HTML5

Presentation by Uplatz

Contact Us: https://training.uplatz.com/

Email: info@uplatz.com

Phone:+44 7836 212635

Uplatz

Table Of Contents:

ARIA:

role="radiogroup“:

- A group of radio buttons.

```
<div role="radiogroup">
 <input role="radio" type="radio" aria-checked="true">
One<br>
 <input role="radio" type="radio" aria-
checked="false"> Two<br>
 <input role="radio" type="radio" aria-
checked="false"> Three
</div>
```

role="row“:

- A row of cells in a tabular container.

```
<table>
 <thead>
```

Uplatz

```
<!-- etc -->
 </thead>
 <tbody>
 <tr role="row">
 <!-- etc -->
 </tr>
 </tbody>
</table>
```

role="rowgroup“:

- A group containing one or more row elements in a grid.

```
<table>
 <thead role="rowgroup">
 <!-- etc -->
```

```
</thead>
 <tbody role="rowgroup">
 <!-- etc -->
 </tbody>
</table>
```

role="rowheader“:

- A cell containing header information for a row in a grid.

```
<table role="grid">
 <thead>
 <!-- etc -->
 </thead>
 <tbody>
 <tr>
 <th role="rowheader">Day 1</th>
```

```
<td>65</td>
</tr>
<tr>
<th role="rowheader">Day 2</th>
<td>74</td>
</tr>
</tbody>
</table>
```

role="scrollbar“:

- A graphical object that controls the scrolling of content within a viewing area, regardless of whether the content is fully displayed within the viewing area.

```
<div id="content1">Lorem ipsum...</div>
<div
```

Uplatz

```
role="scrollbar"
aria-controls="content1"
aria-orientation="vertical"
aria-valuemax="100"
aria-valuemin="0"
aria-valuenow="25">
<div class="scrollhandle"></div>
</div>
```

role="search“:

- A landmark region that contains a collection of items and objects that, as a whole, combine to create a search facility.

```
<div role="search">
```

```
 <input role="searchbox" type="text">
 <button role="button">Search</button>
</div>
```

role="searchbox“:

- A type of textbox intended for specifying search criteria.

```
<div role="search">
 <input role="searchbox" type="text">
 <button role="button">Search</button>
</div>
```

role="separator“:

- A divider that separates and distinguishes sections of content or groups of menuitems.

```
<p>Lorem ipsum...</p>
```

```
<hr role="separator">
<p>Lorem ipsum...</p>
```

role="slider“:

- A user input where the user selects a value from within a given range.

```
<div
 role="slider"
 aria-valuemax="100"
 aria-valuemin="0"
 aria-valuenow="25">
 <div class="sliderhandle"></div>
</div>
```

role="spinbutton“:

- A form of range that expects the user to select from among discrete choices.

```
<input
 role="spinbutton"
 aria-valuemax="100"
 aria-valuemin="0"
 aria-valuenow="25"
 type="number"
 value="25">
```

role="status“:

- A container whose content is advisory information for the user but is not important enough to justify an alert, often but not necessarily presented as a status bar.

```
<div role="status">Online</div>
```

role="switch“:

- A type of checkbox that represents on/off values, as opposed to checked/unchecked values.

```
<select role="switch" aria-checked="false">
 <option>On</option>
 <option selected>Off</option>
</select>
```

role="tab":

- A grouping label providing a mechanism for selecting the tab content that is to be rendered to the user.

```
<ul role="tablist">
 <li role="tab">Introduction</li>
 <li role="tab">Chapter 1</li>
 <li role="tab">Chapter 2</li>
</ul>
```

.

role="table“:

- A section containing data arranged in rows and columns.
- The table role is intended for tabular containers which are not interactive.

```
<table role="table">
 <thead>
 <!-- etc -->
 </thead>
 <tbody>
 <!-- etc -->
 </tbody>
</table>
```

role="tablist“:

- A list of tab elements, which are references to tabpanel elements.

```
<ul role="tablist">
 <li role="tab">Introduction</li>
 <li role="tab">Chapter 1</li>
 <li role="tab">Chapter 2</li>
</ul>
```

role="tabpanel“:

- A container for the resources associated with a tab, where each tab is contained in a tablist.

```
<ul role="tablist">
 <li role="tab">Introduction</li>
 <li role="tab">Chapter 1</li>
 <li role="tab">Chapter 2</li>
</ul>
```

```
<div role="tabpanel">
 <!-- etc -->
</div>
```

role="textbox“:

Input that allows free-form text as its value

```
<textarea role="textbox"></textarea>
```

role="timer"

- A type of live region containing a numerical counter which indicates an amount of elapsed time from a start point, or the time remaining until an end point.

```
<p>
 <span role="timer">60</span> seconds remaining.
</p>
```

role="toolbar“:

- A collection of commonly used function buttons represented in compact visual form.

```
<ul role="toolbar">
 <li><img alt="New" src="new.png"></li>
 <li><img alt="Open" src="open.png"></li>
 <li><img alt="Save" src="save.png"></li>
 <li><img alt="Close" src="close.png"></li>
</ul>
```

role="tooltip“:

- A contextual popup that displays a description for an element.

```
<span aria-describedby="slopedesc">Slope</span>
<div role="tooltip" id="slopedesc">y=mx+b</div>
```

- Typically, the tooltip would be hidden.
- Using JavaScript, the tooltip would be displayed after a delay when the user hovers over the element that it describes.

role="tree“:

- A type of list that may contain sub-level nested groups that can be collapsed and expanded.

```
<ul role="tree">
 <li role="treeitem">
 Part 1
 <ul>
 <li role="treeitem">Chapter 1</li>
 <li role="treeitem">Chapter 2</li>
 <li role="treeitem">Chapter 3</li>
 </ul>
```

```
</li>
 <li role="treeitem">
 Part 2
 <ul>
 <li role="treeitem">Chapter 4</li>
 <li role="treeitem">Chapter 5</li>
 <li role="treeitem">Chapter 6</li>
 </ul>
 </li>
 <li role="treeitem">
 Part 3
 <ul>
<li role="treeitem">Chapter 7</li>
 <li role="treeitem">Chapter 8</li>
```

```
 <li role="treeitem">Chapter 9</li>
 </ul>
 </li>
</ul>
```

role="treegrid“:

- A grid whose rows can be expanded and collapsed in the same manner as for a tree.

role="treeitem“:

- An option item of a tree.
- This is an element within a tree that may be expanded or collapsed if it contains a sublevel group of treeitems.

```
<ul role="tree">
 <li role="treeitem">
 Part 1
 <ul>
```

```
<li role="treeitem">Chapter 1</li>
 <li role="treeitem">Chapter 2</li>
 <li role="treeitem">Chapter 3</li>
 </ul>
 </li>
 <li role="treeitem">
 Part 2
 <ul>
 <li role="treeitem">Chapter 4</li>
 <li role="treeitem">Chapter 5</li>
 <li role="treeitem">Chapter 6</li>
 </ul>
 </li>
 <li role="treeitem">
```

Part 3

```
 <ul>
 <li role="treeitem">Chapter 7</li>
 <li role="treeitem">Chapter 8</li>
 <li role="treeitem">Chapter 9</li>
 </ul>
 </li>
</ul>
```

What is Responsive Web Design?

- Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones):

Setting The Viewport

- When making responsive web pages, add the following <meta> element in all your web pages:

<meta name="viewport" content="width=device-width, initial-scale=1.0">

Using the max-width Property:

- If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size:

<img src="img_girl.jpg" style="max-width:100%;height:auto;">

Show Different Images Depending on Browser Width:

- The HTML <picture> element allows you to define different images for different browser window sizes.
- Resize the browser window to see how the image below change depending on the width:

```
<picture>
  <source srcset="img_smallflower.jpg" media="(max-width: 600px)">
  <source srcset="img_flowers.jpg" media="(max-width: 1500px)">
  <source srcset="flowers.jpg">
  <img src="img_smallflower.jpg" alt="Flowers">
</picture>
```

Responsive Text Size:

- The text size can be set with a "vw" unit, which means the "viewport width".
- That way the text size will follow the size of the browser window:

```
<h1 style="font-size:10vw">Hello World</h1>
```

Media Queries:

- In addition to resize text and images, it is also common to use media queries in responsive web pages.
- With media queries you can define completely different styles for different browser sizes.
- **Example**: resize the browser window to see that the three div elements below will display horizontally on large screens and stacked vertically on small screens:

```
<style>
.left, .right {
  float: left;
  width: 20%; /* The width is 20%, by default */
}
```

```
main {
  float: left;
  width: 60%; /* The width is 60%, by default */
}

/* Use a media query to add a breakpoint at 800px: */
@media screen and (max-width: 800px) {
  .left, .main, .right {
    width: 100%; /* The width is 100%, when the viewport is
800px or smaller */
  }
}
</style>
```

HTML Entities:

- Reserved characters in HTML must be replaced with character entities.

- Characters that are not present on your keyboard can also be replaced by entities.

HTML Entities

- Some characters are reserved in HTML.
- If you use the less than (<) or greater than (>) signs in your text, the browser might mix them with tags.
- Character entities are used to display reserved characters in HTML.

A character entity looks like this:

&entity_name;

OR

&#entity_number;

- To display a less than sign (<) we must write: < or <

Non-breaking Space:

- A common character entity used in HTML is the non-breaking space:
- A non-breaking space is a space that will not break into a new line.
- Two words separated by a non-breaking space will stick together (not break into a new line).
- This is handy when breaking the words might be disruptive.

Examples:

§ 10

10 km/h

10 PM

Combining Diacritical Marks:

- A diacritical mark is a "glyph" added to a letter.
- Some diacritical marks, like grave (`) and acute (´) are called accents.
- Diacritical marks can appear both above and below a letter, inside a letter, and between two letters.
- Diacritical marks can be used in combination with alphanumeric characters to produce a character that is not present in the character set (encoding) used in the page.

