

# **IT - 23 : ADVANCED INTERNET TECHNOLOGIES (2020 Pattern)**

(Semester - II)

Time : 2½ Hours]

[Max. Marks : 50 ]

Instructions to the candidates:

- 1) All questions are compulsory.
- 2) Draw neat and labeled diagram wherever necessary.

PA-2558 [Total No. of Pages : 4

Q1) Choose the correct option (½ mark each) [10]

a) Choose the item below that is not semantic element for text in HTML5.

- i) <wbr>
- ii) <time>
- iii) <mark>
- iv) <article>

Ans- I <wbr>

b) \_\_\_\_\_ is not a HTML5 tag

- i) <source>
- ii) video
- iii) <slider>
- iv) <canvas>

Ans-iii) slider tag

c) Which of the property in CSS used to change the background color of an element is \_\_\_\_\_

- i) Bgcolor
- ii) Color
- iii) Background-color
- iv) All of the above
- v) Ans-iii) Background-color

d) Which selector is used to selects the elements that are the default among a set of similar elements?

- i) : : after
  - ii) : disabled
  - iii) : default
  - iv) : checked
- Ans-iii) :default

e) Have to start Node REPL?

- i) \$ node start
  - ii) \$ node
  - iii) \$ node repl
  - iv) \$ node console
- Ans-ii) node

f) How to get an absolute path?

- i) Os.resolve ('main. js')
- ii) Path. resolve ('main. js')
- iii) fs. resolve ('main. js')
- iv) None of the above

Ans-

Iv None of the above

g) Which module is required for operating system specific operations?

- I) OS module
- II) fs module
- III) net module
- IV) None of the mentioned

Ans-i) OS module

h) Which command will show all the modules installed globally?

- i)\$ npm ls- g
- ii) \$ npm ls
- iii) \$ node ls- g
- iv)\$ node ls

Ans- i) )\$ npm ls- g

i) REPL is for

- I) Research Eval Program Learn
- II) Read Eval Print Loop
- III) Read Earn Point Learn
- IV) Read Eval Point Loop

Ans-II) Read Eval Print Loop

- j) What is the use of angular controllers in the application?
- I) angular controller are used for controlling the data
  - II) angular controller are used for displaying the data
  - III) Both of the above are correct
  - IV) None of the above

Ans-I) angular controller are used for controlling the data

- k) Which of the following is an advantages of AngularJS?
- i) AngularJS code is unit testable
  - ii) AngularJS provides reusable components
  - iii) AngularJS uses dependency injection and makes use of separation of concerns
  - iv) All of the mentioned

Ans- ii) AngularJS provides reusable components

- l) AngularJS is a\_\_\_\_\_framework
- i) HTML
  - ii) Java
  - iii) Javascript
  - iv) SQL

Ans- iii) Javascript

- m) On which of the architectural pattern AngularJS is based?
- i) Observer pattern
  - ii) Decorater patteren
  - iii) MVC architecture pattern
  - iv) MVVM architectural pattern

Ans- iii) MVC architecture pattern

- n) Select the directive which hide or shows the HTML element.

- i) ng- list
- ii) ng- hidi
- iii) ng- copy
- iv) ng- open

Ans- ii) ng-hide

o) In PHP array index starts with position\_\_\_\_\_

- i) 1
- ii) 0
- iii) 2
- iv) -1

Ans- ii) 0

p) \_\_\_\_\_is not a variable scope in PHP

- i) Extern
- ii) Local
- iii) Static
- iv) Global

Ans- i) Extern

q) What is the use of isset ( ) function in PHP?

- I) It is used to check whether variable is set or not
- II) It is used to check whether variable is free or not
- III) It is used to check whether variable is string or not
- IV) None of the mentioned
- V) Ans-I) It is used to check whether variable is set or not

r) Which of the following is the correct syntax to add comment in php?

- i) &-----&
- ii) // -----
- iii) / \* ----- \*/
- iv) both b & c

Ans- iv) both b & c

s) Which of the following is the correct way of defining a variable in PHP

- i) \$ Variable name = value;
  - ii) \$ Variable \_ name = value;
  - iii) \$ Variable = name = value;
  - iv) \$ Variable name as value;
- Ans- ii) \$ Variable \_ name = value;

t) Which of the following is the use of strlen( ) function in PHP?

- i) It returns the type of string
  - ii) It returns the length of string
  - iii) It returns the value of string
  - iv) It returns both value & type of string
- Ans- ii) It returns the length of string

Q2) a) What is Node.js module? Explain its types in brief. [5]

### Node.js Modules

In Node.js, **Modules** are the blocks of encapsulated code that communicate with an external application on the basis of their related functionality. Modules can be a single file or a collection of multiple files/folders. The reason programmers are heavily reliant on modules is because of their reusability as well as the ability to break down a complex piece of code into manageable chunks.

#### Modules are of three types:

- Core Modules
- local Modules
- Third-party Modules

**Core Modules:** Node.js has many built-in modules that are part of the platform and come with Node.js installation. These modules can be loaded into the program by using the **required** function.

#### Loading Core Modules

In order to use Node.js core or NPM modules, you first need to import it using require() function as shown below.

```
var module = require('module_name');
```

As per above syntax, specify the module name in the require() function. The require() function will return an object, function, property or any other JavaScript type, depending on what the specified module returns.

The following example demonstrates how to use Node.js http module to create a web server.

**Syntax:**

```
const module = require('module_name');
```

The require() function will return a JavaScript type depending on what the particular module returns. The following example demonstrates how to use the Node.js http module to create a web server.

Core Modules	Description
http	creates an HTTP server in Node.js.
assert	set of assertion functions useful for testing.
fs	used to handle file system.
path	includes methods to deal with file paths.
process	provides information and control about the current Node.js process.
os	provides information about the operating system.
querystring	utility used for parsing and formatting URL query strings.
url	module provides utilities for URL resolution and parsing.

**Local Modules:** Unlike built-in and external modules, local modules are created locally in your Node.js application. Let's create a simple calculating module that calculates various operations. Create a calc.js file that has the following code:

```
exports.add = function (x, y) {  
  return x + y;  
};
```

```
exports.sub = function (x, y) {  
  return x - y;  
};
```

```
exports.mult = function (x, y) {  
  return x * y;  
};
```

```
};
```

```
exports.div = function (x, y) {  
  return x / y;  
};
```

Since this file provides attributes to the outer world via exports, another file can use its exported functionality using the require() function.

**Filename: index.js**

```
const calculator = require('./calc');
```

```
let x = 50, y = 10;
```

```
console.log("Addition of 50 and 10 is "  
  + calculator.add(x, y));
```

```
console.log("Subtraction of 50 and 10 is "  
  + calculator.sub(x, y));
```

```
console.log("Multiplication of 50 and 10 is "  
  + calculator.mult(x, y));
```

```
console.log("Division of 50 and 10 is "  
  + calculator.div(x, y));
```

**Step to run this program:** Run the **index.js** file using the following command:

```
node index.js
```

### **Output:**

Addition of 50 and 10 is 60

Subtraction of 50 and 10 is 40

Multiplication of 50 and 10 is 500

Division of 50 and 10 is 5

**Note:** This module also hides functionality that is not needed outside of the module.

**Third-party modules:** Third-party modules are modules that are available online using the Node Package Manager(NPM). These modules can be installed in the project folder or globally. Some of the popular third-party modules are Mongoose, express, angular, and React.

### **Example:**

- npm install express
- npm install mongoose
- npm install -g @angular/cli

## Node.js Local Module

Local modules are modules created locally in your Node.js application. These modules include different functionalities of your application in separate files and folders. You can also package it and distribute it via NPM, so that Node.js community can use it. For example, if you need to connect to MongoDB and fetch data then you can create a module for it, which can be reused in your application.

### Writing Simple Module

Let's write simple logging module which logs the information, warning or error to the console.

In Node.js, module should be placed in a separate JavaScript file. So, create a Log.js file and write the following code in it.

```
var log = {
  info: function (info) {
    console.log('Info: ' + info);
  },
  warning: function (warning) {
    console.log('Warning: ' + warning);
  },
  error: function (error) {
    console.log('Error: ' + error);
  }
};
```

```
module.exports = log
```

### Log.js

In the above example of logging module, we have created an object with three functions - info(), warning() and error(). At the end, we have assigned this object to **module.exports**. The module.exports in the above example exposes a log object as a module.

The *module.exports* is a special object which is included in every JS file in the Node.js application by default. Use **module.exports** or **exports** to expose a function, object or variable as a module in Node.js.

### Loading Local Module

To use local modules in your application, you need to load it using require() function in the same way as core module. However, you need to specify the path of JavaScript file of the module.



The following example demonstrates how to use the above logging module contained in Log.js.

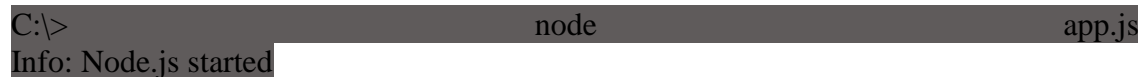
app.js

```
var myLogModule = require('./Log.js');  
  
myLogModule.info('Node.js started');
```

In the above example, app.js is using log module. First, it loads the logging module using require() function and specified path where logging module is stored. Logging module is contained in Log.js file in the root folder. So, we have specified the path './Log.js' in the require() function. The '.' denotes a root folder.

The require() function returns a log object because logging module exposes an object in Log.js using module.exports. So now you can use logging module as an object and call any of its function using dot notation e.g myLogModule.info() or myLogModule.warning() or myLogModule.error()

Run the above example using command prompt (in Windows) as shown below.



A screenshot of a Windows command prompt window. The title bar shows 'C:\>' on the left, 'node' in the center, and 'app.js' on the right. The command prompt itself shows the command 'node app.js' has been executed, and the output is 'Info: Node.js started'.

Thus, you can create a local module using module.exports and use it in your application.

### Export Module in Node.js

Here, you will learn how to expose different types as a module using module.exports.

The `module.exports` is a special object which is included in every JavaScript file in the Node.js application by default. The `module` is a variable that represents the current module, and `exports` is an object that will be exposed as a module. So, whatever you assign to `module.exports` will be exposed as a module.

## Export Literals

As mentioned above, `exports` is an object. So it exposes whatever you assigned to it as a module. For example, if you assign a string literal then it will expose that string literal as a module.

The following example exposes simple string message as a module in Message.js.

Message.js

```
module.exports = 'Hello world'
```

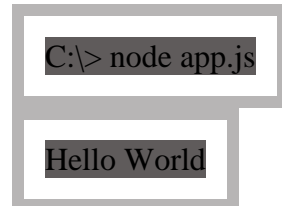
Now, import this message module and use it as shown below.

```
var msg = require('./Message.js');
```

```
console.log(msg);
```

```
app.js
```

Run the above example and see the result, as shown below.



#### Note:

You must specify `./` as a path of root folder to import a local module. However, you do not need to specify the path to import Node.js core modules or NPM modules in the `require()` function.

### Export Object

The `exports` is an object. So, you can attach properties or methods to it. The following example exposes an object with a string property in `Message.js` file.

```
Message.js
```

```
exports.SimpleMessage = 'Hello world';
```

```
//or
```

```
module.exports.SimpleMessage = 'Hello world';
```

In the above example, we have attached a property `SimpleMessage` to the `exports` object. Now, import and use this module, as shown below.

```
app.js
```

```
var msg = require('./Messages.js');
```

```
console.log(msg.SimpleMessage);
```

In the above example, the `require()` function will return an object `{ SimpleMessage : 'Hello World' }` and assign it to the `msg` variable. So, now you can use `msg.SimpleMessage`.

Run the above example by writing `node app.js` in the command prompt and see the output as shown below.

```
C:\> node app.js

Hello World
```

In the same way as above, you can expose an object with function. The following example exposes an object with the `log` function as a module.

`Log.js`

```
module.exports.log = function (msg) {  
  console.log(msg);  
};
```

The above module will expose an object- `{ log : function(msg){ console.log(msg); } }`. Use the above module as shown below.

`app.js`

```
var msg = require('./Log.js');
```

```
msg.log('Hello World');
```

Run and see the output in command prompt as shown below.

```
C:\> node app.js

Hello World
```

You can also attach an object to `module.exports`, as shown below.

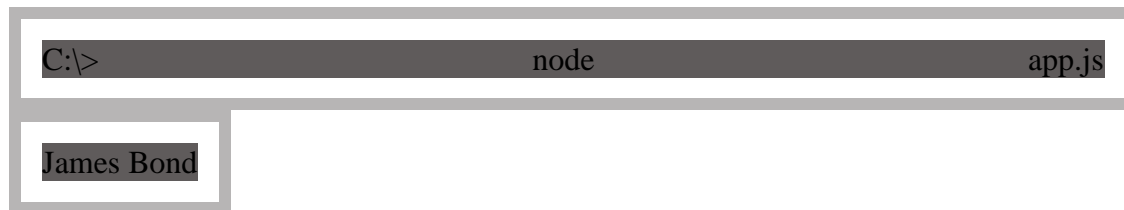
`data.js`

```
module.exports = {  
  firstName: 'James',  
  lastName: 'Bond'  
}
```

`app.js`

```
var person = require('./data.js');  
console.log(person.firstName + ' ' + person.lastName);
```

Run the above example and see the result, as shown below.



b) Write a program to demonstrate ngif. & ngswitch statements. [5]

Structural directives are responsible for the Structure and Layout of the DOM Element. It is used to hide or display the things on the DOM. Structural Directives can be easily identified using the '\*'. Every Structural Directive is preceded by a '\*' symbol. Some of the Build in Structural Directives with Examples are as follows:

## NgIf

---

The NgIf directive is used when you want to display or remove an element based on a condition.

If the condition is false the element the directive is *attached to* will be *removed* from the DOM.

### ***Important***

The difference between [hidden]='false' and \*ngIf='false' is that the first method simply *hides* the element. The second method with ngIf *removes* the element completely from the DOM.

We define the condition by passing an expression to the directive which is evaluated in the context of its host component.

ngIf is used to display or hide the DOM Element based on the expression value assigned to it. The expression value may be either true or false.

The syntax is: \*ngIf="<condition>"

```
<div *ngIf="boolean"> </div>
```

In the above Syntax, boolean stands for either true or false value. Hence, it leads to 2 valid syntaxes as below :

```
<div *ngIf="true"> </div>
```

```
<div *ngIf="false"> </div>
```

Let's use this in an example, we've taken the same code sample as we used for the NgFor lecture but changed it slightly. Each person now has an age as well as a name.

Let's add an NgIf directive to the template so we only show the element if the age is less than 30, like so:

## TypeScript

```
Copy@Component({
  selector: 'ngif-example',
  template: `
<h4>NgIf</h4>
<ul *ngFor="let person of people">
  <li *ngIf="person.age < 30"> (1)
    {{ person.name }} ({{ person.age }})
  </li>
</ul>
`
})
class NgIfExampleComponent {

  people: any[] = [
    {
      "name": "Douglas Pace",
      "age": 35
    },
    {
      "name": "McLeod Mueller",
      "age": 32
    },
    {
      "name": "Day Meyers",
      "age": 21
    }
  ]
}
```

```

    },
    {
      "name": "Aguirre Ellis",
      "age": 34
    },
    {
      "name": "Cook Tyson",
      "age": 32
    }
  ];
}

```

### Example of \*ngIf:

```

<div *ngIf="false">
  This text will be hidden
  <h1 [ngStyle]='{color:'#FF0000}'">
    Structural Directive Example
  </h1>
</div>
<div *ngIf="true">
  This text will be displayed
  <h1 [ngStyle]='{color:'#00FF00}'">
    Structural Directive Example
  </h1>
</div>

```

### Output:

Structural Directive Example

### 2. \*ngIf-else:

ngIf-else works like a simple If-else statement, wherein if the condition is true then 'If' DOM element is rendered, else the other DOM Element is rendered. Angular uses ng-template with element selector in order to display the else section on DOM.

### Syntax:

```

<div *ngIf="boolean; else id_selector"> </div>

<ng-template #id_selector> </ng-template>

```

In the above Syntax, boolean stands for either true or false value. If the boolean value is true then Element in If is rendered on the DOM, else another element is rendered on the DOM .

**Example of \*ngIf- else:**

```

<div *ngIf="false;else id_selector">
  This text will be hidden
  <h1 [ngStyle]='{color:'#FF0000}'">
    GFG Structural Directive
    If Part
  </h1>
</div>
<ng-template #id_selector>
  This text will be displayed
  <h1 [ngStyle]='{color:'#00FF00}'">
    Structural Directive
    Else Part
  </h1>
</ng-template>

```

**Output:**

Structural Directive

**3. \*ngFor:**

\*ngFor is used to loop through the dynamic lists in the DOM. Simply, it is used to build data presentation lists and tables in HTML DOM.

**Syntax:**

```
<div *ngFor="let item of item-list"> </div>
```

**Example of \*ngFor:**

Consider that you are having a list as shown below:

```
items = ["DYP 1", " DYP 2", " DYP 3", " DYP 4"];
```

```

<div *ngFor="let item of items">
  <p> {{item}} </p>
</div>

```

**OUTPUT**

DYP 1  
 DYP 2  
 DYP 3  
 DYP 4  
 DYP 5

**4. \*ngSwitch :**

ngSwitch is used to choose between multiple case statements defined by the expressions inside the \*ngSwitchCase and display on the DOM Element according to that. If no expression is matched, the default case DOM Element is displayed.

**Syntax:**

```

<div [ngSwitch]="expression">

  <div *ngSwitchCase="expression_1"></div>

  <div *ngSwitchCase="expression_2"></div>

```

```
<div *ngSwitchDefault></div>
</div>
```

In the above syntax, the expression is checked with each case and then the case matching with the expression is rendered on DOM else the Default case is rendered on the DOM.

```
<div [ngSwitch]="one">
  <div *ngSwitchCase="one">One is Displayed</div>
  <div *ngSwitchCase="two">Two is Displayed</div>
  <div *ngSwitchDefault>Default Option is Displayed</div>
</div>
```

In the above example, the expression 'one' in ngSwitch is matched to the expression in ngSwitchCase. Hence, the Element displayed on DOM is " One is Displayed ".

