****
****

HTML5

****
****

1. What is the relationship between SGML,HTML , XML and XHTML?
SGML (Standard generalized markup language) is a standard which tells how to specify document markup. It's only a Meta language which describes how a document markup should be. HTML is a markup language which is described using SGML.

So by SGML they created DTD which the HTML refers and needs to adhere to the same. So you will always find "DOCTYPE" attribute at the top of HTML page which defines which DTD is used for parsing purpose.

Hide Copy Code
<!--!doctype-->
Now parsing SGML was a pain so they created XML to make things better. XML uses SGML. For example in SGML you have to start and end tags but in XML you can have closing tags which close automatically ("").

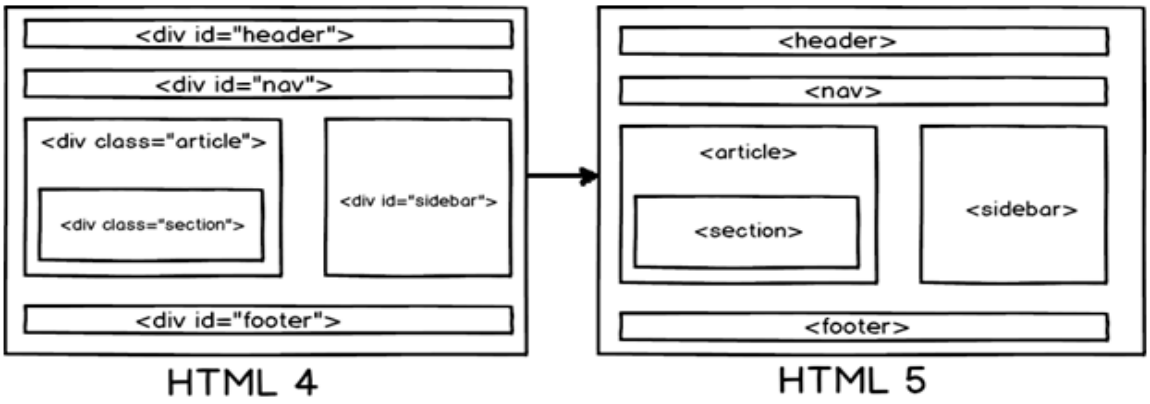
XHTML was created from XML which was used in HTML 4.0. So for example in SGML derived HTML "" is not valid but in XHTML it's valid. You can refer XML DTD as shown in the below code snippet.

2.What is HTML 5?
HTML 5 is a new standard for HTML whose main target is to deliver everything without need to any additional plugins like flash, Silverlight etc. It has everything from animations, videos, rich GUI etc.
HTML5 is cooperation output between World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).

3.How is the page structure of HTML 5 different from HTML 4 or previous HTML?

A typical web page has headers, footers, navigation, central area and side bars. Now if we want to represent the same in HTML 4 with proper names to the HTML section we would probably use a DIV tag.

But in HTML 5 they have made it more clear by creating element names for those sections which makes your HTML more readable.



Below are more details of the HTML 5 element s which form the page structure.

- <header>: Represents header data of HTML.
- <footer>: Footer section of the page.
- <nav>: Navigation elements in the page.
- <article>: Self-contained content.
- <section>: Used inside article to define sections or group content in to sections.
- <aside>: Represent side bar contents of a page.

4. What is datalist in HTML 5?

Datalist element in HTML 5 helps to provide autocomplete feature in a textbox as shown below.

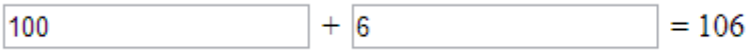
```
<input list="Country">
<datalist id="Country">
<option value="India">
<option value="Italy">
<option value="Iran">
<option value="Israel">
<option value="Indonesia">
</datalist>
```

5. What are the different new form element types in HTML 5?

There are 10 important new form elements introduced in HTML 5:-

Color.
Date
Datetime-local
Email
Time
Url
Range
Telephone
Number
Search

6. What is output element in HTML 5?
Output element is needed when you need calculation from two inputs to be summarized in to a label. For instance you have two textboxes(see the below figure) and you want to add numbers from these textboxes and send them to a label.



Below goes the code of how to use output element with HTML 5.

Hide Copy Code

```
<form onsubmit="return false"  &ouml;ninput="o.value = parseInt(a.value) + parseInt(b.value)">
<input name="a" type="number"> +
<input name="b" type="number"> =
<output name="o" />
</form>
```

You can also replace “parseInt” with “valueAsNumber” for simplicity. You can also use “for” in the output element for more readability.

Hide Copy Code

```
<output name="o" for="a b"></output>
```

7. How can you apply CSS style using ID value?

So let’s say you have a HTML paragraph tag with id “mytext” as shown in the below snippet.

Hide Copy Code

```
<p id="mytext">This is HTML interview questions.</p>
```

You can create a style using “#” selector with the “id” name and apply the CSS value to the paragraph tag. So to apply style to “mytext” element we can use “#mytext” as shown in the below CSS code.

Hide Copy Code

```
<style>
#mytext
```

```
{
background-color:yellow;
}
</style>
```

Quick revision of some important selectors.

1. Set all paragraph tags back ground color to yellow.

Hide Copy Code

```
P,h1
{
background-color:yellow;
}
```

2. Sets all paragraph tags inside div tag to yellow background.

Hide Copy Code

```
div p
{
background-color:yellow;
}
```

3. Sets all paragraph tags following div tags to yellow background.

Hide Copy Code

```
div+p
{
background-color:yellow;
}
```

4. Sets all attribute with “target” to yellow background.

Hide Copy Code

```
a[target]

{

background-color:yellow;

}



<a href="http://www.questpond.com">ASP.NET interview questions</a>
<a href="http://www.questpond.com" target="_blank">c# interview questions</a>
<a href="http://www.questpond.org" target="_top">.NET interview questions with answers</a>
```

5. Set all elements to yellow background when control gets focus.

Hide Copy Code

```
input:focus

{

background-color:yellow;
```

```
}
```

6. Set hyperlinks according to action on links.

Hide Copy Code

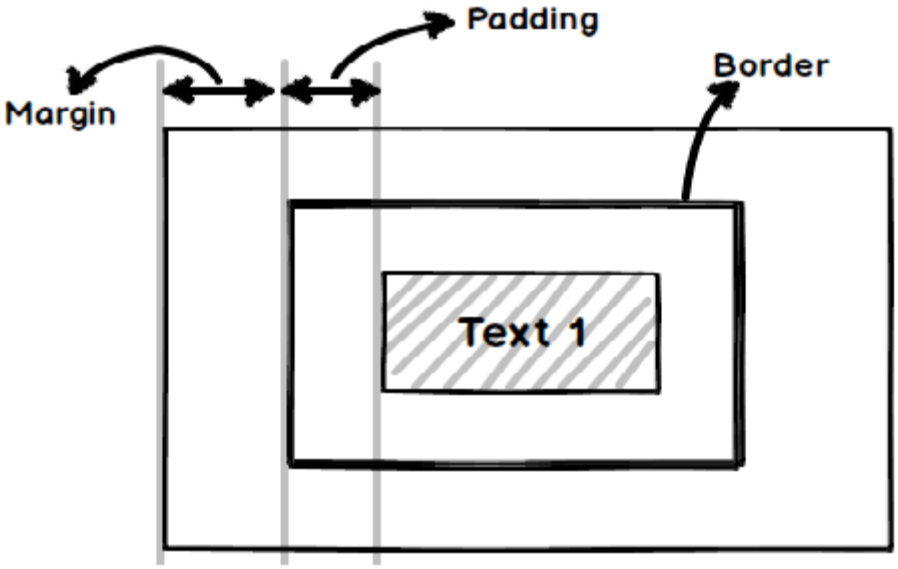
```
a:link    {color:green;}
a:visited {color:green;}
a:hover   {color:red;}
a:active  {color:yellow;}
```

8. CSS Box Model?

CSS box model is a rectangular space around a HTML element which defines border, padding and margin.
Border: - This defines the maximum area in which the element will be contained. We can make the border visible, invisible, define height and width etc.

Padding: - This defines the spacing between border and element.

Margin: - This defines the spacing between border and any neighboring elements.



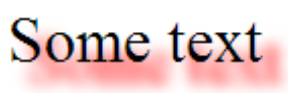
9. Can you explain some text effects in CSS 3?

Here the interviewer is expecting you to answer one of two text effects by CSS.Below are two effects which are worth noting.

Shadow text effect

Hide Copy Code

```
.specialtext
{
text-shadow: 5px 5px 5px #FF0000;
}
```



Word wrap effect

Hide Copy Code

```
<style>
.breakword
{word-wrap:break-word;}
</style>
```

10. What are web workers and why do we need them ?

Web workers at long last bring multi-threading to JavaScript.

A web worker is a script that runs in the background (i.e., in another thread) without the page needing to wait for it to complete. The user can continue to interact with the page while the web worker runs in the background. Workers utilize thread-like message passing to achieve parallelism.

Consider the below heavy for loop code which runs above million times.

Hide Copy Code

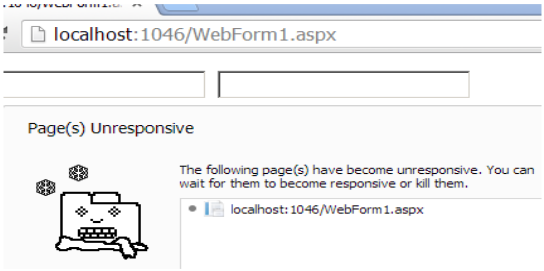
```
function  SomeHeavyFunction()
{
for (i = 0; i < 10000000000000; i++)
{
x = i + x;
}
}
```

Let’s say the above for loop code is executed on a HTML button click. Now this method execution is synchronous. In other words the complete browser will wait until the for loop completes.

Hide Copy Code

```
<input type="button" onclick="SomeHeavyFunction();" />
```

This can further lead to browser getting freezed and unresponsive with an error message as shown in the screen below.



So if we can move this heavy for loop in a JavaScript file and run it asynchronously that means the browser does need to wait for the loop then we can have a more responsive browser. That's what web worker are for.

Web worker helps to execute JavaScript file asynchronously.

11. What is local storage concept in HTML 5?

Many times we would like to store information about the user locally in the computer. For example let's say user has half-filled a long form and suddenly the internet connection breaks off. So the user would like you to store this information locally and when the internet comes back.He would like to get that information and send it to the server for storage.

Modern browsers have storage called as "Local storage" in which you can store this information.

12. How can we add and remove data from local storage?

Data is added to local storage using "key" and "value". Below sample code shows country data "India" added with key value "Key001".

Hide Copy Code

```
localStorage.setItem(&ldquo;Key001&rdquo;,&rdquo;India&rdquo;);
```

To retrieve data from local storage we need to use "getItem" providing the key name.

Hide Copy Code

```
var country = localStorage.getItem(&ldquo;Key001&rdquo;);
```

You can also store JavaScript object's in the local storage using the below code.

Hide Copy Code

```
var country = {};
country.name = &ldquo;India&rdquo;;
country.code = &ldquo;I001&rdquo;;
localStorage.setItem(&ldquo;I001&rdquo;, country);
var country1 = localStorage.getItem(&ldquo;I001&rdquo;);
```

If you want to store in JSON format you can use "JSON.stringify" function as shown in the below code.

Hide Copy Code

```
localStorage.setItem(&ldquo;I001&rdquo;,JSON.stringify(country));
```

12. What is the lifetime of local storage?

Local storage does not have a life time it will stay until either the user clear it from the browser or you remove it using JavaScript code.

13. What is the difference between local storage and cookies?

	Cookies	**Local storage**
Client side / Server side.	Data accessible both at client side and server side. Cookie data is sent to the server side with every request.	Data is accessible only at the local browser side. Server cannot access local storage until deliberately sent to the server via POST or GET.
Size	4095 bytes per cookie.	5 MB per domain.
Expiration	Cookies have expiration attached to it. So after that expiration the cookie and the cookie data get's deleted.	There is no expiration data. Either the end user needs to delete it from the browser or programmatically using JavaScript we need to remove the same.

14. What is WebSQL?

WebSQL is a structured relational database at the client browser side. It's a local RDBMS inside the browser on which you can fire SQL queries.

15. What were some of the key goals and motivations for the HTML5 specification?

HTML5 was designed to replace both HTML 4, XHTML, and the HTML DOM Level 2.
Major goals of the HTML specification were to:

- Deliver rich content (graphics, movies, etc.) without the need for additional plugins (e.g., Flash).
- Provide better semantic support for web page structure through the introduction of new structural element tags.
- Provide a stricter parsing standard to simplify error handling, ensure more consistent cross-browser behavior, and simplify backward compatibility with documents written to older standards.
- Provide better cross-platform support (i.e., to work well whether running on a PC, Tablet, or Smartphone).

16. What are some of the key new features in HTML5?

Key new features of HTML5 include:

- Improved support for embedding graphics, audio, and video content via the new `<canvas>`, `<audio>`, and `<video>` tags.

- Extensions to the JavaScript API such as geolocation and drag-and-drop as well for storage and caching.

- Introduction of "web workers".

- Several new semantic tags were also added to complement the structural logic of modern web applications. These include the `<main>`, `<nav>`, `<article>`, `<section>`, `<header>`, `<footer>`, and `<aside>` tags.

- New form controls, such as `<calendar>`, `<date>`, `<time>`, `<email>`, `<url>`, and `<search>`.

17. Describe the relationship between the <header> and <h1> tags in HTML5.

In previous specifications of HTML, only one `<h1>` element was typically present on a page, used for the heading of the entire page. HTML5 specifies that `<h1>` represents the top-level heading of a "section", whether that be the page `<body>`, or an `<article>` or `<section>` element. In fact, every `<header>` element should at least contain an `<h1>` element. If there is no natural heading for the section, it is a good indication it should not use an `<article>` or `<section>` tag.

18. Write the code necessary to create a 300 pixel by 300 pixel <canvas>. Within it, paint a blue 100 pixel by 100 pixel square with the top-left corner of the square located 50 pixels from both the top and left edges of the canvas.

Here is one simple implementation:

```
<canvas id="c" width="300" height="300"></canvas>

<script>
    var canvas = document.getElementById( "c" );
    var drawing_context = canvas.getContext( "2d" );
    drawing_context.fillStyle = "blue";
    drawing_context.fillRect( 50, 50, 100, 100 );
</script>
```

19. What is HTML5 Web Storage? Explain localStorage and sessionStorage?

With HTML5, web pages can store data locally within the user's browser.

