

# INTERVIEW QUESTION ON CSS

## 1. Why we use css?

Ans:

- o CSS stands for Cascading Style Sheet.
- o CSS is used to design HTML tags.
- o CSS is a widely used language on the web.
- o It helps the web designers to apply style on HTML tags.
- o **CSS** is used to control the style of a web document in a simple and easy way.

It is generally used with HTML to change the style of web pages and user interfaces

## What does CSS do

- o You can add new looks to your old HTML documents.
- o You can completely change the look of your website with only a few changes in CSS code.

---

## Applications of CSS

As mentioned before, CSS is one of the most widely used style language over the web. I'm going to list few of them here:

- **CSS saves time** - You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- **Pages load faster** - If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
- **Easy maintenance** - To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Superior styles to HTML** - CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Multiple Device Compatibility** - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different

versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.

- **Global web standards** - Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

## 2. How many ways we can apply styles in css?

Ans:

# Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

## External CSS

With an external style sheet, you can change the look of an entire website by changing just one file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```
"mystyle.css"
body {
  background-color: lightblue;
}

h1 {
  color: navy;
  margin-left: 20px;
}
```

## Internal CSS

An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

# Inline CSS

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

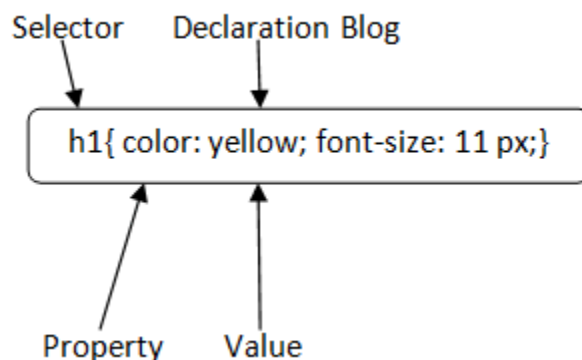
## 3. What is inline, internal and external style?

## 4. What is selector?

Ans:

# CSS Syntax

A CSS rule set contains a selector and a declaration block.



**Selector** – A selector is an HTML tag at which a style will be applied. This could be any tag like `<h1>` or `<table>` etc. etc.

**Declaration Block:** The declaration block can contain one or more declarations separated by a semicolon. For the above example, there are two declarations:

1. color: yellow;
2. font-size: 11 px;

Each declaration contains a property name and value, separated by a colon.

**Property:** A Property is a type of attribute of HTML element. It could be color, border etc.

**Value:** Values are assigned to CSS properties. In the above example, value "yellow" is assigned to color property.

1. Selector{Property1: value1; Property2: value2; .....;}

We can divide CSS selectors into five categories:

### 1. Simple selectors (select elements based on name, id, class)

## The CSS element Selector

The element selector selects HTML elements based on the element name.

```
p {  
  text-align: center;  
  color: red;  
}
```

## The CSS id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

```
#para1 {  
  text-align: center;  
  color: red;  
}
```

## The CSS class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

```
.center {  
  text-align: center;  
  color: red;  
}
```

## The CSS Universal Selector

The universal selector (\*) selects all HTML elements on the page.

```
* {  
  text-align: center;  
  color: blue;  
}
```

## The CSS Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

```
h1, h2, p {  
  text-align: center;  
  color: red;  
}
```

## 2. [Combinator selectors](#) (select elements based on a specific relationship between them)

# CSS Combinators

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:

- descendant selector (space)

## 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:

```
<!DOCTYPE html>
<html>
<head>
<style>
div p {
  background-color: yellow;
}
</style>
</head>
<body>

<h2>Descendant Selector</h2>
<p>The descendant selector matches all elements that are descendants of a specified element.</p>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section>
</div>
```

```
<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
</html>
```

- child selector (>)

## 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:

```
<!DOCTYPE html>
<html>
<head>
<style>
div > p {
  background-color: yellow;
}
</style>
</head>
<body>

<h2>Child Selector</h2>
<p>The child selector (>) selects all elements that are the children of a specified element.</p>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section> <!-- not Child but Descendant -->
  <p>Paragraph 4 in the div.</p>
</div>

<p>Paragraph 5. Not in a div.</p>
```



```
<p>Paragraph 6. Not in a div.</p>

</body>
</html>
```

- adjacent sibling selector (+)

## 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:

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

```
<!DOCTYPE html>
<html>
<head>
<style>
div + p {
  background-color: yellow;
}
</style>
</head>
<body>

<h2>Adjacent Sibling Selector</h2>

<p>The + selector is used to select an element that is directly after another specific element.</p>
<p>The following example selects the first p element that are placed immediately after div elements:</p>
```

```
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
</div>

<p>Paragraph 3. After a div.</p>
<p>Paragraph 4. After a div.</p>

<div>
  <p>Paragraph 5 in the div.</p>
  <p>Paragraph 6 in the div.</p>
</div>

<p>Paragraph 7. After a div.</p>
<p>Paragraph 8. After a div.</p>

</body>
</html>
```

- general sibling selector (~)

## General Sibling Selector (~)

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

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

```
<!DOCTYPE html>
<html>
<head>
<style>
div ~ p {
  background-color: yellow;
}
</style>
</head>
<body>
```

```
<h2>General Sibling Selector</h2>
<p>The general sibling selector (~) selects all elements that are siblings of a s
pecified element.</p>

<p>Paragraph 1.</p>

<div>
  <p>Paragraph 2.</p>
</div>

<p>Paragraph 3.</p>
<code>Some code.</code>
<p>Paragraph 4.</p>

</body>
</html>
```

### 3. Pseudo-class selectors (select elements based on a certain state)

## What are 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

```
<!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */
a:link {
```

```

    color: red;
}

/* visited link */
a:visited {
    color: green;
}

/* mouse over link */
a:hover {
    color: hotpink;
}

/* selected link */
a:active {
    color: blue;
}
</style>
</head>
<body>

<h2>CSS Links</h2>
<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order to be effective.</p>

</body>
</html>

```

**Ex:**

```

<!DOCTYPE html>
<html>
<head>
<style>
p:first-child {
    color: blue;
}
</style>
</head>
<body>

```

```
<p>This is some text.</p>
<p>This is some text.</p>

</body>
</html>
```

**Ex:**

```
<!DOCTYPE html>
<html>
<head>
<style>
p:nth-child(2) {
  color: blue;
}
</style>
</head>
<body>

<p>This is some text.</p>
<p>This is some text.</p>

</body>
</html>
```

#### 4. [Pseudo-elements selectors](#) (select and style a part of an element)

## What are Pseudo-Elements?

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

## The ::first-line Pseudo-element

The `::first-line` pseudo-element is used to add a special style to the first line of a text.

The following example formats the first line of the text in all <p> elements:

[illegible][illegible]

## The ::first-letter Pseudo-element

The `::first-letter` pseudo-element is used to add a special style to the first letter of a text.

The following example formats the first letter of the text in all <p> elements:

```
<!DOCTYPE html>
<html>
<head>
<style>
p::first-letter {
  color: #ff0000;
  font-size: xx-large;
}
</style>
</head>
<body>

<p>You can use the ::first-letter pseudo-element to add a special effect to the f
irst character of a text!</p>

</body>
</html>
```

**Y**ou can use the ::first-letter pseudo-element to add a special effect to the first character of a text!

**Ex:**

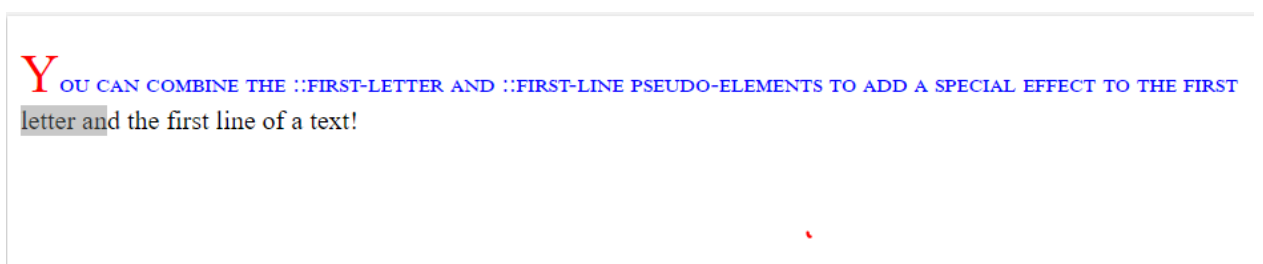
```
<!DOCTYPE html>
<html>
<head>
<style>
p::first-letter {
  color: #ff0000;
  font-size: xx-large;
}

p::first-line {
  color: #0000ff;
  font-variant: small-caps;
}
```

```
</style>
</head>
<body>

<p>You can combine the ::first-letter and ::first-line pseudo-elements to add a special effect to the first letter and the first line of a text!</p>

</body>
</html>
```



## CSS – The ::before Pseudo-element

The `::before` pseudo-element can be used to insert some content before the content of an element.

The following example inserts an image before the content of each `<h1>` element:

## CSS – The ::after Pseudo-element

The `::after` pseudo-element can be used to insert some content after the content of an element.

The following example inserts an image after the content of each `<h1>` element:

**Ex:**



```

<!DOCTYPE html>
<html>
<head>
<style>
h1::before {
  content: url(smiley.gif);
}


h2::after {
  content: url(smiley.gif);
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>The ::before pseudo-element inserts content before the content of an element.<
/p>


<h2>This is a heading</h2>

</body>
</html>

```

 **This is a heading**

The ::before pseudo-element inserts content before the content of an element.

**This is a heading** 

## CSS – The ::marker Pseudo-element

The **::marker** pseudo-element selects the markers of list items.

The following example styles the markers of list items:

```
<!DOCTYPE html>
<html>
<head>
<style>
::marker {
  color: red;
  font-size: 23px;
}
</style>
</head>
<body>

<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>

<ol>
  <li>First</li>
  <li>Second</li>
  <li>Third</li>
</ol>

</body>
</html>
```

## 5. Attribute selectors (select elements based on an attribute or attribute value)

# CSS [attribute] Selector

The `[attribute]` selector is used to select elements with a specified attribute.

The following example selects all `<a>` elements with a target attribute:

```
<!DOCTYPE html>
<html>
<head>
```

```

<style>
a[target] {
  background-color: yellow;
}
</style>
</head>
<body>

<h2>CSS [attribute] Selector</h2>
<p>The links with a target attribute gets a yellow background:</p>

<a href="https://www.w3schools.com">w3schools.com</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>

</body>
</html>

```

## CSS [attribute] Selector

The links with a target attribute gets a yellow background:

[w3schools.com](https://www.w3schools.com) [disney.com](http://www.disney.com) [wikipedia.org](http://www.wikipedia.org)

## CSS [attribute="value"] Selector

The `[attribute="value"]` selector is used to select elements with a specified attribute and value.

The following example selects all `<a>` elements with a `target="_blank"` attribute:

```

<!DOCTYPE html>
<html>
<head>
<style>

```

```

a[target=_blank] {
  background-color: yellow;
}
</style>
</head>
<body>

<h2>CSS [attribute="value"] Selector</h2>
<p>The link with target="_blank" gets a yellow background:</p>

<a href="https://www.w3schools.com">w3schools.com</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>

</body>
</html>

```

## CSS [attribute="value"] Selector

The link with target="\_blank" gets a yellow background:

[w3schools.com](https://www.w3schools.com) [disney.com](http://www.disney.com) [wikipedia.org](http://www.wikipedia.org) .

## Styling Forms

The attribute selectors can be useful for styling forms without class or ID:

```

<!DOCTYPE html>
<html>
<head>
<style>
input[type=text] {
  width: 150px;
  display: block;

```

```

margin-bottom: 10px;
background-color: yellow;
}

input[type=button] {
width: 120px;
margin-left: 35px;
display: block;
}
</style>
</head>
<body>

<h2>Styling Forms</h2>

<form name="input" action="" method="get">
  Firstname:<input type="text" name="Name" value="Peter" size="20">
  Lastname:<input type="text" name="Name" value="Griffin" size="20">
  <input type="button" value="Example Button">
</form>

</body>
</html>

```

## Styling Forms

Firstname:

Lastname:



[https://www.w3schools.com/css/css\\_attribute\\_selectors.asp](https://www.w3schools.com/css/css_attribute_selectors.asp)

### 5. What is pseudo selector?

## 6. How to provide colors in single cell of a table?

**Ans:**

```
<!DOCTYPE html>
<html>
<head>
  <style>
    table,th,td
    {
      border:2px solid red;
    }
    table
    {
      border-collapse:collapse;
      width:20%;
    }
    td
    {
      height:40px;
    }
    tr
    {
      background-color:green;
      color:white;
    }
    th
    {
      background-color:yellow;
      color:black;
    }
    td:nth-child(1) {
      background-color:blue;
      color:white;
    }
  #one{
    background-color: blueviolet;
  }
</style>
</head>
<body>
  <table>
    <tr>
      <th>Roll No</th>
      <th>Name</th>
```

