------------------------------------------------------------------------------

HTML & CSS

-------------------------------------------------------------------------------

What is an example of a block element and an inline element? What is the difference between the two? In what situation would you use one over the other?

Ans. Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The <div> element is a block-level element.

Examples of block-level elements:

* <div>
* <h1> - <h6>
* <p>
* <form>

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline <span> element inside a paragraph.

Examples of inline elements:

* <span>
* <a>
* <img>

Why use CSS? Why can't I simply use a center tag or a font tag to style my page?

Ans. The <center> element was deprecated because it defines the presentation of its contents -- it doesn't describe its contents.

One method of centering is to set the margin-left and margin-right properties of the element to auto, and then set the parent element's text-align property to center. This guarantees that the element will be centered in all modern browsers.

What are the different ways (or locations) you can define styles on an element?

I can think of 3: External, Internal, In-line and Bootstrap.

What are the different values for the following properties? What are the effects of each of those values?

- display: block, inline, inline-block

- position: absolute, relative, fixed, static

- overflow: auto, hidden, scroll, visible

- float: left, right

What is the box model?

## Ans. The CSS Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:

Explanation of the different parts:

* Content - The content of the box, where text and images appear
* Padding - Clears an area around the content. The padding is transparent
* Border - A border that goes around the padding and content
* Margin - Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements.

What are the two different box-sizing values? How do they work?

Ans. The CSS3 box-sizing property allows us to include the padding and border in an element's total width and height.

If you set box-sizing: border-box; on an element padding and border are included in the width and height:

What is z-index? How does it work? In what situation would I use it?

Ans.

The z-index property specifies the stack order of an element.

An element with greater stack order is always in front of an element with a lower stack order.

Note: z-index only works on positioned elements (position:absolute, position:relative, or position:fixed).

z-index: auto|*number*|initial|inherit;

What is the difference between visibility: hidden and display: none?

Ans. display: none will not be available in the page and does not occupy any space. visibility: hidden hides an element, but it will still take up the same space as before. The element will be hidden, but still affect the layout. visibility: hidden preserve the space, whereas display: none doesn't preserve the space.

What is specificity? How do you calculate it?

## Ans

## The concept:

Specificity is the means by which browsers decide which CSS property values are the most relevant to an element and, therefore, will be applied. Specificity is based on the matching rules which are composed of [CSS selectors](https://developer.mozilla.org/en/CSS/CSS_Reference#Selectors) of different sorts.

## How is it calculated?

Specificity is a weight that is applied to a given CSS declaration, determined by the number of each [selector type](https://developer.mozilla.org/en/docs/Web/CSS/Specificity#Selector_Types) in the matching selector. When specificity is equal to any of the multiple declarations, the last declaration found in the CSS is applied to the element. Specificity only applies when the same element is targeted by multiple declarations. As per CSS rules [directly targeted element](https://developer.mozilla.org/en/docs/Web/CSS/Specificity#directly-targeted-elements) will always take precedence over rules that an element inherits from an ancestor.

Note: [Proximity of elements](https://developer.mozilla.org/en/docs/Web/CSS/Specificity#tree-proximity-ignorance) in the document tree has no effect on the specificity.

### Selector Types

The following list of selector types is by increasing specificity:

1. Type selectors (e.g., h1) and pseudo-elements (e.g., :before).
2. Class selectors (e.g., .example), attributes selectors (e.g., [type="radio"]) and pseudo-classes (e.g., :hover).
3. ID selectors (e.g., #example).

Universal selector (\*), combinators (+, >, ~, ' ') and negation pseudo-class (:not()) have no effect on specificity. (The selectors declared inside :not() do, however.)

Inline styles added to an element (e.g., style="font-weight:bold") always overwrite any styles in external stylesheets and thus can be thought of as having the highest specificity.

What does !important do? Why is it considered a bad practice?

Ans. When an important rule is used on a style declaration, this declaration overrides any other declarations. Although technically !important has nothing to do with specificity, it interacts directly with it. Using !important is **bad practice** and should be avoided because it makes debugging more difficult by breaking the natural [cascading](https://developer.mozilla.org/en-US/docs/Web/CSS/Cascade) in your stylesheets. When two conflicting declarations with the !important rule are applied to the same element, the declaration with greater specificity will be applied.

How would you go about building a 3 column layout?

Ans.

Flexbox consists of flex containers and flex items.

A flex container is declared by setting the display property of an element to either flex (rendered as a block) or inline-flex (rendered as inline).

Inside a flex container there is one or more flex items.

**Note:** Everything outside a flex container and inside a flex item is rendered as usual. Flexbox defines how flex items are laid out inside a flex container.

Flex items are positioned inside a flex container along a flex line. By default there is only one flex line per flex container.

<style>   
.flex-container {  
    display: -webkit-flex;  
    display: flex;  
    width: 400px;  
    height: 250px;  
    background-color: lightgrey;  
}  
  
.flex-item {  
    background-color: cornflowerblue;  
    width: 100px;  
    height: 100px;  
    margin: 10px;  
}  
</style>  
</head>  
<body>  
  
<div class="flex-container">  
  <div class="flex-item">flex item 1</div>  
  <div class="flex-item">flex item 2</div>  
  <div class="flex-item">flex item 3</div>   
</div>

If you have a div on a page that matches on all of the rules below, what would be the computed (finally applied) style on the element?

.myclass1 {

color: black;

border: 2px solid black;

overflow: scroll;

}

div.myclass2 {

color: red;

float: right;

}

div#myclass1 {

color: yellow;

border: 2px solid red;

}

Ans: div#myclass1 {

color: yellow;

border: 2px solid red;

}

Specificity rule

-------------------------------------------------------------------------------

SASS

-------------------------------------------------------------------------------

What are the benefits of using SASS over CSS?

Ans.

**Features**

* Fully CSS-compatible
* Language extensions such as variables, nesting, and mixins
* Many [useful functions](http://sass-lang.com/documentation/Sass/Script/Functions.html) for manipulating colors and other values
* Advanced features like [control directives](http://sass-lang.com/documentation/file.SASS_REFERENCE.html#control_directives__expressions) for libraries
* Well-formatted, customizable output

-------------------------------------------------------------------------------

JavaScript

-------------------------------------------------------------------------------

How would you create an array? Get its length? Access one of its elements?

Add and remove elements?

Ans.

var arr = [‘neha’ , ‘ipsha’ ];

arr.length();

arr[0];

arr.push(‘krunal’);

arr.pop(); //removes last element

arr.shift(); //removes first element

arr.unshift(‘mansi’);

How would you create a simple object? Get the number of keys? Access one of its values? Add and remove keys?

Ans.

var obj1 = {name:’neha’, age: 22 }; //literal method

var person = new Object(); //new keyword  
person.firstName = "John";  
person.lastName = "Doe";

//constructor method

function person(first, last, age, eye) {  
    this.firstName = first;  
    this.lastName = last;  
    this.age = age;  
    this.eyeColor = eye;  
}  
var myFather = new person("John", "Doe", 50, "blue");  
var myMother = new person("Sally", "Rally", 48, "green");

//find length of properties in the object

var x = Object.keys(obj1).length;

//access obj property

obj1.name;

//add new property

obj1.roll\_no = 22;

//delete property

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};  
delete person.age;

How would you decide when to use an array or when to use an object?

Ans.

Use array for homogeneous and ordered elements.

Use object for heterogeneous and un-ordered elements.

What is the global object?

The term "global objects" (or standard built-in objects) here is not to be confused with the **global object**. Here, global objects refer to **objects in the global scope** (but only if ECMAScript 5 strict mode is not used; in that case it returns [undefined](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/undefined)). The **global object** itself can be accessed using the [this](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/this) operator in the global scope. In fact, the global scope **consists of** the properties of the global object, including inherited properties, if any.

What kind of scoping do you have in JavaScript? And related to that, what is hoisting?

Ans.

Local

Global

//variable hoisting

x = 5;

Var x;

What are closures? How are they useful?

Ans.

**Closures are functions that refer to independent (free) variables (variables that are used locally, but defined in an enclosing scope). In other words, these functions 'remember' the environment in which they were created.**

function init() {

var name = "Mozilla"; // name is a local variable created by init

function displayName() { // displayName() is the inner function, a closure

alert(name); // use variable declared in the parent function

}

displayName();

}

init();

init() creates a local variable name and then a function called displayName(). displayName() is an inner function that is defined inside init() and is only available within the body of that function. displayName() has no local variables of its own, however it has access to the variables of outer functions and so can use the variable namedeclared in the parent function.

When defining objects -

- what is the constructor pattern, it's drawback?

- what is the prototype patter, it's drawback?

- give the above drawbacks, what approach would you go with?

- how would you implement inheritance between classes?

What are the different patterns? When would you use each one?

Ans.

* [Constructor Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#constructorpatternjavascript)
* [Module Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#modulepatternjavascript)
* [Revealing Module Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#revealingmodulepatternjavascript)
* [Singleton Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#singletonpatternjavascript)
* [Observer Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#observerpatternjavascript)
* [Mediator Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#mediatorpatternjavascript)
* [Prototype Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#prototypepatternjavascript)
* [Command Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#commandpatternjavascript)
* [Facade Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#facadepatternjavascript)
* [Factory Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#factorypatternjavascript)
* [Mixin Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#mixinpatternjavascript)
* [Decorator Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#decoratorpatternjavascript)
* [Flyweight Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#detailflyweight)

In simple terms, what is a DOM?

Ans.

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

How would you pick elements from a page? For each of these standard methods what is the input passed in and what would be the output? And what object can these methods be used on?

Ans.

Document.getElementById();

And so on…..

How would one update the styles on element?

Ans.

function display(){

var x=document.getElementById("demo").value;

document.getElementById("main").style.backgroundColor=x;

}

How can I create an element, add it to a page, and then remove it?

Ans.

var para = document.createElement("P");                       // Create a <p> element  
var t = document.createTextNode("This is a paragraph");       // Create a text node  
para.appendChild(t);                                          // Append the text to <p>  
document.body.appendChild(para);                              // Append <p> to <body>

var elem = document.getElementById("myDiv");

elem.remove();

How do I attach event listeners on an element?

Ans.

var x = document.getElementById("myBtn");

x.addEventListener("mouseover", myFunction);

function myFunction() {

document.getElementById("demo").innerHTML += "Moused over!<br>";

}

What are the two different event models?

-------------------------------------------------------------------------------

jQuery

-------------------------------------------------------------------------------

Why use jQuery?

Ans.

jQuery is a fast and concise JavaScript Library created by John Resig in 2006 with a nice motto − **Write less, do more**.

jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.

jQuery is a JavaScript toolkit designed to simplify various tasks by writing less code. Here is the list of important core features supported by jQuery −

* **DOM manipulation** − The jQuery made it easy to select DOM elements, traverse them and modifying their content by using cross-browser open source selector engine called **Sizzle**.
* **Event handling** − The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
* **AJAX Support** − The jQuery helps you a lot to develop a responsive and feature-rich site using AJAX technology.
* **Animations** − The jQuery comes with plenty of built-in animation effects which you can use in your websites.
* **Lightweight** − The jQuery is very lightweight library - about 19KB in size ( Minified and gzipped ).
* **Cross Browser Support** − The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
* **Latest Technology** − The jQuery supports CSS3 selectors and basic XPath syntax.

There are some standard native lookups in JavaScript, what are they corresponding versions in jQuery?

https://www.sitepoint.com/jquery-vs-raw-javascript-1-dom-forms/

What is the advantage of using these standard lookups in jQuery as well?

What is the difference between .find() and .children()? When would you use one over the other?

Ans.

The find() method returns descendant elements of the selected element.

A descendant is a child, grandchild, great-grandchild, and so on.

The children() method returns all direct children of the selected element.

How would one update the styles on element?

Ans.

$("li").children().css({"color": "red", "border": "2px solid red"});

How can I create an element, add it to a page, and then remove it?

Ans.

## Creating New Elements

jQuery offers a trivial and elegant way to create new elements using the same $() method used to make selections:

|  |  |
| --- | --- |
| 1  2  3 | *// Creating new elements from an HTML string.*  $( "<p>This is a new paragraph</p>" );  $( "<li class=\"new\">new list item</li>" ); |
| 1  2  3  4  5  6 | *// Creating a new element with an attribute object.*  $( "<a/>", {  html: "This is a <strong>new</strong> link",  "class": "new",  href: "foo.html"  }); |

Note that the attributes object in the second argument above, the property name class is quoted, although the property names html and href are not. Property names generally do not need to be quoted unless they are [reserved words](https://mathiasbynens.be/notes/reserved-keywords) (as class is in this case).

When you create a new element, it is not immediately added to the page. There are several ways to add an element to the page once it's been created.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | *// Getting a new element on to the page.*    **var** myNewElement = $( "<p>New element</p>" );    myNewElement.appendTo( "#content" );    myNewElement.insertAfter( "ul:last" ); *// This will remove the p from #content!*    $( "ul" ).last().after( myNewElement.clone() ); *// Clone the p so now we have two.* |

The created element doesn't need to be stored in a variable – you can call the method to add the element to the page directly after the $(). However, most of the time you'll want a reference to the element you added so you won't have to select it later.

You can also create an element as you're adding it to the page, but note that in this case you don't get a reference to the newly created element:

|  |  |
| --- | --- |
| 1  2 | *// Creating and adding an element to the page at the same time.*  $( "ul" ).append( "<li>list item</li>" ); |

The syntax for adding new elements to the page is easy, so it's tempting to forget that there's a huge performance cost for adding to the DOM repeatedly. If you're adding many elements to the same container, you'll want to concatenate all the HTML into a single string, and then append that string to the container instead of appending the elements one at a time. Use an array to gather all the pieces together, then join them into a single string for appending:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | **var** myItems = [];  **var** myList = $( "#myList" );    **for** ( **var** i = 0; i < 100; i++ ) {  myItems.push( "<li>item " + i + "</li>" );  }    myList.append( myItems.join( "" ) ); |

## Removing Elements

There are two ways to remove elements from the page: .remove() and .detach(). Use .remove() when you want to permanently remove the selection from the page. While .remove()does return the removed element(s), those elements will not have their associated data and events attached to them if you return them to the page.

Use .detach() if you need the data and events to persist. Like .remove(), it returns the selection, but it also maintains the data and events associated with the selection, so you can restore the selection to the page at a later time.

The .detach() method is extremely valuable if you are doing heavy manipulation on an element. In that case, it's beneficial to .detach() the element from the page, work on it in your code, then restore it to the page when you're done. This limits expensive "DOM touches" while maintaining the element's data and events.

If you want to leave the element on the page but remove its contents, you can use .empty() to dispose of the element's inner HTML.

How can I figure out the dimensions of an element?

Ans.

* [width()](http://www.w3schools.com/jquery/css_width.asp) - Sets or returns the width of an element
* [innerWidth()](http://www.w3schools.com/jquery/html_innerwidth.asp) - Returns the width of an element (includes padding)
* [innerHeight()](http://www.w3schools.com/jquery/html_innerheight.asp) - Returns the height of an element (includes padding)
* [outerWidth()](http://www.w3schools.com/jquery/html_outerwidth.asp) - Returns the width of an element (includes padding and border)
* [outerHeight()](http://www.w3schools.com/jquery/html_outerheight.asp) - Returns the height of an element (includes padding and border)
* height() - The height() method sets or returns the height of the selected elements.

How do I attach event listeners on an element?

Ans.

$("#p1").mouseenter(function(){

alert("You entered p1!");

});

And so on….

How would you go about implementing a div based pop-up that is triggered by clicking on a link? It needs to be centered across the screen as well.

How would go about implementing a feature where I need a fixed header and a fixed footer (i.e., these sections should be visible at all times and the header should appear at the top of the screen while the footer at the bottom of the screen). And the portion in the middle of the page (i.e., the content) is the only one that should be scrollable.

Ans.

Solved

-------------------------------------------------------------------------------

Angular JS

-------------------------------------------------------------------------------

How is Angular JS useful?

Ans.

Data binding.

Dependency Injection.

Unit testing.

Custom directives and filters.

What do the following directives do? Give an example.

**ng-app** : Use this directive to **auto-bootstrap** an AngularJS application. The ngApp directive designates the **root element** of the application and is typically placed near the root element of the page - e.g. on the <body> or <html> tags.

**ng-bind** : 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.

Example

<script>

angular.module('bindExample', [])

.controller('ExampleController', ['$scope', function($scope) {

$scope.name = 'Whirled';

}]);

</script>

<div ng-controller="ExampleController">

<label>Enter name: <input type="text" ng-model="name"></label><br>

Hello <span ng-bind="name"></span>!

</div>

**ng-repeat** : The ngRepeat directive instantiates a template once per item from a collection. Each template instance gets its own scope, where the given loop variable is set to the current collection item, and $index is set to the item index or key.

Example:

<header ng-repeat-start="item in items">

Header {{ item }}

</header>

<div class="body">

Body {{ item }}

</div>

<footer ng-repeat-end>

Footer {{ item }}

</footer>

**ng-init** : The ngInit directive allows you to evaluate an expression in the current scope.

ng-controller.

Example:

<div ng-app="" ng-init="quantity=1;cost=5">

<p>Total in dollar: {{ quantity \* cost }}</p>

</div>

How would $scope behave with nested controllers?

What is $rootScope? When would you use it?

What are the various filters?

How do I add a custom filter?

How would you go about form validation?

How can I load a template? Scenario - I have some HTML that gives me the necessary labels and inputs for the user to input their address. I'd like to make use of this as part of two other forms which are part of two separate views.

In what scenario would you make use of a value, constant, factory, service or provider?

How would one implement routing?

How do I make an AJAX request?

How would you handle an asynchronous request?

How would I add a custom directive?

What are the different criteria I can use to match?

How is link useful?

What does transclude do?

If there are multiple custom directives that match on an element, how would you control which one would apply first?