HTML

Introduction

XML is Extensible markup language.

XML remains a meta-language like SGML, allowing users to create any tags needed

(Hence "extensible" with .xml extension) and then describing those tags and their permitted uses.

XML is now widely used for communicating data between applications.

HTML is hypertext markup language.

Markup language - annotating a document in a way to differentiate it from text.

HTML is used to create web pages.

Both HTML n XML r derived from SGML but are way more simplified to use than SGML.

HTML editors: Notepad++ (windows), sublime text, TextEdit (Mac), Netbeans

Web browsers (Example Google chrome, Firefox) can read HTML files (.html extension) and render them into visible or audible web pages containing text, images, hyperlink, etc.

HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.

HTML defines a web page with the help of HTML elements and other items.

HTML is focused on the separation of content (the visible text and images) from presentation (like color, font size, and layout).

Example: title is an element represents title of the document.

HTML: content of a page

CSS: presentation/layout of page

JAVASCRIPT: interactive/responsive /behavior

BOOTSTRAP : responsive designs

PHP: server-side scripting

MySQL: It stores the data in an orgainsed way, works with php codes

JavaScript is commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed.

It is Front End Script which makes your website more interactive.

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language.

While PHP originally stood for Personal Home Page,[4] it now stands for PHP: Hypertext Preprocessor.

When you request a webpage by putting the url in your browser or opening it's link, the web server accepts the request, take the HTML page, run the associated PHP script in it, convert the result obtained from PHP to HTML and then return it to your browser. In the browser all CSS and JavaScript runs.

It decides what to present to the html to show to you. Works behind the scene.

PHP is generally used for getting and putting data to/from the database (mostly MySQL).

Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++, or Java.

Python can serve as a scripting language for web applications, e.g., via mod wsgi.

Python IDEs PyCharm, PyDev, Wing IDE, Komodo, Spyder, Ninja Ide Etc.

Bootstrap is a free front-end framework for faster and easier web development

Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins

Bootstrap also gives you the ability to easily create responsive designs

FRONT END WEB DEVELOPMENT

The front end of a website is the part that users interact with. Everything that you see when you're navigating around the Internet, from fonts and colors to dropdown menus and sliders, is a combo of HTML, CSS, and JavaScript being controlled by your computer's browser.

SKILLS AND TOOLS

Front-end developers are responsible for a website's user-facing code and the architecture of its immersive user experiences. In order to execute those objectives, front-end devs must be adept at three main languages: HTML, CSS, and Javascript programming. In addition to fluency in these languages, front-end devs need to be familiar with frameworks **like Bootstrap, Foundation, Backbone, AngularJS, and EmberJS,** which ensure great-looking content no matter the device, and libraries like jQuery and LESS, which package code into a more useful, time-saving form. Ajax, a widely used technique for using Javascript that lets pages dynamically load by downloading server data in the background.

HOW IT TRANSLATES

A designer crafts the logo and graphics, a photographer takes the pictures, and a copywriter writes the text.

But a front-end dev assembled all of those pieces, translated them into web-speak, and built the experience you have with each page.

BACK END WEB DEVELOPMENT

The back end of a website consists of a server, an application, and a database. A back-end developer builds and maintains the technology that powers those components which, together, enable the user-facing side of the website to even exist in the first place.

SKILLS AND TOOLS

In order to make the server, application, and database communicate with each other, back-end devs use server-side languages like PHP, Ruby, Python, Java, and .Net to build an application, and tools like MySQL, Oracle, and SQL Server to find, save, or change data and serve it back to the user in front-end code. PHP frameworks like Zend, Symfony, and CakePHP are important; experience with version control software like SVN, CVS, or Git; and experience with Linux as a development and deployment system.

HOW IT TRANSLATES

When you navigated to a website, the servers send information to your computer or mobile device, which turned into a page you can see.That process is the result of a back-end developer's work.

FULL STACK WEB DEVELOPMENT

Full stack developers are jacks-of-all-trades.

A full stack developer can work cross-functionally on the full "stack" of technology, i.e. both the front end and back end. Full stack developers offer the full package.

SKILLS AND TOOLS

Full stack developers work, like back-end devs, on the server side of web programming, but they can also fluently speak the front-end languages that control how content looks on a site's user-facing side. They're jacks-of-all-trades.

HTML ELEMENTS

1. Document metadata

Metadata contains information about the page. This includes information about styles, scripts and data to help software (search engines, browsers, etc.) use and render the page. Metadata for styles and scripts may be defined in the page or link to another file that has the information.

Element	Description
<base>	The **HTML <base> element** specifies the base URL to use for all relative URLs contained within a document. There can be only one <base> element in a document.
<head>	The **HTML <head> element** provides general information (metadata) about the document, including its title and links to/definitions of scripts and style sheets. Thus the following can go inside head (title, base, style, link, meta, script, noscript).
<style>	The **HTML <style> element** contains style information for a document, or part of a document. By default, the style instructions written inside that element are expected to be CSS.
<title>	The **HTML <title> element** defines the title of the document, shown in a browser's title bar or on the page's tab. It can only contain text and any contained tags are not interpreted.

```
<head>
<base href="http://www.w3schools.com/images/" target="_blank">
</head>
<body>
<p><img src="stickman.gif" width="24" height="39" alt="Stickman"> - Notice that we have only specified a relative address for the image. Since we have specified a base URL in the head section, the browser will look for the image at "http://www.w3schools.com/images/stickman.gif".</p>
<p><a href="http://www.w3schools.com">W3Schools</a> - Notice that the link opens in a new window, even if it has no target="_blank" attribute. This is because the target attribute of the base element is set to "_blank".</p>
</body>
```

```
<head>
<base href="http://www.w3schools.com/images/" target="_blank">
<title>Page Title</title>
<style>
     body {background-color:yellow;}
      h1 {color:red;}
      p {color:blue;} </style>
<link rel="stylesheet" href="mystyle.css">
<meta name="keywords" content="HTML, CSS, XML, XHTML, JavaScript">
<script>
     document.write("Hello World!") </script>

<noscript>Your browser does not support JavaScript!</noscript>
</head>
```

2. Content sectioning

Content sectioning elements allow you to organize the document content into logical pieces. Use the sectioning elements to create a broad outline for your page content, including header and footer navigation, and heading elements to identify sections of content.

Element	Description
<address>	The **HTML <address> element** supplies contact information for its nearest <article> or <body> ancestor; in the latter case, it applies to the whole document. If the <address> element is inside the <body> element, it represents contact information for the document. If the <address> element is inside an <article> element, it represents contact information for that article. The text in the <address> element usually renders in *italic*. Most browsers will add a line break before and after the address element by default.
<article>	The **HTML <article> element** represents a self-contained composition in a document, page, application, or site, which is intended to be independently distributable or reusable (e.g., in syndication). This could be a forum post, a magazine or newspaper article, a blog entry, an object, 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.
<footer>	The **HTML <footer> element** represents a footer for its nearest sectioning content or sectioning root element. A footer typically contains information about the author of the section, copyright data or links to related documents.
<header>	The **HTML <header> element** represents a group of introductory or navigational aids. It may contain some heading elements but also other elements like a logo, wrapped section's header, a search form, and so on.
<h1>, <h2>, <h3>, <h4>, <h5>, <h6>	**Heading** elements implement six levels of document headings, <h1> is the most important and <h6> is the least. A heading element briefly describes the topic of the section it introduces. Heading information may be used by user agents, for example, to construct a table of contents for a document automatically.

Element	Description
<hgroup>	The **HTML <hgroup> Element** (*HTML Headings Group Element*) represents the heading of a section. It defines a single title that participates in the outline of the document as the head3ing of the implicit or explicit section that it belongs to.
<nav>	The **HTML <nav> element** (*HTML Navigation Element*) represents a section of a page that links to other pages or to parts within the page: a section with navigation links.
<section>	The **HTML <section> element** represents a generic section of a document, i.e., a thematic grouping of content, typically with a heading. Each <section> should be identified, typically by including a heading (<h1>-<h6> element) as a child of the <section> element.

```
<address>
   Written by <a href="mailto:webmaster@example.com">Jon Doe</a>.<br>
   Visit us at:<br>
   Example.com<br> </address>

<article>
  <h1>Google Chrome</h1>
  <p>Google Chrome is a free, open-source web</p> </article>

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

<header>
    <h1>Most important heading here</h1>
    <h3>Less important heading here</h3> </header>

<nav>
  <a href="/html/">HTML</a> | <a href="/css/">CSS</a> |
  <li><a href="/examples/">Examples</a></li>
 <li><a href="/references/">References</a></li> </nav>


<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is....</p></section>
```

3. Text content

Use HTML text content elements to organize blocks or sections of content placed between the opening <body> and closing </body> tags. Important for accessibility and SEO, these elements identify the purpose or structure of that content.

Element	Description
<dl>	The **HTML <dl> element** (or *HTML Description List Element*) encloses a list of pairs of terms and descriptions. Common uses for this element are to implement a glossary or to display metadata (a list of key-value pairs).
<dt>	The **HTML <dt> element** (or *HTML Definition Term Element*) identifies a term in a definition list. This element can occur only as a child element of a <dl>. It is usually followed by a <dd> element; however, multiple <dt> elements in a row indicate several terms that are all defined by the immediate next <dd> element.
<dd>	The **HTML <dd> element** (*HTML Description Element*) indicates the description of a term in a description list (<dl>) element. This element can occur only as a child element of a definition list and it must follow a <dt> element.
	The **HTML element** (or *HTML Document Division Element*) is the generic container for flow content, which does not inherently represent anything. It can be used to group elements for styling purposes (using the **class** or **id** attributes), or because they share attribute values, such as **lang**. It should be used only when no other semantic element (such as <article> or <nav>) is appropriate.
<figcaption>	The **HTML <figcaption> element** represents a caption or a legend associated with a figure or an illustration described by the rest of the data of the <figure> element which is its immediate ancestor which means <figcaption> can be the first or last element inside a <figure> block. Also, the *HTML Figcaption Element* is optional; if not provided, then the parent figure element will have no caption.
<figure>	The **HTML <figure> element** represents self-contained content, frequently with a caption (<figcaption>), and is typically referenced as a single unit. While it is related to the main flow, its position is independent of the main flow. Usually this is an image, an illustration, a diagram, a code snippet, or a schema that is referenced in the main text, but that can be moved to another page or to an appendix without affecting the main flow.

Element	Description
<hr>	The **HTML <hr> element** represents a thematic break between paragraph-level elements (for example, a change of scene in a story, or a shift of topic with a section). In previous versions of HTML, it represented a horizontal rule. It may still be displayed as a horizontal rule in visual browsers, but is now defined in semantic terms, rather than presentational terms.
	The **HTML element** (or *HTML List Item Element*) is used to represent an item in a list. It must be contained in a parent element: an ordered list ( ), an unordered list ( ), or a menu (<menu>). In menus and unordered lists, list items are usually displayed using bullet points. In ordered lists, they are usually displayed with an ascending counter on the left, such as a number or letter.
<main>	The **HTML <main> element** represents the main content of the <body> of a document or application. The main content area consists of content that is directly related to, or expands upon the central topic of a document or the central functionality of an application. This content should be unique to the document, excluding any content that is repeated across a set of documents such as sidebars, navigation links, copyright information, site logos, and search forms (unless the document's main function is as a search form).
	The **HTML Element** (or *HTML Ordered List Element*) represents an ordered list of items. Typically, ordered-list items are displayed with a preceding numbering, which can be of any form, like numerals, letters or Romans numerals or even simple bullets. This numbered style is not defined in the HTML description of the page, but in its associated CSS, using the list-style-type property.
<p>	The **HTML <p> element** *(*or *HTML Paragraph Element)* represents a paragraph of text.
<pre>	The **HTML <pre> element** (or *HTML Preformatted Text*) represents preformatted text. Text within this element is typically displayed in a non-proportional font exactly as it is laid out in the file. Whitespaces inside this element are displayed as typed.
	The **HTML element** (or *HTML Unordered List Element*) represents an unordered list of items, namely a collection of items that do not have a numerical ordering, and their order in the list is meaningless. Typically, unordered-list items

Element	Description
	are displayed with a bullet, which can be of several forms, like a dot, a circle or a squared. The bullet style is not defined in the HTML description of the page, but in its associated CSS, using the list-style-type property.

```
<dl>
	<dt>Dog</dt>
	<dd>A carnivorous mammal of the family Canidae.</dd>

	<dt>HTML</dt>
	<dd>HyperText Markup Language.</dd>
	<dd>A language used to format web documents.</dd>
</dl>


<div id =”design”>

	<h1> this will be linked with  id and can be formatted individually using
html/css<h1></div>

<figure>

	<img src="obelisk.jpg" alt="obelisk">

	<figcaption>Tixall Obelisk</figcaption>

</figure>

<hr>

	<ol> // ordered list

		 <li>Coffee</li>

		 <li>Milk</li>

	</ol>

	<ul> // unordered list

		<li>Coffee</li>

		<li>Tea</li>

	</ul>
```

4. Inline text semantics

Use the HTML inline text semantic to define the meaning, structure, or style of a word, line, or any arbitrary piece of text.

Element	Description
<abbr>	The **HTML <abbr> element** (or *HTML Abbreviation Element*) represents an abbreviation and optionally provides a full description for it. If present, the title attribute must contain this full description and nothing else.
<b>	The **HTML <b> Element** represents a span of text stylistically different from normal text, without conveying any special importance or relevance. It is typically used for keywords in a summary, product names in a review, or other spans of text whose typical presentation would be boldfaced. Another example of its use is to mark the lead sentence of each paragraph of an article.
<bdi>	The *HTML <bdi> Element* (or Bi-Directional Isolation Element) isolates a span of text that might be formatted in a different direction from other text outside it. It is useful when embedding user-generated content with an unknown directionality.
<bdo>	The **HTML <bdo> Element** (or *HTML bidirectional override element*) is used to override the current directionality of text. It causes the directionality of the characters to be ignored in favor of the specified directionality.
 	The HTML element *line break* produces a line break in text (carriage-return). It is useful for writing a poem or an address, where the division of lines is significant.
<cite>	The *HTML Citation Element* (**<cite>**) represents a reference to a creative work. It must include the title of a work or a URL reference, which may be in an abbreviated form according to the conventions used for the addition of citation metadata.
<code>	Summary
<data>	The **HTML <data> Element** links a given content with a machine-readable translation. If the content is time- or date-related, the <time> must be used.
<dfn>	The *HTML Definition Element* (**<dfn>**) represents the defining instance of a term.

Element Description

 The HTML element *emphasis* **** marks text that has stress emphasis. The element can be nested, with each level of nesting indicating a greater degree of emphasis.

```
<p>This web site is about <b><abbr title="HyperText Markup
Language">HTML</abbr><b></p>

<ul>

   <li>User <bdi>hrefs</bdi>: 60 points</li>

   <li>User <bdi>إيان</bdi>: 90 points</li>

</ul>

<p><bdo dir="rtl">This paragraph will go right-to-left.</bdo></p>

<p><bdo dir="ltr"> ti esrever </bdo></p>  <br>

<p><cite>The Scream</cite> by Edward Munch. Painted in 1893.</p>

<code>#include <stdio.h></code><br>

<input list="tutorials" /> <datalist id="tutorials">

  <option value="Java">

  <option value="PHP">

  <option value="Ruby">

  <option value="jQuery">

</datalist>

<p><dfn>HTML</dfn> is the standard markup language for creating web
pages.</p>

<em>Emphasized text</em><br><i> italic text</i>

<kbd>Keyboard input</kbd><br>
```

<i> The **HTML <i> Element** represents a range of text that is set off from the normal text for some reason, for example, technical terms, foreign language phrases, or fictional character thoughts. It is typically displayed in italic type.

<kbd> The *HTML Keyboard Input Element* (**<kbd>**) represents user input and produces an inline element displayed in the browser's default monospace font.

Element	Description
<mark>	The *HTML Mark Element* (<mark>) represents highlighted text, i.e., a run of text marked for reference purpose, due to its *relevance* in a particular context. For example it can be used in a page showing search results to highlight every instance of the searched-for word.
<q>	The *HTML Quote Element* (**<q>**) indicates that the enclosed text is a short inline quotation. This element is intended for short quotations that don't require paragraph breaks; for long quotations use <blockquote> element.
<rp>	The **HTML <rp> element** is used to provide fall-back parenthesis for browsers non-supporting ruby annotations. Ruby annotations are for showing pronounciation of East Asian characters, like using Japanese furigana or Taiwainese bopomofo characters. The <rp> element is used in the case of lack of <ruby> element support its content has what should be displayed in order to indicate the presence of a ruby annotation, usually parentheses.
<rt>	The **HTML <rt> Element** embraces pronunciation of characters presented in a ruby annotations, which are used to describe the pronunciation of East Asian characters. This element is always used inside a <ruby> element.
<rtc>	The **HTML <rtc> Element** embraces semantic annotations of characters presented in a ruby of <rb> elements used inside of <ruby> element. <rb> elements can have both pronunciation (<rt>) and semantic (<rtc>) annotations.
<ruby>	The **HTML <ruby> Element** represents a ruby annotation. Ruby annotations are for showing pronunciation of East Asian characters.
<s>	The *HTML Strikethrough Element* (<s>) renders text with a strikethrough, or a line through it. Use the <s> element to represent things that are no longer relevant or no longer accurate. However, <s> is not appropriate when indicating document edits; for that, use the <del> and <ins> elements, as appropriate.
<samp>	The HTML <samp> element is an element intended to identify sample output from a computer program. It is usually displayed in the browser's default monotype font (such as Lucida Console).

