### **Introduction to HTML/CSS**

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

**Solution:** Every HTML element has a default display value depending on what type of element it is.

The two display values are: block and inline.

**Block-level Elements:** A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The <div> element is a block-level element.

<div>Hello World</div>

**Inline Elements:** An inline element does not start on a new line and only takes up as much width as necessary. This is an inline <span> element inside a paragraph.

<span>Hello World</span>

**The <div> Element**

The <div> element is often used as a container for other HTML elements.

The <div> element has no required attributes, but style, class and id are common.

When used together with CSS, the <div> element can be used to style blocks of content

**The <span> Element**

The <span> element is often used as a container for some text.

The <span> element has no required attributes, but style, class and id are common.

When used together with CSS, the <span> element can be used to style parts of the text

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

## **Solution: Display: none**

display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them. display:none means that the tag in question will not appear on the page at all. There will be no space allocated for it between the other tags.

Hiding an element can be done by setting the display property to none. The element will be hidden, and the page will be displayed as if the element is not there:

### **Example**

h1.hidden {

display: none;

}

**visibility:hidden;**

It 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:

### Example

h1.hidden {

visibility: hidden;

}

**3. Explain the clear and float properties.**

## **Solution:**

The CSS float property specifies how an element should float.

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

## **The float Property**

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

## **The clear Property**

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

The most common way to use the clear property is after you have used a float property on an element.

When clearing floats, you should match the clear to the float: If an element is floated to the left, then you should clear to the left. Your floated element will continue to float, but the cleared element will appear below it on the web page.

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

## **Solution:**

## The Position Property: The position property specifies the type of positioning method used for an element.

There are four different position values:

* static
* relative
* fixed
* absolute

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

## **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;

Here is the CSS that is used:

### **Example**

.staticexp {

position: static;

border: 3px solid #73AD21;

}

## **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:

### **Example**

.relativeexp {

position: relative;

left: 30px;

border: 3px solid #73AD21;

}

## **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:

### **Example**

.fixedexp {

position: fixed;

bottom: 0;

right: 0;

width: 300px;

border: 3px solid #73AD21;

}

This <div> element has position: fixed;

## **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 static.

Here is a simple example:

This <div> element has position: relative;

This <div> element has position: absolute;

Here is the CSS that is used:

### **Example**

.relative {

position: relative;

width: 400px;

height: 200px;

border: 3px solid #73AD21;

}

.absolute {

position: absolute;

top: 80px;

right: 0;

width: 200px;

height: 100px;

border: 3px solid #73AD21;

}

**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.**

**Table.html**

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport"

content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

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

<title>TABLE</title>

</head>

<body>

<div style="overflow-x:auto">

<table style="width:100%">

<tr>

<th>ID</th>

<th>Employee Name</th>

<th>Designation</th>

<th>Department</th>

</tr>

<tr>

<td>1001</td>

<td>Smith</td>

<td>Accounting</td>

<td>Finance</td>

</tr>

<tr>

<td>1002</td>

<td>Jackson</td>

<td>Engineering</td>

<td>IT</td>

</tr>

<tr>

<td>1003</td>

<td>Jack</td>

<td>Sales</td>

<td>Marketing</td>

</tr>

<tr>

<td>1004</td>

<td>Rocky</td>

<td>Engineering</td>

<td>IT</td>

</tr>

<tr>

<td>1005</td>

<td>Watson</td>

<td>Sales</td>

<td>Marketing</td>

</tr>

<tr>

<td>1006</td>

<td>Michelle</td>

<td>Accounting</td>

<td>Finance</td>

</tr>

</table>

</div>

</body>

</html>

**Tablestyle.css**

table {

border-collapse: collapse; /\*The border-collapse property sets whether the table borders should be collapsed into a single border: \*/

width: 100%;

}

table, th, td {

border-bottom: 1px solid #ddd;

height: 50px;

text-align: left;

padding: 15px;

}

tr:hover {

background-color: #f5f5f5;

