Q1} What is jQuery? Explain features supported by jQuery.

jQuery is a popular and widely-used JavaScript library designed to simplify the client-side scripting of HTML. It provides a concise and efficient way to manipulate HTML documents, handle events, create animations, make Ajax requests, and manage the overall client-side behaviour of a website.

Here are some key features supported by jQuery:

1. DOM Manipulation: jQuery allows you to easily select and manipulate HTML elements using a simple and intuitive syntax. You can change element attributes, add or remove classes, modify CSS styles, create new elements, and manipulate the DOM structure.

2. Event Handling: jQuery provides powerful event handling capabilities. You can attach event handlers to HTML elements and respond to events such as clicks, mouse movements, keyboard inputs, and form submissions. This allows you to create interactive and dynamic web pages.

3. Ajax Support: jQuery simplifies the process of making Ajax requests to the server and handling the responses. It provides methods to perform asynchronous HTTP requests, send data to the server, and update parts of the page without requiring a full page reload. This enables you to build responsive and interactive web applications.

4. Animations and Effects: jQuery offers a range of animation and visual effects that can be applied to HTML elements. You can create smooth transitions, fade-ins and fade-outs, slide effects, and more. These features add interactivity and enhance the user experience on your website.

5. Utilities and Helper Functions: jQuery includes a set of utility functions that simplify common tasks in JavaScript development. These functions provide features like iterating over arrays and objects, manipulating strings, handling data types, working with events, and managing animations. They help streamline your code and make it more readable and maintainable.

6. Cross-Browser Compatibility: jQuery abstracts away many of the inconsistencies and browser-specific quirks in JavaScript and provides a unified API that works consistently across different browsers. It handles the complexities of browser compatibility, allowing you to write code that works reliably across a wide range of browsers and versions.

Q2} Write a short note on jQuery ready() function.

The jQuery ready() function is a commonly used method that ensures the execution of JavaScript code when the Document Object Model (DOM) is fully loaded and ready to be manipulated. It is designed to address the timing issue that occurs when JavaScript code attempts to manipulate HTML elements before they have been fully rendered by the browser.

- The syntax of the ready() function is straightforward:

```
$(document).ready(function() {
        // Code to be executed when the DOM is ready
});
```

- Alternatively, you can use the shorthand version of the ready() function:

```
$(function() {          // Code to be executed when the DOM is ready
});
```

- Here's how the ready() function works:

1. The $ (or jQuery) is a function that references the jQuery library.

2. The $ function is used to select elements or create new ones, but in the case of $(document), it selects the entire HTML document.

3. The ready() function is invoked on the document object.

4. Inside the ready() function, you write the code that you want to execute as soon as the DOM is ready.

By using the ready() function, you can ensure that your JavaScript code is executed at the appropriate time. This is particularly important when your code relies on specific HTML elements or needs to interact with the DOM structure.

The ready() function is often used to set up event handlers, perform initializations, bind functions to buttons or other elements, and generally prepare the page for user interaction. It provides a convenient way to organize and encapsulate your code, making it more maintainable and easier to understand.

Q3} List jQuery actions and explain any two.

1. hide() and show(): These actions allow you to hide or show HTML elements on a webpage. The hide() function hides the selected elements by setting their CSS display property to "none", effectively removing them from the visible layout. Conversely, the show() function sets the CSS display property to its original value, making the hidden elements visible again. These actions can be used to create interactive effects, toggle visibility, or control the display of content based on user interactions.

2. addClass() and removeClass(): These actions enable you to add or remove CSS classes from HTML elements. The addClass() function adds one or more classes to the selected elements, allowing you to dynamically change their appearance or apply specific styles defined in CSS. Conversely, the removeClass() function removes specified classes from the selected elements. This action is often used to apply or remove styling based on user interactions, such as highlighting selected elements or toggling styles for active elements.

3. attr(): The attr() action is used to get or set attributes of HTML elements. It can retrieve the value of a specified attribute from the first matched element in the jQuery collection or set the value of an attribute for all matched elements. For example, you can use attr('src') to retrieve the value of the src attribute of an image or attr('href', 'newlink') to set a new value for the href attribute of a link. This action is handy for dynamically manipulating attributes, such as changing image sources or updating links based on user interactions or data changes.

