SNS COLLEGE OF TECHNOLOGY

(An AUTONOMOUS INSTITUTION)

RE-ACCREDITED BY NAAC WITH A+ GRADE, ACCREDITED BY NBA(CSE, IT, ECE, EEE & MECHANICAL)

APPROVEDY BY AICTE, NEW DELHI, RECOGNIZED BY UGC, AFFILIATED TO ANNA UNIVERSITY, CHENNAI

DEPARTMENT OF MASTER COMPUTER APPLICATION

HTML INTRODUCTION

19CAT601 - WEB PROGRAMMING ESSENTIALS

UNIT-II – CSS - Advanced CSS4 Concepts

I MCA – I SEM

Advanced CSS 4 concepts

Table of Contents

Advanced CSS 4 concepts

Advanced CSS

Rounded Corners: Corners. That are rounded.
Shadows: Adding “pop” to boxes and text.
Universal, Child, and Adjacent Selectors: More precise aim with clever selectors.
Advanced Colors: Alpha **transparency** and **HSL**.
At-Rules: Importing style sheets, styles for different **media types**, specifying the character set of a stylesheet and **embedded fonts**.
Attribute Selectors: Targeting boxes by their elements’ HTML attributes.
CSS Transitions: Creating smooth animations.
Backgrounds: Multiples, Size, and Origin
Transformations: Molding the size and shape of a box and its contents.
Gradients: Linear and radial gradients without image files.
Media Queries: Optimizing pages for different devices and screen sizes.

Advanced CSS 4 concepts

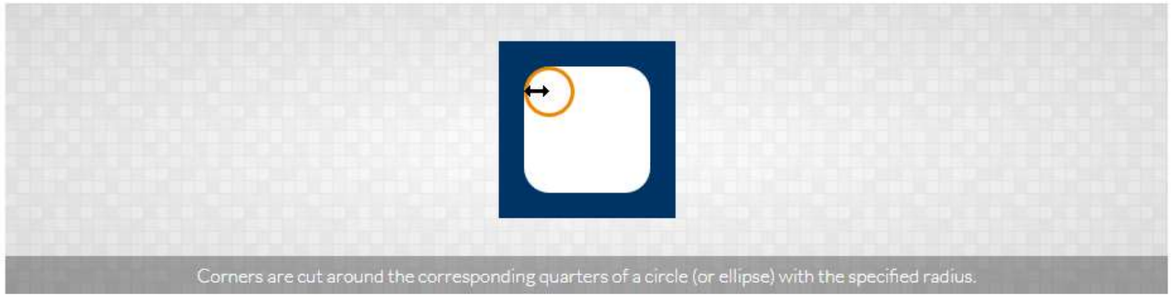
Advanced CSS

Border radius?

Yeah. Border radius. Fear not, though — you don't have to have borders. The `border-radius` property can be used to add a corner to each corner of a box.

```
#marilyn {
    background: #fff;
    width: 100px;
    height: 100px;
    border-radius: 20px;
}
```

And there you have it. Rounded corners, see? Well, you will if you've got a sensible browser (see note below).



Corners are cut around the corresponding quarters of a circle (or ellipse) with the specified radius.

Multiple values

`border-top-left-radius`, `border-top-right-radius`, `border-bottom-right-radius`, and `border-bottom-left-radius` can also be used to target specific corners.

Ever so slightly less horribly wordy, you can also define all corner radii (what a great word) individually with a space-separated list of values, working clockwise from top-left, just like other shorthand properties:

```
#monroe {
    background: #fff;
    width: 100px;
    height: 100px;
    border-radius: 6px 12px 18px 24px;
}
```

Multiple-value border-radius.

Advanced CSS 4 concepts

Advanced CSS

Ellipses

Are circles a bit too square for you? You can specify different horizontal and vertical radiiii by splitting values with a "/".

```
#nanoo {
        background: #fff;
        width: 100px;
        height: 150px;
        border-radius: 50px/100px;
        border-bottom-left-radius: 50px;
        border-bottom-right-radius: 50px;
}
```

Advanced CSS 4 concepts

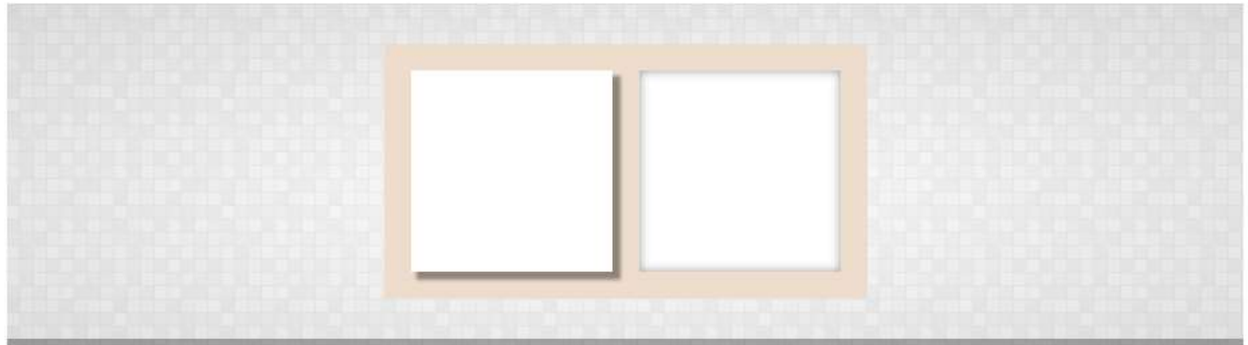
Advanced CSS

Inner shadows

You can also apply shadows to the inside of a box by adding "inset" to the list:

```
box-shadow: inset 0 0 7px 5px #ddd;
```

Splendid!



A basic drop-shadow and an inner-shadow.

i

You might come across browser-specific versions of `box-shadow`, such as `-moz-box-shadow` and `-webkit-box-shadow`. Ignore 'em. They're old and stupid. The majority of modern browsers understand `box-shadow`, including Internet Explorer versions 9 and above.

Advanced CSS 4 concepts

Advanced CSS

Universal selectors

Using an **asterisk** (" * "), you can target **everything** under the sun. You can use it by itself to set global styles for a page, or as a descendant of a selector to set styles of everything within something.

