EXERCISE : HTML AND CSS

1.How are inline and block elements different from each other?

a element is used as an inline element and a element as a block level element.

Basically, an inline element does not cause a line break (start on a new line) and does not take up the full width of a page, only the space bounded by its opening and closing tag. It is usually used within other HTML elements.

A block-level element always starts on a new line and takes up the full width of a page, from left to right. A block-level element can take up one line or multiple lines and has a line break before and after the element.

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

<div>Hello World</div>

<p>The DIV element is a block element, and will always start on a new
line and take up the full width available (stretches out to the left and
right as far as it can).</p>

</body>
</html>
```

Hello World

The DIV element is a block element, and will always start on a new line and take up the full width available (stretches out to the left and right as far as it can).

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

<p>This is an inline span <span >Hello World</span> element inside a
paragraph.</p>

<p>The SPAN element is an inline element, and will not start on a new
line and only takes up as much width as necessary.</p>

</body>
</html>
```

This is an inline span Hello World element inside a paragraph.

The SPAN element is an inline element, and will not start on a new line and only takes up as much width as necessary.

2.Explain the difference between visibility:hidden and display:none

display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.

The <script> element uses display: none; as default.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1.hidden {
  display: none;
}
</style>
</head>
<body>

<h1>This is a visible heading</h1>
<h1 class="hidden">This is a hidden heading</h1>
<p>Notice that the h1 element with display: none; does not take up any
space.</p>

</body>
</html>
```

This is a visible heading

Notice that the h1 element with display: none; does not take up any space.

visibility:hidden; also hides an element.

However, the element will still take up the same space as before. The element will be hidden, but still affect the layout:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1.hidden {
  visibility: hidden;
}
</style>
</head>
<body>

<h1>This is a visible heading</h1>
<h1 class="hidden">This is a hidden heading</h1>
<p>Notice that the hidden heading still takes up space.</p>

</body>
</html>
```

This is a visible heading

Notice that the hidden heading still takes up space.

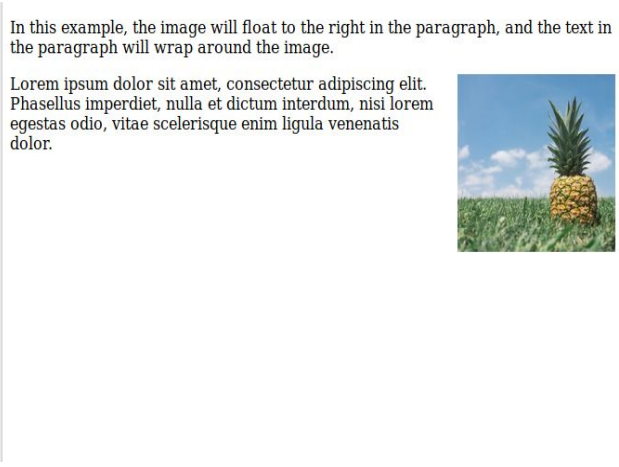
3. Explain the clear and float properties.

The float property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The float property can have one of the following values:

- left - The element floats to the left of its container
- right - The element floats to the right of its container
- none - The element does not float (will be displayed just where it occurs in the text). This is default
- inherit - The element inherits the float value of its parent

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

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

<p><img src="pineapple.jpg" alt="Pineapple" style="width:170px;
height:170px;margin-left:15px;">
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus
imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae
scelerisque enim ligula venenatis dolor. </p>

</body>
</html>
```



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

The clear property can have one of the following values:

- none - Allows floating elements on both sides. This is default
- left - No floating elements allowed on the left side
- right- No floating elements allowed on the right side
- both - No floating elements allowed on either the left or the right side
- inherit - The element inherits the clear value of its parent

```
.div4 {

  clear: left;
}
</style>
</head>
<body>

<h2>Without clear</h2>
<div class="div1">div1</div>
<div class="div2">div2 - Notice that div2 is after div1 in the HTML
code. However, since div1 floats to the left, the text in div2 flows
around div1.</div>
<br><br>

<h2>With clear</h2>
<div class="div3">div3</div>
<div class="div4">div4 - Here, clear: left; moves div4 down below the
floating div3. The value "left" clears elements floated to the left.
You can also clear "right" and "both".</div>

</body>
</html>
```

Without clear

div1
div2 - Notice that div2 is after div1 in the HTML code. However, since div1 floats to the left, the text in div2 flows around div1.

With clear

div3
div4 - Here, clear: left; moves div4 down below the floating div3. The value "left" clears elements floated to the left. You can also clear "right" and "both".

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.