```

        <th>Team</th>
    </tr>
    <tr>
        <td>1001</td>
        <td>John</td>
        <td>Red</td>
    </tr>
    <tr>
        <td id="one">1002</td>
        <td>Peter</td>
        <td>Blue</td>
    </tr>
    <tr>
        <td>1003</td>
        <td>Henry</td>
        <td>Green</td>
    </tr>
</table>

<!-- <script>
    document.getElementById("op").style.background = "#f2f2c3";
</script> -->
</body>
</html>

```

<http://www.corelangs.com/css/table/tablecolor.html>

<http://www.corelangs.com/js/basics/arrays.html>

**7. What is pseudo classes?**

**Ans:**

**8. What is media query?**

**Ans:**

Media query is a CSS technique introduced in CSS3.

It uses the `@media` rule to include a block of CSS properties only if a certain condition is true.

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 to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: pink;
}

@media screen and (min-width: 480px) {
  body {
    background-color: lightgreen;
  }
}
</style>
</head>
<body>

<h1>Resize the browser window to see the effect!</h1>
<p>The media query will only apply if the media type is screen and the viewport is 480px wide or wider.</p>

</body>
</html>
```

## 9. Write syntax for media query?



**10. Write a syntax for small device, medium device and large device using media query?**

**Ans:**

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
.example {
padding: 20px;
color: white;
}
/* Extra small devices (phones, 576px and down) max-width means apply condition s
tart from 0px to 600px */
@media only screen and (max-width: 576px) {

.extrasmall {background: red;}

}

/* Small devices (portrait tablets and large phones, 600px and up) min-width mea
ns start 600px start to rest*/
@media only screen and (min-width: 600px) {
.small {background: green;}
}

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {
.medium {background: blue;}
}

/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) {
.large {background: orange;}
}

/* Extra large devices (large laptops and desktops, 1200px and up) */
@media only screen and (min-width: 1200px) {
.extralarge {background: pink;}
}
</style>
</head>
<body>
```

```

<h2>Typical Media Query Breakpoints</h2>

<h4 class="extrasmall">Extra small devices (phones, 576px and down)</h4>
<h4 class="small">Small devices (portrait tablets and large phones, 600px and up)
</h4>
<h4 class="medium">Medium devices (landscape tablets, 768px and up)</h4>
<h4 class="large">Large devices (laptops/desktops, 992px and up)</h4>
<h4 class="extralarge">Extra large devices (large laptops and desktops, 1200px and up)</h4>

</body>
</html>

```

## Typical Media Query Breakpoints

Extra small devices (phones, 576px and down)

Small devices (portrait tablets and large phones, 600px and up)

Medium devices (landscape tablets, 768px and up)

Large devices (laptops/desktops, 992px and up)

Extra large devices (large laptops and desktops, 1200px and up)

## 11. Have you used flex box and css grid before? What are the difference between them?

**Ans:**

Before the Flexbox Layout module, there were four layout modes:

- Block, for sections in a webpage
- Inline, for text
- Table, for two-dimensional table data
- Positioned, for explicit position of an element

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

<https://www.srijan.net/blog/css-grid-vs-flexbox>

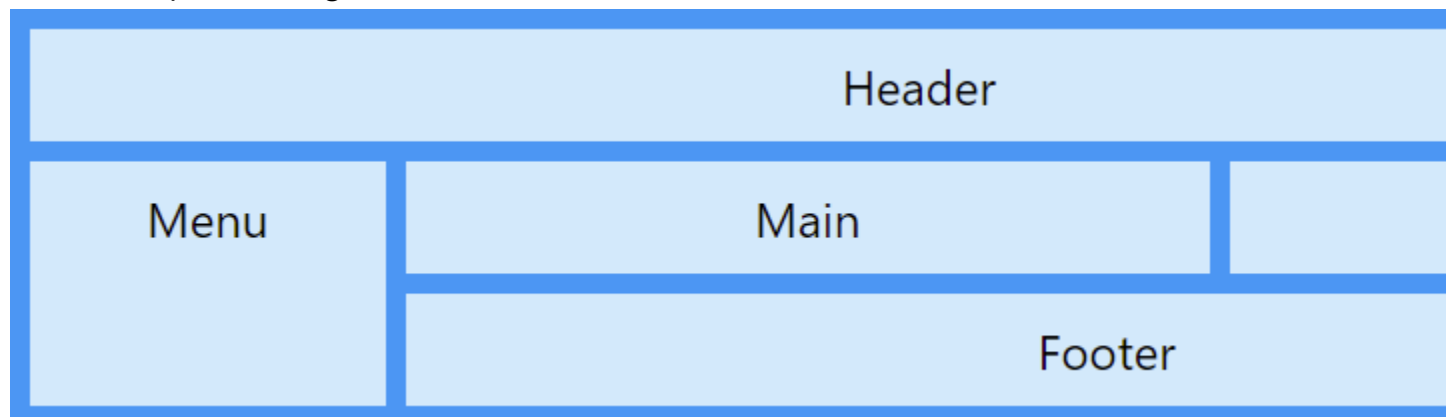
## Grid or Flexbox?

While CSS Flexbox offers a one-dimensional layout model powerful alignment and space distribution, CSS Grid lets you quickly create flexible and two-dimensional layouts.

They can both be used to place, size, align and [architect the website designs](#). But dimensions aren't the only value proposition between the two. Let's learn about CSS Grid and Flexbox together!

## The Fundamentals

The CSS Grid Layout offers a two-dimensional grid-based [layout system](#), with rows and columns. It makes designing web pages easier, without having to use floats and positioning.



*Layout design of CSS Grid*

The CSS Flexbox offers a one-dimensional layout. It is helpful in allocating and aligning the space among items in a container (made of grids). It works with all

kinds of display devices and screen sizes.



*Layout design of CSS Flexbox*

## A Quick Comparison

Element	Grid	Flexbox
Dimension	Two dimensional	One dimensional
Support type	Layout-first	Content-first
Page Responsiveness	Yes	Yes
Other features	Can flex a combination of items through space-occupying features	Can push content elements to extreme alignment
Browser Support	Firefox, Chrome, Safari, IE10+, IOS Safari	Firefox, Chrome, Safari, IE10+, IOS Safari, Opera

Let's take a look at the two major points of difference between Flex and Grid

## One vs Two Dimension

**Grid is made for two-dimensional layout while Flexbox is for one.** This means Flexbox can work on either row or columns at a time, but Grids can work on both.

When working on either element (row or column), you are most associated with the content. Flexbox, here, gives you more flexibility. HTML markup and CSS will be easy to manage in this type of scenario.

However, when working with the overall web page layouts, CSS Grid does away with the nightmares of coding. It gives you more flexibility to move around the blocks irrespective of your HTML markup.

## Content-First vs Layout-First

Another major difference between Flexbox and Grids is that the former works on content while the latter is based on the layout. Let's take a quick look:

### Flexbox

Let's say we want to recreate this design in Flex



HTML:

```
<header>

  <div>Home</div>

  <div>Search</div>

  <div>Logout</div>

</header>
```

CSS:

```
header {
  display: flex;
}

header > div:nth-child(3) {
  margin-left: auto;
}
```



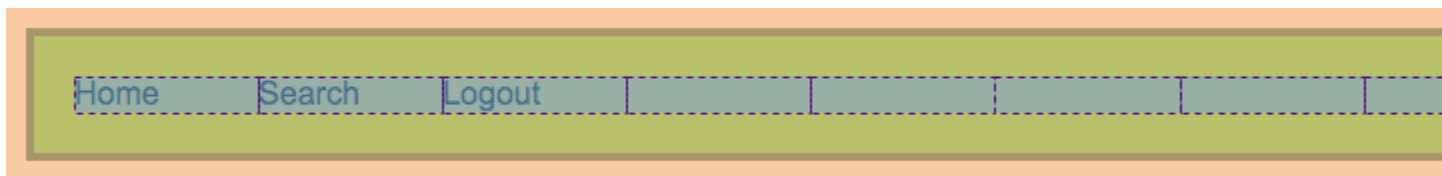
Items will place automatically in one line and you don't have to add anything else except `display: flex;` so the items are free to take their appropriate place.

Let's see how we will achieve the same design with CSS Grids.

```
header {  
  display: grid;  
  grid-template-columns: repeat(10, 1fr);  
}
```



Although they both look similar, the difference in layout can be seen with Chrome dev tools.



Can you see the grid columns & content? So what did Grid actually do?

It divided the wrapper into 10 columns and placed each item into those grid cells. It forced us to divide the layout into 10 columns. But this limitation was not there in the flexbox since it adjusts the items automatically.

We move forward with shifting one element (say, log out) to the extreme right.

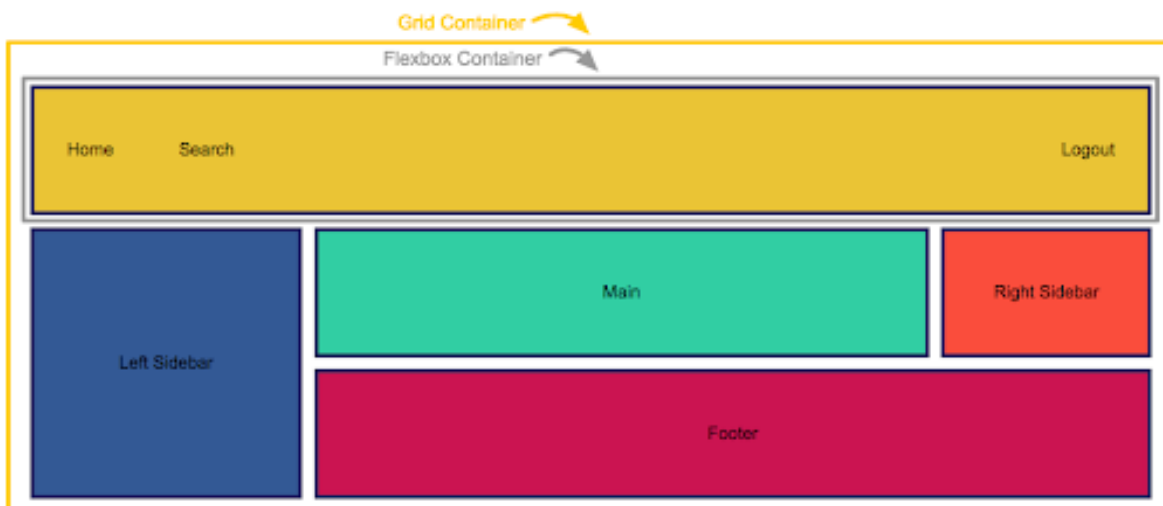
Here's how we do it.

```
header > div:nth-child(3) {  
  grid-column: 10;  
}
```



In this case, we are forcefully moving log out to the last column, which is wrong.

This is how it will look like when we will combine both the header and other web page blocks.



## Conclusion

- CSS Grids helps you create the outer layout of the webpage. You can build complex as well responsive design with this. This is why it is called 'layout first'.

- Flexbox mostly helps align content & move blocks.
- CSS grids are for 2D layouts. It works with both *rows* and *columns*.
- Flexbox works better in one dimension only (either rows OR columns).
- It will be more time saving and helpful if you use both at the same time. While Grid and Flexbox can be defined as the parent-children element, both have their own strengths and limitations. It is wise to understand their ability before deciding upon which one to choose.

## 12. What are @function/@mixin ?

**Ans:**

## 13. What are variable used for?

**Ans:**

## 14. What are attributed and how are they used?

**Ans:**

**An attribute is used to provide extra or additional information about an element.**

1. All HTML elements can have attributes. Attributes provide additional information about an **element**.
2. It takes two parameters : a name and a value. ...
3. Every name has some value that must be written within quotes.



## 15. Do you used any css pre processor and which do you prefer?

**Ans:**

A **CSS preprocessor** is a program that lets you generate [CSS](#) from the preprocessor's own unique [syntax](#). There are many CSS preprocessors to choose from, however most CSS preprocessors will add some features that don't exist in pure CSS, such as mixin, nesting selector, inheritance selector, and so on.

To use a CSS preprocessor, you must install a CSS compiler on your web [server](#); Or use the CSS preprocessor to compile on the development environment, and then upload compiled CSS file to the web server.

## 16. What is LESS and SASS?

**Ans:**

Sass is the most mature, stable, and powerful professional grade [css](#) extension language in the world.

Less (which stands for Leaner Style Sheets) is a backwards-compatible language extension for CSS. This is the official documentation for Less, the language and Less.js, the JavaScript tool that converts your Less styles to CSS styles.

## 17. Difference between css and css3?

**Ans:**

# Difference between CSS and CSS3

- Difficulty Level : [Expert](#)
- Last Updated : 15 Oct, 2020

**CSS:** CSS stands for Cascading Style Sheet. Its main objective is to provide styling and fashion to the web page. CSS provides color, layout, background,

font, and border properties. CSS features allow better content accessibility, enhanced flexibility, and control, as well as the specification of the characteristics of presentation.

**CSS3:** CSS3 stands for Cascading Style Sheet level 3, which is the advanced version of CSS. It is used for structuring, styling, and formatting web pages. Several new features have been added to CSS3 and it is supported by all modern web browsers. The most important feature of CSS3 is the splitting of CSS standards into separate modules that are simpler to learn and use.

#### **Difference between CSS and CSS3:**

S.No

CSS	CSS3
1 CSS is capable of positioning texts and objects. CSS is somehow backward compatible with CSS3.	On the other hand, CSS3 is capable of making the web page more attractive and take less time to create it. If you write CSS3 code in CSS, it will be invalid.
2 Responsive designing is not supported in CSS	CSS3 is the latest version, hence it supports responsive design.
3 CSS cannot be split into modules.	Whereas, CSS3 can be breakdown into modules.
4 Using CSS, we cannot build 3D animation and transformation.	But in CSS3 we can perform all kinds of animation and transformations as it supports animation and 3D transformations.
5 CSS is very slow as compared to CSS3	Whereas, CSS3 is faster than CSS.
6 In CSS we have a good collection of unique color schemas and standard color.	Whereas, CSS3 has a good collection of HSL RGBA, HSLA, and gradient colors.

7	In CSS we can only use single text blocks.	But in CSS3 we can use multi-column text blocks
8	CSS does not support media queries.	But CSS3 supports media queries
9	CSS codes are not supported by all types of modern browsers.	Being the latest version, CSS3 codes are supported by all modern browsers.
10	In CSS, designers have to manually develop rounded gradients and corners.	But CSS3 provide codes for setting rounded gradients and corners
11	There is no special effect like shadowing text, text animation, etc. in CSS. The animation was coded in jQuery and JavaScript.	CSS3 has many advance features like text shadows, visual effects, and a wide range of font style and color.
12	In CSS, the user can add background colors to list items and lists, set images for the list items, etc.	Whereas, CSS3 list has a special <i>display</i> property defined in it. Even list items also have counter reset properties.

### **New features of CSS3:**

1. **Combinator:** CSS3 has a new General sibling combinator which matches up with sibling elements via the tilde (~) combinator.
2. **CSS Selectors:** CSS3 selectors are much advanced in comparison to simple selectors offered by CSS, and are termed as a sequence of easy to use and simple selectors.

3. **Pseudo-elements:** Plenty of new pseudo-elements have been added to CSS3 to give easy styling in depth. Even a new convention of double colons :: is also added.
4. **Border Style:** The latest CSS3 also has new border styling features like *border-radius*, *image-slice*, *image-source*, and values for “width stretch”, etc.
5. **Background style properties:** New features like *background-clip*, *size*, *style*, and *origin* properties have been added to CSS3.

## 18. What are the fluid properties of flex flow?

**Ans:**

## Definition and Usage

The `flex-flow` property is a shorthand property for:

- [flex-direction](#)
- [flex-wrap](#)

**Note:** If the elements are not flexible items, the `flex-flow` property has no effect.

```
<!DOCTYPE html>
<html>
<head>
<style>
#main {
  width: 200px;
  height: 200px;
  border: 1px solid #c3c3c3;
  display: flex;
  flex-flow: row-reverse wrap;
}

#main div {
  width: 50px;
  height: 50px;
}
</style>
```

```

</head>
<body>

<h1>The flex-flow Property</h1>

<div id="main">
  <div style="background-color:coral;">A</div>
  <div style="background-color:lightblue;">B</div>
  <div style="background-color:khaki;">C</div>
  <div style="background-color:pink;">D</div>
  <div style="background-color:lightgrey;">E</div>
  <div style="background-color:lightgreen;">F</div>
</div>

<p><b>Note:</b> Internet Explorer 10 and earlier versions do not support the flex-
-flow property.</p>

</body>
</html>

```

## 19. What is opacity?

**Ans:**

The **opacity** property specifies the opacity/transparency of an element.

```

<!DOCTYPE html>
<html>
<head>
<style>
img {
  opacity: 0.5;
}
</style>
</head>
<body>

<h1>Image Transparency</h1>
<p>The opacity property specifies the transparency of an element. The lower the v
alue, the more transparent:</p>

<p>Image with 50% opacity:</p>


```

```
</body>  
</html>
```

## 20. What is class and id in css?

**Ans:**

The HTML **class** attribute is used to specify a class for an HTML element. Multiple HTML elements can share the same class.

---

## Using The class Attribute

The **class** attribute is often used to point to a class name in a style sheet. It can also be used by a JavaScript to access and manipulate elements with the specific class name.

In the following example we have three **<div>** elements with a **class** attribute with the value of "city". All of the three **<div>** elements will be styled equally according to the **.city** style definition in the head section:

The HTML **id** attribute is used to specify a unique id for an HTML element.

You cannot have more than one element with the same id in an HTML document.

---

## Using The id Attribute

The **id** attribute specifies a unique id for an HTML element. The value of the **id** attribute must be unique within the HTML document.

The **id** attribute is used to point to a specific style declaration in a style sheet. It is also used by JavaScript to access and manipulate the element with the specific id.

The syntax for id is: write a hash character (#), followed by an id name. Then, define the CSS properties within curly braces {}.

In the following example we have an **<h1>** element that points to the id name "myHeader". This **<h1>** element will be styled according to the **#myHeader** style definition in the head section:

**21. How do we want to repeat image in css? Background image is very thin, not in pixels also that I want to show fully background image how ?**

**Ans:**

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("https://picsum.photos/200");
  background-repeat: repeat;
  background-size: cover;

}

h1,p{
  color: white;
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<p>Here, a background image is repeated only horizontally!</p>

</body>
</html>
```

22. Do you know photoshop?

Ans:

no

23. What is module in css ?

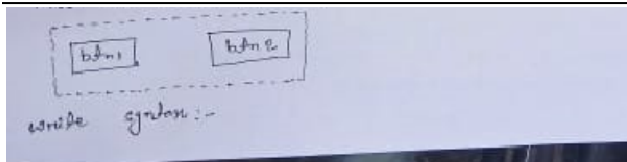
Ans:

*A CSS Module is a CSS file in which all class names and animation names are scoped locally by default.*

The key words here are **scoped locally**. With CSS Modules, your CSS class names become similar to local variables in JavaScript.

<https://www.javascriptstuff.com/what-are-css-modules/>

24. Btn1 btn2 =====> syntax for these two buttons ? ans :



```
<!DOCTYPE html>
<html>
<head>
<style>
.btn{
  border: 2px dashed black;
  width: 16%;
  padding: 10px;
  margin: 10px;
}
button{
```

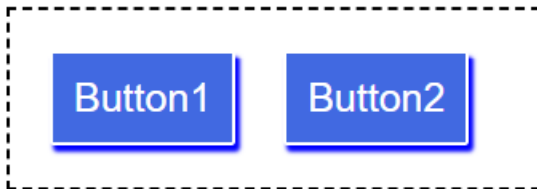


```
background-color: royalblue;
padding: 10px;
margin: 10px;
font-size: 20px;
border-radius: 2px;
border: 2px solid white;
box-shadow: 2px 3px 2px blue;
outline: none;
color: white;
}
</style>
</head>
<body>

    <div class="btn">
        <button>Button1</button>
        <button>Button2</button>

    </div>

</body>
</html>
```



Ans:

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS TASK1</title>
```

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

.dot{
    border-style: dashed;
    margin: 15px 15px 15px 15px;
    padding: 25px;
    width:250px;
    height:150px;
    text-align: center;
}

    .button {
background-color: #4CAF50; /* Green */
border: none;
color: white;
padding: 16px 32px;
text-align: center;
text-decoration: none;
display: inline-block;
font-size: 16px;
margin: 4px 2px;
transition-duration: 0.4s;
cursor: pointer;
    margin-top: 45px;
}

.button1{
    background-color: white;
    color: black;
    border: 2px solid #4CAF50;
}

.button1:hover{
    background-color: #4CAF50;
    color: white;
}

.button2 {
    margin-left: 25px;
    background-color: white;
    color: black;
    border: 2px solid #008CBA;
}

.button2:hover{
    background-color: #008CBA;
```

```

    color:white;
}

</style>
</head>
<body>

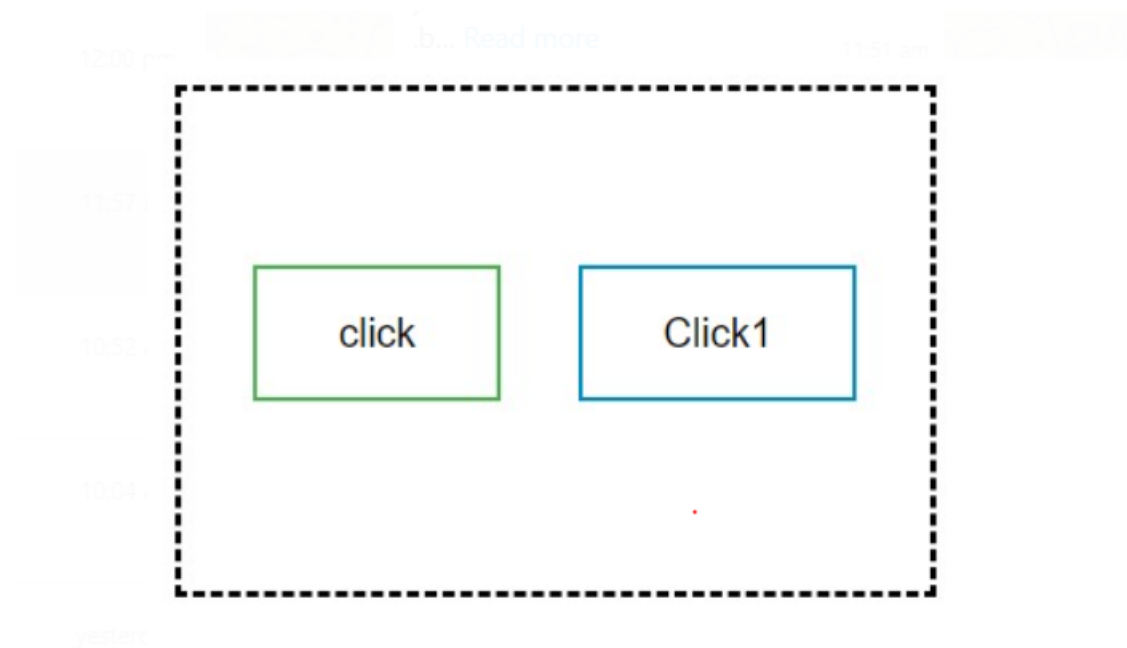
<div class="dot">
<button class="button button1">click</button>
<button class="button button2">Click1</button>

</div>

</body>
</html>

```

**o/p:**



**25. How many positions are there in css ?**

## Ans:

The **position** property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

# position: static;

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with **position: static;** is not positioned in any special way; it is always positioned according to the normal flow of the page:

## 1. Static

`position: static` is the **default value**. Whether we declare it or not, elements are positioned in a normal order on the webpage.

Let's give an example:

First, we define our HTML structure:

```
<!DOCTYPE html>
<html>
<head>
<style>
.box-orange {
    /* without any position declaration */
    background: orange;
    height: 100px;
    width: 100px;
}

.box-blue {
    background: lightskyblue;
    height: 100px;
    width: 100px;
    position: static;
    /* Declared as static */
}
```

```
</style>
</head>
<body>

    <div class="box-orange"></div>
    <div class="box-blue"></div>

</body>
</html>
```

## 2. Relative

# position: relative;

An element with `position: relative;` is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

This `<div>` element has `position: relative;`

Here is the CSS that is used:

`position: relative;` **An element's new position relative to its normal position.**

Starting with `position: relative` and for all **non-static** position values, we are able to change an element's **default** position by using the **helper properties** that I've mentioned above.

Let's move the orange box next to the blue one.

```
<!DOCTYPE html>
<html>
<head>
```

```

<style>
.box-orange {
  position: relative;
  /* // We are now ready to move the element */
  background: orange;
  width: 100px;
  height: 100px;
  top: 100px;
  /* // 100px from top relative to its old position */
  left: 100px;
  /* // 100px from left */
}

.box-blue {
  background: lightskyblue;
  height: 100px;
  width: 100px;
}
</style>
</head>
<body>

  <div class="box-orange"></div>
  <div class="box-blue"></div>

</body>
</html>

```

### 3. Absolute

## position: absolute;

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

**Note:** A "positioned" element is one whose position is anything except

In `position: relative`, the element is positioned **relative to itself**. However, an **absolutely** positioned element is **relative to its parent**.

An element with `position: absolute` is removed from the normal document flow. It is positioned automatically to the starting point (**top-left corner**) of its parent element. If it doesn't have any parent elements, then the initial **document <html>** will be its parent.

Since `position: absolute` removes the element from the document flow, other elements **are affected** and behave as the element is removed completely from the webpage.

