Determine whether the following comparisons are true or false:

10 > 5
20 < 8
5 === 5
10 != 10
15 >= 15

Answers:

10 > 5 (true)
20 < 8 (false)
5 === 5 (true)

© 2024 Du Wun Aung. All rights reserved

10 != 10 (false)
15 >= 15 (true)

Exercise 3:

Evaluate the following logical expressions:

(10 > 5) && (20 < 8)
(5 === 5) || (10 != 10)
!(15 >= 15)
(8 < 5) && (6 === 6)
(4 > 3) || (7 < 5)

Answers:

(10 > 5) && (20 < 8) => (true) && (false) => false
(5 === 5) || (10 != 10) => (true) || (false) => true
!(15 >= 15) => !(true) => false
(8 < 5) && (6 === 6) => (false) && (true) => false
(4 > 3) || (7 < 5) => (true) || (false) => true

Exercise 4:

Concatenate the following strings:

"Hello, " + "world!"
"My favorite color is " + "blue."
"The value of pi is approximately " + 3.14
"Today is " + "Monday."
"I am " + 25 + " years old."

Answers:

"Hello, " + "world!" => "Hello, world!"
"My favorite color is " + "blue." => "My favorite color is blue."
"The value of pi is approximately " + 3.14 => "The value of pi is approximately 3.14"
"Today is " + "Monday." => "Today is Monday."
"I am " + 25 + " years old." => "I am 25 years old."

These exercises should help reinforce your understanding of arithmetic, comparison, logical operators, and string concatenation in JavaScript.

© 2024 Du Wun Aung. All rights reserved

Conclusion

Operators and expressions are fundamental concepts in JavaScript for performing various operations and making decisions in your code. In this module, we've covered arithmetic, comparison, logical, and string concatenation operators and how to use them effectively to manipulate data and control program flow. Understanding these concepts will enable you to write more dynamic and expressive JavaScript code.

© 2024 Du Wun Aung. All rights reserved

Module 4: Control Flow and Loops

In JavaScript, control flow and loop structures are essential for executing code conditionally and repetitively. This module covers various control flow statements and loop constructs, including conditional statements, switch statements, and different types of loops.

Conditional Statements

Conditional statements allow you to execute different blocks of code based on conditions. The most common conditional statements in JavaScript are if-else statements.

```
let num = 10;

if (num > 0) {
 console.log("The number is positive.");
} else if (num < 0) {
 console.log("The number is negative.");
} else {
 console.log("The number is zero.");
}
```

Switch Statement for Multi-way Branching

The switch statement is used for multi-way branching based on the value of an expression.

```
let day = new Date().getDay();

switch (day) {
 case 0:
 console.log("Sunday");
 break;
 case 1:
 console.log("Monday");
 break;
 case 2:
 console.log("Tuesday");
 break;
 // and so on...
 default:
 console.l
```


© 2024 Du Wun Aung. All rights reserved

Loops

Loops allow you to execute a block of code repeatedly until a certain condition is met. JavaScript provides different loop constructs, including for loops, while loops, and do-while loops.

a. For Loop

The for loop is used to execute a block of code a specified number of times.

```
for (let i = 0; i < 5; i++) {
 console.log("Iteration " + i);
}
```

b. While Loop

The while loop is used to execute a block of code as long as a condition is true.

```
let count = 0;
while (count < 5) {
 console.log("Count: " + count);
 count++;
}
```

Looping Through Arrays and Objects

Arrays and objects are fundamental data structures in JavaScript. You can iterate through them using loops to access and manipulate their elements.

a. Looping Through Arrays

```
let fruits = ["apple", "banana", "orange"];

for (let i = 0; i < fruits.length; i++) {
 console.log(fruits[i]);
}
```

b. Looping Through Objects

© 2024 Du Wun Aung. All rights reserved

```
let person = {
 name: "John",
 age: 30,
 city: "New York"
};

for (let key in person) {
 console.log(key + ": " + person[key]);
}
```

Exercises

Exercise 1: Check Even or Odd

Write a program that takes an integer as input and prints whether it's even or odd.

Answer:

```
let num = 7;

if (num % 2 === 0) {
 console.log(num + " is even.");
} else {
 console.log(num + " is odd.");
}
```

Exercise 2: Multiplication Table

Write a program to print the multiplication table (up to 10) of a given number.

Answer:

```
let number = 5;

for (let i = 1; i <= 10; i++) {
 console.log(number + " * " + i + " = " + (number * i));
}
```

Exercise 3: Factorial Calculation

Write a program to calculate the factorial of a given number.

