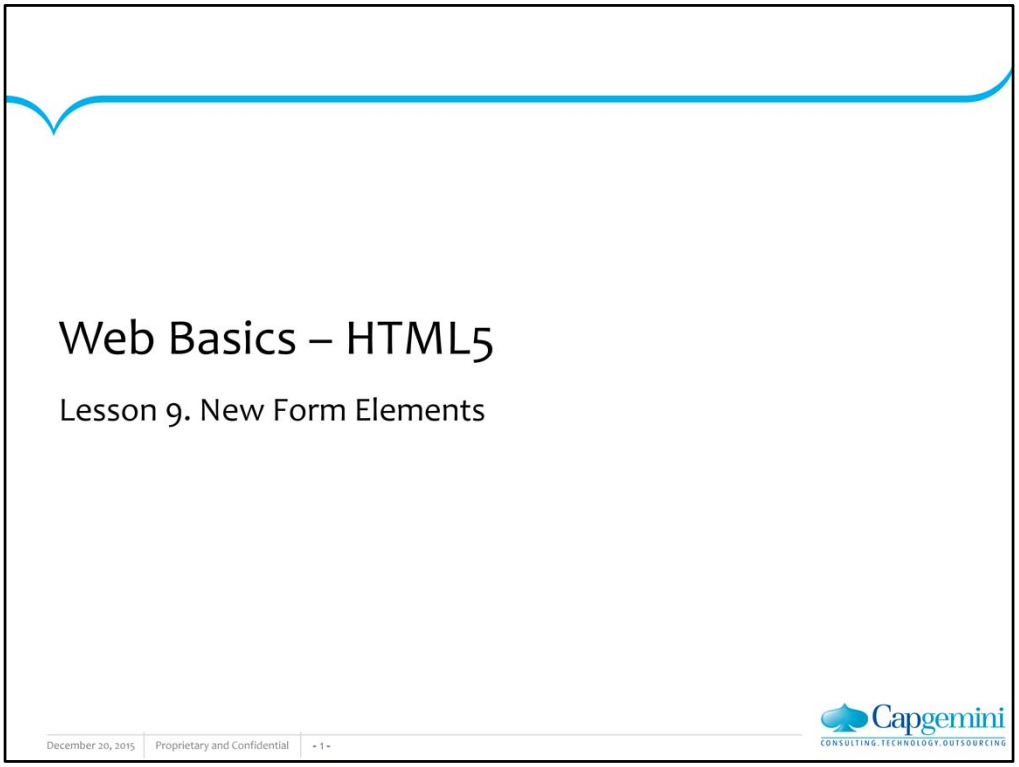
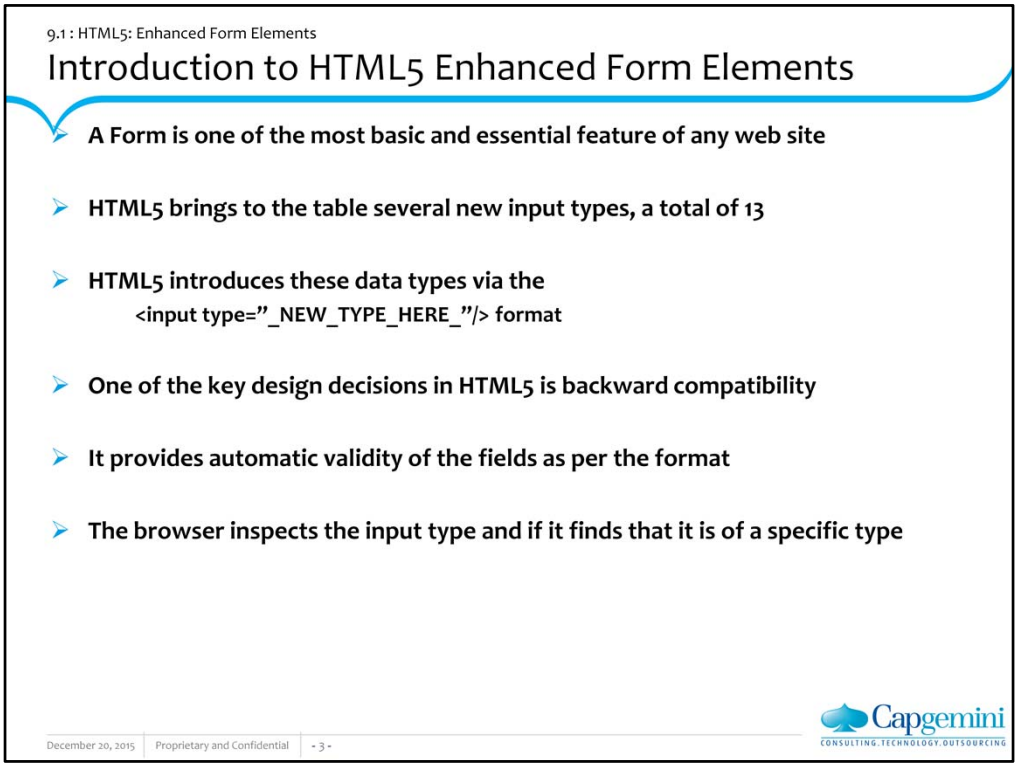

Web Basics – HTML5
Lesson 9. New Form Elements
December 20, 2015
Proprietary and Confidential
- 1 -
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Lesson Objectives

- **In this lesson you will learn about:**
  - HTML5: Enhanced Form Elements
  - An HTML5 Detection Library: Modernizr
  - Using Canvas in HTML5

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING



Introduction to HTML Enhanced Form Elements:

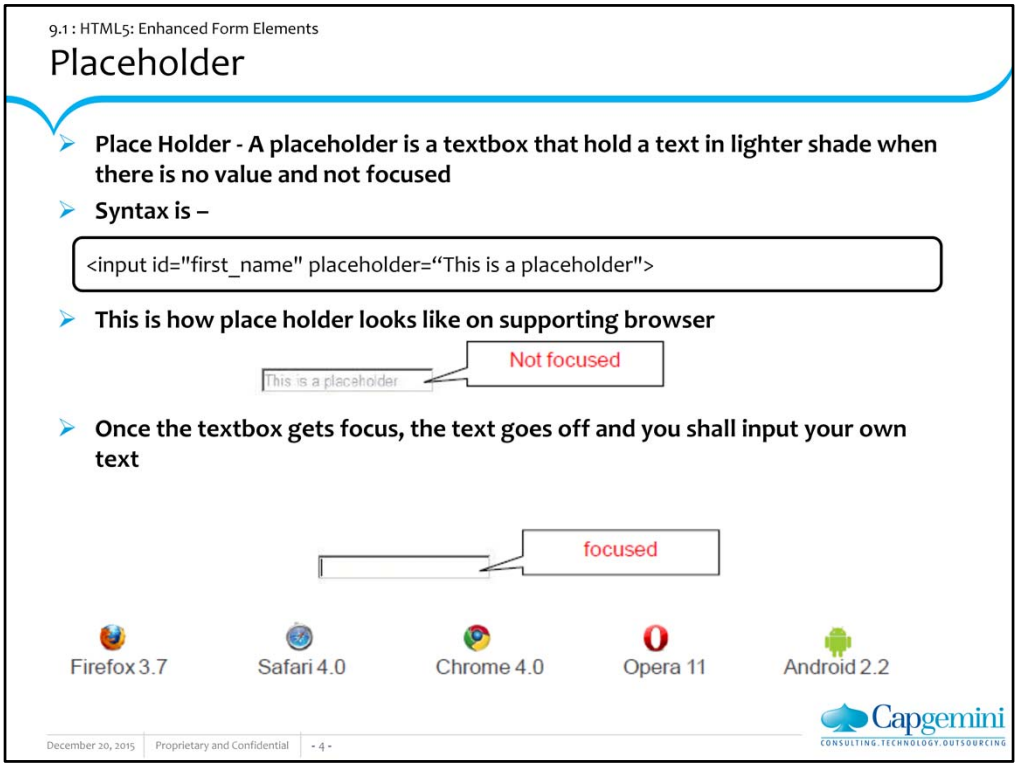
A Form is one of the most basic and essential features of any web site. The form elements available in HTML so far include the textbox, checkbox, radio, button, drop -down list, password and file picker. While these have sufficed so far, there is a clear need for newer form elements. The question is not just of newer form elements, but the ability to inject behaviour into existing form elements so that usability and validity, which is a cornerstone of any good UI, is given highest consideration.

One of the key design decisions in HTML5 is backward compatibility. What this means is that if the new input types are not supported, then by default it falls back to **<input type="text"…./>**, so it will be rendered as a plain text box, which the user can then fill data in.

Advantages of new input types:

a) You get automatic validity of the fields as per the format. This means that the form is not going to get submitted if the value entered is not as per the default validation of that type

b) The browser inspects the input type and if it finds that it is of a specific type, then it does something quite clear to aid the input of that data. For e.g. On the Smart Phones, which do not have a physical keyboard but instead a virtual keyboard, the keyboard that will be shown up will only contain keys that will aid the user in filling out the data.



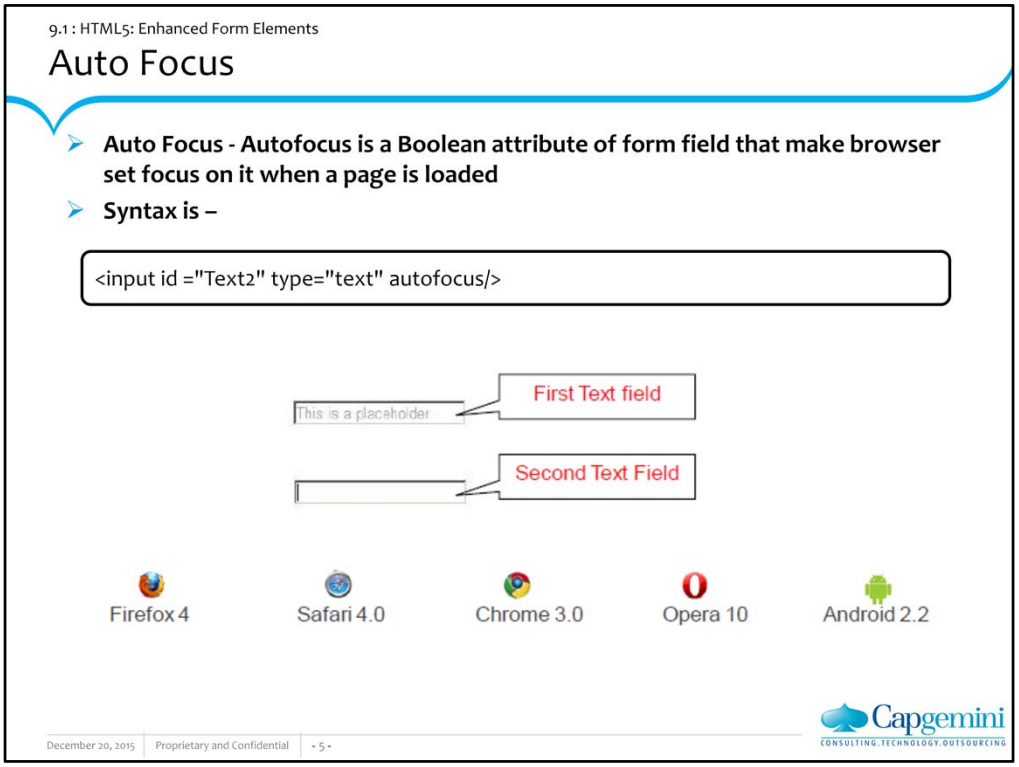
Place Holder:

The first improvement HTML5 brings to web forms is the ability to set placeholder text in an input field. Placeholder text is displayed inside the input field as long as the field is empty and not focused. As soon as you click on (or tab to) the input field, the placeholder text disappears.

Here's how you can include placeholder text in your own web forms:

```
<form>
 <input name="name" placeholder="Enter your name">
 <input type="submit" value="Search">
</form>
```

Browser's that don't support placeholder attribute will simply ignore it. But if you want to make it work in other browsers, you can use some **JavaScript** to create The same behavior. There is an excellent **jQuery plugin** called HTML5 Placeholder Plugin that will go through all input fields with placeholders attached to them and make them work in all browsers.

