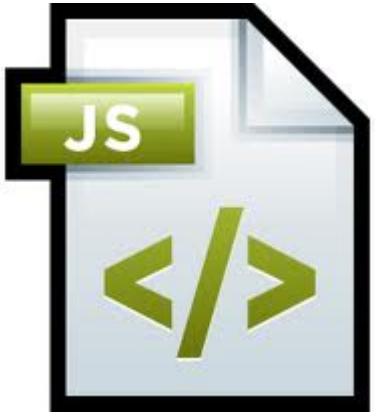


HTML5 storage

HTML5 Forms

Geolocation

Event listeners

canvas element

video and audio element

HTML5 Storage 1

- A way for web pages to store named key/value pairs locally, within the client web browser
  - You can store up to about 5 MB of data
  - Everything is stored as strings
- Like cookies, this data persists even after you navigate away from the web site
- Unlike cookies, this data is never transmitted to the remote web server
- Properties and methods belong to the window object
- sessionStorage work as localStorage but for the session

```
localStorage[key] = value;              // set
localStorage.setItem(key, value);       // alternative
var text = localStorage[key];           // get
var text = localStorage.getItem(key); // alternative
console.log(localStorage.key(0));       // get key name
localStorage.removeItem(key);           // remove
localStorage.clear(key);      // clears the entire storage
```

local-session-storage.html

HTML5 Storage 2

- Web SQL Database (formerly known as “WebDB”) provides a thin wrapper around a SQL database, allowing you to do things like this from JavaScript

```
function testWebSQLDatabase() {
     openDatabase('documents', '1.0', 'Local document storage', 5*1024*1024, function (db) {
       db.changeVersion('', '1.0', function (t) {
         t.executeSql('CREATE TABLE docids (id, name)');
          //t.executeSql('drop docids');
       }, error);
     });
}
```

localstorage.html

Not working - check: http://html5demos.com/ for working sample

- As you can see, most of the action resides in the string you pass to the executeSql method. This string can be any supported SQL statement, including SELECT, UPDATE, INSERT, and DELETE statements. It’s just like backend database programming, except you’re doing it from JavaScript!
- A competing Web DB standard is IndexedDB which uses a non-SQL API to something called ***object store*** - demos are only available this far

HTML5 Forms

- There are basically five areas of improvements when it comes to form features in HTML5
- New input types as email, url, number, search, date etc.
  - The great thing about input types too is that if it's not supported in the web browser, it will default back to a regular <input type="text"> element: no errors, no multiple code versions to handle!
- New attributes as: placeholder, autofocus, required
- New elements as: meter, progress
- Form validation is always on! Works on <input type=
  - email, url, number and its min and max attributes
- APIs, such as the File API
- http://robertnyman.com/2011/08/16/html5-forms-input-types-attributes-and-new-elements-demos-tips-and-tricks/

13 new form input types

- Demos in the following files: html5forms-attributes.html, html5forms-elements.html, html5forms-input-types.html

```
<input type="color" value="#b97a57">

<input type="date" value="2011-06-08">

<input type="datetime"
     value="2011-06-09T20:35:34.32">

<input type="datetime-local"
     value="2011-06-09T22:41">

<input type="email"
     value="robert@robertnyman.com">

<input type="month" value="2011-06">

<input type="number" value="4">

<input type="range" value="15">
<!-- Note: If not set, 'default
attribute values are min="0",
max="100", step="1". -->
```


```
<input type="search"
     value="[Any search text]">

<!-- Note: In WebKit-based web browsers
(Google Chrome, Safari) you can add the
non-standard results attribute to get a
looking glass icon to click to see the
latest searches, and the attribute autosave
to keep them across page loads. -->

<input type="tel" value="[Any numeric value]">

<!-- Note: Most web browsers seem to let
through any value at this time. -->

<input type="time" value="22:38">

<input type="url" value="http://robertnyman.com">

<!-- Note: requires a protocol like
http://, ftp:// etc in the beginning. -->

<input type="week" value="2011-W24">
```

HTML5 form input attributes 1

- autocomplete
  - An option to turn off automatic form completion of values for a field. Possible values are “on” and “off”.
- **autofocus**
  - Whether focus should be set to this field as soon as it has loaded.
- formaction
  - For buttons that submit a form (e.g. <input type=”submit”>, <input type=”image”> and <button> elements) to be able to override the action attribute of the form; for instance if different buttons should submit the form to different URLs. No more JavaScript to do this!
- formenctype
  - For buttons that submit a form to be able to override the form’s specified encoding
- formmethod
  - For buttons that submit a form to be able to override the form’s method attribute, in case a button should change the method.

HTML5 form input attributes 2

- formnovalidate
  - Append to a submit button to bypass form validation.
- formtarget
  - For buttons that submit a form to be able to override the form's target attribute.
- list
  - To connect with a <datalist> element by its id, to use its <option> elements as suggestions.
- max
  - Maximum value for the value that can be put in.
- min
  - Minimum value for the value that can be put in.
- multiple
  - Allows for selection of multiple files for <input type="file"> elements, and for multiple e-mail addresses separated by a comma.

HTML5 form input attributes 3

- **novalidate**
  - Applies only to the <form> element, and prevents a form from being validated before submitted.
- **pattern**
  - Declaring what pattern should be used for validating a field's value, in the form of a regular expression.
- **placeholder**
  - Meant to be able to display a hint to the end user what to input. Disappears when field is entered

- readonly
  - If a field should be readonly.
- **required**
  - For validation purposes, if a field is required or not.


```
<form>
  <input name="q" required>
  <input type="submit" value="Go">
</form>
```

- spellcheck
  - Lets the web browser know if it should spell check the contents or not.
- step
  - Possibility to control the size of each step for elements that accepts number or date input.

HTML5 form elements

- datalist
  - Contains a number of <option> elements with values that can be used as suggestions for other form elements through the usage of the list attribute on them.
- keygen
  - Offers a way to create a public/private key pair where the public key is sent with the form. (Ok, I'll be honest – probably not the new element that will get you the most excited… Also it seems like Internet Explorer are not interested in implementing it either)
- meter
  - The meter element is for displaying values on a bar, where you can custom control min, max and assigned value. You can also specify low, high and optimum to set up different kind of areas of the bar.
- output
  - Dedicated to output the result of a calculation in the page, for instance sliding a <input type="range"> back and forth.
- progress
  - Meant to be used to indicate progress of any kind in a web page, for instance file upload progress.

Geolocation API 1

- Geolocation is the art of figuring out where you are in the world - one can use
  - Your IP address, wireless network connection, which cell tower your phone is talking to or dedicated GPS hardware to know it
- Geolocation support is an opt-in (user cannot be forced to share location)
  - User get a question like this

geolocation.html


```
// show_map and handle_error arguments are callback functions
// the optional third is a PositionOptions object with three properties,
// time is in ms {enableHighAccuracy: false, timeout: 15000, maximumAge: 120000}
// The call to getCurrentPosition() will return immediately
function get_location() {
    navigator.geolocation.getCurrentPosition(show_map, handle_error);
}
```

Geolocation API 2 geolocation.html

- Getting the callback and receiving a position object
- Example - geolocation.html
- To enhance it with maps - check the developers guide at
  - http://code.google.com/apis/maps/documentation/javascript/

```
function show_map(position) {
      var latitude = position.coords.latitude;
      var longitude = position.coords.longitude;
      // let's show a map or do something interesting!
}
```

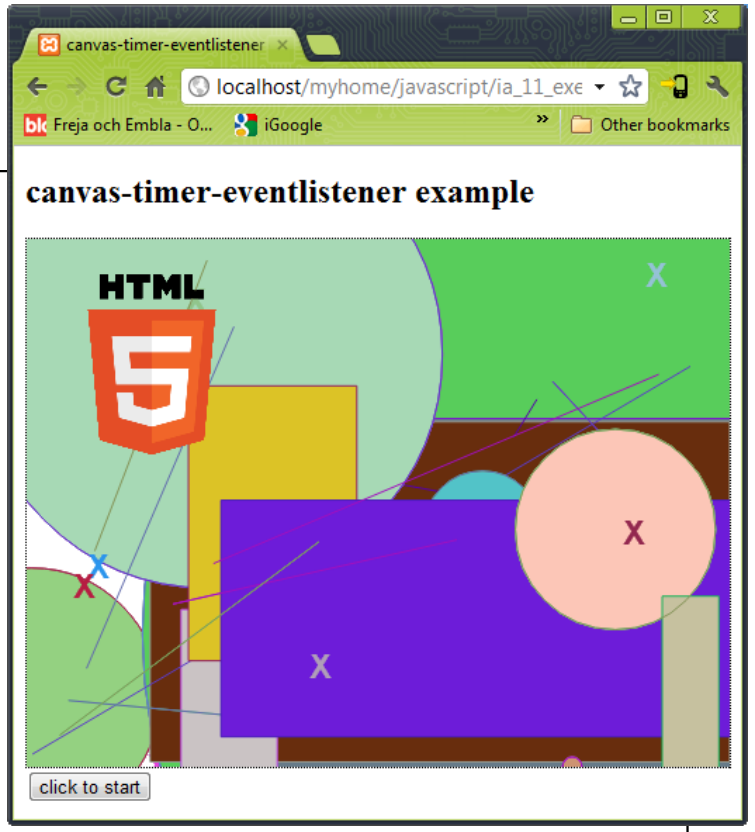
```
function handle_error(err) {
      if (err.code == 1) {
            // user said no!
      }
}
```

POSITION OBJECT

Property	Type	Notes
coords.latitude	double	decimal degrees
coords.longitude	double	decimal degrees
coords.altitude	double or null	meters above the reference ellipsoid
coords.accuracy	double	meters
coords.altitudeAccuracy	double or null	meters
coords.heading	double or null	degrees clockwise from true north
coords.speed	double or null	meters/second
timestamp	DOMTimeStamp	like a Date() object

Canvas element 1

- Good intro tutorial in chapter 4 - Dive Into HTML5
- One can have more than one canvas on a page
- X, Y (0, 0) is in upper left corner
- Reset: g_canvas.width = g_canvas.width;



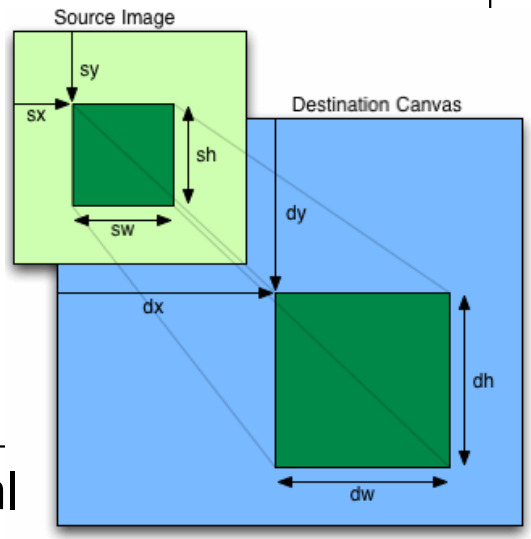

```
<head>
<script>
      var g_canvas, g_context;
      function drawRect(x, y, w, h) {
            g_canvas = document.getElementById("cvs1");
            g_context = g_canvas.getContext("2d");
            g_context.fillStyle = "#FF0000";
            b_context.fillRect(x, y, w, h);
            // color the rect edges
            g_context.strokeStyle = "#0000FF";
            // just draws the rect edges
            g_context.strokeRect(x, y, w, h);
      }
</script>
</head>

<body onload="drawRect();">
      <canvas id="cvs1" width="500" height="375" style="border: 1px dotted;"></canvas>
</body>
```

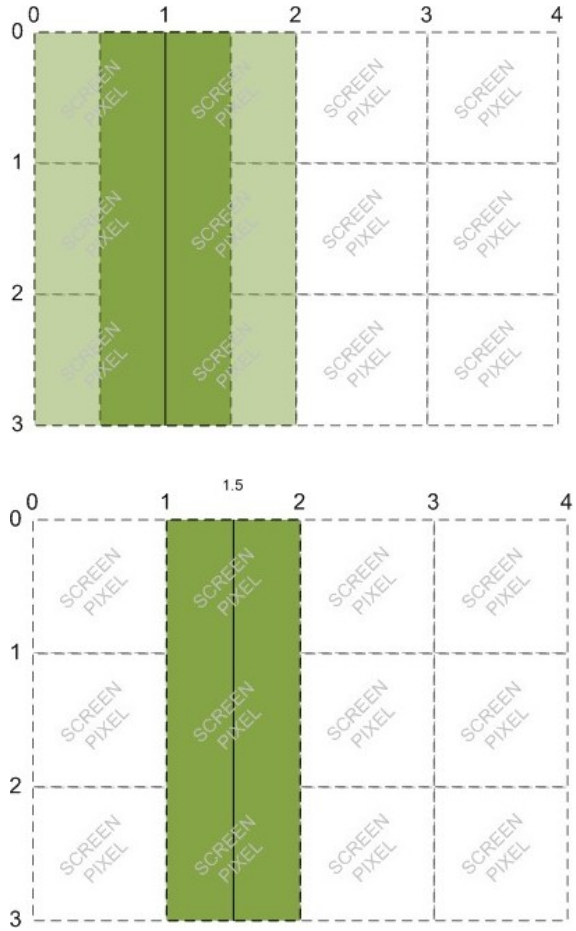
canvas-timer-eventlistener.html

Canvas element 2

- Drawing paths (lines) with width one pixel
- drawLine(1, 0, 1, 3) vs. drawLine(1.5, 0, 1.5, 3)
- Images are possible to draw as well

```
// draw a line on the canvas
function drawLine(x0, y0, x1, y1)
{
     g_context.moveTo(x0, y0);
     g_context.lineTo(x1, y1);
     g_context.strokeStyle = "#00FF00";
     g_context.stroke();
}
// draw an image on the canvas
function drawImg(dx, dy, dw, dh)
{
     var pic = new Image();
     pic.src = "html5.png";
     g_context.drawImage(pic,
          dx, dy, dw, dh);
}
```





canvas-timer-eventlistener.html

Event-driven JavaScript 1

- Events have a ***name***, such as ”click”, ”change”, ”load”, ”mouseover”, ”keypress” or ”readystatechange” that indicates the general type of event that has occurred
- Events also have a ***target***, which is the object on which they occurred
- If we want our program to respond to an event, we write a function known as an ”event handler”, “event listener”, or a “callback”
- In most browsers, for most kinds of events, event handlers are passed an object as an argument, and the properties of this object provide details about the event
- Events whose targets are elements in a document often propagate up the document tree in a process known as “bubbling”

```
// event examples
window.onload = function() { ... };

document.getElementById("button1").onclick = function() { ... };

xmlHttpRequest.onreadystatechange = handleResponse;
function handleResponse() { ... }
```

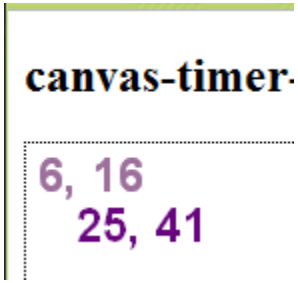
Event-driven JavaScript 2

- If you need to register more than one event handler function for a single event, or if you want to write a module of code that can safely register event handlers even if another module has already registered a handler for the same event on the same target, you have to use another event handler registration technique
- Most objects that can be event targets have a method named addEventListener(), which allows the registration of multiple listeners
- The first argument to the function is the name of the event without "on", the second the function and the third ”false” indicate that event shall bubble up the DOM tree as normal

```
// event listener examples
window.addEventListener("load", function() {...}, false);

xmlHttpRequest.addEventListener("readystatechange", function() {...}, false);

// in IE8 and earlier, you must use a similar method, named attachEvent():
window.attachEvent("onload", function() {...});
```

Event-driven JavaScript 3

- Check out chapter 13 in Eloquent JavaScript
- Example



canvas-timer-eventlistener.html

```
// register a click event handler for a canvas element
registerEventHandler(g_canvas, "click", canvasOnClick);

// Point object definition
function Point (x, y) {
     this.x = x;
     this.y = y;
}
// Point object method
Point.prototype.toString = function() {
     return this.x + ', ' + this.y;
};
function registerEventHandler(node, event, handler) {
     if (typeof node.addEventListener == "function")
          node.addEventListener(event, handler, false);
     else
          node.attachEvent("on" + event, handler); // for < IE9
}
// catch mouse clicks on canvas and draw point text
function canvasOnClick(ev) {
     var point = getCursorPosition(ev);
     drawText(point.toString(), point.x, point.y);
}
```

video element

- Tag works like the audio tag with some additional attributes
- Read chapter 5 in Dive Into HTML5 for a good intro to digital video, describing difference between a codec and a container
- Video containers
  - MPEG 4 (.mp4 or .m4v), Flash Video (.flv), Ogg (.ogv), WebM (.webm) which is similar to Matroska (.mkv), Audio Video Interleave (.avi)
- Video codecs
  - H.264, THEORA, VP8
- When you “watch a video,” your video player is doing at least three things at once
  - Interpreting the container format to find out which video and audio tracks are available, and how they are stored within the file so that it can find the data it needs to decode next
  - Decoding the video stream and displaying a series of images on the screen
  - Decoding the audio stream and sending the sound to your speakers

What works on the web?

- For maximum compatibility, here's what your video workflow will look like
  - Make one version that uses WebM (VP8 + Vorbis).
  - Make another version that uses H.264 baseline video and AAC "low complexity" audio in an MP4 container.
  - Make another version that uses Theora video and Vorbis audio in an Ogg container.
  - Link to all three video files from a single <video> element, and fall back to a Flash-based video player.

```
<!-- Your browser will go down the list of video and audio sources in order,
     and play the first one it's able to play -->
<video id="movie" width="480" height="270" controls="controls" preload="none"
     poster="http://localhost/myhome/javascript/ia_11_exempel/video/eric-schmidt-google-evil.jpg">
     <source src="evilgooglevideo.webmvp8.webm" type="video/webm; codecs='vp8, vorbis'" />
     <source src="evilgooglevideo.theora.ogv" type="video/ogg; codecs='theora, vorbis'" />
     <source src="evilgooglevideo.mp4" type="video/mp4; codecs='avc1.42E01E, mp4a.40.2'" />
</video>
<audio id="audio" controls="controls">
     <source src="ACDC_-_Back_In_Black-sample.ogg" type="audio/ogg" />
     <source src="ACDC_-_Back_In_Black-sample.mp3" type="audio/mpeg" />
</audio>
```

video-audio.html

audio element

- Tag works like the video element but with less attributes
- Audio codecs
  - MP3, AAC, and Ogg Vorbis

Additional video attributes

Attribute	Value	Description
autoplay	autoplay	Specifies that the audio will start playing as soon as it is ready
controls	controls	Specifies that audio controls should be displayed (such as a play/pause button etc).
loop	loop	Specifies that the audio will start over again, every time it is finished
preload	none	Specifies if and how the author thinks the audio should be loaded when the page loads
src	URL	Specifies the URL of the audio file
height	pixels	Sets the height of the video player

HTML5 Links

- Common resources for HTML5 and further links on these pages
- The HTML5 Specification
  - http://dev.w3.org/html5/markup/spec.html
- Roberts talk
  - http://robertnyman.com/html5/
- html5 laboratory
  - http://www.html5laboratory.com/
- HTML5 Rocks
  - http://www.html5rocks.com/
- HTML 5 Demos and Examples
  - http://html5demos.com/

Best free media converter?
https://handbrake.fr/