

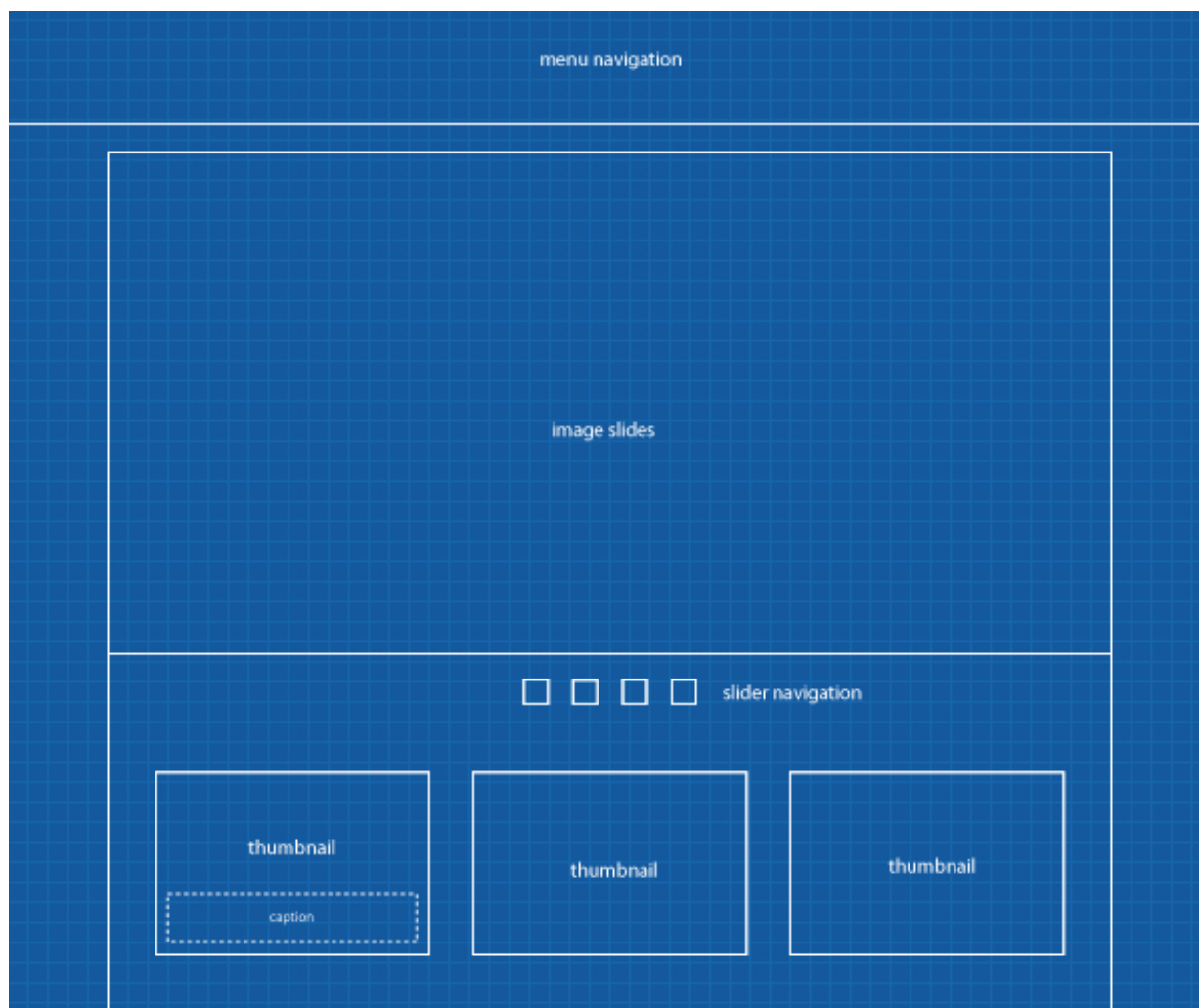
# Practical Use of CSS3: Transforms and Transitions

 [webdesign.tutsplus.com/tutorials/practical-use-of-css3-transforms-and-transitions--pre-33170](http://webdesign.tutsplus.com/tutorials/practical-use-of-css3-transforms-and-transitions--pre-33170)

Thoriq Firdaus

Transforms and Transitions (introduced as part of the CSS3 specification) enable us to create the kind of interaction and experience previously only associated with JavaScript or Flash. You'll likely have read a lot of discussion and tutorials covering these properties, but they tend to be experimental in nature.

In this tutorial we will explore CSS transforms and transitions by creating something more useful. We're going to build a single static HTML page that will utilize transforms and transitions in a slider, plus some interesting hover effects.



Before we continue it's worth noting browser compatibility. This tutorial demonstrates modern techniques, techniques which will be commonplace further down the line, but for the time being suffer incompatibility in older browsers. Take a look at [caniuse.com](http://caniuse.com) for more detailed analysis of which browsers support the various methods discussed here.

## Preparation


Alright, at this point we already know where we are about to go, so we can decide what we need to prepare.

*Before anything else, preparation is the key to success. ~ Alexander G. Bell*

## Basic Skills

First, we need a basic understanding of HTML and CSS. Assuming you have that, we are ready to go.

If, however, you are really new in this area, I would advise you take some time to get up to date on the topic; fortunately Envato has provided a [30 days course](#) which will teach you all the essential things you need to know about HTML & CSS for free.

Course Index						
Intro	1	2	3	4	5	6
Welcome!	Your First Web Page	Your Code Editor	Lists	Parent-Child Relationships	Heading Tags	Blockquotes
7	8	9	10	11	12	13
Anchors	Your First Stylesheet	Clean Project Structures	Images	The Necessity of Divs	IDs and Classes	An Assignment
14	15	16	17	18	19	20
Assignment Solutions	Floats, and a Simple Layout	Navigation Lists	An Introduction to Forms	Image Replacement	The Basics of Typography	Relative and Absolute Positioning
21	22	23	24	25	26	27
Reproduce a Website Fragment	The Importance of Validation	Zen Coding	Resets and Normalizing	CSS Frameworks	Creating the Markup	How to Slice a PSD
28	29	30	Conclusion	 Get notified about new web dev and design courses from <a href="#">Tuts+ Premium</a> .		
How to Create Snippets	The CSS for our website	Completing the website	You're Done!			

## Find Images

We'll obviously need some images to make the image slider, as well as the image captions for the website. [PhotoDune](#) will serve our purposes very well. The following are the images that we will be using in this tutorial.

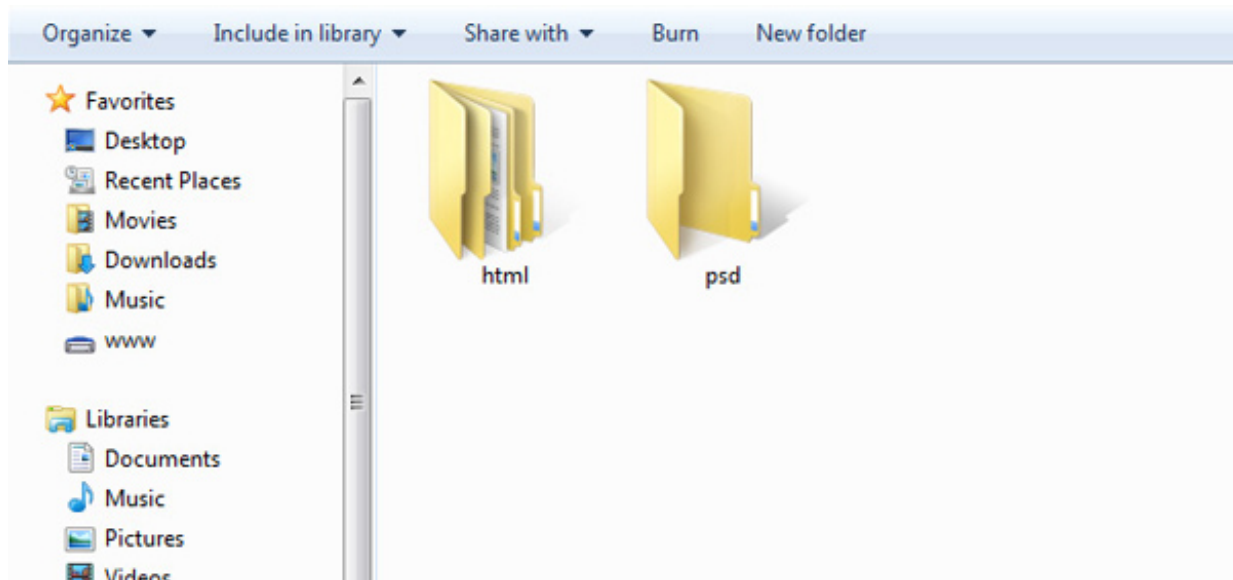


We will also need a texture for the background. My favorite place to find textures at the moment is [Subtle Patterns](#) and after browsing through all the pages, I finally decided to use [this example](#).

---

## Managing Folders

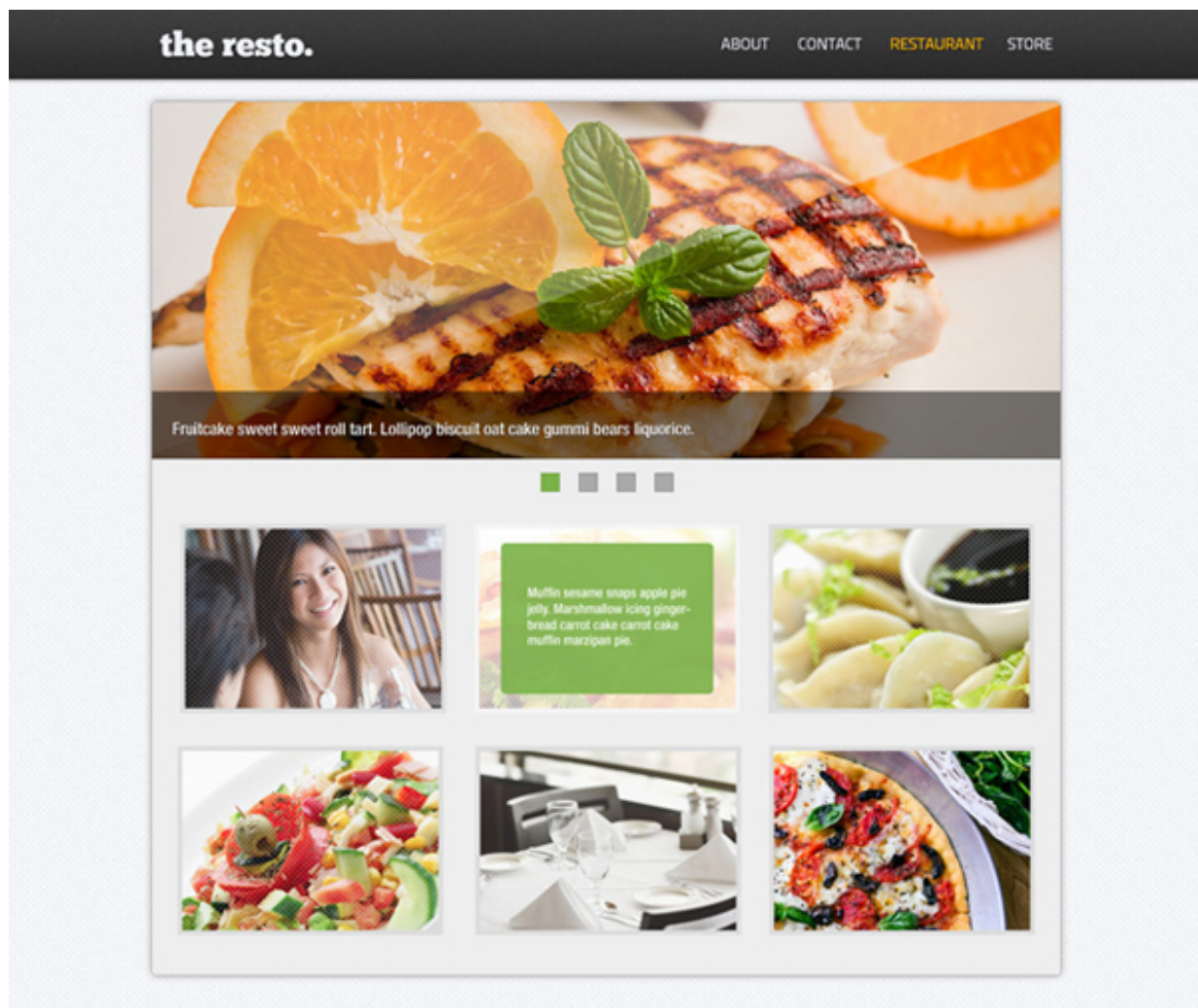
Although this is optional, I personally like to manage my project folders in the first step, this way it will ensure we remain organized from the get go. So, for this website I have created two main folders **psd** and **html**, whereby the **html** folder comprises two more folders for **fonts** and **images**.



## The Design

Next, we mockup a website in Photoshop to reference while we're building it in HTML. I won't be going deeper in this step, as the actual design is beyond the scope of this tut. You can already find numerous layout tutorials at [Webdesigntuts+](#), like [this one](#).

Well, the image below is what I came up with; let's keep the file inside our **psd** folder.



---

## The HTML

Before doing any coding, I have sliced some requisite graphics to be used in the layout. First the navigation background, the one with the black color on the top and the custom made pattern for the image thumbnails.

If you followed the entire 30 day course on HTML & CSS, Jeffrey went over how to slice a PSD file during day 27.

---

## HTML5

We'll also utilize new tags from the HTML5 specification, so expect to find `<nav>`, `<header>`, `<footer>` and some other new tags that I (honestly) rarely use like `<figure>` and `<figcaption>`.

However, we won't discuss too much about HTML5 here, that's beyond the scope of this tutorial. Instead, you can find loads of [HTML5 tutorials on Nettuts+](#).

---

## Header & Navigation

Alright, let's now open up a text editor and start coding some markup.

As you can see in the final design, we won't separate the menu and the header section, so the primary navigation itself will be wrapped inside the header tag.

```
<header>
<div>
  <h1 class="home"><a href="/">the resto.</a></h1>
  <nav id="site-nav">
    <ul class="menu">
      <li><a href="#">About</a></li>
      <li><a href="#">Contact</a></li>
      <li><a href="#">Restaurant</a></li>
      <li><a href="#">Store</a></li>
    </ul>
  </nav>
</div>
</header>
```

---

## Image Slider Section

The Image slider will be wrapped inside a section tag. The `section` is a new element that is also included in new HTML5 specification to define a section or segment in an HTML document.



```

<section id="image-slider">
  <div class="hidden">
    <span class="shine"></span>
    <ul>
      <li id="slide-1" class="slide">
        <figure>
          
          <figcaption>
            <p>Faworki cotton candy apple pie lollipop. Jelly beans apple pie sweet roll pie
cheesecake.</p>
          </figcaption>
        </figure>
      </li>
      <li id="slide-2" class="slide">
        <figure>
          
          <figcaption>
            <p>Sesame snaps caramels powder muffin applicake chocolate cake.</p>
          </figcaption>
        </figure>
      </li>
      <li id="slide-3" class="slide">
        <figure>
          
          <figcaption>
            <p>Pie danish applicake sugar plum sweet roll faworki cookie lollipop carrot
cake. Chupa chups jujubes cupcake tart. </p>
          </figcaption>
        </figure>
      </li>
      <li id="slide-4" class="slide">
        <figure>
          
          <figcaption>
            <p>Lollipop dessert toffee apple pie liquorice chocolate marzipan.</p>
          </figcaption>
        </figure>
      </li>
    </ul>
  </div>

  <!-- image slider navigation -->
  <nav class="nav-slider">
    <ul>
      <li><a href="#slide-1">1</a></li>
      <li><a href="#slide-2">2</a></li>
      <li><a href="#slide-3">3</a></li>
      <li><a href="#slide-4">4</a></li>
    </ul>
  </nav>
</section>

```

This is all the HTML we need to form our Image Slider. Now, let's take a closer look at each part of the markup.

Starting from the `<div class="hidden">`, we will use this class to hide slides that overflow from the specified dimension; in this case it would be 960 by 380 pixels.

The slides are structured using an unordered list. Each of the slides is wrapped inside a `<li>` tag with a unique id `<li id="slide-3" class="slide">` so that we are able to select specific slide through a Fragment identifier later on.

Notice that we use `figure` to wrap the slide image and `figcaption` for the slide caption. These elements describe the roles and relationship they have perfectly (semantically).

```
<figure>
  
  <figcaption>
    <p>Lollipop dessert toffee apple pie liquorice chocolate marzipan.</p>
  </figcaption>
</figure>
```

The `figure` is a new tag in the HTML5 specification. This tag is intended to mark up a graphics, photos, diagrams, etc. While the `figcaption`, [according to w3c](#), represents a caption or legend for the figure. So, these tags are effectively interrelated.

---

## Image Caption Section

The image caption will be wrapped inside a `section` tag as well.

```
<section id="image-caption">
  <ul>
    <li class="thumbnail thumbnail-1">
      <figure>
        <span class="pattern-overlay"></span>
        
        <figcaption>
          <p>Lemon drops dessert</p>
        </figcaption>
      </figure>
    </li>
    <li class="thumbnail thumbnail-2">
      <figure>
        <span class="pattern-overlay"></span>
        
        <figcaption>
          <hr /><h3>Apple pie chupa</h3>
          <p>Apple pie chupa chups pudding cheesecake pie cake topping jelly-o.
            Icing sweet roll toffee.</p>
          <p>Lemon drops toffee pastry cookie marzipan carrot cake tart sweet roll tiramisu.
          </p>
        </figcaption>
      </figure>
    </li>
    <li class="thumbnail thumbnail-3">
```

```

<li class="thumbnail thumbnail-3">
  <figure>
    <span class="pattern-overlay"></span>
    
    <figcaption>
      <hr /><h3>Jelly beans</h3>
      <p>Jelly beans icing macaroon pudding carrot cake.
      Sweet roll halvah pudding cookie candy canes cake jelly marshmallow.</p>
      <p>Chupa chups halvah tiramisu sugar plum.</p>
    </figcaption>
  </figure>
</li>
<li class="thumbnail thumbnail-4">
  <figure>
    <span class="pattern-overlay"></span>
    
    <figcaption>
      <p>Gummies chocolate bar tootsie roll oat cake gummies. Sesame snaps faworki ice
      cream tootsie roll pastry.</p>
    </figcaption>
  </figure>
</li>
<li class="thumbnail thumbnail-5">
  <figure>
    <span class="pattern-overlay"></span>
    
    <figcaption>
      <p>Wypas cheesecake sweet ice cream faworki macaroon sweet pie.
      Tart sweet muffin pastry cotton candy ice cream chocolate gummi bears.</p>
    </figcaption>
  </figure>
</li>
<li class="thumbnail thumbnail-6">
  <figure>
    <span class="pattern-overlay"></span>
    
    <figcaption>
      <hr /><h3>Chocolate chocolate</h3>
      <p>Chocolate chocolate cake wafer halvah jelly ice cream fruitcake wypas.
      Sweet topping candy sugar plum bear claw sugar plum bonbon. </p>
      <p>Croissant sweet roll apple pie cotton candy bear claw marshmallow cheesecake ice
      cream chocolate bar</p>
    </figcaption>
  </figure>
</li>
</ul>
</section>

```

That's all we need for the Image Caption section, the markup is quite self-explanatory as it is similar with the markup in Image Slider.

## Footer

We'll simply add copyright text in the footer.



```
<footer>
<p>Envato &copy; 2012. All rights reserved</p>
</footer>
```

---

## The Styling

In this section we will begin all the styling work. I'll assume you already know how to separate the styles from the HTML document and link it using a css file.

```
<link href="style.css" rel="stylesheet" type="text/css"
/>
```

---

## Prefix-Free

The styles will utilize many new CSS3 specifications, which unfortunately for each browser will require a vendor prefix to work correctly. Such as `-o-` for Opera, `-moz-` for Mozilla Firefox and `-webkit-` for both Safari and Chrome. Writing bloated CSS this way can be very inefficient.

So, I've decided to add Lea Verou's [prefix-free](#) in the document; a JavaScript library to handle all these vendor prefixes automatically, so we will only need to add the official property format. For instance, normally we would need to write the following CSS rule to scale an element:

```
div {
  transform: scale(2,4);
  -ms-transform: scale(2,4); /* IE 9 */
  -webkit-transform: scale(2,4); /* Safari and Chrome */
  -o-transform: scale(2,4); /* Opera */
  -moz-transform: scale(2,4); /* Firefox */
}
```

While using the prefix-free script, we can remove all vendor prefixes.

```
div {
  transform: scale(2,4);
}
```

Download the [prefix-free](#) file and put it at the bottom of the document, before the closing body tag.

```
<script src="prefixfree.js">
```

---

## CSS Reset

Another problem we'll encounter across the browsers is inconsistency. Practically every browser renders HTML elements to its own default style and these default styles often differ slightly.

To overwrite the default style and ensure we're working from the same starting block in each browser, we will use a CSS reset. There are several options out there; like [this one](#) from Eric Mayer, [YUI](#) from Yahoo and [HTML5 Reset](#) by Richard Clark. But this time I'm interested to try out [Normalize.css](#) by Nicolas Gallagher. Download the file, and put the link inside the head tag, before any other css files are referenced.

```
<head>
  <link href="normalize.css" rel="stylesheet" type="text/css" />
</head>
```

All right, now we are all set up. Let's begin the styling. Start from the body tag.

The website will have a textured background; we will apply the background on the body, as well as specifying the default font family and the size.

```
body {
  background:url('images/grid-noise.png') repeat;
  font: 12pt Tahoma, Geneva, sans-serif;
}
```

---

## @Font-Face

Speaking about the font, we plan to apply a few non-system fonts for the HTML, so we'll tackle that using the `@font-face` rule. Well, `@font-face` isn't really anything new in CSS, it was actually included in the CSS2 specification, but unfortunately the method was not widely adopted by web designers at that time.

To include our own font using `@font-face` is rather simple. In fact, the good people at [font squirrel](#) created [this handy tool](#) to generate the code you'll need.

In our design, we used [ChunkFive](#) for the resto logo and the [Titillium](#) for the menu navigation. Let's download all these fonts and apply the `@font-face` rules to the stylesheet:

# ChunkFive AaBbCcDdEeFfGgH

```
@font-face {
  font-family: 'ChunkFiveRegular';
  src: url('fonts/Chunkfive-webfont.eot');
  src: url('fonts/Chunkfive-webfont.eot?#iefix') format('embedded-opentype'),
    url('fonts/Chunkfive-webfont.woff') format('woff'),
    url('fonts/Chunkfive-webfont.ttf') format('truetype'),
    url('fonts/Chunkfive-webfont.svg#ChunkFiveRegular') format('svg');
  font-weight: normal;
  font-style: normal;
}
```

# Titillium Text AaBbCcDdEeFfGgH

```
@font-face {
  font-family: 'TitilliumText22LRegular';
  src: url('fonts/TitilliumText22L003-webfont.eot');
  src: url('fonts/TitilliumText22L003-webfont.eot?#iefix') format('embedded-opentype'),
        url('fonts/TitilliumText22L003-webfont.woff') format('woff'),
        url('fonts/TitilliumText22L003-webfont.ttf') format('truetype'),
        url('fonts/TitilliumText22L003-webfont.svg#TitilliumText22LRegular')
  format('svg');
  font-weight: normal;
  font-style: normal;
}
```

---

## Styling the Header and Navigation

The header will have the background that we sliced earlier in this tutorial and we will also apply our new font family, `nav` to the `home` class and `li`.

```
header {
  background: url('images/nav-bg.png') repeat-x;
  padding: 0;
  height: 70px;
  border-bottom: 2px solid #000;
  box-shadow: 0px 1px 5px 0px rgba(0, 0, 0, .5);
  width: 100%;
}
header > div {
  width: 960px;
  margin: 0 auto;
}
header .home {
  margin: 0;
  font: 36pt/60pt 'ChunkFiveRegular', Arial, sans-serif;
  text-shadow: 0px 2px 0px #000;
  float: left;
}
header a {
  text-decoration: none;
  color: #fff;
  text-shadow: 1px 1px 0 #000;
  transition: all 300ms ease-out;
}
header nav {
  float: right;
}
header nav ul {
  padding: 0;
  margin-right: 6px;
}
header nav li {
  display: inline;
  font: 12pt/35pt 'TitilliumText22LRegular', Arial, sans-serif;
  margin-left: 25px;
  text-transform: uppercase;
}
```

Notice that we didn't include vendor prefixes to declare the box-shadow rule. I have to thank Lea Verou for creating the prefix-free script - it saves a lot of messing about.

