Advance PHP

Q. What Is Object Oriented Programming?

Ans. **OOP** stands for Object-Oriented Programming.
Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions .

Advantages of OOP:-

Object-oriented programming has several advantages over procedural programming:

1. OOP is faster and easier to execute.
2. OOP provides a clear structure for the programs.
3. OOP provide securities.
4. OOP helps to keep the PHP code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug.
5. OOP makes it possible to create full reusable applications with less code and shorter development time

Q. What Are Properties Of Object Oriented Systems?

Ans. Properties of object oriented programming are:

1. Class
2. Object
3. Encapsulation
4. Inheritance
5. Access Modifier
6. Polymorphism

Class:-
Making group of data member (variable) and member function that called class
A class is defined by using the class keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods go inside the braces.

Object:-
Classes are nothing without objects! We can create multiple objects from a class. Each object has all the properties and methods defined in the class, but they will have different property values.
Objects of a class are created using the new keyword.
Object is also known as the instance of the class.

Encapsulation:-
OOPs concept of Encapsulation in PHP means, enclosing the internal details of the object to protect from external sources. It describes, combining the class, data variables and member function that work on data together within a single unit to form an object. Otherwise, it's the bundling of properties and behavior in a single class unit.
Data is not accessed directly, in fact, they are accessed through functions (GET or SET) written inside the class. Attributes are kept private but getter (GET) and setter (SET) methods are kept public for manipulation of these attributes.

Inheritance:-
It is a concept of accessing the features of one class from another class.
We can reuse our code by using Inheritance.
Types:

1. Single-level inheritance
2. Multi-level inheritance
3. Multiple inheritance
4. Hierarchical Inheritance
5. Hybrid Inheritance

Access Modifier:-
Properties and methods can have access modifiers which control where they can be accessed.

Types:

1. Public
2. Private
3. Protected

Polymorphism:-

This word is can from Greek word poly and morphism.
Poly means "many" and morphism means “property” which helps us to assign more than one property.
There are two ways to achieve polymorphism in PHP:

1. Overloading
2. Overriding

Note: PHP does not support Overloading

Q. What Is Difference Between Class And Interface?

Ans.

1. “Class” keyword is used to create class, while “interface” keyword is used to create interface in PHP.
2. “Extends” keyword is used to inherit a class into another class, while “implements” keyword is used to inherit interface into a class.
3. In PHP, class does not support multiple inheritance and interface supports multiple inheritance.

Q. What Is Overloading?

Ans.

Function overloading in PHP allows creating multiple functions with the same name but different implementations, using magic methods to dynamically create properties and methods within a class.
Note: - We can’t overload method in PHP.
Function signature are based on their names and do not include argument lists, so you can’t have two functions with the same name.

Q. What are the differences between abstract classes and interfaces?

Ans.

1. Interface can't have properties, while Abstract class can.
2. All interface functions must be pubic, while abstract class functions is public or protected.
3. Abstract class can have abstract functions as well as non-abstract function, while interface can only have abstract method. All methods in an interface are abstract, so they can't be implemented in code and the abstract keyword is not necessary.
4. Abstract class does not support multiple inheritance, while interface supports multiple inheritance.
5. Abstract keyword is used to create abstract class, while interface keyword is used to create.
6. Abstract class can inherit other class and interface, while interface can inherit only interface.
7. Extends keyword is used to inherit abstract class, while implements keyword is used to inherit interface.

Q. Define Constructor and Destructor?

Ans.

Constructor: -

A constructor allows you to initialize an object's properties upon creation of the object. Also called magic function.

If you create a __construct() function, PHP will automatically call this function when you create an object from a class.

Notice that the construct function starts with two underscores (__).

Example of constructor:-

```
<?php
 class abc
{
        function simple()
        {
                echo "Simple Function <br>";
        }

        function __construct()
        {
                echo "Magic function run automatically <br>";
        }

        function autocall()
{

                $this->simple();  // normal function call in function with this
                                                                    keywords
                abc::__construct(); // call by ::(scope resolution)
        }

}

$obj=new abc;
$obj->autocall();

?>
```


Destructor:-

In PHP, destructor method is named as __destruct.
During shutdown sequence too, objects will be destroyed.
Destructor method doesn't take any arguments, neither does it return any data type.

Example of destructor:-

```
<?php
class a
{

   public function __construct()
   {
      echo "I'm alive! <br>";
        }

   public function __destruct()// object() destroy & call in last
   {
      echo "I'm dead now <br>";
   }
        public function display()
   {
      echo "I'm display now <br>";
   }
}

$a = new a();
$a->display();

?>
```


Q. How to Load Classes in PHP?

Ans.

PHP load classes are used for declaring its object etc. in object oriented applications. PHP parser loads it automatically, if it is registered with spl_auto_register() function. PHP parse gets the least chance to load class/interface before emitting an error.

Syntax:

```
spl_automated_register(function ($class_name){
 include $class_name '.php';
});
```

Q. How to Call Parent Constructor?

Ans.

Case 1:

We can't run directly the parent class constructor in child class if the child class defines a constructor. In order to run a parent constructor, a call to parent::__construct() within the child constructor is required.

Example:-

```
<?php
        class my_constructor
         {
                public function __construct()
                {
                   echo   "Hello Sir I'm Constructor";
                }
         }
        class abc extends my_constructor
        {
                public function __construct()
                {
                        patrent::__construct();    //or my_constructor::__construct();
                        echo "I am not constructor sir";
                }
        }
        $obj = new abc();
?>
```

Case 2:

If the child does not define a constructor then it may be inherited from the parent class just like a normal class method (if it was not declared as private).

Example:-

```
<?php
        class dadaji
        {
                Public function __construct ()
                {
                        echo “Sir… you are the best”;
                }
        }
        class papa extends dadaji
        {
                function display ()
                {
                        echo “display function called”;
                }

        }
        $obj =new papa ();
        $obj->display();
?>
```


Q. Are parent constructor called implicitly when creating an object of class?

Ans.

Yes, parent constructor called implicitly when creating an object of class, if the child class does not define a constructor.

But in case, if we create a constructor in child class, then we have to call patrent constructor explicitly by using scope resolution operator (::).

Example:-

```
<?php
        class my_constructor
         {
                public function __construct()
                 {
```

```
            echo   "Hello Sir I'm Constructor";
        }
    }
    class abc extends my_constructor
    {
        public function __construct()
        {
            patrent::__construct();    //or my_constructor::__construct();
            echo "I am not constructor sir";
        }
    }
    $obj = new abc();
?>
```

Q. What happen, if constructor is defined as private or protected?

Ans. _r_a_v_i_i_____

Case 1:-

If we create constructor as private and create object of child class, then it will show fatal error, because protected data member can be accessed only in own class. They can't be accessed out of the class.

Case 2:-

If we create constructor as protected, then it will be visible for own class and child/inherit class. It will not be accessed out of the class.
If we want to access protected constructor in child class, then we have to create a function or constructor in child and call it with creating instance (object) of child class. In function or constructor we of child class we have to call parent constructor explicitly by using scope resolution operator (::).

Example:-

```
<?php
        class parent
        {
                protected function __construct()
                {
                        echo “Parent constructor called”;
                }
        }
        class child extends parent
        {
                function __construct()
                {
                        parent::__construct();
                        echo “Child constructor called”;
                }
        }
$obj=new child();
?>
```


Q. What are PHP magic methods/functions? list them. Write program for static keyword in PHP?

Ans. **Magic methods:-**Magic methods are special methods which override PHP's default action when certain actions are performed on an object.

__construct(),
__destruct(),
__call(),
__callStatic(),
__get(),
__set(),
__isset(),
__unset(),
__sleep(),

__wakeup(),
__serialize(),
__unserialize(),
__toString(),
__invoke(),
__set_state(),
__clone(), and __debugInfo()

Static keyword:-

Static properties can be called directly without creating an instance(object) of a class. Static properties are declared with the static keyword.
To access a static property use the class name, scope resolution operator (::), and the property name. Visibility must be public all time.

Example:-

```
class abc
{
        public static $my_static="I am static";
        public $simple_var="I am simple ";
        public function static_fool()
        {
            echo $this->simple_var; // normal variable call by $this
            echo abc::$my_static;
            //echo self::$my_static;  // fool::$my_static;
        }
}
class xyz extends abc
{
        public function static_bar()
        {
            echo $this->$simple_var;
            echo abc::$my_static;
            //echo parent::$my_static;

        }
}
```

```
echo abc::$my_static;
?>
```

Q. Create multiple traits and use it in to a single class?

Ans. Multiple traits are used as multiple inheritance in PHP.

Example:-

```
<?php
trait abc
{
        public function test()
                {
                echo "This is test method";
                }
}
trait xyz
{
        public function sample()
        {
                echo "this is sample method";
        }
}

class c
{
        use abc,xyz;  // multiple inheritance
}
$obj=new c();
$obj->test();
$obj->sample();
?>
```

Q. Write PHP script of object iteration?

Ans. Iteration=Looping

```
<?php
  class myclass {
    private $var;
    protected $var1;
    public $x, $y, $z;
    public function __construct() {
      $this->var="Hello World";
      $this->var1=array(1,2,3);
      $this->x=100;
      $this->y=200;
      $this->z=300;
    }
  }
  $obj = new myclass();
  foreach($obj as $key => $value) {
    print "$key => $value <br>";
  }
?>
```


Q. Use of the $this keyword.

Ans.

$this is a reserved keyword in PHP that refers to the calling object. It is usually the object to which the method belongs, but possibly another object if the method is called statically from the context of a secondary object. This keyword is only applicable to internal methods.

Example:-

```
<?php

        Class MyClass

        {

                Public $mySample=”Hello !”;
```

```
        Public function printSample()
        {
            Echo $this->mySample;
        }
    }
    $obj=new MyClass();
    $obj->printSample();
?>
```

Q. Consider the exercise11and add a edit link near delete link e.g. Clicking up on edit button a particular row should be open in

adding an "edit" link next to a "delete" link in a table or list.

Ans.

