**CSS**

- CSS stands for Cascading Style Sheets
- CSS is the language we use to style a Web page.
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

1. What is the Box model in CSS? Which CSS properties are a part of it?

A rectangle box is wrapped around every HTML element. The box model is used to determine the height and width of the rectangular box. The CSS Box consists of Width and height (or in the absence of that, default values and the content inside), padding, borders, margin.
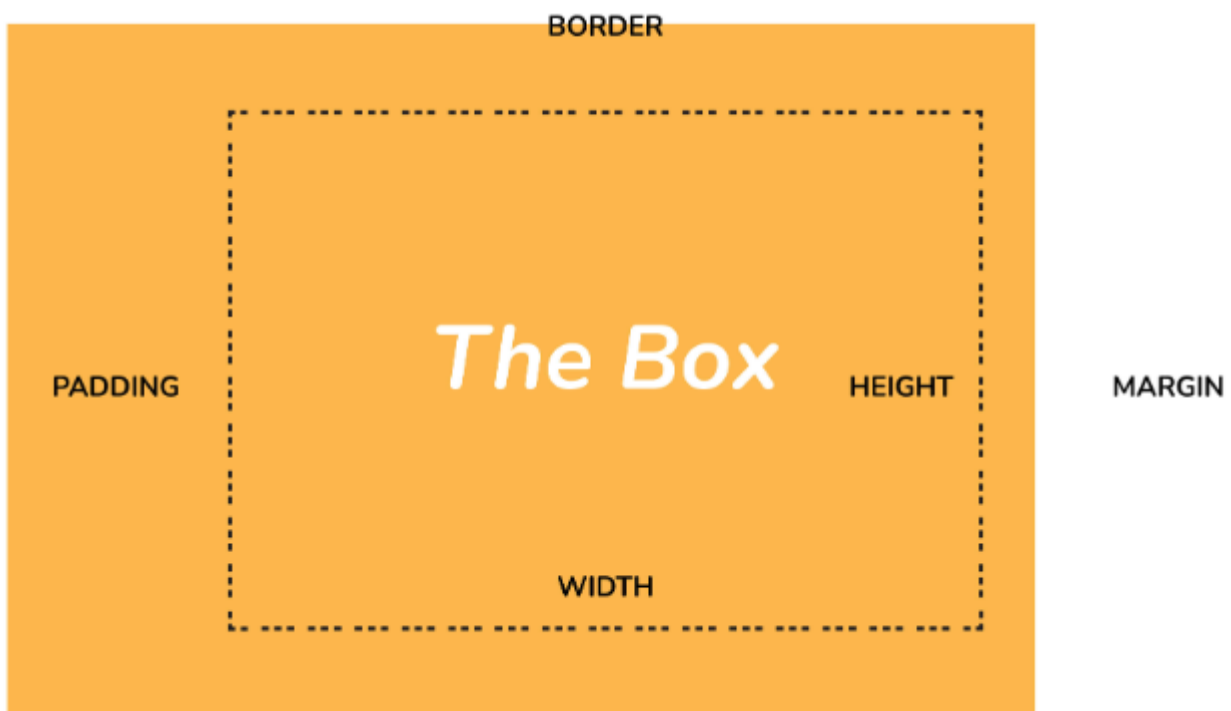
1. **Content:** Actual Content of the box where the text or image is placed.
2. **Padding:** Area surrounding the content (Space between the border and content).
3. **Border:** Area surrounding the padding.
4. **Margin:** Area surrounding the border.

2. What are the advantages of using CSS?

The main advantages of CSS are given below:

1. **Separation of content from presentation -** CSS provides a way to present the same content in multiple presentation formats in mobile or desktop or laptop.
2. **Easy to maintain -** CSS, built effectively can be used to change the look and feel complete by making small changes. To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.

3. **Bandwidth -** Used effectively, the style sheets will be stored in the browser cache and they can be used on multiple pages, without having to download again.

3. What are the limitations of CSS?

Disadvantages of CSS are given below:

1. **Browser Compatibility:** Some style selectors are supported and some are not. We have to determine which style is supported or not using the @support selector).
2. **Cross Browser issue:** Some selectors behave differently in a different browser).
3. **There is no parent selector:** Currently, Using CSS, you can't select a parent tag.

4. How to include CSS in the webpage?

There are different ways to include a CSS in a webpage,

1 - External Style Sheet: An external file linked to your HTML document: Using link tag, we can link the style sheet to the HTML page.

```
<link rel="stylesheet" type="text/css" href="styles.css" />
```

2 - Embed CSS with a style tag: A set of CSS styles included within your HTML page.

```
<style type="text/css">

    /*Add style rules here*/

</style>
```

Add your CSS rules between the opening and closing style tags and write your CSS exactly the same way as you do in stand-alone stylesheet files.

3 - Add inline styles to HTML elements(CSS rules applied directly within an HTML tag.): Style can be added directly to the HTML element using a style tag.

```
<h2 style="color:red;background:black">Inline Style</h2>
```

4 - Import a stylesheet file (An external file imported into another CSS file): Another way to add CSS is by using the @import rule. This is to add a new CSS file within CSS itself.

```
@import "path/to/style.css";
```

5. What are the different types of Selectors in CSS?

A CSS selector is the part of a CSS ruleset that actually selects the content you want to style. Different types of selectors are listed below.

Universal Selector: The universal selector works like a wildcard character, selecting all elements on a page. In the given example, the provided styles will get applied to all the elements on the page.

```css
* {
  color: "green";
  font-size: 20px;
  line-height: 25px;
}
```

Element Type Selector: This selector matches one or more HTML elements of the same name. In the given example, the provided styles will get applied to all the ul elements on the page.

```css
ul {
  line-style: none;
  border: solid 1px #ccc;
}
```

ID Selector: This selector matches any HTML element that has an ID attribute with the same value as that of the selector. In the given example, the provided styles will get applied to all the elements having ID as a container on the page.

```css
#container {
  width: 960px;
  margin: 0 auto;
}
<div id="container"></div>
```

Class Selector: The class selector also matches all elements on the page that have their class attribute set to the same value as the class. In the given example, the provided styles will get applied to all the elements having ID as the box on the page.

```css
.box {
  padding: 10px;
  margin: 10px;
  width: 240px;
}
<div id="container"></div>
```

Descendant Combinator: The descendant selector or, more accurately, the descendant combinator lets you combine two or more selectors so you can be more specific in your selection method.

```css
#container .box {
  float: left;
  padding-bottom: 15px;
}
<div id="container">
    <div class="box"></div>
```

```
        <div class="box-2"></div>
    </div>

    <div class="box"></div>
```

This declaration block will apply to all elements that have a class of box that is inside an element with an ID of the container. It's worth noting that the .box element doesn't have to be an immediate child: there could be another element wrapping .box, and the styles would still apply.

Child Combinator: A selector that uses the child combinator is similar to a selector that uses a descendant combinator, except it only targets immediate child elements.

```
#container> .box {
    float: left;
    padding-bottom: 15px;
}
<div id="container">
    <div class="box"></div>

    <div>
        <div class="box"></div>
    </div>
</div>
```

The selector will match all elements that have a class of box and that are immediate children of the #container element. That means, unlike the descendant combinator, there can't be another element wrapping .box it has to be a direct child element.

General Sibling Combinator: A selector that uses a general sibling combinator to match elements based on sibling relationships. The selected elements are beside each other in the HTML.

```
h2 ~ p {
    margin-bottom: 20px;
}
<h2>Title</h2>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<div class="box">
    <p>Paragraph example.</p>
</div>
```

In this example, all paragraph elements (<p>) will be styled with the specified rules, but only if they are siblings of <h2> elements. There could be other elements in between the <h2> and <p>, and the styles would still apply.

Adjacent Sibling Combinator: A selector that uses the adjacent sibling combinator uses the plus symbol (+), and is almost the same as the general sibling selector. The difference is that the targeted element must be an immediate sibling, not just a general sibling.

```
p + p {
    text-indent: 1.Sem;
    margin-bottom: 0;
}
<h2>Title</h2>
    <p>Paragraph example.</p>
    <p>Paragraph example.</p>
    <p>Paragraph example.</p>

    <div class="box">
        <p>Paragraph example.</p>
        <p>Paragraph example.</p>
    </div>
```

The above example will apply the specified styles only to paragraph elements that immediately follow other paragraph elements. This means the first paragraph element on a page would not receive these styles. Also, if another element appeared between two paragraphs, the second paragraph of the two wouldn't have the styles applied.

Attribute Selector: The attribute selector targets elements based on the presence and/or value of HTML attributes, and is declared using square brackets.

```
input [type="text"] {
    background-color: #444;
    width: 200px;
}
<input type="text">
```

6. What is a CSS Preprocessor?

A CSS Preprocessor is a tool used to extend the basic functionality of default vanilla CSS through its own scripting language. It helps us to use complex logical syntax like – variables, functions, mixins, code nesting, and inheritance to name a few, supercharging your vanilla CSS. Types: Sass, Less and Stylus.

7. What is VH/VW (viewport height/ viewport width) in CSS?

It's a CSS unit used to measure the height and width in percentage with respect to the viewport. It is used mainly in responsive design techniques. The measure VH is equal to 1/100 of the height of the viewport. If the height of the browser is 1000px, 1vh is equal to 10px. Similarly, if the width is 1000px, then 1 vw is equal to 10px.

8. Difference between reset vs normalize CSS? How do they differ?

Reset CSS: 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.

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

9. What is the difference between inline, inline-block, and block?

Block Element: The block elements always start on a new line. They will also take space for an entire row or width. List of block elements are <div>, <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.

Inline Block Elements: Inline-block elements are similar to inline elements, except they can have padding and margins and set height and width values.

10. Is it important to test the webpage in different browsers?

It's most important to test a website in different browsers when you're first designing it, or when making major changes. However, it's also important to repeat these tests periodically, since browsers go through a lot of updates and changes.

11. What are Pseudo elements and Pseudo classes?

Pseudo-elements allows us to create items that do not normally exist in the document tree, for example ::after.

1. ::before
2. ::after
3. ::first-letter
4. ::first-line
5. ::selection

In the below example, the color will appear only on the first line of the paragraph.

```css
p: :first-line {
   color: #ffOOOO;
   font-variant: small-caps;
}
```

Pseudo-classes select regular elements but under certain conditions like when the user is hovering over the link.

1. :link
2. :visited
3. :hover
4. :active
5. :focus

Example of the pseudo-class, In the below example, the color applies to the anchor tag when it's hovered.

```css
/* mouse over link */
```

```
a:hover {
   color: #FFOOFF;
}
```

12. How do you specify units in the CSS?. What are the different ways to do it?

There are different ways to specify units in CSS like px, em, pt, percentage (%). px(Pixel) gives fine-grained control and maintains alignment because 1 px or multiple of 1 px is guaranteed to look sharp. px is not cascade. em maintains relative size. you can have responsive fonts. Em, will cascade 1em is equal to the current font-size of the element or the browser default. If u sent font-size to 16px then 1em = 16px. The common practice is to set default body font-size to 62.5% (equal to 10px).

pt(point) are traditionally used in print. 1pt = 1/72 inch and it is a fixed-size unit.

%(percentage) sets font-size relative to the font size of the body. Hence, you have to set the font-size of the body to a reasonable size.

13. Does margin-top or margin-bottom have an effect on inline elements?

No, it doesn't affect the inline elements. Inline elements flow with the contents of the page.

14. What property is used for changing the font face?

We can use the font-family property for achieving this. The font-family property is used for specifying what font needs to be applied on the targetted DOM element. It can hold several font names as part of "fallback" mechanism in case the browser does not support the fonts. For example, we can use:

```
p {
   font-family: "Times New Roman", Times, serif;
}
```

In the above piece of code, we are applying font-family property to the paragraph element.

1. It tells the browser to look for "Times New Roman" font and apply it.
2. If the "Times New Roman" font is not installed or supported, then it asks the browser to use Times font.
3. If both "Times New Roman" and Times are not supported, then it asks the browser to use any supported generic font belonging to serif.