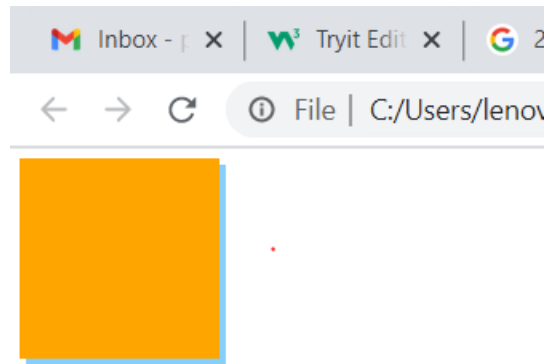
Let's add a **container** as parent element:

```
<!DOCTYPE html>
<html>
<head>
<style>
.box-orange {
    position: absolute;
    background: orange;
    width: 100px;
    height: 100px;
    left: 5px;
    top: 5px;
}

.box-blue {
    background: lightskyblue;
    height: 100px;
    width: 100px;
}
</style>
</head>
<body>

    <div class="container">
        <div class="box-orange"></div>
```

```
    <div class="box-blue"></div>
  </div>
</body>
</html>
```



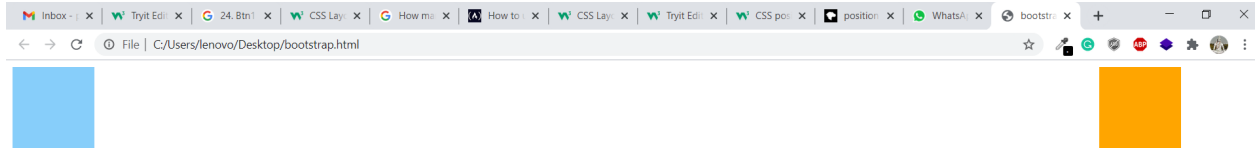
```
<!DOCTYPE html>
<html>
<head>
<style>
.box-orange {
  position: absolute;
  background: orange;
  width: 100px;
  height: 100px;
  right: 100px;
}

.box-blue {
  background: lightskyblue;
  height: 100px;
  width: 100px;
}
</style>
</head>
<body>

  <div class="container">
    <div class="box-orange"></div>
```



```
<div class="box-blue"></div>
</div>
</body>
</html>
```



## position: fixed;

An element with `position: fixed;` is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

### 4. Fixed

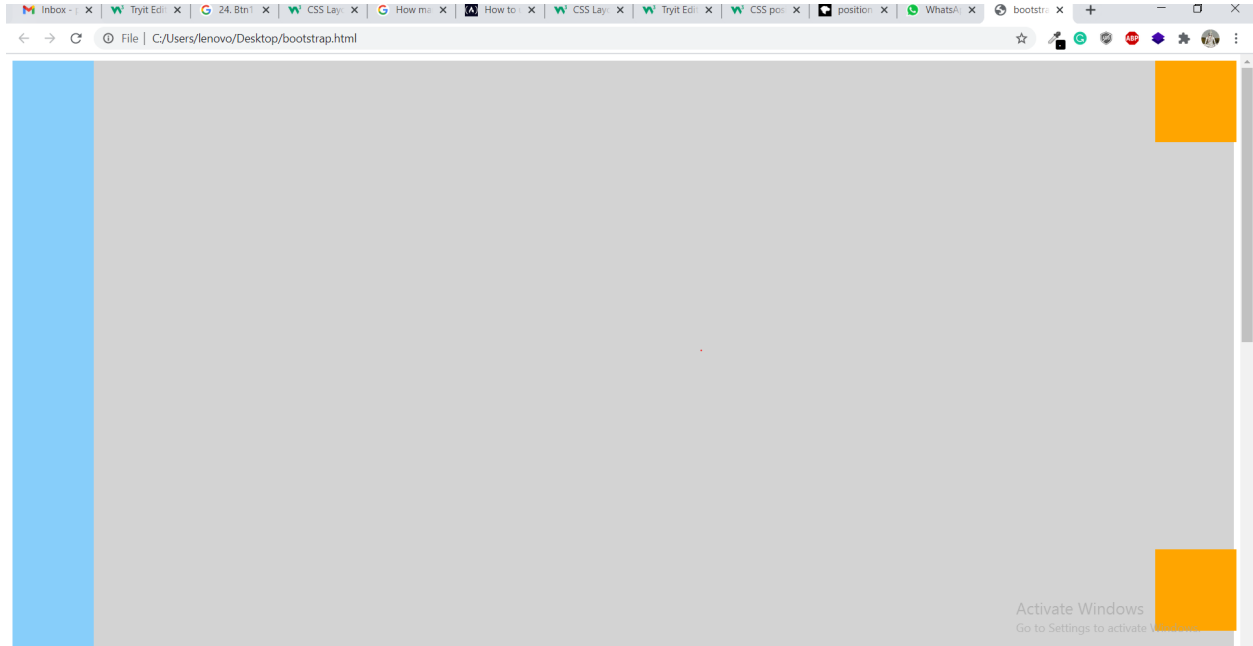
Like `position: absolute`, fixed positioned elements are also removed from the normal document flow. The differences are:

- They are **only relative to the <html> document**, not any other parents.
- They are **not affected by scrolling**.

```
<!DOCTYPE html>
<html>
<head>
<style>
```

[illegible]

```
</html>
```



## 5. Sticky

`position: sticky` can be explained as a mix of `position: relative` and `position: fixed`.

It behaves until a declared point like `position: relative`, after that it changes its behavior to `position: fixed`. The best way to explain **`position: sticky`** is by an example:

```
position: sticky;
```

An element with `position: sticky;` is positioned based on the user's scroll position.

A sticky element toggles between `relative` and `fixed`, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like `position: fixed`).

```
<!DOCTYPE html>
<html>
<head>
<style>

.box-blue {
  background: lightskyblue;
  height: 100px;
  width: 100px;
}

.container {
  position: relative;
  background: lightgray;
}

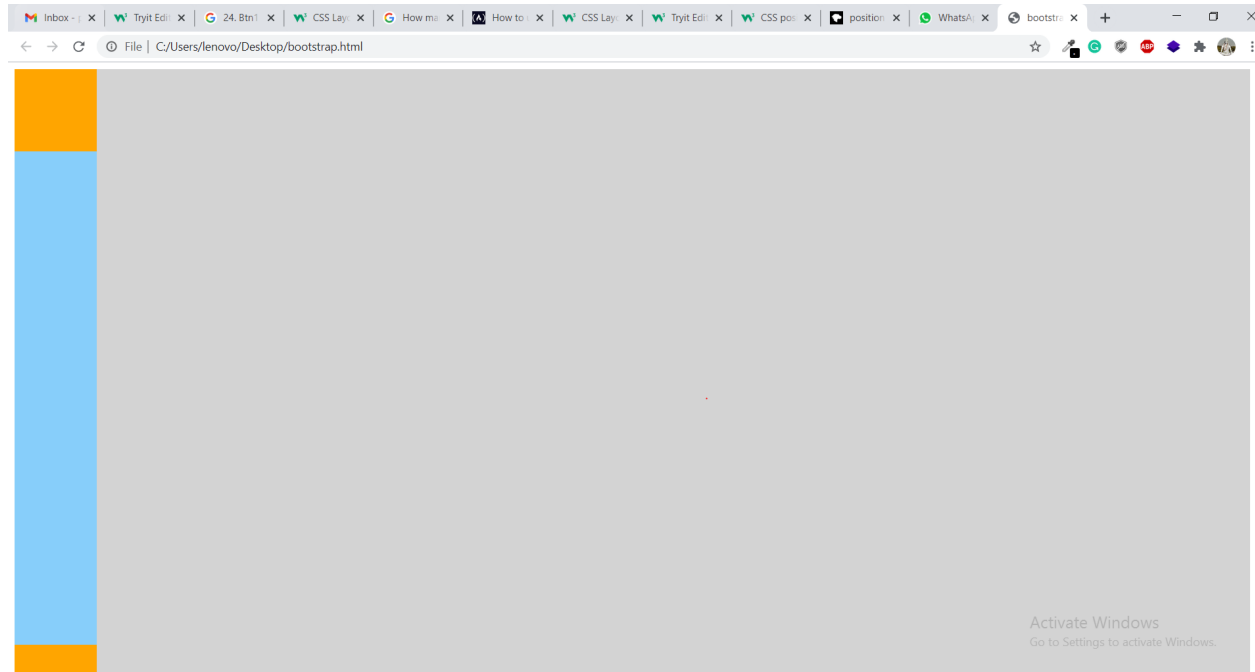
.box-orange {
  position: sticky;
  background: orange;
  width: 100px;
  height: 100px;
  right: 5px;
  /* // 5px relative to the most-right of parent */
}
</style>
</head>
<body>

  <div class="container">
    <div class="box-orange"></div>
    <div class="box-blue"></div>
    <div class="box-blue"></div>
    <div class="box-blue"></div>
    <div class="box-blue"></div>
    <div class="box-blue"></div>
    <div class="box-blue"></div>
    <div class="box-orange"></div>
    <div class="box-blue"></div>
    <div class="box-blue"></div>
    <div class="box-blue"></div>
    <div class="box-blue"></div>
    <div class="box-blue"></div>
    <div class="box-blue"></div>
    <div class="box-blue"></div>
  </div>
</body>
</html>
```

```

        <div class="box-blue"></div>
        <div class="box-blue"></div>
        <div class="box-blue"></div>
    </div>
</body>
</html>

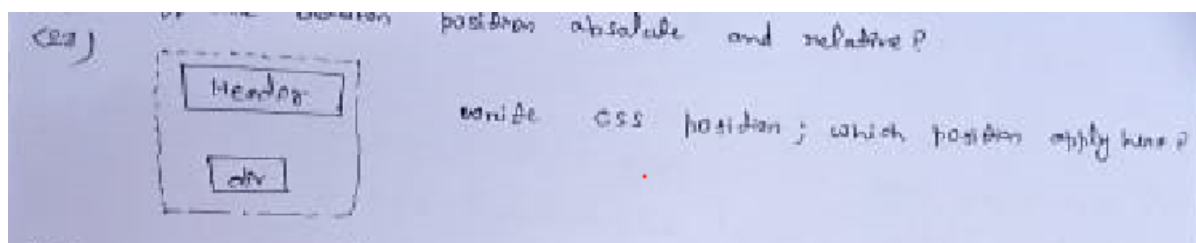
```



**26. Difference between position absolute and relative ?**

**27. Header**

**Div ==> write css position; which position apply here?**



```

<!DOCTYPE html>
<html>

```

```
<head>
<style>

.header {
  background: lightskyblue;
  height: 100px;
  width: 300px;
  margin: 30px;
}

.box-blue {
  position: relative;
  background: orange;
  width: 300px;
  height: 100px;
  margin: 30px;

  /* // 5px relative to the most-right of parent */
}

.container{
  border: 2px dashed black ;
  font-weight: bold;
  width: 40%;
  position: relative;
}
h1{
  text-align: center;
  padding-top: 30px;
  position: relative;
}
</style>
</head>
<body>
  <div class="container">

    <div class="header">
      <h1>Header</h1>
    </div>

    <div class="box-blue">
```

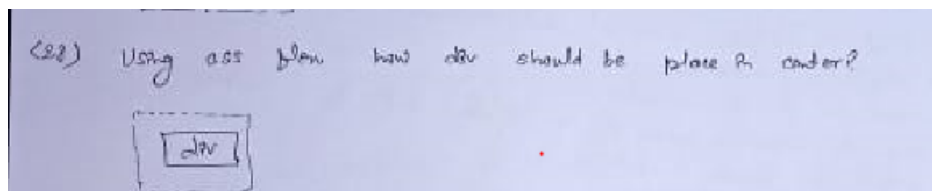
```
<h1>Div</h1>
</div>
</div>
</body>
</html>
```

**o/p:**



**28. Using css flex how div should be place in center?**

**Div**



```
<!DOCTYPE html>
<html>
<head>
<style>
```

```

.box {
  background: orange;
  width: 100px;
  height: 100px;
  margin: 30px;

  /* // 5px relative to the most-right of parent */
}

.container{
  display: flex;
  justify-content: center;
  align-items: center;
  border: 2px dashed black ;
  font-weight: bold;
  width: 300px;
  height: 300px;
}
h1{
  text-align: center;
  padding-top: 10px;
}
</style>
</head>
<body>
  <div class="container">

    <div class="box">
      <h1>Div</h1>
    </div>
  </div>
</body>
</html>

```

**o/p:**





**29. Write syntax how many ways we can placed a div at center?**

**Ans:**

```
<!DOCTYPE html>
<html>
<head>
<style>

h1 {
  color:green;
  padding-left : 47px;
}
h2 {
  padding-left: 11px;
}

</style>
</head>
```

```
<body>

  <h1>GeeksforGeeks</h1>
<h2>Center alignment of inside div</h2>
<div style="background-color:#4dff4d;width:20%;text-align:center;padding:7px;">
<div style="background-color:orange;width:50%;text-align:center;padding:2px;margin:0 auto">
  <h3 style="font-size:2vw;">Example of div inside a div.</h3>

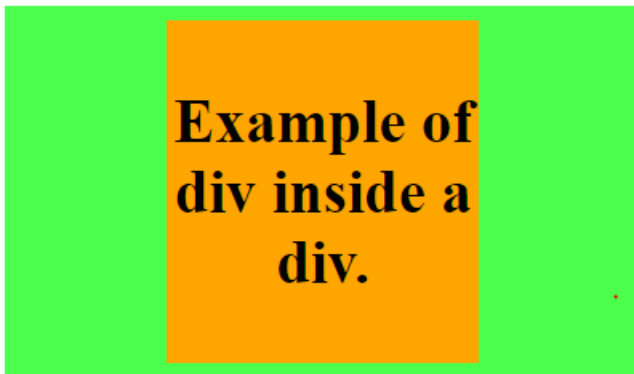
</div>
</div>

</body>
</html>
```

**o/p;**

# GeeksforGeeks

## Center alignment of inside div



<https://www.w3.org/Style/Examples/007/center.en.html>

### 1. CENTERING A BLOCK OR IMAGE

```
<!DOCTYPE html>
<html>
<head>
<style>

p.blocktext {
  margin-left: auto;
  margin-right: auto;
  width: 200px
  /* margin:auto */
}

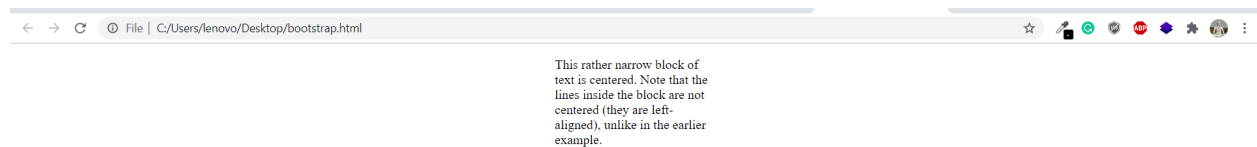
</style>
```

```
</head>
<body>

<P class="blocktext">
  This rather narrow block of text is centered. Note that the lines inside
  the block are not centered (they are left-aligned), unlike in the earlier exam
ple.
</P>

</body>
</html>
```

**o/p:**



**Image center:**

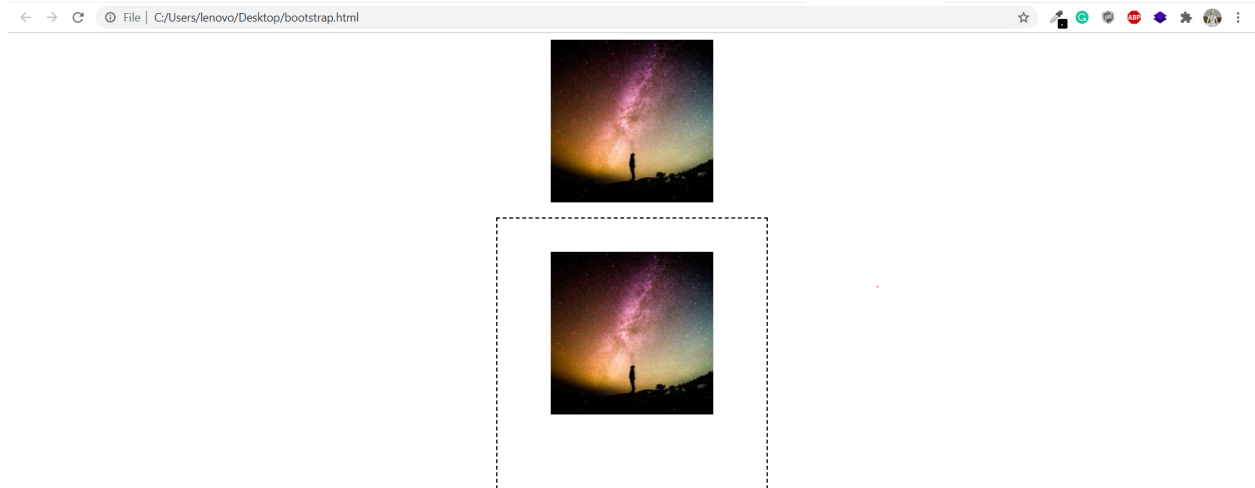
```
<!DOCTYPE html>
<html>
<head>
<style>
.displayed{
margin: auto;
display: block;

}

.displayed2{
  margin: auto;
display: block;
```

```
}  
.container{  
width: 250px;  
height: 250px;  
border: 2px dashed black;  
padding: 40px;  
margin: auto;  
}  
</style>  
</head>  
<body>  
  
    
  
  <br>  
  
  <div class="container">  
      
  
  </div>  
  
</body>  
</html>
```

**o/p:**



## CENTERING VERTICALLY AND HORIZONTALLY IN CSS LEVEL 3

```
<!DOCTYPE html>
<html>
<head>
<style>
div.container4 {
    height: 10em;
    position: relative ;
    border: 2px dashed black;
    width: 200px;
}

    div.container4 p {
    margin: 0;
    background: yellow;
    position: absolute;
    top: 50%;
    left: 50%;
    margin-right: -50%;
    transform: translate(-50%, -50%)
    }
/* by using flex */
    div.container6 {
    height: 10em;
    display: flex;
    align-items: center;
```

```

    justify-content: center;
    border: 2px dashed black;
    width: 200px;

}
div.container6 p {
    margin: 0 ;
    background-color: yellowgreen;
}
</style>
</head>
<body>

<!-- css3 -->
<div class=container4>
    <p>Centered!</p>
</div>
<br></br>
<!-- flex -->
<div class=container6>
    <p>Centered!</p>
</div>

</body>
</html>

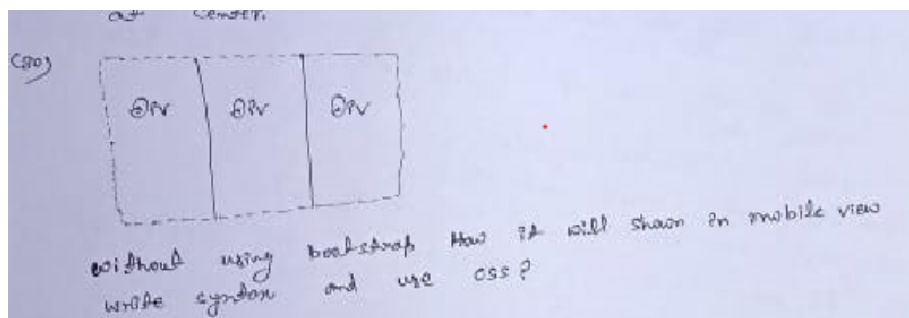
```

**o/p:**



30. Div div div => without using bootstrap how it will shown in mobile view write syntax and use css?

Ans:





```
<!DOCTYPE html>
<html>
<head>
<style>
div.one{
  width: 100px;
  height: 100px;
  background-color: yellowgreen;
}

div.two{
  width: 100px;
  height: 100px;
  background-color: lightcoral;
}

div.three{
  width: 100px;
  height: 100px;
  background-color: lightseagreen;
}
h1{
  text-align: center;
  padding-top: 10px;
}

.container{
  display: flex;
}
</style>
</head>
<body>

<div class="container">
  <div class="one">
    <h1>Div1</h1>
  </div>

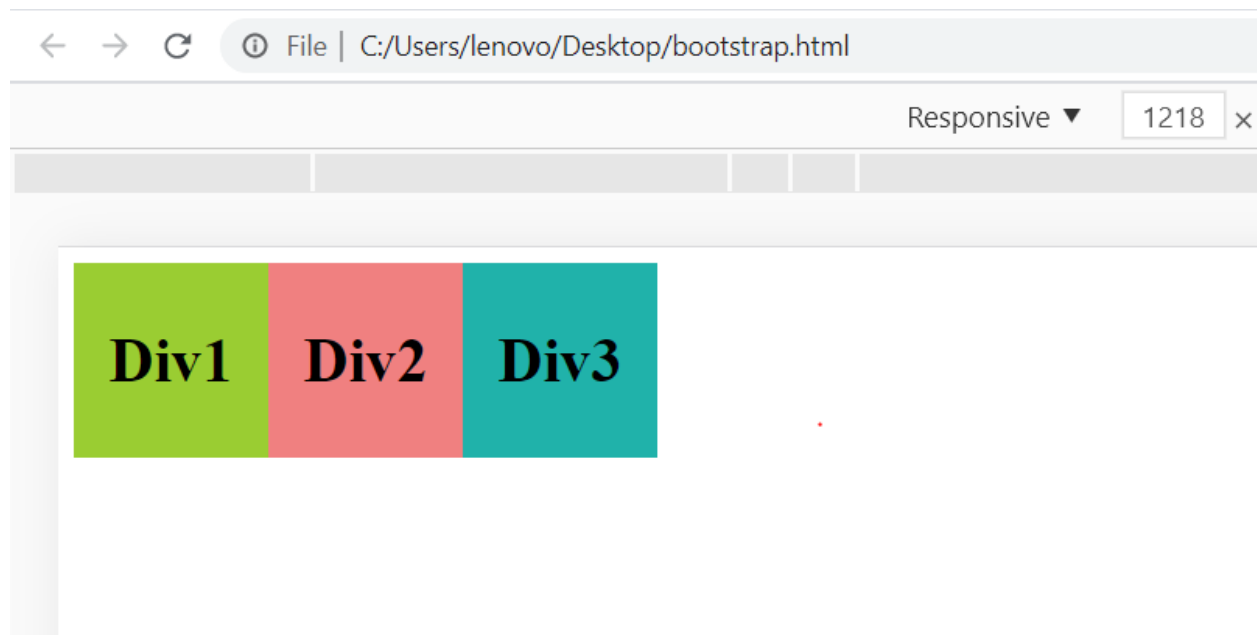
  <div class="two">
    <h1>Div2</h1>
  </div>

  <div class="three">
```

```
<h1>Div3</h1>
</div>
</div>

</body>
</html>
```

**o/p:**



**31. What is animations in css?**

**Ans:**

[https://www.w3schools.com/css/css3\\_animations.asp](https://www.w3schools.com/css/css3_animations.asp)

## What are CSS Animations?

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

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

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

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

```
<!DOCTYPE html>
<html>
<head>
<style>
h2 {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}

@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
</style>
</head>
<body>

<h2></h2>

</body>
</html>

<!-- example 2 -->

<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  width: 100px;
```

```

    height: 100px;
    background-color: red;
    position: relative;
    animation-name: example;
    animation-duration: 4s;
}

@keyframes example {
    0%    {background-color:red; left:0px; top:0px;}
    25%   {background-color:yellow; left:200px; top:0px;}
    50%   {background-color:blue; left:200px; top:200px;}
    75%   {background-color:green; left:0px; top:200px;}
    100%  {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>

<p><b>Note:</b> This example does not work in Internet Explorer 9 and earlier versions.</p>
<h1></h1>

</body>
</html>

<!-- example 3 -->

<!DOCTYPE html>
<html>
<head>
<style>
div {
    width: 100px;
    height: 100px;
    background-color: red;
    position: relative;
    animation-name: example;
    animation-duration: 5s;
    animation-timing-function: linear;
    animation-delay: 2s;
    animation-iteration-count: infinite;
    animation-direction: alternate;
}

@keyframes example {

