CSS

What is box model

CSS box model is a box that wraps around every single element in HTML which includes the content padding inside and then the border and the margin

<div></div>

div {

height:100px;

width:100px;

padding: 10px;

background:red;

border: 20px solid blue;

margin :10px;

}

By default every box is a content box

Box sizing: content -box. Which means that the content which is center part of it and the height and with is 100px\*100px which is inside and everything else is outside but the border box is if I say box sizing :

Box-size:border-box

Content becomes bit small now the border is part of the box itself so the content shrinks

Before it was like 100px height and 100 px width and the content was also 100px

Because of the border is 20px it now gives the border inside so the overall box is the same size adjustable the content shrinks to that 60 by 60

2. what is specificity?

For an element and if u have 2 conflicting style, so lets say if u have one styke that is changing the color or background color and u have second style changing the background color of the sam eelement then the browser has to decide which one to apply and it choses based on the specificity and the specificity is nothing but set of rules that it has

<div>name</div>

Div{

Background-color:green;

}

I wuld get name with green background color,

What if I have a same div but diff background color the n which one browser would choose

The ruleis if u have this kind of situation where the same element tage and it has different conflicting styles then it would pik the last one

Rule 2.

<div class=”name>name</div>

.name{

Background-color:green;

}

div.name{

Background-color:green;

}

In this case it should pick div.name because t is more specific then the regular div

Rule 3:

<div class=”name” id=”myId”>name</div>

Div#myiD{

Background-color:green;

}

ID WINS, because id has higher priority than class,

Rule 4:

What if I have !important

<div class=”name” id =”myiD”>name</div>

Div#myId {

Background-color: red;

}

div.name{

background-color: green !important

}

In this case it would pick green, enforcing this to pick it up

3. How to align a block element inside another element (aligning lining center)

<div class="out">

<div class="in">

</div>

</div>

.out {

width: 300px;

height: 300px;

background-color: yellow;

position: relative;

}

.in {

width: 100px;

height: 100px;

background-color: green;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%)

}

4. Difference between Static, Relative, Absolute and fixed position

<div class="one">one</div>

<br/>

<div class="two">two</div>

<br/>

<div class="three">three</div>

div {

width: 160px;

height: 40px;

padding: 20px;

border: 8px solid green;

background: yello;

}

I will get 4 box; by defalt they are all static, by the nature of the flow,

Which means if the 1st element here it will b stay there and second elemnt come after that, it follows the order

A picture containing graphical user interface

Description automatically generated

Now if I change 2 div to position: relative nothing would happen but if I apply top or left right bottom, to 50%

It would move 50% from original position, I will have 50p top, without affecting the element 1 and 3(lement 1 and 3 exactly where they were 2 is moving around top and bottom, the advantage is that

This doesn’t disturb environment as much compared to position absolute:

A picture containing graphical user interface

Description automatically generated

Position : absolute:

div {

width: 160px;

height: 40px;

padding: 20px;

border: 8px solid green;

background: yello;

}

.two{

position:absolute;

}

A screenshot of a computer screen

Description automatically generated

3 actually moved up as if 2 doesn’t exist 2 is not actually in separate space in a way so that disturbs the env,and it actually reference to its parent position if I say top 0 means it go upto parent

Graphical user interface

Description automatically generated

Position:fixed

Fixed is almost like absolut but its reference to the entire page not to its parant so it will stay wherever it is right now

Graphical user interface

Description automatically generated

2 actaully stays where it is and everything else is moving around it so (oif u scrolling down 2 stays whwre it is)

5. Difference between visibility:hidden and display:none

display:none

gap will not be there , element doesn’t exist inside the dom, so it doesn’t hold its position visibility hidden the elemnt is in the dom but its not visible

Graphical user interface, text, application

Description automatically generated

Visibility:hidden;

Only the element is hidden but the top part is there but it not being displayed, that’s why it is a gap here

Top part is there that’s why there I a gap at thetop

Graphical user interface, text, application

Description automatically generated

What is shadow DOM?

CSS is basically global which means If I write css for one element it can impact another element. Because it is global, often if we build web component the web component are built independenatly and they r used in diff projects so it means it uses its own local css so when I use in any project it shouldn’t impact the component so we can use shadow dom

Which is nothing but a scoped sub tree nside your elemnt

Graphical user interface, text, application

Description automatically generated

A screenshot of a computer screen

Description automatically generated

7. how to build triangle in css?

Text

Description automatically generated

Text

Description automatically generated

10. what are psudo element?

Psedo elements are used to give elements your selector and special effect and it will allow the some extra markup for the elemt without disturbing its envioronment

After:it add after the content

Before: it add befor the content

First later, firstline

Graphical user interface, text, application, email

Description automatically generated

Utility of this: if I want to inject tooltip on an elemnt I can do it by injecting a conent of th tooltip basically it shows it when I hover on the elemnt

:hover: single colon is pseudo clas and double colon is spedo elemt

What are data Attribute? And the tilities:

In HTML if you want to store data u can use data attribute because regular attributes are not suitable to store data and data, data attribute are designed to staore data in

Graphical user interface, text

Description automatically generated

Graphical user interface, text, email

Description automatically generated

What is the difference between classes and IDs in CSS? What's the difference between "resetting" and "normalizing" CSS? Which would you choose, and why? Describe Floats and how they work. Describe z-index and how stacking context is formed. Describe BFC(Block Formatting Context) and how it works. What are the various clearing techniques and which is appropriate for what context? Explain CSS sprites, and how you would implement them on a page or site. What are your favourite image replacement techniques and which do you use when? How would you approach fixing browser-specific styling issues? How do you serve your pages for feature-constrained browsers? What techniques/processes do you use? What are the different ways to visually hide content (and make it available only for screen readers)? Have you ever used a grid system, and if so, what do you prefer? Have you used or implemented media queries or mobile specific layouts/CSS? Are you familiar with styling SVG? How do you optimize your webpages for print? What are some of the "gotchas" for writing efficient CSS? What are the advantages/disadvantages of using CSS preprocessors? Describe what you like and dislike about the CSS preprocessors you have used. How would you implement a web design comp that uses non-standard fonts? Explain how a browser determines what elements match a CSS selector. Describe pseudo-elements and discuss what they are used for. Explain your understanding of the box model and how you would tell the browser in CSS to render your layout in different box models. What does \* { box-sizing: border-box; } do? What are its advantages? List as many values for the display property that you can remember. What's the difference between inline and inline-block? What's the difference between a relative, fixed, absolute and statically positioned element? The 'C' in CSS stands for Cascading. How is priority determined in assigning styles (a few examples)? How can you use this system to your advantage? What existing CSS frameworks have you used locally, or in production? How would you change/improve them? Have you played around with the new CSS Flexbox or Grid specs? How is responsive design different from adaptive design? Have you ever worked with retina graphics? If so, when and what techniques did you use? Is there any reason you'd want to use translate() instead of absolute positioning, or vice-versa? And why?

<https://codeburst.io/clearing-your-front-end-job-interview-css-95bdd82871f2>

1. What is the difference between classes and IDs in CSS?

When comparing CSS class vs ID, the difference is that CSS class applies a style to multiple elements. ID, on the other hand, applies a style to one unique element. ID is also special in that you can use a special URL to link directly to an element and it’s used by JavaScript.

* **IDs** — Meant to be unique within the document. Can be used to identify an element when linking using a fragment identifier. Elements can only have one id attribute.
* **Classes** — Can be reused on multiple elements within the document. Mainly for styling and targeting elements.
  1. What’s the difference between “resetting” and “normalizing” CSS? Which would you choose, and why?
* **Resetting CSS**
* Browser uses it’s internal(build-in) CSS for some HTML element like margin, padding, list-style, font-size etc . In order to make HTML page from scratch you need to override these styles.
* **Normalize CSS**
* Normalize CSS means “What You Write is What You See/Get” across all browsers. e.g If you write CSS for h1,h2,h3,h4 like font-size: “14px”, font-weight: 600 . then all hq,h2,h3.h4 will look same across all borwsers
* **Resetting** — Resetting is meant to strip all default browser styling on elements. For e.g. margins, paddings, font-sizes of all elements are reset to be the same. You will have to redeclare styling for common typographic elements.
* **Normalizing** — Normalizing preserves useful default styles rather than “unstyling” everything. It also corrects bugs for common browser dependencies.

I would choose resetting when I have very a customized or unconventional site design such that I need to do a lot of my own styling do not need any default styling to be preserved.

## Describe floats and how they work.

Float is a CSS positioning property. Floated elements remain a part of the flow of the page, and will affect the positioning of other elements (e.g. text will flow around floated elements), unlike position: absolute elements, which are removed from the flow of the page.

The CSS clear property can be used to be positioned below left/right/both floated elements.

If a parent element contains nothing but floated elements, its height will be collapsed to nothing. It can be fixed by clearing the float after the floated elements in the container but before the close of the container.

The .clearfix hack uses a clever CSS pseudo selector (:after) to clear floats. Rather than setting the overflow on the parent, you apply an additional class like clearfix to it. Then apply this CSS:

.clearfix:after {  
 content: '.';  
 visibility: hidden;  
 display: block;  
 height: 0;  
 clear: both;  
}

Alternatively, give overflow: auto or overflow: hidden property to the parent element which will establish a new block formatting context inside the children and it will expand to contain its children.

## Describe z-index and how stacking context is formed.

The z-index property in CSS controls the vertical stacking order of elements that overlap. z-index only effects elements that have a position value which is not static.

Without any z-index value, elements stack in the order that they appear in the DOM (the lowest one down at the same hierarchy level appears on top). Elements with non-static positioning (and their children) will always appear on top of elements with default static positioning, regardless of HTML hierarchy.

A stacking context is an element that contains a set of layers. Within a local stacking context, the z-index values of its children are set relative to that element rather than to the document root. Layers outside of that context — i.e. sibling elements of a local stacking context — can't sit between layers within it. If an element B sits on top of element A, a child element of element A, element C, can never be higher than element B even if element C has a higher z-index than element B.

Each stacking context is self-contained — after the element’s contents are stacked, the whole element is considered in the stacking order of the parent stacking context. A handful of CSS properties trigger a new stacking context, such as opacity less than 1, filter that is not none, and transform that is notnone`.

## Describe Block Formatting Context (BFC) and how it works.

A Block Formatting Context (BFC) is part of the visual CSS rendering of a web page in which block boxes are laid out. Floats, absolutely positioned elements, inline-blocks, table-cells, table-captions, and elements with overflowother than visible (except when that value has been propagated to the viewport) establish new block formatting contexts.

A BFC is an HTML box that satisfies at least one of the following conditions:

* The value of float is not none.
* The value of position is neither static nor relative.
* The value of display is table-cell, table-caption, inline-block, flex, or inline-flex.
* The value of overflow is not visible.

In a BFC, each box’s left outer edge touches the left edge of the containing block (for right-to-left formatting, right edges touch).

Vertical margins between adjacent block-level boxes in a BFC collapse. Read more on [collapsing margins](https://www.sitepoint.com/web-foundations/collapsing-margins/).

## What are the various clearing techniques and which is appropriate for what context?

* Empty div method - <div style="clear:both;"></div>.
* Clearfix method — Refer to the .clearfix class above.
* overflow: auto or overflow: hidden method - Parent will establish a new block formatting context and expand to contains its floated children.

In large projects, I would write a utility .clearfix class and use them in places where I need it. overflow: hiddenmight clip children if the children is taller than the parent and is not very ideal.

## What are the different ways to visually hide content (and make it available only for screen readers)?

These techniques are related to accessibility (a11y).

* visibility: hidden. However the element is still in the flow of the page, and still takes up space.
* width: 0; height: 0. Make the element not take up any space on the screen at all, resulting in not showing it.
* position; absolute; left: -99999px. Position it outside of the screen.
* text-indent: -9999px. This only works on text within the block elements.

I would go with the `absolute` positioning approach, as it has the least caveats and works for most elements.

## How do you optimize your webpages for print?

* Create a stylesheet for print or use media queries.

<!-- Main stylesheet on top -->  
<link rel="stylesheet" href="/global.css" media="all" />  
<!-- Print only, on bottom -->  
<link rel="stylesheet" href="/print.css" media="print" />

Make sure to put non-print styles inside @media screen { ... }.

@media print {  
 ...  
}

* Deliberately add page breaks.

<style>  
.page-break {   
 display: none;  
 page-break-before: always;   
}  
</style>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Fusce eu felis. Curabitur sit amet magna. Nullam aliquet. Aliquam ut diam...  
<div class="page-break"></div>  
Lorem ipsum dolor sit amet, consectetuer adipiscing elit....

## Describe pseudo-elements and discuss what they are used for.

A CSS pseudo-element is a keyword added to a selector that lets you style a specific part of the selected element(s). They can be used for decoration (:first-line, :first-letter) or adding elements to the markup (combined with content: ...) without having to modify the markup (:before, :after).

* :first-line and :first-letter can be used to decorate text.
* Used in the .clearfix hack as shown above to add a zero-space element with clear: both.
* Triangular arrows in tooltips use :before and :after. Encourages separation of concerns because the triangle is considered part of styling and not really the DOM, but not really possible to draw a triangle with just CSS styles

## Explain your understanding of the box model and how you would tell the browser in CSS to render your layout in different box models.

The CSS box model is responsible for calculating:

* How much space a block-level element takes up.
* Whether or not borders and/or margins overlap, or collapse.
* A box’s dimensions.

The box model has the following rules:

* The dimensions of a block element are calculated by width, height, padding, borders, and margins.
* If no height is specified, a block element will be as high as the content it contains, plus padding (unless there are floats, for which see below).
* If no width is specified, a non-floated block element will expand to fit the width of its parent minus padding.
* The height of an element is calculated by the content's height.
* The width of an element is calculated by the content's width.
* By default, paddings and borders are not part of the width and height of an element.

## What does \* { box-sizing: border-box; } do? What are its advantages?

* By default, elements have box-sizing: content-box applied, and only the content size is being accounted for.
* box-sizing: border-box changes how the width and height of elements are being calculated, border and padding are also being included in the calculation.
* The height of an element is now calculated by the content's height + vertical padding + vertical border width.
* The width of an element is now calculated by the content's width + horizontal padding + horizontal borderwidth.

## List as many values for the display property that you can remember.

* none, block, inline, inline-block, table, table-row, table-cell, list-item.

## What’s the difference between inline and inline-block?

I shall throw in a comparison with block for good measure.

**Block**

* Size — Fills up the width of its parent container.
* Positioning — Start on a new line and tolerates no HTML elements next to it (except when you add float).
* Can specify width and height — Yes.
* Can be aligned with vertical-align — Yes.
* Margins and paddings — All sides respected.

**Inline-Block**

* Size — Depends on content.
* Positioning — Flows along with other content and allows other elements beside.
* Can specify width and height — Yes.
* Can be aligned with vertical-align — Yes.
* Margins and paddings — All sides respected.

**Inline**

* Size — Depends on content.
* Positioning — Flows along with other content and allows other elements beside.
* Can specify width and height — No. Will ignore if being set.
* Can be aligned with vertical-align — Only horizontal sides respected. Vertical sides, if specified, do not affect layout. Vertical space it takes up depends on line-height, even though the borderand padding appear visually around the content.
* Margins and paddings — Becomes like a block element where you can set vertical margins and paddings.

## What’s the difference between a relative, fixed, absolute and static-ally positioned element?

A positioned element is an element whose computed position property is either relative, absolute, fixed or sticky.

* static - The default position; the element will flow into the page as it normally would. The top, right, bottom, left and z-index properties do not apply.
* relative - The element's position is adjusted relative to itself, without changing layout (and thus leaving a gap for the element where it would have been had it not been positioned).
* absolute - The element is removed from the flow of the page and positioned at a specified position relative to its closest positioned ancestor if any, or otherwise relative to the initial containing block. Absolutely positioned boxes can have margins, and they do not collapse with any other margins. These elements do not affect the position of other elements.
* fixed - The element is removed from the flow of the page and positioned at a specified position relative to the viewport and doesn't move when scrolled.
* sticky - Sticky positioning is a hybrid of relative and fixed positioning. The element is treated as relative positioned until it crosses a specified threshold, at which point it is treated as fixed positioned.

## What existing CSS frameworks have you used locally, or in production? How would you change/improve them?

* **Bootstrap** — Slow release cycle. Bootstrap 4 has been in alpha for almost 2 years. Add a spinner button component, as it is widely-used.
* **Semantic UI**— Source code structure makes theme customization is extremely hard to understand. Painful to customize with unconventional theming system. Hardcoded config path within the vendor library. Not well-designed for overriding variables unlike in Bootstrap.
* **Bulma** — A lot of non-semantic and superfluous classes and markup required. Not backward compatible. Upgrading versions breaks the app in subtle manners.

## Have you played around with the new CSS Flexbox or Grid specs?

Yes. Flexbox is mainly meant for 1-dimensional layouts while Grid is meant for 2-dimensional layouts.

Flexbox solves many common problems in CSS, such as vertical centering of elements within a container, sticky footer, etc. Bootstrap and Bulma are based on Flexbox, and it is probably the recommended way to create layouts these days. Have tried Flexbox before but ran into some browser incompatibility issues (Safari) in using `flex-grow`, and I had to rewrite my code using `inline-blocks` and math to calculate the widths in percentages, it wasn’t a nice experience.

Grid is by far the most intuitive approach for creating grid-based layouts (it better be!) but browser support is not wide at the moment

## Is there any reason you’d want to use translate() instead of absolute positioning, or vice-versa? And why?

translate() is a value of CSS transform. Changing transform or opacity does not trigger browser reflow or repaint, only compositions, whereas changing the absolute positioning triggers reflow. transform causes the browser to create a GPU layer for the element but changing absolute positioning properties uses the CPU. Hence translate() is more efficient and will result in shorter paint times for smoother animations.

When using translate(), the element still takes up its original space (sort of like position: relative), unlike in changing the absolute positioning.