If you do not want the font-face of the paragraph element to be Times New Roman/Times/serif font, and you want to use the Arial/Helvetica/sans-serif font, then we can just update the CSS property of paragraph element as:

```
p {
   font-family: Arial, Helvetica, sans-serif;
}
```

15. What are the differences between adaptive design and responsive design?

Adaptive Design	Responsive Design
Adaptive design focuses on developing websites based on multiple fixed layout sizes.	Responsive design focuses on showing content on the basis of available browser space.
When a website developed using adaptive design is opened on the desktop browser, first the available space is detected and then the layout with most appropriate sizes are picked and used for the display of contents. Resizing of browser window has no affect on the design.	When a website developed using responsive design is opened on a desktop browser and when we try to resize the browser window, the content of the website is dynamically and optimally rearranged to accomodate the window.
Usually, adaptive designs use six standard screen widths - 320 px, 480 px, 760 px, 960 px, 1200 px, 1600 px. These sizes are detected and appropriate layouts are loaded.	This design makes use of CSS media queries for changing styles depending on the target devices properties for adapting to different screens.
It takes a lot of time and effort to first examine the options and realities of the end users and then design best possible adaptive solutions them.	Generally, Responsive design takes much less work to build and design fluid websites that can accomodate content from screen depending on the screen size.
Gives a lot of control over the design to develop sites for specific screens.	No much control over the design is offered here.

16. How are the CSS selectors matched against the elements by the browser?

The order of matching selectors goes from **right to left** of the selector expression. The elements in the DOM are filtered by browsers based on the key selectors and are then traversed up to the parent elements for determining the matches. The speed of determining the elements depends on the length of the chain of selectors. Consider an example:

```
p span{
    color: black;
}
```

Here, the browser first finds all span elements in the DOM and then it traverses to each of its parent elements to check if they are the paragraph p elements.

Once the browser finds **all** matching span tags having paragraph elements as parent and applies the color of black to the content, the matching process is stopped.

**17. How is border-box different from content-box?**

content-box is the default value box-sizing property. The height and the width properties consist only of the content by excluding the border and padding. Consider an example as shown:

```
div{
```

```
   width:300px;
   height:200px;
   padding:15px;
   border: 5px solid grey;
   margin:30px;
   -moz-box-sizing:content-box;
   -webkit-box-sizing:content-box;
   box-sizing:content-box;
}
```

Here, the box-sizing for the div element is given as content-box. That means, the height and width considered for the div content exclude the padding and border. We will get full height and width parameters specified for the content as shown in the below image.
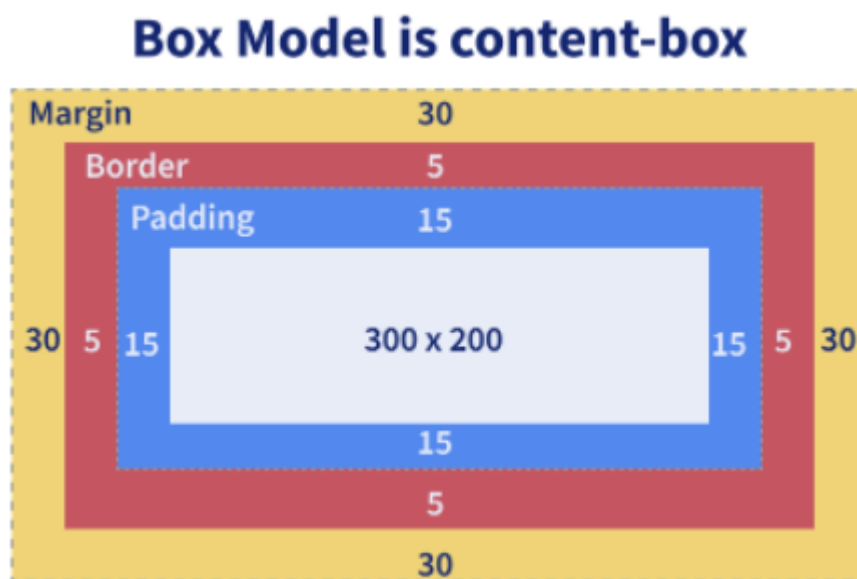
The border-box property includes the content, padding and border in the height and width properties. Consider an example as shown:

```
div{
   width:300px;
   height:200px;
   padding:15px;
   border: 5px solid grey;
   margin:30px;
   -moz-box-sizing:border-box;
   -webkit-box-sizing:border-box;
   box-sizing:border-box;
}
```

Here, the box-sizing for the div element is given as border-box. That means the height and width considered for the div content will also include the padding and border. This means that the actual height of the div content will be:

actual height = height -

       padding on top and bottom -
       border on top and bottom
   = 200 - (15*2) - (5*2)

   = 160 px

and the actual width of the div content would be:
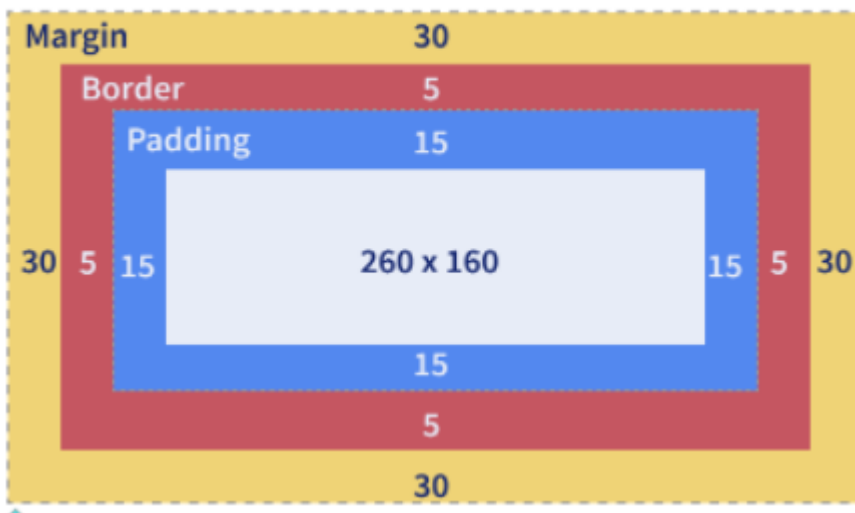
actual width = width -
       padding on right and left -

      border on right and left

   = 300 - (15*2) - (5*2)