4. fadeIn() and fadeOut(): These actions provide smooth fade-in and fade-out effects for HTML elements. The fadeIn() function gradually increases the opacity of the selected elements, making them appear gradually. Conversely, the fadeOut() function gradually decreases the opacity, causing the elements to fade away. These actions are useful for creating visually appealing transitions or animations, such as displaying tooltips, modals, or notifications with smooth fade-in and fade-out effects.

Q4} Explain in brief jQuery validation plugin.

The jQuery Validation Plugin is a popular jQuery extension that provides an easy and efficient way to validate HTML forms on the client side. It simplifies the process of validating user inputs, such as checking for required fields, validating email addresses, ensuring correct formatting, and handling custom validation rules.

- Here are some key features and benefits of the jQuery Validation Plugin:

1. Form Validation: The plugin allows you to define validation rules for form fields using simple and intuitive syntax. You can specify rules such as required fields, minimum and maximum length, email format, numeric values, and more. It provides a wide range of built-in validation methods that cover common use cases. You can also create custom validation rules to suit your specific requirements.

2. Error Messages: When form inputs fail validation, the plugin displays error messages next to the corresponding fields, indicating the validation errors. You can customize the appearance and placement of error messages to match your website's design and layout. The error messages are automatically shown and hidden based on user interactions, providing real-time feedback to users.

3. Validation Events: The plugin offers various validation events that can be triggered during the validation process. You can use these events to customize the validation behavior, perform additional actions, or integrate with other JavaScript functions. Events such as submit, focusin, focusout, and keyup allow you to control when and how validation is triggered.

4. AJAX Validation: The jQuery Validation Plugin supports AJAX-based validation, allowing you to perform server-side validation asynchronously without submitting the form. This feature is useful when you need to validate form inputs against server-side data or perform complex validation checks that require server interaction. You can define custom AJAX validation methods and handle the response accordingly.

5. Validation Options and Configuration: The plugin provides a range of options and configuration settings to tailor the validation behavior to your needs. You can customize error messages, change the default error styles, specify the order of validation, define validation groups, and more. These options give you fine-grained control over the validation process and allow you to adapt it to your specific requirements.

The jQuery Validation Plugin is widely used in web development to enhance the user experience by ensuring the correctness and integrity of user inputs before they are submitted to the server. It helps prevent unnecessary form submissions, reduces server-side processing, and provides immediate feedback to users, leading to a smoother and more user-friendly form interaction.

The plugin is highly customizable, well-documented, and actively maintained by the jQuery community. It integrates seamlessly with other jQuery plugins and libraries, making it a powerful tool for form validation in web applications.

Q5} Write categories of jQuery UI and explain jQuery UI Accordion.

jQuery UI is a comprehensive user interface (UI) library built on top of jQuery. It provides a collection of interactive and customizable UI components that can be easily integrated into web applications. The jQuery UI library is categorized into the following main components:

1. Interactions: These components enable users to interact with elements on the page. Examples include draggable elements, resizable elements, selectable elements, sortable lists, and droppable elements.

2. Widgets: jQuery UI offers a variety of UI widgets that enhance the user experience and provide advanced functionality. Some popular widgets include datepickers, tooltips, sliders, progress bars, auto-complete fields, accordions, dialogs, tabs, and menus.

3. Effects: jQuery UI provides a range of animation effects and transitions that can be applied to HTML elements. These effects include fading elements in and out, sliding elements, highlighting elements, and creating smooth transitions between different states.

4. Utilities: jQuery UI also includes utility functions and methods that help with common tasks, such as positioning elements on the page, handling mouse interactions, theming, and localization.

The jQuery UI Accordion:

jQuery UI Accordion: The Accordion widget allows you to create collapsible content sections, commonly used to organize and present information in a compact and space-efficient manner. It consists of a series of collapsible panels, where only one panel is expanded at a time. Clicking on a panel's header expands or collapses its associated content.

- The Accordion widget provides the following features:

1. Multiple Panels: You can define multiple panels within the Accordion, each consisting of a header and content section. Only one panel is open (expanded) at a time, and the others are collapsed. This helps conserve screen space and presents information in a structured and organized way.

