1. What is CSS? Explain Different types of CSS

- CSS stand for Cascading Style Sheets.
- CSS, fondly referred to as CSS is a simple design language intended to simplify the process of making web pages presentable.
- CSS handles the look and feel part of a web page.
- Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, as well as a variety of other effects.
- CSS is easy to learn and understand but it provides a powerful control over the presentation of HTML document.
- Most commonly, CSS is combined with the markup languages HTML or XHTML.

- CSS can be added to HTML documents in 3 ways:

1) Inline - by using the style attribute inside

HTML elements.

2) Internal by using a <style> element in the <head> section.

3) External - by using <link> element to link to an external css file.

1) Inline CSS

- An inline css is used to apply a unique style to a single HTML element.
- An inline css uses the style attribute of a HTML element.
- The following example sets the text color of the <h1> element to blue, and the text color of the <p> element to red.

```
<!DOCTYPE html>
<html>
<body>
<h1 style="color:blue;">A Blue Heading</h1>
<p style="color:red;">A red paragraph.</p>
</body>
</html>
```

2) Internal CSS

- An internal css is used to define a style for a single HTML page.
- An internal css is defined in the <head> section of an HTML page, within a <style> element.
- The following example sets the text color of ALL the <h1> elements (on the page) to blue,

and the text color of ALL the <p> elements to red. In addition, the page will be displayed with a "powderblue" background color:

```
<html>
<head>
<style>
body {background-color : powderblue ;}
h1 { color : blue ; }
p { color : red ; }
</style>
</head>
<body>
<h1>This is a heading </h1>
<p> This is a paragraph </p>
</body>
</html>
```

3) External css

- An external style sheet is used to define the style for many HTML pages.
- To use an external style sheet, add a link to it in the <head> section of each HTML page.

```
<html>
<head>
<link rel = "stylesheet " href = " styles.css">
</head>
<body>
<h1> This is a heading </h1>
<p> This is a paragraph. </p>
```

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

- The external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a css extension.

Here is what the "styles.css" file looks like:

"styles.css"

```
body { background-color: powderblue; }
h1 { color : blue; }
P { color : red; }
```

3. Explain Different Module in Node.js

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

Modules ~~are~~ are of three types:

- Core Modules
- local Modules
- Third-party Modules

1) Core Modules :

- Node.js has many built-in modules that are part of the platform and comes with Node.js installation.
- These modules can be loaded into the program by using the require function.

Syntax :

```
var 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.

```
var http = require ('http');
http.createServer (function (req, res)
{
  res.writeHead (200, {'content-Type' : 'text/html'});
  res.write ('Welcome to this page!');
  res.end ();
}).lesten (3000);
```

- In the above example, the require () function returns an object because the Http module returns its functionality as an object.
  The function http.createServer() method will be executed when someone tries to access the computer on port 3000.
- The res.writeHead () method is the stuts code where 200 means it is: ~~ok~~ OK, while the second

argument is an object containing the response headers.

The following list contains some of the important core modules in Node.js :

Core Modules	Description
1) http	Creates an HTTP server in Node.js.
2) assert	Set of assertion functions useful for testing.
3) fs	used to handle file system.
4) path	includes methods to deal with file paths.
5) process	provide's information and control about the current Node.js process.
6) os	provides information about the operating system.
7) querystring	utility used for parsing & formatting URL query strings.
8) url	module provides utilities for URL resolution and parsing.

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

Filename : calc.js

```
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.

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

3) Third-part 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

4) Explain various jquery Effects methods ?

:-

JQuery Effects

- Hide and show
- Fading
- Sliding
- Animation
- Stop
- Callback
- Chaining

1) Hide and Show effect

jQuery hide() and show()

Examples

jQuery hide()

- Demonstrates a simple jQuery hide() method.
- Another hide() demonstration. How to hide parts of text.

Syntax :

```
$(selector).hide(speed, callback);
$(selector).show(speed, callback);
```

- The optional speed parameter specifies the speed of the hiding/showing and can take the following values: "slow", "fast", or milliseconds
- The optional speed parameter is a function to be executed after the hide() or show() method completes (you will learn more about callback functions in a later chapter).
- With jQuery, you can hide and show HTML elements with the hide() and show() methods:

ex:

```
$("#hide").click(function(){
  $("p").hide();
});
```

2) Fading effect

With jQuery you can fade an element in and out of visibility.

jQuery has the following fade methods.

- fadeIn()
- fadeOut()
- fadeToggle()
- fadeTo()

Syntax:

```
$(selector).fadeIn(speed,callback);
```

- The optional speed parameter specifies the duration of the effect.
- It can take the following values: "slow", "fast", or milliseconds.

- The optional callback parameter is a function to be executed after the fading completes.
- The following example demonstrates the fadeIn() method with different parameters :

ex:

```
$("button").click(function (){
 $("#div1").fadeIn();
 $("#div2").fadeIn("slow");
 $("#div3").fadeIn(3000);
```

3) jQuery Slide effect

- With jQuery you can create a sliding effect on elements.
- jQuery has the following slide methods:
  - SlideDown()
  - SlideUp()
  - SlideToggle()

Syntax:

```
$(selector).slideDown(speed,callback);
```

ex.

```
$("#flip").click(function (){
 $("#panel").slideDown();
 });
```

4) jQuery Animations effect

- The jQuery animate() method is used to create custom animations.

Syntax:

```
$(selector).animate({params}, speed, callback);
```

The required params parameter defines the CSS properties to be animated.

ex:

```
$("button").click(function(){
  $("div").animate({left:'250px'});
});
```

5) stop effect

- The jQuery stop() method is used to stop an animation or effect before it is finished.
- The stop() method works for all jQuery effect functions, including sliding, fading and custom animations.

Syntax:

```
$(selector).stop(stopAll, goToEnd);
```

ex.

```
$("button").click(function(){
  $("div").stop();
});
```

6) callback effect

- JavaScript statements ~~are~~ are executed line by line.
- However, with effects, the next line of code can be run even though the effect is not

finished. This can create errors.
- To prevent this, you can create a callback function.
- A callback function is executed after the current effect is finished.

Syntax:

```
$(selector).hide(speed,callback);
```

Example with callback

```
$("button").click(function(){
    $("p").hide("slow",function(){
        alert("The paragraph is now hidden");
    });
});
```

The example has a callback parameter that is a function that will be executed after the hide effect is completed:

Example without callback

The example below has no callback parameter and the alert box will be displayed before the hide effect is completed:

```
$("button").click(function(){
  $("p").hide(1000);
alert("The paragraph is now hidden");
});
```

7) chaining effect

- chaining allows us to run multiple jQuery methods (on the same element) within a single statement.
- Until now we have been writing jQuery statements one at a time (one after the other).
- However, there is a technique called chaining, that allows us to run multiple jQuery ~~com~~ commands, one ~~other~~ after the other, on the same element.
- To chain an action, you simply append the action to the previous action.
- The following example chains together the css(), slideUp(), and slideDown() methods. The "p" element first changes to red, then it slides up and then it ~~sl~~ slides down:

```
$("button").click("color","red").SlideUp
(2000).slideDown(2000);
});
});
```

5. What is new media tag in HTML 5?

HTML5 Media Tags.

<audio>, <video>, <embed>, <source>, <track>

1) <audio>

- HTML audio tag is used to define sounds such as music and other audio clips. Currently there are three supported file formate for HTML5

audio tag.

- The HTML <audio> element is used to play an audio file on a web page.
- The controls attribute adds audio controls, like play, pause and volume.
- The <source> element allows you to specify alternative audio files which the.
- Browser may choose from The browser will used the first recognized format.
- The text between the <audio> and </audio> tags will only be displayed in
- HTML5 supports <video> and <audio> controls. The flash, silverlight and similar technologies are used to play the multi media items.

ex:
```
<audio controls autoplay loop>
<source src="horse.ogg" type="audio/ogg">
<source src="horse.mp3" type="audio/mpeg">
</audio>
```

- Attributes of HTML Audio Tag

1) Controls, 2) src, 3) loop, 4) preload, 5) muted
6) autoplay : It specifies that the audio will start playing as soon as it is ready.

2) HTML 5 <video>

- HTML5 supports <video> tag also.
- The HTML video tag is used for streaming video files such as a movie clip, song clip onthe web page.

Attributes of HTML Video Tag

- Controls: It defines the video controls which is displayed with play/pause buttons.

- height : It is used to set ~~to~~ the height of the ~~the~~ video player.
- width : It is used to set the width of the video player.
- poster : It specifies the image which is displayed on the screen when the video is not played.
- autoplay : It specifies that the video will start playing as soon as it is ready.
- loop : It specifies that the video file will start over again, every time when it is completed.
- muted : It is used to mute the video output.
- preload : It specifies the author view to upload video file when the page loads.
- src : It specifies the source URL of the video file.

ex: <video width = "320" ~~hee~~ height = "240" controls autoplay loop>
<source src = "movie.mp4" type = "video/mp4">
</video>

3) <embed> Tag

The <embed> tag defines a container for an external resource, such as a web page, a picture, a media player or a plug-in application.

Attribute ~~Height~~	Value	Description
height	pixels	specifies the height of the embedded content.
src	URL	specifies the address of the external file to embed.
type	media-type	specifies the media type of the embedded content.
width	pixels	specifies the width of the embedded content.

- Global Attributes
  The <embed> tag also supports the Global Attributes in HTML.

- Event Attributes
  The <embed> tag also supports the Event Attributes in HTML.

- Default css settings
  Most browsers will display the <embed> element with the following default values:

```
embed : focus {
            outline : none;
        }
```

- An embedded HTML page :

```
<embed type="text/html" src="snippet.html"
    width="500" height="200">
```

4) <source> tag

- The <source> tag is used to specify mutiple media resources for media elements.

- The <source> tag allows you to specify alternative video / audio / image files which the browser may choose from, based on browser support or viewport width. The browser will choose the first <source> it supports.

Attribute	Value	Description
• media	media-query	Accepts any valid media query that would normally be defined in a CSS specifies
• size		specifies image sizes for different page layouts.
• src		
• srcset	URL	Required when <source> is used in <picture>. Specifies the URL of the image to use in different situations.
• type	MIME-type	specifies the MIME-type of the resource.

- Global Attributes
  The <source> tag also supports the Global Attributes in HTML.
- Event Attributes
  The <source> tag also supports the event Attributes.

ex: <video width="320" height="240" controls>
<source src="movie.mp4" type="video/mp4">
</video>

5) <track>

- The <track> tag specifies text tracks for <audio> or <video> elements.
- This element is used to specify subtitles, caption files or other files containing text, that should be visible when the media is playing.
- Tracks are formatted in web VTT format (.vtt files).

Optional Attributes.

Attribute	Value	Description
default	default	specifies that the track is to be enabled if the user's preferences do not indicate that another track would be more appropriate.
kind	captions chapters descriptions metadata subtitles	Specifies the kind of text track.
label	text	Specifies the title of the text track.
src	URL	Required specifies the URL of the track file.
srclang	language-code	Specifies the language of the track text data (required if kind="subtitles")

- The <track> tag also supports the Global Attributes.
- The <track> tag also supports the Event Attributes.

ex.

```
<video width = "320" height = "240" controls>
  src = "forrest_gump.mp4" type = "video/mp4">
<source
  src = "forrest_gump.ogg" type = "video/ogg">
<track
  src = "fgsubtitles_en.vtt" kind = "subtitles"
srclang = "en" label = "English">
<track
  src = "fgsubtitle_no.vtt" kind = "subtitles"
  srclang = "no" label = "Norwegian">
</video>.
```

6. Explain different 10 properties in css.

—

- The Display property is css defines how the components (div, hyperlink, heading, etc.) are going to be placed on the web page.
- As the name suggests, this property is used to define the display of the different parts of a web page.

1) Background Properties

- The css background properties are used to add background effects for elements.
- In these chapters you will learn about the following css background properties:

i) background - color
ii) background - image
iii) background - repeat
iv) background - attachment

v) background - position
vi) background (shorthand property)

ex: div {
background-color : green;
opacity : 0.3;
} [opacity / ∓ Transparency : The opacity property specifies the opacity/Transparency of an element. It can take a value from 0.0-1.0. The lower value, the ~~not Transp~~ not Transparent:]

2) Block Properties
- The display : inline-block value.
- Compared to display : inline, the major difference is that display: inline_block allows to set a width and height on the element.
- Also, with display : inline-block, the top and ~~bool~~ bottom margins/paddings are respected, but with display : inline they are not.
- Compared to display : block, the major difference is that display : inline-block does not add a line-break after the element, so the element can sit next to other elements.

3) CSS Colors
i) CSS Background Color : properly specifies the bgcolor of an element.
Syntax:
<h1 style = "background-color: DodgerBlue;">
Hello World </h1>

- ? Explain different

v) background - position
vi) background (shorthand property)
ex: div {
background-color : green;
opacity : 0.3;
} [opacity / Transparency : The opacity property specifies the opacity/Transparency of an element. It can take a value from 0.0 - 1.0. The lower value, the ~~more Transp~~ more Transparent.]

2) Block Properties
- The display : inline-block value.
- Compared to display : inline, the major difference is that display : inline_block allows to set a width and height on the element.
- Also, with display : inline-block, the top and ~~bool~~ bottom margins / paddings are respected, but with display : inline they are not.
- Compared to display : block, the major difference is that display : inline-block does not add a line-break after the element, so the element can sit next to other elements.

3) CSS Colors
i) CSS Background Color : properly specifies the bgcolor of an element.
Syntax:
<h1 style = "background-color: DodgerBlue;">
Hello World </h1>

ii) CSS Text color: property specifies the font color of an element.

Syntax:

```
<h1 style = " color : Tomato; " > Hello. World </h1>
```

iii) CSS Border color:

property specifies the border color and other decorative effects.

We can define the color of an element by using the following ways:

- RGB format
- RGBA format
- Hexadecimal notation
- HSL
- HSLA
- Built-in color

4). CSS Borders:

The border-style property specifies what kind of border to display.

The border-style property can have from one to four value (for the top border, right border, bottom border, and the left border.).

The following border-style values are allowed:

1) dotted - Defines a dotted border.
2) dashed - Defines a dashed border.
3) solid -
4) double -
5) groove - Define a 3D grooved border. The effect depends on the bordercolor value.
6) ridge - Defines a 3D ridged border.

ii) CSS Text color: property specifies the font color of an element.

Syntax:

```
<h1 style = "color: Tomato;">Hello.World</h1>
```

iii) CSS Border color:

property specifies the border color and other decorative effects.

We can define the color of an element by using the following ways:

- RGB format
- RGBA format
- Hexadecimal notation.
- HSL
- HSLA
- Built-in color

4). CSS Borders:

The border-style property specifies what kind of border to display.

The border-style property can have from one to four value (for the top border, right border, ~~boo~~ bottom border, and the left border.).

The following border-style values are allowed:

1) dotted - Defines a dotted border.
2) dashed - Defines a dashed border.
3) solid -
4) double -
5) groove - Define a 3D grooved border. The effect depends on the bordercolor value.
6) ridge - Defines a 3D ridged border.

7) inset - Define a 3D inset border.
8) outset - Define a 3D outset border.
9) none - Define no border
10) hidden - Defines a hidden border.

ex:
```
<style> P.mix {border-style : dotted dashed solid
                double ; } </style>
```

5) CSS Border Width :
- The border-width property specifies the width of the four borders.
- The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.

```
p{
   border-~~sty~~style : solid;
   border-width : 5 px ;
   }
```

6) CSS Margins :
- Margins are used to create space around elements, outside of any defined borders.
- With CSS, you have full control over the ~~margins~~ margins. There are properties for setting the margin for each side of an element. (Top, right, bottom and left).
- All the margin properties can have -

i) auto - the browser calculates the margin.
ii) length - specifies a margin in px, pt, cm, etc.
iii) % - specifies a margin in % of the ~~m~~ width

the containing element.

iv) inherit - specifies that the margin should be inherited from the parent element.

Syntax:

1) <style>
div{ border : 1px solid black;
margin : 25 px 50 px 75 px 100 px;
} </style>

2) <style>
div { border : 1px solid black;
margin-top : 25 px;
margin-right : 50 px;
margin-bottom : 75 px;
margin-left : 100 px;
</style>

7) css Padding :

- Padding is used to create space around an elements content, inside of any defined borders.
- With css, you have full control over the padding.
- There are properties for setting the padd for each side of an element (Top, right, bot and left).
- All the padding properties can have the -

1) length - specifies a padding in px, pt, cm, etc.
2) % - specifies a padding in % of the width of the containing element.

3) inherit - specifies that the padding should be inherited from the parent element.

syntax :

```
<style>
div { border : 1px solid black;
• Padding : 25 px 50px 75 px 100px;
~~padd~~
Padding-Top : 25 px;
Padding-right : 50 px;
Padding-bottom: 75 px;
Padding-left : 100 px;
}
</style>
```

8) CSS Height and Width:

- The CSS height and width properties are used to set the height and width of an element.
- The css max-width property is used to set the maximum width of an element.
- The height and width properties do not include padding, borders, or margins.
- It sets the height and width of the area inside the padding, border, and margin of the element.
- The height and width properties may have-

i) auto - This is default. The browser calculates the height and width.

ii) length - Defines the height / width in px, cm, etc.

iii) % - Defines the % of the containing block.

iv) initial - sets the height / width to its default value.

v) inherit - The height/width will be inherited from its parent value.

syntax :

```
<style>
 div {height : 200 px;
      width  = 50%;
     }
</style>
```

9) CSS Lists :

In HTML, there are two main types of lists :

1) unordered lists (<ul>) - the list items are marked with bullets
2) ordered lists (<ol>) - the list items are marked with numbers or letters.

- The CSS list properties allow you to :

1) Set different list item markers for ordered lists
2) Set different list item markers for unordered lists
3) Set an image as the list item marker.
4) Add background colors to lists and list items.
5) The list-style-type property specifies the type of list item marker.
6) The some of the available list item markers have -
   1) circle
   2) square
   3) upper-roman
   4) lower-alpha

- Position the List Item Markers

1) The list-style-position property specifies the position of the list-item markers (bullet points).
2) "List-style-position: outside;" means that the bullet points will be outside the list item.
3) The start of each line of a list item will be aligned vertically. This is default.
4) "List-style-position: Inside;" means that the bullet points will be inside the list item.
5) As it is part of the list item, it will be part of the text and push the text at the start;

- List with colors: /Styling List with colors:

1) We can also style lists with colors, to make them look a little more interesting.
2) Anything added to the <ol> or <ul> tag, affects the entire list, while properties.
3) Added to the <li> tag will affect the individual list items:

Syntax:

```
<style>
    ul.a {list-style-type : square;
          list-style-image : url('sqpurple.gif');

          list-style-position : outside;
    //    list-style-position : inside;
    // list-style : square inside url("sqpurple.gif");

    // background : #ff9999;
    // padding : 20px;
    }
</style>
```

10) CSS Tables

1) Table Borders:

- To specify table borders is CSS, use the border property.
- The example below specifies a black border for <table>, <th>, and <td> elements:

```
table, th, td {
    border: 1px solid black;
}
```

2) Full-Width Table:

- The table above might seem small in some cases.
- If you need a table that should span the entire screen (full-width), add width: 100% to the <table> element.

```
table { width: 100%; }
```

3) Table Width and Height:

- The width and height of a table are defined by width and height properties.
- The example below sets the width of the table and the height of the <th> elements to 70px:

```
table { width: 100%; }
  th { height: 70px }
```

4) CSS Table Alignment :
i) Horizontal Alignment.
ii) Vertical Alignment.

5) Table Padding
6) Hoverable Table
7) Table color
8) Responsive Table.

8. Explain What is Bootstrap? How to responsive website using bootstrap?

- Bootstrap

1) Bootstrap is a free front-end framework for faster and easier web delopment.
2) Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other as well as optional Javascript plugins.
3) Bootstrap also gives you the ability to easily create responsive designs.

Responsive website using bootstrap.

i) Responsive web design makes your web page look good on all devices.
ii) Responsive web design is not a program or a JavaScript.
iii) Responsive web design uses only HTML and CSS.
iv) Designing for the Best Experience for All Users.
v) Web pages can be viewed using many different devices : desktop, tablets and phone, your web page should look good and be easy to use.

regardless of the device.

vi) Web pages should not leave out information to fit smaller devices, but rather adapt its content to fit any device:

Desktop, Tablet, Phone

vii) It is called responsive web design when you use CSS and HTML to resize, hide, shrink, enlarge or move the content to make it look good on any screen.

9. What NodeJs & Explain Advantages & Disadvantages of NodeJs!

- Node.Js

Node.js an open-source and cross-platform runtime environment for executing JavaScript code outside a browser. You need to remember that Node.Js is not a framework and it's not a programming language. Most people are confused and understand it's a framework or a programming language. We often use Node.js for building back-end services like APIs like Web App or Mobile App. It's used in production by large companies such as Paypal, Uber, Netflix, Walmart and so on.

* Features of NodeJs:

i) There are other programming languages also which we can use to build back-end services so what makes Node.js different I am going

regardless of the device.

vi) Web pages should not leave out information to fit smaller devices, but rather adapt its content to fit any device:

Desktop, Tablet, Phone

vii) It is called responsive web design when you use CSS and HTML to resize, hide, shrink, enlarge or move the content to make it look good on any screen.

9. What NodeJs & Explain Advantages & Disadvantages of NodeJs:

- Node.Js

Node.js an open-source and cross-platform runtime environment for executing JavaScript code outside a browser. You need to remember that Node.Js is not a framework and it's not a programming language. Most people are confused and understand it's a framework or a programming language. We often use Node.js for building back-end services like APIs like Web App or Mobile App. It's used in production by large companies such as Paypal, Uber, Netflix, Walmart and so on.

* Features of NodeJs:

i) There are other programming languages also which we can use to build back-end services what makes Node.js different I am

to explain.

ii) It's easy to get started and can be used for prototyping and agile dep development.

iii) It provides fast and highly scalable services.

iv) It uses JavaScript everywhere so, it's easy for a JavaScript programmer to build back-end services using Node.js.

v) Source code cleaner and consistent.

vi) Large ecosystem for open source library.

vii) It has Asynchronous or Non-blocking nature.

Advantages of Node JS : Here are the ~~befefi~~ benefits of using Node.js

1) Easy Scalability : Developers prefer to use Node.js because it is easily scaling the application in both horizontal and vertical directions. We can also add extra resources during the scalability of the application.

2) Real-time web apps : If you are building a web app you can also use PHP and it will take the same amount of time when you use Node.js, But if I am talking about building chat apps or gamming apps Node.js is much more preferable because of faster synchronization. Also, the event loop avoids HTTP overloaded for Node.js development.

3) Fast suite : NodeJs runs on the V8 engine developed by Google. Event loop in NodeJS handles all synchronous operation so NodeJS acts

like a fast suite and all the operations can be done quickly like reading or writing in the database, network connection, or file system.

4) Easy to learn and code: NodeJs is easy to learn and code because it uses Javascript. If you are a front-end developer and have a good grasp of Javascript you can easily learn and build the application on NodeJs.

5) Advantage of caching : It provides the ~~cat~~ caching of a single module. Whenever there is any request for the first module, it gets cached in the ~~ex~~ application memory, so you don't need to re-execute the code.

6) Data Streaming : In NodeJs HTTP request and response are considered as two separate events. They are data stream so when you process a file at the time of loading it will reduce the overall time and will make it faster when the data presented in the form of transmissions. It also allows you to stream audio and video files at lightning speed.

7) Hosting : Paas (Platform as a service) and Heroku are the hosting platforms for NodeJs application deployment which is easy to use without facing any issue.

8) Corporate Support : Most of the well-known companies like walmart, Paypal, Microsoft, Yahoo are using NodeJs for building the applications. NodeJs uses JavaScript, so most of the companies are combining front-end and backend Teams together into a single unit.

* Disadvantages of Node.js

1) Reduce Performance when handling heavy computing tasks.

2) Node.js invites a lot of code changes due to unstable API.

3) ~~Asy~~ Asynchronous Programming model makes it difficult to maintain code.

4) High demand with a few experienced node.js developer.

5) Choose wisely - lack of library support can endanger your code.

7. What is the difference between Get and Post method?

Get	Post
1) It is default method.	1) It is not default method.
2) It is designed to Get data from the server.	2) It is designed to send data to the server.
3) It supports catching.	3) It does not support catching.
4) Get request can be bookmarked.	4) Post request can not be bookmarked.
5) Get request is more efficient & used more than post.	5) Post request is less efficient & used less than get.
6) Get allows only ASCII data.	6) Post has no restrictions, it allows binary data as well.
7) Normally get is used to retrieve data	7) Post is mainly used to data insertion & data updates.
8) Data is visible to everyone in the URL.	8) Data is not displayed in the URL.
9) In case of get request, only limited amound of data can be sent because data is sent in header.	9) In case of post request, large amount of data can be sent because data is sent in body.

10) It can be store data 2000 character only.	10) It can store data up to 8 ~~m~~ Mb.

10. Explain various Datemethods and string methods in JavaScript.

- The JavaScript string is an object that represents a sequence of characters.

There are 2 ways to create, string in JavaScript.

1) By string literal.
2) By string object (using new keyword).

1) By string literal.

The string literal is created using ~~"~~ double quotes.

1. var stringname = "string value";

```
<!Doctype html>
<html>
  <head> </head>
    <body>
    < script >
var str= "This is string literal";
  document.write (str);
   </script>
   </body>
   </html>
```

2) By string object (using new keyword)

The syntax of creating string object using new keyword is given below;

1. var stringname = new string ("string literal");

Here, new keyword is used to create instance of string.

```
<!Doctype html>
<html>
<body>
<script>
var stringname = new string ("hello JavaScript string");
document.write(stringname);
</script>
</body>
</html>
```

* JavaScript string Method.

1) char At ()
It provides the char value present at the specified index.

2) char Code At ()
It provides the Unicode value of a character present at the specified index.

3) Concat ()
It provides a combination of two or more strings.

4) index of ()
It provides the position of a char value present in the given string.

5) lostIndex of ()
It provides the position of a char value present in the given string by searching a character from the lost position.

6) Search ()
It searches a specified regular expression in a given string and returns its position if a match occurs.

7) Match ()
It searches a specified regular expression in a given string and returns that regular expression if a match occurs.

8) replace ()
It replaces a given string with the specified replacement.

9) Substr ()
It is used to fetch the part of the given string on the basis of the specified starting position and length.

10) Substring ()
It is used to fetch the part of the given string on the basis of the specified index.

11) toLowerCase ()
It converts the given string into lowercase letter.

5) lastIndex of ()
It provides the position of a char value present in the given string by searching a character from the last position.

6) search ()
It searches a specified regular expression in a given string and returns its position if a match occurs.

7) Match ()
It searches a specified regular expression in a given string and returns that regular expression if a match occurs.

8) replace ()
It replaces a given string with the specified replacement.

9) substr ()
It is used to fetch the part of the given string on the basis of the specified starting position and length.

10) Substring ()
It is used to fetch the part of the given string on the basis of the specified index.

11) toLowerCase ()
It converts the given string into lowercase letter.

12) toLocaleLowerCase()
It converts the given string into lowercase letter on the basic of host's current locale.

13) toUpperCase()
It converts the given string into uppercase letter.

14) toString()
It provides a string representing the particular object.

15) Split()
Split string into substring array, then returns the newly created array.

16) trim()
white space.

* JavaScript Date Methods.
The javaScript Date object can be used to get year, month and day. You can display a timer on the webpage by the help of javascript date object.

1) getDate()
It returns the integer value between 1 & 31 that represents the day for the specified date on the basis of local time.

