COURSEWARE

Professional Skills

Agile Fundamentals

Jira

Git

Databases Introduction

Java Beginner

Maven

Testing (Foundation)

Java Intermediate

HTML

CSS

- Introduction to CSS3
- Fonts
- Transform
- Selectors
- Selectors - Pseudo-Class / Pseudo-Elements
- Spatial Attributes
- Alignment
- Overflow
- Display and Positioning
- Background and Borders
- Bootstrap Introduction
- Bootstrap Nav
- Bootstrap Collapse
- Bootstrap Grid
- Bootstrap Forms & Inputs
- Bootstrap Modal
- Bootstrap Cards

Javascript

Spring Boot

Selenium

Sonarqube

Advanced Testing (Theory)

Display and Positioning

Contents

- Overview
  - Display Property
    - Block-level Elements
    - Inline Elements
    - Overriding Default Display Value
    - Hide an element
  - Position Property
    - Static
    - Relative
    - Fixed
    - Absolute
    - Sticky
  - Overlapping Elements
    - Positioning Text In an Image
  - All CSS Positioning Properties
- Tutorial
- Exercises

Overview

In this module, we will be discussing the `display` and `position` property within CSS for controlling layout.

Display Property

The `display` property is the most important CSS property for controlling layout. It specifies if/how a element is displayed.

Every HTML element has a default display value depending on what type of element it is.
The default display value for most elements is `block` or `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).

Examples of block-level elements:

- `<div>`
- `<h1>` - `<h6>`
- `<p>`
- `<form>`
- `<header>`
- `<footer>`
- `<section>`

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

```
<p> This is an <span> inline span element inside </span> a paragraph </p>
```

Examples of inline elements include:

- `<span>`

Cucumber
MongoDB
Express
NodeJS
React
Express-Testing
Networking
Security
Cloud Fundamentals
AWS Foundations
AWS Intermediate
Linux
DevOps
Jenkins Introduction
Jenkins Pipeline
Markdown
IDE Cheatsheet

- `<a>`
- `<img>`

Overriding Default Display Value

As mentioned previously, every element has a default display value. However, we can override this.

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.

A common example is making inline `<li>` elements for horizontal menus:

```
<!DOCTYPE html>
<html>
<head>
    <style>
        li {
            display: inline;
        }
    </style>
</head>
<body>
    <p> Display a list of links as a horizontal menu: </p>
    <ul>
        <li><a href="www.qa.com" target="_blank">QA </a></li>
        <li><a href="www.google.com" target="_blank">Google</a></li>
        <li><a href="https://www.w3.org/TR/css-display-3/">W3 Display
Reference</a></li>
    </ul>
</body>
</html>
```

*Note: Setting the display property of an element only changes **how the element is displayed**, NOT what kind of element it is. So, an inline element with `display:block` is not allowed to have other block elements inside it.*

Hide an element

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:

```
<!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>
```

Another approach we could take is using `visibility:hidden;` - this 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>
```

Position Property

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:

1. `static`
2. `relative`
3. `fixed`
4. `absolute`
5. `sticky`

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.

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.

```
div.static{
    position:static;
    border: 3px solid #f9f9f9;
}
```

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.

```
div.relative {
    position:relative;
    left: 30px;
    border: 3px solid #ff0000;
}
```

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 be located.

```
div.fixed {
    position: fixed;
    bottom: 0;
    right: 0;
    width: 300px;
    border: 3px solid #00FFFF;
}
```

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 of whose position is anything except `static`.

```
<!DOCTYPE html>
<html>
<head>
    <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 viewport, like fixed):</p>

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

</body>
</html>
```

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`).

Note: You must also specify at least one of `top`,`right`,`bottom` or `left` for sticky positioning to work.

```css
div.sticky {
    position: sticky;
    top: 0;
    background-color: purple;
    border: 2px solid #f9f9f9;
}
```

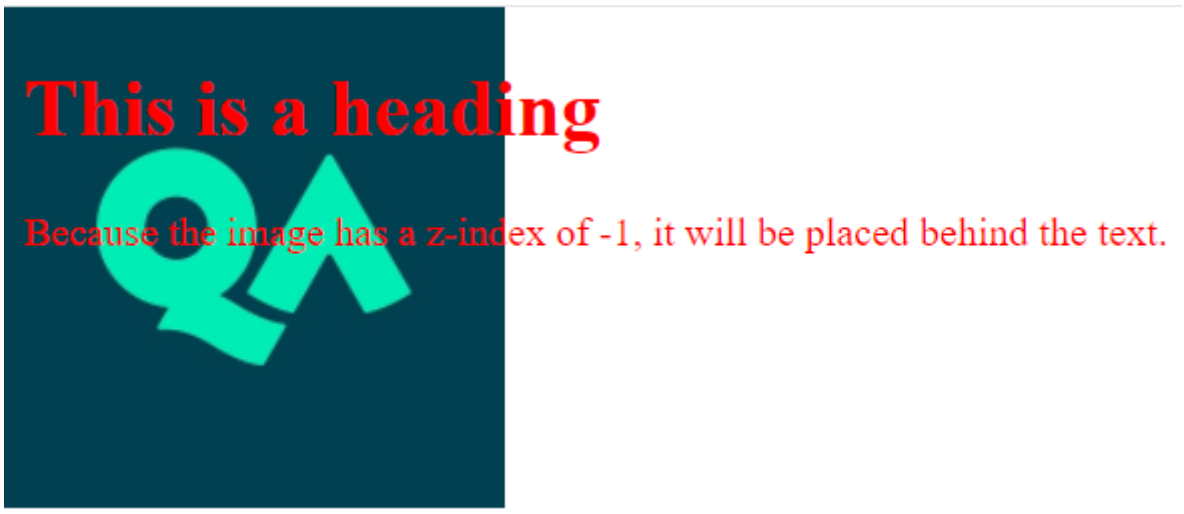
Overlapping Elements

When elements are positioned, they can overlap other elements.

The `z-index` property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

An element can have a position or negative stack order:

```css
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
  width:200px;
  height:200px;
}
```



An element with greater stack order is always in front of an element with a lower stack order.

Note: If two positioned elements overlap without a z-index specified, the element positioned last in the HTML ode will be shown on top.

Positioning Text In an Image

To position text in an image, we place our image inside a container, as well as our text and position it on top of each other.

```
<!DOCTYPE html>
<html>
<head>
    <style>
        .container {
        position: relative;
        }

        .center {
        position: absolute;
        top: 50%;
        width: 100%;
        text-align: center;
        font-size: 18px;
        }

        img {
        width: 100%;
        height: auto;
        opacity: 0.3;
        }
    </style>
</head>
<body>

    <h2>Image Text</h2>
    <p>Center text in image:</p>

    <div class="container">
        <img src="https://i.imgur.com/jdUj4kh.jpg" alt="South of France"
width="1000" height="300">
        <div class="center">Centered</div>
    </div>

</body>
</html>
```

All CSS Positioning Properties

Property	Description
`bottom`	Sets the bottom margin edge for a positioned box
`clip`	Clips an absolutely positioned element
`left`	Sets the left margin edge for a positioned box
`position`	Specifies the type of positioning for an element
`right`	Sets the right margin edge for a positioned box
`top`	Sets the top margin edge for a positioned box
`z-index`	Sets the stack order of an element

Tutorial

1. Create the following HTML document

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</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>

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

    <p>The overflow property specifies what to do if the
    content of an element exceeds the size of the element's box.</p>

    <p>overflow:scroll</p>
    <div class="scroll">You can use the overflow property when you want to
have better control of the layout. The default value is visible.</div>

    <p>overflow:hidden</p>
    <div class="hidden">You can use the overflow property when you want to
have better control of the layout. The default value is visible.</div>

    <div class="ex1">This div element has width: 500px;</div>
    <br>

    <div class="ex2">This div element has max-width: 500px;</div>

    <p class="stay">
        <strong>Tip:</strong> Drag the browser window to smaller than
500px wide, to see the difference between
        the two divs!
    </p>
</body>
</html>
```

2. Set the heading with `class=hidden` to be hidden using the `display` property

```
<style>
    ...
    h1.hidden{
        display: none;
    }
</style>
```

3. Set the heading with the `class=vis` to be hidden using the `visibility` property.

```
<style>
    ...
    h1.vis{
         visibility: hidden;
     }
</style>
```

> Observe the difference between the last two steps.

4. Set the background-colour to sky blue, and overflow to scroll for the `<div>` element with class `scroll`

```
<style>
     ...
     div.scroll {
          background-color: #00FFFF;
          width: 100px;
          height: 100px;
          overflow: scroll;
     }
 </style>
```

5. Set the background-colour to sky green, and overflow to hidden for the `<div>` element with class `hidden`

```
<style>
     ...
     div.hidden {
          background-color: #00FF00;
          width: 100px;
          height: 100px;
          overflow: hidden;
     }
 </style>
```

6. Set the width of the `<div>` with class `ex1` to 500px

```
<style>
     ...
     div.ex1 {
          width: 500px;
          margin: auto;
          border: 3px solid #8AC007;
     }
 </style>
```

7. Set the max-width of `<div>` with class `ex2` to 500px

```
<style>
     ...
     div.ex2 {
          max-width: 500px;
          margin: auto;
          border: 3px solid #8AC007;
     }
 </style>
```

> Observe the difference between the last two steps.

8. Set the last paragraph with the class `stay` to `fixed` using the property `position`

```
<style>
     ...
     p.stay {
          position:fixed;
          bottom: 0;
          background-color: purple;
          border: 2px solid #f9f9f9;
     }
</style>
```

The final code should look like this:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        h1.hidden{
            display: none;
        }
         h1.vis {
         visibility: hidden;
        }
        div.scroll {
            background-color: #00FFFF;
            width: 100px;
            height: 100px;
            overflow: scroll;
        }

        div.hidden {
            background-color: #00FF00;
            width: 100px;
            height: 100px;
            overflow: hidden;
        }
        p.stay {
            position:fixed;
            bottom: 0;
            background-color: purple;
            border: 2px solid #f9f9f9;
        }
    </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>

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

    <p>The overflow property specifies what to do if the
    content of an element exceeds the size of the element's box.</p>

    <p>overflow:scroll</p>
    <div class="scroll">You can use the overflow property when you want to have
better control of the layout. The default value is visible.</div>

    <p>overflow:hidden</p>
    <div class="hidden">You can use the overflow property when you want to have
better control of the layout. The default value is visible.</div>

    <div class="ex1">This div element has width: 500px;</div>
    <br>

    <div class="ex2">This div element has max-width: 500px;</div>

    <p class="stay">
        <strong>Tip:</strong> Drag the browser window to smaller than 500px
wide, to see the difference between
        the two divs!
    </p>

</body>
</html>
```

Exercises

1. Given the following HTML code, write CSS code to meet the following requirements:

    - Hide the `<h1>` element. It should still take up the same space as before *Hint: Use the visibility property*
    - Display the list of items as inline elements.
    - Display the `<strong>` elements as block elements.

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

   </style>
</head>
<body>

   <h1>This is my shopping list</h1>
   <ul>
       <li>Apple</li>
       <li>Orange</li>
       <li>Pear</li>
   </ul>
   <p>This is a <strong>paragraph</strong>, with some words more
<strong>important</strong> than others </p>
   <p>This is another paragraph.</p>

</body>
</html>
```

2. Create a HTML document that contains an image with some text positioned over the bottom right of the image.

    ▶ Solution