1.HTML Structure: Ensure each row has a unique identifier. You might use the `id` attribute for this purpose.

```
<table>
<tr id="row1">
<td>Data 1</td>
<td>Data 2</td>
<td><a href="#" class="edit-link" onclick="editRow('row1')">Edit</a></td>
<td><a href="#" class="delete-link" onclick="deleteRow('row1')">Delete</a></td>
</tr>
<!-- More rows... -->
</table>
```

2.JavaScript Functions: Add JavaScript functions to handle the edit and delete actions. These functions will be responsible for performing the necessary actions on the data or UI.

```
<script>
functioneditRow(rowId)
{
   // Add logic to open the edit form or perform edit action
console.log("Editing row with ID: " + rowId);
 }
functiondeleteRow(rowId)
{
   // Add logic to delete the row
console.log("Deleting row with ID: " + rowId);
 }
</script>
```


3.Styling: You might want to style the edit and delete links to make them visually distinct.

```
.edit-link, .delete-link
{
margin-right: 10px;
color: blue; /* Edit link color */
}
.delete-link
{
 //code
```

}

- **editing mode**

  - Visual Code Studio
  - Notepad++
  - Sublime Text
  - Net Beans
  - Atom
  - PHP Strom

- **e.g. on the Particular row there should be filled text box with data and on the option column there should be a confirm button clicking upon it arrow should be updated.**

1. **Particular Row :**
   - This likely refers to a specific entry or record in your data.
2. **Text Box With Data :-**
   - This is where information related to the particular row is displayed or entered.
3. **Option Column :-**
   - This could be a column proving different actions or options for each row.
4. **Confirm Button :-**
   - Clicking on this button likely triggers a specific action related to the row.
5. **Arrow Update :-**
   - The arrow updates suggests a visual change, possibly indicating the completion or confirmation of an action.


```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Interactive Table</title>

<style>

table

{
```

```
border-collapse: collapse;
width: 100%;
    }

th, td
{
border: 1px solid #ddd;
padding: 8px;
text-align: left;
    }
button
{
padding: 5px 10px;
cursor: pointer;
    }
</style>
</head>
<body>
<table>
<thead>
<tr>
<th>Particular</th>
<th>Text Box with Data</th>
<th>Options</th>
```

```
</tr>
</thead>
<tbody>
<tr>
<td>Row 1</td>
<td><input type="text" id="dataInput1"></td>
<td>
<button onclick="confirmAction(1)">Confirm</button>
</td>
</tr>
<tr>
<td>Row 2</td>
<td><input type="text" id="dataInput2"></td>
<td>
<button onclick="confirmAction(2)">Confirm</button>
</td>
</tr>
<!-- Add more rows as needed -->
</tbody>
</table>
<script>
functionconfirmAction(rowNumber)
{
    // Perform actions related to the confirmation, e.g., update arrow
```

```
alert('Confirmed for Row ' + rowNumber);
      // You can update the arrow or perform other actions here
   }
</script>


</body>
</html>
```

- **Create Hotel Room Booking System User can book room by 3 ways**

1.Website Booking:

- **User Interface:**
  - Homepage: Displays available rooms, promotions, and general information.
  - Search Page: Allows users to filter rooms based on criteria (e.g., date, number of guests).
  - Room Details: Provides information about each room, including amenities and prices.
  - Booking Form: Users can input details such as check-in/out dates, the number of guests, and payment information.
- **Backend Logic:**
  - Validate user inputs and check room availability.
  - Calculate total cost based on room rates and selected dates.
  - Secure payment processing.
  - Send confirmation email with booking details.
- **Database:**
  - Store room information (availability, rates, amenities).
  - Maintain user data (bookings, personal information).

2.Mobile App Booking:

- **User Interface:**
  - Similar to the website but optimized for mobile devices.
  - Responsive design for various screen sizes.
- **Backend Logic:**
  - Utilize APIs for seamless communication between the app and server.
  - Implement push notifications for booking updates.
  - Offline functionality for viewing bookings without an internet connection.

- **Database:**
  - Synchronize data between the app and the central database.

3.Phone Call Booking:

- **Interactive Voice Response (IVR) System:**
  - Welcome and guide the user through the booking process.
  - Voice recognition for user input.
  - Provide options for room selection, dates, and payment.
- **Backend Logic:**
  - Convert voice inputs to text and process them.
  - Check room availability and calculate the total cost.
  - Generate a booking confirmation.
- **Database:**
  - Update room availability and store booking details.

- **Full day**

1. User Registration:

- Users need to register for an account to book a room.
- Collect basic information like name, email, and password for registration.
- Implement authentication to secure user accounts.

2. Room Selection:

- Display a list of available rooms with details (e.g., room type, price, amenities).
- Allow users to select a room for booking.

3. Booking Methods:

- **a. Online Booking:**
  - Users can log in, choose the desired room, and make an online payment.
  - Integrate a payment gateway for secure transactions.
  - Upon successful payment, confirm the booking and provide a booking ID.
- **b. Walk-in Booking:**
  - Users can physically visit the hotel and book a room at the reception.
  - Receptionist enters user details and assigns a room.
  - Generate a booking ID and provide it to the user.
- **c. Phone Booking:**

- Users can call the hotel to inquire about room availability.
- Hotel staff can check availability, take user details over the phone, and book the room.
- Provide a booking confirmation with a unique ID.