Earlier, this was done with cookies. However, Web Storage is more secure and faster. The data is not included with every server request, but used ONLY when asked for.
The data is stored in name/value pairs, and a web page can only access data stored by itself. Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.
The difference between localStorage and sessionStorage involves the lifetime and scope of the storage. Data stored through localStorage is permanent: it does not expire and remains stored on the user's computer until a web app deletes it or the user asks the browser to delete it. SessionStorage has the same lifetime as the top-level window or browser tab in which the script that stored it is running. When the window or tab is permanently closed, any data stored through sessionStorage is deleted.
Both forms of storage are scoped to the document origin so that documents with different origins will never share the stored objects. But sessionStorage is also scoped on a per-window basis. If a user has two browser tabs displaying documents from the same origin, those two tabs have separate sessionStorage data: the scripts running in one tab cannot read or overwrite the data written by scripts in the other tab, even if both tabs are visiting exactly the same page and are running exactly the same scripts.

20. What is custom attributes in HTML5?

A custom data attribute starts with data- and would be named based on your requirement. Following is the simple example−

```
<div class="example" data-subject="physics" data-level="complex">
    ...
</div>
```

The above will be perfectly valid HTML5 with two custom attributes called data-subject and data-level. You would be able to get the values of these attributes using JavaScript APIs or CSS in similar way as you get for standard attributes.

21. What are the drawbacks of cookies?

Cookies have following drawbacks−

- Cookies are included with every HTTP request, thereby slowing down your web application by transmitting the same data.
- Cookies are included with every HTTP request, thereby sending data unencrypted over the internet.
- Cookies are limited to about 4 KB of data . Not enough to store required data.

22. What are web sockets?

Web Sockets is a next-generation bidirectional communication technology for web applications which operates over a single socket and is exposed via a JavaScript interface in HTML 5 compliant browsers.

Once you get a Web Socket connection with the web server, you can send data from browser to server by calling a send() method, and receive data from server to browser by an onmessage event handler.

Following is the API which creates a new WebSocket object.

```
var Socket = new WebSocket(url, [protocal] );
```

Here first argument, url, specifies the URL to which to connect. The second attribute, protocol is optional, and if present, specifies a sub-protocol that the server must support for the connection to be successful.

23. What is the purpose of Socket.readyState atribute of WebSocket?

The readonly attribute readyState represents the state of the connection. It can have the following values:

- A value of 0 indicates that the connection has not yet been established.
- A value of 1 indicates that the connection is established and communication is possible.
- A value of 3 indicates that the connection has been closed or could not be opened.
- A value of 2 indicates that the connection is going through the closing handshake.

24. What is the purpose of Socket.bufferedAmount atribute of WebSocket?

The readonly attribute bufferedAmount represents the number of bytes of UTF-8 text that have been queued using send() method.

25. What is Geolocation API in HTML?

HTML5 Geolocation API lets you share your location with your favorite web sites. A Javascript can capture your latitude and longitude and can be sent to backend web server and do fancy location-aware things like finding local businesses or showing your location on a map.

Today most of the browsers and mobile devices support Geolocation API. The geolocation APIs work with a new property of the global navigator object ie. Geolocation object which can be created as follows:

```
var geolocation = navigator.geolocation;
```

The geolocation object is a service object that allows widgets to retrieve information about the geographic location of the device.

26.What is purpose of getCurrentPosition() method of geolocation object of HTML5?

This method retrieves the current geographic location of the user.

27. What is purpose of watchPosition() method of geolocation object of HTML5?

This method retrieves periodic updates about the current geographic location of the device.

28. What is purpose of clearPosition() method of geolocation object of HTML5?

This method cancels an ongoing watchPosition call.

29.What are new HTML 5 <form> elements?

New HTML 5 <form> elements are

- **<datalist>**
- **<keygen>**

- **<output>**

30.Describe Server Sent Event ?

Server Sent Event allows our webpages to get automatically updated from the server.

31. What is feature detection?

Feature detection detect wheather some new feature of HTML 5 is running on the given browsers or not?

Feature detection can done by two ways:
1. Using modernizr.js
2. Using Simple JavaScript

```
if(typeof(Storage) !== "undefined") {
Code for localStorage/sessionStorage.
} else {
Sorry No Browser support for Storage }
```

What is modernizr?

Modernizr is an open source, MIT-licensed JavaScript library that detects support for many HTML5 & CSS3 new features. You should always use the latest version. It runs automatically

```
<script src="modernizr.min.js"></script>
if (Modernizr.canvas) {
// let's draw some wonderful shapes!
} else {
// no browser support for canvas available
}
```

http://www.webdevelopmenthelp.net/2013/04/html5-interview-questions.html

31.Benefits of using an IIFE?

Reduce scope lookups:

A small performance benefit of using IIFE's is the ability to pass commonly used global objects (window, document, jQuery, etc) to an IIFE's anonymous function, and then reference these global objects within the IIFE via a local scope.

Minification Optimization

Another small performance benefit of using IIFE's is that it helps with minification optimization. Since we are able to pass global objects into the anonymous function as local values (parameters), a minifier is able to reduce the name of each global object to a one letter word (as long as you don't already have a variable of the same name). Let's look at the above code sample after minification:

Readability

One of the con's with using IIFE's is readability. If you have a lot of JavaScript code inside of an IIFE, and you want to find the parameters that you are passing into an IIFE, then you need to scroll all the way to the bottom of the page. Luckily, there is a more readable pattern that I have begun to use:

This IIFE pattern allows you see what global objects you are passing into your IIFE, without you having to scroll to the very bottom of a potentially very long file. Happy IIFE'ing!

Javascript Closure:

http://stackoverflow.com/questions/111102/how-do-javascript-closures-work

http://javascriptissexy.com/understand-javascript-closures-with-ease/

****

CSS3

****

1. **Explain what a class selector is and how it's used:**

A class can be thought of as a grouped collection of CSS attributes applied to HTML elements. This allows you to apply the same styling to multiple HTML elements by placing them in the same CSS class.

•Class methods can be called by inserting a 'class' property and name within an HTML element, then calling the class name with a '.' in the CSS doc.

2. **What are pseudo classes and what are they used for?**

•Pseudo classes are similar to classes, but are not explicitly defined in the markup, and are used to add additional effects to selected HTML elements such as link colors, hover actions, etc.

•Pseudo classes are defined by first listing the selector, followed by a colon and then pseudo-class element. E.g., a:link{ color: blue }, or a:visited { color: red }

•Pseudo-class syntax:

selector:pseudo-class {
property:value;}

•Syntax for using a pseudo class within a CSS class:

selector.class:pseudo-class {
property:value;}

•:link, :visited, :hover, :active, :first_line are all examples of pseudo classes, used to call a specific action on an element, such as the changing of a link color after it has been visited.

3. **Explain the three main ways to apply CSS styles to a Web page:**

•**Inline:** Though this method often goes against best practices, it's easily done by inserting a 'style' attribute inside an HTML element:

•e.g.) Lorem Ipsum

•**Embedded/Internal:** Done by defining the head of an HTML document by wrapping characteristics in a <style> tag.

•**Linked/External:** CSS is placed in an external .css file, and linked to the HTML document with a <link> tag. This can also be accomplished using the '**@import**', however, this can slow page load time and is generally not advised.

4. **What is grouping and what is it used for?**

•Grouping allows you to apply the same style to multiple elements with a single declaration. This is done by grouping the selectors into a list, separated by commas.

e.g.) h1, h2 { font-family: Helvetica; font-size: 20; }
•Grouping helps memory usage and enhances readability.

5. **What is a Class selector and how does it differ from an ID selector?**

The main difference is that the same class selector can be applied to multiple HTML elements, whereas ID selectors are unique.

6. **What are child selectors?**

•Child selectors are another way to group and style a set of elements that descend from a parent element.

•A child selector is matched by calling two or more elements, separated by a '>' sign to indicate inheritance.

e.g.)

```
body > p {
font-family: Helvetica;
font-size: 20;
}
```

•In this example, the same styling would be applied to all paragraphs within the body.

7.**How to restore the default property value using CSS?**

- In short, there's no easy way to restore to default values to whatever a browser uses . The closest option is to use the '**initial**' property value, which will restore it to the default CSS values, rather than the browser's default styles.

8.**What is the purpose of pseudo-elements and how are they made?**

- Pseudo elements are made using a double colon (::) followed by the name of the pseudo element.
- Pseudo-elements are used to add special effects to some selectors, and can only be applied to block-level elements.
- Most commonly used pseudo-elements are ::first_line, ::first_letter, ::before, ::after

9.**How are inline and block elements different from each other?**

- A block element is an element that takes up the full width available, and has a line break before and after it. <h1>, <p>, <li>, and are all examples of block elements.
- An inline element only takes up as much width as necessary, cannot accept width and height values, and does not force line breaks. <a> and are examples of inline elements.

10.**What is the purpose of the z-index and how is it used?**

- The z-index helps specify the stack order of positioned elements that may overlap one another. The z-index default value is zero, and can take on either a positive or negative number.
- An element with a higher z-index is always stacked above one with a lower index.
- Z-Index can take the following values:

Auto: Sets the stack order equal to its parents.
Number: Orders the stack order.
Initial: Sets this property to its default value (0).
Inherit: Inherits this property from its parent element.

11. **What are the advantages and disadvantages of External Style Sheets?**

- The biggest advantages of external style sheets are that they can be applied to multiple documents while being managed from a single style sheet. This keeps code DRY and improves efficiency and convenience.
- The disadvantages are that it may decrease loading time in some situations. It may also not be practical if there are not enough styling conditions to justify an external sheet.

12. **Explain the difference between visibility:hidden and display:none**

- **visibility:hidden** simply hides the element, while it will still take up space and affect the layout of the document.
- **display:none** also hides the element, but will not take up space and the page will appear as if the element is not present.

13. **What are some of the new features and properties in CSS3?**

- Box model
- New Web fonts
- Rounded corners
- Border Images
- Box Shadows, Text Shadows
- New Color schemes (RGBA)
- Transform property
- New Pseudo-classes
- Multi-column layout
- New Gradients

14. **Why shouldn't I use fixed sized fonts ?**

•Often times, fixed font sizes will show up incorrectly on the user end and will prohibit responsiveness. Using relative sizing will keep fonts proportionate in their relationships to each other and will allow for greater end user flexibility.

15. **Which font names are available on all platforms ?**

•Only five basic font families(serif, sans-serif, cursive, fantasy, and monsospace) are recognized across platforms, rather than specific fonts.

•Specific font name recognitions will vary by browser.

16.**What are the advantages/disadvantages of using CSS preprocessors? (SASS, Compass, Stylus, LESS)**

•Here is another opportunity to discuss your personal preferences on use of CSS preprocessors and why.

•While there's no right or wrong answer here, below are some commonly cited pros and cons of using preprocessors:

Benefits: Ability to use nested sytax, define variables and mixins, use of mathematical and operational functions, and the ability to join multiple files into a single one.
Disadvantages: Difficulties tracking file size, maintenance and updating issues, difficulties debugging.

8. Who maintains the CSS specifications?
World Wide Web Consortium maintains the CSS specifications.

10. What benefits and demerits do External Style Sheets have?
Benefits:

- One file can be used to control multiple documents having different styles.
- Multiple HTML elements can have many documents, which can have classes.

- To group styles in composite situations, methods as selector and grouping are used.

Demerits:

- Extra download is needed to import documents having style information.
- To render the document, the external style sheet should be loaded.
- Not practical for small style definitions.

11. Discuss the merits and demerits of Embedded Style Sheets?

Merits of Embedded Style Sheets:

- Multiple tag types can be created in a single document.
- Styles, in complex situations, can be applied by using Selector and Grouping methods.
- Extra download is unnecessary.

Demerits of Embedded Style Sheets:

- Multiple documents cannot be controlled.

15. Differentiate Style Sheet concept from HTML?

While HTML provides easy structure method, it lacks styling, unlike Style sheets. Moreover, style sheets have better browser capabilities and formatting options.

19. Enlist the various fonts' attributes?

They are:

- Font-style
- Font-variant
- Font-weight
- Font-size/line-height
- Font-family
- Caption
- Icon

24. What is Pseudo-elements ?

Pseudo-elements are used to add special effects to some selectors. CSS in used to apply styles in HTML mark-up. In some cases when extra mark-up or styling is not possible for the document, then there is a feature available in CSS known as pseudo-elements. It will allow extra mark-up to the document without disturbing the actual document.

28. Enlist the various Media types used?

Different media has different properties as they are case insensitive.

They are:

- Aural – for sound synthesizers and speech
- Print – gives a preview of the content when printed
- Projection- projects the CSS on projectors.
- Handheld- uses handheld devices.
- Screen- computers and laptop screens.

29. What is CSS Box Model and what are its elements?

This box defines design and layout of elements of CSS. The elements are:

Margin: the top most layer, the overall structure is shown

Border: the padding and content option with a border around it is shown. Background color affects the border.

Padding: Space is shown. Background colour affects the border.

Content: Actual content is shown.

31. Compare RGB values with Hexadecimal color codes ?

A color can be specified in two ways:

- A color is represented by 6 characters i.e. hexadecimal color coding. It is a combination of numbers and letters and is preceded by #. e.g.: g {color: #00cjfi}
- A color is represented by a mixture of red, green and blue. The value of a color can also be specified. e.g.:rgb(r,g,b): In this type the values can be in between the integers 0 and 255.rgb(r%,g%,b%): red, green and blue percentage is shown.

32. Define Image sprites with context to CSS ?

When a set of images is collaborated into one image, it is known as ‘Image Sprites’. As the loading every image on a webpage consumes time, using image sprites lessens the time taken and gives information quickly.

CSS coding:

```
1 img.add { width: 60px; height: 55px; background: url (image.god) 0 0; }
```

In this case, only the part needed is used. The user can save substantial margin and time through this.

34. How can the dimension be defined of an element ?

Dimension properties can be defined by:

- Height
- Max-height
- Max-width
- Min-height
- Min-width
- Width

36. How does Z index function?

Overlapping may occur while using CSS for positioning HTML elements. Z index helps in specifying the overlapping element. It is a number which can be positive or negative, the default value being zero.

40. How can the gap under the image be removed?

As images being inline elements are treated same as texts, so there is a gap left, which can be removed by:

CSS

```
1 img { display: block ; }
```

41. Why is @import only at the top?

@import is preferred only at the top, to avoid any overriding rules. Generally, ranking order is followed in mostprogramming languages such as Java, Modula, etc. In C, the # is a prominent example of a @import being at the top.

