WEB-2 LONG QUESTION-ANSWERS[1-15]

1 Write a note on jQuery.

jQuery is a lightweight, "write less, do more", JavaScript library.

The purpose of jQuery is to make it much easier to use JavaScript on your website.

jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.

jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

2 Write a note on jQuery events.

All the different visitors' actions that a web page can respond to are called events.

An event represents the precise moment when something happens.

Examples:

- moving a mouse over an element
- selecting a radio button
- clicking on an element

The term **"fires/fired"** is often used with events. Example: "The keypress event is fired, the moment you press a key".

Here are some common DOM events:

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

3 Write a note on effects of jQuery.

jQuery Effect Methods

The following table lists all the jQuery methods for creating animation effects.

Method	Description
animate()	Runs a custom animation on the selected elements

clearQueue()	Removes all remaining queued functions from the selected elements
delay()	Sets a delay for all queued functions on the selected elements
dequeue()	Removes the next function from the queue, and then executes the function
fadeIn()	Fades in the selected elements
fadeOut()	Fades out the selected elements
fadeTo()	Fades in/out the selected elements to a given opacity
fadeToggle()	Toggles between the fadeIn() and fadeOut() methods
finish()	Stops, removes and completes all queued animations for the selected elements
hide()	Hides the selected elements
queue()	Shows the queued functions on the selected elements
show()	Shows the selected elements
slideDown()	Slides-down (shows) the selected elements
slideToggle()	Toggles between the slideUp() and slideDown() methods
slideUp()	Slides-up (hides) the selected elements
stop()	Stops the currently running animation for the selected elements
toggle()	Toggles between the hide() and show() methods

4 What are the types of selectors in jQuery? Explain each.

jQuery Selectors

jQuery selectors allow you to select and manipulate HTML element(s).

jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.

All selectors in jQuery start with the dollar sign and parentheses: $().

The element Selector

The jQuery element selector selects elements based on the element name.

You can select all <p> elements on a page like this:

```
$("p")
```

Example

When a user clicks on a button, all <p> elements will be hidden:

Example

```
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
```

Try it Yourself »

The #id Selector

The jQuery *#id* selector uses the id attribute of an HTML tag to find the specific element.

An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.

To find an element with a specific id, write a hash character, followed by the id of the HTML element:

```
$("#test")
```

Example

When a user clicks on a button, the element with id="test" will be hidden:

Example

```
$(document).ready(function(){
  $("button").click(function(){
    $("#test").hide();
  });
});
```

Try it Yourself »

The .class Selector

The jQuery *.class* selector finds elements with a specific class.

To find elements with a specific class, write a period character, followed by the name of the class:

```
$(".test")
```

Example

When a user clicks on a button, the elements with class="test" will be hidden:

Example

```
$(document).ready(function(){
  $("button").click(function(){
    $(".test").hide();
  });
});
```

Try it Yourself »

More Examples of jQuery Selectors

Syntax	Description	Example
$("*")	Selects all elements	Try it
$(this)	Selects the current HTML element	Try it
$("p.intro")	Selects all <p> elements with class="intro"	Try it
$("p:first")	Selects the first <p> element	Try it
$("ul li:first")	Selects the first element of the first	Try it
$("ul li:first-child")	Selects the first element of every	Try it
$("[href]")	Selects all elements with an href attribute	Try it
$("a[target='_blank']")	Selects all <a> elements with a target attribute value equal to "_blank"	Try it
$("a[target!='_blank']")	Selects all <a> elements with a target attribute value NOT equal to "_blank"	Try it
$(":button")	Selects all <button> elements and <input> elements of type="button"	Try it
$("tr:even")	Selects all even <tr> elements	Try it
$("tr:odd")	Selects all odd <tr> elements	Try it

5 Explain jQuery methods for CSS manipulation.

jQuery css() Method

The `css()` method sets or returns one or more style properties for the selected elements.

Return a CSS Property

To return the value of a specified CSS property, use the following syntax:

```
css("propertyname");
```

The following example will return the background-color value of the FIRST matched element:

Example

```
$("p").css("background-color");
```

Try it Yourself »

Set a CSS Property

To set a specified CSS property, use the following syntax:

```
css("propertyname","value");
```

The following example will set the background-color value for ALL matched elements:

Example

```
$("p").css("background-color", "yellow");
```

Try it Yourself »

Set Multiple CSS Properties

To set multiple CSS properties, use the following syntax:

```
css({"propertyname":"value","propertyname":"value",...});
```

The following example will set a background-color and a font-size for ALL matched elements:

Example

```
$("p").css({"background-color": "yellow", "font-size": "200%"});
```

Try it Yourself »

6 Explain jQuery insert methods.

Add New HTML Content

We will look at four jQuery methods that are used to add new content:

- `append()` - Inserts content at the end of the selected elements
- `prepend()` - Inserts content at the beginning of the selected elements
- `after()` - Inserts content after the selected elements
- `before()` - Inserts content before the selected elements

jQuery append() Method

The jQuery `append()` method inserts content AT THE END of the selected HTML elements.

Example

```
$("p").append("Some appended text.");
```

Try it Yourself »

jQuery prepend() Method

The jQuery `prepend()` method inserts content AT THE BEGINNING of the selected HTML elements.

Example

```
$("p").prepend("Some prepended text.");
```

Try it Yourself »

Add Several New Elements With append() and prepend()

In both examples above, we have only inserted some text/HTML at the beginning/end of the selected HTML elements.

However, both the `append()` and `prepend()` methods can take an infinite number of new elements as parameters. The new elements can be generated with text/HTML (like we have done in the examples above), with jQuery, or with JavaScript code and DOM elements.

In the following example, we create several new elements. The elements are created with text/HTML, jQuery, and JavaScript/DOM. Then we append the new elements to the text with the `append()` method (this would have worked for `prepend()` too) :

Example

```
function appendText() {
  var txt1 = "<p>Text.</p>";               // Create element with HTML
  var txt2 = $("<p></p>").text("Text.");   // Create with jQuery
  var txt3 = document.createElement("p");  // Create with DOM
  txt3.innerHTML = "Text.";
  $("body").append(txt1, txt2, txt3);      // Append the new elements
}
```

Try it Yourself »

jQuery after() and before() Methods

The jQuery `after()` method inserts content AFTER the selected HTML elements.

The jQuery `before()` method inserts content BEFORE the selected HTML elements.

Example

```
$("img").after("Some text after");

$("img").before("Some text before");
```

Try it Yourself »

Add Several New Elements With after() and before()

Also, both the `after()` and `before()` methods can take an infinite number of new elements as parameters. The new elements can be generated with text/HTML (like we have done in the example above), with jQuery, or with JavaScript code and DOM elements.

In the following example, we create several new elements. The elements are created with text/HTML, jQuery, and JavaScript/DOM. Then we insert the new elements to the text with the `after()` method (this would have worked for `before()` too) :

Example

```
function afterText() {
  var txt1 = "<b>I </b>";                  // Create element with HTML
  var txt2 = $("<i></i>").text("love ");   // Create with jQuery
  var txt3 = document.createElement("b");  // Create with DOM
  txt3.innerHTML = "jQuery!";
  $("img").after(txt1, txt2, txt3);        // Insert new elements after <img>
}
```

7 Explain jQuery methods to modify DOM.

Whatever we do use these methods, in the end, are focusing on changing dom events. In other words, we can say using these methods we are manipulating dom. Some of dom manipulation jQuery methods are:

- **addClass():** In the set of matched elements, it adds the specified class or classes to each element.
- **after():** In the set of matched elements, insert data or content specified by the parameter after each element tag.
- **append():** In short words, we can say using this method we insert data or content to the specific element.
- **attr():** This method gives the attribute value of the particular element.
- **before():** Using this method we can insert any data or content before each specified element.
- **clone():** Using this method we deep copy the sourced element.
- **css():** Using this method we change the css of the specified element.
- **detach():** Using this method we simply delete the specified element from the dom tree.
- **hasClass():** Using this method we find the specified class attached to the particular element.
- **empty():** This method deletes all the child nodes attached to the particular element dom tree.
- **height():** This method gives the value of the height of the specified element.
- **offset():** This method sets the coordinate of the specified element relative to the document.
- **position():** This method gets the current position of the specified element relative to its parent element.
- **prop():** This method gives the value of the currently selected element property.
- **replaceAll():** This method replaces all the target element with the specified element.

- **text():** This method sets or gets the text string value of specified elements.
- **Val():** This method gives the current value of the first specified elements.
- **width():** This method gives the computed width value of the specified elements.
- **wrap():** This method wraps the particular html structure around each element in the sets of specified elements.
- **wrapInner():** This method wraps the html structure around the content of each specified elements.

8 Write features of AJAX.

There are several implementation wise important features of AJAX in web development.