The following, for example, will set the margin and padding on everything in a page to zero and everything within an element with the ID "contact" to be displayed as a block:

```
* {
    margin: 0;
    padding: 0;
}

#contact * {
    display: block;
}
```

i

Using a standalone universal selector is commonly used to "reset" many of a browser's default styles. Setting a margin to zero, for example, will kill all spacing around the likes of paragraphs, headings and blockquotes.

Child selectors

A **greater-than** symbol (">") can be used to specify something that is a child of something else, that is, something **immediately nested** within something.

So, with this HTML...

```
<ul id="genus_examples">
    <li>Cats
        <ul>
            <li>Panthera</li>
            <li>Felis</li>
            <li>Neofelis</li>
        </ul>
    </li>
    <li>Apes
        <ul>
            <li>Pongo</li>
            <li>Pan</li>
            <li>Homo</li>
        </ul>
    </li>
</ul>
```

...and the following CSS...

```
#genus_examples > li { border: 1px solid red }
```

...a red border would be drawn around "Cats" and "Apes" only, rather than around every single list item (which would be the case with `#genus_examples li { border: 1px solid red }`). This is because the likes of "Panthera" and "Felis" are **grandchildren** of "genus_examples", not **children**.

Advanced CSS 4 concepts

Advanced CSS

Adjacent selectors

A plus sign ("+") is used to target an adjacent sibling of an element, essentially, something **immediately following** something.

With the following HTML:

```
<h1>Clouded leopards</h1>
<p>Clouded leopards are cats that belong to the genus Neofelis.</p>
<p>There are two extant species: Neofelis nebulosa and Neofelis diardi.</p>
```

...and CSS...

```
h1 + p { font-weight: bold }
```

Only the first paragraph, that following the heading, will be made bold.

A further, CSS 3, "general sibling" selector uses a **tilde** ("~") and will match an element following another regardless of its immediacy. So, in the above example, `h1 ~ p { font-weight: bold }` will style all paragraphs after the top-level heading but if there were any `p`s preceding the `h1`, these would not be affected.

Advanced CSS 4 concepts

Advanced CSS

Advanced Colors

We already know that colors can be defined by name, RGB, or hex values, but CSS 3 also allows you to paint away with **HSL** — hue, saturation, and lightness — as well as stipulating **transparency**.

There are no super special properties at play here — **HSL** and **RGBa** (the "a" standing for "alpha", as in "alpha transparency") can be applied to any property that has a color value, such as `color`, `background-color`, `border-color` or `box-shadow`, to name a mere handful.

Alpha transparency

RGBa opens up an exciting new dimension to web design, allowing you to set the transparency of a box or text. If you wanted a smidgen of a snazzy background image to peep through a heading, for example, you might use something like this:

```
h1 {
    padding: 50px;
    background-image: url(snazzy.jpg);
    color: rgba(0,0,0,0.8);
}
```

A standard value of `rgb(0,0,0)` would set the heading to pure black but that fourth value, in `rgba`, sets the level of transparency, "1" being completely opaque, "0" being completely transparent. So `rgba(0,0,0,0.8)` is saying red="0", green="0", blue="0", alpha="0.8", which, all together, makes it 80% black.

This doesn't only apply to text, of course, you could apply a transparent background color to an entire box, a transparent box shadow... anywhere where you can use `rgb`, you can used `rgba`.

Advanced CSS 4 concepts

Advanced CSS

Hue, saturation, and lightness

Color names aside, web colors have always been red-green-blue biased, be that through hex codes or explicit RBG (or RGBa). Although mildly less straightforward (especially if your brain is trained to break down colors into red, green and blue), HSL can actually be more intuitive because it gives you direct control over the aspects of a color's shade rather than its logical ingredients.

It is used in a similar way to `rgb`:

```
#smut { color: hsl(36, 100%, 50%) }
```

Rather than each sub-value being a part of the color spectrum, however, they are:

- **Hue** ("36" in the above example): Any angle, from 0 to 360, taken from a typical color wheel, where "0" (and "360") is red, "120" is green and "240" is blue.
- **Saturation** ("100%" in the example): How saturated you want the color to be, from 0% (none, so a level of grey depending on the lightness) to 100% (the whole whack, please).
- **Lightness** ("50%" in the example): From 0% (black) to 100% (white), 50% being "normal".

So the example used here will produce an orange (36°) that is rich (100% saturation) and vibrant (50% lightness). It is the equivalent of `#ff9900`, `#f90`, and `rgb(255, 153, 0)`.

HSLa

Hey, man, this funky fresh transparency and HSL can be combined?! You'd better believe it. Here's **HSLa**:

```
#rabbit { background: hsla(0, 75%, 75%, 0.5) }
```

You can figure out what that does, right?

i

`hsl` and `hsla` are supported by most modern browsers, including IE versions 9 and above.

Advanced CSS 4 concepts

Advanced CSS

At-Rules: @import, @media, and @font-face

At-rules are clever, powerful little huggers that encapsulate a bunch of CSS rules and apply them to something specific. They can be used to **import** other CSS files, apply CSS to a particular **media**, or embed funkysexy uncommon **fonts**.

Each at-rule starts with an apetail, or an "at sign", if you want to be boring about it ("@").

Importing

Let's start with the simple `@import` rule. This is used to bolt another stylesheet onto your existing one.

```
@import url(morestyles.css);
```

This can be used if a site requires long, complex stylesheets that might be easier to manage if they are broken down into smaller files.

@import rules **must** be placed at the top of a stylesheet, before any other rules.

Advanced CSS 4 concepts

Advanced CSS

Targeting media types

`@media` can be used to apply styles to a specific media, such as print.

```
@media print {
    body {
        font-size: 10pt;
        font-family: times, serif;
    }

    #navigation {
        display: none;
    }
}
```

Values that follow "`@media`" can include `screen`, `print`, `projection`, `handheld`, and `all`, or a comma-separated list of more than one, such as:

```
@media screen, projection {
    /* ... */
}
```

It doesn't stop there, oh no. CSS 3 allows you to target not only specific media but also variables relating to that media, such as screen size (particularly helpful in targeting phones). Have a gander at the Media Queries page for more.