```

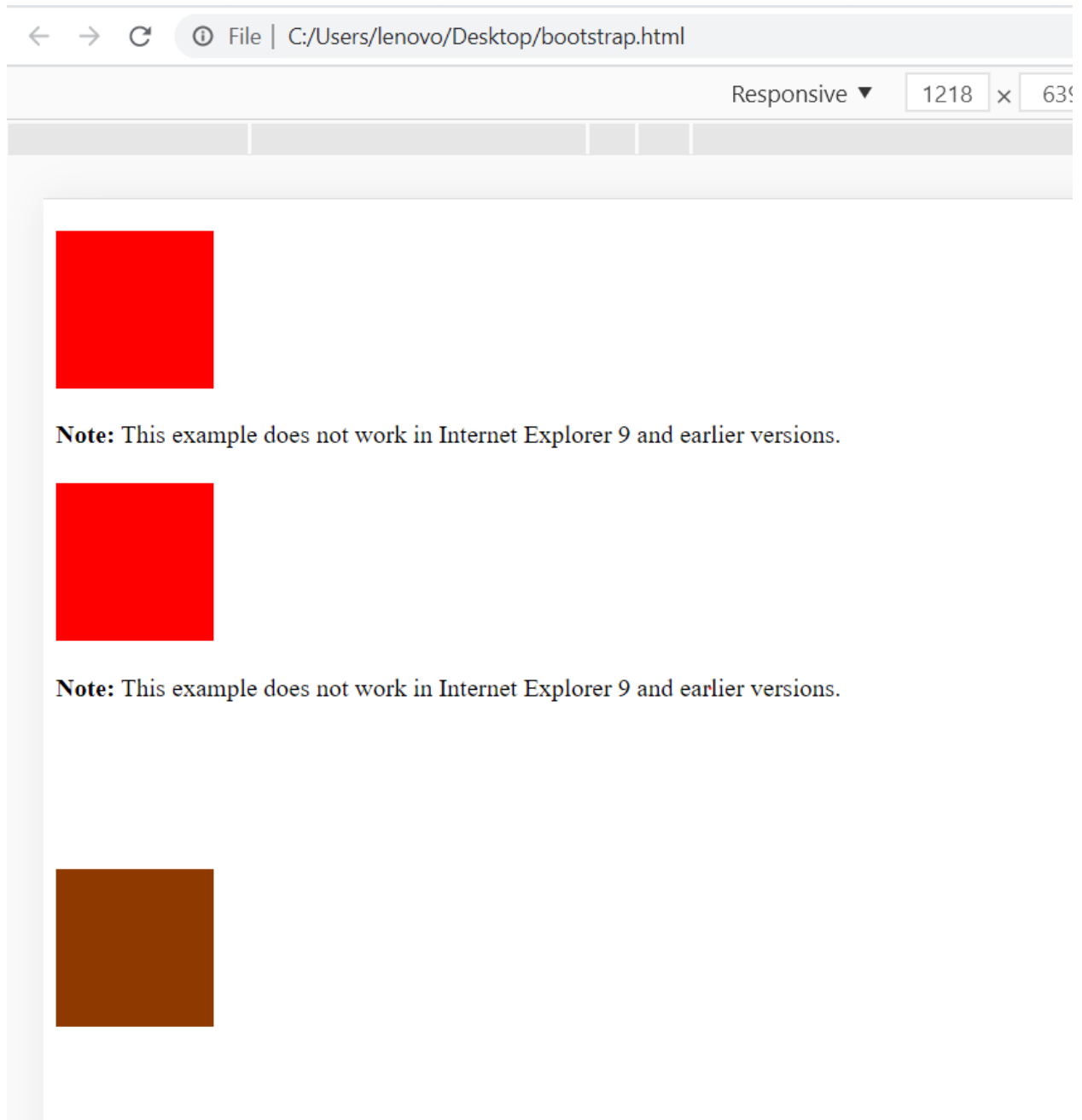
```
0% {background-color:red; left:0px; top:0px;}
25% {background-color:yellow; left:200px; top:0px;}
50% {background-color:blue; left:200px; top:200px;}
75% {background-color:green; left:0px; top:200px;}
100% {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>

<p><b>Note:</b> This example does not work in Internet Explorer 9 and earlier versions.</p>

<div></div>

</body>
</html>
```

**o/p:**



**32. Div1 div2 => I want to move div1 to div2 how can we do that? Use css?**

**Ans:**

```
<!DOCTYPE html>
<html>
<head>
```

```
<style>
div.one{
  width: 100px;
  height: 100px;
  background-color: yellowgreen;
  position: absolute;
  margin-left: 10px;
}

div.two{
  width: 100px;
  height: 100px;
  background-color: lightcoral;
}

h1{
  text-align: center;
  padding-top: 10px;
}

.container{
  display: flex;
}
</style>
</head>
<body>

<div class="container">
  <div class="one">
    <h1>Div1</h1>
  </div>

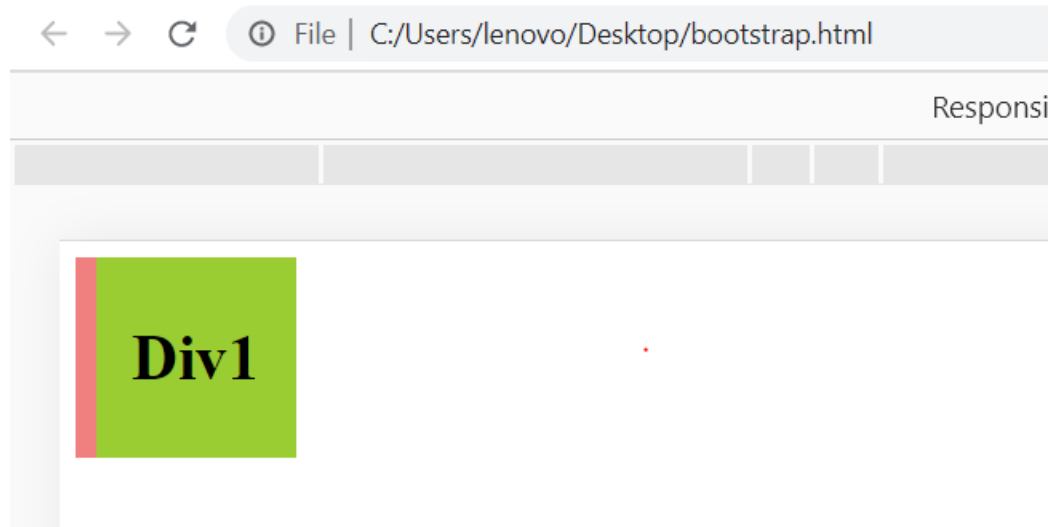
  <div class="two">
    <h1>Div2</h1>
  </div>

</div>

</body>
```

```
</html>
```

**o/p:**



### **33. What is box model?**

**Ans:**

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

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

## **CSS Box Model**

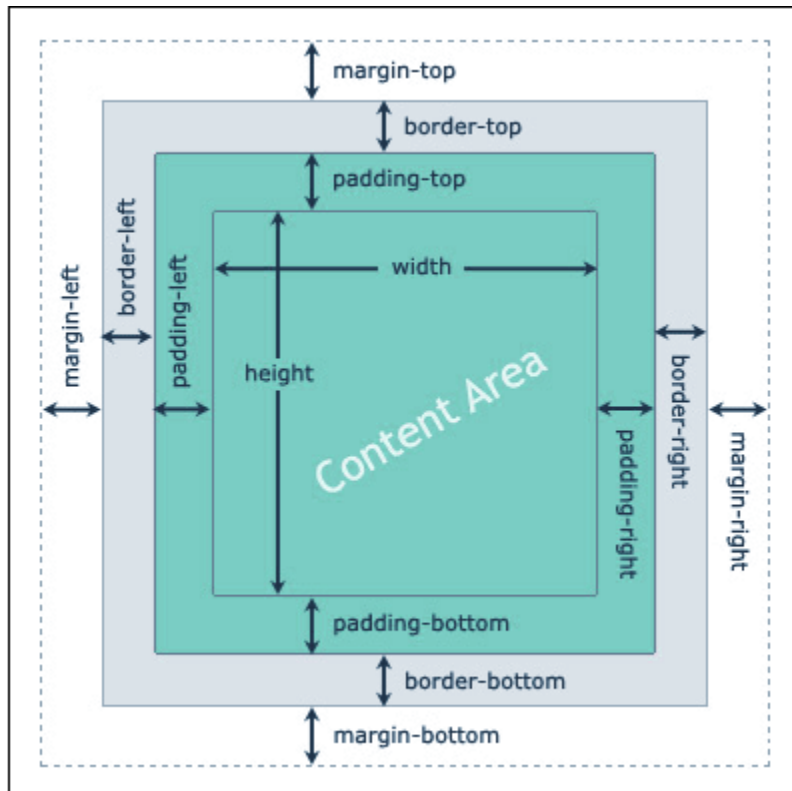
The components that can be depicted on the web page consist of one or more than one rectangular box.

A CSS box model is a compartment that includes numerous assets, such as edge, border, padding and material. It is used to develop the design and structure of a web page. It can be used as a set of tools to personalize the layout of different components.



According to the CSS box model, the web browser supplies each element as a square prism.

The following diagram illustrates how the CSS properties of [width](#), [height](#), [padding](#), [border](#) and [margin](#) dictate that how much space an attribute will occupy on a web page.



Explanation of the different parts:

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

### Border Field

It is a region between the padding-box and the margin. Its proportions are determined by the width and height of the boundary.

### Margin Field

This segment consists of the area between the boundary and the edge of the border.

The proportion of the margin region is equal to the margin-box width and height. It is better to separate the product from its neighbor nodes.

## Padding Field

This field requires the padding of the component. In essence, this area is the space around the subject area and inside the border-box. The height and the width of the padding box decide its proportions.

## Content Field

Material such as text, photographs, or other digital media is included in this area.

It is constrained by the information edge, and its proportions are dictated by the width and height of the content enclosure.

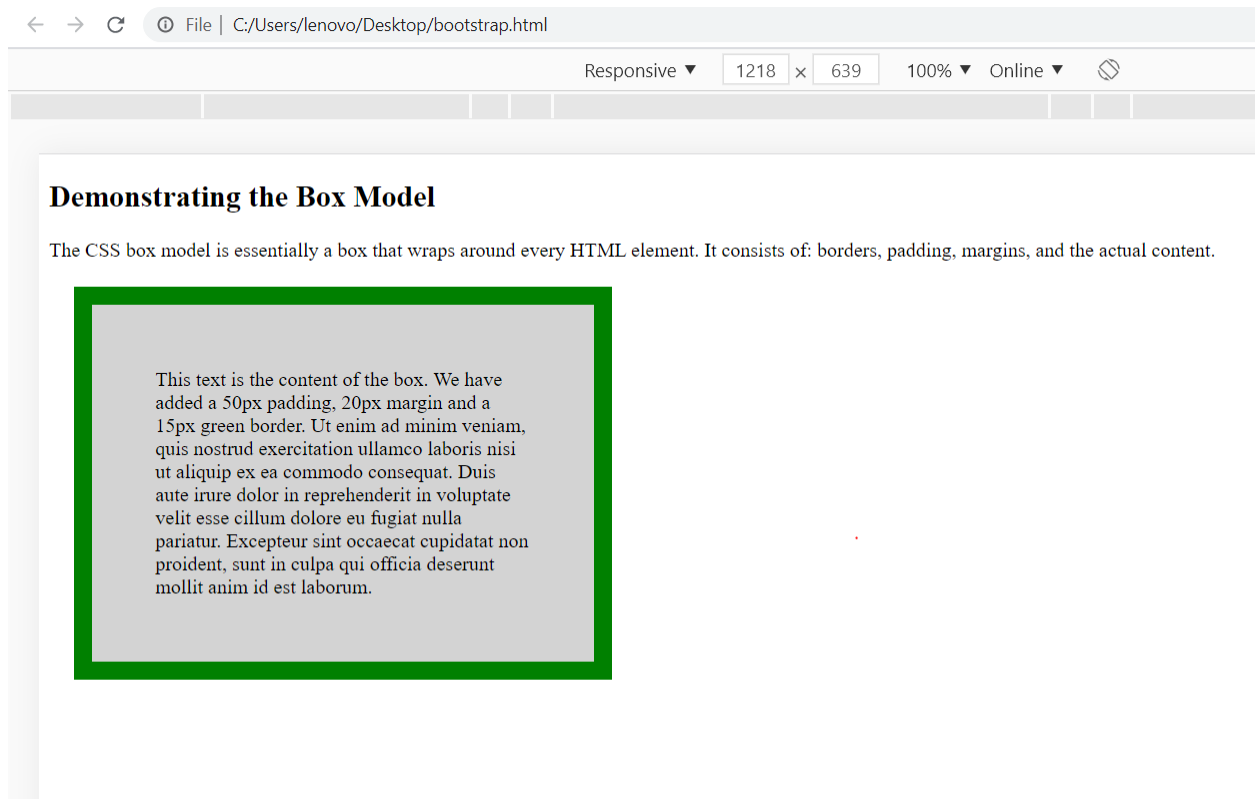
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: lightgrey;
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
</style>
</head>
<body>