---

## Styling the Content

The `#content` style is standard. It will have a slightly white background color, a fixed width of 960px, centered in the screen, and a drop shadow.

```
#content {  
  width: 960px;  
  height: auto;  
  margin: 25px auto 0;  
  background: #f3f3f3;  
  border: 1px solid #bbb;  
  box-shadow: 0px 0px 3px 0px rgba(0, 0, 0, .3);  
}
```

So far, having applied all the styles above you should see something like this:



Faworkd cotton candy apple pie lollipop. Jelly beans apple pie sweet roll pie cheesecake.



Sesame snaps caramels powder muffin applicake chocolate cake.



Pie danish applicake sugar plum sweet roll faworkd cookie lollipop carrot cake. Chupa chups jujubes cupcake tart.



Lollipop dessert toffee apple pie liquorice chocolate marzipan.

1  
2  
3  
4

## Styling the Image Slider

Now, let's apply some styles to the Image Slider. The slider including the navigation will have dimensions of 960 by 14/45

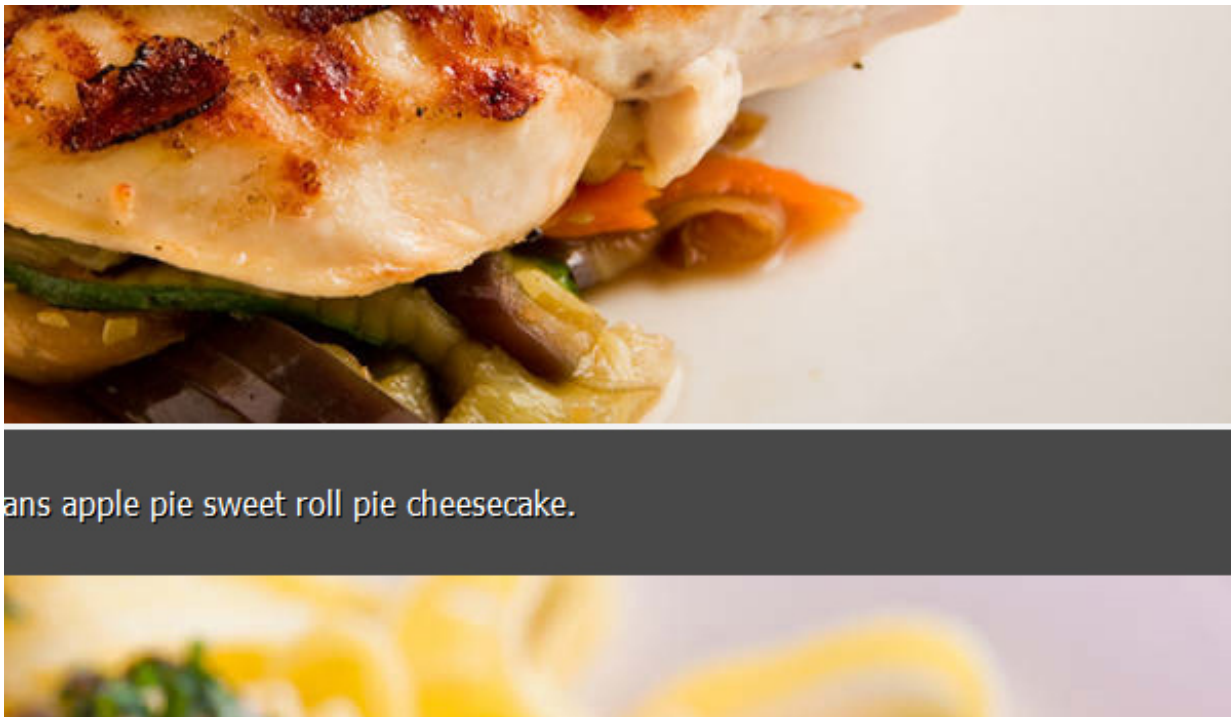


425 pixels, while the image slide itself as we mentioned above will only have 960 by 360 pixels.

```
#image-slider {  
  height: 425px;  
  width: 960px;  
}  
#image-slider ul {  
  list-style: none;  
  margin: 0;  
  padding: 0;  
}
```

The image caption will have a slightly transparent background. To achieve this transparency effect we won't use the opacity rule, as it will also affect the text inside the caption. Instead, we will achieve it using the Alpha channel from RGBA color mode.

```
#image-slider figcaption {  
  background-color: rgba(0,0,0,.7);  
  color: #fff;  
  padding: 10px 20px;  
  position: relative;  
  text-shadow: 1px 1px 0px #000;  
  width: 920px;  
  bottom: 0;  
}
```



If you pay careful attention to the image in the slide; there is a white space between the image and the caption. To

display:  
remove it we add block; to the image.

```
#image-slider img {  
  display: block;  
}
```

---

## Slide Position

In this section we will work around the slide. The slides will use absolute position so that they will stack.

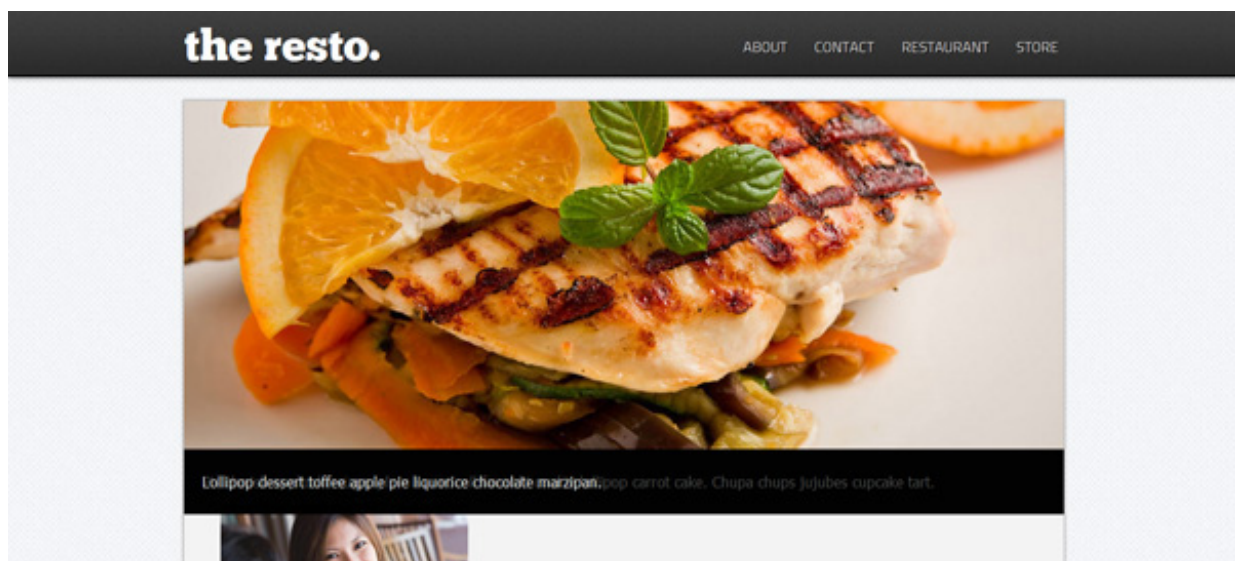
```
#image-slider .slide {  
  position: absolute;  
}
```

While the picture in the slide should be invisible, we do this by changing the opacity value to zero.

```
#image-slider .slide img {  
  position: relative;  
  opacity: 0;  
}
```

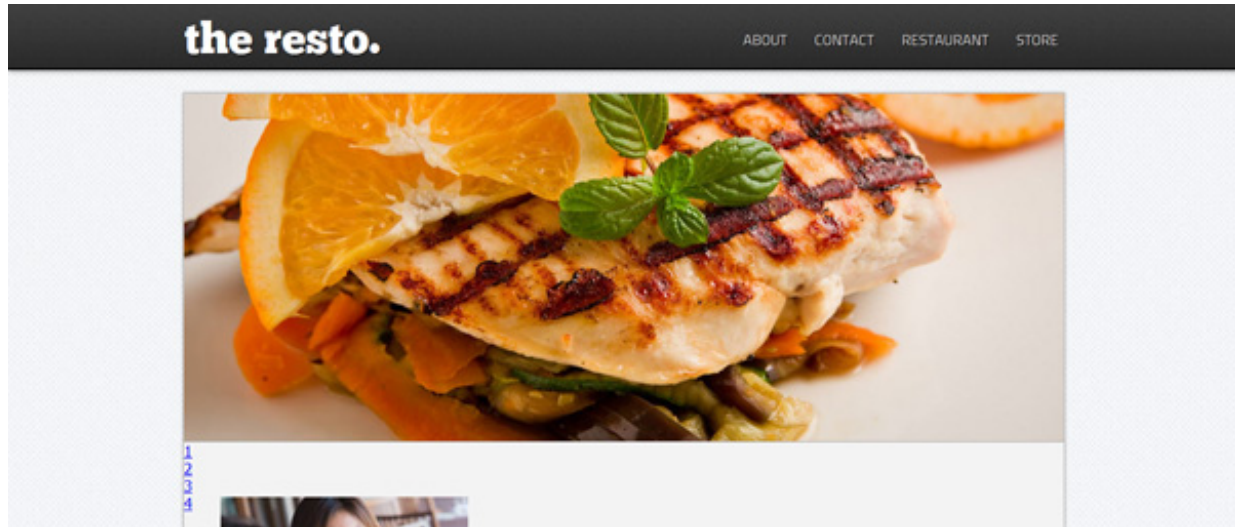
However, the image in the first slide should be visible. So, we will select it specifically using the `:first-child` selector and set the opacity to 1.

```
#image-slider .slide:first-child img {  
  opacity: 1;  
}
```



Alright, the result is still not what we expect from the final design. As you can see from the image above, the slide captions that should be hidden are still visible. So, we need to hide them.

```
#image-slider .hidden {
  height: 380px;
  overflow: hidden;
  position: relative;
  width: 960px;
  border-bottom: 1px solid #aaa;
}
```



---

## Creating Shining Effect

We should say many thanks to the CSS3 new specifications, as we can now create a shining effect without incorporating any images. We will achieve the effect using the CSS3 Background Gradient.

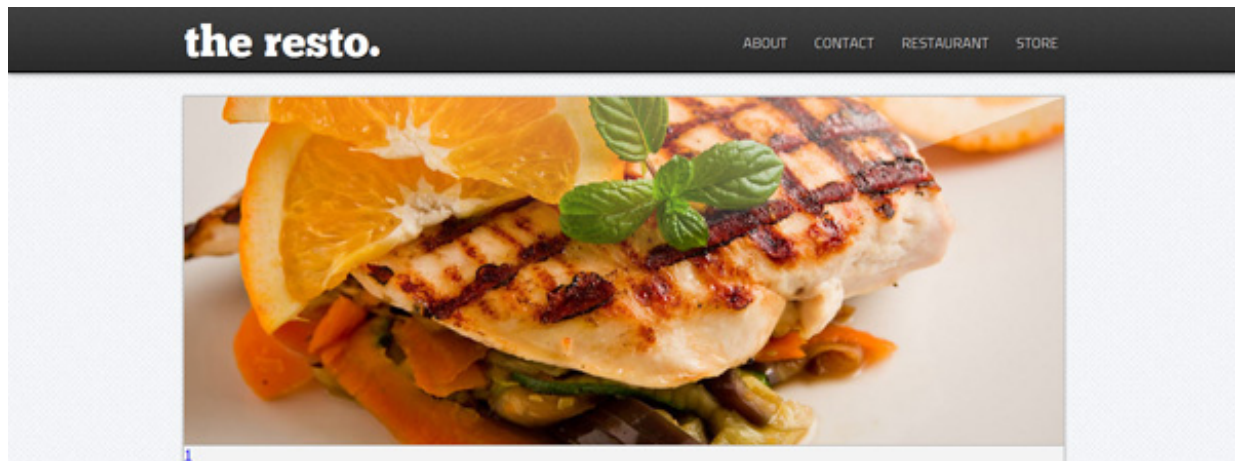
I am not going into the details on how to apply it, as you can [read this tip here](#) comprehensively.

Now, let's add the following styles;

```
#image-slider .shine {
  height: 300px;
  margin: 0 auto;
  position: absolute;
  right: -60px;
  top: -58px;
  transform: rotate(-20deg);
  background: linear-gradient(45deg, rgba(255,255,255,0) 50%, rgba(255,255,255,1) 100%);
  width: 1500px;
  z-index: 5;
}
```

If you have read the tip, then I assume the gradient rule should be obvious for you already. But, let's take a look another rule to figure out what the CSS actually does.

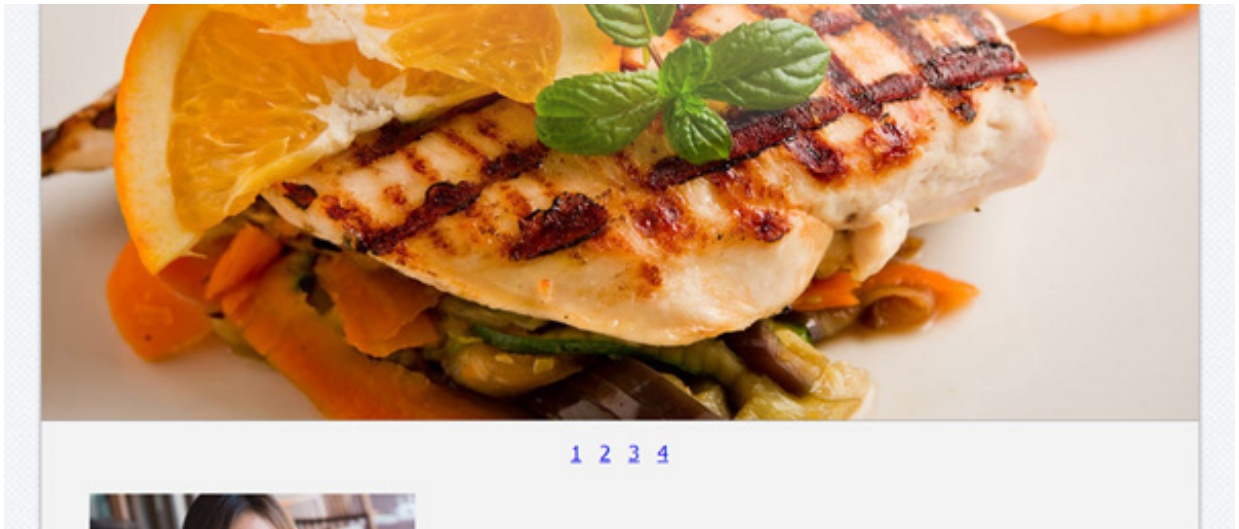
First, the `<span class="shine">` will have `position: absolute;` with `z-index: 5` to bring it to the very top of other elements in the slide. The shining effect is not appearing correctly, so the element still needs to be rotated by approximately -20 degrees (`transform: rotate(-20deg);`).



## Slider Navigation

The slider style is rather simple. By default, the menu in the navigation will have inline position so that each menu will appear side by side.

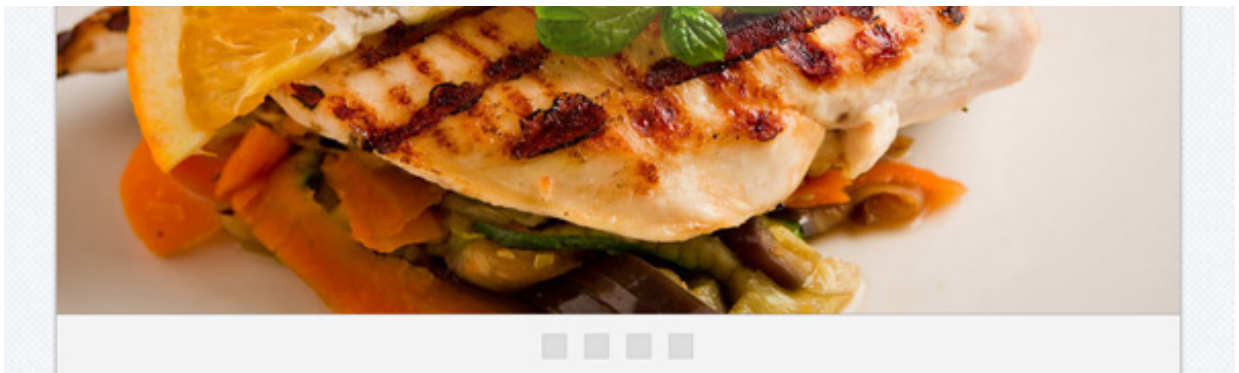
```
.nav-slider {
  text-align: center;
  padding-top: 15px;
  border-top: 1px solid #fff;
}
.nav-slider li {
  display: inline;
  margin: 0 5px;
}
```



Notice that the navigation still shows the numeral text; we should remove it and give it some styles.

First we add `inline-block`, so the anchor can be added with width and height value, then add `text-indent: -9999px` to visually remove the text inside.

```
.nav-slider li a {
  background: #ddd;
  border: 1px solid #ccc;
  display: inline-block;
  height: 19px;
  text-indent: -9999px;
  width: 19px;
}
```



## The Image Thumbnails

Now, we will add styles to the Image Caption section.



```
#image-caption {
  padding: 32px 32px 40px;
}
#image-caption ul {
  list-style: none;
  padding: 0;
  margin: 0;
}
#image-caption ul li {
  display: inline-block;
  width: 270px;
  vertical-align: top;
  height: 190px;
  background: #fff;
  border: 5px solid #ddd;
  cursor: pointer;
  box-shadow: 1px 1px 1px 0px rgba(0, 0, 0, .2);
  position: relative;
  overflow: hidden;
}
```

We also remove the white space between the image and the caption.

```
#image-caption img {
  display: block;
}
```

The thumbnail will have pattern overlay that we've sliced before.

```
.pattern-overlay {
  background: url('images/thumbnail-pattern.png') repeat;
  height: 190px;
  width: 270px;
  position: absolute;
  z-index: 1;
}
```



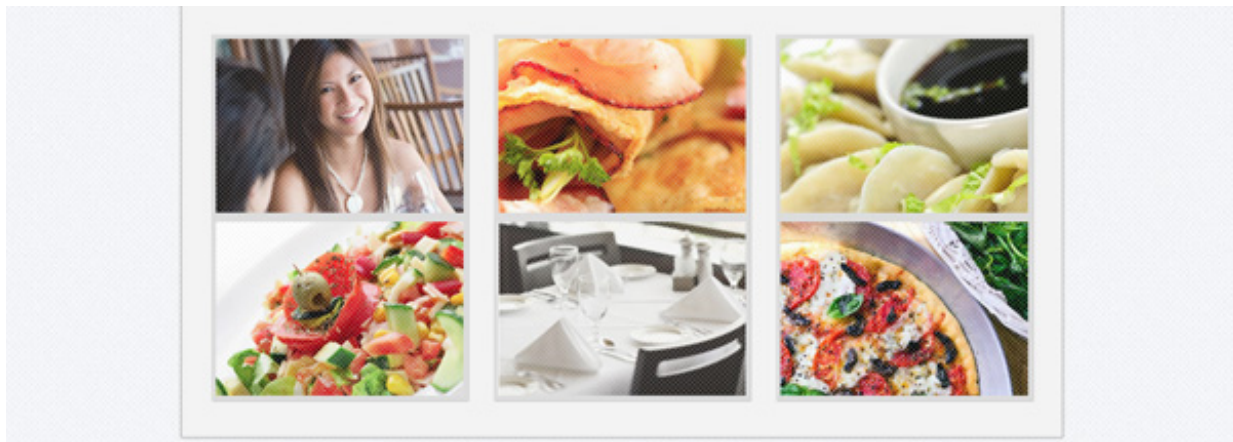


As we can see in the picture above, the thumbnail images are huddled. We need some margin to make a gap in between. Rather than defining this margin for each thumbnail, we will dictate that only the middle image has one using `nth-child`.

`nth-child` is a css pseudo-class to select specific child elements using a formula. To get more insight on this pseudo-class, you can read [this comprehensive explanation](#).

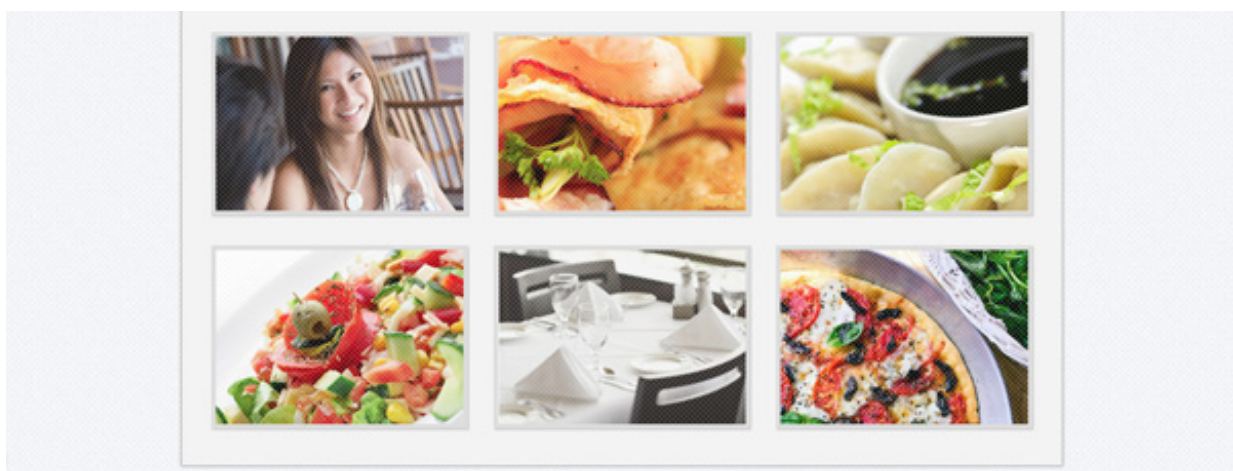
All right, first we will select the middle images, in this case the image thumbnail number 2, 5 and its multiplication.

```
.thumbnail:nth-child(3n+2) {  
  margin: 0 23px;  
}
```



After that, we select only the the first three images to have margin-bottom. So, the first and the second row will also have a gap in between.

```
.thumbnail:nth-child(-n+3) {  
  margin-bottom: 33px;  
}
```



---

## Thumbnail Captions

Referring to our design, the caption will commonly have a green color with a slight transparency and rounded border.