- AJAX is a user-friendly approach.
- Does not depend on server technology.
- Makes web pages faster.
- Increases the performance of the web page.
- Support for client-side template rendering.
- Assists in the data view control.
- Support for live data binding.
- Reduces consumption of server resources.
- Responsive and rich user interfaces.
- Needs no traditional form to submit and the whole page refresh.
- Only some part of the page is refreshed/tweaked.
- Processing is similar for all browser types.
- Faster interaction and development of web applications.
- The server uses a reduced amount of bandwidth.
- Improves user's interactivity.
- Offers better usability.

9 Explain working / architecture of AJAX with diagram.

AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

AJAX is not a programming language.

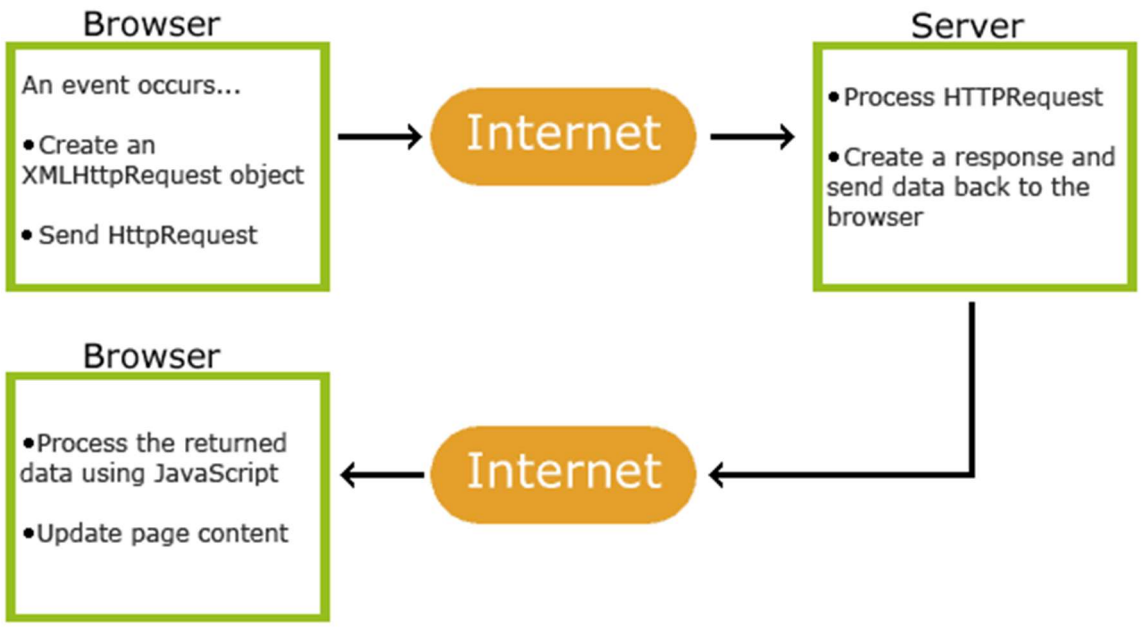
AJAX just uses a combination of:

- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

How AJAX Works



- 1. An event occurs in a web page (the page is loaded, a button is clicked)
- 2. An XMLHttpRequest object is created by JavaScript
- 3. The XMLHttpRequest object sends a request to a web server
- 4. The server processes the request
- 5. The server sends a response back to the web page
- 6. The response is read by JavaScript
- 7. Proper action (like page update) is performed by JavaScript

10 Explain various methods of XMLHttpRequest

XMLHttpRequest Methods

- **abort()**
  Cancels the current request.
- **getAllResponseHeaders()**
  Returns the complete set of HTTP headers as a string.
- **getResponseHeader(headerName)**
  Returns the value of the specified HTTP header.
- **open(method, URL)**
- **open(method, URL, async)**
- **open(method, URL, async, userName)**
- **open(method, URL, async, userName, password)**
  Specifies the method, URL, and other optional attributes of a request.
  The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods such as "PUT" and "DELETE" (primarily used in REST applications) may be possible.
  The "async" parameter specifies whether the request should be handled asynchronously or not. "true" means that the script processing carries on after the send() method without waiting for a response, and "false" means that the script waits for a response before continuing script processing.
- **send(content)**
  Sends the request.
- **setRequestHeader(label, value)**
  Adds a label/value pair to the HTTP header to be sent.

