The Complete JavaScript Development

FULL-STACK Development

UI Development + MEAN-STACK

(MongoDB Express AngularJS Node.js)

Linux
HTML 5
CSS3 and Bootstrap
Javascript

Git
SQL (MYSQL)
NOSQL (MongoDB)

Node.js
Express.js
Angular JS
SDLC and Agile(Jira or Agilefant)

HTML

HTML stands for Hypertext Markup Language, and it is the most widely used language to write Web Pages.

- Hypertext refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.
- As its name suggests, HTML is a Markup Language which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.

Now, HTML is being widely used to format web pages with the help of different tags available in HTML language

Prerequisites:

You will need a text editor, such as Notepad ,Atom etc and an Internet browser, such as Internet Explorer or Netscape.

Text editors

- Visual Studio Code - this is currently the best on the market for MEAN stack development url : https://code.visualstudio.com/
- Sublime Text - this is what we will use during this course url :https://www.sublimetext.com/
- Atom - this is what we will uses during this course url : https://atom.io/

Note: all of these are available on multiple platforms

The browsers

There are five major browsers used on desktop today: Chrome, Internet Explorer, Firefox, Safari and Opera.

The browser's main functionality

The main function of a browser is to present the web resource you choose, by requesting it from the server and displaying it in the browser window. The resource is usually an HTML document, but may also be a PDF, image, or some other type of content. The location of the resource is specified by the user using a URI (Uniform Resource Identifier).

The way the browser interprets and displays HTML files is specified in the HTML and CSS specifications. These specifications are maintained by the W3C (World Wide Web Consortium) organization, which is the standards organization for the web. For years browsers conformed to only a part of the specifications and developed their own extensions. That caused serious compatibility issues for web authors. Today most of the browsers more or less conform to the specifications.
Browser user interfaces have a lot in common with each other. Among the common user interface elements are:

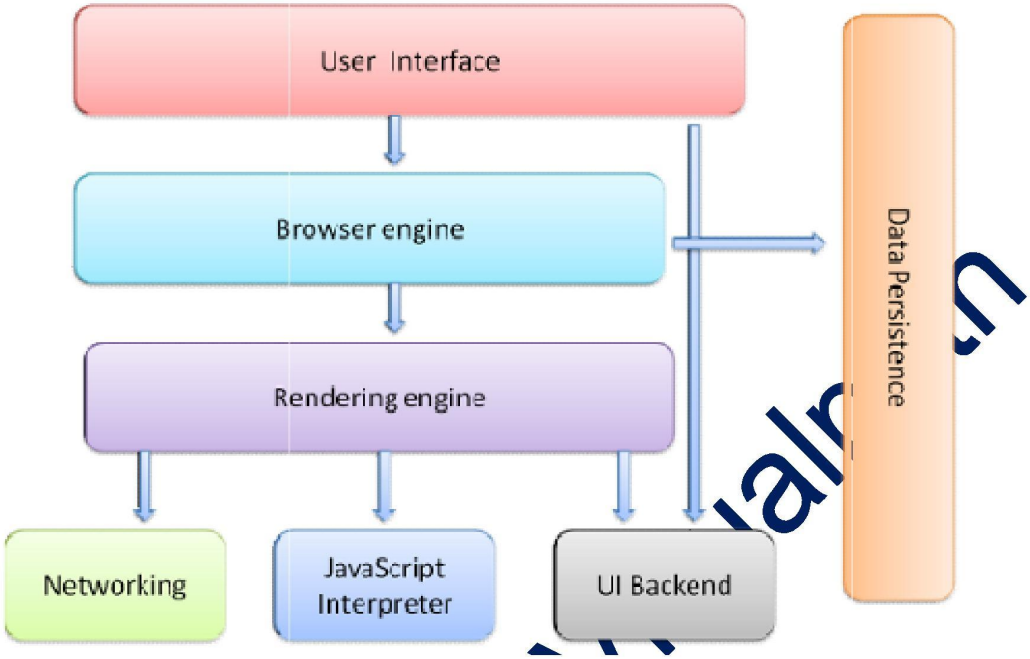
- Address bar for inserting a URI
- Back and forward buttons
- Bookmarking options
- Refresh and stop buttons for refreshing or stopping the loading of current documents
- Home button that takes you to your home page

Strangely enough, the browser's user interface is not specified in any formal specification, it just comes from good practices shaped over years of experience and by browsers imitating each other. The HTML5 specification doesn't define UI elements a browser must have, but lists some common elements. Among those are the address bar, status bar and tool bar. There are, of course, features unique to a specific browser like Firefox's downloads manager.

The browser's high level structure

The browser's main components are :

1. **The user interface**: this includes the address bar, back/forward button, bookmarking menu, etc. Every part of the browser display except the window where you see the requested page.
2. **The browser engine**: marshals actions between the UI and the rendering engine.
3. **The rendering engine** : responsible for displaying requested content. For example if the requested content is HTML, the rendering engine parses HTML and CSS, and displays the parsed content on the screen.
4. **Networking**: for network calls such as HTTP requests, using different implementations for different platform behind a platform-independent interface.
5. **UI backend**: used for drawing basic widgets like combo boxes and windows. This backend exposes a generic interface that is not platform specific. Underneath it uses operating system user interface methods.
6. **JavaScript interpreter**. Used to parse and execute JavaScript code.
7. **Data storage**. This is a persistence layer. The browser may need to save all sorts of data locally, such as cookies. Browsers also support storage mechanisms such as localStorage, IndexedDB, WebSQL and FileSystem.



The rendering engine

The responsibility of the rendering engine is well... Rendering, that is display of the requested contents on the browser screen.

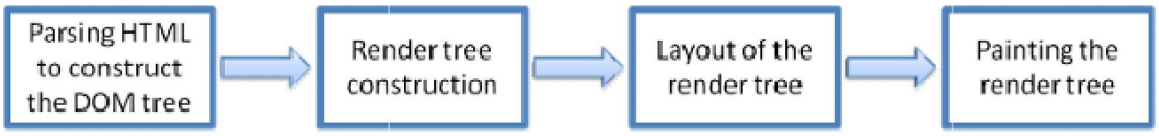
By default the rendering engine can display HTML and XML documents and images. It can display other types of data via plug-ins or extension; for example, displaying PDF documents using a PDF viewer plug-in. However, in this chapter we will focus on the main use case: displaying HTML and images that are formatted using CSS.

Rendering engines

Different browsers use different rendering engines: Internet Explorer uses Trident, Firefox uses Gecko, Safari uses WebKit. Chrome and Opera (from version 15) use Blink, a fork of WebKit.

The main flow

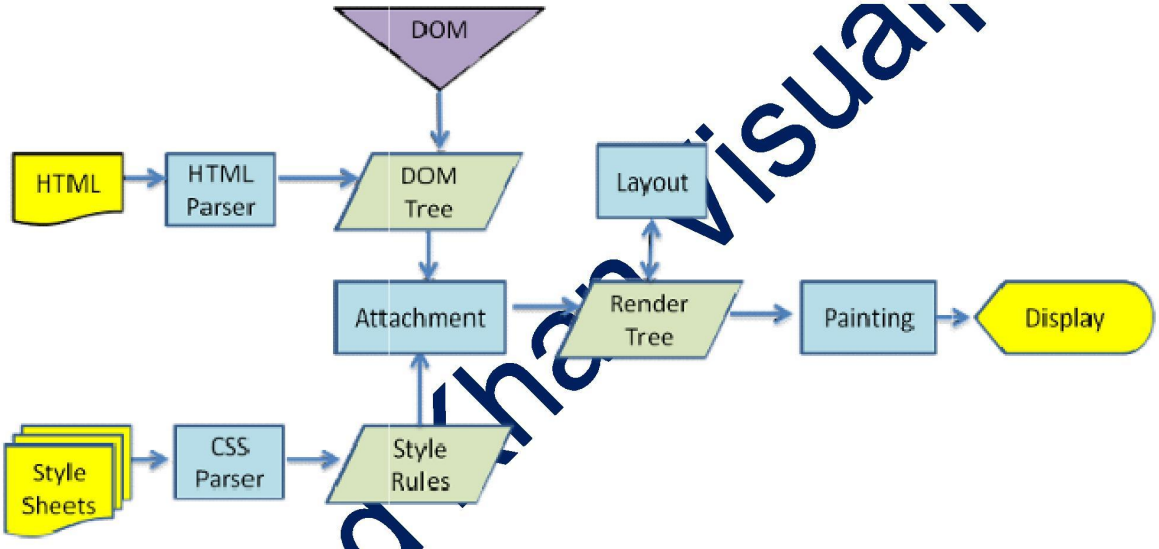
The rendering engine will start getting the contents of the requested document from the networking layer. This will usually be done in 8kB chunks.
After that, this is the basic flow of the rendering engine:



The rendering engine will start parsing the HTML document and convert elements to **DOM** nodes in a tree called the "content tree". The engine will parse the style data, both in external CSS files and in style elements. Styling information together with visual instructions in the HTML will be used to create another tree: the **render tree**.

After the construction of the render tree it goes through a "**layout**" process. This means giving each node the exact coordinates where it should appear on the screen. The next stage is **painting**–the render tree will be traversed and each node will be painted using the UI backend layer.

Main flow examples



Few Useful Terminologies:

How the Internet Works ?

The World Wide Web is the most popular part of the Internet by far. The Web allows rich and diverse communication by displaying text, graphics, animation, photos, sound and video. The Web physically consists of your personal computer, web browser software, a connection to an Internet Service Provider, computers called servers that host digital data, and routers and switches to direct the flow of information.

Components of the Internet:

The Internet is comprised of many components such a Email, FTP and Usenet News. The World Wide Web is simply one of these components.

World Wide Web (World Wide Web is like an Internet Library with millions of books)

- FTP
- E-mail & E-mail Discussion Groups
- Telnet
- Usenet News
- HTTP
- Chat & Instant Messaging

What is the World Wide Web?

One simple definition of the WWW is The WWW is a Hypertext Information System

Hypertext browsing:

- Non-Linear structure (not a book)
- You read what you want next
- Click on Hypertext links to navigate the WWW

Features of the WWW are:

- Graphical
- Easy to use
- Cross Platform
- Distributed
- Dynamic
- Interactive (forms, Java)

What is Url?

URLs (Uniform Resource Locators) are the addresses of the WWW pages

To view / read the WWW pages you must have a special application i.e. a web browser

A web server is a program that runs on web sites and is responsible for replying to a web browsers request for files

Web server: a system on the internet contains one or more web site. Server are basically ment to serve request.

Web site: a collection of one or more web pages

Web pages: single disk file with a single file name

Home pages: first page in website

What is an html File?

HTML is a format that tells a computer how to display a web page. The documents themselves are plain text files with special "tags" or codes that a web browser uses to interpret and display information on your computer screen.

- HTML stands for Hyper Text Markup Language.
- An HTML file is a text file containing small markup tags.
- The markup tags tell the Web browser how to display the page.
- An HTML file must have an htm or html file extension.

Try It? Open your text editor and type the following text:

```
<html>
  <head>
    <title>My First Webpage</title>
  </head>
  <body>This is my first homepage. <b>This text is bold</b> </body>
</html>
```

This is my first homepage. **This text is bold** Save the file as mypage.html. Start your browser. Select Open (or Open Page) in the File menu of your browser. A dialog box will appear. Select Browse (or Choose File) and locate the html file you just created - mypage.html - select it and click Open.

HTM or HTML Extension?

When you save an HTML file, you can use either the .htm or the .html extension. The .htm extension comes from the past when some of the commonly used software only allowed three letter extensions. It is perfectly safe to use either .html or .htm, but be consistent. mypage.htm and mypage.html are treated as different files by the browser.

How to View HTML Source?

A good way to learn HTML is to look at how other people have coded their html pages. To find out, simply click on the View option in your browsers toolbar and select Source or Page Source. This will open a window that shows you the actual HTML of the page. Go ahead and view the source html for this page.

HTML Tags What are HTML tags?

- HTML tags are used to mark-up HTML elements.
- HTML tags are surrounded by the two characters < and >
- The surrounding characters are called angle brackets
- HTML tags normally come in pairs like **and **
- The first tag in a pair is the start tag, the second tag is the end tag
- The text between the start and end tags is the element content
- HTML tags are not case sensitive, **means the same as <B>**

Why Use Lowercase Tags?

You may notice we've used lowercase tags even though I said that HTML tags are not case sensitive.<B> means the same as <b>. The World Wide Web Consortium (W3C), the group responsible for developing web standards, recommends lowercase tags in their HTML 4 recommendation, and XHTM (the next generation HTML) requires lowercase tags.

