DZone Refcardz

Get More Refcardz! Visit refcardz.com

www.dzone.com

CONTENTS INCLUDE:

HTML5: New Standards for Web Interactivity

By James Sugrue

ABOUT HTML5

HTML5 is a standard for structuring and presenting content on the Web. It incorporates features such as geolocation, video playback and drag-and-drop. HTML5 allows developers to create rich internet applications without the need for third party APIs and browser plug-ins.

HTML5 is still under specification, and is currently in the Working Draft stage in the W3C, but many aspects of HTML5 are now stable and can be implemented in browsers.

This DZone Refcard highlights the main features in HTML5 and illustrates the JavaScript APIs available to work with those features.

CHANGES TO EXISTING COMPONENTS

Simplified Syntax

There are a number of simplifications to the syntax of HTML introduced in the HTML5 specification.

Document Type

The `<doctype>` for an HTML document has changed from its verbose DTD reference to a much simpler format, simply stating the document is an HTML document type:

```
<!doctype html>
```

This change allows HTML5 to be fully backward compatible with previous versions of HTML.

Character Encoding

The `<meta>` tag for a document allows you to set the character encoding using the simple `charset` attribute, replacing declarations such as:

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

with

```
<meta charset=UTF-8">
```

Hot Tip

Other Encoding Options

You can also set the encoding of the document by setting the Byte Order Mark at the start of the file, or by setting the HTTP Content-Type header at transport level.

Script and Link Elements

The `<script>` element has been stripped down, removing the need for the type attribute. The reason for this is that scripts are typically written in JavaScript. The `<link>` element has lost its type attribute due to the prevalence of CSS.

Hot Tip

Embedding Scalar Vector Graphics (SVG)

Both SVG and MathML elements can be directly embedded into your HTML5 document utilizing their syntax features.

NEW ELEMENTS IN HTML5

The following table presents an overview of the new elements that have been added to HTML5.

Element	Description
<article>	An independent piece of content for a document e.g. blog entry, forum entry
<aside>	A piece of content that is somehow related to the rest of the page
<audio>	Audio media content
<canvas>	A component for rendering dynamic bitmap graphics on the fly. e.g games
<command>	A command that the user can invoke: a button, radio button or checkbox
<datalist>	Together with the new `list` attribute for the `<input>` element can be used to make combo boxes
<details>	Additional information or controls that the user can obtain on demand, to provide details on the document, or parts of it
<embed>	Used for plug-in content
<figure>	A piece of self-contained flow content referenced as a single unit from the main flow of the document
<figcaption>	Caption for a `<figure>`
<footer>	Footer for a section; may contain information about author, copyright information, etc.
<header>	A group of introductory or navigation aids
<hgroup>	Header of a section
<keygen>	A key pair generation control for user authentication in forms
<mark>	A run of text in one document marker or highlighted for reference purposes

<meter>	A measurement, such as disk usage, when the minimum and maximum values are known.
<nav>	A section of the document intended for navigation
<output>	Output such as a calculation done through scripting
<progress>	Represents progress of a task such as downloading or performing other expensive operations
<rt> <rp> <ruby>	Enables Ruby annotation markup <rt> gives an explanation of the Ruby annotation <rp> is what the browser should show if it does not support the <ruby> element
<section>	A generic document or application section
<source>	Used to specify multiple media resources on elements such as <audio> and <video>
<time>	Date and time definition
<video>	Video media content

ATTRIBUTE CHANGES

The following section presents an overview of the attributes that have been added or changed in existing HTML elements.

Global Attributes

The following global attributes have been added for use across a number of elements.

Attribute	Description
contenteditable	Indicates that the element is an editable area
contextmenu	Points to a context menu provided by the author
data-*	All author defined attributes need to be prefixed by **data-**, preventing clashes with future versions of HTML
draggable	Used with the drag & drop Javascript API
hidden	Indicates that an element is not relevant
role aria-*	Instructs assistive technology
spellcheck	Enables spell check indicators if content can be checked

Input Element

The input element's type attribute now has these new attributes: `color`, `date`, `datetime`, `datetime-local`, `email`, `month`, `number`, `range`, `search` and `tel`.

Presentational Elements

HTML5 has removed the use of presentational attributes such as `align`, `background` (for body), `bgcolor` and `border`, as they are better handled in CSS.

Hot Tip

Presentational Elements

HTML5 has removed the use of presentational attributes such as align, background (for body), bgcolor and border, as they are better handled in CSS.

REMOVED ELEMENTS

The following elements have been removed from HTML5 because they are more effectively represented using CSS: `basefont`, `big`, `center`, `font`, `s`, `strike`, `tt` and `u`.

Other elements have been removed because they have a negative effect on usability and accessibility. These include: `frame`, `frameset` and `noframes`.

This last set of elements has been removed due to their lack of frequent use. They also caused confusion at times: `acronym` (use `abbr` for abbreviations), `applet` (`object` replaces its use), `isIndex` and `dir` (use `ul` instead).

CANVAS

`<canvas>`, is probably the most dramatic element to be added to HTML5. It allows graphics on the client's browser to be dynamically updated.

HTML Representation

The `<canvas>` element has the following attributes:

Attribute	Description
height	The height of the canvas, specified in pixels. Default is 150
id	An identifier for the canvas so that it is accessible via Javascript
width	The width of the canvas, specified in pixels. Default is 300

JavaScript API

All dynamic behavior for the canvas is specified in JavaScript. Once you have access to the canvas element, you must create a context in which the canvas can be drawn on.

The following code snippet shows how to access the Canvas and retrieve its graphical context.

```
var canvas = document.getElementById("documentCanvas");
var context = canvas.getContext("2d");
```

Currently the only valid context for the Canvas element is 2D. There will probably be a 3D context in the future. The Context object has the a number of different methods. These include transformations, compositing, state, coloring and shape methods.

Transformations

When creating shapes and paths, transformations are applied to coordinates. Transformations must be performed in reverse order.

Transformation Methods
context.rotate(angle) Changes the transformation matrix to apply a clockwise rotation expressed in radians
context.scale(x,y) Applies a scaling transformation, where x represents scale factor in the horizontal direction and y represents the scale factor in the vertical direction, specified in multiples
context.setTransform(m11, m12, m21, m22, dx, dy) Resets the current transformation matrix to the identity matrix, then the given transformation matrix is applied.
context.tranform(m11, m12, m21, m22, dx, dy) Transforms the context by multiplying the current transformation matrix with the matrix described by `m11 m21 dx m21 m22 dy 0 0 0`
context.translate(x,y) Applies a translation transformation to the context where x is the distance along the horizontal and y is the distance along the vertical, specified in coordinate space units

Context State

A context has a stack of drawing states including:

- The current transformation matrix
- The current clipping region
- Values for a number of attributes

The following methods can be used to manage state:

Context State Methods
context.restore() Pops the top state on the stack and acts as an undo action and restores the context to that state
context.save() Pushes the current state onto the stack

Compositing

All drawing operations are affected by the global compositing values.

Compositing Methods

context.globalAlpha [= value]
Returns or sets the current alpha value applied to rendering operations. Values outside the range 0.0 (fully transparent) to 1.0 (no transparency) are ignored.

context.globalCompositeOperation [=value]
Returns or sets the current composition operation from the list below:

- **copy**
  Display source image instead of destination image
- **destination-atop**
  Display destination image wherever both are opaque. Display source image where source is opaque but destination is transparent
- **destination-in**
  Display destination image wherever both destination and source images are opaque
- **destination-out**
  Display destination image wherever destination image is opaque and source image is transparent
- **destination-over**
  Display destination image wherever destination image is opaque. Source image elsewhere
- **lighter**
  Display the sum of source and destination images
- **source-atop**
  Display source image wherever both images are opaque. Display destination image where destination is opaque but source is transparent
- **source-in**
  Display source image wherever both source image and destination image are opaque
- **source-out**
  Display source image wherever source image is opaque and destination image is transparent
- **source-over (default)**
  Display source image wherever source image is opaque. Destination image elsewhere
- **vendorName-operationName**
  Vendor specific operations follow this format
- **xor**
  Exclusive OR of source and destination image

Colors and Styles

Style Methods

context.fillStyle[=value]
Returns or sets the current style used for filling shapes. Can be a string containing a CSS color, or a `CanvasGradient` /`CanvasPattern`

context.strokeStyle[=value]
Returns or sets the current style used for the stroke of shapes. Can be a string containing a CSS color, or a `CanvasGradient` /`CanvasPattern`

To create the appropriate `CanvasGradient` object use either the `createLinearGradient()` or `createRadialGradient()` from the `Context` object. The resulting `CanvasGradient` object has the following method available:

CanvasGradient Methods

gradient.addColorStop(offset, color)
Adds a new stop to the gradient at a point between 0.0 and 1.0, each representing each end of the gradient. The color parameter must be a CSS color

To create a `CanvasPattern` object use the following method:

CanvasPattern Method

context.createPattern(image, repetition)
Creates a new CanvasPattern object using the given image.The image is repeated in one of these specified directions:

- `repeat`: both directions
- `repeat-x`: horizontal only
- `repeat-y`: vertical only
- `no-repeat`: neither

The image can be either an `img`, `video` or `canvas`.

Line Styles

The following line styles can be applied from the context:

Line Style Methods

context.lineCap[=value]
The ending that will be placed at the end of the line. Either `butt`, `round` or `square`. Initially set to `butt`

context.lineJoin[=value]
The corners that will be placed where two lines meet. Either `bevel`, `round` or `miter`. Initially set to `miter`

context.lineWidth[=value]
Width of the line, in coordinate space units. Initially set to 1.0

context.miterLimit[=value]
Maximum allowed ratio of the miter length to half the line width. Miter length is the distance from the point where the join occurs to the intersection of the line edges on the outside of the join. Initially set to 10.0.

Shadows

There are four global shadow attributes that affect all drawing operations:

Shadow Methods

context.shadowBlur[=value]
The size of the blurring effect, initially set to 0.

context.shadowColor[=value]
The color of the shadow, initially transparent black. Specified as a CSS color.

context.shadowOffsetX[=value]
context.shadowOffsetY[=value]
The distance that a shadow will be offset in horizontal and vertical distance

Shapes

Three methods are available to create simple shapes, or rectangles:

Simple Shape Methods

context.clearRect(x,y,w,h)
Clears the pixels in the specified rectangle that also intersect the current clipping region to fully transparent black, erasing the previous image

context.fillRect(x,y,w,h)
Paints the specified rectangle using the defined fillStyle

context.strokeRect(x,y,w,h)
Draws the outline of the rectangle path using strokeStyle, lineWidth, lineJoin and miterLimit attributes

The context always has a current path, which can have zero or more sub-paths. These are used to create complex shapes.

Complex Shape Methods

context.arc(x,y,radius, startAngle, endAngle, anticlockwise)
Creates an arc described by the circumference of the circle in the arguments, in the given direction

context.arcTo(x1,y1,x2,y2,radius)
Adds a point to the path that is connected to the previous point by an arc

context.beginPath()
Resets the current path

context.bezierCurveTo(cp1x,cp1y,cp2x,cp2y,x,y)
Adds a point to the path that is connected to the previous point by a cubic Bezier curve

context.clip()
Constrains the clipping region to a given path

context.closePath()
Marks the current subpath as closed, and starts a new subpath

context.fill()
Fills the subpaths with the current fill style

context.lineTo(x,y)
Adds a point to the current subpath that is connected to the previous point by a line

context.moveTo(x,y)
Creates a new subpath with the given point as its first and only point

context.quadrativeCurveTo(cpx,cpy,x,y)
Adds a point to the current path that is connected to the previous one by a quadratic Bezier curve

context.rect(x,y,w,h)
Adds a new closed subpath to the path, representing the given rectangle

context.stroke()
Creates the strokes of the subpaths with the current stroke style

context.isPointInPath(x,y)
Returns true if the given point is in the current path

Text

The following attributes and operations are available for manipulating text:

Text Methods
context.font[=value] Returns or sets the current font setting in the CSS `font` property syntax
context.textAlign[=value] Returns or sets the current text alignment settings. Acceptable values are `start` (default), `end`, `left`, `right` or `center`
context.textBaseline[=value] Returns or sets the current baseline alignment settings. Can be one of `top`, `middle`, `hanging`, `alphabetic` (default), `ideographic` or `bottom`
context.fillText(text,x,y,maxWidth) Renders fill for given text at a given position. Text is scaled to fit `maxWidth` if provided
context.strokeText(text, x, y, maxWidth) Renders stroke for given text at a given position. Text is scaled to fit `maxWidth` if provided
metrics = context.measureText(text) Returns a `TextMetrics` object with the metrics of the given text in the current font
metrics.width Returns the advance width of the text that was passed to the `measureText()` method

Images

There are three image functions available to draw an image on the Canvas:

Image Methods
context . drawImage(image, dx, dy) context . drawImage(image, dx, dy, dw, dh) context . drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh) Draws the given image onto the Canvas. The `image` attribute must be one of `<img>`, canvas or `<video>`

In order to manipulate pixels, you must first create an `ImageData` object using one of the following methods:

ImageData Creation Methods
context.createImageData(sw,sh) Returns an `ImageData` object with the given dimensions in CSS pixels
context.createImageData(imagedata) Returns an `ImageData` object with the same dimensions as the `imagedata` parameter
context.getImageData(sx,sy,sw,sh) Returns an `ImageData` object containing the image data for the given rectangle of the canvas

The `ImageData` object has the following properties:

ImageData Properties
imagedata.data A one dimensional array containing the data for the `ImageData` object (as a `CanvasPixelArray`)
imagedata.height imagedata.width Actual dimensions of the data in the `ImageData` object

The following method is used when you want to utilize the `ImageData` object:

ImageData Methods
context.putImageData(imagedata, dx, dy, dirtyX, dirty, dirtyWidth, dirtyHeight) Paints the data of the given `ImageData` object onto the Canvas

MEDIA PLAYBACK

Video

In place of third party plug-ins like Flash, HTML5 provides a standard way to include video with the `<video>` element.

Hot Tip

Video Formats

Currently supported video formats are ogg with Thedora or VP8 video and Vorbis codec, and mpeg4 with H.264 video and AAC audio codec.

The video element has the following attributes:

Attribute	Description
autoplay	Specifies that the video will start playing as soon as the content is loaded. Value of the attribute should be "autoplay"
controls	Specifies play button and volume widget controls will be displayed. Value of the attribute should be "controls"
height	The height of the video player, specified in pixels
loop	Specifies that the video will play in a continuous loop. Value of the attribute should be "loop"
preload	Specifies that the video should load when the page loads. Ignored if autoplay attribute is used. Value of the attribute should be "preload"
src	The URL of the video to play
width	The width of the video player, specified in pixels

The following example shows how to specify a video for a page with two different formats. This is for wider browser support. Note that you can specify a message to display within the video tag for browsers that do not support the tag.

```
<video width="800" height="600" controls="controls"
autoplay="autoplay">
  <source src="http://mysite.com/movie.ogg" type="video/ogg" />
  <source src="http://mysite.com/movie.mp4" type="video/mp4" />
       Your browser does not support the video tag.
</video>
```

Audio

The `<audio>` element allows the embedding of audio media directly into your HTML page, rather than using third party plug-ins.

Hot Tip

Audio Formats

The three supported audio formats are MP3, Wav and Ogg Vorbis.

The audio element has the following attributes:

Attribute	Description
autoplay	Specifies that the video will start playing as soon as the content is loaded. Value of the attribute should be "autoplay"
controls	Specifies that the play button and volume widget controls will be displayed. Value of the attribute should be "controls"
loop	Specifies that the audio will play in a continuous loop. Value of the attribute should be "loop"
preload	Specifies that the audio should load when the page loads. Ignored if autoplay attribute is used. Value of the attribute should be "preload"
src	The URL of the audio file to play

The following example shows how to specify an audio file for an HTML page.

```
<audio controls="controls" autoplay="autoplay">
  <source src="http://mysite.com/audio.mp3" type="audio/mpeg" />
  <source src="http://mysite.com/audio.ogg" type="audio/ogg" />
       Your browser does not support the audio tag.
</video>
```

LOCAL OFFLINE STORAGE

As opposed to using cookies, client data can be stored in two ways—either using longer term localStorage or with single session based sessionStorage. Cookies have traditionally passed data on every server request, but in HTML5 data is passed, only when requested.

To utilize data storage and retrieval you will need to use JavaScript calls.

Session Storage

Session storage should be used only when you require data for a single session. The `sessionStorage` object is used for this. Any attribute can be written to within this object:

Writing the data is as simple as the following:

```
sessionStorage.username = "James"
```

Retrieval of the same attribute:

```
document.write("Welcome back" + sessionStorage.username);
```

Local Storage

Local storage should be used when you have more persistent data with no time limit. Data can be accessed using the same approach shown above using the `localStorage` object.

Hot Tip

Data Access

A site can only access data that it has previously stored. This prevents cases where different websites can access other site's data.

HTML5 EVENT HANDLING

HTML5 defines a number of new event handling attributes. They are listed here.

Media Events

The following events are applicable to all media elements such as `audio`, `video`, `embed` and `img`.

Event Attribute	Trigger
oncanplay	Media can start to play, but might has to stop for buffering
oncanplaythrough	Media doesn't need to wait for buffering
ondurationchange	Length of media has changed
onemptied	Media resource becomes "empty" due to a loss of network connection or other errors
onended	Media has reached the end
onerror	Error occurs during loading of media
onloadeddata	Media has loaded
onloadedmetadata	Meta data of media, such as duration, has loaded
onloadstart	Browser starts to load data
onpause	Media has paused
onplay	Media is about to play
onplaying	Media is has started playing
onprogress	Browser is retrieving media data
onratechange	Playing rate has changed
onreadystatechange	Ready-state changed
onseekend	When seeking has ended (media seeking attribute is no longer true)
onseeking	When seeking has begun (seeking attribute is true)
onstalled	Error retrieving media data
onsuspend	Retrieval of data stopped before completion
ontimeupdate	Media changes playing position
onvolumechange	Media volume changed
onwaiting	Media stopped playing but should resume

Window Events

The following new window events are applicable to the `<body>` element:

Event Attribute	Trigger
onafterprint	Document has printed
onbeforeprint	Document is about to be printed
onbeforeonload	Before the document is loaded
onerror	Error has occurred
onhaschange	Document has a change
onmessage	Message is triggered
onoffline	Document goes offline
ononline	Document goes online
onpagehide	Window is hidden
onpageshow	Window is visible
onpopstate	Window history changes
onredo	Redo is performed
onresize	Window is resized
onstorage	Document loads
onundo	Undo is performed
onunload	User leaves the document

Mouse Events

Mouse events can happen with any element on the page. The main changes include support for drag and drop.

Event Attribute	Trigger
ondrag	Element is dragged
ondragend	Drag operation is completed
ondragenter	Element is dragged to a valid drop target
ondragleave	Element leaves drop target
ondragover	Element dragged over drop target
ondragstart	Drag operation has begun
ondrop	Element is dropped onto drop target
onmousewheel	Mouse wheel is rotated
onscroll	Element's scrollbar is being scrolled

Form Events

The following new event attributes are applicable to form elements, but can be used across other elements:

Event Attribute	Trigger
oncontextmenu	Context menu is invoked
onformchange	Form changes
onforminput	User inputs data into form
oninput	Element gets user input
oninvalid	Element is invalid

GEOLOCATION

The Geolocation API allows you to share your location with trusted websites. This is achieved by retrieving the longitude (long) and latitude through JavaScript.