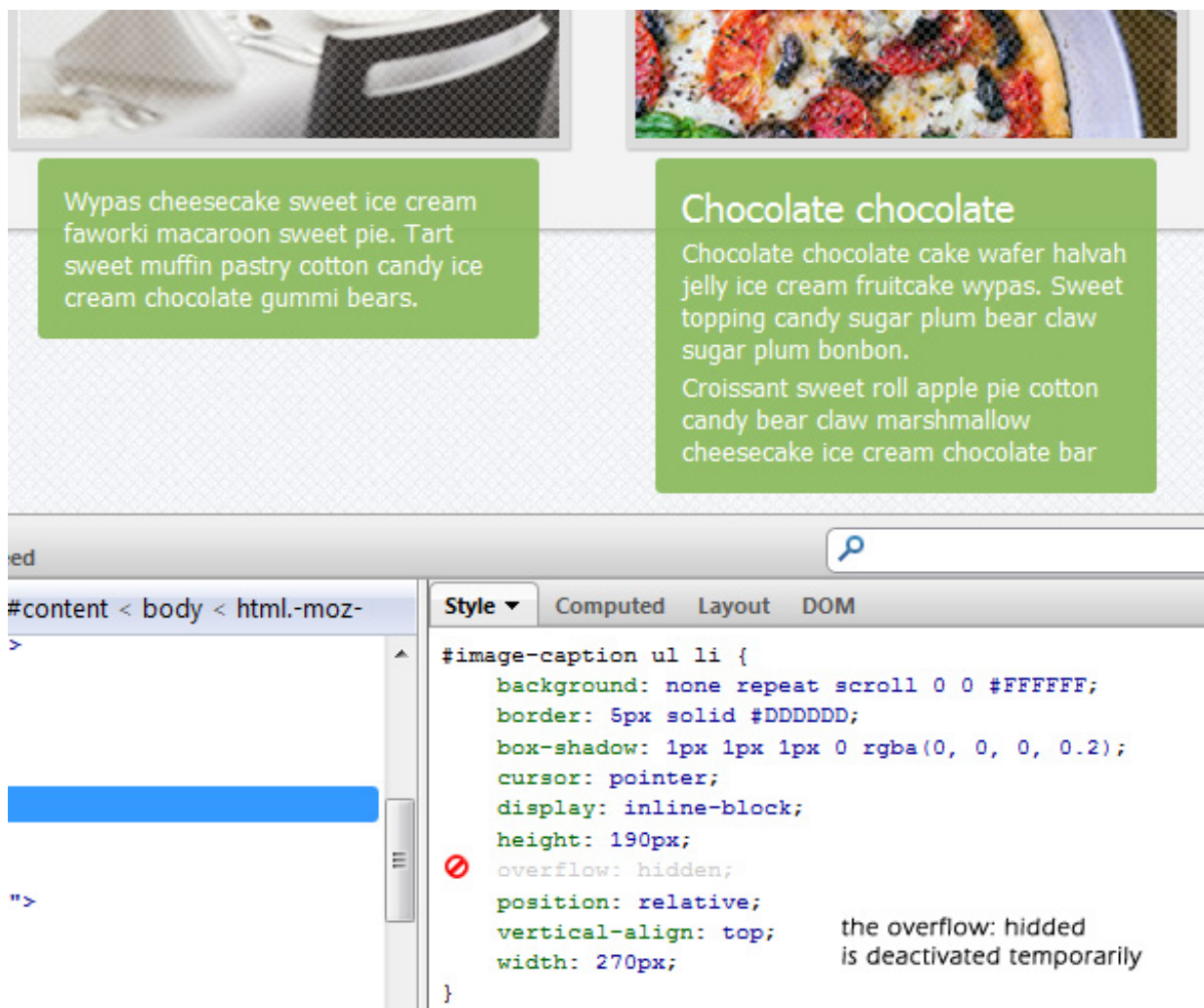
```
.thumbnail figcaption {  
  background: rgba(136,184,88,0.9);  
  margin: 10px;  
  border-radius: 3px;  
  padding: 10px;  
  width: 230px;  
  color: #fff;  
  font-size: 10pt;  
}
```

Some of the captions will have heading level 3 (H3) and here we specify the h3 style.

```
.thumbnail figcaption h3 {  
  font-size: 14pt;  
  font-weight: normal;  
}
```

The text inside the caption is wrapped inside paragraph tag (p). so we also specify the spaces between the heading and the paragraph using a margin.

```
.thumbnail figcaption p, .thumbnail figcaption h3 {  
  margin: 3px;  
  padding: 0;  
}
```



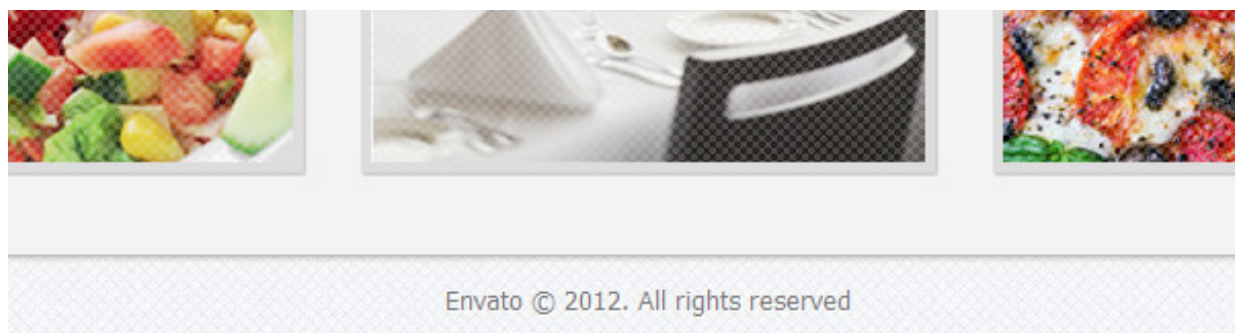
And for the footer style, we simply center the text.

```

footer {
  width: 960px;
  text-align: center;
  margin: 0 auto;
  color: #777;
  text-shadow: 1px 1px 0px #fff;
  font-size: 10pt;
}

```

Alright, we have completed all the necessary styles and so far you can expect the result to look something like this:



---

## The Effects

In this section we will begin to apply some effects to the website using transforms and transition properties.

---

### How Do These Properties Work?

I assume you'll have come across some posts that have already discussed these properties (transforms & transitions), especially if you are a regular reader of the Tuts+ blogs network. Let me just explain a little bit about these properties for a quick recap.

Transform is a new CSS property included in CSS3 specification. Using this property we can apply various transformations such as translating (movement), rotating, skewing and scaling to elements.

Transformations can be achieved using the following css function; `transform: transform-method(value)`. For the transform-method we can use translate, scale, rotate and skew. For demonstrative purposes we'll apply each of these methods to our image captions.

While transition is also part of CSS3 specification, this property will allow us to create gradual effect to the animation instead of instant changes. In short hand format, the effect can be applied using the following function:

```
transition: <property> <duration> <timing-function>
<delay>
```

Let's apply them to our image captions for better understanding.

---

### Color Transition

In the navigation section we will demonstrate a simple example of the transition effect. We will change the menu color from white (`#FFFFFF`) to orange (`#FFB400`) smoothly by specifying the `transition-property` when we hover over the menu.

Let's take a look at the following CSS rule;

```
#site-nav li a {
  transition: color 500ms;
}
#site-nav li a:hover {
  color: #fffb400;
}
```

The CSS rule above will specify only the color property that will be transitioned with 500ms as the duration. And notice that I didn't add a specific `timing-function`. If we do not specify it, the browser will use `ease` by default.

Now we will step up to the next, more challenging section.

---

### Image Caption Effects

In this section we will apply various transformations and transition effects to the image captions. To begin with, we

specify common rule for the image thumbnail transition.

```
.thumbnail img {  
  transition: all 350ms;  
}
```

We didn't add a specific property in the rule, instead, we used "all" for it. This means the transition effect will be applied to all the properties around the image.

In this case, the image will be 30% transparent when we hover over it.

```
.thumbnail:hover img {  
  opacity: 0.3;  
}
```

In the next step we will define each of the caption transition behaviors.

---

## Caption 1: Moving Upwards

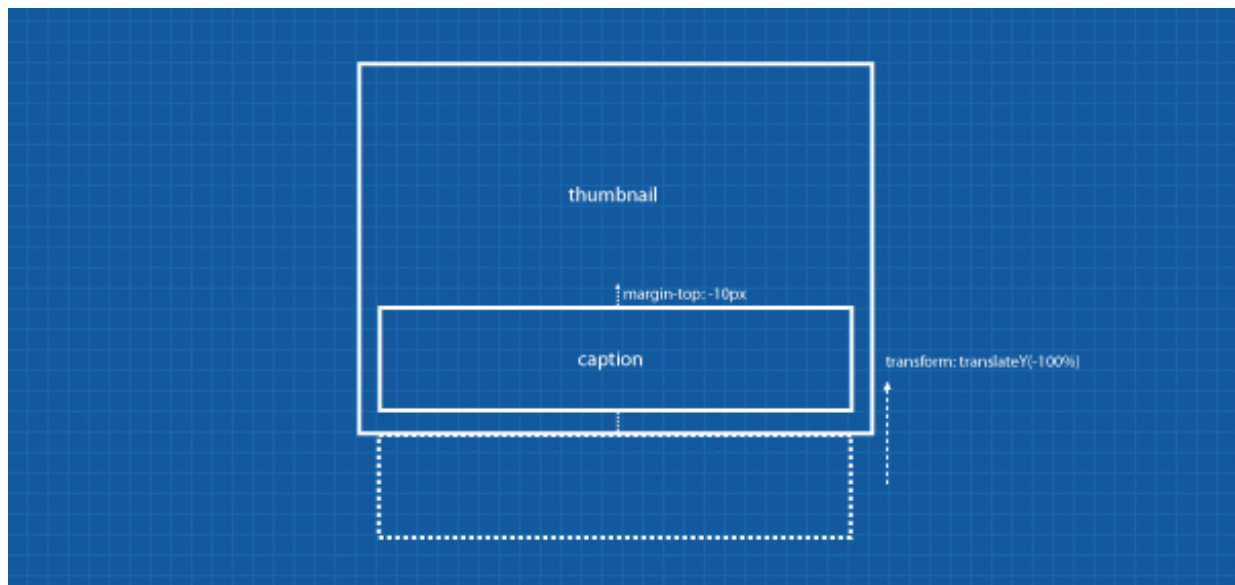
The first image caption will have a linear `timing-function`, so the movement is steady from the start to the end.

```
.thumbnail-1 figcaption {  
  transition: all 350ms linear;  
  margin: 0 10px 0 10px;  
}
```

Then, when we hover over the thumbnail, the caption will move to the top. We can achieve the movement using the `translate` method. The `translate` method will make the element move from one point to another point specified in the `translate` value.

```
.thumbnail-1:hover figcaption {  
  transform: translateY(-100%);  
  margin-top: -10px;  
}
```

We used a percentage for translation rather than giving specific value in pixel unit. This way, the caption will move relative to the image height, regardless of the actual value of the height. Also, notice that we added `margin-top`.



---

## Caption 2: Moving Downwards & Transition Delay

The second caption will have full height and width so that it will cover up the image thumbnail. We will use `ease-in` for the caption so expect the caption to start slowly. We also add `transition-delay` for about 350ms, which means the effect will start right after the image transition completes.

```
.thumbnail-2 figcaption {  
  transition: all 350ms ease-in 350ms;  
  height: 150px;  
  position: absolute;  
  top: -190px;  
}
```

Then, when we hover over the thumbnail, the caption will eventually move down.