11 Explain various properties of XMLHttpRequest

- **onreadystatechange**
  An event handler for an event that fires at every state change.
- **readyState**
  The readyState property defines the current state of the XMLHttpRequest object.
  The following table provides a list of the possible values for the readyState property –

State	Description
0	The request is not initialized.
1	The request has been set up.
2	The request has been sent.
3	The request is in process.

4	The request is completed.

readyState = 0 After you have created the XMLHttpRequest object, but before you have called the open() method.

readyState = 1 After you have called the open() method, but before you have called send().

readyState = 2 After you have called send().

readyState = 3 After the browser has established a communication with the server, but before the server has completed the response.

readyState = 4 After the request has been completed, and the response data has been completely received from the server.

- **responseText**
  Returns the response as a string.
- **responseXML**
  Returns the response as XML. This property returns an XML document object, which can be examined and parsed using the W3C DOM node tree methods and properties.
- **status**
  Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").
- **statusText**
  Returns the status as a string (e.g., "Not Found" or "OK").

12 Explain XMLHttpRequest objects.

- **new XMLHttpRequest():** It creates a new XMLHttpRequest object.
- **abort():** This method will cancel the current request to exchange data.
- **getAllResponseHeaders():** It will return a set of HTTP headers in form of a string.
- **getResponseHeader():** It will return the specified HTTP header information.
- **open(method, URL, async, userName, password):** This method specifies the method, URL, and other parameters of a request. In this function call, the method parameter defines the operation that has to be performed on the data like **GET, POST, HEAD,** and some other HTTP methods such as **PUT, DELETE.** The **async** parameter specifies the asynchronous behavior of the request. It holds two values *true* and *false***.** If the value is *true* then the script processing will continue after **send()** method without waiting for the response while *false* value means that the script will wait for a response before processing it.
- **send():** It will send the request to the server for data exchange. To GET the requests, **send()** is used.
- **send(string):** It also sends the request to the server for data exchange, but It is used to POST the requests.
- **setRequestHeader():** It will add the label and value pair to the header that has to be sent.

13. Write difference between Synchronous and Asynchronous web application.

Difference between asynchronous and synchronous data transmission

Sr.no	asynchronous	synchronous
1	Transfer one character at a time	Transfer a group or block of character at a time
2	Low data transfer rate	High. Data transfer rate
3	Sysnchronous character is not used.	Sysnchronous character is used.
4	Start and stop bit is added in each character	No. Start and stop bit is added in each character

14 Explain important concept, features and working of Node JS.

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009 and its latest version is v0.10.36. The definition of Node.js as supplied by its official documentation is as follows –

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

```
Node.js = Runtime Environment + JavaScript Library
```

Features of Node.js

Following are some of the important features that make Node.js the first choice of software architects.

- **Asynchronous and Event Driven** – All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
- **Very Fast** – Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- **Single Threaded but Highly Scalable** – Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
- **No Buffering** – Node.js applications never buffer any data. These applications simply output the data in chunks.
- **License** – Node.js is released under the MIT license

15 What is working of http module explain in detail.

World Wide Web Communication

The World Wide Web is about communication between web **clients** and web **servers**.

Clients are often browsers (Chrome, Edge, Safari), but they can be any type of program or device.

Servers are most often computers in the cloud.

Web Client

Cloud

Web Server

HTTP Request / Response

Communication between clients and servers is done by **requests** and **responses**:

1. A client (a browser) sends an **HTTP request** to the web
2. A web server receives the request
3. The server runs an application to process the request
4. The server returns an **HTTP response** (output) to the browser
5. The client (the browser) receives the response

The HTTP Request Circle

A typical HTTP request / response circle:

1. The browser requests an HTML page. The server returns an HTML file.
2. The browser requests a style sheet. The server returns a CSS file.
3. The browser requests an JPG image. The server returns a JPG file.
4. The browser requests JavaScript code. The server returns a JS file
5. The browser requests data. The server returns data (in XML or JSON).

XHR - XML Http Request

All browsers have a built-in **XMLHttpRequest Object (XHR)**.

XHR is a JavaScript object that is used to transfer data between a web browser and a web server.

XHR is often used to request and receive data for the purpose of modifying a web page.

Despite the XML and Http in the name, XHR is used with other protocols than HTTP, and the data can be of many different types like HTML, CSS, XML, JSON, and plain text.

The XHR Object is a **Web Developers Dream**, because you can:

- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background

The XHR Object is the underlying concept of **AJAX** and **JSON**: