CSS:

1. **W3C** is an acronym standing for the World Wide Web Consortium. This is the group that determines the standards for the technology behind the Web

2. You have a HTML table and you want to give white color to odd columns and green color to even columns. How will you do this?

   Answer:
   Table td:nth-child(even) {
    Background: green;
   }

   [note that the count starts from 0. So each row, the first, third , fifth etc columns will be green]

3. What are the various CSS selectors? Ask some of them

http://www.w3schools.com/cssref/css_selectors.asp

some of them:

Selector	**Example**	**Example description**
.class	.intro	Selects all elements with class="intro"
#id	#firstname	Selects the element with id="firstname"
*	*	Selects all elements
element	p	Selects all <p> elements
element,element	div,p	Selects all <div> elements and all <p> elements
element element	div p	Selects all <p> elements inside <div> elements
element>element	div>p	Selects all <p> elements where the parent is a <div> element

element+element	div+p	Selects all <p> elements that are placed immediately after <div> elements
element1~element2	p~ul	Selects every <ul> element that are preceded by a <p> element. This is only In CSS3.

4. Set a background color for all ul elements that are preceded by a p element with the same parent:

Answer:
p~ul
{
background:#ff0000;
}

5. What is the meaning of

   a < img { border: none; }

   Answer: it would select a tags but only if they contained an img tag.

6. What is the meaning of
   div>p
   {
   background-color:yellow;
   }

   Answer: Select and style every <p> element where the parent is a <div> element:

   Q again: can you write it as p<div ?
   Answer: NO

7. You have an HTML file that calls an external file inside it by using load() method of **jQuery**. However, the CSS in main HTML file and CSS of external file conflict. How do you prevent it.

```
Example:

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>
<script>
function ext() {
   $('#aaa').load('external.txt');
}
</script>

<body onLoad="ext()">

<style>
   h1 {
      color:green;
   }
</style>

<h1>green</h1>

<div id="aaa"></div>
```

external.txt

```
<style>
h1 {
color:red;
}
</style>

<h1>red</h1>
```

Answer:

You could simply just apply a class to the h1 like so: `<h1 class="red">Red</h1>` and then just create a class like so `.red { color: red; }`

8. What are the differences between these two selectors?

```
ul li { margin: 0 0 5px 0; }
ul > li { margin: 0 0 5px 0; }
```

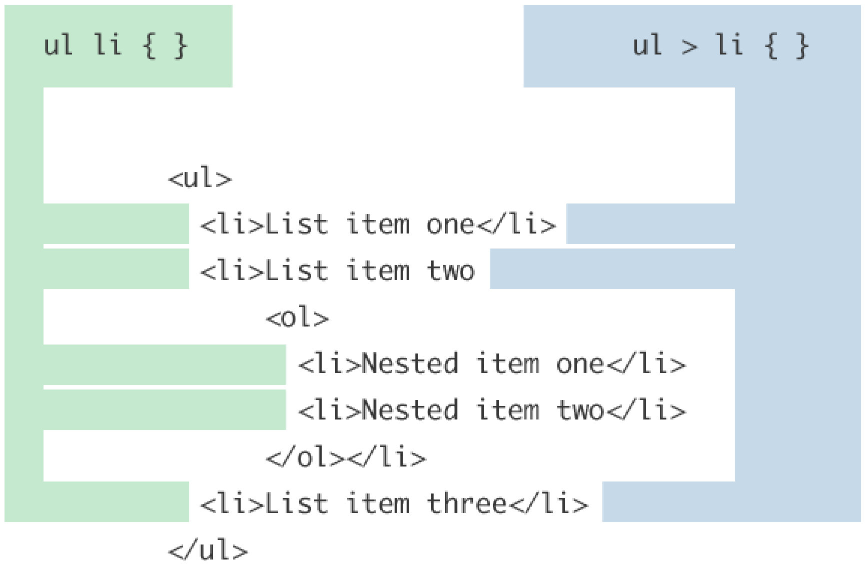
The first selector above is a *decendant selector*. It will select any list items that are anywhere underneath an unordered list in the markup structure. The list item could be buried three levels deep within other nested lists, and this selector will still match it.

The second selector above is a *child combinator selector*. This means it will only select list items that are direct children of an unordered list. In otherwords, it only looks *one level*down the markup structure, no deeper. So if there was another unordered list nested deeper, the list item children of it will not be targeted by this selector.

9. Other selectors that are similar to descendant/child selectors category

   other selectors in this style: the **child combinator**, the **adjacent sibling combinator**, and the **general sibling combinator**.

10. Child combinator



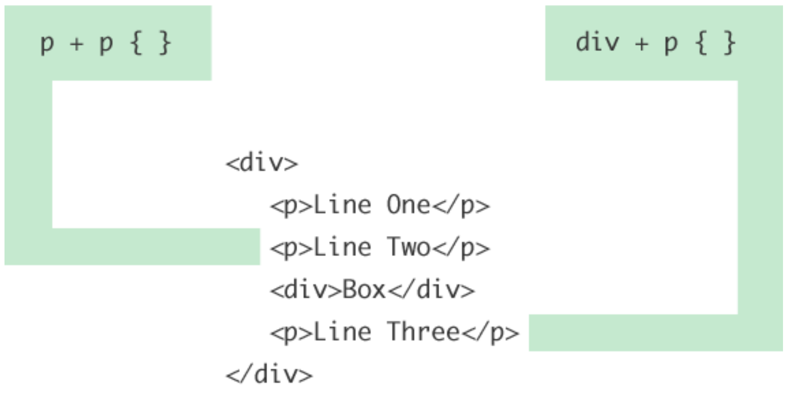
```
ul li { }                         ul > li { }

        <ul>
          <li>List item one</li>
          <li>List item two
              <ol>
                <li>Nested item one</li>
                <li>Nested item two</li>
              </ol></li>
          <li>List item three</li>
        </ul>
```

11. Adjacent sibling combinator

    An adjacent sibling combinator selector allows you to select an element that is **directly** after another specific element

p + p { font-size: smaller; } /* Selects all paragraphs that follow another paragraph */

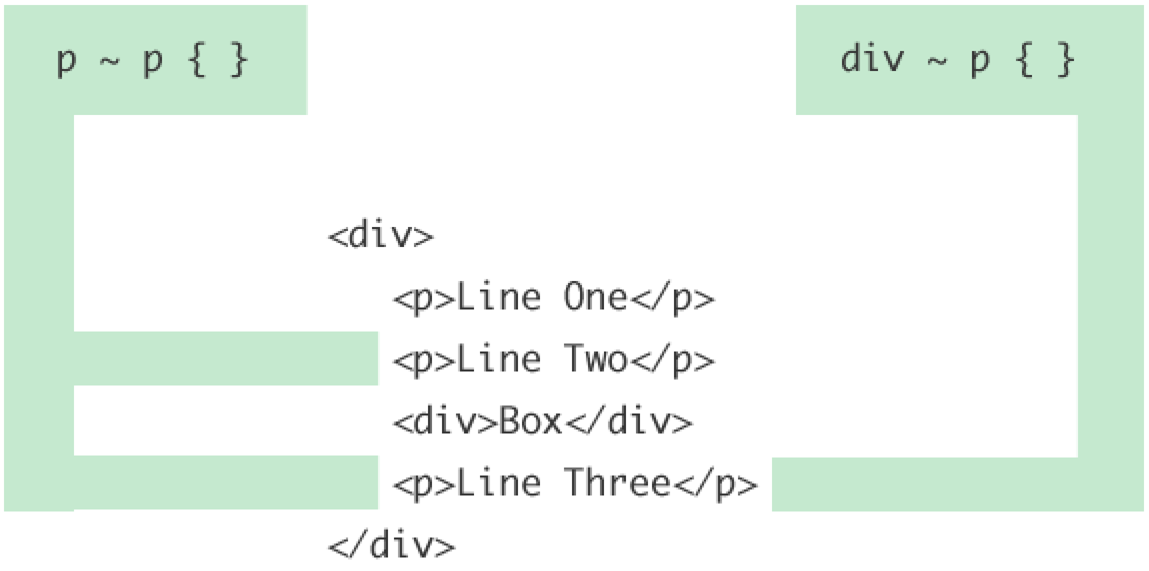
#title + ul { margin-top: 0; } /* Selects an unordered list that directly follows the element with ID title */



12. General sibling combinator

    The general sibling combinator selector is very similar to the adjacent sibling combinator selector we just looked at. The difference is that that the element being selected *doesn't need immediately succeed* the first element, but can appear anywhere after it.

If we use the same example structure as above, the last <p> element will be selected by p ~ p as well, because it is preceded by another <p> element, even though not*directly*.



13. How to select a <p> that has <img>

Answer: This is not supported by CSS. You can do this in jquery.

```
$("p:has(img)")
```

13. Where is the specification for the CSS 3 selectors?

Answer: http://www.w3.org/TR/css3-selectors/#selectors

14. How CSS gets evaluated [by the browser]??

Answer: The style of an element is evaluated on element creation

We often think of our pages as these full and complete documents full of elements and content. However, browsers are designed to handle documents like a stream. They begin to receive the document from the server and can render the document before it has completely downloaded. Each node is evaluated and rendered to the viewport as it is received.

Take a look at the body of an example document:

```
<body>
  <div id="content">
    <div class="module intro">
      <p>Lorem Ipsum</p>
    </div>
    <div class="module">
      <p>Lorem Ipsum</p>
      <p>Lorem Ipsum</p>
      <p>Lorem Ipsum <span>Test</span></p>
    </div>
  </div>
</body>
```

The browser starts at the top and sees a body element. At this point, it thinks it's empty. It hasn't evaluated anything else. The browser will determine what the computed styles are and apply them to the element. What is the font, the color, the line height? After it figures this out, it paints it to the screen.

Next, it sees a div element with an ID of content. Again, at this point, it thinks it's empty. It hasn't evaluated anything else. The browser figures out the styles and then the div gets painted. The browser will determine if it needs to repaint the body—did the element get wider or taller? (I suspect there are other considerations but width and height changes are the most common effects child elements have on their parents.)

This process continues on until it reaches the end of the document.

15 . If you have a css defined like this ‘div#content p { color: #003366; } ’ , how is this evaluated?

Answer: CSS gets evaluated from right to left.

To determine whether a CSS rule applies to a particular element, it starts from the right of the rule and works it's way left.

If you have a rule like body div#content p { color: #003366; } then for every element—as it gets rendered to the page—it'll first ask if it's a paragraph element. If it is, it'll work its way up the DOM and ask if it's a div with an ID of content. If it finds what it's looking for, it'll continue its way up the DOM until it reaches the body.

By working right to left, the browser can determine whether a rule applies to this particular element that it is trying to paint to the viewport much faster. To determine which rule is more or less performant, you need to figure out how many nodes need to be evaluated to determine whether a style can be applied to an element.

16. In terms of Performance, what do you take care while writing CSS selectors

Answer:

Avoid descendant selectors

Avoid child or adjacent selectors

17. Suppose I have a CSS declared like this, what does this mean and what do you feel about the performance?

#content * { color: #039; }

- Performance with id and * [Universal and Tag Selectors]
- Within its recommendations for descendant, child or adjacent selectors, it says to avoid the universal and tag selectors.

With the ID selector there, your initial thought might be that this is really fast. **The problem is that with the browser engine evaluating from right to left, the universal selector matches first**. For the browser to determine whether this element should be this deep shade of blue, it now has to check every ancestor element until it finds an element with an ID of content.

And it'll have to do this for every single element on the page.

Now that we understand when an element gets evaluated, how the selectors are determined, and how it might impact performance

18. What are the differences between CSS2 and CSS3?

http://webdesign.about.com/od/css3/a/differences-css2-css3.htm

The biggest difference between CSS2 and CSS3 is that CSS3 has been split up into different sections, called modules. Each of these modules is making its way through the W3C in various stages of the recommendation process. CSS2 was submitted as a single document with all the Cascading Style Sheets information within it. Because each of the modules is being worked on individually, we have a much wider range of browser support for CSS3 modules

CSS3 offers a bunch of new ways you can write CSS rules with new CSS selectors, as well as a new combinator, and some new pseudo-elements.

Three new attribute selectors:

- attribute beginning matches exactly
  `element[foo^="bar"]`
  The element has an attribute called foo that begins with "bar" e.g. <element foo="barn">
- attribute ending matches exactly
  `element[foo$="bar"]`
  The element has an attribute called foo that ends with "bar" e.g. <element foo="rebar">
- attribute contains the match

`element[foo*="bar"]`
The element has an attribute called foo that contains the string "bar" e.g. <element foo="rebaring">

16 new pseudo-classes:

- :root
  The root element of the document. In HTML this is always <html>
- :nth-child(n)
  use this to match exact child elements or use variables to get alternating matches
- :nth-last-child(n)
  match exact child elements counting up from the last one
- :nth-of-type(n)
  match sibling elements with the same name before it in the document tree
- :nth-last-of-type(n)
  match sibling elements with the same name counting up from the bottom
- :last-child
  match the last child element of the parent
- :first-of-type
  match the first sibling element of that type
- :last-of-type
  match the last sibling element of that type
- :only-child
  match the element that is the only child of its parent
- :only-of-type
  match the element that is the only one of its type
- :empty
  match the element that has no children (including text nodes)
- :target
  match an element that is the target of the referring URI
- :enabled
  match the element when it's enabled
- :disabled
  match the element when it's disabled
- :checked
  match the element when it's checked (radio button or checkbox)
- :not(s)
  match when the element does not match the simple selector s

One new combinator:

- elementA ~ elementB
  match when elementB follows somewhere after elementA, not necessarily immediately

19. What are CSS3 Modules? What are these modules and how are they different from CSS2? Can you explain these modules in a simple better way? And why these modules make CSS3 different from CSS2?

http://www.w3.org/TR/CSS/#css2 – CSS2 or CSS Level 2

http://www.w3.org/TR/CSS/#css3 – CSS3 or CSS Level 3

CSS2 specification - http://www.w3.org/TR/2008/REC-CSS2-20080411/

CSS Level 2 is "a specification"

CSS Level 3 is "a collection of specifications". Each specification (**module**) defines a (largely) self-contained set of related features, such as borders and backgrounds.
This allows the various specifications to progress to recommendations/implementations without having to wait for everything else.

20. Is CSS3 fully enabled in every browser?

http://stackoverflow.com/questions/9628120/is-css3-fully-enabled-in-every-browser?lq=1

The CSS3 specification itself is incomplete, so it cannot possibly be "fully enabled" in any browser at the moment

21. What are CSS 3 filters?

Answer:

http://css3.bradshawenterprises.com/filters/

It's worth noting that right now, CSS Filter Effects are an unoffical specification – however, the editors of the spec include representatives from Adobe, Apple and Opera, and we have already got implementations in Chrome, Safari and iOS 6. The specficiation can be found here.
https://dvcs.w3.org/hg/FXTF/raw-file/tip/filters/index.html

A filter effect is a graphical operation that is applied to an element as it is drawn into the document. It is an image-based effect, in that it takes zero or more images as input, a number of parameters specific to the effect, and then produces an image as output. The output image is either rendered into the document instead of the original element, used as an input image to another filter effect, or provided as a CSS image value.

```
.thing_you_want_to_filter {
  /*
```

```
    these are all default values, note that hue-rotate and blur have units.

    You'll also need to include the vendor prefixes.

  */

  filter: grayscale(0);

  filter: sepia(0);

  filter: saturate(1);

  filter: hue-rotate(0deg);

  filter: invert(0);

  filter: opacity(1);

  filter: brightness(1);

  filter: contrast(1);

  filter: blur(0px);


  /* Drop shadow has the same syntax as box-shadow – see below for why it's
amazing! */

  filter: drop-shadow(5px 5px 10px #ccc);

}
```

How to apply a CSS filter?

To use filters now, you need to use the vendor prefixed version of the ‘filter’ property

div { +filter: grayscale(100%); }

div { +filter: grayscale(50%); }

eg:

```
.mask{
    position:absolute;
    width:100%;
    top:11%;
    background:rgba(100,100,100,0.3);
    -webkit-filter:sepia(100%);
    height:100px;
    z-index:2;
}
```

- IMPORTANT: If you don't include "**-webkit**", the filter does not work. As it is not completely implemented by all browsers as of now [early 2014]

22. You have a 'ul' and list of 'li' and you want to display it as a table.
[display: table] . Now, for each 'li', you define this css -

```
border-bottom: 1px solid #bfd1ed;
border-right: 1px solid #bfd1ed;
```

Now the requirement is that I have to remove the bottommost border of the last row. I don't need it really.

Answer:

I don't think there is any way in CSS to select the "bottom row" of elements in this way. However, you can obscure the bottom border regardless with a negative margin on the container:

```
.centering-horizontal-nav {
    margin-bottom: -1px;
}
```

23. CSS Fundamental Interfaces

http://www.w3.org/TR/DOM-Level-2-Style/css.html

Interface CSSStyleSheet (introduced in DOM Level 2)

The CSSStyleSheet interface is a concrete interface used to represent a CSS style sheet i.e., a style sheet whose content type is "text/css".

IDL Definition

```
// Introduced in DOM Level 2:

interface CSSStyleSheet : stylesheets::StyleSheet {

  readonly attribute CSSRule         ownerRule;

  readonly attribute CSSRuleList     cssRules;

  unsigned long     insertRule(in DOMString rule,

                               in unsigned long index)

                                        raises(DOMException);

  void              deleteRule(in unsigned long index)

                                        raises(DOMException);

};
```

24. [Indirect question on CSS Box Model]

Assume that you had only 250px of space. Let's make an element with a total width of 250px: You need some border as well. Also check if he is confusing with top and bottom , which is actually associated with the height.

Answer:

```
width:220px;
padding:10px;
```

```
border:5px solid gray;
margin:0px
```

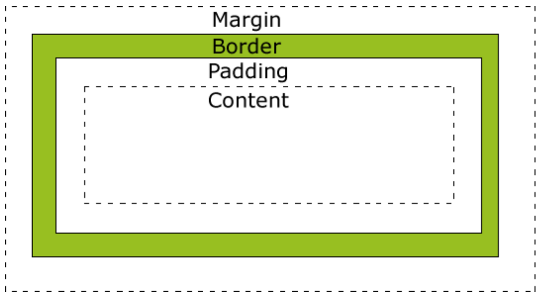
Let's do the math:
220px (width)
+ 20px (left + right padding)
+ 10px (left + right border)
+ 0px (left + right margin)
= 250px

The total width of an element should be calculated like this:

Total element width = width + left padding + right padding + left border + right border + left margin + right margin

The total height of an element should be calculated like this:

Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin



25. CSS Positioning

Answer: Static, Fixed, relative, Absolute
[should also relate with Box Model]

To understand positioning in CSS you must first understand the box model. For display purposes, every element in a document is considered to be a rectangular box which has a content area surrounded by padding, a border and margins

Margins are always transparent - [Make a question to puzzle : How to make margins transparent! ☺]

Margins, borders and padding are **all optional** but for purposes of calculating positions and sizes they are given a **default width of zero** if not specified. Different widths can be set for each individual side (top, right, bottom and left) if desired. **Margins** can even have **negative values.**

http://www.w3schools.com/css/css_positioning.asp
http://www.brainjar.com/css/positioning/

Static - HTML elements are positioned static by default. A static positioned element is always positioned according to the normal flow of the page.
Static positioned elements are not affected by the top, bottom, left, and right properties

Fixed Positioning - An element with fixed position is positioned relative to the browser window.
It will not move even if the window is scrolled:

Relative Positioning - A relative positioned element is positioned relative to its normal position. The content of relatively positioned elements can be moved and overlap other elements, but the reserved space for the element is still preserved in the normal flow.

Absolute Positioning - An absolute position element is positioned relative to the first parent element that has a position other than static. If no such element is found, the containing block is <html>. Absolutely positioned elements are removed from the normal flow. The document and other elements behave like the absolutely positioned element does not exist.
Absolutely positioned elements can overlap other elements.

26. Box Types [comes as part of Box Model/ CSS positioning]

There are two basic types of boxes, *block* and *inline.* Block boxes are generated by elements such as P, DIV or TABLE. Inline boxes are generated by tags such as B, I or SPAN and actual content like text and images.

The box type may also be set using the `display` property. Setting a value of `block` on an inline element, for example, will cause it to be treated as a block element. Note that if you set the display to `none,` no box is created. The browser acts as if the element did not exist. Likewise, any nested elements are ignored as well, even if they specifically declare some other display value.

27. CSS **Pseudo-elements**

[They are also part of CSS selectors]

CSS pseudo-elements are used to add special effects to some selectors.

The "first-line" pseudo-element is used to add a special style to the first line of a text.

The "first-line" pseudo-element can only be applied to block-level elements.
p:first-line

```
{
color:#ff0000;
font-variant:small-caps;
}
```

28. What is a block element?

A block element is an element that takes up the full width available, and has a line break before and after it.

Example: h1, p, div

Align block elements for layout purposes

How to horizontally align block elements for layout purposes.

29. How can you center-align a div element?

[Answer: look for margin auto, mentioning width, IE8 etc]

Block elements can be center-aligned by setting the left and right margins to "auto".

How it works? Setting the left and right margins to auto specifies that they should split the available margin equally. The result is a centered element.

Note: Using margin:auto; will not work in IE8 and earlier, unless a !DOCTYPE is declared.

Example:

```
.center
{
margin-left:auto;
margin-right:auto;
width:70%;
background-color:#b0e0e6;
}
```

Tip: Center-aligning has no effect if the width is 100%.

30. What does `<div align="center"> does?`

`Answer: It does text align center`, not the div. Only the text context inside it.

31. How do you left align and right align a div element?

One method of aligning elements is to use absolute positioning: [Other one is using float. See below Q:33]

```
.right
{
position:absolute;
right:0px;
width:300px;
background-color:#b0e0e6;
}
```

Cons: Absolute positioned elements are removed from the normal flow, and can overlap elements. Once positioned, browser doesn't care someone is sitting out there!

How do you overcome this??

32. How do you overcome cross-browser compatibility issues while trying to align elements using 'position' attribute or 'float' atttribute?

Crossbrowser Compatibility Issues

When aligning elements like this, it is always a good idea to predefine margin and padding for the <body> element. This is to avoid visual differences in different browsers.

There is a problem with IE8 and earlier, when using the position property. If a container element (in our case <div class="container">) has a specified width, and the !DOCTYPE declaration is missing, IE8 and earlier versions will add a 17px margin on the right side. This seems to be space reserved for a scrollbar. Always set the !DOCTYPE declaration when using the position property:

Example

```
body
{
margin:0;
padding:0;
}
.container
{
position:relative;
width:100%;
}
.right
{
position:absolute;
right:0px;
width:300px;
background-color:#b0e0e6;
}
```

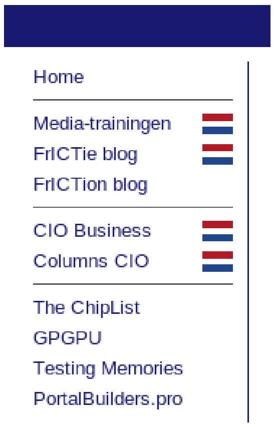
33. Left and Right Aligning Using the float Property

```
.right
{
float:right;
width:300px;
background-color:#b0e0e6;
}
```

33. Difference between floating, aligning and positioning?

http://portalbuilders.pro/PBblog/css-the-difference-between-floating-aligning-and-positioning/

34. If you build a menu like this



you want the text in the menu items to align to the left, and at the same time to align the flags to the right. What you do?

Answer:

float the flag image to the right within the `div` defining the menu item - works only in certain browsers.

Float is not the answer. Neither align. Position is the answer.

Floating a tag takes it out of the current text flow

The alignment defines the direction of the flow within an element, while a float takes an element out of the current flow.

Correct Answer: we deploy the positioning of elements:

div#left_menu div.left_menu_item {

```
  position: relative;

  ...

  }


div#left_menu div.left_menu_item img.flag {

  position: absolute;

  top: 2px;

  right: 10px;

  ...

  }
```

Setting the position to absolute allows us to use the top, bottom, left, and right parameters to explicitly align the flag relative to its containing element (or more precisely: the first containing element that has a position other than the default static; here: the *menu_item* `div`). Since we do not want to take the menu item block from the normal flow, we set its position to relative, but do not set the other positioning parameters.

35.

More CSS interview Questions:

http://programmers.stackexchange.com/questions/148213/how-can-i-evaluate-a-candidates-knowledge-of-html-css-during-an-interview

http://css-tricks.com/interview-questions-css/

19. http://webdesign.about.com/od/typemeasurements/qt/how-big-is-an-em.htm

---

JavaScript questions

Basic questions:

14. Difference between Javascript and Jquery
15. Var c = 2; ?
16. C= 3; // do you need to declare a variable before usage in javascript?
17. Scopes in javascript?how is it defined in javascript

18. this keyword in javascript
    http://www.quirksmode.org/js/this.html
19. DOM objects
20. is javascript Object oriented ?
21. when is DOM created? When is DOM called? When are HTML elements created?
22. who calls the document.ready() function? Is it a callback function
    How jquery auto detect that the DOM has been loaded
23. How is the jquery data connection works?
24. what is jquery success and failure? Are they function names in the jquery api?
25. what is a callback function? How is it different from normal function call of
    function3(){

Function1()
Function2()

}

Advanced Javascript Questions

1. What is the difference between == and ===? Which one would you use?

[me: equal meaning, value equal]

The equality (==) operator will compare for equality after doing necessary type casting,
the identity operator (===) doesn't do any conversions. A good practice suggested by Douglas Crockford is to always use strict equality

`===`, or 'strict comparison' means is the same type and equal

`==` simply means equal

```
“1" == 1 // true
"1" === 1 // false
```

An example of type coercion at work. Basically anytime your value is the "same" but the type isn't then `==` works.
Please use `===` everywhere. There's no need to use `==`. checking for types is always better

```
'' == '0'           // false
0 == ''             // true
0 == '0'            // true

false == 'false'    // false
false == '0'        // true

false == undefined // false
false == null       // false
null == undefined   // true
```

JavaScript has both strict and type-converting equality comparison. For strict equality the objects being compared must have the same type and:

- Two strings are strictly equal when they have the same sequence of characters, same length, and same characters in corresponding positions.
- Two numbers are strictly equal when they are numerically equal (have the same number value). NaN is not equal to anything, including NaN. Positive and negative zeros are equal to one another.
- Two Boolean operands are strictly equal if both are true or both are false.
- Two objects are strictly equal if they refer to the same Object.
- Null and Undefined types are == (but not ===).

2. How would you check if a variable is null/undefined?

```
//check if bar is null

bar === null
```

```
//check if bar is undefined

typeof bar === "undefined"
```

3. Safely turning a JSON string into an object

jQuery.parseJSON(jsonString); // using jquery

JSON.parse(jsonString); // without jquery

4. What is the difference between jquery.parsejson and json.parse?

jQuery will use the native JSON.parse method if it is available, and otherwise it will try to evaluate the data with a new Function, which is kind of like eval.
So yes, you should definitely use jQuery.parseJSON.

5. event.preventDefault() vs. return false [Add more points if answered correctly]

http://stackoverflow.com/questions/1357118/event-preventdefault-vs-return-false?rq=1

#1 event.preventDefault()

```
$('a').click(function (e) {
    // custom handling here
    e.preventDefault();
});
```

#2 return false

```
$('a').click(function () {
    // custom handling here
    return false;
});
```

Is there any significant difference between those two methods of stopping event propagation?

Answer1:

`return false` from *within a jQuery event handler* is effectively the same as calling both `e.preventDefault` and `e.stopPropagation` on the passed jQuery.Event object.

`e.preventDefault()` will prevent the default event from occuring, `e.stopPropagation()` will prevent the event from bubbling up and `return false` will do both. Note that this behaviour differs from *normal* (non-jQuery) event handlers, in which, notably, `return false` does *not* stop the event from bubbling up.

5.1 Q - > Explain event bubbling example in this scenario?

Answer:

```
<div onclick="alert('Outer clicked')">

  Outer Div<br><br>

  <a id="inner" href="http://www.google.com">Google</a>

</div>


document.getElementById('inner').addEventListener('click', function(e){

  alert(e.type);

  return false // DOES NOT STOP EVENT BUBBLING

}, false)
```

- Here, when you click on Google link, you get two alerts ‘click’ and ‘Outer clicked’

Asnwer2:

There is at least one clear advantage when using event.preventDefault() over using return false. Suppose you are capturing the click event on an anchor tag, otherwise which it would be a big problem if the user were to be navigated away from the current page. If your click handler uses return false to prevent browser navigation, it opens the possibility that the interpreter will not reach the return statement and the browser will proceed to execute the anchor tag's default behavior.

```
$('a').click(function (e) {
  // custom handling here

  // oops...runtime error...where oh where will the href take me?

  return false;
});
```

The benefit to using event.preventDefault() is that you can add this as the first line in the handler, thereby guaranteeing that the anchor's default behavior will not fire, regardless if the last line of the function is not reached (eg. runtime error).

```
$('a').click(function (e) {
  e.preventDefault();
  // custom handling here

  // oops...runtime error, but at least the user isn't navigated away.
});
```