<h2>Demonstrating the Box Model</h2>

<p>The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.</p>

<div>This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</div>

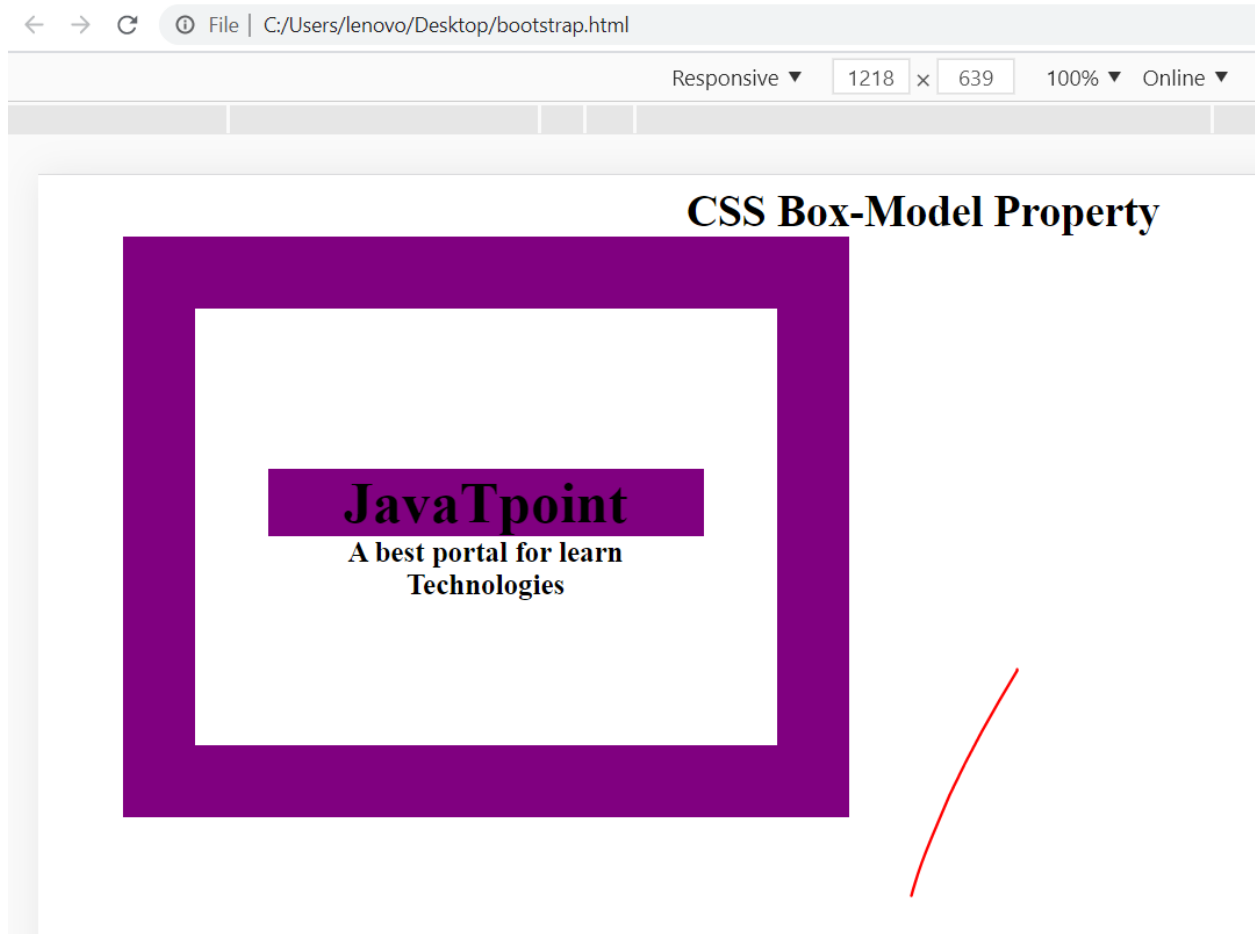
</body>
</html>
```



## Example 2

```
<!DOCTYPE html>
<head>
<title>CSS Box Model</title>
<style>
    .main
{
    font-size:30px;
    font-weight:bold;
    Text-align:center;
}
    .gfg
{
    margin-left:50px;
    border:50px solid Purple;
    width:300px;
```

```
        height:200px;
        text-align:center;
        padding:50px;
    }
    .gfg1
{
        font-size:40px;
        font-weight:bold;
        color:black;
        margin-top:60px;
        background-color:purple;
    }
    .gfg2
{
        font-size:20px;
        font-weight:bold;
        background-color:white;
    }
</style>
</head>
<body>
<div class = "main">CSS Box-Model Property</div>
    <div class = "gfg">
        <div class = "gfg1">JavaTpoint</div>
        <div class = "gfg2">A best portal for learn Technologies</div>
    </div>
</body>
</html>
```



**34. If we don't apply positions, then by default which position applied?**

**Ans:**

`position: static;`

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with `position: static;` is not positioned in any special way; it is always positioned according to the normal flow of the page:

This `<div>` element has `position: static;`

```
<!DOCTYPE html>
<html>
```

```

<head>
<style>
div.static {
  position: static;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: static;</h2>

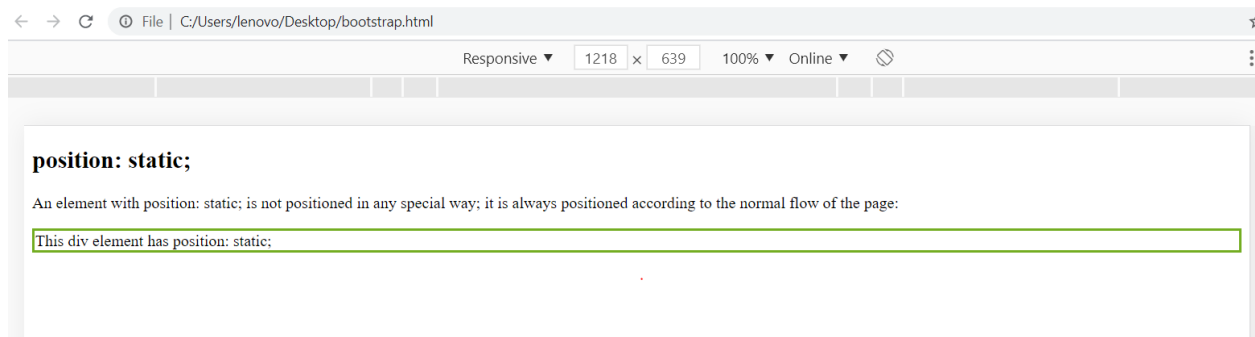
<p>An element with position: static; is not positioned in any special way; it is
always positioned according to the normal flow of the page:</p>

<div class="static">
  This div element has position: static;
</div>

</body>
</html>

```

**o/p:**



### 35. What are new things in css3 ?

**Ans:**

**CSS3** is the **latest** evolution of the Cascading Style Sheets language and aims at extending CSS2. 1. It brings a lot of **new features** and additions, like rounded corners, shadows, gradients, transitions or animations, as well as **new** layouts like multi-columns, flexible box or grid layouts.

Look at this list of CSS3 tutorials explaining the majority of modules:

1. User Interface
2. Box Model
3. Selectors
4. Backgrounds
5. Borders
6. Media Queries
7. Transforms
8. Text Effects
9. Multiple Column Layout
10. Animations

**36. If css1 and css2 not support responsive design then how people are doing responsive design?**

**Ans:**

**Bootsrap 3**

**37. What is the main purpose of text-align center ?**

**Ans:**

To just **center** the **text** inside an element, use **text-align: center;** This **text** is **centered**.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.a {
  text-align: center;
}

div.b {
```

```
    text-align: left;
}

div.c {
    text-align: right;
}

div.d {
    text-align: justify;
}
</style>
</head>
<body>

<h1>The text-align Property</h1>

<div class="a">
<h2>text-align: center:</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at
erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, conseq
uat gravida libero rhoncus ut.</p>
</div>

<div class="b">
<h2>text-align: left:</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at
erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, conseq
uat gravida libero rhoncus ut.</p>
</div>

<div class="c">
<h2>text-align: right:</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at
erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, conseq
uat gravida libero rhoncus ut.</p>
</div>

<div class="d">
<h2>text-align: justify:</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at
erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, conseq
uat gravida libero rhoncus ut.</p>
</div>

</body>
```



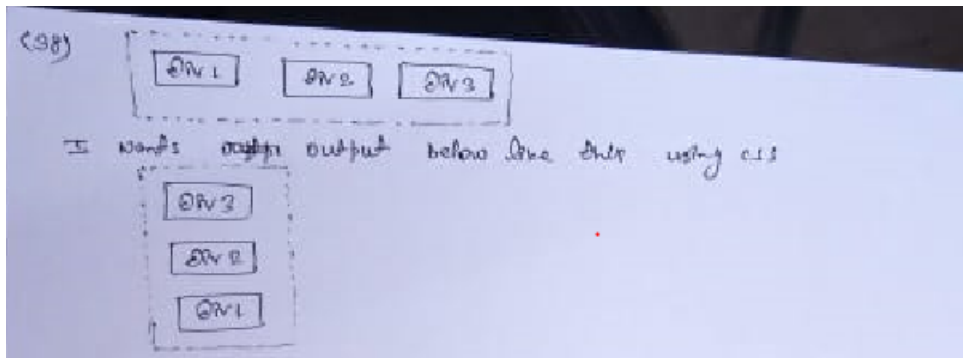
```
</html>
```

38. Div1 div2 div3 I want output below like this using css

Div3

Div2

Div1



```
<!DOCTYPE html>
<html>
<head>
<style>
div.one{
  width: 100px;
  height: 100px;
  background-color: yellowgreen;
}

.container{
  /* display: block; */
  /* display: flex; */
```

```
}

.one{
  /* float: left; */
  display: inline-block;
}
</style>
</head>
<body>

<div class="container">
  <div class="one">
    <h1>Div1</h1>
  </div>

  <div class="one">
    <h1>Div2</h1>
  </div>

  <div class="one">
    <h1>Div3</h1>
  </div>

</div>

</body>
</html>
```



```
<!DOCTYPE html>
<html>
<head>
<style>
div.one{
  width: 100px;
  height: 100px;
  background-color: yellowgreen;
}

.container{
  display: block;
  /* display: flex; */
}

.one{
  /* float: left; */
  /* display: inline-block; */
}
</style>
</head>
<body>
```

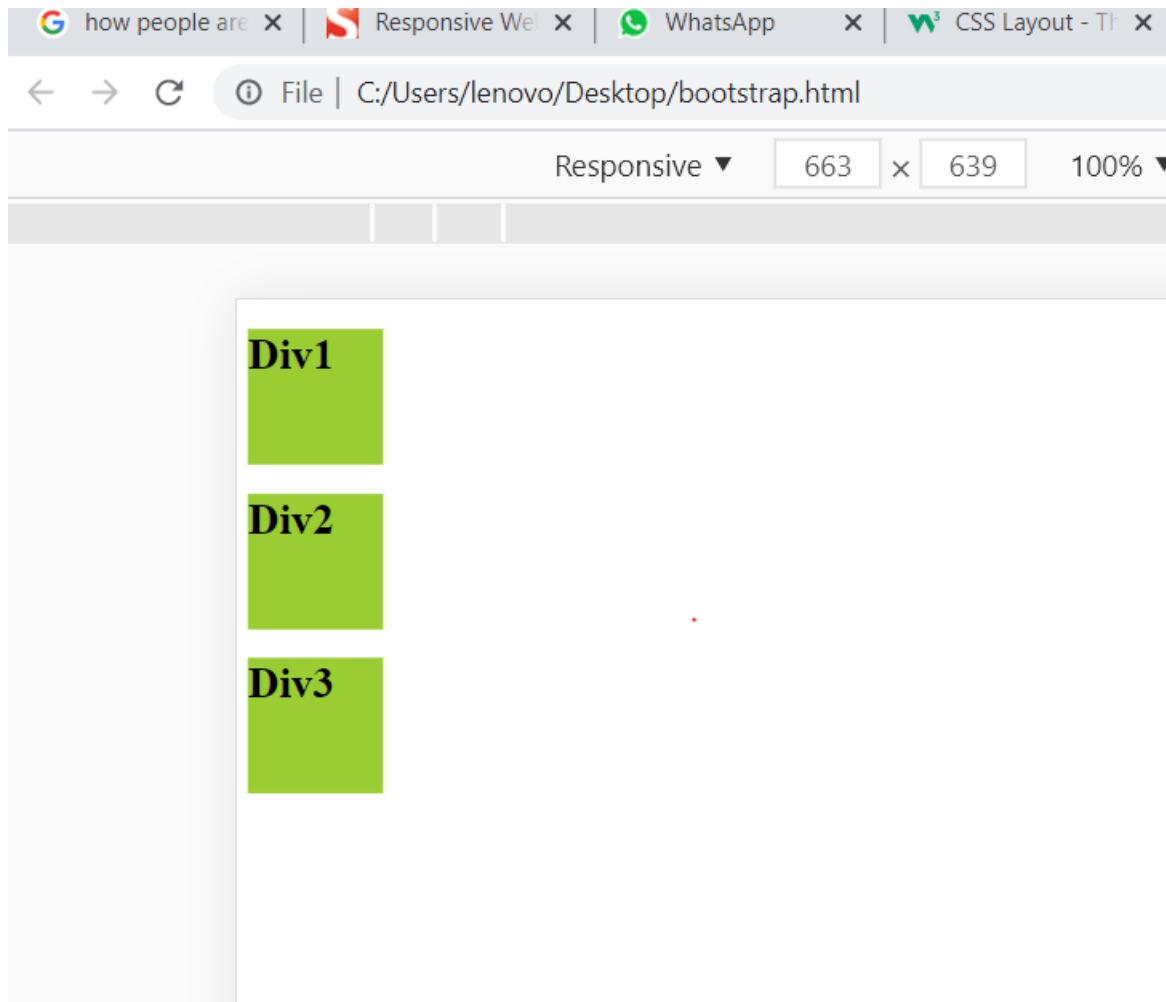
```
<div class="container">
  <div class="one">
    <h1>Div1</h1>
  </div>

  <div class="one">
    <h1>Div2</h1>
  </div>

  <div class="one">
    <h1>Div3</h1>
  </div>

</div>

</body>
</html>
```



**39. Explain what elements will match each of the following css selectors?**

- i. **Div, p => select all <div> elements and all <p> elements**
- ii. **Div p => select all <p> elements that are anywhere inside a element <div>**
- iii. **Div > p => select all <p> elements where the immediate point is a <div>....**
- iv. **Div + p => select all <p> elements that are placed immediately after a <div> element**

- v. **Div ~ p => select all <p> elements that are anywhere preceded by a <div> element**

#### 40. What is the difference between normal css and nested css?

##### Describe what a “reset” css file how it is useful?

A **CSS Reset** (or “**Reset CSS**”) is a short, often compressed (minified) set of **CSS** rules that resets the styling of all HTML elements to a consistent baseline. In case you didn't know, every browser has its own default 'user agent' **stylesheet**, that it uses to make unstyled websites appear more legible.

Ex:

```
/* CSS reset */
body,div,dl,dt,dd,ul,ol,li,h1,h2,h3,h4,h5,h6,pre,form,fieldset,input,textarea,p,blockquote,th,td {
    margin:0;
    padding:0;
}
html,body {
    margin:0;
    padding:0;
}
table {
    border-collapse:collapse;
    border-spacing:0;
}
fieldset,img {
    border:0;
}
input{
    border:1px solid #b0b0b0;
    padding:3px 5px 4px;
    color:#979797;
    width:190px;
}
address,caption,cite,code,dfn,th,var {
    font-style:normal;
    font-weight:normal;
}
ol,ul {
    list-style:none;
}
caption,th {
```

```

    text-align:left;
}
h1,h2,h3,h4,h5,h6 {
    font-size:100%;
    font-weight:normal;
}
q:before,q:after {
    content:'';
}
abbr,acronym { border:0;
}

```

**Are you familiar with normalize css? Do you understand how they differ?**

#### **41. What are gradients in css?**

**Ans:**

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines two types of gradients:

- **Linear Gradients (goes down/up/left/right/diagonally)**
- **Radial Gradients (defined by their center)**

**Ex:**

```

<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
    height: 100px;
    background-color: red; /* For browsers that do not support gradients */
    background-image: linear-gradient(0deg, red, yellow);
}

#grad2 {

```

```

height: 100px;
background-color: red; /* For browsers that do not support gradients */
background-image: linear-gradient(90deg, red, yellow);
}

#grad3 {
height: 100px;
background-color: red; /* For browsers that do not support gradients */
background-image: linear-gradient(180deg, red, yellow);
}

#grad4 {
height: 100px;
background-color: red; /* For browsers that do not support gradients */
background-image: linear-gradient(-90deg, red, yellow);
}
</style>
</head>
<body>

<h1>Linear Gradients - Using Different Angles</h1>

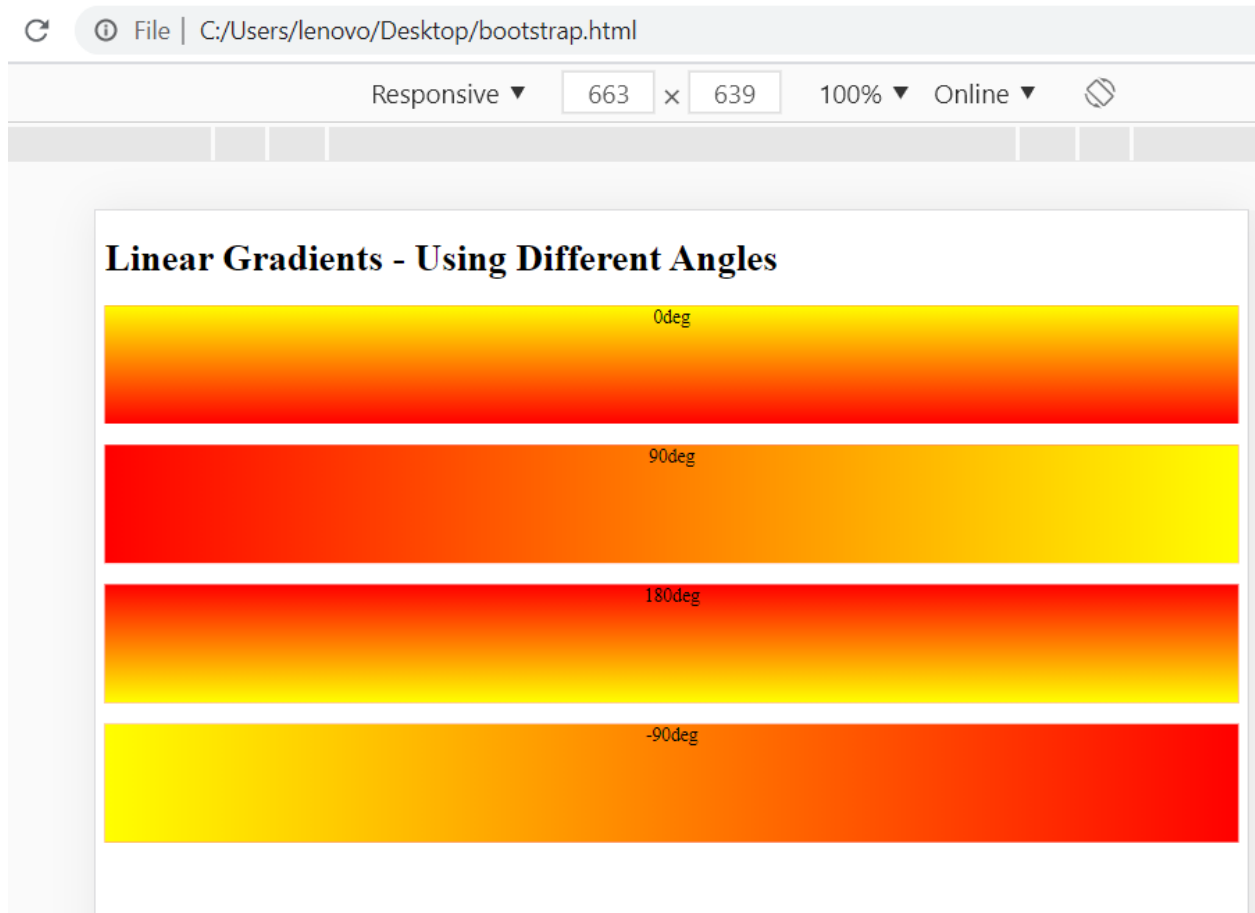
<div id="grad1" style="text-align:center;">0deg</div><br>
<div id="grad2" style="text-align:center;">90deg</div><br>
<div id="grad3" style="text-align:center;">180deg</div><br>
<div id="grad4" style="text-align:center;">-90deg</div>

</body>
</html>

```

**o/p:**





## 42. What is the float property of css?

**Ans:**

The **float CSS property** places an element on the left or right side of its container, allowing text and inline **elements** to wrap around it. The element is removed from the normal flow of the page, though still remaining a part of the flow (in contrast to absolute positioning).

The **float** property specifies how an element should float.

**Note:** Absolutely positioned elements ignore the **float** property!

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  float: right;
```

```

}
</style>
</head>
<body>
<h1>The float Property</h1>

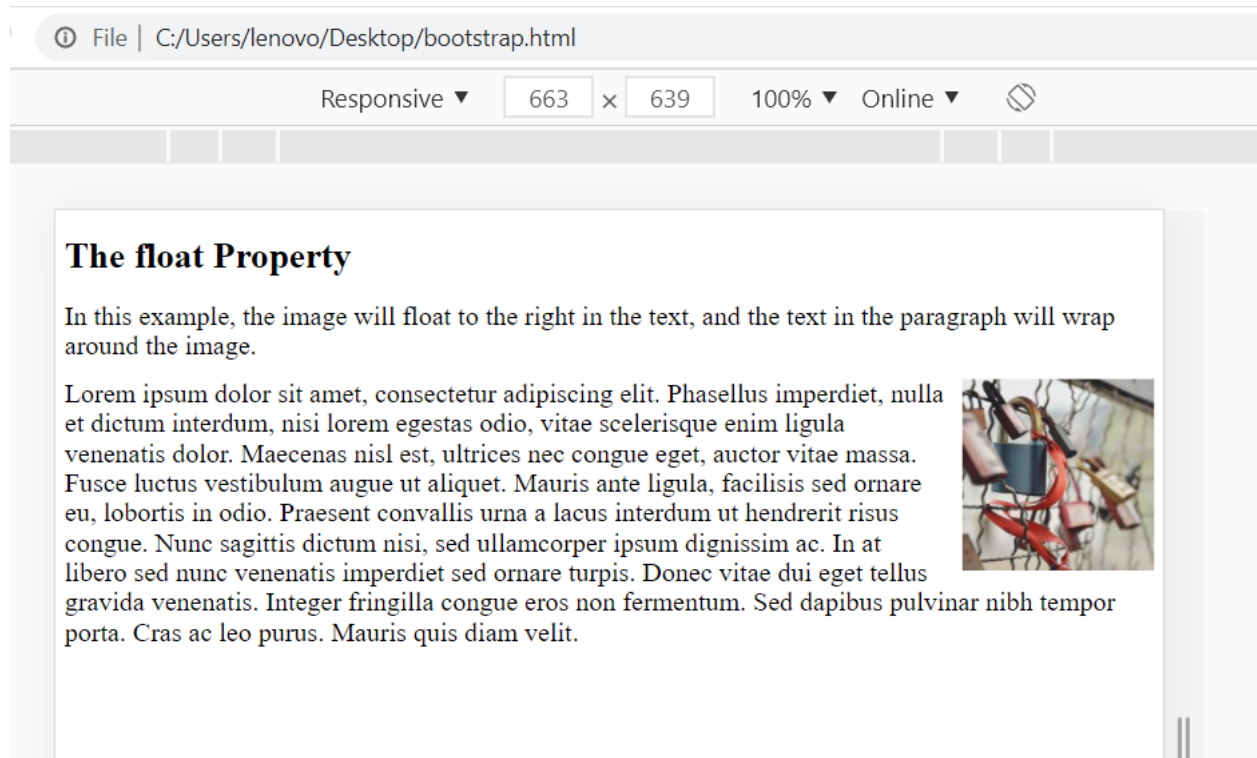
<p>In this example, the image will float to the right in the text, and the text i
n the paragraph will wrap around the image.</p>

<p>
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nul
la et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula ven
enatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fu
sce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare
eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus
congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at lib
ero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus g
ravidam venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvin
ar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.</p>

</body>
</html>

```

o/p:



#### 43. What is the difference between `visibility: hidden` and `display: none`?

**Ans:**

`visibility: hidden` means that unlike `display: none`, the tag is not visible, but space is allocated for it on the page. The tag is rendered, it just isn't seen on the page.

`display: none` means that the tag in question will not appear on the page at all (although you can still interact with it through the dom). There will be no space allocated for it between the other

#### 44. `<div class="div" id="div"> hello world </div>`

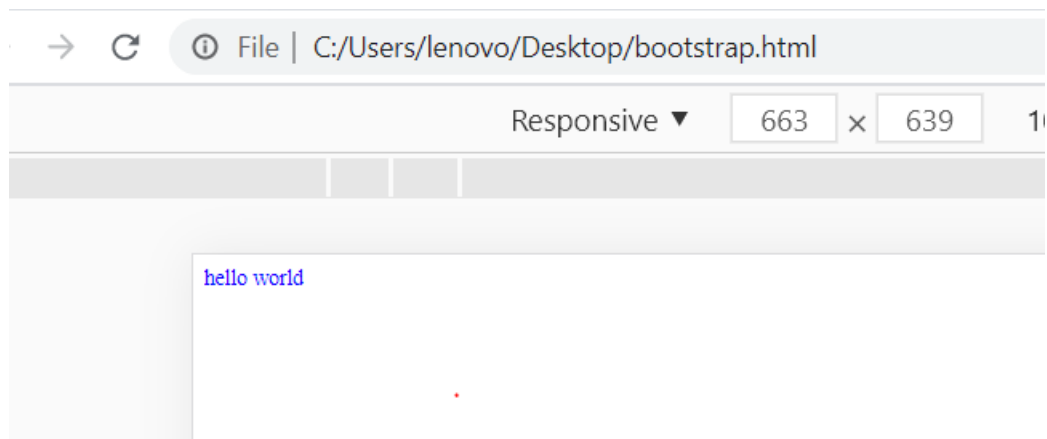
`.div{color:red;}, #div{color:blue;}`

**==> which color apply on the element?**

```
<!DOCTYPE html>
<html>
<head>
<style>

```

**o/p:**



Yes, in **one** single division you **can use** both but it's not very common. While styling you will call both so it will cause some ambiguity if you don't properly choose "x" and "y". **Use # for ID** and **.** for **class**

Nope, perfectly acceptable. A **class** is defined using a **.** and an **ID** is defined using a **#**. So as far as the browser is concerned, they're two totally separate items