Module-4 Box Model and Its Properties

1. Explain Overflow & its property.

The overflow property controls what happens to content that breaks outside of its bounds: imagine a div in which you've explicitly set to be 200px wide, but contains an image that is 300px wide. That image will stick out of the div and be visible by default. Whereas if you set the overflow value to hidden, the image will cut off at 200px.

```
div {
  overflow:  visible | hidden | scroll | auto | inherit

}
```

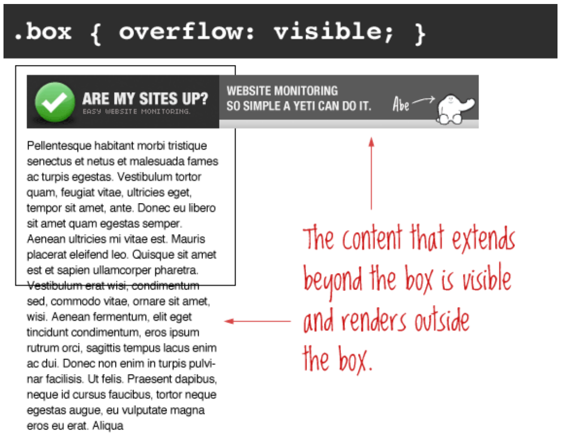
Values

- visible: content is not clipped when it proceeds outside its box. This is the default value of the property
- hidden: overflowing content will be hidden.
- scroll: similar to hidden except users will be able to scroll through the hidden content
- auto: if the content proceeds outside its box then that content will be hidden whilst a scroll bar should be visible for users to read the rest of the content.
- initial: uses the default value which is visible
- inherit: sets the overflow to the value of its parent element.

Remember that text will naturally wrap at the end of an element (unless white-space is changed) so text will rarely be the cause of overflow. Unless a height is set, text will just push an element taller as well. Overflow comes into play more commonly when explicit widths and heights are set and it would be undesirable for any content to spill out, or when scrolling is explicitly being avoided.

Visible

If you don't set the overflow property at all, the default is visible. So in general, there is no reason to explicitly set this property to visible unless you are overriding it from being set elsewhere.

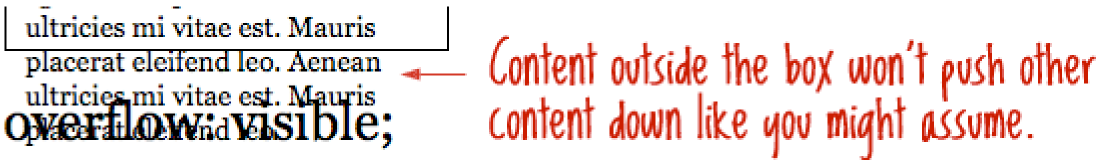


```
.box { overflow: visible; }
```

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi. Aenean fermentum, elit eget tincidunt condimentum, eros ipsum rutrum orci, sagittis tempus lacus enim ac dui. Donec non enim in turpis pulvinar facilisis. Ut felis. Praesent dapibus, neque id cursus faucibus, tortor neque egestas augue, eu vulputate magna eros eu erat. Aliqua

The content that extends beyond the box is visible and renders outside the box.

The important thing to remember here is that even though the content is visible outside of the box, that content does not affect the flow of the page. For example:

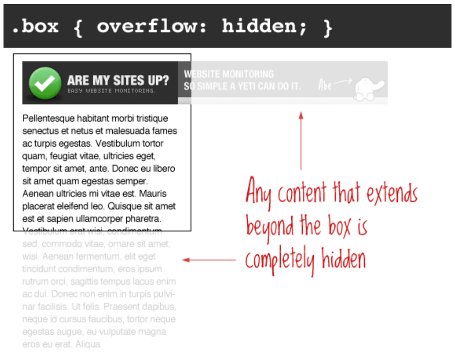

ultricies mi vitae est. Mauris
placerat eleifend leo. Aenean
ultricies mi vitae est. Mauris
placerat eleifend leo.
overflow: visible;

Content outside the box won't push other content down like you might assume.

Generally, you shouldn't be setting static heights on boxes with web text in them anyway, so it shouldn't come up.

Hidden:

The opposite of the default visible is hidden. This literally hides any content that extends beyond the box.



```
.box { overflow: hidden; }
```

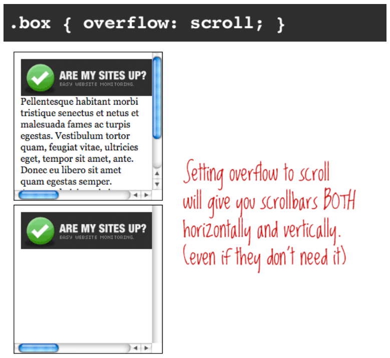
Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi. Aenean fermentum, elit eget tincidunt condimentum, eros ipsum rutrum orci, sagittis tempus lacus enim ac dui. Donec non enim in turpis pulvinar facilisis. Ut felis. Praesent dapibus, neque id cursus faucibus, tortor neque egestas augue, eu vulputate magna eros eu erat. Aliqua

Any content that extends beyond the box is completely hidden

This is particularly useful in use with dynamic content and the possibility of an overflow causing serious layout problems. However, bear in mind that content that is hidden in this way is utterly inaccessible (short of viewing the source). So for example a user has their default font size set larger than you would expect, you may be pushing text outside of a box and hiding it completely from their view.

Scroll

Setting the overflow value of a box to scroll will hide the content from rendering outside the box, but will offer scrollbars to scroll the interior of the box to view the content.



```
.box { overflow: scroll; }
```

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper.

Setting overflow to scroll will give you scrollbars BOTH horizontally and vertically. (even if they don't need it)

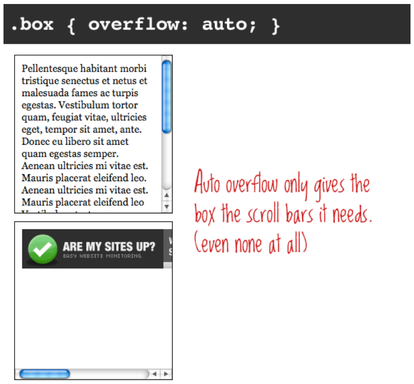
Of note with this value is that you get BOTH horizontal and vertical scrollbars no matter what, even if the content requires only one or the other.

iOS' momentum scrolling can be enabled for this value with -webkit-overflow-scrolling.

Note: In OS X Lion, when scrollbars are set to only show when being used, scroll behaves more like auto, in that only needed scrollbars will show up.

Auto

Auto overflow is very similar to the scroll value, only it solves the problem of getting scrollbars when you don't need them. The scrollbars will only show up if there is content that actually breaks out of the element.



```
.box { overflow: auto; }
```

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Aenean ultricies mi vitae est. Mauris placerat eleifend leo

Auto overflow only gives the box the scroll bars it needs. (even none at all)

overflow-x and overflow-y

It's also possible to manipulate the overflow of content horizontally or vertically with the overflow-x and overflow-y properties. For example in the demo below the horizontal overflow can be scrolled through whilst the text that extends beyond the height of the box is hidden:

```
.box {
  overflow-y: hidden;
  overflow-x: scroll;

}
```

2. What is the Box model in CSS? Which CSS properties are a part of it?

   A rectangle box is wrapped around every HTML element. The box model is used to determine the height and width of the rectangular box. The CSS Box consists of Width and height (or in the absence of that, default values and the content inside), padding, borders, margin.
   Box Model In CSS
   - Content: Actual Content of the box where the text or image placed.
   - Padding: Area surrounding the content (Space between the border and content).
   - Border: Area surrounding the padding.
   - Margin: Area surrounding the border.

3. What is the difference between padding and margin?
   Answer: In CSS, the margin is the property by which we can create space around elements. We can even create space to the exterior defined borders.
   In CSS, we have margin property as follows:
   - margin-top
   - margin-right
   - margin-bottom

- Margin-left

Margin property has some defined values as shown below.

- Auto – Using this property browser calculates the margin.
- Length – It sets the margin values in px,pt,cm etc.
- % – It sets the width % of the element.
- Inherit – By this property we can inherit the margin property from the parent element.

In CSS, padding is the property by which we can generate space around an element’s content as well as inside any known border.

CSS padding also has properties like,

1. Padding-top
2. Padding-right
3. Padding-bottom
4. Padding-left

Negative values are not allowed in padding.

```
div {

padding-top: 60px;

padding-right: 40px;

padding-bottom: 50px;

padding-left: 70px;

}
```

4. What is the use of the Box Model in CSS?

Answer: In CSS, the box model is a box that binds all the HTML elements and it includes features like margins, border, padding, and the actual content.

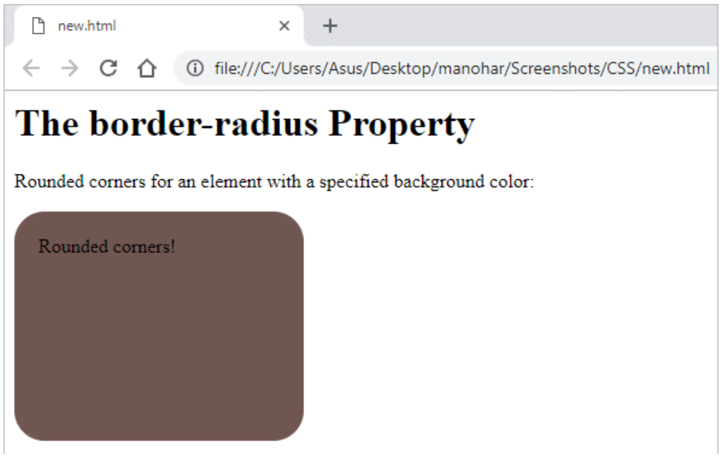
By using a box model we will get the authority to add the borders all around the elements and we can also define the space between the elements.

5. How do we make a rounded corner by using CSS?

Answer: We can make a rounded corner by using the property “border-radius”. We can apply this property to any element.

Example:

```
<html>

<head>



<style>

#rcorners1 {
  border-radius: 25px;
  background: #715751;
  padding: 20px;
  width: 200px;
  height: 150px;
}

</style>
</head>

<body>
<h1>The border-radius Property</h1>
<p>Rounded corners for an element with a specified background color:</p>
<p id="rcorners1">Rounded corners!</p>
</body>

</html>
```


new.html
file:///C:/Users/Asus/Desktop/manohar/Screenshots/CSS/new.html

The border-radius Property

Rounded corners for an element with a specified background color:

Rounded corners!

6. How will you add border images to an HTML element?
   Answer: We can set the image to be used as the border-image alongside an element by using the property of CSS "border-image".
   Example:

```
#borderimg {

   border: 15px solid transparent;

   padding: 20px;

   border-image: url(border.png) 30 round;

}
```

7. Explain Box shadow property.

   The box-shadow property in CSS is for putting shadows on elements (sometimes referred to as "drop shadows", ala Photoshop/Figma).

```
.card {
  box-shadow: 0 3px 10px rgb(0 0 0 / 0.2);
}
```

   That syntax is:

```
box-shadow: [horizontal offset] [vertical offset] [blur radius] [optional spread radius] [color];
```

   1. The horizontal offset (required) of the shadow, positive means the shadow will be on the right of the box, a negative offset will put the shadow on the left of the box.
   2. The vertical offset (required) of the shadow, a negative one means the box-shadow will be above the box, a positive one means the shadow will be below the box.
   3. The blur radius (required), if set to 0 the shadow will be sharp, the higher the number, the more blurred it will be, and the further out the shadow will extend. For instance a shadow with 5px of horizontal offset that also has a 5px blur radius will be 10px of total shadow.
   4. The spread radius (optional), positive values increase the size of the shadow, negative values decrease the size. Default is 0 (the shadow is same size as blur).
   5. Color (required) – takes any color value, like hex, named, rgba or hsla. If the color value is omitted, box shadows are drawn in the foreground color (text color). But be aware, older WebKit browsers (pre Chrome 20 and Safari 6) ignore the rule when color is omitted.

   Using a semi-transparent color like rgba(0, 0, 0, 0.4) is most common, and a nice effect, as it doesn't completely/opaquely cover what it's over, but allows what's underneath to show through a bit, like a real shadow.