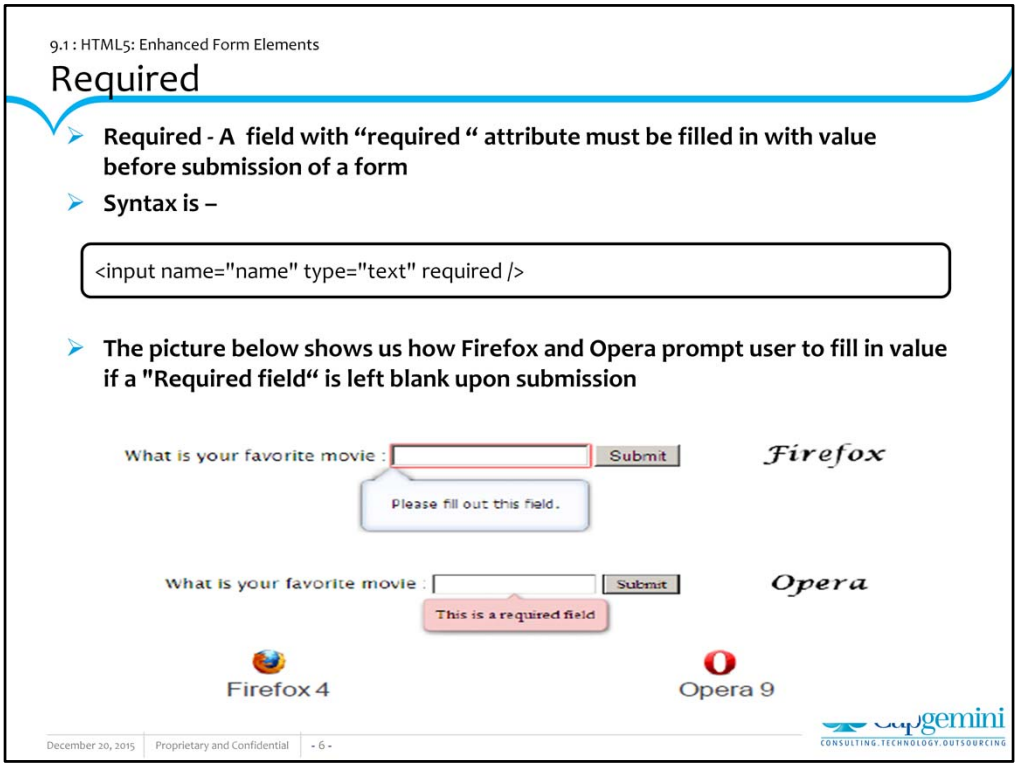


Auto Focus

This attribute when applied to any form element, will result in the field receiving focus. For e.g. consider the example shown below:

```
<form>
 <label for="firstname">First Name</label>
 <input type="text" id="firstname" name="firstname" autofocus>
 <label for="lastname">Last Name</label>
 <input type="text" id="lastname" name="lastname">
 <input type="submit" label="Go">
</form>
```

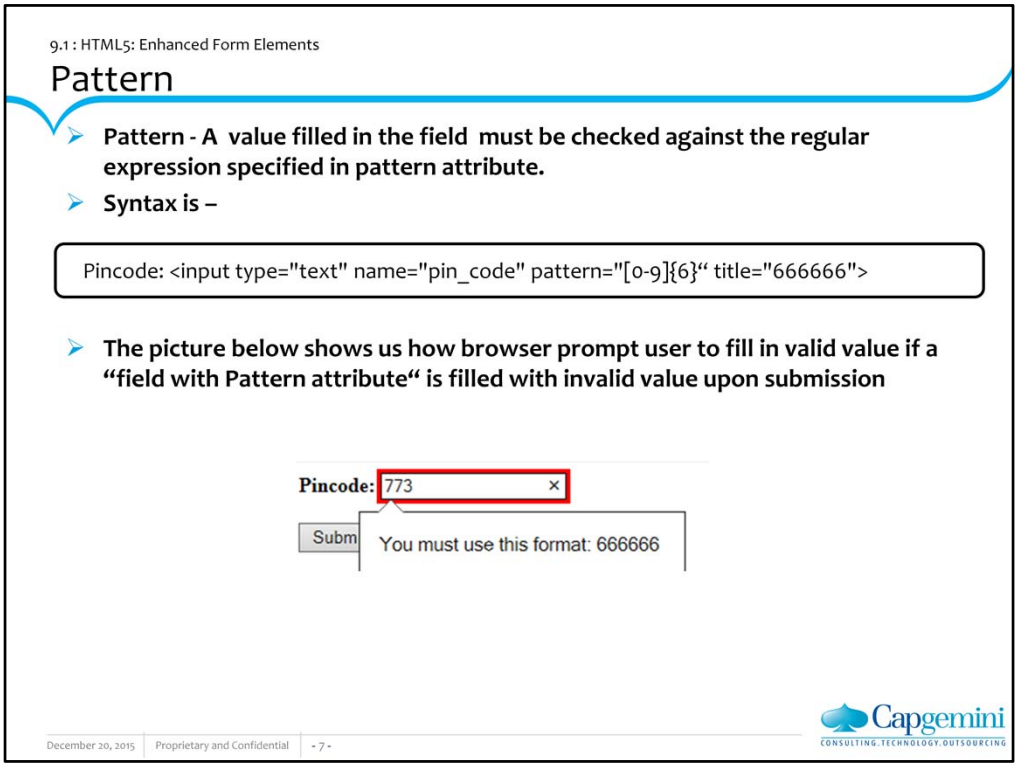
We have added the attribute autofocus to the firstname input field. When the form loads, you will find that the focus is already set on that field, thereby making it easier for the user to start filling the form.



Required:
If the **required** attribute is present, then the field must contain a value when the form is submitted. This informs the (*HTML5*-aware) web browser that the field is to be considered mandatory. Different browsers may mark the input box in some way (Firefox 4 Beta adds a red box-shadow by default), display a warning (Opera) or even prevent the form from being submitted if this field has no value. Hopefully these behaviours will converge in future releases.

Here's an example of an input field for a required email address that ensures that the field has a value and that the value is a valid email address.

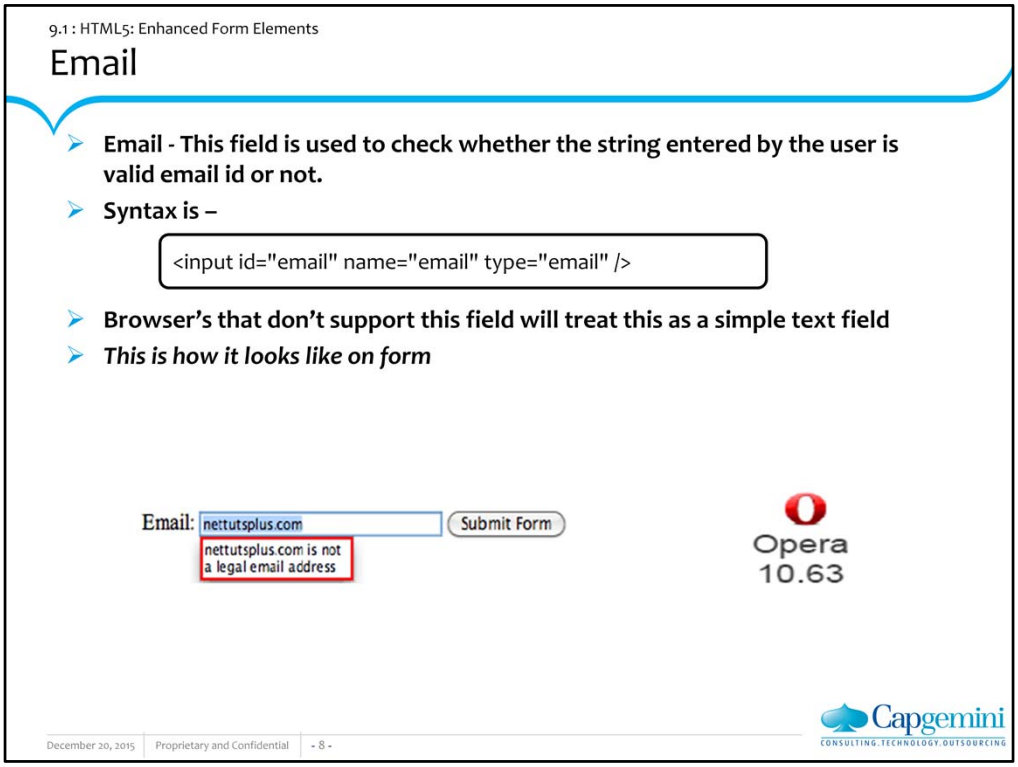
Example:
<input type="email" id="email_addr" name="email_addr" required />



Required:
If the **required** attribute is present, then the field must contain a value when the form is submitted. This informs the (*HTML5*-aware) web browser that the field is to be considered mandatory. Different browsers may mark the input box in some way (Firefox 4 Beta adds a red box-shadow by default), display a warning (Opera) or even prevent the form from being submitted if this field has no value. Hopefully these behaviours will converge in future releases.