```
<style>
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}

div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: absolute;</h2>

<p>An element with position: absolute; is positioned relative to the
nearest positioned ancestor (instead of positioned relative to the
```

position: absolute;

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

This div element has position: relative;

This div element has position: absolute;

4. explain difference between absolute, relative,fixed and static.

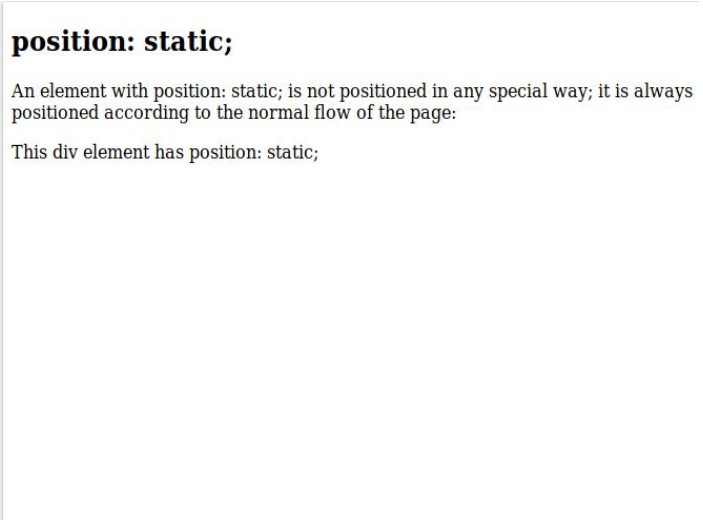
position: static;

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:

```
<!DOCTYPE html>
<html>
<head>
<style>
div.static {
  position: static;
}
</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>
```



position: static;

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

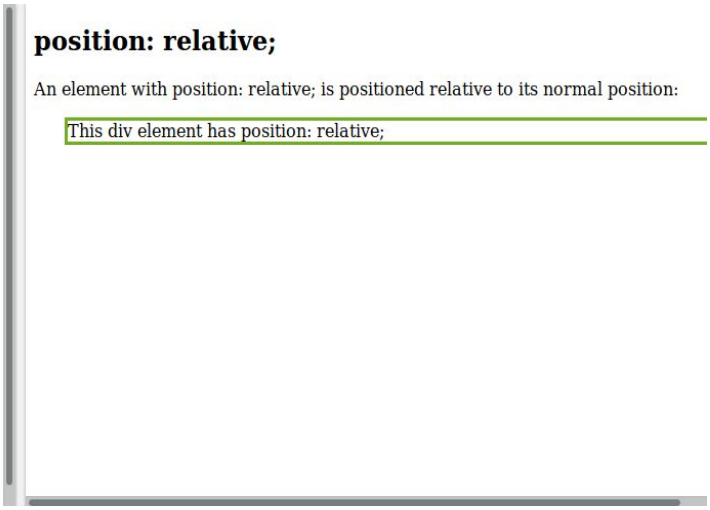
This div element has position: static;

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.

```html
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  left: 30px;

}
</style>
</head>
<body>

<h2>position: relative;</h2>

<p>An element with position: relative; is positioned relative to its
normal position:</p>