   = 260 px

This is represented in the image below:

Box Model is border-box

18. How is opacity specified in CSS3?

Opacity refers to the degree to which the content is transparent or opaque. We can use the property named opacity which takes the values ranging from 0 to 1. 0 specifies that the element is completely transparent where 1 means that the element is completely opaque. We can use the opacity property as follows:

```css
div {
  opacity: 0.6;
}
```

In the above example, an opacity of 60% is applied to the div section. The opacity property is not supported by the internet explorer browser. To make it work there, we need to use filter property as polyfill as shown in the example below.

```css
div {
   opacity: 0.6;
   filter: alpha(opacity=60);
}
```
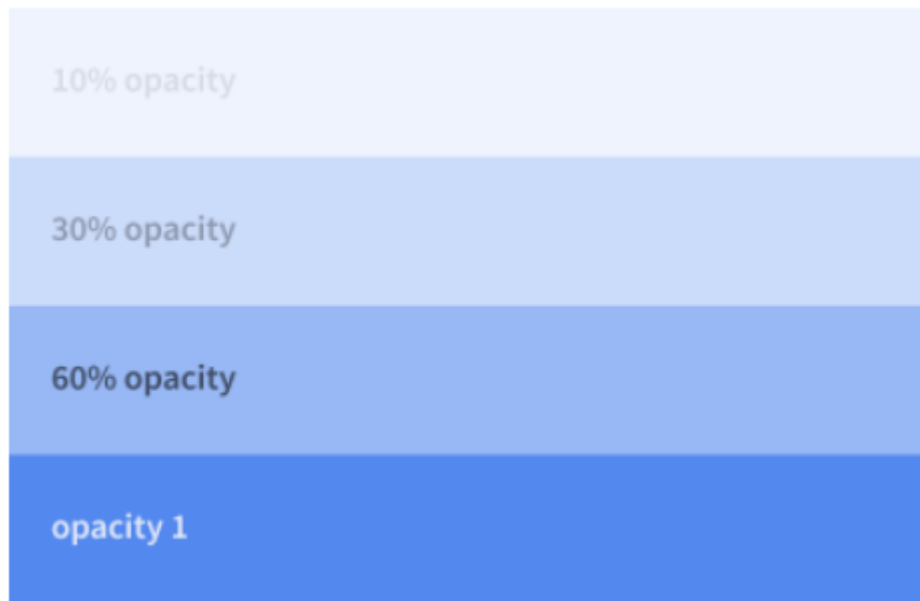

19. Why should we use float property in CSS?

The float property is used for positioning the HTML elements horizontally either towards the left or right of the container. For instance,

```css
float-demo {
   float: right;
}
```

Here, the element to which the class is applied ensures that the element is positioned on the right of the container. If you specify the value of float as left, then the element will be placed on the left side of the container.

20. What is a z-index, how does it function?

z-index is used for specifying the vertical stacking of the overlapping elements that occur at the time of its positioning. It specifies the vertical stack order of the elements positioned that helps to define how the display of elements should happen in cases of overlapping.

The default value of this property is 0 and can be either positive or negative. Apart from 0, the values of the z-index can be:

1.   Auto: The stack order will be set equal to the parent.

2. Number: The number can be positive or negative. It defines the stack order.
3. Initial: The default value of 0 is set to the property.
4. Inherit: The properties are inherited from the parent.

The elements having a lesser value of z-index is stacked lower than the ones with a higher z-index.
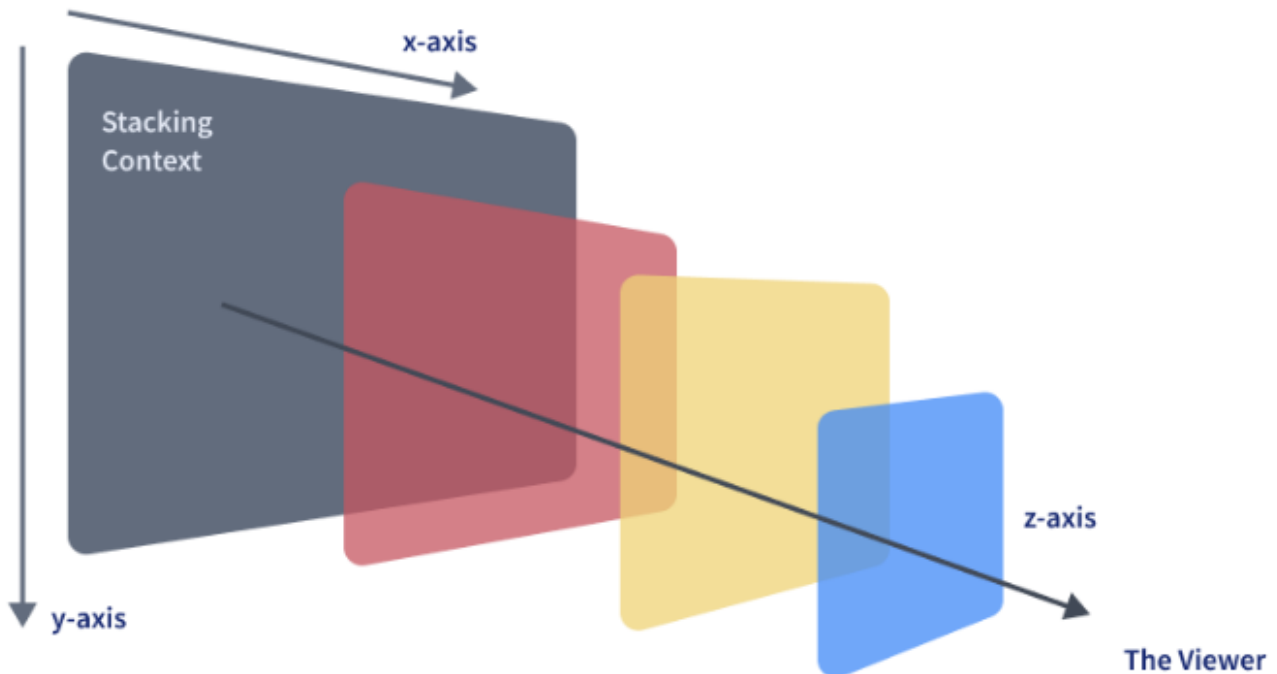
The z-index property determines the order of the element on the z-axis of the stacking contect.

From the above figure, we can see that as the value of the z-index increases along the z-axis, the order of stacking would be towards the top of other elements along the vertical axis.

21. What are the properties of flexbox?

Flexbox stands for flexible box and it was introduced around 2017 in CSS with the purpose of providing an efficient way to handle layouts, align elements within them and distribute spaces amongst the items in dynamic/responsive conditions. It provides an enhanced ability to alter the dimensions of the items and make use of the available space in the container efficiently. In order to achieve this, CSS3 provides some properties.

The properties of flexbox are as follows:

1. **flex-direction**: This property helps in defining the direction the container should stack the items targetted for flex. The values of this property can be
2. **row**: Stacks items horizontally from left to right in the flex container.
3. **column**: Stacks items vertically from top to bottom in the flex container.
4. **row-reverse:** Stacks items horizontally from right to left in the flex container.
5. **column-reverse:** Stacks items vertically from bottom to top in the flex container.

6. **flex-wrap**: This property specifies of the flex items should be wrapped or not. Possible values are:
7. **wrap**: The flex items would be wrapped if needed.
8. **nowrap**: This is the default value that says the items won't be wrapped.
9. **wrap-reverse**: This specifies that the items will be wrapped if needed but in reverse order.
10. **flex-flow**: This property is used for setting both flex-direction and flex-wrap properties in one statement.
11. **justify-content**: Used for aligning the flex items. Possible values are:
12. **center**: It means that all the flex items are present at the center of the container.
13. **flex-start**: This value states that the items are aligned at the start of the container. This is the default value.
14. **flex-end:** This value ensures the items are aligned at the end of the container.
15. **space-around**: This value displays the items having space between, before, around the items.
16. **space-between**: This value displays items with spaces between the lines.
17. **align-items**: This is used for aligning flex items.
18. **align-content**: This is used for aligning the flex lines.

22. What is cascading in CSS?

"Cascading" refers to the process of going through the style declarations and defining weight or importance to the styling rules that help the browser to select what rules have to be applied in times of conflict. The conflict here refers to multiple rules that are applicable to a particular HTML element. In such cases, we need to let the browser know what style needs to be applied to the element. This is done by cascading down the list of style declarations elements.

For example, if we have the below style:

```
p{
   color:white;
}
```

and we also have the following declaration below it or in another stylesheet that has been linked to the page:

```
p{
   color: black;
}
```

We have a conflict in color property here for the paragraph elements. Here, the browser just cascades down to identify what is the most recent and most specific style and applies that. Since we have the color:black; as the most specific declaration, the color black is applied to the paragraph elements. Now if you want to ensure color white is applied to the paragraph, we can define weight to that style by adding !important as shown below:

```
p{
   color:white !important;
}
```

!important ensures that the property has the maximum weight in presence of other conflicting properties.

23. Explain CSS position property?

1. **Absolute:** To place an element exactly where you want to place it. absolute position is actually set relative to the element's parent. if no parent is available then the relative place to the page itself (it will default all the way back up to the element).
2. **Relative:** "Relative to itself". Setting position: relative; on an element and no other positioning attributes, it will no effect on its positioning. It allows the use of z-index on the element and it limits the scope of absolutely positioned child elements. Any child element will be absolutely positioned within that block.
3. **Fixed:** The element is positioned relative to the viewport or the browser window itself. viewport doesn't change if you scroll and hence the fixed element will stay right in the same position.
4. **Static:** Static default for every single page element. The only reason you would ever set an element to position: static is to forcefully remove some positioning that got applied to an element outside of your control.
5. **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.
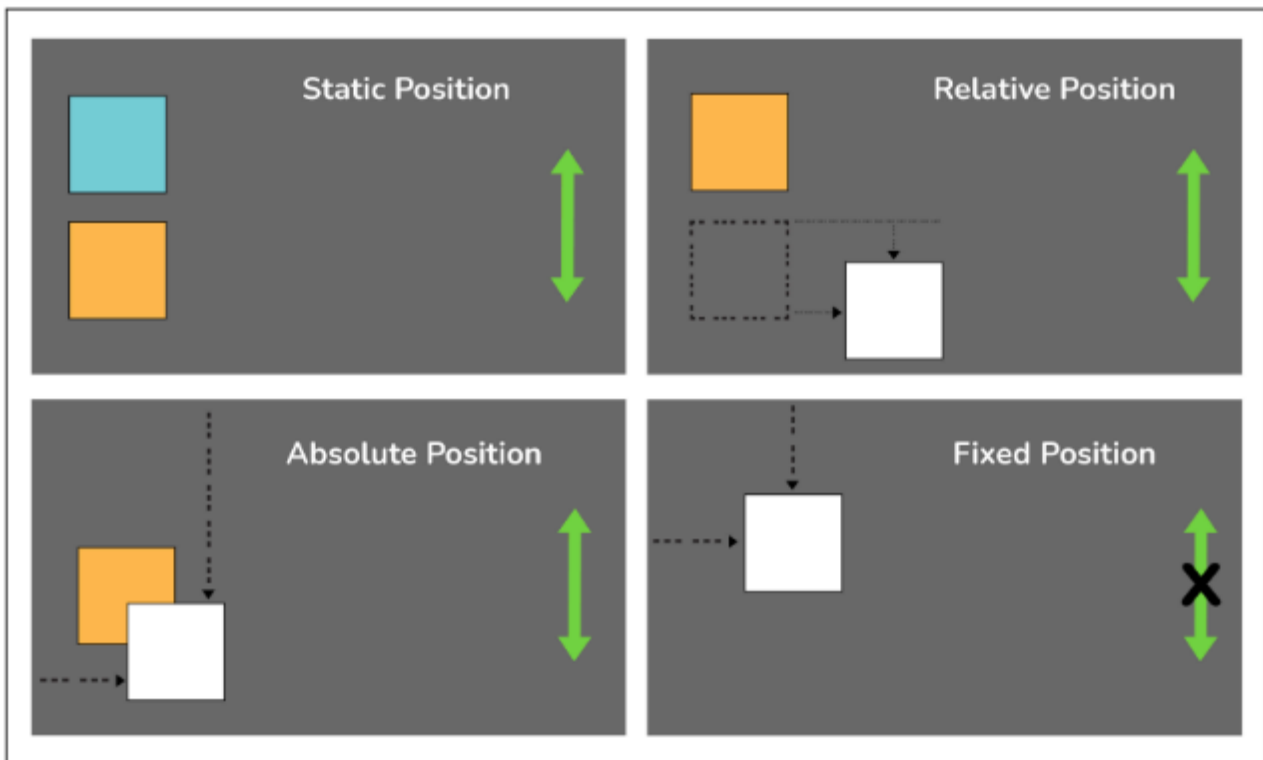
24. When does DOM reflow occur?

