HTML 5 Podcast

Project name	R+L Carriers
Project Id	22394
Project Location	Akruti, Mumbai
BU	CPRD-Retail
Client	R+L Carriers
Speaker's Name	Shruti Kamath
Speaker's email ids	shruti.kamath@igate.com
Actual Start Date (DD-Mon-YY)	NA
Actual End Date (DD-Mon-YY)	NA
Project type (development/ Maintenance/Migration etc.)	Maintenance/Development
Team Size	30
Domain / Vertical	Retail
Technology	HTML 5
Keywords (this would be used during search of podcast on KM portal)	HTML 5
Summary of the podcast in a paragraph	This podcast provides an overview of HTML 5 its basic structure, new tags added some deprecated tags and how browser compatibility can be tackled.

Introduction: Hi everyone!! Thank you and welcome all for this podcast. I would like to introduce myself before I get this started. I am Shruti Kamath.

Here in this podcast I would like to talk about HTML 5 in general the list of deprecated tags its basic semantic structure and all the new tags that have been added.

So to begin with…

✓ **What is HTML 5?**

HTML 5 is a new standard for HTML. The previous version of HTML was - HTML 4.01. HTML 5 is designed to deliver almost everything you want to do online without requiring additional plug-ins.HTML5 provide one common interface to make loading elements easier. For example, there is no need to install a Flash plug-in in HTML5 because the element will run by itself. It does everything from animation to apps, music to movies.

Let's have a look at the list of deprecated tags and attributes.

✓ **Deprecated tags**

The following elements are not available in HTML5 anymore and their function is better handled by CSS

Tags (Elements)	Description
<applet>	Defines an applet
<center>	Defines centered text
<font>	Defines text font, size, and color
<frame>	Defines a frame
<frameset>	Defines a set of frames

<noframes>	Defines a noframe section
<s>	Defines strikethrough text
✓ <strike>	Defines strikethrough text
✓ ✓ <u>	Defines underlined text

✓ **Deprecated Attributes**

HTML5 has none of the presentational attributes that were in HTML4 as their functions are better handled by CSS. Some attributes from HTML4 are no longer allowed in HTML5 at all and they have been removed completely.

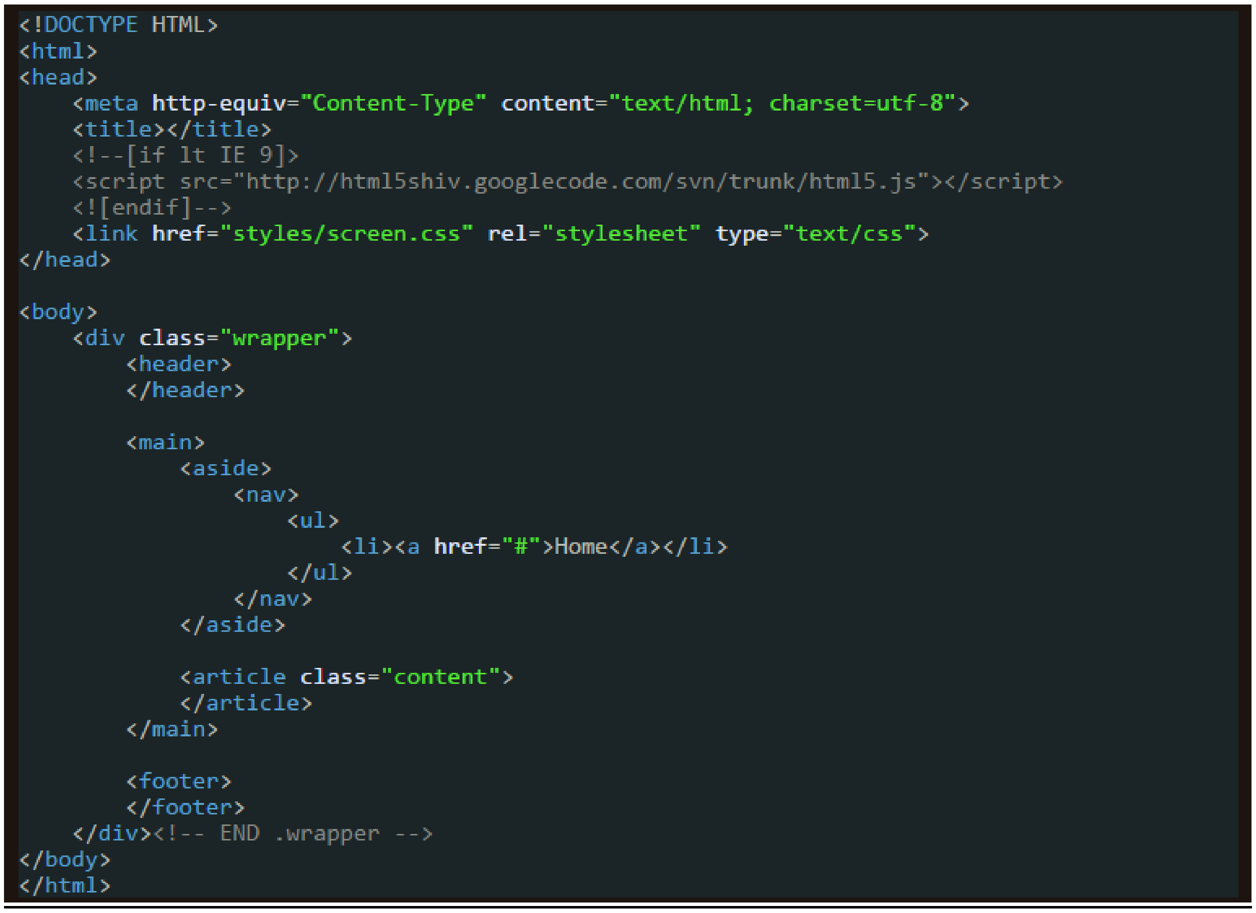
Following is the table having removed attributes and their corresponding impacted tags (elements) i.e. elements from which those attributes have been removed permanently –

Removed Attributes	From the Elements
charset	link and a
target	link
version	html
name	img
scheme	meta
abbr	td and t

align	caption, iframe, img, input, object, legend, table, hr, div, h1, h2, h3, h4, h5, h6, p, col, colgroup, tbody, td, tfoot, th, thead and tr.
link	body
background	body
bgcolor	table, tr, td, th and body.
border	table and object.
cellpadding	table
cellspacing	table
clear	br
frame	table
frame	table
frameborder	iframe
hspace	img and object.
vspace	img and object.
marginheight	iframe

marginwidth	iframe
nowrap	td and th
rules	table
scrolling	iframe
size	hr
type	li, ol and ul.
valign	col, colgroup, tbody, td, tfoot, th, thead and tr
width	hr, table, td, th, col, colgroup and pre.

✓ HTML5 basic Semantic Structure



```
<!DOCTYPE HTML>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title></title>
    <!--[if lt IE 9]>
    <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
    <link href="styles/screen.css" rel="stylesheet" type="text/css">
</head>

<body>
    <div class="wrapper">
        <header>
        </header>

        <main>
            <aside>
                <nav>
                    <ul>
                        <li><a href="#">Home</a></li>
                    </ul>
                </nav>
            </aside>

            <article class="content">
            </article>
        </main>

        <footer>
        </footer>
    </div><!-- END .wrapper -->
</body>
</html>
```

As we can see the template shows the basic semantic structure which should ideally be followed when we design a web page.

Starting with the

1. New Doctype

Still using that pesky, impossible-to-memorize XHTML doctype?

```
<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

If so, why? Switch to the new HTML5 doctype and it's so simple and easy to remember.

```
<! DOCTYPE html>
```

In fact, did you know that it truthfully isn't even really necessary for HTML5? However, it's used for current and older browsers that require a specified doctype. Browsers that do not understand this doctype will simply render the contained markup in standards mode. So, without worry, feel free to throw caution to the wind, and embrace the new HTML5 doctype.
Next we see is a commented line of code which is used to resolve the browser compatibility issues when executing on Internet Explorer 8 and earlier, which does not allow styling of unknown elements. We shall look into this section at a little later phase.

2. The Semantic Header and Footer

Gone are the days of:

```
<div id="header">
   ...
</div>

<div id="footer">
   ...
</div>
```

Divs, by nature, have no semantic structure -- even after an id is applied. Now, with HTML5, we have access to the <header> and <footer> elements. The mark-up above can now be replaced with:

```
<header>
   ...
</header>

<footer>
   ...
</footer>
```

It's fully appropriate to have multiple headers and footers in your project*s.*

Try not to confuse these elements with the "header" and "footer" of your website. They simply refer to their container. As such, it makes sense to place, for example, meta information at the bottom of a blog post within the footer element. The same holds true for the header.

3. The Figure Element

Consider the following mark-up for an image:

```
<img src="path/to/image" alt="About image" />
<p>Image of Mars. </p>
```

There unfortunately isn't any easy or semantic way to associate the caption, wrapped in a paragraph tag, with the image element itself. But HTML5 rectifies this, with the introduction of the <figure> element. When combined with the <figcaption>element, we can now semantically associate captions with their image counterparts.

```
<figure>
   <img src="path/to/image" alt="About image" />
   <figcaption>
      <p>This is an image of something interesting. </p>
   </figcaption>
</figure>
```

4. No More "Types" for Scripts and Links

You possibly still add the type attribute to your link and script tags as you can view

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

<scripttype="text/javascript" src="path/to/script.js"></script>
```

This is no longer necessary. It's implied that both of these tags refer to stylesheets and scripts, respectively. As such, we can remove the type attribute all together.

```
<link rel="stylesheet" href="path/to/stylesheet.css" />
<script src="path/to/script.js"></script>
```

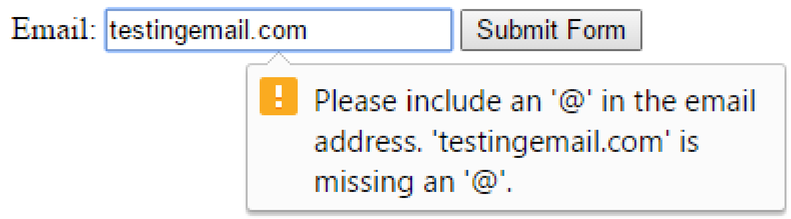
5. Email Inputs

If we apply a type of "email" to form inputs, we can instruct the browser to *only* allow strings that confirm to a valid email address structure. That's right; built-in form validation will soon be here! We can't 100% rely on this just yet, for obvious reasons. In older browsers that do not understand this "email" type, they'll simply fall back to a regular textbox.

```
<!DOCTYPE html>

<html lang="en">
<head>
  <meta charset="utf-8">
  <title>untitled</title>
</head>
<body>
  <form action="" method="get">
    <label for="email">Email:</label>
    <input id="email" name="email" type="email" />

    <button type="submit"> Submit Form </button>
  </form>
</body>
</html>
```



Please include an '@' in the email address. 'testingemail.com' is missing an '@'.

At this time, we cannot depend on browser validation. A server/client side solution must still be implemented.

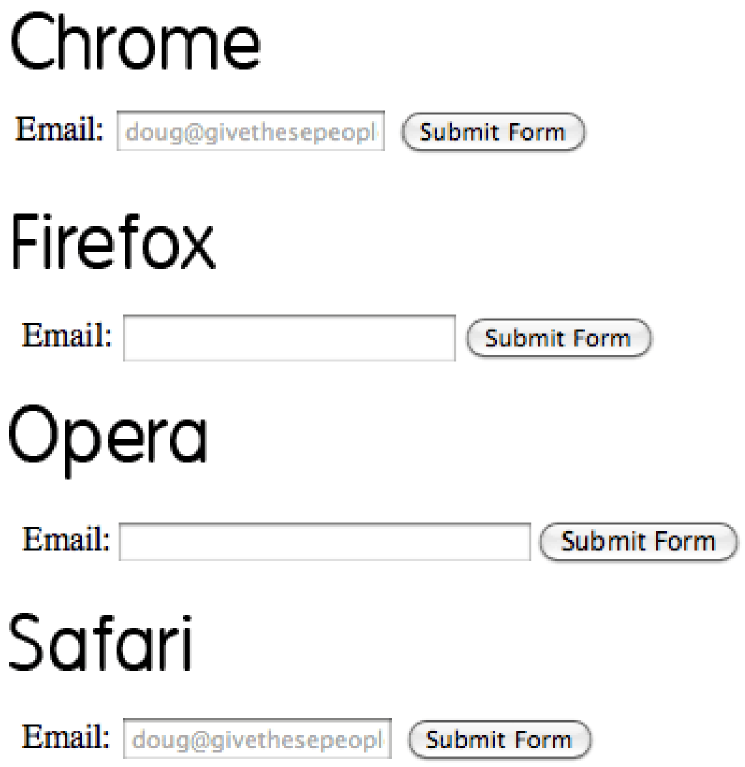
It should also be noted that all the current browsers are a bit wonky when it comes to what elements and attributes they do and don't support. For example, Opera seems to support email validation, just as long as the name attribute is specified. However, it does not support the placeholder attribute, which we'll learn next. Bottom line, don't depend on this form of validation just yet...but you can still use it!

6. Placeholders

Before, we had to utilize a bit of JavaScript to create placeholders for textboxes. Sure, you can initially set the value attribute how you see fit, but, as soon as the user deletes that text and clicks away, the input will be left blank again. The placeholder attribute remedies this.

```
<input name="email" type="email" placeholder="doug@givethesepeopleair.com" />
```

Again, support is shady at best across browsers; however, this will continue to improve with every new release. Besides, if the browser, like Firefox and Opera, don't currently support the placeholder attribute, no harm done.
As we can see different browsers shows different output for the placeholder attribute.



7. Internet Explorer and HTML5

Unfortunately, the Internet Explorer requires a bit of wrangling in order to understand the new HTML5 elements and all elements, by default, have a display of inline.

In order to ensure that the new HTML5 elements render correctly as block level elements, it's necessary at this time to style them as such.

```
header, footer, article, section, nav, menu {
  display: block;
}
```

Unfortunately, Internet Explorer will still ignore these styling's, because it has no clue what, as an example, the header element even is. Luckily, there is an easy fix:

```
document.createElement("article");
document.createElement("footer");
document.createElement("header");
document.createElement("nav");
document.createElement("menu");
```

Strangely enough, this code seems to trigger Internet Explorer. To simply this process for each new application, Thankfully, Sjoerd Visscher created the "HTML5 Enabling JavaScript", "the shiv". This script also fixes some printing issues as well.

```
<!--[if IE]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

The link to the shiv code must be placed in the <head> element, because Internet Explorer needs to know about all new elements before reading them.

8. Required Attribute

Forms allow for a new required attribute, which specifies, naturally, whether a particular input is required. Dependent upon your coding preference, you can declare this attribute in one of two ways:

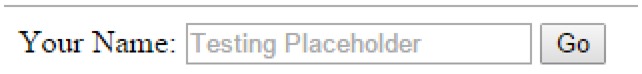
```
<input type="text" name="someInput" required>
```

Or, with a more structured approach.

```
<input type="text" name="someInput" required="required">
```

Either method will do. With this code, and within browsers that support this attribute, a form cannot be submitted if that "someInput" input is blank. Here's a quick example; we'll also add the placeholder attribute as well, as there's no reason not to.

```
<form method="post" action="">
  <label for="someInput"> Your Name: </label>
  <input type="text" id="someInput" name="someInput" placeholder="Testing Placeholder" required>
  <button type="submit">Go</button>
</form>
```



If the input is left blank, and the form is submitted, the textbox will be highlighted.

9.Autofocus Attribute

Again, HTML5 removes the need for JavaScript solutions. If a particular input should be "selected," or focused, by default, we can now utilize the autofocus attribute.

```
<input type="text" name="someInput" placeholder=" Testing Placeholder " required autofocus>
```

Interestingly enough, while I personally tend to prefer a more XHTML approach (using quotation marks, etc.), writing "autofocus=autofocus" feels odd. As such, we can stick to any of the approach.

10. Regular Expressions

How often have you found yourself writing some quickie regular expression to verify a particular textbox? Thanks to the new pattern attribute, we can insert a regular expression directly into our markup.

```
<form action="" method="post">
  <label for="username">Create a Username: </label>
  <input type="text"
    name="username"
    id="username"
    placeholder="4 <> 10"
    pattern="[A-Za-z]{4,10}"
    autofocus
    required>
  <button type="submit">Go </button>
</form>
```

If you're moderately familiar with regular expressions, you'll be aware that this pattern: [A-Za-z]{4,10} accepts only upper and lowercase letters. This string must also have a minimum of four characters, and a maximum of ten.

11. Detect Support for Attributes

What good are these attributes if we have no way of determining whether the browser recognizes them? Well, good point; but there are several ways to figure this out. We'll discuss two. The first option is to utilize the excellent Modernizr library. Alternatively, we can create and dissect these elements to determine what the browsers are capable of. For instance, in our previous example, if we want to determine if the browser can implement the pattern attribute, we could add a bit of JavaScript to our page:

```
alert( 'pattern' in document.createElement('input') ) // boolean;
```

In fact, this is a popular method of determining browser compatibility. The jQuery library utilizes this trick. Above, we're creating a new input element, and determining whether the pattern attribute is recognized within. If it is, the browser supports this functionality. Otherwise, it of course does not.

```
<script>
if (!'pattern' in document.createElement('input') ) {
	// do client/server side validation
}
</script>
```

Note that this relies on JavaScript!

12. Audio Support

No longer do we have to rely upon third party plugins in order to render audio. HTML5 now offers the <audio> element. Well, at least, ultimately, we won't have to worry about these plugins. For the time being, only the most recent of browsers offer support for HTML5 audio. At this time, it's still a good practice to offer some form of backward compatibility.

```
<audio autoplay="autoplay" controls="controls">
   <source src="file.ogg" />
   <source src="file.mp3" />
   <a>Download this file.</a>
</audio>
```

Mozilla and Webkit don't fully get along just yet, when it comes to the audio format. Firefox will want to see an .ogg file, while Webkit browsers will work just fine with the common .mp3 extension. This

means that, at least for now, you should create two versions of the audio.
When Safari loads the page, it won't recognize that .ogg format, and will skip it and move on to the mp3 version, accordingly. Please note that IE, per usual, doesn't support this, and Opera 10 and lower can only work with .wav files.

13. Video Support

Much like the <audio> element, we also, of course, have HTML5 video as well in the new browsers!
While Safari, and Internet Explorer 9 can be expected to support video in the H.264 format (which Flash players can play), Firefox and Opera are sticking with the open source Theora and Vorbis formats. As such, when displaying HTML5 video, you must offer both formats.

```
<video controls preload>
   <source src="cohagenPhoneCall.ogv" type="video/ogg; codecs='vorbis, theora'" />
   <source src="cohagenPhoneCall.mp4" type="video/mp4; 'codecs='avc1.42E01E, mp4a.40.2'" />
   <p> Your browser is old. <a href="cohagenPhoneCall.mp4">Download this video instead.</a> </p>
</video>
```

Chrome can correctly display video that is encoded in both the "ogg" and "mp4" formats.

There are a few things worth noting here.
We aren't technically required to set the type attribute; however, if we don't, the browser has to figure out the type itself. Save some bandwidth, and declare it yourself.
Not all browsers understand HTML5 video. Below the source elements, we can either offer a download link, or embed a Flash version of the video instead. It's up to you.

14. Preload Videos

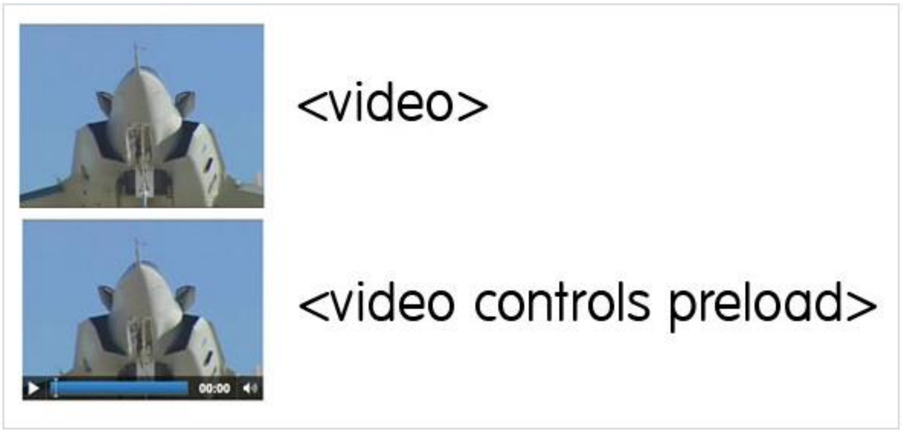
The preload attribute does exactly what you'd guess. Though, with that said, you should first decide whether or not you want the browser to preload the video. Is it necessary? Perhaps, if the visitor accesses a page, which is specifically made to display a video, you should definitely preload the video, and save the visitor a bit of waiting time. Videos can be preloaded by setting preload="preload", or by simply adding preload. I prefer the latter solution; it's a bit less redundant.

```
<video preload>
```

15. Display Controls

If you're working along with each of these tips and techniques, you might have noticed that, with the code above, the video above appears to be only an image, without any controls. To render these play controls, we must specify the controls attribute within the video element.

<video preload controls>



Please note that each browser renders its player differently from one another.

Conclusion:
I hope this podcast would have given you a fair insight of what HTML 5 is all about. Once again I would like to thank you for listening this Podcast.
For any comments or feedbacks, please do write it to me at shruti.kamath@igate.com.

Thank You!!