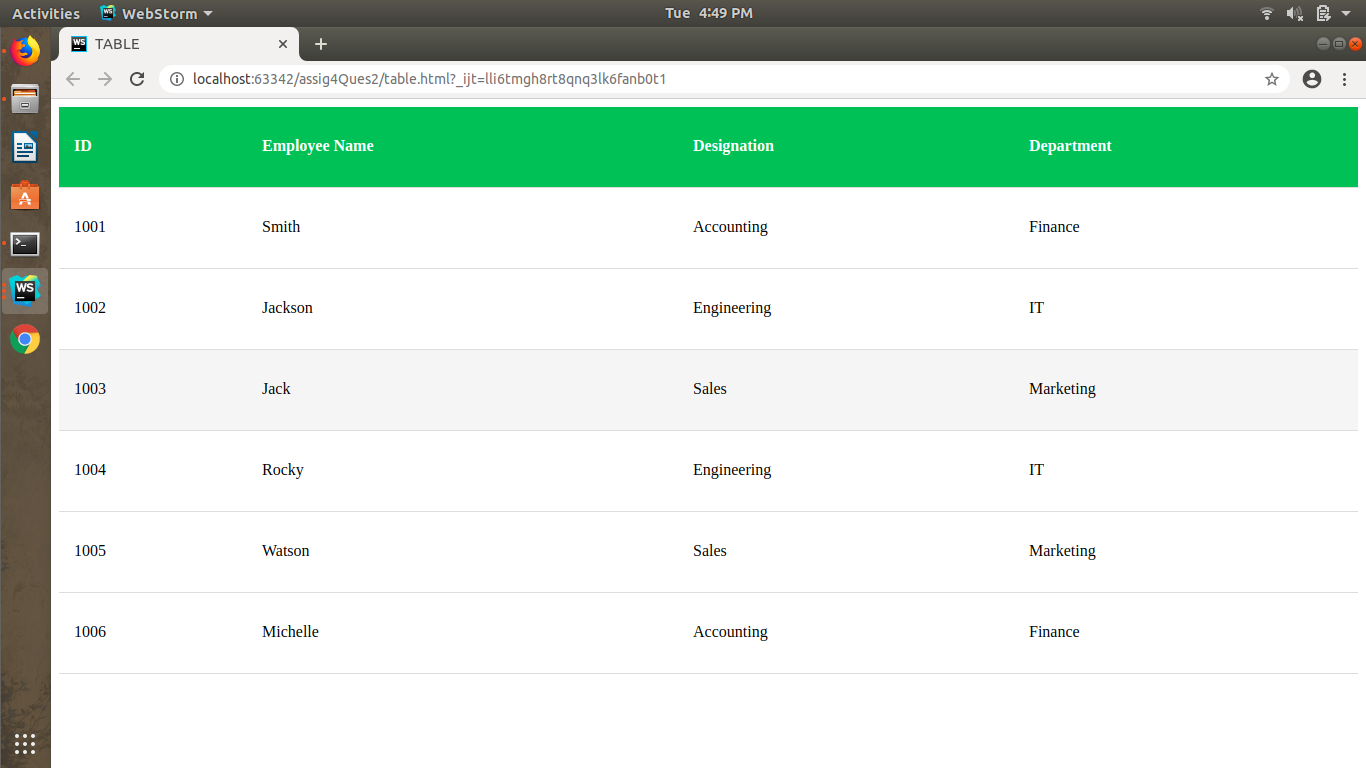
}

th {

background-color: #4CAF50;

color: white;

}

****

**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.

### **Example**

**Describe metadata within an HTML document:**

<head>

<meta charset="UTF-8">

<meta name="description" content="Free Web tutorials">

<meta name="keywords" content="HTML,CSS,XML,JavaScript">

<meta name="author" content="John Doe">

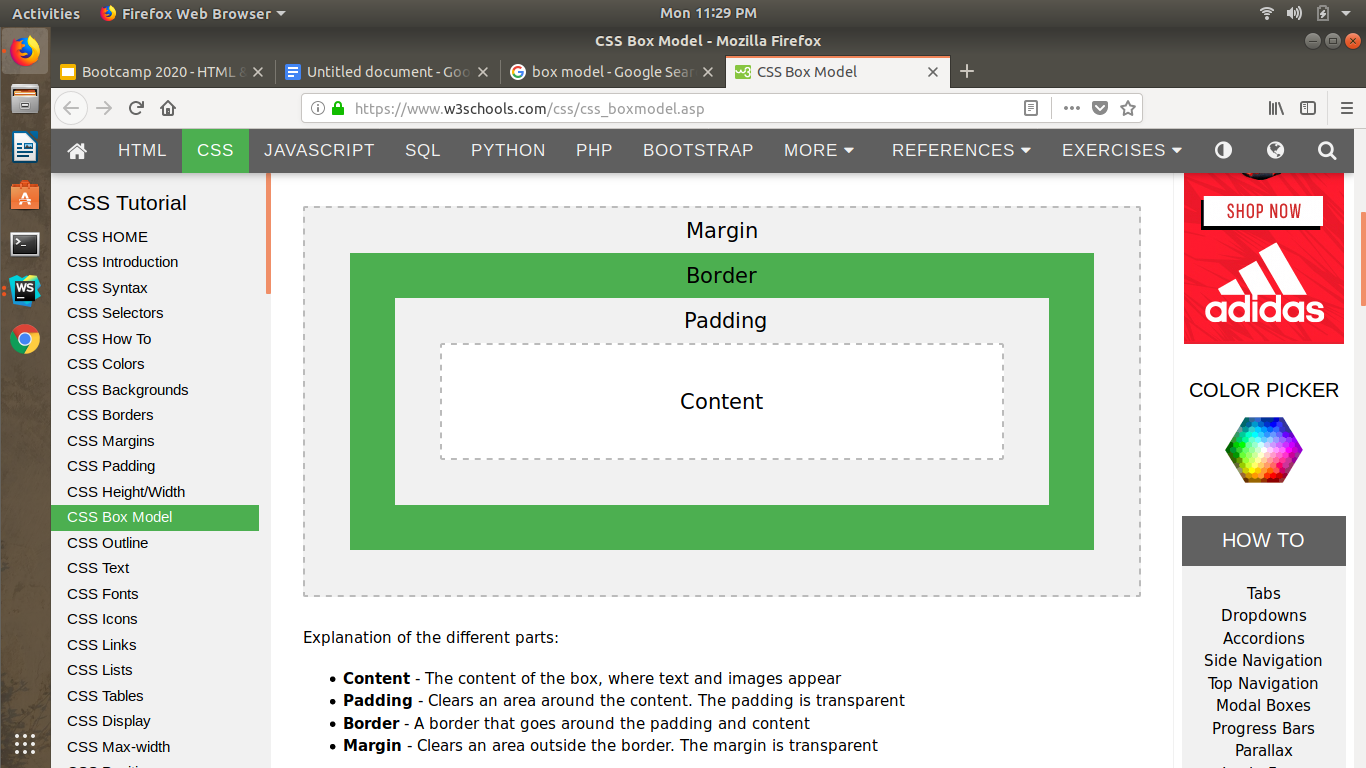
<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

**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

**Example**

Demonstration of the box model:

div {

width: 300px;

border: 15px solid green;

padding: 50px;

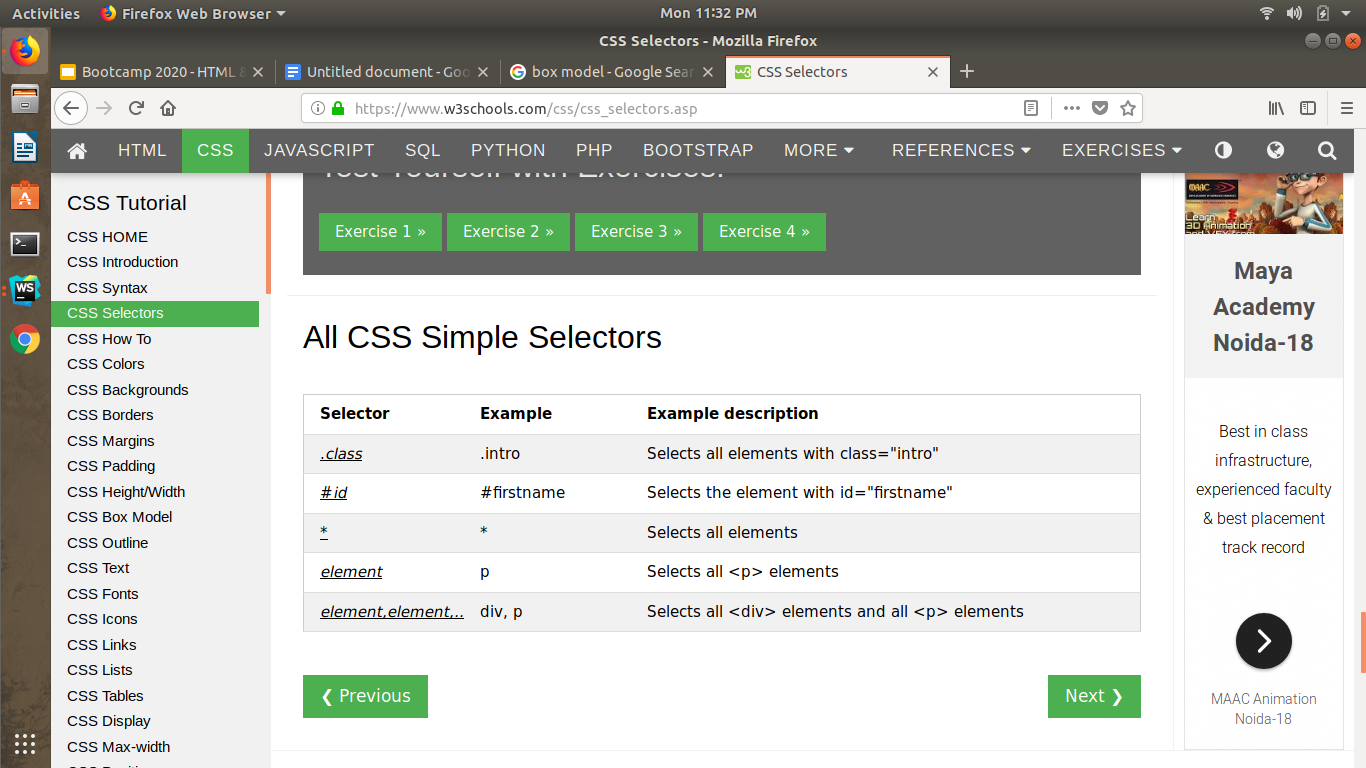
margin: 20px;

}

**8. What are the different types of CSS Selectors?**

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

All CSS Simple Selectors



1. **The CSS element Selector**

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

**Example**

Here, all <p> elements on the page will be center-aligned, with a red text color:

p {

text-align: center;

color: red;

}

1. **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.

**Example**

The CSS rule below will be applied to the HTML element with id="para1":

#para1 {

text-align: center;

color: red;

}

Note: An id name cannot start with a number!

1. **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.

**Example**

In this example all HTML elements with class="center" will be red and center-aligned:

.center {

text-align: center;

color: red;

}

You can also specify that only specific HTML elements should be affected by a class.

Example

In this example only <p> elements with class="center" will be center-aligned:

p.center {

text-align: center;

color: red;

}

1. **The CSS Universal Selector**

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

**Example**

The CSS rule below will affect every HTML element on the page:

\* {

text-align: center;

color: blue;

}

1. **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 {

text-align: center;

color: red;

}

h2 {

text-align: center;

color: red;

}

p {

text-align: center;

color: red;

}

It will be better to group the selectors, to minimize the code.

To group selectors, separate each selector with a comma.

**Example**

In this example we have grouped the selectors from the code above:

h1, h2, p {

text-align: center;

color: red;

}

**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.

Tip: Always add the <!DOCTYPE> declaration to your HTML documents, so that the browser knows what type of document to expect.

**Example**

<!DOCTYPE html>

<html>

<head>

<title>Title of the document</title>

</head>

<body>

The content of the document......

</body>

</html>

**10. Explain 5 HTML5 semantic tags.**

Semantics is the study of the meanings of words and phrases in a language.

Semantic elements = elements with a meaning.

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

Examples of non-semantic elements: <div> and <span> - Tells nothing about its content.

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

**New Semantic Elements in HTML5**

Many web sites contain HTML code like:

<div id="nav"> <div class="header"> <div id="footer"> to indicate navigation, header, and footer.

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

1. **HTML5 <section> Element**

The <section> element defines a section in a document.

According to W3C's HTML5 documentation: "A section is a thematic grouping of content, typically with a heading."

A home page could normally be split into sections for introduction, content, and contact information.

**Example**

<section>

<h1>WWF</h1>

<p>The World Wide Fund for Nature (WWF) is....</p>

</section>

1. **HTML5 <article> Element**

The <article> element specifies independent, self-contained content.

An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.

Examples of where an <article> element can be used:

* Forum post
* Blog post
* Newspaper article

**Example**

<article>

<h1>What Does WWF Do?</h1>

<p>WWF's mission is to stop the degradation of our planet's natural environment,

and build a future in which humans live in harmony with nature.</p>

</article>

1. **HTML5 <header> Element**

The <header> element specifies a header for a document or section.

The <header> element should be used as a container for introductory content.

You can have several <header> elements in one document.

The following example defines a header for an article:

**Example**

<article>

<header>

<h1>What Does WWF Do?</h1>

<p>WWF's mission:</p>

</header>

<p>WWF's mission is to stop the degradation of our planet's natural environment,

and build a future in which humans live in harmony with nature.</p>

</article>

1. **HTML5 <footer> Element**

The <footer> element specifies a footer for a document or section.

A <footer> element should contain information about its containing element.

A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

You may have several <footer> elements in one document.

**Example**

<footer>

<p>Posted by: Hege Refsnes</p>

<p>Contact information: <a href="mailto:someone@example.com">

someone@example.com</a>.</p>

</footer>

1. **HTML5 <nav> Element**

The <nav> element defines a set of navigation links.

Notice that NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major block of navigation links.

**Example**

<nav>

<a href="/html/">HTML</a> |

<a href="/css/">CSS</a> |

<a href="/js/">JavaScript</a> |

<a href="/jquery/">jQuery</a>

</nav>

1. **HTML5 <aside> Element**

The <aside> element defines some content aside from the content it is placed in (like a sidebar).

The <aside> content should be related to the surrounding content.

**Example**

<p>My family and I visited The Epcot center this summer.</p>

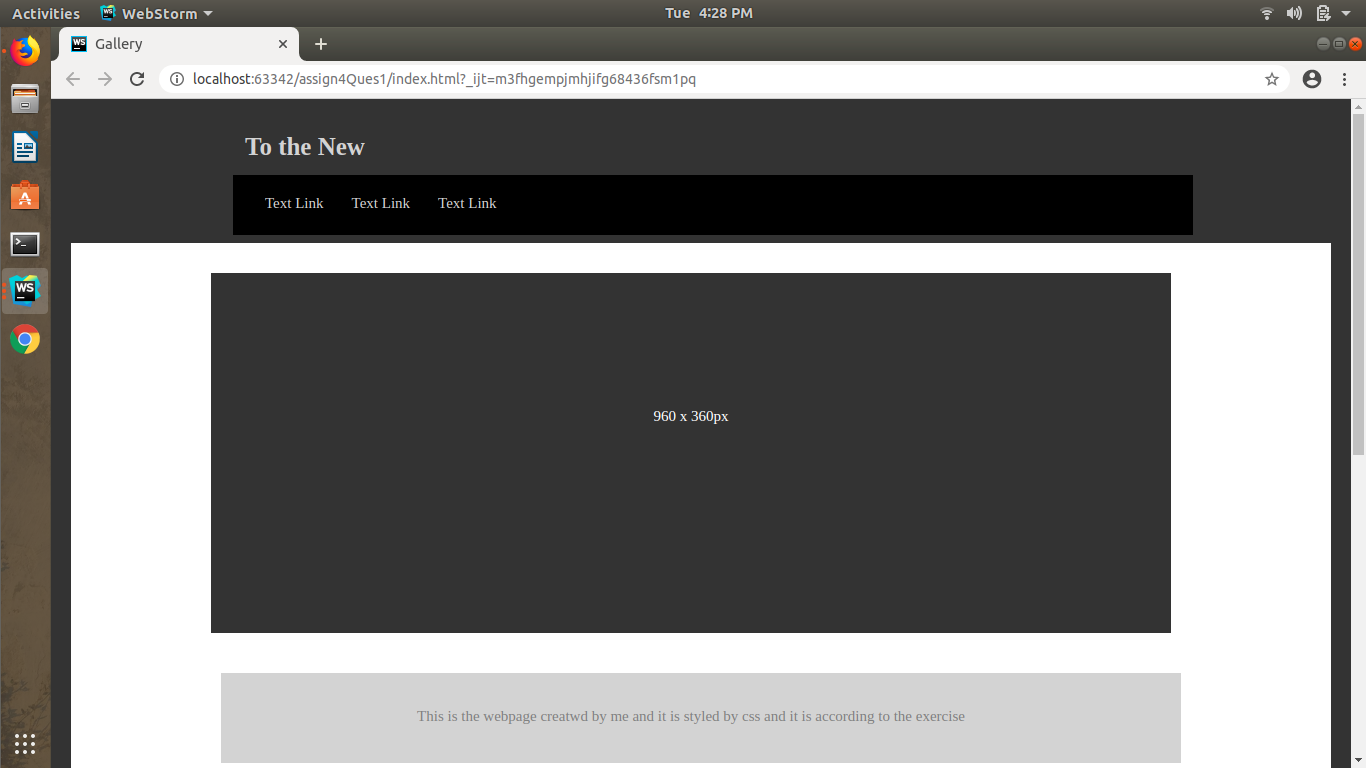
<aside>

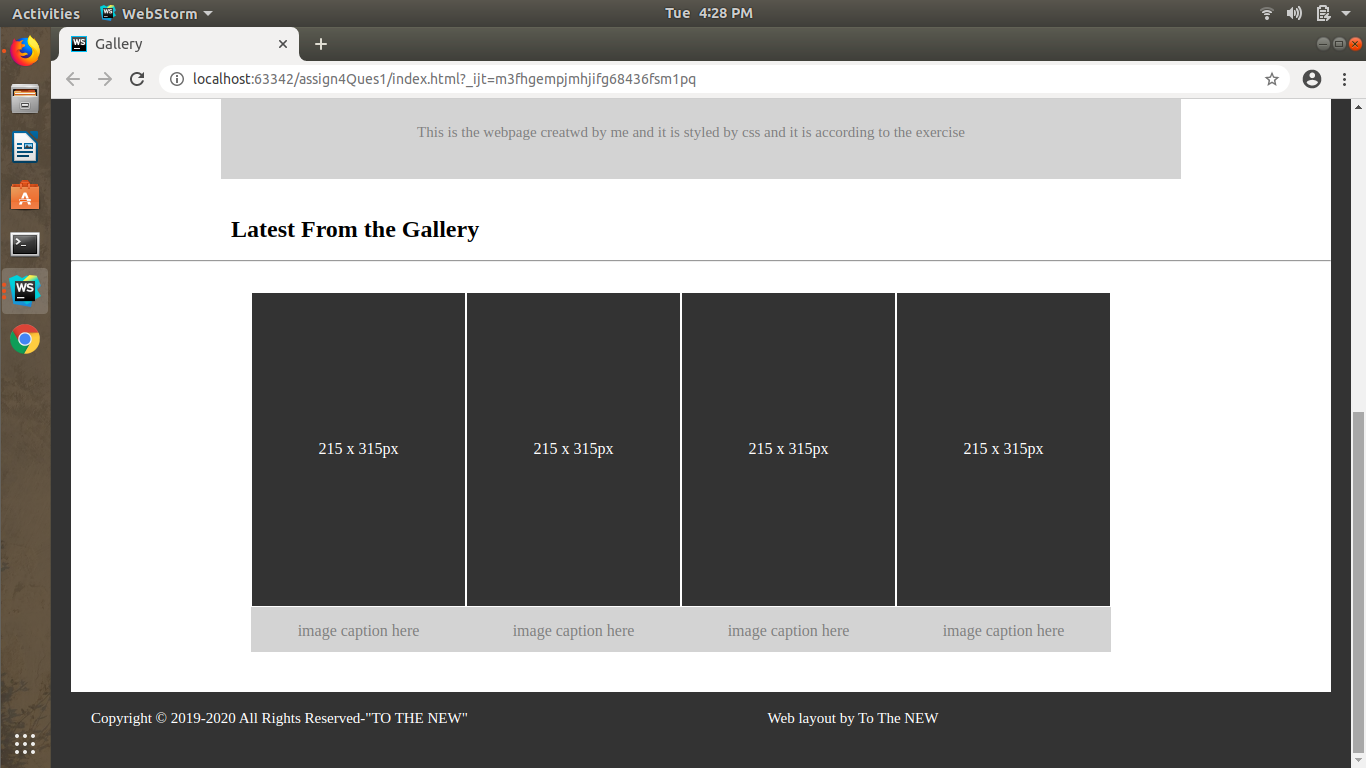
<h4>Epcot Center</h4>

<p>The Epcot Center is a theme park in Disney World, Florida.</p>

</aside>

**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)**

