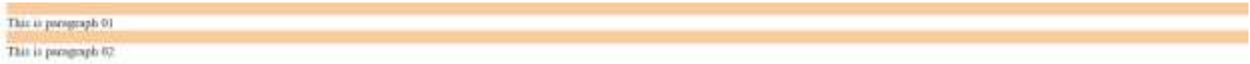
1. Give a brief description of the Box Model

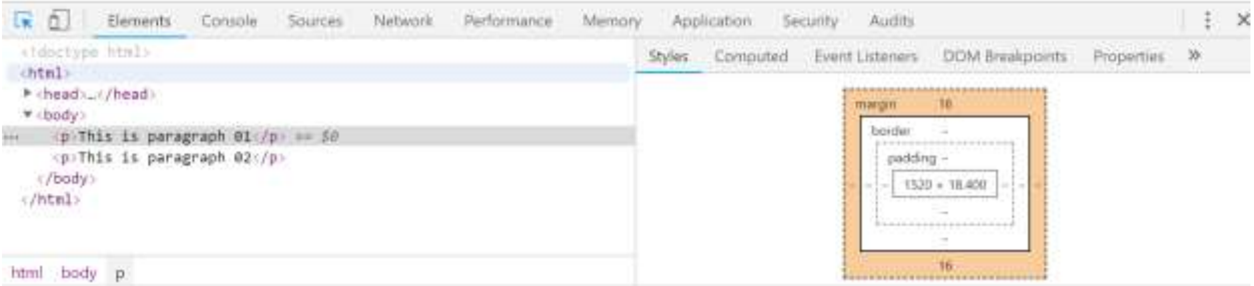
   Every html element can be considered as boxes which contain its content, padding, border and margin. In box model, elements place in vertically one after another occupying whole horizontal space by default.

   Every HTML elements have their own default properties.

   Below image shows how block model works.
   I have introduced two HTML <p> elements, but I haven't used any CSS styles for those <p> tags. But we can see there are two margins (16px) from the top and bottom of the paragraph-1 in peach color. This is because there is browser predefined properties in HTML elements.



This is paragraph 01
This is paragraph 02



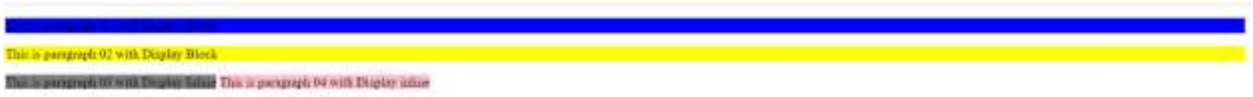
```
<!doctype html>
<html>
▶ <head>…</head>
▼ <body>
    <p>This is paragraph 01</p> == $0
    <p>This is paragraph 02</p>
  </body>
</html>
```

   Sometimes we don't need this much space in between two paragraphs. So depending on the requirement we can customized in those box models.

2. What is the difference between `display: inline;` and `display: block`?

   Display: block
   Default box model.
   Vertically sets elements one after another on the webpage. Always occupy space as much as possible introduced in the default box model. Always show browser default padding, margins and borders.

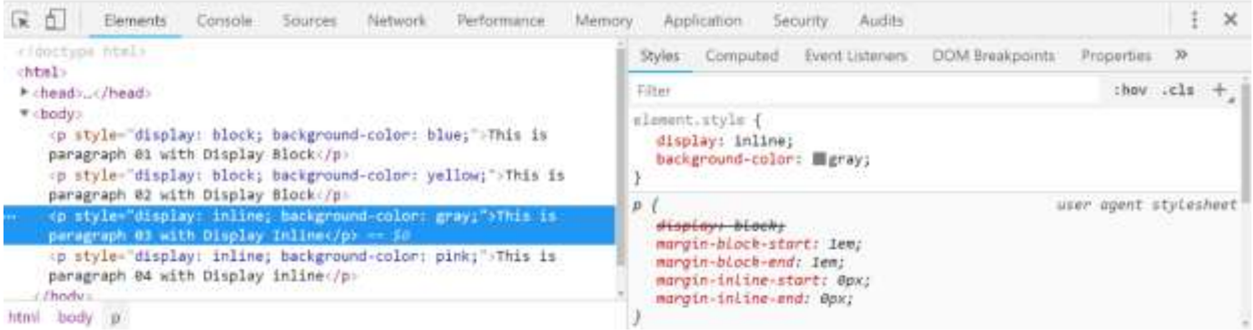
   Display: inline is opposite to box model, because it occupies only content space as a line, no line breaks, no heights and widths, but can only assign left and right paddings. Horizontally sets elements one after another on the webpage.



This is paragraph 01 with Display Block

This is paragraph 02 with Display Block

This is paragraph 03 with Display Inline This is paragraph 04 with Display inline



```
<!doctype html>
<html>
▸<head>…</head>
▾<body>
    <p style="display: block; background-color: blue;">This is
    paragraph 01 with Display Block</p>
    <p style="display: block; background-color: yellow;">This is
    paragraph 02 with Display Block</p>
…   <p style="display: inline; background-color: gray;">This is
    paragraph 03 with Display Inline</p> == $0
    <p style="display: inline; background-color: pink;">This is
    paragraph 04 with Display inline</p>
  </body>
```

```
element.style {
   display: inline;
   background-color: ■gray;
}
p {                                              user agent stylesheet
   ~~display: block;~~
   margin-block-start: 1em;
   margin-block-end: 1em;
   margin-inline-start: 0px;
   margin-inline-end: 0px;
}
```

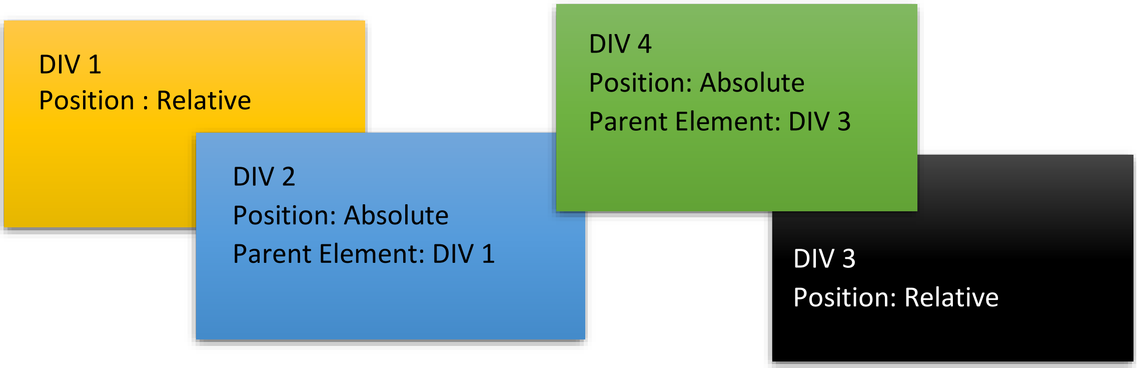
3. How is the `z-index` property used and why?

   It specifies the stack order of the elements. Simply z-index allows us to stack different elements in different depths in the view port. This allows us to arrange elements stack on each other. It always works with positioned elements. As an example if four div elements set as follows,

   <DIV id=”1”> <DIV id=”2”> </DIV>/</DIV>

   <DIV id=”3”> <DIV id=”4”> </DIV>/</DIV>

   [Without Z Index]
   IF we coded in in sequence of DIV1-DIV2 and DIV3- DIV4 without z-index, the browser chooses to make the DIV 4 in the most front following the order of occurrence.



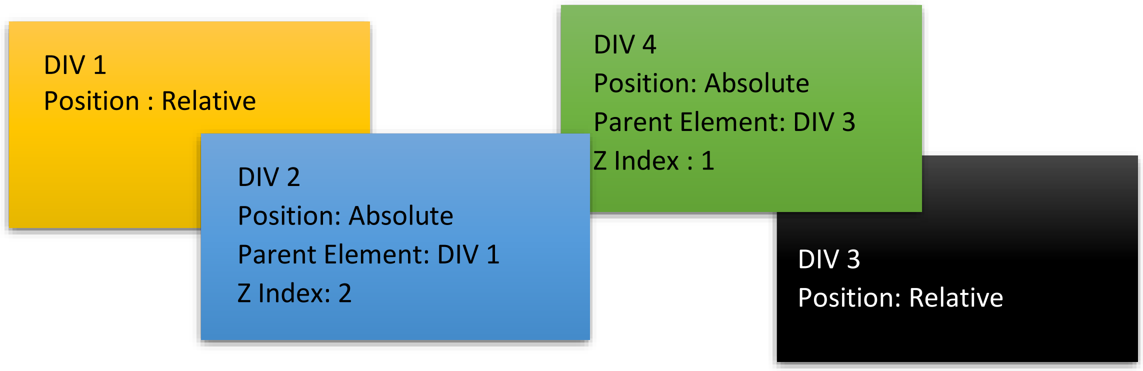
DIV 1
Position : Relative

DIV 2
Position: Absolute
Parent Element: DIV 1

DIV 4
Position: Absolute
Parent Element: DIV 3

DIV 3
Position: Relative

[With Z Index]
But if we want to make the arrangement as below, we have to introduce z-index which assigns them stacking order.



Even though DIV 4 and DIV 2 are not directly related, the stacking order can be arranged with the z index.

4. Can you give an example of when you would use absolute positioning?

   If we need to position an icon relative to its parent element positioning.

   In the CSS/HTML exercise my search icon inside the input element holds an absolute positioning relative to its parent; input element. Otherwise when the browser changes its size (Responsive), the icon will appear in inappropriate different positions relative to its view port.

   Child follows where ever parent goes ☺

5. How would you override an inline style?

   Prioritize the external/internal CSS style by making it's attribute as ***!important***. Therefore, browser will follow the external/internal style and override inline style.

Although browser gives priority to the inline CSS properties by default, if we override the property with ***!important*** attribute from a external/internal CSS style, the browser will follows external/internal CSS rule, unless the inline CSS is also set ***!important.***