Reflow is the name of the web browser process for re-calculating the positions and geometries of elements in the document, for the purpose of re-rendering part or all of the document.

Reflow occurs when:

1. Insert, remove or update an element in the DOM.
2. Modify content on the page, e.g. the text in an input box.
3. Move a DOM element.

4. Animate a DOM element.
5. Take measurements of an element such as offsetHeight or getComputedStyle.
6. Change a CSS style.

25. Different Box Sizing Property?

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

1. **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.
2. **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.
3. **Border-box:** Width and height values apply to the content, padding, and border.

26. How to center align a div inside another div?

1. Centering with Table:

HTML:

```
<div class="cn"><div class="inner">your content</div></div>
```

CSS:

```
.cn {
   display: table-cell;
   width: 500px;
   height: 500px;
   vertical-align: middle;
   text-align: center;
}

.inner {
   display: inline-block;
   width: 200px; height: 200px;
}
```

2. Centering with Transform

HTML:

```
<div class="cn"><div class="inner">your content</div></div>
```

CSS:

```
.cn {
   position: relative;
   width: 500px;
   height: 500px;
}
```

```
.inner {
    position: absolute;
    top: 50%; left: 50%;
    transform: translate(-50%,-50%);
    width: 200px;
    height: 200px;
}
```

3. Centering with Flexbox

HTML:

```
<div class="cn"><div class="inner">your content</div></div>
```

CSS:

```
.cn {
    display: flex;
    justify-content: center;
    align-items: center;
}
```

4. Centering with Grid

HTML:

```
<div class="wrap_grid">
    <div id="container">vertical aligned text<br />some more text here
    </div>
</div>
```

CSS:

```
.wrap-grid {
    display: grid;
    place-content: center;
}
```

27. Can you name the four types of @media properties?

The four types of @media properties are:

1. All → It's the default property. Used for all media-type devices.
2. Screen → Used for computer screen, mobile screen.
3. Print → Used for printers.
4. Speech → Used for screen readers.

28. What is the grid system?

CSS Grid Layout is the most powerful layout system available in CSS. It is said to be a 2-dimensional system, meaning it can handle both columns and rows, unlike flexbox which is largely a 1-dimensional system.

29. What are the different ways to hide the element using CSS?

1. Using display property (**display: none**). It's not available for screen readers. The element will not exist in the DOM if display: none is used.
2. Using visibility property (**visibility: hidden**), will take up the space of the element. It will be available to screen reader users. The element will actually be present in the DOM, but not shown on the screen.
3. Using position property (**position: absolute**). Make it available outside the screen.

30. What does the :root pseudo-class refer to?

The :root selector allows you to target the highest-level "parent" element in the DOM, or document tree. It is defined in the CSS Selectors Level 3 specification.

31. How do I restore the default value of a property?

The keyword initial can be used to reset it to its default value.

32. Difference between CSS grid vs flexbox?

1. CSS Grid Layout is a two-dimensional system, meaning it can handle both columns and rows. Grid layout is intended for larger-scale layouts which aren't linear in design.
2. Flexbox is largely a one-dimensional system (either in a column or a row). Flexbox layout is most appropriate to the components of an application.

33. How does Calc work?

The CSS3 calc() function allows us to perform mathematical operations on property values. Instead of declaring, for example, static pixel values for an element's width, we can use calc() to specify that the width is the result of the addition of two or more numeric values.

```
.foo {
    Width: calc(100px + 50px)
}
```

34. What do CSS Custom properties variables mean?

Custom properties (sometimes referred to as CSS variables or cascading variables) are defined by users that contain specific values to be reused throughout a document. The value is set using -- notion. And the values are accessed using the var() function.

```
:root {
    --main-bg-color: brown
}
```

```
.one {
  color: white;
  background-color· var (--main-bg-color);
  margin: l0px,
  width: 50px,
  height: 5Opx;
  display: inline-block;
}
```

35. What is the difference between CSS variables and pre-processor (SASS, LESS, Stylus) variables?

1. CSS variables can be used without the need for a preprocessor. Currently, all the major browsers support the CSS variables.
2. CSS variable cascade. But the preprocessor variables don't cascade.
3. CSS variable can be accessed and manipulated in javascript.

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

1. It makes every element in the document include the padding and border in the element's inner dimension for the height and width computation.
2. In box-sizing: border-box, The height of an element is now calculated by the content's height + vertical padding + vertical border width.
3. The width of an element is now calculated by the content's width + horizontal padding + horizontal border width.

37. What does !important mean in CSS?

The style is having the important will have the highest precedence and it overrides the cascaded property.

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

38. 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 the class value (or pseudo-class or attribute selector), the universal selector (*) has no specificity. ID selectors have a higher specificity than attribute selectors.

39. What are the advantages of using translate() instead of absolute position?

Translate() does not cause the browser to trigger repaint and layout and instead only acts on the compositor. The absolute position triggers the repaint or DOM reflow. So, translate() gives better performance.

40. How to determine if the browser supports a certain feature?

The @support in CSS can be very useful to scan if the current browser has support for a certain feature.