****

JavaScript

http://www.toptal.com/javascript/interview-questions
http://javascript.info/tutorial/events-and-timing-depth
https://www.codementor.io/javascript/tutorial/21-essential-javascript-tech-interview-practice-questions-answers
http://www.tutorialspoint.com/javascript/javascript_interview_questions.htm
http://career.guru99.com/top-85-javascript-interview-questions/
http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers
https://medium.com/javascript-scene/10-interview-questions-every-javascript-developer-should-know-6fa6bdf5ad95#.c1ipb8i38
http://www.sitepoint.com/5-typical-javascript-interview-exercises/

****

1. What is a potential pitfall with using typeof bar === "object" to determine if bar is an object? How can this pitfall be avoided?

Although `typeof bar === "object"` *is* a reliable way of checking if `bar` is an object, the surprising gotcha in JavaScript is that `null` is *also* considered an object!
Therefore, the following code will, to the surprise of most developers, log `true` (not `false`) to the console:

```
var bar = null;
console.log(typeof bar === "object");  // logs true!
```

As long as one is aware of this, the problem can easily be avoided by also checking if `bar` is `null`:

```
console.log((bar !== null) && (typeof bar === "object"));  // logs false
```

To be entirely thorough in our answer, there are two other things worth noting:

First, the above solution will return `false` if `bar` is a function. In most cases, this is the desired behavior, but in situations where you want to also return `true` for functions, you could amend the above solution to be:

```
console.log((bar !== null) && ((typeof bar === "object") || (typeof bar === "function")));
```

Second, the above solution will return `true` if `bar` is an array (e.g., if `var bar = [];`). In most cases, this is the desired behavior, since arrays are indeed objects, but in situations where you want to also `false` for arrays, you could amend the above solution to be:

```
console.log((bar !== null) && (typeof bar === "object") && (toString.call(bar) !== "[object Array]"));
```

Or, if you're using jQuery:

```
console.log((bar !== null) && (typeof bar === "object") && (! $.isArray(bar)));
```

2.What will the code below output to the console and why?

```
(function(){
    var a = b = 3;
})();

console.log("a defined? " + (typeof a !== 'undefined'));
console.log("b defined? " + (typeof b !== 'undefined'));
```

Since both `a` and `b` are defined within the enclosing scope of the function, and since the line they are on begins with the `var` keyword, most JavaScript developers would expect `typeof a` and `typeof b` to both be *undefined* in the above example.

However, that is *not* the case. The issue here is that most developers *incorrectly* understand the statement `var a = b = 3;` to be shorthand for:

```
var b = 3;
var a = b;
```

But in fact, `var a = b = 3;` is actually shorthand for:

```
b = 3;
var a = b;
```

As a result (if you are *not* using strict mode), the output of the code snippet would be:

```
a defined? false
b defined? true
```

But how can "b" be defined *outside* of the scope of the enclosing function? Well, since the statement `var a = b = 3;` is shorthand for the statements `b = 3;` and `var a = b;`, "b" ends up being a global variable

(since it is not preceded by the `var` keyword) and is therefore still in scope even outside of the enclosing function.

Note that, in strict mode (i.e., with `use strict`), the statement `var a = b = 3;` will generate a runtime error of `ReferenceError: b is not defined`, thereby avoiding any headfakes/bugs that might othewise result. (Yet another prime example of why you should use `use strict` as a matter of course in your code!)

3. What is the significance, and what are the benefits, of including 'use strict' at the beginning of a JavaScript source file?

The short and most important answer here is that `use strict` is a way to voluntarily **enforce stricter parsing and error handling on your JavaScript code** at runtime. Code errors that would otherwise have been ignored or would have failed silently will now generate errors or throw exceptions. In general, it is a good practice.

Some of the key benefits of strict mode include:

- **Makes debugging easier.** Code errors that would otherwise have been ignored or would have failed silently will now generate errors or throw exceptions, alerting you sooner to problems in your code and directing you more quickly to their source.
- **Prevents accidental globals.** Without strict mode, assigning a value to an undeclared variable automatically creates a global variable with that name. This is one of the most common errors in JavaScript. In strict mode, attempting to do so throws an error.
- **Eliminates `this` coercion**. Without strict mode, a reference to a `this` value of null or undefined is automatically coerced to the global. This can cause many headfakes and pull-out-your-hair kind of bugs. In strict mode, referencing a a `this` value of null or undefined throws an error.
- **Disallows duplicate property names or parameter values.** Strict mode throws an error when it detects a duplicate named property in an object (e.g., `var object = {foo: "bar", foo: "baz"};`) or a duplicate named argument for a function (e.g., `function foo(val1, val2, val1){}`), thereby catching what is almost certainly a bug in your code that you might otherwise have wasted lots of time tracking down.
- **Makes eval() safer.** There are some differences in the way `eval()` behaves in strict mode and in non-strict mode. Most significantly, in strict mode, variables and functions declared inside of an `eval()` statement are *not* created in the containing scope (they *are* created in the containing scope in non-strict mode, which can also be a common source of problems).
- **Throws error on invalid usage of `delete`.** The `delete` operator (used to remove properties from objects) cannot be used on non-configurable properties of the object. Non-strict code will fail silently when an attempt is made to delete a non-configurable property, whereas strict mode will throw an error in such a case.

4. What is NaN? What is its type? How can you reliably test if a value is equal to NaN?

The NaN property represents a value that is “not a number”. This special value results from an operation that could not be performed either because one of the operands was non-numeric (e.g., "abc" / 4), or because the result of the operation is non-numeric (e.g., an attempt to divide by zero).
While this seems straightforward enough, there are a couple of somewhat surprising characteristics of NaNthat can result in hair-pulling bugs if one is not aware of them.
For one thing, although NaN means “not a number”, its type is, believe it or not, Number:
console.log(typeof NaN === "number"); // logs "true"
Additionally, NaN compared to anything – even itself! – is false:
console.log(NaN === NaN); // logs "false"
A semi-reliable way to test whether a number is equal to NaN is with the built-in functionisNaN(), but even using isNaN()is an imperfect solution.
A better solution would either be to usevalue !== value, which would only produce true if the value is equal to NaN. Also, ES6 offers a newNumber.isNaN()function, which is a different and more reliable than the old globalisNaN()function.

5.Write a simple function (less than 80 characters) that returns a boolean indicating whether or not a string is a palindrome.

The following one line function will return true if str is a palindrome; otherwise, it returns false.

```
function isPalindrome(str) {
   str = str.replace(/\W/g, '').toLowerCase();
   return (str == str.split('').reverse().join(''));
}
```
For example:

```
console.log(isPalindrome("level"));                 // logs 'true'
console.log(isPalindrome("levels"));                // logs 'false'
console.log(isPalindrome("A car, a man, a maraca"));  // logs 'true'
```

6. How to duplicate array in javascript?
Simplest way is use **concat method**
array.concat(array);

7. Find unique elements in array?
Array.prototype.contain=function(ele){

```
  for(var i=0;i<this.length;i++){
     if(this[i]===ele)
     return true;
   }
   return false;
}
Array.prototype.unique=function(){
          var arr=[];
          for(i=0;i<this.length;i++){
          if(!arr.contain(this[i]))
                    arr.push(this[i]);
          }
         return arr;
}

console.log([1,1,2,3,2,3,4,1,3,42,2].unique());
```

8.