4.Reservation Management:

- Implement a system to manage reservations.
- Avoid double bookings by updating room availability in real-time.
- Provide an admin interface to view and manage bookings.

5. User Dashboard:

- Users can log in to their accounts to view their booking history.
- Display upcoming and past bookings with details.

6. Notifications:

- Send email/SMS notifications for successful bookings and reminders.
- Notify users of any changes to their reservations.

7. Cancellation:

- Allow users to cancel bookings with a refund policy.
- Implement cancellation confirmation and refund processing.

8. Reporting:

- Generate reports for the hotel staff to analyze booking trends, occupancy rates, etc.

9. Security Measures:

- Implement secure coding practices.
- Use HTTPS for secure data transmission.
- Regularly update and patch the system to prevent security vulnerabilities.

- **Half day**

1. User Registration:

- Users should be able to create an account or log in if they already have one.
- Collect necessary information such as name, email, contact details, and password.

2. Room Selection:

- Display a list of available rooms with details like room type, price, and availability.
- Allow users to choose a room based on their preferences.
- Implement a calendar to show room availability for half-day bookings.

3. Booking Process:

- Users can initiate a booking by selecting the check-in and check-out dates and times.
- For half-day bookings, provide options for morning or afternoon check-in/check-out.
- Calculate the total cost based on the selected room and duration.

4. Payment Integration:

- Integrate a secure payment gateway to handle transactions.
- Accept various payment methods (credit/debit cards, online wallets, etc.).
- Confirm the booking only after successful payment.

5. Booking Confirmation:

- Send a confirmation email or message to the user with booking details.
- Display a confirmation page with a summary of the booking.

6. User Dashboard:

- Provide a user dashboard where users can view their booking history.
- Allow users to cancel or modify bookings within a specified timeframe.

7. Admin Panel:

- Implement an admin panel for hotel staff to manage room availability, bookings, and user accounts.
- Allow admins to add new rooms, update prices, and view booking reports.

8. Notifications:

- Send reminders to users before their check-in/check-out time.
- Notify users of successful bookings, cancellations, or any changes to their reservation.

9. Reviews and Feedback:

- Allow users to leave reviews and ratings for their stay.
- Display reviews to help future users make informed decisions.

10. Security:

- Implement secure authentication and authorization mechanisms.
- Ensure that sensitive information such as payment details is encrypted.

11. Mobile Responsiveness:

- Design the system to be mobile-friendly for users who prefer booking on smartphones or tablets.

12. Analytics:

- Integrate analytics to track user behavior, popular rooms, and booking patterns.

- **Custom**

Hotel Room Booking System Components:

1.Database:

- Create a database to store information about rooms, reservations, and users. Tables could include `Rooms`, `Reservations`, and `Users`.

2.Backend:

- Use a backend framework (e.g., Flask, Django, Express.js) to handle server-side logic and interact with the database.

3.User Authentication:

- Implement a user authentication system to secure user accounts and booking history.

4.Web Interface:

- Create a web interface for users to book rooms. Use HTML, CSS, and JavaScript for the frontend. Ensure a responsive design for different devices.

5.Mobile App:

- Develop a mobile app (iOS/Android) using a framework like React Native or Flutter, providing similar functionalities to the web interface.

6.Phone Booking:

- Allow users to book rooms over the phone by calling a dedicated reservation hotline. Staff members can use a secure admin panel to manually enter reservations made via phone.

7.Room Availability:

- Implement a mechanism to check room availability for specific dates. Update the availability status in real-time.

8.Payment Integration:

- Integrate a payment gateway for online bookings. Ensure secure handling of payment information.

• If user select for the full day than user only have selection for the checking checkout date

1.Radio Buttons for Stay Option:

- ☐ Full Day
- ☐ Custom Stay

2.Date Picker for Check-In:

- ☐ Check-In Date: [Date Picker]

3.Date Picker for Check-Out (Conditional):

- ☐ Check-Out Date: [Date Picker]

• If user select Half day than user have option of date and slot option(like user want to book room for first half – Morning (8AM to 6PM) if user select for second halfit‟s for evening (7PM to Morning 7AM)). Do proper validation like if user can book only available slot. (havetouse jQuery -> Ajax, validation, Json passing).

To implement the described functionality with jQuery, Ajax, and JSON passing for booking a room with date and slot options, you can follow these steps:

1. HTML Structure :

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Room Booking</title>
<!-- Include jQuery -->
<scriptsrc="https://code.jquery.com/jquery-3.6.4.min.js"></script>
</head>
<body>
<h2>Room Booking</h2>

<label for="bookingType">Select Booking Type:</label>
<select id="bookingType">
<option value="fullDay">Full Day</option>
<option value="halfDay">Half Day</option>
</select>
<div id="dateSlotOptions" style="display:none;">
<label for="bookingDate">Select Date:</label>
<input type="date" id="bookingDate">
<label for="bookingSlot">Select Slot:</label>
<select id="bookingSlot">
<option value="morning">Morning (8AM to 6PM)</option>
<option value="evening">Evening (7PM to 7AM)</option>
</select>
```

```
</div>
<button id="submitBooking">Submit Booking</button>
<script src="script.js"></script>
</body>
</html>
```

1. **JavaScript (script.js):**

```
$(document).ready(function()
{
  // Event listener for booking type change
  $("#bookingType").change(function()
{
if ($(this).val() === "halfDay")
{
     $("#dateSlotOptions").show();
   }
else
{
     $("#dateSlotOptions").hide();
   }
  });
  // Event listener for submit button
  $("#submitBooking").click(function()
{
```

```
   // Validate and gather user inputs
varbookingType = $("#bookingType").val();
varbookingDate = $("#bookingDate").val();
varbookingSlot = $("#bookingSlot").val();
   // Validate user inputs
if (bookingType === "halfDay" && (!bookingDate || !bookingSlot))
{
alert("Please select date and slot for half-day booking.");
return;
   }


   // Construct data to send via Ajax
varbookingData =
{
type: bookingType,
date: bookingDate,
slot: bookingSlot
   };
   // Perform Ajax request
   $.ajax({
url: "your_booking_endpoint.php", // Replace with your server-side endpoint
type: "POST",
data: JSON.stringify(bookingData),
contentType: "application/json; charset=utf-8",
```

```
dataType: "json",
success: function(response)
{
      // Handle success response from server
alert("Booking successful!");
    },
error: function(error)
{
      // Handle error response from server
alert("Booking failed. Please try again.");
    }
  });
 });
});
```


2. **Server-side (your_booking_endpoint.php):**

```
<?php
header('Content-Type: application/json');

// Retrieve JSON data from the request
$data = json_decode(file_get_contents('php://input'), true);

// Validate and process booking
if ($data['type'] === 'halfDay')
{
   // Perform validation for half-day booking and process accordingly
   // Replace this with your actual validation and booking logic
   $isValidBooking = true; // Replace with your validation logic
```

```
if ($isValidBooking)
{
      // Booking successful
      $response = array('status' => 'success', 'message' => 'Booking successful!');
   }
else
{
      // Booking failed
      $response = array('status' => 'error', 'message' => 'Invalid date or slot for half-day
booking.');
   }
}
else
{
   // Process full-day booking
   // Replace this with your logic for full-day booking
   $response = array('status' => 'success', 'message' => 'Booking successful!');
}

// Send JSON response back to the client
echojson_encode($response);
?>
```


Jquery

- **What is jQuery?**
  jQuery is a fast, small, and feature-rich JavaScript library. It simplifies the process of traversing HTML documents, handling events, creating animations, and managing Ajax interactions. jQuery was created by John Resig and was first released in 2006. It quickly gained popularity due to its ease of use and the ability to perform common tasks with less code compared to raw JavaScript.
  Some key features of jQuery include:

1.DOM Manipulation: jQuery simplifies the process of selecting and manipulating HTML elements on the page. It provides a concise syntax for common tasks like changing content, attributes, and styles.

2.Event Handling: jQuery makes it easy to handle various events, such as clicks, keypresses, and mouse movements. Event handling in jQuery is straightforward and cross-browser compatible.

3.Ajax Support: jQuery simplifies asynchronous JavaScript and XML (Ajax) requests, allowing developers to fetch data from a server and update parts of a web page without requiring a full page refresh.

4.Animation Effects: jQuery includes built-in functions for creating animations and transitions, making it easier to add visual effects to web pages.

5. Utilities: jQuery includes a variety of utility functions that streamline common tasks, such as working with arrays and objects, making AJAX requests, and handling browser compatibility issues.

- **How are JavaScript and jQuery different?**

1.Core Purpose:

- **JavaScript:** It is a general-purpose programming language that can be used for a wide range of tasks, not limited to web development. It is supported by all major web browsers and can be used for both client-side and server-side development.
- **jQuery:** It is a lightweight, cross-platform JavaScript library designed to simplify the client-side scripting of HTML. jQuery makes it easier to perform common tasks like DOM manipulation, event handling, animation, and AJAX with less code.

2.Syntax:

- **JavaScript:** It is a programming language with its syntax. It provides a way to interact with the DOM (Document Object Model) and handle events, among other things.
- **jQuery:** It is essentially a set of functions written in JavaScript. jQuery syntax is often considered simpler and more concise than raw JavaScript, especially for tasks like DOM manipulation.

3.DOM Manipulation:

- **JavaScript:** It uses native methods to manipulate the DOM. For example, `document.getElementById()`, `document.createElement()`, etc.
- **jQuery:** It provides a simplified and often more cross-browser compatible syntax for DOM manipulation. For instance, `$("#elementId")` or `$(".className")` to select elements.

4.Event Handling:

- **JavaScript:** Event handling in JavaScript can be done using the `addEventListener` method or by assigning event attributes directly in the HTML.
- **jQuery:** It simplifies event handling with methods like `click()`, `change()`, etc. For example, `$("#myButton").click(function() { /* do something */ });`.

5.Ajax:

- **JavaScript:** Native JavaScript provides the `XMLHttpRequest` object for making AJAX requests.
- **jQuery:** jQuery simplifies AJAX requests with methods like `$.ajax()` or shorthand methods like `$.get()`, `$.post()`.

6.Cross-browser Compatibility:

- **JavaScript:** Developers need to be aware of and handle cross-browser compatibility issues themselves.
- **jQuery:** One of the main reasons jQuery gained popularity was its ability to abstract away many cross-browser inconsistencies, providing a more consistent interface.

7.Size and Performance:

- **JavaScript:** Native JavaScript code can sometimes be more lightweight because it doesn't include the additional abstraction layer that jQuery provides.
- **jQuery:** While it simplifies coding, it adds an extra layer, and the library itself adds some file size overhead. In modern web development, where bandwidth and performance are crucial, some developers opt to use JavaScript directly for smaller projects.

- **Which is the starting point of code execution in jQuery?**
  The starting point of code execution is often inside a document-ready event handler. The document-ready event occurs when the DOM (Document Object Model) has been fully loaded and is ready to be manipulated. This ensures that the jQuery code runs after the HTML document has been parsed.

Syntax:-