The global navigator object has access to the Geolocation object, which has the following API:

Geolocation Interface
getCurrentPosition(in PositionCallback successCallback, in optional PositionErrorCallback errorCallback, in optional PositionOptions options) Asynchronously attempts to obtain the current location of the device. If successful, the successCallback is invoked with a new Position object
long watchPosition(in PositionCallback successCallback, in optional PositionErrorCallback errorCallback, in optional PositionOptions options) Immediately returns a long representing the id of a watch operation, and asynchronously handles the watch operation which tracks the position of the device
clearWatch (in long watchId) Stops the watch with the supplied identifier. If no such watch exists it simply returns

The `PositionOptions` object has three attributes:

Attribute	Description
enableHighAccuracy	A hint that the application would like the best possible results, which may lead to slower response times or higher power consumption
timeout	Maximum length of time in milliseconds that is allowed to pass from the call to getCurrentPosition or watchPosition until the successCallback is invoked
maximumAge	The maximum age of a cached position that the application will accept

The `Position` object has two attributes:

Attribute	Description
coords	The geographic coordinates along with their associated accuracy. The Coordinates object has attributes (as double) for latitude, longitude, altitude, heading, speed, accuracy, and altitudeAccuracy
timestamp	The time at which the Position was acquired

The `PositionCallback` and `PositionErrorCallback` interfaces have one `handleEvent` method each. `PositionCallback` accepts a `Position`, while `PositionErrorCallback` accepts a `PositionError`.

BROWSER SUPPORT FOR HTML5 FEATURES

The table below illustrates when each major browser gained support for key HTML5 features. The numbers indicate the browser's version when support for each HTML5 feature began.

Feature	IE	Firefox	Safari	Opera	Chrome
contenteditable	6.0+	3.5+	3.2+	10.1+	5.0+
canvas	9.0+	3.0+	3.2+	10.1+	5.0+
offline storage	8.0+	3.0+	4.0+	10.5+	5.0+
audio	9.0+	3.5+	4.0+	10.1+	5.0+
video	9.0+	3.5+	3.2+	10.5+	5.0+
Canvas Text API	9.0	3.5+	4.0+	10.5+	5.0+
Drag and Drop	6.0	3.5+	4.0+	-	5.0+
Geolocation	9.0+	3.5+	5.0+	10.6+	5.0+

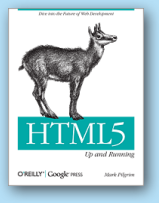
Compatability

http://caniuse.com and http://html5rocks.com are excellent resources to view the compatibility of HTML5 features across a range of browsers.

ABOUT THE AUTHOR

James Sugrue has been editor at both Javalobby and Eclipse Zone for over two years, and loves every minute of it. By day, James is a software architect at Pilz Ireland, developing killer desktop software using Java and Eclipse all the way. While working on desktop technologies such as Eclipse RCP and Swing, James also likes meddling with up-and-coming technologies such as Eclipse e4. His current obsession is developing for the iPhone and iPad, having convinced himself that it's a turning point for the software industry.

RECOMMENDED BOOK



If you don't know about the new features available in HTML5, now's the time to find out. This book provides practical information about how and why the latest version of this markup language will significantly change the way you develop for the Web. *HTML5: Up & Running* carefully guides you though the important changes in this version with lots of hands-on examples, including markup, graphics, and screenshots. You'll learn how to use HTML5 markup to add video, offline capabilities, and more—and you'll be able to put that functionality to work right away.

Browse our collection of over 100 Free Cheat Sheets

Free PDF

Upcoming Refcardz
OSMF
Clojure
HTML 5
Test Driven Development

DZone

DZone communities deliver over 6 million pages each month to more than 3.3 million software developers, architects and decision makers. DZone offers something for everyone, including news, tutorials, cheat sheets, blogs, feature articles, source code and more. **"DZone is a developer's dream,"** says PC Magazine.

DZone, Inc.
140 Preston Executive Dr.
Suite 100
Cary, NC 27513

888.678.0399
919.678.0300

Refcardz Feedback Welcome
refcardz@dzone.com

Sponsorship Opportunities
sales@dzone.com

ISBN-13: 978-1-936502-03-5
ISBN-10: 1-936502-03-8
50795
9 781936 502035
$7.95

Version 1.0

Copyright © 2010 DZone, Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.