1. how to add comments on css?

Ans: In CSS, comments starts with /* and ends with */.

2. Why do we use pseudo-class?

Ans: A pseudo-class is used to define a special state of an element. For ex. Change style of element when mouse over that element, etc.

3. How is specificity applied?

Ans: Every CSS selector has its place in the specificity hierarchy.

There are four categories which define the specificity level of a selector:

1. Inline styles - Example: <h1 style="color: pink;">
2. IDs - Example: #navbar
3. Classes, pseudo-classes, attribute selectors - Example: .test, :hover, [href]
4. Elements and pseudo-elements - Example: h1, :before

If there are two or more CSS rules that point to the same element, the selector with the highest specificity value will "win", and its style declaration will be applied to that HTML element.

4. What method allows an element to be moved from its current position?

Ans: The translate() method moves an element from its current position

5. what properties does flex model have?

Ans:

- flex-grow
- flex-shrink
- flex-basis
- flex-direction
- flex-wrap

6. What is the difference between flex and grids?

Ans: The basic difference between CSS Grid Layout and CSS Flexbox Layout is that flexbox was designed for layout in one dimension - either a row or a column. Grid was designed for two-dimensional layout - rows, and columns at the same time.

1. Dimensionality and Flexibility:

Flexbox offers greater control over alignment and space distribution between items. Being one-dimensional, Flexbox only deals with either columns or rows.

Grid has two-dimension layout capabilities which allow flexible widths as a unit of length. This compensates for the limitations in Flex.

2. Alignment:

Flex Direction allows developers to align elements vertically or horizontally, which is used when developers create and reverse rows or columns.

CSS Grid deploys fractional measure units for grid fluidity and auto-keyword functionality to automatically adjust columns or rows.

3. Item Management

Flex Container is the parent element while Flex Item represents the children. The Flex Container can ensure balanced representation by adjusting item dimensions. This allows developers to design for fluctuating screen sizes.

Grid supports both implicit and explicit content placement. Its inbuilt automation allows it to automatically extend line items and copy values into the new creation from the preceding item.

7. Give an example where we have to use grids and where you have to use flexbox?

Ans: Flexbox works from the content out. An ideal use case for flexbox is when you have a set of items and want to space them out evenly in a container. You let the size of the content decide how much individual space each item takes up. If the items wrap onto a new line, they will work out their spacing based on their size and the available space on that line.

Grid works from the layout in. When you use CSS Grid Layout you create a layout and then you place items into it, or you allow the auto-placement rules to place the items into the grid cells according to that strict grid. It is possible to create tracks that respond to the size of the content, however, they will also change the entire track.

If you are using flexbox and find yourself disabling some of the flexibility, you probably need to use CSS Grid Layout. An example would be if you are setting a percentage width on a flex item to make it line up with other items in a row above. In that case, a grid is likely to be a better choice.

8. What are combinators? give examples of how you can use them.

Ans: A combinator is something that explains the relationship between the selectors.

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS:

1. descendant selector (space)
2. child selector (>)
3. adjacent sibling selector (+)
4. general sibling selector (~)

Descendant Selector

The descendant selector matches all elements that are descendants of a specified element.

The following example selects all <p> elements inside <div> elements:

Example:

```
div p {
  background-color: yellow;
}
```

Child Selector (>)

The child selector selects all elements that are the children of a specified element.

The following example selects all <p> elements that are children of a <div> element:

Example

```
div > p {
  background-color: yellow;
}
```

Adjacent Sibling Selector (+)

The adjacent sibling selector is used to select an element that is directly after another specific element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

The following example selects the first <p> element that are placed immediately after <div> elements:

Example

```
div + p {
  background-color: yellow;
}
```

General Sibling Selector (~)

The general sibling selector selects all elements that are next siblings of a specified element.

The following example selects all <p> elements that are next siblings of <div> elements:

Example

```
div ~ p {
  background-color: yellow;
}
```

9. What does object-fit do?

Ans: The CSS object-fit property is used to specify how an <img> or <video> should be resized to fit its container.

This property tells the content to fill the container in a variety of ways; such as "preserve that aspect ratio" or "stretch up and take up as much space as possible".

The object-fit property can take one of the following values:

1. fill - This is default. The image is resized to fill the given dimension. If necessary, the image will be stretched or squished to fit
2. contain - The image keeps its aspect ratio, but is resized to fit within the given dimension
3. cover - The image keeps its aspect ratio and fills the given dimension. The image will be clipped to fit
4. none - The image is not resized
5. scale-down - the image is scaled down to the smallest version of none or contain

10. What does rotate do?

Ans: The rotate() CSS function defines a transformation that rotates an element around a fixed point on the 2D plane, without deforming it. Its result is a <transform-function> data type.

The fixed point that the element rotates around — mentioned above — is also known as the transform origin.

The amount of rotation created by rotate() is specified by an <angle>. If positive, the movement will be clockwise; if negative, it will be counter-clockwise. A rotation by 180° is called point reflection.

11. What rule can be used to define animations?

Ans: An animation lets an element gradually change from one style to another.

We can change as many CSS properties you want, as many times as you want.

To use CSS animation, we must first specify some keyframes for the animation.

Keyframes hold what styles the element will have at certain times.

The @keyframes Rule

When we specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, we must bind the animation to an element.

The sub-properties of the animation property are:

animation-name

Specifies the name of the @keyframes at-rule describing the animation's keyframes.

animation-duration

Configures the length of time that an animation should take to complete one cycle.

animation-timing-function

Configures the timing of the animation; that is, how the animation transitions through keyframes, by establishing acceleration curves.

animation-delay

Configures the delay between the time the element is loaded and the beginning of the animation sequence.

animation-iteration-count

Configures the number of times the animation should repeat; you can specify infinite to repeat the animation indefinitely.

animation-direction

Configures whether or not the animation should alternate direction on each run through the sequence or reset to the start point and repeat itself.

animation-fill-mode

Configures what values are applied by the animation before and after it is executing.

animation-play-state

Lets you pause and resume the animation sequence.

12. When working with attribute selectors, how can you select elements which contain a particular attribute value?

Ans: The [attribute*="value"] selector is used to select elements whose attribute value contains a specified value.

The following example selects all elements with a class attribute value that contains "te":

Example

```
[class*="te"] {
  background: yellow;
}
```

13. What does @media do?

Ans: The @media rule is used in media queries to apply different styles for different media types/devices.

Media queries can be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

Using media queries are a popular technique for delivering a tailored style sheet (responsive web design) to desktops, laptops, tablets, and mobile phones.

We can also use media queries to specify that certain styles are only for printed documents or for screen readers (mediatype: print, screen, or speech).

In addition to media types, there are also media features. Media features provide more specific details to media queries, by allowing to test for a specific feature of the user agent or display device. For example, we can apply styles to only those screens that are greater, or smaller, than a certain width.

14. What can be used to override properties of an element

Ans: To override the CSS properties of a class using another class, we can use the !important directive. In CSS, !important means "this is important", and the property:value pair that has this directive is always applied even if the other element has higher specificity.

15. What is the ranking of selectors with respect to specificity

Ans: The following list of selector types ranked by specificity:

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

Universal selector (*), combinators (+, >, ~, " ", ||) and negation pseudo-class (:not()) have no effect on specificity.

16. how can we apply same styles to multiple selectors?

Ans: When you group CSS selectors, you apply the same styles to several different elements without repeating the styles in your stylesheet. Instead of having two, three, or more CSS rules that do the same thing (set the color of something to red, for example), you use a single CSS rule that accomplishes the same thing. The secret to this efficiency-boosting tactic is the comma.

You can place any valid selector in a group, and all elements in the document that match all the grouped elements will have the same style based on that style property.

17. What are the differences between relative and absolute in CSS?

Ans When you set the position relative to an element, without adding any other positioning attributes (top, bottom, right, left) nothing will happen. When you add an additional position, such as left: 20px the element will move 20px to the right from its normal position. Here, you can see that this element is relative to itself. When the element moves, no other element on the layout will be affected.

Absolute Position: This type of positioning allows you to place your element precisely where you want it.

The positioning is done relative to the first relatively (or absolutely) positioned parent element. In the case when there is no positioned parent element, it will be positioned related directly to the HTML element (the page itself).