2) getDay()
It returns the integer value between 0 and

12) toLocaleLowerCase()
It converts the given string into lowercase letters on the basic of host's current locale.

13) toUpperCase()
It converts the given string into uppercase letters.

14) toString()
It provides a string representing the particular object.

15) Split()
Split string into substring array, then returns the newly created array.

16) trim()
white space.

* JavaScript Date Methods.
The javascript Date object can be used to get year, month and day. You can display a timer on the webpage by the help of javascript date object.

1) get date()
It returns the integer value between 1 & 31 that represents the day for the specified date on the basis of local time.

2) getDay()
It returns the integer value between 0 and

that represents the day of the week on the basis of Local time.

3) getFullyears()
It returns the integer value that represents the year on the basis of Local time.

4) getHours ()
It returns the integer value between 0 and 23 that represents the hours on the basis of Local time.

5) getMilliseconds ()
It returns the integer value between 0 and 999 that represents the milliseconds on the basis of local time.

6) getMinutes ()
It returns the integer value between 0 and 59 that represents the minutes on the basis of local time.

7) getMonth ()
It returns the integer value between 0 and 11 that represents the month on the basis of local time.

8) getSeconds ()
It returns the integer value between 0 and 60, that represents the seconds on the basis of local time.

9) getUTCDate()
It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of universal time.

10) getUTCDay()
It returns the integer value between 0 and 6 that represents the day of the week on the basis of universal time.

11) getUTCFullYear()
It returns the integer value that represents the year on the basis of universal time.

12) getUTCHours()
0 & 23 that represents the hours on the basis universal time.

13) getUTCMinutes()
0 & 59 minutes on the basic of universal time

14) getUTCSeconds()
0 & 60 that represents the seconds on the basis of universal time.

15) setDate()
it set the date value for the specified data on the basis of Local time.

16) setDay()
set the particular day of the week on the basis of local time.

17) setFullyears()
sets the year value for the specified date on the basis of local time.

18) setHours()
It sets the hour value for the specified date on the basis of local time.

19) toString - It returns the date in the form of string.

20) Value of :
It returns the Primitive value of Date Object.

21) toDateString : It return the date in the form ISO format string.

2. Explain in brief Rounting in Angular js.

1) If you want to navigate to diff^{n} pages in your application, but you also want to the application to be a single page application with no page reloading you can use the ngRoute mocle.
2) The ngRoute module routes your application to diff^{n} pages without reloading the entire application.
3) The ngRoute module helps your application to become a single page application.

4) $routeProvider is used to configure the routes it helps to define what page to display when a user clicks a links, it accepts either when() or otherwise() method.

5) The ngRoute must be added as a dependant in the application module.

12. How to create from using html.

—

```
<!Doctype html>
<html>
<head>
<title> Form in HTML </title>
</head>
<body>
<h2> Registration Form </h2>
<form>
<fieldset>
<legend> User Personal information </legend>
<label> Enter Your Full name </label> <br>
<input type ="text" name ="name"> <br>
<label> Enter your email </label> <br>
<input type ="email" name= "email"> <br>
<label> Enter your password </label> <br>
<input type = "Password" name= "Pass"> <br>
<br> <label> Enter your gender </label> <br>
<input type = "radio" id = "gender" name= "gender
value = "male" /> Male <br>
<input type= "radio" id= "gender" name = "gender"
value = "Female" /> Female <br />
<input type = "radio" id = "gender" name= "gender"
```

```
value="others"/> others <br/>
<br> Enter your address <br>
<textarea> </textarea> <br>
<input type="submit" value="sign-up">
</fieldset>
</form>
</body>
</html>
```

Registration Form

User personal information

Enter your full name

Enter your email

Enter your password

Confirm your password

Enter your gender

- Male
- Female
- Other

Enter your address :

[ ]

[sign-up]

13. Explain what is Bootstrap Grid System ?

—

Bootstrap Grid System.

1) Bootstrap's grid system allows up to 12 columns across the page.
2) If you do not want to use all 12 columns individually, you can group the colums together to create wider columns.
3) Bootstrap's grid system is responsive the columns will re-arrange depending on the screen size on a big screen it might look ~~beet~~ better with the content organized in three columns, but on a small screen it would be better it the content items were stacked on top of each other.

* Bootstrap 4 grid system has 5 classes.

1) • col (extra small devices - screen width < 576 px)
2) • col-sm- (small devices - screen width => 576 px)
3) • col-md- (medium devices - 11 => 768 px)
4) • col-lg- (large devices - 11 => 992 px)
5) • col-xl- (xlarge devices 11 => 1200 px)

* Grid System Rules

1) Rows must be placed within a container (fixed-width) or container-fluid (full-width) for proper alignment & padding.

2) Use rows to create horizontal groups of columns.

3) Content should be placed within columns ~~and~~ and only columns may be immediate children of rows.

4) Predefined classes like row and .col-sm-4 are available for quickly making grid system.

5) Column width are in percentage, so they are always fluid & sized relative to their parent element.

15. Explain following term in JavaScript

I) number methods
II) Data types
III) operators
IV) Variables

I) Number methods

1) isfinite() - it determines whether the given value is a finite number.

2) isInteger() - it determines whether the given value on a integer.

3) Parsefloat() - converts the given string into a floating point number.

4) ParseInt() - string into an integer number.

5) tofixed() - It returns the string that represents a number with exact digits after a decimal point.

6) toExponential() - It returns the string that represents exponential notation of the given number.

7) toprecision() - It returns the string that, represents a number of specified precision.

8) tostring() - return the given number in the form to string.

2) Data types

JavaScript provides different data types to hold different types of values. There are 2 types of data types in JavaScript.

1) Primitive data type.
2) Non- Primitive data type.

```
var a=40; // holding number.
var b="Rahul"; // holding string.
```

* JavaScript Primitive data types.

Data type

1) String - represent seq of characters "hello"
2) Number - represent numeric value eg. 100
3) Boolean - represent boolean value either False or true
4) undefind - represent undefind $ value.
5) Null - represent null i.e. no. value at all.

* JavaScript non-primitive data types
Data type.
1) object - represent instance through we can access members.
2) Array - represents group of ~~sim~~ similar values.

Reg Exp - represents regular expression.

3) Operators
JavaScript operators are symbols that are used to perform operations on operands.
1) Arithmetic operators
operation on operands

Operator	Description	eg.
+	Addition	10+20=30
-	Subtraction	20-10 = 10
*	multiplication	10X20= 200
/	Division	20/10 = 2
%	modulus	20%10=0
++	increment	var a=10; a++ => a=11
--	decrement	var a=10; a-- =>a=9

2) Comparision operator
compare 2 operands

operator	Description	eg.
==	is equal to	10==20=> false
===	indentical	10==20 => false
!=	not equal to	10!=20 => True
!==	not identical	20!==20 = False
>	Greater than	20>10 => true

>=	Greater than equal to	20 >= 10 => true
<=	less than equal to	20 <= 10 => false
<	less than	20 < 10 => false

3) Bitwise operator
operations on operands.

Operator	Description	eg.
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==30) = false
^	=||= XOR	(10==20 ^ 20==30) = false
~	=||= NOT	(~10) = -10
<<	-||- left shift	(10<<2) = 400
>>	-||- right shift	(10>>2) = 2
>>>	-||- Right shift	(10>>>2) = 2

4) Logical operator

&&	Logical	AND	(10==20 && 20==33) = false
||	-||-	OR	(10==20 || 20==33) = false
!	-||-	NOT	!(10==20) = true

5) Assignment operators

=	Assign	10 + 10 = 20
+=	Add assign	var a = 10, a += 20 => a = 30
-=		a = 20; a -= 10 => a = 10
*=	multiply ||	var a = 10; a *= 20 => 200
/=	divided ||	a = 10; a /= 2 => 5
%=	modulus ||	a = 10; a %= 2 => a = 0

6) Special operators

(!;) = conditional operator return value
, = multiple expression
delete = delete property from the object
in = checks if object has the given property
instance of = instance give type
new = creates an instance.
type of = checks the type of object.
void = return value
yield = generator interator.

4) Variables - A JavaScript variable is simply a name of storage location. There are 2 types of variables in JS:
local variable & global variables.

* Rules for declaring variables
1) Name must start with a letter [a to z or A to z]
2) Javascript variable case sensitive

1) JavaScript local variable
A JavaScript local variable is declared inside block or Fn it is accessible within the Fn or block only.

```
<script>
Fn abc (){
var x = 10; // local variable
}
</script>
```

6) Special operators

(!;) = conditional operator return value
, = multiple expression
delete = delete property from the object
in = checks if object has the given property
instance of = instance give type
new = creates an instance.
type of = checks the type of object.
void = return value
yield = generator interator.

4) Variables - A JavaScript variable is simply a name of storage location. There are 2 types of variables in JS:
local variable & global variables.

* Rules for declaring variables
1) Name must start with a letter [a to z or A to z]
2) JavaScript variable case sensitive

1) JavaScript local variable
A JavaScript local variable is declared inside block or Fn it is accessible within the Fn or block only.

```
<script>
Fn abc () {
var x = 10; // local variable
}
</script>
```

2) JavaScript global variable

A JavaScript global variable is accessible from only function. A variable i.e. declared outside the fn or declared with window object is known as global variable.

```
<script>
var data=200; // global variable
    function a(){
    }
    </script>
```