Name: Kristiyan Milanov Petsanov

Faculty: FCST

Fac. No.: 121224086

Title: **How HTML5 Works?**

Author and

Publishing house: [Stephanie Crawford](https://www.howstuffworks.com/about-author.htm#scrawford) / [How Stuff Works?](https://computer.howstuffworks.com/html-five.htm)

**TEXT**

**How HTML5 Works?**

Table of Contents

[I. Introduction 2](#_Toc183590294)

[II. HTML5 Goes Beyond HTML 3](#_Toc183590295)

[III. HTML5 Foundation Technologies 3](#_Toc183590296)

[IV. Basic HTML Content 4](#_Toc183590297)

[V. Forms 5](#_Toc183590298)

[VI. Semantic Elements 6](#_Toc183590299)

[VII. Media Elements and the Flash Rivalry 7](#_Toc183590300)

[VIII. The Canvas 8](#_Toc183590301)

[IX. Creating the Look with CSS3 10](#_Toc183590302)

[X. JavaScript as a Central Component 11](#_Toc183590303)

[XI. Web Application Technologies 11](#_Toc183590304)

[XII. Groundbreaking Proofs of Concept 12](#_Toc183590305)

How HTML works

# Introduction

Hypertext Markup Language (HTML) has been a core technology for the web since the early 1990s. Tim Berners-Lee created HTML in 1989 as a simple but effective way to encode electronic documents. In fact, the original purpose of a web browser was to serve as a reader for such documents. Two decades later, the browser itself has become a portal to a world of online media. That's why HTML5 isn't just another HTML revision, but a comprehensive standard for [how web pages work](https://computer.howstuffworks.com/web-page.htm).

To better understand what makes HTML5 unique, let's turn the clock back a bit. Back in 1994, HTML was still in its first revision, Mosaic and Netscape dominated the browser market, and most people had yet to experience this new thing called the World Wide Web. That year, HTML creator Berners-Lee headed a newly established web standards group known as the World Wide Web Consortium (W3C).

Though W3C is a respected standards authority today, the commercial players in the 1990s browser market largely ignored those standards and blazed their own paths. By 1995, W3C had published the second revision of the HTML standard, and web newcomer [Microsoft](https://computer.howstuffworks.com/microsoft.htm) was gaining ground with its Internet Explorer (IE) browser. Microsoft largely ignored standards, and Netscape struggled to maintain a respectable market share while IE began to dominate.

During these early browser wars, web developers were challenged to keep their sites compatible with each new release of the major browsers as well as with the lesser-used Opera and Apple Safari browsers. Even though W3C had published HTML 3.2 in 1997, followed by HTML 4 in 1998, following the standards seemed less important than keeping up with browser-specific features. This went on until 2003 when the community-driven Mozilla Foundation broke the trend. After its original Mozilla browser release, followed by its [Firefox](https://computer.howstuffworks.com/internet/basics/firefox.htm) browser in 2004, the Mozilla quickly hacked away at IE's dominance. In addition, these new browsers actually followed existing W3C standards while doing so.

While Mozilla's Firefox continued growing using the aging HTML 4 standard, Mozilla joined Apple and Opera in 2004 to form a group called the Web Hypertext Application Technology Working Group (WHATWG). The goal of WHATWG is to keep HTML development alive. Though it originally hesitated, W3C joined the HTML revival in 2006. Together, WHATWG and W3C combined existing specifications for HTML and XHTML and further developed them to create the new HTML5 specification. That specification is currently maintained and published by W3C.

This article explores this new HTML5 technology. We'll look at XHTML and other technologies that have gone into HTML5 and cover the basic points of how to use HTML5 to create attractive, standards-compliant web content. We'll also check out some exciting ways people are using HTML5 on the web. Let's start by looking at the goals of HTML5 and why it's far more than just HTML.

# HTML5 Goes Beyond HTML

By itself, HTML focuses on presenting static documents, meaning text, links and little else. Through HTML 4, the HTML standard was one in the same with the HTML syntax used to accomplish this objective. The HTML5 standard has a much broader goal to describe the content, styling and application interfaces behind a web page when it's loaded in your browser. More specifically, HTML5 establishes a single syntax for interacting with all the elements that web page has loaded into your computer's memory.

There's a potential point of confusion we should clarify about the term "HTML5" before we go on. There is a new version of the markup language itself called HTML5. However, that language is just one part of the broader standard that shares the name HTML5. This article is covering all the components of the HTML5 standard, of which the HTML5 language is a centerpiece.

With that distinction in mind, how does the browser load a page written according to the newer HTML5 standard and not the long-standing HTML 4 language? After all, we usually just rely on our browsers to load web pages without worrying about the HTML behind them. The answer is simple: As long as the browser itself can support HTML5, it can handle anything you throw at it without needing to distinguish between HTML revisions.

To make this magic happen, the HTML5 standard integrates new versions of multiple technologies working together toward common goals. These goals include the following:

* Keeping the language simple and intuitive
* Ensuring the code is easy to read and maintain
* Addressing pages as interactive applications rather than static documents
* Relying on Cascading Style Sheets (CSS) for styling the content
* Recognizing JavaScript as a central scripting component for web pages
* Embracing dynamic content from server-side technologies such as PHP and ASP

Now that you know that the HTML5 is more than the HTML markup itself, let's examine the foundation technologies in the standard.

# HTML5 Foundation Technologies

A markup language uses a series of tags to mark elements in a file. The markup relies on some sort of model that defines what those elements are and how to use them. Since HTML has always been about encoding documents, it relies on something called a document object model (DOM). The DOM is incorporated within the web browser application itself. Thus, the markup language is just a way of giving the browser instructions based on a model that it already understands.

To better understand the relationship between a DOM and HTML, think of the DOM as a box of rubber stamps. Each stamp is a different element within the DOM. Each page written in HTML is just a series of imprints made from the stamps, which are filled in with your own custom content. The web browser can display anything created from the stamps in that box.

Technically speaking, the DOM is an application programming interface (API) providing a platform-neutral way to process either HTML or XML documents. (Extensible Markup Language, or XML, looks similar to HTML, but it lets the programmer add a customized set of elements you can use in a document.) The HTML5 standard requires the DOM known as DOM5 HTML. Prepared for the growth in HTML5, the latest versions of [Chrome](https://computer.howstuffworks.com/google-chrome-browser.htm), [Firefox](https://computer.howstuffworks.com/internet/basics/firefox.htm), IE, Safari and Opera browsers can all process pages under the DOM5 HTML.

Another important foundation for HTML5 is its associated Multipurpose Internet Mail Extensions ([MIME](https://computer.howstuffworks.com/question334.htm)) types. MIME is an Internet Engineering Task Force (IETF) standard for alerting Internet-capable software what type of content it's trying to process. This typically corresponds to a specific type of file, such as an HTML document or MP3 audio file.

Web developers specify MIME types when they encode a web page. This prompts the browser to load the corresponding files appropriately according to their types. There are two MIME types that join DOM5 HTML to form the basis of the new HTML5 standard: the HTML MIME type (text/html) and the XML MIME type (application/xhtml+xml). HTML5 includes updates in the syntax for both of these MIME types.

In short, the fundamental technologies in the HTML5 standard are DOM5 HTML and the HTML5-compatible MIME types for HTML and XML. Next, let's move on to how to put together a web page using HTML5.

# Basic HTML Content

At a minimum, HTML5 includes version 5 of the basic HTML syntax. The sample code in the sidebar on this page shows what this HTML code might look like for a basic web page. When you're browsing the web, use the "View source" or similar option in your browser to see the HTML (and other code) loaded to create the page you're currently viewing. All of the code shown in our example is also backward compatible with HTML 4, meaning that it doesn't have any of the new additions available in HTML5.

Though it's not present in all web pages, it's a good practice to start your HTML file with a document type (doctype) declaration as shown in the example. This confirms that the browser should expect standard HTML when interpreting the document. Some developers rely on extensions to the HTML standard, which use document type definition (DTD) files. If that's the case, the developer specifies the location of the DTD as part of this doctype.

Besides the doctype, the remaining code in the example is standard HTML syntax. If you're not familiar with this syntax, see our article [How Web Pages Work](https://computer.howstuffworks.com/web-page.htm).

So, what's new to these basic elements in HTML5? Here's the list, and we'll take a closer look at each in the pages that follow.

* New and updated form elements -- laying out the form structure and processing form input from the user
* Semantic elements -- identifying content based on its role within a web page
* Media elements -- embedding audio, video and other interactive media without the need for third-party browser extensions
* Ruby elements -- supporting internationalization for Asian language web pages

# Forms

Though forms (sections of a website where users can input specific data, like names, addresses, etc.) were part of the earliest elements available in HTML, they lacked many useful features before HTML5. To make up for this, third party developers created web form software that make basic HTML forms seem nearly obsolete. HTML5 breathed new life into forms, though, with new form elements, input types and attributes for existing elements.

Without distinguishing between old and new syntax, let's examine the overall approach to creating a form for your web page using HTML5. Here some recommended steps to get started:

1. Establish a form block using the <form> tags.
2. Designate a block of fields within the form using the <fieldset> tags.
3. Lay out each field in the field set using <label> tags. Use the "for" attribute for each label, as you would the id attribute, to identify that label for use in other portions of your code. Example: <label for="name"></label>
4. Within each label tag, add the field name as you want it displayed plus the <input /> tag for the field itself.
5. Use the type attribute within the input tag (<input type="">) that prompts the browser to validate the user's input based on that type. This eliminates the need for additional code to handle validation and prompt for re-entry. HTML5 includes the following input types: date, time, datetime, datetime-local, month, week, number, e-mail, tel, url, range, color and search. You can also use the pattern attribute with a regular expression to create a custom validation pattern.
6. Indicate required fields by adding the required attribute for an input element.
7. Use the autofocus attribute to set a particular input element as the first element in the form. This is where a web browser will automatically place the user's cursor after loading the form.
8. Use the placeholder attribute if desired to add text hints about what to enter in a given field.
9. Set CSS styles for ":required" and ":invalid" so the user has a visual cue to missing information or input that won't validate for a given input type.

Check an up-to-date HTML5 reference guide for a complete list of form elements and their attributes. Also, note that support for some form fields varies between web browsers. Be sure to test your form in each type of browser you want your website to support.

While HTML form elements have more features in version 5, the next element type is completely new in HTML5.

# Semantic Elements

Semantic elements in HTML5 are, as their name implies, those that refer to the meaning of certain content. Just using <body> tags alone isn't enough, since most websites today contain several different types of content on a single page. web developers need to apply separate rules for how each type of content should look or behave.

Without semantic elements, web developers have used <div> and <span> elements to identify blocks of content, associating each with a particular class attribute. Each class could be associated with specific styles using CSS or with specific behavior using scripts.

Over time, certain content types became very common on websites, such as headers, footers, menus and navigation points. HTML5 incorporates these by adding standard semantic elements. This means you can break up your <body> block into helpful semantic elements rather than a series of <div> and <span> tags.

The following is the list of semantic page elements with a brief definition of each. Be sure to find an HTML5 reference you can trust for a complete list of valid attributes and child elements for each semantic element.

Block elements:

* **section** -- a generic page division used to help break up a larger block of content
* **article** -- content that is originally from an outside source, sometimes added dynamically on page load by using an aggregator script
* **header** and **footer** -- blocks that appear at the top and bottom of each page
* **hgroup** -- a grouping of multiple related headers, such as a title and subtitle
* **menu** -- a list of commands (see the command element), and attributes that specify the menu's behavior
* **nav** -- identifies a block that's strictly for website navigation, typically some unordered list of links to other pages on the site
* **address** -- includes contact information for the author of the content within a block, such as an article, section or entire page body
* **aside** -- indicates that the content should be treated as a sidebar

Inline elements:

* **summary** and **details** -- toggle between a teaser/summary and full details for the same content
* **figure** and **figcaption** -- elements used to apply common behavior to images, no matter what media elements (img, svg or canvas) are used to include them
* **time** -- text representing a calendar date, clock time or both, formatted so that the browser can adjust for time zone differences if necessary
* **command** -- a label and a behavior associated with that label when you use the keyboard or mouse to interact with it, typically used inside a menu block
* **dfn** -- a term that's being defined within the content
* **wbr** -- a tag indicating an acceptable place to break text within a word when it wraps across multiple lines

Semantic elements were created as an improvement over more generic HTML elements. The next new element type, though, aims to be an all-out replacement for third-party media extensions in web browsers.

# Media Elements and the Flash Rivalry

One of the most talked-about features of HTML5 is its new approach to embedding media on a [web page](https://computer.howstuffworks.com/web-page.htm). We'll take a look at some amazing applications of this HTML5 feature later in this article. HTML5 accomplishes this embedding without requiring an external browser plugin. The following are the media elements available in HTML5:

* **audio** -- embeds audio in the page and includes attributes for specifying how to play it; supported file formats will vary between web browsers
* **video** -- embeds video in the page and includes attributes for specifying how to play it; supported file formats will vary between web browsers
* **source** -- used with audio or video elements to identify a source; multiple sources could be specified for a single item
* **embed** -- embeds and specifies the media type for content that might lack support within other media elements
* **canvas** -- sets aside part of the page, or the computer screen, where JavaScript can draw images; coming up, we'll find out more about how this important new feature works
* **svg** -- embed vector graphics encoded with SVG markup language, allowing them to be scaled dynamically to the area of the page in which they're loaded without losing any graphic quality

As HTML5 developed, market experts questioned whether it would replace Flash, which is currently controlled by Adobe. Flash is a media player that you can add as a plugin to your web browser. Because Flash has all the features necessary for a site to embed different types of media, sites like YouTube have depended on it for years. There are even some web applications that rely entirely on Flash to power their interactive interfaces. Flash adoption was so widespread in 2007 that, upon the release of the much anticipated Apple iPhone, many consumers were upset to hear that its [operating system](https://computer.howstuffworks.com/operating-system.htm), Apple iOS, didn't support Flash.

When [Google](https://computer.howstuffworks.com/internet/basics/google.htm), arguably the biggest brand on the Internet, dove head-first into developing its apps with HTML5, Flash supporters took notice. Before HTML5, the only threat to Flash's dominance was web application developers willing to build and maintain custom HTML/JavaScript alternatives. With HTML5, such cumbersome customizations are no longer necessary when creating an interactive media experience to rival Flash technology.

So, who will win: Flash or HTML5? The short answer is, "Neither." Each technology has its own features, advantages and drawbacks. In addition, Adobe and Google have worked on Flash to HTML5 conversion tools. At the time of this writing, it seems that though HTML5 is gaining status, Flash is likely to retain its own popularity for a while, too.

Flash over to the next page where we'll break down the canvas element and how to use it in your HTML5 web pages.

# The Canvas

One of the most important new features to HTML5 is the canvas. The canvas feature lets you identify a section of a [web page](https://computer.howstuffworks.com/web-page.htm), draw content within it and add interactive functionality. All of this is possible with through a combination of HTML and [JavaScript](https://computer.howstuffworks.com/javascript.htm) code. Let's look at the three parts you'll need to set up a canvas in HTML5:

**Part 1:** Use the <canvas> tag to target a rectangular portion of your computer screen for a new canvas window. You can specify the height and width of the canvas, measured from its upper-left corner. The identity attribute (id) for the canvas is important because you'll need it to match its corresponding JavaScript code. The following is an example of using the canvas tag in HTML5 code:

<canvas id="sampleCanvas"

width="400"

height="300>

If you see this text, your browser

does not have HTML5 canvas support.

</canvas>

**Part 2:** Create the JavaScript that draws content into the canvas. Later, we'll find out how to add and manage the JavaScript code associated with an HTML5 web page. Here, let's just look at some sample JavaScript you can use for canvas objects. The following code draws a rectangle within the canvas above:

function drawRectangle() {

var targetCanvas = document.getElementById("sampleCanvas");

var rectContext = sampleCanvas.getContext("2d");

rectContext.fillStyle = "blue";

rectContext.fillRect(50, 25, 150, 100);

}

This function draws a solid blue rectangle within your canvas. The two "var" lines create variables you can use. The first (targetCanvas) identifies the canvas element you're targeting using its ID from the HTML. The second (rectContext) sets up a two-dimensional context for drawing. Think of the context as a clear layer of glass laid on top of the canvas, and you'll call functions to draw on that glass.

The "fillStyle" line identifies the color blue by name, which HTML recognizes and associates with one of its named colors. You can also use any color, pattern or gradient as you would when writing style code in HTML or CSS. The "fillRect" line draws the rectangle starting at an offset from the upper left, the first two numbers being the x and y offsets as if the upper left of the canvas was at coordinates (0,0). The second two numbers indicate the width and height of the rectangle, respectively.

**Part 3:** Call the JavaScript function from the HTML code. This ties the two pieces together. You could call the function as soon as the web page loads, which makes sense if you're using canvas to draw interactive page contents that need to appear as soon as the page loads. Another option is to create a button, menu or other element in the HTML that triggers the drawing event. The following is an example of a button you could use to run the example script above:

<button type="button"

onclick="drawRectangle()">

Click here to see the drawing.

</button>

Dive Into HTML5 has an easy-to-follow guide to creating canvas elements for your web pages. In addition, a reliable HTML5 reference should provide extensive resources for drawing sophisticated graphics, both still and moving. If you want to make extensive use of the canvas in HTML5, you'll need to be comfortable programming in JavaScript. Next, let's explore how CSS3 creates the look and feel of an HTML5 web page.

# Creating the Look with CSS3

Each HTML element has a number of attributes you use in the code to modify the look or function of that element. The style and related attributes adjust the size, color, placement, borders and other physical aspects of the element. Over the years of the web's lifespan, site developers widely adopted the practice of separating style from content. By making this separation, developers enjoy the following advantages:

* Less code to maintain: Multiple elements across multiple web pages can share the same styles without repeating the styling code for each one.
* Easier maintenance over time: Updating to a new style, or fixing an existing one, means changing only one piece of styling code rather than updating each element using that code.

Therefore, while styling can be placed in line with the HTML code, the more popular choice for web developers is to use Cascading Style Sheets (CSS) to manage all the styling code associated with a website. The HTML for each page can reference one or more style sheets, which your browser should load when the page loads. These style sheets, in the form of CSS files, are intended to mimic what you might enter using <style> tags within the <head> block in HTML.

While a guide to writing and using CSS is beyond the scope of this article, be sure to grab your own copy of a comprehensive reference for CSS3. The following is a summary of things you can do with CSS3 which complement the HTML5 standard:

* Apply a style to an element based on it having a specific attribute.
* Remove a style for all elements in a specific class.
* Lay out the page with the flexible box layout to counter the challenges and shortcomings of floating div blocks.
* Include a font file and associate it with a custom font-family label to use throughout the CSS.
* Add a shadow (text-shadow) or outline (text-stroke) to the font for text elements.
* Specify a color using the RGBA format.
* Make use of popular visual elements, including rounded corners, gradients, shadows, reflections and border images.
* Adjust the opacity of an element.
* Add more elaborate transition animations for elements.
* Move, rotate, scale or skew an element with the transform attribute.

Note that while most major browsers support the majority of items on the list above, some browsers still haven't fully implemented these CSS3 features. As with HTML5, be sure to test your CSS code in each browser you want your website to support. Next, let's look at the core technology that's helping achieve HTML5's goals: JavaScript.

# JavaScript as a Central Component

[JavaScript](https://computer.howstuffworks.com/javascript.htm) is a language specifically designed for the web. The syntax for writing functions and referencing objects in JavaScript is similar to Java and other object oriented languages. That's where the similarity ends, though. JavaScript is not compiled to create an executable file. Instead, JavaScript associated with a web page is loaded in your web browser when that page is loaded, and JavaScript functions are called based on the interactions you have with the page.

You could write the JavaScript code within the HTML code using tags around it, similar to using <style> tags for inline CSS. However, as with CSS styles, the best practice for script code is to put it in a separate file and reference that file from the HTML. This also gives you the flexibility to load the script only if the browser meets certain requirements. In HTML5, your web browser will also cache JavaScript and other files that support a web page, so the page will function properly in offline mode, too.

The following is an example of including a JavaScript file named "example.js" which is in a scripts directory one level above the referencing web page:

<script language="javascript" type="text/javascript"

src="../scripts/example.js"></script>

As with HTML and CSS, JavaScript basics are beyond the scope of this article, and you should pick up a good reference to use while you program. Be sure the reference covers JavaScript for HTML5 so you don't miss these HTML5 enhancements:

* Reference elements by class name, tag name or a custom query selection.
* Allocate and use up to 5 MB of local storage on the client system rather than setting and using cookies in the web browser.
* Create and use a WebSQL database as part of that local storage.
* Make use of web sockets, a new feature that maintains a live client-server connection between your browser and a web server.
* Retrieve geolocation data from the client computer via the browser, and use that data for other operations, such as displaying a map of that location.

So far, we've looked at the client-side part of HTML5. In other words, we've looked at the code your web browser downloads from the [Internet](https://computer.howstuffworks.com/internet/basics/internet-infrastructure.htm) and processes on your local computer. HTML5 standards also reach to the other side of that connection, though. Next, let's look at the server-side technology behind HTML5.

# Web Application Technologies

Earlier, we determined that HTML5 is approaching web content not as a set of documents, but as a network of web applications. As such, your browser isn't just a place to process the content you download. It's also a portal to additional software residing on remote [web servers](https://computer.howstuffworks.com/web-server.htm). In the HTML5 standard, this remote software is consistently available to support your interactive web experience.

So what kind of software are we talking about? Two widely used examples are Hypertext Preprocessor (PHP) and Active Server Pages (ASP). These are web development languages that assemble content on demand from the server side. For example, when you answer a poll question on a website and get back a graphic showing how the votes have been distributed, there's probably some PHP at work on the page. You won't see PHP or ASP code itself, even if you view the source of a page. Instead, by default, you'll only see the dynamically assembled HTML and [JavaScript](https://computer.howstuffworks.com/javascript.htm) produced by the server before it's loaded in your browser window.

Another server-side technology that can support HTML5 web pages is database software. For example, both MySQL and Oracle can store and manage large amounts of data and keep it secure from unauthorized queries. A web application can accept credentials and access that data as needed, responding to queries and allowing the app to retrieve the requested information.

With the flexibility of the web and the ever-expanding features of technology like HTML5, it's possible that more web applications will call on server-side software for support. The growing field of cloud computing already depends on this client-server relationship through the web. HTML5 is ready to accommodate this relationship from the client side.

Now that we've looked at the technical aspects behind HTML5, let's look at some highly publicized applications of HTML5 that have helped establish its reputation.

# Groundbreaking Proofs of Concept

A proof of concept is a demonstration that a new technology can live up to its claims in a practical, real-world application. As you've read so far, HTML5 by itself is a revision of application programming interfaces (APIs) and the standards for using them. For HTML5, the proofs of concept come through demonstrations of the products using the technology.

Today, there are many games programmed in HTML5, but they don't typically show off everything HTML5 can do for a website. With Google leading the charge for HTML5 adoption, it's published several entertaining demonstrations of what HTML5 can do in its [Chrome web browser](https://computer.howstuffworks.com/google-chrome-browser.htm). You can find these at the [Chrome Experiments](http://www.chromeexperiments.com/) site.

In one of the most talked-about demonstrations back in August 2010, the band Arcade Fire teamed up with Google to produce an interactive music video experience called "[The Wilderness Downtown](http://www.thewildernessdowntown.com/)." To start the app, you enter the address of your childhood home. After it incorporates that input, you click to launch the multiple window video experience. The video incorporates geolocation images overlaid with silhouettes of birds flying overhead and a person running through the streets.

In July 2011, the band OK Go collaborated with innovative performance art dance company Pilobolus for its own HTML5 Chrome Experiment called the "All Is Not Lost Video Dance Messenger." Similar to the Arcade Fire video, you start by entering some text and letting the application process that input. Then, when you launch the app, the HTML5 code draws and resizes multiple windows on your screen as it displays portions of the video in each one. Eventually, the message you entered at the beginning is spelled out with dancers' feet, filmed from underneath a glass floor.

For the typical website you visit, you probably won't know it's an HTML5 site unless it tells you. One thing's for certain, though: Web designers have been able to take their art to a new level, thanks to HTML5 features like the canvas. Designers and developers alike should see HTML5 as an essential technology to master in the coming years.

In this article, we've discovered the components of HTML5 and how they can work together to create web masterpieces. We've even glimpsed some innovative HTML5 creations showing off its versatility. For much more on HTML5, including links to online resources you can reference when developing with HTML5, keep reading.