```
$(document).ready(function()
{
   // jQuery code goes here
});
```

- **Document Load Vs Window. Load() jQuery**

$(document).ready():

- This event occurs when the DOM (Document Object Model) is ready, which means the HTML structure of the page has been fully loaded and parsed.
- It is triggered as soon as the DOM is ready, even if other external resources like images, stylesheets, and scripts are still loading.
- It is suitable for tasks that don't require waiting for all external resources to be fully loaded.

Example:

```
$(document).ready(function()
{
   // Your code here
});
```

$(window).load():

- This event occurs when all assets on the page, including images and other resources, have finished loading.
- It waits for the entire page, including external resources, to be fully loaded before triggering.
- It is suitable for tasks that depend on the dimensions or content of images, as it ensures that all images have been loaded.

Example:

```
$(window).load(function()
{
   // Your code here
});
```

- **What is the difference between prop and attr?**

1.Attribute (attr):

- In HTML, attributes provide additional information about HTML elements and are always included in the opening tag of an element.
- Attributes define the properties of an HTML element and are used to configure or modify the behavior of the element.
- Examples of attributes include `class`, `id`, `src`, `href`, `alt`, etc.
- Attributes are part of the HTML document and can be accessed and modified using JavaScript.

Example:

<imgsrc="image.jpg" alt="An example image" class="my-image">

2.Property (prop):

- In the context of JavaScript and the Document Object Model (DOM), properties are values associated with JavaScript objects.
- Elements in the DOM are represented as objects, and these objects have properties that can be accessed and modified using JavaScript.
- Properties often represent the current state of an element, and they can be dynamic, changing based on user interactions or other events.
- Examples of properties include `innerHTML`, `value`, `style`, etc.

Example:

```
varmyElement = document.getElementById("exampleElement");
console.log(myElement.innerHTML);     // Accessing the innerHTML property of an element
```

- **Explain Difference Between JQuery And JavaScript?**

1.Core Purpose:

- **JavaScript:** It is a general-purpose programming language that can be used for a wide range of tasks, not limited to web development. It is supported by all major web browsers and can be used for both client-side and server-side development.
- **jQuery:** It is a lightweight, cross-platform JavaScript library designed to simplify the client-side scripting of HTML. jQuery makes it easier to perform common tasks like DOM manipulation, event handling, animation, and AJAX with less code.

2.Syntax:

- **JavaScript:** It is a programming language with its syntax. It provides a way to interact with the DOM (Document Object Model) and handle events, among other things.
- **jQuery:** It is essentially a set of functions written in JavaScript. jQuery syntax is often considered simpler and more concise than raw JavaScript, especially for tasks like DOM manipulation.

3.DOM Manipulation:

- **JavaScript:** It uses native methods to manipulate the DOM. For example, `document.getElementById()`, `document.createElement()`, etc.
- **jQuery:** It provides a simplified and often more cross-browser compatible syntax for DOM manipulation. For instance, `$("#elementId")` or `$(".className")` to select elements.

4.Event Handling:

- **JavaScript:** Event handling in JavaScript can be done using the `addEventListener` method or by assigning event attributes directly in the HTML.
- **jQuery:** It simplifies event handling with methods like `click()`, `change()`, etc. For example, `$("#myButton").click(function() { /* do something */ });`.

5.Ajax:

- **JavaScript:** Native JavaScript provides the `XMLHttpRequest` object for making AJAX requests.
- **jQuery:** jQuery simplifies AJAX requests with methods like `$.ajax()` or shorthand methods like `$.get()`, `$.post()`.

6.Cross-browser Compatibility:

- **JavaScript:** Developers need to be aware of and handle cross-browser compatibility issues themselves.
- **jQuery:** One of the main reasons jQuery gained popularity was its ability to abstract away many cross-browser inconsistencies, providing a more consistent interface.

7.Size and Performance:

- **JavaScript:** Native JavaScript code can sometimes be more lightweight because it doesn't include the additional abstraction layer that jQuery provides.
- **jQuery:** While it simplifies coding, it adds an extra layer, and the library itself adds some file size overhead. In modern web development, where bandwidth and performance are crucial, some developers opt to use JavaScript directly for smaller projects.

- **How We Can Select The Specified <li> Element From The ListOf Elements In ?**
  To select a specific `<li>` element from a list of `<li>` elements within a `<ul>` (unordered list) using JavaScript, you can use various methods.

Example:

1. Using JavaScript and the DOM:

Assuming you have an unordered list with an id, let's say "myList":

