HTML5

Internet Engineering

Fall 2016

Bahador Bakhshi

CE & IT Department, Amirkabir University of Technology

Questions

- Q9) What are the new futures in web pages?
  - In other words, why new version of HTML?
- Q9.1) Search engines don't show my page!
- Q9.2) I don't want use Flash for multimedia!
- Q9.3) Why JS for form validation every time?
- Q9.4) Cookies are good, but I need more!!
- Q9.5) Can I implement games under web?!
- Q9.6) I want use elements on page as objects!
- Q9.7) How does the Gmail off-line work?

Outline

- Introduction
- Page Structure
- Multimedia
- Forms
- Storage
- Drag & Drop
- Canvas
- Other Features

Outline

- Introduction
- Page Structure
- Multimedia
- Forms
- Storage
- Drag & Drop
- Canvas
- Other Features

Introduction

- HTML5 is the next generation of HTML
  - HTML5 is recently standardized
  - Most modern browsers have some HTML5 support
- HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG)
  - In 2006, they decided to cooperate and create a new version of HTML

HTML5 Goals

- Fill the HTML 4 gaps for new modern web apps.
- New features should be based on HTML, CSS, DOM, and JavaScript
  - Reduce the need for external plugins like Flash, …
- Standardize common elements
  - More markup to replace scripting
- Standardize common usage & applications
- Better error handling
  - A consistent DOM for any bad markup

HTML5 Standard Status

- W3C Recommendations
  - Note status
    - People at W3C start discussing some issues
  - Working Draft
    - W3C invites comments
  - Candidate Recommendation
    - Bugs, issues, …
  - Recommendation

2004	WHATWG started
2008	W3C Working Draft
2012	W3C Candidate Recommendation
28 Oct. 2014	W3C Recommendation

HTML5 New Features

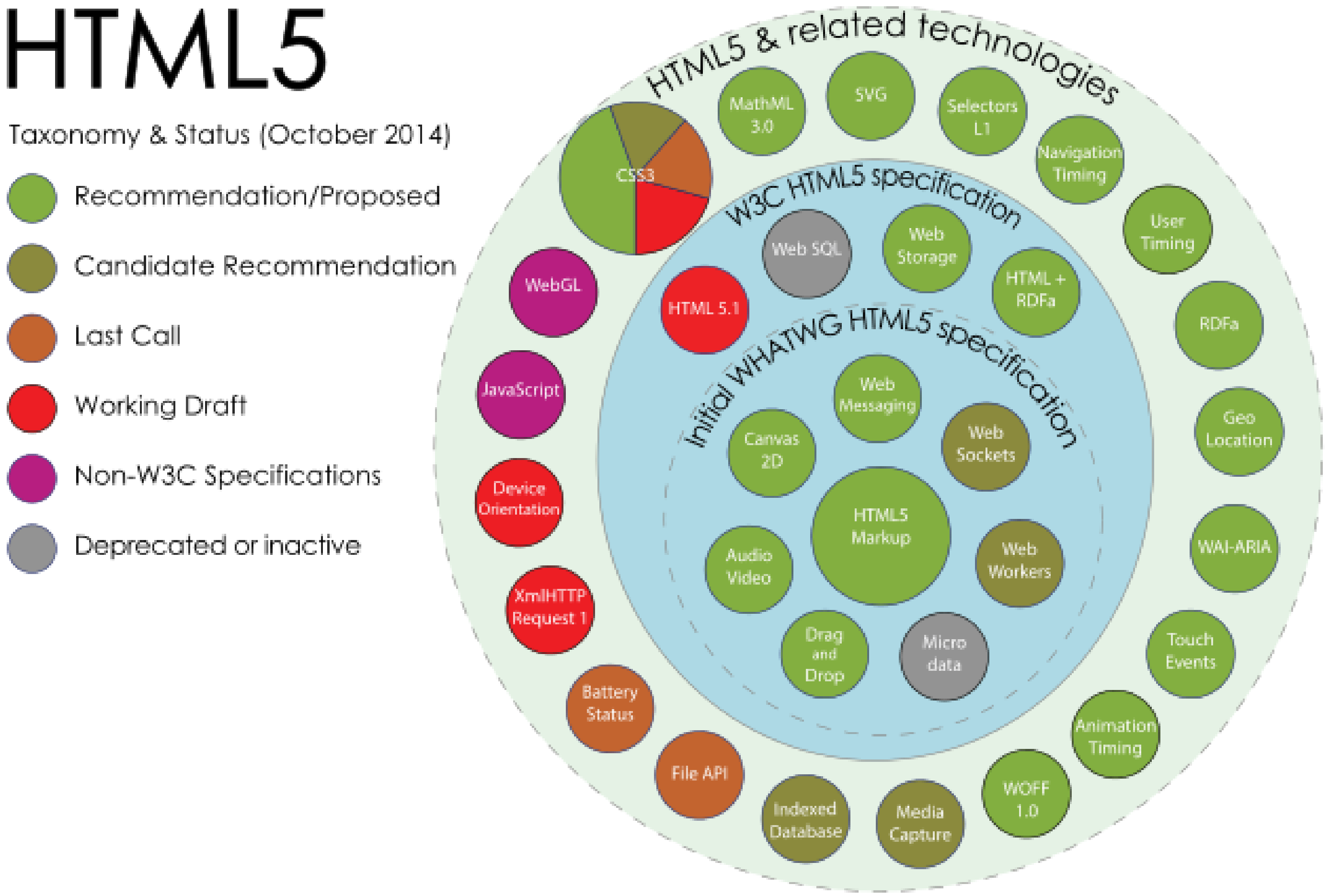


Image source: Wikipedia

HTML5 Document Type & Encoding

- HTML5 is not based on XML or SGML
- However, browsers need to see `<!DOCTYPE …>`
  - To work in standard compliance mode
- HTML5 (dummy) Document type
  - It is dummy because does NOT determine a DTD!

```
<!DOCTYPE html>
```

- Character encoding

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

HTML5 Syntax

- HTML5 syntax (not XHTML5) is very lax
- These are equivalent

```
<meta CHARSET=utf-8 >
<meta charset="utf-8" />
<META charset="utf-8" >
```

- Following tags are not required!!!
  - Document is successfully validated without them

```
<html> <head> <body>
```

Some New Global Attributes

Attribute	Value	Description
contenteditable	true false inherit	Specifies whether a user can edit the content of an element or not
draggable	true false auto	Specifies whether a user is allowed to drag an element or not
hidden	hidden	Specifies that an element should be hidden
spellcheck	true false	Specifies if the element must have its spelling and grammar checked

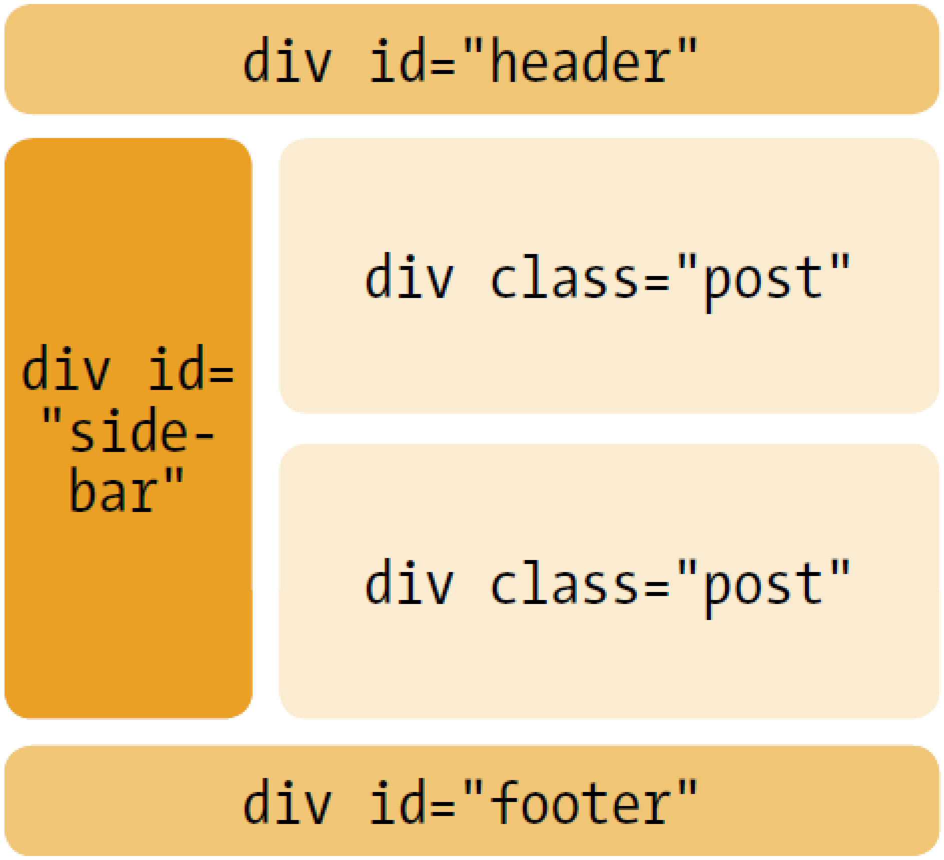
Outline

Page Structure

- In XHTML, page is organized by **`div`**s
  - Assign meaningful ID or Class for **`div`**s
    - E.g., header, navigation, footer, content, …
  - This approach is ad-hoc
- HTML5: new tags for the common divs
  - **`<header>`**, **`<nav>`**, **`<footer>`**, …
- These new elements are to emphasize semantic web rather than new presentation style!
  - HTML is not just presentation
  - Each page portion has its own meaning
    - Semantic web & better search engines
- HTML5 recommends the usage of these tags

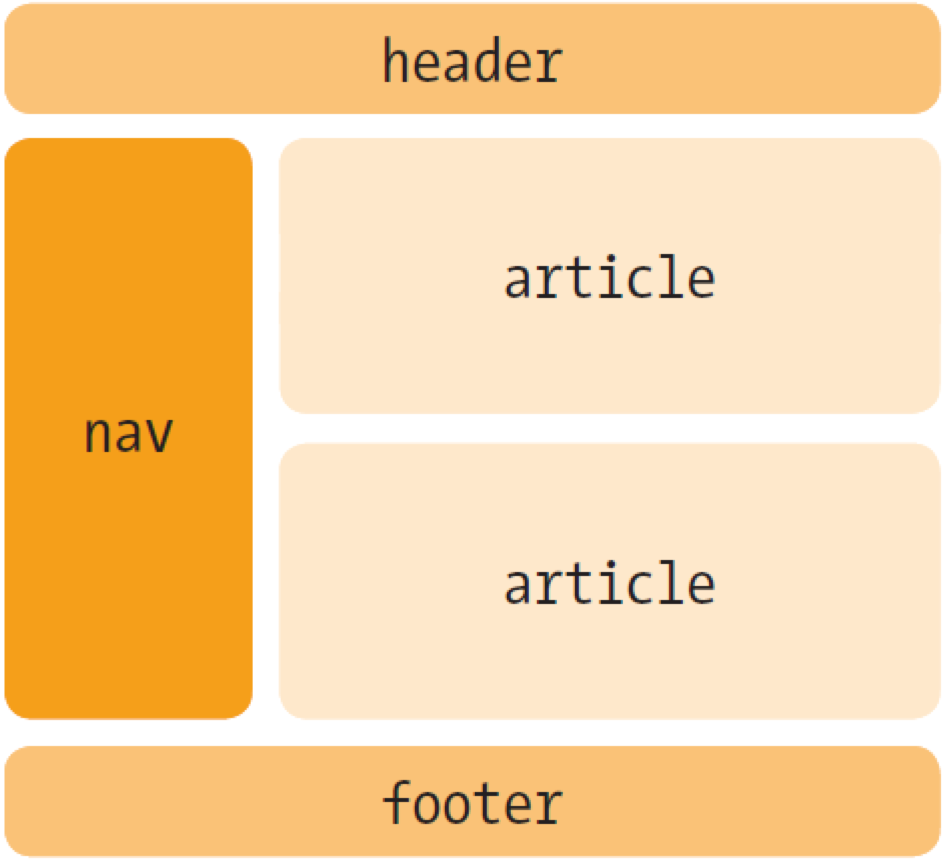
XHTML Based Page Design

- A sample weblog
- id & class to divs
  - CSS to arrange divs
- Search engines do not understand the meaning of each div
  - Footer is as important as header!