****

AngularJS

****

1. Angular Directive:
   Scope Parameters:
   Each scope parameter can be passes through HTML attributes:

```
    scope: {
     param1: '=',  //two way binding (reference)
```

```
  param2: '@',  //one way expression (top down)
  param3: '&'     //one way behavior (bottom up)
 }
```

'=' usedful to communicate between directive and outside caller.
'@' Caller can specify a string with interpolation value in it
'&'   example ng-click, one way behavior'

link(scope,element,attributes) function is executed once per element.

2. What are the ways to communicate between modules of your application using core AngularJS functionality? Name three ways.

Communication can happen:

- Using services
- Using events
- By assigning models on $rootScope
- Directly between controllers, using $parent, nextSibling, etc
- Directly between controllers, using ControllerAs, or other forms of inheritence

3. How do you reset a “$timeout”, and disable a “$watch()”?

The key to both is assigning the result of the function to a variable.

To cleanup the timeout, just “.cancel()” it:

```
var customTimeout = $timeout(function () {
  // arbitrary code
}, 55);
```

$timeout.cancel(customTimeout);
The same applies to “$interval()”.

To disable a watch, just call it.

```
// .$watch() returns a deregistration function that we store to a variable
var deregisterWatchFn = $rootScope.$watch('someGloballyAvailableProperty', function (newVal) {
  if (newVal) {
    // we invoke that deregistration function, to disable the watch
    deregisterWatchFn();
    ...
  }
});
```

4. Name and describe the phases of a directive definition function execution, or describe how directives are instantiated.

The flow is as follows:
First, the “$compile()” function is executed which returns two link functions, preLink and postLink. That function is executed for every directive, starting from parent, then child, then grandchild.
Secondly, two functions are executed for every directive: the controller and the prelink function. The order of execution again starts with the parent element, then child, then grandchild, etc.
The last function postLink is executed in the inverse order. That is, it is first executed for grandchild, then child, then parent.
A great explanation of how directives are handled in AngularJS is available in the AngularJS Tutorial: Demystifying Custom Directivespost on the Toptal blog.

5. How does interpolation, e.g. “{{ someModel }}”, actually work?
It relies on $interpolation, a service which is called by the compiler. It evaluates text and markup which may contain AngularJS expressions. For every interpolated expression, a “watch()” is set.
$interpolation returns a function, which has a single argument, “context”. By calling that function and providing a scope as context, the expressions are “$parse()”d against that scope.

6. How does the digest phase work?
In a nutshell, on every digest cycle all scope models are compared against their previous values. That is dirty checking. If change is detected, the watches set on that model are fired. Then another digest cycle

executes, and so on until all models are stable.

It is probably important to mention that there is no ".$digest()" polling. That means that every time it is being called deliberately. As long as core directives are used, we don't need to worry, but when external code changes models the digest cycle needs to be called manually. Usually to do that, ".$apply()" or similar is used, and not ".$digest()" directly.

7. $q?
A service that helps you run functions asynchronously, and use their return values (or exceptions) when they are done processing.

This is an implementation of promises/deferred objects inspired by Kris Kowal's Q.

$q can be used in two fashions --- one which is more similar to Kris Kowal's Q or jQuery's Deferred implementations, and the other which resembles ES6 promises to some degree.

New instance of deferred is constructed by calling $q.defer().

The purpose of the deferred object is to expose the associated Promise instance as well as APIs that can be used for signaling the successful or unsuccessful completion, as well as the status of the task.

Methods

resolve(value) – resolves the derived promise with the value. If the value is a rejection constructed via $q.reject, the promise will be rejected instead.
reject(reason) – rejects the derived promise with the reason. This is equivalent to resolving it with a rejection constructed via $q.reject.
notify(value) - provides updates on the status of the promise's execution. This may be called multiple times before the promise is either resolved or rejected.
Properties

promise – {Promise} – promise object associated with this deferred.

8. $emit vs $broadcat?
http://www.bennadel.com/blog/2807-using-rootscope-emit-as-a-performance-optimization-in-

angularjs.htm
http://www.dotnet-tricks.com/Tutorial/angularjs/HM0L291214-Understanding-$emit,-$broadcast-and-$on-in-AngularJS.html

$rootScope.$emit only lets other $rootScope listeners catch it. This is good when you don't want every $scope to get it. Mostly a high level communication. Think of it as adults talking to each other in a room so the kids can't hear them.

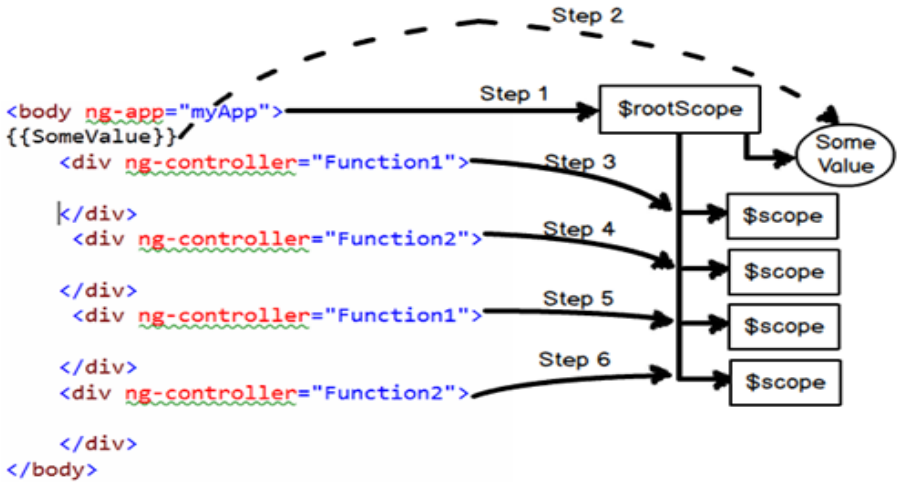
$rootScope.$broadcast is a method that lets pretty much everything hear it. This would be the equivalent of parents yelling that dinner is ready so everyone in the house hears it.

$scope.$emit is when you want that $scope and all its parents and $rootScope to hear the event. This is a child whining to their parents at home (but not at a grocery store where other kids can hear).

$scope.$broadcast is for the $scope itself and its children. This is a child whispering to its stuffed animals so their parents can't hear.

9. $rootscope?

"$rootScope" is a parent object of all "$scope" angular objects created in a web page.



10. How to make HTTP get and post call in AngularJS?

To make HTTP calls we need to use the "$http" service of Angular. In order to use the http services you need to make provide the "$http" as a input in your function parameters as shown in the below code.

function CustomerController($scope,$http)

```
{
        $scope.Add = function()
        {
            $http({
```

```
                    method: "GET",
                    url: "http://localhost:8438/SomeMethod"
                   }).success(function (data, status, headers, config)
                  {
                     // Here goes code after success
                  }
          }
}
```

11.What is dependency injection and how does it work in Angular?

Dependency injection is a process where we inject the dependent objects rather than consumer creating the objects. DI is everywhere in Angular or we can go one step ahead and say Angular cannot work without DI.

For example in the below code “$scope” and “$http” objects are created and injected by the angular framework. The consumer i.e. “CustomerController” does not create these objects himself rather Angular injects these objects.

```
function CustomerController($scope,$http)
{
// your consumer would be using the scope and http objects
}
```

12. How does DI benefit in Angular?

There are two big benefits of DI: - Decoupling and Testing.

