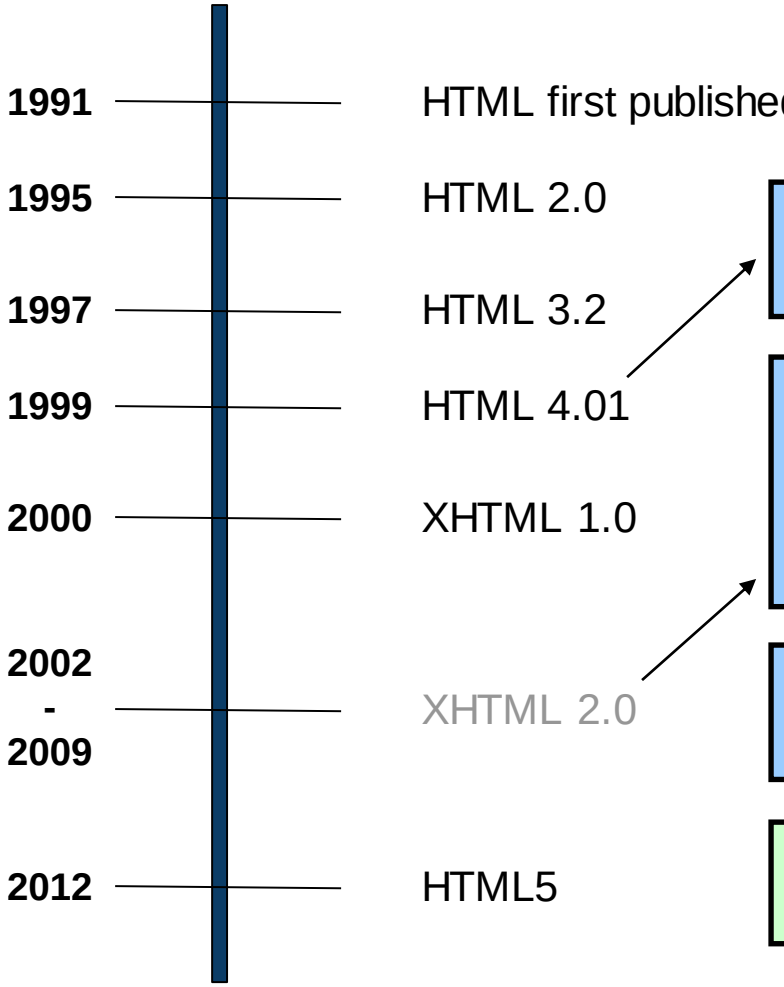
Introduction to HTML5

Puneeth BR

Assistant Professor

NMAMIT NITTE

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 HTML5?

- HTML5 is the newest version of HTML, only recently gaining partial support by the makers of web browsers.
- It incorporates all features from earlier versions of HTML, including the stricter XHTML.
- It adds a diverse set of new tools for the web developer to use.
- It is still a work in progress. No browsers have full HTML5 support. It will be many years – perhaps not until 2018 or later - before being fully defined and supported.

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.

New Elements in HTML5

<article>
<aside>
<audio>
<canvas>
<datalist>
<figure>
<figcaption>
<footer>
<header>
<hgroup>
<mark>
<nav>
<progress>
<section>
<source>
<svg>
<time>
<video>

These are just some of the new elements introduced in HTML5. We will be exploring each of these during this course.

Other New Features in HTML5

- Built-in audio and video support (without plugins)
- Enhanced form controls and attributes
- The Canvas (a way to draw directly on a web page)
- Drag and Drop functionality
- Support for CSS3 (the newer and more powerful version of CSS)
- More advanced features for web developers, such as data storage and offline applications.

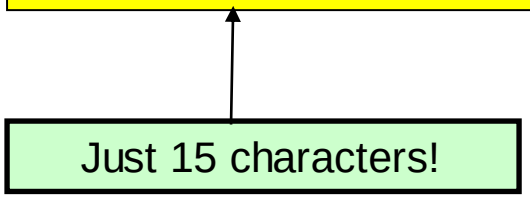
First Look at HTML5

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>
```



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 <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.

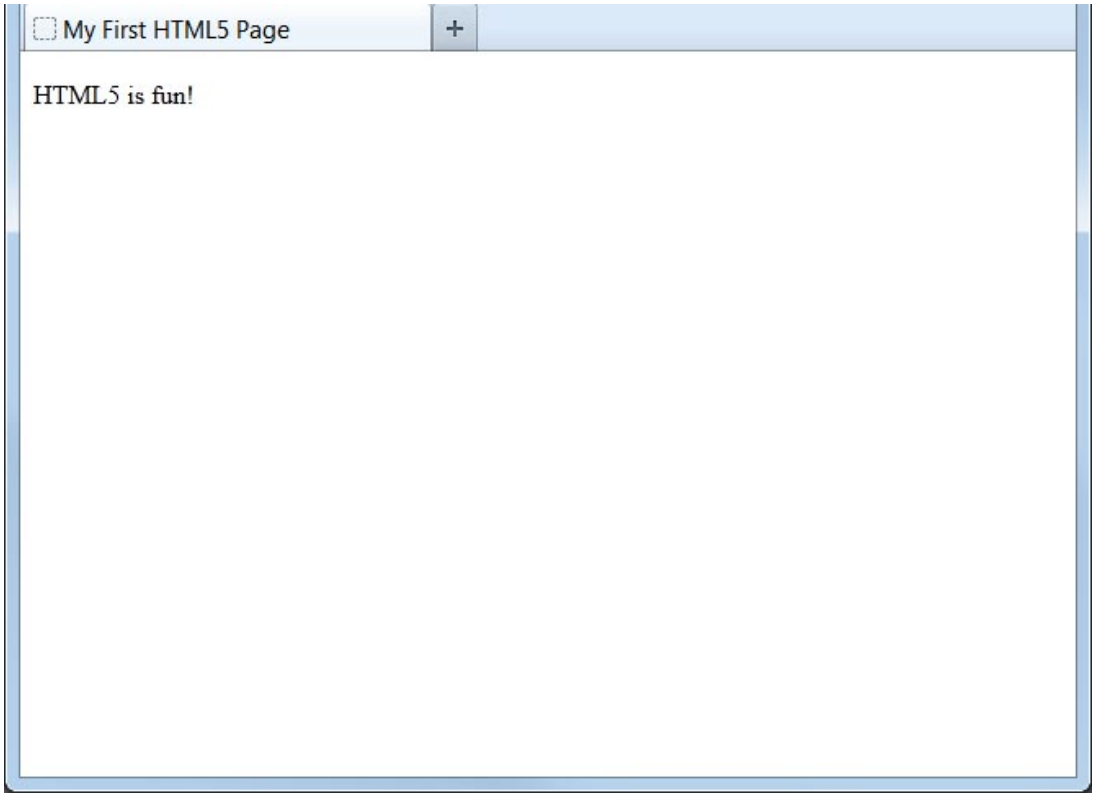
Basic HTML5 Web Page

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.

<iframe>

The <iframe> tag specifies an inline frame.

An inline frame is used to embed another document within the current HTML document.

<iframe src="a.html" width="150" height="150" scrolling="auto"

frameborder="no"> </iframe>

HTML Layouts

Websites often display content in multiple columns (like a magazine or newspaper).

HTML5 offers new semantic elements that define the different parts of a web page:

HTML5 Semantic Elements

<header> - Defines a header for a document or a section

<nav> - Defines a container for navigation links

<section> - Defines a section in a document

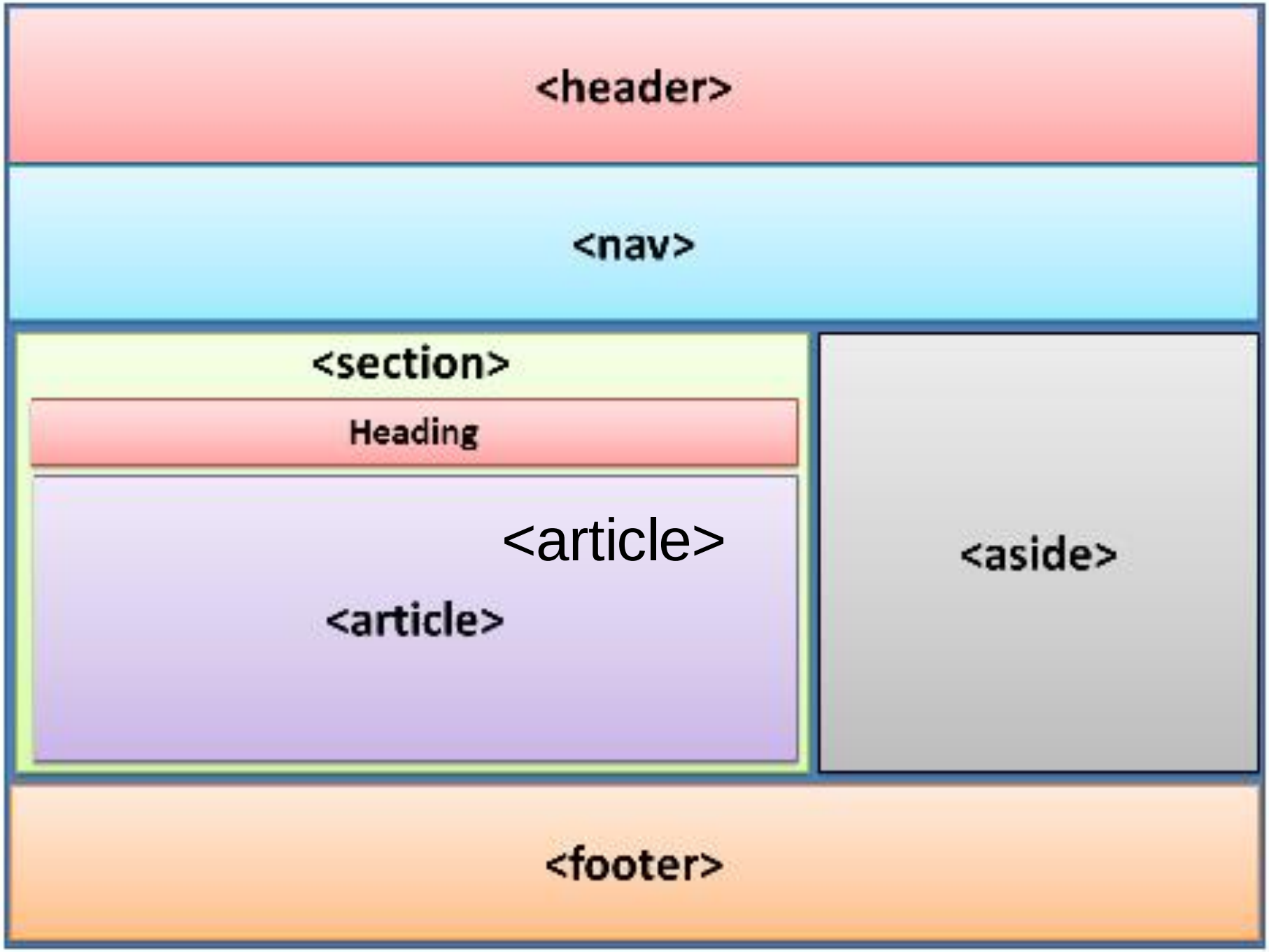
<article> - Defines an independent self-contained article

<aside> - Defines content aside from the content (like a sidebar)

<footer> - Defines a footer for a document or a section

<details> - Defines additional details

<summary> - Defines a heading for the <details> element


<header>
<nav>
<section>
Heading
<article>
<article>
<aside>
<footer>

<header>

```
<header>
<h1>This is page heading</h1>
</header>
```

<nav>

The <nav> section can contain links to the other pages from the website or to other parts of the same web page.

```
<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">About Us</a></li>
    <li><a href="#">Contact Us</a></li>
  </ul>
</nav>
```

section

<section> element represents a generic section of a document or application. The <section> element should not be confused with the element. The <section> element is a thematic grouping of content whereas the doesn't have any such restriction

```
<section>
 <h1>This is a section heading</h1>
 <p>
   Hello world! Hello world! Hello world!
   Hello world! Hello world! Hello world!
   Hello world! Hello world! Hello world!
 </p>
</section>
```

<article>

an independent item section of content such as a blog post, forum post or a comment

```
<article>
  <h1>This is article heading</h1>
  <p>
    Hello world! Hello world! Hello world!
    Hello world! Hello world! Hello world!
    Hello world! Hello world! Hello world!
  </p>
</article>
```

aside

The <aside> tag defines some content aside from the content it is placed in.

```
<aside>
  <figure>
    <img src="images/laptop.png" height="100px" width="100px" />
    <figcaption>Figure caption goes here</figcaption>
  </figure>
  <p>
    Hello world! Hello world! Hello world!
    Hello world! Hello world! Hello world!
  </p>
</aside>
```

<footer>

The <footer> element specifies a footer for a document or section.

A <footer> element should contain information about its containing element.

A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

```
<footer>
 <p>Posted by: Hege Refsnes</p>
 <p>Contact information: <a
  href="mailto:someone@example.com">
 someone@example.com</a>.</p>
</footer>
```

Viewport

HTML. Viewport meta tag for Responsive Web Design. A Browser's viewport is the area of web page in which the content is visible to the user. The viewport does not have the same size, it varies with the variation in screen size of the devices on which the website is visible.

<meta name="viewport" content= "width=device-width, initial-scale=1.0">

width: Width of the virtual viewport of the device.

height: Height of the virtual viewport of the device.

initial-scale: Zoom level when the page is first visited.

minimum-scale: Minimum zoom level to which a user can zoom the page.

maximum-scale: Maximum zoom level to which a user can zoom the page.

user-scalable: Flag which allows the device to zoom in or out.(value= yes/no).

computer code

The HTML <code> tag is used for indicating a piece of code. The code tag surrounds the code being marked up.

Tag	Description
	Renders as emphasized text
	Defines important text
<code>	Defines a piece of computer code
<samp>	Defines sample output from a computer program
<kbd>	Defines keyboard input
<var>	Defines a variable

```
<code>
x = 5;
y = 6;
z = x + y;
</code>
```

new semantics elements

Semantic elements = elements with a meaning.

Examples of non-semantic elements: and - Tells nothing about its content.

Examples of semantic elements: <form>, <table>, and <article> - Clearly defines its content.

Tag Description

<article> : Defines an article

<aside> : Defines content aside from the page content

<details>: Defines additional details that the user can view or hide

<figcaption>: Defines a caption for a <figure> element

<figure> : Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

<footer> : Defines a footer for a document or section

<header> : Specifies a header for a document or section

<main> : Specifies the main content of a document

<mark> : Defines marked/highlighted text

<nav> : Defines navigation links

<section>: Defines a section in a document

<summary>: Defines a visible heading for a <details> element

<time> :Defines a date/time

HTML5 Migration

how to convert an HTML4 page into an HTML5 page, without destroying anything of the original content or structure.

You can migrate from XHTML to HTML5, using the same recipe.

Typical HTML4

Typical HTML5

<header>

<nav>

<section>

<article>

<footer>

Change doctype and Character Encoding

Html5

<meta charset="utf-8">

Html4

Doctype <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<!DOCTYPE html>

```
<!DOCTYPE html>
<html><head><style>
header{
background:cyan; height:100px;
}
footer{
background:red; height:100px;
}
section{
background:#eee; height:200px;
}
</style></head><body><header">
<p>This is Header section</p> </header>
<section">
<p>Switch from Html4 to Html5</p> </section>
<footer">
<p>This is Footer section</p> </footer> </body>
</html>
```

HTML4

```
<!DOCTYPE html>
<html><head><style>
.header{
background:cyan; height:100px;
}
.footer{
background:red; height:100px;
}
.content{
background:#eee; height:200px;
}
</style></head>
<body> <div class="header">
<p>This is Header section</p>
</div>
<div class="content">
<p>Switch from Html4 to Html5</p>
</div>
<div class="footer">
<p>This is Footer section</p> </div> </body> </html>
```

HTML5

```
<!DOCTYPE html>
<html><head><style>
header{
background:cyan; height:100px;
}
footer{
background:red; height:100px;
}
section{
background:#eee; height:200px;
}
</style></head><body><header">
<p>This is Header section</p> </header>
<section">
<p>Switch from Html4 to Html5</p> </section>
<footer">
<p>This is Footer section</p> </footer> </body>
</html>
```

canvas

The HTML <canvas> element is used to draw graphics.

The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.

Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

Find the Canvas Element

First of all, you must find the <canvas> element.

This is done by using the HTML DOM method getElementById():

```
var canvas =
  document.getElementById("myCanvas");
```

Step 2: Create a Drawing Object

Secondly, you need a drawing object for the canvas.

The getContext() is a built-in HTML object, with properties and methods for drawing:

```
var ctx = canvas.getContext("2d");
```

Step 3: Draw on the Canvas

Finally, you can draw on the canvas.

Set the fill style of the drawing object to the color red:

```
ctx.fillStyle = "#FF0000";
```

The fillStyle property can be a CSS color, a gradient, or a pattern. The default fillStyle is black.

The fillRect(x,y,width,height) method draws a rectangle, filled with the fill style, on the canvas:

```
ctx.fillRect(0, 0, 150, 75);
```

//Start at the upper-left corner (0,0) and draw a 150x75 pixels rectangle.

Draw a Line

To draw a straight line on a canvas, use the following methods:

moveTo(x,y) - defines the starting point of the line

lineTo(x,y) - defines the ending point of the line

To actually draw the line, you must use one of the "ink" methods, like stroke().

Example

Define a starting point in position (0,0), and an ending point in position (200,100). Then use the stroke() method to actually draw the line:

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.moveTo(0, 0);
ctx.lineTo(200, 100);
ctx.stroke();
```

Draw Text

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

<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #d3d3d3;">
</canvas>

<script>
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World",10,50);
</script>

</body>
</html>
```

Canvas - Images

To draw an image on a canvas, use the following method:

drawImage(image,x,y)

HTML Media

Multimedia comes in many different formats. It can be almost anything you can hear or see.

Examples: Images, music, sound, videos, records, films, animations, and more.

Web pages often contain multimedia elements of different types and formats.

HTML5 <audio>

The HTML5 <audio> element specifies a standard way to embed audio in a web page.

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">

</audio>
```

Autoplay: This Boolean attribute specifies that the audio will automatically start playing as soon as it can do so without stopping to finish loading the data.

Controls: If specified, the browsers will display controls to allow the user to control audio playback, such as play/pause, volume, etc.

Loop: This Boolean attribute specifies that the audio will automatically start over again, upon reaching the end.

Muted: This Boolean attribute specifies whether the audio will be initially silenced. The default value is false, meaning that the audio will be played.

src URL : Specifies the location of the audio file. Alternatively, you can use the preferred <source> tag as it allows for multiple options.

HTML5 <video>

The HTML5 <video> element specifies a standard way to embed a video in a web page.

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
</video>
```

Controls: attribute adds video controls, like play, pause, and volume.

It is a good idea to always include width and height attributes. If height and width are not set, the page might flicker while the video loads.

<source>: element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.

Google Map

HTML Google Map is used to display maps on your webpage

```
<!DOCTYPE html>
<html>
<body>
<h1>First Google Map Example</h1>
<div id="map">My map will go here...</div>
</body>
</html>
```

Set the Map Size

```
<div id="map"
style="width:400px;height:400px;backgrou
nd:grey"></div>
```

```
<!DOCTYPE html>  <html>  <body>
<h1>My First Google Map</h1>
<div id="map" style="width:400px;height:400px;background:grey"></div>
<script>
function myMap() {
var mapOptions = {
    center: new google.maps.LatLng(51.5, -0.12),
    zoom: 10,
    mapTypeId: google.maps.MapTypeId.HYBRID      }
var map = new google.maps.Map(document.getElementById("map"), mapOptions);   }
</script>
<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBu-
  916DdpKAjTmJNIgngS6HL_kDIKU0aU&callback=myMap"></script>
</body>  </html>
```

mapOptions: It is a variable which defines the properties for the map.

center: It specifies where to center the map (using latitude and longitude coordinates).

zoom: It specifies the zoom level for the map (try to experiment with the zoom level).

mapTypeId: It specifies the map type to display. The following map types are supported: ROADMAP, SATELLITE, HYBRID, and TERRAIN.

Differences Between SVG and Canvas

- Resolution independent
- SVG has better scalability. So it can be printed with high quality at any resolution
- No support for event handlers
- You can save the resulting image as .png or .jpg
- Well suited for graphic-intensive games
- SVG gives better performance with smaller number of objects or larger surface.
- SVG can be modified through script and CSS
- SVG is vector based and composed of shapes.

- Resolution dependent
- Canvas has poor scalability. Hence it is not suitable for printing on higher resolution
- Support for event handlers
- Slow rendering if complex (anything that uses the DOM a lot will be slow)
- Not suited for game application
- Canvas gives better performance with smaller surface or larger number of objects.
- Canvas can be modified through script only
- Canvas is raster based and composed of pixel.

SVG

SVG stands for Scalable Vector Graphics

SVG is used to define graphics for the Web

SVG is a W3C recommendation

SVG Circle:

```
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green" stroke-
    width="4" fill="yellow" />
</svg>
```

SVG Rectangle

Example

```
<svg width="400" height="100">
  <rect width="400" height="100"
   style="fill:rgb(0,0,255);stroke-
   width:10;stroke:rgb(0,0,0)" />
</svg>
```

SVG Ellipse - <ellipse>

The <ellipse> element is used to create an ellipse.

An ellipse is closely related to a circle. The difference is that an ellipse has an x and a y radius that differs from each other, while a circle has equal x and y radius:

```
<svg height="140" width="500">
 <ellipse cx="200" cy="80" rx="100" ry="50"
 style="fill:yellow;stroke:purple;stroke-width:2" />
</svg>
```

SVG Line - <line>

The <line> element is used to create a line:

```
<svg height="210" width="500">
 <line x1="0" y1="0" x2="200" y2="200"
  style="stroke:rgb(255,0,0);stroke-width:2"
  />
</svg>
```

SVG Text - <text>

The <text> element is used to define a text.

```
<svg height="30" width="200">
 <text x="0" y="15" fill="red">I love
  SVG!</text>
</svg>
```

YouTube

```
 <iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY">
</iframe>
```