Introduction to HTML & CSS - Exercise Solutions
February 08, 2019

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

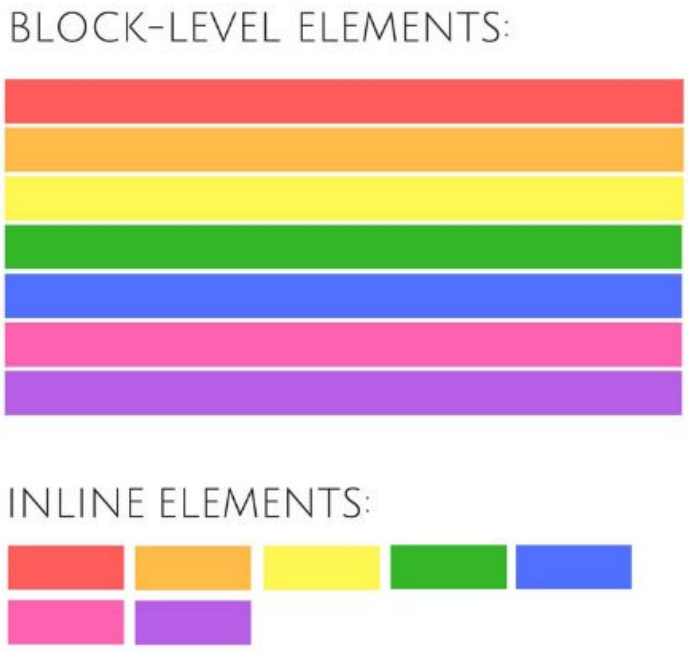
The display property specifies the display behavior (the type of rendering box) of an element.

Inline: span, button, input, label, select, etc

- An inline element has no line break before or after it, and it tolerates HTML elements next to it
- It respects left & right margins and padding, but not top & bottom
- It cannot have a width and height set
- It allows other elements to sit to their left and right

Block: h1 to h6, p, div, table, etc

- A block element has some whitespace above and below it and does not tolerate any HTML elements next to it
- It respects top, bottom, left, and right margins and padding
- It forces a line break after the block element
- It acquires full-width if width not defined



Inline-block:

- An inline-block element is placed as an inline element (on the same line as adjacent content), but it behaves as a block element
- It allows other elements to sit to their left and right
- It respects top & bottom margins and padding
- It respects height and width

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

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

Question 3:
Explain the clear and float properties

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 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

In its simplest use, the float property can be used to wrap text around images.

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 we have used a float property on an element.

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

Question 4:
Explain difference between absolute, relative, fixed, and static

The position property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).
There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

Elements are then positioned using the top, bottom, left, and right properties.

`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:

`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.

`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.

`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.

Question 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: https://github.com/Deee92/bootcamp_assignments/tree/master/html_question5

Page:

Question 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.
<meta> tags always go inside the <head> element.
Metadata is always passed as name/value pairs.

Example 1 - Define keywords for search engines:

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

Example 2 - Define a description of your web page:

```
<meta name="description" content="Free Web tutorials on HTML and CSS">
```

Example 3 - Define the author of a page:

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

Example 4 - Refresh document every 30 seconds:

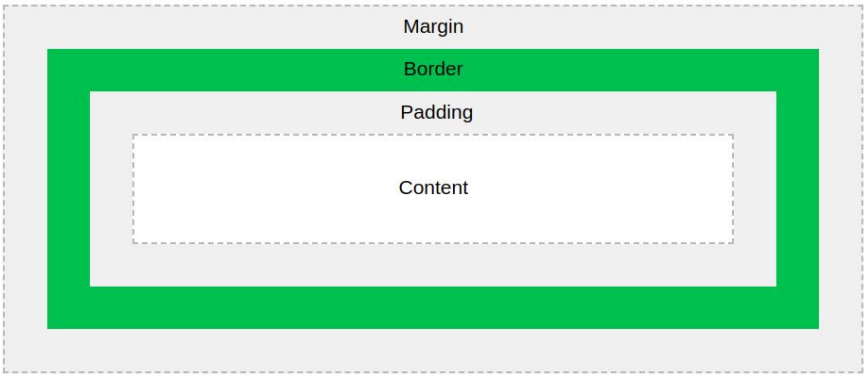
```
<meta http-equiv="refresh" content="30">
```

Example 5 - Setting the viewport to make your website look good on all devices:
HTML5 introduced a method to let web designers take control over the viewport (the user's visible area of a web page), through the <meta> tag.
The viewport is the user's visible area of a web page. It varies with the device, and will be smaller on a mobile phone than on a computer screen.
You should include the following <meta> viewport element in all your web pages:
<meta name="viewport" content="width=device-width, initial-scale=1.0">
A <meta> viewport element gives the browser instructions on how to control the page's dimensions and scaling.
The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

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

Question 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:

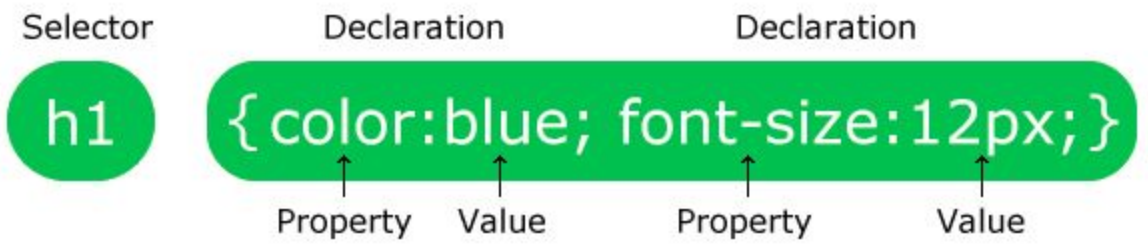
- Content - The content of the box, where text and images appear
- Padding - Clears an area around the content. The padding is transparent
- Border - A border that goes around the padding and content
- Margin - Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements.

Question 8:
What are the different types of CSS Selectors?

A CSS rule-set consists of a selector and a declaration block.

CSS selector
The selector points to the HTML element you want to style.



CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

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

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

The id Selector
The id selector uses the id attribute of an HTML element to select a specific element.
The id of an element should be 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 class Selector
The class selector selects elements with a specific class attribute.
To select elements with a specific class, write a period (.) character, followed by the name of the class.

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

We can also group the selectors, to minimize the code. To group selectors, separate each selector with a comma.

Pseudo-classes selectors
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

```
selector:pseudo-class {
  property:value;
}
```

(https://www.w3schools.com/css/css_pseudo_classes.asp for a complete list)

Question 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.

It is advisable to always add the <!DOCTYPE> declaration to your HTML documents, so that the browser knows what type of document to expect.

`<!DOCTYPE html>` or `<!doctype html>`

Question 10:
Explain 5 HTML5 semantic tags

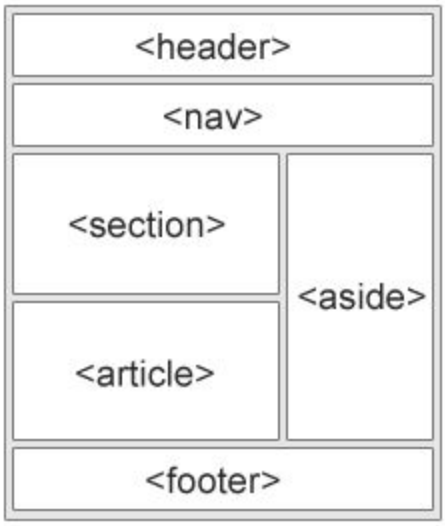
Semantics is the study of the meanings of words and phrases in a language.
Semantic elements = elements with a meaning.

A semantic element clearly describes its meaning to both the browser and the developer. With HTML4, developers used their own id/class names to style elements: header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav, etc.
This made it impossible for search engines to identify the correct web page content.
With the new HTML5 elements (<header> <footer> <nav> <section> <article>), this will become easier.
According to the W3C, a Semantic Web: "Allows data to be shared and reused across applications, enterprises, and communities."

Examples of non-semantic elements: and - Tells nothing about its content.
Examples of semantic elements: <form>, <table>, and <article> - Clearly defines its content.

HTML5 offers new semantic elements, which are supported in all modern browsers, to define different parts of a web page:

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



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

```
<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>
```

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

```
<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>
```

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.

```
<footer>
  <p>Posted by: Hege Refsnes</p>
  <p>Contact information: <a href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>
```

HTML5 <nav> Element

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

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

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

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.

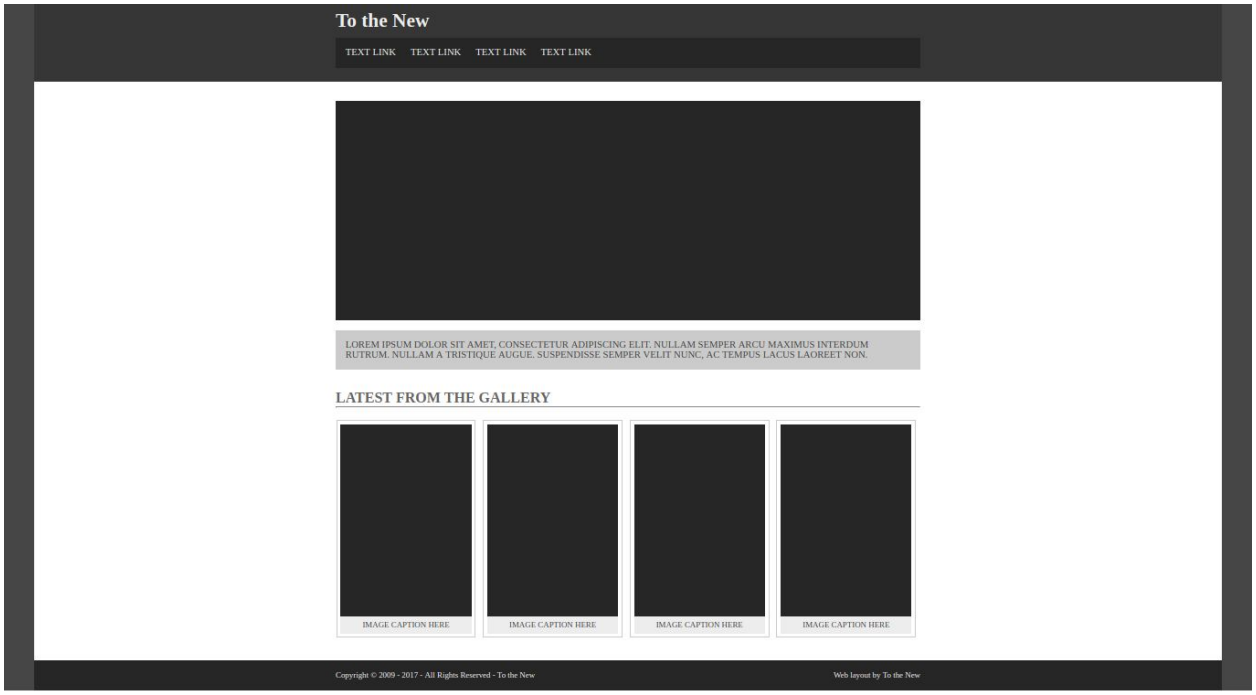
```
<p>My family and I visited The Epcot center this summer.</p>

<aside>
  <h4>Epcot Center</h4>
  <p>The Epcot Center is a theme park in Disney World, Florida.</p>
</aside>
```

Question 11:
Create HTML for web-page.jpg

Code: https://github.com/Deee92/bootcamp_assignments/tree/master/html_exercise_one

Page:



Question 12:
Create HTML for form.png

Code: https://github.com/Deee92/bootcamp_assignments/tree/master/html_exercise_two

Page: