Build COMPETENCY
across your TEAM

SYNERGETICS
GET IT RIGHT

Microsoft Partner
Gold Cloud Platform
Silver Learning

# Day 3. HTML 5

Smita B Kumar

Google Cloud Platform

Windows Azure

amazon web services

APACHE kafka
A distributed streaming platform

Apache Software Foundation

APACHE Spark

Java

hadoop

# HTML 5 - Topics Overview

- What is HTML5?
- Responsive Web Design
- HTML5 – New Form
- Audio and Video Support
- Canvas

Build COMPETENCY
across your TEAM

SYNERGETICS
GET IT RIGHT

Microsoft Partner
Gold Cloud Platform
Silver Learning

# Introduction to HTML5

# Objectives

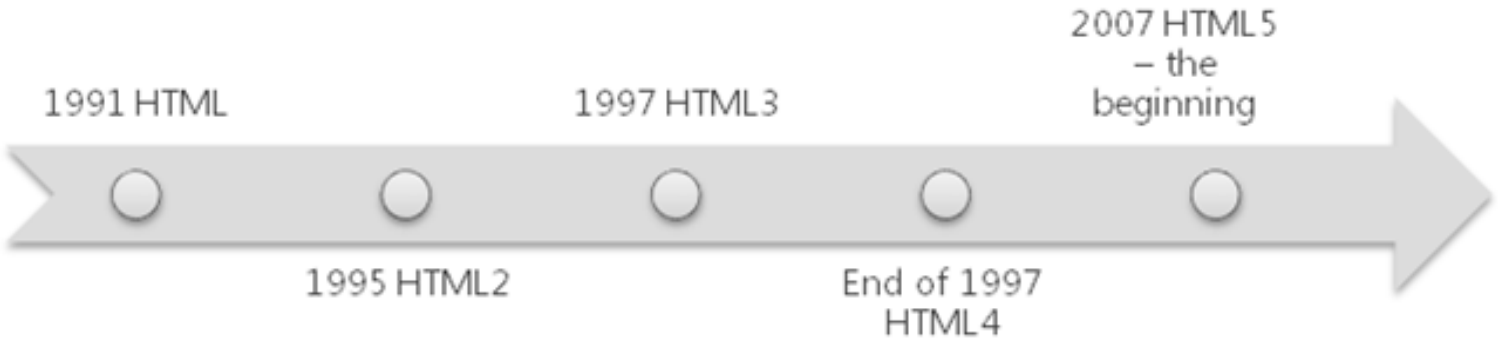
SYNERGETICS GET IT RIGHT | Microsoft Partner Gold Cloud Platform Silver Learning

**At the end of this sub-module you will know:**

- **What is HTML 5?**
- **Rules for HTML 5**
- **Browser support**
- **Features of HTML 5**
- **New elements in HTML 5**
- **New markup elements**

# What Is HTML 5

- The new version of HTML
- A new and emerging set of web standards and specifications
- HTML5 is HTML + CSS3 + JavaScript APIs
- The future of the web
- Still under development
- A lot of features supported by modern browsers

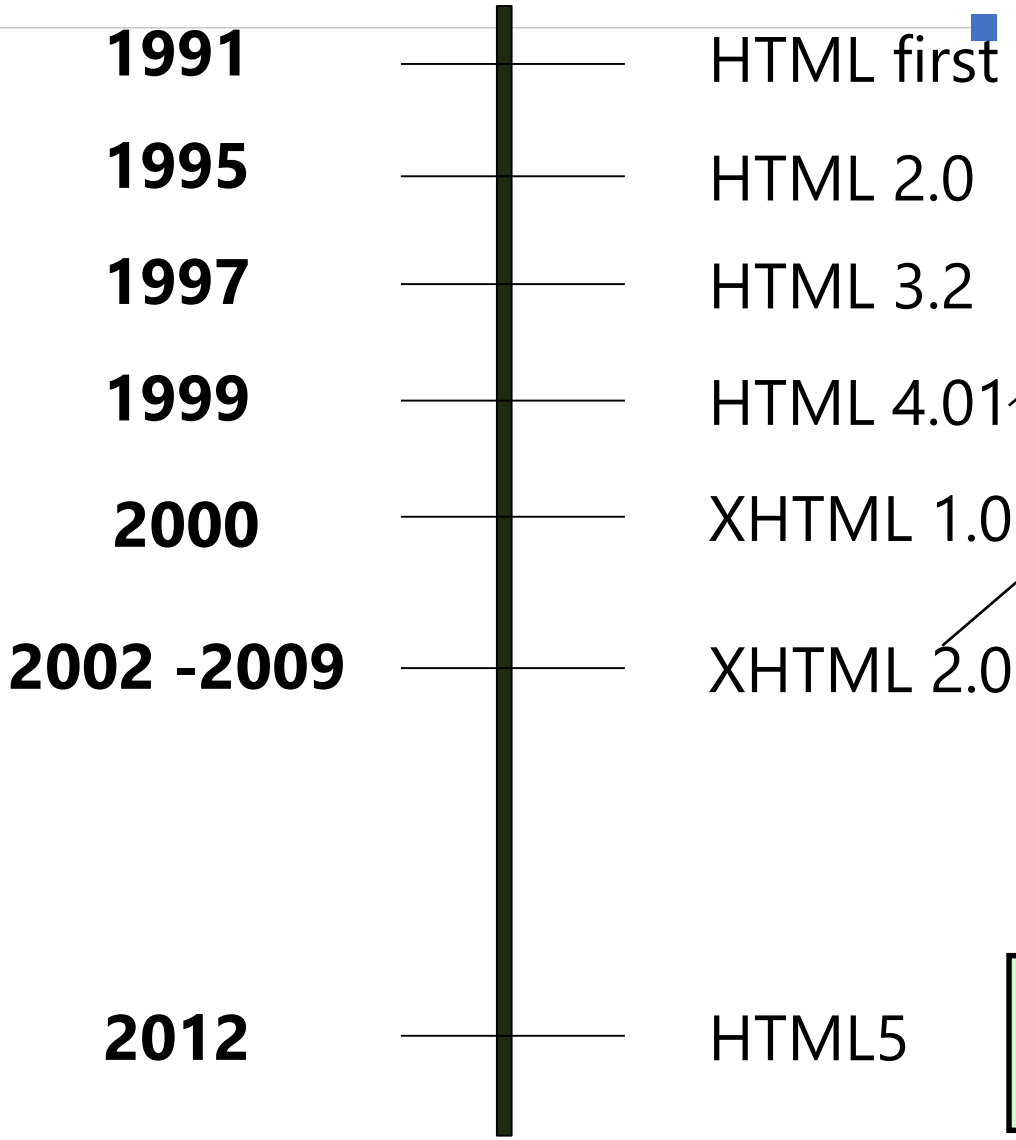



# Why to use HTML5?

SYNERGETICS GET IT RIGHT | Microsoft Partner Gold Cloud Platform Silver Learning

- ✓ **Accessibility**
- ✓ **Searchability**
- ✓ **Interoperability**
- ✓ **Many new**
  - **HTML elements and attributes**
  - **JavaScript APIs and improved capabilities**
  - **CSS capabilities**

# History of HTML




After HTML 4.01 was released, focus shifted to XHTML and its stricter standards.

XHTML 2.0 had even stricter standards than 1.0, rejecting web pages that did not comply. It fell out of favor gradually and was abandoned completely in 2009.

HTML5 is much more tolerant and can handle markup from all the prior versions.

Though HTML5 was published officially in 2012, it has been in development since 2004.

# What is HTML 5 (Contd.).



- HTML5 is the next generation of HTML
- It will be the new standard for HTML,XHTML, and HTML DOM
- HTML5 work is still in progress and most modern browsers have some HTML5 support
- It is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHTWG)
- WHATWG was working with web forms and applications, and W3C was working with XHTML 2.0
- In 2006, they decided to collaborate and create a new version of HTML

# What is HTML 5 (Contd.).



- It is a language for structuring and presenting content for the WWW
- It is a core technology of the Internet originally proposed by Opera Software
- The core aim of HTML5 is
  - Is to improve the language with support for multimedia
  - Easily read by humans
  - Understood by computers and devices
- The HTML5 working group includes Apple, Google, AOL, IBM, Microsoft, Mozilla, Nokia, Opera, and many hundreds of other vendors.

# Goals of HTML5

- Support all existing web pages. With HTML5, there is no requirement to go back and revise older websites.
- Reduce the need for external plugins and scripts to show website content.
- Improve the semantic definition (i.e. meaning and purpose) of page elements.
- Make the rendering of web content universal and independent of the device being used.
- Handle web documents errors in a better and more consistent fashion.

# Rules for HTML 5



- New features should be based on HTML, CSS, DOM, and JavaScript
- Usage of external plug-ins (like Flash) should be reduced
- Strict parsing rules are introduced to handle any errors.
- More markup to replace scripting
- Device independence should be there

# HTML5 and the Open Web Platform



- The Open Web Platform (OWP) is a collection of web technologies
- It is developed by W3C and other web standardization bodies
- HTML5 is part of the OWP (Open Web Platform)

# New Features of HTML5



- New canvas element is available for drawing
- New video and audio elements are available for media playback
- Better support for local offline storage
- New content specific elements, like article, mark, progress, meter
- New form controls, like calendar, date, time, email, url, search
- Drag and Drop functionality
- Support for CSS3 (the newer and more powerful version of CSS)
- Geolocation, Working with GoogleMaps
- Web Workers
- WebSockets
- Server Sent Event

# The New Elements in HTML5



- The **canvas** element
- The **video** and **audio** elements
- New content-specific elements:
  - **article**, **footer**, **header**, **nav**, **section**, and more
- New form controls:
  - **calendar**, **date**, **time**, **email**, **url**, **search**, and more

# New Semantic/Structural Elements

- <article>
- <aside>
- <bdi>
- <command>
- <details>
- <dialog>
- <summary>
- <figure>
- <figcaption>
- <footer>
- <header>
- <mark>
- <meter>
- <nav>
- <progress>
- <ruby>
- <rt>
- <section>
- <time>
- <wbr>

# Browser Support

- No browsers have full HTML5 support as HTML 5 is yet not an official standard

- But all major browsers (Chrome, Safari, Firefox, Opera, Internet Explorer) continue to add new HTML5 features to their latest versions

  - Internet Explorer 9, Firefox, Opera, Chrome, and Safari support the <video> element.

Note: Internet Explorer 8 and earlier versions, do not support the <video> element.

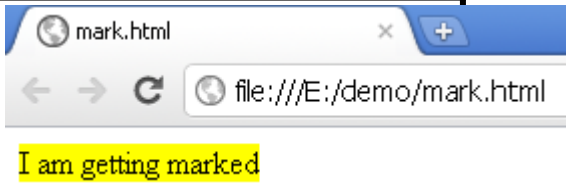
# New Elements in HTML5



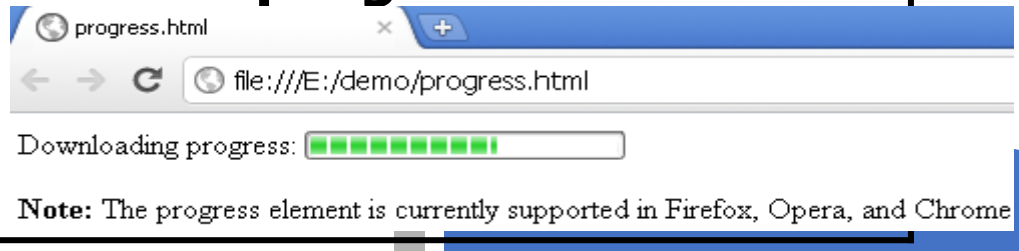
- HTML5 includes new elements for better structure, drawing, media content and better form handling
- New media elements like <audio>, <video>,<source>
- Canvas element <canvas> uses JavaScript to make drawing on a web page
- The Input elements type attribute has many new values, for better input control before sending it to the server

# Some new markup elements

| | |
|---|---|
| <article> | Used to specify a news-article, blog post, forum post, or other articles which can be distributed independently from the rest of the site |
| <mark> | For text that should be highlighted |
| <meter> | For a measurement, used only if the maximum and minimum values are known |
| <progress> | The <progress> tag represents the progress of a task |







# First Look at HTML5

Remember the DOCTYPE declaration from XHTML?

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

In HTML5, there is just one possible DOCTYPE declaration and it is simpler:

```
<!DOCTYPE html>
```

Just 15 characters!

The DOCTYPE tells the browser which type and version of document to expect. This should be the last time the DOCTYPE is ever changed. From now on, all future versions of HTML will use this same simplified declaration.

# The Doctype Element

- A declaration of document type
- Must be provided by any HTML document
- Defines the rendering mode of the browser
- Was created to enable HTML parsing and validation

```
<!DOCTYPE html>
<html>
```

# The <html> Element

This is what the <html> element looked like in XHTML:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

Again, HTML5 simplifies this line:

```
<html lang="en">
```

The **lang** attribute in the <html> element declares which language the page content is in. Though not strictly required, it should always be specified, as it can assist search engines and screen readers.

Each of the world's major languages has a two-character code, e.g. Spanish = "es", French = "fr", German = "de", Chinese = "zh", Arabic = "ar".

# The <head> Section

Here is a typical XHTML <head> section:

```
<head>
  <meta http-equiv="Content-type" content="text/html; charset=UTF-8" />
  <title>My First XHTML Page</title>
  <link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

And the HTML5 version:

```
<head>
  <meta charset="utf-8">
  <title>My First HTML5 Page</title>
  <link rel="stylesheet" href="style.css">
</head>
```

Notice the simplified character set declaration, the shorter CSS stylesheet link text, and the removal of the trailing slashes for these two lines.

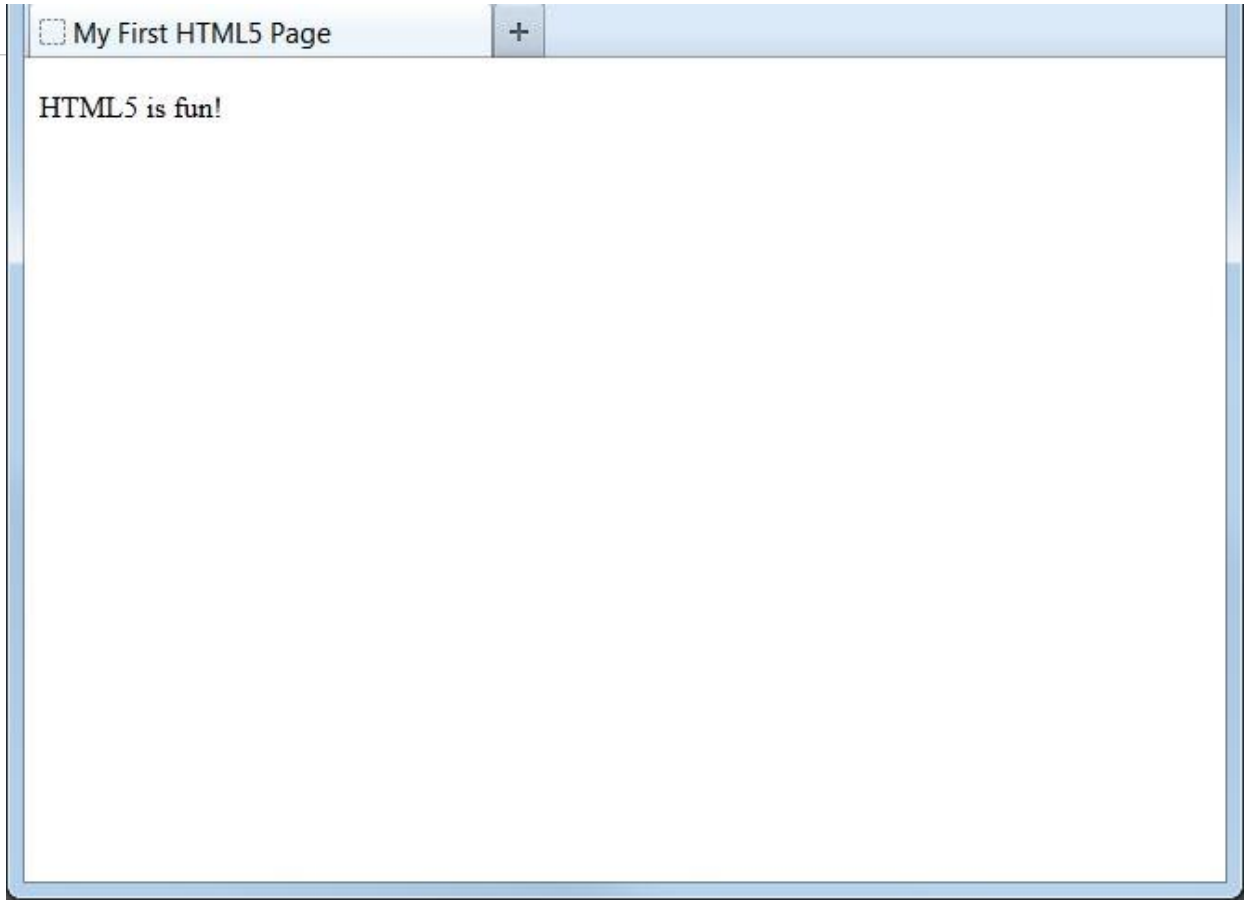
# Basic HTML5 Web Page

Putting the prior sections together, and now adding the <body> section and closing tags, we have our first complete web page in HTML5:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>My First HTML5 Page</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <p>HTML5 is fun!</p>
</body>
</html>
```

Let's open this page in a web browser to see how it looks…

# Viewing the HTML5 Web Page




Even though we used HTML5, the page looks exactly the same in a web browser as it would in XHTML. Without looking at the source code, web visitors will not know which version of HTML the page was created with.

# HTML 4.01 elements removed from HTML%

- The below are the  list of HTML 4.01 elements which are removed from HTML5.
  - <acronym>
  - <applet>
  - <basefont>
  - <big>
  - 
  - <dir>
  - <font>
  - <frame>
  - <frameset>
  - <noframes>
  - <strike>
  - <tt>

# Summary

- **At the end of this sub module:**
  - **Understand HTML 5**
  - **Different browser support for HTML 5**
  - **Different HTML 5 elements**

# HTML 5 Structural Elements

# Objectives

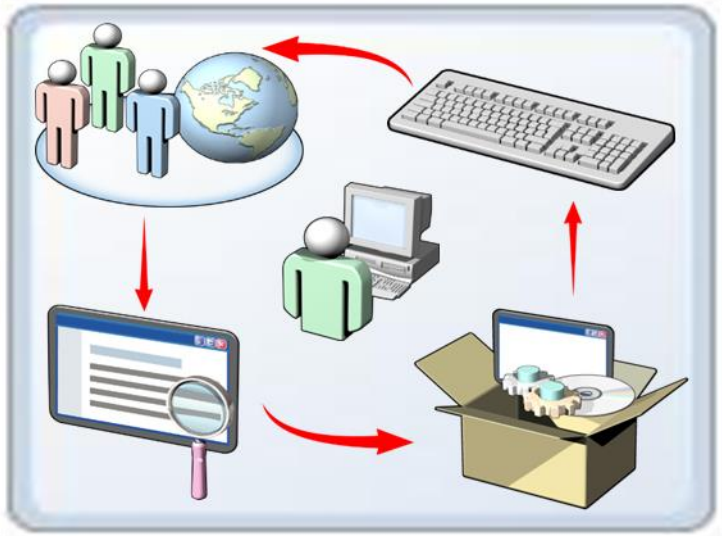
- Explain the rationale behind the new HTML5 elements:
- Describe where and how to use the new HTML5 structural elements
- Describe what are the new semantic block-level elements
- Describe what are the new inline semantic elements
- Describe what are the new interactive elements

# New HTML 5 – Rationale Behind

SYNERGETICS GET IT RIGHT | Microsoft Partner Gold Cloud Platform Silver Learning

## New HTML5 elements enable:

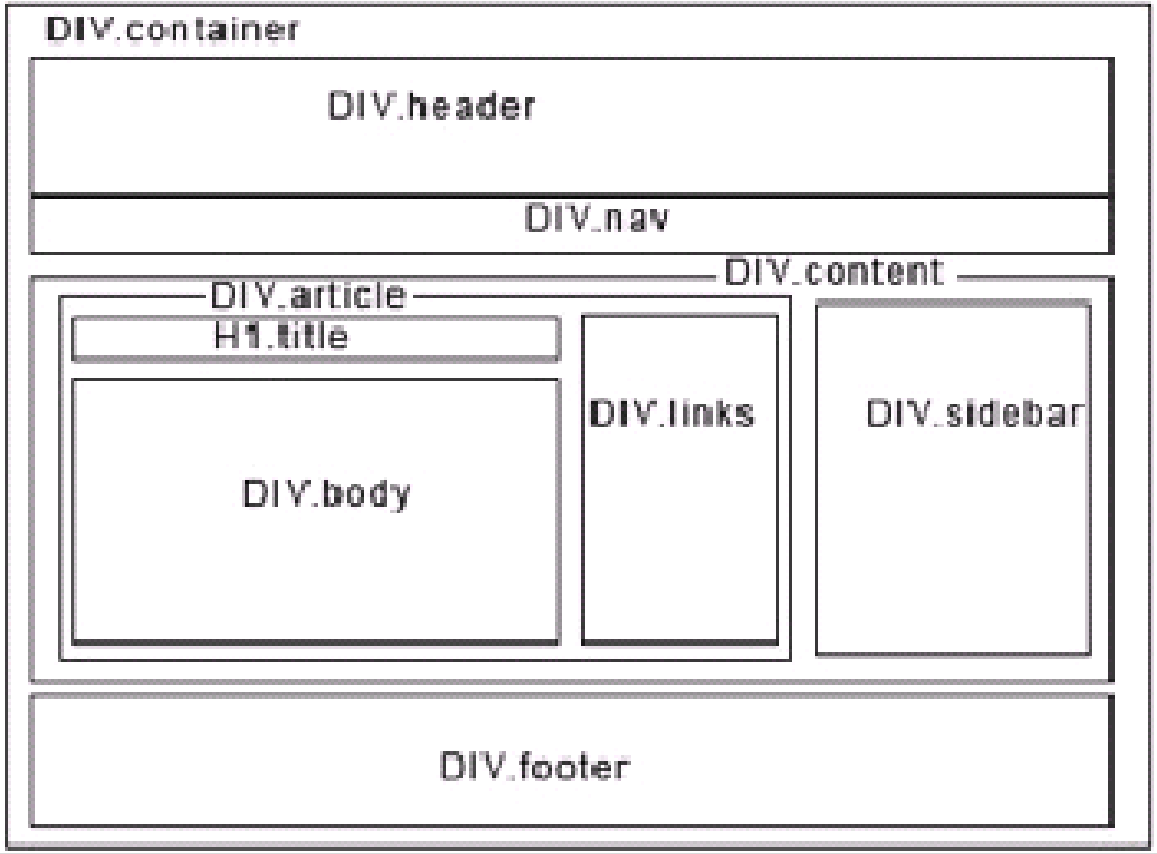
- Accessibility
- Searchability
- Internationalization
- Interoperability



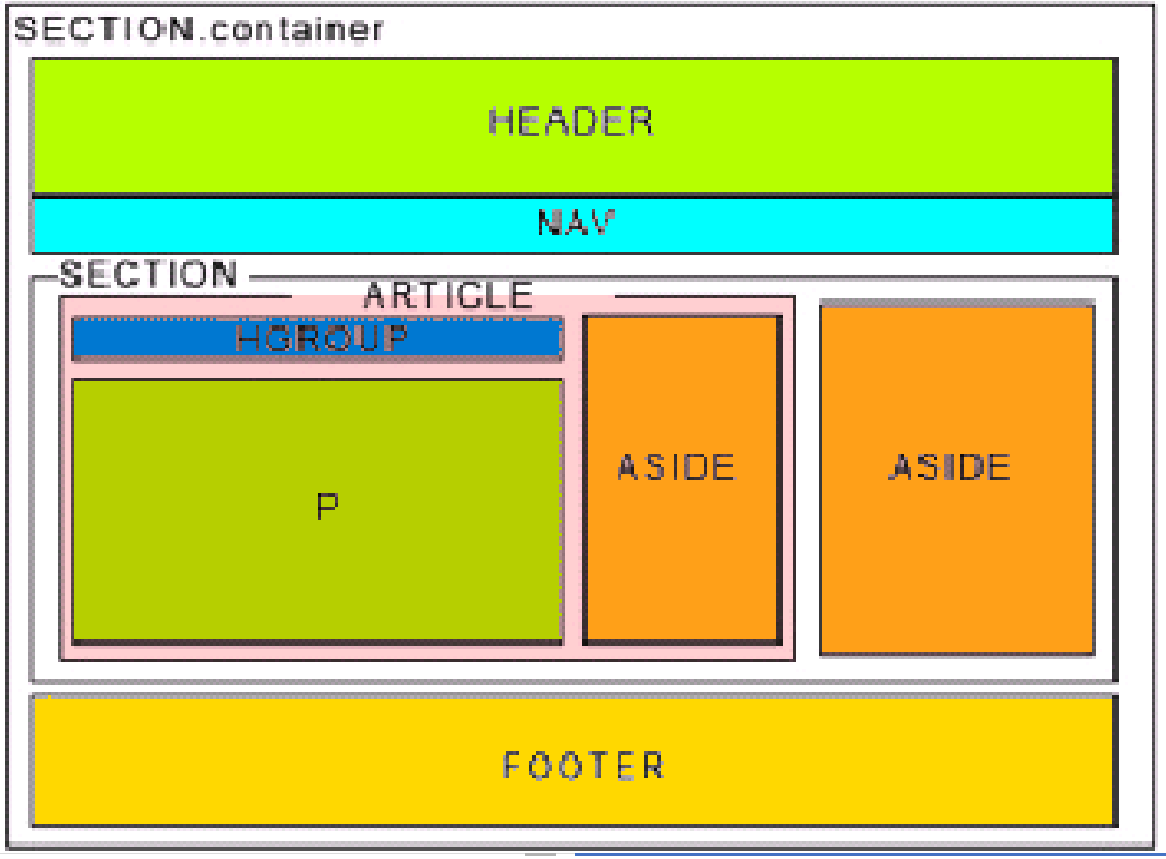
# Structural Elements

- No need for div elements all over the place
- New ways to mean what you actually meant to mean

## HTML 4




## HTML 5



# The New Structural Elements

- New structural HTML5 elements:
  - **Article**
  - **Footer**
  - **Header**
  - **Hgroup**
  - **Section**
  - **Address**
  - **Nav**
  - **Aside**
- No need for **div** elements all over the webpage

```
<body>
    <header>
        <nav>
            <ul>
                <li>Navigation</li>
            </ul>
        </nav>
    </header>
    <section>
        <article>
            <p>Article content</p>
        </article>
    </section>
    <footer>
        <p>Copyright</p>
    </footer>
</body>
```

# Main Structural Elements

- **Article :** Defines an article (for example within a section)
- **Footer :** Footer elements contain information about their containing element: who wrote it, copyright, etc.
- **Header :** The page header shown on the page, not the same as <head>
- **Nav :** Collection of links to other pages
- **Section :** A part or chapter in a book, or essentially the content body of the page

**And More**

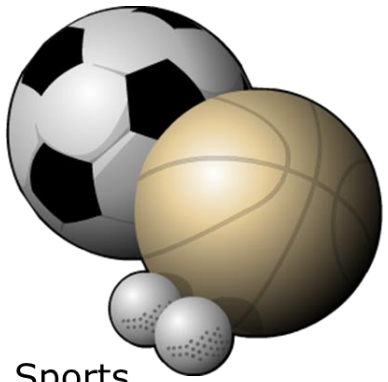
# Semantic Block-Level Elements



## Semantic Block-Level Elements

- Block elements that have a specific semantic meaning
- Include the figure element

```
<figure>
  <img src="imageurl" alt="image alt">
  <figcaption>Sports</figcaption>
</figure>
```

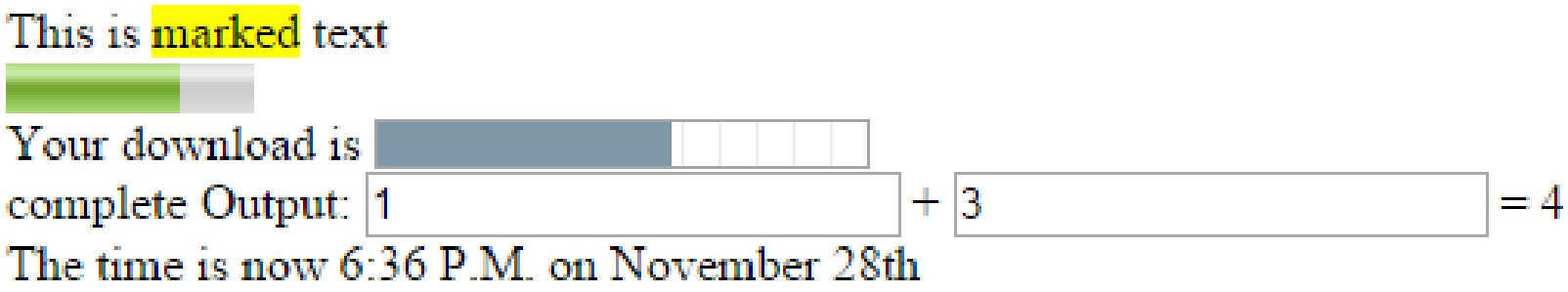


Sports

# Inline Semantic Elements

- Inline elements affect their content.
- Examples:
  - **Mark**
  - **Meter**
  - **Progress**
  - **Output**
  - **Time**

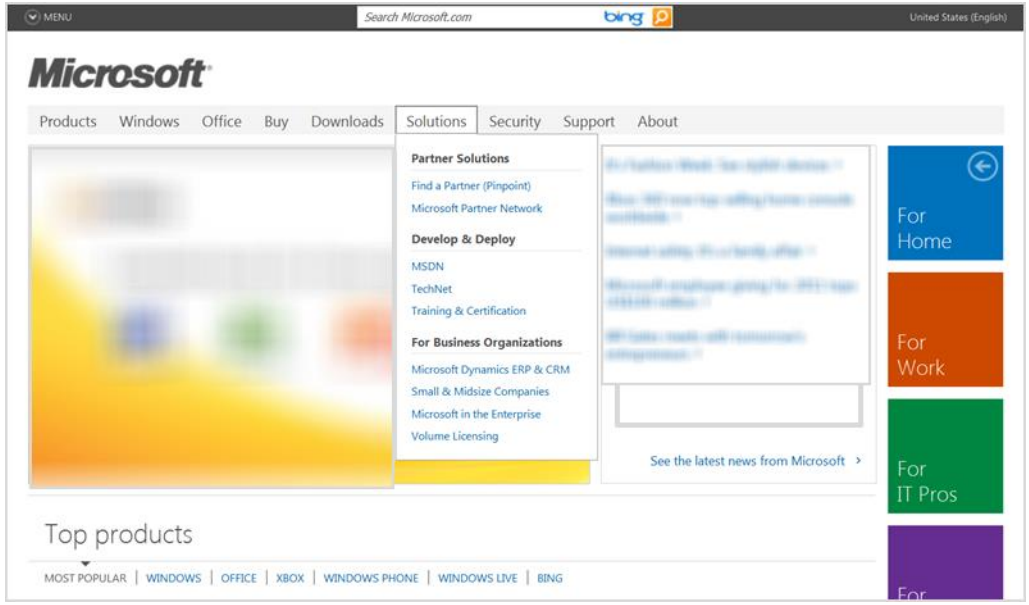
```
This is <mark>marked</mark> text
<meter value="70" min="0" max="100" low="54" high="100"
optimum="100">70</meter>
Your download is <progress max="100" value="60">60%</progress>
complete
Output: <input type="text" value="1" /> + <input type="text" value="3" />
= <output>4</output>
The time is now <time datetime="2011-09-28T18:36">6:36 P.M. on
November 28th</time>
```



# Interactive Elements

- Interactive elements include:
  - **Details**
  - **Summary**
  - **Menu**
  - **Command**

```
<details>
  <summary>The summary</summary>
  <p>Some details</p>
</details>
```




▼ The summary

Some details

# Summary

SYNERGETICS GET IT RIGHT | Microsoft Partner Gold Cloud Platform Silver Learning

**In the module, you learned about:**

- **The new HTML5 elements and the rationale behind the creation of these elements.**
- **How and when to use them.**
- **The use of navigation and menus.**

# Hands On Assignment

SYNERGETICS GET IT RIGHT | Microsoft Partner Gold Cloud Platform Silver Learning

**Practice the below exercises in the system**

- **Create simple web pages using HTML tags**
- **For the above webpages create a simple home page using menu and navigation tags**

# Creating Form Input and Validation

# Objectives

SYNERGETICS GET IT RIGHT | Microsoft Partner Gold Cloud Platform Silver Learning

**At the end of this sub-module you will know:**

- **HTML 5 new Input types**
- **HTML 5 new form elements**
- **HTML 5 new form attributes**
- **Using Browser Detection, Feature Detection, and Modernizr**

# Challenges with HTML 4.0 Forms



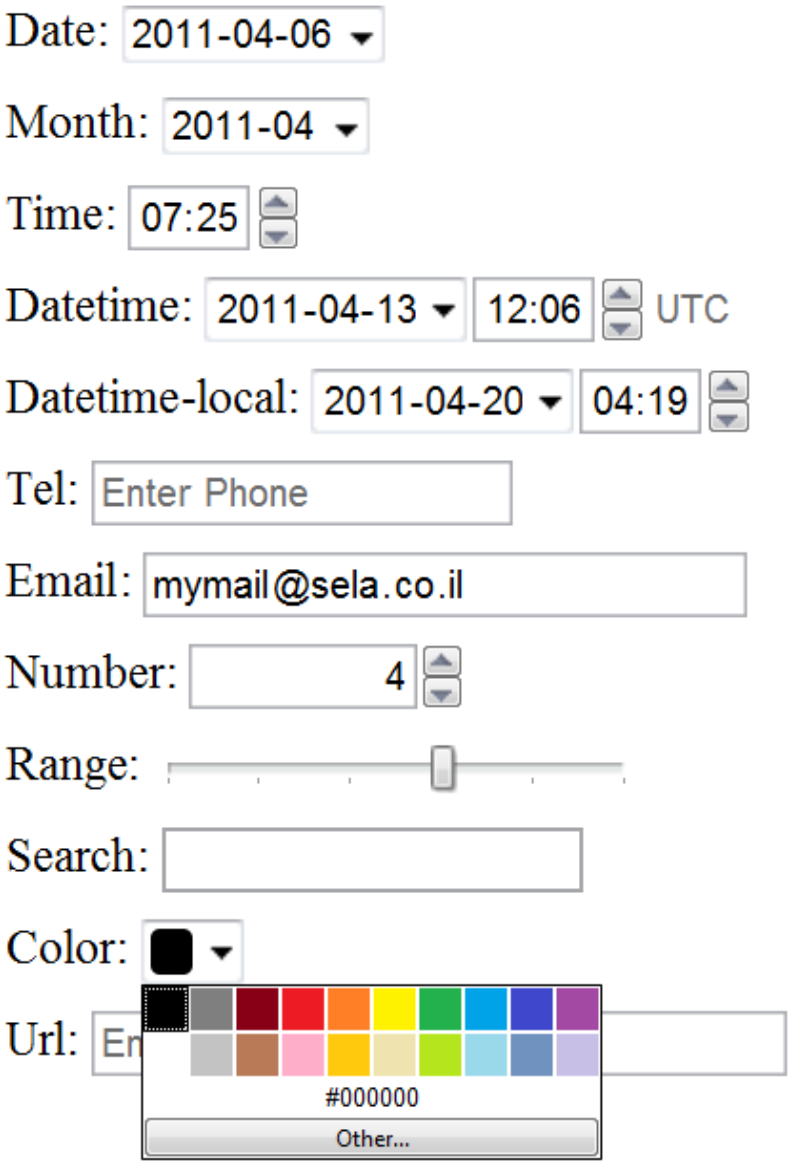
- The HTML 4 form fields are not validated
- The validation to the input fields can be done by
  - Writing JavaScript code in the Client side Code
  - Or the validation needs to be done at the Server side
- Almost every web page has some kind of form with search, email, sign-up etc
- It would be great if browsers had built in validation for some of the common data types we collect

# HTML – New Input types

- HTML 5 has lot of new input types
- These new features allow for better input control and validation
  - color
  - date
  - datetime
  - datetime-local
  - email
  - month
  - number
  - range
  - search
  - tel
  - time
  - url
  - week

# HTML – New Input types (Contd.)

| Types |
| --- |
| Email |
| Url |
| Tel |
| Number |
| Range |
| Search |
| Color |
| Date pickers (date, month, week, time, datetime, datetime local) |



# Input type - email

- It represents a control for editing a list of e-mail addresses given in the element's value
- HTML email address validation will only work in browsers which have support for it

```
<input type="email" placeholder="Please enter your email" required
        multiple="multiple">
```

- required - specifies that the element is a required part of form submission
- placeholder - a short phrase to help the user when entering data into the control

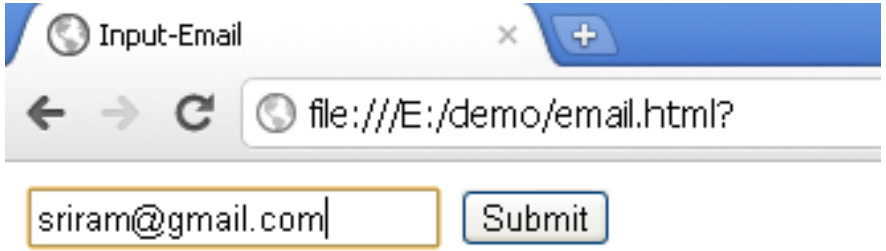
- multiple - specifies that the element allows multiple values

# Input type – email (Contd.).

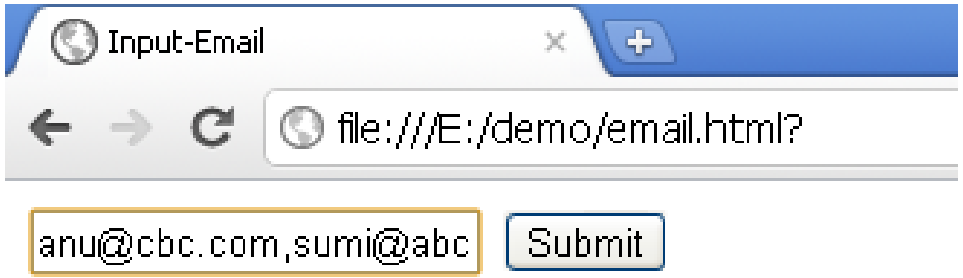
## Field is empty


## The value should be an email


## Valid email – with a single email




## Valid-More than 1 email



# Input type - url

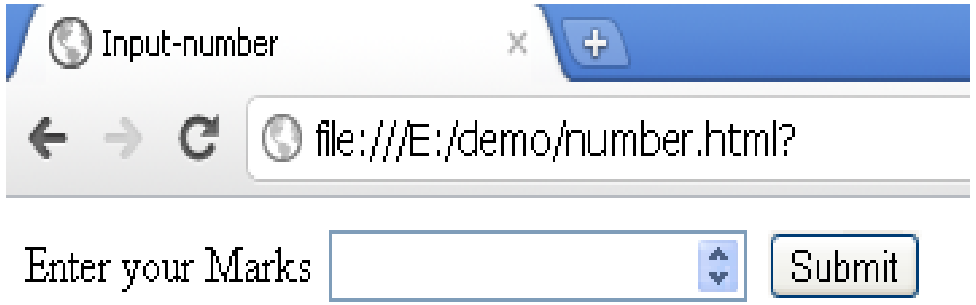
```
<input type="url" name="URL">
```

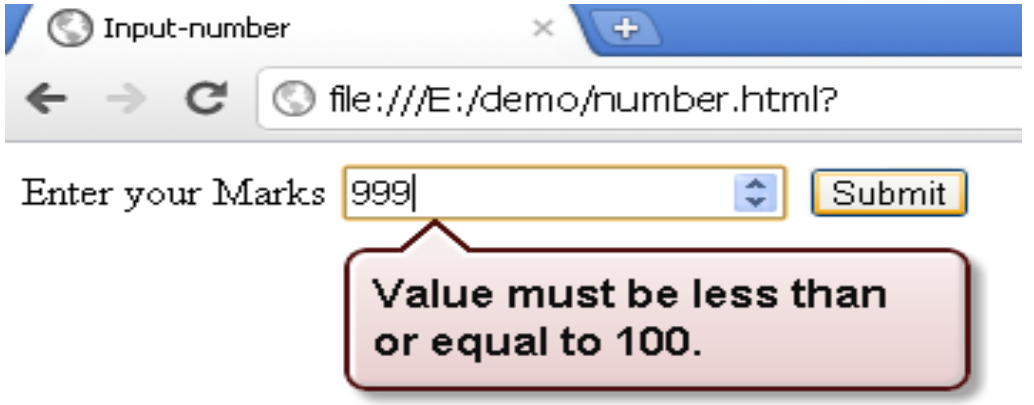
- It is used for input fields that should contain a URL address
- The value of the url field is automatically validated when the form is submitted

# Input type - number

<input type="number" min="0" max="100">

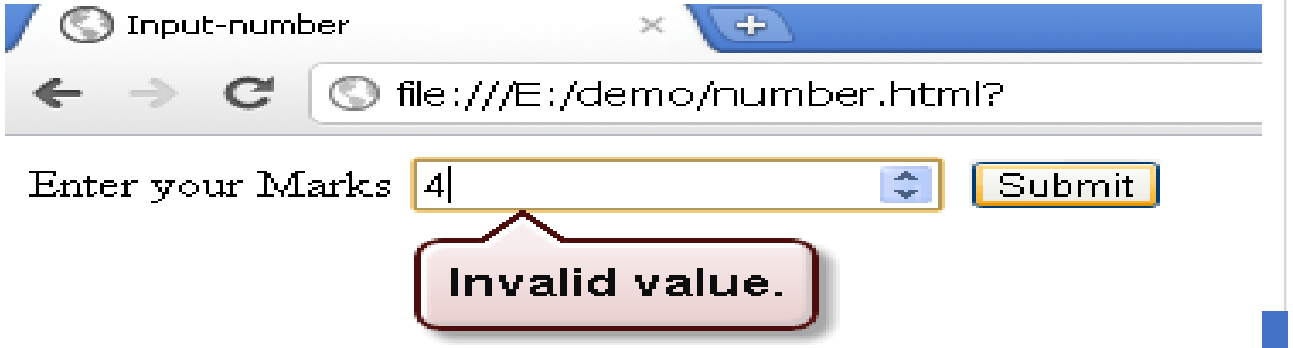
- Used to take a number as input




<input type="number" min="0 " max="100" step="3">

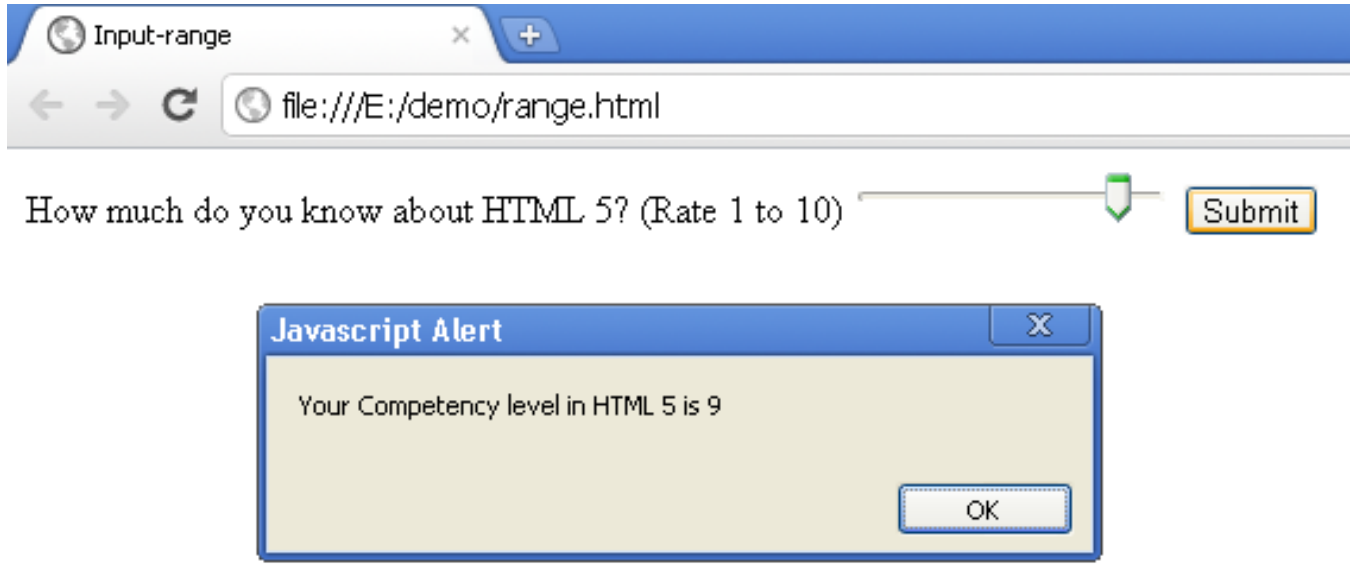
- step - specifies legal number intervals (if step="3", legal numbers could be -3,0,3,6, etc)

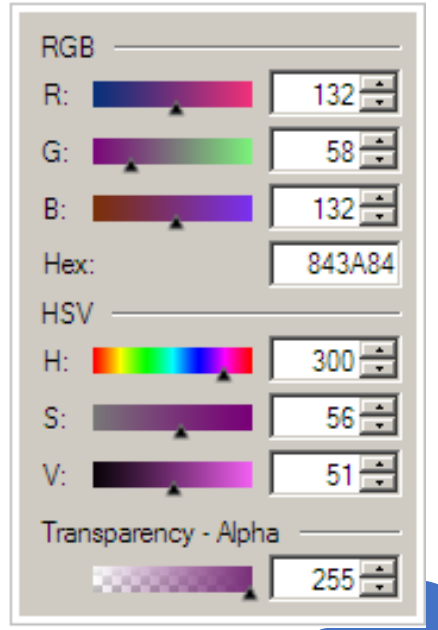


# Input type - range

- Slider control is a very useful user interface to set a number within a range

```
<input type="range" name="Range" min="1" max="10" required>
```

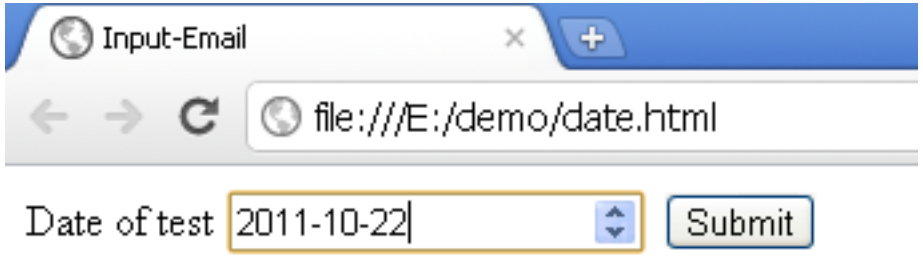


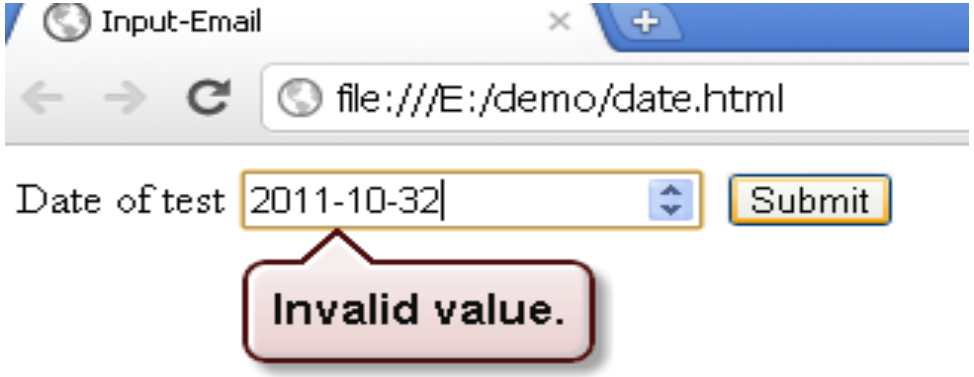



# Input type – Date pickers - date

- Date and time field can be easily found in many web forms
- Typical applications are like ticket booking, appointment booking etc
- HTML5 the web browser ensures user can only enter a valid date time string into the input textbox

```
<input type="date" name="set_date" />
```

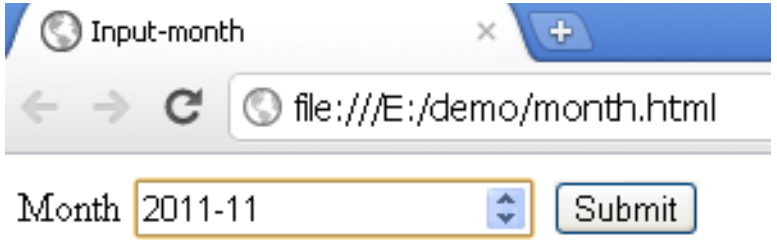


# Input type – Date pickers – date (Contd.).

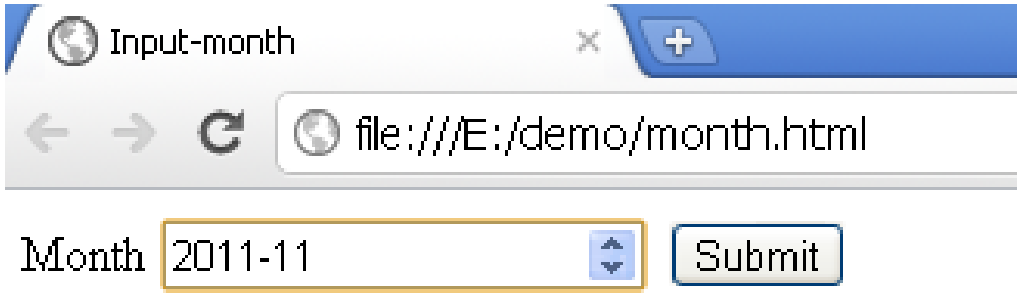
```
<input type="month" name="month" />
```

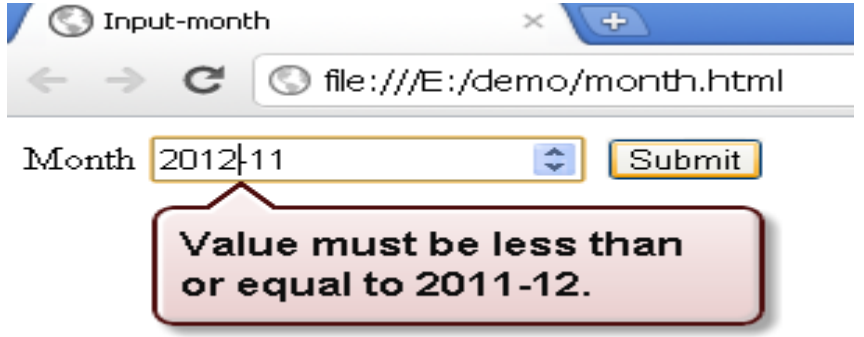
- Creates a date input control for specifying a particular month in a year





```
<input type="month" name="month" min="2011-1" max="2011-12"/>
```

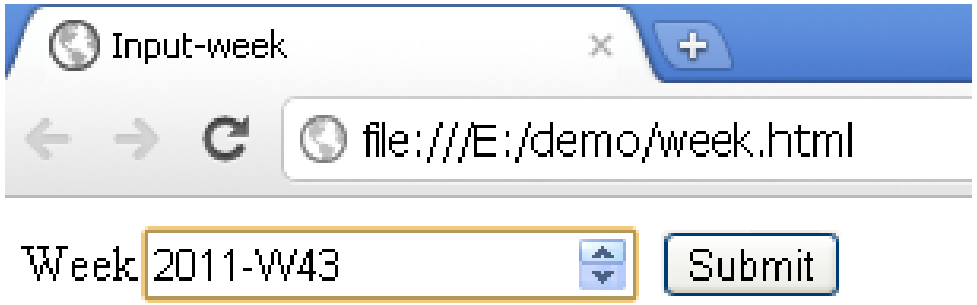


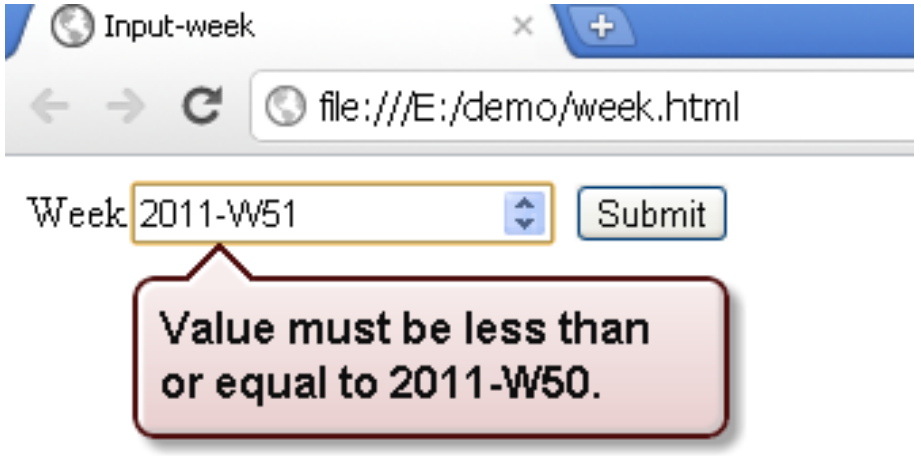



# Input type – Date pickers – week

```
<input type="week" name="week" min="2011-W15" max="2011-W50"/>
```

- Allows entry and validation of a week number

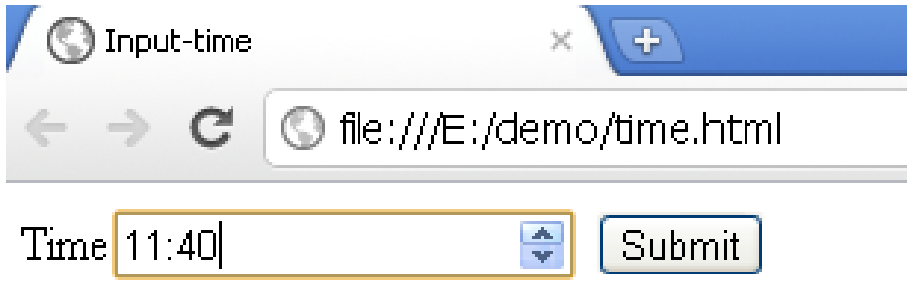


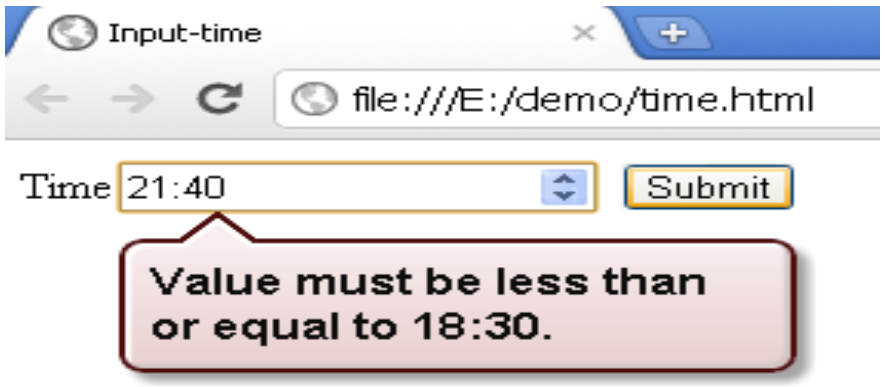



# Input type – Date pickers – time

```
<input type="time" name="Time" min="11:40" max="18:30">
```

- **Allows entry and validation of a valid time**

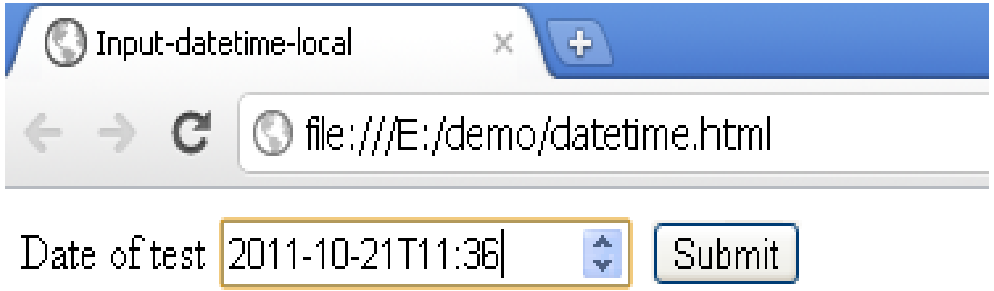


# Input type – Date pickers – datetime-local

```
<input type="datetime-local" name="set_date" />
```

- It creates a combined date/time input field
- The value is an ISO formatted date and time




# Input type - Search

```
<input type="search" name="Search">
```

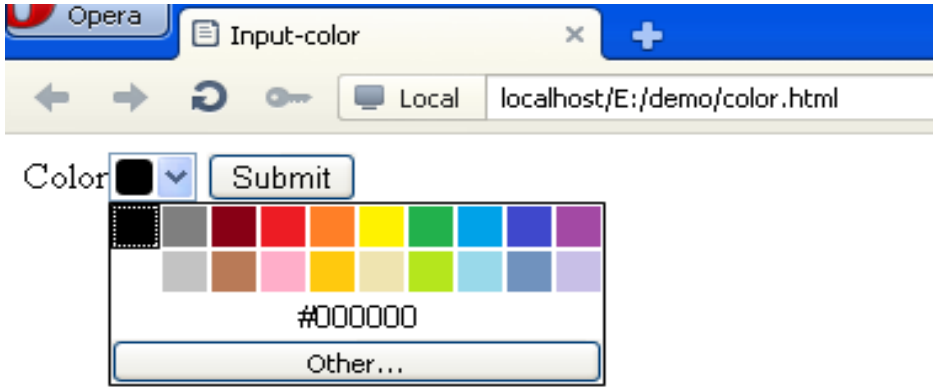
- It is used for search fields, like a site search, or Google search
- The difference between search and text type is only stylistic
- It takes the operating system's default rounded-corners style for search
- Currently Google Chrome does not support this

# Input type - Color

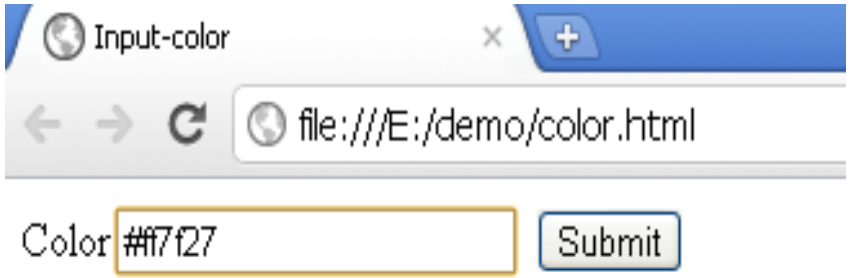
- Color <input type="color" name="Color" required>
- Used to create a color control for selecting a color value
- The Opera browser will allow you to select a color from a color picker

Color <input type="color" name="Color" required>

In Opera 11.52




In Google chrome 14.0



# HTML 5 Form Elements

- New Form elements added to HTML 5 are
  - <datalist> - A list of options for input values
  - <output> - For different types of output, such as output written by a script
  - <keygen> The purpose of the <keygen> element is to provide a secure way to authenticate users.

# HTML Form Attributes

- New Form Attributes
  - autocomplete
  - novalidate
- New Input Attributes
  - autocomplete
  - autofocus
  - form
  - formaction
  - formenctype
  - formmethod
  - formnovalidate
  - formtarget
  - height and width
  - min and max
  - multiple
  - pattern (regexp)
  - placeholder
  - required
  - Step
  - list

# Autocomplete attributes

- The **autocomplete** attribute instructs the browser to remember field values
- It work with <form> and the following input types
  - text, search, url, telephone, email, password, date pickers, range and color
- It can have two values on and off
- When the user starts to type in an auto complete field the browser should display options to fill in the field

**Enabling Auto-Completion for a Form Field**

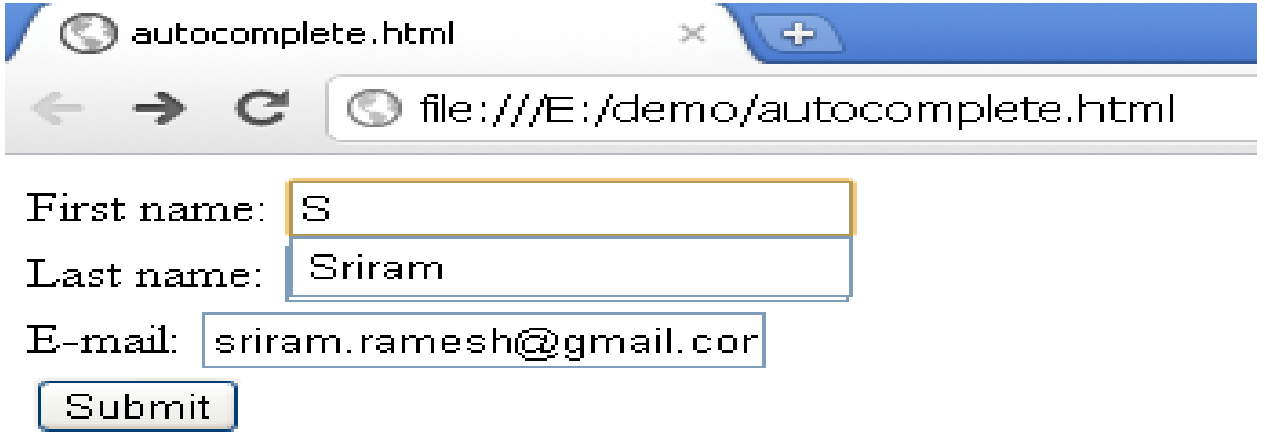
```
<input type="text" name="Username" autocomplete="on" />
```

**Disabling Auto-Completion for a Sensitive Form Field**

```
<input type="text" name="CreditCardNumber" autocomplete="off" />
```

# Autocomplete attribute (Contd.).

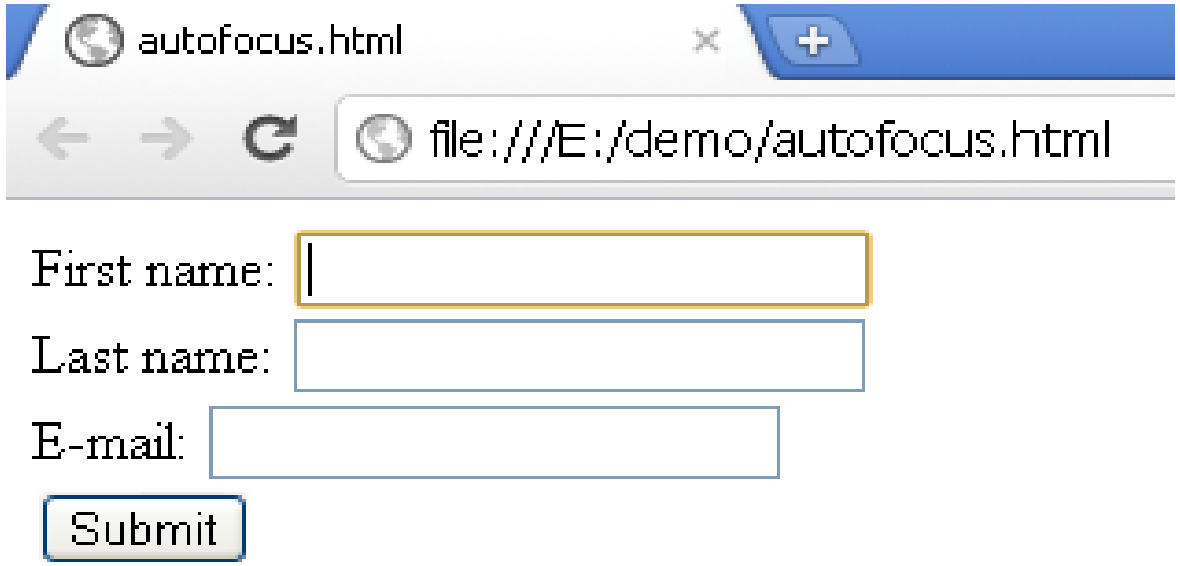
```
<form action="/webapp/demo1" method="get" autocomplete="on">
First name: <input type="text" name="fname" /><br />
Last name: <input type="text" name="lname" /><br />
E-mail: <input type="email" name="email" autocomplete="off" /><br />
<input type="submit" />
</form>
```



# autofocus attribute

- It specifies that a field should automatically get focus when a page is loaded
- This attribute works with all <input> types

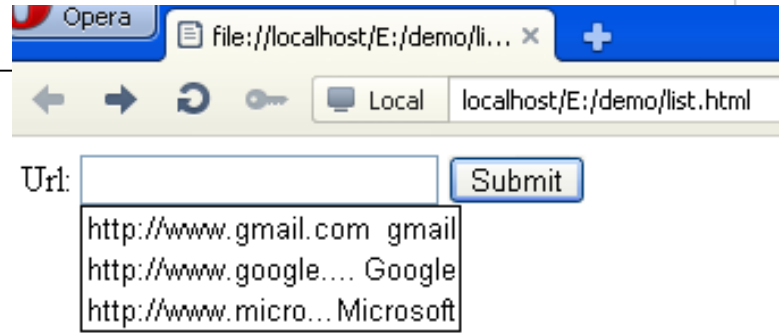
First name: <input type="text" name="fname" autofocus="autofocus" />



# list attribute

- The list attribute specifies a datalist for an input field
- A datalist is a list of options for an input field
- The list attribute works with the following <input> types
  - text, search, url, telephone, email, date pickers, number, range, and color

```
Url: <input type="text" list="url_list" name="url" />
<datalist id="url_list">
   <option label="gmail" value="http://www.gmail.com"
   <option label="Google" value="http://www.google.com" />
   <option label="Microsoft" value="http://www.microsoft.com" />
</datalist>
```

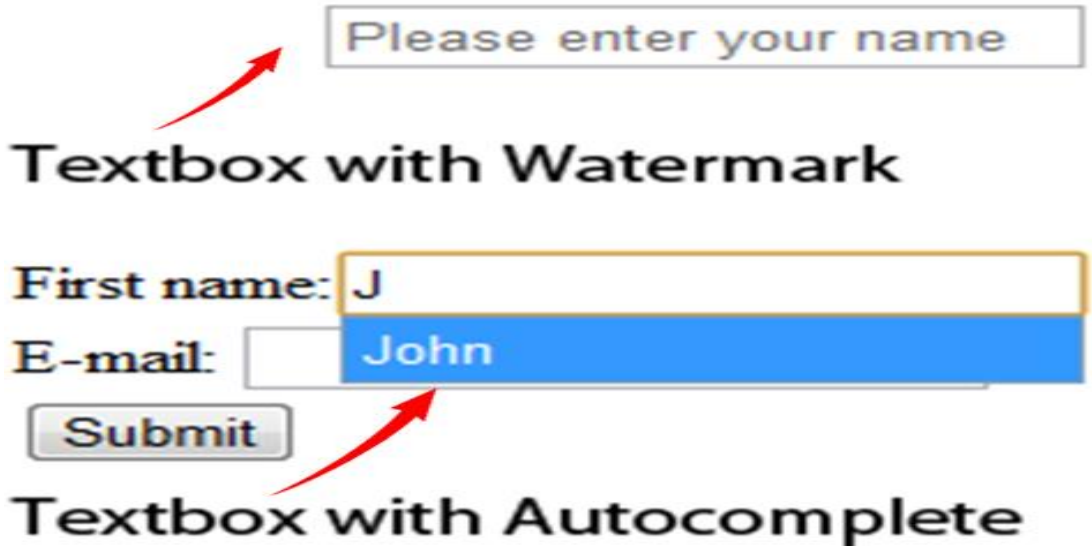


# list Attribute and the datalist Element

- The **list** attribute provides another mechanism for auto-completion.
- Whereas the **autocomplete** attribute instructs the browser to suggest values based on historical entries that the browser itself manages, the **list** attribute gives the web developer more control over the suggestions.
- A list of pre-defined values can be specified in a **datalist** element somewhere on the page, and referenced by any text field.
- The browser then uses the pre-defined values as suggestions when the user fills the text field.
- To specify the suggestions made available by a **datalist** element, use **option** elements.

# Usability Attributes

```
<input type="text" placeholder="Enter your full name" name="FullName" />
<input type="text" name="FullName" autofocus="autofocus" />
<input type="text" name="Username" autocomplete="on" />
```



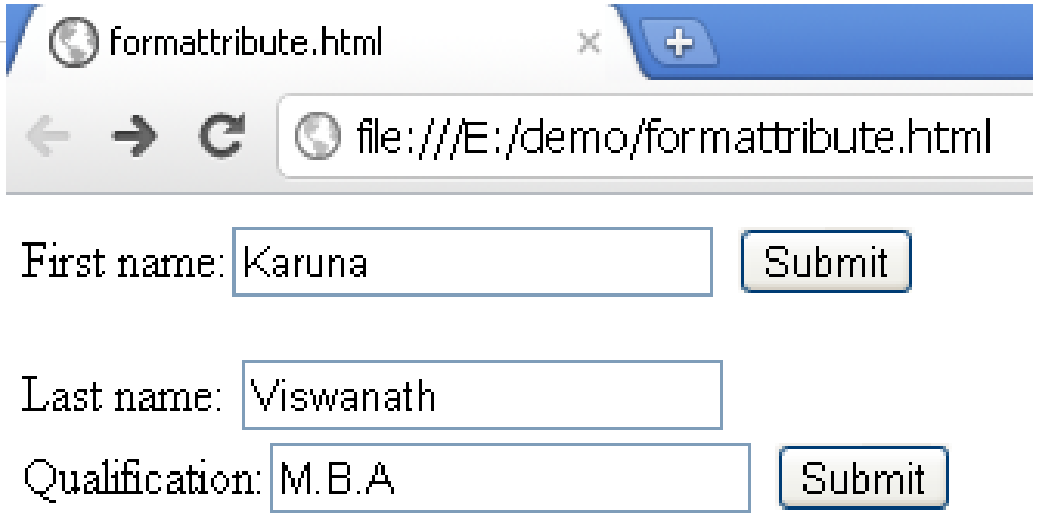
# form attribute

- In an HTML page there can be many number of forms but only one form can be submitted
- HTML 5 allows an element to use the form attribute and specify the form to which it belongs to

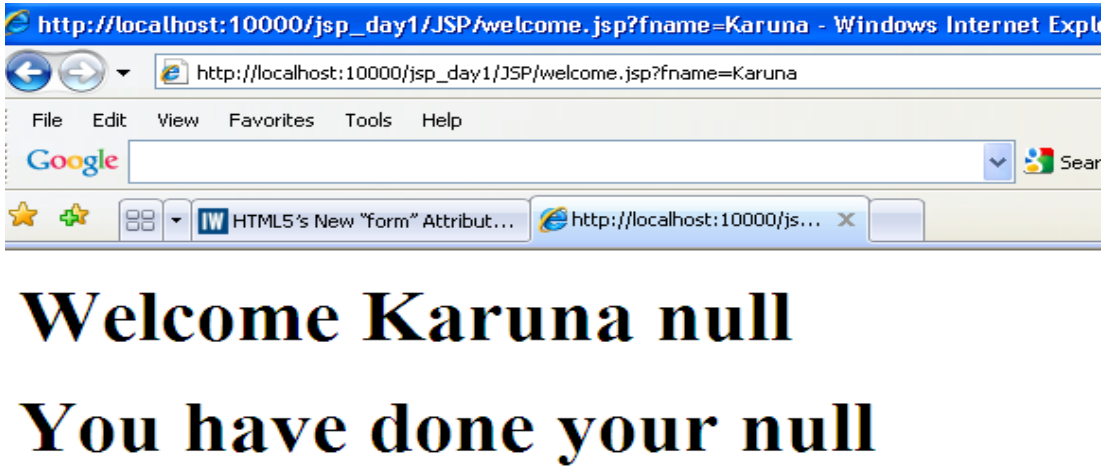
```
<form action="http://localhost:10000/webApp/default.aspx" method="get"
   id="user_form">
     First name:<input type="text" name="fname" />
     <input type="submit" />
</form>
Last name: <input type="text" name="lname" form="user_form" />

<form action="http://localhost:10000/webApp/default1.aspx" method="get"
   id="user_form2">
   Qualification:<input type="text" name="qualification" form="user_form"/>
   <input type="submit" /></form>
```

# form attribute (Contd.).




It works like this in an unsupported browser




Output in a browser with form attribute support

# formoverrides attribute

- It allows you to override some of the attributes set for the form element
- The various form override attributes are:
  - formaction – It overrides the form action attribute
  - formmethod – It overrides the form method attribute
  - formnovalidate – It overrides the form novalidate attribute
  - formtarget – It overrides the form target attribute
- The form override attributes works with the following <input> types submit and image

# formaction attribute

- It overrides the form action attribute

```
<form action="/WebApp/user.aspx" method="get">
Email: <input type="email" name="userid">
<input type="submit" value="user">
<input type="submit" formaction="/WebApp/admin.aspx"
  value="admin">
</form>
```

- If the user clicks the user button it submits the form to user.aspx
- If the user clicks the admin button it submits the forms to admin.aspx

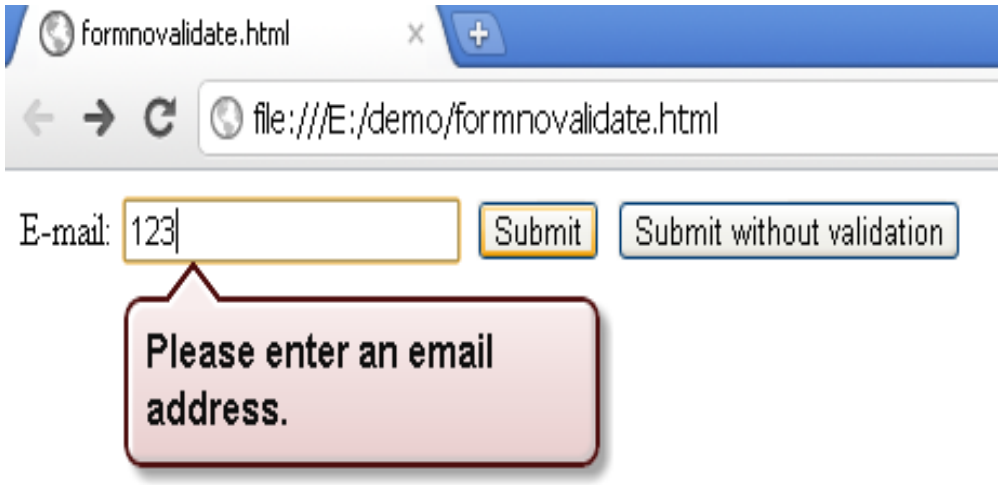
# formmethod attribute

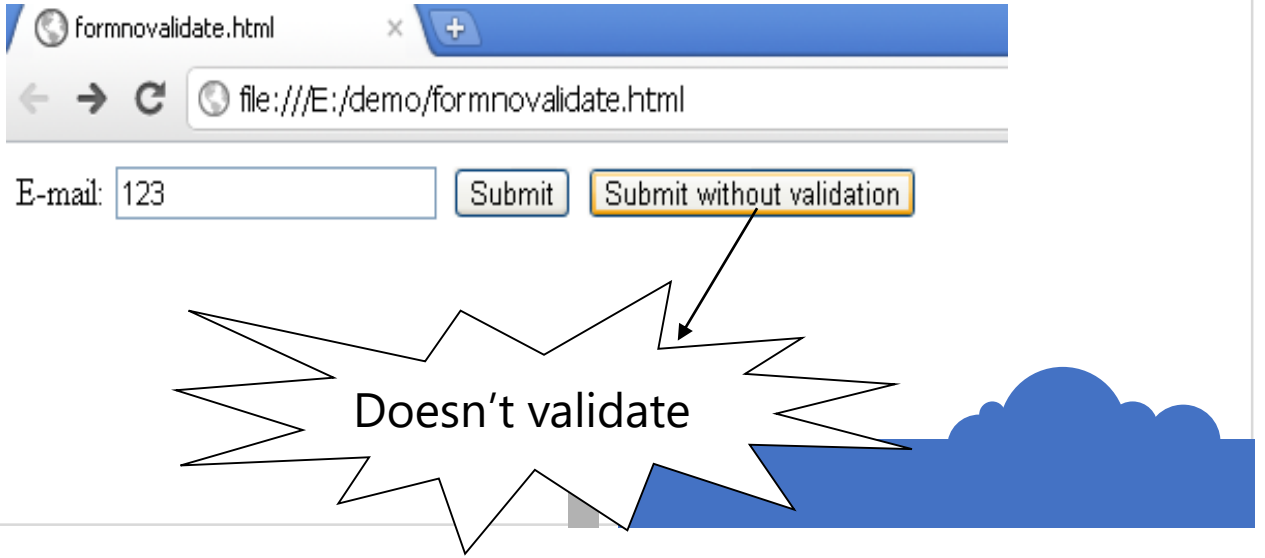
- Helps to override the form method attribute (get / post)

```
<form action="one.aspx" method="get">
FName: <input type="text" name="firstname" /><br />
LName: <input type="text" name="lastname" /><br />
<input type="submit" value="Submit" />
<input type="submit" formmethod="post" formaction="two.aspx"
  value="Submit using POST" />
</form>
```

# formnovalidate attribute

```
<form action="demo.aspx" method="get">
E-mail: <input type="email" name="userid">
<input type="submit">
<input type="submit" formnovalidate="true" value="Submit without
   validation">
</form>
```



# formtarget attribute

- Used to override the target attribute of the form element

<input formtarget="_blank|_self|_parent|_top|*framename*" />

| | |
|---|---|
| _blank | Submits the form to a document in a new window or tab |
| _self | Submits the form to a document in the same frame (this is default) |
| _parent | Submits the form to a document in the parent frame |
| _top | Submits the form to a document in the full body of the window |
| framename | Submits the form to a document in a named iframe |

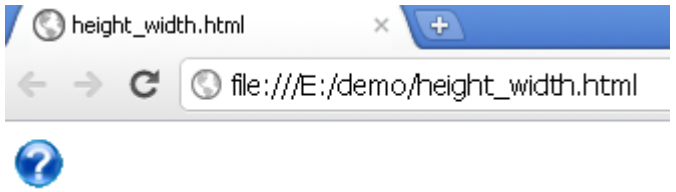
# formtarget attribute (Contd.).

```
<form action="demo1.aspx" method="get">
 First name: <input type="text" name="firstname" /><br />
 Last name: <input type="text" name="lastname" /><br />
<input type="submit" value="Submit" />
<input type="submit" formtarget="_blank" value="Submit to a new
  window/tab" />
</form>
```

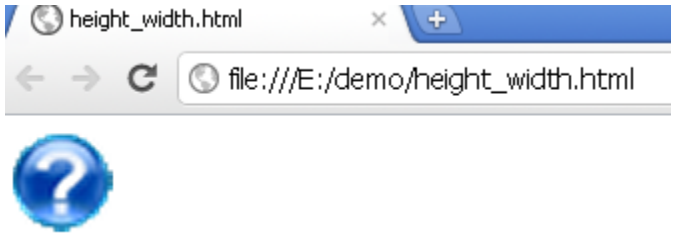
# height and width attribute

- height - specifies the height of the image
- width - specifies the width of the image

```
<input type="image" src="q-mark.gif" />
```




```
<input type="image" src="q-mark.gif" height="50" width="50" />
```



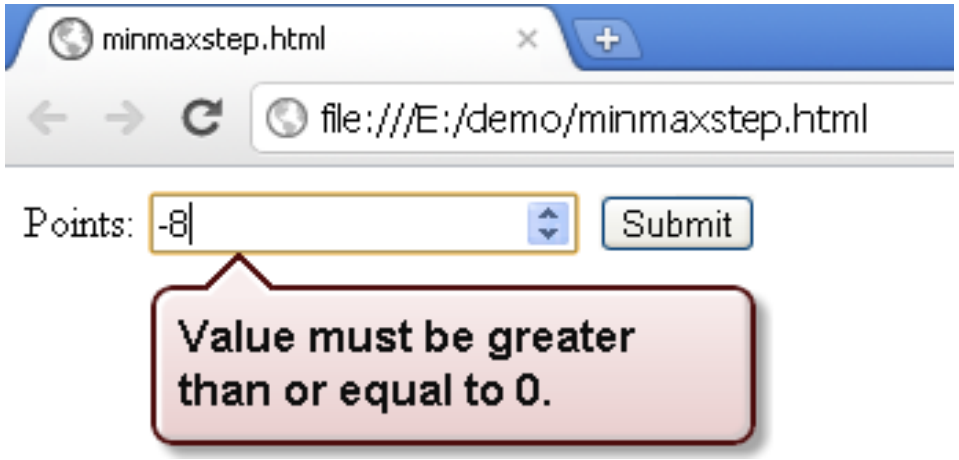
# New Attributes

| Attribute | Description |
|---|---|
| **Min, Max** | Accepted min and max values |
| **Multiple** | Related to file input type, allows selection of multiple files |
| **Pattern** | Specifies a pattern used to validate an input field |
| **Placeholder** | A short hint intended to aid the user with data entry |
| **Required** | Boolean attribute to indicate that the element is required |
| **Step** | Limits allowed values, thus indicating the granularity required |

# min, max and step attribute

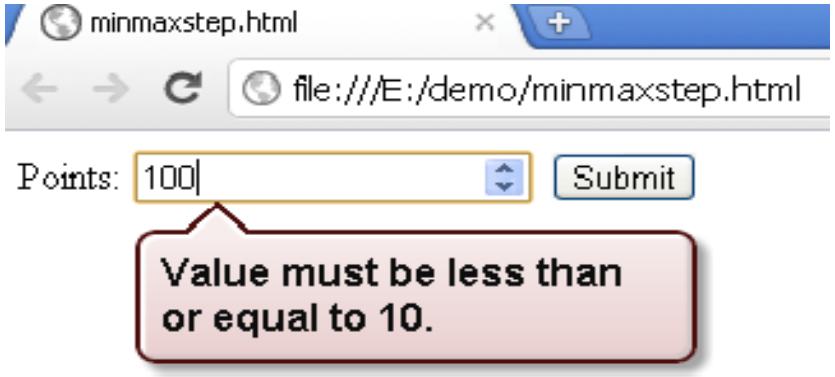
- max – used to specify the maximum value allowed for the input field
- min - used to specify the minimum value allowed for the input field
- step - specifies the legal number intervals for the input field (if step="4", legal numbers could be -4,0,4,8, etc)
- The min, max, and step attributes works with the following <input> types
  - date pickers, number, and range

Points: <input type="number" name="pts" min="0" max="10" step="3"/>

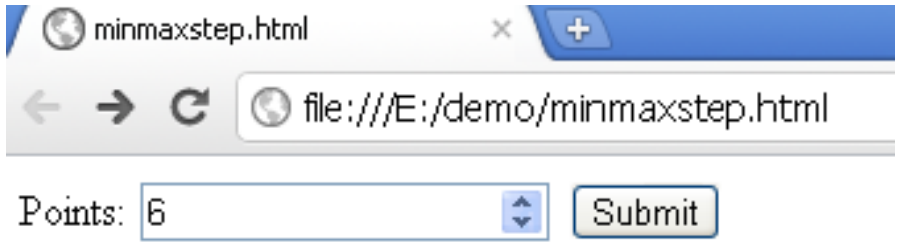


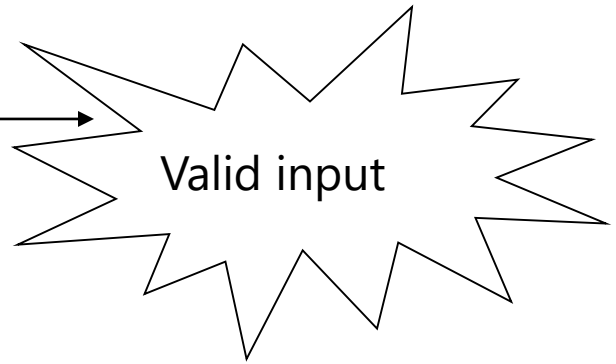
# min,max and step attribute (Contd.).

Points: <input type="number" name="pts" min="0" max="10" step="3"/>





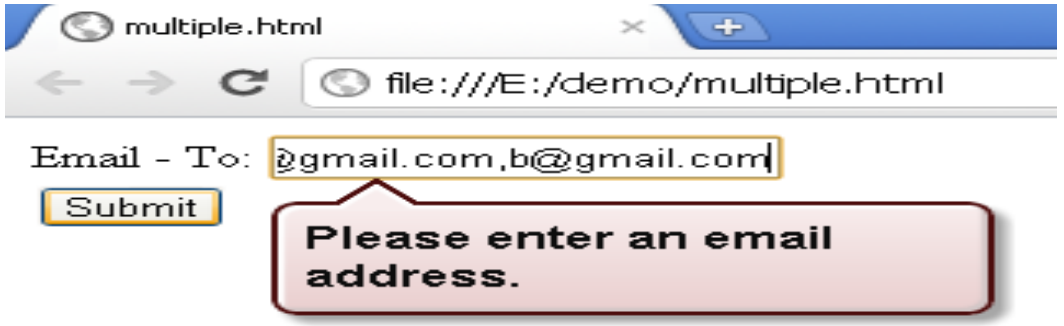


# multiple attribute

- The multiple attribute is used to indicate if the user can be allowed to specify more than one value

Email - To: <input type="email" name="to" />

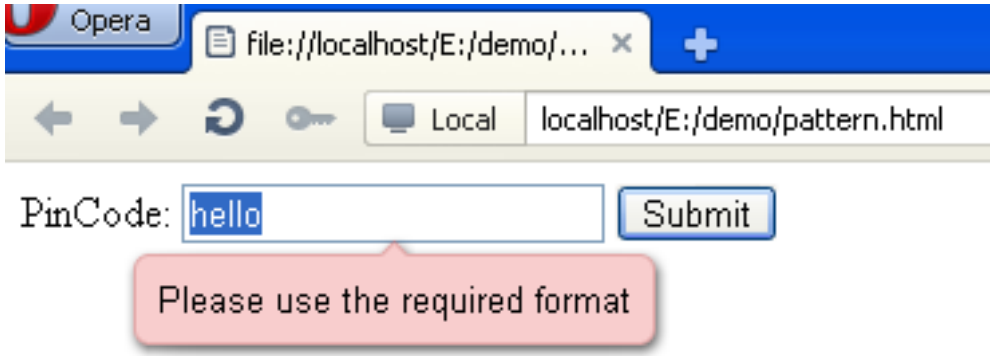



Email - CC: <input type="email" name="cc" multiple />

# pattern attribute

- The pattern attribute is used to specify a pattern to validate an input field
- The pattern is a regular expression
- The pattern attribute works with the following <input> types
  - text, search, url, telephone, email, and password

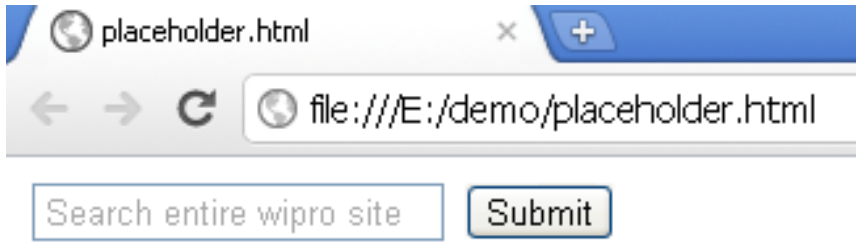
```
<input type="text" pattern="[0-9]{6}" name="pincode" placeholder="A
  pincode is a 6 digit number"/>
```




# placeholder attribute

- It provides description about the expected value of an input field
- It works with the following <input> types
  - text, search, url, telephone, email, and password
- The hint is displayed in the input field when it is empty, and disappears when the field gets focus

```
<input type="search" name="usr_search" placeholder="Search entire wipro site"
/>
```



# required attribute

- The required attribute specifies that an input field must be filled out before submitting
- It works with the following <input> types
  - text, search, url, telephone, email, password, date pickers, number, checkbox, radio, and file

```
Name: <input type="text" name="user_name" required="required" />
```

# Cross-Browser Interoperability

- One of the main problems with earlier versions of HTML was cross-browser interoperability
- Developers strive for code that will work in all browsers
  - This is impossible because of legacy browsers and the difference between browser engines

# Browser Detection



- A technique to check which browser is running the code
- Can make dozens of assumptions based on this single check
- Should be avoided most of the time
- Should only be used when applying code to a specific legacy browser of a known version

```
<!--[if IE 7]><div>rendered in Internet Explorer 7 only</div><![endif]-->
```

# Feature Detection

- A technique to check the availability of a feature before it is used
- Standard features should be looked for first
  - Browsers sometimes support both the standards and a legacy implementation
  - Looking for standart features first will ensure the use of standards if available
- Avoid making any assumptions about existing features

```
if (window.addEventListener) {
    // HTML that will be rendered if addEventListener is available
}
else {
    // HTML that will be rendered if addEventListener is not available
}
```

# Modernizr



- A small JavaScript library
- Detects availability of HTML5 and CSS3 specifications and whether features are natively implemented in the browser
- Uses feature detection under the hood

```
if (Modernizr.canvas) {
   // HTML that will be rendered if the Canvas element is available
}
```

# Summary

- Describe the new HTML5 input types.
- Apply date pickers on webpages.
- Use the new text box types on webpages.
- Use the new interactive text box types on webpages.

# Hands On

- Create a form to accept Customer Name, Address , Email and telephone number
  - Code the form with autocomplete active.
  - The **Name** field you create should have autofocus, placeholder text, and be required. Don't forget to select the appropriate type for this field as well as all the fields that follow.
  - The **Telephone** field should have placeholder text, a pattern to restrict entry, and be required.
  - The **Email address** field should have placeholder text and allow multiple entries. This field should also be required.

# Graphics and Multimedia Elements

# Objectives

SYNERGETICS GET IT RIGHT | Microsoft Partner Gold Cloud Platform Silver Learning

**At the end of this sub-module you will learn:**

- **Use the canvas element and manipulate it by using JavaScript.**
- **Use the audio and video elements, and control them by using JavaScript.**
- **Add support for multiple audio and video codecs.**

# Canvas Basics

- Working with the Canvas
- Drawing Shapes
- Applying Styles
- Drawing Images
- Drawing Text
- Transformations

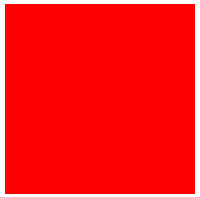
# HTML5 <canvas> tag

- It is used to draw graphics on a web page
- It is a rectangular area and every pixel of it can be controlled
- Used to draw simple diagrams, animations, charts and graphics
- Uses JavaScript to do the drawing
- Has several methods for drawing paths, boxes, circles, characters, and adding images.

# HTML5 Canvas (Contd.)

- **A block element that allows developers to draw 2D or 3D graphics using JavaScript**
  - ✓ **For 3D you use WebGL API**

```
<canvas id="myCanvas" width="200" height="200">
Your browser doesn't support Canvas, sorry.
</canvas>
<script type="text/javascript">
var example = document.getElementById("myCanvas");
var context = example.getContext("2d");
context.fillStyle = "rgb(255,0,0)";
context.fillRect(30, 30, 50, 50);
</script>
```



# Working with the Canvas

- The canvas element is a drawing surface
- The canvas element is lookless

- The “2d” context draws two-dimensional drawings on the canvas

```
<canvas id="canvas" width="300"
height="150"></canvas>
```

- After the canvas is declared, the element can be retrieved by using Javascript

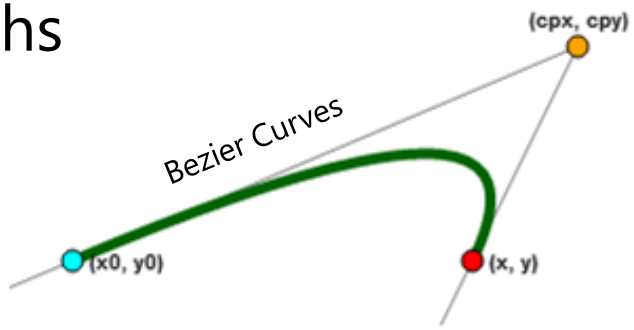
```
var canvas = document.getElementById("canvas");
var context = canvas.getContext("2d");
```

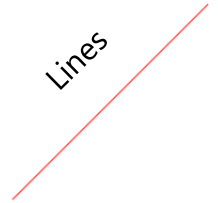
**The canvas is fun!**

The canvas is fun!

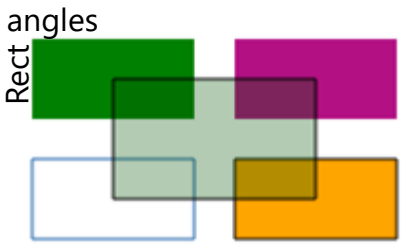
# Drawing Shapes

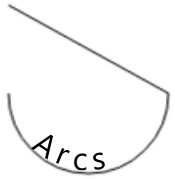
- Simple Shapes: Rectangles
- Complex Shapes: Paths
  - Straight Lines
  - Arcs
  - Bezier Curves
  - Rectangles
  - Clipping Regions




```
context.fillRect(x, y, width, height)
context.strokeRect(x, y, width, height)
context.clearRect(x, y, width, height)
```



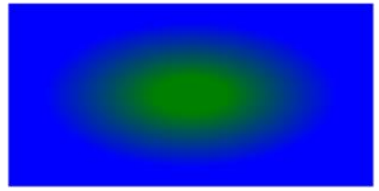



# Applying Styles

- Colors
- Gradients
  - Linear Gradients
  - Radial Gradients
- Line Settings

Line Settings


Linear Gradient


Radial Gradients

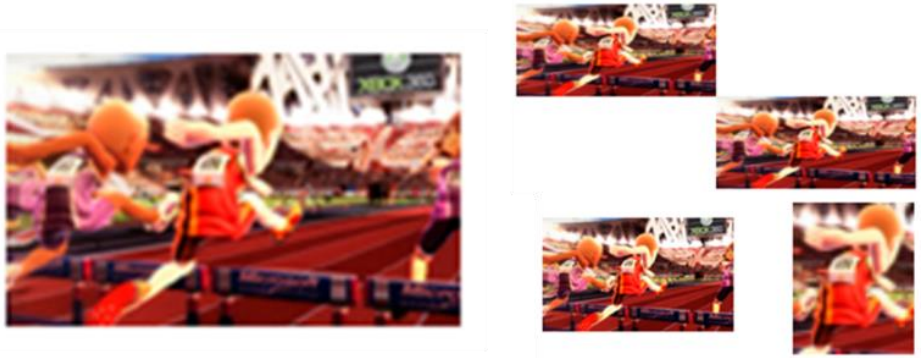
```
context.fillStyle = style
context.strokeStyle = style

context.createLinearGradient(x0, y0, x1, y1)
gradient.addColorStop(0.25, "red");
gradient.addColorStop(0.5, "green");
context.fillStyle = gradient;
```

# Drawing Images



- Images can be drawn from various sources:
  - **img** elements
  - **video** elements
  - **canvas** elements
- The **drawImage** overloads can draw, scale, and slice images.
- Patterns of repeated images can be used as stroke and fill styles.



```
var image = document.createElement("img");
image.onload = function () {
    context.drawImage(image, 20, 20);
};
image.src = "image.png";
```

# Drawing Text



- The **font** property is set by using a CSS-style declaration
- Text on the canvas uses many CSS paradigms
- Text can be aligned horizontally or vertically
- Multiline text is not supported
- Use the start and end alignments for multilingual support

```
context.fillText(text, x, y);
context.strokeText(text, x, y);
```

# Scalable Vector Graphics (SVG)

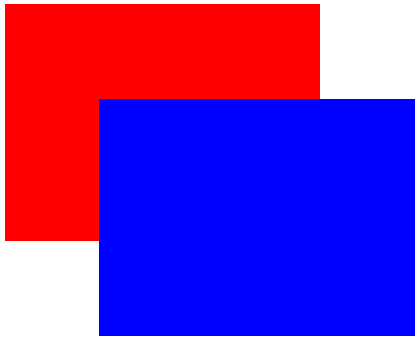


- Create and draw 2D vector graphics
  - ✓ Vector images are composed of shapes instead of pixels
  - ✓ Based on the SVG 1.1 2nd Edition Full specification

- Support for:
  - ✓ Full DOM access to SVG elements
  - ✓ Document structure, scripting, styling, paths, shapes, colors, transforms, gradients, patterns, masking, clipping, markers, linking and views

# 2D Vector Graphics

```
<svg width="400" height="200“>
        <rect fill="red" x="20" y="20" width="100" height="75"
        />
        <rect fill="blue" x="50" y="50" width="100" height="75"
        />
```



# Differences Between Canvas and SVG

| | Canvas | SVG |
|---|---|---|
| **Abstraction** | Pixel based (dynamic bitmap) | Shape based |
| **Elements** | Single HTML element | Multiple graphical elements which become part of the Document Object Model (DOM) |
| **Driver** | Modified through Script only | Modified through Script and CSS |
| **Event Model** | User Interaction is granular (x,y) | User Interaction is abstracted (rect, path) |
| **Performance** | Performance is better with smaller surface and/or larger number of objects | Performance is better with smaller number of objects and/or larger surface. |

# Playing Multimedia

- The **video** and **audio** elements are interchangeable
- The media file is specified with the **src** attribute
- The **autoplay** and **loop** attributes control automatic playback
- The **preload** attribute recommends a buffering strategy
- The **volume** and **muted** attributes control the volume
- The **controls** attribute toggles the built-in media controls
- Add fallback content to support legacy browsers

```
<video src="video.m4v"></video>
<audio src="audio.m4a"></audio>
```

# HTML5 <video> Element

- **Enables to play video natively in the browser**
- ✓ **Video can be composited with anything else on the page**
  - **HTML content, images, SVG graphics**
  - **Include standard codecs like: h.264, ogg and webm**
  - **Hardware accelerated, GPU-based decoding in most of the browsers**

```
<video src="video.mp4" id="videoTag" width="640px" height="360px">
<!-- Only shown when browser support video -->
<!-- You Could Embed Flash or Silverlight Video Here -->
</video>
```

# Video-Specific Attributes

- The **video** element can be laid out easily by using CSS
- The **width** and **height** attributes specify the dimensions of the viewing area
- The video always maintains its innate aspect ratio
- The **poster** frame image is shown before the video is played

```
<video src="video.m4v" poster="first_frame.png" width=”100px” height=”100px”></video>
```

# HTML5 <audio> Element

- **Enables to play audio natively in the browser**
  - ✓ **Industry-standard MP3 and AAC audio**
  - ✓ **Fully scriptable via the DOM**

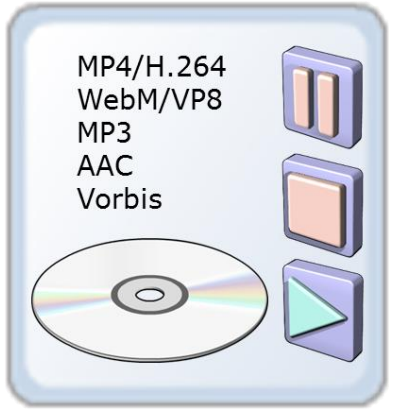
```
<audio src="audio.mp3" id="audioTag" autoplay controls>
<!-- Only shown when browser support audio -->
<!-- You could embed Flash or Silverlight audio here -->
</audio>
```

# Alternative Sources

- The **source** element specifies alternate media resources
- Multiple **source** elements can be specified
- The browser uses only the first **source** element it supports
- Specifying the MIME type with the **type** attribute avoids downloading unsupported files
- The **media** attribute constrains the environment by using *media queries*

```
<video>
  <source src="video.m4v" " type="video/mp4" />
</video>
<audio>
  <source src="audio.m4a" " type="video/mp4" />
</audio>
```

# Video and Audio Formats




| Format | IE | Firefox | Opera | Chrome | Safari |
|---|---|---|---|---|---|
| Ogg | No | Yes | Yes | Yes | No |
| MP3 | Yes | No | No | Yes | Yes |
| Wav | No | Yes | Yes | Yes | Yes |

# Controlling Playback



- The **play** method starts playback
- The **pause** method pauses playback
- The **playbackRate** property changes the playback speed
- The **currentTime** property reports the current position and seeks
- The **duration** property reports the length of the video or audio
- The **buffered** property lists the currently downloaded data
- Many properties and events report status and progress

```
var video = document.getElementById("video");
video.play();
window.setTimeout(function () {
    video.pause();
}, 3000);
```

# Integrating Video and the Canvas



- Video uses time, and the canvas does not
- Time loops are used to simulate time
- The **setInterval** and **setTimeout** methods drive the time loop
- The Video and Canvas APIs work well together

```
function frame() {
    var context = document.getElementyId("canvas").getContext("2d");
    var video = document.getElementById("video");
    context.drawImage(video, 0, 0);
    window.setTimeout(frame, 100);
}

frame(); // calling the function directly triggers the first iteration
```

# Drag and Drop Feature

- Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.
- In HTML5, drag and drop is part of the standard, and any element can be draggable.

# Summary

- Introduced some of the most exciting features in HTML5. The multimedia elements—**video** and **audio**
- Many accessibility features.
- Understanding  APIs that programmers can use to control
- multimedia.

# Hands On

- **To the page created in previous hands on , insert a suitable image**

SYNERGETICS GET IT RIGHT | Microsoft Partner
Gold Cloud Platform
Silver Learning

# Q & A

Contact: smitakumar@synergetics-india.com

SYNERGETICS GET IT RIGHT | Microsoft Partner
Gold Cloud Platform
Silver Learning

# Thank You

**Have a wonderful day ☺**