THE CROSS BROWSER HANDBOOK

DANIEL HERKEN

Contents

HTML 5

HTML 5 introduces many great new features into the HTML standard. As the time of writing the HTML 5 specification is still not finalized, but all major browser vendors have already started to implement many of these features. In the following I'll describe the newest features that browsers already support, and how you get support ready in older browsers starting with Internet Explorer 7. As mentioned earlier, you will be hard-pressed to get these features going in Internet Explorer 6.

Please remember that this book has no auto-update function, so be sure to check our online resource at www.crossbrowserbook.com/Knowledge to get the latest information about browser support. Moreover, you can include the great library Modernizr (http://modernizr.com) in your page; it detects all HTML 5 features you can use in the current browser.

Of course HTML 5 contains many more features, elements and attributes than are described here. You could fill a book just about HTML 5 (someone already has: http://diveintohtml5.info).

Multimedia and Drawing

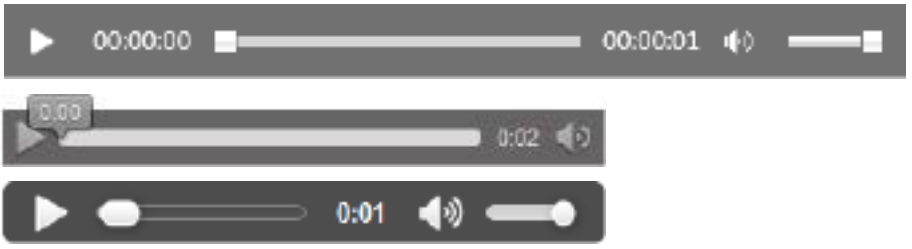
There are quite a few multimedia and drawing capabilities available in HTML 5. You can play sound and videos without the need for any additional plugin. Moreover you can show vector images and draw on a canvas using JavaScript.

<audio>

The `<audio>` tag is one of the great new features included in HTML 5. It allows for native supported sound or audio streams to the user without the use of any plug-in (like Flash). You can use the `<audio>` tag like this:

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

Browsers that support this tag display a media player control so the user can start, stop and resume the audio playback. Of course these media player controls does not have a consistent look across all the major browsers.



Moreover, not every browser supports the same audio formats. Because of this you can add more than one source tag inside the `<audio>` tag.

The browser will automatically select the audio source it supports (if any). Here are the audio formats currently supported across the different browsers:

								
9+ MP3, AAC	3.5+ MP3, OGG	3+ MP3, OGG, WAV, AAC	3.1+ MP3 AAC, WAV	10+ OGG WAV	2.3+ MP3, AAC WAV	4.0+ MP3, AAC, WAV	7.5+ MP3, AAC	7.0+ MP3, AAC, WAV

Unfortunately, the browsers Internet Explorer 7 and 8 don't support this tag out-of-the-box, so you will need to provide a fallback solution for these browsers. The simplest fallback would be a link to download the file manually. This link will not be visible in browsers that support the `<audio>` tag.

```
<audio controls="controls">
   <source src="music.mp3" type="audio/mp3"/>
   <source src="music.ogg" type="audio/ogg"/>
   Not audio playback support, please download
   the <a href="music.mp3">file</a> here.
</audio>
```

To get `<audio>` tag support for older Internet Explorer versions you can use the following:

jPlayer (http://github.com/happyworm/jPlayer) uses Flash to simulate the audio tag for older Internet Explorer versions.

audio.js (http://kolber.github.com/audiojs) uses Flash to simulate the audio tag for older Internet Explorer versions.

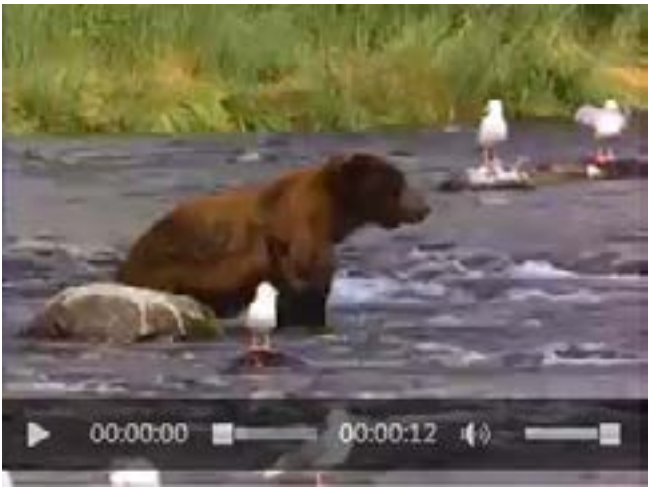
SoundManager2 (http://www.schillmania.com/projects/soundmanager2/) uses a flash fallback

<video>

The <video> tag introduced in HTML 5 can be used to directly embed video content without the need of an additional plug-in. You can use the <video> tag like this:

```
<video width="640" height="320" controls="controls">
   <source src="video.mp4" type="video/mp4"/>
   <source src="video.ogg" type="video/ogg"/>
</video>
```

Browsers supporting the <video> tag will now display media player controls to control the video playback. Like the <audio> tag, the different browsers display this media player different:



Again, the format support differs between the browsers:

9+ MP4	3.5+ WebM, OGG	3+ MP4, WebM, OGG	3.1+ MP4	10+ WebM OGG	2.2+ MP4	3.2+ MP4	7.5+ MP4	7.0+ MP4

To provide a fallback for older browsers like Internet Explorer 6, 7 and 8 you can provide a download link:

```
<video width="640" height="320" controls="controls">
   <source src="video.mp4" type="video/mp4"/>
   <source src="video.ogg" type="video/ogg"/>
   Not video playback support. Please
   download the <a href="video.mp4">video</a>.
</video>
```

To get video support in older Internet Explorer versions you can use the following:

html5media (http://html5media.info/) uses Flash to simulate the `<audio>` and `<video>` tag for older Internet Explorer versions.

mediaelementsjs (http://mediaelementjs.com/) uses Flash and Silverlight to emulate the `<audio>` and `<video>` tag for older Internet Explorer versions.

videojs (http://videojs.com/) uses Flash as a fallback solution

<canvas>

The `<canvas>` tag provides script access to a resolution-dependent bitmap canvas. This canvas can be used to render graphs, game graphics or other visual images on the fly using JavaScript. You can use the `<canvas>` tag like this:

```
<canvas id="canvas">
  Your browser does not support the HTML 5 canvas tag.
</canvas>
<script type="text/javascript">
   var canvas = document.getElementById("canvas");
   var drawingContext = canvas.getContext("2d");
   drawingContext.fillStyle="#000000";
   drawingContext.fillRect(0,0,200,100);
</script>
```

This code will draw a rectangle (200x100 pixel) filled with black color. Luckily, the drawing is consistent across all supporting browsers. These are:

9+	1.5+	1+	2+	9+	2.1+	3.2+	7.5+	7.0+

To get canvas support in older Internet Explorer versions you can use the following:

FlashCanvas (http://flashcanvas.net/) simulates the `<canvas>` tag in older Internet Explorer versions using Flash and JavaScript.

explorercanvas (http://code.google.com/p/explorercanvas/) simulates the `<canvas>` tag in older Internet Explorer versions using JavaScript and VML.

fxCanvas (http://code.google.com/p/fxcanvas/) simulates the `<canvas>` tag using flash

<svg>

The `<svg>` tag can be used to create scalable vector graphics directly in HTML or an external resource. These graphics don't lose quality if they get scaled or zoomed. A quick example:

```
<svg xmlns="http://www.w3.org/2000/
svg" version="1.1" height="100">
   <polygon points="100,10 40,100 100,60 10,60"
   style="fill:lime;stroke:green;stroke-width:2;">
</svg>
```

This code will result in a browser drawing the following (odd) graphic:

Currently most modern browsers support the `<svg>` tag:

								
9+	1.5+	1+	3+	8+	3.0+	3.2+	7.0+	7.0+

To get `<svg>` support in older browsers you can use the following Scripts:

websvg (http://code.google.com/p/svgweb/) uses JavaScript to simulate `<svg>` if needed

Raphaeljs (http://raphaeljs.com/) uses JavaScript to simulate `<svg>` if needed

Ample SDK (http://www.amplesdk.com/examples/markup/svg/) is an open-source GUI Framework

SIE (http://sie.sourceforge.jp/) emulates svg using VML

Dojo Toolkit (http://livedocs.dojotoolkit.org/) includes fallback via VML, Canvas, Silverlight and Flash

Web Applications

With HTML 5 it got much easier to create full featured web applications. Features like geolocation, web storage, web workers and many more allow for feature rich web applications directly run by the browser.

Application Cache

With the use of the HTML 5 application cache it's easy to get your web application to work offline. This can be accomplished by creating a cache manifest file. In this manifest file you can specific resources the browser should download and use in case your application is running offline. An example of this manifest file you look like this:

```
CACHE MANIFEST
/style.css
/logo.png
/application.js

NETWORK:
login.asp

FALLBACK:
/html/ /offlinemode.html
```

To get this working you need to specify this manifest file in the main page of your application.

```
<html manifest="app.appcache">
```

All major browsers starting with Internet Explorer 10 support this:

10+	3.5+	4+	4+	10+	2.1+	3.2+	8.0+	7.0+

Currently there is no polyfill available to get this functionality working in older browsers. You can find a unfinished implementation using Google Gears here: http://code.google.com/p/html5-gears/.

File API

One major shortcoming of web applications is working with files on the client side. Uploading and manipulating files was often only possible by using plug-ins like flash. The File API enables you to simply manipulate files and enhances the file upload user experience. A quick file upload example could look like this:

```
function startRead() {
   var file = document.getElementById('fl').files[0];
   getAsText(file);
}

function getAsText(readFile) {
   var reader = new FileReader();
   reader.readAsText(readFile, "UTF-16");
   reader.onprogress = updateProgress;
   reader.onload = loaded;
   reader.onerror = errorHandler;
}

function updateProgress(event) {
   if (event.lengthComputable) {
      var loaded = (event.loaded / event.total);

      if (loaded < 1) {
        // Here you could show the process to the user
      }
   }
}
```

```
function loaded(event) {
   var fileString = event.target.result;
   // Here you could manipulate the file
}

function errorHandler(event) {
   if(event.target.error.name == "NotReadableError") {
      // Error handling goes here
   }
}
```

Starting with Internet Explorer 10 every modern browser supports the file API functionality:

								
10+	6+ moz	13+ webkit	6+	12+	4.0+	6.0+	8.0+	10.0+

There is a script available to get the file API ready in older browsers:

FileReader.js (https://github.com/Jahdrien/FileReader)

Dropfile.js (https://github.com/MrSwitch/dropfile)

FileSaver.js (https://github.com/eligrey/FileSaver.js)

Geolocation

The HTML 5 geolocation feature allows you to detect the physical location of your websites users. In all implementing browsers the user has to approve this to happen. Of course the location is much more accurate on mobile devices which have access to GPS data. You can detect the location using JavaScript:

```
<script type="text/javascript">
    function getLocation() {
       if (navigator.geolocation) {
         navigator.geolocation
                  .getCurrentPosition(showLatAndLong);
       }
       else {
          alert("Geolocation not supported.");
       }
    }

    function showLatAndLong(position) {
       alert("Latitude: "
             + position.coords.latitude +
             " Longitude: "
             + position.coords.longitude);
    }
</script>
```

Starting with Internet Explorer 9 every modern browser supports the geolocation functionality:

								
9+	3.5+	5+	5+	10+	2.0+	3.2+	7.0+	7.0+

There are a few scripts available to get geolocation ready in older browsers:

webshim (https://github.com/aFarkas/webshim) enables a geolocation fallback using JavaScript

geolocation shim API (https://github.com/manuelbieh/Geolocation-API-Polyfill) enables geolocation using JavaScript

History API

The back and forward buttons are the most used buttons inside the browser UI. The standard browser functionality works very well for static web pages but modern web applications don't work the same way. The HTML 5 History API enables you to manipulate the history state using JavaScript.

The history API allows for history navigation:

```
window.history.back();
window.history.forward();
window.history.go(-1);
window.history.go(4);
```

Moreover you can manually push and replace the history state:

```
var stateObj = { foo: "bar" };
history.pushState(stateObj, "page", "bar.html");
history.replaceState(stateObj);
```

Starting with Internet Explorer 10 every modern browser supports the history API functionality:

								
10+	4+	5+	5+	11+	2.0+	3.2+	8.0+	7.0+

There are scripts available to get the history API ready in older browsers:

HTML 5 history API (https://github.com/devote/HTML5-History-API)

History.js (https://github.com/balupton/history.js)

Server-Sent Events

In the classic AJAX and JavaScript world every interaction starts with the client connecting to the server. So if a webpage needs periodical updates it can only ask the server for it. With the introduction of the server-sent events this is no longer true. The server-send event can be used to push updates down to the client. A quick example:

```
<script type="text/javascript">
   if (typeof (EventSource) !== "undefined") {
      var source = new EventSource("serverside.php");
      source.onmessage = function (event) {
         alert(event.data);
      };
   }
   else {
      alert("No server-send event support");
   }
</script>
```

Not every modern browser supports these server-sent events, including Internet Explorer.

								
-	6+	9+	5+	11+	-	4.1+	-	-

To get support for server-sent events you can use this script:

Portal (https://github.com/flowersinthesand/portal)

Web Sockets

The HTML 5 web sockets are a technology designed to get real-time communication between the web server and browser using only HTTP communication. It works directly over TCP and can be used by JavaScript:

```
<script type="text/javascript">
   function Test() {
      if ("WebSocket" in window) {
         var socket = new WebSocket("ws://theserver");
         socket.onopen = function () {
            socket.send("Something");
            alert("Sent something...");
         };
         socket.onmessage = function (event) {
            alert(event.data);
         };
         socket.onclose = function () {
            alert("Connection closed");
         };
      }
      else {
         alert("No web socket support");
      }
   }
</script>
```

Currently all modern browsers, starting with Internet Explorer 10, do support web sockets.

								
10+	11+	16+	5+	11+	4.0+	4.1+	8.0+	7.0+

To get support for web sockets in older browsers you can use the following scripts:

Portal (https://github.com/flowersinthesand/portal) uses JavaScript to add web socket support

Kaazing Web Socket (http://kaazing.com/products/kaazing-websocket-gateway.html) is a commercial product to enable web sockets in older browsers

Web Storage

The web storage integrated in HTML 5 allows for a better local storage than cookies. The advantage over the usage of cookies is that the web storage is only attached attached to a request if it is actually requested. The web storage can be separated in local storage and local session storage. It is available using JavaScript:

```
<script type="text/javascript">
    if (typeof (Storage) !== "undefined") {
        localStorage.testvalue = "The value";
        alert(localStorage.testvalue);
    }
    else {
        alert("No web storage support.");
    }
</script>
```

And to use the session storage:

```
<script type="text/javascript">
    if (typeof (Storage) !== "undefined") {
        sessionStorage.testvalue = "The value";
        alert(sessionStorage.testvalue);
    }
    else {
        alert("No web storage support.");
    }
</script>
```

Starting with Internet Explorer 8 every modern browser supports the web storage:

								
8+	3.5+	4+	4+	10+	2.0+	3.2+	7.5+	7.0+

To get web storage support in older browsers you can use the following scripts:

Storage polyfill (https://gist.github.com/350433)

Real storage (http://code.google.com/p/realstorage/)

Sessionstorage (http://code.google.com/p/sessionstorage/)

YUILibrary (http://yuilibrary.com/yui/docs/cache/#offline)

jStorage (http://www.jstorage.info/)

Web Workers

In the old HTML 4 world JavaScript is single threaded. This means if your JavaScript takes up a lot of time to execute the page appears to "hang" and the user cannot interact with it while JavaScript is running. This can be overcome by the HTML 5 web workers which enable you to run JavaScript in the background, without blocking the page interaction. You can start, send messages and terminate a web worker using JavaScript.

```
<script type="text/javascript">
   var worker;

   function startWorker() {
      if (typeof (Worker) !== "undefined") {
         worker = new Worker("workers.js");
      }
      worker.onmessage = function (event) {
         alert(event.data);
      };
      else
      {
         alert("No web worker support.");
      }
   }
   function stopWorker() {
      worker.terminate();
   }
</script>
```

The web workers are supported by all major browsers starting with Internet Explorer 10.

								
10+	3.5+	3+	4+	10+	-	5.0+	8.0+	7.0+

Currently there is no working script available to get this working in older Internet Explorer versions.

Web Forms

HTML 5 includes a wide array of new features to simplify the building of web forms. Unfortunately, none of these are supported in Internet Explorer 6, 7 and 8.

Placeholder text

One of the improvements is that you can define a placeholder text that is displayed inside an input field as long as the field is empty. You can use the placeholder text like this:

```
<input type="text" placeholder="Enter something here"></
input>
```

Currently, this placeholder is supported in most browsers:

								
10+	4+	11+	5+	11+	2.1+	3.2+	8.0+	7.0+

To get placeholder support in older browser versions you can use the following scripts:

jQuery.HTML5Support (https://github.com/amiel/html5support) creates placeholders in older Internet Explorer versions.

Placeholder.js (https://github.com/jamesallardice/Placeholders.js) enables placeholders in older Internet Explorer versions (including Internet Explorer 6).

Autofocus fields

Many pages automatically focus on one element like, for example, the Google search field. This can be done using JavaScript or, as introduced in HTML 5, using the autofocus attribute. You can use this attribute like this:

```
<input type="text" autofocus="autofocus"></input>
```

Most browsers already support the autofocus attribute:

								
10+	4+	11+	5.1+	11+	2.1+	3.2+	8.0+	7.0+

To get autofocus support in older browser versions you can use the following:

jQuery.HTML5Support (https://github.com/amiel/html5support) adds support for the autofocus attribute to older Internet Explorer versions.

New Input types

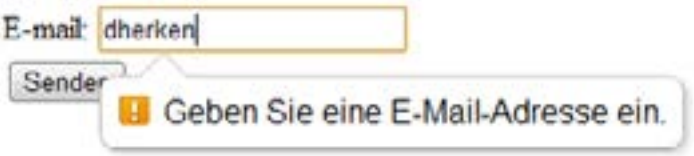
HTML 5 introduces more new input types that help the user fill out web forms faster and with greater ease. The great thing is, even though all browsers do not already support the new input types, you can still start using them. If the browser does not support an input type, it will simply fall back and display a raw text input field. Let's take a look at a few of the new input types:

Email

About 50% of all input fields used on the web are there to enter an email address. And nearly every developer runs his own email validation using server processing, JavaScript or both. Not anymore! This new input type gives the browser the ability to validate email addresses without the need for any code. You can use the email input type like this:

```
<input type="email"></input>
```

Now, if the user submits the web form the browser automatically validates this input field and displays a message to the user if necessary. For example, Google Chrome shows a small notification:



Url

The input type url does exactly the same as for email but for webpage addresses. You simply define the input type and the browser validates the field content for you. You can use the input type like this:

```
<input type="url"></input>
```

Likewise, the browser notifies the user if anything needs to be checked:

Date

There are many jQuery and JavaScript plug-ins out there that let the user select a date by displaying a simple calendar. Unfortunately, this means many different UI-Styles and user confusion all around. HTML 5 introduces the date input type to get a consistent look and feel (at least inside a browser). You can use the date input type like this:

```
<input type="date"></input>
```

The user will then be able to select a date using a calendar widget.

Range

Sometimes your users need to select a value in a range of possibilities. Using a desktop application, the old slider control is always a welcome solution for this, but recreating this using JavaScript is not easy. HTML 5 introduces the new input type called range, which gives you native browser support for the slider. You can use the input type like this:

```
<input type="range" min="1" max="100"></input>
```

As expected, the browser shows a slider control:

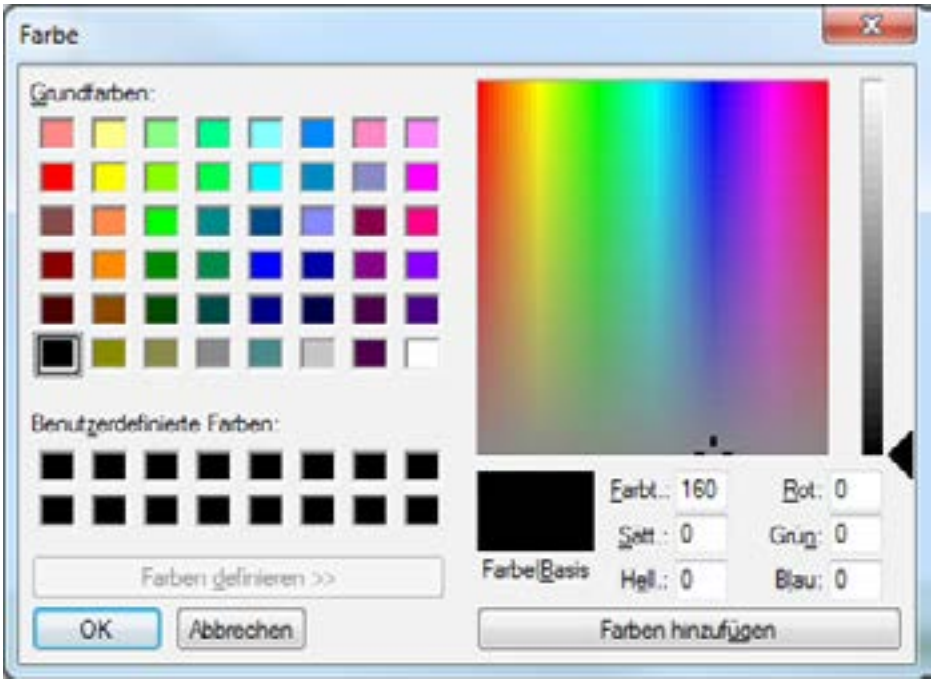
Color

Sometimes the user just needs to select a color. Maybe the page wants to create a cross-browser gradient for you (Colorzilla anyone? http://www.colorzilla.com/gradient-editor/). Now, in the HTML 4 world you could use a jQuery plug-in or some JavaScript to let the user select a color.

In HTML 5 you can use the input type color, which gives the user a color picker to easily select a color. You can use the input type like this:

```
<input type="color"></input>
```

And the browser displays something like:



Form validation

In HTML 5 web forms are improved with a wide range of automatic validation options that get feedback to the user without the need of a page refresh or any JavaScript validation. There are attributes like required and novalidate that do exactly what the names suggest. You can use these attributes like this:

```
<form novalidate="novalidate">
   <input type="email" required="required"></input>
</form>
```

Only some browsers already support one or more of these new HTML 5 input types and validation rules:

									
EMail	10+	15+	18+	5.1+	12+	2.2+	5.0+	8.0+	-
Url	10+	15+	18+	5.1+	12+	2.2+	5.0+	8.0+	-
Date	-	-	18+	5.1+	12+	-	5.0+	-	10.0+
Range	10+	-	18+	5.1+	12+	2.1+	5.0+	8.0+	-
Color	-	-	18+	-	12+	-	-	-	-
Noval-idate	10+	15+	18+	5.1+	12+	2.1+	3.2+	8.0+	7.0+
Re-quired	10+	15+	18+	5.1+	12+	2.1+	3.2+	8.0+	7.0+

You can enable support for many new HTML 5 input types and validation rules using one of these scripts:

WebShimsLib (http://afarkas.github.com/webshim/demos/index.html) adds support for email, url, date, number, range and many new validation attributes.

HTML5 Form Polyfill (http://demo.dsheiko.com/html5form-shim/) adds support for email, url, number, tel and many new validation attributes.

Semantic elements

HTML 5 introduces some semantic elements to help group and separate content on a website. These semantic elements help search engines and software (like screen readers) to understand the meaning of the information on a webpage. A good example for this is the `<b>` tag which will toggle a bold typeface inside the web browser. A screen reader would interpret this as louder speech while a search engine may use the `<b>` tag to identify more important content.

Let's take a quick look at the new tags, the browser compatibility table, a complete site example on how to use these tags and some workarounds to get support them supported in Internet Explorer 7 and 8.

<header>

The `<header>` tag represents the header element of a webpage. You can use it to wrap a group of headlines or an `<hgroup>`. Alternatively, you can integrate a classic page header (e.g. logo, the navigation, search box...) into the `<header>` tag.

<hgroup>

The `<hgroup>` tag represents the heading of a section. The tag can be used to group some h1, h2, h3, h4, h5 and h6 elements (which represent subheadings, alternative titles or taglines).

<nav>

The <nav> tag represents a section of a page or application that links to other pages or parts within the current page. Simply a section with navigation links.

<article>

The `<article>` tag represents a self-contained composition in a page or application. Examples for this could be a forum post, newspaper article or blog entry. When `<article>` tags are nested, the inner article elements represent articles that are related to the outer article. An example of this are the user comments for a blog entry.

<section>

The `<section>` tag represents a generic section of a webpage or application. It can be used to group content on a page. This could be chapters, tabbed pages or numbered sections of a thesis. For example, a webpage could be split into sections like introduction, news items and contact information.

<aside>

The `<aside>` tag represents a part of the page that consists of content that is related to the content outside the aside element. An example is a sidebar with related content or related advertising.

<figure>

The `<figure>` tag represents some content that is self-contained and typically referenced as a single unit from the main document. For example, this tag can be used to annotate illustrations, diagrams, photos or code.

<figcaption>

The `<figcaption>` tag represents a caption or legend for the rest of the parent figure element.

<footer>

The `<footer>` tag represents the footer for its nearest ancestor element. This could be the page footer containing copyright data and some links. Alternatively, it could represent the footer for a blog entry or news article containing information like the author and publishing date. The footer element does not need to appear at the end of a page.

Site example

Following is an example site that shows how you can use these new HTML tags. The site represents a blog post.
For readability I've omitted the `<head>` tag and most of the real content.

```
<body>
   <header>
      <hgroup>
         <h1>Blog</h1>
         <h2>The blog about something</h2>
      </hgroup>
      <nav>
         <a href="home.html">Home</a>
         <a href="contact.html">Contact</a>
         <a href="imprint.html">Imprint</a>
      </nav>
   </header>
   <div>
      <article>
         <header>
            <h1>Here could be your headline</h1>
         </header>
         <div>
            <p>Some content would go here</p>
            <aside>
               Maybe place an ad here?
            </aside>
            <p>Some more content …</p>
            <figure>
               <img src="statistic.png"/>
               <figurecaption>
                  The statistic data
```

```
                </figurecaption>
            </figure>
        </div>
    </article>
</div>
<footer>
    <p>Copyright 2012 by Daniel Herken</p>
</footer>
</body>
```

Most browsers support these semantic tags out-of-the box:

								
9+	15+	18+	5.1+	11+	2.2+	4.0+	7.5+	7.0+

To use this tags in Internet Explorer 6, 7 and 8, you can include one of these scripts. Both scripts use JavaScript to simulate these tags.

HTML5shiv (http://code.google.com/p/html5shiv/)

More HTML 5 tags

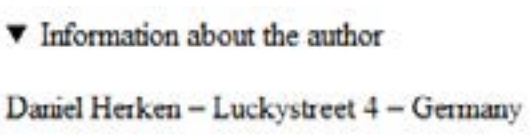
Following are some more HTML tags that are not yet widely supported by any browser or rendering engine. But there are some solutions that simulate these tags using JavaScript or a plug-in.

<details> and <summary>

The `<details>` tag represents additional information that should include a `<summary>` tag. The user can hide or view the information inside the `<details>` tag on demand. If the tag gets closed by the user, only the content of the `<summary>` tag is displayed. You can put any type of content inside the `<details>` tag. It can be used like this:

```
<details>
   <summary>Information about the author</summary>
   <p>Daniel Herken - Luckystreet 4 - Germany</p>
</details>
```

The browser displays the example like this:



Currently, not many browsers support this tag:

-	-	12+	-	12+	4.0+	6.0+	7.5+	10.0+

To get support for the `<details>` and `<summary>` tag you can use the following:

Element.details (https://github.com/termi/Element.details) adds support for `<details>` and `<summary>` to all browsers (including Internet Explorer 6).

jQuery-details (http://akral.bitbucket.org/details-tag/) adds support for `<details>` and `<summary>` tag using jQuery and JavaScript to all browsers.

<meter>

Another new tag introduced in HTML 5 is the `<meter>` tag. It defines a measurement within a known range or a factional value. It is displayed just like a progress bar, but if you want to display fast changing progress then you should use the `<progress>` tag. You can use the `<meter>` tag like this:

```
<meter value="0.4"></meter>
<meter value="8" min="0" max="10"></meter>
```

The browser displays the example like this:

Currently, not many browsers support this tag:

-	-	6+	5.2+	11+	-	-	-	10.0+

To get support for the `<meter>` tag in all other browsers you can use the following:

meter fallback (https://gist.github.com/667320) uses JavaScript to add `<meter>` support to all major browsers.

meter jQuery plug-in (https://github.com/xjamundx/HTML5-Meter-Shim) uses JavaScript to add `<meter>` support to all browsers.

<progress>

As mentioned earlier, there is a new HTML 5 element that should be used to display fast changing progress to the user. It can be accessed using JavaScript to update the currently displayed progress. You can use the `<progress>` tag like this:

```
<progress value="45" max="100"></progress>
```

The browser displays the example like this:

Currently, only some browsers support this tag:

10+	6+	6+	5.2+	11+	-	-	-	10.0+

To get support for the `<progress>` tag in all other browsers you can use the following:

`progress polyfill` (https://github.com/LeaVerou/HTML5-Progress-polyfill) to enable the `<progress>` tag in all major browsers.

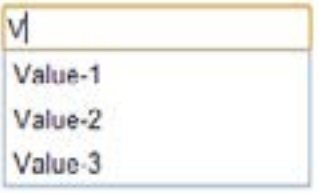
<datalist>

The last HTML 5 element is called `<datalist>` and it can be used to help users fill out an input field. The `<datalist>` tag specifies a list of options for an input. The user will see a drop-down of options once he starts to input his data. You can use the `<datalist>` like this:

```
<input list="preDefinedValues" />

<datalist id="preDefinedValues">
   <option value="Value-1">
   <option value="Value-2">
   <option value="Value-3">
</datalist>
```

The browser displays the example like this:



Currently, only some browsers support this tag:

-	15+	20+	-	12+	-	-	-	-

To get support for the `<datalist>` tag in all other browsers you can use the following script:

jQuery HTML 5 datalist plug-in (http://miketaylr.com/code/datalist.html)