Element	Description
<small>	The HTML Small Element (<small>) makes the text *font size* one size smaller (for example, from large to medium, or from small to x-small) down to the browser's minimum font size. In HTML5, this element is repurposed to represent side-comments and small print, including copyright and legal text, independent of its styled presentation.
	The **HTML element** is a generic inline container for phrasing content, which does not inherently represent anything. It can be used to group elements for styling purposes (using the class or id attributes), or because they share attribute values, such as lang. It should be used only when no other semantic element is appropriate. is very much like a element, but is a block-level element whereas a is an inline element.
	The HTML Strong Element () gives text strong importance, and is typically displayed in bold.
<sub>	The *HTML Subscript Element* (<sub>) defines a span of text that should be displayed, for typographic reasons, lower, and often smaller, than the main span of text.
<sup>	The *HTML Superscript Element* (<sup>) defines a span of text that should be displayed, for typographic reasons, higher, and often smaller, than the main span of text.
<time>	The **HTML <time> element** represents either a time on a 24-hour clock or a precise date in the Gregorian calendar (with optional time and timezone information).
<u>	The *HTML Underline Element* (<u>) renders text with an underline, a line under the baseline of its content.
<var>	The HTML Variable Element (<var>) represents a variable in a mathematical expression or a programming context.
<wbr>	The HTML element *word break opportunity* <wbr> represents a position within text where the browser may optionally break a line, though its line-breaking rules would not otherwise create a break at that location.

```
<p>Do not forget to buy <mark>milk</mark> today.</p>

<p>So I asked Bob and he said <q>I know as much about quotations as I do about
pigeon fancying</q></p>

<p> now its updated to <s>HTML4</s> HTML5</p>

<samp>Sample output from a computer program</samp><br>

<p>W3Schools.com <small> the world's largest web development site.</small></p>

<p>This is  <span style="color:blue">crazy</span></p>

<p> This is <strong> Strong </strong></p>

<p> This is <sub> Subscript </sub><sup> Superscript </sup></p>

<p>We open at <time>10:00</time> every morning.</p>

<p>I have a date on <time datetime="2008-02-14 20:00">Valentines
day</time>.</p>

<p><u> underline</u></p>

<var>Variable</var>

<p> This is a
veryveryveryveryveryveryveryveryveryveryveryveryvery<wbr>longwordthatwillbreakatsp
ecific<wbr>placeswhenthebrowserwindowisresized.</p>

<p><b>Note:</b> The wbr element is not supported in IE.</p>
```

5. Image and multimedia

HTML supports various multimedia ressources such as images, audio, and video.

Element	Description
<area>	The *HTML <area> element* defines a hot-spot region on an image, and optionally associates it with a hypertext link. This element is used only within a <map> element.
<audio>	The **HTML <audio> element** is used to embed sound content in documents. It may contain several audio sources, represented using the src attribute or the <source> element; the browser will choose the most suitable one. Currently, there are 3 supported file formats for the <audio> element: MP3, Wav, and Ogg:

Element Description

<map> The **HTML <map> element** is used with <area> elements to define an image map (a clickable link area).

<track> The **HTML <track> element** is used as a child of the media elements—<audio> and <video>. It lets you specify timed text tracks (or time-based data), for example to automatically handle subtitles. The tracks are formatted in WebVTT format (.vtt files) — Web Video Text Tracks.

<video> Use the **HTML <video> element** to embed video content in a document. The video element contains one or more video sources. To specify a video source, use either the src attribute or the <source> element; the browser will choose the most suitable one.

```
<img src="planets.gif" width="145" height="126" alt="Planets"
usemap="#planetmap">
<map name="planetmap">
  <area shape="rect" coords="0,0,82,126" href="sun.htm" alt="Sun">
  <area shape="circle" coords="90,58,3" href="mercur.htm" alt="Mercury">
  <area shape="circle" coords="124,58,8" href="venus.htm" alt="Venus">
</map>

<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
  Your browser does not support the audio tag.
</audio>

<video width="320" height="240" controls>
  <source src="forrest_gump.mp4" type="video/mp4">
  <source src="forrest_gump.ogg" type="video/ogg">
  <track src="subtitles_en.vtt" kind="subtitles" srclang="en"
label="English">
  <track src="subtitles_no.vtt" kind="subtitles" srclang="no"
label="Norwegian">
</video>
```

6. Embedded content

In addition to regular multimedia content, HTML can include a variety of other content, even if it's not always easy to interact with.

Element	Description
<embed>	The **HTML <embed> Element** represents an integration point for an external application or interactive content (in other words, a plug-in).
<object>	The *HTML Embedded Object Element* (**<object>**) represents an external resource, which can be treated as an image, a nested browsing context, or a resource to be handled by a plugin.
<param>	The **HTML <param> Element** (or *HTML Parameter Element)* defines parameters for <object>.
<source>	The **HTML <source> element** is used to specify multiple media resources for <picture>, <audio> and <video> elements. It is an empty element. It is commonly used to serve the same media in multiple formats supported by different browsers.

```
<embed src="helloworld.swf">


<object width="400" height="400"
data="helloworld.swf"></object>


<object data="horse.wav">
  <param name="autoplay" value="true">
</object>


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

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

7. Scripting

In order to create dynamic content and Web applications, HTML supports the use of scripting languages, most prominently JavaScript. Certain elements support this capability.

Element	Description
<canvas>	The **HTML <canvas> Element** can be used to draw graphics via scripting (usually JavaScript). For example, it can be used to draw graphs, make photo compositions or even perform animations. You may (and should) provide alternate content inside the <canvas> block. That content will be rendered both on older browsers that don't support canvas **and** in browsers with JavaScript disabled.
<noscript>	The **HTML <noscript> Element** defines a section of html to be inserted if a script type on the page is unsupported or if scripting is currently turned off in the browser.
<script>	The *HTML Script Element* (**<script>**) is used to embed or reference an executable script within an HTML or XHTML document.

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

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

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

- HTML Canvas Can Draw Text - Canvas can draw colorful text, with or without animation.
- HTML Canvas Can Draw Graphics - Canvas has great features for graphical data presentation with an imagery of graphs and charts.
- HTML Canvas Can be Animated - Canvas objects can move. Everything is possible: from simple bouncing balls to complex animations.
- HTML Canvas Can be Interactive - Canvas can respond to JavaScript events, Canvas can respond to any user action (key clicks, mouse clicks, button clicks, finger movement).
- HTML Canvas Can be Used in Games

Draw on the Canvas With JavaScript

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

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

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

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);
```

Canvas Coordinates

The HTML canvas is a two-dimensional grid.

The upper-left corner of the canvas has the coordinates (0,0)

In the previous chapter, you saw this method used: fillRect(0,0,150,75).

This means: 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 a Circle

To draw a circle on a canvas, use the following methods:

- beginPath();
- arc(x,y,r,start,stop)

Example

Define a circle with the arc() method. Then use the stroke() method to actually draw the circle:

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

```
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
```

Canvas - Gradients

Gradients can be used to fill rectangles, circles, lines, text, etc. Shapes on the canvas are not limited to solid colors.There are two different types of gradients:

- createLinearGradient(*x,y,x1,y1*) - creates a linear gradient
- createRadialGradient(*x,y,r,x1,y1,r1*) - creates a radial/circular gradient

Once we have a gradient object, we must add two or more color stops.

The addColorStop() method specifies the color stops, and its position along the gradient. Gradient positions can be anywhere between 0 to 1.

To use the gradient, set the fillStyle or strokeStyle property to the gradient, then draw the shape (rectangle, text, or a line).

Using createLinearGradient()

Example: Create a linear gradient. Fill rectangle with the gradient:

```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");

// Create gradient
var grd=ctx.createLinearGradient(0,0,200,0);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");

// Fill with gradient
ctx.fillStyle=grd;
ctx.fillRect(10,10,150,80);
```

Using createRadialGradient():

Example: Create a radial/circular gradient. Fill rectangle with the gradient:

```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");

// Create gradient
var grd=ctx.createRadialGradient(75,50,5,90,60,100);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");
```

```
// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
```

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:

JavaScript:

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

✓ **Using strokeText()**

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

JavaScript:

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World",10,50);
```

Add Color and Center Text

Example: Set font to 30px "Comic Sans MS" and write a filled red text in the center of the canvas:

JavaScript:

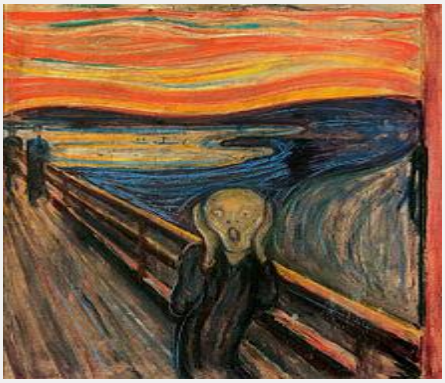
```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.font = "30px Comic Sans MS";
ctx.fillStyle = "red";
ctx.textAlign = "center";
ctx.fillText("Hello World", canvas.width/2, canvas.height/2);
```

Canvas - Images

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

- drawImage(*image,x,y*)

Example



JavaScript:

```
window.onload = function() {
    var canvas = document.getElementById("myCanvas");
    var ctx = canvas.getContext("2d");
    var img = document.getElementById("scream");
    ctx.drawImage(img, 10, 10);
};
```

HTML Canvas Reference

Description

The HTML5 <canvas> tag is used to draw graphics, on the fly, via scripting (usually JavaScript).

However, the <canvas> element has no drawing abilities of its own (it is only a container for graphics) - you must use a script to actually draw the graphics.

The getContext() method returns an object that provides methods and properties for drawing on the canvas.

This reference will cover the properties and methods of the getContext("2d") object, which can be used to draw text, lines, boxes, circles, and more - on the canvas.

Colors, Styles, and Shadows

Property	Description
fillStyle	Sets or returns the color, gradient, or pattern used to fill the drawing
strokeStyle	Sets or returns the color, gradient, or pattern used for strokes
shadowColor	Sets or returns the color to use for shadows
shadowBlur	Sets or returns the blur level for shadows
shadowOffsetX	Sets or returns the horizontal distance of the shadow from the shape
shadowOffsetY	Sets or returns the vertical distance of the shadow from the shape

Method	Description
createLinearGradient()	Creates a linear gradient (to use on canvas content)

createPattern()	Repeats a specified element in the specified direction
createRadialGradient()	Creates a radial/circular gradient (to use on canvas content)
addColorStop()	Specifies the colors and stop positions in a gradient object

Line Styles

Property	Description
lineCap	Sets or returns the style of the end caps for a line
lineJoin	Sets or returns the type of corner created, when two lines meet
lineWidth	Sets or returns the current line width
miterLimit	Sets or returns the maximum miter length

Rectangles

Method	Description
rect()	Creates a rectangle
fillRect()	Draws a "filled" rectangle
strokeRect()	Draws a rectangle (no fill)
clearRect()	Clears the specified pixels within a given rectangle

Paths

Method	Description
fill()	Fills the current drawing (path)
stroke()	Actually draws the path you have defined
beginPath()	Begins a path, or resets the current path
moveTo()	Moves the path to the specified point in the canvas, without creating a line
closePath()	Creates a path from the current point back to the starting point
lineTo()	Adds a new point and creates a line to that point from the last specified point in the canvas
clip()	Clips a region of any shape and size from the original canvas
quadraticCurveTo()	Creates a quadratic Bézier curve
bezierCurveTo()	Creates a cubic Bézier curve
arc()	Creates an arc/curve (used to create circles, or parts of circles)
arcTo()	Creates an arc/curve between two tangents
isPointInPath()	Returns true if the specified point is in the current path, otherwise false

Transformations

Method	Description
scale()	Scales the current drawing bigger or smaller
rotate()	Rotates the current drawing
translate()	Remaps the (0,0) position on the canvas
transform()	Replaces the current transformation matrix for the drawing
setTransform()	Resets the current transform to the identity matrix. Then runs transform()

Text

Property	Description
font	Sets or returns the current font properties for text content
textAlign	Sets or returns the current alignment for text content
textBaseline	Sets or returns the current text baseline used when drawing text

Method	Description
fillText()	Draws "filled" text on the canvas
strokeText()	Draws text on the canvas (no fill)

measureText()	Returns an object that contains the width of the specified text

Image Drawing

Method	Description
drawImage()	Draws an image, canvas, or video onto the canvas

Pixel Manipulation

Property	Description
width	Returns the width of an ImageData object
height	Returns the height of an ImageData object
data	Returns an object that contains image data of a specified ImageData object

Method	Description
createImageData()	Creates a new, blank ImageData object
getImageData()	Returns an ImageData object that copies the pixel data for the specified rectangle on a canvas
putImageData()	Puts the image data (from a specified ImageData object) back onto the canvas

Compositing

Property	Description
globalAlpha	Sets or returns the current alpha or transparency value of the drawing
globalCompositeOperation	Sets or returns how a new image are drawn onto an existing image

Other

Method	Description
save()	Saves the state of the current context
restore()	Returns previously saved path state and attributes
createEvent()	-
getContext()	-
toDataURL()	-

8. Demarcating edits

These elements let you provide indications that specific parts of the text have been altered.

Element	Description
<del>	The *HTML Deleted Text Element* (**<del>**) represents a range of text that has been deleted from a document. This element is often (but need not be) rendered with strike-through text.
<ins>	The **HTML <ins> Element** (or *HTML Inserted Text*) HTML represents a range of text that has been added to a document. Ex: My favorite color is ~~blue~~ <ins>red</ins>!

9. Table content

The elements here are used to create and handle tabular data.

Element	Description
<caption>	The **HTML <caption> Element** (or *HTML Table Caption Element*) represents the title of a table. Though it is always the first descendant of a <table> (<caption> must be inserted immediately after <table> and You can specify only one caption per table), its styling, using CSS, may place it elsewhere, relative to the table.
<col>	The *HTML Table Column Element* (**<col>**) defines a column within a table and is used for defining common semantics on all common cells. It is generally found within a <colgroup> element. The <col> tag specifies column properties for each column within a <colgroup> element. The <col> tag is useful for applying styles to entire columns, instead of repeating the styles for each cell, for each row.
<colgroup>	The *HTML Table Column Group Element* (**<colgroup>**) defines a group of columns within a table.
<table>	The *HTML Table Element* (<table>) represents data in two dimensions or more.

Element	Description
<tbody>	The HTML **Table Body Element (<tbody>)** defines one or more <tr> element data-rows to be the body of its parent <table> element (as long as no <tr> elements are immediate children of that table element.) In conjunction with a preceding <thead> and/or <tfoot> element, <tbody> provides additional semantic information for devices such as printers and displays. Of the parent table's child elements, <tbody> represents the content which, when longer than a page, will most likely differ for each page printed; while the content of <thead> and <tfoot> will be the same or similar for each page printed. For displays, <tbody> will enable separate scrolling of the <thead>, <tfoot>, and <caption> elements of the same parent <table> element. Note that unlike the <thead>, <tfoot>, and <caption> elements however, multiple <tbody> elements are permitted (if consecutive), allowing the data-rows in long tables to be divided into different sections, each separately formatted as needed.
<td>	The *Table cell* HTML element (**<td>**) defines a cell of a table that contains data. It participates in the *table model*.
<tfoot>	The *HTML Table Foot Element* (<tfoot>) defines a set of rows summarizing the columns of the table.
<th>	The HTML element *table header cell* <th> defines a cell that is a header for a group of cells of a table. The group of cells that the header refers to is defined by the scope and headers attribute.
<thead>	The *HTML Table Head Element* (<thead>) defines a set of rows defining the head of the columns of the table.
<tr>	The HTML element *table row* <tr> defines a row of cells in a table. Those can be a mix of <td> and <th> elements.

```
<table>
<caption> Monthly savings </caption>
<tr>  <th>Month</th>
      <th>Savings</th> </tr>
      <tr>  <td>January</td>
      <td>$100</td>  </tr>
</table>

<head>
      <style> table, th, td { border: 1px solid black; } </style> </head>
<body> <table>
      <colgroup>
      <col span="2" style="background-color:red">
      <col style="background-color:yellow"></colgroup>
      <tr> <th>Title</th> </tr>
      <tr> <td>My first HTML</td> </tr>
      <tr> <td>My first CSS</td> </tr>
</table> </body>

<table>
<thead>
<tr> <th>Month</th>
<th>Savings</th> </tr></thead>
<tfoot>
<tr>  <td>Sum</td>
<td>$180</td> </tr></tfoot>
<tbody>
<tr>  <td>January</td>
<td>$100</td> </tr>
<tr> <td>February</td>
<td>$80</td> </tr></tbody>
</table><p><b>Tip:</b> The thead, tbody, and tfoot elements will not affect
the layout of the table by default. However, you can use CSS to style</p>
```

10. Forms

HTML provides a number of elements which can be used together to create forms which the user can fill out and submit to the Web site or application. There's a great deal of added information about this available in the HTML forms guide.

Element	Description
<button>	The **HTML <*button*> Element** represents a clickable button.
<datalist>	The *HTML Datalist Element* (**<datalist>**) contains a set of <option> elements that represent the values available for other controls.
<fieldset>	The **HTML <fieldset> element** is used to group several controls as well as labels (<label>) within a web form.
<form>	The **HTML <form> element** represents a document section that contains interactive controls to submit information to a web server.
<input>	The HTML element <input> is used to create interactive controls for web-based forms in order to accept data from the user. How an <input> works varies considerably depending on the value of its type attribute.
<keygen>	The HTML <keygen> element exists to facilitate generation of key material, and submission of the public key as part of an HTML form. This mechanism is designed for use with Web-based certificate management systems. It is expected that the <keygen> element will be used in an HTML form along with other information needed to construct a certificate request, and that the result of the process will be a signed certificate.
<label>	The *HTML Label Element* (**<label>**) represents a caption for an item in a user interface. It can be associated with a control either by placing the control element inside the <label> element, or by using the for attribute. Such a control is called the *labeled control* of the label element. With one input can be assosiated few labels.
<legend>	The **HTML <legend> Element** (or *HTML Legend Field Element*) represents a caption for the content of its parent <fieldset>.

Element	Description
<meter>	The **HTML <meter> Element** represents either a scalar value within a known range or a fractional value.
<optgroup>	In a Web form, the **HTML <optgroup> element** creates a grouping of options within a <select> element.
<option>	In a Web form, the **HTML <option> element** is used to create a control representing an item within a <select>, an <optgroup> or a <datalist> HTML5 element.
<output>	The **HTML <output> element** represents the result of a calculation or user action.
<progress>	The **HTML <progress> Element** is used to view the completion progress of a task. While the specifics of how it's displayed is left up to the browser developer, it's typically displayed as a progress bar. Javascript can be used to manipulate the value of progress bar.
<select>	The HTML *select* (<select>) element represents a control that presents a menu of options. The options within the menu are represented by <option> elements, which can be grouped by <optgroup> elements. Options can be pre-selected for the user.

```
1. <button type="button" autofocus disabled onclick="alert('Hello
world!')">Click Me!</button>

2. <form action="demo_form.asp" method="get" id="nameform">

     First name: <input type="text" name="fname"><br>

     Last name: <input type="text" name="lname"><br>

   </form>

<p>button below is outside form element, but still part of form.</p>

<button type="submit" form="nameform" value="Submit">Submit</button>

3.  <form action="demo_form.asp" method="get">

      Choose your favorite subject:

    <button name="subject" type="submit" value="HTML">HTML</button>
</form>

4.  <button type="submit" value="Submit">Submit</button>

    <button type="reset" value="Reset">Reset</button>
```

```
<input list="browsers"> <datalist id="browsers">
                          <option value="Internet Explorer">
                          <option value="Firefox">
                           <option value="Chrome"> </datalist>
<form>  <fieldset>
             <legend>Personalia:</legend>
              Name: <input type="text"><br>
              Email: <input type="text"><br> </fieldset> </form>
<form action="demo_keygen.asp" method="get">
              Username: <input type="text" name="usr_name">
              Encryption: <keygen name="security">
              <input type="submit"> </form>
<form action="demo_form.asp">
             <label for="male">Male</label>
             <input type="radio" name="sex" id="male" value="male"><br>
             <input type="submit" value="Submit"> </form>
<form> <fieldset> <legend>Personalia:</legend>
                          Name: <input type="text"><br>
                          Email: <input type="text"><br> </fieldset> </form>
<p>Display a gauge:</p>
      <meter value="2" min="0" max="10">2 out of 10</meter><br>
      <meter value="0.6">60%</meter></p>
<select> <optgroup label="Swedish Cars">
             <option value="volvo">Volvo</option>
             <option value="saab">Saab</option> </optgroup>
         <optgroup label="German Cars">
             <option value="mercedes">Mercedes</option>
             <option value="audi">Audi</option> </optgroup> </select>
<select> <option value="1">1</option> <option value="2">2</option></select>
<form  oninput="x.value=parseInt(a.value)+parseInt(b.value)">0
      <input type="range" id="a" value="50">100 +
      <input type="number" id="b" value="50">=<output name="x" for="a
               b"></output> </form>
Downloading progress:<progress value="22" max="100"></progress>
```

11. Interactive elements

HTML offers a selection of elements which help to create interactive user interface objects.

Element	Description
<details>	The *HTML Details Element* (**<details>**) is used as a disclosure widget from which the user can retrieve additional information.
<dialog>	The **HTML <dialog> element** represents a dialog box or other interactive component, such as an inspector or window. <form> elements can be integrated within a dialog by specifying them with the attribute method="dialog". When such a form is submitted, the dialog is closed with a returnValue attribute set to the value of the submit button used.
<menu>	The **HTML <menu> element** represents a group of commands that a user can perform or activate. This includes both list menus, which might appear across the top of a screen, as well as context menus, such as those that might appear underneath a button after it has been clicked.
<menuitem>	The **HTML <menuitem> element** represents a command that a user is able to invoke through a popup menu. This includes context menus, as well as menus that might be attached to a menu button.
<summary>	The HTML *summary* element (<summary>) is used as a summary, caption, or legend for the content of a <details> element.

```
<details>
  <summary>Copyright 1999-2014.</summary>
  <p> - by Refsnes Data. All Rights Reserved.</p>
  <p>All content and graphics on this web site are the property of the company</p>
</details>

<dialog open>
  <p>Greetings, one and all!</p>
</dialog>

<!-- A button, which displays a menu when clicked. -->
<button type="menu" menu="dropdown-menu"> Dropdown </button>
       <menu type="context" id="dropdown-menu">
              <menuitem label="Action">
              <menuitem label="Another action"> <hr>
              <menuitem label="Separated action">
       </menu>

<details>
  <summary>Some details</summary>
  <p>More info about the details.</p> </details>
```

12. Web Components

Web Components is an HTML-related technology which makes it possible to, essentially, create and use custom elements as if it were regular HTML. In addition, you can even create custom versions of standard HTML elements, as well.

Note: The elements for Web Components are are defined in the World Wide Web Consortium (W3C) Web Components collection of specifications rather than the HTML specification. In addition, the Web Components specification has not been finalized and is subject to change.

Element	Description
<content>	The **HTML <content>** element is used inside of Shadow DOM as an insertion point. It is not intended to be used in ordinary HTML. It is used with Web Components.
<decorator>	
<element>	The **HTML <element> element** is used to define new custom DOM elements.
<shadow>	The **HTML <shadow>** element is used as a shadow DOM insertion point. You might use it if you have created multiple shadow roots under a shadow host. It is not useful in ordinary HTML. It is used with Web Components.
<template>	The **HTML template element <template>** is a mechanism for holding client-side content that is not to be rendered when a page is loaded but may subsequently be instantiated during runtime using JavaScript.

HTML ATTRIBUTES

- Attributes provide additional information about an element
- Attributes are always specified in the start tag
- Attributes come in name/value pairs like: name="value"

Example 1:

Example 2: <div id>

Example 3: <div id = ‘ ’ >

Attribute Name	Elements	Description
accept	<form>, <input>	List of types the server accepts, typically a file type.
accept-charset	<form>	List of supported charsets.
accesskey	Global attribute	Defines a keyboard shortcut to activate or add focus to the element.
action	<form>	The URI of a program that processes the information submitted via the form.
align	<applet>, <caption>, <col>, <colgroup>, <hr>, <iframe>, <img>, <table>, <tbody>, <td>, <tfoot> , <th>, <thead>, <tr>	Specifies the horizontal alignment of the element.
Alt	<applet>, <area>, <img>, <input>	Alternative text in case an image can't be displayed.
async	<script>	Indicates that the script should be executed asynchronously.
autocomplete	<form>, <input>	Indicates whether controls in this form can by default have their

Attribute Name	Elements	Description
		values automatically completed by the browser.
autofocus	<button>, <input>, <keygen>, <select>, <textarea>	The element should be automatically focused after the page loaded.
autoplay	<audio>, <video>	The audio or video should play as soon as possible.
autosave	<input>	Previous values should persist dropdowns of selectable values across page loads.
bgcolor	<body>, <col>, <colgroup>, <marquee>, <table>, <tbody>, <tfoot>, <td>, <th>, <tr>	Background color of the element. **Note:** This is a legacy attribute. Please use the CSS background-color property instead.
border	<img>, <object>, <table>	The border width. **Note:** This is a legacy attribute. Please use the CSS border property instead.
buffered	<audio>, <video>	Contains the time range of already buffered media.
challenge	<keygen>	A challenge string that is submitted along with the public key.
charset	<meta>, <script>	Declares the character encoding of the page or script.
checked	<command>, <input>	Indicates whether the element should be checked on page load.
cite	<blockquote>, <del>, <ins>, <q>	Contains a URI which points to the source of the quote or change.

Attribute Name	Elements	Description
class	Global attribute	Often used with CSS to style elements with common properties.
code	<applet>	Specifies the URL of the applet's class file to be loaded and executed.
codebase	<applet>	This attribute gives the absolute or relative URL of the directory where applets' .class files referenced by the code attribute are stored.
color	<basefont>, <font>, <hr>	This attribute sets the text color using either a named color or a color specified in the hexadecimal #RRGGBB format. **Note:** This is a legacy attribute. Please use the CSS color property instead.
cols	<textarea>	Defines the number of columns in a textarea.
colspan	<td>, <th>	The colspan attribute defines the number of columns a cell should span.
content	<meta>	A value associated with http-equiv or name depending on the context.
contenteditable	Global attribute	Indicates whether the element's content is editable.
contextmenu	Global attribute	Defines the ID of a <menu> element which will serve as the element's context menu.
controls	<audio>, <video>	Indicates whether the browser should show playback controls to the user.

Attribute Name	Elements	Description
coords	<area>	A set of values specifying the coordinates of the hot-spot region.
data	<object>	Specifies the URL of the resource.
data-*	Global attribute	Lets you attach custom attributes to an HTML element.
datetime	<del>, <ins>, <time>	Indicates the date and time associated with the element.
default	<track>	Indicates that the track should be enabled unless the user's preferences indicate something different.
defer	<script>	Indicates that the script should be executed after the page has been parsed.
dir	Global attribute	Defines the text direction. Allowed values are ltr (Left-To-Right) or rtl (Right-To-Left)
dirname	<input>, <textarea>	
disabled	<button>, <command>, <fieldset>, <input>, <keygen>, <optgroup>, <option>, <select>, <textarea>	Indicates whether the user can interact with the element.
download	<a>, <area>	Indicates that the hyperlink is to be used for downloading a resource.
draggable	Global attribute	Defines whether the element can be dragged.
dropzone	Global attribute	Indicates that the element accept the dropping of content on it.
enctype	<form>	Defines the content type of the form date when the method is POST.

Attribute Name	Elements	Description
for	<label>, <output>	Describes elements which belongs to this one.
form	<button>, <fieldset>, <input>, <keygen>, <label>, <meter>, <object>, <output>, <progress>, <select>, <textarea>	Indicates the form that is the owner of the element.
formaction	<input>, <button>	Indicates the action of the element, overriding the action defined in the <form>.
headers	<td>, <th>	IDs of the <th> elements which applies to this element.
height	<canvas>, <embed>, <iframe>, <img>, <input>, <object>, <video>	Specifies the height of elements listed here. For all other elements, use the CSS height property. **Note:** In some instances, such as , this is a legacy attribute, in which case the CSS height property should be used instead.
hidden	Global attribute	Prevents rendering of given element, while keeping child elements, e.g. script elements, active.
high	<meter>	Indicates the lower bound of the upper range.
href	<a>, <area>, <base>, <link>	The URL of a linked resource.
hreflang	<a>, <area>, <link>	Specifies the language of the linked resource.
http-equiv	<meta>	
icon	<command>	Specifies a picture which represents the command.
id	Global attribute	Often used with CSS to style a specific element. The value of this attribute must be unique.

Attribute Name	Elements	Description
ismap	<img>	Indicates that the image is part of a server-side image map.
itemprop	Global attribute	
keytype	<keygen>	Specifies the type of key generated.
kind	<track>	Specifies the kind of text track.
label	<track>	Specifies a user-readable title of the text track.
lang	Global attribute	Defines the language used in the element.
language	<script>	Defines the script language used in the element.
list	<input>	Identifies a list of pre-defined options to suggest to the user.
loop	<audio>, <bgsound>, <marquee>, <video>	Indicates whether the media should start playing from the start when it's finished.
low	<meter>	Indicates the upper bound of the lower range.
manifest	<html>	Specifies the URL of the document's cache manifest.
max	<input>, <meter>, <progress>	Indicates the maximum value allowed.
maxlength	<input>, <textarea>	Defines the maximum number of characters allowed in the element.
media	<a>, <area>, <link>, <source>, <style>	Specifies a hint of the media for which the linked resource was designed.

Attribute Name	Elements	Description
method	<form>	Defines which HTTP method to use when submitting the form. Can be GET (default) or POST.
min	<input>, <meter>	Indicates the minimum value allowed.
multiple	<input>, <select>	Indicates whether multiple values can be entered in an input of the type email or file.
name	<button>, <form>, <fieldset>, <iframe>, <input>, <keygen>, <object>, <output>, <select>, <textarea>, <map>, <meta>, <param>	Name of the element. For example used by the server to identify the fields in form submits.
novalidate	<form>	This attribute indicates that the form shouldn't be validated when submitted.
open	<details>	Indicates whether the details will be shown on page load.
optimum	<meter>	Indicates the optimal numeric value.
pattern	<input>	Defines a regular expression which the element's value will be validated against.
ping	<a>, <area>	
placeholder	<input>, <textarea>	Provides a hint to the user of what can be entered in the field.
poster	<video>	A URL indicating a poster frame to show until the user plays or seeks.
preload	<audio>, <video>	Indicates whether the whole resource, parts of it or nothing should be preloaded.
radiogroup	<command>	

Attribute Name	Elements	Description
readonly	<input>, <textarea>	Indicates whether the element can be edited.
rel	<a>, <area>, <link>	Specifies the relationship of the target object to the link object.
required	<input>, <select>, <textarea>	Indicates whether this element is required to fill out or not.
reversed		Indicates whether the list should be displayed in a descending order instead of a ascending.
rows	<textarea>	Defines the number of rows in a text area.
rowspan	<td>, <th>	Defines the number of rows a table cell should span over.
sandbox	<iframe>	
scope	<th>	
scoped	<style>	
seamless	<iframe>	
selected	<option>	Defines a value which will be selected on page load.
shape	<a>, <area>	
size	<input>, <select>	Defines the width of the element (in pixels). If the element's type attribute is text or password then it's the number of characters.
sizes	<link>, <img>, <source>	
span	<col>, <colgroup>	
spellcheck	Global attribute	Indicates whether spell checking is allowed for the element.

Attribute Name	Elements	Description
src	<audio>, <embed>, <iframe>, <img>, <input>, <script>, <source>, <track>, <video>	The URL of the embeddable content.
srcdoc	<iframe>	
srclang	<track>	
srcset	<img>	
start		Defines the first number if other than 1.
step	<input>	
style	Global attribute	Defines CSS styles which will override styles previously set.
summary	<table>	
tabindex	Global attribute	Overrides the browser's default tab order and follows the one specified instead.
target	<a>, <area>, <base>, <form>	
title	Global attribute	Text to be displayed in a tooltip when hovering over the element.
type	<button>, <input>, <command>, <embed>, <object>, <script>, <source>, <style>, <menu>	Defines the type of the element.
usemap	<img>, <input>, <object>	
value	<button>, <option>, <input>, , <meter>, <progress>, <param>	Defines a default value which will be displayed in the element on page load.

Attribute Name	Elements	Description
width	<canvas>, <embed>, <iframe>, <img>, <input>, <object>, <video>	For the elements listed here, this establishes the element's width. **Note:** For all other instances, such as , this is a legacy attribute, in which case the CSS width property should be used instead.
wrap	<textarea>	Indicates whether the text should be wrapped.

GLOBAL ATTRIBUTES

Global attributes are attributes common to all HTML elements; they can be used on all elements, though the attributes may have no effect on some elements.

Global attributes may be specified on all HTML elements, *even those not specified in the standard*. That means that any non-standard elements must still permit these attributes, even though using those elements means that the document is no longer HTML5-compliant. For example, HTML5-compliant browsers hide content marked as <foo hidden>...<foo>, even though <foo> is not a valid HTML element.

In addition to the basic HTML global attributes, the following global attributes also exist:

- **xml:lang** and **xml:base** — these are inherited from the XHTML specifications and deprecated, but kept for compatibility purposes.
- The multiple **aria-*** attributes, used for improving accessibility.
- The event handler attributes: **onabort**, **onautocomplete**, **onautocompleteerror**, **onblur**, **oncancel**, **oncanplay**, **oncanplaythrough**, **onchange**, **onclick**, **onclose**, **oncontextmenu**, **oncuechange**, **ondblclick**, **ondrag**, **ondragend**, **ondragenter**, **ondragexit**, **ondragleave**, **ondragover**, **ondragstart**, **ondrop**, **ondurationchange**, **onemptied**, **onended**, **onerror**, **onfocus**, **oninput**, **oninvalid**, **onkeydown**, **onkeypress**, **onkeyup**, **onload**, **onloadeddata**, **onloadedmetadata**, **onloadstart**, **onmousedown**, **onmouseenter**, **onmouseleave**, **onmousemove**, **onmouseout**, **onmouseover**, **onmouseup**, **onmousewheel**, **onpause**, **onplay**, **onplaying**, **onprogress**, **onratechange**, **onreset**, **onresize**, **onscroll**, **onseeked**, **onseeking**, **onselect**, **onshow**, **onsort**, **onstalled**, **onsubmit**, **onsuspend**, **ontimeupdate**, **ontoggle**, **onvolumechange**, **onwaiting**.

Description

accesskey

Provides a hint for generating a keyboard shortcut for the current element. This attribute consists of a space-separated list of characters. The browser should use the first one that exists on the computer keyboard layout.

class

Is a space-separated list of the classes of the element. Classes allows CSS and JavaScript to select and access specific elements via the class selectors or functions like the method Document.getElementsByClassName().

contenteditable

Is an enumerated attribute indicating if the element should be editable by the user. If so, the browser modifies its widget to allow editing. The attribute must take one of the following values:

- true or the *empty string*, which indicates that the element must be editable;

- false, which indicates that the element must not be editable.

contextmenu

Is the **id** of an <menu> to use as the contextual menu for this element.

data-*

Forms a class of attributes, called custom data attributes, that allow proprietary information to be exchanged between the HTML and its DOM representation that may be used by scripts. All such custom data are available via the HTMLElement interface of the element the attribute is set on. The HTMLElement.dataset property gives access to them.

dir

Is an enumerated attribute indicating the directionality of the element's text. It can have the following values:

- ltr, which means *left to right* and is to be used for languages that are written from the left to the right (like English);
- rtl, which means *right to left* and is to be used for languages that are written from the right to the left (like Arabic);
- auto, which let the user agent decides. It uses a basic algorithm as it parses the characters inside the element until it finds a character with a strong directionality, then apply that directionality to the whole element.

draggable

Is an enumerated attribute indicating whether the element can be dragged, using the Drag and Drop API. It can have the following values:

- true, which indicates that the element may be dragged
- false, which indicates that the element may not be dragged.

dropzone

Is an enumerated attribute indicating what types of content can be dropped on an element, using the Drag and Drop API. It can have the following values:

- copy, which indicates that dropping will create a copy of the element that was dragged
- move, which indicates that the element that was dragged will be moved to this new location.
- link, will create a link to the dragged data.

hidden

Is a Boolean attribute indicates that the element is not yet, or is no longer, *relevant*. For example, it can be used to hide elements of the page that can't be used until the login

process has been completed. The browser won't render such elements. This attribute must not be used to hide content that could legitimately be shown.

id

Defines a unique identifier (ID) which must be unique in the whole document. Its purpose is to identify the element when linking (using a fragment identifier), scripting, or styling (with CSS).

itemid

itemprop

itemref

itemscope

itemtype

These attributes are related to the WHATWG HTML Microdata feature.

lang

Participates in defining the language of the element, the language that non-editable elements are written in or the language that editable elements should be written in. The tag contains one single entry value in the format defines in the *Tags for Identifying Languages (BCP47)* IETF document. **xml:lang** has priority over it.

spellcheck

Is an enumerated attribute defines whether the element may be checked for spelling errors. It may have the following values:

- true, which indicates that the element should be, if possible, checked for spelling errors;
- false, which indicates that the element should not be checked for spelling errors.

style

Contains CSS styling declarations to be applied to the element. Note that it is recommended for styles to be defined in a separate file or files. This attribute and the <style> element have mainly the purpose of allowing for quick styling, for example for testing purposes.

tabindex

Is an integer attribute indicates if the element can take input focus (is *focusable*), if it should participate to sequential keyboard navigation, and if so, at what position. It can takes several values:

- a *negative value* means that the element should be focusable, but should not be reachable via sequential keyboard navigation;
- 0 means that the element should be focusable and reachable via sequential keyboard navigation, but its relative order is defined by the platform convention;
- a *positive value* which means should be focusable and reachable via sequential keyboard navigation; its relative order is defined by the value of the attribute: the sequential follow the increasing number of the **tabindex**. If several elements share the same tabindex, their relative order follows their relative position in the document).

title

Contains a text representing advisory information related to the element it belongs to. Such information can typically, but not necessarily, be presented to the user as a tooltip.

translate

Is an enumerated attribute that is used to specify whether an element's attribute values and the values of its Text node children are to be translated when the page is localized, or whether to leave them unchanged. It can have the following values:

- empty string and "yes", which indicates that the element will be translated.
- "no", which indicates that the element will not be translated.

Link Types

In HTML, the following link types indicate the relationship between two documents, in which one links to the other using an <a>, <area>, or <link> element.

List of the defined link types and their significance in HTML			
Link Type	**Description**	**Allowed in these elements**	**Not allowed in these elements**

alternate	• If the element is <link> and the rel attribute also contains the stylesheet type, the link defines an alternative style sheet; in that case the title attribute must be present and not be the empty string.
	• If the type is set to application/rss+xml or application/atom+xml, the link defines a syndication feed. The first one defined on the page is the default.
	• Otherwise, the link defines an alternative page, of one of these types:
	  ○ for another medium, like a handheld device (if the media attribute is set)
	  ○ in another language (if the hreflang attribute is set),
	  ○ in another format, such as a PDF (if the type attribute is set)
	  ○ a combination of these
		<a>, <area>, <link>	*None.*

archives	Defines a hyperlink to a document that contains an archive link to this one. For example, a blog entry could link to a monthly index page this way. **Note:** Although recognized, the singular archive is incorrect and must be avoided.	<a>, <area>, <link>	*None.*
author	Defines a hyperlink to a page describing the author or providing a way to contact the author. **Note:** This may be a mailto: hyperlink, but this is not recommended on public pages as robot harvesters will quickly lead to a lot of spam sent to the address. In that case, it is better to lead to a page containing a contact	<a>, <area>, <link>	*None.*

List of the defined link types and their significance in HTML			
Link Type	**Description**	**Allowed in these elements**	**Not allowed in these elements**
	form. Although recognized, the rev attribute on <a>, <area> or<link> elements with a link type of made is incorrect and should be replaced by the rel attribute with this link type.		
bookmark	Indicates that the hyperlink is a permalink for the nearest ancestor <article> element. If none, it is a permalink for the section that the element is most closely associated to. This allows for bookmarking a single article in a page containing multiple articles, such as on a monthly summary blog page, or a blog aggregator.	<a>, <area>	<link>
dns-prefetch	Hints to the browser that a resources is needed allowing the browser to do a DNS lookup and protocol handshaking before a user clicks the link.	<link>	<a>, <area>
external	Indicates that the hyperlink leads to a resource outside the site of the current page; that is, following the link will make the user leave the site.	<a>, <area>	<link>
first	Indicates that the hyperlink leads to the first resource of the *sequence* the current page is in. **Note:** Other link types related to linking resources in the same sequence are last, prev, next. Although recognized, the synomyns begin and start are incorrect and must be avoided.	<a>, <area>, <link>	*None.*
help	• If the element is <a> or <area>, it indicates that the hyperlink leads to a resource giving further help about the parent of the element, and its descendants.	<a>, <area>, <link>	*None.*

List of the defined link types and their significance in HTML			
Link Type	**Description**	**Allowed in these elements**	**Not allowed in these elements**
	• If the element is <link> it indicates that the hyperlink leads to a resource giving further help about the page as a whole.		
icon	Defines a resource for representing the page in the user interface, usually an icon (auditory or visual). The media, type and sizes attributes allow the browser to select the most appropriate icon for its context. If several resources match, the browser will select the last one declared, in tree order. As these attributes are merely hints, and the resources may be inappropriate upon further inspection, the browser will then select another one, if appropriate. **Note:** Apple's iOS does not use this link type, nor the sizes attribute, like others mobile browsers do, to select a webpage icon for Web Clip or a start-up placeholder. Instead it uses the non-standard apple-touch-icon and apple-touch-startup-image respectively. The shortcut link type is often seen before icon, but this link type is non-conforming, ignored and **web authors must not use it anymore**.	<link>	<a>, <area>
index	Indicates that the page is part of a *hierarchical* structure and that the hyperlink leads to the top level resource of that structure. If one or several up link types are also present, the number of these up indicates the depth of the current page in the hierarchy.	<a>, <area>, <link>	*None.*
last	Indicates that the hyperlink leads to the *last* resource of the *sequence* the current page is in.	<a>, <area>, <link>	*None.*

List of the defined link types and their significance in HTML			
Link Type	**Description**	**Allowed in these elements**	**Not allowed in these elements**
	Note: Other link types related to linking resources in the same sequence are first, prev, next. Although recognized, the synomyn end is incorrect and must be avoided.		
license	Indicates that the hyperlink leads to a document describing the licensing information. If not inside the <head> element, the standard doesn't distinguish between a hyperlink applying to a specific part of the document or to the document as a whole. Only the data on the page can indicate this. **Note:** Although recognized, the synonym copyright is incorrect and must be avoided.	<a>, <area>, <link>	*None.*
next	Indicates that the hyperlink leads to the *next* resource of the *sequence* the current page is in. **Note:** Other link types related to linking resources in the same sequence are first, prev, last.	<a>, <area>, <link>	*None.*
nofollow	Indicates that the linked document is not endorsed by the author of this one, for example if it has no control over it, if it is a bad example or if there is commercial relationship between the two (sold link). This link type may be used by some search engines that use popularity ranking techniques.	<a>, <area>	<link>
noreferrer	Prevents the browser, when navigating to another page, to send this page name, or any other value, as referrer via the Referer: HTTP header. (In Firefox, before Firefox 37, this worked only in links found in pages. Links clicked in the UI, like "Open in a new tab" via the contextual menu, doesn't abide for this value)	<a>, <area>	<link>

List of the defined link types and their significance in HTML			
Link Type	**Description**	**Allowed in these elements**	**Not allowed in these elements**
pingback	Defines an external resource URI to call if one make a comment or a citation about the webpage. The protocol used to make such a call is defined in the Pingback 1.0 specification. **Note:** if the X-Pingback: HTTP header is also present, this header has precedence over the <link> element with this link type	<link>	<a>, <area>
preconnect	Hints the browser to open in advance the connection to the linked web site, without disclosing any private information.	<link>	<a>, <area>
prefetch	Hints the browser to fetch in advance the linked resource, as it will likely be requested by the user. **Note:** the Link Prefetch FAQ has details on which links can be prefetched and on alternative methods.	<a> Unimplemented, <area> Unimplemented, <link>	*None.*
preload	Tells the browser to download a resource because this resource will be needed later during the current navigation.	<link>	<a>, <area>
prerender		<link>	<a>, <area>
prev	Indicates that the hyperlink leads to the *preceding* resource of the *sequence* the current page is in. **Note:** other link types related to linking resources in the same sequence are first, last, next. Although recognized, the synomyn previous is incorrect and must be avoided.	<a>, <area>, <link>	*None.*
search	Indicates that the hyperlink reference a document whose interface is specially designing for searching in	<a>, <area>, <link>	*None.*

List of the defined link types and their significance in HTML			
Link Type	**Description**	**Allowed in these elements**	**Not allowed in these elements**
	this document, or site, and its resources. If the type attribute is set to application/opensearchdescription+xml the resource is an OpenSearch plugin that can be easily added to the interface of some browsers like Firefox or Internet Explorer.		
stylesheet	Defines an external resource to be used as a stylesheet. If the type is not set, the browser should assume it is a text/css stylesheet until further inspection. If used in combination with the alternate keyword, it defines an alternative style sheet; in that case the title attribute must be present and not be the empty string.	<link>	<a>, <area>
sidebar	Indicates that the hyperlink leads to a resource that would be better suited for a secondary browsing context, like a *sidebar*. Browsers, that don't have such a context will ignore this keyword.	<a>, <area>, <link>	*None.*
tag	Indicates that the hyperlink refers to a document describing a *tag* that applies to this document. **Note:** this link type should not be set on links to a member of a tag cloud as these do not apply to a single document but to a set of pages.	<a>, <area>	<link>
up	Indicates that the page is part of a *hierarchical* structure and that the hyperlink leads to the higher level resource of that structure. The number of uplink types indicates the depth difference between the current page and the linked resource.	<a>, <area>, <link>	*None.*

Tips for Authoring Fast Loading HTML Page

Reduce page weight

Page weight is by far the most important factor in page-load performance.

Reducing page weight through the elimination of unnecessary whitespace and comments, commonly known as minimization, and by moving inline script and CSS into external files, can improve download performance with minimal need for other changes in the page structure.

Tools such as HTML Tidy can automatically strip leading whitespace and extra blank lines from valid HTML source. Other tools can "compress" JavaScript by reformatting the source or by obfuscating the source and replacing long identifiers with shorter versions.

Minimize the number of files

Reducing the number of files referenced in a web page lowers the number of HTTP connections required to download a page.

Depending on a browser's cache settings, it may send an If-Modified-Since request to the web server for each CSS, JavaScript or image file, asking whether the file has been modified since the last time it was downloaded.

By reducing the number of files that are referenced within a web page, you reduce the time required for these requests to be sent, and for their responses to be received.

If you use background images a lot in your CSS, you can reduce the amount of HTTP lookups needed by combining the images into one, known as an image sprite. Then you just apply the same image each time you need it for a background, and adjust the x/y coordinates appropriately. This technique works best with elements that will have limited dimensions, and will not work for every use of a background image. However, the fewer HTTP requests and single image caching can help reduce page-load time.

Too much time spent querying the last modified time of referenced files can delay the initial display of a web page, since the browser must check the modification time for each CSS or JavaScript file, before rendering the page.

Reduce domain lookups

Since each separate domain costs time in a DNS lookup, the page load time will grow along with the number of separate domains appearing in CSS link(s) and JavaScript and image src(es).

This may not always be practical; however, you should always take care to use only the minimum necessary number of different domains in your pages.

Cache reused content

Make sure that any content that can be cached, is cached, and with appropriate expiration times.

In particular, pay attention to the Last-Modified header. It allows for efficient page caching; by means of this header, information is conveyed to the user agent about the file it wants to load, such as when it was last modified. Most web servers automatically append the Last-Modified header to static pages (e.g. .html, .css), based on the last-modified date stored in the file system. With dynamic pages (e.g. .php, .aspx), this, of course, can't be done, and the header is not sent.

So, in particular for pages which are generated dynamically, a little research on this subject is beneficial. It can be somewhat involved, but it will save a lot in page requests on pages which would normally not be cacheable.

More information:

1. HTTP Conditional Get for RSS Hackers
2. HTTP 304: Not Modified
3. HTTP ETag on Wikipedia
4. Caching in HTTP

Optimally order the components of the page

Download page content first, along with any CSS or JavaScript that may be required for its initial display, so that the user gets the quickest apparent response during the page loading. This content is typically text, and can therefore benefit from text compression in transit, thus providing an even quicker response to the user.

Any dynamic features that require the page to complete loading before being used, should be initially disabled, and then only enabled after the page has loaded. This will cause the JavaScript to be loaded after the page contents, which will improve the overall appearance of the page load.

Reduce the number of inline scripts

Inline scripts can be expensive for page loading, since the parser must assume that an inline script could modify the page structure while parsing is in progress. Reducing the use of inline scripts in general, and reducing the use of document.write() to output content in particular, can improve overall page loading. Move the inline scripts away from HTML and CSS. Use modern AJAX methods to manipulate page content for modern browsers, rather than the older approaches based on document.write().

Use modern CSS and valid markup

Use of modern CSS reduces the amount of markup, can reduce the need for (spacer) images, in terms of layout, and can very often replace images of stylized text -- that "cost" much more than the equivalent text-and-CSS.

Using valid markup has other advantages. First, browsers will have no need to perform error-correction when parsing the HTML (this is aside from the philosophical issue of whether to allow format variation in user input, and then programmatically "correct" or normalize it; or whether, instead, to enforce a strict, no-tolerance input format).

Moreover, valid markup allows for the free use of other tools which can *pre-process* your web pages. For example, HTML Tidy can remove whitespace and optional ending tags; however, it will refuse to run on a page with serious markup errors.

Chunk your content

Tables for layouts are a legacy method that should not be used any more. Layouts utilizing blocks, and in the near future, CSS3 Multi-column Layout or CSS3 Flexible Box Layout, should be used instead.

Tables are still considered valid markup, but should be used for displaying tabular data. To help the browser render your page quicker, you should avoid nesting your tables.

Rather than deeply nesting tables as in:

```
<TABLE>
  <TABLE>
    <TABLE>
        ...
    </TABLE>
  </TABLE>
</TABLE>
```

use non-nested tables or divs as in

```
<TABLE>...</TABLE>
<TABLE>...</TABLE>
<TABLE>...</TABLE>
```

See also: CSS3 Multi-column Layout Spec and CSS3 Flexible Box Layout

Minify and compress SVG assets

SVG produced by most drawing applications often contains unnecessary metadata which can be removed. Configure your servers apply gzip compression for SVG assets.

Specify sizes for images and tables

If the browser can immediately determine the height and/or width of your images and tables, it will be able to display a web page without having to reflow the content. This not only speeds the

display of the page but prevents annoying changes in a page's layout when the page completes loading. For this reason, height and width should be specified for images, whenever possible.

Tables should use the CSS selector: property combination:-

Example: table-layout: fixed;

and should specify widths of columns using the COL and COLGROUP HTML tags.

Choose your user-agent requirements wisely

To achieve the greatest improvements in page design, make sure that reasonable user-agent requirements are specified for projects. Do not require your content to appear pixel-perfect in all browsers, especially not in down-version browsers.

Ideally, your basic minimum requirements should be based on the consideration of modern browsers that support the relevant standards. This can include recent versions of Firefox, Internet Explorer, Google Chrome, Opera, and Safari.

Note, however, that many of the tips listed in this article are common-sense techniques which apply to any user agent, and can be applied to any web page, regardless of browser-support requirements.

Example page structure

· HTML

· HEAD

· LINK ...
CSS files required for page appearance. Minimize the number of files for performance while keeping unrelated CSS in separate files for maintenance.

· SCRIPT ...
JavaScript files for functions **required** during the loading of the page, but not any DHTML that can only run after page loads.

Minimize the number of files for performance while keeping unrelated JavaScript in separate files for maintenance.

· BODY

· User visible page content in small chunks (tables / divs) that can be displayed without waiting for the full page to download.

· SCRIPT ...
Any scripts which will be used to perform DHTML. DHTML script typically can only run after the page has completely loaded and all necessary objects have been initialized.

There is no need to load these scripts before the page content. That only slows down the initial appearance of the page load.

Minimize the number of files for performance while keeping unrelated JavaScript in separate files for maintenance.

If any images are used for rollover effects, you should preload them here after the page content has downloaded.

Use async and defer, if possible

Make the JavaScript scripts such that they are compatible with both the async and the defer and use async whenever possible, specially if you have multiple script tags.
With that, the page can stop rendering while JavaScript is still loading. Otherwise the browser will not render anything that is after the script tags that do not have these attributes.

Note: Even though these attributes do help a lot for the first time a page is loaded, you should use them but not rely that it will work in all browsers. If you follow all guidelines to make good JavaScript code, there is no need to change your code.

HIGHLIGHTS:

- **HTML – home plan (where should be kitchen, room, etc., its size)**
- **CSS – color of carpet, curtains, etc.**
- **JavaScript – interactive elements such as garage door opener, TV remote.**
- **Never determine screen size in pixel (Ex: 200px) instead define in terms of % (Ex: 10%)**
- **Responsive screen- changes screen size when screen width reduced.**
- **Divide everything into boxes, then code, then test and compare it with actual design than fix the small changes.**
- **Use 12 column grid size: coz 12 can be easily divided to 2, 3, 4 evenly sized columns.**
- **Row of 100% page width + columns 1/12-12/12 page width**
  .i.e. 8.33%, 16.66%, … 100%.
- **placeholder.it/100/30 , placepuppy.it/100/30 , placekitten.com/100/30 ,**
  www.google.com/fonts.
- **Bootstrap, foundation.zurb, yaml.de, 960.gs , susy.oddbird, framelessgrid,**
  frameworks for HTML, CSS, JavaScript.