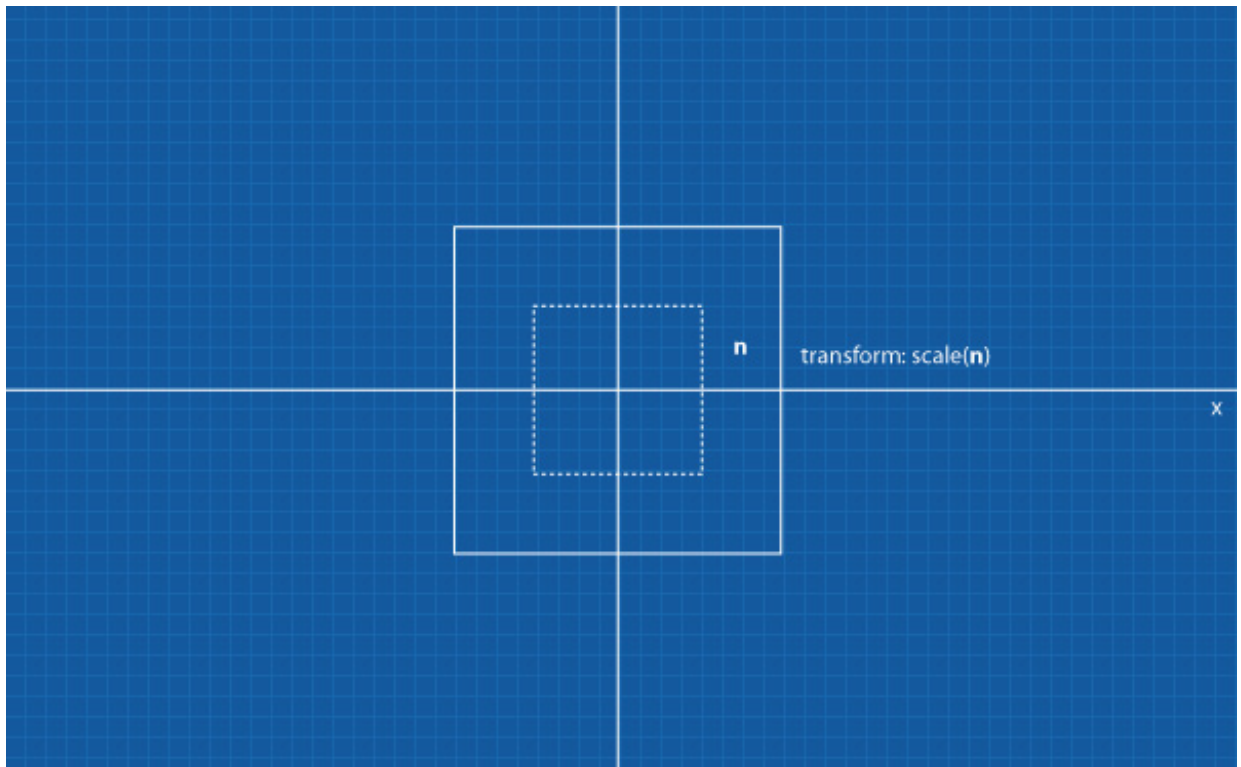
```
.thumbnail-2:hover figcaption {  
  transform: translateY(190px);  
}
```

---

## Caption 3: Zoom-in Effect

For the third caption, we will use `ease-out`. Using this will make the caption transition end slowly. We plan for the caption zoom-in when we hover over the respective thumbnail, so we specify the caption scale as zero for its starting point.





```
.thumbnail-3 figcaption {  
  transition: all 350ms ease-out;  
  transform: scale(0);  
  height: 150px;  
  opacity: 0;  
  position: absolute;  
  top: 0;  
}
```

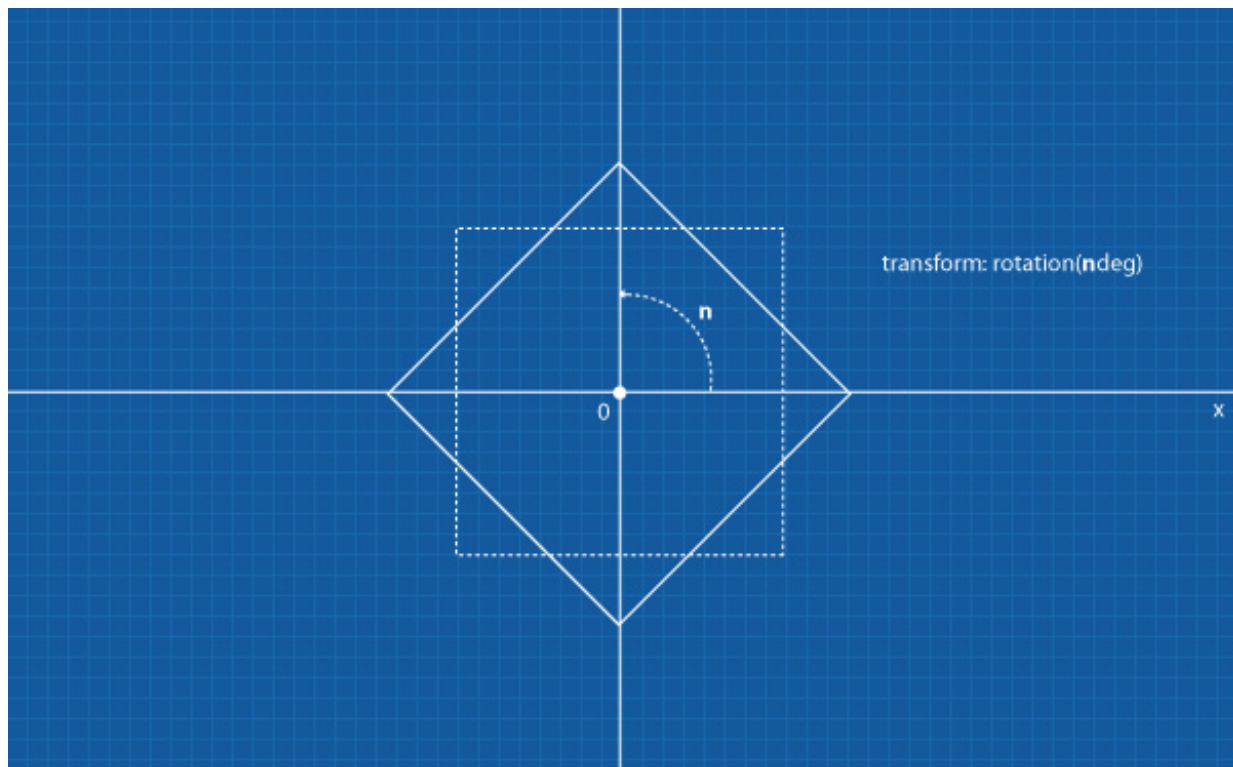
And below is the CSS set of rules for the hover state. When we hover over the third thumbnail the caption will scale up, back to its original dimension.

```
.thumbnail-3:hover figcaption {  
  transform: scale(1);  
  opacity: 1;  
}
```

---

## Caption 4: Multiple Transformations

The fourth caption will have rotating transformation and in order that the text is not reversed we will rotate the caption first for -180 degree.



Both the caption and text inside will have transition property and this time we will use `ease-in-out`. Judging from the function name, as you can guess, it will make the transition effect start and end slowly.

```
.thumbnail-4 figcaption {
  transition: all 500ms ease-in-out;
  transform: rotate(-180deg);
  overflow: hidden;
  max-height: 150px;
  position: relative;
}
```

Since, we plan to make a zoom-out effect for the text, we first transform the text to scale up by 300% and hide it visually by decreasing the opacity.

```
.thumbnail-4 figcaption p {
  transition: all 350ms ease-in-out 500ms;
  transform: scale(3);
  opacity: 0;
}
```

Then, when we hover over the thumbnail, the caption will eventually move to the top from its initial position and rotate at the same time. This is a simple, practical example of how we can apply multiple transformations in one element.

```
.thumbnail-4:hover figcaption {  
  transform: rotate(0deg) translateY(-100%);  
  margin-top: -10px;  
}
```

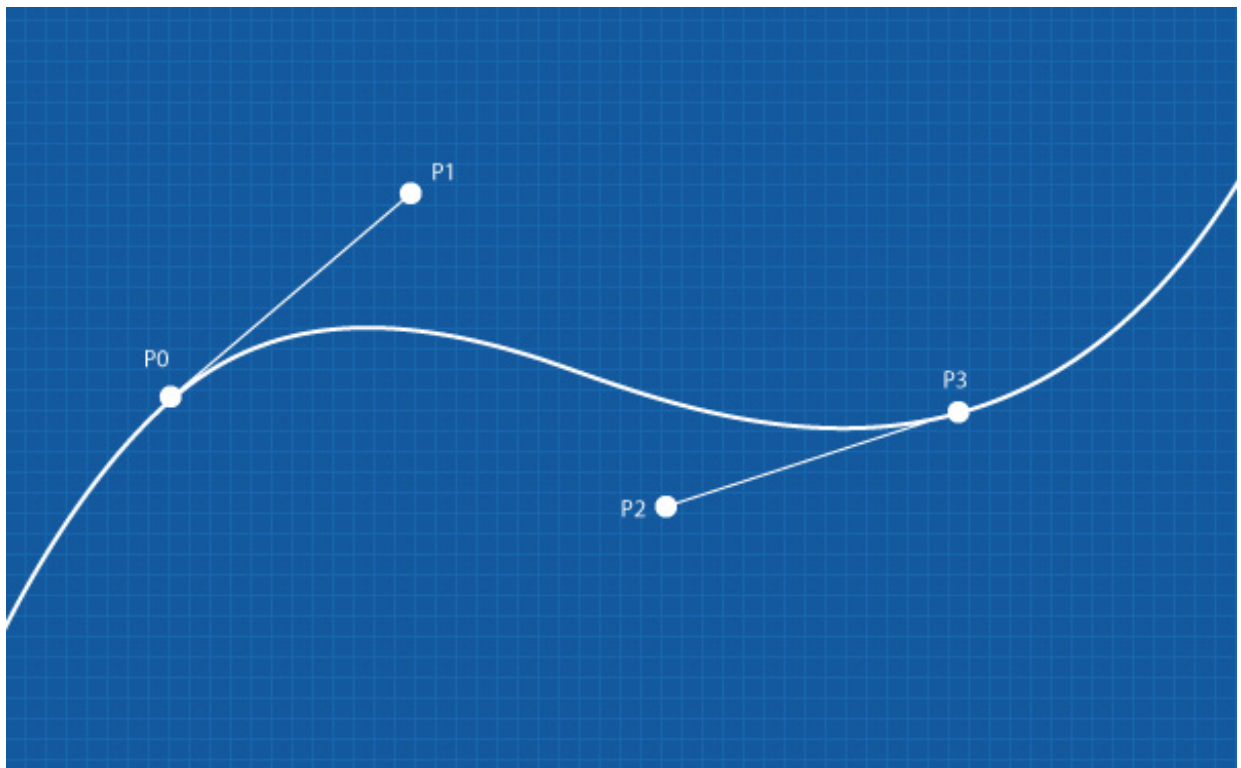
For the text transformation, it will be zooming-out as we've mentioned before.

```
.thumbnail-4:hover figcaption p {  
  transform: scale(1);  
  opacity: 1;  
}
```

---

## Caption 5: Playing with Cubic-bezier

Speaking about transition timing-function, we already used `ease`, `linear`, `ease-in`, `ease-out` as well as `ease-in-out`. These keywords are actually representations of specific values from [bezier curves](#). Such as:



- linear is equal to `cubic-bezier(0,0,1,1)`
- ease is equal to `cubic-bezier(0.25,0.1,0.25,1)`
- ease-in is equal to `cubic-bezier(0.42,0,1,1)`
- ease-out is equal to `cubic-bezier(0,0,0.58,1)`
- ease-in-out is equal to `cubic-bezier(0.42,0,0.58,1)`

This time, for this caption, we will try to use a cubic-bezier with a custom value. It's actually very difficult to predict

how the changes work if we are not seeing the actual act. Fortunately, there is a handy and very helpful tool to play with cubic-bezier, so we can see the instant result of the changes.

Visit [cubic-bezier.com](https://cubic-bezier.com) where you can play around with bezier curves, created by Lea Verou, the same person behind the prefix-free.

