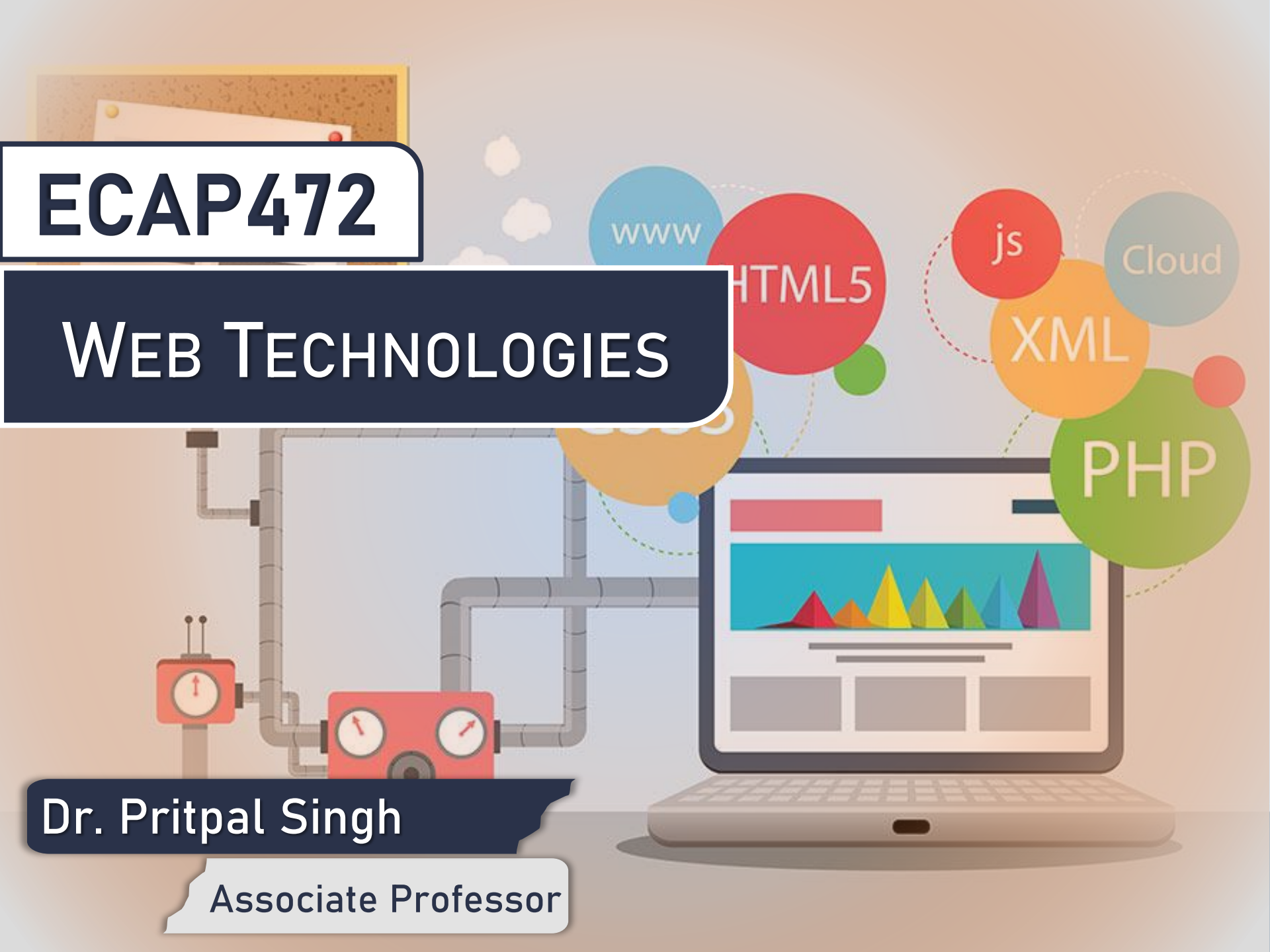
ECAP472

Web Technologies



Dr. Pritpal Singh

Associate Professor

Learning Outcomes

After this lecture, you will be able to

- Understand JavaScript variables.
- Understand JavaScript functions .

JavaScript Variables

4 Ways to Declare a JavaScript Variable:

- Using var
- Using let
- Using const
- Using nothing

What are Variables?

- Variables are containers for storing data (storing data values).
- In this example, x, y, and z, are variables, declared with the var keyword:
- Example
  - var x = 5;
  - var y = 6;
  - var z = x + y;

Variables

- In this example, x, y, and z, are variables, declared with the let keyword:
- Example
  - let x = 5;
  - let y = 6;
  - let z = x + y;

Variables

- In this example, x, y, and z, are undeclared variables:
- Example
  - x = 5;
  - y = 6;
  - z = x + y;

When to Use JavaScript var?

Always declare JavaScript variables with var, let, or const.

The var keyword is used in all JavaScript code from 1995 to 2015.

The let and const keywords were added to JavaScript in 2015.

If you want your code to run in older browser, you must use var.

When to Use JavaScript var?

Always declare JavaScript variables with var, let, or const.

The var keyword is used in all JavaScript code from 1995 to 2015.

The let and const keywords were added to JavaScript in 2015.

If you want your code to run in older browser, you must use var.

When to Use JavaScript var?

Always declare JavaScript variables with var, let, or const.

The var keyword is used in all JavaScript code from 1995 to 2015.

The let and const keywords were added to JavaScript in 2015.

If you want your code to run in older browser, you must use var.

When to Use JavaScript var?

Always declare JavaScript variables with var, let, or const.

The var keyword is used in all JavaScript code from 1995 to 2015.

The let and const keywords were added to JavaScript in 2015.

If you want your code to run in older browser, you must use var.

JavaScript Identifiers

- All JavaScript variables must be identified with unique names.
- These unique names are called identifiers.
- Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

JavaScript Identifiers

- The general rules for constructing names for variables (unique identifiers) are:
- Names can contain letters, digits, underscores, and dollar signs.
- Names must begin with a letter
- Names can also begin with $ and _ (but we will not use it in this tutorial)
- Names are case sensitive (y and Y are different variables)
- Reserved words (like JavaScript keywords) cannot be used as names

The Assignment Operator

- In JavaScript, the equal sign (=) is an "assignment" operator, not an "equal to" operator.
- This is different from algebra. The following does not make sense in algebra:
- x = x + 5

JavaScript Data Types

- JavaScript variables can hold numbers like 100 and text values like "John Doe".
- In programming, text values are called text strings.
- JavaScript can handle many types of data, but for now, just think of numbers and strings.
- Strings are written inside double or single quotes. Numbers are written without quotes.
- If you put a number in quotes, it will be treated as a text string.

One Statement, Many Variables

- You can declare many variables in one statement.
- Start the statement with let and separate the variables by comma:
- Example
- let person = "John Doe", carName = "Volvo", price = 200;

JavaScript Functions

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).

Example

```
function myFunction(p1, p2)
{
  return p1 * p2;   // The function returns the product of p1
and p2
}
```

JavaScript Function Syntax

- A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().
- Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
- The parentheses may include parameter names separated by commas:
- (parameter1, parameter2, ...)
- The code to be executed, by the function, is placed inside curly brackets: {}

Example

- function name(parameter1, parameter2, parameter3)

  {

  // code to be executed

  }

- Function parameters are listed inside the parentheses () in the function definition.
- Function arguments are the values received by the function when it is invoked.
- Inside the function, the arguments (the parameters) behave as local variables

Function Invocation

- he code inside the function will execute when "something" invokes (calls) the function:
- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

Function Return

- When JavaScript reaches a return statement, the function will stop executing.
- If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.
- Functions often compute a return value. The return value is "returned" back to the "caller":

Function Return

- Example
- Calculate the product of two numbers, and return the result:

- let x = myFunction(4, 3); // Function is called, return value will end up in x

- function myFunction(a, b) {
- return a * b; // Function returns the product of a and b
- }

Why Functions?

- You can reuse code: Define the code once, and use it many times.
- You can use the same code many times with different arguments, to produce different results.

JavaScript Events

- HTML events are "things" that happen to HTML elements.
- When JavaScript is used in HTML pages, JavaScript can "react" on these event

HTML Events

- An HTML event can be something the browser does, or something a user does.

- Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked
- Often, when events happen, you may want to do something.

- JavaScript lets you execute code when events are detected.

That’s all for now…