Advanced CSS 4 concepts

Advanced CSS

Embedding fonts

`@font-face` has been around for a long time but was nigh-on useless for much of its life. CSS 3 has polished it up and it is now widely used as a technique for embedding fonts in a web page so that a typeface can be used even if it isn't sitting on the user's computer. So you no longer need to rely on "web safe" fonts such as Arial or Verdana. Exciting times.

Jumping in at the deep end...

```
@font-face {
    font-family: "font of all knowledge";
    src: url(fontofallknowledge.woff);
}
```

What this does is create a font named "font of all knowledge" using the `font-family` **descriptor** and links the font file "fontofallknowledge.woff" to that name using the `src` descriptor. "font of all knowledge" can then be used in a standard font rule, such as:

```
p { font-family: "font of all knowledge", arial, sans-serif; }
```

The font will be downloaded (in this case from the same directory as the CSS file) and applied to paragraphs. If the browser is too decrepit to deal with sparkly new font-faces, it will simply revert to the next, standard, font in the list. Magic!

You can also look for a number of fonts to apply to the rule with a comma-separated list. Checking to see if a font is already present on a user's computer, removing the need to download it, can also be accomplished by replacing "url" in the `src` value with "local".

And because a font file might contain a whole host of weights or styles, you might also want to specify which one you're interested in. So the `@font-face` rule could end up looking something like this:

```
@font-face {
    font-family: "font of all knowledge";
    src: local("font of all knowledge"), local(fontofallknowledge), url(fontofallknowledge.woff);
    font-weight: 400;
    font-style: normal;
}
```

Advanced CSS 4 concepts

Advanced CSS

Attribute Selectors

What? More selectors? Damn straight. Because someone somewhere really wants you to keep your HTML as tidy as possible. So before you add a class attribute to a tag see if you can simply target it by an attribute that might already be there.

Attribute selectors allow you to style an element's box based on the presence of an HTML attribute or of an attribute's value.

With the attribute...

Appending an attribute name enclosed in square brackets, to style something that simply contains a certain **attribute**, you can do something like this:

```
abbr[title] { border-bottom: 1px dotted #ccc }
```

Advanced CSS 4 concepts

Advanced CSS

With the attribute and its value...

You can further specify that you want to style something with a specific **attribute value**.

```
input[type=text] { width: 200px; }
```

This example will apply a width of 200 pixels only to `input` elements that are specified as being text boxes (`<input type="text">`).

This approach can certainly be useful in forms where `input` elements can take on many guises using the `type` attribute.

You can also target more than one attribute at a time in a way similar to the following:

```
input[type=text][disabled] { border: 1px solid #ccc }
```

This will only apply a gray border to text inputs that are disabled.

CSS 3 further allows you to match an attribute without being exact:

- `[attribute^=something]` will match a the value of an attribute that **begins** with something.
- `[attribute$=something]` will match a the value of an attribute that **ends** with something.
- `[attribute*=something]` will match a the value of an attribute that **contains** something, be it in the beginning, middle, or end.

So, as an example, you could style external links (eg. "http://www.htmldog.com") differently to internal links (eg. "overhere.html") in the following way:

```
a[href^=http] {
    padding-right: 10px;
    background: url(arrow.png) right no-repeat;
}
```

Advanced CSS 4 concepts

Advanced CSS

CSS Transitions

Transitions allow you to easily **animate** parts of your design without the need for the likes of JavaScript. How dangerous.

At the most simplistic level, think of a traditional `:hover` state, where you might change the appearance of a link when a cursor fondles it:

```
a:link {
    color: hsl(36,50%,50%);
}
a:hover {
    color: hsl(36,100%,50%);
}
```

This creates a binary animation; a link switches from a subdued orange to a rich orange when it is hovered over.

The `transition` property, however, is much more powerful, allowing smooth animation (rather than a jump from one state to another). It is a shorthand property that combines the following properties (which can be used individually if you so choose):

- `transition-property`: which property (or properties) will transition.
- `transition-duration`: how long the transition takes.
- `transition-timing-function`: if the transition takes place at a constant speed or if it accelerates and decelerates.
- `transition-delay`: how long to wait until the transition takes place.

Returning to the shorthand property, if a transition is applied like so...

```
a:link {
    transition: all .5s linear 0;
    color: hsl(36,50%,50%);
}
a:hover {
    color: hsl(36,100%,50%);
}
```

...when the link is hovered over, the color will gradually change rather than suddenly switch. The `transition` property here is saying it wants all properties transitioned over half a second in a linear fashion with no delay.

i

For a transition to take place, only `transition-duration` is required, the rest defaulting to `transition-property: all; transition-timing-function: ease; transition-delay: 0;`. So you could, for example, simply declare `transition: .5s`.

Targeting specific properties

While "all" can be used in transition-property (or transition), you can tell a browser only to transition certain properties, by simply plonking in the property name you want to change. So the previous example could actually include `transition: color .5s ease 0`, given only the color changes.

If you want to target more than one property (without using "all"), you can offer up a comma-separated list with `transition-property`:

```
a:link {
    transition: .5s;
    transition-property: color, font-size;
...
```

Advanced CSS 4 concepts

Advanced CSS

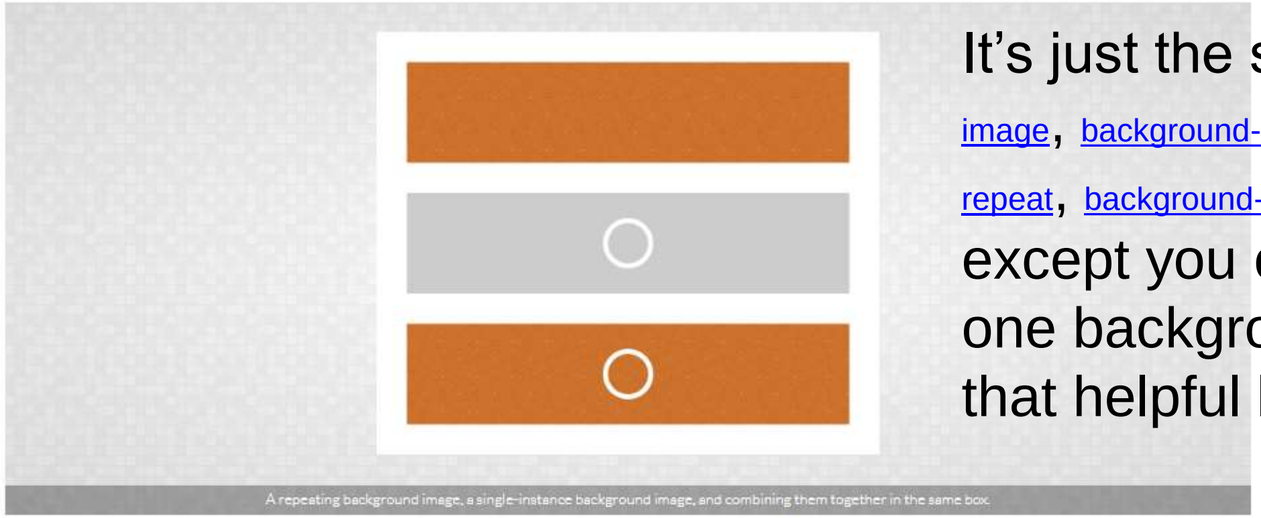
```
background:
url(bg.png),
url(bullet.png) 0
50% no-repeat,
url(arrow.png)
right no-repeat;
```