**Output:**

**One is Displayed**

OR a) What are different services in angular? [5]

AngularJS service is a function, which can use for the business layer of an application. It is like a constructor function that will invoke only once at runtime with new. Services in AngularJS are stateless and singleton objects because we get only one object in it regardless of which application interface created it.

We can use it to provide functionality in our web application. Each service performs a specific task.

### **When to use Services in AngularJS?**

We can use AngularJS services when we want to create things that act as an application interface. It can be used for all those purposes for which constructor is used.

Service is a piece of reusable code with a focused purpose. A code that you will use across multiple components in your application.

Our components need to access the data. You can write data access code in each Component, but this is very inefficient and breaks the rule of single responsibility. The Component should focus on presenting the data to the user.

The task of receiving data from the back-end server should be delegated to another class. We call a class a service class because it provides each Component with the data it needs.

### **What is Angular Services used for?**

Features independent of components such a logging services

- Share logic or data across components



- Encapsulate external interactions like data access
- Services are easier to test.
- They are easier to Debug.
- We can reuse the service at many places.

## How to create a Service in Angular

An Angular service is just a JavaScript function. All we have to do is create a class and add methods and properties. Then we can create an instance of this class in our Component and call its methods.

One of the best uses of services is to get data from a data source. Let us create a simple service, which receives product data and sends it to our Component.

## Product Model

Please create a new file under the folder src/app and call it product.ts

### product.ts

```
export class Product {
  constructor(productID:number,  name: string ,  price:number) {
    this.productID=productID;
    this.name=name;
    this.price=price;
  }
  productID:number ;
  name: string ;
  price:number;
}
```

The Product class above is our domain model.

## Product Angular Service

Next, we create an Angular service, which returns a list of products.

Please create a new file under the src/app folder and call it product.service.ts.**product.service.ts**

```

import {Product} from './product'
export class ProductService{
  public getProducts() {
    let products:Product[];
    products=[
      new Product(1,'Memory Card',500),
      new Product(1,'Pen Drive',750),
      new Product(1,'Power Bank',100)
    ]
    return products;
  }
}

```

First, we set the Product model to product.ts. import from

Next, create a ProductService class and export it. We need to export so that components and other service classes import it and use it

The Get Products method returns a collection of products. In this example, we have hardcoded the products. In real life, you would send an HTTP GET request to your back end API to get the data service is ready.

Note that the above class is a simple JavaScript function. There is nothing angular about it.

### Invoking the ProductService

The Next step is to invoke the ProductService from the Component. Open the app.component.ts and add the following code.

#### app.component.ts

```

import { Component } from '@angular/core';
import { ProductService } from './product.service';
import { Product } from './product';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
})
export class AppComponent

```

```

{
  products:Product[];
  productService;
  constructor(){
    this.productService=new ProductService();
  }
  getProducts() {
    this.products=this.productService.getProducts();
  }
}

```

### **We start with importing Product & ProductService**

We create an instance of ProductService in the AppComponent's constructor. In real-life Angular apps, we use dependency injection in Angular to inject ProductService into the constructor. We will learn this in the next tutorial.

The getProducts method calls the getProducts method of the ProductService. It returns a list of products, which we store in local variable products.

b) Write a program to read the query string using url property in Node js[5]

The **Query String** module used to provides utilities for parsing and formatting URL query strings.It can be used to convert query string into JSON object and vice-versa. The Query String is the part of the URL that starts after the question mark(?).

**Requiring Module:** You can include the module using the following code:

```
const querystring = require('querystring');
```

**Note:** It's not a global object, so need to install it explicitly.

**Install Module:**

```
npm install querystring
```

**Example 1:** Using **parse()**:

We will now look into the below six methods in the next section.

- querystring.decode()
- querystring.encode()
- querystring.escape(str)
- querystring.parse(str[, sep[, eq[, options]]])
- querystring.stringify(obj[, sep[, eq[, options]]])
- querystring.unescape(str)

## Query String methods with description

Let us look into a real example to understand the important Query string methods.

Let us setup a basic Node application by giving the command `npm init -y` in terminal, inside a folder. I had created an empty NodeJS folder for the same.

### **querystring.parse() Method**

The `querystring.parse()` method is used to parse the URL query string into an object that contains the key value pair. The object which we get is not a JavaScript object, so we cannot use Object methods like `obj.toString`, or `obj.hasOwnProperty()`.

The latest UTF-8 encoding format is assumed unless we specify a different encoding format. But we should stick to UTF-8 encoding as it is the standard and contains all international characters like the Chinese characters and the hindi characters. After that also if we need alternative character encoding, then the `decodeURIComponent` option should be used.

The syntax for the method is below.

As seen from the above syntax the method accepts four parameters, and they are described below.

- **str:** This is the only required string field and it specifies the query string that has to be parsed.
- **sep:** It is an optional string field, which if given specifies the substring used to delimit the key and value pairs in the query string. The default value which is generally used is “&”.
- **eq:** It is an optional string field that specifies the substring used to delimit keys and values in the query string. The default value which is generally used is “=”.
- **options:** It is an optional object field which is used to modify the behaviour of the method. It can have the following parameters:
  - **decodeURIComponent:** It is a function that would be used to specify the encoding format in the query string. The default value is `querystring.unescape()`, about which we will learn later.
  - **maxKeys:** It is the number which specifies the maximum number of keys that should be parsed. A value of “0” would remove all the

counting limits, and it can parse any number of keys. The default value is set at “1000”.

The below example shows the various options used in `querystring.parse()` method.

Add the below code in `querystring.js` file, which we created earlier

Now, run the command `node querystring.js` from the Integrated terminal in VSCode or any terminal. Note that you need to be inside the folder NodeJS, which we had created earlier. The output of the same will be below.

### **querystring.stringify() Method**

The `querystring.stringify()` method is used to produce a query string from a given object, which contains a key value pair. It is exactly the opposite of `querystring.parse()` Method.

It can be used to convert the string, numbers and Boolean values for the key. You can also use an array of string, numbers or Boolean as values. This method of changing an object to query string is called serialized.

The latest UTF-8 encoding format is assumed unless we specify a different encoding format. But we should stick to UTF-8 encoding as it is the standard and contains all international characters like the Chinese characters and the Hindi characters. If we still need an alternative character encoding, then the **decodeURIComponent** option should be used.

Syntax for the method is below.

As from the above syntax the method accepts four parameters, and they are described below.

- **obj:** This is the only required object field and it specifies the object that has to be serialized.
- **sep:** It is an optional string field, which if given specifies the substring used to delimit the key and value pairs in the query string. The default value which is generally used is “&”.

- **eq:** It is an optional string field that specifies the substring used to delimit keys and values in the query string. The default value which is generally used is “=”.
- **options:** It is an optional object field which is used to modify the behaviour of the method. It can have the following parameters:
  - **decodeURIComponent:** It is a function that would be used to specify the encoding format in the query string. The default value is `querystring.escape()`, about which we will learn later.

The below example shows the various options used in `querystring.stringify()` method. Add the below code in `querystring.js` file, which we created earlier.  
`queryString);`

Now, run the command `node querystring.js` from the Integrated terminal in VSCode or any terminal. Note that you need to be inside the folder NodeJS, which we had created earlier. The output of the same will be below.

### Query String

**1:** `name=nabendu&access=true&role=developer&role=architect&role=manager`

**Query String 2:** `name:Parag, access:false, role:editor, role:HR`

**Query String 3:** `name==Parag&&&access==false&&&role==editor&&&role==HR`

### `querystring.decode()` Method

The `querystring.decode()` method is nothing but an alias for `querystring.parse()` method. In our parse example, we can use it. So, add the below code in `querystring.js` file, which we created earlier

As earlier, run the command `node querystring.js` from a terminal. And the output will be same as that with `querystring.parse()` method.: `'false' }`

### `querystring.encode()` Method

The `querystring.encode()` method is nothing but an alias for `querystring.stringify()` method. In our stringify example, we can use it. So, add the below code in `querystring.js` file, which we created earlier.

As earlier run the command `node querystring.js` from a terminal. And the output will be same as that with `querystring.stringify()` method.

### **querystring.escape(str) Method**

The `querystring.escape()` method is used by `querystring.stringify()` method and is generally not used directly.

### **querystring.unescape(str) Method**

The `querystring.unescape()` method is used by `querystring.parse()` method and is generally not used directly.

#### **Example : Using `stringify()`:**

```
// Importing the model
import querystring from 'querystring'

// Specify the object
// to be serialized
const q2=querystring.stringify({
    name:'Testing',
    company:'GeeksforGeeks',
    content:'Article',
    date:'9thMarch2021'
});

// Print the result.
console.log(q2);
```

#### **Output:**

```
name=Testing&company=GeeksforGeeks&
content=Article&date=9thMarch2021
```

Q3) a) What is NPM? How packages installed locally and globally? [5]

Node Package Manager provides two main functionalities:

- It provides online repositories for node.js packages/modules which are searchable on [search.npmjs.org](https://search.npmjs.org)

- It also provides command line utility to install Node.js packages, do version management and dependency management of Node.js packages.

The npm comes bundled with Node.js installables in versions after that v0.6.3. You can check the version by opening Node.js command prompt and typing the following command:

1. npm version

### **Installing Modules using npm**

Following is the syntax to install any Node.js module:

1. npm install <**Module** Name>

Let's install a famous Node.js web framework called express:

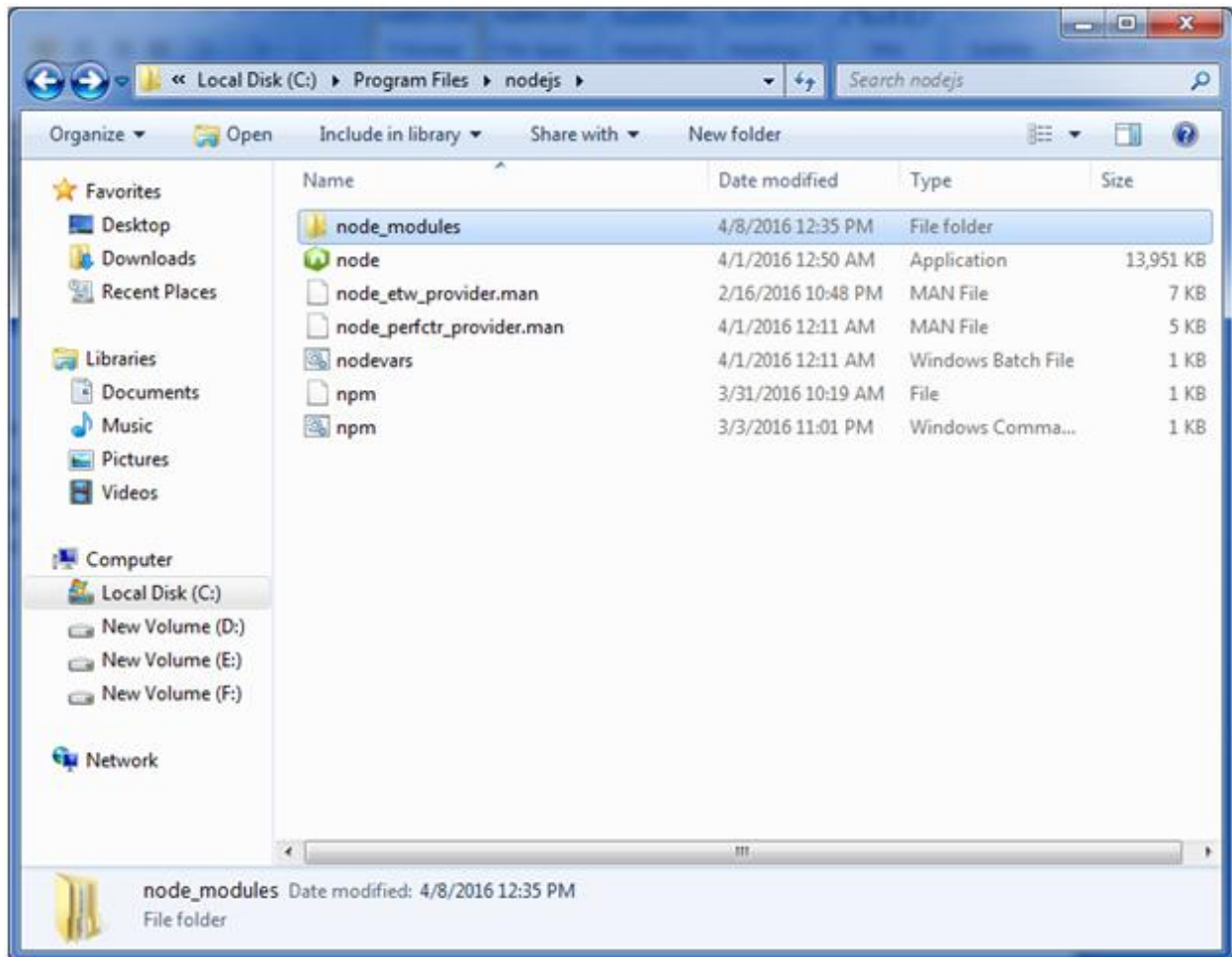
Open the Node.js command prompt and execute the following command:

1. npm install express

### **Global vs Local Installation**

By default, npm installs dependency in local mode. Here local mode specifies the folder where Node application is present. For example if you installed express module, it created node\_modules directory in the current directory where it installed express module.





You can use npm ls command to list down all the locally installed modules.

Open the Node.js command prompt and execute "npm ls":

Globally installed packages/dependencies are stored in system directory. Let's install express module using global installation. Although it will also produce the same result but modules will be installed globally.

Open Node.js command prompt and execute the following code:

1. npm install express -g

## Uninstalling a Module

To uninstall a Node.js module, use the following command:

```
npm uninstall express
```

The Node.js module is uninstalled. You can verify by using the following command:

```
npm ls
```

## Searching a Module

"npm search express" command is used to search express or module.

```
npm search express
```

The main difference between local and global packages is this:

1. **local packages** are installed in the directory where you run `npm install <package-name>`, and they are put in the `node_modules` folder under this directory
2. **global packages** are all put in a single place in your system (exactly where depends on your setup), regardless of where you run `npm install -g <package-name>`

b) Create angular program which will demonstrate the usage of component directive.  
[5]

The answer is in your practical file.

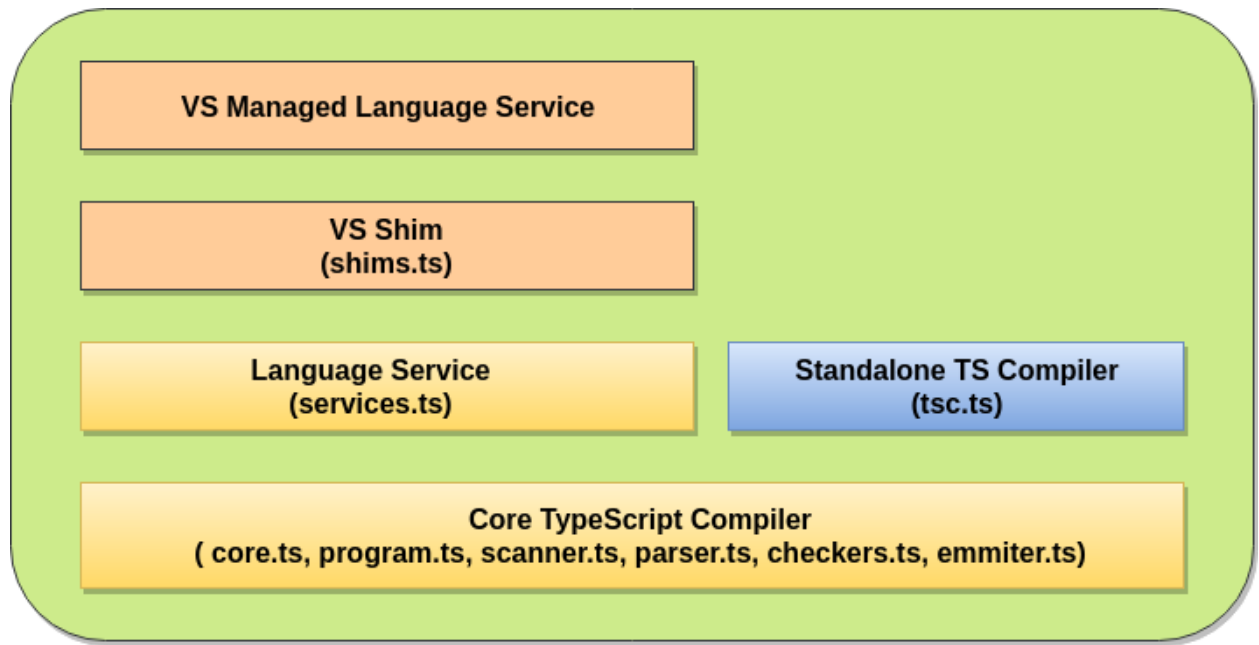
OR

a) What are the Typescript components? [5]

Components of TypeScript

The TypeScript language is internally divided into three main layers. Each of these layers is divided into sublayers or components. In the following diagram, we can see the three layers and each of their internal components. These layers are:

1. Language
2. The TypeScript Compiler
3. The TypeScript Language Services



**Components of TypeScript**

## 1. Language

It features the TypeScript language elements. It comprises elements like syntax, keywords, and type annotations.

## 2. The TypeScript Compiler

The TypeScript compiler (TSC) transform the TypeScript program equivalent to its JavaScript code. It also performs the parsing, and type checking of our TypeScript code to JavaScript code.