2. Collapsible Panels: Accordion panels can be expanded or collapsed by clicking on their headers. This allows users to control the visibility of content sections, revealing or hiding information as needed.

3. Animation Effects: The Accordion widget supports animation effects when expanding or collapsing panels. You can specify the animation speed and easing function to create smooth and visually appealing transitions.

4. Customization: The Accordion widget can be customized to match the design and style of your website or application. You can define custom headers, apply themes, modify styles, and customize the behaviour of the Accordion using various options and call-backs provided by jQuery UI.

Q6} Describe jQuery UI Tabs with example.

jQuery UI Tabs is a powerful UI component that allows you to create tabbed content sections on a web page. It enables you to organize and present information in a tabbed format, where each tab represents a different content section. Users can switch between tabs to view the corresponding content without having to navigate to separate pages.

Here's an example that demonstrates the usage of jQuery UI Tabs:

HTML structure:

```
<div id="tabs">
  <ul>
    <li><a href="#tab1">Tab 1</a></li>
    <li><a href="#tab2">Tab 2</a></li>
    <li><a href="#tab3">Tab 3</a></li>
  </ul>
  <div id="tab1">
    <h2>Tab 1 Content</h2>
    <p>This is the content for Tab 1.</p>
  </div>
  <div id="tab2">
```

```
  <h2>Tab 2 Content</h2>
  <p>This is the content for Tab 2.</p>
 </div>
 <div id="tab3">
  <h2>Tab 3 Content</h2>
  <p>This is the content for Tab 3.</p>
 </div>
</div>
```

JavaScript code:

```
$(function() {
 $("#tabs").tabs();
});
```

In the HTML structure, we have a element with an id of "tabs" that serves as the container for the tabs. Inside the container, we have an unordered list (<ul>) containing individual list items (<li>) for each tab. Each list item has an anchor (<a>) element with an href attribute pointing to the corresponding content section defined by its id value.

In the JavaScript code, we select the element with an id of "tabs" using the jQuery selector $("#tabs") and apply the tabs() method to initialize the tabs functionality.

Once the tabs are initialized, the jQuery UI Tabs component takes care of the rest. It transforms the HTML structure into a set of tabs, where clicking on a tab header reveals the associated content section.

The result is a visually appealing and interactive tabbed interface. Users can click on different tabs to switch between content sections, viewing the relevant information without leaving the page.

Q7} Describe jQuery UI Tooltip with example.

jQuery UI Tooltip is a component that allows you to create customizable tooltips for HTML elements. A tooltip is a small pop-up box that appears when the user hovers over or interacts with an element, providing additional information or context about the element.

- Here's an example that demonstrates the usage of jQuery UI Tooltip:

HTML structure:

```
<span title="Tooltip text" class="tooltip">Hover over me</span>
```

JavaScript code:

```
$(function() {
 $(".tooltip").tooltip();
```

});

In the HTML structure, we have a element with a title attribute that contains the text we want to display as a tooltip. We also assign the class "tooltip" to the element to indicate that we want to apply the tooltip functionality to it.

In the JavaScript code, we use the jQuery selector $(".tooltip") to select all elements with the class "tooltip" and apply the tooltip() method to initialize the tooltip functionality.

Once the tooltips are initialized, the jQuery UI Tooltip component takes care of displaying the tooltips when the user interacts with the elements. By default, the tooltip appears when the user hovers over the element, but you can customize the trigger event if desired.

The result is that when the user hovers over the element with the class "tooltip," a small box with the tooltip text will appear near the element, providing additional information or context.

- The jQuery UI Tooltip component offers various customization options, including:

- Positioning: You can specify where the tooltip should be positioned relative to the element, such as above, below, to the left, or to the right.

- Styling: You can customize the appearance of the tooltips by applying CSS styles to match your website's design and layout.

- Delay: You can control the delay before the tooltip appears and how long it stays visible.

- Content: In addition to the title attribute, you can provide tooltip content dynamically using AJAX or other methods.

- Events and Callbacks: You can handle tooltip-related events and customize the behavior using callback functions.