**1. What is CORS?**

url: https://www.maxcdn.com/one/visual-glossary/cors/

A)​​**defination**:

Cross-origin resource sharing (CORS) is a standard for accessing web resources on different domains.

CORS allows web scripts to interact more openly with content outside of the original domain, leading to better integration between web services.

**overview:**

To prevent websites from tampering with each other, web browsers implement a security measure known as the same-origin policy.

The same-origin policy lets resources (such as JavaScript) interact with resources from the same domain, but not with resources from a different domain.

This provides security for the user by preventing abuse, such as running a script that reads the password field on a secure website.

**how it works**:

When a browser executes a script that references a resource on another domain,

it requests the content directly from the second domain.

The second domain determines whether or not to serve the content by validating the first domain, which is included as part of the request.

The second domain then returns either the content or an error message back to the browser, bypassing the first domain entirely.

// in node js

$ npm install cors

var express = require('express');

var cors = require('cors');

var app = express();

app.use(cors());

app.use(function (req, res, next) {

*res.header("Access-Control-Allow-Origin", "\*");*

*res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");*

*next();*

});

**2. what is CDN?**

A) The use of a Content Delivery Network (CDN) for public web sites is quite common.

If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).

One big advantage of using the hosted jQuery from Google or Microsoft:

1.Many users already have downloaded jQuery from Google or Microsoft when visiting another site. As a result, it will be loaded from cache when they visit your site, which leads to faster loading time.

Also, most CDN's will make sure that once a user requests a file from it, it will be served from the server closest to them, which also leads to faster loading time.

2.The value of a CDN lies in the likelihood of the user having already visited another site calling that same file from that CDN, and becomes increasingly valuable depending on the size of the file.

The likelihood of this being the case increases with the ubiquity of the file being requested and the popularity of the CDN.

->is that CDN will allow users to pull the files from a "local" or at least closer (to the user) server. So if your server is in Europe, users from South America or Russia, for example, will still get the files relatively fast.

3.If you are working on a website that is expected to work in intranet only then, use local server directory. If you are working on a website that is expected to work on internet as well, then you should try CDN

**Q3) What is component in Angular2?**

A).Angular 2 or 4 is a component-based framework with typescript, Components are major building blocks of Angular 2 or 4.

A component can be defined as an independent cohesive block of code which has the required logic, view, and data as one unit.

When written using TypeScript, a component is a TypeScript class decorated with @Component() decorator.

**Q4) Library vs Framework vs platform**

**Library**-> is just a set of tools for you to use. It won't normally enforce a workflow on you and your code will be executing library code. Math is a good example of a library.

**Framework**-> enforces workflow as it is a working thing and only gives you extension points to put your code in and it will run your code and control things for you.

For example you have no control when spring framework will call your controller. You are given guidelines on what you need to do for Spring to think your class is a controller.

**The platform**-> defines a standard around which a system can be developed. It will have a set of standard APIs that expose specific components (streams, file system, ports, etc) for your use.

JVM is a good example of a platform as well as operating systems.

**Q5) language vs framework**

A **language**-> is syntax, grammar, semantics (and perhaps a core library) that implementers are required to support.

A **framework**-> is a cohesive set of library code that together simplifies programming in any given language.

**Q6) Cross Browser Compatible**

1. Cross-browser means web application works with all versions of all browsers.

To claim cross-browser compatibility, the website is nowadays expected to support browsers such as Mozilla Firefox,Google Chrome, Opera, and Safari in addition to Internet Explorer and Netscape.

Here are the basic tips to make your website cross browser compatible.

**Tips**:

1. Define Valid Doctype:

Your doctype tells the browser what rules you’ll be using in your code. If you don’t specify, the browser has to guess, and different browsers will guess differently. When you design in “strict mode,” Internet Explorer 6 and above will show behavior much closer to that of Firefox, Safari and other modern browsers.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"

"http://www.w3.org/TR/html4/strict.dtd">

Otherwise, Internet Explorer runs in a “quirks mode” where it attempts to emulate the behavior of older browsers.

2. CSS Reset:

By default, Different browsers behave as per their default css rules. You need to explicitly define css to make same behavior for different browsers. I like Eric’s CSS reset:

http://meyerweb.com/eric/thoughts/2007/05/01/reset-reloaded/

http://meyerweb.com/eric/tools/css/reset/index.html

3. Conditional Comments:

Try to use conditional comments instead of CSS hacks.

With conditional comments you can link to separate style sheets for different browsers.

<link type="text/css" href="style.css" />

<!--[If IE]>

<link type="text/css" href="IEHacks.css" />

<![endif]-->

<!--[if !IE]>

<link type="text/css" href="NonIEHacks.css" />

<![endif]-->

ex2.

<!--[if IE 5]>

<p>This message is only displayed in IE5.</p>

<![endif]-->

<!--[if !IE 5]>

<p>This message is only displayed in browsers other than IE5.</p>

<![endif]-->

4. Try to use javascript library like jQuery, YahooUI, MooTools, Dojo abstract away the differences in the DOM, AJAX and JavaScript.

5. You can use Css Frameworks like BluePrint, 960 Grid. These are mostly cross browser compatible.

See following to know which css framework is suitable for you.

http://net.tutsplus.com/tutorials/html-css-techniques/which-css-grid-framework-should-you-use-for-web-design/

6. Validate:

The W3C Validation Service validates multiple versions of XHTML and HTML, outputting many useful errors and warnings to help users create a perfect website.

W3C Validator: http://validator.w3.org/

W3C Css Validator: http://jigsaw.w3.org/css-validator/

7. Testing:

I would suggest you to start with firefox and then move to IE. Use Firebug, a firefox addon and IE Developer Toolbar for IE to set layout. To test website on different browser, there are some online and offline utilities like

browsershots,

IETester,

browsercam.

http://www.monitis.com/blog/5-css-techniques-for-better-cross-browser-compatibility/ (for css)

**Q7) make responcive website(media querys)**

A)

1. Flexible Layouts:

The divs you use to create your web page layouts need to consist of relative length units. This means you shouldn't use fixed widths in your CSS, rather use percentages.

The formula to convert sizes from a design to percentages is (target/context)x100 = result

// image here(URL:https://stackoverflow.com/questions/20210223/how-do-i-make-a-website-responsive)

Lets take the picture above as an example of a design. To calculate what the size of the div on the left is going to be calculated like this: (300px/960px)x100 = 30.25%

The CSS would look something like this:

.leftDiv

{

width: 30.25%;

float: left;

}

.rightDiv

{

width: 65%;

float: left;

}

For text to automatically resize you can use a unit called VW (ViewWidth)

.myText

{

font-size: 1vw;

}

This ensures that the text automatically resize relative to the view width.

2. Flexible Media:

Flexible media applies to images, videos and canvasses which automatically resize relative to its parent.

Example:

img, video, canvass

{

max-width: 100%;

}

This ensures that these elements resize automatically inside its parent.

3. **Media Queries**:

The next step is to use media queries like you've done in your question, these media queries define certain CSS statements for certain screen sizes. I normally use only three media queries for computers screens, tablets and phone screens. Its not necessary to have more than this because the Flexible Layouts and Flexible Media will ensure relative resizing if done correctly.

/\* Landscape phone to portrait tablet \*/

@media (min-width: 340px) and (max-width: 500px) {

#content {

height: 215px;

}

}

/\* Landscape phone to portrait tablet \*/

@media (min-width: 501px) and (max-width: 767px) {

#content {

height: 660px;

}

}

You may find this helpful: https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media\_queries

<URL:https://stackoverflow.com/questions/20210223/how-do-i-make-a-website-responsive>

**Q8) Screen sizes for media queries**

1. Device = Desktops

Screen = 1281px to higher resolution desktops

@media (min-width: 1281px) {

}

2. Device = Laptops, Desktops

Screen = B/w 1025px to 1280px

@media (min-width: 1025px) and (max-width: 1280px) {

}

3. Device = Tablets, Ipads (portrait)

Screen = B/w 768px to 1024px

@media (min-width: 768px) and (max-width: 1024px) {

}

4. Device = Tablets, Ipads (landscape)

Screen = B/w 768px to 1024px

@media (min-width: 768px) and (max-width: 1024px) and (orientation: landscape) {

}

5. Device = Low Resolution Tablets, Mobiles (Landscape)

Screen = B/w 481px to 767px

@media (min-width: 481px) and (max-width: 767px) {

}

6. Device = Most of the Smartphones Mobiles (Portrait)

Screen = B/w 320px to 479px

@media (min-width: 320px) and (max-width: 480px) {

}

**Q8) AJAX**

A)AJAX = Asynchronous JavaScript And XML.

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes.

This means that it is possible to update parts of a web page, without reloading the whole page.

Syntax for creating an XMLHttpRequest object:

var xhttp = new XMLHttpRequest();

->To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

xhttp.open("GET/POST", "ajax\_info.txt", true);

xhttp.send();

->**Server Response**:

The readyState property holds the status of the XMLHttpRequest.

The onreadystatechange property defines a function to be executed when the readyState changes.

The status property and the statusText property holds the status of the XMLHttpRequest object.

**XMLHttpRequest Properties**

**onreadystatechange**

An event handler for an event that fires at every state change.

**readyState**

The readyState property defines the current state of the XMLHttpRequest object.

The following table provides a list of the possible values for the readyState property −

**State** **Description**

0 The request is not initialized.

1 The request has been set up.

2 The request has been sent.

3 The request is in process.

4 The request is completed.

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

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

readyState = 2 After you have called send().

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

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

**responseText**

Returns the response as a string.

**responseXML**

Returns the response as XML. This property returns an XML document object, which can be examined and parsed using the W3C DOM node tree methods and properties.

**status**

Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").

**statusText**

Returns the status as a string (e.g., "Not Found" or "OK").

example:

function loadDoc() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("demo").innerHTML =

this.responseText;

}

};

xhttp.open("GET", "ajax\_info.txt", true);

xhttp.send();

}

`