13. What are the different custom directive types in AngularJS?

There are different flavors of Angular directives depending till what level you want to restrict your custom directive.

In other words do you want your custom directive to be applied only on HTML element or only on an attribute or just to CSS etc.

So in all there are four different kinds of custom directives:-

Element directives (E)

Attribute directives (A)

CSS class directives (C)

Comment directives (M)

14. What is services in AngularJS?

In AngularJS services are the singleton objects or functions that are used for carrying out specific tasks. It holds some business logic and these function can be called as controllers, directive, filters and so on.

15. Explain what is data binding in AngularJS ?

Automatic synchronization of data between the model and view components is referred as data binding in AngularJS. There are two ways for data binding

1. Data mining in classical template systems

2. Data binding in angular templates

16. Mention the steps for the compilation process of HTML happens?

Compilation of HTML process occurs in following ways

1.Using the standard browser API, first the HTML is parsed into DOM

2. By using the call to the $compile () method, compilation of the DOM is performed.

3.The method traverses the DOM and matches the directives.

4.Link the template with scope by calling the linking function returned from the previous step

17.Explain what is linking function and type of linking function?

Link combines the directives with a scope and produce a live view. For registering DOM listeners as well as updating the DOM, link function is responsible. After the template is cloned it is executed.

Pre-linking function: Pre-linking function is executed before the child elements are linked. It is not considered as the safe way for DOM transformation.

Post linking function: Post linking function is executed after the child elements are linked. It is safe to do DOM transformation by post-linking function

18. Explain what is the difference between link and compile in Angular.js?

Compile function: It is used for template DOM Manipulation and collect all of the directives.

Link function: It is used for registering DOM listeners as well as instance DOM manipulation. It is executed once the template has been cloned.

19.Mention what are the styling form that ngModel adds to CSS classes ?

ngModel adds these CSS classes to allow styling of form as well as control

1. ng- valid

2. ng- invalid

3. ng-pristine

4. ng-dirty

20.What is ngBind?

The ngBind attribute tells Angular to replace the text content of the specified HTML element with the value of a given expression, and to update the text content when the value of that expression changes.

Typically, you don't use ngBind directly, but instead you use the double curly markup like {{ expression }} which is similar but less verbose.

It is preferable to use ngBind instead of {{ expression }} if a template is momentarily displayed by the browser in its raw state before Angular compiles it. Since ngBind is an element attribute, it makes the bindings invisible to the user while the page is loading.

An alternative solution to this problem would be using the ngCloak directive.

21.what is immutability?

In object-oriented and functional programming, an immutable object is an object whose state cannot be modified after it is created. This is in contrast to a mutable object, which can be modified after it is created.

http://blog.mgechev.com/2015/03/02/immutability-in-angularjs-immutablejs/

https://www.npmjs.com/package/angular-immutable

22. What is hoisting?

In Javascript, you can have multiple var-statements in a function. All of these statements act as if they were declared at the top of the function. Hoisting is the act of moving the declarations to the top of the function.

```
var myvar = 'my value';

(function() {
  alert(myvar);           // undefined
  var myvar = 'local value';
})();
```

Variable Declarations are Hoisted

Within its current scope, regardless of where a variable is declared, it will be, behind the scenes, hoisted to the top. However, only the declaration will be hoisted. If the variable is also initialized, the current value, at the top of the scope, will initially be set to undefined.

Functions are hoisted too..

function definition hoisting only occurs for function declarations, not function expressions.

```
// Outputs: "Definition hoisted!"
definitionHoisted();
// TypeError: undefined is not a function
definitionNotHoisted();
function definitionHoisted() { console.log("Definition hoisted!"); }
var definitionNotHoisted = function () { console.log("Definition not hoisted!"); };
```

23. What is the difference between the $parse, $interpolate and $compile services?

http://stackoverflow.com/questions/17900588/what-is-the-difference-between-the-parse-interpolate-and-compile-services

Those are all examples of services that aid in AngularJS view rendering (although $parse and $interpolate could be used outside of this domain). To illustrate what is the role of each service let's take example of this piece of HTML:

```
var imgHtml = '<img ng-src="/path/{{name}}.{{extension}}">'
```

and values on the scope:

$scope.name = 'image';

$scope.extension = 'jpg';

Given this markup here is what each service brings to the table:

$compile - it can take the whole markup and turn it into a linking function that, when executed against a certain scope will turn a piece of HTML text into dynamic, live DOM with all the directives (here: ng-src) reacting to model changes. One would invoke it as follows: $compile(imgHtml)($scope) and would get a DOM element with all the DOM event bounds as a result. $compile is making use of $interpolate (among other things) to do its job.

$interpolate knows how to process a string with embedded interpolation expressions, ex.: /path/{{name}}.{{extension}}. In other words it can take a string with interpolation expressions, a scope and turn it into the resulting text. One can think of the $interpolation service as a very simple, string-based template language. Given the above example one would use this service like: $interpolate("/path/{{name}}.{{extension}}")($scope) to get the path/image.jpg string as a result.

$parse is used by $interpolate to evaluate individual expressions (name, extension) against a scope. It can be used to both read and set values for a given expression. For example, to evaluate the name expression one would do: $parse('name')($scope) to get the "image" value. To set the value one would do: $parse('name').assign($scope, 'image2')

All those services are working together to provide a live UI in AngularJS. But they operate on different levels:

$parse is concerned with individual expressions only (name, extension). It is a read-write service.

$interpolate is read only and is concerned with strings containing multiple expressions (/path/{{name}}.{{extension}})

$compile is at the heart of AngularJS machinery and can turn HTML strings (with directives and interpolation expressions) into live DOM.

24. What is ng-Cloak?

The ngCloak directive is used to prevent the Angular html template from being briefly displayed by the browser in its raw (uncompiled) form while your application is loading. Use this directive to avoid the undesirable flicker effect caused by the html template display.

The directive can be applied to the <body> element, but the preferred usage is to apply multiple ngCloak directives to small portions of the page to permit progressive rendering of the browser view.

gCloak works in cooperation with the following css rule embedded within angular.js and angular.min.js. For CSP mode please add angular-csp.css to your html file (see ngCsp).

```
[ng\:cloak], [ng-cloak], [data-ng-cloak], [x-ng-cloak], .ng-cloak, .x-ng-cloak {
  display: none !important;
}
```

When this css rule is loaded by the browser, all html elements (including their children) that are tagged with the ngCloak directive are hidden. When Angular encounters this directive during the compilation of the template it deletes the ngCloak element attribute, making the compiled element visible.

For the best result, the angular.js script must be loaded in the head section of the html document; alternatively, the css rule above must be included in the external stylesheet of the application.

25. Event bubbling?

Event bubbling and capturing are two ways of event propagation in the HTML DOM API, when an event occurs in an element inside another element, and both elements have registered a handle for that event. The event propagation mode determines in which order the elements receive the event.

With bubbling, the event is first captured and handled by the innermost element and then propagated to outer elements.

With capturing, the event is first captured by the outermost element and propagated to the inner elements.

Capturing is also called "trickling", which helps remember the propagation order:

trickle down, bubble up

Back in the old days, Netscape advocated event capturing, while Microsoft promoted event bubbling. Both are part of the W3C Document Object Model Events standard (2000).

IE < 9 uses only event bubbling, whereas IE9+ and all major browsers support both. On the other hand, the performance of event bubbling may be slightly lower for complex DOMs.

We can use the addEventListener(type, listener, useCapture) to register event handlers for in either bubbling (default) or capturing mode. To use the capturing model pass the third argument as true.

Example

```
<div>
   <ul>
      <li></li>
```

```
    </ul>
</div>
```

In the structure above, assume that a click event occurred in the li element.

In capturing model, the event will be handled by the div first (click event handlers in the div will fire first), then in the ul, then at the last in the target element, li.

In the bubbling model, the opposite will happen: the event will be first handled by the li, then by the ul, and at last by the div element.

27. Arrow Function?

ES6 fat arrow functions have a shorter syntax compared to function expressions and lexically bind the this value. Arrow functions are always anonymous and effectively turn function (arguments) { expression } into arguments => expression. If using an expression after an arrow, the return is implicit, so no return is required.

28.

****

GruntJS

****

1. Q1. What is Grunt?

Ans. Grunt is a JavaScript task runner, which can automate tasks like minification, compilation, unit testing, checking js errors. Once configured, one doesn't have to worry about these tasks.

2. Why to use Grunt?

Ans. Grunt has become very popular and has tons of plugins to choose from. These plugins are great assets for any app to automate various things with minimum efforts.

3. How do you install Grunt?

Ans. Grunt and Grunt plugins are installed and managed via npm, the Node.js package manager. To install grunt, first ensure the npm is installed properly. And then run following command.

```
npm install grunt –save-dev
```

4. How do you setup/configure Grunt?

Ans. Once installed, you need to add 2 files to setup Grunt.

1. **package.json:** This file is used by npm to store metadata for projects published as npm modules. So basically, there will be list of all Grunt plugins, along with grunt which your project is using.

2. **Gruntfile**: This file is named Gruntfile.js and is used to configure or define tasks and load Grunt plugins.

5. What is --save-dev option while installing the grunt?

Ans. As mentioned in previous answer that “package.json” file holds the metadata for grunt plugins. So when grunt is installed using --save-dev option, the metadata is added to package.json. So you don’t have to add it manually. And this is how your package.json will look like,

```
{
  "name": "my-project-name",
  "version": "0.1.0",
  "devDependencies": {
    "grunt": "~0.4.5",
    "grunt-contrib-jshint": "~0.10.0",
    "grunt-contrib-nodeunit": "~0.4.1",
    "grunt-contrib-uglify": "~0.5.0"
  }
}
```

6.What is the difference between --save and --save-dev?

Ans. Before will look at the difference, it is important to understand the difference between dependencies and devDependencies.

devDependencies are for the development-related scripts, e.g. unit testing, packaging scripts, documentation generation, etc. where dependencies are required for production use, and assumed required for dev as well. As for example, you can include some plugin which you require during development like (for debugging or unit testing) but you don't need them on production.

So **--save adds packages under dependencies and --save-dev adds under devdependencies section**.

```
{
  "name": "my-project-name",
  "version": "0.1.0",
  "dependencies": {
    "grunt": "~0.4.5",
    "grunt-contrib-nodeunit": "~0.4.1",
    "grunt-contrib-uglify": "~0.5.0"
  }
  "devDependencies": {
    "grunt": "~0.4.5",
    "grunt-contrib-jshint": "~0.10.0",}}
```

7. What are Grunt modules/plugins?

Ans. Grunt modules are distributed through Node's NPM directory. Normally, they are prefixed with grunt- and official grunt plugins are prefixed with grunt-contrib. Example: grunt-contrib-uglify. You can get list of all grunt plugins

8. How do you install a Grunt plugin?

Ans. The syntax remains name but the only thing which changes is module/plugin name.

npm install <module> --save-dev

For example, to install uglify plugin,

npm install grunt-contrib-uglify --save-dev

By default, it always installs the latest version available. But if you wish to install specific version then same you can include in the command.

9.What is Grunt-cli?

Ans. Grunt cli is command line interface to run grunt commands. In other words, it's a tool to access Grunt from command line anywhere in the system. To install, grunt –cli execute following command

```
npm install -g grunt-cli
```

This will put the grunt command in your system path, allowing it to be run from any directory. Please note, that installing grunt-cli, doesn't install grunt on your system.

-g means global which means it adds to PATH variables of the system so that you can run grunt from anywhere (without going to specific folder on command prompt).

The reason for having two components is to ensure that we can run different grunt versions side-by-side (i.e. legacy versions in older projects). Hence it is recommended to install grunt-cli globally while grunt should be installed on a per-project basis.

10. How does Gruntfile.js uses package.json?

Ans. Task configuration is specified in your Gruntfile via the grunt.initConfig method. Inside of grunt.initConfig(), we read the information from package.json and saved it to a pkg property. With this, we can use the attributes from our package.json file. We can call the name of our project using pkg.name and the version with pkg.version.

```
module.exports = function(grunt) {
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json')
  });
};
```

11. Where do you define configuration of grunt plugin?

Ans. Grunt plugin configuration needs to be defined within grunt.initConfig method. See below sample code.

```
module.exports = function(grunt) {
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    concat: {
    // concat task configuration goes here.
    },
   uglify: {
    // uglify task configuration goes here.
   }
  });
};
```

12.Can you override default settings of a plugin? If yes, then how?

Ans. Yes, we can override. Inside a task configuration, an options property may be specified to override built-in defaults. In addition, each target may have an options property which is specific to that target.

```
module.exports = function(grunt) {
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    concat: {
     options: {
```

```
      separator: ';'
    }
  }
 });
};
```

13. What is a task in Grunt?

Ans. Tasks are grunt's bread and butter. Every time Grunt is run, you specify one or more tasks to run, which tells Grunt what you'd like it to do. See below code.

```
grunt.initConfig({
 concat: {
  development: {
   // concat task "development" target options and files go here.
  },
  production: {
   // concat task "production" target options and files go here.
  },
 },
 uglify: {
  development: {
   // uglify task "development" target options and files go here.
  },
 },
});
```

Here, there are 2 tasks defined concat and uglify. And for each task, we defined targets. For concat, there are 2 targets "development" and "production" and only "development" for uglify. Creating target allows us to define separate settings for different objectives. Here, we can have different option for development and production version. It's not compulsory to define a target.

Target's can also have their own option parameters which will override the settings of options defined

for the task.

14.How do you execute grunt task?

Ans. You can execute the grunt task from command line. Remember, we have grunt command line interface. To execute contact module with grunt for all the tasks.

grunt concat

To execute task with particular target

grunt concat:development

15. How do we load grunt plugins in Gruntfile.js?

Ans. grunt.loadNpmTasks() is used for loading grunt plugins. Before loading, please ensure that these plugins are already installed via npm.

```
grunt.loadNpmTasks('grunt-contrib-uglify');
```

16.How to get a stack trace when an error occurs?

Ans. Use the --stack option to see stack traces. Such as grunt task –stack.

17.Which are the most used grunt plugins?

Ans. Though there are tons of plugins which you can use, but below are the most used.

watch: Run predefined tasks whenever watched file patterns are added, changed or deleted.

jshint: Validate files with JSHint

uglify: Minify files with UglifyJS

concat: Concatenate files.

cssmin: Minify CSS

less: Compile LESS files to CSS.