What is the DOM?

The DOM (Document Object Model) is a cross-platform API that treats HTML and XML documents as a tree structure consisting of nodes. These nodes (such as elements and text nodes) are objects that can be programmatically manipulated and any visible changes made to them are reflected live in the document. In a browser, this API is available to JavaScript where DOM nodes can be manipulated to change their styles, contents, placement in the document, or interacted with through event listeners.

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 affects 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 not none.

Why is it generally a good idea to position CSS `<link>` s between `<head></head>` and JS `<script>` s just before `</body>`? Do you know any exceptions?

- The main reason as to why JS files are linked at the bottom of the body is because whenever a browser encounters any JS, it parses it and executes that on the spot. Hence if it was to be added at the top, it would make the page

rendering slow and thus it would take more time for page load. Moreover since the DOM won't be rendered fully, JS won't be able to manipulate the elements.

- However if you use Jquery, that won't be an issue since it would execute only after the document is ready. But since in any case, the browser would parse it, it would slow the page load.
- On the contrary, CSS files are linked in the head because they get applied regardless of DOM already rendered or not. Hence the webpage looks elegant as soon as the page loads. However just like JS you can link the CSS at the end which would mean that the webpage first loads with just plain HTML and then the CSS is applied to it. This shift is clearly visible to the user and moreover an important thing to remember is that the page would load with bare minimum HTML and if the user has slow Internet connection, the CSS load would take considerable amount of time, which means that the webpage shows just the HTML meanwhile. This might make the user close the website without waiting for it to load fully.
- To avoid such things, a CSS file is linked at the head while a JS file is linked at the bottom.

What is HTML5 Web Storage? Explain localStorage and sessionStorage.

With HTML5, web pages can store data locally within the user's browser. The data is stored in name/value pairs, and a web page can only access data stored by itself.

Differences between `localStorage` and `sessionStorage` regarding lifetime:

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 data got stored. When the tab is permanently closed, any data stored through sessionStorage is deleted.
Differences between `localStorage` and `sessionStorage` regarding storage scope: Both forms of storage are scoped to the document origin so that documents with different origins will never share the stored objects.

sessionStorage is also scoped on a per-window basis. Two browser tabs with documents from the same origin have separate sessionStorage data.
Unlike in `localStorage`, the same scripts from the same origin can't access each other's `sessionStorage` when opened in different tabs.

Briefly describe the correct usage of the following HTML5 semantic elements: <header>, <article>,<section>, <footer>

<header> is used to contain introductory and navigational information about a section of the page. This can include the section heading, the author's name, time and date of publication, table of contents, or other navigational information.

<article> is meant to house a self-contained composition that can logically be independently recreated outside of the page without losing its meaning. Individual blog posts or news stories are good examples.

<section> is a flexible container for holding content that shares a common informational theme or purpose.

<footer> is used to hold information that should appear at the end of a section of content and contain additional information about the section. Author's name, copyright information, and related links are typical examples of such content.

What are the building blocks of HTML5?

Semantics: allowing you to describe more precisely what your content is.
Connectivity: allowing you to communicate with the server in new and innovative ways.

Offline and storage: allowing web pages to store data on the client-side locally and operate offline more efficiently.

Multimedia: making video and audio first-class citizens in the Open Web.
2D/3D graphics and effects: allowing a much more diverse range of presentation options.

Performance and integration: providing greater speed optimization and better usage of computer hardware.

Device access: allowing for the usage of various input and output devices.
Styling: letting authors write more sophisticated themes.

What is the difference between a and a ?

** ** is a block level element which means it will render it on it's own line with a width of a 100% of the parent element.

**** is an inline element which means it will render on the same line as the previous element, if it is also an inline element, and it's width will be determined by its content

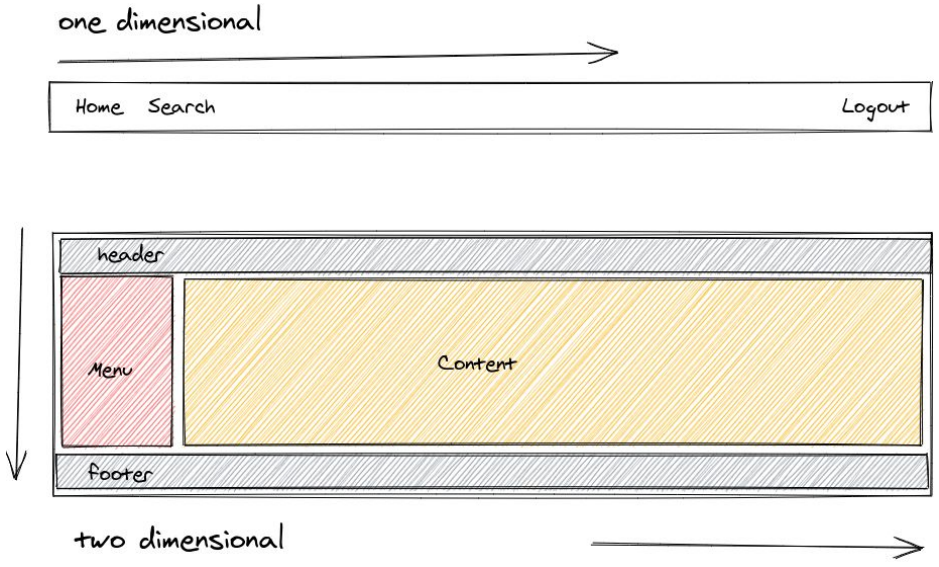
When to use css grid and flexbox?