HTML Symbols:

HTML Symbol Entities:

- HTML entities were described in the previous chapter.

- Many mathematical, technical, and currency symbols, are not present on a normal keyboard.
- To add such symbols to an HTML page, you can use an HTML entity name.
- If no entity name exists, you can use an entity number, a decimal, or hexadecimal reference.

Example:

** I will display € **

** I will display € **

** I will display € **

OUTPUT:

I will display €

I will display €

I will display €

Uplatz

HTML Helpers (Plug-ins):

- Helper applications (plug-ins) are computer programs that extend the standard functionality of a web browser.

Examples of well-known plug-ins are Java applets.

- Plug-ins can be added to web pages with the <object> tag or the <embed> tag.
- Plug-ins can be used for many purposes: display maps, scan for viruses, verify your bank id, etc.

The <object> Element:

- The <object> element is supported by all browsers.
- The <object> element defines an embedded object within an HTML document.
- It is used to embed plug-ins (like Java applets, PDF readers, Flash Players) in web pages.

<object width="400" height="50"data="bookmark.swf"></object>

HTML YouTube Videos:

- Converting videos to different formats can be difficult and time-consuming.
- An easier solution is to let YouTube play the videos in your web page.

Playing a YouTube Video in HTML:

- To play your video on a web page, do the following:

Upload the video to YouTube:

- Take a note of the video id
- Define an <iframe> element in your web page
- Let the src attribute point to the video URL
- Use the width and height attributes to specify the dimension of the player

Add any other parameters to the URL (see below)

Example - Using iFrame (recommended):

YouTube Autoplay:

- You can have your video start playing automatically when a user visits that page by adding a simple parameter to your YouTube URL.
- Note: Take careful consideration when deciding to autoplay your videos. Automatically starting a video can annoy your visitor and end up causing more harm than good.
- **Value 0 (default):** The video will not play automatically when the player loads.
- **Value 1**: The video will play automatically when the player loads.

YouTube Playlist:

- A comma separated list of videos to play (in addition to the original URL).

YouTube Loop:

Value 0 (default): **The video will play only once.**

Value 1: **The video will loop (forever).**

YouTube Controls:

Value 0: Player controls does not display.

Value 1 (default): Player controls display.

YouTube - Using <object> or <embed>

- **Note:** YouTube <object> and <embed> were deprecated from January 2015.
- You should migrate your videos to use <iframe> instead.

HTML5 Geolocation:

- Locate the User's Position
- The HTML Geolocation API is used to get the geographical position of a user.
- Since this can compromise privacy, the position is not available unless the user approves it.

What is HTML Web Storage?

- With web storage, web applications can store data locally within the user's browser.
- Before HTML5, application data had to be stored in cookies, included in every server request.
- Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance.

- Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.
- Web storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

HTML Web Storage Objects:

- HTML web storage provides two objects for storing data on the client:

window.localStorage - stores data with no expiration date

window.sessionStorage - stores data for one session (data is lost when the browser tab is closed)

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.

```
// Store
localStorage.setItem("lastname", "Smith");
// Retrieve
document.getElementById("result").innerHTML =
localStorage.getItem("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 specific browser tab.

The following example counts the number of times a user has clicked a button, in the current session:

```
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.";
```

What is a Web Worker?

- 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.

Create a Web Worker File

- Now, let's create our web worker in an external JavaScript.
- Here, we create a script that counts. The script is stored in the "demo_workers.js" 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 post a message back to the HTML page.

Create a Web Worker Object

- Now that we have the web worker file, we need to call it from an HTML page.
- The following lines checks if the worker already exists, if not - it creates a new web worker object and runs the code in "demo_workers.js":

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

- Then we can send and receive messages from the web worker.
- Add an "onmessage" event listener to the web worker.

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

- When the web worker posts a message, the code within the event listener is executed.
- The data from the web worker is stored in event.data.

Terminate a Web Worker:

- When a web worker object is created, it will continue to listen for messages (even after the external script is finished) until it is terminated.
- To terminate a web worker, and free browser/computer resources, use the terminate() method:

w.terminate();

Reuse the Web Worker:

- If you set the worker variable to undefined, after it has been terminated, you can reuse the code:

w = undefined;

Full Web Worker Example Code

```
<!DOCTYPE html>
<html>
<body>
<p>Count
numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start
Worker</button>
<button onclick="stopWorker()">Stop
Worker</button>
<script>
var w;
function startWorker() {
  if (typeof(Worker) !== "undefined") {
    if (typeof(w) == "undefined") {
      w = new Worker("demo_workers.js");
    }
```

```
 w.onmessage = function(event) {
     document.getElementById("result").innerHTML =
event.data;
    };
  } else {
    document.getElementById("result").innerHTML = "
Sorry! No Web Worker support.";
  }
}
function stopWorker() {
  w.terminate();
  w = undefined;
}
</script>

</body>
</html>
```

Summary:

- ARIA
- Responsive Web Design
- Media Query
- HTML Symbols
- HTML WebWorkers
- HTML WebStorage

Thank You……

If you have any quries please write to info@uplatz.com".