# HTML5 Programming Basic Skills (2)

Wen-Hsiang Lu (盧文祥)
Department of Computer Science and Information Engineering,
National Cheng Kung University
2015/01/07

# Outline

- RichText
- Video & Audio
- New Elements of Form
- Drag & Drop

# RichText

- Make some HTML elements editable
  - Use contenteditable property

- Make a HTML document editable
  - Use designMode property

    <iframe src=“about:blank” width= “300”; height= “200”;
    onload=“this.contentDocument.designMode= ‘on’”;
    <iframe>
- Move a cursor to the end of editor:
  - html5-10-richtext-collapse.htm
- Get selection range: html5-11-richtext-selection.htm
  - select(): select all input texts

# <video> & <audio>

- Play video and voice files

```
<video src=
 "http://m98.nthu.edu.tw/~s9822543/html5_colors/colors_in_beijing.mp4" >
 </video>
<audio src="green.mp3" controls loop autoplay>Play mp3!</audio>
```

- [Ex] html5-17-video.htm

# <source> 指定播放媒體檔案和編碼器

- Use <source> tag

```
<video controls>
   <source src='xx.mp4' type='video/mp4; codec="avc1.4D401E, mp4a.40.2"'>
  </source>
</video>
```

- [Ex] html5-18-video-source.htm

# Use JavaScript to Control <video> & <audio>

- Use JavaScript API
  - load(), play(), pause()

```
<video id="video1" src="colors_in_beijing.mp4"></video>
<script>
var video1=document.getElementById("video1");
   video1.play();
</script>
```

- [Ex] html5-19-video-javascript.htm

# Video & Audio Player

```
function playOrpause(){
  var videoURL= document.getElementById("url").value;
  var video1= document.getElementById("video1");
  if (video1.paused){
    if (videoURL != video1.src){
      video1.src= videoURL;
      video1.load();
    }
    else {  video1.play();  }
    document.getElementById("pbutton").value= "Pause (暫停)";
  }
  else {
    video1.pause();
    document.getElementById("pbutton").value= "Play (播放)";
  }
}
```

[Ex] html5-20-video-player.htm

# New Elements of <Form>

- New types of <input> tag

```
<p>Search:<input type="search">
<p>Telephone:<input type="tel">
<p>URL:<input type="url">
<p>Email:<input type="email">
<p>Datetime:<input type="datetime">
<p>Date:<input type="date" min="2010-12-23" max="2011-1-22" step="2">
<p>Number:<input type="number" min="0" max="10" step="2">
<p>Range:<input type="range" min="0" max="10" step="2">
```

- [Ex] html5-21-form-newelement.htm

# File type of <input>

- <input> select multiple files via “multiple” property

```
<input id ="file" type="file" multiple accept="video/*”
 onchange="showFinfo()">
<script>
function showFinfo(){
  var selectedFiles= document.getElementById("file").files;
  var finfos= [];
  for (var i=0; i<selectedFiles.length; i++){
    var file= selectedFiles[i];
    finfos.push("name:"+file.name+"size:"+file.size);
  }
}
</script>
```

[Ex] html5-22-form-file.htm

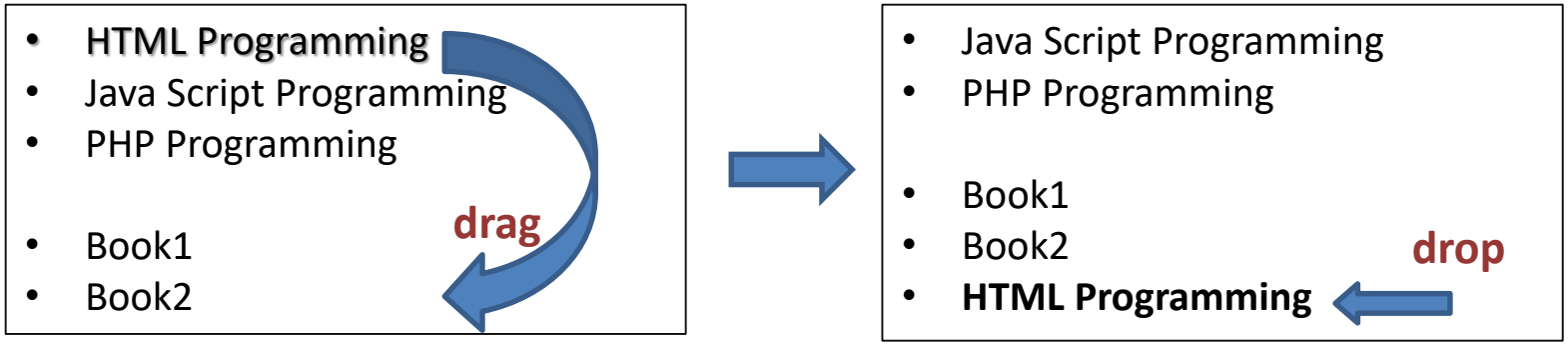
# <progress> & <meter>

- <progress>: display a visual progress bar of an executing task
  - value & max property
- <meter>: display a visual ratio bar
  - max, min & value property

```
<p>progress:<progress value= "0.2" max= "1">
<p>meter1:<meter max="1000" min="0" value="350"></meter>
<p>meter2:<meter max="1000" min="0" value="750"></meter>
```

[Ex] html5-23-form-progress.htm

# Drag & Drop API (1)




- Drag source must be set to “draggable”
  - <li id="OS" draggable="true"> HTML Programming </li>
- DataTransfer object
  - setData(type, data)
  - getData(type)
- [Ex] html5-24-drag&drop.htm

# Drag & Drop API (2)

- Relevant events:
  - onDragStart: dataTransfer.setData(type, data)
  - onDrop: dataTransfer.getData(type)
  - onDragOver: event.preventDefault()

- [Ex] html5-26-dragdrop-table.htm

# File API

- Access files at client side
- File object:
  - <input type="file" multiple>
  - Attribute: name, size
- FileReader object:
  - readAsText(fileBlob, encoding), readAsBinaryString(fileBlob)
  - result property: file content

[Ex] html5-27-filereader.htm

# Communication API

- Deliver messages between pages.
- MessageEvent object:
  - data: property
- Sender:
  - postMessage(message,domain)
- Receiver:
  - onMessage: event
  - addEventListener(“message”, process function, false);

[Ex] html5-29-parent.html

# Web worker

- The Web Workers API
  - makes it possible to execute a JavaScript file asynchronously and autonomously.
  - A web worker is essentially a thread executing a JavaScript file.
  - using web workers you can achieve multi threading in web applications.
- Worker method: worker = new Worker("html5-30-webworker.js");
- Sending message:
  - postMessage(message)
- Receiving message:
  - onMessage: event
  - addEventListener(“message”, process function, false);

[Ex] html5-30-webworker.html