TechLift
KICKSTART YOUR CAREER

WEB ENGINEERING

.Net

P@SHA
Pakistan Software Houses Association for IT & ITES

PAKISTAN
PSEB
SOFTWARE EXPORT BOARD

Week 2 - Day 4

Learning Objectives

By the end of this session, the students will have reinforced to further developed an understanding of:

- Functions
- Data Types
- Operators and Arithmetic Operations
- Error Handling
- JQuery

: A Quick Review

This week is focused on everyone (students with technical/non technical background) starting from the same ground with HTML & JavaScript revisions.

Functions

Functions

- Fundamental building blocks in JavaScript
- Perform a set of statements
  - Calculate a value
  - Perform a task
- Define on the scope from where you want to call it

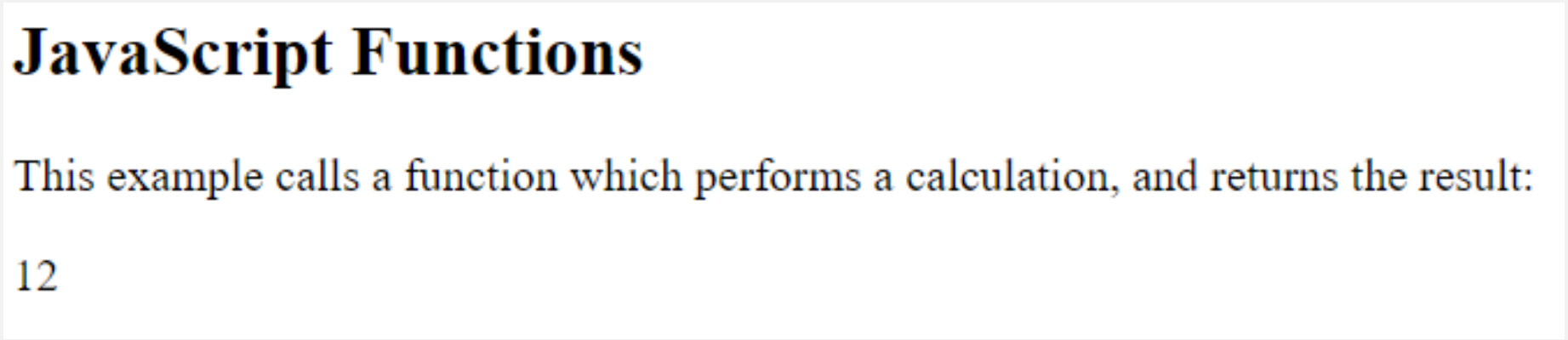
Functions

- Fundamental building blocks in JavaScript to perform a particular task
- Perform a set of statements
  - Calculate a value
  - Perform a task
- Define on the scope from where you want to call it

(A JavaScript function is executed when "something" invokes it (calls it).)

Functions - Example

Write the code to get the following output and then check the code from the link given below:



Check the output here.

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**: {}**

JavaScript Function Syntax

```
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.
- A Function is much the same as a Procedure or a Subroutine, in other programming languages.

Function Invocation

```
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.
- A Function is much the same as a Procedure or a Subroutine, in other programming languages.

Functions

Syntax

```
<script type = "text/javascript">

  function functionname(parameter-list)

{ statements }

 </script>
```

Example

```
<script type ="text/javascript">
 function sayHello()
      {alert("Hello there"); }
 </script>
```

Calling a Function

```
<html>  <head>
        <script type = "text/javascript">
                function sayHello()
                        {       document.write ("Hello there!");       }
        </script>   </head>
 <body>
 <p>Click the following button to call the function</p>
      <form>       <input type = "button" onclick = "sayHello()" value = "Say Hello">     </form>
  <p>Use different text in write method and then try...</p>
  </body>
</html>
```

The return Statement

This is required to return a value from a function. This statement should be the last statement in a function.

```
<html>
 <head>
      <script type = "text/javascript">
               function concatenate(first, last)
                    {
                              var full;
                              full = first + last;
                              return full;
                     }
              function secondFunction()
                    {         var result;
                              result = concatenate('snehal', 'smruti');
                              document.write (result );
                     }
         </script>
   </head>
 <body>
    <p>Click the following button to call the function</p>
       <form>
              <input type = "button" onclick = "secondFunction()" value = "Call Function">
        </form>

  </body>
</html>
```

Data Types

Data Types

There are two types of Data Types in JavaScript

– Primitive data type

– Non-primitive (reference) data type

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

Primitive data type

- String
- Number
- Boolean
- Null
- Undefined

Primitive data type (String)

A series of characters enclosed in quotation marks either single quotation marks (') or double quotation marks (")

Example :

```
var name = “Webstack Academy”;
var name = 'Webstack Academy';
```

Primitive Data Type (Numbers)

JavaScript has only one type of numbers.

Numbers can be written with, or without decimals:

Example

```
let x1 = 34.00;      // Written with decimals
let x2 = 34;         // Written without decimals
```

Execute the example and check the output here.

JavaScript Booleans

Booleans can only have two values: true or false. Booleans are often used in conditional testing.

Example

```
let x = 5;
let y = 5;
let z = 6;
(x == y)       // Returns true
(x == z)       // Returns false
```

Execute the example and check the output here.

JavaScript Arrays

- JavaScript arrays are written with square brackets.
- Array items are separated by commas.
- The following code declares (creates) an array called cars, containing three items (car names):

Example

```
const cars = ["Saab", "Volvo", "BMW"];
```

check the output here.

Array indexes are zero-based, which means the first item is [0], second is [1], and so on.

You will learn more about arrays later in this tutorial.

JavaScript Objects

- JavaScript objects are written with curly braces {}.
- Object properties are written as name:value pairs, separated by commas.

Example

```
const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

Check the output here.

The object (person) in the example above has 4 properties: firstName, lastName, age, and eyeColor.

Undefined

In JavaScript, a variable without a value, has the value undefined. The type is also undefined.

Example:

```
let car;    // Value is undefined, type is undefined
```

Check the output here.

Undefined

Any variable can be emptied, by setting the value to undefined. The type will also be undefined.

Example:

```
car = undefined;     // Value is undefined, type is undefined
```

Check the output here.

Operators

Operators

JavaScript supports the following types of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators

Let's have a look on all operators one by one.

Arithmetic Operators

JavaScript supports the following arithmetic operators −

Assume variable A holds 10 and variable B holds 20, then −

Sr.No.	Operator & Description
1	+ (Addition) Adds two operands Ex: A + B will give 30
2	- (Subtraction) Subtracts the second operand from the first Ex: A - B will give -10
3	* (Multiplication) Multiply both operands Ex: A * B will give 200
4	/ (Division) Divide the numerator by the denominator Ex: B / A will give 2

Arithmetic Operators

Assume variable A holds 10 and variable B holds 20, then −

Sr.No.	Operator & Description
5	% (Modulus) Outputs the remainder of an integer division Ex: B % A will give 0
6	++ (Increment) Increases an integer value by one Ex: A++ will give 11
7	-- (Decrement) Decreases an integer value by one Ex: A-- will give 9

Arithmetic Operators

Practice Task

Try writing the code given in the example here.
After writing it, check the output.

Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

Comparison Operators

Practice Task

Try writing the code given in the example here.
After writing it, check the output.

Logical Operators

The following operators are known as JavaScript logical operators.

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

Logical Operators

Practice Task

Try writing the code given in the example here.
After writing it, check the output.

Assignment Operators

The following operators are known as JavaScript assignment operators.

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30
-=	Subtract and assign	var a=20; a-=10; Now a = 10
*=	Multiply and assign	var a=10; a*=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0

Assignment Operators

Try writing the code given in the example here.
After writing it, check the output.

Practice Task

Errors & Exceptions Handling

Syntax Errors

Find out the reason that why does the following line causes a syntax error:

```
<script type = "text/javascript">
<!--
window.print(;
//-->
</script>
```

Runtime Errors

Find out the reason that why does the following line causes a runtime error

```
<script type = "text/javascript">
<!--
window.printme();
//-->
</script>
```

Logical Errors: The try...catch...finally Statement

1

Here is an example where we are trying to call a non-existing function which in turn is raising an exception. Let us see how it behaves without **try...catch−**

Try writing this code and then tally your output.

Check the output here.

```
<html>
   <head>
      <script type = "text/javascript">
         <!--
            function myFunc() {
               var a = 100;
               alert("Value of variable a is : " + a );
            }
         //-->
      </script>
   </head>

   <body>
      <p>Click the following to see the result:</p>

      <form>
         <input type = "button" value = "Click Me" onclick = "myFunc();" />
      </form>
   </body>
</html>
```

2

Now let us try to catch this exception using **try...catch** and display a user-friendly message. You can also suppress this message, if you want to hide this error from a user.

Try writing this code and then tally your output.

Check the output here.

```
<html>
   <head>

      <script type = "text/javascript">
         <!--
            function myFunc() {
               var a = 100;
               try {
                  alert("Value of variable a is : " + a );
               }
               catch ( e ) {
                  alert("Error: " + e.description );
               }
            }
         //-->
      </script>

   </head>
   <body>
      <p>Click the following to see the result:</p>

      <form>
         <input type = "button" value = "Click Me" onclick = "myFunc();" />
      </form>

   </body>
</html>
```

3

You can use finally block which will always execute unconditionally after the try/catch. Here is an example.

Try writing this code and then tally your output.

Check the output here.

```
<html>
   <head>

      <script type = "text/javascript">
         <!--
            function myFunc() {
               var a = 100;

               try {
                  alert("Value of variable a is : " + a );
               }
               catch ( e ) {
                  alert("Error: " + e.description );
               }
               finally {
                  alert("Finally block will always execute!" );
               }
            }
         //-->
      </script>

   </head>
   <body>
      <p>Click the following to see the result:</p>

      <form>
         <input type = "button" value = "Click Me" onclick = "myFunc();" />
      </form>

   </body>
</html>
```

The Throw Statement

- You can use throw statement to raise your built-in exceptions or your customized exceptions.
- Later these exceptions can be captured and you can take an appropriate action.

The Throw Statement

This example demonstrates how to use a throw statement.

Try writing this code and then tally your output.

Check the output here.

```
<html>
   <head>

      <script type = "text/javascript">
         <!--
            function myFunc() {
               var a = 100;
               var b = 0;

               try {
                  if ( b == 0 ) {
                     throw( "Divide by zero error." );
                  } else {
                     var c = a / b;
                  }
               }
               catch ( e ) {
                  alert("Error: " + e );
               }
            }
         //-->
      </script>

   </head>
   <body>
      <p>Click the following to see the result:</p>

      <form>
         <input type = "button" value = "Click Me" onclick = "myFunc();" />
      </form>

   </body>
</html>
```

The onerror() Method

- The onerror event handler was the first feature to facilitate error handling in JavaScript.
- The error event is fired on the window object whenever an exception occurs on the page.

The onerror() Method

Try writing this code and then tally your output.

Check the output here.

```
<html>
   <head>

      <script type = "text/javascript">
         <!--
            window.onerror = function () {
               alert("An error occurred.");
            }
         //-->
      </script>

   </head>
   <body>
      <p>Click the following to see the result:</p>

      <form>
         <input type = "button" value = "Click Me" onclick = "myFunc();" />
      </form>

   </body>
</html>
```

The onerror() Method

- The **onerror** event handler provides three pieces of information to identify the exact nature of the error −
- **Error message** − The same message that the browser would display for the given error
- **URL −** The file in which the error occurred
- **Line number**− The line number in the given URL that caused the error

The onerror() Method

Here is the example to show how to extract this information.

Try writing this code and then tally your output.

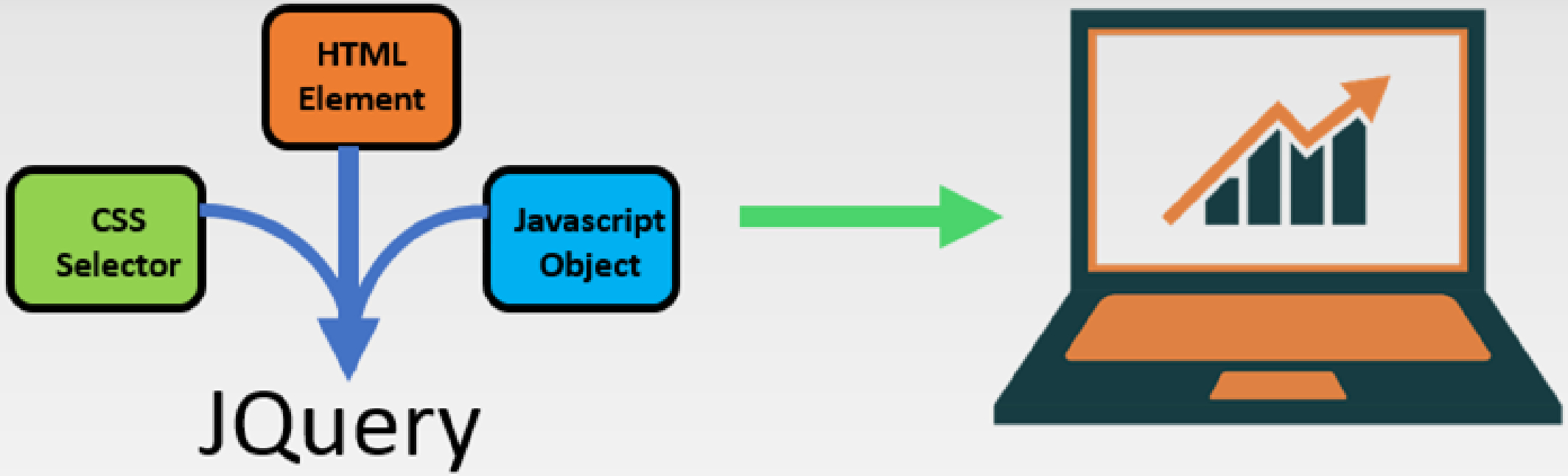
Check the output here.

```
<html>
   <head>

      <script type = "text/javascript">
         <!--
            window.onerror = function (msg, url, line) {
               alert("Message : " + msg );
               alert("url : " + url );
               alert("Line number : " + line );
            }
         //-->
      </script>

   </head>
   <body>
      <p>Click the following to see the result:</p>

      <form>
         <input type = "button" value = "Click Me" onclick = "myFunc();" />
      </form>

   </body>
</html>
```

: Basics

Install JQuery

JQuery will be installed in all the computers before starting its introduction.



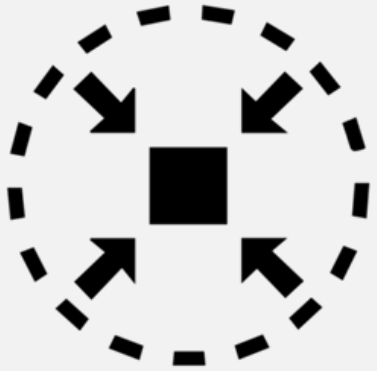
What is jQuery?

What is jQuery?

edureka!

Simplifies JavaScript

Event Handling



Lightweight

Animations

What is JQuery

- **Simplifies JavaScript:**

  It simplifies DOM manipulation and event handling for rapid web development

- **Event handling:**

  jQuery offers an effective manner to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code

- **Lightweight:**

  jQuery is a compact, lightweight library of about 19KB in size

- **Animations:**

  It comes with plenty of built-in animation effects which you can use in your web app to make it more interactive

What is it?

A library of JavaScript functions, just like...

- Prototype
- YUI
- Dojo
- mooTools

What is it?

- The jQuery library contains the following features:
  - HTML element selections
  - HTML element manipulation
  - CSS manipulation
  - HTML event functions
  - JavaScript effects and animations
  - HTML DOM traversal and modification
  - AJAX
  - Utilities
- An open source project, maintained by a group of developers with active support base and well written documentation.

jQuery philosophy

- Focus on the interaction between JavaScript and HTML
- (Almost) every operation boils down to:
  - Find some stuff
  - Do something to it


jQuery
Top Uses Of
jQuery

jQuery

- a fast, small JavaScript library included in a single .js file
- built-in functions
- a framework built using JavaScript capabilities.
- all the functions and other capabilities available in JavaScript can be used
- Any JavaScript editor can be used to write jQuery such as
  - Notepad
  - Visual Studio
  - Eclipse
  - Ultra edit

Advantages of jQuery

- **Easy to learn:** jQuery is easy to learn because it supports same JavaScript style coding.
- **Write less do more**: jQuery provides a rich set of features that increase developers' productivity by writing less and readable code.
- **Excellent API Documentation:** jQuery provides excellent online API documentation.
- **Cross-browser support:** jQuery provides excellent cross-browser support without writing extra code.
- **Unobtrusive: j**Query is unobtrusive which allows separation of concerns by separating html and jQuery code.

Features of jQuery

jQuery includes the following features:

- DOM element selections using the cross-browser open source selector engine Sizzle, a spinoff out of the jQuery project
- DOM traversal and modification (including support for CSS 1-3)
- DOM manipulation based on CSS selectors that uses node elements name and node elements
- attributes (id and class) as criteria to build selectors
- Events
- Effects and animations
- Ajax
- Extensibility through plug-ins
- Cross-browser support

Features of jQuery

- jQuery simplifies various tasks of a progammer by writing less code. Here is the list of important core features supported by jQuery −
- DOM manipulation − The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called Sizzle.
- Event handling − The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
- AJAX Support − The jQuery helps you a lot to develop a responsive and feature rich site using AJAX technology.

Features of jQuery

- Animations − The jQuery comes with plenty of built-in animation effects which you can use in your websites.
- Lightweight − The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).
- Cross Browser Support − The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
- Latest Technology − The jQuery supports CSS3 selectors and basic XPath syntax.

Setting up jQuery

There are two ways to use jQuery.

- **Local Installation −** You can download jQuery library on your local machine and include it in your HTML code.

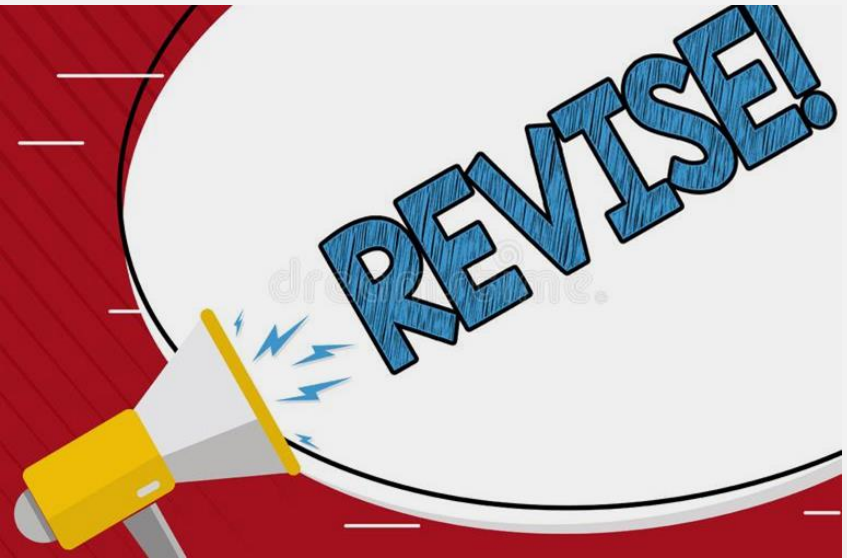
- **CDN Based Installation −** You can include jQuery library into your HTML code directly from Content Delivery Network (CDN).

Homework

1- Students will practice exercises of the topics covered today from this link:

JavaScript Exercises

2- They will develop a multiplication table with the help of this link.



They will come prepared for a marked quiz to be held in the next class.

Reinforcement Tutorial

This tutorial can be shared with the students for further practice and reinforcement: https://www.guru99.com/interactive-javascript-tutorials.html

Learning Objectives

By the end of this session, the students have practised

- Functions
- Data Types
- Operators and Arithmetic Operations
- Error Handling
- JQuery

Conclusion & Q/A

See you tomorrow!