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

</body>
</html>
```

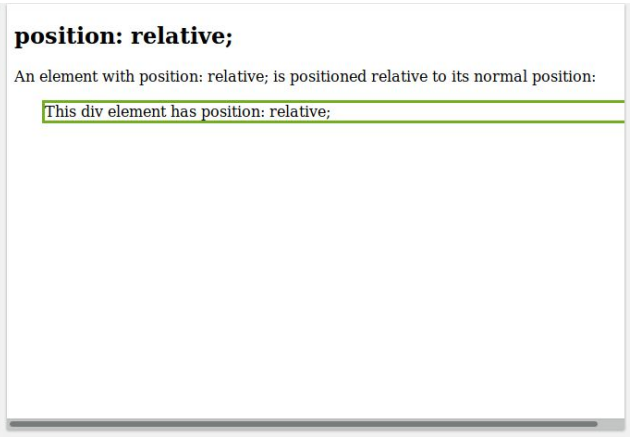


position: fixed;

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.

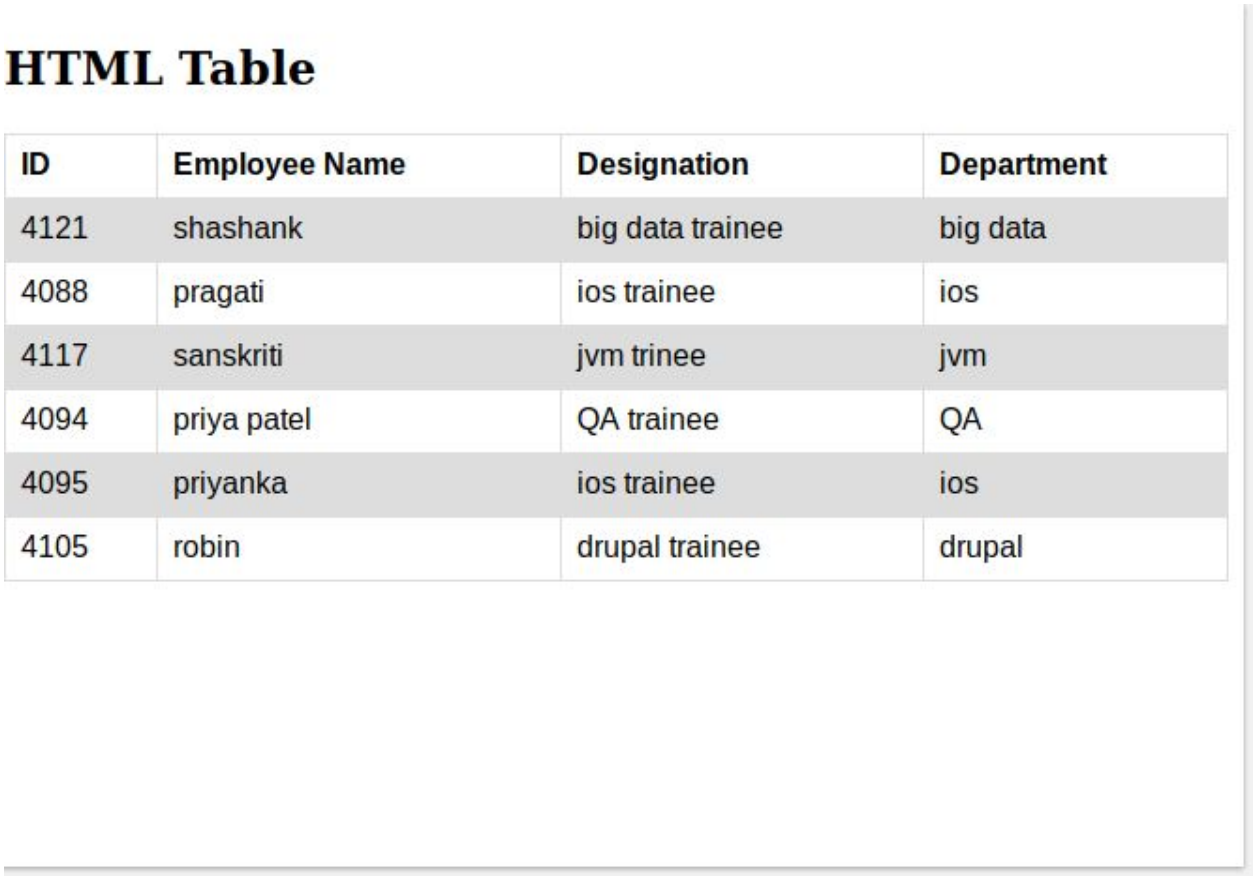
A fixed element does not leave a gap in the page where it would normally have been located.

```html
<html>
<head>
<style>
div.relative {
  position: relative;
  left: 30px;

}
</style>
</head>
<body>

<h2>position: relative;</h2>

<p>An element with position: relative; is positioned relative to its
normal position:</p>

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

</body>
</html>
```



5. Write the HTML code to create a table in which there are 4 columns(ID , Employee Name, Designation, Department) and at least 6 rows. Also do some styling to it.

Code is available in five.html file which is in git rep



HTML Table

ID	Employee Name	Designation	Department
4121	shashank	big data trainee	big data
4088	pragati	ios trainee	ios
4117	sanskriti	jvm trinee	jvm
4094	priya patel	QA trainee	QA
4095	priyanka	ios trainee	ios
4105	robin	drupal trainee	drupal

6. Why do we use meta tags?

Metadata is data (information) about data.

The <meta> tag provides metadata about the HTML document. Metadata will not be displayed on the page, but will be machine parsable.

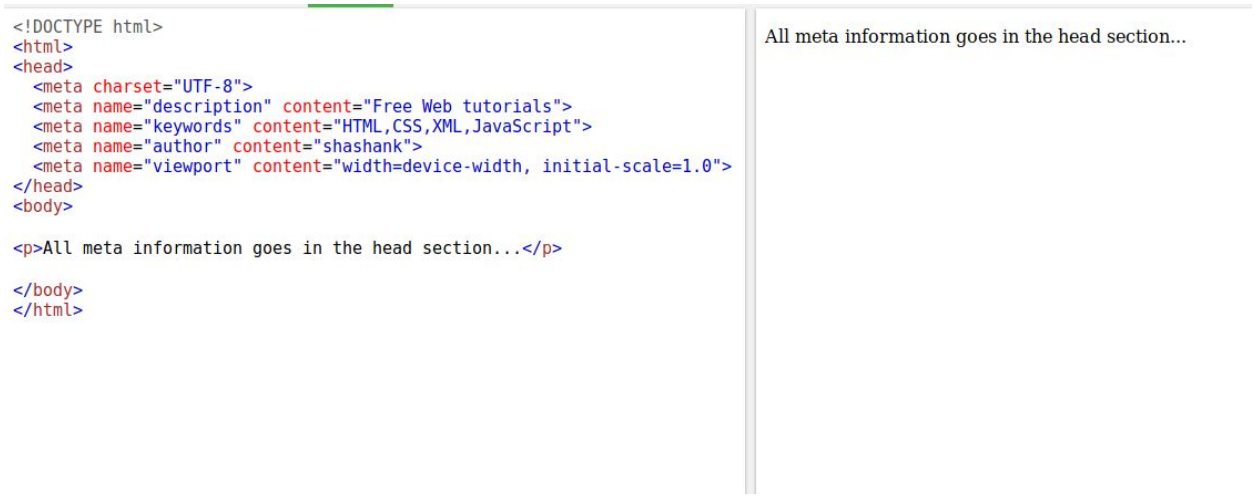
Meta elements are typically used to specify page description, keywords, author of the document, last modified, and other metadata.

The metadata can be used by browsers (how to display content or reload page), search engines (keywords), or other web services.

Note: <meta> tags always go inside the <head> element.

Note: Metadata is always passed as name/value pairs.

Note: The content attribute MUST be defined if the name or the http-equiv attribute is defined. If none of these are defined, the content attribute CANNOT be defined.



7. Explain box model.

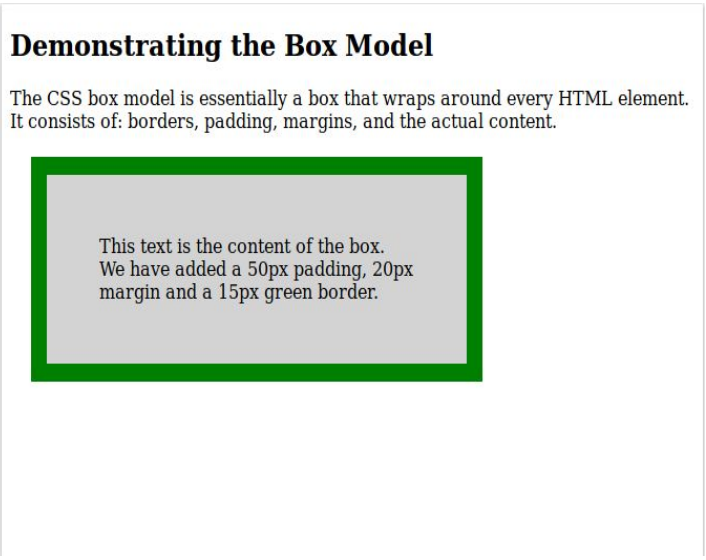
All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

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:

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

```html
<!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. </div>

</body>
```



8. What are the different types of CSS Selectors?

CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

```html
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<p>Every paragraph will be affected by the style.</p>
<p id="paral">Me too!</p>
<p>And me!</p>

</body>
</html>
```

Every paragraph will be affected by the style.

Me too!

And me!

9. Define Doctype.

The <!DOCTYPE> declaration must be the very first thing in your HTML document, before the <html> tag.

The <!DOCTYPE> declaration is not an HTML tag; it is an instruction to the web browser about what version of HTML the page is written in.

In HTML 4.01, the <!DOCTYPE> declaration refers to a DTD, because HTML 4.01 was based on SGML. The DTD specifies the rules for the markup language, so that the browsers render the content correctly.

HTML5 is not based on SGML, and therefore does not require a reference to a DTD.

```html
<!DOCTYPE html>
<html>
<head>
<title>Title of the document</title>
</head>

<body>
The content of the document......
</body>

</html>
```

The content of the document.....

10. Explain 5 HTML5 semantic tags.

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of non-semantic elements: and - Tells nothing about its content.

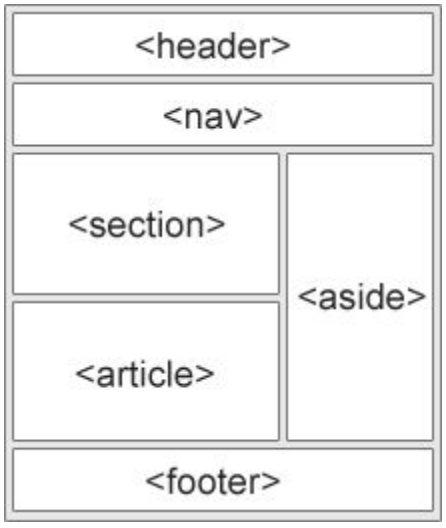
Examples of semantic elements: <form>, <table>, and <article> - Clearly defines its content.

Many web sites contain HTML code like:

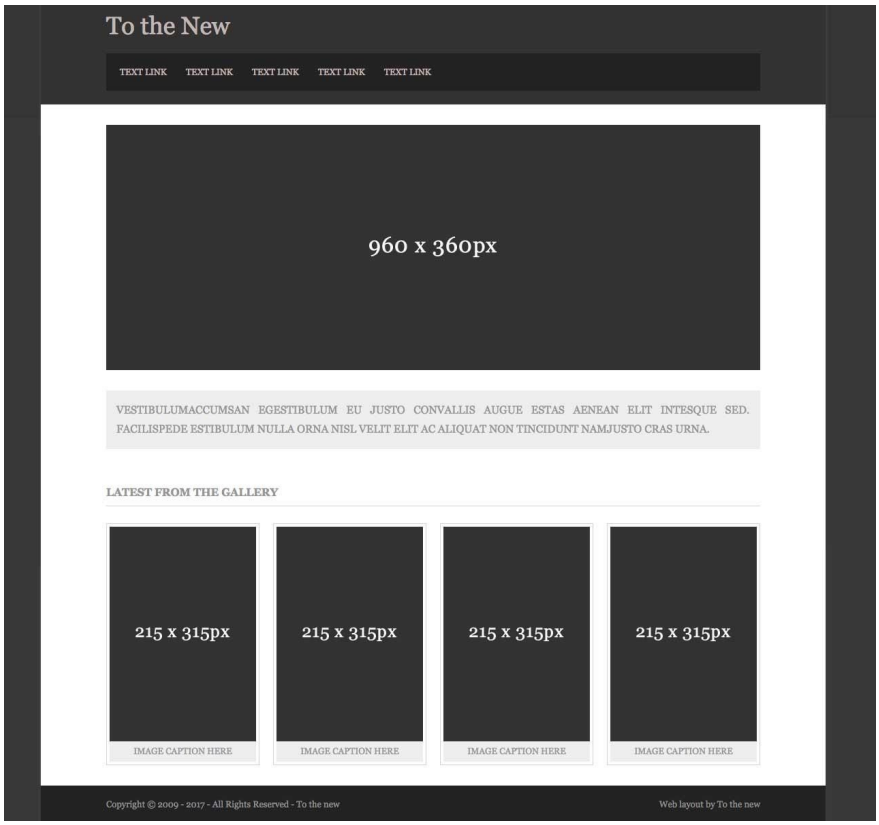
to indicate navigation, header, and footer.

HTML5 offers new semantic elements to define different parts of a web page:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



11. Create HTML for web-page.jpg (check resources, highest weightage for answers)



12. Create HTML for form.png (check resources, highest weightage for answers)

Code is at repository as form.html and form.css