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

41. How does the absolute positioning work?

Absolute positioning is a very powerful positioning mechanism that allows users to place any element wherever they want in an exact location. The CSS properties right, left, top, bottom and define the exact locations where you need to place the element. In absolute positioning, the following points need to be considered:

1. The element to which the absolute positioning is applied is removed from the normal workflow of the HTML document.
2. The HTML layout does not create any space for that element in its page layout.
3. The element is positioned relative to the closest positioned ancestor. If no such ancestor is present, then the element is placed relative to the initial container block.
4. The final position of the element is determined based on values provided to the top, right, left, bottom.

42. How does this property work overflow: hidden?

The overflow property in CSS is used for specifying whether the content has to be clipped or the scrollbars have to be added to the content area when the content size exceeds the specified container size where the content is enclosed. If the value of overflow is hidden, the content gets clipped post the size of the container thereby making the content invisible. For example,

```css
div {
  width: 150px;
  height: 50px;
  overflow: hidden;
}
```

If the content of the div is very large and exceeds the height of 50px, the content gets clipped post 50px and the rest of the content is not made visible.

43. How will you align content inside the p tag at the exact center inside the div?

We can add the text-align: center property inside the parent div for aligning the contents horizontally. But it will not align the contents vertically. We can align the content vertically by making the parent element have relative positioning and the child element have absolute positioning. The child element should have the values of top, bottom, right, left as 0 to center it in the middle vertically. Then we need to set the margin as auto. It is assumed that both the child and mother elements will have height and width values.

Consider we have a div element of height and width taking 20% of the screen size, and we have a paragraph element taking the height of 1.2em and width of 20%. If we want to align the paragraph element at the center (vertically and horizontally), we write the following styles:

```
div {
    position : relative;  // Make position relative
    height : 20%;
    width : 20%;
    text-align : center; //Align to center horizontally
}
p {
    position : absolute; // Make position absolute
    top:0;              // Give values of top, bottom,left, right to 0
    bottom:0;
    left:0;
    right:0;
    margin : auto;       // Set margin as auto
    height : 1.2 em;
    width : 20%;
}
```

44. How is margin different from padding in CSS?

Margin property using which we can create space around the elements. We can also create space for borders defined at the exteriors. We have the following properties for defining the margin:

1. margin-top
2. margin-right
3. margin-bottom
4. margin-left
   margin property by itself has the values as:
5. auto – The browser auto-calculates the margin while we use this.
6. length – The value of this property can be in px, pt, cm, em etc. The values can be positive or negative.
7. % – We can also give percentage value as margin to the element.
8. inherit – Using this property, the margin properties can be inherited from the parent elements.

The padding property is used for generating the space around the element's content and inside any known border. The padding also has sub-properties like:

1. padding-top
2. padding-right
3. padding-bottom
4. padding-left

It is to be noted that the padding does not allow negative values.

From the below image, we can see that the Margin is the outermost entity of the CSS Box Model that lies outside of the borders whereas the padding lies within the borders.
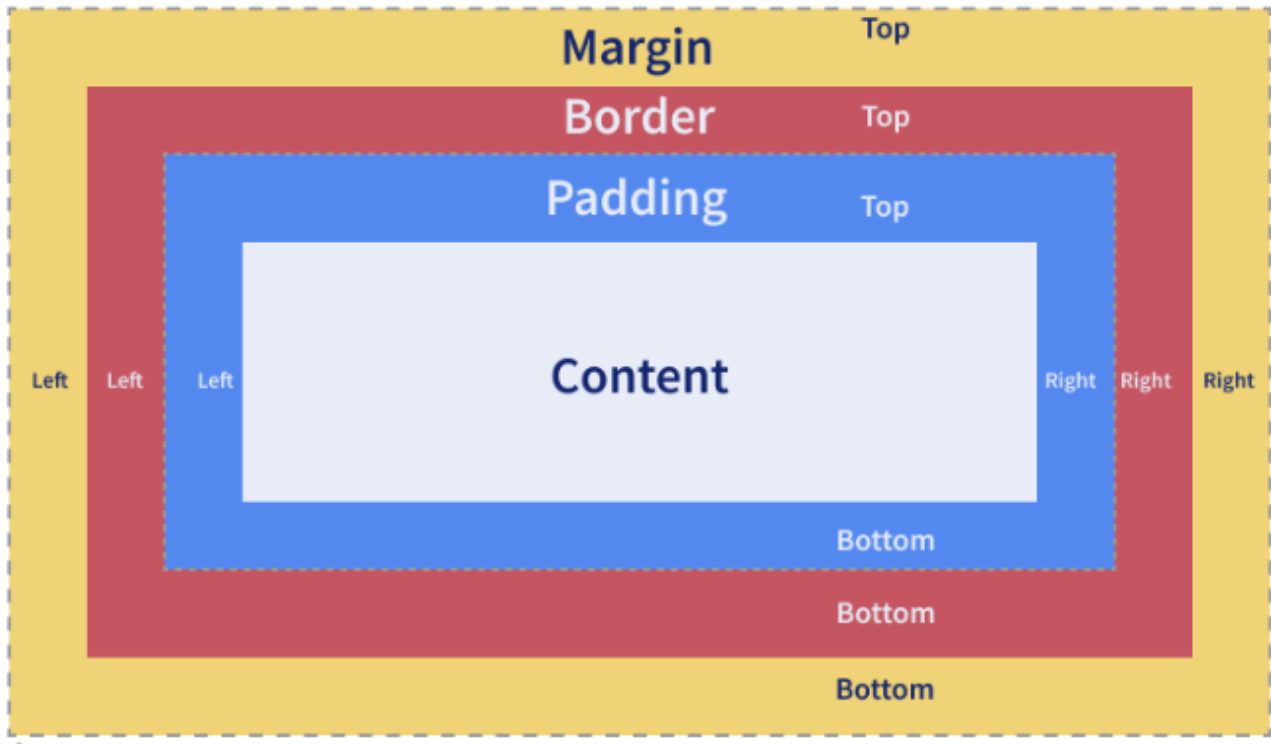
45. What do you have to do to automatically number the heading values of sections and categories?

We can use the concept of CSS counters. This lets us adjust the appearance of the content based on the location in a document. While using this, we need to first initialize the value of the counter-reset property which is 0 by default. The same property is also used for changing the value to any number that we need. Post initialization, the counter's value can be incremented or decremented by using the counter-increment property. The name of the counter cannot be CSS keywords like "none", "initial", "inherit" etc. If the CSS keywords are used, then the declaration would be ignored.

Consider an example as shown below:

```
body {
  counter-reset: header;  /* define counter named 'header' whose initial value is 0 by default */
}

h2::before {
  counter-increment: header;  /* The value of header counter by 1.*/
  content: "Header " counter(header) ": ";  /* To display word Header and the value of the counter with
colon before it.*/
}
```

Here, we are trying to achieve auto count increment and display feature for the h2 tag. Wherever we use h2 tag, the content will be prefixed by "Header 1 : " , "Header 2 : ", "Header 3 : " etc.

46. How is the nth-child() different from nth of type selectors?

Both are pseudo-classes (Pseudo-classes are those keywords that specifies the special state of the selected element). The nth-child() pseudo-class is used for matching elements based on the number that represents

the position of an element based on the siblings. The number is used to match an element on the basis of the element's position amongst its siblings.

For example, in the below piece of code, if we give nth-child(4) for the example class, then the 4th child of the example class is selected irrespective of the element type. Here, the fourth child of the example class is the div element. The element is selected and a background of black is added to it.

```
.example:nth-child(4) {
   background: black;
}
<div class="example">
   <p>This is a paragraph.</p>
   <p>This is a paragraph.</p>
   <p>This is a paragraph.</p>
   <div>This is a div.</div> <!-- 4th Element to select and apply style-->
   <div>This is a div.</div>
   <p>This is a paragraph.</p>
   <p>This is a paragraph.</p>
   <div>This is a div.</div>
</div>
```

The nth-of-type() pseudo-class is similar to the nth-child but it helps in matching the selector based on a number that represents the position of the element within the elements that are the siblings of its same type. The number can also be given as a function or give keywords like odd or even.

For example, in the below piece of code, if we give p:nth-of-type(even) for the example class, then all the even paragraph tags are selected within the example class and the style of background black is applied to them. The selected elements are marked in comments in the below code:

```
.example p:nth-of-type(even) {
   background: black;
}
<div class="example">
   <p>This is a paragraph.</p>
   <p>This is a paragraph.</p> <!-- Select this and apply style-->
   <p>This is a paragraph.</p>
   <div>This is a div.</div>
   <div>This is a div.</div>
   <p>This is a paragraph.</p> <!-- Select this and apply style-->
   <p>This is a paragraph.</p>
   <div>This is a div.</div>
   <p>This is a paragraph.</p> <!-- Select this and apply style-->
   <div>This is a div.</div>
</div>
```

47. Why do we need to use clear property along with floats in CSS?

The clear property along with floats is used for specifying which side of floating elements is not supposed to float. An element having clear property ensures that the element does not move up adjacent to the float. But the element will be moved down past the float.

Let us understand this with the help of an example. We know that the floated objects do not add to the height of the objects where they reside. Consider we have a div element with class "floated_div" within another div element with id "main_div".

```html
<html>
  <head>
  <style>
    #main_div {
        width: 400px;
        margin: 10px auto;
        border: 4px solid #cccccc;
        padding: 5px;
    }

    .floated_div {
        float: left;
        width: 50px;
        height: 50px;
        border: 2px solid #990000;
        margin: 10px;
    }
  </style>
  </head>
  <body>
    <div id="main_div">
        <p>Clear Float Demo</p>
        <div class="floated_div"></div>
        <div class="floated_div"></div>
        <div class="floated_div"></div>
        <div class="floated_div"></div>
        <div class="floated_div"></div>
    </div>
  </body>
</html>
```

The result of this code would be as shown below. We see that the squares that are expected to be within dev are not within the main parent div. How do we fix this?

Clear Float Demo
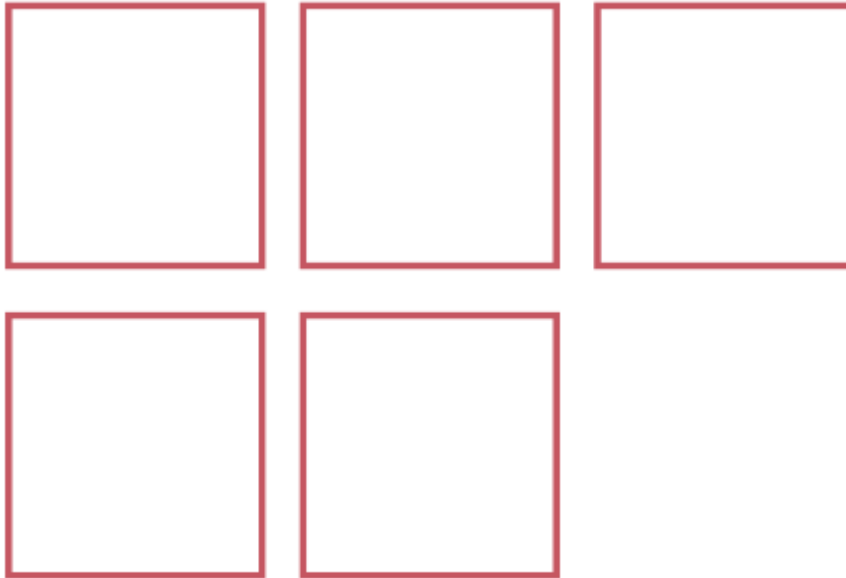
We can do it just by adding <div style="clear:both"></div> line at the end of the last floated element so that the floated elements are fit in properly within the main div container.

```html
<html>
  <head>
  <style>
    #main_div {
        width: 400px;
        margin: 10px auto;
        border: 4px solid #cccccc;
        padding: 5px;
    }

    .floated_div {
        float: left;
        width: 50px;
        height: 50px;
        border: 2px solid #990000;
        margin: 10px;
    }
  </style>
  </head>
  <body>
    <div id="main_div">
        <p>Clear Float Demo</p>
```

```
        <div class="floated_div"></div>
        <div class="floated_div"></div>
        <div class="floated_div"></div>
        <div class="floated_div"></div>
        <div class="floated_div"></div>
        <div style="clear:both"></div>   <!-- Adding this fixed the issue -->
    </div>
  </body>
</html>
```

The result of this will be: