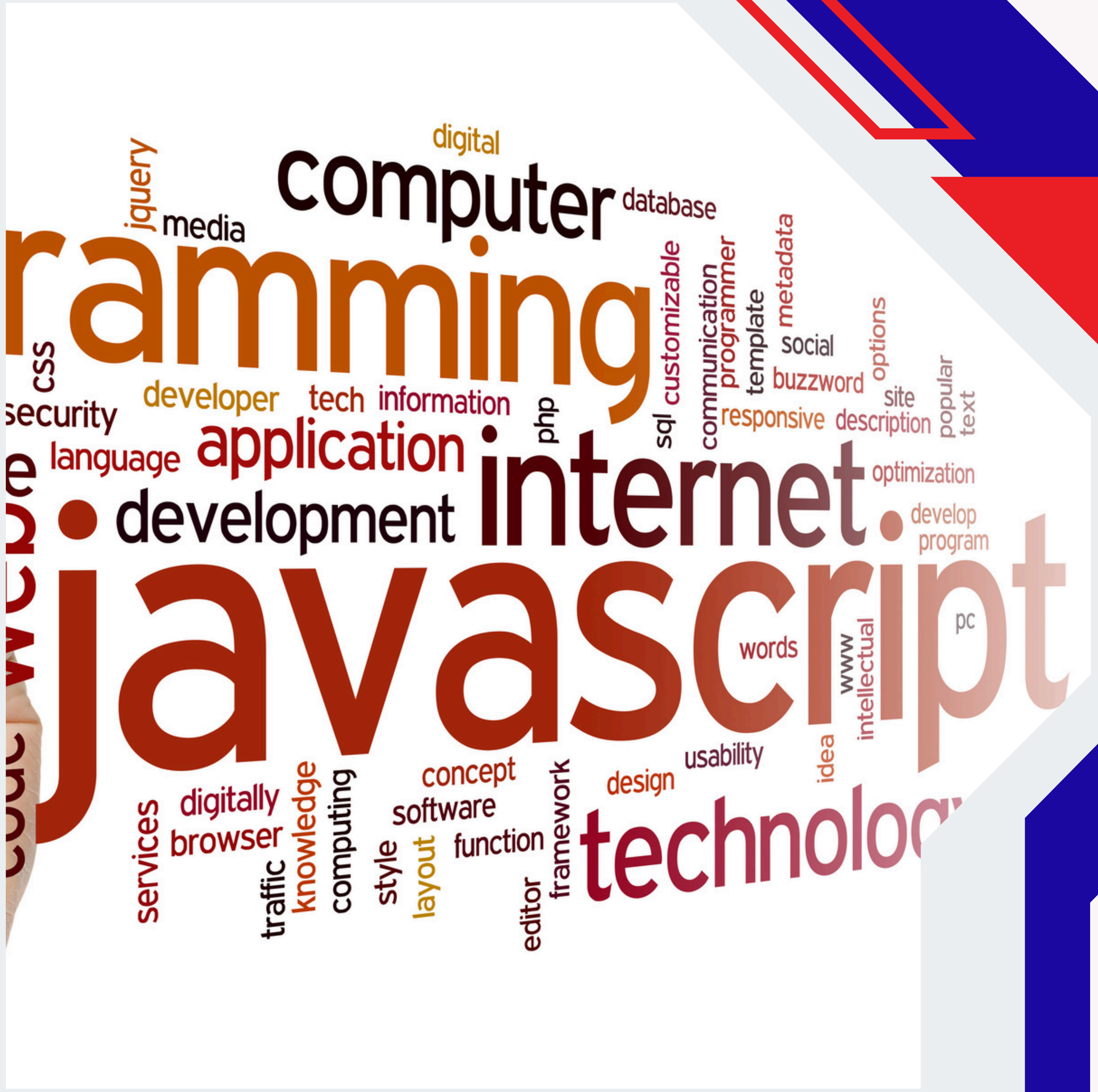
JAVASCRIPT DEVELOPMENT



WEB APPLICATION DEVELOPMENT | IT6315

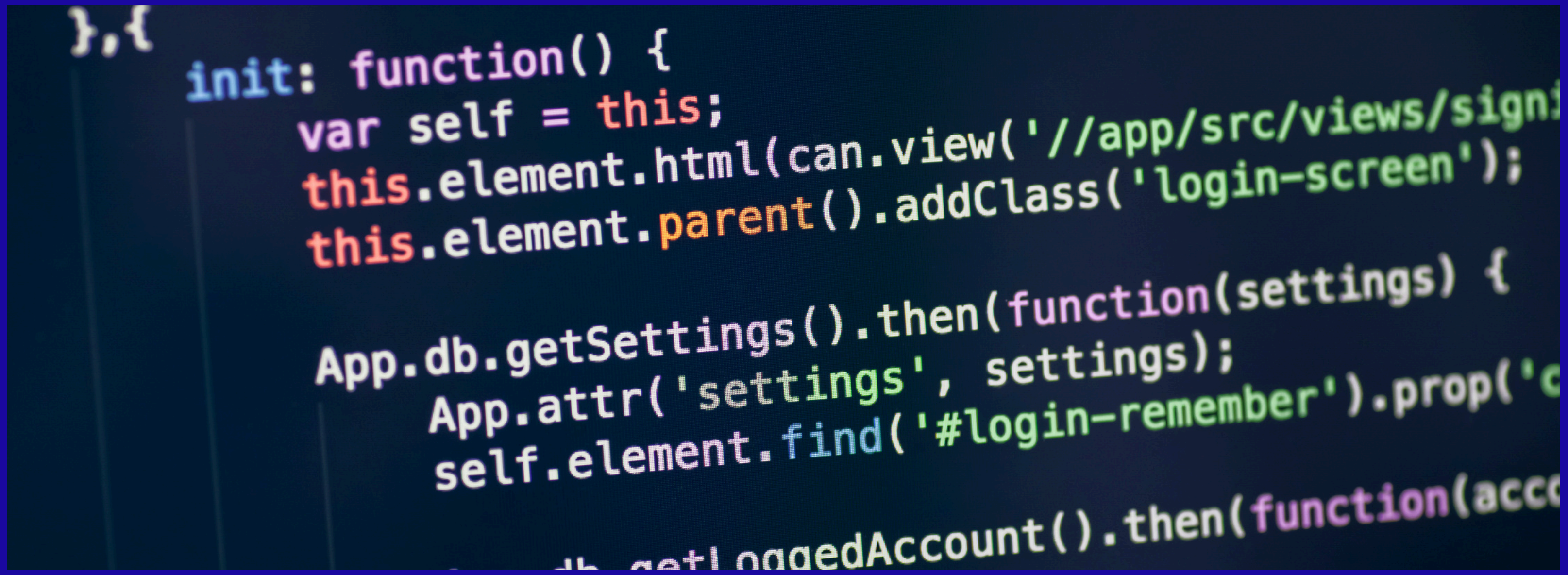
JOHN PAUL M. MANUEL | ACLC COLLEGE OF TAYTAY

ACLC COLLEGE OF TAYTAY
(A Member of AMA Education System)

VERSION 01-250119-2425

WHAT IS JAVASCRIPT

JavaScript is a scripting or programming language that allows you to implement complex features on web pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. — you can bet that JavaScript is probably involved.



```
},{
  init: function() {
    var self = this;
    this.element.html(can.view('//app/src/views/signi
    this.element.parent().addClass('login-screen');

    App.db.getSettings().then(function(settings) {
      App.attr('settings', settings);
      self.element.find('#login-remember').prop('c

      db.getLoggedAccount().then(function(acc
```

JAVASCRIPT DEVELOPMENT

WHY TO LEARN JAVASCRIPT?

Versatility: JavaScript can be used to develop (using ElectronJS) websites, games (Using Phaser and Three.js), mobile apps (using React Native), and more.