CSS Grid Layout is a **two-dimensional** system, meaning it can handle both columns and rows, unlike flexbox which is largely a **one-dimensional** system (either in a column or a row).

A core difference between CSS Grid and Flexbox is that — CSS Grid's approach is **layout-first** while Flexbox' approach is **content-first**. If you are well aware of your content before making layout, then blindly opt for Flexbox and if not, opt for CSS Grid.

Flexbox layout is most appropriate to the components of an application (as most of them are fundamentally linear), and **small-scale** layouts, while the Grid layout is intended for **larger-scale** layouts which aren't linear in their design.

If you only need to define a layout as a row or a column, then you probably need flexbox. If you want to define a grid and fit content into it in two dimensions — you need the grid.



How to make page responsive?

Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones).

Setting the viewport

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Responsive Images

If the CSS width property is set to 100%, the image will be responsive and scale up and down

Show different Images depending on Browser Width

The HTML `<picture>` element allows you to define different images for different browser window sizes.

Responsive Text Size

The text size can be set with a "vw" unit, which means the "viewport width". That way the text size will follow the size of the browser window.

Media Queries

Using media queries you can define completely different styles for different browser sizes.

What is difference between Select and Datalist?

For the select element, the user is required to select one of the options you've given. For the datalist element, it is suggested that the user select one of the options you've given, but he can actually enter anything he wants in the input.

Select-Option

```
<select name="browser">
  <option value="firefox">Firefox</option>
  <option value="ie">IE</option>
  <option value="chrome">Chrome</option>
```

```
  <option value="opera">Opera</option>
  <option value="safari">Safari</option>
</select>
```

Datalist-Option

```
<input type="text" list="browsers">
<datalist id="browsers">
  <option value="Firefox">
  <option value="IE">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

What are optional closing tag?

`<p>, <li>, <td>, <tr>, <th>, <html>, <body>`, etc. don't have to provide end tag. Whenever browser hits a new tag it automatically ends the previous tag.

Why you would like to use semantic tag?

- Search Engine Optimization, accessibility, repurposing, light code.
- Many visually impaired person rely on browser speech and semantic tag helps to interpret page content clearly.
- Search engine needs to understand page content to rank and semantic tag helps.
- Semantic code aids accessibility. Specially, many people whose eyes are not good rely on speech browsers to read pages to them. These programs cannot interpret pages very well unless they are clearly explained.
- Help Search engines to better understand pages. Search engine need to understand what your content is about when rank you properly on search engines. Semantic code tends to improve your placement on search engines, as it is easier for the "search engine spiders" to understand.
- It’s easier to read and edit, which saves time and money during maintenance.

Explain the difference between layout, painting and compositing?

JavaScript: Typically JavaScript is used to handle work that will result in visual changes, whether it's jQuery's animate function, sorting a data set, or adding DOM elements to the page. It doesn't have to be JavaScript that triggers a visual change, though: CSS Animations, Transitions, and the Web Animations API are also commonly used.

Style calculations: This is the process of figuring out which CSS rules apply to which elements based on matching selectors, for example, .headline or .nav > .nav__item. From there, once rules are known, they are applied and the final styles for each element are calculated.

Layout: Once the browser knows which rules apply to an element it can begin to calculate how much space it takes up and where it is on screen. The web's layout model means that one element can affect others, for example the width of the element typically affects its children's widths and so on all the way up and down the tree, so the process can be quite involved for the browser.

Paint: Painting is the process of filling in pixels. It involves drawing out text, colors, images, borders, and shadows, essentially every visual part of the elements. The drawing is typically done onto multiple surfaces, often called layers.

Compositing: Since the parts of the page were drawn into potentially multiple layers they need to be drawn to the screen in the correct order so that the page renders correctly. This is especially important for elements that overlap another, since a mistake could result in one element appearing over the top of another incorrectly.

The difference between block, inline and inline-block element?

Block Elements

The block elements always start on a new line. They will also take space of an entire row or width. List of block elements are `<p>, <h1>, <div>, <header>.`

Example:

```
<p>
  Lorem ipsum dolor sit amet consectetur adipisicing elit.
Unde autem,
  consequatur deleniti nobis beatae quo dolore nemo corporis.
Ad delectus
  dignissimos pariatur illum eveniet dolor rem eius laborum
sed iure!
</p>

<p>
  Lorem ipsum dolor sit amet consectetur adipisicing elit.
Unde autem,
  consequatur deleniti nobis beatae quo dolore nemo corporis.
Ad delectus
  dignissimos pariatur illum eveniet dolor rem eius laborum
sed iure!
</p>
```

Inline Elements

Inline elements don't start on a new line, they appear on the same line as the content and tags beside them. Some examples of inline elements are `<a>`, `<span>` , `<strong>`, and `<img>` tags.

When it comes to margins and padding, browsers treat inline elements differently. You can add space to the left and right on an inline element, but you cannot add height to the top or bottom padding or margin of an inline element.

```
<a href="#">Link</a>
<img src="https://picsum.photos/30" />
<span>Span</span>
<strong>Strong Player</strong>
```

Inline-Block Elements

Inline-block elements are similar to inline elements, except they can have padding and margins added on all four sides. One common use for using inline-block is for creating navigation links horizontally. Some examples of inline-block elements are `<input>`, `<button>`, `<select>`, `<textarea>` etc.