```
<ul id="myList">
```

```
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
</ul>
```

JavaScript code to select the second `<li>` element:

```
// Get the <ul> element by its id
varmyList = document.getElementById("myList");

// Get the second <li> element (index 1, as indices start from 0)
varsecondLi = myList.getElementsByTagName("li")[1];

// Now 'secondLi' contains the reference to the second <li> element
console.log(secondLi.textContent);
```

2. Using querySelector:

```
// Use querySelector to directly select the second <li> element
varsecondLi = document.querySelector("#myListli:nth-child(2)");

// Now 'secondLi' contains the reference to the second <li> element
console.log(secondLi.textContent);
```


3. Using querySelectorAll:

```
varallLiElements = document.querySelectorAll("#myList li"); // Use querySelectorAll to select all <li> elements and then choose the second one

varsecondLi = allLiElements[1]; // Get the second <li> element (index 1, as indices start from 0)

console.log(secondLi.textContent);  // Now 'secondLi' contains the reference to the second <li> element
```

- **In <table> Design Change The Color Of Even <tr> Elements To “green” And Change The Color Off Odd <tr> Elements To “blue” Color? Give An Example Code?**

  the even and odd `<tr>` elements differently. Here's an example code:

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

```
<style>
table
	{
width: 100%;
border-collapse: collapse;
	}

tr:nth-child(even)
{
background-color: green;
	}

tr:nth-child(odd)
{
background-color: blue;
	}

th, td
{
border: 1px solid black;
padding: 8px;
text-align: left;
	}
</style>
</head>
<body>

<table>
<thead>
<tr>
<th>Header 1</th>
<th>Header 2</th>
<th>Header 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 1, Cell 1</td>
<td>Row 1, Cell 2</td>
<td>Row 1, Cell 3</td>
</tr>
<tr>
<td>Row 2, Cell 1</td>
<td>Row 2, Cell 2</td>
<td>Row 2, Cell 3</td>
</tr>
```

```
<tr>
<td>Row 3, Cell 1</td>
<td>Row 3, Cell 2</td>
<td>Row 3, Cell 3</td>
</tr>
</tbody>
</table>
</body>
</html>
```

- **How We Can Implement Animation Effects In Jquery?**
  Implementing animation effects in jQuery involves using the built-in animation functions provided by jQuery. These functions allow you to animate the properties of HTML elements, such as their position, size, opacity, and more.

Include jQuery:
Ensure that you include the jQuery library in your HTML file. You can either download it and host it locally or use a CDN (Content Delivery Network). For example:

```
<scriptsrc="https://code.jquery.com/jquery-3.6.4.min.js"></script>
```


Select Elements: Use jQuery to select the HTML elements you want to animate. You can do this using the `$` function.

```
<div id="myElement">Animate me!</div>
varmyElement = $("#myElement");
```

Animate Properties: Use jQuery's animation functions to change the properties of the selected elements. Common properties include `height`, `width`, `opacity`, `left`, `top`, etc.

```
myElement.animate(
{
opacity: 0.5,
left: '+=50',
height: 'toggle'
}, 1000); // 1000 is the duration in milliseconds
```

Chaining Animations: You can chain multiple animations together using the `queue` option or by simply calling additional animation functions.

```
myElement
```

```
  .animate({ left: '+=50' }, 500)
  .animate({ opacity: 0.5 }, 500);
```

Callback Functions: You can use callback functions to execute code after an animation completes.

```
myElement.animate({ opacity: 0.5 }, 1000, function() {
  // Code to execute after the animation is complete
});
```

Easing: jQuery provides easing options to control the speed of animations. You can use built-in options like `'swing'` or `'linear'`, or include a custom easing function.

```
myElement.animate({ left: '+=50' }, { duration: 500, easing: 'linear' });
```

Stop Animation: You can stop an animation in progress using the `stop` function.

```
myElement.stop();
```



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<scriptsrc="https://code.jquery.com/jquery-3.6.4.min.js"></script>
<title>jQuery Animation Example</title>
</head>
<body>

<button id="animateButton">Animate</button>
<div id="animatedDiv">Hello, I'm animated!</div>

<script>
    $(document).ready(function()
{
       $("#animateButton").click(function()
{
         $("#animatedDiv").animate(
          {
left: '+=100',
opacity: 0.5
         }, 1000);
       });
```

```
    });
</script>

</body>
</html>
```

- **Apply jQuery validation using library.**
  jQuery Validation is a popular library for handling form validation in web applications. To use jQuery Validation, you need to include the jQuery library and the jQuery Validation plugin in your HTML file. Here's a step-by-step guide:

Include jQuery: Make sure to include the jQuery library in your HTML file. You can do this by adding the following line in the `<head>` section of your HTML:

```
<scriptsrc="https://code.jquery.com/jquery-3.6.4.min.js"></script>
```

Include jQuery Validation: Include the jQuery Validation plugin by adding the following line below the jQuery script tag:

```
<script src="https://cdn.jsdelivr.net/jquery.validation/1.19.3/jquery.validate.min.js"></script>
```


Create a Form: Create an HTML form that you want to validate. For example:

```
<form id="myForm">
<label for="name">Name:</label>
<input type="text" id="name" name="name" required>

<label for="email">Email:</label>
<input type="email" id="email" name="email" required>

<input type="submit" value="Submit">
</form>
```

Initialize jQuery Validation: Add a script to initialize the jQuery Validation on your form. Place this script after including the jQuery and jQuery Validation scripts:

```
<script>
  $(document).ready(function ()
  {
    $("#myForm").validate();
  });
</script>
```

- **Create custom dynamic function for require field validator.**
  To create a custom dynamic function for a required field validator in JavaScript, you can define a function that takes an object as input and checks if the required fields are present. Here's an example of a simple dynamic required field validator function:

```
functioncreateRequiredFieldValidator(requiredFields)
{
return function (data)
{
constmissingFields = [];

   // Check each required field
requiredFields.forEach(field =>
{
if (!data.hasOwnProperty(field) || data[field] === null || data[field] === undefined ||
data[field] === '')
{
missingFields.push(field);
    }
   });

   // Return the result
if (missingFields.length === 0)
{
return { isValid: true };
   }
else
{
return { isValid: false, missingFields };
   }
  };
}
```


- **Get state data by country selection (Ajax).**
  To retrieve state data based on country selection using Ajax, you'll need to create a web page with HTML, JavaScript, and use a server-side script (like PHP, Node.js, or any server-side technology of your choice) to handle the data fetching. Below is a simple example using HTML, JavaScript, and PHP:

(HTML)Index.php

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Country and State Selection</title>
</head>
<body>
<label for="country">Select Country:</label>
<select id="country" onchange="getStates()">
<!-- Populate this with your list of countries -->
<option value="usa">USA</option>
<option value="canada">Canada</option>
<!-- Add more options as needed -->
</select>

<label for="state">Select State:</label>
<select id="state"></select>

<script>
functiongetStates()
  {
varcountrySelect = document.getElementById('country');
varstateSelect = document.getElementById('state');
varselectedCountry = countrySelect.value;

    // Make an Ajax request to get states based on the selected country
varxhr = new XMLHttpRequest();
xhr.onreadystatechange = function()
   {
if (xhr.readyState == 4 &&xhr.status == 200)
     {
        // Parse the JSON response
var states = JSON.parse(xhr.responseText);

        // Clear existing options
stateSelect.innerHTML = '';

        // Populate the state dropdown with new options
for (vari = 0; i<states.length; i++)
```

```
        {
var option = document.createElement('option');
option.value = states[i];
option.text = states[i];
stateSelect.add(option);
          }
        }
      };

      // Replace 'get_states.php' with the path to your server-side script
xhr.open('GET', 'get_states.php?country=' + selectedCountry, true);
xhr.send();
    }
</script>
</body>
</html>
```

PHP (get_states.php):


```
<?php
// Replace this with your own logic to fetch states based on the selected country
if(isset($_GET['country'])) {
  $selectedCountry = $_GET['country'];

  // Example: Provide states based on the selected country
  $states = [];
if($selectedCountry == 'usa') {
    $states = ['New York', 'California', 'Texas'];
  } elseif($selectedCountry == 'canada') {
    $states = ['Ontario', 'Quebec', 'British Columbia'];
  }

  // Send the states as a JSON response
echojson_encode($states);
}
?>
```

- **Image uploading with preview.**

=> To implement image uploading with a preview in a web application, you can use HTML, JavaScript, and CSS.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Image Upload with Preview</title>
<style>
   #preview-container
  {
display: flex;
justify-content: center;
align-items: center;
height: 200px;
border: 2px dashed #ccc;
margin-bottom: 20px;
   }

   #preview-img
  {
max-width: 100%;
max-height: 100%;
   }
</style>
</head>
<body>

<input type="file" id="imageInput" accept="image/*" onchange="previewImage(event)">
<div id="preview-container">
<img id="preview-img" src="#" alt="Image Preview">
</div>

<script>
functionpreviewImage(event)
{
const input = event.target;
constpreviewImg = document.getElementById('preview-img');
constpreviewContainer = document.getElementById('preview-container');

const file = input.files[0];

if (file)
   {
const reader = new FileReader();
```

```
reader.onload = function (e)
      {
previewImg.src = e.target.result;
       };

reader.readAsDataURL(file);
previewContainer.style.display = 'block';
      }
else
    {
previewImg.src = '#';
previewContainer.style.display = 'none';
      }
    }
</script>

</body>
</html>
```