Answer:

© 2024 Du Wun Aung. All rights reserved

```
let n = 5;
let factorial = 1;

for (let i = 1; i <= n; i++) {
 factorial *= i;
}
console.log("Factorial of " + n + " is: " + factorial);
```

Exercise 4: Print Even Numbers

Write a program to print all even numbers from 1 to 20.

Answer:

```
for (let i = 2; i <= 20; i += 2) {
 console.log(i);
}
```

Exercise 5: Reverse Counting

Write a program to print numbers from 10 to 1 in reverse order.

Answer:

```
for (let i = 10; i >= 1; i--) {
 console.log(i);
}
```

Exercise 6: Print Fibonacci Series

Write a program to print the Fibonacci series up to a certain limit (e.g., 50).

Answer:

```
let limit = 50;
let n1 = 0, n2 = 1, nextTerm;

console.log("Fibonacci Series:");
console.log(n1);
console.log(n2);

nextTerm = n1 + n2;

while (nextTerm <= limit) {
 console.log(nextTerm);
```


© 2024 Du Wun Aung. All rights reserved

```
 n1 = n2;
 n2 = nextTerm;
 nextTerm = n1 + n2;
}
```

Note:

The Fibonacci series is a sequence of numbers where each number is the sum of the two preceding ones, usually starting with 0 and 1. The sequence looks like this:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

The Fibonacci sequence can be defined recursively by the following formula:

F(n) = F(n-1) + F(n-2)

Where F(n) is the nth term in the sequence, and F(n-1) and F(n-2) are the previous two terms.

Here's how the sequence is generated:

- *Start with 0 and 1 as the first two numbers.*
- *Each subsequent number in the sequence is the sum of the two preceding numbers.*

So, for example:

- *The third number is 0 + 1 = 1*
- *The fourth number is 1 + 1 = 2*
- *The fifth number is 1 + 2 = 3*
- *The sixth number is 2 + 3 = 5*
- *And so on...*

The Fibonacci sequence has many interesting properties and applications in mathematics, computer science, and nature, making it a fascinating topic of study.

Exercise 7: Check Prime Number

Write a program to check whether a given number is prime or not.

Answer:

```
let num = 11;
let isPrime = true;

for (let i = 2; i <= Math.sqrt(num); i++) {
 if (num % i === 0) {
 isPrime = false;
 break;
```


© 2024 Du Wun Aung. All rights reserved

```
 }
}

if (isPrime) {
 console.log(num + " is a prime number.");
} else {
 console.log(num + " is not a prime number.");
}
```

Exercise 8: Sum of Array Elements

Write a program to find the sum of elements in an array.

Answer:

```
let numbers = [1, 2, 3, 4, 5];
let sum = 0;

for (let i = 0; i < numbers.length; i++) {
 sum += numbers[i];
}

console.log("Sum of array elements: " + sum);
```

These exercises cover a range of control flow and loop concepts in JavaScript. Practice them to enhance your understanding and proficiency in using these constructs.

Conclusion

In this module, you've learned about control flow statements such as if-else and switch, different types of loops including for and while loops, and how to loop through arrays and objects in JavaScript. Practice these concepts to become proficient in controlling the flow of your programs and performing repetitive tasks efficiently.

© 2024 Du Wun Aung. All rights reserved

Module 5: Functions

In JavaScript, functions are blocks of reusable code that perform a specific task. They allow you to break down your program into smaller, manageable pieces and make your code more modular and organized. This module covers the fundamentals of functions in JavaScript.

Introduction to Functions

Functions are blocks of code that can be defined once and executed repeatedly. They can take inputs, perform actions, and return results. Functions help in organizing code, promoting reusability, and making code more readable.

Declare and Invoke Functions

Declaring a function involves defining its name, parameters (optional), and the code block to be executed. Functions can then be invoked or called to execute the code within them.

```
// Function declaration
function greet() {
 console.log("Hello!");
}

// Function invocation
greet();
```

Parameters and Return Values

Functions can accept inputs called parameters, which are used as variables within the function's scope. Functions can also return values using the return statement.

```
function add(a, b) {
 return a + b;
}

let result = add(3, 5); // result = 8
```


© 2024 Du Wun Aung. All rights reserved

Scope

Scope refers to the visibility of variables within a program. JavaScript has two types of scope: global scope and local scope. Variables declared outside of any function have global scope, while variables declared within a function have local scope.

```
let globalVar = "I'm global";

function myFunction() {
 let localVar = "I'm local";
 console.log(globalVar); // Accessible
}

console.log(globalVar); // Accessible
console.log(localVar); // Error: localVar is not defined
```