```
input {
  width: 300px;
  height: 50px;
}

button {
  width: 100px;
  height: 50px;
  margin-top: 20px;
}

<input type="text" /> <button>Submit</button>
```

What is pseudo element?

Pseudo Element: A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

Style the first letter, or line, of an element
Insert content before, or after, the content of an element

Example **CSS Pseudo Elements:**
https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements

What is pseudo class?

Pseudo-classes: A pseudo-class is used to define a special state of an element.

For example, it can be used to:

Style an element when a user mouses over it

Style visited and unvisited links differently

Style an element when it gets focus

Example **CSS Pseudo Classes**
https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes

What elements will match each of the following CSS selectors

div, p Selects all elements and all elements

div p Selects all <p> elements that are anywhere inside a element
div > p Selects all <p> elements where the immediate parent is a element
div + p Selects all <p> elements that are placed immediately after a element
div ~ p Selects all <p> elements that are anywhere preceded by a Element

Explain the meaning of each of these CSS units for expressing length:

cm centimeters em elements (i.e., relative to the font-size of the element; e.g., 2 em means 2 times the current font size) in inches mm millimeters pc picas (1 pc = 12 pt = 1/6th of an inch) pt points (1 pt = 1/72nd of an inch) px pixels (1 px = 1/96th of an inch)

In CSS3, how would you select

Every <a> element whose href attribute value begins with “https”.

```
a[href^="https"]
```

Every <a> element whose href attribute value ends with “.pdf”.

```
a[href$=".pdf"]
```

Every <a> element whose href attribute value contains the substring “css”.

```
a[href*="css"]
```

What is the purpose of the box-sizing property?

The box-sizing CSS property sets how the total width and height of an element is calculated.

content-box: the default width and height values apply to the element's content only. The padding and border are added to the outside of the box.

padding-box: Width and height values apply to the element's content and its padding. The border is added to the outside of the box. Currently, only Firefox supports the padding-box value.

border-box: Width and height values apply to the content, padding, and border. inherit: inherits the box sizing of the parent element.

Example:

```
box-sizing: content-box;
width: 100%;
border: solid rgb(90,107,204) 10px;
padding: 5px;
```

How to create a zebra striped table with CSS?

To create a zebra-striped table, use the nth-child() selector and add a background-color to all even (or odd) table rows:

```
tr:nth-child(even) {
    background-color: #f2f2f2
}
```

What is the difference between RGBa, HEX and HSLa?

RGB (Red/Green/Blue) is a color model.

```
p {
  color: rgba(37, 84, 127, 1);
}
```

HEX (Hexadecimal color values)

```
p {
  color: #25547f;
}
```

HSLa (Hue Saturation Lightness alpha)

```
p {
  color: hsla(209, 55%, 32%, 1);
}
```

What is CSS preprocessor?

Pre-processors extend CSS with variables, operators, interpolations, functions, mixins and many more other usable assets. After development, these specific files are compiled into regular CSS that any browser can understand. Pre-processor help writing reusable, easily maintainable and extensible codes in CSS.

CSS preprocessors

- SASS (SCSS)
- LESS
- Stylus
- PostCSS

Advantages:

- CSS is made more maintainable.
- Easy to write nested selectors.
- Variables for consistent theming. Can share theme files across different projects.
- Mixins to generate repeated CSS.
- Splitting your code into multiple files. CSS files can be split up too but doing so will require an HTTP request to download each CSS file

What is the difference between "resetting" and "normalizing" CSS?

1. Resetting: CSS resets aim to remove all built-in browser styling. For example margins, paddings, font-sizes of all elements are reset to be the same. You will have to redeclare styling for common typographic elements.

Example

```
html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p,
blockquote, pre, a, abbr,
acronym, address, big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp, small,
strike, strong,
```

```
sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form, label,
legend, table,
caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas, details, embed,
figure, figcaption,
footer, header, hgroup,  menu, nav, output, ruby, section, summary, time, mark,
audio, video {
   margin: 0;
   padding: 0;
   border: 0;
   font-size: 100%;
   font: inherit;
   vertical-align: baseline;
}
```

2. Normalizing: Normalize CSS aims to make built-in browser styling consistent across browsers. It also corrects bugs for common browser dependencies.