Client Side: JavaScript is the main language for client-side logic and is supported by almost all browsers. There is a big list of frameworks and libraries like React JS, Angular JS, and Vue JS.

Server-Side: With runtime environments like Node.js and Frameworks like Express.js, JavaScript is now widely used for building server-side applications.

Machine Learning: With Libraries like Tensorflow.JS, JavaScript can be used to develop and train machine learning models.

JAVASCRIPT DEVELOPMENT

DIFFERENCE BETWEEN JAVASCRIPT AND JAVA

JavaScript and Java are distinct languages with different purposes. Java is an object-oriented programming language for building standalone applications, while JavaScript is primarily used to enhance web pages with interactivity and dynamic content.

JAVASCRIPT DEVELOPMENT

VARIABLES AND DATATYPES

Variables

A variable is like a container that holds data that can be reused or updated later in the program. In JavaScript, variables are declared using the keywords var, let, or const.

The **var** keyword is used to declare a variable. It has a function-scoped or globally-scoped behaviour.

The **let** keyword is introduced in ES6, has block scope and cannot be re-declared in the same scope.

The **const** keyword declares variables that cannot be reassigned. It's block-scoped as well.

VARIABLES AND DATATYPES

Data Types

JavaScript supports various datatypes, which can be broadly categorized into primitive and non-primitive types.

Primitive Datatypes

Primitive datatypes represent single values and are immutable.

1. **Number:** Represents numeric values (integers and decimals).

```
let n = 42;
let pi = 3.14;
```

2. **String**: Represents text enclosed in single or double quotes.

```
let s = "Hello, World!";
```

JAVASCRIPT DEVELOPMENT

VARIABLES AND DATATYPES

3. **Boolean:** Represents a logical value (true or false).

```
let bool= true;
```

4. **Undefined:** A variable that has been declared but not assigned a value.

```
undefined
```

5. **Null**: Represents an intentional absence of any value.

```
let empty = null;
```

6. **Symbol**: Represents unique and immutable values, often used as object keys.

```
let sym = Symbol('unique');
```

7. **BigInt:** Represents integers larger than Number.MAX_SAFE_INTEGER.

```
let bigNumber = 123456789012345678901234567890n;
```

JAVASCRIPT DEVELOPMENT

VARIABLES AND DATATYPES

Non-Primitive Datatypes

Non-primitive types are objects and can store collections of data or more complex entities.

1. **Object**: Represents key-value pairs.

```
let obj = {
    name: "Amit",
    age: 25
};
```

2. **Array**: Represents an ordered list of values.

```
let a = ["red", "green", "blue"];
```

VARIABLES AND DATATYPES

3. **Function**: Represents reusable blocks of code.

```
function fun() {
 console.log("This is a function");
}
```

JAVASCRIPT DEVELOPMENT

CONSOLE.LOGO

The console.log() method is used to print messages to the browser’s developer console. Open the console (usually with F12 or Ctrl + Shift + J) to see the message “Hello, World!” displayed.

```
console.log("Hello, World!"); // Prints Hello, World! to the console
```

JAVASCRIPT DEVELOPMENT

JAVASCRIPT ARITHMETIC OPERATORS

These are the operators that work on numerical values and then return a number. These are basically used to perform mathematical operations.

Addition (+)
Add two numbers or concatenate the string.

Subtraction (-)
Difference between the two operators.

Multiplication (*)
Multiply two numbers.

Division (/)
Find the quotient of two operands.

Modulus (%)
Find the remainder of two operands.

Exponentiation (**)
Raise the Left operator to the power of the right operator.

Increment (++)
Increase the operand by one.

Decrement (--)
Decrease the operand by one.

JAVASCRIPT DEVELOPMENT

JAVASCRIPT COMPARISON OPERATORS

These are the operators that are used to perform equality or difference comparisons between the values. It checks whether an element is greater, smaller equal, or unequal to the other element.

Equality (==)
Compares the equality of two operators.

Inequality (!=)
Compares inequality of two operators.

Strict Equality (===)
Compares both value and type of the operand.

Strict Inequality (!==)
Compares inequality with type.

Greater than (>)
Checks if the left operator is greater than the right operator.

Greater than or equal (>=)
Checks if the left operator is greater than or equal to the right operator.

Less than (<)
Checks if the left operator is smaller than the right operator.

Less than or equal (<=)
Checks if the left operator is smaller than or equal to the right operator.

JAVASCRIPT LOGICAL OPERATORS

These are the operators which allow us to compare variables or values. The logical operator is mostly used to make decisions based on conditions specified for the statements. It can also be used to manipulate a boolean or set termination conditions for loops.

NOT (!)
Converts operator to boolean and returns flipped value.
AND (&&)
Evaluates operands and return true only if all are true.
OR (||)
Returns true even if one of the multiple operands is true.

JAVASCRIPT DEVELOPMENT

JAVASCRIPT ASSIGNMENT OPERATORS

These operators assign the value of the right-hand operand to its left-hand operand. That is if a = b assigns the value of b to a.

Addition Assignment (+=)
Add the value of the right operand to left and assign it to the left operand.

Subtraction Assignment (-=)
Subtracts the value of the right operand to left and assigns it to the left operand.

Multiplication Assignment (*=)
Multiplies the value of the right operand to left and assign it to the left operand.

Division Assignment (/=)
Divides the value of the right operand by the left and assigns it to the left operand.

Remainder Assignment (%=)
Finds the remainder after the division of the right operand with the left and assign it to the left operand.

Exponentiation Assignment (=)**
Raises left operand to power of right operand and assign it to the left operand.

CONTROL STATEMENTS IN JAVASCRIPT

JavaScript control statement is used to control the execution of a program based on a specific condition. If the condition meets then a particular block of action will be executed otherwise it will execute another block of action that satisfies that particular condition.

Types of Control Statements in JavaScript

- **Conditional Statement:** These statements are used for decision-making, a decision is made by the conditional statement based on an expression that is passed. Either YES or NO.

- **Iterative Statement:** This is a statement that iterates repeatedly until a condition is met. Simply said, if we have an expression, the statement will keep repeating itself until and unless it is satisfied.

IF STATEMENT

In this approach, we are using an if statement to check a specific condition, the code block gets executed when the given condition is satisfied.

```
if ( condition_is_given_here ) {
    // If the condition is met,
    //the code will get executed.
}
```

CONTROL STATEMENTS IN JAVASCRIPT

IF-ELSE STATEMENT

The if-else statement will perform some action for a specific condition. If the condition meets then a particular code of action will be executed otherwise it will execute another code of action that satisfies that particular condition.

```
if (condition1) {
    // Executes when condition1 is true
    if (condition2) {
        // Executes when condition2 is true
    }
}
```

SWITCH STATEMENT

The switch case statement in JavaScript is also used for decision-making purposes. In some cases, using the switch case statement is seen to be more convenient than if-else statements.

```
switch (expression) {
    case value1:
        statement1;
        break;
    case value2:
        statement2;
        break;
    default:
        statementDefault;
}
```

TERNARY OPERATOR (CONDITIONAL OPERATOR)

The conditional operator, also referred to as the ternary operator (?:), is a shortcut for expressing conditional statements in JavaScript.

```
condition ? value if true : value if false
```

FOR LOOP

In this approach, we are using for loop in which the execution of a set of instructions repeatedly until some condition evaluates and becomes false

```
for (statement 1; statement 2; statement 3) {
    // Code here . . .
}
```

```
for (let i = 0; i <= 10; i++) {
  if (i % 2 === 0) {
    console.log(i);
  }
};
```

CONTROL STATEMENTS IN JAVASCRIPT

WHILE LOOP

The_while loop repeats a block of code as long as the condition is true.

```
let i = 1;
while (i <= 5) {
    console.log(i);
    i++;
}
```