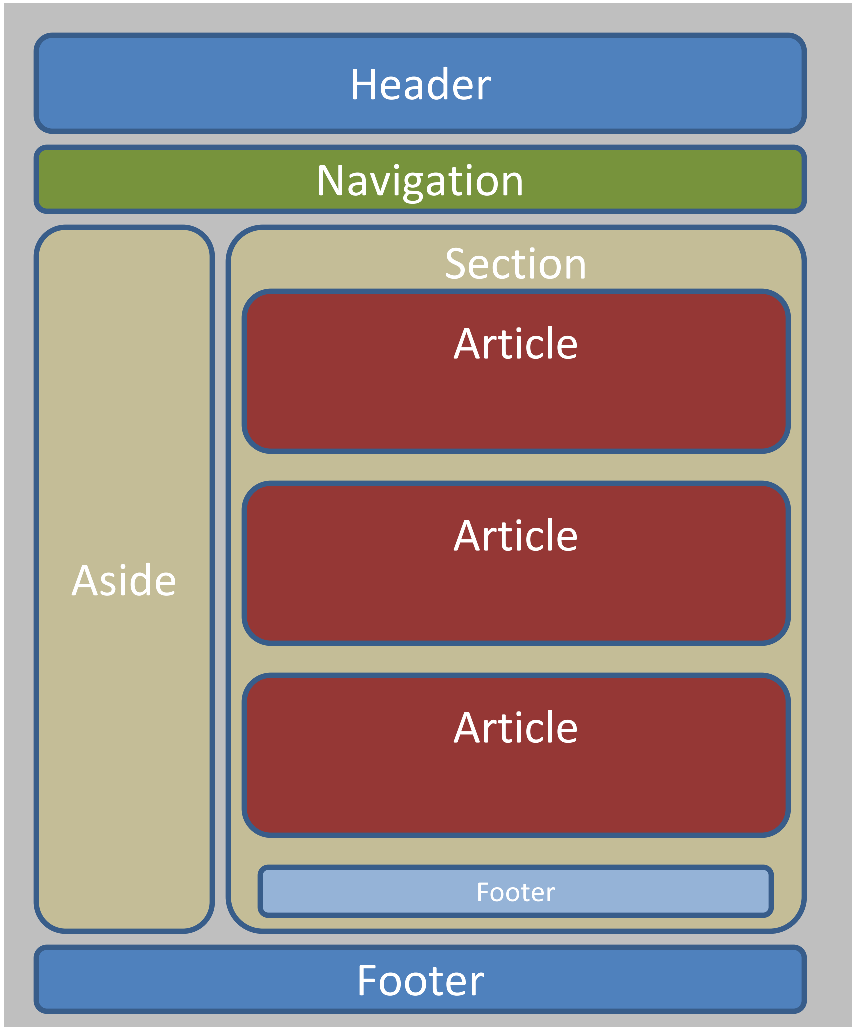


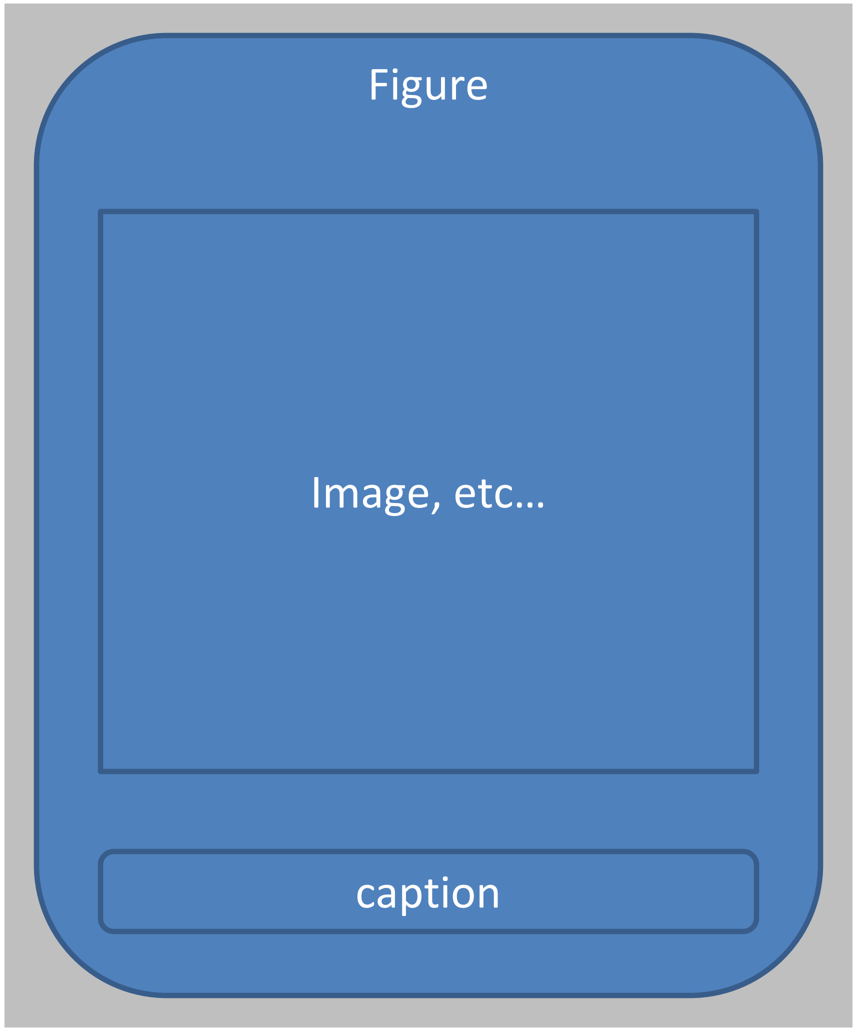
HTML5 Based Page Design

- The weblog using HTML5 semantic tags
- Browsers do not have default style for the new tags
- Use CSS
  - display:block
  - …
- Now, search engine knows that header >> article >> footer
  - Better page ranking!



Sample HTML5 Page Structure





Example: http://netstream.ru/htmlsamples/html5-blog/index.html

HTML5 Page Structure Tags

- **`header`**
  - Represents a group of introductory
- **`section`**
  - Represents a generic document section
- **`article`**
  - Represents an independent piece of content of a document, such as newspaper article
- **`aside`**
  - Represents an independent piece of content of a document, such as a blog entry

HTML5 Page Structure Tags

- **`hgroup`**
  - Groups multiple headers, title and subtitle in the header
- **`footer`**
  - Represents a footer for a section
- **`nav`**
  - Represents a section intended for navigation
- **`figure`**
  - Used to associate a caption together with some embedded content
  - **`<img>`** is used to insert (source) the image
  - **`<figcaption>`** provides caption to the figure

Outline

HTML5 Multimedia

- Until now, there hasn't been a standard multimedia on web
- Most videos are shown through a plugin
  - However, not all browsers have the same plugins
- HTML5 specifies a standard way to include multimedia contents (video or audio) on web
  - **`<video>`** element to insert video
  - **`<audio>`** element to play audio sound
  - **`<source>`** element to source the content

HTML5 Video

- Video element:

```
<video width="320" height="240"
controls="controls">
  . . .
 Your browser does not support video tag.
</video>
```

- The `control` attribute adds video controls, like play, pause, and volume
- If height and width are set
  - Required space is reserved on page load
  - Without these attributes, no space is reserved, Page layout will change during loading

HTML5 Video (Cont'd)

- The video file is specified by `<source>` in `<video>`

```
<source src="URL" type="codec type" />
```

- Three codecs: Ogg, MPEG 4, WebM
  - However, Not all browsers support all formats
    - Firefox, Opera, and Chrome: Ogg, WebM
    - Internet Explorer, Chrome, and Safari: MPEG4
- The `<video>` element allows multiple `<source>`
  - The browser will use the first recognized format
  - To cover all the major browsers, use two `<source>` elements: MPEG4 and Ogg or WebM

HTML5 Video (Cont’d)

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

Video Attributes

Attribute	Value	Description
autoplay	autoplay	The video will start playing as soon as it is ready
controls	controls	Video controls should be displayed
height	pixels	Sets the height of the video player
loop	loop	Video will start over again, every time it is finished
muted	muted	Audio output of the video should be muted
poster	URL	An image to be shown while the video is downloading, or until the user hits the play button
width	pixels	Sets the width of the video player

HTML5 Audio

- Three formats: Ogg, WAV, MP3
  - Ogg: Firefox, Opera, Chrome
  - MP3: IE9, Chrome, Safari

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

- control, multiple source, and content text is similar to **`<video>`** element

HTML5 Audio (cont'd)

Attribute	Value	Description
autoplay	autoplay	Audio will start playing as soon as it is ready
controls	controls	Audio controls should be displayed (such as a play/pause button etc).
loop	loop	Audio will start over again, every time it is finished

(Some) Media Events

Attribute	Description
oncanplay	When a file is ready to start playing (when it has buffered enough to begin)
onemptied	When something bad happens and the file is suddenly unavailable (like unexpectedly disconnects)
onended	When the media has reach the end (a useful event for messages like "thanks for listening")
onerror	When an error occurs when the file is being loaded
onloadstart	When the file begins to load before anything is actually loaded
onpause	When the media is paused either by the user or programmatically
onplay	When the media is ready to start playing
onplaying	When the media actually has started playing
onvolumechange	When the volume is changed which (includes setting the volume to "mute")

Video example

- Example

Outline

- Introduction
- Page Structure
- Multimedia
- Forms
- Storage
- Drag & Drop
- Canvas
- Other Features

Forms Current & Future

- One of the problems with HTML 4 forms is that they are just dumb fields
- Validation is required on the server, of course
- But we have to duplicate it in the user's browser with JavaScript
- If browsers had built-in validation for some of the most common data types → ☺
  - HTML5 defines new input elements

HTML5 New Input Types

- New input types in HTML5

Type	Description
`url`	The input value is a URL
`email`	The input value is one or more email addresses
`date`	The input value is a date
`month`	The input value is a month
`week`	The input value is a week
`time`	The input value is of type time
`number`	The input value is a number
`range`	The input value is a number in a given range
`color`	The input value is a hexadecimal color, like #FF8800

HTML5 New Input Types (cont'd)

- Support of the new types
  - Opera & Chrome have (almost) full support
  - Limited support by other browsers
- Automatic validation by browser, usually
  - email & url & number are checked
  - date, week, month, range, color are selectable only from valid options
- Backward compatibility
  - Unsupported types are treated as **`type="text"`**

HTML5 New Input Types Attributes

- Input data in number & range can be controlled

Attribute	Value	Description
max	*number*	Specifies the maximum value allowed
min	*number*	Specifies the minimum value allowed
step	*number*	Specifies legal number intervals step="3", legal numbers could be -3,0,3,6)
value	*number*	Specifies the default value

```
Points: <input type="number" name="points"
  min="1" max="10" step="2"/>
```

Example

- Opera
- Chrome
- Firefox !!!
- IE :-P

HTML5 New Input Types Attributes

- **list**: datalist
  - A list of options for an input field
  - For text, url, email, …

```
web page: <input type="url" list="url_list"
  name="link" />
<datalist id="url_list">
  <option label="W3Schools"
  value="http://www.w3schools.com" />
  <option label="Google"
  value="http://www.google.com" />
  <option label="Microsoft"
  value="http://www.microsoft.com" />
</datalist>
```

HTML5 New Input Types Attributes

- **`pattern`**: regular expression
  - The pattern that input should match with

```
Country DNS Domain: <input type="text"
name="country_code" pattern="[A-z]{2}" />
```

- **`required`**: required
  - Input field must be filled out before submitting
- **`autofocus`**: autofocus
  - Field should automatically get focus when a page is loaded

HTML5 Forms Association

- In HTML4, input elements must be inside a form to be associated with it
- In HTML5, input elements can be associated with forms using `form` attribute

```
<form id="foo">
...
</form>
<textarea form="foo"></textarea>
```

Outline

Storing Data on the Client

- Store data on user side (client, browser)
- Earlier, this was done with cookies
  - Not suitable for large amounts of data
  - By default, passed on by EVERY request to server
    - Making it very slow and inefficient
- HTML5 offers two new objects for storing data on the client
  - **`localStorage`**: stores data with no time limit
  - **`sessionStorage`**: stores data for one session

Storing Data on the Client

- HTML5 data storage
  - Data is not passed on by every server request
    - Used in client side only when asked for
  - To store large amounts of data (~ 2-5MB configurable) in client side without affecting website's performance
    - Completely client-side web application!
      - Stores as filesystem resources for the application
  - Per-site data is stored in different areas
    - A website can only access data stored by itself
- HTML5 uses JavaScript to store and access data

sessionStorage

- To store the data for one session
- Data is deleted when session finishes
    - Typically, when the **browser** **tab** is closed!
- How to create and access sessionStorage:

```
<script type="text/javascript">
  sessionStorage.lastname="Karimi";
  var name = sessionStorage.lastname;
</script>
```

sessionStorage Example

- Count page visits in current session

```
<script type="text/javascript">
  if (sessionStorage.pagecount){

      sessionStorage.pagecount =

      Number(sessionStorage.pagecount) + 1;
  }
  else{

      sessionStorage.pagecount = 1;

  }
  document.write("Visits "+
  sessionStorage.pagecount + " time(s).");

</script>
```

localStorage

- The **localStorage** Object
- Stores data with no time limit
  - The data will be available the next day, week, …
- How to create and access a **localStorage**:

```
<script type="text/javascript">
   localStorage.lastname="Karimi";
   var name = localStorage.lastname;
</script>
```

- How to remove/clear **localStorage**:

```
<script type="text/javascript">
   localStorage.removeItem(lastname);
   localStorage.clear();
</script>
```

`localStorage` Example

```
<script type="text/javascript">
function saveNote(){
    var note = document.getElementById("mynote");
    localStorage.note=note.innerHTML;
}
function loadNote(){
    var note = document.getElementById("mynote");
    if(localStorage.note){ note.innerHTML = localStorage.note; }
    else{  note.innerHTML = "My Note"; }
}
</script>
=====================
<body onload="loadNote()">
Insert your note here
<div id="mynote" style="border-style:solid; background-color:yellow; width:
    50%;" contenteditable="true"> </div>
<input type="button" value="Save Note" onclick="saveNote()">
</body>
```

Outline

- Introduction
- Page Structure
- Multimedia
- Forms
- Storage
- Drag & Drop
- Canvas
- Other Features

Drag & Drop

- HTML5 supports drag-and-drop operations
- Move elements & text around the browser window using mouse, e.g.,
  - Move items into a shopping cart
  - Customize page layout!!
- From HTML5 point of view
  - By default, all links, text, image are draggable
  - To make an element draggable:

```
<div id="ID" draggable="true"> </div>
```

Drag & Drop: Events

- `ondragstart` event
  - Occurs in *draggable* elements when users start dragging them
- `ondragenter` event
  - Occurs in a *drop* target when a draggable move over that target
- `ondragover` event
  - Occurs in a drop *target* while users drag a draggable element over that target
  - If the drop is to be accepted, then this event (dragover) has to return false
    - If true is returned → Drop is cancelled

Drag & Drop: Events

- `ondrop` event
  - Occurs in a drop *target* while users drop a draggable element onto that target
  - To prevent further processing by browser → return false
- `ondragend` event
  - Occurs in *draggable* elements when users stop dragging them

Drag & Drop: Data Management

- Dragging objects == Carrying data
- To access the data, we use **`event`** object
  - When an objected is dragging → an event
    - **`event`** is an object that is passed to all event-handlers (we use anywhere)
    - It contains useful information (e.g., data, mouse location, …)
- **`event.dataTransfer`** contains the data
- To get the data: **`event.dataTransfer.getData`**
    - Data depends on element
    - e.g., *Text*: text, *Image*: source, *link*: href
- To set data: **`event.dataTransfer.setData`**(type, data)
  - Customize the transferred data

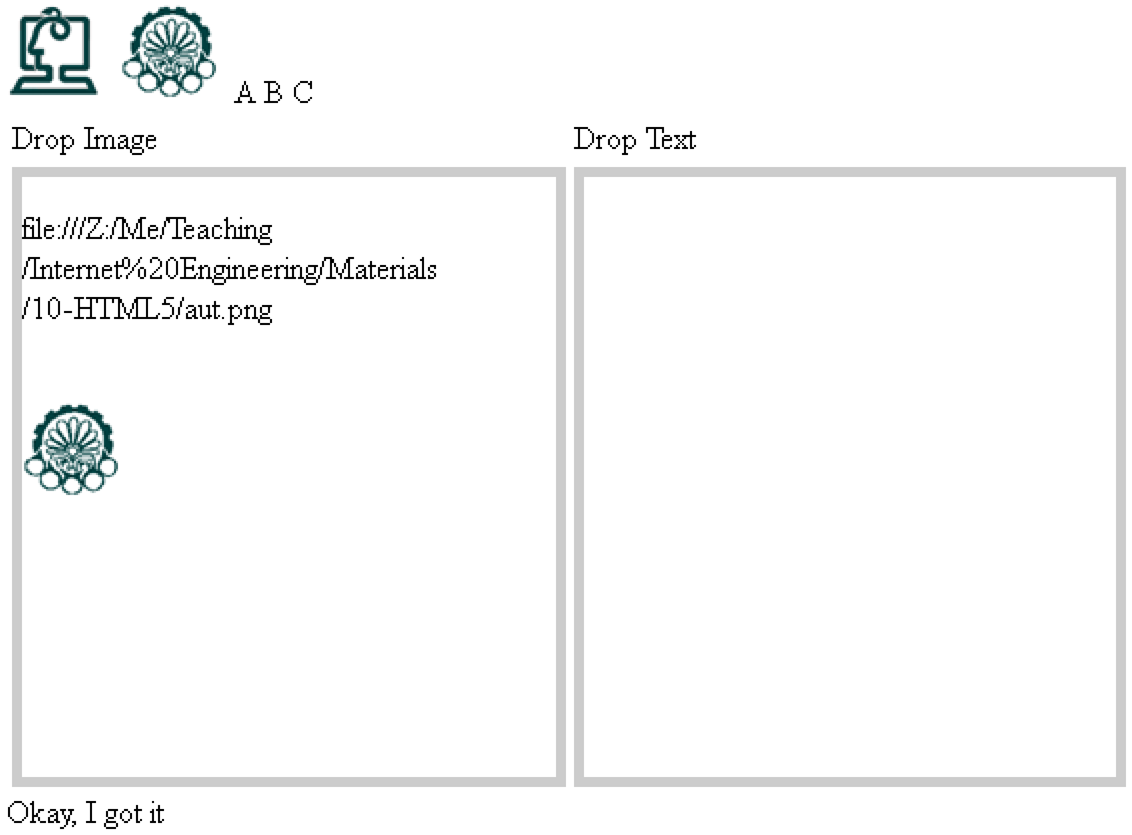
Example 1

```
<body onload="init(); init2()">
<img src="ceit.png" alt="CEIT" ondragstart="ds()"
   ondragend="de()" />
<img src="aut.png" alt="AUT" />
<span id="txt" contenteditable="true"> A B C</span>

<table>
<tr> <td>Drop Image</td> <td> Drop Text</td> </tr>
<tr> <td> <div class="d" id="drop"> </div> </td>
     <td> <div class="d" id="drop2"> </div> </td>
</tr>
</table>
<span id="msg2"> </span> <br />
<span id="msg3"> </span> <br />
</body>
```

Example 1 (cont'd)

```
function init(){
    var drop = document.getElementById("drop");
    drop.ondrop = function (event) {
         this.innerHTML += "<img src='"+event.dataTransfer.getData('Text')+"'
         alt='abc' />";
         document.getElementById("msg2").innerHTML = "Okay, I got it";
         return false;
    };
    drop.ondragover = function(){ return false; };
    drop.ondragenter = function(){
         document.getElementById("msg2").innerHTML =
         "I am ready to get the dragged object"; };
}

function init2(){
    var drop = document.getElementById("drop2");
    drop.ondrop = function (event) {
         this.innerHTML += "<p>" + event.dataTransfer.getData("Text") +
    "</p>";
         return false;
    };
    drop.ondragover = function(){ return false; };
}
```

Example 1 (cont'd)


A B C

Drop Image

Drop Text

file:///Z:/Me/Teaching
/Internet%20Engineering/Materials
/10-HTML5/aut.png

Okay, I got it

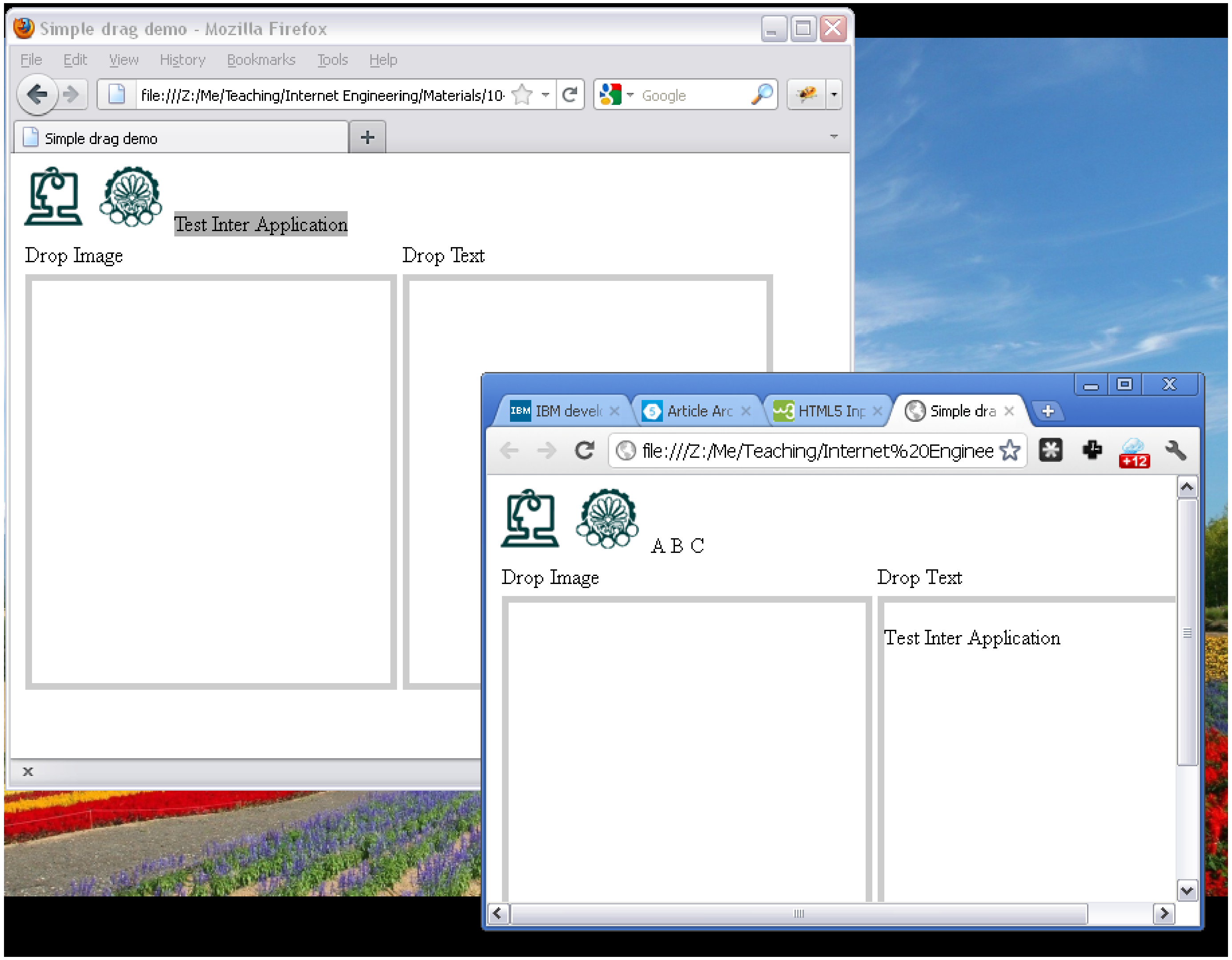
Example 2

```
<span id="txt" ondragstart="ds2(event)"
   contenteditable="true"> A B C</span>


function ds2(event){
   event.dataTransfer.setData('text/plain',
   "Data = " +
   document.getElementById("txt").innerHTML);

}
```

Dragging Objects Out of Web Page

- E.g.: Drag file from browser to download accelerator
- Only the `dragstart` and `dragend` events fire when the drag-and-drop process begins and finishes.
  - All other events are omitted because the drop point is outside of the web browser
- The information stored in `event.dataTransfer` will then be transferred to your operating system
- It is then up to the desktop or application to interpret that data correctly

Example 3


Simple drag demo - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///Z:/Me/Teaching/Internet Engineering/Materials/10·
Google
Simple drag demo
IBM devel
Article Arc
HTML5 Inp
Simple dra
file:///Z:/Me/Teaching/Internet%20Enginee

Test Inter Application

Drop Image

Drop Text

A B C

Drop Image

Drop Text

Test Inter Application

Dragging Objects Into Web Page

- E.g., upload file
- Only the **`dragenter`**, **`dragover`**, and **`drop`** events fire when dragging something from your desktop into your web page.
- All other events are omitted because the drag start point is outside of the web browser
- The information being brought into your web page can be read using **`event.dataTransfer.getData()`**,
  - The information is set by operating system/application
  - Can be a path to the local file

Outline

- Introduction
- Page Structure
- Multimedia
- Forms
- Storage
- Drag & Drop
- Canvas
- Other Features

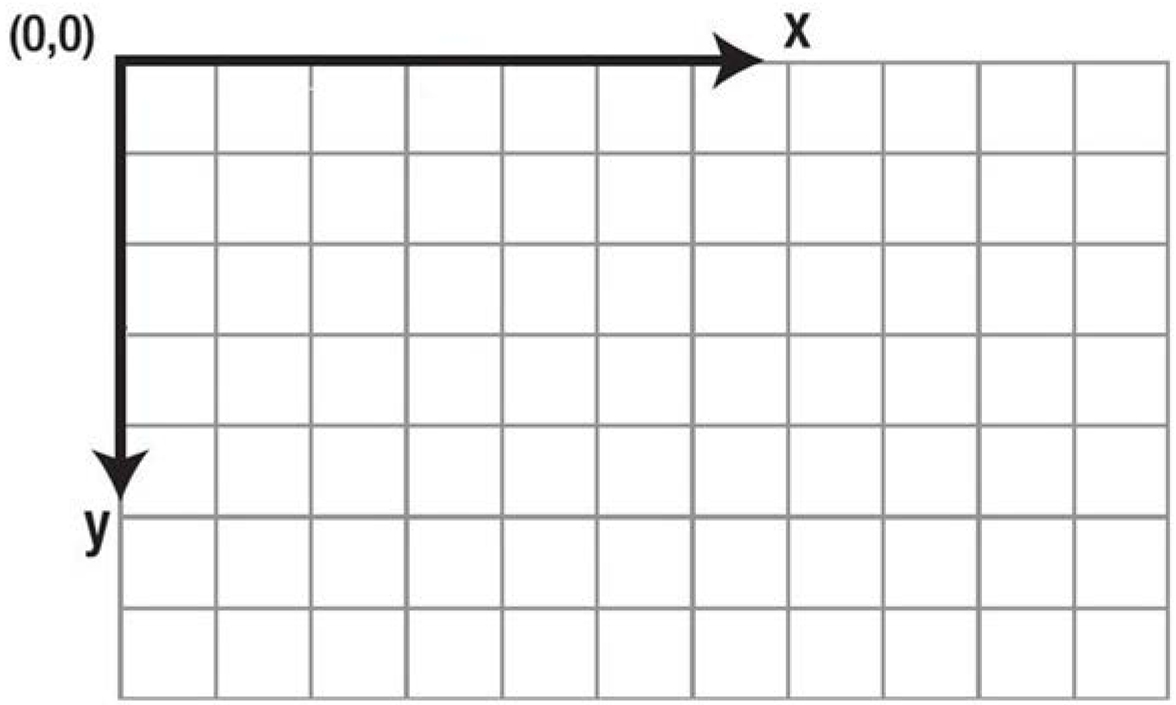
Canvas

- The canvas element provides an API for 2D drawing lines, fills, images, text, and so on
- A canvas is a rectangular area, and we can control every pixel of it
  - Nothing by itself
  - JavaScript uses Canvas to draw graphics

```
<canvas id="canvas_id" width="200"
  height="100">
  Your browser does not support Canvas
</canvas>
```

Canvas (cont'd)

- Canvas axis



Access to Canvas

- Access to the canvas in JavaScript function

```
var canvas =
   document.getElementByID("canvas_id");
var ctx = canvas.getContext("2d");
```

- A really huge API to draw lines, … using the `ctx` object

Canvas API: Drawing Processes

- Two general kinds of shapes (states)
  - Filled
    - Flooding the area within a closed shape with a color
  - Stroke
    - Drawing a line or a border with a specific thickness and color along a path, shape, or text
- Global settings
  - Styling

    ```
    ctx.fillStyle = "rgb(R, G, B)";
    ctx.strokeStyle = "rgb(R, G, B)";
    ```

  - Line width

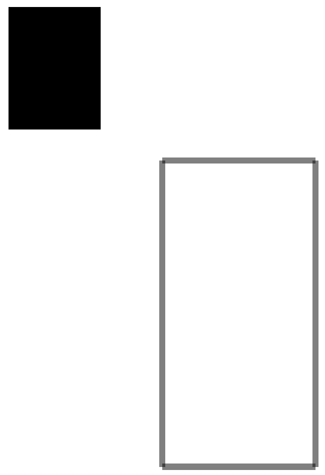
    ```
    ctx.lineWidth = 5;
    ```

Canvas API: Shapes

- Simple shapes in canvas are rectangles

```
ctx.fillRect(X, Y, W, H);
ctx.strokeRect(X, Y, W, H);
```

```
ctx.fillRect(50,50,30,40);
ctx.strokeRect(100,100,50,100);
```

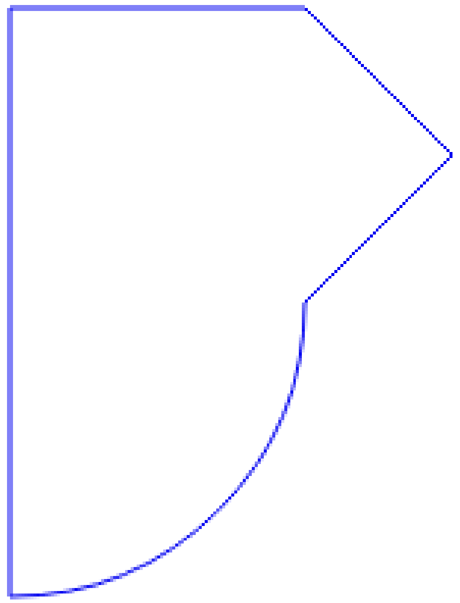


Canvas API: Shapes

- Complex shapes are created by path
  - Path is an invisible line, can be filled or stroked
- A new path: `ctx.beginPath()`
- Move the pen: `ctx.moveTo(X,Y)`
- Line: `ctx.lineTo(X,Y)`
- Arc: `ctx.arc(X,Y,R,sAngle,eAngle,anticlock);`
- Close the path: `ctx.closePath()`

Canvas API: Complex Shapes

```
ctx.beginPath();
ctx.strokeStyle = "rgb(0, 0, 240)";
ctx.arc(200,200,100,0,90 * TO_RADIANS,0);
ctx.lineTo(200, 100);
ctx.lineTo(300, 100);
ctx.lineTo(350, 150);
ctx.closePath();
ctx.stroke();
```



Canvas API: Text Drawing

- To place a single line of text, using a single font configuration, anywhere in the canvas
- `ctx.fillText(text,x,y,maxWidth)`

  `ctx.strokeText(text,x,y,maxWidth)`

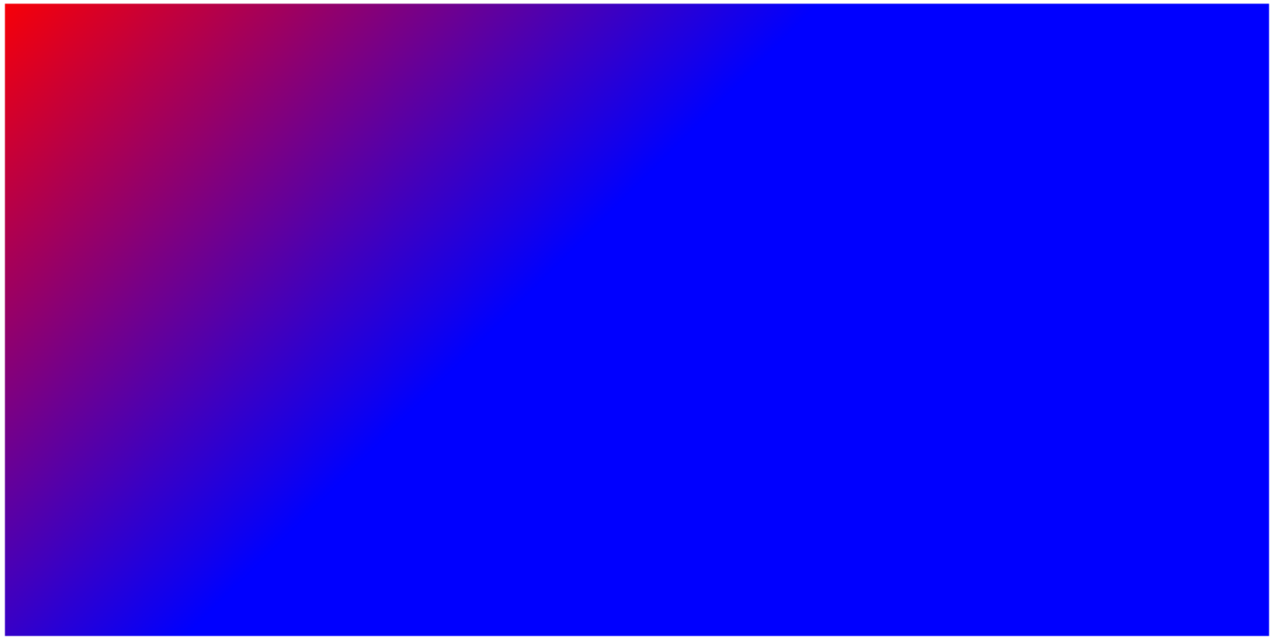
  Draws text in the current font at (x,y) using a solid **`fillStyle`**/**`strokeStyle`** setting
  - An optional maxWidth size determines how much text should actually be drawn.
- metrics = `ctx.measureText(text)`: Measures text in the current font to determine the width it would occupy without actually drawing it

Canvas API: Colors

- Color options: solid color, gradient, or pattern
  - Assign value to `fillStyle` or `strokeStyle` attribute
- Solid color: CSS-like color syntax: `red`, `#ff0000`, or `rgb(255,0,0)`
- Gradient color
  - `gradient = ctx.createLinearGradient(x0,y0,x1,y1)`
    - Creates a linear gradient that contains gradient colors that change from (x0,y0) to (x1,y1).
  - `gradient = ctx.createRadialGradient(x0,y0,r0,x1,y1,r1)`
    - Creates a radial gradient that contains gradient colors that change from (x0,y0) and a radius of r0, outward to (x1,y1) and a radius of r1.
  - The output of these function is a gradient object that first must receive at least two `addColorStop`() calls before it is assigned to `fillStyle` or `strokeStyle`

Canvas API: Colors

```
gradient = ctx.createLinearGradient(0, 0, 300, 300);
gradient.addColorStop(0, "#F00");
gradient.addColorStop(1, "#00F");
ctx.fillStyle = gradient;
ctx.fillRect(10, 10, 900, 450);
```



Canvas API: Insert Image

- Put image in canvas

```
var img=new Image();
img.src="URL";
cxt.drawImage(img,0,0);
```

Canvas API: Plane Transformations

- This transformation is applied to consequence drawings in the canvas
- `ctx.scale(x,y)`: The default state is 1x and 1y
- `ctx.translate(x,y)`: Reposition the origin to any new location point.
- `ctx.rotate(angle)`: Rotates the x- and y-axes by pivoting on the origin point.

Mouse Event Handling

- Step 1) Register desired event handlers

```
canvas.addEventListener('eventname',
  handler);
```

- Step 2) Implement the handler
  - We need mouse coordination pointer
  - It is obtained from the event object (DOM2 event handling!!), which is passed to the event handler
    - It is tricky, the event object has multiple coordination: **`event.clientX, event.clientY, event.offsetX, event.offsetY, ...`**

Save & Restore

- To save and restore canvas state
  - Not the drawings
  - **`ctx.save()`**: push state on stack
  - **`ctx.restore()`**: pop sate from stack
- Save & Restore drawing

```
imagedata=
ctx.getImageData(X,Y,width, height);
ctx.putImageData(imagedata, X, Y);
```

Save Canvas as Image

- Canvas can be saved using “Data URL”
  - A new kind of URL which is defined by HTML5

```
var canv = document.getElementById("canvas");
imageurl = canv.toDataURL("image/png");
```

  - `imageurl` contains the image in PNG format

- An easy way to save it

```
canimg = document.getElementById("canvasimage");
canimg.innerHTML='<image src="'+imageurl+'">'
```

Canvas

- Example

Outline

Offline Web Applications

- Example: Offline Gmail
- Browser should cache what application needs when it does not connect to server
  - In the simplest case, when the application does not connect to server, browser serves the needed file from its cache
- Application specifies what needs to be cached

```
<html lang="en" manifest="files.manifest">
```

Offline Web Applications

- **`files.manifest:`**

```
CACHE MANIFEST

CACHE:
file1.html
file2.js

FALLBACK:
neededfile.html fallback-replacement.html

NETWORK:
*
```

File which should be cached

Files which cannot be cached, but can be replaced by other files in offline mode

Files which should not be cached, if application is in online mode, they are retrieved from server

WebSocket

- Ajax is a mechanism to generate a HTTP request from web page (pull data from server)
- How to receive data from server (push data to client)?!
  - E.g., chat: we don't send request but we get data
- We need to listen a (TCP) socket in webpage!!
- HTML5's solution is the WebSocket

```
var socket = new WebSocket('ws://server:port/')
```

  - `socket.send()`: To send data
  - `socket.onmessage(`event`)`: To receive data
    - accessible from `event.data`
- We need a *WebSocket server* to handle the messages
  - This mechanism is not for directly browser-to-browser communication

File-API

- Extended version of drag-and-drop to access the content of files in browser
  - E.g., drag files from desktop to web page
  - In fact, it is not part of HTML5 but related!
- This is different from uploading files by <file>
  - We read the file in browser! We have access to file content
- If browser supports
  - `reader = new FileReader()`
- Different method to read files
  - **`readAsText`**: Output is string
  - **`readAsDataURL`**: Output is a data URL

Web-Workers

- Multi-threading inside web pages
  - To run JavaScript in parallel on a web page, without blocking user interface
- Multi-threading → Race condition !!!
  - Thread safety is always a big challenge
    - Accessing shared data from multiple threads
- To be thread-safe, web-works cannot access to data which is shared by the main page
  - One of the shared data is DOM → Web-workers does not access to DOM!!!
- So, if we cannot access to DOM, what/how can we do?
  - Web-Workers work by message passing
    - JS in main page creates a web-worker
      - The code of the worker is given by another JS file
    - JS in main page post & get messages from the worker

Answers

- Q9.1) Search engines don't show my page!!!
  - Organize your page by semantic tags!!
- Q9.2) I don't want use Flash for multimedia?
  - Ok! HTML5 has built in support
- Q9.3) Why JS for form validation every time?
  - Instead of HTML4 + JS use HTML5 forms
- Q9.4) Cookies are good, but I need more!!
  - HTML5 storage is for you
- Q9.5) Can I implement games under web?!
  - Simple games! Yes, use canvas for drawing
- Q9.6) I want use elements on page as objects!
  - At least you can drag & drop them in a page!!
- Q9.7) How does the Gmail off-line work?
  - It uses HTML5 off-line!!

What are the Next?!

- WebGL
  - API for rendering interactive 3D computer graphics and 2D graphics within any compatible web browser
- Touch Events
  - Set of events that represent points of contact and changes of those points with respect to DOM elements displayed upon it
- Geo Location
  - API is used to get the geographical position of a user
- MathML (mathematical markup language)
  - Describing mathematical notations, integrating mathematical formulae into Web pages

References

- Bruce Lawson and Remy Sharp, “Introducing HTML5”
- http://www.w3schools.com/html5/