```
/*
  Correct the font size and margin on `h1` elements within `section`
  and `article` contexts in Chrome, Firefox, and Safari.
*/
 h1 {  font-size: 2em;  margin: 0.67em 0;}
```

Describe clear Property in CSS?

The clear property specifies what elements can float beside the cleared element and on which side.

Sl.No	Properties	Description
01.	clear: none	Allows floating elements on both sides. This is default
02.	clear: left	No floating elements allowed on the left side
03.	clear: right	No floating elements allowed on the right side
04.	clear: both	No floating elements allowed on either the left or the right side
05.	clear: inherit	The element inherits the clear value of its parent

What is a focus ring? What is the correct solution to handle them?

A focus ring is a visible outline given to focusable elements such as buttons and anchor tags. It varies depending on the vendor, but generally it appears as a blue outline around the element to indicate it is currently focused.

In the past, many people specified outline: 0; on the element to remove the focus ring. However, this causes accessibility issues for keyboard users because the focus state may not be clear. When not specified though, it causes an unappealing blue ring to appear around an element.

In recent times, frameworks like Bootstrap have opted to use a more appealing box-shadow outline to replace the default focus ring. However, this is still not ideal for mouse users.

The best solution is an upcoming pseudo-selector :focus-visible which can be polyfilled today with JavaScript. It will only show a focus ring if the user is using a keyboard and leave it hidden for mouse users. This keeps both aesthetics for mouse use and accessibility for keyboard use.

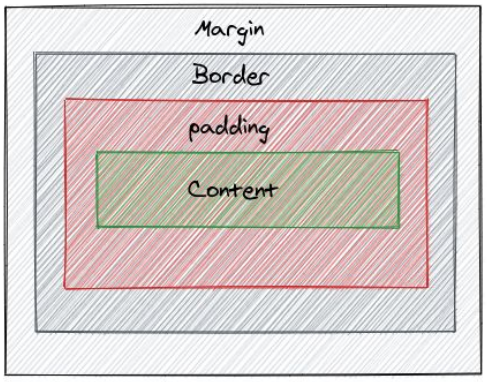
Describe the layout of the CSS Box Model and briefly describe each component.

Content: The inner-most part of the box filled with content, such as text, an image, or video player. It has the dimensions content-box width and content-box height.

Padding: The transparent area surrounding the content. It has dimensions padding-box width and padding-box height.

Border: The area surrounding the padding (if any) and content. It has dimensions border-box width and border-box height.

Margin: The transparent outer-most layer that surrounds the border. It separates the element from other elements in the DOM. It has dimensions margin-box width and margin-box height.



What is the difference between em and rem units?

Both **em** and **rem** units are based on the font-size CSS property. The only difference is where they inherit their values from.

em units inherit their value from the **font-size** of the parent element
rem units inherit their value from the **font-size** of the root element (html)
In most browsers, the **font-size** of the root element is set to **16px** by default.

Using flexbox, create a 3-column layout where each column takes up a col-{n} / 12 ratio of the container.

```
<div class="row">
  <div class="col-2"></div>
  <div class="col-7"></div>
  <div class="col-3"></div>
</div>
```

Set the .row parent to display: flex; and use the flex shorthand property to give the column classes a flex-grow value that corresponds to its ratio value.

```
.row {
  display: flex;
}

.col-2 {
  flex: 2;
}

.col-7 {
  flex: 7;
}

.col-3 {
  flex: 3;
}
```

Can you name the four types of @media properties?

all, which applies to all media type devices

print, which only applies to printers
screen, which only applies to screens (desktops, tablets, mobile etc.)
speech, which only applies to screen readers

What are the advantages of using CSS sprites and how are they utilized?

CSS sprites combine multiple images into one image, limiting the number of HTTP requests a browser has to make, thus improving load times. Even under the new HTTP/2 protocol, this remains true.

Under HTTP/1.1, at most one request is allowed per TCP connection. With HTTP/1.1, modern browsers open multiple parallel connections (between 2 to 8) but it is limited. With HTTP/2, all requests between the browser and the server are multiplexed on a single TCP connection. This means the cost of opening and closing multiple connections is mitigated, resulting in a better usage of the TCP connection and limits the impact of latency between the client and server. It could then become possible to load tens of images in parallel on the same TCP connection.

However, according to benchmark results, although HTTP/2 offers 50% improvement over HTTP/1.1, in most cases the sprite set is still faster to load than individual images.

To utilize a spritesheet in CSS, one would use certain properties, such as background-image, background-position and background-size to ultimately alter the background of an element.

Can you describe how CSS specificity works?

Assuming the browser has already determined the set of rules for an element, each rule is assigned a matrix of values, which correspond to the following from highest to lowest specificity:

- Inline rules (binary - 1 or 0)
- Number of id selectors
- Number of class, pseudo-class and attribute selectors
- Number of tags and pseudo-element selectors

When two selectors are compared, the comparison is made on a per-column basis (e.g. an id selector will always be higher than any amount of class selectors, as ids have higher specificity than classes). In cases of equal specificity between multiple

rules, the rules that comes last in the page's style sheet is deemed more specific and therefore applied to the element.

What is the purpose of the alt attribute on images?

The alt attribute provides alternative information for an image if a user cannot view it. The alt attribute should be used to describe any images except those which only serve a decorative purpose, in which case it should be left empty.

What is the purpose of cache busting and how can you achieve it?

Browsers have a cache to temporarily store files on websites so they don't need to be re-downloaded again when switching between pages or reloading the same page. The server is set up to send headers that tell the browser to store the file for a given amount of time. This greatly increases website speed and preserves bandwidth.

However, it can cause problems when the website has been changed by developers because the user's cache still references old files. This can either leave them with old functionality or break a website if the cached CSS and JavaScript files are referencing elements that no longer exist, have moved or have been renamed.

Cache busting is the process of forcing the browser to download the new files. This is done by naming the file something different to the old file.

A common technique to force the browser to re-download the file is to append a query string to the end of the file.

```
src="js/script.js" => src="js/script.js?v=2"
```

The browser considers it a different file but prevents the need to change the file name.

Can a web page contain multiple <header> elements? What about <footer> elements?

Yes to both. The W3 documents state that the tags represent the header(<header>) and footer(<footer>) areas of their nearest ancestor "section". So not only can the page <body> contain a header and a footer, but so can every <article> and <section> element.

What are defer and async attributes on a <script> tag?

If neither attribute is present, the script is downloaded and executed synchronously, and will halt parsing of the document until it has finished executing (default behavior). Scripts are downloaded and executed in the order they are encountered.

The defer attribute downloads the script while the document is still parsing but waits until the document has finished parsing before executing it, equivalent to executing inside a DOMContentLoaded event listener. defer scripts will execute in order.

The async attribute downloads the script during parsing the document but will pause the parser to execute the script before it has fully finished parsing. async scripts will not necessarily execute in order.

Note: both attributes must only be used if the script has a src attribute (i.e. not an inline script).

```
<script src="myscript.js"></script>
<script src="myscript.js" defer></script>
<script src="myscript.js" async></script>
```

Discuss the differences between an HTML specification and a browser's implementation thereof.

HTML specifications such as HTML5 define a set of rules that a document must adhere to in order to be "valid" according to that specification. In addition, a specification provides instructions on how a browser must interpret and render such a document.

A browser is said to "support" a specification if it handles valid documents according to the rules of the specification. As of yet, no browser supports all aspects of the HTML5 specification (although all of the major browser support most of it), and as a result, it is necessary for the developer to confirm whether the aspect they are making use of will be supported by all of the browsers on which they hope to display their content. This is why cross-browser support continues to be a headache for developers, despite the improved specifications.

What is the difference between HTML and React event handling?

In HTML, the attribute name is in all lowercase and is given a string invoking a function defined somewhere:

```
<button onclick="handleClick()"></button>
```

In React, the attribute name is camelCase and are passed the function reference inside curly braces:

```
<button onClick={handleClick} />
```

In HTML, false can be returned to prevent default behavior, whereas in React preventDefault has to be called explicitly.

```
<a href="#" onclick="console.log('The link was clicked.');
return false" />
function handleClick(e) {
  e.preventDefault()
  console.log("The link was clicked.")
}
```

What are some differences that XHTML has compared to HTML?

Some of the key differences are:

- An XHTML element must have an XHTML <DOCTYPE>
- Attributes values must be enclosed in quotes
- Attribute minimization is forbidden (e.g. one has to use checked="checked" instead of checked)
- Elements must always be properly nested
- Elements must always be closed
- Special characters must be escaped

Where and why is the rel="noopener" attribute used?

The rel="noopener" is an attribute used in <a> elements (hyperlinks). It prevents pages from having a window.opener property, which would otherwise point to the page from where the link was opened and would allow the page opened from the hyperlink to manipulate the page where the hyperlink is.

What's the difference between a relative, fixed, absolute and statically 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 extremely hard to understand. Its unconventional theming system is a pain to customize. 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.

How would you implement a web design comp that uses non-standard fonts?

Use @font-face and define font-family for different font-weights.

Explain how a browser determines what elements match a CSS selector.

This part is related to the above about writing efficient CSS. Browsers match selectors from rightmost (key selector) to left. Browsers filter out elements in the DOM according to the key selector and traverse up its parent elements to determine matches. The shorter the length of the selector chain, the faster the browser can determine if that element matches the selector.

For example with this selector p span, browsers firstly find all the elements and traverse up its parent all the way up to the root to find the <p> element. For a particular , as soon as it finds a <p>, it knows that the matches and can stop its matching.

What are the building blocks of HTML5?

- **Semantics:** allowing you to describe more precisely what your content is.
- Connectivity: allowing you to communicate with the server in new and innovative ways.
- **Offline and storage:** allowing webpages to store data on the client-side locally and operate offline more efficiently.
- **Multimedia**: making video and audio first-class citizens in the Open Web.
- 2D/3D graphics and effects: allowing a much more diverse range of presentation options.
- **Performance and integration:** providing greater speed optimization and better usage of computer hardware.
- **Device access:** allowing for the usage of various input and output devices.
- **Styling:** letting authors write more sophisticated themes.

What is Critical Rendering Path?

- Constructing the DOM Tree
- Constructing the CSSOM Tree
- Running JavaScript - parser blocking resource
- Creating the Render Tree
- Generating the Layout
- Painting

What is the purpose of main element?

The HTML <main> element represents the dominant content of the of a document. The main content area consists of content that is directly related to or expands upon the central topic of a document, or the central functionality of an application.

```
<main role="main">
    <p>Geckos are a group of usually small, usually nocturnal
lizards.
       They are found on every continent except Australia.</p>
    <p>Many species of gecko have adhesive toe pads which
enable them to climb walls and even windows.</p>
</main>
```

Note: A document mustn't have more than one <main> element that doesn't have the hidden attribute specified.

Name 3 ways to decrease page load?

- LocalStorage
- Caching resources
- DNS-prefetch (sample below)
- Keep resources on a CDN

Explain some of the pros and cons for CSS animations versus JavaScript animations.

Regarding optimization and responsiveness the debate bounces back and forth but, the concept is:

CSS animations allows the browser to choose where the animation processing is done, CPU or the GPU. (Central or Graphics Processing Unit)

That said, adding many layers to a document will eventually have a performance hit.

JS animation means more code for the user to download and for the developer to maintain.

Applying multiple animation types on an element is harder with CSS since all transforming power is in one property transform

CSS animations being declarative are not programmable therefore limited in capability.

What does CORS stand for and what issue does it address?

Cross-Origin Resource Sharing (CORS) is a W3C spec that allows cross-domain communication from the browser. By building on top of the XMLHttpRequest object, CORS allows developers to work with the same idioms as same-domain requests. CORS gives web servers cross-domain access controls, which enable secure cross-domain data transfers.

What is desktop first and mobile first design approach

Desktop first : General selectors and styles designed to make the site look good on DESKTOP screens defined globally. But they affect all devices, and must be overridden by max-width media queries targeting minimum screen size

Mobile First : General selectors and styles designed to make the site look good on small MOBILE screens go here. But they affect all devices, and must be overridden by min-width media queries targeting maximum scrren size

In desktop first approach the media queries will be written with respect to max-width whereas in mobile first approach media queries will be written with respect to min-width

What are data- attributes good for?

The HTML5 data attribute lets you assign custom data to an element. When we want to store more information/data about the element when no suitable HTML5 element or attribute exists

What is a self closing tag?

In HTML5 it is not strictly necessary to close certain HTML tags. The tags that aren't required to have specific closing tags are called "self closing" tags.

An example of a self closing tag is something like a line break (
) or the meta tag (<meta>). This means that the following are both acceptable:

```
<meta charset="UTF-8">
...
```

```
<meta charset="UTF-8" />
```

Does the following trigger http request at the time of page load?

```
<img src="mypic.jpg" style="visibility: hidden" alt="My
photo">

<div style="display: none;">
    <img src="mypic.jpg" alt="My photo">
</div>
```

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 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 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 overflow other than visible (except when that value has been propagated to the viewport) establish new block formatting contexts.

Knowing how to establish a block formatting context is important, because without doing so, the containing box will not contain floated children. This is similar to collapsing margins, but more insidious as you will find entire boxes collapsing in odd ways.

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.

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

- Empty div method - .
- 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: hidden might clip children if the children is taller than the parent and is not very ideal.

How would you approach fixing browser-specific styling issues?

- After identifying the issue and the offending browser, use a separate style sheet that only loads when that specific browser is being used. This technique requires server-side rendering though.
- Use libraries like Bootstrap that already handles these styling issues for you.
- Use autoprefixer to automatically add vendor prefixes to your code.
- Use Reset CSS or Normalize.css.
- If you're using Postcss (or a similar transpiling library), there may be plugins which allow you to opt in for using modern CSS syntax (and even W3C proposals) that will transform those sections of your code into corresponding safe code that will work in the targets you've used.

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

These techniques are related to accessibility (a11y).

- 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.
- Metadata. For example by using Schema.org, RDF, and JSON-LD.
- WAI-ARIA. A W3C technical specification that specifies how to increase the accessibility of web pages.

Even if WAI-ARIA is the ideal solution, I would go with the absolute positioning approach, as it has the least caveats, works for most elements and it's an easy technique.

Are you familiar with styling SVG?

Yes, there are several ways to color shapes (including specifying attributes on the object) using inline CSS, an embedded CSS section, or an external CSS file. Most SVG you'll find around the web use inline CSS, but there are advantages and disadvantages associated with each type.

Basic coloring can be done by setting two attributes on the node: fill and stroke. fill sets the color inside the object and stroke sets the color of the line drawn around the object. You can use the same CSS color naming schemes that you use in HTML, whether that's color names (that is red), RGB values (that is rgb(255,0,0)), Hex values, RGBA values, etc.

```
<rect
  x="10"
  y="10"
  width="100"
  height="100"
  stroke="blue"
  fill="purple"
  fill-opacity="0.5"
  stroke-opacity="0.8"
/>
```

The above fill="purple" is an example of a presentational attribute. Interestingly, and unlike inline styles like style="fill: purple" which also happens to be an attribute, presentational attributes can be overriden by CSS styles defined in a stylesheet. So, if you did something like svg { fill: blue; } it would override the purple fill we've defined.

What are some of the "gotchas" for writing efficient CSS?

Firstly, understand that browsers match selectors from rightmost (key selector) to left. Browsers filter out elements in the DOM according to the key selector and traverse up its parent elements to determine matches. The shorter the length of the selector chain, the faster the browser can determine if that element matches the selector. Hence avoid key selectors that are tag and universal selectors. They match a large number of elements and browsers will have to do more work in determining if the parents do match.

BEM (Block Element Modifier) methodology recommends that everything has a single class, and, where you need hierarchy, that gets baked into the name of the class as well, this naturally makes the selector efficient and easy to override.

Be aware of which CSS properties trigger reflow, repaint, and compositing. Avoid writing styles that change the layout (trigger reflow) where possible.

What are the advantages/disadvantages of using CSS preprocessors?

Advantages:

- CSS is made more maintainable.
- Easy to write nested selectors.

- Variables for consistent theming. Can share theme files across different projects.
- Mixins to generate repeated CSS.
- Sass features like loops, lists, and maps can make configuration easier and less verbose.
- Splitting your code into multiple files. CSS files can be split up too but doing so will require an HTTP request to download each CSS file.

Disadvantages:

- Requires tools for preprocessing. Re-compilation time can be slow.
- Not writing currently and potentially usable CSS. For example, by using something like postcss-loader with webpack, you can write potentially future-compatible CSS, allowing you to use things like CSS variables instead of Sass variables. Thus, you're learning new skills that could pay off if/when they become standardized.

Describe what you like and dislike about the CSS preprocessors you have used.

Likes:

- Mostly the advantages mentioned above.
- Less is written in JavaScript, which plays well with Node.

Dislikes:

- I use Sass via node-sass, which is a binding for LibSass written in C++. I have to frequently recompile it when switching between node versions.
- In Less, variable names are prefixed with @, which can be confused with native CSS keywords like @media, @import and @font-face rule.

Explain how a browser determines what elements match a CSS selector.

This part is related to the above about writing efficient CSS. Browsers match selectors from rightmost (key selector) to left. Browsers filter out elements in the DOM according to the key selector and traverse up its parent elements to determine matches. The shorter the length of the selector chain, the faster the browser can determine if that element matches the selector.

For example with this selector p span, browsers firstly find all the elements and traverse up its parent all the way up to the root to find the <p> element. For a particular , as soon as it finds a <p>, it knows that the matches and can stop its matching.

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

- Make every element in the document include the padding and border in the element's inner dimensions; making it easier to reason about the layout of elements on the page.
- 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 border width.
- Taking into account paddings and borders as part of our box model resonates better with how designers actually imagine content in grids.

List display property in CSS?

The display property specifies the display behavior (the type of rendering box) of an element.

Example:

```
p.ex1 {display: none;}
p.ex2 {display: inline;}
p.ex3 {display: block;}
p.ex4 {display: inline-block;}
```

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

translate() is a value of CSS transform. Changing transform or opacity does not trigger browser reflow or repaint but does trigger 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 occupies its original space (sort of like position: relative), unlike in changing the absolute positioning.

Example:

If we combine position:relative with one of the offset properties top, bottom, left or right the element will be moved from its original place in the layout whilst preserving the space in the document it once occupied. The element will be moved on to a new layer and its "layer order" or its stacking order can then be controlled with the z-index property.

```
.thing {
  position: relative;
  top: 100px;
  left: 50px;
}
```

In the above example the element will be moved 100px away from the top and 50px away from the left of its original position.

When using transform:translate(x,y) we get a very similar visual result to using relative position. The same result as above could be achieved with the following snippet:

```
.thing {
  transform: translate(50px, 100px);
}
```

In this case, we are translating the coordinates of the element by 50px along the x-axis and 100px along the y-axis. The end result is visually the same as the previous position example.

What is Accessibility (a11y) in a web application?

Accessibility refers to how software or hardware combinations are designed to make a system accessible to persons with disabilities, such as:

- Visual impairment

- Hearing loss
- Limited dexterity

For example, a website developed with accessibility in mind might have text-to-speech capabilities or output for special braille hardware geared toward individuals with visual impairments.

What is UI/UX?

1) UI or User Interface: is how a product or website is laid out and how you interact with it: Where the buttons are, how big the fonts are, and how menus are organized are all elements of UI.

2) UX or User Experience: is how you feel about using a product or a website. So, your love for the way the new Apple Watch looks or your excitement that there's finally a tablet-sized iPhone to watch those Corgi videos you're obsessed with are reflections of UX. So the new look of the Facebook news feed involves a change to UI, and the way you navigate that new page is the UX.

Which property is used to change the face of a font?

The font-family property is used to change the face of a font.

Which property is used to make a font italic or oblique?

The font-style property is used to make a font italic or oblique.

Which property is used to create a small-caps effect?

The font-variant property is used to create a small-caps effect.

Which property is used to increase or decrease how bold or light a font appears?
The font-weight property is used to increase or decrease how bold or light a font appears.

Which property is used to add or subtract space between the letters that make up a word?

The letter-spacing property is used to add or subtract space between the letters that make up a word.

Which property is used to add or subtract space between the words of a sentence?

The word-spacing property is used to add or subtract space between the words of a sentence.

Which property is used to indent the text of a paragraph?

The text-indent property is used to indent the text of a paragraph.

Which property is used to align the text of a document?

The text-align property is used to align the text of a document.

Which property is used to underline, overline, and strikethrough text?

The text-decoration property is used to underline, overline, and strikethrough text.

Which property is used to capitalize text or convert text to uppercase or lowercase letters?

The text-transform property is used to capitalize text or convert text to uppercase or lowercase letters.

Which property allows you to control the shape or appearance of the marker of a list?

The list-style-type allows you to control the shape or appearance of the marker.

How do I restore the default value of a property?

The keyword initial can be used to resets it to its default value, which is defined in the CSS specification of the given property.

What is specificity? How to calculate specificity?

A process of determining which css rule will be applied to an element. It actually determines which rules will take precedence. Inline style usually wins then ID then class value (or pseudo-class or attribute selector), universal selector (*) has no specificity. ID selectors have a higher specificity than attribute selectors.

Selector Types

The following list of selector types increases by specificity:

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

```
/*wins*/
a#a-02 { background-image : url(n.gif); }
a[id="a-02"] { background-image : url(n.png); }
```

Contextual selectors are more specific than a single element selector.The embedded style sheet is closer to the element to be styled. The last rule defined overrides any previous, conflicting rules.

```
p { color: red; background: yellow }
p { color: green } // wins
```

A class selector beats any number of element selectors.

```
.introduction {} //wins
html body div div h2 p {}
```

What do you know about CSS Transitions?

CSS Transitions allows to add an effect while changing from one style to another. You can set the which property you want to transition, duration, how you want to transit (linear, ease, ease-in, ease-out, cubic-bezier) and delay when transition will start.

CSS Transition Properties

Sl.No	Property	Description
01.	transition	A shorthand property for setting the four transition properties into a single property
02.	transition-delay	Specifies a delay (in seconds) for the transition effect

03.	transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete
04.	transition-property	Specifies the name of the CSS property the transition effect is for
05.	transition-timing-function	Specifies the speed curve of the transition effect

Example:

```
div {
  width: 100px;
  height: 100px;
  background: red;
  -webkit-transition: width 2s; /* Safari prior 6.1
*/
  transition: width 2s;
}
```

What does !important mean in CSS?

It overrides the cascade and gives the style rule the highest precedence.

```
p {
    color: red !important;
}
#thing {
    color: green;
}
<p id="thing">Will be RED.</p>
```

What is CSS opacity?

The opacity CSS property sets the opacity of an element. Opacity is the degree to which content behind an element is hidden, and is the opposite of transparency.

```
div { background-color: yellow; }
.light {
  opacity: 0.2; /* Barely see the text over the background */
}
.medium {
  opacity: 0.5; /* See the text more clearly over the background */
}
.heavy {
  opacity: 0.9; /* See the text very clearly over the background */
}
<div class="light">You can barely see this.</div>
<div class="medium">This is easier to see.</div>
<div class="heavy">This is very easy to see.</div>
```

What is contextual selector?

Contextual selector addresses specific occurrence of an element. It is a string of individual selectors separated by white space (search pattern), where only the last element in the pattern is addressed providing it matches the specified contex.

It also check the context of the class in the html tree, assigning the style to the element through a specific route, taking into account the order of depth in the tree.

How is the concept of inheritance applied in CSS?

Inheritance is a concept in which the child class will inherit the properties of its parent class. It is used in CSS to define the hierarchy from the top level to the bottom level. Inherited properties can be overridden by the children class if the child uses the same name.

How do you handle browser differences in your user base?

The @supports query in CSS can be very useful to scan if the user’s current browser has a certain feature. The @supports CSS at-rule lets you specify declarations that depend on a browser's support for one or more specific CSS features. This is called a feature query. The rule may be placed at the top level of your code or nested inside any other conditional group at-rule.

```
@supports (display: grid) {
  div {
    display: grid;
  }
}

@supports not (display: grid) {
  div {
    float: right;
  }
}
```

What is Cascade?

Cascade is a method of defining the weight (importance) of individual styling rules thus allowing conflicting rules to be sorted out should such rules apply to the same selector.

```
P {color: white ! important} /* increased weight */
P (color: black} /* normal weight */

How Do I Have A Fixed (non-scrolling) Background Image?
```

In CSS, we can use the background-attachment property. The background attachment can be included in the shorthand background property, as in this example:

```
body {
  background: white url(example.gif) fixed ;
  color: black ;
}
```

Describe the following common CSS units of length: cm, em, in, mm, pc, pt, and px.

There are many ways to express units of length within CSS, but these are just some of the more common ones.

cm: centimeters
em: a relative unit of measurement based on the size of a font
in: inches

mm: millimeters
pc: pica, a unit of length equivalent to 12 points, or 1/6 of an inch
pt: 1/72 of an inch
px: a device-specific relative measurement equivalent to a certain number of pixels on a display

What are CSS vendor prefixes?

Vendor prefixes are extensions to CSS standards that can be added to these features to prevent incompatibilities from arising when the standard is extended. CSS vendor prefixes for some common platforms are listed below.

-webkit-: Android, Chrome, iOS, and Safari
-moz-: Mozilla Firefox
-ms-: Internet Explorer
-o-: Opera

CSS3 Colors

The color keyword list has been extended in the CSS3 color module to include 147 additional keyword colors (that are generally well supported), CSS3 also provides us with a number of other options: HSL, HSLA, RGBA and Opacity.

```
div.halfopaque {
  background-color: rgb(0, 0, 0);
  opacity: 0.5;
  color: #000000;
}
div.halfalpha {
  background-color: rgba(0, 0, 0, 0.5);
  color: #000000;
}
```

Rounded Corners: border-radius

```
border-radius: 25px;
```

Drop Shadows

```
box-shadow: 2px 5px 0 0 rgba(72,72,72,1);
```

Text Shadow

```
text-shadow: topOffset leftOffset blurRadius color;
```

Linear Gradients

```
Syntax: background: linear-gradient(direction, color-stop1,
color-stop2, ...);
/* Example */
#grad {
  background: linear-gradient(to right, red , yellow);
}
```

Radial Gradients

```
Syntax : background: radial-gradient(shape size at position,
start-color, ..., last-color);
/* Example */
#grad {
  background: radial-gradient(red, yellow, green);
}//Default
#grad {
  background: radial-gradient(circle, red, yellow, green);
}//Circle
```

What are the CSS positioning?

Keyword	Value	Description
position	static	The default mode, block element is positioned in the flow. Top, left etc. are ignored.
position	relative	The block element is positioned relative to its position in the flow.
position	absolute	Block element is positioned relative to its container.

position	fixed	Block element is positioned relative to the window and won't scroll.
top	Number [px, cm, in…]	Positions the block down from the reference point at the specified distance.
bottom	Number [px, cm, in…]	Positions the block up from the reference point at the specified distance.
left	Number [px, cm, in…]	Positions the block right from the reference point at the specified distance.
right	Number [px, cm, in…]	Positions the block left from the reference point at the specified distance.