HTML 5

Author : Thanh C Tran

Date: 07/09/2013

Objectives

- HTML Basic – an Overview
- HTML 5

Course Timetable

- Duration: 180 minutes
- Break: 15 minutes

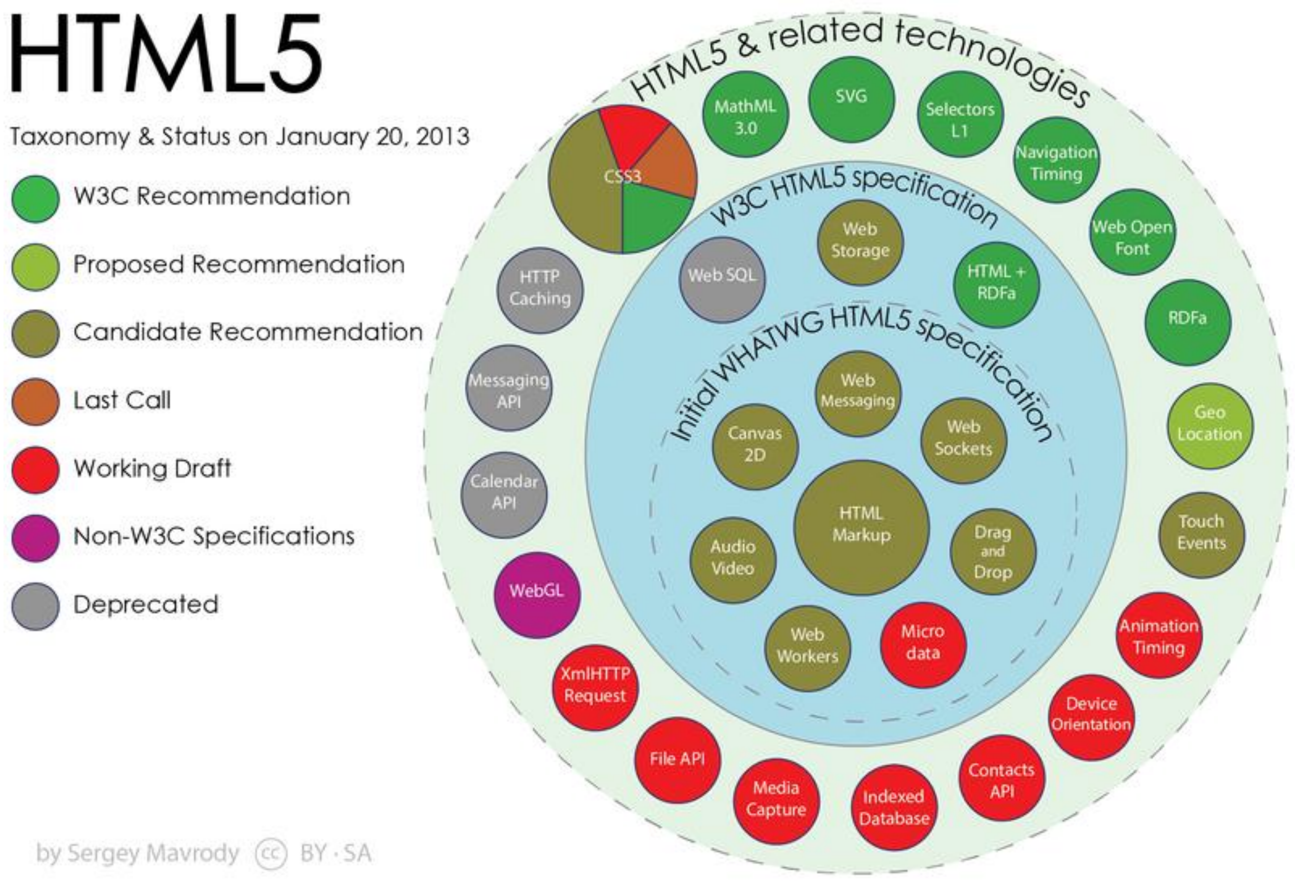
Prerequisites

- HTML
- CSS
- JavaScript

Agenda

HTML 5

- Introduction
- New Elements
- Canvas
- SVG
- Drag/Drop
- Geolocation
- Video and Audio
- Input Types
- Form Elements & Attributes
- Sematic
- Web Storage
- App Cache
- Web Workers


HTML5
Taxonomy & Status on January 20, 2013
W3C Recommendation
Proposed Recommendation
Candidate Recommendation
Last Call
Working Draft
Non-W3C Specifications
Deprecated
HTML5 & related technologies
W3C HTML5 specification
Initial WHATWG HTML5 specification
CSS3
MathML 3.0
SVG
Selectors L1
Navigation Timing
Web Open Font
RDFa
Geo Location
Touch Events
Animation Timing
Device Orientation
Contacts API
Indexed Database
Media Capture
File API
XmlHTTP Request
WebGL
Calendar API
Messaging API
HTTP Caching
Web SQL
Web Storage
HTML + RDFa
Web Messaging
Canvas 2D
Web Sockets
HTML Markup
Audio Video
Drag and Drop
Web Workers
Micro data
by Sergey Mavrody (cc) BY · SA

HTML5 - Introduction

What is HTML5?

- HTML5 is a new standard for HTML
- HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).
- Some rules for HTML5 were established:
  - New features should be based on HTML, CSS, DOM, and JavaScript
  - Reduce the need for external plugins (like Flash)
  - Better error handling
  - More markup to replace scripting
  - HTML5 should be device independent
  - The development process should be visible to the public

HTML5 - Introduction

What’s new?

- DOCTYPE declaration
  - It becomes shorter and simpler as below:

  `<!DOCTYPE html>`

- New Features:
  - The <canvas> element for 2D drawing
  - The <video> and <audio> elements for media playback
  - Support for local storage
  - New content-specific elements, like <article>, <footer>, <header>, <nav>, <section>
  - New form controls, like calendar, date, time, email, url, search

HTML5 - New Elements

Overview

- The internet, and the use of the internet, has changed a lot since HTML 4.01 became a standard in 1999.
- Today, several elements in HTML 4.01 are obsolete, never used, or not used the way they were intended. All those elements are removed or re-written in HTML5.
- To better handle today's internet use, HTML5 also includes new elements for drawing graphics, adding media content, better page structure, better form handling, and several APIs to drag/drop elements, find Geolocation, include web storage, application cache, web workers, etc.

HTML5 - New Elements – cont.

- The new <canvas> element
- New Media elements:
  - <audio>
  - <video>
  - <source>
  - <embed>
  - <track>
- New Form elements:
  - <datalist>
  - <keygen>
  - <output>
- New Semantic/Structural elements:
  - <article>
  - <aside>
  - <bdi>
  - <command>
  - ...

HTML5 - New Elements – cont.

- Removed Elements:
  - <acronym>
  - <applet>
  - <basefont>
  - <big>
  - <center>
  - <dir>
  - <font>
  - <frame>
  - <frameset>
  - <noframes>
  - <strike>
  - <tt>

HTML5 - Canvas

- A canvas is a rectangular area on an HTML page, and it is specified with the <canvas> element.
- The HTML5 <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, characters, and adding images.
- Create a canvas

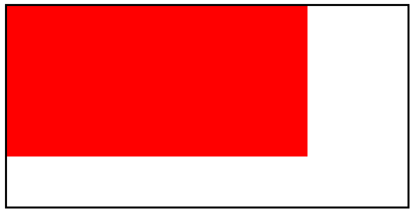
```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

- 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.

HTML5 – Canvas – cont.

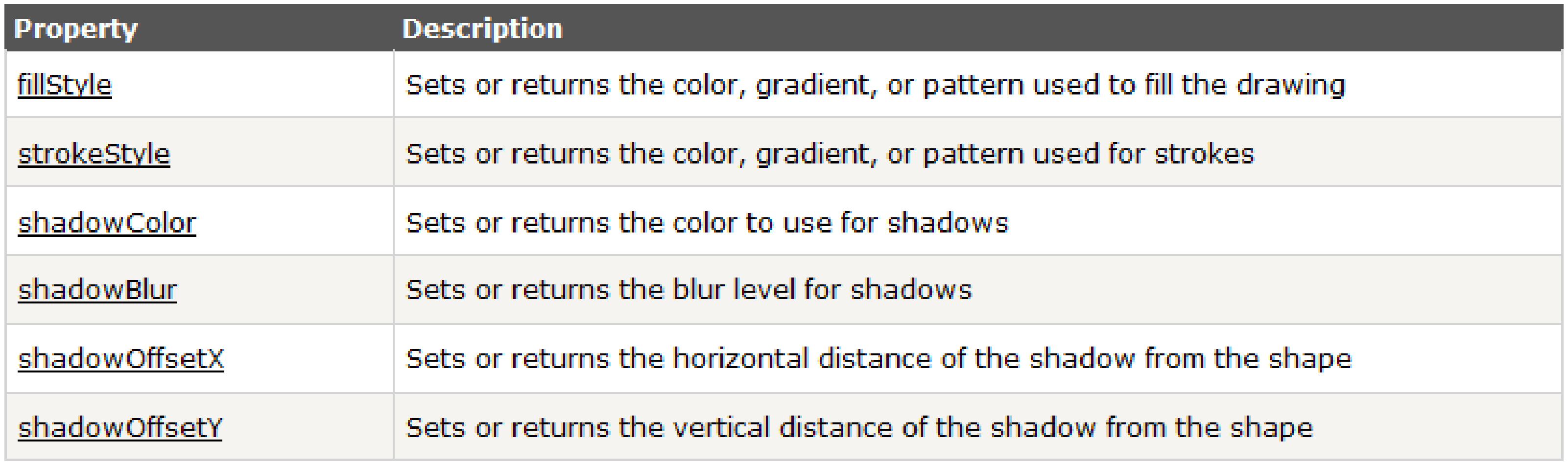
Example

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML5 - Canvas</title>
</head>
<body>
    <canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
        Your browser does not support the HTML5 canvas tag.
    </canvas>
    <script>
        var c=document.getElementById("myCanvas");
        var ctx=c.getContext("2d");
        ctx.fillStyle="#FF0000";
        ctx.fillRect(0,0,150,75);
    </script>
</body>
</html>
```

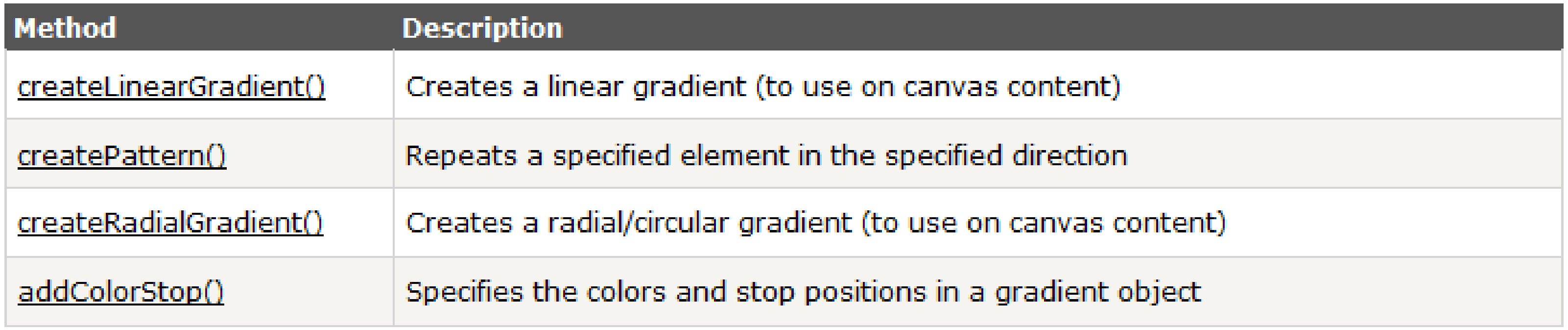


HTML5 – Canvas – cont.

- Colors, Styles, and Shadows



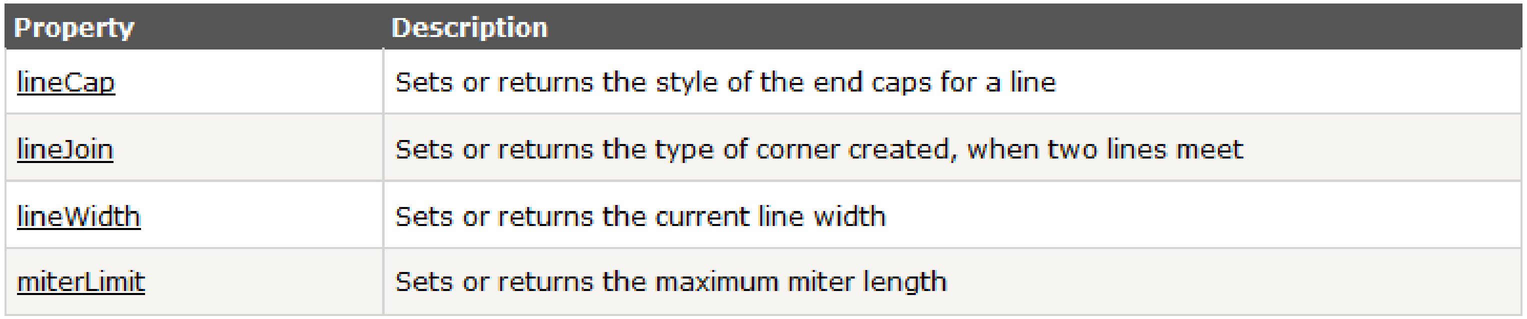
Property	Description
fillStyle	Sets or returns the color, gradient, or pattern used to fill the drawing
strokeStyle	Sets or returns the color, gradient, or pattern used for strokes
shadowColor	Sets or returns the color to use for shadows
shadowBlur	Sets or returns the blur level for shadows
shadowOffsetX	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

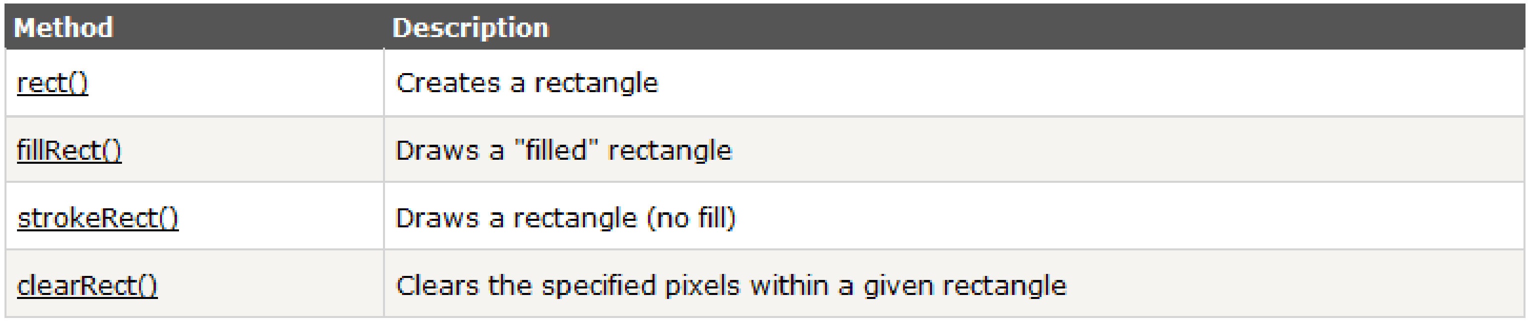
HTML5 – Canvas – cont.

- 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

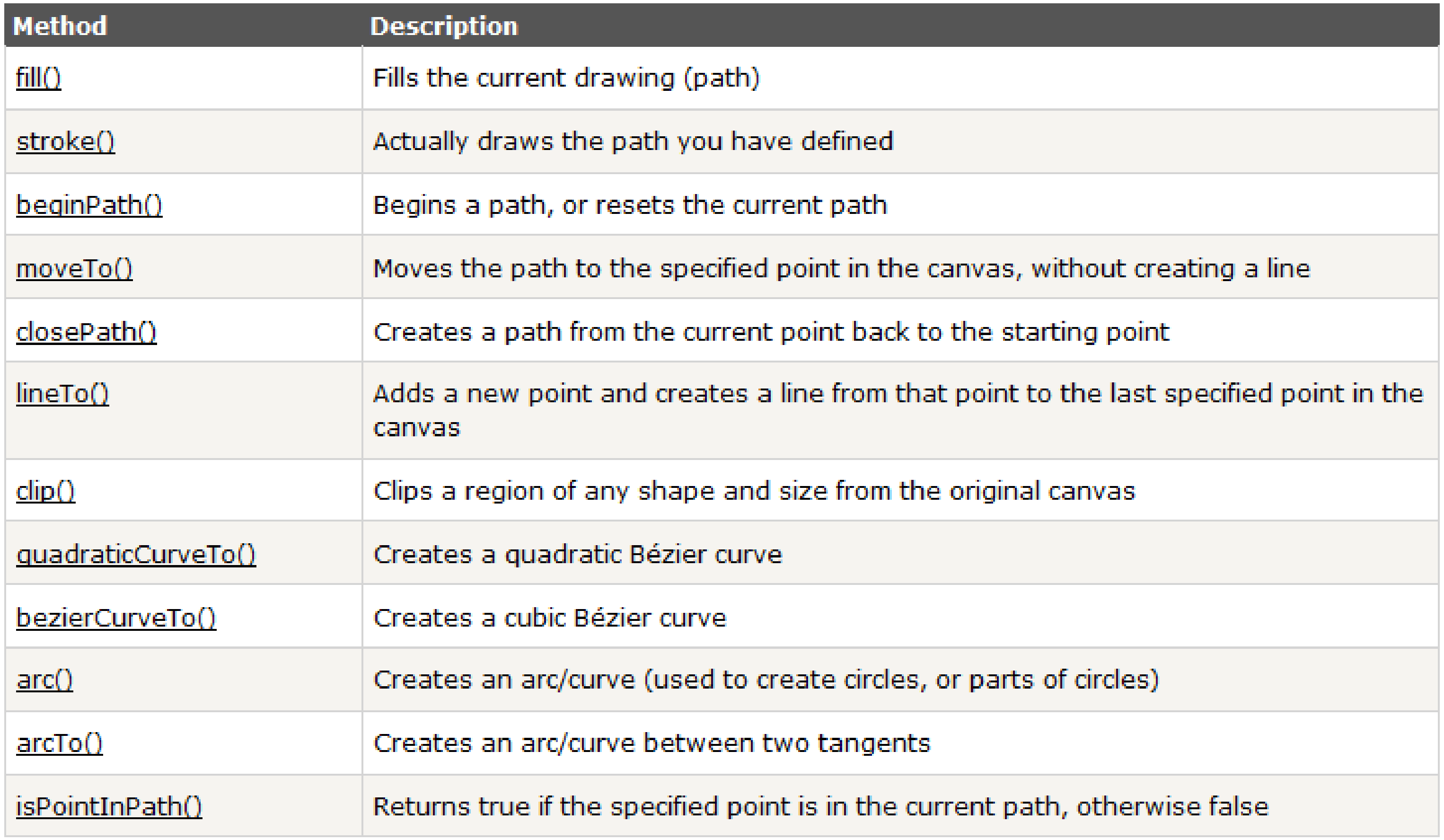
- Rectangles



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

HTML5 – Canvas – cont.

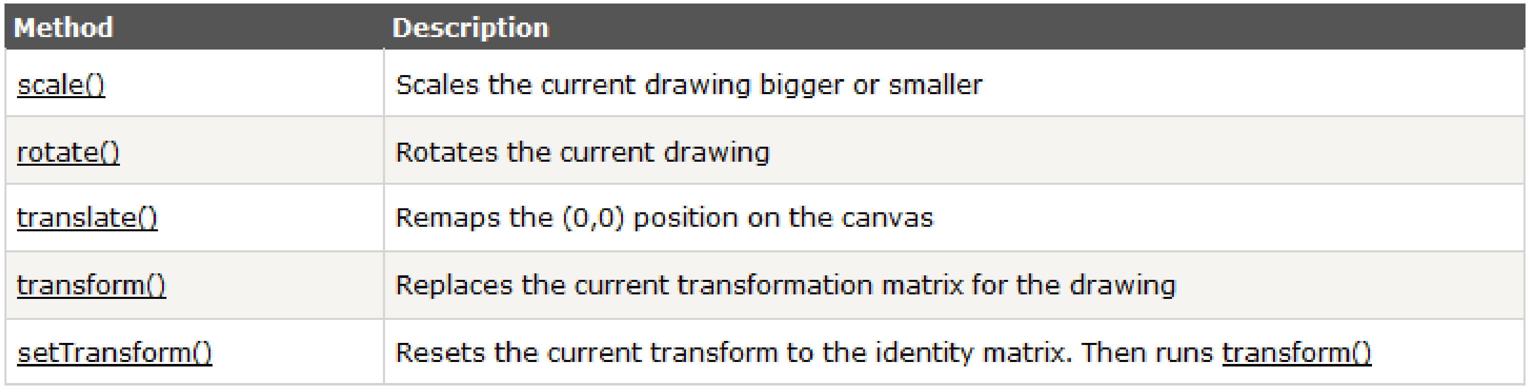
- Paths



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 from that point to 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

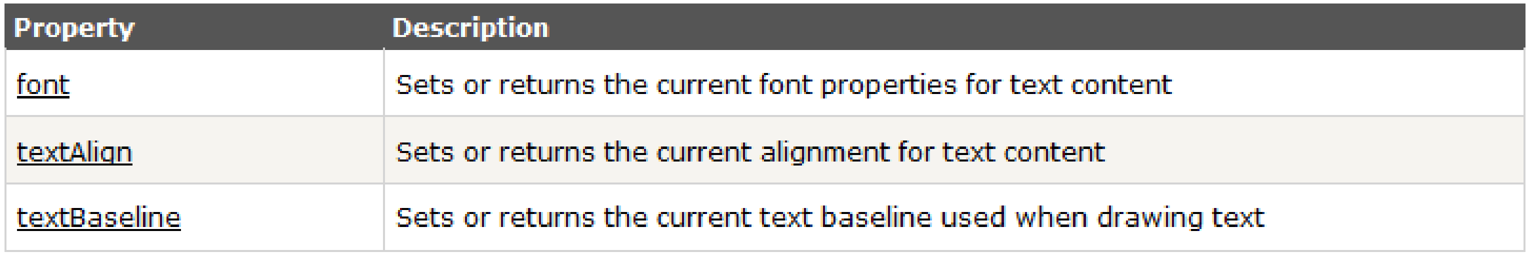
HTML5 – Canvas – cont.

- Transformations

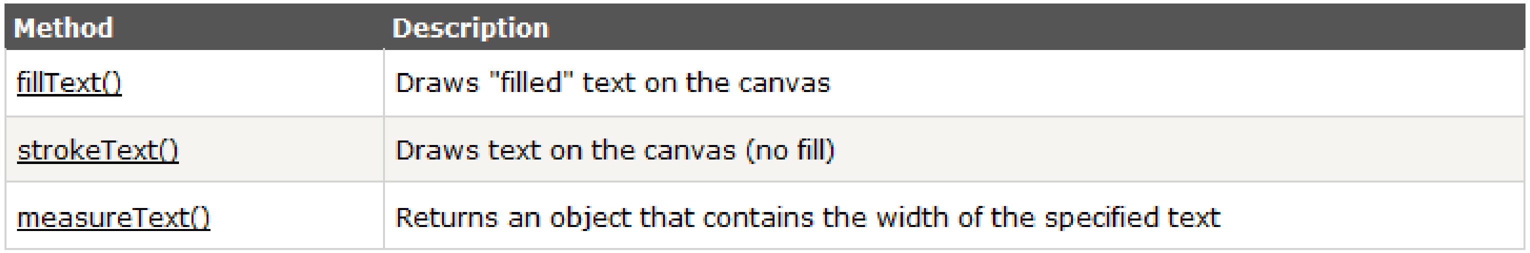


Method	Description
scale()	Scales the current drawing bigger or smaller
rotate()	Rotates the current drawing
translate()	Remaps the (0,0) position on the canvas
transform()	Replaces the current transformation matrix for the drawing
setTransform()	Resets the current transform to the identity matrix. Then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

- Text



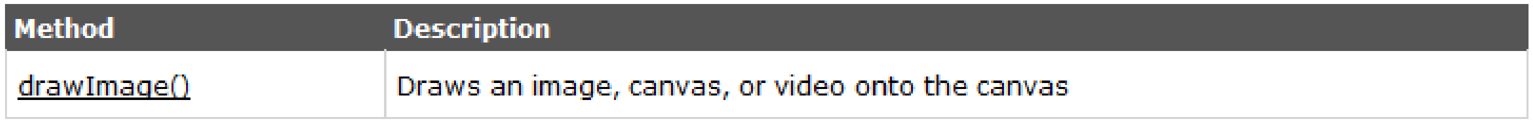
Property	Description
font	Sets or returns the current font properties for text content
textAlign	Sets or returns the current alignment for text content
textBaseline	Sets or returns the current text baseline used when drawing text



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

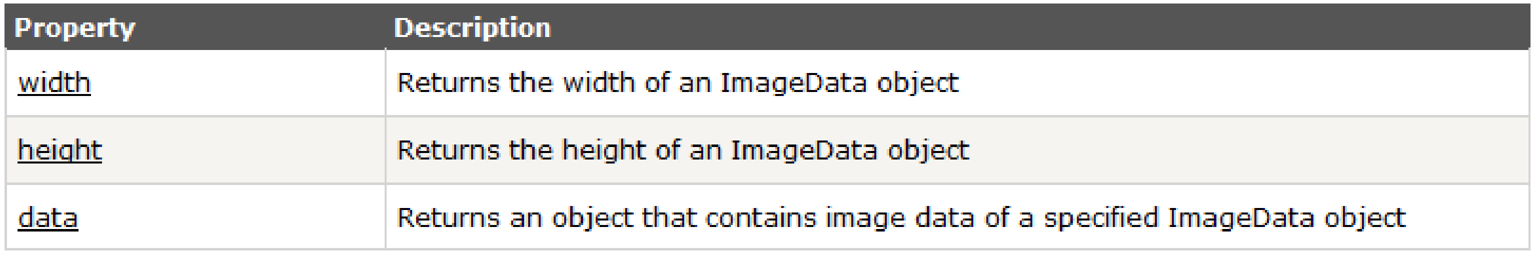
HTML5 – Canvas – cont.

- Image Drawing

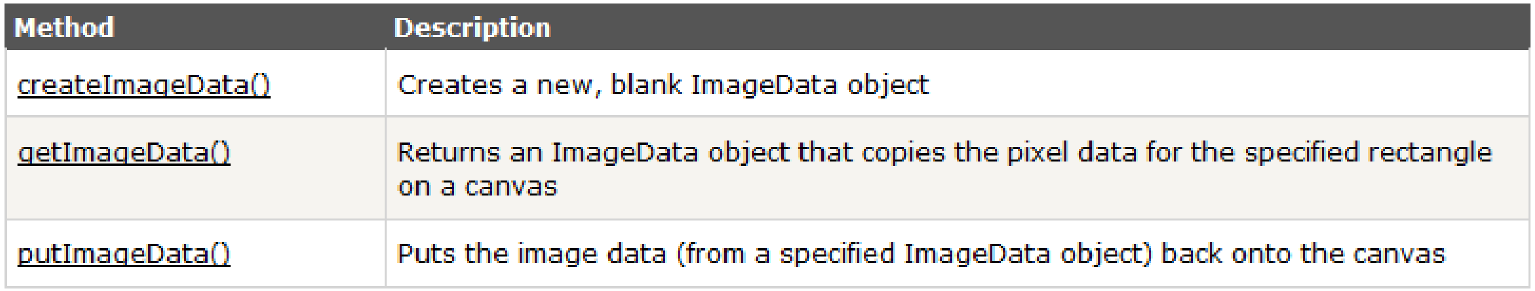


Method	Description
drawImage()	Draws an image, canvas, or video onto the canvas

- Pixel Manipulation



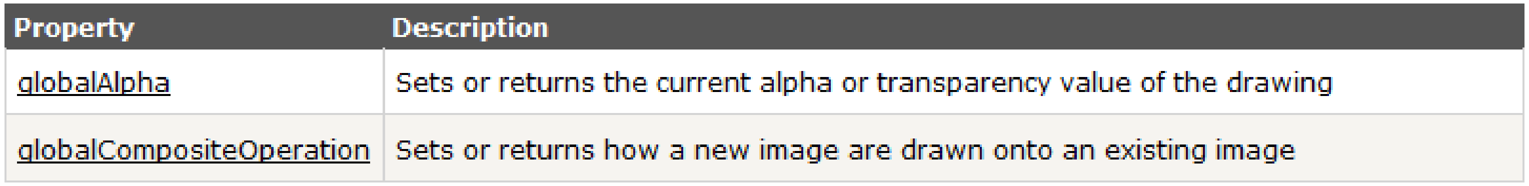
Property	Description
width	Returns the width of an ImageData object
height	Returns the height of an ImageData object
data	Returns an object that contains image data of a specified ImageData object



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

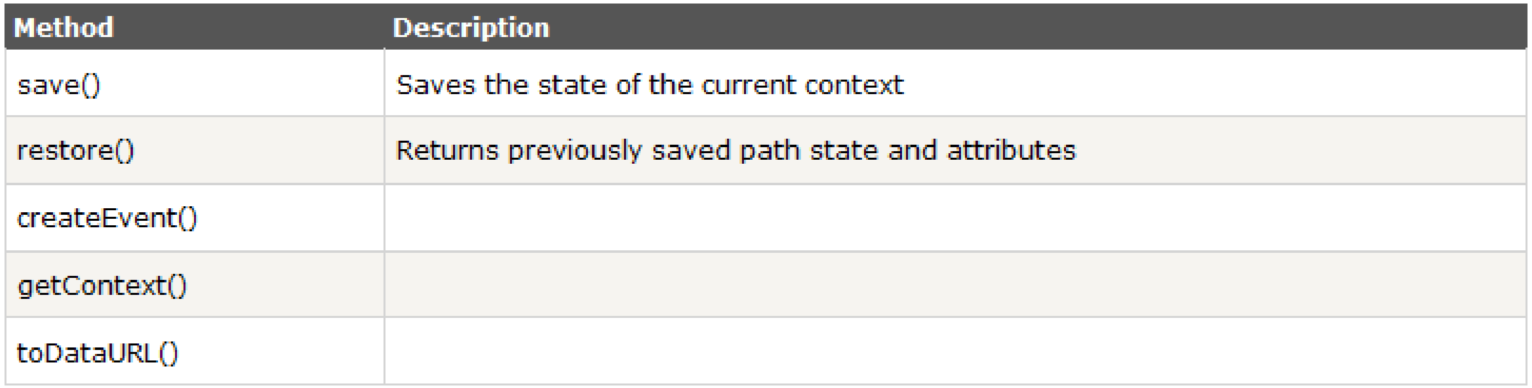
HTML5 – Canvas – cont.

- 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()	

HTML5 – Inline SVG

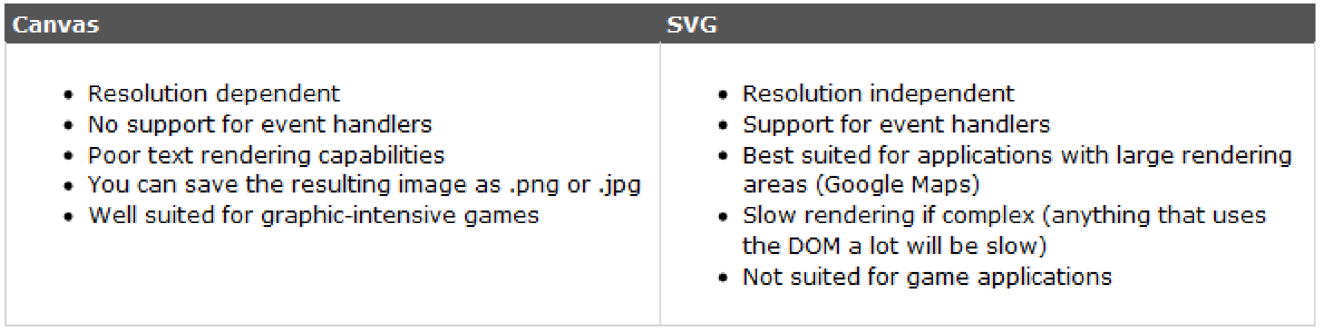
- SVG stands for **Scalable Vector Graphics.**
- SVG is used to define vector-based graphics for the Web.
- SVG defines the graphics in XML format.
- SVG graphics do NOT lose any quality if they are zoomed or resized.
- Every element and every attribute in SVG files can be animated.
- SVG is a W3C recommendation.
- Advantages of using SVG over other image formats (like JPEG and GIF) are:
  - SVG images can be created and edited with any text editor
  - SVG images can be searched, indexed, scripted, and compressed
  - SVG images are scalable
  - SVG images can be printed with high quality at any resolution
  - SVG images are zoomable (and the image can be zoomed without degradation)

HTML5 – Inline SVG – cont.

- In HTML5, you can embed SVG elements directly into your HTML page:

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1" height="190">
  <polygon points="100,10 40,180 190,60 10,60 160,180"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;">
</svg>
```

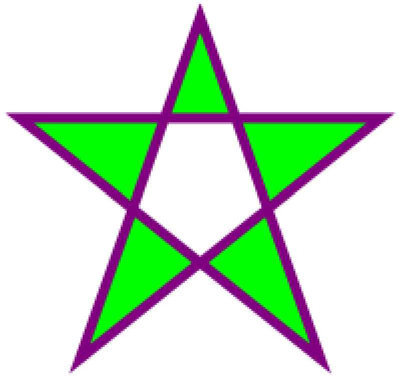
- Differences Between SVG and Canvas:
  - SVG is a language for describing 2D graphics in XML.
  - Canvas draws 2D graphics, on the fly (with a JavaScript).
  - SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element.
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



Canvas	SVG
• Resolution dependent • No support for event handlers • Poor text rendering capabilities • You can save the resulting image as .png or .jpg • Well suited for graphic-intensive games	• Resolution independent • Support for event handlers • Best suited for applications with large rendering areas (Google Maps) • Slow rendering if complex (anything that uses the DOM a lot will be slow) • Not suited for game applications

HTML5 – Inline SVG – cont.

```
<!DOCTYPE html>
<html>
<body>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1" height="190">
  <polygon points="100,10 40,180 190,60 10,60 160,180"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;">
</svg>
</body>
</html>
```



HTML5 – Drag and Drop

- Drag and drop is a part of the HTML5 standard.
- Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.
- In HTML5, drag and drop is part of the standard, and any element can be draggable.
- Browser Support:
  - Internet Explorer 9+
  - Firefox
  - Opera
  - Chrome
  - and Safari

HTML5 – Drag and Drop

- Make an Element Draggable
  - First of all: To make an element draggable, set the draggable attribute to true:

```
<img draggable="true">
```

- What to Drag, uses ondragstart and setData()
  - Then, specify what should happen when the element is dragged. The dataTransfer.setData() method sets the data type and the value of the dragged data:

```
function drag(ev) {
  ev.dataTransfer.setData("Text",ev.target.id);
}
```

- Where to Drop, ondragover
  - The ondragover event specifies where the dragged data can be dropped. By default, data/elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element.

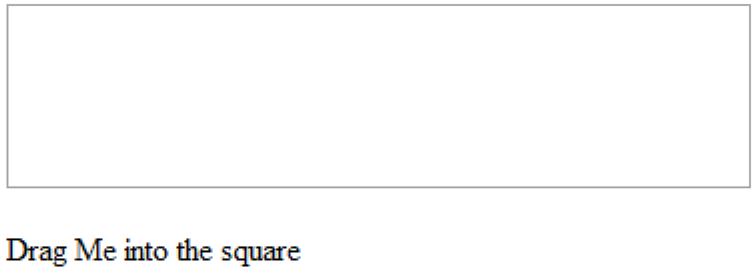
```
event.preventDefault()
```

- Do the Drop, ondrop
  - When the dragged data is dropped, a drop event occurs.

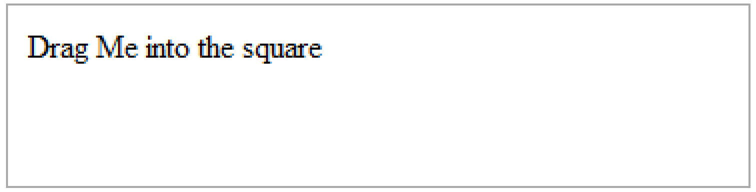
HTML5 – Drag and Drop

Example

```
<!DOCTYPE HTML>
<html><head>
<style type="text/css">
#div1 {width:350px;height:70px;padding:10px;border:1px solid #aaaaaa;}
</style>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}
function drag(ev) {
  ev.dataTransfer.setData("Text",ev.target.id);
}
function drop(ev) {
  ev.preventDefault();
  var data=ev.dataTransfer.getData("Text");
  ev.target.appendChild(document.getElementById(data));
}
</script></head>
<body>
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<br>
<div id="drag1" draggable="true" ondragstart="drag(event)" width="336"
height="69">Drag Me into the square</div>
</body></html>
```



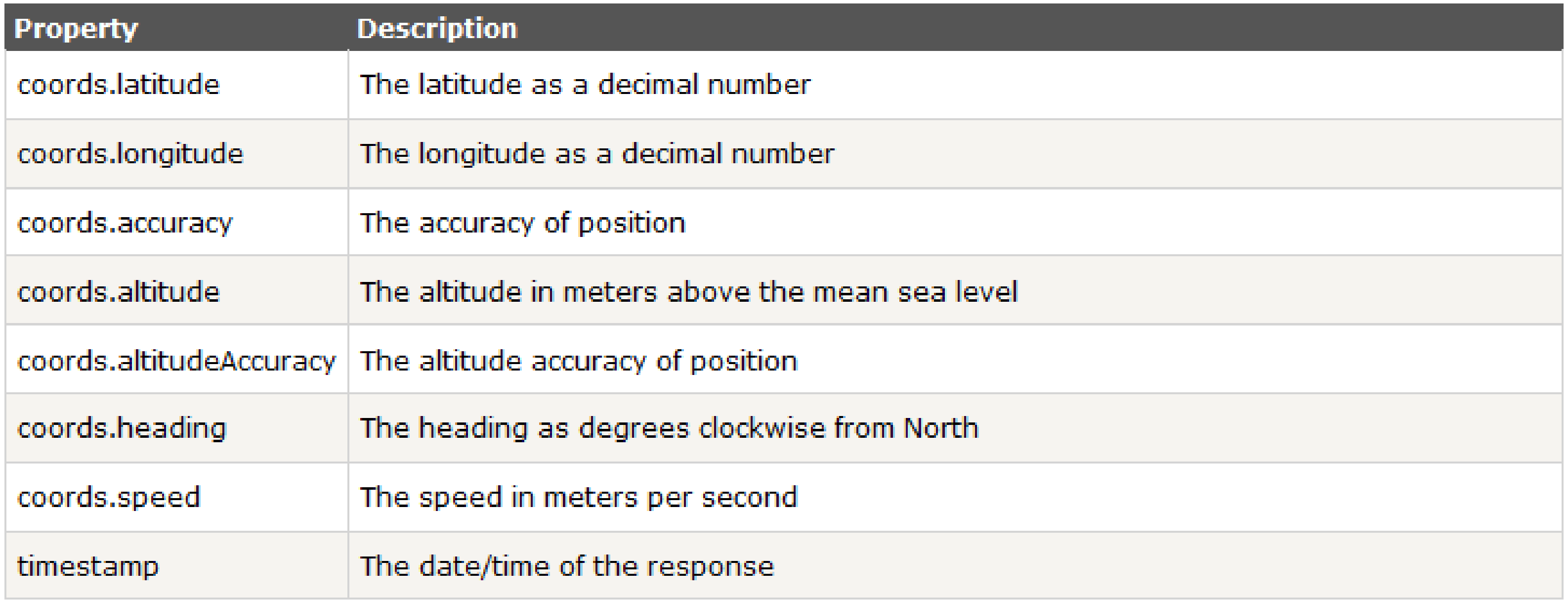
Drag Me into the square



Drag Me into the square

HTML5 – Geolocation

- The HTML5 Geolocation API is used to get the geographical position of a user.
- Since this can compromise user privacy, the position is not available unless the user approves it.
- Using Geolocation
  - Use the getCurrentPosition() method to get the user's position. The getCurrentPosition() method returns an object if it is successful. The latitude, longitude and accuracy properties are always returned. The other properties below are returned if available.



Property	Description
coords.latitude	The latitude as a decimal number
coords.longitude	The longitude as a decimal number
coords.accuracy	The accuracy of position
coords.altitude	The altitude in meters above the mean sea level
coords.altitudeAccuracy	The altitude accuracy of position
coords.heading	The heading as degrees clockwise from North
coords.speed	The speed in meters per second
timestamp	The date/time of the response

HTML5 – Geolocation – cont.

- Handling Errors and Rejections, the second parameter of the getCurrentPosition() method is used to handle errors. It specifies a function to run if it fails to get the user's location:

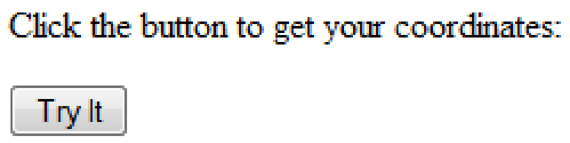
```
function showError(error)  {
   switch(error.code) {
      case error.PERMISSION_DENIED:
         x.innerHTML="User denied the request for Geolocation."
         break;
      case error.POSITION_UNAVAILABLE:
          x.innerHTML="Location information is unavailable."
          break;
      case error.TIMEOUT:
           x.innerHTML="The request to get user location timed out."
           break;
      case error.UNKNOWN_ERROR:
            x.innerHTML="An unknown error occurred."
            break;
   }
}
```

HTML5 – Geolocation – cont.

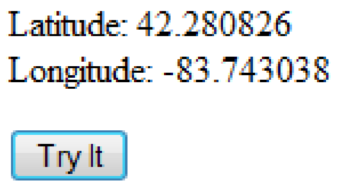
- Other interesting Methods:
  - watchPosition() - Returns the current position of the user and continues to return updated position as the user moves (like the GPS in a car).
  - clearWatch() - Stops the watchPosition() method.
- Example:

```
<script>
   var x=document.getElementById("demo");
      function getLocation() {
         if (navigator.geolocation) {
            navigator.geolocation.watchPosition(showPosition);
         } else{x.innerHTML="Geolocation is not supported by this browser.";}
      }
      function showPosition(position) {
         x.innerHTML="Latitude: " + position.coords.latitude +   "<br>Longitude: " +
position.coords.longitude;
      }
</script>
```

HTML5 – Geolocation – cont.



Click the button to get your coordinates:



Latitude: 42.280826
Longitude: -83.743038

```
<!DOCTYPE html>
<html>
<body>
  <p id="demo">Click the button to get your coordinates:</p>
  <button onclick="getLocation()">Try It</button>
  <script>
    var x=document.getElementById("demo");
    function getLocation() {
      If (navigator.geolocation) {
navigator.geolocation.getCurrentPosition(showPosition,showErr
or);
      } else{x.innerHTML="Geolocation is not supported by this
browser.";}
    }
    function showPosition(position)  {
      x.innerHTML="Latitude: " + position.coords.latitude +
      "<br>Longitude: " + position.coords.longitude;
    }
```

```
 function showError(error) {
        switch(error.code) {
          case error.PERMISSION_DENIED:
            x.innerHTML="User denied the request for
Geolocation."
            break;
          case error.POSITION_UNAVAILABLE:
            x.innerHTML="Location information is unavailable."
            break;
          case error.TIMEOUT:
             x.innerHTML="The request to get user location
timed out."
            break;
          case error.UNKNOWN_ERROR:
             x.innerHTML="An unknown error occurred."
             break;
        }
    }
</script>
</body>
</html>
```

HTML5 – Video & Audio

- Until now, there has not been a standard for showing a video/movie/audio on a web page.
- Today, most videos are shown through a plug-in (like flash). However, different browsers may have different plug-ins.
- HTML5 defines a new element which specifies a standard way to embed a video/movie on a web page: the <video> element. For audio, we have <audio> element.
- To show a video in HTML5, this is all you need:

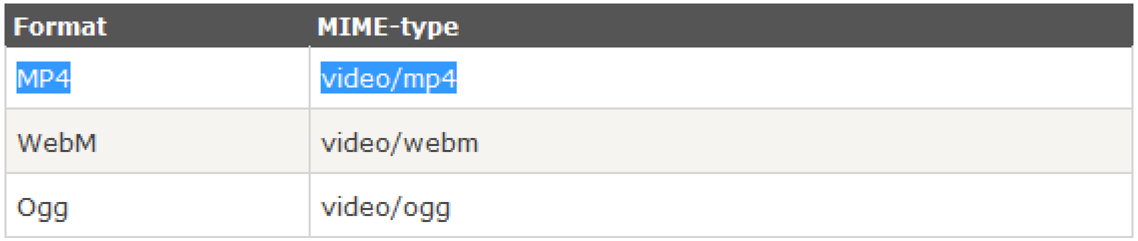
```
<video width="320" height="240" controls>
   <source src="movie.mp4" type="video/mp4">
   <source src="movie.ogg" type="video/ogg">
   Your browser does not support the video tag.
</video>
```

- To play an audio file in HTML5, this is all you need:

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

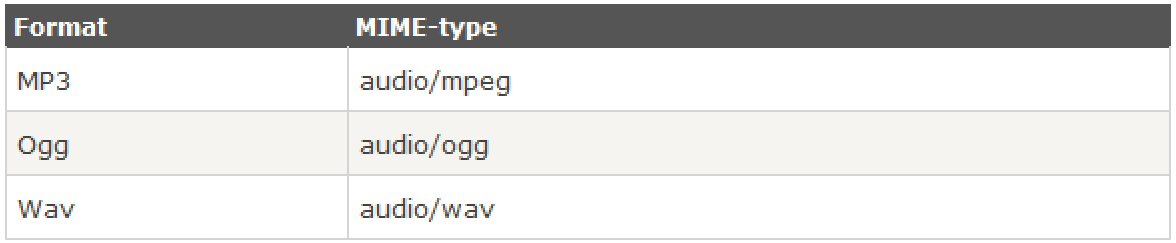
HTML5 – Video & Audio – cont.

- MIME Types for Video Formats



Format	MIME-type
MP4	video/mp4
WebM	video/webm
Ogg	video/ogg

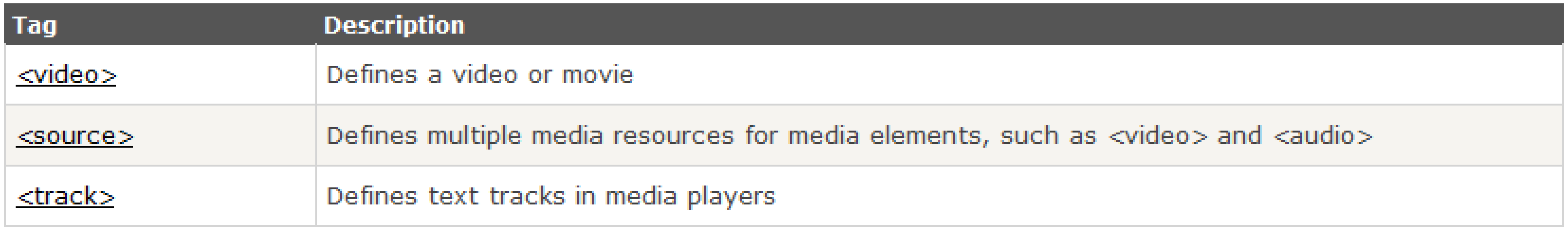
- MIME Types for Audio Formats



Format	MIME-type
MP3	audio/mpeg
Ogg	audio/ogg
Wav	audio/wav

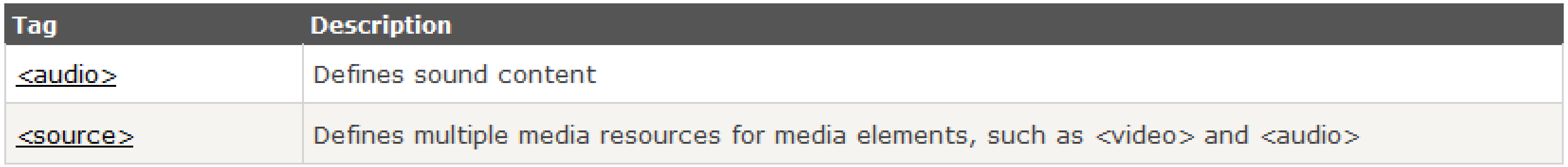
HTML5 – Video & Audio – cont.

- Video Tags



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

- Audio Tags



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

HTML5 – Input Types

- HTML5 has several new input types for forms. These new features allow better input control and validation.
  - color, the color type is used for input fields that should contain a color:

    `<input type="color" name="favcolor">`
  - date, the date type allows the user to select a date:

    `<input type="date" name="bday">`
  - datetime, the datatime allows the user to select date and time (with time zone):

    `<input type="datetime" name="bdaytime">`
  - datetime-local, allows the user to select a data and time (without time zone)

    `<input type="datetime-local" name="bdaytime">`
  - email, is used for the input fields that should contain an email address

    `<input type="email" name="email">`
  - month, allows the user to select a month and year.

    `<input type="month" name="bdaymonth">`
  - number, is used for input fields that should contain a numeric value. We can also set restrictions on what numbers are accepted:

    `<input type="number" name="quantity" min="1" max="5">`

Note: *Not all major browsers support all the new input types. However, you can already start using them; If they are not supported, they will behave as regular text fields.*

HTML5 – Input Types – cont.

- search, is used for search fields (a search field behaves like a regular text field).

  `<input type="search" name="googlesearch">`
- tel, define a field for entering a telephone number:

  `<input type="tel" name="usrtel">`
- time, allows the user to select a time:

  `<input type="time" name="usr_time">`
- url, is used for input fields that should contain a URL address. The value of the url field is automatically validated when the form is submitted:

  `<input type="url" name="homepage">`
- week, allows the user to select a week and year:

  `<input type="week" name="week_year">`
- range, is used for input fields that should contain a value from a range of numbers. We can also set restrictions on what numbers are accepted:

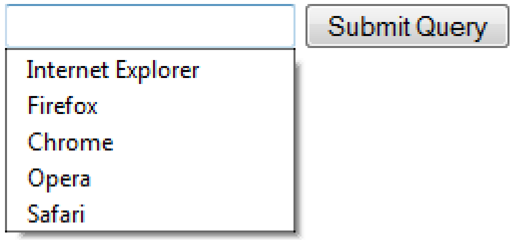
  `<input type="range" name="points" min="1" max="10">`

Note: *Not all major browsers support all the new input types. However, you can already start using them; If they are not supported, they will behave as regular text fields.*

HTML5 – Form elements

- <datalist> Element,
  - specifies a list of pre-defined options for an <input> element.
  - is used to provide an "autocomplete" feature on <input> elements. Users will see a drop-down list of pre-defined options as they input data.
  - Use the <input> element's list attribute to bind it together with a <datalist> element.

```
<input list="browsers">
<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```



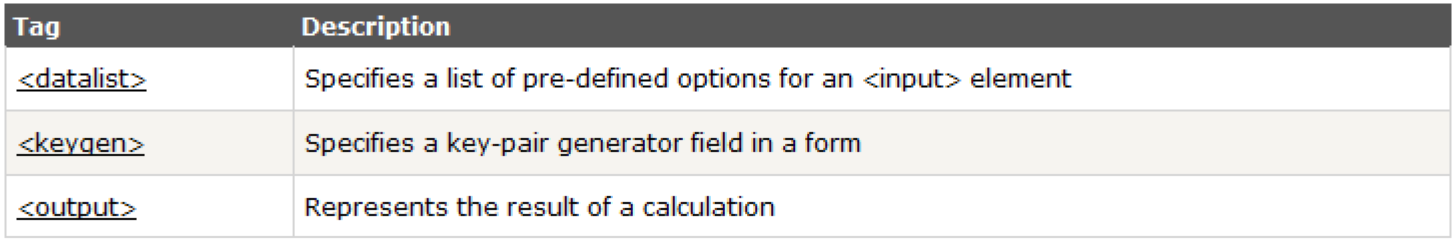
- <keygen> Element
  - The purpose of the <keygen> element is to provide a secure way to authenticate users.
  - The <keygen> tag specifies a key-pair generator field in a form.
  - When the form is submitted, two keys are generated, one private and one public.
  - The private key is stored locally, and the public key is sent to the server. The public key could be used to generate a client certificate to authenticate the user in the future.

```
<form action="demo_keygen.jsp" method="get">
      Username: <input type="text" name="usr_name">
      Encryption: <keygen name="security">
     <input type="submit">
 </form>
```

HTML5 – Form elements – cont.

- **<output> Element**
  - The <output> element represents the result of a calculation (like one performed by a script).

```
<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>
```



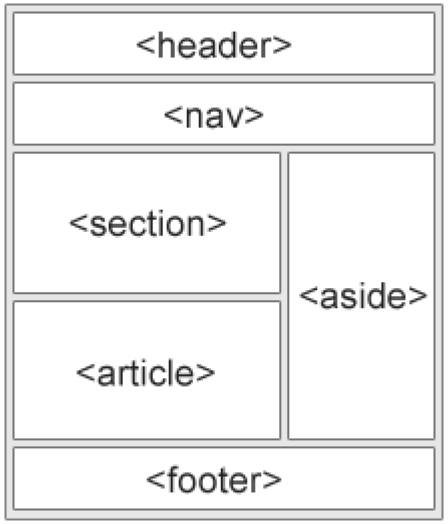
Tag	Description
<datalist>	Specifies a list of pre-defined options for an <input> element
<keygen>	Specifies a key-pair generator field in a form
<output>	Represents the result of a calculation

HTML5 – Form attributes

- HTML5 has several new attributes for <form> and <input>.
- New attributes for <form>:
  - autocomplete
  - novalidate
- New attributes for <input>:
  - autocomplete
  - autofocus
  - form
  - formaction
  - formenctype
  - formmethod
  - formnovalidate
  - formtarget
  - height and width
  - list
  - min and max
  - multiple
  - pattern (regexp)
  - placeholder
  - required
  - step

HTML5 – Semantic elements

- A semantic element clearly describes its meaning to both the browser and the developer:
  - Examples of **non-semantic** elements: <div> and <span> - Tells nothing about its content.
  - Examples of **semantic** elements: <form>, <table>, and <img> - Clearly defines its content.
- Many of existing web sites today contains HTML code like this: <div id="nav">, <div class="header">, or <div id="footer">, to indicate navigation links, header, and footer. HTML5 offers new semantic elements to clearly define different parts of a web page:
  - <header>
  - <nav>
  - <section>
  - <article>
  - <aside>
  - <figcaption>
  - <figure>
  - <footer>



HTML5 – Semantic elements

- <section> Element, defines a section in a document.

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

- <article> Element, specifies independent, self-contained content. An article should make sense on its own and it should be possible to distribute it independently from the rest of the web site.

```
<article>
  <h1>Internet Explorer 9</h1>
  <p>Windows Internet Explorer 9 (abbreviated as IE9) was released to
  the  public on March 14, 2011 at 21:00 PDT.....</p>
</article>
```

- <nav> Element, defines a set of navigation links. The <nav> element is intended for large blocks of navigation links. However, not all links in a document should be inside a <nav> element!

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

HTML5 – Semantic elements

- <aside> Element, defines some content aside from the content it is placed in (like a sidebar). The aside content should be related to the surrounding content.

```
<aside>
  <h4>Epcot Center</h4>
  <p>The Epcot Center is a theme park in Disney World, Florida.</p>
</aside>
```

- <header> Element, specifies a header for a document or section. It should be used as a container for introductory content. We can have several <header> elements in one document.

```
<article>
  <header>
    <h1>Internet Explorer 9</h1>
    <p><time pubdate datetime="2011-03-15"></time></p>
  </header>
  <p>Windows Internet Explorer 9 (abbreviated as IE9) was released to
  the  public on March 14, 2011 at 21:00 PDT.....</p>
</article>
```

- <footer> Element, specifies a footer for a document or section. It should contain information about its containing element. We can have several <footer> elements in one document.

```
<footer>
  <p>Posted by: Hege Refsnes</p>
  <p><time pubdate datetime="2012-03-01"></time></p>
</footer>
```

HTML5 – Semantic elements

- <figure> and <figcaption> Elements,
  - The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
  - While the content of the <figure> element is related to the main flow, its position is independent of the main flow, and if removed it should not affect the flow of the document.
  - The <figcaption> tag defines a caption for a <figure> element.
  - The <figcaption> element can be placed as the first or last child of the <figure> element.

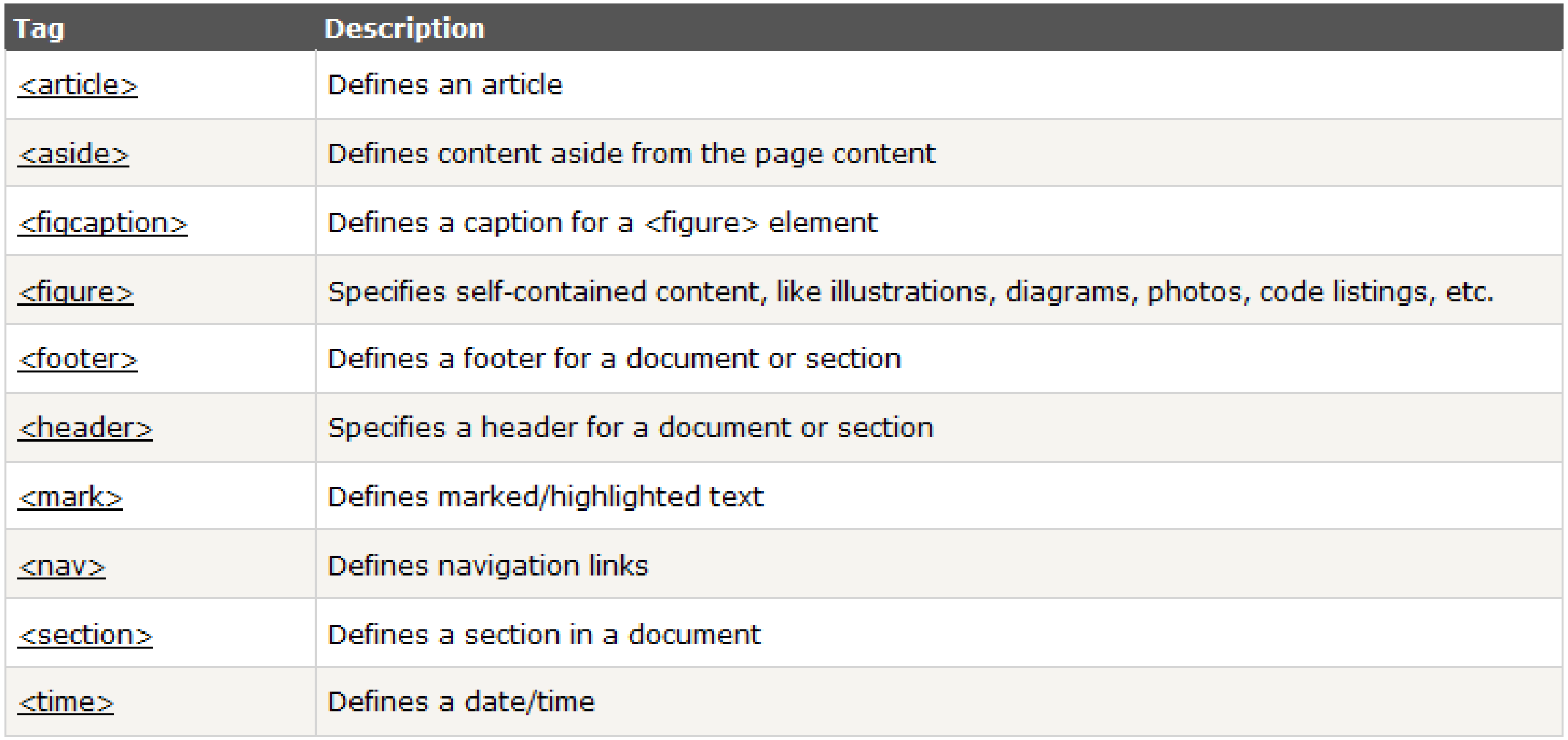
```
<figure>
  <img src="img_pulpit.jpg" alt="The Pulpit Rock" width="304" height="228">
  <figcaption>Fig1. - The Pulpit Pock, Norway.</figcaption>
</figure>
```

HTML5 – Semantic elements

- The following browsers support
  - Internet Explorer 9+
  - Firefox
  - Chrome
  - Safari
  - Opera
- But Internet Explorer 8 and earlier does not support these elements. However, there is a solution for handling this by the checking as below:

```
<!--[if lt IE 9]>
<script src="html5shiv.js"></script>
<![endif]-->
```

HTML5 – Semantic elements



Tag	Description
<article>	Defines an article
<aside>	Defines content aside from the page content
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<time>	Defines a date/time

HTML5 – Web Storage

- HTML5 web storage, a better local storage than cookies
- With HTML5, web pages can store data locally within the user's browser.
- Earlier, this was done with cookies. However, Web Storage is more secure and faster. The data is not included with every server request, but used ONLY when asked for. It is also possible to store large amounts of data, without affecting the website's performance.
- The data is stored in key/value pairs, and a web page can only access data stored by itself.
- Web storage is supported in Internet Explorer 8+, Firefox, Opera, Chrome, and Safari. Internet Explorer 7 and earlier versions, do not support web storage.
- There are two new objects for storing data on the client:
  - localStorage, stores data with no expiration date
  - sessionStorage, stores data for one session
  - Before using web storage, check browser support for localStorage and sessionStorage:

```
if(typeof(Storage)!=="undefined")   {
   // Yes! localStorage and sessionStorage support!
   // Some code.....
} else {
   // Sorry! No web storage support..
}
```

HTML5 – Web Storage

- **The localStorage Object,** The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

```
localStorage.lastname="Smith";
document.getElementById("result").innerHTML="Last name: “+ localStorage.lastname;
```

- **The sessionStorage Object,** The sessionStorage object is equal to the localStorage object, **except** that it stores the data for only one session. The data is deleted when the user closes the browser window.

```
if (sessionStorage.clickcount) {
  sessionStorage.clickcount=Number(sessionStorage.clickcount)+1;
} else {
  sessionStorage.clickcount=1;
}
document.getElementById("result").innerHTML="You have clicked the button " +
sessionStorage.clickcount + " time(s) in this session.";
```

HTML5 – Application Cache

- HTML5 introduces application cache, which means that a web application is cached, and accessible without an internet connection.
- Application cache gives an application three advantages:
  - Offline browsing - users can use the application when they're offline
  - Speed - cached resources load faster
  - Reduced server load - the browser will only download updated/changed resources from the server
- **Cache Manifest Basics,** to enable application cache, we include the manifest attribute in the document's <html> tag as below:

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

  - Every page with the manifest attribute specified will be cached when the user visits it. If the manifest attribute is not specified, the page will not be cached (unless the page is specified directly in the manifest file).
  - The recommended file extension for manifest files is: ".appcache"

HTML5 – Application Cache

- **The Manifest File,** the manifest file is a simple text file, which tells the browser what to cache (and what to never cache). The manifest file has three sections:
  - CACHE MANIFEST, this is the first line and is required.

```
CACHE MANIFEST
/theme.css
/common.js
```

  - NETWORK,
    - this section specifies that the resources should never be cached, and will not be available offline:

```
NETWORK:
login.jsp
```

    - An asterisk can be used to indicate that all other resources/files require an internet connection

```
NETWORK:
*
```

  - FALLBACK, is the section specifies that resources will be served in case an internet connection cannot be established:

```
FALLBACK:
/html/ /offline.html
```

The first URL is resources and the second is the fallback.

HTML5 – Application Cache

- **Updating the Cache,** Once an application is cached, it remains cached until one of the following happens
  - The user clears the browser's cache
  - The manifest file is modified (see tip below)
  - The application cache is programmatically updated
- **An Example:**

```
CACHE MANIFEST
/theme.css
/common.js

NETWORK:
login.jsp

FALLBACK:
/html/ /offline.html
```

HTML5 – Web Workers

- When executing scripts in an HTML page, the page becomes unresponsive until the script is finished.
- A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.
- Check Web Worker Support
  - Before creating a web worker, check whether the user's browser supports it:

```
if(typeof(Worker)!=="undefined")  {
  // Yes! Web worker support!
  // Some code.....
} else {
  // Sorry! No Web Worker support..
}
```

- Create a Web Worker File

```
var i=0;
function timedCount() {
   i=i+1;
   postMessage(i);
   setTimeout("timedCount()",500);
}
timedCount();
```

The important part of the code above is the **postMessage()** method - which is used to posts a message back to the HTML page.

HTML5 – Web Workers

- Create a Web Worker Object

```
if(typeof(w)=="undefined")  {
  w=new Worker("workers.js");
}

w.onmessage=function(event) {
   document.getElementById("result").innerHTML=event.data;
};

w.terminate();
```

HTML5 – Web Workers

```
var w;
function startWorker() {
   if(typeof(Worker)!=="undefined") {
      if(typeof(w)=="undefined")     {
         w=new Worker("workers.js");
      }
      w.onmessage = function (event) {
          document.getElementById("result").innerHTML=event.data;
      };
   } else {
      document.getElementById("result").innerHTML="Sorry, your browser does not
support Web Workers...";
   }
}

function stopWorker() {
   w.terminate();
}
```

Questions & Answers

CSC

Thanks for Your Attention