Here's an example of an input field for a required email address that ensures that the field has a value and that the value is a valid email address.

Example:
<input type="email" id="email_addr" name="email_addr" required />

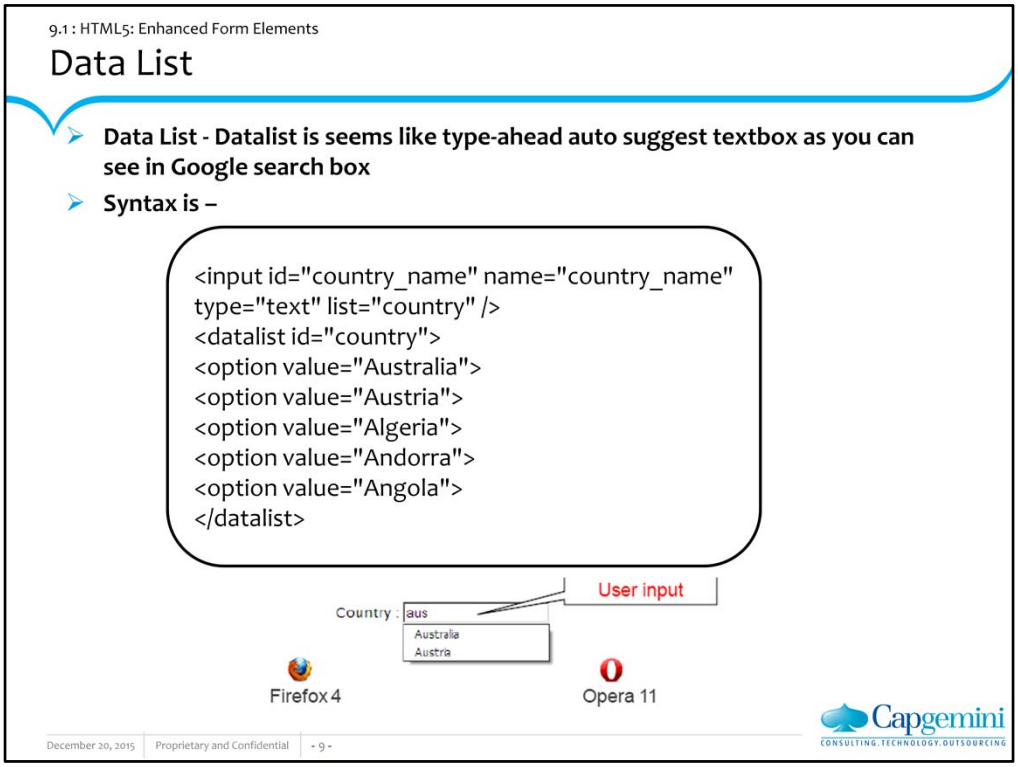


Email:
The email type is used for input fields that should contain an e-mail address. The value of the email field is automatically validated when the form is submitted.

Example

E-mail: <input type="email" name="user_email" />

Safari on the iPhone recognizes the email input type, and changes the on-screen keyboard to match it (adds @ and .com options).

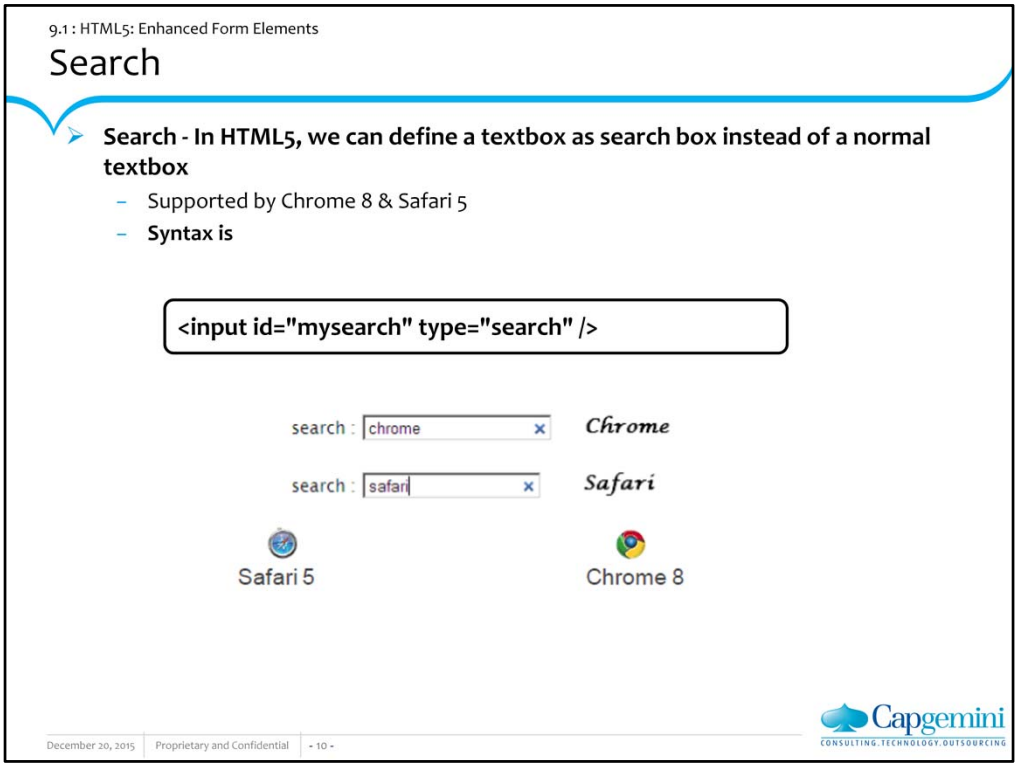


Data List:

The datalist element specifies a list of options for an input field. The list is created with option elements inside the datalist. To bind a datalist to an input field, let the list attribute of the input field refer to the id of the datalist:

Example

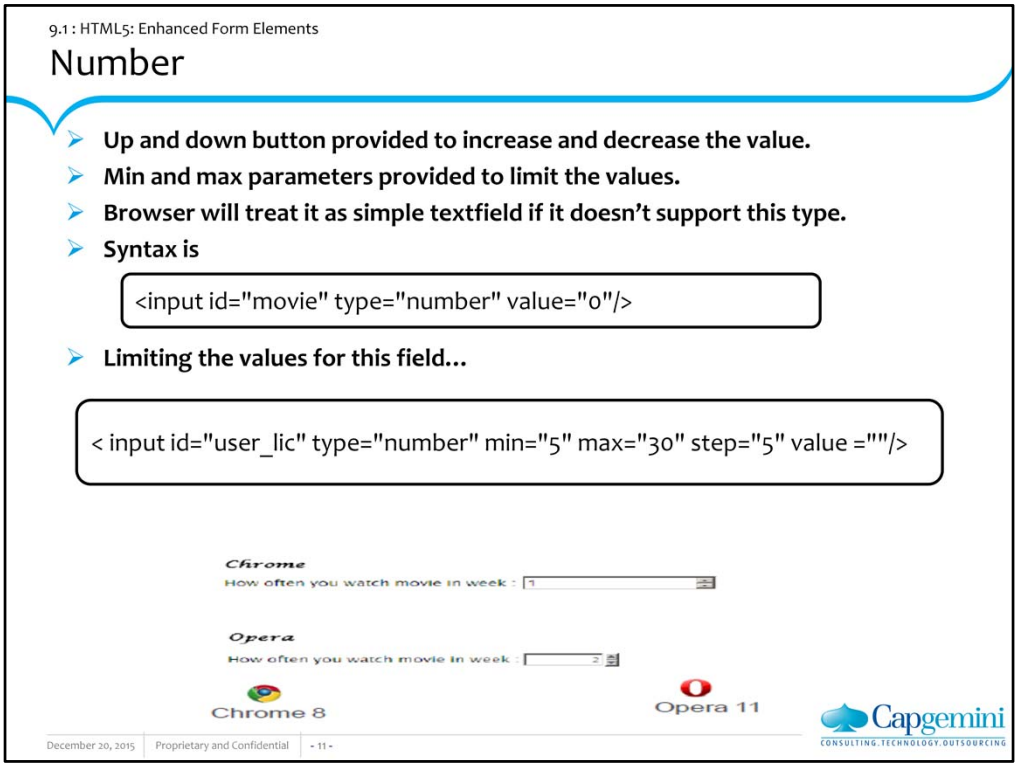
```
Webpage: <input type="url" list="url_list" name="link" />
 <datalist id="url_list">
 <option label="W3Schools" value="http://www.w3schools.com" />
 <option label="Google" value="http://www.google.com" />
 <option label="Microsoft" value="http://www.microsoft.com" />
 </datalist>
```



Search:
Setting the type attribute of an **<input>** tag to **search** is mainly of cosmetic value. It doesn't automatically create a search field. As with all form elements, you still need to code the server-side logic yourself or use a third-party script. Most browsers simply display a search field as a normal text field. However, Safari and Chrome on Mac OS X automatically add rounded corners to match the operating system's standard look.
Apple.com uses **<input type="search">** for their site-search box, to help give their Site a "Mac-like" feel. But there's nothing Mac-specific about it. It's just markup, so each browser on each platform can choose to render it according to platform-specific conventions. As with all the other new input types, browsers that don't recognize type="search" will treat it like type="text" .

Example:

```
<form>
 <input name="q" type="search">
 <input type="submit" value="Find">
</form>
```



Number:

Asking for a number is trickier than asking for an email address or web address. First of all, numbers are more complicated than you might think. You don't often ask for "just a number." It's more likely that you'll ask for a number in a particular range. You may only want certain kinds of numbers within that range- maybe whole numbers but not fractions or decimals.

Example:

<input type="number" min="0" max="10" step="2" value="6">

Let's take that one attribute at a time.

1. type="number" means that this is a number field.
2. min="0" specifies the minimum acceptable value for this field.
3. max="10" is the maximum acceptable value.
4. step="2", combined with the min value, defines the acceptable numbers in the range: 0, 2, 4, and so on, up to the maxvalue.
5. value="6" is the default value.

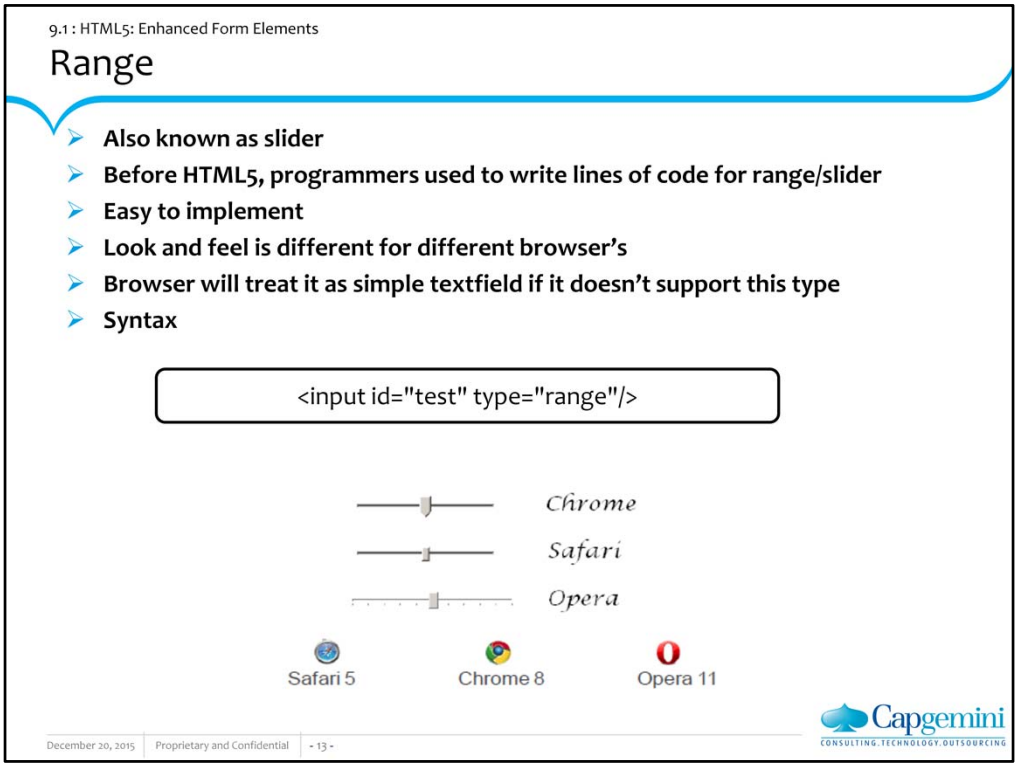
That's the markup side of a number field. Keep in mind that all of those attributes are optional. If you have a minimum but no maximum, you can specify a min attribute but no max attribute. The default step value is 1, and you can omit the step attribute unless you need a different step value. If there's no default value, then the value attribute can be the empty string or even omitted altogether.

Visualizing Number field :

On the iPhone, where input is difficult to begin with, the browser once again optimizes the virtual keyboard for numeric input.

In the desktop version of Opera, the same type="number" field is rendered as a "spinbox" control, with little up and down arrows that you can click to change the value.

Opera respects the min, max, and step attributes, so you'll always end up with an acceptable numeric value. If you bump up the value to the maximum, the up arrow in the spinbox is greyed out.

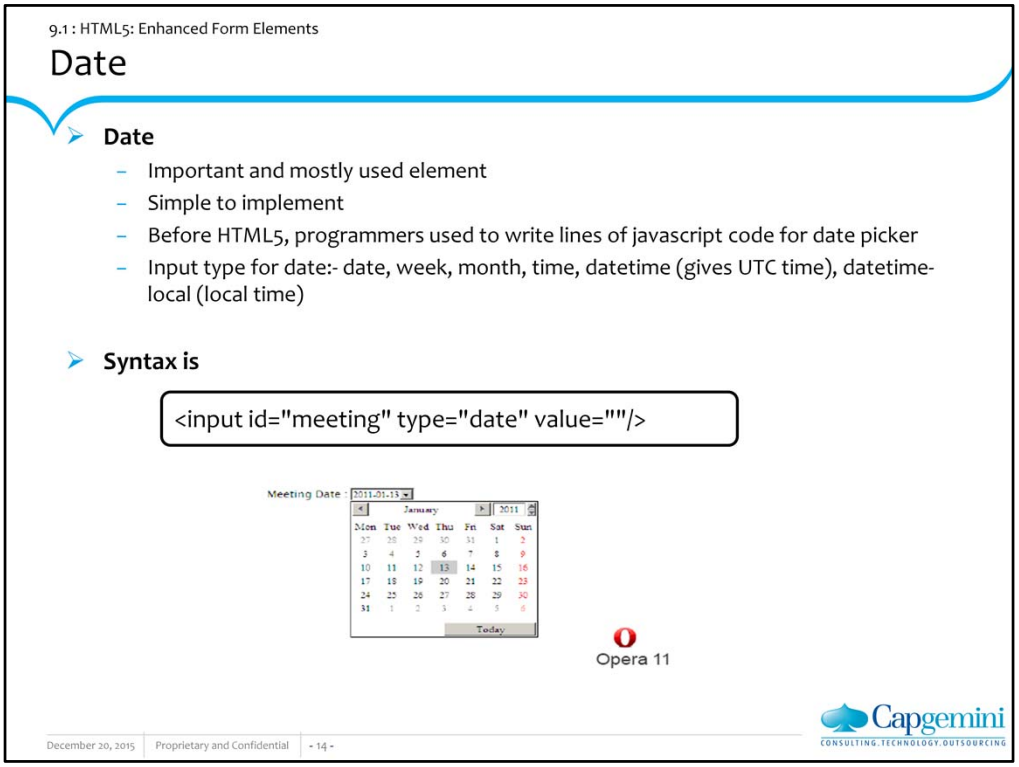


Range:
Spinboxes are not the only way to represent numeric input. You've probably also seen "slider" controls.
You can now have slider controls in your web forms, too.

Example:

<input type="range" min="0" max="10" step="2" value="6">

All the available attributes are the same as type="number" — min, max, step, value — and they mean the same thing. The only difference is the user interface. Instead of a field for typing, browsers are expected to render type="range" as a slider control. Safari, Chrome, and Opera all do this. (Sadly, the iPhone renders it as a simple text box. It doesn't even optimize its on-screen keyboard for numeric input.) All other browsers simply treat the field as type="text".

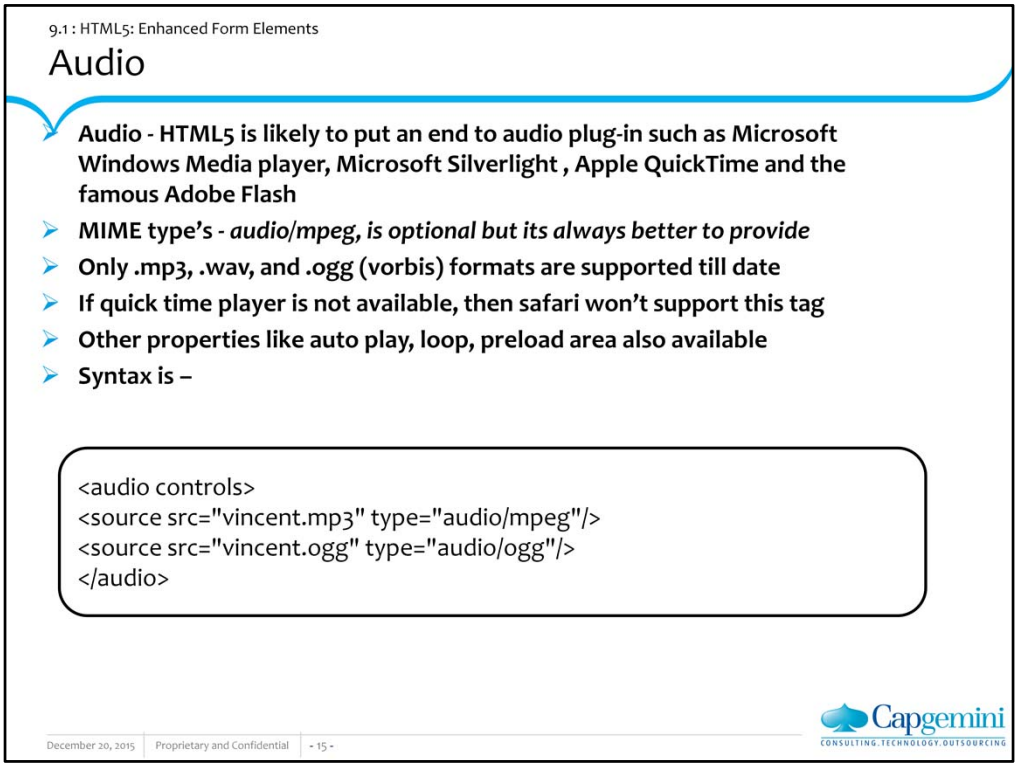


Date:

HTML 4 did not include a date picker control. JavaScript frameworks have picked up the slack (Dojo, jQuery UI, YUI, Closure Library), but of course each of these solutions requires "buying into" the framework on which the date picker is built.

HTML5 finally defines a way to include a native date picker control without having to script it yourself. In fact, it defines six input types: date, month, week, time, date + time, and date + time - timezone.

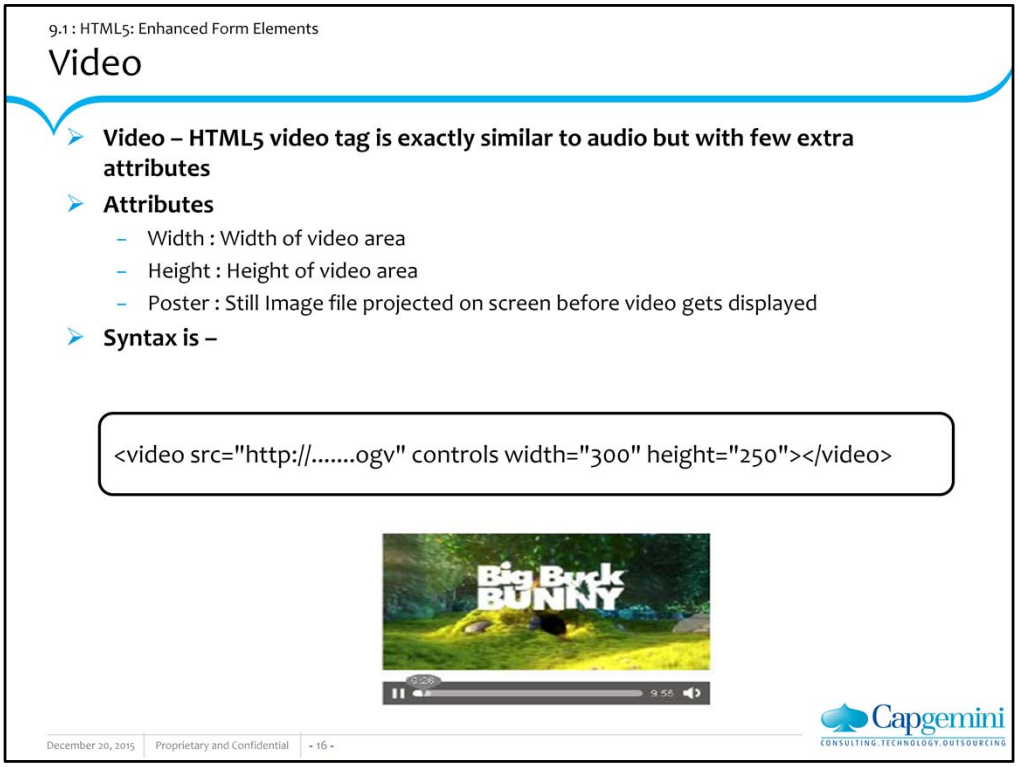
So far, support is… sparse.



Audio & Video:

HTML5 is fairly intelligent about picking the right default for presenting the most optimum audio or video. Couple that with the absolute minimum coding that's needed to handle multimedia in HTML5 and you have a pretty good situation for developers.

The bad news is that because the people diligently working on the HTML5 specification tried to compromise between open formats and de facto standard formats and so on, support for native codecs in HTML5 is slightly lacking: there isn't any. It's up to the browser to support formats, and up to the developer to supply them. What's emerged from that are a few relatively new standards.



Examples:

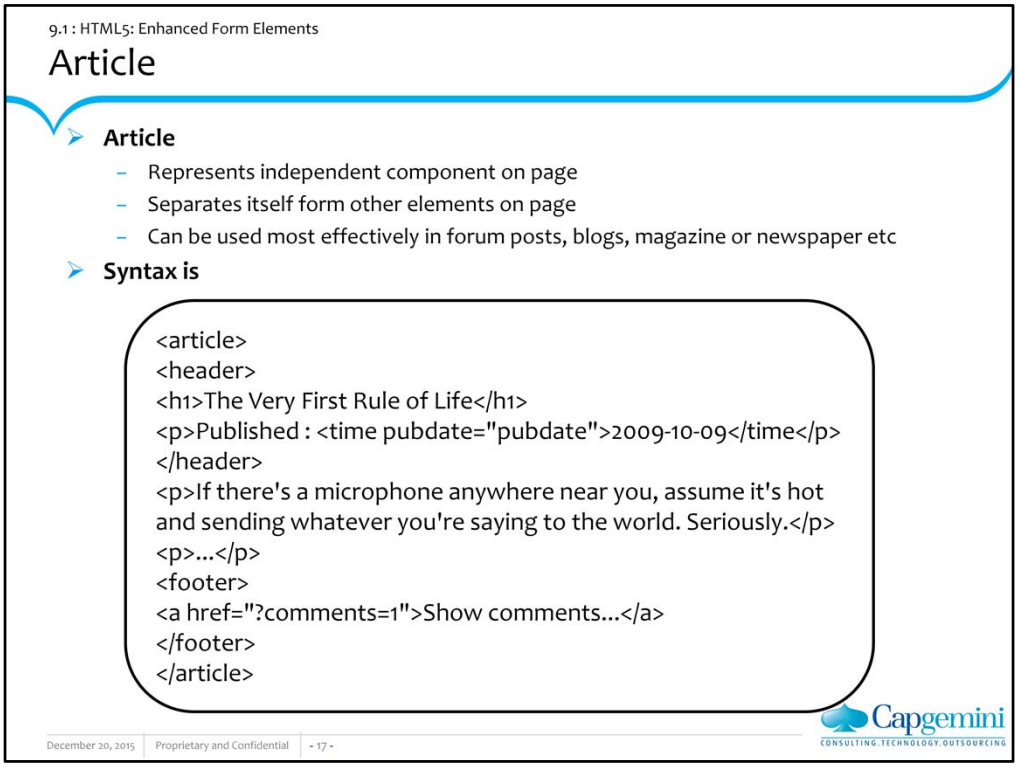
Audio

<audio controls preload="metadata">
<source src="sample.mp3">
<source src="sample.ogg"> </audio>

Video

<video controls width="640" height="480" src="sample.mp4" poster="sample.jpg">
</video>

Both the <video> and <audio> can take another important parameter, shown above, called preload (which was formerly implemented as autobuffer with slightly different syntax). The values for preload can be auto (download the media file to the browser in advance), none (do not preload the media), or metadata (just download enough metadata to discover the duration and other information of the media file). So with a little extra care in coding, you can make the user's experience much better.



Article:

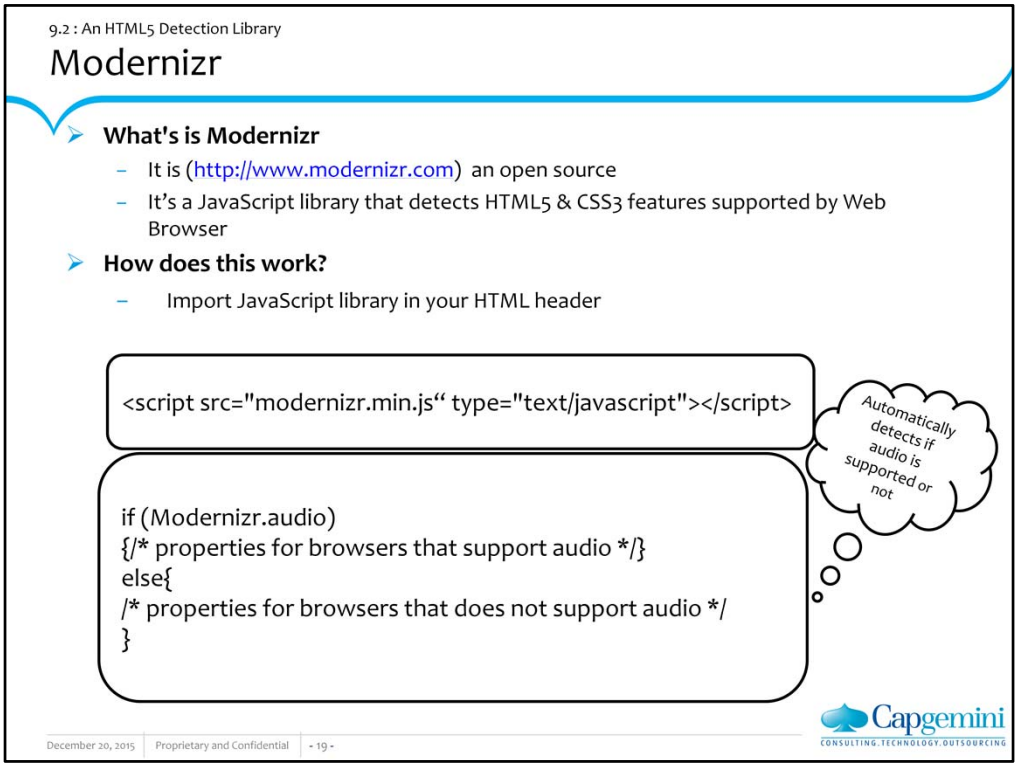
The article element represents a component of a page that consists of a self-contained composition in a document, page, application, or site and that is intended to be independently distributable or reusable, e.g. in syndication. This could be a forum post, a magazine or newspaper article, a Web log entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content.

9.1 : **HTML5: Enhanced Form Elements**

Demo

- **Demonstration on the use of new form elements**

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING



Modernizr

Modernizr is a JavaScript library that helps you add HTML5 capabilities to your websites. HTML5 is a combination of HTML, JavaScript, and CSS3. The problem with HTML5 is that not all browsers support HTML5 markup and those that do support it, do not always support it the same way. Modernizer helps you get a consistent experience for your end users across multiple browsers. Even older browsers like Internet Explorer 6 can benefit from Modernizr.

Modernizr detects the actual HTML5 features that a browser supports. It does this by first creating an element, setting a style instruction on the element, and then retrieving it. If a browser does not support the instruction, it will return an error or undefined.

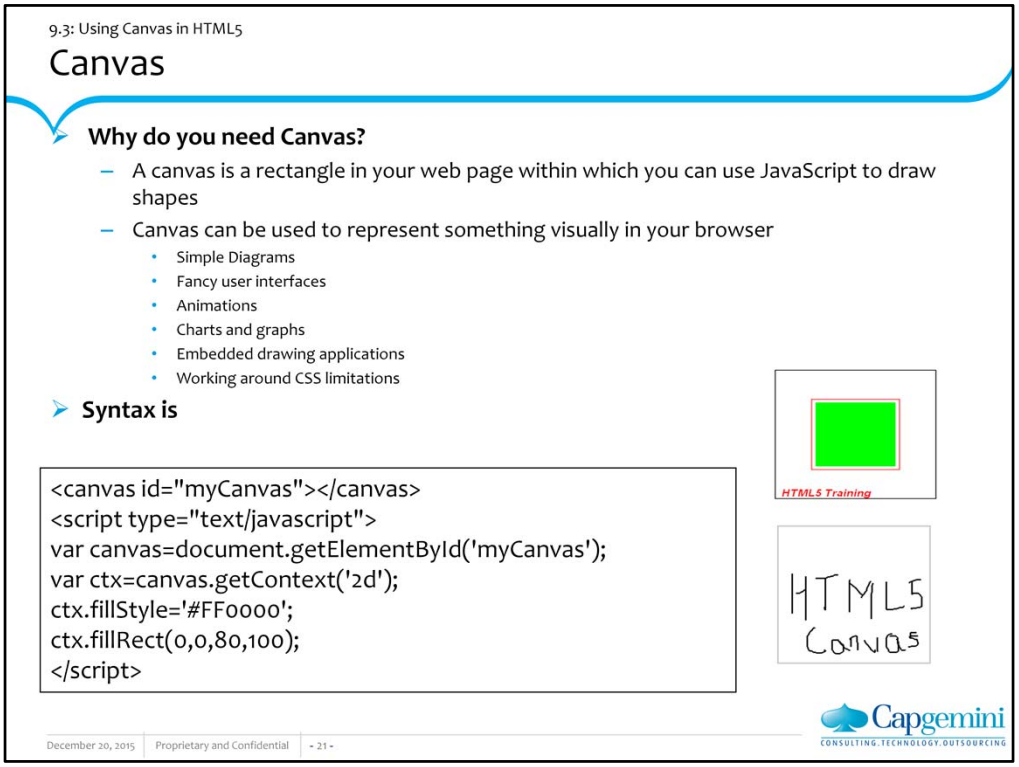
The first thing you need to do is get the Modernizr library. There are two ways to do this. First, you can use a Content Delivery Network (CDN) like the one from Microsoft. The second (and best) way is to use a custom build from Modernizr.com This option allows you to choose only the portions you want. This results in a smaller file.

Modernizr is referenced just like any JavaScript code. Modernizr should be place after your CSS references.
Place a script reference in the head tag of your HTML document:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Dive Into HTML5</title>
  <script src="modernizr.min.js"></script>
</head>
<body>
  ...
</body>
</html>
```

Modernizr runs automatically. There is no modernizr_init() function to call. When it runs, it creates a global object called Modernizr, that contains a set of Boolean properties for each feature it can detect. For example, if your browser supports the canvas API, the Modernizr.canvas property will be true. If your browser does not support the canvas API, theModernizr.canvas property will be false.

```
if (Modernizr.canvas)
{ // let's draw some shapes! } else { // no native canvas support available :( }
```



Canvas

HTML5 defines the <canvas> element as "a resolution-dependent bitmap canvas that can be used for rendering graphs, game graphics, or other visual images on the fly." A canvas is a rectangle in your page where you can use JavaScript to draw anything you want. HTML5 defines a set of functions ("the canvas API") for drawing shapes, defining paths, creating gradients, and applying transformations.
If your browser supports the canvas API, the DOM object it creates to represent a <canvas> element will have a getContext() method. If your browser doesn't support the canvas API, theDOM object it creates for a <canvas> element will only have the set of common properties, but not anything canvas-specific.

function supports_canvas()
{ return !!document.createElement('canvas').getContext; }

This function starts by creating a dummy <canvas> element. But the element is never attached to your page, so no one will ever see it. It's just floating in memory, going nowhere and doing nothing.

As soon as you create the dummy <canvas> element, you test for the presence of a getContext() method. This method will only exist if your browser supports the canvas API. This function will detect support for most of the canvas API, including shapes, paths, gradients & patterns. It will not detect the third-party explorercanvas library that implements the canvas API in Microsoft Internet Explorer.

Instead of writing this function yourself, you can use Modernizr to detect support for the canvas API as seen earlier.

Summary

- **In this lesson, you have learnt about:**
  - HTML5 introduces new and enhanced form elements those who caters to every need of modern Web Designing
  - It also introduces a new JavaScript library, Modernzr, that helps you detect HTML5 feature capabilities of your websites

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Review Question

- **Question 1: A _____ is a textbox that hold a text in lighter shade when there is no value and not focused**
- **Question 2: A _____ is a JavaScript library that helps you add HTML5 capabilities to your websites**

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING