Chapter 16

Visibility and Positioning in CSS

Visibility

To hide an element form view, a property called visibility can be implemented. This property can be used along with JavaScript to create very complex menu and also to develop a complex webpage layouts.

Visibility property can be used to hide error messages that are only displayed if the user needs to see them, or to hide answers to a quiz until the user selects an option.

NOTE – Remember that the source code will still contain whatever is in the invisible paragraph, therefore visibility property should not be used to hide sensitive information such as credit card details or passwords.

Following are the values that is accepted by the visibility property -

|  |  |
| --- | --- |
| Value | Description |
| Visible | The box and its contents are shown to the user. |
| Hidden | The box and its content are made invisible, although they still affect the layout of the page. |
| Collapse | This is for use only with dynamic table columns and row effects |

* Example 1

<html>

<head>

</head>

<body>

<p> This paragraph should be visible in normal way.

</p>

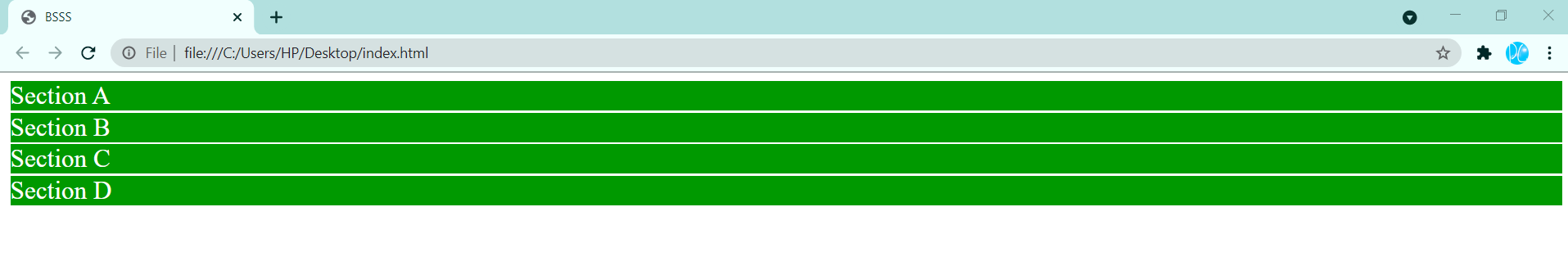
<p style = "visibility:hidden;">

</p>

</body>

</html>

The above HTML document will produce the following result –



This paragraph should be visible in normal way.

This paragraph should not be visible.

**Positioning**

CSS Positioning could be used to set a layout of a webpage, instead of using table(s). By using the CSS positioning one can make a page more dynamic. It could be specified whether the element should be positioned relative to its natural position in the page or absolute based on its parent element.

Following are types of positioning:

* Normal Flow/Static Positioning
* Relative Positioning
* Absolute Positioning
* Fixed Positioning

Now, we will see all the CSS positioning related properties with examples –

Normal Flow

Normal Flow is the default behavior of a web browser. It is not required to specify this in the style sheet since it is the default. With normal flow boxes will show up in the order that it is placed in the code, and each box level element is stacked on the next.

<styletype="text/css">

#main {

border:1pxsolid#00F;

}

#content {

border:1pxsolid#F00;

}

</style>

Static Positioning

Static positioning is applied by the declaration position: static. This places the element in the normal flow. Since normal flow is the default it is not normally necessary to explicitly use this.

Where it is useful is over-riding another rule of lower specifity, e.g.

div { position:absolute; }

#notMe { position:static; }

would absolutely position all div elements except the one whose id attribute has the value *notMe*.

The left, top, right and bottom properties are not needed since they don't influence static positioning. They are used below to show they have no influence.

|  |  |  |
| --- | --- | --- |
| <styletype="text/css">  #main {  border:1pxsolid#00F;  }  #content {  border:1pxsolid#F00;  position:static;  left:100px;  top:125px;  right:50px;  bottom:30px;  }  </style> |  |  |

## Relative Positioning

The browser first lays out the element as though it was in the normal flow. The element is then displaced by the amount specified by the left or right properties and the top or bottom properties. A gap is left in the normal flow at the point the element should have appeared. Relative positioning does not allow an element to change size. If both left and right are specified right will be ignored in languages which are written left to right such as English. bottom is ignored if top is specified.

Relative positioning changes the position of the HTML element relative to where it normally appears. So "left:20" adds 20 pixels to the element's LEFT position.

You can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

* Move Left - Use a negative value for *left*.
* Move Right - Use a positive value for *left*.
* Move Up - Use a negative value for *top*.
* Move Down - Use a positive value for *top*.

**NOTE** − You can use *bottom* or *right* values as well in the same way as *top* and *left*.

* Example 2

<html>

<head>

</head>

<body>

<div style = "**position:relative; left:80px; top:2px;** background-color:yellow;">

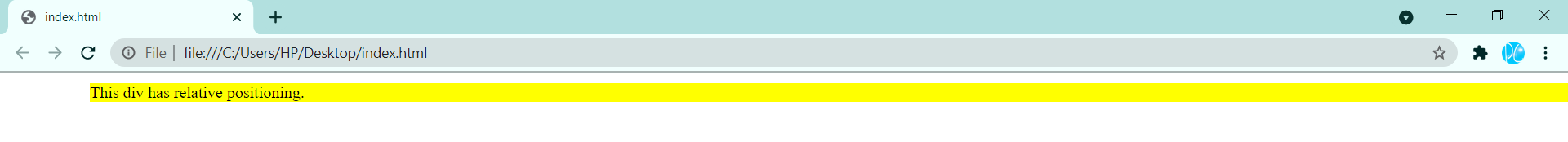
This div has relative positioning.

</div>

</body>

</html>

The above HTML document will produce the following result –



## Absolute Positioning

This positions a box relative to its containing block. However, unlike relative positioning the gap in the normal flow left by removing the element closes up. The containing block is the nearest ancestor with a 'position' of 'absolute', 'relative' or 'fixed'.

You can use any one or combination of left, top, right, and bottom properties to position the box. The co-ordinates for absolute position have (0,0) at the top left of the containing block. Increasing the value of top moves the element down the page.

Since absolutely positioned boxes are taken out of the normal flow they can be positioned anywhere on the page regardless of their position in the document's source order.

An element with **position: absolute** is positioned at the specified coordinates relative to your screen top-left corner.

You can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

* Move Left - Use a negative value for *left*.
* Move Right - Use a positive value for *left*.
* Move Up - Use a negative value for *top*.
* Move Down - Use a positive value for *top*.

**NOTE** − You can use *bottom* or *right* values as well in the same way as top and left.

* Example 3

<html>

<head>

</head>

<body>

<div style = "**position:absolute; left:80px; top:20px;** background-color:yellow;">

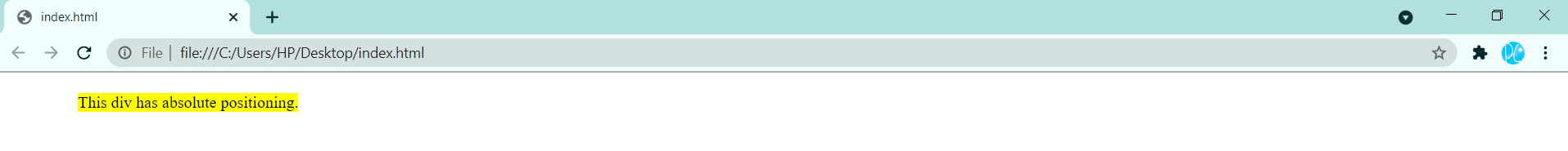
This div has absolute positioning.

</div>

</body>

</html>

The above HTML document will produce the following result –



## Fixed Positioning

Fixed positioning is a subcategory of absolute positioning. The only difference is that for a fixed positioned box, the containing block is established by the browser window size. A fixed element does not move when a web page is scrolled as all other elements do. It is calculated in the same way as absolute positioning with respect to containing blocks in that it pulls the positioned box out of the normal flow.

Fixed positioning allows you to fix the position of an element to a particular spot on the page, regardless of scrolling. Specified coordinates will be relative to the browser window.

You can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

* Move Left - Use a negative value for *left*.
* Move Right - Use a positive value for *left*.
* Move Up - Use a negative value for *top*.
* Move Down - Use a positive value for *top*.

**NOTE** − You can use *bottom* or *right* values as well in the same way as *top* and *left*.

* Example 4

<html>

<head>

</head>

<body>

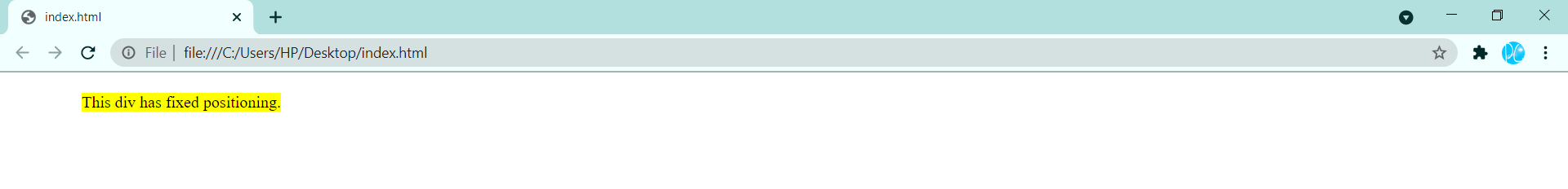
<div style = "**position:fixed; left:80px; top:20px;** background-color:yellow;"> This div has fixed positioning.

</div>

</body>

</html>

The above HTML document will produce the following result –



Validation

Validation is the process of checking something against a rule. When you are a beginner, it is very common that you will commit many mistakes in writing your CSS rules. How you will make sure whatever you have written is 100% accurate and up to the W3 quality standards?

If you use CSS, your code needs to be correct. Improper code may cause unexpected results in how your page looks or functions.

But if you want to validate your CSS style sheet embedded in an (X)HTML document, you should first check that the (X)HTML you use is valid.

Tool to check the validity of (X)HTML document: Validate (X)HTML document.

You can use the following tools to check the validity of your CSS.

|  |  |
| --- | --- |
|  | W3C CSS Validator (World Wide Web Consortium), This validator checks your CSS by either file upload, direct input, or using URI – one page at a time. This validator helps you to locate all the errors in your CSS. |
|  | The WDG CSS check validator, lets you validate your CSS by direct input, file upload, and using URI. Errors will be listed by line and column numbers if you have any. Errors usually come with links to explain the reason of error. |

A CSS validator checks your Cascading Style Sheets to make sure that they comply with the CSS standards set by the W3 Consortium. There are a few validators which will also tell you which CSS features are supported by which browsers (since not all browsers are equal in their CSS implementation).

Why Validate Your HTML Code?

There are a number of reasons why you should validate your code. But major ones are-

1. It helps Cross-Browser, Cross-Platform, and Future Compatibility.
2. A good quality website increases search engine visibility.
3. Professionalism: As a web developer, your code should not raise errors while seen by the visitors.

Floating Elements

Elements can be made to float within the normal flow. Boxes are moved left or right as far as they can go. Elements after the float box will move up to fill any gap left behind thus flowing around the box with the float position.

Notice that float is not a position property, but it acts like one. Float is applied with the float property not the position property. The float is positioned to the left in the example, but you could easily make it positioned to the right. A width was added to the content block so you can see how elements after the float block move up and wrap around the area the content block does not occupy.

You must set the width property when floating block-level elements otherwise they will expand to fill the entire width of their container.

|  |  |  |
| --- | --- | --- |
| <styletype="text/css">  #main {  border:1pxsolid#00F;  }  #content {  border:1pxsolid#F00;  left:100px;  top:125px;  right:50px;  bottom:30px;  float:left;  width:425px;  }  </style> |  | Float Positioning |