Exercises

Write a function that takes two numbers as parameters and returns their sum.
Answer:

```
nction sum(a, b) {
    return a + b;
}
console.log(sum(3, 5)); // Output: 8
```

Write a function that takes a string as a parameter and returns its length.
Answer:

```
function stringLength(str) {
    return str.length;
}
console.log(stringLength("Hello")); // Output: 5
```

Create a function that checks if a given number is even or odd and returns the result.
Answer:

```
function checkEvenOrOdd(num) {
    if (num % 2 === 0) {
```


© 2024 Du Wun Aung. All rights reserved

```
        return "Even";
    } else {
        return "Odd";
    }
}
console.log(checkEvenOrOdd(7)); // Output: Odd
```

Write a function that takes an array of numbers and returns the sum of all elements.
Answer:

```
function sumArray(arr) {
    let sum = 0;
    for (let i = 0; i < arr.length; i++) {
        sum += arr[i];
    }
    return sum;
}
console.log(sumArray([1, 2, 3, 4, 5])); // Output: 15
```

Implement a function that calculates the factorial of a given number recursively.
Note: Factorial is a mathematical operation that is denoted by the symbol '!'. It represents the product of all positive integers up to a given number. For example, the factorial of 5 (written as 5!) is calculated as:

5! = 5 × 4 × 3 × 2 × 1 = 120

Answer:

```
function factorial(n) {
    if (n === 0 || n === 1) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}
console.log(factorial(5)); // Output: 120
```

Create a function that takes two strings as parameters and concatenates them.
Answer:

```
function concatenateStrings(str1, str2) {
    return str1 + str2;
```


© 2024 Du Wun Aung. All rights reserved

```
}
console.log(concatenateStrings("Hello", "World")); // Output: HelloWorld
```

Write a function that takes an array of numbers as a parameter and returns the largest number in the array.
Answer:

```
function findLargestNumber(arr) {
    let max = arr[0];
    for (let i = 1; i < arr.length; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}
console.log(findLargestNumber([1, 5, 3, 9, 2])); // Output: 9
```

Implement a function that checks if a given year is a leap year and returns true or false. (Note: In the Gregorian calendar, which is the most widely used calendar system today, a year is a leap year if it is divisible by 4, except for years that are divisible by 100. However, years that are divisible by 400 are still considered leap years.)
Answer:

```
function isLeapYear(year) {
    if ((year % 4 === 0 && year % 100 !== 0) || year % 400 === 0) {
        return true;
    } else {
        return false;
    }
}
console.log(isLeapYear(2024)); // Output: true
```

Conclusion

Functions are a fundamental concept in JavaScript programming. By understanding how to declare, invoke, and utilize parameters and return values effectively, along with understanding scope, you'll be able to write more modular and reusable code.

© 2024 Du Wun Aung. All rights reserved

Practice with the provided exercises to solidify your understanding of functions in JavaScript.

© 2024 Du Wun Aung. All rights reserved

Module 6: Manipulating the Document Object Model (DOM)

In web development, the Document Object Model (DOM) is a programming interface for web documents. It represents the structure of an HTML document as a tree of objects, allowing JavaScript to interact with and manipulate the content, structure, and style of a web page dynamically.

Introduction to the DOM

The DOM is a hierarchical representation of an HTML document. Each HTML element, attribute, and text node in the document is represented as an object in the DOM tree. JavaScript can access and manipulate these objects to dynamically modify the web page.

Accessing DOM Elements

JavaScript provides methods to access DOM elements by their IDs, classes, tag names, or CSS selectors. Some common methods include `getElementById`, `getElementsByClassName`, `getElementsByTagName`, `querySelector`, and `querySelectorAll`.

Example:

```
// Accessing an element by ID
let element = document.getElementById("myElement");

// Accessing elements by class name
let elements = document.getElementsByClassName("myClass");

// Accessing elements by tag name
let paragraphs = document.getElementsByTagName("p");

// Accessing an element using a CSS selector
let element = document.querySelector("#myElement");
```

Manipulating DOM Elements

Once you've selected DOM elements, you can manipulate them by changing their properties, attributes, styles, or content. Common methods for manipulation include `innerHTML`, `textContent`, `setAttribute`, `style`, `classList`, etc.

© 2024 Du Wun Aung. All rights reserved

Example:

```
// Changing text content
element.textContent = "New text";

// Changing attribute value
element.setAttribute("src", "newImage.jpg");

// Changing CSS styles
element.style.color = "red";

// Adding or removing classes
element.classList.add("newClass");
element.classList
```

Creating and Removing DOM Elements

JavaScript allows you to create new DOM elements dynamically and add them to the document using methods like `createElement` and `appendChild`. Similarly, you can remove existing elements from the document using methods like `removeChild`.

Example:

```
// Creating a new element
let newElement = document.createElement("div");

// Appending the new element to an existing element
parentElement.appendChild(newElement);

// Removing an existing element
parentElement.removeChild(existingElement);
```

Event Handling

The DOM enables you to respond to user interactions (events) such as clicks, mouse movements, keyboard input, etc. You can attach event listeners to DOM elements to execute specific actions when events occur, using methods like `addEventListener`.

Example:

```
// Adding a click event listener
element.addEventListener("click", function() {
 console.log("Element clicked!");
```


© 2024 Du Wun Aung. All rights reserved

```
});
```

Exercises

Exercise 1: Change Text Content

Create an HTML document with a paragraph element with ID "demo". Using JavaScript, change the text content of the paragraph to "Hello, World!".

Answer:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Exercise 1</title>
</head>
<body>
 <p id="demo">Original text</p>
 <script>
 document.getElementById("demo").textContent = "Hello, World!";
 </script>
</body>
</html>
```

Exercise 2: Change Background Color

Create an HTML document with a div element with class "box". Using JavaScript, change the background color of the div to "red".

Answer:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Exercise 2</title>
 <style>
 .box {
 width: 100px;
 height: 100px;
 background-color: blue; /* Initial color */
 }
 </style>
```


© 2024 Du Wun Aung. All rights reserved

```
</head>
<body>
 <div class="box"></div>
 <script>
 document.querySelector(".box").style.backgroundColor = "red";
 </script>
</body>
</html>
```

Exercise 3: Create New Element

Create an HTML document with an empty unordered list (ul) element with ID "myList". Using JavaScript, create a new list item (li) element with text content "Item 1" and append it to the unordered list.

Answer:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Exercise 3</title>
</head>
<body>
 <ul id="myList"></ul>
 <script>
 let newListElement = document.createElement("li");
 newListElement.textContent = "Item 1";
 document.getElementById("myList").appendChild(newListElement);
 </script>
</body>
</html>
```

Exercise 4: Remove Element

Create an HTML document with a paragraph element with ID "removeMe". Using JavaScript, remove the paragraph element from the document.

Answer:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Exercise 4</title>
```


© 2024 Du Wun Aung. All rights reserved

```
</head>
<body>
 <p id="removeMe">This paragraph will be removed.</p>
 <script>
 let elementToRemove = document.getElementById("removeMe");
 elementToRemove.parentNode.removeChild(elementToRemove);
 </script>
</body>
</html>
```

Exercise 5: Event Handling

Create an HTML document with a button element with ID "myButton". Using JavaScript, add a click event listener to the button that alerts "Button clicked!" when clicked.

Answer:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Exercise 5</title>
</head>
<body>
 <button id="myButton">Click me</button>
 <script>
 document.getElementById("myButton").addEventListener("click", function() {
 alert("Button clicked!");
 });
 </script>
</body>
</html>
```

These exercises demonstrate various aspects of DOM manipulation in JavaScript. Practice with them to improve your understanding and proficiency in working with the DOM.

Conclusion

Understanding how to manipulate the DOM is crucial for building dynamic and interactive web pages with JavaScript. By mastering DOM manipulation techniques,

© 2024 Du Wun Aung. All rights reserved

you can create engaging user experiences and enhance the functionality of your web applications.

Module 7: Practical Project: Interactive To-Do List

In this module, we'll create an interactive to-do list application using HTML, CSS, and JavaScript. This project will demonstrate how to manipulate the DOM to add, remove, and update tasks dynamically.

Project Overview

The to-do list will have the following features:

Ability to add new tasks.
Mark tasks as completed.
Delete tasks.
Filter tasks based on their completion status (all, active, completed).
Clear all completed tasks at once.

HTML Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>To-Do List</title>
 <link rel="stylesheet" href="styles.css">
</head>
<body>
 <div class="container">
 <h1>To-Do List</h1>
 <input type="text" id="taskInput" placeholder="Add new task...">
 <ul id="taskList"></ul>
 <div class="footer">
 <div class="task-count" id="taskCount"></div>
 <div class="filter-options">
 <button id="allTasks">All</button>
 <button id="activeTasks">Active</button>
 <button id="completedTasks">Completed</button>
 </div>
 <button id="clearCompleted">Clear Completed</button>
 </div>
 </div>
 <script src="script.js"></script>
</body>
```


© 2024 Du Wun Aung. All rights reserved

```
</html>
```

CSS (styles.css)

```
body {
 font-family: Arial, sans-serif;
}

.container {
 max-width: 400px;
 margin: 20px auto;
 padding: 0 20px;
}

h1 {
 text-align: center;
}

input[type="text"] {
 width: 100%;
 padding: 10px;
 margin-top: 10px;
 box-sizing: border-box;
}

ul {
 list-style-type: none;
 padding: 0;
}

li {
 display: flex;
 align-items: center;
 padding: 10px;
 border-bottom: 1px solid #ddd;
}

.completed {
 text-decoration: line-through;
 color: #aaa;
}

.footer {
 margin-top: 20px;
 display: flex;
 justify-content: space-between;
 align-items: center;
}

.filter-options button {
```


© 2024 Du Wun Aung. All rights reserved

```
 margin-right: 5px;
}

button {
 cursor: pointer;
}
```

JavaScript (script.js)

```
const taskInput = document.getElementById("taskInput");
const taskList = document.getElementById("taskList");
const taskCount = document.getElementById("taskCount");
const allTasksBtn = document.getElementById("allTasks");
const activeTasksBtn = document.getElementById("activeTasks");
const completedTasksBtn = document.getElementById("completedTasks");
const clearCompletedBtn = document.getElementById("clearCompleted");

let tasks = [];

function renderTasks() {
 taskList.innerHTML = "";
 tasks.forEach((task, index) => {
 const taskItem = document.createElement("li");
 taskItem.innerHTML = `
 <input type="checkbox" id="task${index}" ${task.completed ? "checked" : ""}>
 <label for="task${index}" class="${task.completed ? "completed" :
""}">${task.text}</label>
 <button class="delete-btn">X</button>
 `;
 taskList.appendChild(taskItem);
 });
 updateTaskCount();
}

function updateTaskCount() {
 const uncompletedTasks = tasks.filter(task => !task.completed).length;
 taskCount.textContent = `${uncompletedTasks} ${uncompletedTasks === 1 ? "task" :
"tasks"} left`;
}

function addTask(text) {
 tasks.push({ text, completed: false });
 renderTasks();
}

function toggleTask(index) {
 tasks[index].completed = !tasks[index].completed;
 renderTasks();
}
```


© 2024 Du Wun Aung. All rights reserved

```
function deleteTask(index) {
 tasks.splice(index, 1);
 renderTasks();
}

function clearCompletedTasks() {
 tasks = tasks.filter(task => !task.completed);
 renderTasks();
}

taskInput.addEventListener("keypress", function(event) {
 if (event.key === "Enter" && taskInput.value.trim() !== "") {
 addTask(taskInput.value.trim());
 taskInput.value = "";
 }
});

taskList.addEventListener("change", function(event) {
 if (event.target.type === "checkbox") {
 const index = event.target.id.slice(4);
 toggleTask(index);
 }
});

taskList.addEventListener("click", function(event) {
 if (event.target.classList.contains("delete-btn")) {
 const index = event.target.parentElement.querySelector("input").id.slice(4);
 deleteTask(index);
 }
});

allTasksBtn.addEventListener("click", function() {
 renderTasks();
});

activeTasksBtn.addEventListener("click", function() {
 const activeTasks = tasks.filter(task => !task.completed);
 taskList.innerHTML = "";
 activeTasks.forEach((task, index) => {
 const taskItem = document.createElement("li");
 taskItem.innerHTML = `
 <input type="checkbox" id="task${index}">
 <label for="task${index}">${task.text}</label>
 <button class="delete-btn">X</button>
 `;
 taskList.appendChild(taskItem);
 });
});

completedTasksBtn.addEventListener("click", function() {
 const completedTasks = tasks.filter(task => task.completed);
 taskList.innerHTML = "";
 completedTasks.forEach((task, index) => {
```


© 2024 Du Wun Aung. All rights reserved

```
 const taskItem = document.createElement("li");
 taskItem.innerHTML = `
 <input type="checkbox" id="task${index}" checked>
 <label for="task${index}" class="completed">${task.text}</label>
 <button class="delete-btn">X</button>
 `;
 taskList.appendChild(taskItem);
 });
});

clearCompletedBtn.addEventListener("click", function() {
 clearCompletedTasks();
});

// Initial rendering
renderTasks();
```

Conclusion

This project demonstrates how to create an interactive to-do list application using HTML, CSS, and JavaScript. The application allows users to add, mark as complete, delete, and filter tasks dynamically.

© 2024 Du Wun Aung. All rights reserved

Next Steps

As you move forward from completing this book, consider the following next steps:

Apply What You've Learned: Take the knowledge and skills you've gained from this book and apply them in practical scenarios. Whether it's implementing code examples in your projects or using the concepts discussed in your daily work, practical application is key to solidifying your understanding.

Further Learning: Continuously expand your knowledge by exploring additional resources, such as online tutorials, courses, books, and documentation. There's always more to learn, and staying curious and open to new information will help you grow as a developer.

Build Projects: Practice your skills by building projects that interest you. Whether it's a small personal website, a web application, or a mobile app, hands-on projects provide valuable experience and help reinforce your learning.

Collaborate and Network: Engage with other developers, join online communities, attend meetups, and participate in hackathons. Collaboration and networking not only expose you to different perspectives and ideas but also provide opportunities for learning and growth.

Stay Updated: Stay abreast of the latest developments and trends in web development. Technology is constantly evolving, so staying updated with the latest tools, frameworks, and best practices is essential to remain competitive in the field.

Reflect and Iterate: Take time to reflect on your learning journey and identify areas for improvement. Set goals for yourself, experiment with new techniques, and iterate on your skills to continually progress as a developer.

Remember that learning is a lifelong journey, and each step you take brings you closer to your goals. Embrace challenges, stay curious, and keep pushing yourself to reach new heights in your web development journey.

© 2024 Du Wun Aung. All rights reserved

In addition to the concepts covered in this book, consider exploring the following programming languages to broaden your skillset and deepen your understanding of web development:

Python: Python is a versatile and beginner-friendly language known for its simplicity and readability. It's widely used in web development, data analysis, machine learning, and more. Learning Python can open up opportunities in various domains beyond web development.

Ruby: Ruby is a dynamic, object-oriented programming language known for its elegant syntax and developer-friendly environment. It's commonly used with the Ruby on Rails framework for web development, making it a valuable language to explore for building web applications.

Java: Java is a robust, cross-platform language commonly used for building enterprise-level web applications, Android mobile apps, and backend services. Understanding Java can be beneficial for developing scalable and secure web applications.

C#: C# is a powerful, statically typed language developed by Microsoft and commonly used for building desktop, web, and mobile applications within the .NET ecosystem. Learning C# can open up opportunities in web development using frameworks like ASP.NET.

Go: Go, also known as Golang, is a statically typed, compiled language developed by Google. It's known for its simplicity, performance, and concurrency support, making it a popular choice for building scalable web services and backend systems.

Swift: Swift is a modern, open-source programming language developed by Apple for building iOS, macOS, watchOS, and tvOS applications. If you're interested in mobile app development for Apple platforms, learning Swift can be advantageous.

JavaScript Frameworks: Dive deeper into JavaScript by exploring popular frameworks and libraries such as React, Angular, Vue.js, and Node.js. These frameworks offer powerful tools and patterns for building interactive web applications and scalable backend services.

Exploring these programming languages will not only expand your technical skills but also provide you with a broader perspective on software development. As you continue your learning journey, don't hesitate to experiment with different languages and technologies to find what resonates best with your interests and career goals.

© 2024 Du Wun Aung. All rights reserved

Dear Reader,

I want to take a moment to express my heartfelt gratitude to you, the reader, for choosing to engage with this book. Your interest and support mean the world to me, and I am genuinely grateful for the opportunity to share my knowledge and insights with you.

Creating this book has been a labor of love, and I hope that you find it informative, inspiring, and enjoyable to read. Your willingness to explore new ideas, learn, and grow is truly admirable, and I commend you for your dedication to personal and intellectual enrichment.

As you delve into the pages of this book, I encourage you to approach the content with an open mind and a curious spirit. Embrace the journey of discovery, ask questions, seek understanding, and above all, apply what you learn to enrich your life and the lives of those around you.

Once again, thank you for your support and enthusiasm. I am deeply grateful for the opportunity to embark on this journey with you, and I look forward to the insights and discoveries that await us.

Warm regards,
Du Wun Aung

© 2024 Du Wun Aung. All rights reserved