HTML 5 Features

Lecture 12

HTML 5 Features

- Canvas
- Audio and Video
- Geolocation
- Local Storage
- Web Workers
- Native Drag-n-Drop
- Microdata
- Semantic Tags

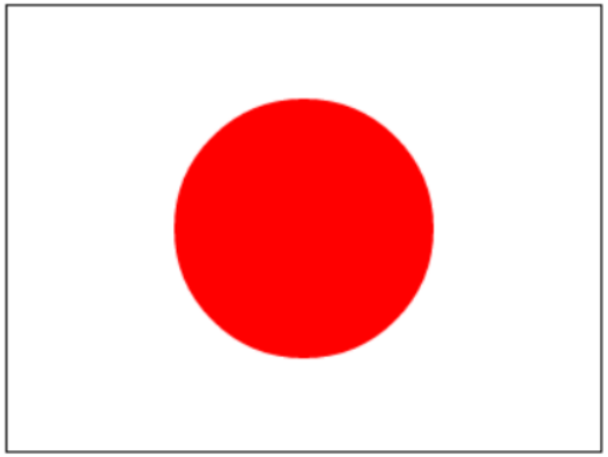
Why HTML5?

- No need to install additional plugins such as Flash, Java Applet, Silverlight, ActiveX on browser for rich user experience.
- Native audio and video players
- Enhancements on handling geolocation, background workers, validating forms and having local storage within browser.
- Cross-browser compatibility

The Canvas

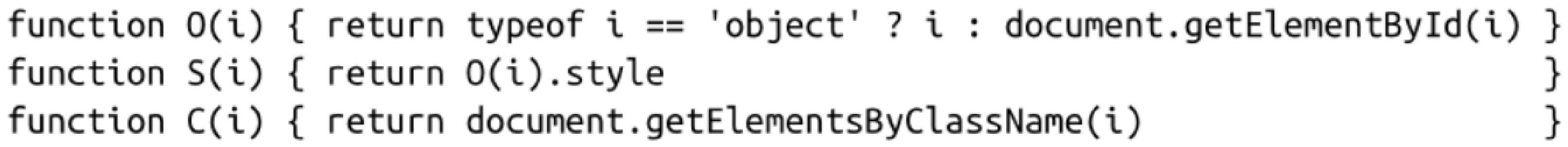
- *canvas* element enables drawing graphics in a web page using JavaScript
- if *canvas* is not supported by the browser, it displays the text.

```
<canvas id='mycanvas' width='320' height='240'>
  This is a canvas element given the ID <i>mycanvas</i>
  This text is only visible in non-HTML5 browsers
</canvas>
 <script>
   canvas           = O('mycanvas')
   context          = canvas.getContext('2d')

   context.fillStyle = 'red'
   S(canvas).border  = '1px solid black'

   context.beginPath()
   context.moveTo(160, 120)
   context.arc(160, 120, 70, 0, Math.PI * 2, false)
   context.closePath()
   context.fill()
 </script>
```



OSC functions

- Minimal replacement for jQuery:



```
function O(i) { return typeof i == 'object' ? i : document.getElementById(i) }
function S(i) { return O(i).style                                            }
function C(i) { return document.getElementsByClassName(i)                    }
```

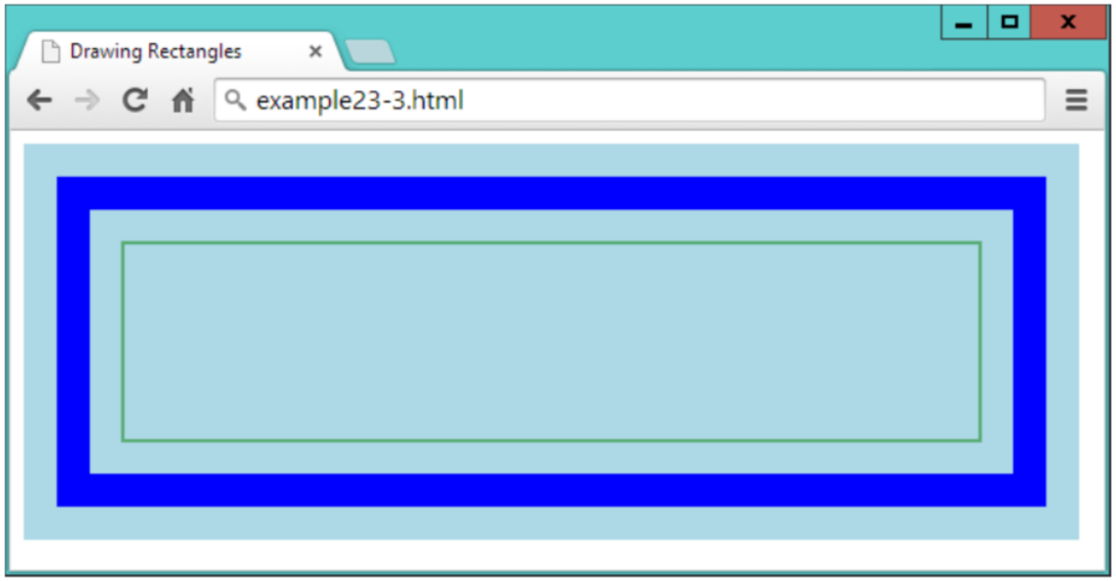
- If you are using jQuery, use it for your needs, instead of OSC functions!

Creating Rectangles

- Create a rectangle
  - *context.fillRect(x0, y0, x1, y1)*
- Set fill color
  - *context.fillStyle='blue'*
- Erase in rectangular shape
  - *context.clearRect(x0, y0, x1, y1)*
- Creating outlined rectangle
  - *context.strokeRect(x0, y0, x1, y1)*
- Set line color
  - *context.strokeStyle='green'*

Example

```
<script>
  canvas                  = O('mycanvas')
  context                 = canvas.getContext('2d')
  S(canvas).background = 'lightblue'
  context.fillStyle       = 'blue'
  context.strokeStyle     = 'green'

  context.fillRect(  20, 20, 600, 200)
  context.clearRect( 40, 40, 560, 160)
  context.strokeRect(60, 60, 520, 120)
</script>
```



Creating Gradients and Patterns

- Create Linear gradient
  - *var g=context.createLinearGradient(x0, y0, x1, y1)*
- Add color stops
  - *g.addColorStop(position, color)*
- Creating radial gradient
  - *var c=context.createRadialGradient(x0,y0, rad0, x1, y1, rad1)*
- Creating patterns
  - *var img=new Image();*
  - *img.src='image.png';*
  - *var pattern=context.createPattern(img, 'repeat');*

Writing Texts

- Stroke Text:
  - *context.strokeText('WickerpediA', 0, 0)*

WickerpediA

- Fill Text:
  - *context.fillText('WickerpediA', 0, 0)*
  - *context.fillStyle=pattern*

WickerpediA

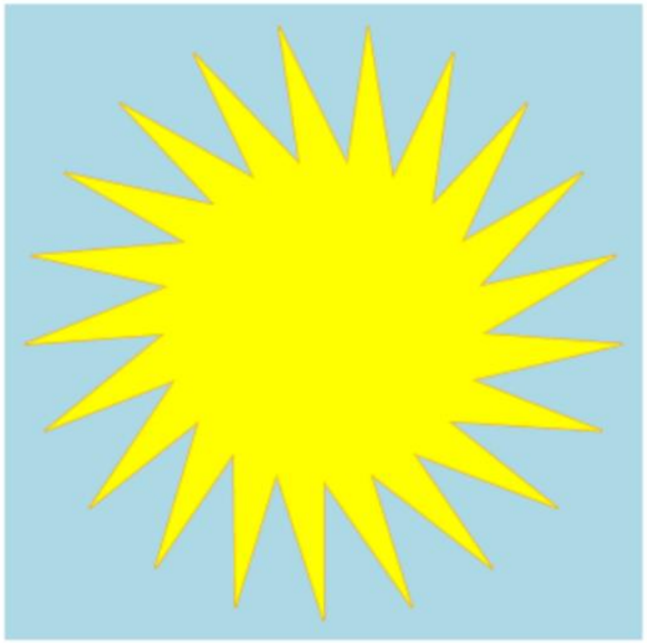
- Measure Text:
  - *var metrics=context.measureText('WickerpediA')*
  - *var width=metrics.width*

Drawing Lines

- Using Paths
  - *context.beginPath()*
  - *context.moveTo(20,100)*
  - *context.lineTo(20,20)*
- Not closing the path:
  - *context.stroke()*
  - *context.closePath()*
- Closing the path
  - *context.closePath()*
  - *context.stroke()*
- Line attributes:
  - *context.lineWidth=3*
  - *context.lineCap = 'round'* *// other values: butt, square*
  - *context.lineJoin = 'round'* *// other values: bevel, miter*

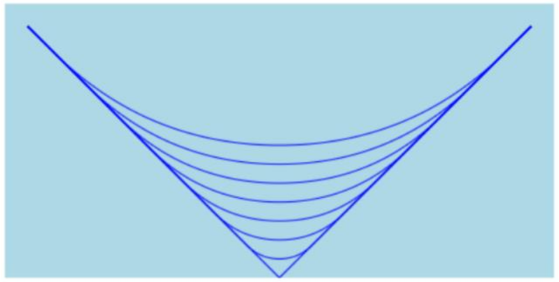
Example

```
canvas                = O('mycanvas')
context               = canvas.getContext('2d')
S(canvas).background = 'lightblue'
context.strokeStyle  = 'orange'
context.fillStyle    = 'yellow'

orig   = 160
points = 21
dist   = Math.PI / points * 2
scale1 = 150
scale2 = 80

context.beginPath()

for (j = 0 ; j < points ; ++j)
{
  x = Math.sin(j * dist)
  y = Math.cos(j * dist)
  context.lineTo(orig + x * scale1, orig + y * scale1)
  context.lineTo(orig + x * scale2, orig + y * scale2)
}

context.closePath()
context.stroke()
context.fill()
```



Drawing Curves

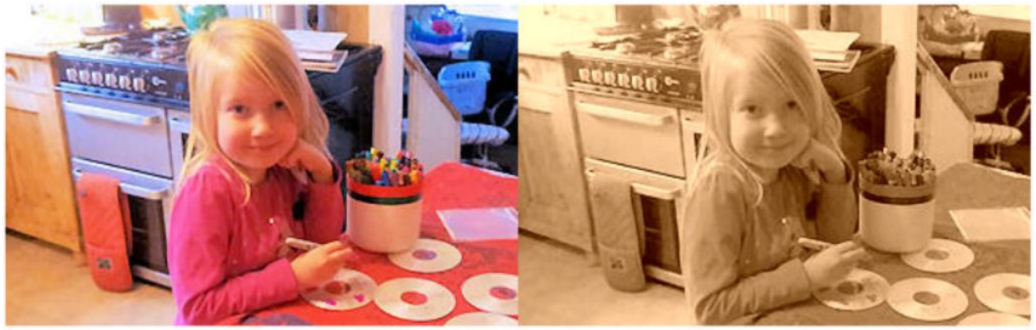
- Drawing arc:
  - *context.arc(x0, y0, radius, angle0, angle1, isCounterClockwise)*
- Drawing arc between 2 points:
  - *context.arcTo(x0, y0, x1, y1, radius)*



- Drawing quadratic curve between 2 points:
  - *context.moveTo(x0, y0)*
  - *context.quadraticCurveTo(cpx, cpy, x1, y1)*

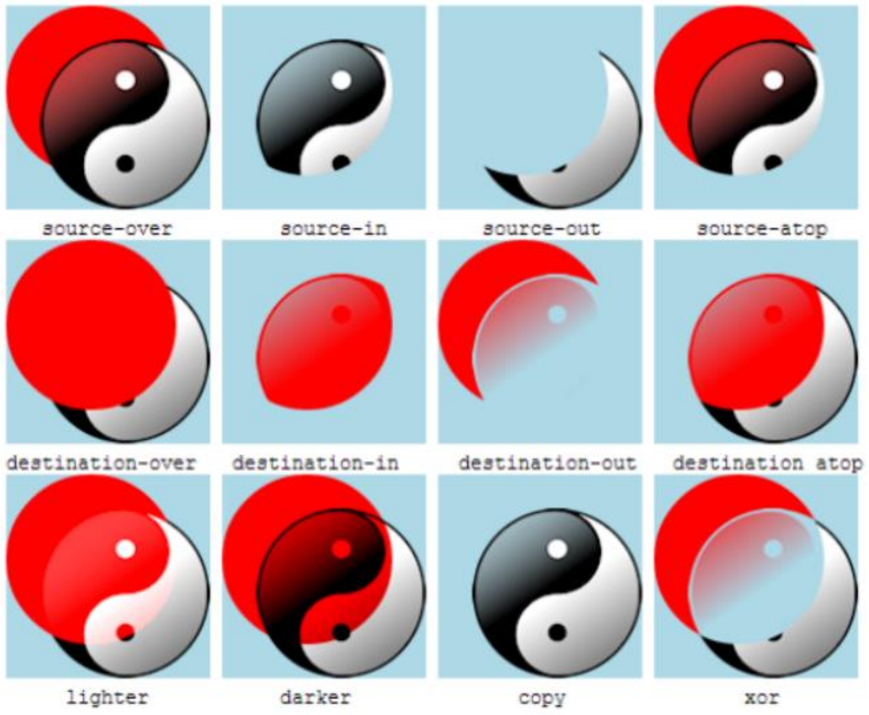
Pixel Level Editing

```
myimage      = new Image()
myimage.src = 'photo.jpg'

myimage.onload = function()
{
  context.drawImage(myimage, 0, 0)
  idata = context.getImageData(0, 0, myimage.width, myimage.height)

  for (y = 0 ; y < myimage.height ; ++y)
  {
    pos = y * myimage.width * 4
    for (x = 0 ; x < myimage.width ; ++x)
    {
      average =
      (
        idata.data[pos]     +
        idata.data[pos + 1] +
        idata.data[pos + 2]
      ) / 3

      idata.data[pos]     = average + 50
      idata.data[pos + 1] = average
      idata.data[pos + 2] = average - 50
      pos += 4;
    }
  }
  context.putImageData(idata, 320, 0)
}
```



Graphical Compositions

```
image     = new Image()
image.src = 'image.png'

image.onload = function()
{
  types =
  [
    'source-over',     'source-in',        'source-out',
    'source-atop',     'destination-over', 'destination-in',
    'destination-out', 'destination-atop', 'lighter',
    'darker',          'copy',             'xor'
  ]

  for (j = 0 ; j < 12 ; ++j)
  {
    canvas                 = O('c' + (j + 1))
    context                = canvas.getContext('2d')
    S(canvas).background = 'lightblue'
    context.fillStyle      = 'red'
     context.arc(50, 50, 50, 0, Math.PI * 2, false)
     context.fill()
     context.globalCompositeOperation = types[j]
     context.drawImage(image, 20, 20, 100, 100)
  }
}
```



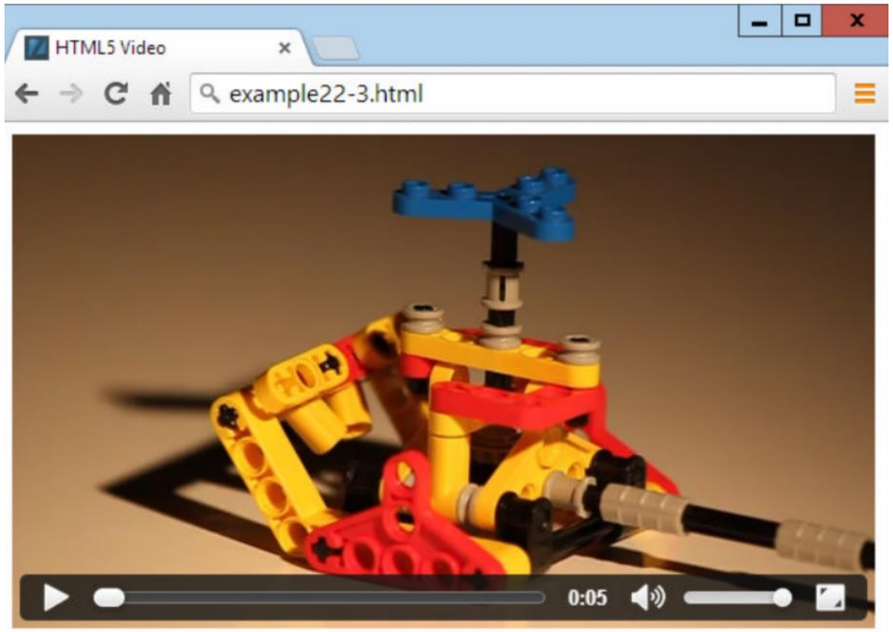
More about HTML5 graphics

- SVG tutorial:
  - https://www.w3schools.com/html/html5_svg.asp
- 3D Graphics using WebGL
  - http://learningwebgl.com/blog/?page_id=1217
- Other Visualisation Tools:
  - https://d3js.org/ - mostly for data visualisation/charts
  - https://threejs.org/ -mostly for 3D graphics
  - http://paperjs.org/ - mostly for 2D graphics
  - http://fabricjs.com/ - interactive graphics
  - http://www.babylonjs.com/ - mostly for 3D games on browser

Audio and Video

- *audio* and *video* tags can seamlessly embed audio and video files into your web page

```
<video width='560' height='320' controls>
  <source src='movie.mp4'  type='video/mp4'>
  <source src='movie.webm' type='video/webm'>
  <source src='movie.ogv'  type='video/ogg'>
</video>
```



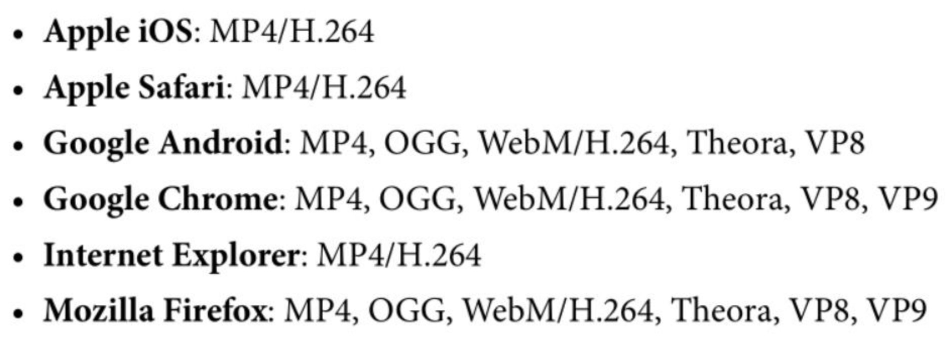
Audio Codecs (enCOder/DECoders)

- **AAC:** Apple's iTunes Advanced Audio Encoding. MIME type: *audio/aac*
- **MP3:** MPEG Audio Layer 3, MIME type: *audio/mpeg*
- **PCM:** Pulse Coded Modulation, lossless codec and usually many time larger than AAC and MP3. Extension *.wav.* MIME Type: *audio/wav*
- **Vorbis:** *.ogg* not patented and free of charge. MIME type: *audio/oga*

- **Apple iOS**: AAC, MP3, PCM
- **Apple Safari**: AAC, MP3, PCM
- **Google Android**: 2.3+ AAC, MP3, Vorbis
- **Google Chrome**: AAC, MP3, Vorbis
- **Microsoft Internet Explorer**: AAC, MP3
- **Mozilla Firefox**: MP3, PCM, Vorbis
- **Opera**: PCM, Vorbis

Video Codecs (enCOder/DECoders)

- **MP4:** MPEG-4 standard. MIME type: *video/mp4*
- **OGG:** Free open container format. MIME Type: *video/ogg, video/ogv*
- **WebM:** Open compression supporting codecs H.264, Theora, VP8 and VP9



- **Apple iOS**: MP4/H.264
- **Apple Safari**: MP4/H.264
- **Google Android**: MP4, OGG, WebM/H.264, Theora, VP8
- **Google Chrome**: MP4, OGG, WebM/H.264, Theora, VP8, VP9
- **Internet Explorer**: MP4/H.264
- **Mozilla Firefox**: MP4, OGG, WebM/H.264, Theora, VP8, VP9

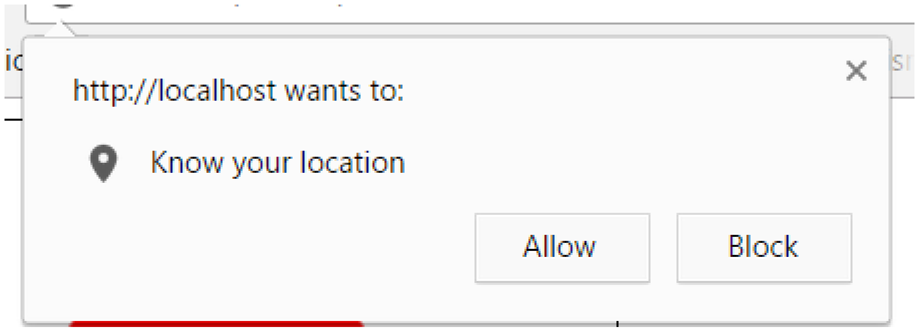
Geolocation

- Browser can send geolocation information to web server which comes from GPS chip on your laptop or mobile phone, from your IP address or from nearby Wi-Fi hotspots.
- User should give permission before this info is sent
- Loads of uses:
  - Navigation
  - Local maps
  - Notifying about local restaurants, wi-fi hotspots or other spots
  - Nearest gas station
  - Friends near-by your

Geolocation

- Requesting Geolocation:

```
if (typeof navigator.geolocation=='undefined') {
    alert('Geolocation not supported');
} else {
    navigator.geolocation.getCurrentPosition(
        function(position){
            alert("Position granted: Lat:" + position.coords.latitude + ", Lng: " + position.coords.longitude);
        },
        function(error){
            switch(error.code){
                case 1: alert("Permission Denied"); break;
                case 2: alert("Position unavailable"); break;
                case 3: alert("Operation timed out"); break;
                case 4: alert("Unknown error"); break;
            }
        });
}
```

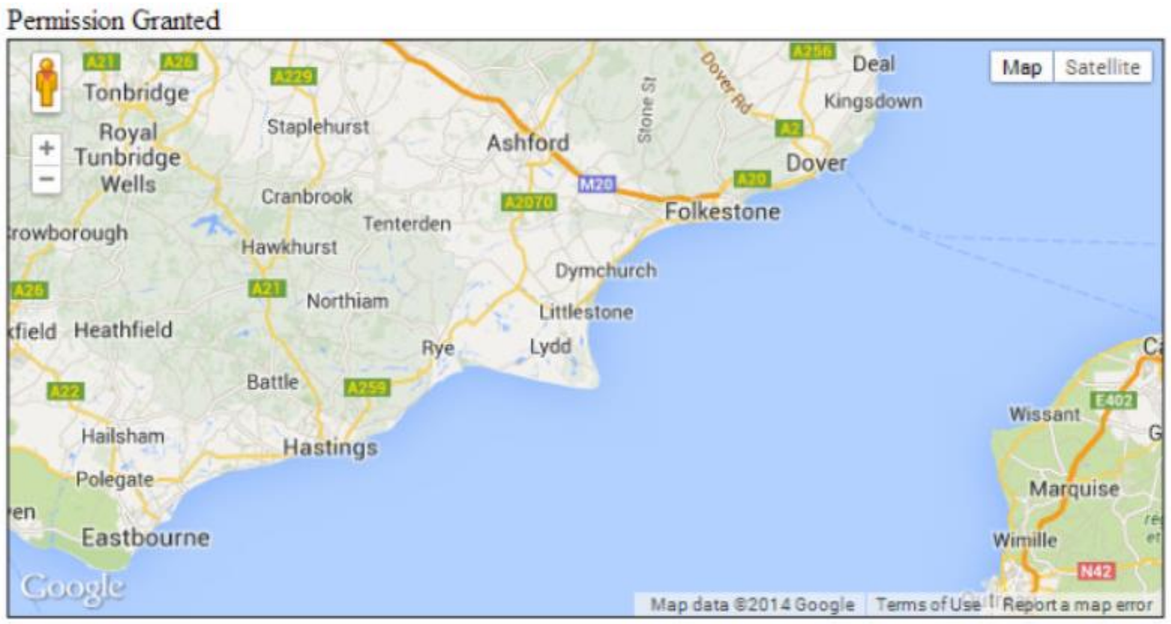

http://localhost wants to:

Know your location

Google Maps

```
<script src="https://maps.googleapis.com/maps/api/js?sensor=false"></script>
```

```
<div id='status'></div>
<div id='map'></div>

<script>
  if (typeof navigator.geolocation == 'undefined')
     alert("Geolocation not supported.")
  else
    navigator.geolocation.getCurrentPosition(granted, denied)

  function granted(position)
  {
    O('status').innerHTML = 'Permission Granted'
    S('map').border       = '1px solid black'
    S('map').width        = '640px'
    S('map').height       = '320px'

    var lat   = position.coords.latitude
    var long  = position.coords.longitude
    var gmap  = O('map')
    var gopts =
    {
      center: new google.maps.LatLng(lat, long),
      zoom: 9, mapTypeId: google.maps.MapTypeId.ROADMAP
    }
    var map = new google.maps.Map(gmap, gopts)
  }

  function denied(error)
  {
    var message

    switch(error.code)
    {
      case 1: message = 'Permission Denied'; break;
      case 2: message = 'Position Unavailable'; break;
      case 3: message = 'Operation Timed Out'; break;
      case 4: message = 'Unknown Error'; break;
    }

    O('status').innerHTML = message
  }
</script>
```


Permission Granted

Local Storage

- Cookies provide limited local storage on client's computer
- However it would be more useful if we could store more data in client's PC for applications such as
  - word processor
  - spreadsheets
  - graphical editors
  - music playlists
- Advantages of local storage:
  - provide up to 10 MBs of storage on client's computer
  - unlike Cookies, the data in local storage is not sent to server with every request
  - remove the burden of hosting user data at server
  - provide better performance for slow-connection
  - offline web applications are possible
  - private information can be stored on client computer only

Using Local Storage

```
if (typeof localStorage == 'undefined')
{
  alert("Local storage is not available")
}
else
{
  username = localStorage.getItem('username')
  password = localStorage.getItem('password')
  alert("The current values of 'username' and 'password' are\n\n" +
    username + " / " + password + "\n\nClick OK to assign values")

  localStorage.setItem('username', 'ceastwood')
  localStorage.setItem('password', 'makemyday')
  username = localStorage.getItem('username')
  password = localStorage.getItem('password')
  alert("The current values of 'username' and 'password' are\n\n" +
    username + " / " + password + "\n\nClick OK to clear values")

  localStorage.removeItem('username')
  localStorage.removeItem('password')
  username = localStorage.getItem('username')
  password = localStorage.getItem('password')
  alert("The current values of 'username' and 'password' are\n\n" +
    username + " / " + password)
}
```

Web Workers

- Web workers can run in the background and communicate with the main JavaScript thread through event handlers.
- Workers are terminated by calling: *worker.terminate()*

```
<span id='result'>0</span>

<script>
  if (!!window.Worker)
  {
    var worker = new Worker('worker.js')

    worker.onmessage = function (event)
    {
      O('result').innerHTML = event.data;
    }
  }
  else
  {
    alert("Web workers not supported")
  }
</script>
```

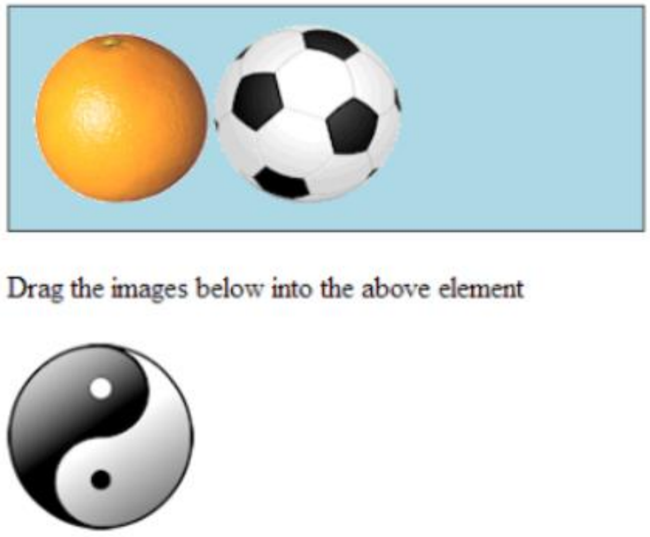
worker.js

```
var n = 1

search: while (true)
{
  n += 1

  for (var i = 2; i <= Math.sqrt(n); i += 1)
  {
    if (n % i == 0) continue search
  }

  postMessage(n)
}
```

Native Drag-n-Drop

```
<div id='dest' ondrop='drop(event)' ondragover='allow(event)'></div><br>
Drag the image below into the above element<br><br>

<img id='source1' src='image1.png' draggable='true' ondragstart='drag(event)'>
<img id='source2' src='image2.png' draggable='true' ondragstart='drag(event)'>
<img id='source3' src='image3.png' draggable='true' ondragstart='drag(event)'>
```

```
function allow(event)
{
  event.preventDefault()
}

function drag(event)
{
  event.dataTransfer.setData('image/png', event.target.id)
}

function drop(event)
{
  event.preventDefault()
  var data=event.dataTransfer.getData('image/png')
  event.target.appendChild(O(data))
}
```

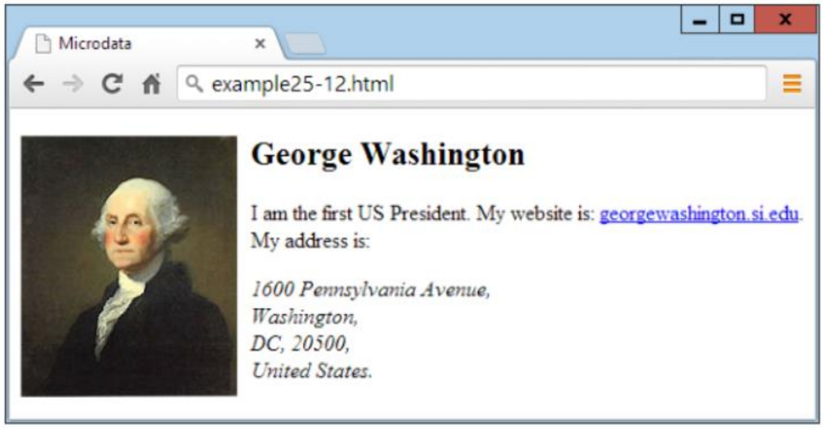

Drag the images below into the above element

Microdata

- Microdata is a set of tag attributes that attach semantic meaning to each HTML tag
- It is parsed by search engines and social networks to better represent the data in different web sites.
- Below is the list of attributes:
  - itemscope
  - itemtype
  - itemid
  - itemref
  - itemprop

Microdata describing a person in HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Microdata</title>
  </head>
  <body>
    <section itemscope itemtype='http://schema.org/Person'>
      <img itemprop='image' src='gw.jpg' alt='George Washington'
        align='left' style='margin-right:10px'>
      <h2 itemprop='name'>George Washington</h2>
      <p>I am the first <span itemprop='jobTitle'>US President</span>.
      My website is: <a itemprop='url'
        href='http://georgewashington.si.edu'>georgewashington.si.edu</a>.
      My address is:</p>
      <address itemscope itemtype='http://schema.org/PostalAddress'
        itemprop='address'>
        <span itemprop='streetAddress'>1600 Pennsylvania Avenue</span>,<br>
        <span itemprop='addressLocality'>Washington</span>,<br>
        <span itemprop='addressRegion'>DC</span>,<br>
        <span itemprop='postalCode'>20500</span>,<br>
        <span itemprop='addressCountry'>United States</span>.
      </address>
    </section>
  </body>
</html>
```



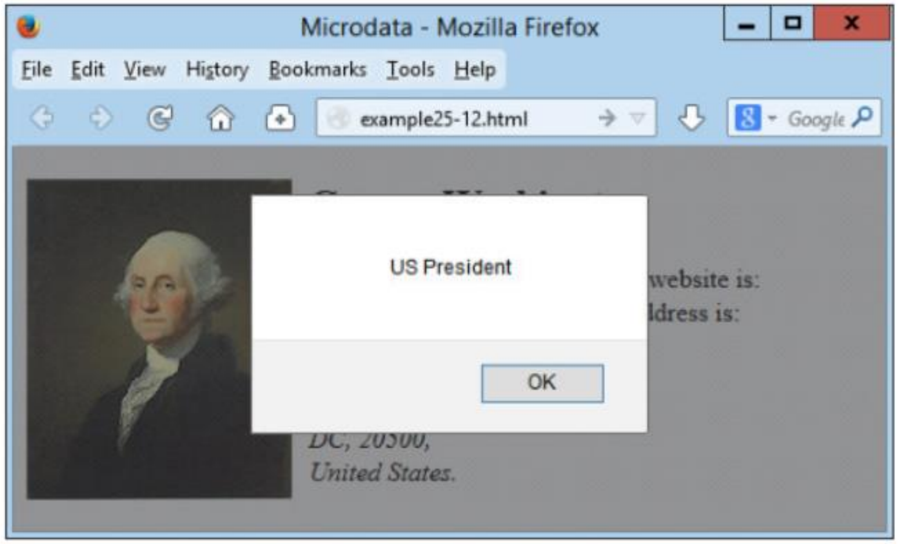
George Washington

I am the first US President. My website is: georgewashington.si.edu.
My address is:

1600 Pennsylvania Avenue,
Washington,
DC, 20500,
United States.

Accessing Microdata using JavaScript

```
window.onload = function()
{
  if (!!document.getItems)

  {
    data = document.getItems('http://schema.org/Person')[0]
    alert(data.properties['jobTitle'][0].textContent)
  }
}
```


Microdata - Mozilla Firefox
File Edit View History Bookmarks Tools Help
example25-12.html
Google
OK

US President

website is:
ldress is:
DC, 20500,
United States.

New HTML5 Tags

Semantic/Structural Tags

<article>	Defines an article in a document
<aside>	Defines content aside from the page content
<bdi>	Isolates a part of text that might be formatted in a different direction from other text outside it
<details>	Defines additional details that the user can view or hide
<dialog>	Defines a dialog box or window
<figcaption>	Defines a caption for a <figure> element
<figure>	Defines self-contained content
<footer>	Defines a footer for a document or section
<header>	Defines a header for a document or section

New HTML5 Tags

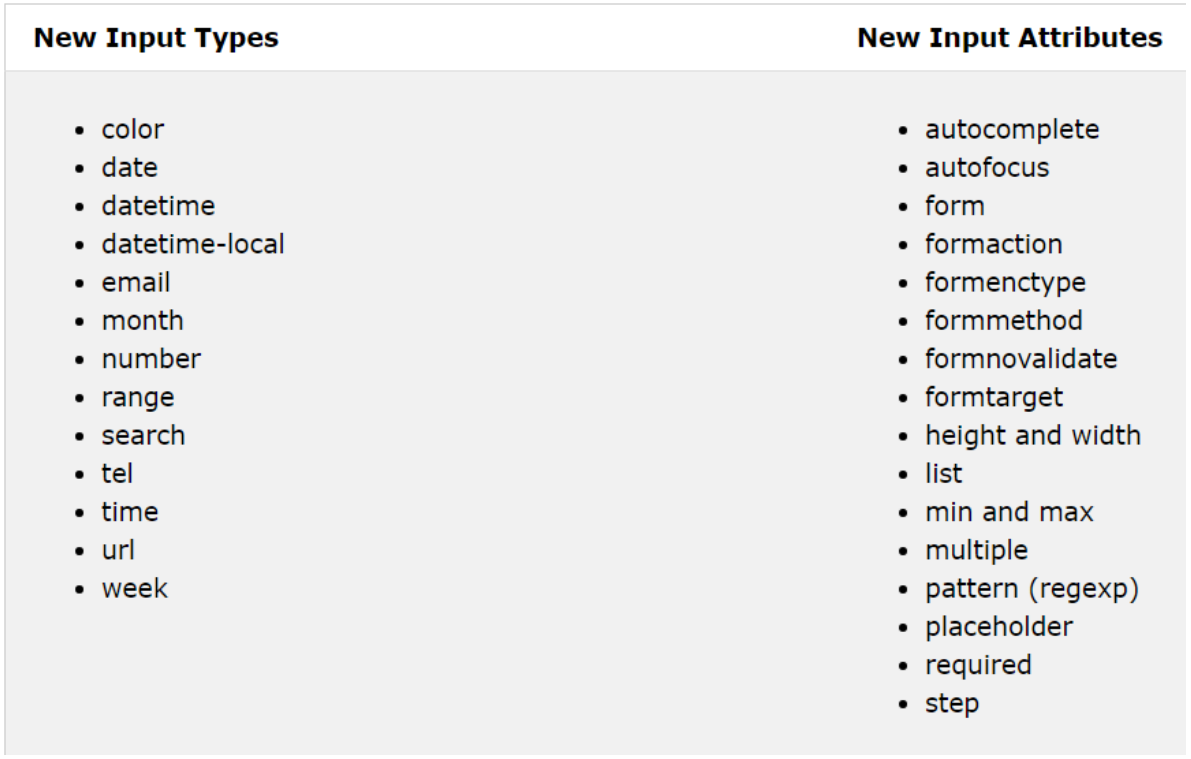
Semantic/Structural Tags

<main>	Defines the main content of a document
<mark>	Defines marked/highlighted text
<menuitem>	Defines a command/menu item that the user can invoke from a popup menu
<meter>	Defines a scalar measurement within a known range (a gauge)
<nav>	Defines navigation links
<progress>	Represents the progress of a task
<rp>	Defines what to show in browsers that do not support ruby annotations
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography)
<ruby>	Defines a ruby annotation (for East Asian typography)
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time
<wbr>	Defines a possible line-break

New HTML5 Tags

Form Tags

<datalist>	Specifies a list of pre-defined options for input controls
<keygen>	Defines a key-pair generator field (for forms)
<output>	Defines the result of a calculation



New Input Types

- color
- date
- datetime
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

New Input Attributes

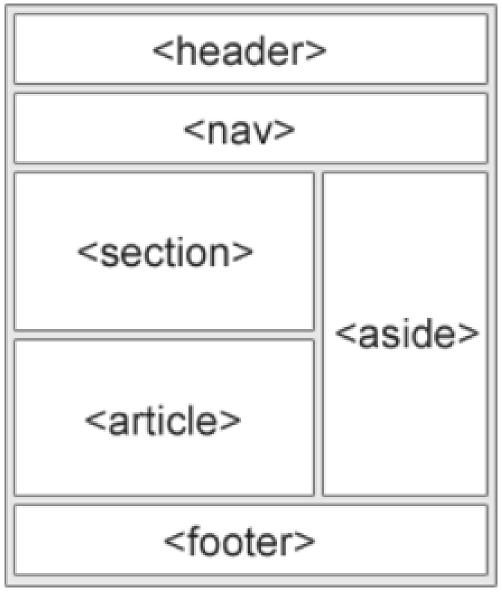
- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list
- min and max
- multiple
- pattern (regexp)
- placeholder
- required
- step

More info: https://www.w3schools.com/html/html_form_input_types.asp

Semantic Tags

- Non-semantic (Structural) tags such as *div* and *span* tell nothing about its content
- Semantic tags such as *table, form, article* provide some meta information about the content and its structure

  - <article>
  - <aside>
  - <details>
  - <figcaption>
  - <figure>
  - <footer>
  - <header>
  - <main>
  - <mark>
  - <nav>
  - <section>
  - <summary>
  - <time>



More about HTML5 tools

- Audio Video Players:
  - https://github.com/adrienjoly/playemjs
  - https://github.com/videojs/video.js
- Maps:
  - http://leafletjs.com/
  - http://cesiumjs.org/
- Local Storage:
  - https://github.com/marcuswestin/store.js
  - https://github.com/mozilla/localForage
- Notifications:
  - https://jaredreich.com/projects/notie
  - https://sciactive.com/pnotify/
  - http://ned.im/noty