CSS Backgrounds: Multiples, Size, and Origin

As well as plastering a single or tiling background image across parts of your page, you can also apply **multiple backgrounds**, adjust the **size of background images**, and define the **origin of a background** based on levels of the box model.

Multiple backgrounds

This is the boogie web designers have been crying out for since Bing Crosbie was topping the charts.



A repeating background image, a single-instance background image, and combining them together in the same box.

CSS3 allows you to apply multiple background images to a single box by simply putting image locations in a comma-separated list:

```
background-image: url(this.jpg), url(that.gif), url(theother.png);
```

It's just the same as using background-image, background-position, background-repeat, background-attachment, and background, except you can specify more than one background with the aid of that helpful little comma.

Advanced CSS 4 concepts

Advanced CSS

Background size

The `background-size` property allows you to stretch or compress a background image.

`background-size: contain` and `background-size: cover`

The values can be:

- `auto`, which maintains the background image's **original size and width/height ratio**.
- **lengths**, a width and a height, such as `100px 50px` (100px wide, 50px high). Specifying a single length, such as `100px` will result the equivalent of `100px auto`.
- **percentages**, a width and a height, such as `50% 25%` (50% of the width of the background area, 25% of the height of the background area). Specifying a single percentage, such as `50%` will result the equivalent of `50% auto`.
- A **combination** of lengths, percentages, and `auto`, such as `80px auto` (80px wide, automatic height to maintain the image's original ratio)
- `contain`, which maintains the background image's **original ratio** and makes it **as large as possible** whilst **fitting entirely within the box's background area**.
- `cover`, which maintains the background image's **original ratio** and makes it **large enough to fill the entire background area**, which may result in cropping of either the height or width.

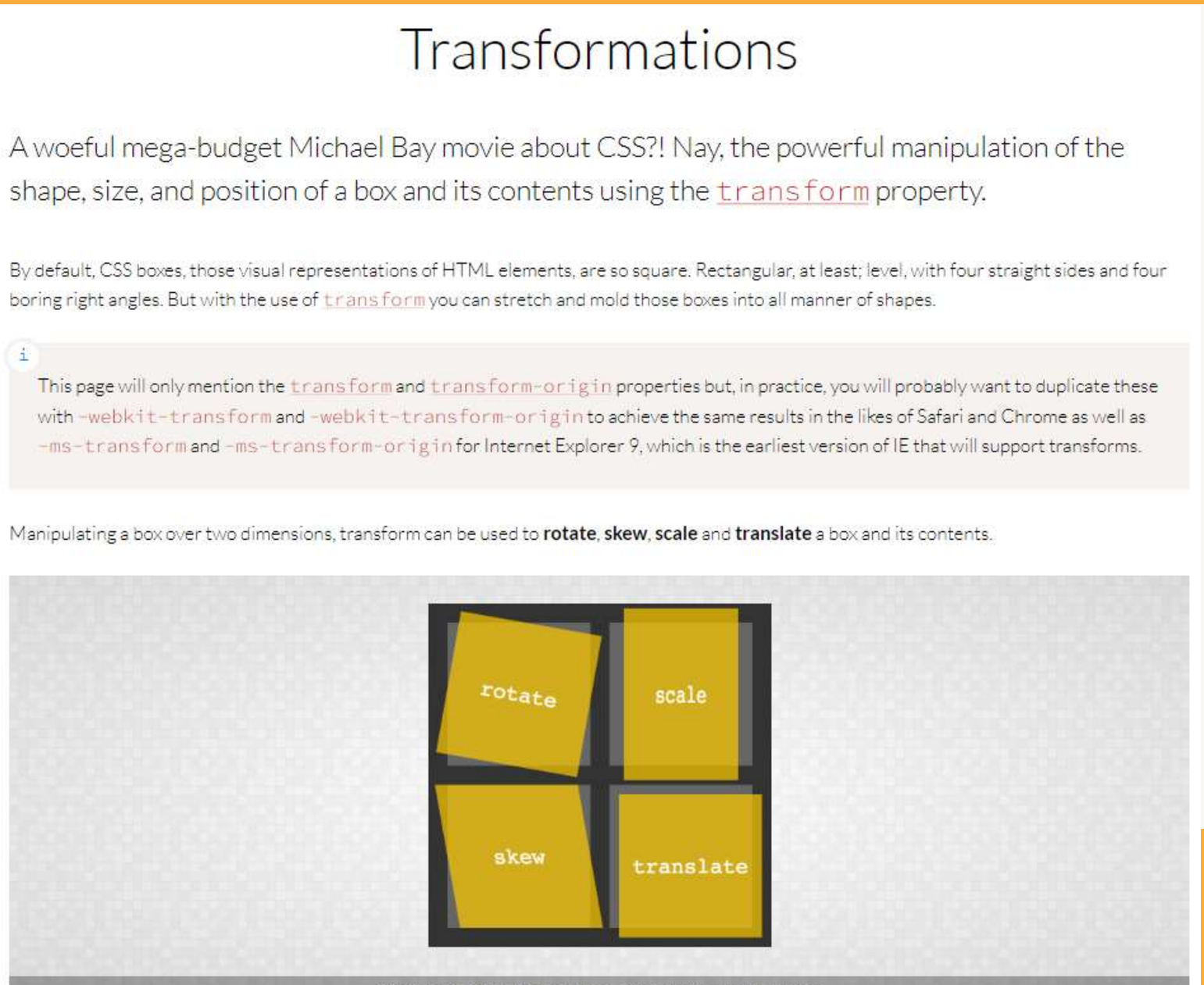
Advanced CSS 4 concepts

Advanced CSS

Rotating

The following would result in a square orange box with all of its contents — text, images, whatever — obediently standing to attention:

```
.note { width: 300px;
height: 300px;
background:
hsl(36,100%,50%); }
transform: rotate(-10deg);
```



Skewing
transform: skew(20deg,10 deg);

Scaling
transform: scale(2);
transform: scale(1,2);

Translating
transform: translate(100px,200px);

Combining Transformations

transform: rotate(-10deg) scale(2);
transform: matrix(2,-0.35,0.35,2,0,0);

Advanced CSS

```
background: linear-gradient(to right, orange, red);
```

Or one corner:

```
background: linear-gradient(to bottom right, orange, red);
```

Or any angle that tickles your fancy:

```
background: linear-gradient(20deg, orange, red);
```

Note that the “to” is excluded with angles because there isn’t an explicit destination.

Gradients

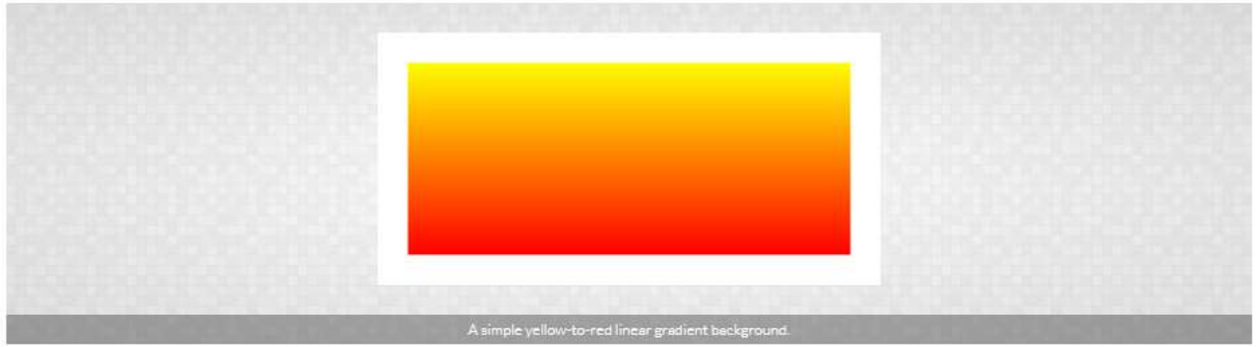
Images showing a smooth dissolve from one color to another are plastered all over the web. However, CSS 3 allows you to place them in your designs without having to create an actual image file.

There is no special property for this; you simply use the `background` or `background-image` property and define your gradient in its value. You can create both **linear** and **radial** gradients this way.

Linear gradients

For an even spread of two colors, fading from one at the top to another at the bottom, a declaration can simply be something like:

```
background: linear-gradient(yellow, red);
```



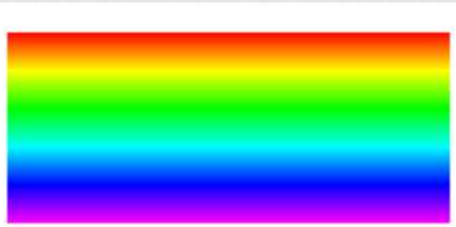
A simple yellow-to-red linear gradient background.

To manipulate the angle of the fading, you slot in "to" and the destination you want the transition to head to. You can head to one side:

```
background: linear-gradient(to right, orange, red);
```

Advanced CSS

```
background: linear-
gradient(hsl(0,100%,50%),hsl(60,100%,50%),hsl(120,100%,50%),hsl(180,100%,50%),hsl(240,100%,50%),hsl(3
```



A colorful gradient extravaganza.

!

To get gradients to work on as many browsers as possible, you will probably also want to use `-webkit-linear-gradient` to cover Safari and older versions of Chrome. **The values of these must also exclude the "to" part**, if used.

CSS gradients won't play at all with IE 9 and below, but you can still make them, and any other incapable browser, use the traditional method of a good old fashioned image by specifying that first, as a fallback (proceeding declarations will just be ignored).

So, all-in, your gradients might look something like this:

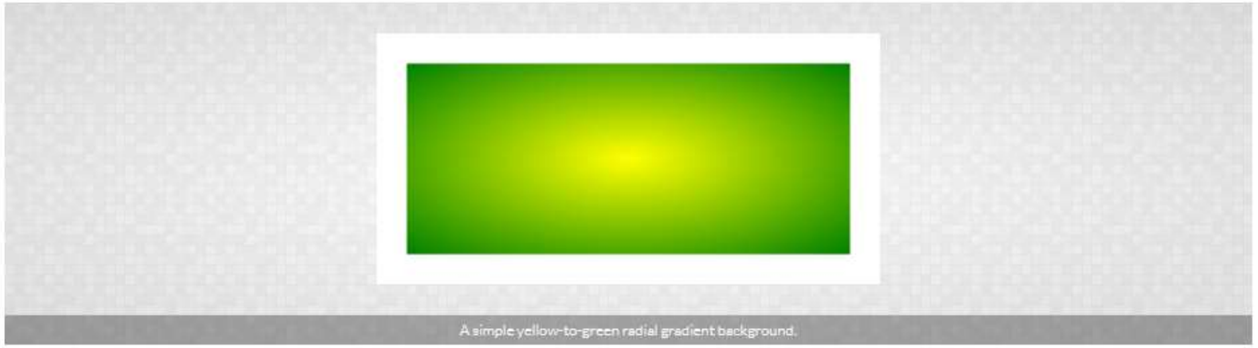
```
body {
    background: #666 url(fade.png) 0 0 repeat-y;
    background: -webkit-linear-gradient(right, #000, #666);
    background: linear-gradient(to right, #000, #666);
}
```

Advanced CSS

Radial gradients

Radial gradients, with one color starting from a central point and fading to another color, use a similar syntax:

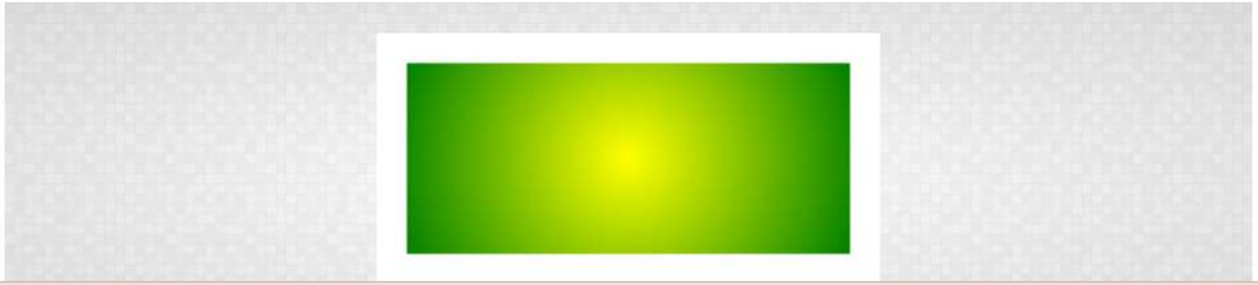
```
background: radial-gradient(yellow, green);
```



A simple yellow-to-green radial gradient background.

You can also specify the shape of the fade. By default it is an ellipse, spreading to fill the background box, but you can force it to be circular, regardless of the shape of the box:

```
background: radial-gradient(circle, yellow, green);
```



Using "closest-side", "closest-corner", "farthest-side" and "farthest-corner" you can also specify if the gradient is contained by the sides or corners of the box nearest to or furthest away from the origin:

```
background: radial-gradient(circle closest-side, yellow, green);
```

And if you wanted to place the origin of the gradient somewhere specific, you can also use "at":

```
background: radial-gradient(at top left, yellow, green);
```

A radial gradient emanating from the top-left corner of a box.

!

More compatibility notes: additional `-webkit-radial-gradient` usage is wise. The order of the parameters needs to be changed here, and, like "to", "at" is excluded. So:

```
background: radial-gradient(circle closest-side at 100px 200px , black, white);
```

Would be accompanied by:

```
background: -webkit-radial-gradient(100px 200px, circle closest-side, black, white);
```

Advanced CSS

Color stops

If you don't want a uniform blend across your gradient, you can specify exactly where in the gradient each color kicks in, straight after each color, starting at "0" and ending at "100%" (although lengths can also be used).

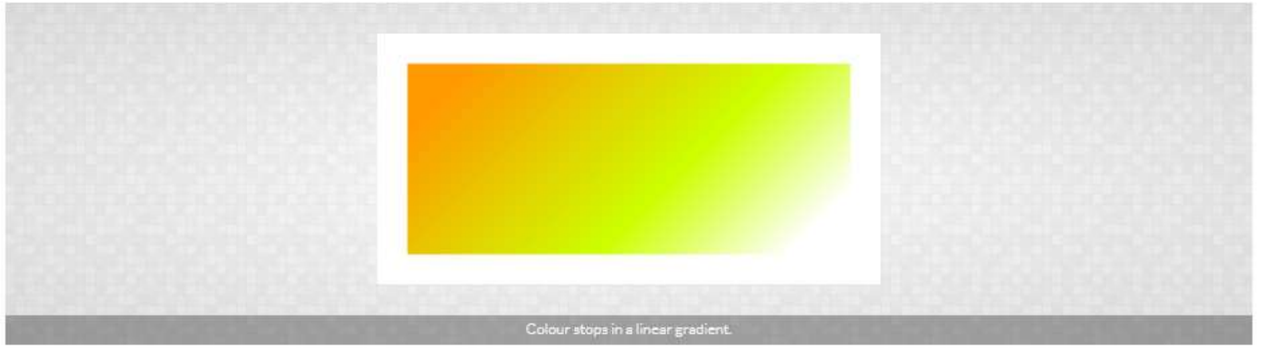
So, just to make it clear before tinkering:

- `linear-gradient(black 0, white 100%)` is the equivalent of `linear-gradient(black, white)`
- `radial-gradient(#06c 0, #fc0 50%, #039 100%)` is the same as `radial-gradient(#06c, #fc0, #039)`
- `linear-gradient(red 0%, green 33.3%, blue 66.7%, black 100%)` will have the same result as `linear-gradient(red, green, blue, black)`

That's because, when the positions are stated in these examples, they evenly space out the colors, which is the default when no color stops are explicitly defined.

So, to get on with that tinkering, you can pull and stretch away with those stops:

```
background: linear-gradient(135deg, hsl(36,100%,50%) 10%, hsl(72,100%,50%) 60%, white 90%);
```



Colour stops in a linear gradient.

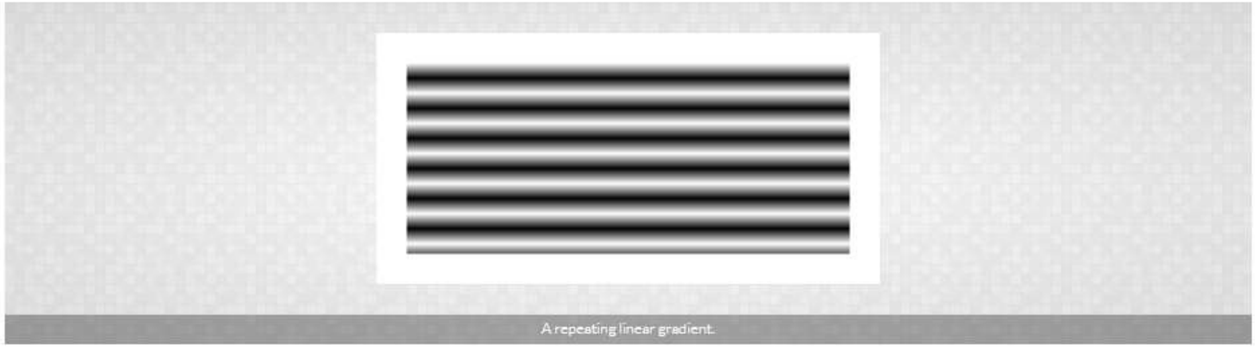
Advanced CSS

Repeating gradients

A single gradient will fill a box with the previous methods but you can use "repeating-linear-gradient" and "repeating-linear-gradient" to build on the color stops and, well, repeat the gradient.

For basic bars of black-and-white bars, for example:

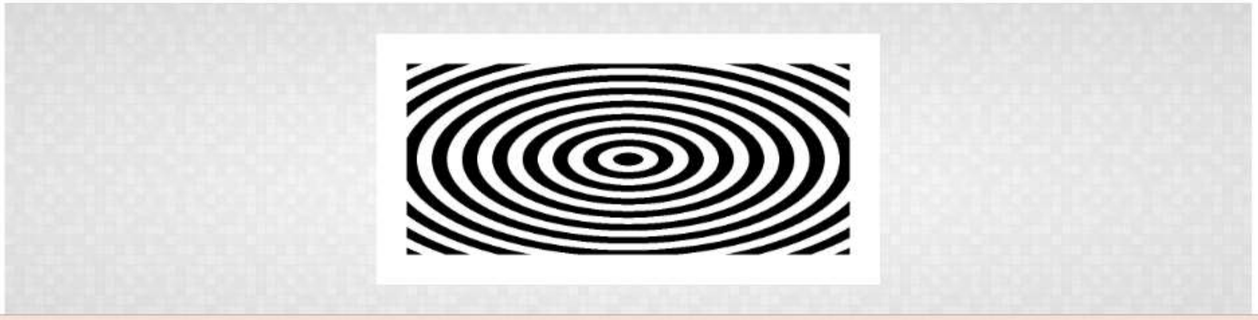
```
background: repeating-linear-gradient(white, black 15px, white 30px);
```



A repeating linear gradient.

Or something a bit more solid:

```
background: repeating-radial-gradient(black, black 15px, white 15px, white 30px);
```



Advanced CSS

Cascading Style Sheets (CSS) handles the look and feel part of a web page. It is used to change the document's style, such as colors, layout, and size of the text. CSS is easy to learn and understand, and it provides robust control over the presentation of an HTML document.

CSS Pseudo-Classes

The pseudo-class selector in CSS is used to select the element that is in a specific state.

Syntax:

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

There are many pseudo-classes, but the ones that are commonly used are:

Advanced CSS

:hover pseudo-class

This pseudo-class is used to add a special effect to an element when our mouse pointer is over it.

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS</title>
    <style>
    .box{
        background-color: pink;
        width: 300px;
        height: 200px;
        margin: auto;
        font-size: 40px;
        text-align: center;
    }

    .box:hover{
        background-color: cyan;
    }

    h1, h2{
        color: black;
        text-align: center;
    }
    </style>
</head>

<body>
    <h1>Simplilearn</h1>
    <h2>:hover Pseudo-class</h2>
    <div class="box">
        My color changes if you hover over me!
    </div>
</body>
```

Advanced CSS

:active pseudo-class

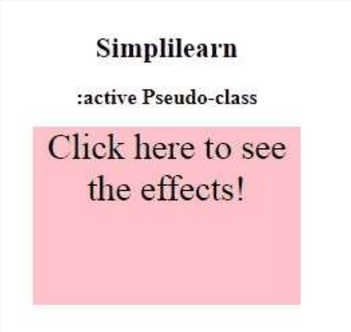
The :active pseudo-class represents an element that the user has activated either by a pointing device or by a touchscreen device.

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS</title>
    <style>
    .box{
        background-color: pink;
        width: 300px;
        height: 200px;
        margin: auto;
        font-size: 40px;
        text-align: center;
    }

    .box:active{
        background-color: cyan;
    }

    h1, h2{
        color: black;
        text-align: center;
    }
    </style>
</head>

<body>
    <h1>Simplilearn</h1>
    <h2>:active Pseudo-class</h2>
    <div class="box">
        Click here to see the effects!
    </div>
</body>
</html>
```



Advanced CSS

CSS Pseudo-Elements

A CSS pseudo-element is particularly used in styling specific parts of an element.

Syntax:

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

The most commonly used pseudo-elements are:

::first-letter pseudo-element

The ::first-letter applies a style to the first letter of the first line of a block-level element.

Syntax:

```
selector::first-letter
{
  property: value;
}
```

Advanced CSS

First Letter Pseudo Element

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS</title>
    <style>
    body{
    text-align: center;
    }
    h1::first-letter {
    font-family: Lucida Calligraphy;
    font-size: 3cm;
    color: red;
    text-shadow: 5px 8px 9px cyan;
    }
    h1{
    color: red;
    }
    </style>
</head>

<body>
    <h1>Welcome to Simplilearn</h1>
    <h2> This is an example of ::first-letter pseudo-element</h2>
</body>
</html>
```

The ::before Pseudo-element

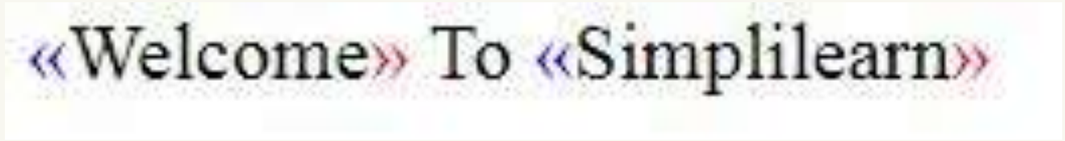
The ::before pseudo-element helps to insert specific content before the element's content.

```
<!DOCTYPE html>
<html>
<head>
<style>
*{
   font-size: 30px;

}
   q::before {
  content: "«";
  color: blue;
}

q::after {
  content: "»";
  color: red;
}

</style>
</head>
<body>
   <q>Welcome</q> To <q>Simplilearn</q>
</body>
```



CSS Z-Index

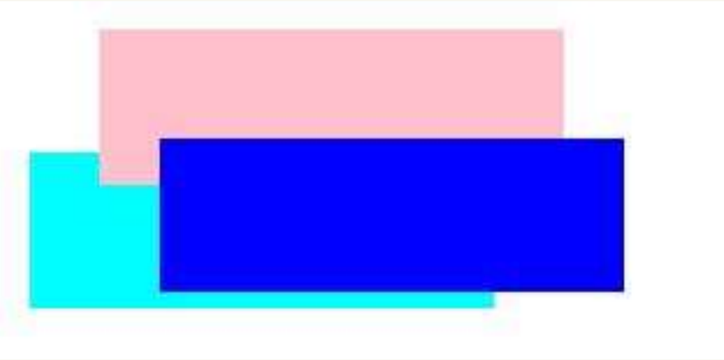
The z-index property defines the order of the elements on the z-axis. An element with a higher z-index is displayed in front of an element with a lower z-index. One thing to note is that the z-index only works with positioned elements.

```
<html>
   <head>
   </head>

   <body align="center">
      <div style = "background-color:pink;
         width:300px;
         height:100px;
         position:relative;
         top:10px;
         left:80px;
         z-index:2">
      </div>

      <div style = "background-color:cyan;
         width:300px;
         height:100px;
         position:relative;
         top:-10px;
         left:35px;
         z-index:1;">
      </div>

      <div style = "background-color:blue;
         width:300px;
         height:100px;
         position:relative;
         top:-120px;
         left:120px;
         z-index:3;">
      </div>
   </body>
</html>
```



Advanced CSS

CSS Minify

Minification (minify) refers to the process of removing unnecessary data like comments, unused codes, changing longer variables and function names to shorter ones, and many more without affecting your code display on browsers.

How to Minify the CSS file?

There are multiple sources available online to minify the CSS file. You can use them to reduce the size of the CSS file.

CSS Loader

A loader is an animation that shows the visitor about the page when it is loading. It is helpful when a page takes some seconds to load the webpage content.

Advanced CSS

CSS Loader

```
<!DOCTYPE html>
<html>
<head>
   <title></title>
   <style type="text/css">
      h1{
            color: orange;
      }
      div{
            display: block;
            position: absolute;
            width: 10px;
            height: 10px;
            left: calc(50% - 1em);
            border-radius: 1.3em;
            transform-origin: 1.3em 2em;
            animation: rotate;
            animation-iteration-count: infinite;
            animation-duration: 2s;
      }

      @keyframes rotate {
            0% { transform: rotate(0deg); }
            100% { transform: rotate(360deg); }
      }
      .a1 {
            animation-delay: 0.1s;
            background-color: cyan;
      }
      .a2 {
            animation-delay: 0.2s;
            background-color: pink;
      }
      .a3 {
            animation-delay: 0.3s;
            background-color: purple;
      }
      .a4 {
            animation-delay: 0.4s;
            background-color: green;
      }
      .a5 {
            animation-delay: 0.5s;
            background-color: black;
      }
   </style>
```

```
   </style>
</head>
<body>
<center>
         <h1>Welcome To Simplilearn</h1>

            <div class='a1'></div>
            <div class='a2'></div>
            <div class='a3'></div>
            <div class='a4'></div>
            <div class='a5'></div>

   </center>
</body>
</html>
```

Advanced CSS

CSS User Interface

The CSS user interface is a module that lets you define the rendering and functionality of features to the user interface. Here you will discuss the following user interface property:

Resize

Outline-offset

Resize Property

The resize property specifies whether an element should be resizable by the user.

Advanced CSS

Resize Property

The resize property specifies whether an element should be resizable by the user.

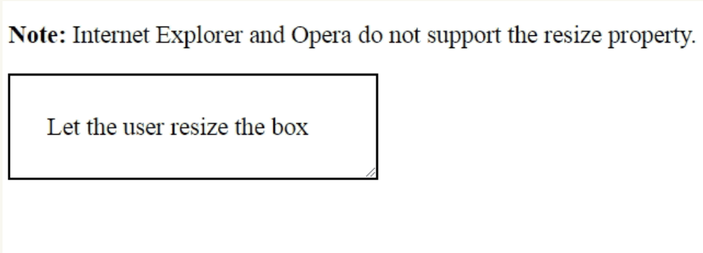
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    border: 3px solid;
    padding: 50px 50px;
    width: 400px;
    resize: both;
    overflow: auto;
    font-size: 34px;
}

body{
   font-size: 34px;
}
</style>
</head>
<body>

<p><b>Note:</b> Internet Explorer and Opera do not support the resize property.</p>

<div>Let the user resize the box</div>

</body>
```

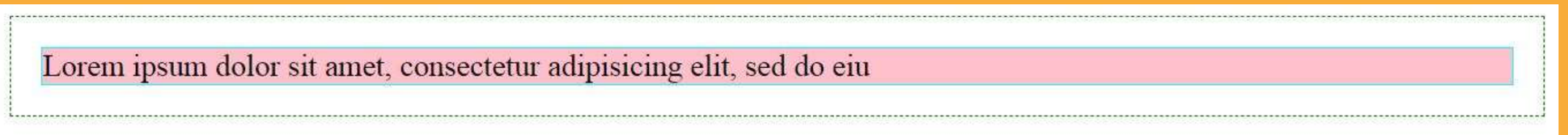


Advanced CSS

Outline-Offset

The outline-offset CSS property sets and determines the space between an outline and the edge or border of an element.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  outline: 1px dashed green;
  outline-offset: 40px;
  background: pink;
  border: 1px solid cyan;
  margin: 60px;
  font-size: 40px;
}
</style>
</head>
<body>
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiu</p>
</body>
```

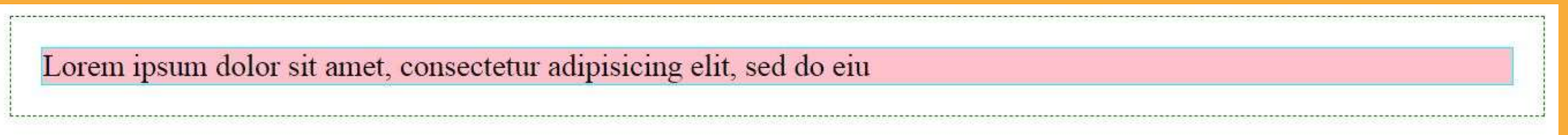


Advanced CSS

Outline-Offset

The outline-offset CSS property sets and determines the space between an outline and the edge or border of an element.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  outline: 1px dashed green;
  outline-offset: 40px;
  background: pink;
  border: 1px solid cyan;
  margin: 60px;
  font-size: 40px;
}
</style>
</head>
<body>
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiu</p>
</body>
```



- References

https://www.simplilearn.com/tutorials/css-tutorial/advanced-css

https://www.htmldog.com/guides/css/advanced/

Thank You!