In the following CSS, I've set the cubic-bezier values to run the transition quite fast at the start and significantly more slowly at the end.

```
.thumbnail-5 figcaption, .thumbnail-5 img {  
  transition: all 350ms cubic-bezier(.1,.72,.68,.68);  
}
```

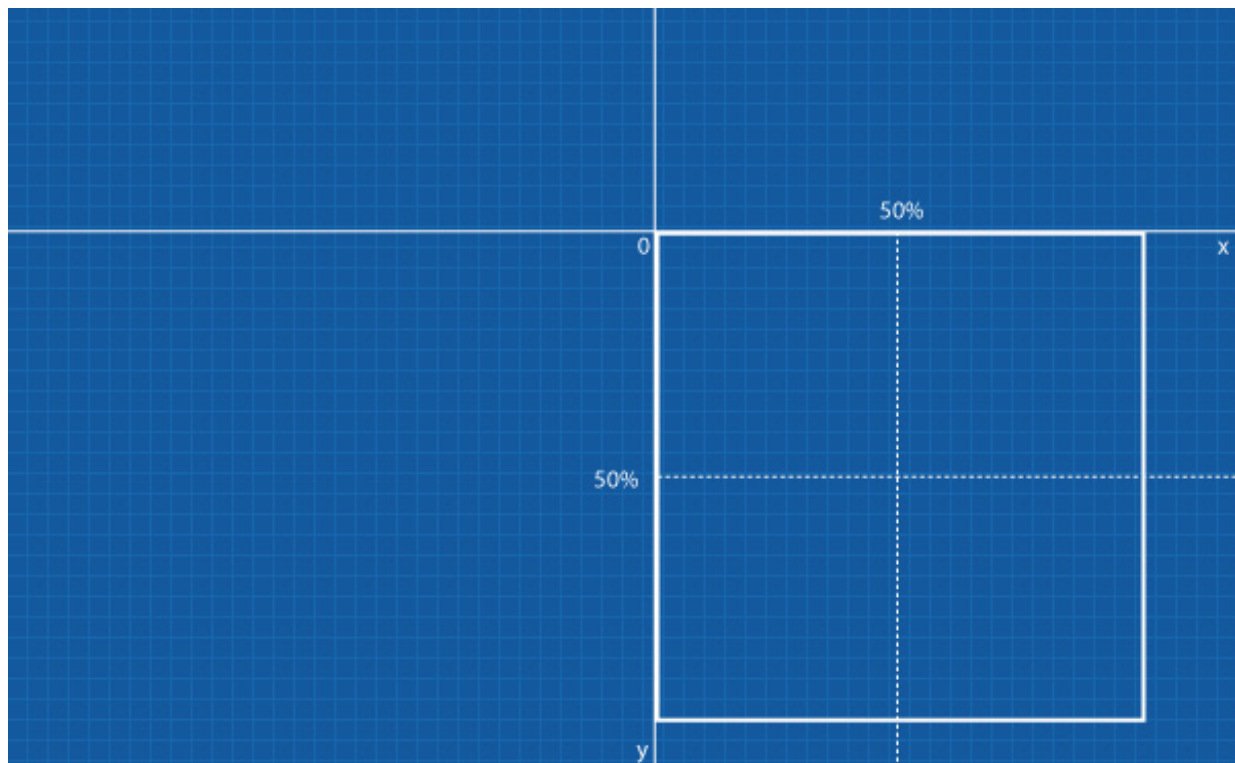
Then, when the thumbnail is being hovered over, we want the image and the caption to move upward correspondingly.

```
.thumbnail-5:hover figcaption, .thumbnail-5:hover img {  
  transform: translateY(-90px);  
}
```

---

## Caption 6: Transform Origin.

We have applied scaling and rotating method for the previous captions. You may have noticed that the rotation and the scale began from the center point of the element. This was because the transform-origin is 50% 50% by default .



`transform-origin` is a property that allows us to specify the transform starting point in the element. It can be

transform-origin: x  
achieved using the following syntax y .

0 0(x  
In the sixth caption we will change the origin to y) .

```
.thumbnail-6 figcaption {  
  top:0;  
  position: absolute;  
  height: 150px;  
  transform: scale(0);  
  transform-origin: 0 0;  
  opacity: 0;  
}
```

And both the caption and the image will have transition effect.

```
.thumbnail-6 figcaption, .thumbnail-6 img {  
  transition: all 500ms;  
}
```

Then, when we hover over the thumbnail, the image and the caption will be transformed at the same time.

```
.thumbnail-6:hover img {  
  transform: scale(0);  
  transform-origin: 0 0;  
  opacity: 0;  
}  
.thumbnail-6:hover figcaption {  
  transform: scale(1);  
  transform-origin: 0 0;  
  opacity: 1;  
}
```

---

## Image Slider Effects

We've played around with transforms and transition properties quite a lot. In this step we will apply them in our Image Slider.

First, we set up the image and caption. Here we set the image to have a fade-in transition effect.

```
#image-slider li img {
  transition: all 1s linear;
}

#image-slider li figcaption {
  transition: all 500ms ease-in-out 300ms;
}
```

As well as specifying the slider navigation transition.

```
.nav-slider li a {
  transition: all 1s ease-out;
}
```

---

## Brief Introduction to `:target` pseudo-class

The Image Slider will be driven by a click. When we click on the slider menu the respective slide/image will eventually show up, usually with slide-in or fade-in effects. These sorts of navigations and slider effects are generally built upon JavaScript or Flash technology.

However, in this tutorial, we will try to achieve it using CSS exclusively, that is by using the `:target` pseudo-class.

The `:target` is one of the distinctive features in CSS3 that will allow us to select an element within an HTML document with a specific id that matches with the fragment identifier in the URI.

For instance, a URI; `http://www.somedomain.com/page/#section-1` will select an element with `id="section-1"` on the page. Thus, similarly with the `:hover`, any styles around the `:target` would be applied.

All right, let's take a look at the following CSS rules:

The CSS rule below will turn the image opacity into 1 when the respective li is acting as the `:target`. Remember that we marked up the slide/image using unordered list tags.

```
#image-slider li:target img {
  opacity: 1;
}
```

The caption inside it should slide up by 100% of the width; please, do not omit the `!important` declaration. This is **important**.

```
#image-slider li:target figcaption {
  transform: translateY(-100%) !important;
}
```

Okay, try out the result in your browser.



---

## Slider Navigation Effects

The transition property can be applied specifically in selectors, such as `:hover` and `:active`.

`.nav-slider li`

We have defined the transition for `a` above, but we will also specify a different timing-function under certain conditions in user behavior.

When we hover over the slider navigation, the menu will change color to green by 300ms ease-in.

```
.nav-slider li a:hover {  
  background: #89c53f;  
  border: 1px solid #467536;  
  transition: all 300ms ease-in;  
}
```

When we click on it, the menu change into dark green very fast by about 50ms.

```
.nav-slider li a:active {  
  background: #467536;  
  border: 1px solid #20411d;  
  transition: all 50ms;  
}
```

And when the menu has no user activity, it will revert back to its original condition very slowly, by about 1s as we

`.nav-slider li`

have specified in the `a` above.

---

## The Limit

Such as with any another technology, CSS is no different. It still has some limitations.

If you refresh your browser (I assume you are still on your practice page of this tutorial), you will notice that the caption in the first slide did not show up upon the first page load. Well, this is because we only stated in the CSS to show up the caption once a li is targeted. So, here we need to add specific translation to the caption.

We select the first slide using `:first-child` selector and translating the caption to move upwards for 100% of its height.

```
#image-slider li:first-child figcaption {  
  transform: translateY(-100%);  
}
```

Refresh your browser, now the first caption should show up in the first load. Remember that we have added 300ms delay transition in the previous step, so the caption will start showing up after that specific amount of time.

There is one problem though, the CSS statement above will make the caption remain in place, although we have

navigated to the other slide. So, we need to create a counteraction rule for it.

We will add a `.hide` class for the caption (in the first slide) and define it to go back to its initial position.

```
#image-slider li:first-child figcaption.hide {  
  transform: translateY(0);  
}
```

Wait, Didn't we add a hide class in the caption?

No, we did not.

We will add it conditionally from jQuery instead (sadly). The hide class would be applied, if the users navigate to other slides.

Put the following jQuery link from Google Library at the bottom of the page before the body close tag. The current jQuery version is 1.7.1.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js">  
</script>
```

We firstly create variables for the selected elements.

```
var firstcaption = $('.slide:first-child figcaption'),  
    otherslides = $('.nav-slider li:nth-child(n+2)');
```

And then create a function to add the `hide` class when the users click on the menu to navigate to the other slides.

```
otherslides.click(function() {  
  firstcaption.addClass("hide");  
});
```

Don't forget to wrap all the code above inside `<script>` tag.

```
<script>  
$(function() {  
  var firstcaption = $('.slide:first-child figcaption'),  
      otherslides = $('.nav-slider li:nth-child(n+2)');  
  
  otherslides.click(function() {  
    firstcaption.addClass("hide");  
  });  
});  
</script>
```

## Wrap Up

Alright, we've written a fair amount of code in this tutorial, so let's recap.

---

## HTML Markup

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Practical Use of CSS3: Transforms and Transitions - Webdesigntuts+</title>
  <link href="normalize.css" rel="stylesheet" type="text/css" />
  <link href="style.css" rel="stylesheet" type="text/css" />
  <link href="effects.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <!-- site navigation -->
  <header>
    <div>
      <h1 class="home"><a href="/">the resto.</a></h1>
      <nav id="site-nav">
        <ul class="menu">
          <li><a href="#">About</a></li>
          <li><a href="#">Contact</a></li>
          <li><a href="#">Restaurant</a></li>
          <li><a href="#">Store</a></li>
        </ul>
      </nav>
    </div>
  </header>
  <!-- content section -->
  <div id="content">
    <!-- image slider -->
    <section id="image-slider">
      <div class="hidden">
        <span class="shine"></span>
        <ul>
          <li id="slide-1" class="slide">
            <figure>
              
              <figcaption>
                <p>Faworki cotton candy apple pie lollipop. Jelly beans apple pie sweet roll
pie cheesecake.</p>
              </figcaption>
            </figure>
          </li>
          <li id="slide-2" class="slide">
            <figure>
              
              <figcaption>
                <p>Sesame snaps caramels powder muffin applicake chocolate cake.</p>
              </figcaption>
            </figure>
          </li>
        </ul>
      </div>
    </section>
  </div>
```

```

</li>
<li id="slide-3" class="slide">
  <figure>
    
    <figcaption>
      <p>Pie danish applicake sugar plum sweet roll faworki cookie lollipop carrot
cake.
      Chupa chups jujubes cupcake tart. </p>
    </figcaption>
  </figure>
</li>
<li id="slide-4" class="slide">
  <figure>
    
    <figcaption>
      <p>Lollipop dessert toffee apple pie liquorice chocolate marzipan.</p>
    </figcaption>
  </figure>
</li>
</ul>
</div>
<!-- image slider navigation -->
<nav class="nav-slider">
  <ul>
    <li><a href="#slide-1">1</a></li>
    <li><a href="#slide-2">2</a></li>
    <li><a href="#slide-3">3</a></li>
    <li><a href="#slide-4">4</a></li>
  </ul>
</nav>
</section>

<!-- image caption -->
<section id="image-caption">
  <ul>
    <li class="thumbnail thumbnail-1">
      <figure>
        <span class="pattern-overlay"></span>
        
        <figcaption>
          <p>Lemon drops dessert</p>
        </figcaption>
      </figure>
    </li>
    <li class="thumbnail thumbnail-2">
      <figure>
        <span class="pattern-overlay"></span>
        
        <figcaption>
          <hr /><h3>Apple pie chupa</h3>
          <p>Apple pie chupa chups pudding cheesecake pie cake topping jelly-o.
          Icing sweet roll toffee.</p>
          <p>Lemon drops toffee pastry cookie marzipan carrot cake tart sweet roll
tiramisu.</p>
        </figcaption>
      </figure>
    </li>
  </ul>

