

# Internet Programming I

## Chapter 2 Web Development Using HTML



Chere L. (M. Tech) and Biruk G. (MSc.)  
Lecturer, Dept. of Software Eng.

Nov 2021, AASTU

# Objectives



After success completion of the chapter you will be able to:

- Identify the core web development technologies
- Understand the HTML document structure and contents models.
- Identify and use HTML elements and attributes
- Build website using HTML

# Lesson 6

## HTML Elements

### (Semantic Elements)

- Page Layout Content sectioning
  - Traditional HTML Layout
  - HTML5 Layout Elements
- Interactive Elements
- iFrame and Portal Elements
- Scalar Graphics Vector (SVG)

# HTML Semantic Elements

# 1. Traditional HTML Layout

- **Content sectioning** elements allow you to organize the document content into logical pieces.
- For a long time, the `<div>` *element* is used to group together related elements on the web page (such as *a header, an article, footer or sidebar*).
- Web developers are used *class or id attributes* to indicate the *role of the <div> element* in the structure of the page.

<body>

```
<div id="page">
```

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

```
<div id="nav">
```

```
<div id="content">
```

```
<div class="article">
```

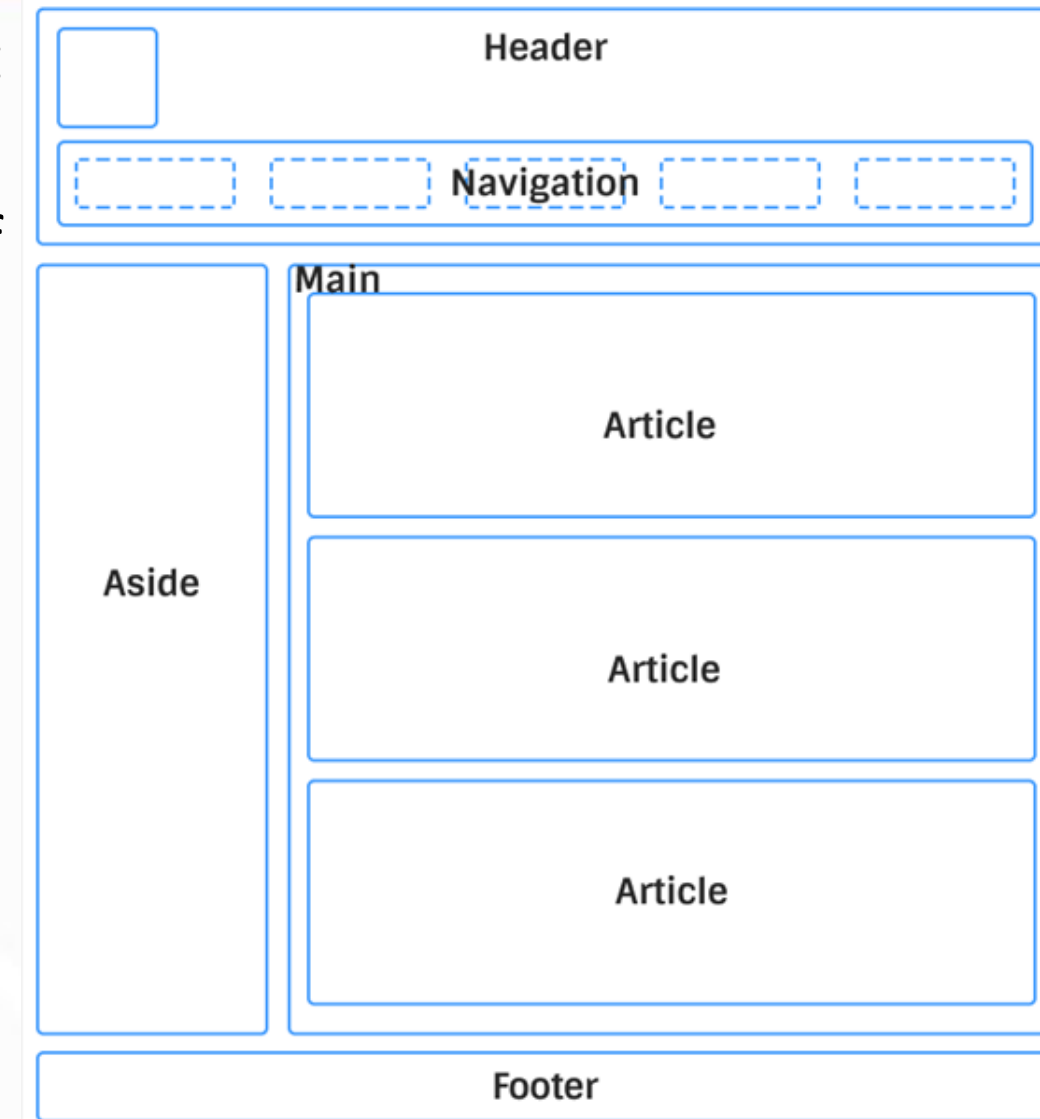
```
<div class="article">
```

```
<div id="sidebar">
```

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

## 2. HTML5 Layout Elements

- HTML5 introduces a new set of elements that allow you to divide up the parts of a page.
- The names of these elements indicate the kind of content found in them.
- HTML5 content sectioning - **semantic elements**
  - header
  - nav
  - main
  - section
  - article
  - aside
  - footer
- **Why semantics?**
  - Clearly describes its meaning to both the browser and the developer
  - Help to structure the code and add readability

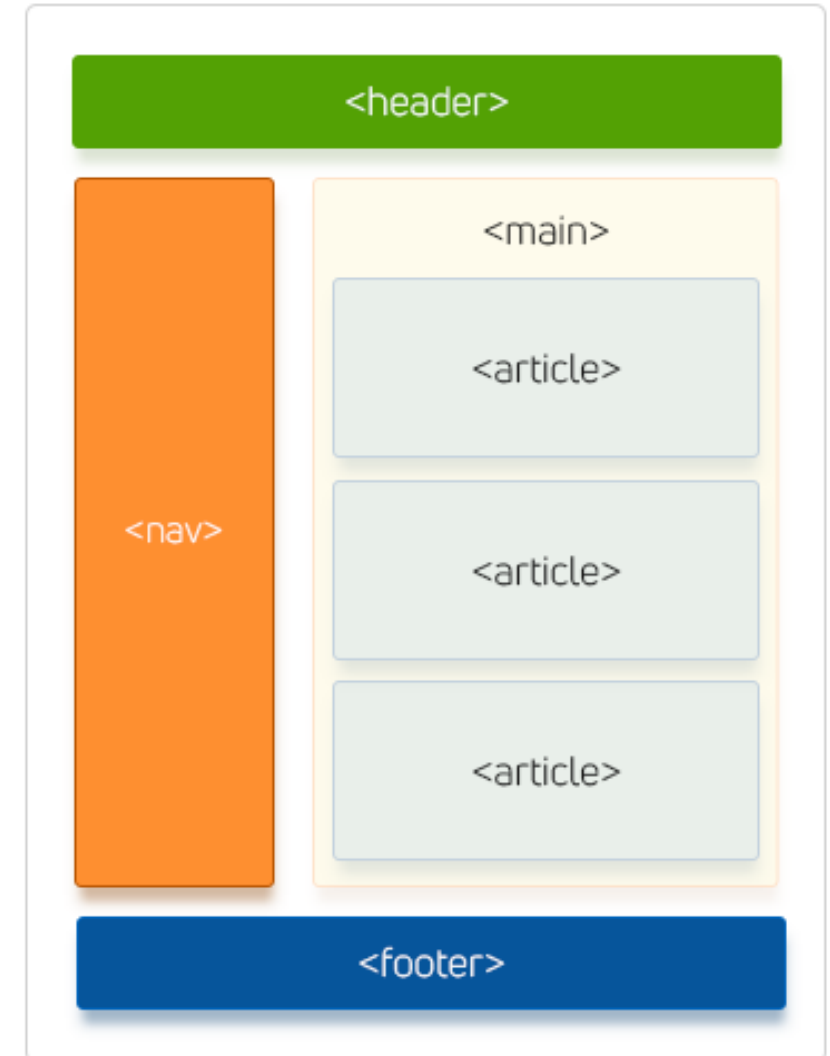


## Traditional Layout Vs. HTML5 Page Layout Elements

### HTML4: Lost of Classes/id



### HTML5: Semantic Tags/Sections





## Exercise

- Identify the section of the page
- Indicate which HTML5 semantic elements are used to define the section

Logo

Primary navPrimary navPrimary navPrimary nav

Search

Submit

Secondary nav

Secondary nav

Secondary nav

Secondary nav

Secondary nav

## Main article heading

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quae sunt igitur communia vobis cum antiquis, iis sic utamur quasi concessis; Nihil acciderat ei, quod nollet, nisi quod anulum, quo delectabatur, in mari abiecerat. Unum est sine dolore esse, alterum cum voluptate. Laboro autem non sine causa; Theophrasti igitur, inquit, tibi liber ille placet de beata vita? Nihil opus est exemplis hoc facere longius. Duo Reges constructio interrete. Graecum enim hunc versum nostis omnes Suavis laborum est praeteritorum memoria. Haec et tu ita posuisti, et verba vestra sunt.

### Article secondary heading

Nos commodius agimus. A mene tu? Tantum dico, magis fuisse vestrum agere Epicuri diem natalem, quam illius testamento cavere ut ageretur. Tenesne igitur, inquam, Hieronymus Rhodius quid dicat esse summum bonum, quo putet omnia referri oportere? Nihilò beatiorem esse Metellum quam Regulum. Sed quanta sit alias, nunc tantum possitne esse tanta. Philosophi autem in suis lectulis plerumque moriuntur. Esse enim, nisi eris, non potes.

Sunt enim quasi prima elementa naturae, quibus ubertas orationis adhiberi vix potest, nec equidem eam cogito consecrari. Id Sextilius factum negabat. Quorum sine causa fieri nihil putandum est. Quae autem natura suae primae institutionis oblita est?

Share

Facebook

Twitter

Email

Recommended

Related article

Article description

Related article

Article description

Footer linkFooter linkFooter linkFooter linkFooter link

## (a) The <header> element

- Specifies a *header* for a document or a section.
- For introductory content <header> element should be used as container.
- Generally it contains one or more heading elements, logo or icons or author's information.
- Several <header> elements can be used in one document,
- However, it cannot be placed within a <footer>, <address> or another <header> element.
- It is *not sectioning content* and therefore does not introduce a new section in the outline.
- Intended to usually contain the surrounding section's heading (h1–h6 element).

## (b) The <footer> element

- Specifies a *footer* for a document or a section.
- It is generally used in the last of the section (bottom of the page).
- Typically contains information about the
  - ✓ *author , contact, and copyright information*
  - ✓ *Sitemap, back to top links,*
  - ✓ *Links to related documents etc.*
- Not sectioning content and therefore doesn't introduce a new section.
- To put information like address, e-mail etc. about the author on your web page, all the relevant elements should be included into the footer element.
- For example, enclose information about the author in an <address> element that can be included into the <footer> element.

## (c) The <nav> element

- Represents a section of a page whose purpose is to **provide navigation links**, either within the current document or to other documents.
- Common examples of navigation sections are menus, tables of contents, and indexes.
- It's not necessary for all links to be contained in a <nav> element.
- It is intended only for major block of navigation links; typically the <footer> element often has a list of links.
- A document may have several <nav> elements, for example, one for site navigation and one for intra-page navigation.
- User agents, such as screen readers, can use this element to determine whether to omit the initial rendering of navigation-only content.

## (d) The <main> element

- Represents the **dominant content** of the <body> of a document.
- It is written within <body> tag.
- It is used to accurately describe the primary content of a page.
- Its content should directly related to or expands upon the central topic of a page, or the central functionality of an application.
- Doesn't contribute to the document's outline.
- It should be unique to the document and cannot be nested inside other semantic elements like <article>, <aside>, <header>, <footer> elements.
- Content that is repeated across other document sections such as sidebars, navigation links, site logos, and search forms shouldn't be included
- If the **search form** is the main function of the page, it can be included.

## (e) The <article> element

- Defines an **independent self-contained** content in a *document, page, application or a site*.
- The content makes sense on its own and intended to be independently distributable or reusable (e.g., in syndication).
- It generally used on *Forum post, a magazine or newspaper article, Blog post, News story, a product card, a user-submitted comment, an interactive widget or gadget, or any other independent item of content*
- Each <article> should be **identified**, typically by including a heading (<h1>-<h6> element) as a child of the <article> element.
- When an <article> element is nested, the inner element represents an article related to the outer element.

## (f) The <section> element

- Represents a *generic standalone section* of a document, which doesn't have a more specific semantic element to represent it.
- Sections should always have a heading, with very few exceptions.
- When you put your content on a web page, it may contains many chapters, headers, footers, or other sections on a web page that is why HTML <section> tag is used.
- Example: A home page could normally be divided into sections for introduction, content, and contact information
- To reiterate, each <section> should be identified, typically by including a heading (<h1>-<h6> element) as a child of the <section> element, wherever possible.



## (g) The <aside> element

- Represents a portion of a document whose content is only **indirectly related** to the document's main content.
- According to W3C definition, the <aside> element represents content that forms the main textual flow of a document.
- It frequently presented as *sidebars or call-out boxes*.
- Do not use the <aside> element to *tag parenthesized text*, as this kind of text is considered part of the main flow.



## Example:

```
<!DOCTYPE html>
<html lang="en">
<body>
  <table width = "90%" border = "1"
    cellpadding = "10" align = "center">
    <tr><td colspan = "3" align="center">
      <header>
        <h1>Welcome On Our Website!</h1>
        <p align = "center" >
          Here is our logo and slogan.
        </p>
      </header></td>
    </tr>

    <tr><td width = "25%">
      <nav><header>
        <h3>Choose Your Interest</h3> </header>
        <ul>
          <li>Menu 1</li>
          <li>Menu 2</li>
          <li>Menu 3</li>
        </ul>
      </nav>
    </td>
```

```
<td>
  <article> <header>
    <h1>Title of Article</h1>
    <h2>Subtitle of Article</h2>
  </header>

  <section>
    <h4>First Logical Part (e.g. "Theory")</h4>
    <p>Paragraph 1 in first section</p>

    <h5>Some Other Subheading in First Section</h5>
    <p>Paragraph 2 in first section</p>
  </section>

  <footer>
    <h5>Author Bio</h5>
    <p>Paragraph in Article's Footer</p>
  </footer>
</article>
</td>
```

```
<td width = "25%">
  <aside>
    <h3>Get To Know Us Better</h3>
    <section>
      <h4>Popular Posts</h4>
      <ul>...</ul>
    </section>
    <section>
      <h4>Partners</h4>
      <ul>...</ul>
    </section>
    <section>
      <h4>Testimonials</h4>
      <ul>...</ul>
    </section> </aside>
</td> </tr>
<tr> <td colspan = "3" align="center">
  <footer>
    <p> Copyright Info and Social Media Links </p>
  </footer>
</td></tr>
</table>
</body>
</html>
```

## Output

<h1>Welcome On Our Website!</h1> <p>Here is our logo and slogan.</p>		
<p><b>Choose Your Interest</b></p> <ul style="list-style-type: none"> <li>Menu 1</li> <li>Menu 2</li> <li>Menu 3</li> </ul>	<p><b>Title of Article</b></p> <p><b>Subtitle of Article</b></p> <p>First Logical Part (e.g. "Theory")</p> <p>Paragraph 1 in first section</p> <p>Some Other Subheading in First Section</p> <p>Paragraph 2 in first section</p> <p>Author Bio</p> <p>Paragraph in Article's Footer</p>	<p><b>Get To Know Us Better</b></p> <p>Popular Posts</p> <p>...</p> <p><b>Partners</b></p> <p>...</p> <p><b>Testimonials</b></p> <p>...</p>
<p>Copyright Info and Social Media Links</p>		

# Interactive Elements

# 3. Interactive Elements

## (a) The <details> element

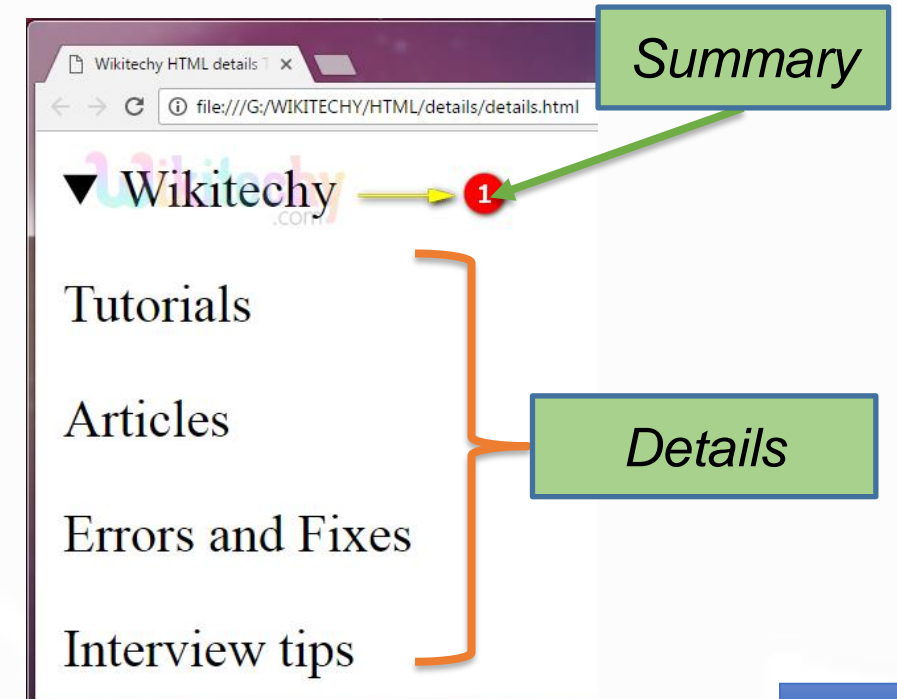
- Creates a disclosure widget in which information is visible only when the widget is *toggled into an "open" state*.
- Often used to create an interactive widget that the user can open and close.
- By default, the widget is closed.
- When open, it expands, and displays the content within.
- Any sort of content can be put inside the <details> tag.
- A disclosure widget is typically presented onscreen using a **small triangle** which *rotates (or twists)* to indicate open/closed status, with a label next to the triangle.

## (b) The <summary> element

- Defines a **visible heading** for the <details> element.
- The heading can be clicked to view/hide the details.
- It should be the first child element of the <details> element.

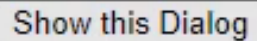


```
<!DOCTYPE html>
<html>
  <head>
    <title>Wikitechy HTML details Tag</title>
  </head>
  <body>
    <details>
      <summary>Wikitechy</summary>
      <p>Tutorials</p>
      <p>Articles</p>
      <p>Errors and Fixes</p>
      <p>Interview tips</p>
    </details>
  </body>
</html>
```



## (c) The <dialog> element

- Represents a dialog box or other interactive component, such as a dismissible alert, inspector, or sub-window.
- Makes it easy to create popup dialogs and modals on a web page



```
<!DOCTYPE>
<html>
  <body>
    <button onclick="showThisDialog()">Show this Dialog</button>

    <dialog id="this-dialog">
      <p> I am a dialog-box </p>
      <button onclick="hideThisDialog()">Close Me! </button>
    </dialog>

    <!-- JavaScript to provide the "Show/Close" functionality -->
    <script type="text/JavaScript">
      var dialog = document.querySelector('dialog');
      dialogPolyfill.registerDialog(dialog);

      function showThisDialog() {
        document.getElementById('this-dialog').show();
      }

      function hideThisDialog() {
        document.getElementById('this-dialog').close();
      }
    </script>
  </body>
</html>
```



## (d) The <menu> element

- Specifies a *list or menu of commands* that a user can perform or activate.
- It is used for creating *context menu* as well as *lists menu*.
- It is a semantic alternative to <ul>.
- It represents an unordered list of items that represented by <li> elements.
- It can contain one or more *<li> elements* within it.
- It is different from <ul> element that it intended for interactive items, to act on whilst <ul> primarily contains items for display.

```
<!DOCTYPE html>
<html>
<body>
<h4>Example of Menu Tag</h4>
<menu>
  <li>Home</li>
  <li>Registration</li>
  <li>Contact-us</li>
  <li>About-us</li>
</menu>
</body>
</html>
```

### Example of Menu Tag

- Home
- Registration
- Contact-us
- About-us

# Embedded Contents



# 4. Embedded Elements

## (a) The <iframe> element

- Represents a *nested browsing context*, embedding another HTML page into the current one (a webpage within a webpage).
- Defines an inline frame, hence it is also called as an Inline frame.
- It embeds another document within the current HTML document in the rectangular region.
- The webpage content and iframe contents can interact with each other using JavaScript.
- Each embedded browsing context has its *own session history* and document.
- The browsing context that embeds the others is called the “parent browsing context”.

## Syntax:

**<iframe src="URL"></iframe>**

- "src" attribute specifies the web address (URL) of the inline frame page.

### ➤ Example 1:

```
<!DOCTYPE html>
<html>
<body>
  <h2>Iframe - Target for a Link</h2>

  <p><a href="https://html.spec.whatwg.org/"
    target="iframe_a">HTML Living Standard</a></p>
  <p>The name of iframe and link target must have
    same value else link will not open as frame.</p>

    <iframe height="300px" width="100%"
      src="" name="iframe_a"></iframe>

</body>
</html>
```

### Iframe - Target for a Link

[HTML Living Standard](https://html.spec.whatwg.org/)

The name of iframe and link target must have same value else link will not open as frame.



Attribute	Value	Description
allow		Specifies a feature policy for the <iframe>
src	<i>URL</i>	Specifies the address of the document to embed in the <iframe>
srcdoc	<i>HTML_code</i>	Specifies the HTML content of the page to show in the <iframe>
height, width	<i>pixels</i>	<ul style="list-style-type: none"><li>➤ Specifies the height and width of an &lt;iframe&gt; respectively.</li><li>➤ Default height and width are 150 pixels and 300 pixels respectively.</li></ul>
name	<i>text</i>	Specifies the name of an <iframe>
allowfullscreen	true false	Set to true if the <iframe> can activate fullscreen mode by calling the requestFullscreen() method
allowpaymentrequest	true false	Set to true if a cross-origin <iframe> should be allowed to invoke the Payment Request API
loading	eager lazy	Specifies whether a browser should load an iframe immediately or to defer loading of iframes until some conditions are met

# Cont'd

## Example 2: Embed YouTube video using iframe

```
<!DOCTYPE html>
<html>
<body style="background-color: #f0f8ff">
  <h3>Play videos using iframe</h3>
  <p>In first video full screen is available and
    in second video full screen is not available</p>

  <iframe width="550" height="315"
    src="https://www.youtube.com/embed/JHq3pL4cdy4" frameborder="0"
    allow="accelerometer; autoplay; encrypted-media; gyroscope;
    picture-in-picture" allowfullscreen style="padding:20px;">
  </iframe>

  <iframe width="550" height="315"
    src="https://www.youtube.com/embed/O5hShU06wxs" frameborder="0"
    allow="accelerometer; autoplay; encrypted-media; gyroscope;
    picture-in-picture" style="padding:20px;">
  </iframe>

</body>
</html>
```

### Note

- It is a good practice to always include a title attribute for the <iframe> which used by screen readers to read out what the content of the <iframe> is.

### Play videos using iframe

In first video full screen is available and in second video full screen is not available



# HTML Graphics

## (5) HTML5 SVG

- The **HTML SVG** is an acronym which stands for *Scalable Vector Graphics*.
- It is a modularized language **W3C recommendation** which is used to describe *two-dimensional vector and mixed vector/raster graphics in XML*.
- SVG images and their behaviors are defined (*image is drawn out*) using a series of statements that follow the XML schema.
- This means SVG images can be created and edited with any text editor, such as Notepad.
- But generally drawing programs like inkspace are preferred to create it.
- Every element and attribute in SVG files can be animated.
- SVG is mostly used for *vector type diagrams* like pie charts, 2-Dimensional graphs in an X,Y coordinate system etc.

## Embedding SVG into HTML Pages

- The **<svg> element** specifies the root of a SVG fragment.
- **Syntax:**

```
<svg xmlns = "http://www.w3.org/2000/svg"
      //Content Here
</svg>
```

### Note:

- The **xmlns attribute** define XML namespace URIs to be used when parsing this element and its child elements
- At least an xmlns is required on the root <svg> element in standalone SVG
- It is only required on the **outermost svg element** of SVG documents.
- It is unnecessary for inner svg elements or inside HTML documents.



## SVG Basic shapes

- To draw SVG, there are various basic shapes. The purpose of these shapes is easily understandable through their name.

Shape	Attributes
Line	<ul style="list-style-type: none"><li>➤ <b>X1, Y1, X2, Y2</b> - defines the <i>start of the line</i> on the x-axis and y-axis</li><li>➤ <b>X1, Y1, X2, Y2</b> - defines the <i>end of the line</i> on the x-axis and y-axis</li></ul>
Rectangle	<ul style="list-style-type: none"><li>➤ <b>X, Y</b> – define the <i>top position</i> of the rectangle</li><li>➤ <b>Width, height</b> – define the rectangle <i>width and height</i> respectively</li></ul>
Circle	<ul style="list-style-type: none"><li>➤ <b>r, cx, cy</b> – define <i>radius, x coordinates and y coordinates</i> of the circle respectively</li></ul>
Ellipse	<ul style="list-style-type: none"><li>➤ <b>rx, ry, cx, cy</b> – define <i>horizontal radius, vertical radius, x coordinates and y coordinates</i> of the center of the ellipse respectively</li></ul>
Polygon	<ul style="list-style-type: none"><li>➤ An arbitrary straight-line closed shape, possibly including crisscrossing edges.</li><li>➤ <b>Points attribute</b> - The list of vertices (corner-points) of the polygon.</li></ul>
Polyline	Polyline is a connected straight line group. It has all the points of line in one attribute



## Common CSS property of SVG basic shape

- **Fill** - *define the fill color of the shape.*
- **Fill-opacity** - *defines the opacity of the fill color (legal range: 0 to 1)*
- **Stroke** - *define the color of text, line or outline of any element.*
- **Stroke-width** - *define the thickness of text, line or outline of any element.*
- **stroke-linecap** - *define different types of ending of a line or outline of any path.*
- **Stroke-opacity** – *defines the opacity of the stroke color (legal range: 0 to 1)*
- **stroke-dasharray** - *used to create dashed lines.*

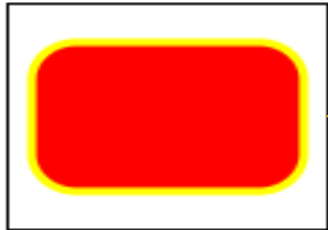
# Cont'd

## Example 1:

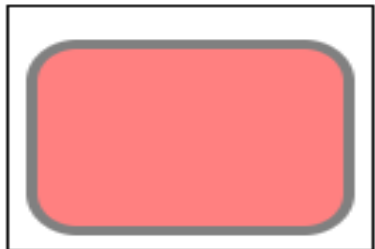
SVG Line



SVG Rectangle

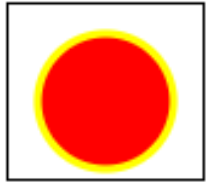


SVG Rounded Rectangle



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Create a Line with HTML5 SVG</title>
  <style> svg { border: 1px solid black;} </style>
</head>
<body>
  <table cellpadding = "10px" cellspacing = "10px">
    <tr><td> <h3> SVG Line </h3>
      <svg width="100" height="120">
        <line x1="20" y1="30" x2="80" y2="100"
          stroke="red" stroke-width="4"/>
      </svg>
    </td><tr>
    <tr><td> <h3> SVG Rectangle </h3>
      <svg height="140" width="160">
        <rect x = "20" y = "10" rx = "20" ry = "20" width="120" height="120"
          style="fill:red; stroke:yellow; stroke-width:5;" />
      </svg>
    </td></tr>
    <tr><td> <h3> SVG Rounded Rectangle </h3>
      <svg width="160" height="130">
        <rect x="10" y="20" rx="20" ry="20" width="140" height="100"
          style="fill:red;stroke:black;stroke-width:5;opacity:0.5" />
      </svg>
    </td> </tr>
```

# Cont'd



SVG Polygon



SVG Logo Using Eclipse



```
<tr><td> <h3> SVG Circle </h3>
  <svg width="100" height="90">
    <circle cx="50" cy="50" r="35" stroke="yellow"
      stroke-width="4" fill="red" />
  </svg></td></tr>
<tr><td> <h3> SVG Polygon </h3>
  <svg height="160" width="160">
    <polygon points="80,10 20,148 150,60 10,60, 120,148"
      style="fill:red; stroke:yellow; stroke-width:5;
        fill-rule:nonzero;" />
  </svg> </td></tr>
<tr> <td> <h3> SVG Logo Using Eclipse </h3>
  <svg height="130" width="200">
    <ellipse cx="100" cy="70" rx="85" ry="55"
      fill="url(#grad1)" />
    <text fill="#ffffff" font-size="45"
      font-family="Verdana" x="50" y="86">SVG</text>
    <defs> <linearGradient id="grad1"
      x1="0%" y1="0%" x2="100%" y2="0%">
      <stop offset="0%" style="stop-color:rgb(255,255,0);
        stop-opacity:1" />
      <stop offset="100%" style="stop-color:rgb(255,0,0);
        stop-opacity:1" />
    </linearGradient> </defs>
    Sorry, your browser does not support inline SVG.
  </svg> </td> </tr>
</table>
</body>
</html>
```

## Example 3 – SVG Eclipse and polyline:

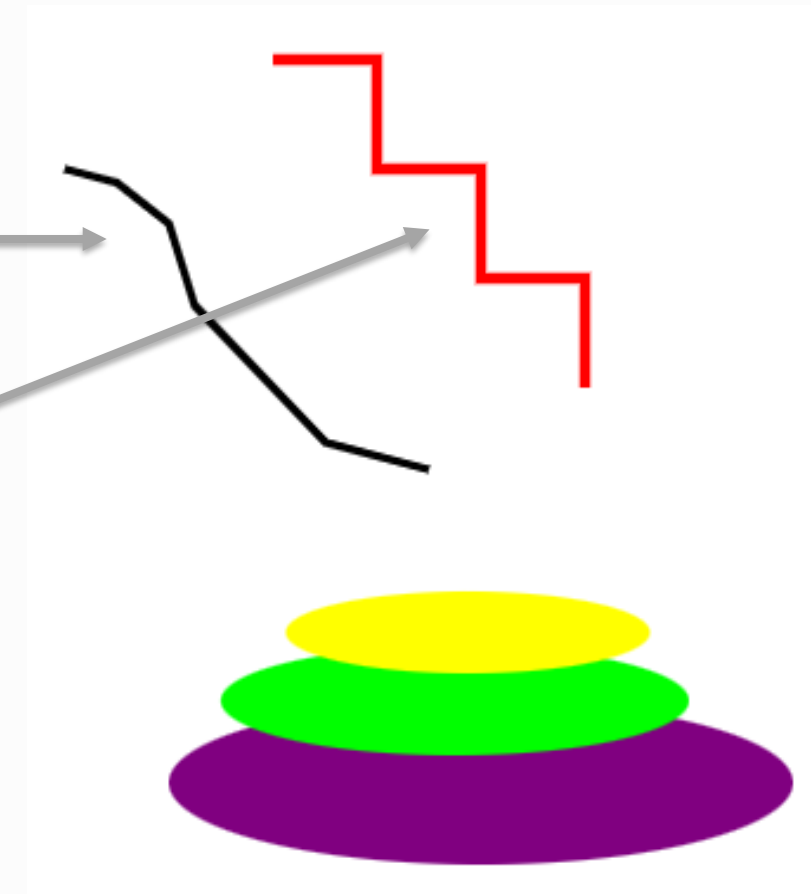
```
<!DOCTYPE html>
<html>
<body>

<svg height="200" width="500">
  <polyline
    points="20,80 40,85 60,100 70,130 120,180 160,190"
    style="fill:none;stroke:black;stroke-width:3" />

  <polyline
    points="100,40 140,40 140,80 180,80 180,120 220,120 220,160"
    style="fill:white;stroke:red;stroke-width:4" />
  Sorry, your browser does not support inline SVG.
</svg>

<svg height="150" width="500">
  <ellipse cx="180" cy="100" rx="120" ry="30" style="fill:purple"/>
  <ellipse cx="170" cy="70" rx="90" ry="20" style="fill:lime"/>
  <ellipse cx="175" cy="45" rx="70" ry="15" style="fill:yellow"/>
</svg>

</body>
</html>
```



## SVG Text

- The `<text>` element - used to draw text
- With `<tspan>` element, the `<text>` element can be arranged in any number of sub-groups.
- The **Hyper link Text** used to create the text which has hyper link.

## Attributes

- *X, Y* - defines the position of the top left corner top position of the text respectively.
- *Width, height* - defines the width and height respectively.
- *Transform* – defines the rotation of the text

# Cont'd

## Example:

### SVG Text

Text:  
WWW.TutorailA

### Text with Rotate

*I love SVG*

Multiline Text:  
WWW.JavaTPoint.COM  
This is Simple Easy learning.

Text as Link:

WWW.JavaTPoint.COM

```
<!DOCTYPE html>
<html>
  <title>SVG Text</title>
  <body>
    <h3>SVG Text</h3>
    <svg width="150" height="50">
      <g>
        <text x="40" y="23" >Text: </text>
        <text x="40" y="40"
              fill="rgb(121,0,121)">WWW.TutorailAndExample.COM</text>
      </g></svg>

    <h3> Text with Rotate </h3>
    <svg height="60" width="200">
      <text x="0" y="15" fill="red" transform="rotate(30 20,40)">I love SVG</text>
    </svg>

    <svg width="570" height="100">
      <g>
        <text x="40" y="23" >Multiline Text: </text>
        <text x="40" y="40" fill="rgb(121,0,121)">WWW.JavaTPoint.COM
        <tspan x="40" y="60" font-weight="bold">
          This is Simple Easy learning.
        </tspan></text>
      </g> </svg>

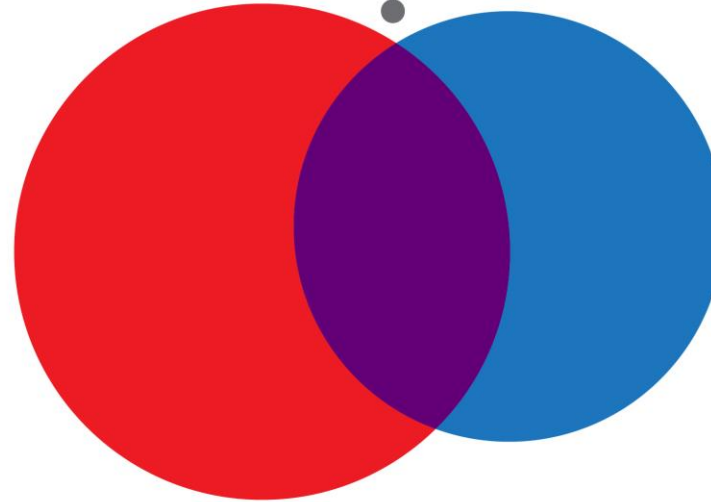
    <svg width="800" height="800">
      <g>
        <text x="20" y="10" >Text as Link: </text>
        <a xlink:href="https://www.javatpoint.com/svg-tutorial" target="_blank">
          <text font-family="Verdana" font-size="20" x="40" y="40"
              fill="rgb(121,0,121)">WWW.JavaTPoint.COM</text>
        </a>
      </g></svg>
  </body>
</html>
```

## Why SVG is preferred over other image formats?

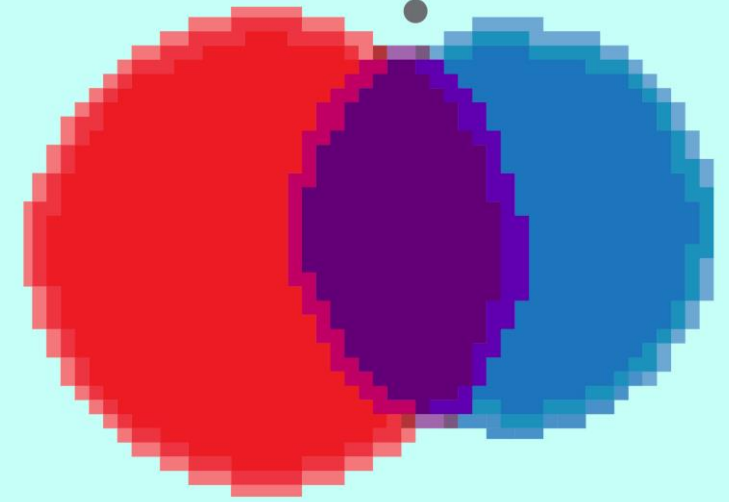
- The advantages of using SVG over other image formats like JPEG, GIF, PNG
  - a) The SVG images can be;
    - saved as the smallest size possible or zoomed to a certain level without degradation of the picture quality.
    - searched , indexed, scripted, and compressed.
    - created and modified using JavaScript in real time.
    - printed with high quality at any resolution because unlike bitmap image formats like JPG or PNG, it does not contain a fixed set of dots. .
    - contain hyperlinks to other documents.
  - b) The content can be animated using the built-in animation elements.

## Closer Look of Vector Vs. Raster images

**Vector**



**Raster**



AI

EPS

CGM

PDF

SVG

CDR

BMP

TIFF

PCX

GIF

PNG

JPEG



## (6) HTML Graphics Canvas Element

- The *HTML5 canvas element* can be used to draw graphics on the webpage via JavaScript.
- The canvas was originally introduced by Apple for the Mac OS dashboard widgets and to power graphics in the Safari web browser.
- Later it was adopted by the Firefox, Google Chrome and Opera.
- Now the canvas is a part of the new HTML5 specification for next generation web technologies.
- Canvas is a low level, procedural model that updates a bitmap and does not have a built-in scene.
- Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

- The **<canvas> element**
  - Provides HTML a bitmapped surface to work with.
  - Used to draw graphics on the web page, **on the fly** using scripting language like JavaScript
  - Allows for *dynamic and scriptable rendering of 2D* shapes and bitmap images.
  - Only a container for graphics and must need to use JavaScript to draw the graphics.
  - The *canvas scripting API or the WebGL API* can be used to actually draw graphics and animations.
- The canvas is a *two-dimensional rectangular* area.
- The coordinates of the top-left corner of the canvas are (0, 0) – the origin, and the coordinates of the bottom-right corner are (canvas width, canvas height).
- By default the <canvas> element has *300px of width and 150px* of height without any border and content.

## ■ Canvas Example:

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

```
  <canvas id="myCanvas" width="200" height="100"
    style="border:1px solid #000000;">
```

Your browser does not support the HTML canvas tag.

```
  </canvas>
</body>
</html>
```



## Note:

- Always specify an
  - ***id attribute*** (to be referred to in a script),
  - A ***width and height attribute*** to define the size of the canvas.
- To add a border, use the style attribute.
- After creating the rectangular canvas area, you must add a JavaScript to do the drawing.

***(will be discussed chapter 4)***



Canvas	SVG
<ul style="list-style-type: none"> <li>■ Draws 2D graphics, on the fly (with a JavaScript)</li> <li>■ No support for event handlers</li> <li>■ Resolution dependent, rendered pixel by pixel and once drawn it forgotten by the browser.</li> <li>■ Poor text rendering capabilities</li> <li>■ The resulting image can be saved as .png or .jpg</li> <li>■ Well suited for graphic-intensive games</li> </ul>	<ul style="list-style-type: none"> <li>➤ A language for describing 2D graphics in XML</li> <li>➤ Resolution independent, the drawn shape is remembered as an object and any change can be rendered by browser</li> <li>➤ Support for event handlers, and JavaScript event handlers for an element.</li> <li>➤ Best suited for applications with large rendering areas (Google Maps)</li> <li>➤ Slow rendering if complex (anything that uses the DOM a lot will be slow)</li> <li>➤ Not suited for game applications</li> </ul>

- HTML5 semantic elements include
  - <header>,
  - <nav>,
  - <main>,
  - <section>,
  - <article>,
  - <aside>,
  - <footer>
- Embedded element - HTML iFrame
- Interactive elements - <details>, <summary>, <dialog>, <menu>
- HTML Graphics – SVG and Canvas

# Reading Assignment

## ➤ *Embedded Elements*

- *<math> element*
- *<portal> element*
- *<embed> element*
- *<object> element*
- *<param> element*

## ➤ *Scripting Elements*

- *<script>*
- *<noscript>*
- *<style>*

## ➤ *More on SVG Elements*

- *<g> element*
- *<defs> element*
- *<symbol> element*
- *Pattern*
- *Clip path*
- *SVG Tools*

# Reading Resources/Materials

## *Chapter 17, :*

- ✓ Jon Duckett; HTML and CSS Design and Build Websites, 2011 John Wiley & Sons, Inc., Indianapolis, Indiana

## *Chapter 3:*

- ✓ Paul Deitel, Harvey Deitel, Abbey Deitel (2014). Internet & World Wide Web How To Program (5th Edition), Pearson Education.

## *Chapter 11:*

- ✓ Musciano, C., Kennedy, B. (2006). HTML and XHTML: The Definitive Guide, 6th edition. O'Reilly Media, Inc.

# Thank You For Your Attention!!

Any Questions