Tag Attributes

Tags can have attributes. Attributes can provide additional information about the HTML elements on your page. The <tag> tells the browser to do something, while the attribute tells the browser how to do it. For instance, if we add the bgcolor attribute, we can tell the browser that the background color of your page should be blue, like this: <body bgcolor="blue">.

This tag defines an HTML table: <table>. With an added border attribute, you can tell the browser that the table should have no borders: <table border="0">. Attributes always come in name/value pairs like this: name="value". Attributes are always added to the start tag of an HTML element and the value is surrounded by quotes.

Quote Styles, "red" or 'red'?

Attribute values should always be enclosed in quotes. Double style quotes are the most common, but single style quotes are also allowed. In some rare situations, like when the attribute value itself contains quotes, it is necessary to use single quotes:
name='George "machine Gun" Kelly'

Note: Some tags we will discuss are deprecated, meaning the World Wide Web Consortium (W3C) the governing body that sets HTML, XML, CSS, and other technical standards decided those tags and attributes are marked for deletion in future versions of HTML and XHTML. Browsers should continue to support deprecated tags and attributes, but eventually these tags are likely to become obsolete and so future support cannot be guaranteed.

Basic HTML Tags

The most important tags in HTML are tags that define headings, paragraphs and line breaks.
Basic HTML Tags
Tag Description
<html> Defines an HTML document
<body> Defines the document's body
<h1> to <h6> Defines header 1 to header 6
<p> Defines a paragraph

 Inserts a single line break
<hr> Defines a horizontal rule
<!--> Defines a comment

HTML5

HTML5 standards is not just a new HTML elements.it's a effect of collection of standards.HTML5 is not only there for HTML instead it is for Javascript and CSS tree are part of the new HTML5 definition.

HTML4 standards was so limited and how do we manage we manage with the plugins such as flash plugin.HTML5 solve this difficulty.Operating website is higher on Mobiles than the Desktop PC.HTML5 provide large amount of cross platform functionality .

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. HTML5 is the latest and most enhanced version of HTML.The two major organizations have been involved in developing of HTML5 since its initial time. One is W3C (World Wide Web Consortium) and the other one is WHATWG (Web Hypertext Application Technology Working Group).

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>title of the document</title>
  </head>
  <body>
    <header> starting of  the content </header>

     <main> body part of the content</main>
    <footer>end of the content</footer>
  </body>
</html>
```

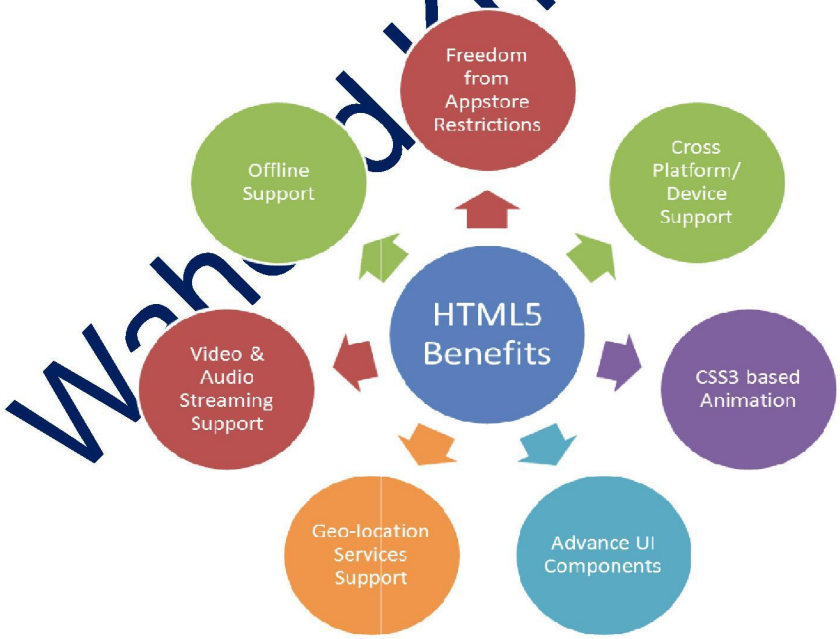
Befor HTML5,we just have to work with general containers and it was a problems for search engines.

```
<html>
  <head>
    <title>title</title>
  </head>
  <body>
    <div id="header">content here</div>

    <div id="main">content here</div>

    <div id="footer">content here</div>
  </body>
</html>
```

New Features:

HTML5 introduces a number of new elements and attributes that helps in building a modern website. Following are great features introduced in HTML5.

- **New Semantic Elements** – These are like <header>, <footer>, and <section>.
- **New Attributes of form elements** – Improvements to HTML web forms where new attributes have been introduced for <input> tag.
- New **graphic elements** -- <svg> and <canvas>.
- **Persistent Local Storage** – To achieve without resorting to third-party plugins.
- **WebSocket** – A a next-generation bidirectional communication technology for web applications.
- **Server-Sent Event**s – HTML5 introduces events which flow from web server to the web browsers and they are called Server-Sent Events (SSE).
- **Canvas** – This supports a two-dimensional drawing surface that you can program with JavaScript.
- **Audio & Video** – You can embed audio or video on your web pages without resorting to third-party plugins.
- **Geolocation** – Now visitors can choose to share their physical location with your web application.
- **Microdata** – This lets you create your own vocabularies beyond HTML5 and extend your web pages with custom semantics.
- **Drag and drop** – Drag and drop the items from one location to another location on a the same webpage.



Exercise 1:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
    <title>Joe's Coffee Store!</title>
    <link rel="stylesheet"  href="Styles/Layout.css"    type="text/css" />
</head>
<body>
      <p> </p>
</body>
</html>
```

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

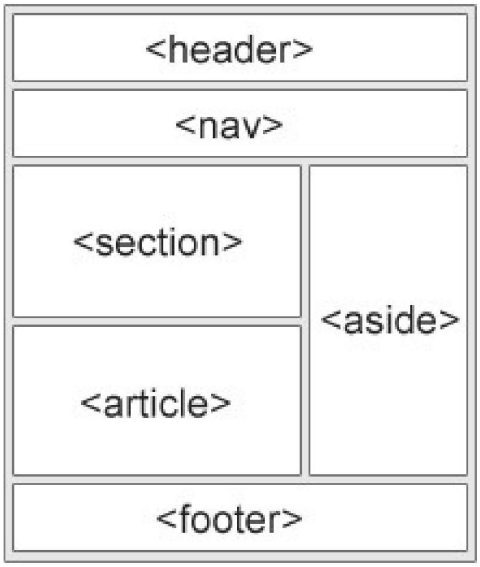
- Examples of non-semantic elements: and - Tells nothing about its content.
- Examples of semantic elements: <form>, <table> and <article> - Clearly defines its content.

New Semantic Elements in HTML5

Many web sites contain HTML code like:

to indicate navigation, header, and footer.
HTML5 offers new semantic elements to define different parts of a web page:

Tags	Description
<article>	Defines an article in a document.
<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>	Defines self-contained content
<footer>	Defines a footer for a document or section
<header>	Defines a header for a document or section
<main>	Defines 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



Why Semantic Elements?

With HTML4, developers used their own id/class names to style elements: header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav, etc.
This made it impossible for search engines to identify the correct web page content.
With the new HTML5 elements (<header> <footer> <nav> <section> <article>), this will become easier.
Semantic Web: "Allows data to be shared and reused across applications, enterprises, and communities."

- **HTML5 <section> Element**

The <section> element defines a section in a document.According to W3C's HTML5 documentation: "A section is a thematic grouping of content, typically with a heading."
A home page could normally be split into sections for introduction, content, and contact information.

```
<!DOCTYPE html>
<html>
<body>
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is an international organization working on issues regarding the conservation and so on....</p>
</section>
<section>
  <h1>WWF's Panda symbol</h1>
  <p>The Panda has become the symbol of WWF. The well-known panda logo of WWF originated from a panda named Chi Chi</p>
</section>
</body>
</html>
```

- **HTML5 <article> Element**

The <article> element specifies independent, self-contained content.
An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.
Examples of where an <article> element can be used:

- Forum post
- Blog post
- Newspaper article

```
<!DOCTYPE html>
<html>
<body>
<article>
  <h1>What Does WWF Do</h1>
  <p>WWF's mission is to stop the degradation of our planet's natural environment, and build a future in which humans live in harmony with nature.</p>
</article>
</body>
</html>
```

Nesting <article> in <section> or Vice Versa?

The <article> element specifies independent, self-contained content.The <section> element defines section in a document.Can we use the definitions to decide how to nest those elements? No, we cannot!
So, on the Internet you will find HTML pages with <section> elements containing <article> elements, and <article> elements containing <sections> elements. You will also find pages with <section> elements containing <section> elements, and <article> elements containing <article> elements.
Example for a newspaper: The sport articles in the sport section, may have a technical section in each article.

- **HTML5 <header> Element**

The <header> element specifies a header for a document or section.The <header> element should be used as a container for introductory content. You can have several <header> elements in one document.

The following example defines a header for an article:

```
<!DOCTYPE html>
<html>
<body>
<article>
  <header>
    <h1>WWF's mission:</h1>
    <p>WWF's mission is to stop the degradation of our planet's natural environment and so on.</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural environment, and build a future in which humans live in harmony with nature.</p>
</article>
</body>
</html>
```

- **HTML5 <footer> Element**

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. You may have several <footer> elements in one document.

```
<!DOCTYPE html>
<html>
<body>
<footer>
  <p>Posted by: someone</p>
  <p>Contact information: <a href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>
</body>
</html>
```

- **HTML5 <nav> Element**

The <nav> element defines a set of navigation links.
Notice: Not all links of a document should be inside a <nav> element. The <nav> element is intended only for major block of navigation links.

```
<!DOCTYPE html>
<html>
<body>
<nav>
 <a href="/html/">HTML</a> |
 <a href="/css/">CSS</a> |
 <a href="/js/">JavaScript</a> |
 <a href="/jquery/">jQuery</a>
</nav>
</body>
</html>
```

- **HTML5 <aside> Element**

The <aside> element defines some content aside from the content it is placed in (like a sidebar). The aside content should be related to the surrounding content.using aside we can display images also.

```
<!DOCTYPE html>
<html>
<body>
<p>My family and I visited the chennai this summer.</p>
<aside>
 <h4>Aside Element</h4>
 <p> Web Design makes your web page look good on all devices (desktops, tablets, and phones).</p>
</aside>
</body>
</html>
```

- **HTML5 <figure> and <figcaption> Elements**

The purpose of a figure caption is to add a visual explanation to an image.In HTML5, an image and a caption can be grouped together in a <figure> element:

```
<!DOCTYPE html>
<html>
<body>
<p>Figure and Figcaption.</p>
<figure>
 <img src="img_pulpit.jpg" alt="The Pulpit Rock" width="304" height="228">
```

```
 <figcaption>Fig.1 - The Pulpit Rock, Norway.</figcaption>
</figure>
</body>
</html>
```

Migration from HTML4 to HTML5

This chapter is entirely about how to migrate from HTML4 to HTML5. This chapter demonstrates how to convert an HTML4 page into an HTML5 page, without destroying anything of the original content or structure.

Typical HTML4	Typical HTML5
	<header>
	<nav>
	<section>
	<article>
	<footer>

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>HTML5</title>
<style>
body {
   font-family: Verdana,sans-serif;
   font-size: 0.9em;
}
header, footer {
   padding: 10px;
   color: white;
   background-color: black;
}
section {
   margin: 5px;
```

```
    padding: 10px;
    background-color: lightgrey;
}
article {
    margin: 5px;
    padding: 10px;
    background-color: white;
}

nav ul {
    padding: 0;
}
nav ul li {
    display: inline;
    margin: 5px;
}
</style>
</head>
<body>
<header>
<h1>India Today</h1>
</header>
<nav>
<ul>
  <li>News</li>
  <li>Sports</li>
  <li>Weather</li>
</ul>
</nav>
<section>
<h2>Sport Section</h2>
<article>
<h2>News 1</h2>
  <p>news1 content</p>
  <p>news1 content</p>
</article>
<article>
<h2>News 2</h2>
  <p>news2 content</p>
  <p>news2 content</p>
</article>
</section>
<footer>
<p>&copy; 2014 India Today. All rights reserved.</p>
</footer>
</body>
</html>
```

HTML Graphics

The HTML <canvas> element is used to draw graphics on a web page. The graphic to the left is created with <canvas>. It shows four elements: a red rectangle, a gradient rectangle, a multicolor rectangle, and a multicolor text.

What is HTML Canvas?

The HTML <canvas> element is used to draw graphics, on the fly, via JavaScript.
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.

Canvas Examples:
A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.
The markup looks like this:
<canvas id="myCanvas" width="200" height="100"></canvas>

Note: Always specify an id attribute (to be referred to in a script), and a width and height attribute to define the size of the canvas. To add a border, use the style attribute.
Here is an example of a basic, empty canvas:

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

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
Your browser does not support the HTML5 canvas tag.
</canvas>

</body>
</html>
```

Steps to implement canvas

Step 1: 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

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.
The fillStyle property can be a CSS color, a gradient, or a pattern. The default fillStyle is black.
Set the fill style of the drawing object to the color red:

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

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

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

All drawing on the HTML canvas must be done with JavaScript:

```
<html>
<body>
<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #c3c3c3;">
Your browser does not support the canvas element.
nvas id="myCanvas" width="200" height="100"
style="border:1px solid #c3c3c3;">
Your browser does not support the canvas element.
```

```
</canvas>
<script>
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.fillStyle = "#FF0000";
ctx.fillRect(0,0,150,75);
</script>
</body>
</html>
```

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:

```
<html>
<body>

<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #d3d3d3;">
Your browser does not support the canvas element.
</canvas>

<script>

var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
</script>
</body>
</html>
```

Canvas - Images

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

- drawImage(image,x,y)

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

<p>Image to use:</p>
<img id="scream" src="img_the_scream.jpg" alt="The Scream" width="220" height="277">

<p>Canvas to fill:</p>
<canvas id="myCanvas" width="250" height="300"
style="border:1px solid #d3d3d3;">
Your browser does not support the HTML5 canvas tag.</canvas>

<p><button onclick="myCanvas()">Try it</button></p>

<script>
function myCanvas() {
   var c = document.getElementById("myCanvas");
   var ctx = c.getContext("2d");
   var img = document.getElementById("scream");
   ctx.drawImage(img,10,10);
}
</script>

</body>
</html>
```


Drawing Text on the Canvas

To draw text on a canvas, the most important property and methods are:

- font - defines the font properties for the text
- fillText(*text,x,y*) - draws "filled" text on the canvas
- strokeText(*text,x,y*) - draws text on the canvas (no fill)

Using fillText()

Example:

Set font to 30px "Arial" and write a filled text on the canvas:


```
<html>
<body>
    <canvas id="myCanvas" width="200" height="100"
    style="border:1px solid #d3d3d3;">
    Your browser does not support the canvas element.
  </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>
```

Using strokeText():

Example:

Set font to 30px "Arial" and write a text, with no fill, on the canva

```
<html>
<body>
<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #d3d3d3;">
Your browser does not support the canvas element.
</canvas>
<script>
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World",10,50);
</script>
</body>
</html>
```

What is SVG?

- SVG stands for Scalable Vector Graphics
- SVG is used to define graphics for the Web
- SVG is a W3C recommendation.
- SVG graphics do NOT lose any quality if they are zoomed or resized

The HTML <svg> Element:

The HTML <svg> element is a container for SVG graphics.
SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

SVG Circle:

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

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40"
  stroke="green" stroke-width="4" fill="yellow" />
Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

- The cx and cy attributes define the x and y coordinates of the center of the circle. If cx and cy are omitted, the circle's center is set to (0,0)
- The r attribute defines the radius of the circle

SVG Rectangle:

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

<svg width="400" height="100">
  <rect width="400" height="100"
  style="fill:rgb(0,0,255);stroke-width:10;stroke:rgb(0,0,0)" />
Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

- The width and height attributes of the <rect> element define the height and the width of the rectangle
- The style attribute is used to define CSS properties for the rectangle
- The CSS fill property defines the fill color of the rectangle
- The CSS stroke-width property defines the width of the border of the rectangle
- The CSS stroke property defines the color of the border of the rectangle

Differences Between SVG and Canvas

- SVG is a language for describing 2D graphics in XML.
- Canvas draws 2D graphics, on the fly (with a JavaScript).
- SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element.
- In SVG, each drawn shape is remembered as an object. If attributes of an SVG object are changed, the browser can automatically re-render the shape.
- Canvas is rendered pixel by pixel. In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

HTML Google Maps

To demonstrate how to add a Google Map to a web page, we will use a basic HTML page:

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

<h1>Google Map</h1>

<div id="map">My map will go here</div>

</body>
</html>
```

Set the Map Size:

Set the size of the map

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

<h1>My First Google Map</h1>

<div id="map" style="width:400px;height:400px;background:yellow"></div>

</body>
</html>
```

Create a Function to Set The Map Properties

This example defines a Google Map centered in india:

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

<h1>Google Map</h1>

<div id="map" style="width:400px;height:400px;"></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>
</body>
</html>
```


The Above Example Explained:

The **mapOptions** variable defines the properties for the map.
The **center** property specifies where to center the map (using latitude and longitude coordinates).
The **zoom** property specifies the zoom level for the map (try to experiment with the zoom level).
The **mapTypeId** property specifies the map type to display. The following map types are supported: ROADMAP, SATELLITE, HYBRID, and TERRAIN.
The line: **var map=new google.maps.Map(document.getElementById("map"), mapOptions);** creates a new map inside the element with id="map", using the parameters that are passed (mapOptions).

Add the Google Maps API

show the map on the page!

The functionality of the map is provided by a JavaScript library located at Google. Add a script to refer to the Google Maps API with a callback to the myMap function:


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

<h1>My First Google Map</h1>

<div id="map" style="width:400px;height:400px;background:yellow"></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>
<!--
To use this code on your website, get a free API key from Google.
Read more at: https://www.w3schools.com/graphics/google_maps_basic.asp
-->

</body>
</html>
```


What is Multimedia?

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.
In this chapter you will learn about the different multimedia formats.

HTML5 Video

Playing Videos in HTML

Before HTML5, a video could only be played in a browser with a plug-in (like flash).
The HTML5 <video> element specifies a standard way to embed a video in a web page.

Tags	Description
<video>	Defines a video or movie
<source>	Defines multiple media resources for media elements, such as <video> and <audio>
<track>	Defines text tracks in media players

Attributes:

The <video> element has several special attributes that can change or enhance its default behavior.

Attributes	Description
autoplay	Tells the browser to immediately start downloading the video and play it as soon as it can. **Note**: mobile browsers generally do not support this attribute, the user must tap the screen to begin video playback.
poster	Provides an image to show before the video loads
controls	Shows the default video controls (play, pause, volume etc)
loop	Tells the browser to automatically loop the video.(how many times you want to repeat the video.)
muted	Mutes the audio from the video
height & width	Sets the width and height of the video.

Example:

```
<!DOCTYPE html>
<html>
<body>
<video  width="400"  height="240" controls  autoplay>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie..ogg" type="video/ogg">
<track src="subtitles_en.vtt"  kind="subtitles"  label="English">
  Your browser does not support HTML5 video.
</video>
<p>
Video courtesy of
<a href="https://www.bigbuckbunny.org/" target="_blank">Big Buck Bunny</a>.
</p>
</body>
</html>
```

How it Works

The 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.
The <source> element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.
The text between the <video> and </video> tags will only be displayed in browsers that do not support the <video> element.

HTML <audio> Element

To play an audio file in HTML, use the <audio> element.
The current HTML5 specification does not specify which audio formats browsers should support in the audio tag. But most commonly used audio formats are ogg, mp3and wav.

HTML Audio - How It Works

The controls attribute adds audio controls, like play, pause, and volume.
The <source> element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.
The text between the <audio> and </audio> tags will only be displayed in browsers that do not support the <audio> element.

HTML5 Audio Tags:

Tag	Description
<audio>	Defines sound content
<source>	Defines multiple media resources for media elements,such as <video>and <audio>

Example:

```
<!DOCTYPE html>
<html>
<body>
<audio controls  loop>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
</body>
</html>
```

Attribute	Description
controls	You need this to make the native audio player appear. Otherwise, you would have to use DOM to control the audio element to play your music
autoplay	If this guy exists, the browser will just play your song or your speech without asking permission from your visitor.
Loop	Keep repeating your music.
Src	The URL of your audio file.
Preload	This attribute was formerly known as "autobuffer" and it was an boolean attribute as "controls". none - do not buffer audio file automatically. metadata - only buffer the metadata of audio file. auto - buffer audio file before it gets played.
Muted	Mutes the audio from the video