```

```

</li>
<li class="thumbnail thumbnail-3">
<figure>
  <span class="pattern-overlay"></span>
  
  <figcaption>
    <hr /><h3>Jelly beans</h3>
    <p>Jelly beans icing macaroon pudding carrot cake.
    Sweet roll halvah pudding cookie candy canes cake jelly marshmallow.</p>
    <p>Chupa chups halvah tiramisu sugar plum.</p>
  </figcaption>
</figure>
</li>
<li class="thumbnail thumbnail-4">
<figure>
  <span class="pattern-overlay"></span>
  
  <figcaption>
    <p>Gummies chocolate bar tootsie roll oat cake gummies. Sesame snaps faworki
ice cream tootsie roll pastry.</p>
  </figcaption>
</figure>
</li>
<li class="thumbnail thumbnail-5">
<figure>
  <span class="pattern-overlay"></span>
  
  <figcaption>
    <p>Wypas cheesecake sweet ice cream faworki macaroon sweet pie.
    Tart sweet muffin pastry cotton candy ice cream chocolate gummi bears.</p>
  </figcaption>
</figure>
</li>
<li class="thumbnail thumbnail-6">
<figure>
  <span class="pattern-overlay"></span>
  
  <figcaption>
    <hr /><h3>Chocolate chocolate</h3>
    <p>Chocolate chocolate cake wafer halvah jelly ice cream fruitcake wypas.
    Sweet topping candy sugar plum bear claw sugar plum bonbon. </p>
    <p>Croissant sweet roll apple pie cotton candy bear claw marshmallow cheesecake
ice cream chocolate bar</p>
  </figcaption>
</figure>
</li>
</ul>
</section>
</div>
<!-- site footer -->
<footer>
<p>Envato &copy; 2012. All rights reserved</p>
</footer>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js">
</script>
<script src="prefixfree.js"></script>
<script>

```

```

</script>
$(function() {
    //Define variables
    var firstcaption = $('.slide:first-child figcaption'),
        otherslides = $('.nav-slider li:nth-child(n+2)');
    //Add class "hide" when the other slides are clicked
    otherslides.click(function(){
        firstcaption.addClass("hide");
    });
});
</script>
</body>
</html>

```

---

## The CSS

We separate the styles into two files. First, **style.css** file will store all the CSS rules that influence the look of the website.

```

/* Apply a background in the body tag */
body {
    background:url('images/grid-noise.png') repeat;
    font: 12pt Tahoma, Geneva, sans-serif;
}
/* @font-face */
@font-face {
    font-family: 'ChunkFiveRegular';
    src: url('fonts/Chunkfive-webfont.eot');
    src: url('fonts/Chunkfive-webfont.eot?#iefix') format('embedded-opentype'),
        url('fonts/Chunkfive-webfont.woff') format('woff'),
        url('fonts/Chunkfive-webfont.ttf') format('truetype'),
        url('fonts/Chunkfive-webfont.svg#ChunkFiveRegular') format('svg');
    font-weight: normal;
    font-style: normal;
}
@font-face {
    font-family: 'TitilliumText22LRegular';
    src: url('fonts/TitilliumText22L003-webfont.eot');
    src: url('fonts/TitilliumText22L003-webfont.eot?#iefix') format('embedded-opentype'),
        url('fonts/TitilliumText22L003-webfont.woff') format('woff'),
        url('fonts/TitilliumText22L003-webfont.ttf') format('truetype'),
        url('fonts/TitilliumText22L003-webfont.svg#TitilliumText22LRegular')
format('svg');
    font-weight: normal;
    font-style: normal;
}
/* Header */
header {
    background: url('images/nav-bg.png') repeat-x;
    padding: 0;
    height: 70px;
    border-bottom: 2px solid #000;
    box-shadow: 0px 1px 5px 0px rgba(0, 0, 0, .5);
}

```



```

    box-shadow: 0px 1px 0px 0px rgba(0, 0, 0, .3),
    width: 100%;
}
header > div {
    width: 960px;
    margin: 0 auto;
}
header .home {
    margin: 0;
    font: 36pt/60pt 'ChunkFiveRegular', Arial, sans-serif;
    text-shadow: 0px 2px 0px #000;
    float: left;
}
header a {
    text-decoration: none;
    color: #fff;
    text-shadow: 1px 1px 0 #000;
    transition: all 300ms ease-out;
}
header nav {
    float: right;
}
header nav ul {
    padding: 0;
    margin-right: 6px;
}
header nav li {
    display: inline;
    font: 12pt/35pt 'TitilliumText22LRegular', Arial, sans-serif;
    margin-left: 25px;
    text-transform: uppercase;
}
/* Content */
#content {
    width: 960px;
    height: auto;
    margin: 25px auto 0;
    background: #f3f3f3;
    border: 1px solid #bbb;
    box-shadow: 0px 0px 3px 0px rgba(0, 0, 0, .3);
}
/* Image Slider Section */
#image-slider {
    height: 425px;
    width: 960px;
}
#image-slider ul {
    list-style: none;
    margin: 0;
    padding: 0;
}
#image-slider img {
    display: block; /** For removing white space **/
}
#image-slider figcaption {
    background-color: rgba(0,0,0,.7);
    color: #fff;

```

```

padding: 10px 20px;
position: relative;
text-shadow: 1px 1px 0px #000;
width: 920px;
bottom: 0;
}
#image-slider .slide {
    position: absolute;
}
#image-slider .slide img {
    position: relative;
    opacity: 0;
}
#image-slider .slide:first-child img {
    opacity: 1;
}
#image-slider .hidden {
    height: 380px;
    overflow: hidden;
    position: relative;
    width: 960px;
    border-bottom: 1px solid #aaa;
}
#image-slider .shine {
    height: 300px;
    margin: 0 auto;
    position: absolute;
    right: -60px;
    top: -58px;
    transform: rotate(-20deg);
    background: linear-gradient(45deg, rgba(255,255,255,0) 50%, rgba(255,255,255,1)
100%);
    width: 1500px;
    z-index: 5;
}
/* Slide Navigation Section */
.nav-slider {
    text-align: center;
    padding-top: 15px;
    border-top: 1px solid #fff;
}
.nav-slider li {
    display: inline;
    margin: 0 5px;
}
.nav-slider li a {
    background: #ddd;
    border: 1px solid #ccc;
    display: inline-block;
    height: 19px;
    text-indent: -9999px;
    width: 19px;
}
/* Image Caption Section */
#image-caption {
    padding: 32px 32px 40px;

```

```

}
#image-caption ul {
  list-style: none;
  padding: 0;
  margin: 0;
}
#image-caption ul li {
  display: inline-block;
  width: 270px;
  vertical-align: top;
  height: 190px;
  background: #fff;
  border: 5px solid #ddd;
  cursor: pointer;
  box-shadow: 1px 1px 1px 0px rgba(0, 0, 0, .2);
  position: relative;
  overflow: hidden;
}
#image-caption img {
  display: block; /** Remove white space **/
}
.pattern-overlay {
  background: url('images/thumbnail-pattern.png') repeat;
  height: 190px;
  width: 270px;
  position: absolute;
  z-index: 1;
}
.thumbnail:nth-child(3n+2) /** Select only the the middle images (2,5,...etc)**/ {
  margin: 0 23px;
}
.thumbnail:nth-child(-n+3) /** Select only the first 3 images **/ {
  margin-bottom: 33px;
}
.thumbnail figcaption {
  background: rgba(136,184,88,0.9);
  margin: 10px;
  border-radius: 3px;
  padding: 10px;
  width: 230px;
  color: #fff;
  font-size: 10pt;
}
.thumbnail figcaption h3 {
  font-size: 14pt;
  font-weight: normal;
}
.thumbnail figcaption p, .thumbnail figcaption h3 {
  margin: 3px;
  padding: 0;
}
.thumbnail-1 figcaption {
  margin: 0 10px 0 10px;
}
.thumbnail-2 figcaption {
  height: 150px;
  position: absolute;

```

```

position: absolute;
top: -190px;
}
.thumbnail-3 figcaption {
transform: scale(0);
height: 150px;
opacity: 0;
position: absolute;
top: 0;
}
.thumbnail-4 figcaption {
transform: rotate(-180deg);
overflow: hidden;
max-height: 150px;
position: relative;
}
.thumbnail-4 figcaption p {
transform: scale(3);
opacity: 0;
}
.thumbnail-5 img {
box-shadow: 1px 1px 1px 0px rgba(0, 0, 1, .5);
position: relative;
z-index: 1;
}
.thumbnail-5 figcaption {
margin: 0;
border-radius: 0;
width: 250px;
}
.thumbnail-6 figcaption {
top:0;
position: absolute;
height: 150px;
transform: scale(0);
transform-origin: 0 0;
opacity: 0;
}
.thumbnail-6, .thumbnail-6 img {
box-shadow: 1px 1px 1px 0px rgba(0, 0, 1, .5);
}
/* Footer */
footer {
width: 960px;
text-align: center;
margin: 0 auto;
color: #777;
text-shadow: 1px 1px 0px #fff;
font-size: 10pt;
}

```

And then effects.css for all the CSS rules which perform the transition and transformation effects in the website.

```

/* Site Navigation */
"-----"

```

```

#site-nav li a {
  transition: all 500ms;
}
#site-nav li a:hover {
  color: #ffb400;
}

/* Image Slider Effects */
#image-slider li img {
  transition: all 1s linear;
}
#image-slider li figcaption {
  transition: all 500ms ease-in-out 300ms;
}
#image-slider li:target img {
  opacity: 1;
}
#image-slider li:target figcaption {
  transform: translateY(-100%) !important;
}
#image-slider li:first-child figcaption {
  transform: translateY(-100%);
}
#image-slider li:first-child figcaption.hide {
  transform: translateY(0);
}

/* Nav Slider */
.nav-slider li a {
  transition: all 1s ease-out;
}
.nav-slider li a:hover {
  background: #89c53f;
  border: 1px solid #467536;
  transition: all 300ms ease-in;
}
.nav-slider li a:active {
  background: #467536;
  border: 1px solid #20411d;
  transition: all 50ms;
}

/* Image Caption */
.thumbnail img {
  transition: all 350ms;
}
.thumbnail-1 figcaption {
  transition: all 350ms linear;
}
.thumbnail-2 figcaption {
  transition: all 350ms ease-in 350ms;
}
.thumbnail-3 figcaption {
  transition: all 350ms ease-out;
}
.thumbnail-4 figcaption {
  transition: all 500ms ease-in-out;
}

```

```

    transition: all 350ms ease-in-out;
}
.thumbnail-4 figcaption p {
    transition: all 350ms ease-in-out 500ms;
}
.thumbnail-5 figcaption, .thumbnail-5 img {
    transition: all 350ms cubic-bezier(.1,.72,.68,.68);
}
.thumbnail-6 figcaption, .thumbnail-6 img {
    transition: all 500ms;
}
.thumbnail:hover img {
    opacity: 0.3;
}
.thumbnail-1:hover figcaption {
    transform: translateY(-100%);
    margin-top: -10px;
}
.thumbnail-2:hover figcaption {
    transform: translateY(190px);
}
.thumbnail-3:hover figcaption {
    transform: scale(1);
    opacity: 1;
}
.thumbnail-4:hover figcaption {
    transform: rotate(0deg) translateY(-100%);
    margin-top: -10px;
}
.thumbnail-4:hover figcaption p {
    transform: scale(1);
    opacity: 1;
}
.thumbnail-5:hover figcaption, .thumbnail-5:hover img {
    transform: translateY(-90px);
}
.thumbnail-6:hover img {
    transform: scale(0);
    transform-origin: 0 0;
    opacity: 0;
}
.thumbnail-6:hover figcaption {
    transform: scale(1);
    transform-origin: 0 0;
    opacity: 1;
}

```

---

## Final Thoughts

We're done! It's been useful seeing how we can accomplish so many effects today only using CSS rules with the bare minimum of JavaScript.

Don't forget the lack of support from older browsers, especially IE. But used conservatively these kinds of effects can enhance websites in a gracefully degrading manner.



Thanks for reading this tutorial; I hope you learned something!