Browser doesn't support the execution of TypeScript code directly. So the program written in TypeScript must be re-written in JavaScript equivalent code which supports the execution of code in the browser directly. To perform this, TypeScript comes with TypeScript compiler named "tsc." The current version of TypeScript compiler supports ES6, by default. It compiles the source code in any module like ES6, SystemJS, AMD, etc.

We can install the TypeScript compiler by locally, globally, or both with any **npm** package. Once installation completes, we can compile the TypeScript file by running "tsc" command on the command line.

### Example:

1. `$ tsc helloworld.ts // It compiles the TS file helloworld into the helloworld.js file.`

### Compiler Configuration

The TypeScript compiler configuration is given in **tsconfig.json** file and looks like the following:

```
{
  "compilerOptions": {
    "declaration": true,
    "emitDecoratorMetadata": false,
    "experimentalDecorators": false,
    "module": "none",
    "moduleResolution": "node",
    "noFallthroughCasesInSwitch": false,
    "noImplicitAny": false,
    "noImplicitReturns": false,
    "removeComments": false,
    "sourceMap": false,
    "strictNullChecks": false,
    "target": "es3"
  },
  "compileOnSave": true
}
```

### Declaration file

When we compile the TypeScript source code, it gives an option to generate a **declaration file** with the extension **.d.ts**. This file works as an interface to the components in the compiled JavaScript. If a file has an extension **.d.ts**, then each root level definition must have the **declare** keyword prefixed to it. It makes clear that there will be no code emitted by TypeScript, which ensures that the declared item will exist at runtime. The declaration file provides IntelliSense for JavaScript libraries like jQuery.

### 3. The TypeScript Language Services

The language service provides information which helps editors and other tools to give better assistance features such as automated refactoring and IntelliSense. It exposes an additional layer around the core-compiler pipeline. It supports some standard typical

editor operations like code formatting and outlining, colorization, statement completion, signature help, etc.

- b) Write a program using NPM which will convert entered string into lower case & upper case.

Q4) a) Explain and tags in HTML5 with suitable examples. [4]

### HTML5 <audio> Tag

Since the release of HTML5, audios can be added to webpages using the “audio” tag. Previously, audios could be only played on web pages using web plugins like Flash. The “audio” tag is an inline element that is used to embed sound files into a web page. It is a useful tag if you want to add audio such as songs, interviews, etc. on your webpage.

#### Syntax:

<audio>

<source src="sample.mp3" type="audio/mpeg">

</audio>

**Attributes:** The various attributes that can be used with the “audio” tag are listed below:

- **Controls:** Designates what controls to display with the audio player.
- **Autoplay:** Designates that the audio file will play immediately after it loads controls.
- **Loop:** Designates that the audio file should continuously repeat.
- **src:** Designates the URL of the audio file.
- **muted:** Designates that the audio file should be muted.

**Supported Formats:** Three formats mp3, ogg, and wav are supported by HTML5.

The support for each format by different browsers is given below :

Browser	MP3	WAV	OGG
Google Chrome	Yes	Yes	Yes
Internet Explorer	Yes	No	No
Firefox	Yes	Yes	Yes
Opera	Yes	Yes	Yes
Safari	Yes	Yes	No

The below Examples explain the audio Tag:

**Example 1 (Adding audio to Webpage):** The controls attribute is used to add audio controls such as play, pause, and volume. The “source” element is used to specify the audio files which the browser may use. The first recognized format is used by the browser.

<!DOCTYPE html>

<html>

```
<body>
  <p>Audio Sample</p>

  <!-- audio tag starts here -->
  <audio controls>
    <source src="test.mp3" type="audio/mp3">
    <source src="test.ogg" type="audio/ogg">
  </audio>
  <!-- audio tag ends here -->

</body>
</html>
output
```

Audio Sample



**Example 2 (Autoplaying audio on a Webpage):** The autoplay attribute is used to automatically begin playback of the audio file whenever the URL of the webpage is loaded.

```
<!DOCTYPE html>
<html>

<body>
  <p>Audio Sample</p>

  <!-- audio tag starts here -->
  <audio controls autoplay>
    <source src="test.mp3" type="audio/mp3">
    <source src="test.ogg" type="audio/ogg">
  </audio>
  <!-- audio tag ends here -->
</body>
</html>
```

**Example 3 (Adding muted audio file on a Webpage):** The muted attribute specifies that the audio should be muted on the webpage.

```
<!DOCTYPE html>
<html>
```

```

<body>
  <p>Audio Sample</p>

  <!-- audio tag starts here -->
  <audio controls muted>
    <source src="test.mp3" type="audio/mp3">
    <source src="test.ogg" type="audio/ogg">
  </audio>
  <!-- audio tag ends here -->
</body>
</html>

```

**Example 4 (Adding audio using the source element):** The source element can be used to add audio files to a webpage. The src attribute is used to specify the source of the file specified.

```

<!DOCTYPE html>
<html>

<body>
  <p>Audio Sample</p>

  <!-- audio tag starts here -->
  <audio controls autoplay>
    <source src="test.mp3" type="audio/mp3">
  </audio>
  <!-- audio tag ends here -->

</body>
</html>

```

**Example 5 (Adding audio with multiple sources):** Multiple sources of audios are specified so that if the browser is unable to play the first source, then it will automatically jump to the second source and try to play it.

```

<!DOCTYPE html>
<html>

<body>
  <p>Audio Sample</p>

  <!-- audio tag starts here -->
  <audio controls autoplay>
    <source src="test.mp3" type="audio/mp3">
    <source src="test.ogg" type="audio/ogg">
    <source src="test.opus" type="audio/ogg">
  </audio>

```

```
<!-- audio tag ends here -->
```

```
</body>
```

```
</html>
```

**Example 6 (Adding audio using the “Embed” tag):** Adding audios to a webpage using the “embed” tag is an old technique. This method does work, but is comparatively less efficient than the other methods. The user must have a plugin like MIDI or QuickTime because the embed tag requires a plugin for support.

```
!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p>Audio Sample</p>
```

```
<!-- embed code starts here -->
```

```
<embed src="test.mp3" width="200" height="50"
      autoplay="true" loop="true">
```

```
<!-- embed code ends here -->
```

```
</body>
```

```
</html>
```

b) Write a PHP script to demonstrate variables in PHP. Use Globals, local & global keyword to access variables. [4]

c) Differentiate between include & require [2]

**PHP require() Function:** The **require()** function in PHP is basically used to include the contents/code/data of one PHP file to another PHP file. During this process if there are any kind of errors then this **require()** function will pop up a warning along with a fatal error and it will immediately stop the execution of the script. In order to use this **require()** function, we will first need to create two PHP files. Using the **include()** function, include one PHP file into another one. After that, you will see two PHP files combined into one HTML file.

**Example 1:**

```
<html>
```

```
<body>
```

```
<h1>Welcome to DYPh1>
```

```
<p>Myself, Gaurav Gandar</p>
```

```
<p>Thank you</p>
```

```
<?php require 'GFG.php'; ?>
```

```
</body>
```

```
</html>
```



GFG.php

```
<?php

    echo "

    <p>visit Again-" . date("Y") . " geeks for geeks.com</p>

    ";

?>
```

**PHP include() Function:** The **include()** function in PHP is basically used to include the contents/code/data of one PHP file to another PHP file. During this process if there are any kind of errors then this **include()** function will pop up a warning but unlike the **require()** function, it will not stop the execution of the script rather the script will continue its process. In order to use this **include()** function, we will first need to create two PHP files. Using the **include()** function, include one PHP file into another one. After that, you will see two PHP files combined into one HTML file.

**Example 2:**

```
<html>
<body>
    <h1>Welcome to geeks for geeks!</h1>
    <p>Myself, Gaurav Gandal</p>

    <p>Thank you</p>

    <?php include 'GFG.php'; ?>
</body>
</html>
GFG.php
```

```
<?php

    echo "

    <p>Visit Again; " . date("Y") . " Geeks for geeks.com</p>
```

",";

?>

### Difference between require() and include():

include()	require()
The <b>include()</b> function does not stop the execution of the script even if any error occurs.	The <b>require()</b> function will stop the execution of the script when an error occurs.
The <b>include()</b> function does not give a fatal error.	The <b>require()</b> function gives a fatal error
The <b>include()</b> function is mostly used when the file is not required and the application should continue to execute its process when the file is not found.	The <b>require()</b> function is mostly used when the file is mandatory for the application.
The <b>include()</b> function will only produce a warning ( <u>E_WARNING</u> ) and the script will continue to execute.	The <b>require()</b> will produce a fatal error ( <u>E_COMPILE_ERROR</u> ) along with the warning.

OR

a) What are semantic elements and how it works in HTML5? [4]

### What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements: <div> and <span> - Tells nothing about its content.

Examples of **semantic** elements: <form>, <table>, and <article> - Clearly defines its content.

## Semantic Elements in HTML

Many web sites contain HTML code like: `<div id="nav">` `<div class="header">` `<div id="footer">` to indicate navigation, header, and footer.

In HTML there are some semantic elements that can be used to define different parts of a web page:

- `<article>`
- `<aside>`
- `<details>`
- `<figcaption>`
- `<figure>`
- `<footer>`
- `<header>`
- `<main>`
- `<mark>`
- `<nav>`
- `<section>`
- `<summary>`
- `<time>`

---

## HTML `<section>` Element

The `<section>` element defines a section in a document.

According to W3C's HTML documentation: "A section is a thematic grouping of content, typically with a heading."

Examples of where a `<section>` element can be used:

- Chapters
- Introduction
- News items
- Contact information

A web page could normally be split into sections for introduction, content, and contact information.

### Example

Two sections in a document:

```
<section>
<h1>WWF</h1>
<p>The World Wide Fund for Nature (WWF) is an international organization working
on issues regarding the conservation, research and restoration of the environment,
formerly named the World Wildlife Fund. WWF was founded in 1961.</p>
</section>

<section>
<h1>WWF's Panda symbol</h1>
<p>The Panda has become the symbol of WWF. The well-known panda logo of WWF
originated from a panda named Chi Chi that was transferred from the Beijing Zoo to the
London Zoo in the same year of the establishment of WWF.</p>
</section>
```

## HTML `<article>` Element

The `<article>` element specifies independent, self-contained content.

An article should make sense on its own, and it should be possible to distribute it independently from the rest of the web site.

Examples of where the `<article>` element can be used:

- Forum posts
- Blog posts
- User comments
- Product cards
- Newspaper articles

### Example

Three articles with independent, self-contained content:

```
<article>
<h2>Google Chrome</h2>
<p>Google Chrome is a web browser developed by Google, released in 2008. Chrome
is the world's most popular web browser today!</p>
</article>

<article>
<h2>Mozilla Firefox</h2>
```

```
<p>Mozilla Firefox is an open-source web browser developed by Mozilla. Firefox has
been the second most popular web browser since January, 2018.</p>
</article>
```

```
<article>
<h2>Microsoft Edge</h2>
<p>Microsoft Edge is a web browser developed by Microsoft, released in 2015.
Microsoft Edge replaced Internet Explorer.</p>
</article>
```

## Example 2

Use CSS to style the <article> element:

```
<html>
<head>
<style>
.all-browsers {
  margin: 0;
  padding: 5px;
  background-color: lightgray;
}

.all-browsers > h1, .browser {
  margin: 10px;
  padding: 5px;
}

.browser {
  background: white;
}

.browser > h2, p {
  margin: 4px;
  font-size: 90%;
}
</style>
</head>
<body>

<article class="all-browsers">
  <h1>Most Popular Browsers</h1>
```

```
<article class="browser">
  <h2>Google Chrome</h2>
  <p>Google Chrome is a web browser developed by Google, released in 2008.
Chrome is the world's most popular web browser today!</p>
</article>
<article class="browser">
  <h2>Mozilla Firefox</h2>
  <p>Mozilla Firefox is an open-source web browser developed by Mozilla. Firefox
has been the second most popular web browser since January, 2018.</p>
</article>
<article class="browser">
  <h2>Microsoft Edge</h2>
  <p>Microsoft Edge is a web browser developed by Microsoft, released in 2015.
Microsoft Edge replaced Internet Explorer.</p>
</article>
</article>

</body>
</html>
```

---

### **Nesting <article> in <section> or Vice Versa?**

The <article> element specifies independent, self-contained content.

The <section> element defines section in a document.

Can we use the definitions to decide how to nest those elements? No, we cannot!

So, you will find HTML pages with <section> elements containing <article> elements, and <article> elements containing <section> elements.

### **HTML <header> Element**

The <header> element represents a container for introductory content or a set of navigational links.

A <header> element typically contains:

- one or more heading elements (<h1> - <h6>)
- logo or icon
- authorship information

**Note:** You can have several <header> elements in one HTML document. However, <header> cannot be placed within a <footer>, <address> or another <header> element.

### Example

A header for an <article>:

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

### HTML <footer> Element

The <footer> element defines a footer for a document or section.

A <footer> element typically contains:

- authorship information
- copyright information
- contact information
- sitemap
- back to top links
- related documents

You can have several <footer> elements in one document.

### Example

A footer section in a document:

```
<footer>
  <p>Author: Hege Refsnes</p>
```

```
<p><a href="mailto:hege@example.com">hege@example.com</a></p>
</footer>
```

## HTML <nav> Element

The <nav> element defines a set of navigation links.

Notice that NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major blocks of navigation links.

Browsers, such as screen readers for disabled users, can use this element to determine whether to omit the initial rendering of this content.

### Example

A set of navigation links:

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

## HTML <aside> Element

The <aside> element defines some content aside from the content it is placed in (like a sidebar).

The <aside> content should be indirectly related to the surrounding content.

### Example

Display some content aside from the content it is placed in:

```
<p>My family and I visited The Epcot center this summer. The weather was nice, and
Epcot was amazing! I had a great summer together with my family!</p>
```



```
<aside>
<h4>Epcot Center</h4>
<p>Epcot is a theme park at Walt Disney World Resort featuring exciting attractions,
international pavilions, award-winning fireworks and seasonal special events.</p>
</aside>
```

## Example 2

Use CSS to style the <aside> element:

```
<html>
<head>
<style>
aside {
  width: 30%;
  padding-left: 15px;
  margin-left: 15px;
  float: right;
  font-style: italic;
  background-color: lightgray;
}
</style>
</head>
<body>

<p>My family and I visited The Epcot center this summer. The weather was nice, and
Epcot was amazing! I had a great summer together with my family!</p>

<aside>
<p>The Epcot center is a theme park at Walt Disney World Resort featuring exciting
attractions, international pavilions, award-winning fireworks and seasonal special
events.</p>
</aside>

<p>My family and I visited The Epcot center this summer. The weather was nice, and
Epcot was amazing! I had a great summer together with my family!</p>
<p>My family and I visited The Epcot center this summer. The weather was nice, and
Epcot was amazing! I had a great summer together with my family!</p>

</body>
</html>
```

---

## HTML <figure> and <figcaption> Elements

The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

The <figcaption> tag defines a caption for a <figure> element.

The <figcaption> element can be placed as the first or as the last child of a <figure> element.

The <img> element defines the actual image/illustration.

### Example

```
<figure>
  
  <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
</figure>
```

## Why Semantic Elements?

According to the W3C: "A semantic Web allows data to be shared and reused across applications, enterprises, and communities."

## Semantic Elements in HTML

Below is a list of some of the semantic elements in HTML.

Tag	Description
<article>	Defines independent, self-contained content

<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element

<time>

Defines a date/time

For a complete list of all available HTML tags, visit our [HTML Tag Reference](#).

b) Write a PHP script to demonstrate any 4 operators[4]

### PHP Operators

PHP Operator is a symbol i.e used to perform operations on operands. In simple words, operators are used to perform operations on variables or values. For example:

1. \$num=10+20;//+ is the operator and 10,20 are operands

In the above example, + is the binary + operator, 10 and 20 are operands and \$num is variable.

PHP Operators can be categorized in following forms:

- Arithmetic Operators
- Assignment Operators
- Bitwise Operators
- Comparison Operators
- Incrementing/Decrementing Operators
- Logical Operators
- String Operators
- Array Operators
- Type Operators
- Execution Operators
- Error Control Operators

We can also categorize operators on behalf of operands. They can be categorized in 3 forms:

- **Unary Operators:** works on single operands such as ++, -- etc.
- **Binary Operators:** works on two operands such as binary +, -, \*, / etc.

- **Ternary Operators:** works on three operands such as "?:".

---

## Arithmetic Operators

The PHP arithmetic operators are used to perform common arithmetic operations such as addition, subtraction, etc. with numeric values.

Operator	Name	Example	Explanation
+	Addition	$\$a + \$b$	Sum of operands
-	Subtraction	$\$a - \$b$	Difference of operands
*	Multiplication	$\$a * \$b$	Product of operands
/	Division	$\$a / \$b$	Quotient of operands
%	Modulus	$\$a \% \$b$	Remainder of operands
**	Exponentiation	$\$a ** \$b$	$\$a$ raised to the power $\$b$

The exponentiation (\*\*) operator has been introduced in PHP 5.6.

---

## Assignment Operators

The assignment operators are used to assign value to different variables. The basic assignment operator is "=".

Operator	Name	Example	Explanation
=	Assign	$\$a = \$b$	The value of right operand is assigned to the left operand
+=	Add then Assign	$\$a += \$b$	Addition same as $\$a = \$a + \$b$

-=	Subtract then Assign	\$a -= \$b	Subtraction same as \$a = \$a - \$b
*=	Multiply then Assign	\$a *= \$b	Multiplication same as \$a = \$a * \$b
/=	Divide then Assign (quotient)	\$a /= \$b	Find quotient same as \$a = \$a / \$b
%=	Divide then Assign (remainder)	\$a %= \$b	Find remainder same as \$a = \$a % \$b

## Bitwise Operators

The bitwise operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
&	And	\$a & \$b	Bits that are 1 in both \$a and \$b are set to 1, otherwise 0.
	Or (Inclusive or)	\$a   \$b	Bits that are 1 in either \$a or \$b are set to 1
^	Xor (Exclusive or)	\$a ^ \$b	Bits that are 1 in either \$a or \$b are set to 0.
~	Not	~\$a	Bits that are 1 set to 0 and bits that are 0 are set to 1
<<	Shift left	\$a << \$b	Left shift the bits of operand \$a \$b steps
>>	Shift right	\$a >> \$b	Right shift the bits of \$a operand by \$b number of places

## Comparison Operators

Comparison operators allow comparing two values, such as number or string. Below the list of comparison operators are given:

Operator	Name	Example	Explanation
==	Equal	\$a == \$b	Return TRUE if \$a is equal to \$b
===	Identical	\$a === \$b	Return TRUE if \$a is equal to \$b, and they are of same data type
!==	Not identical	\$a !== \$b	Return TRUE if \$a is not equal to \$b, and they are not of same data type
!=	Not equal	\$a != \$b	Return TRUE if \$a is not equal to \$b
<>	Not equal	\$a <> \$b	Return TRUE if \$a is not equal to \$b
<	Less than	\$a < \$b	Return TRUE if \$a is less than \$b
>	Greater than	\$a > \$b	Return TRUE if \$a is greater than \$b
<=	Less than or equal to	\$a <= \$b	Return TRUE if \$a is less than or equal \$b
>=	Greater than or equal to	\$a >= \$b	Return TRUE if \$a is greater than or equal \$b
<=>	Spaceship	\$a <=> \$b	Return -1 if \$a is less than \$b Return 0 if \$a is equal \$b Return 1 if \$a is greater than \$b

## Incrementing/Decrementing Operators

The increment and decrement operators are used to increase and decrease the value of a variable.

Operator	Name	Example	Explanation
----------	------	---------	-------------

++	Increment	++\$a	Increment the value of \$a by one, then return \$a
		\$a++	Return \$a, then increment the value of \$a by one
--	decrement	--\$a	Decrement the value of \$a by one, then return \$a
		\$a--	Return \$a, then decrement the value of \$a by one

## Logical Operators

The logical operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
and	And	\$a and \$b	Return TRUE if both \$a and \$b are true
Or	Or	\$a or \$b	Return TRUE if either \$a or \$b is true
xor	Xor	\$a xor \$b	Return TRUE if either \$a or \$b is true but not both
!	Not	! \$a	Return TRUE if \$a is not true
&&	And	\$a && \$b	Return TRUE if either \$a and \$b are true
	Or	\$a    \$b	Return TRUE if either \$a or \$b is true

## String Operators

The string operators are used to perform the operation on strings. There are two string operators in PHP, which are given below:



Operator	Name	Example	Explanation
.	Concatenation	\$a . \$b	Concatenate both \$a and \$b
.=	Concatenation and Assignment	\$a .= \$b	First concatenate \$a and \$b, then assign the concatenated string to \$a, e.g. \$a = \$a . \$b

## Array Operators

The array operators are used in case of array. Basically, these operators are used to compare the values of arrays.

Operator	Name	Example	Explanation
+	Union	\$a + \$y	Union of \$a and \$b
==	Equality	\$a == \$b	Return TRUE if \$a and \$b have same key/value pair
!=	Inequality	\$a != \$b	Return TRUE if \$a is not equal to \$b
===	Identity	\$a === \$b	Return TRUE if \$a and \$b have same key/value pair of same type in same order
!==	Non-Identity	\$a !== \$b	Return TRUE if \$a is not identical to \$b
<>	Inequality	\$a <> \$b	Return TRUE if \$a is not equal to \$b

## Type Operators

The type operator **instanceof** is used to determine whether an object, its parent and its derived class are the same type or not. Basically, this operator determines which certain class the object belongs to. It is used in object-oriented programming.

```
//class declaration
class Developer
{
}
class Programmer
{
}
//creating an object of type Developer
$charu = new Developer();

//testing the type of object
if( $charu instanceof Developer)
{
    echo "Charu is a developer.";
}
else
{
    echo "Charu is a programmer.";
}
echo "</br>";
var_dump($charu instanceof Developer);    //It will return true.
var_dump($charu instanceof Programmer);    //It will return false.
?>
```

### Output:

```
Charu is a developer.
bool(true) bool(false)
```

### Execution Operators

PHP has an execution operator **backticks** (`). PHP executes the content of backticks as a shell command. Execution operator and **shell\_exec()** give the same result.

Operator	Name	Example	Explanation
----------	------	---------	-------------

<code>`</code>	backticks	<code>echo `dir`;</code>	Execute the shell command and return the result. Here, it will show the directories available in current folder.
----------------	-----------	--------------------------	--

**Note:** Note that backticks (```) are not single-quotes.

## Error Control Operators

PHP has one error control operator, i.e., **at (@) symbol**. Whenever it is used with an expression, any error message will be ignored that might be generated by that expression.

Operator	Name	Example	Explanation
@	at	@file ('non_existent_file')	Intentional file error

## PHP Operators Precedence

Let's see the precedence of PHP operators with associativity.

Operators	Additional Information	Associativity
clone new	clone and new	non-associative
[	array()	left
**	arithmetic	right
++ -- ~ (int) (float) (string) (array) (object) (bool) @	increment/decrement and types	right
instanceof	types	non-associative

!	logical (negation)	right
* / %	arithmetic	left
+ - .	arithmetic and string concatenation	left
<< >>	bitwise (shift)	left
< <= > >=	comparison	non-associative
== != === !== <>	comparison	non-associative
&	bitwise AND	left
^	bitwise XOR	left
	bitwise OR	left
&&	logical AND	left
	logical OR	left
?:	ternary	left
= += -= *= **= /= .= %= &=  = ^= <<= >>= =>	assignment	right
and	logical	left
xor	logical	left
or	logical	left

,	many uses (comma)	left
---	-------------------	------

c) What is Var- dump( ) function in PHP? [2]

## Definition and Usage

The var\_dump() function dumps information about one or more variables. The information holds type and value of the variable(s).

## Syntax

```
var_dump(var1, var2, ...);
```

## Parameter Values

Parameter	Description
<i>var1</i> , <i>var2</i> , ...	Required. Specifies the variable(s) to dump information from

## Technical Details

<b>Return Value:</b>	Nothing
<b>Return Type:</b>	-
<b>PHP Version:</b>	4.0+

## Example

```
<?php
$a = 32;
echo var_dump($a) . "<br>";
```

```
$b = "Hello world!";  
echo var_dump($b) . "<br>";
```

```
$c = 32.5;  
echo var_dump($c) . "<br>";
```

```
$d = array("red", "green", "blue");  
echo var_dump($d) . "<br>";
```

```
$e = array(32, "Hello world!", 32.5, array("red", "green", "blue"));  
echo var_dump($e) . "<br>";
```

```
// Dump two variables  
echo var_dump($a, $b) . "<br>";  
?>
```

Q5) a) What are selectors in CSS? Explain with suitable example. [4]

**CSS selectors** are used *to select the content you want to style*. Selectors are the part of CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

## 1) CSS Element Selector

The element selector selects the HTML element by name.

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
p{  
    text-align: center;  
    color: blue;  
}
```

```
</style>
</head>
<body>
<p>This style will be applied on every paragraph.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
</body>
</html>
```

Output:

This style will be applied on every paragraph.

Me too!

And me!

---

## 2) CSS Id Selector

The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element.

It is written with the hash character (#), followed by the id of the element.

Let's take an example with the id "para1".

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
    text-align: center;
    color: blue;
}
</style>
</head>
<body>
```

```
<p id="para1">Hello DYP.com</p>
<p>This paragraph will not be affected.</p>
</body>
</html>
```

Output:

Hello DYP.com

This paragraph will not be affected.

### 3) CSS Class Selector

The class selector selects HTML elements with a specific class attribute. It is used with a period character . (full stop symbol) followed by the class name.

*Note: A class name should not be started with a number.*

Let's take an example with a class "center".

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
  text-align: center;
  color: blue;
}
</style>
</head>
<body>
<h1 class="center">This heading is blue and center-aligned.</h1>
<p class="center">This paragraph is blue and center-aligned.</p>
</body>
</html>
```

Output:



This heading is blue and center-aligned.

This paragraph is blue and center-aligned.

### CSS Class Selector for specific element

If you want to specify that only one specific HTML element should be affected then you should use the element name with class selector.

Let's see an example.

```
<!DOCTYPE html>
<html>
<head>
<style>
p.center {
  text-align: center;
  color: blue;
}
</style>
</head>
<body>
<h1 class="center">This heading is not affected</h1>
<p class="center">This paragraph is blue and center-aligned.</p>
</body>
</html>
```

Output:

This heading is not affected

This paragraph is blue and center-aligned.

#### 4) CSS Universal Selector

The universal selector is used as a wildcard character. It selects all the elements on the pages.

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  color: green;
  font-size: 20px;
}
</style>
</head>
<body>
<h2>This is heading</h2>
<p>This style will be applied on every paragraph.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
</body>
</html>
```

Output:

This is heading

This style will be applied on every paragraph.

Me too!

And me!

---

#### 5) CSS Group Selector

The grouping selector is used to select all the elements with the same style definitions.

Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping.

Let's see the CSS code without group selector.

```
h1 {  
    text-align: center;  
    color: blue;  
}  
h2 {  
    text-align: center;  
    color: blue;  
}  
p {  
    text-align: center;  
    color: blue;  
}
```

As you can see, you need to define CSS properties for all the elements. It can be grouped in following ways:

```
h1,h2,p {  
    text-align: center;  
    color: blue;  
}
```

Let's see the full example of CSS group selector.

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
h1, h2, p {  
    text-align: center;  
    color: blue;  
}  
</style>
```

```
</head>
<body>
<h1>Hello DYP.com</h1>
<h2>Hello DYP.com (In smaller font)</h2>
<p>This is a paragraph.</p>
</body>
</html>
```

- c) Write a PHP script to display employees who belong to the IT department and whose salary is between 30,000 – 80,000 and store found records into another table. [6]  
(Assume suitable table structure)

OR

- a) What is a CSS framework? What are the different sorts of CSS frameworks? [4]

CSS frameworks are tools that UI designers use to make their work easier. Instead of starting from scratch, frameworks allow developers to swiftly create user interfaces that can be changed and iterated throughout a project. The basic purpose of the CSS framework is to bring a more standardized practice for web development. They're helpful for larger projects, bigger teams, and those who need to design a theme that can be used on multiple projects.

CSS framework is a set of prepared and ready-to-use CSS stylesheets. They're designed for typical tasks like putting up navbars and are frequently enhanced with other technologies like SASS and JavaScript. CSS frameworks provide more than just time savings. Frameworks deliver standards to teams with multiple developers, especially larger ones. Frameworks standardize layouts and allow one developer to readily comprehend another developer's code, rather than each developer writing the code by himself from the beginning for each project, making their work hectic and tedious. This saves time and ensures a smoother development cycle with fewer defects and improved team communication.

## CSS Framework

There are numerous CSS frameworks available, and all of them are incredibly beneficial. Though it is difficult to select a good framework from the several available options, here are a few common and popular frameworks you can choose from.

## 1. Bootstrap

Bootstrap has a very nice responsive design, due to which it is regarded as one of the best CSS frameworks. It was created by Twitter and first launched in 2011.

Bootstrap is a powerful front-end framework for building modern web pages and web applications. It is open-source and free to use. However, it comes with many HTML and CSS templates for UI elements like buttons and forms. Bootstrap also supports JavaScript extensions. Bootstrap uses SASS variables and mixins for theming, a responsive grid system for layout, pre-built components for design patterns, and JS plugins for user interaction.

### *Advantages of Bootstrap*

- It is a fully responsive framework; responsiveness is a major reason many developers prefer Bootstrap over its competitors.
- Bootstrap has a strong developer community and is one of the best-documented frameworks available.
- Bootstrap is a grid-based CSS framework that creates a multi-column layout with predefined pixel widths, allowing us to focus on content creation rather than text alignment.
- Bootstrap has a well-maintained and well-tested codebase; developers utilizing it will seldom run into browser compatibility difficulties.
- LESS and SASS are two of the most common preprocessors used in Bootstrap's code.
- It is a good framework in terms of usage and popularity, and it saves a lot of time. It has a relatively short learning curve.

### *Disadvantages of Bootstrap*

- The web pages or front-ends created with Bootstrap seem very similar, and Bootstrap's flexibility is limited.
- Due to the vast number of modules loaded in Bootstrap, CSS projects have a large build size. We can't pick and choose whatever components we want; we will have to change the codebase at our own risk to get rid of them.

### *Bootstrap 4 vs. Bootstrap 3*

Bootstrap 4 is the most recent version of the Bootstrap framework. It became the first choice for developers because it has new SASS variables for global styling and easy theming, bringing new dimensions for developers in building projects.

Except for Internet Explorer 9 and below, Bootstrap 4 is compatible with all major browsers. To support these older browsers, tests are being performed so that these features can be easily implemented in more devices.

***Kickstart Your UI/UX Career Right Here!***

UI/UX Design Expert **EXPLORE PROGRAM**

## **2. Tailwind CSS**

Tailwind CSS is a utility-first CSS framework that differs from other top CSS frameworks like Bulma and Bootstrap. Tailwind CSS provides pre-designed components that may be used as a foundation for subsequent development. It does not come with a pre-made design, but it allows us to include our own personal style rapidly.

### ***Advantages of Tailwind CSS***

- It allows developers to implement unique designs.
- With Tailwind CSS, we can use libraries like Purge CSS to reduce the CSS build size substantially.
- It works at a lower level by giving a developer a set of CSS helper classes.
- Tailwind is designed to be component friendly and encourages breaking down into smaller reusable components, so it has a lower cognitive load.

### ***Disadvantages of Tailwind***

- Tailwind alone makes it challenging to create complex animations.
- Because there are so many classes for styling, readability can be a problem for certain developers.

## **3. Foundation**

Another excellent CSS framework is Foundation. HTML, CSS, SASS, and Javascript are part of this powerful front-end CSS framework. It is best suited for large web applications that require a design host. It was built with a mobile-first approach and is very responsive. It is often called “The most advanced responsive front-end framework in the world.”

### *Advantages of Foundation*

- Foundation is designed to provide front-end developers complete control over their user interfaces.
- Easy to understand and readable code.
- Foundation is made up of lightweight, scalable components with minimal design and easy customization.

### *Disadvantages of Foundation*

- Because Foundation features rely on Javascript, it is incompatible with React and Angular applications.
- It has several features and is more complex than other frameworks. It provides a lot of versatility when it comes to developing front-end templates.

## **4. Bulma**

Bulma is another great flexbox-based CSS framework. It's an open-source CSS library that is 100 percent responsive and comes with several pre-configured components. Bulma creates Metro-style grids with a method called tiles, which makes the website look sleek and smooth. It has a lightweight framework that allows us to import only the elements we need for our current web design.

### *Advantages of Bulma*

- Most browsers are compatible with Bulma-designed websites. This makes developers' work easier, particularly when conducting cross-browser testing.
- It is a framework independent of the environment and sits on top of the logic layer.
- The syntax of Bulma is quite simple and easy to use.

### *Disadvantages of Bulma*

- There isn't a large developer community for Bulma.
- On the Internet Explorer web browser, it performs sluggishly.

***Here's How to Land a Top Software Developer Job***

**Full Stack Development-MEANEXPLORE PROGRAM**

## **How to Choose the Right CSS Framework for a Project?**

Frameworks for CSS are fantastic. They help us to get off to a good start when building a new site by providing structured code that ensures our site is responsive, solid, and practical with no work.

In the modern era, with increasing digitalization, everyone wants to ensure their presence on the internet, which has led to the development of many websites with a lot of pages. Using a pre-built framework or structure can help us with the day-to-day labor of website architecture. So while choosing a framework, we must keep certain points in consideration, as discussed in the next section:

### **What Type of Framework Do We Need?**

To begin, we must have some knowledge of our website. Is it even necessary to have a system? Structures are beneficial to most locations with a large number of pages. The ideal amount of pages can change, but if we find ourselves repeating the same HTML, CSS, or even JavaScript on one of our pages, a layout or structure can help.

- **Framework Language**

We should be familiar with the computer language(s) that our framework employs. As previously stated, some simple frameworks are nothing more than basic HTML templates, while more complicated frameworks may include CSS and JavaScript. Some frameworks construct the CSS using LESS or SASS, while others utilize Ruby or other computer languages to compile the pages once they're built. It will be challenging to build our framework if we are unfamiliar with the language(s) it employs.

- **Framework Features**

A CSS framework, often known as a web framework, is a collection of web-based tools, libraries, and best practices. A CSS framework can be as simple as a one-page template that serves as the foundation for all of our site's pages, or it can be a complicated collection of CSS, HTML, JavaScript, and server-side programs and files that govern the entire site architecture.

- **Whether the Framework is Customizable and Modular**



Customizable frameworks allow us to add our own code to make our site look unique and different from other websites. However, if a framework is overly customized, its benefits are lost, and it will be better to start from scratch.

### **Advantages of CSS Framework**

- Speeds up our development.
- Enables cross-border functionality.
- Gives us clean and symmetrical layouts.

### **Disadvantages of CSS Framework**

- The CSS framework has a standard set of grids, selectors, and other code. It limits what we can design.
- If we're already comfortable and effective with one method of designing and programming and are compelled to use a CSS framework we're unfamiliar with, we may end up wasting time.
- We will inevitably find code in a CSS framework that we don't need. It's quite improbable that we'll use all of a framework's features. As a result, we're stuck with superfluous code.

b) Write a PHP script & insert at least 5 records into it & update specific record in database. Assume student table with required fields in database. [6]  
You already solved this question in practical lab.