Experiment No. 2

Problem: Implement HTML program for nested frame structure in TLB fashion consisting of targeting between frames. Study and implement video and audio features embedding. Also perform image mapping.

The basic concept behind frames is pretty simple:

1. Use the frameset element in place of the body element in an HTML document.
2. Use the frame element to create frames for the content of the web page.
3. Use the src attribute to identify the resource that should be loaded inside each frame.
4. Create a different file with the contents for each frame.

Let's look at a few examples of how this works. First we need a few HTML documents to work with. Let's create four different HTML documents. Here's what the first will contain:

```
<!DOCTYPE html>
<html>
   <body>
      <h1>Frame 1</h1>
      <p>Contents of Frame 1</p>
   </body>
</html>
```

The first document we'll save as frame_1.html. The other three documents will have similar contents and follow the same naming sequence.

Frames

You can divide the browser's window into multiple regions called frames. Each of the frames can contain a distinct and complete HTML document. Framed layout enables you to display several HTML documents at once. Framed layout is popular for:

##Dividing the window into a navigation frame (on the top or left-side of the window) and an actual content frame.

##Dividing the window into a small summary frame and a detail frame. Java API documentation is a good example.

However, use frames with extreme care! Frame (especially the "header" frame) occupied precious screen asset, as it does not scroll away! Framed layout have fallen out of favor over the years. Use <div> and CSS to organize your web page instead.

The tags involved are:

##<frameset>...</frameset>: to sub-divide the window.

##<frame>: defines each of the frames in a frameset.

##<iframe>...</iframe>: for floating or inline or internal frame.

##<noframe>...</noframe>: alternative text if frame is not supported by browser.

Frame Set <frameset>...</frameset>

Function: To divide the window into multiple frames, row-wise or column-wise. <frameset>...<frameset> can be nested for complex layout. <frameset>...<frameset> is to be used in place of <body>...<body> tag. i.e., there should not have an <body> tag in the document with <frameset>.

Syntax:

```
<frameset rows="list-of-row-sizes">
   ...frame-declarations...
</frameset>
 <frameset cols="list-of-column-sizes">
   ...frame-declarations...
</frameset>
```

Attributes:

##rows|cols="n|n%|*, n|n%|*, ...": a list of frame sizes in pixels or percentage separated by commas. Wildcard "*" can be used to indicate the remaining space.

Example:

```
<frameset rows="100, 20%, 150, *">
  ...frame-declarations...
</frameset>
```

Divide the window into four rows of sizes: 100 pixels, 20% of the screen, 150 pixels, and the remaining space.

Individual Frame <frame />

Function: Declare each individual frame, and place the HTML document.

Syntax:

```
<frame
  src="url"
  name="frame-name"
  noresize
  frameborder="0|1"
  scrolling="yes|no|auto" />
```

Attributes:

##src:="url": provides the URL of the document to be displayed inside this frame.

##name: specifies an unique identifier, to be used as the target of other tags, such as <a>, <form>.

##noresize: suppresses resizing by dragging of border.

##frameborder: sets to 1 to show the border, 0 to suppress.

##scrolling: "yes" to show scrollbars and allow scrolling; "no" to suppress scrolling; and "auto" to let the browser to decide (e.g., auto-hide).

Creating Vertical Columns

To create a set of four vertical columns, we need to use the frameset element with the cols attribute.

The cols attribute is used to define the number and size of columns the frameset will contain. In our case, we have four files to display, so we need four frames. To create four frames we need to assign four comma-separated values to the cols attribute.

To make things simple we're going to assign the value * to each of the frames, this will cause them to be automatically sized to fill the available space.

Here's what our HTML markup looks like.

```
<!DOCTYPE html>
<html>
<frameset cols="*,*,*,*">
   <frame src="../file_path/frame_1.html">
   <frame src="frame_2.html">
   <frame src="frame_3.html">
   <frame src="frame_4.html">
</frameset>
</html>
```

And here's how that HTML will render.

Frames arranged into columns

Creating Horizontal Rows

Rows of frames can be created by using the rows attribute rather than the cols attribute as shown in the HTML below.

```
<!DOCTYPE html>
<html>
<frameset rows="*,*,*,*">
   <frame src="frame_1.html">
   <frame src="frame_2.html">
   <frame src="frame_3.html">
   <frame src="frame_4.html">
</frameset>
</html>
```

By making that one change, the frames now load as four rows stacked up on top of eachother.

Frames arranged into rows: Mixing Columns and Rows

Columns and rows of frames can both appear on the same webpage by nesting one frameset inside of another. To do this, we first create a frameset and then nest a child frameset within the parent element. Here’s an example of how we could nest two rows within a set of three columns.

```
<frameset cols="*,*,*">
   <frameset rows="*,*">
      <frame src="frame_1.html">
      <frame src="frame_2.html">
   </frameset>
   <frame src="frame_3.html">
   <frame src="frame_4.html">
</frameset>
```

Here’s the result of that code:

First column split vertically into two rows followed by two full-height columns

Example: Divide the screen into 3 frames, top, left and right.

```
<frameset rows="50, *">
  <frame src="navigation.html" name="navigation" noresize>
  <frameset cols="100, *">
    <frame src="summary.html" name="summary">
    <frame src="detail.html" name="detail">
  </frameset>
</frameset>
```

Nested Frameset:

The nested frameset takes the place of the first frame within the parent element. The nested element can be placed in any position. For example, if we wanted the nested element to appear in the center position we would just rearrange the elements like this.

```
<frameset cols="*,*,*">
   <frame src="frame_1.html">
   <frameset rows="*,*">
      <frame src="frame_2.html">
      <frame src="frame_3.html">
   </frameset>
   <frame src="frame_4.html">
</frameset>
```

Here’s how the rearranged frames would render.

Middle column split vertically into two rows. First and third columns are full height.

Of course, we can also create additional nested frames if we want to.

```
<frameset cols="*,*">
   <frame src="frame_1.html">
   <frameset rows="*,*">
      <frame src="frame_2.html">
      <frameset cols="*,*">
         <frame src="frame_3.html">
         <frame src="frame_4.html">
      </frameset>
   </frameset>
</frameset>
```

That code creates a set of two equally sized columns. We then split the second column into two rows. Finally, we split the second row into two columns. Here's what that actually looks like.

A set of two columns with the second column split vertically into two rows, the bottom row is split horizontally into two columns.

One more way to create a combination of rows and columns is to define a grid of columns and rows in a single frameset. For example, if you wanted a grid of four equally sized frames, you could use the following code.

```
<frameset rows="*,*" cols="*,*">

   <frame src="frame_1.html">

   <frame src="frame_2.html">

   <frame src="frame_3.html">

   <frame src="frame_4.html">

</frameset>
```

The resulting grid of columns and rows looks like this.

A grid of four equally sized frames

How to Style Frames

When styling the presentation of a webpage that uses frames, there are two different types of styling to consider:

Styling within each frame.

Styling the frameset

The presentation of each frame must be defined within the source document. The presentation of the frameset must be defined within the parent document containing the frameset.

In other words, frame_1.html must be styled by CSS rules contained within frame_1.html or within a stylesheet linked to frame_1.html.

Styling Frame Source Documents

Just as with any webpage, the contents of each frame can be styled with CSS. In order to style the contents of each frame, the styles must be added to the source document itself either by linking to an external stylesheet within the source document or by adding internal or inline styles to the source document.

Considering our four source documents, CSS styles have to be applied to each document individually. Applying CSS styles to the webpage that contains the frameset will not cause those styles to apply to each individual document.

If we want to style frame_1.html we need to add styles directly to the document itself either by linking to an external style sheet or by typing them directly into the document. Here's an example of how we might do that:

```
<!DOCTYPE html>
<html>
   <head>
      <style>
         body {background: gray;}
         h1 {color: blue;}
         p {margin: 20px;}
      </style>
   </head>
   <body>
      <h1>Frame 1</h1>
      <p>Contents of Frame 1</p>
   </body>
</html>
```

If we go back to our original example with four equally-sized columns and load the frameset after making these changes to frame_1.html, we get this:

Four columns with styles applied to the first frame

Styling & Formatting the Frameset

There are a few things you can do to affect the presentation of a frameset beyond styling the documents themselves.

The size of each frame can be specified and locked.

The margin between frames can be changed.

The border around frames can be formatted.

These changes aren't made with CSS. Instead, they are made by adding attributes and values to the frame elements.

Sizing Frames

Frames can be sized either in pixels or percentages, or they can be set to automatically adjust in size based on the available space. To specify the size of a frame, insert the desired value in the cols or rows attribute.

By default, unless the attribute noresize is added to a frame, website visitors can use their mouse to drag the border between two frames to resize the frames. If this is undesirable, the attribute noresize can be applied to a frame element to prevent resizing.

Let's put both of these ideas into practice.

Let's create the following layout:

- One full-width row along the top of the webpage.
- Three columns below the top row.
- The first and third columns sizes to create left and right sidebars.
- The middle column sized to create a larger content area.

We can create this layout with the following code.

```
<frameset rows="150px,*">

  <frame noresize src="frame_1.html">

  <frameset cols="20%,*,20%">

    <frame src="frame_2.html">

    <frame src="frame_3.html">

    <frame src="frame_4.html">

  </frameset>

</frameset>
```

What that code creates is a frameset of two rows.

- The first row is 150px tall. The noresize attribute appearing on the firstframe means that it cannot be resized.

- The styles we applied earlier to frame_1.html are preserved, but only affect the contents of that frame.
- The second row expands to fill the remaining space.
- A second frameset is nested in the second row and includes three columns.
- The first and third columns will each cover 20% of the available browser window.
- The second column will resize to fill the space remaining between the first and third columns.
- Since the we did not use the noresize attribute on the columns, they will initially render based on the sizes included in the code, but a website visitor will be able to manually resize them.

That code would create a webpage that rendered like this.

1. Frames with styling and specific sizing rules applied.
2. Formatting Frame Margins & Borders.

Now that we have our layout defined, we can increase or decrease the margin between the frames and also remove the border between the frames if we wish to do so. Using the layout we created in the previous step, let’s remove the borders between the three columns, but leave the border between the upper and lower rows. Let’s also add some margin around the contents of the first frame.

```
<frameset rows="150px,*">
  <frame noresize src="frame_1.html" marginheight="15">
  <frameset cols="20%,*,20%">
    <frame src="frame_2.html" frameborder="0">
    <frame src="frame_3.html" frameborder="0">
    <frame src="frame_4.html" frameborder="0">
  </frameset>
</frameset>
```

The marginheight attribute applied to the first frame will add 15px of margin above and below the content loaded in the first frame. The frameborder value of 0 removes the borders from around the three bottom frames.

If we pull up this code in a browser, here's what it looks like.

Frames with a margin applied to the top frame and the borders removed from the remaining three frames

Targeting Frames with Links

One of the most common uses of frames is to build sticky navigation into a frame that is always visible regardless of the position of the contents of the other frames. When properly implemented, navigation links will cause new resources to load in one frame while the other frames remain static.

Anchors can be formatted to target specific frames by assigning a name attribute to a targeted frame element, and using the target attributed within the a element to load the href in the targeted frame.

If all of that is a little confusing, let's take it step-by-step.

The first step in making this happen is to assign a name to the frame where we want links to open. Using the layout we created just a minute ago, we would probably want to use the left-hand column for our navigation and the center column as our targeted frame. In order to do this, we need to assign a name to our target.

```
<frameset rows="150px,*">

  <frame noresize src="frame_1.html" marginheight="15">

  <frameset cols="20%,*,20%">

    <frame src="frame_2.html" frameborder="0">

    <frame src="frame_3.html" name="mid_col" frameborder="0">

    <frame src="frame_4.html" frameborder="0">

  </frameset>

</frameset>
```

Now that we've named the center column name="mid_col" we can create a couple of links in our left-hand column source document frame_2.html and target the center column.

```
<!DOCTYPE html>
<html>
<body>
   <h1>Frame 2</h1>
   <p>Contents of Frame 2</p>
   <ul>
   <li><a href="frame_1.html" target="mid_col">Load frame_1.html</a></li>
   <li><a href="frame_2.html" target="mid_col">Load frame_2.html</a></li>
   <li><a href="frame_3.html" target="mid_col">Load frame_3.html</a></li>
   <li><a href="frame_4.html" target="mid_col">Load frame_4.html</a></li>
   </ul>
</body>
</html>
```

Now when we load our webpage we have four navigation links in the left sidebar, and when we click a link the contents of that file are loaded in the middle column frame with the attribute name="mid_col".

When we load our page here’s what we initially see.

A frameset with links displayed in a column on the left-hand side of the page

If we click the link Load frame_1.html the contents of that file are loaded in the center column and we get this.

A frameset with a navigation column on the left-hand side and a link opened in the middle frame

If we click the link Load frame_2.html we see the navigation contents both in the left sidebar and the middle column.

Clicking the links Load frame_3.html and Load frame_4.html would behave exactly as you’d expect, loading the contents of those files in the middle column.

If we forgot to add the target="mid_col" attribute to one of the links, when we clicked the link the resource will load in the same frame that contained the link. If we want to reload the entire page, such as when linking to an external website, we need to add the target="_blank" or target="_top" attribute to the anchor element.

Providing a noframes Fallback

In the past, the noframes element was used to create a fallback for browsers that did not support the use of frames. However, all modern browsers do support frames for the time being, and support for noframes is virtually nonexistent. As a result, it is no longer necessary to have a noframes fallback when working with frames.

How to Make Frames Responsive

One of the problems with frames is that they tend to create usability issues for website visitors using smartphones and small tablets. Since frames have been removed from HTML5 entirely and deemed obsolete, it’s important that the owners of websites built with frames begin planning a redesign that does not include frames. However, until a full redesign can be completed, there are a few things webmasters can do to improve the usability of framed designs.

Use Rows Rather than Columns

If possible, organize frames into rows rather than columns. It is much easier to navigate through content vertically than horizontally on a small screen and frames that are arranged into rows are much easier to view on a small screen than those arranged into columns.

If we compress our column and row layouts onto a simulation of an Apple iPhone 6 screen we can see how rows are easier to view than columns.

Comparison of how rows and columns render on a small screen

If you had to look at these two sites and try to read them which would you rather look at?

Use Percentages for Column Widths

When columns are sized using percentages rather than pixels they will automatically resize based on the size of the device being used to view the site. While this may create some issues if certain frames become too small, a visitor’s experience will be better overall when column widths are assigned based on percentages rather than pixels.

How to Migrate Away from Frames

Both the frameset and frame elements have been removed from the latest HTML specification, HTML5. Owners of websites built with frames should complete a redesign of their website to remove frames from the design of the site.

At some point, web browsers will drop support for frames. When that happens, websites that still rely on frames will become unusable. As a result, transitioning away from frames isn’t just a good idea, it’s mandatory.

Evaluate Framed Content

The first step to redesigning a website that uses frames is to think through the reasons why frames were used in the first place.

Were frames used to create a specific layout? If so, CSS can be used to create a similar layout.

Are frames used to create a specifically sized ad container? There are many ways to duplicate that effect using CSS or a widget designed to work with a content management system.

Was a frame used to create a sticky navigation menu? Once again, CSS can duplicate this same effect.

Were frames used to load a resource from an external website? If so, the iframe element, which is part of HTML5, can be used to embed content from an external website.

In virtually every case CSS can be used to duplicate the layout created with frames, and iframes can be used to embed external resources.

Targeting Named Frame

To control which frame to receive the content of a hyperlink, you could include the target attribute in the <a> tag. Using the previous example, a hyperlink in the "navigation" frame targeting the "detail" frame is:

```
<a href="chapter5.html" target="detail">Chapter 5</a>
```

If all the links in the "navigation" frame are targeting the "detail" frame, you can use the <base> tag (in the HEAD section) to set up a global target reference:

```
<!-- navigation.html -->

<head>

  <title>navigation bar</title>
  <base target="detail">
</head>
```

As mentioned, frames has gone out of favor these day. The HTML5 introduced new elements such as <header>, <footer>, <section>, <nav>, <article> to help you in organizing your web page, instead of using frame, division, or table.

No Frame <noframe>...</noframe>

Function: Provides an alternative text if the browser does not support framed layout. <noframe> must be placed within a <frameset>.

Syntax (Container tag):

```
<noframe>...alternative-text...</noframe>
```

Iframe (Inline Frame or Internal Frame) <iframe>...</iframe>

Function: Place an inline frame on the browser window. An iframe is a frame that can be embedded within a regular HTML page, which contains a separate and complete HTML document, set via its src attribute.

Syntax (Container tag):

```
<iframe
   src="URL"
   name="frame-name"
   frameborder="0|1"
   scrolling="yes|no|auto">
     ...alternative-text...
</iframe>
```

You could use CSS to position and style the iframe. Iframe is used extensively by JavaScript and Ajax.

Introduction to frames

HTML frames allow authors to present documents in multiple views, which may be independent windows or subwindows. Multiple views offer designers a way to keep certain information visible, while other views are scrolled or replaced. For example, within the same window, one frame might display a static banner, a second a navigation menu, and a third the main document that can be scrolled through or replaced by navigating in the second frame.

Here is a simple frame document:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
  "http://www.w3.org/TR/html4/frameset.dtd">
<HTML>
<HEAD>
<TITLE>A simple frameset document</TITLE>
</HEAD>
<FRAMESET cols="20%, 80%">
  <FRAMESET rows="100, 200">
      <FRAME src="contents_of_frame1.html">
      <FRAME src="contents_of_frame2.gif">
  </FRAMESET>
  <FRAME src="contents_of_frame3.html">
  <NOFRAMES>
      <P>This frameset document contains:
      <UL>
         <LI><A href="contents_of_frame1.html">Some neat contents</A>
         <LI><IMG src="contents_of_frame2.gif" alt="A neat image">
         <LI><A href="contents_of_frame3.html">Some other neat contents</A>
      </UL>
  </NOFRAMES>
</FRAMESET>
</HTML>
```

If the user agent can't display frames or is configured not to, it will render the contents of the NOFRAMES element.

Layout of frames

An HTML document that describes frame layout (called a frameset document) has a different makeup than an HTML document without frames. A standard document has one HEAD section and one BODY. A frameset document has a HEAD, and a FRAMESET in place of the BODY.

The FRAMESET section of a document specifies the layout of views in the main user agent window. In addition, the FRAMESET section can contain a NOFRAMES element to provide alternate content for user agents that do not support frames or are configured not to display frames.

Elements that might normally be placed in the BODY element must not appear before the first FRAMESET element or the FRAMESET will be ignored.

The FRAMESET element

```
<![ %HTML.Frameset; [

<!ELEMENT FRAMESET - - ((FRAMESET|FRAME)+ & NOFRAMES?) --
window subdivision-->

<!ATTLIST FRAMESET

  %coreattrs;                    -- id, class, style, title --

  rows       %MultiLengths; #IMPLIED  -- list of lengths,

                             default: 100% (1 row) --

  cols       %MultiLengths; #IMPLIED  -- list of lengths,

                             default: 100% (1 col) --

  onload     %Script;      #IMPLIED  -- all the frames have been loaded  --

  onunload   %Script;      #IMPLIED  -- all the frames have been removed --

  >

]]>
```

Attribute definitions

rows = multi-length-list [CN]

This attribute specifies the layout of horizontal frames. It is a comma-separated list of pixels, percentages, and relative lengths. The default value is 100%, meaning one row.

cols = multi-length-list [CN]

This attribute specifies the layout of vertical frames. It is a comma-separated list of pixels, percentages, and relative lengths. The default value is 100%, meaning one column.

Attributes defined elsewhere

• id, class (document-wide identifiers)

•title (element title)

• style (inline style information)

•onload, onunload (intrinsic events)

The FRAMESET element specifies the layout of the main user window in terms of rectangular subspaces.

Rows and columns

Setting the rows attribute defines the number of horizontal subspaces in a frameset. Setting the cols attribute defines the number of vertical subspaces. Both attributes may be set simultaneously to create a grid. If the rows attribute is not set, each column extends the entire length of the page. If the cols attribute is not set, each row extends the entire width of the page. If neither attribute is set, the frame takes up exactly the size of the page.

Frames are created left-to-right for columns and top-to-bottom for rows. When both attributes are specified, views are created left-to-right in the top row, left-to-right in the second row, etc.

The first example divides the screen vertically in two (i.e., creates a top half and a bottom half).

```
<FRAMESET rows="50%, 50%">

...the rest of the definition...

</FRAMESET>
```

The next example creates three columns: the second has a fixed width of 250 pixels (useful, for example, to hold an image with a known size). The first receives 25% of the remaining space and the third 75% of the remaining space.

```
<FRAMESET cols="1*,250,3*">
```

...the rest of the definition...

</FRAMESET>

The next example creates a 2x3 grid of subspaces.

<FRAMESET rows="30%,70%" cols="33%,34%,33%">

...the rest of the definition...

</FRAMESET>

For the next example, suppose the browser window is currently 1000 pixels high. The first view is allotted 30% of the total height (300 pixels). The second view is specified to be exactly 400 pixels high. This leaves 300 pixels to be divided between the other two frames. The fourth frame's height is specified as "2*", so it is twice as high as the third frame, whose height is only "*" (equivalent to 1*). Therefore the third frame will be 100 pixels high and the fourth will be 200 pixels high.

<FRAMESET rows="30%,400,*,2*">

...the rest of the definition...

</FRAMESET>

Absolute lengths that do not sum to 100% of the real available space should be adjusted by the user agent. When underspecified, remaining space should be allotted proportionally to each view. When overspecified, each view should be reduced according to its specified proportion of the total space.

Nested frame sets

Framesets may be nested to any level.

In the following example, the outer FRAMESET divides the available space into three equal columns. The inner FRAMESET then divides the second area into two rows of unequal height.

```
<FRAMESET cols="33%, 33%, 34%">
    ...contents of first frame...
    <FRAMESET rows="40%, 50%">
```

```
        ...contents of second frame, first row...
        ...contents of second frame, second row...
    </FRAMESET>
    ...contents of third frame...
</FRAMESET>
```

Sharing data among frames

Authors may share data among several frames by including this data via an OBJECT element. Authors should include the OBJECT element in the HEAD element of a frameset document and name it with the id attribute. Any document that is the contents of a frame in the frameset may refer to this identifier.

The following example illustrates how a script might refer to an OBJECT element defined for an entire frameset:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
  "http://www.w3.org/TR/html4/frameset.dtd">
<HTML>
<HEAD>
<TITLE>This is a frameset with OBJECT in the HEAD</TITLE>
<!-- This OBJECT is not rendered! -->
<OBJECT id="myobject" data="data.bar"></OBJECT>
</HEAD>
<FRAMESET>
    <FRAME src="bianca.html" name="bianca">
</FRAMESET>
</HTML>
<!-- In bianca.html -->
<HTML>
```

```
<HEAD>
<TITLE>Bianca's page</TITLE>
</HEAD>
<BODY>
...the beginning of the document...
<P>
<SCRIPT type="text/javascript">
parent.myobject.myproperty
</SCRIPT>
...the rest of the document...
</BODY>
</HTML>
```

The FRAME element

```
<![ %HTML.Frameset; [
<!-- reserved frame names start with "_" otherwise starts with letter -->
<!ELEMENT FRAME - O EMPTY             -- subwindow -->
<!ATTLIST FRAME
  %coreattrs;                        -- id, class, style, title --
  longdesc    %URI;         #IMPLIED  -- link to long description
                            (complements title) --
  name       CDATA         #IMPLIED  -- name of frame for targetting --
  src        %URI;         #IMPLIED  -- source of frame content --
  frameborder (1|0)        1        -- request frame borders? --
  marginwidth %Pixels;       #IMPLIED  -- margin widths in pixels --
  marginheight %Pixels;      #IMPLIED  -- margin height in pixels --
  noresize    (noresize)     #IMPLIED  -- allow users to resize frames? --
  scrolling   (yes|no|auto)  auto      -- scrollbar or none --
  >
]]>
```

Attribute definitions

name = cdata [CI]

This attribute assigns a name to the current frame. This name may be used as the target of subsequent links.

longdesc = uri [CT]

This attribute specifies a link to a long description of the frame. This description should supplement the short description provided using the title attribute, and may be particularly useful for non-visual user agents.

src = uri [CT]

This attribute specifies the location of the initial contents to be contained in the frame.

noresize [CI]

When present, this boolean attribute tells the user agent that the frame window must not be resizeable.

scrolling = auto|yes|no [CI]

This attribute specifies scroll information for the frame window. Possible values
•auto: This value tells the user agent to provide scrolling devices for the frame window when necessary. This is the default value.

•yes: This value tells the user agent to always provide scrolling devices for the frame window.

•no: This value tells the user agent not to provide scrolling devices for the frame window.

frameborder = 1|0 [CN]

This attribute provides the user agent with information about the frame border. Possible values: •1: This value tells the user agent to draw a separator between this frame and every adjoining frame. This is the default value.

•0: This value tells the user agent not to draw a separator between this frame and every adjoining frame. Note that separators may be drawn next to this frame nonetheless if specified by other frames.

marginwidth = pixels [CN]

This attribute specifies the amount of space to be left between the frame's contents in its left and right margins. The value must be greater than zero (pixels). The default value depends on the user agent.

marginheight = pixels [CN]

This attribute specifies the amount of space to be left between the frame's contents in its top and bottom margins. The value must be greater than zero (pixels). The default value depends on the user agent.

Attributes defined elsewhere

• id, class (document-wide identifiers)

•title (element title)

• style (inline style information)

The FRAME element defines the contents and appearance of a single frame.

Setting the initial contents of a frame

The src attribute specifies the initial document the frame will contain.

The following example HTML document:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"

  "http://www.w3.org/TR/html4/frameset.dtd">

<HTML>
```

```
<HEAD>
<TITLE>A frameset document</TITLE>
</HEAD>
<FRAMESET cols="33%,33%,33%">
  <FRAMESET rows="*,200">
      <FRAME src="contents_of_frame1.html">
      <FRAME src="contents_of_frame2.gif">
  </FRAMESET>
  <FRAME src="contents_of_frame3.html">
  <FRAME src="contents_of_frame4.html">
</FRAMESET>
</HTML>
```

and cause the user agent to load each file into a separate view.

The contents of a frame must not be in the same document as the frame's definition.

ILLEGAL EXAMPLE:

The following frameset definition is not legal HTML since the contents of the second frame are in the same document as the frameset.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
<HTML>
<HEAD>
<TITLE>A frameset document</TITLE>
</HEAD>
<FRAMESET cols="50%,50%">
```

```
 <FRAME src="contents_of_frame1.html">

 <FRAME src="#anchor_in_same_document">

 <NOFRAMES>

 ...some text...

 <H2><A name="anchor_in_same_document">Important section</A></H2>

 ...some text...

 </NOFRAMES>

</FRAMESET>

</HTML>
```

- Visual rendering of a frame

The following example illustrates the usage of the decorative FRAME attributes. We specify that frame 1 will allow no scroll bars. Frame 2 will leave white space around its contents (initially, an image file) and the frame will not be resizeable. No border will be drawn between frames 3 and 4. Borders will be drawn (by default) between frames 1, 2, and 3.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"

   "http://www.w3.org/TR/html4/frameset.dtd">

<HTML>

<HEAD>

<TITLE>A frameset document</TITLE>

</HEAD>

<FRAMESET cols="33%,33%,33%">

 <FRAMESET rows="*,200">

     <FRAME src="contents_of_frame1.html" scrolling="no">

     <FRAME src="contents_of_frame2.gif"

              marginwidth="10" marginheight="15"

              noresize>
```

```
  </FRAMESET>
  <FRAME src="contents_of_frame3.html" frameborder="0">
  <FRAME src="contents_of_frame4.html" frameborder="0">
</FRAMESET>
</HTML>
```

Specifying target frame information

Note. For information about current practice in determining the target of a frame, please consult the notes on frames in the appendix.

Attribute definitions

target = frame-target [CI]

This attribute specifies the name of a frame where a document is to be opened.

By assigning a name to a frame via the name attribute, authors can refer to it as the "target" of links defined by other elements. The target attribute may be set for elements that create links (A, LINK), image maps (AREA), and forms (FORM).

Please consult the section on target frame names for information about recognized frame names.

This example illustrates how targets allow the dynamic modification of a frame's contents. First we define a frameset in the document frameset.html, shown here:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
<HTML>
<HEAD>
<TITLE>A frameset document</TITLE>
</HEAD>
<FRAMESET rows="50%,50%">
   <FRAME name="fixed" src="init_fixed.html">
   <FRAME name="dynamic" src="init_dynamic.html">
```

```
</FRAMESET>
</HTML>
```

Then, in init_dynamic.html, we link to the frame named "dynamic".

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<TITLE>A document with anchors with specific targets</TITLE>
</HEAD>
<BODY>
...beginning of the document...
<P>Now you may advance to
   <A href="slide2.html" target="dynamic">slide 2.</A>
...more document...
<P>You're doing great. Now on to
   <A href="slide3.html" target="dynamic">slide 3.</A>
</BODY>
</HTML>
```

Activating either link opens a new document in the frame named "dynamic" while the other frame, "fixed", maintains its initial contents.

Note. A frameset definition never changes, but the contents of one of its frames can. Once the initial contents of a frame change, the frameset definition no longer reflects the current state of its frames.

There is currently no way to encode the entire state of a frameset in a URI. Therefore, many user agents do not allow users to assign a bookmark to a frameset.

Framesets may make navigation forward and backward through your user agent's history more difficult for users.

Setting the default target for links

When many links in the same document designate the same target, it is possible to specify the target once and dispense with the target attribute of each element. This is done by setting the target attribute of the BASE element

We return to the previous example, this time factoring the target information by defining it in the BASE element and removing it from the A elements.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

  "http://www.w3.org/TR/html4/loose.dtd">

<HTML>

<HEAD>

<TITLE>A document with BASE with a specific target</TITLE>

<BASE href="http://www.mycom.com/Slides" target="dynamic">

</HEAD>

<BODY>

...beginning of the document...

<P>Now you may advance to <A href="slide2.html">slide 2.</A>

...more document...

<P>You're doing great. Now on to

      <A href="slide3.html">slide 3.</A>

</BODY>

</HTML>
```

Target semantics

User agents should determine the target frame in which to load a linked resource according to the following precedences (highest priority to lowest):

1.If an element has its target attribute set to a known frame, when the element is activated (i.e., a link is followed or a form is processed), the resource designated by the element should be loaded into the target frame.

2.If an element does not have the target attribute set but the BASE element does, the BASE element's target attribute determines the frame.

3.If neither the element nor the BASE element refers to a target, the resource designated by the element should be loaded into the frame containing the element.

4.If any target attribute refers to an unknown frame F, the user agent should create a new window and frame, assign the name F to the frame, and load the resource designated by the element in the new frame.

User agents may provide users with a mechanism to override the target attribute.

Alternate content

Authors should supply alternate content for those user agents that do not support frames or are configured not to display frames.

The NOFRAMES element

```
<![ %HTML.Frameset; [

<!ENTITY % noframes.content "(BODY) -(NOFRAMES)">

]]>

<!ENTITY % noframes.content "(%flow;)*">

<!ELEMENT NOFRAMES - - %noframes.content;

 -- alternate content container for non frame-based rendering -->

<!ATTLIST NOFRAMES

  %attrs;                              -- %coreattrs, %i18n, %events --

  >
```

Attributes defined elsewhere

• id, class (document-wide identifiers)

•lang (language information), dir (text direction)

•title (element title)

• style (inline style information)

•onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup (intrinsic events)

The NOFRAMES element specifies content that should be displayed only by user agents that do not support frames or are configured not to display frames. User agents that support frames must only display the contents of a NOFRAMES declaration when configured not to display frames. User agents that do not support frames must display the contents of NOFRAMES in any case.

The NOFRAMES element is part of both the transitional and frameset DTDs. In a document that uses the frameset DTD, NOFRAMES may be used at the end of the FRAMESET section of the document.

For example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
    "http://www.w3.org/TR/html4/frameset.dtd">
  <HTML>
  <HEAD>
  <TITLE>A frameset document with NOFRAMES</TITLE>
  </HEAD>
  <FRAMESET cols="50%, 50%">
    <FRAME src="main.html">
    <FRAME src="table_of_contents.html">
    <NOFRAMES>
    <P>Here is the <A href="main-noframes.html">
            non-frame based version of the document.</A>
    </NOFRAMES>
  </FRAMESET>
  </HTML>
```

NOFRAMES may be used, for example, in a document that is the source of a frame and that uses the transitional DTD. This allows authors to explain the document's purpose in cases when it is viewed out of the frameset or with a user agent that doesn't support frames.

Long descriptions of frames

The longdesc attribute allows authors to make frame documents more accessible to people using non-visual user agents. This attribute designates a resource that provides a long description of the frame. Authors should note that long descriptions associated with frames are attached to the frame, not the frame's contents. Since the contents may vary over time, the initial long description is likely to become inappropriate for the frame's later contents. In particular, authors should not include an image as the sole content of a frame.

The following frameset document describes two frames. The left frame contains a table of contents and the right frame initially contains an image of an ostrich:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
  "http://www.w3.org/TR/html4/frameset.dtd">
<HTML>
<HEAD>
<TITLE>A poorly-designed frameset document</TITLE>
</HEAD>
<FRAMESET cols="20%, 80%">
  <FRAME src="table_of_contents.html">
  <FRAME src="ostrich.gif" longdesc="ostrich-desc.html">
</FRAMESET>
</HTML>
```

Note that the image has been included in the frame independently of any HTML element, so the author has no means of specifying alternate text other than via the longdesc attribute. If the contents of the right frame change (e.g., the user selects a rattlesnake from the table of contents), users will have no textual access to the frame's new content.

Thus, authors should not put an image directly in a frame. Instead, the image should be specified in a separate HTML document, and therein annotated with the appropriate alternate text:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
  "http://www.w3.org/TR/html4/frameset.dtd">
<HTML>
<HEAD>
<TITLE>A well-designed frameset document</TITLE>
</HEAD>
<FRAMESET cols="20%, 80%">
  <FRAME src="table_of_contents.html">
  <FRAME src="ostrich-container.html">
```

```
</FRAMESET>
</HTML>
<!-- In ostrich-container.html: -->
<HTML>
<HEAD>
<TITLE>The fast and powerful ostrich</TITLE>
</HEAD>
<P>
<OBJECT data="ostrich.gif" type="image/gif">
These ostriches sure taste good!
</OBJECT>
</HTML>
```

Inline frames:

the IFRAME element

```
<!ELEMENT IFRAME - - (%flow;)*         -- inline subwindow -->
<!ATTLIST IFRAME
  %coreattrs;                          -- id, class, style, title --
  longdesc    %URI;         #IMPLIED  -- link to long description
                                       (complements title) --
  name        CDATA         #IMPLIED  -- name of frame for targetting --
  src         %URI;         #IMPLIED  -- source of frame content --
  frameborder (1|0)         1         -- request frame borders? --
  marginwidth %Pixels;      #IMPLIED  -- margin widths in pixels --
  marginheight %Pixels;     #IMPLIED  -- margin height in pixels --
  scrolling   (yes|no|auto) auto      -- scrollbar or none --
  align       %IAlign;      #IMPLIED  -- vertical or horizontal alignment --
  height      %Length;      #IMPLIED  -- frame height --
  width       %Length;      #IMPLIED  -- frame width --
```

>

Attribute definitions

longdesc = uri [CT]

This attribute specifies a link to a long description of the frame. This description should supplement the short description provided using the title attribute, and is particularly useful for non-visual user agents.

name = cdata [CI]

This attribute assigns a name to the current frame. This name may be used as the target of subsequent links.

width = length [CN]

The width of the inline frame.

height = length [CN]

The height of the inline frame.

Attributes defined elsewhere

• id, class (document-wide identifiers)

•title (element title)

• style (inline style information)

•name, src, frameborder, marginwidth, marginheight, scrolling (frame controls and decoration)

• align (alignment)

The IFRAME element allows authors to insert a frame within a block of text. Inserting an inline frame within a section of text is much like inserting an object via the OBJECT element: they both allow you to insert an HTML document in the middle of another, they may both be aligned with surrounding text, etc.

The information to be inserted inline is designated by the src attribute of this element. The contents of the IFRAME element, on the other hand, should only be displayed by user agents that do not support frames or are configured not to display frames.

For user agents that support frames, the following example will place an inline frame surrounded by a border in the middle of the text.

```
<IFRAME src="foo.html" width="400" height="500"
        scrolling="auto" frameborder="1">
[Your user agent does not support frames or is currently configured
not to display frames. However, you may visit
<A href="foo.html">the related document.</A>]
</IFRAME>
```

Inline frames may not be resized (and thus, they do not take the noresize attribute).

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document. A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

Disadvantages of Frames There are few drawbacks with using frames, so it's never recommended to use frames in your webpages: • Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up. • Sometimes your page will be displayed differently on different computers due to different screen resolution. • The browser's back button might not work as the user hopes. • There are still few browsers that do not support frame technology. Creating Frames To use frames on a page we use tag instead of tag.

The tag defines, how to divide the window into frames. The rows attribute of tag defines horizontal frames and cols attribute defines vertical frames. Each frame is indicated by tag and it defines which HTML document shall open into the frame. Example Following is the example to create three horizontal frames: This will produce the following result: Example Let's put the above example as follows, here we replaced rows attribute by cols and changed their width. This will create all the three frames vertically: This will produce the following result: The Tag Attributes Following are important attributes of the tag: Attribute Description cols Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of the four ways: Absolute values in pixels. For example, to create three vertical frames, use cols="100, 500,100". A percentage of the browser window. For example, to create three vertical frames, use cols="10%, 80%,10%". Using a wildcard symbol. For example, to create three vertical frames, use cols="10%, *,10%". In this case wildcard takes remainder of the window. As relative widths of the browser window. For example, to create three vertical frames, use cols="3*,2*,1*". This is an alternative to percentages. You can use relative widths of the browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth. rows This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example, to create two horizontal frames, use rows="10%, 90%". You can specify the height of each row in the same way as explained above for columns. border This attribute specifies the width of the border of each frame in pixels. For example, border="5". A value of zero means no border. frameborder This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example frameborder="0" specifies no border. HTML 79 framespacing This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example framespacing="10" means there should be 10 pixels spacing between each frames. The Tag Attributes Following are the important attributes of tag: Attribute Description src This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src="/html/top_frame.htm" will load an HTML file available in html directory. name This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link. frameborder This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the tag if one is given, and this can take values either 1 (yes) or 0 (no). marginwidth This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth="10". marginheight This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight="10". noresize By default, you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize="noresize". scrolling This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling="no" means it should not have scroll bars. HTML

80 longdesc This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc="framedescription.htm" Browser Support forFrames If a user is using any old browser or any browser, which does not support frames then element should be displayed to the user. So you must place a element inside the element because the element is supposed to replace the element, but if a browser does not understand element then it should understand what is inside the element which is contained in a element. You can put some nice message for your user having old browsers. For example, Sorry!! your browser does not support frames. as shown in the above example. Frame's name and target attributes One of the most popular uses of frames is to place navigation bars in one frame and then load main pages into a separate frame. Let's see following example where a test.htm file has following code: Here, we have created two columns to fill with two frames. The first frame is 200 pixels wide and will contain the navigation menu bar implemented by menu.htm file. The second column fills in remaining space and will contain the main part of the page and it is implemented by main.htm file. For all the three links available in menu bar, we have HTML 81 mentioned target frame as main_page, so whenever you click any of the links in menu bar, available link will open in main page. Following is the content of menu.htm file When we load test.htm file, it produces following result: Now you can try to click links available in the left panel and see the result. The targetattribute can also take one of the following values: Option Description _self Loads the page into the current frame. _blank Loads a page into a new browser window.opening a new window. HTML 82 _parent Loads the page into the parent window, which in the case of a single frameset is the main browser window. _top Loads the page into the browser window, replacing any current frames. targetframe Loads the page into a named targetframe.

Setting up a Link Targeting a Specific HTML Element via the id Attribute

You can setup a link to target a specific HTML element (typically a heading <h1> to <h6>), similar to bookmark, via:

1. Define an id="*idName*" attribute for the targeted element. The id attribute is applicable to ALL HTML elements, including <h1> to <h6>. The id-value is supposed to be unique in the document (i.e., no two elements shall have the same id-value). An anchor name (or bookmark) called *idname* will be set up automatically on the element. You can refer to the anchor point via #*idName*, by prefixing with a # sign.

2. Setup a link targeting the anchor point, i.e., <a href="#*idName*">...</a> for the same document, or <a href="*url#idName*">...</a> for the anchor point in another document identified via *url*.

For example,

```
<h1 id="ch1">Chapter 1</h1>
 ......
 ......
<h1 id="ch2">Chapter 2</h1>
 ......
 ......
Jump to <a href="#ch1">Chapter 1</a>
 ......
Jump to <a href="#ch2">Chapter 2</a>
```

Using Image as Hyperlink

To use an image as a hyperlink, put the image tag <img> between <a href="*url*"> and </a>. For example:

```
<a href="http://abc.com/">
  <img src="logo.gif" alt="logo" width="30" height="20">
</a>
<p>click the above image to visit us</p>
```

Using Image as Hyperlink

To use an image as a hyperlink, put the image tag <img> between <a href="*url*"> and </a>. For example:

```
<a href="http://abc.com/">
  <img src="logo.gif" alt="logo" width="30" height="20">
</a>
<p>click the above image to visit us</p>
```

Task: Implement HTML program for nested frame structure in TLB fashion consisting of targeting between frames and audio, video features and image mapping.

Illustration:

The tags involved are:

- <frameset>...</frameset>: to sub-divide the window.
- <frame>: defines each of the frames in a frameset.
- <iframe>...</iframe>: for floating or inline or internal frame.

- <noframe>...</noframe>: alternative text if frame is not supported by browser.

Frame Set <frameset>...</frameset>

Function: To divide the window into multiple frames, row-wise or column-wise. <frameset>...<frameset> can be nested for complex layout. <frameset>...<frameset> is to be used in place of <body>...<body> tag. i.e., there should not have an <body> tag in the document with <frameset>.

Syntax:

```
<frameset rows="list-of-row-sizes">
  ...frame-declarations...
</frameset>

<frameset cols="list-of-column-sizes">
  ...frame-declarations...
</frameset>
```

Attributes:

- rows|cols="*n*|*n*%|*, *n*|*n*%|*, ...": a list of frame sizes in pixels or percentage separated by commas. Wildcard "*" can be used to indicate the remaining space.

Example:

```
<frameset rows="100, 20%, 150, *">
  ...frame-declarations...
</frameset>
```

Divide the window into four rows of sizes: 100 pixels, 20% of the screen, 150 pixels, and the remaining space.

Individual Frame <frame />

Function: Declare each individual frame, and place the HTML document.

Syntax:

```
<frame
  src="url"
  name="frame-name"
  noresize
  frameborder="0|1"
  scrolling="yes|no|auto" />
```

Attributes:

- src:="*url*": provides the URL of the document to be displayed inside this frame.
- name: specifies an unique identifier, to be used as the target of other tags, such as <a>, <form>.
- noresize: suppresses resizing by dragging of border.
- frameborder: sets to 1 to show the border, 0 to suppress.
- scrolling: "yes" to show scrollbars and allow scrolling; "no" to suppress scrolling; and "auto" to let the browser to decide (e.g., auto-hide).

Example: Divide the screen into 3 frames, top, left and right.

```
<frameset rows="50, *">
  <frame src="navigation.html" name="navigation" noresize>
  <frameset cols="100, *">
   <frame src="summary.html" name="summary">
   <frame src="detail.html" name="detail">
  </frameset>
</frameset>
```

Targeting Named Frame

To control which frame to receive the content of a hyperlink, you could include the target attribute in the <a> tag. Using the previous example, a hyperlink in the "navigation" frame targeting the "detail" frame is:

```
<a href="chapter5.html" target="detail">Chapter 5</a>
```

If all the links in the "navigation" frame are targeting the "detail" frame, you can use the <base> tag (in the HEAD section) to set up a global target reference:

```
<!-- navigation.html -->
<head>
  <title>navigation bar</title>
  <base target="detail">
</head>
```

As mentioned, frames has gone out of favor these day. The HTML5 introduced new elements such as <header>, <footer>, <section>, <nav>, <article> to help you in organizing your web page, instead of using frame, division, or table.

No Frame <noframe>...</noframe>

Function: Provides an alternative text if the browser does not support framed layout. <noframe> must be placed within a <frameset>.

Syntax (Container tag):

```
<noframe>...alternative-text...</noframe>
```

Iframe (Inline Frame or Internal Frame) <iframe>...</iframe>

Function: Place an inline frame on the browser window. An iframe is a frame that can be embedded within a regular HTML page, which contains a separate and complete HTML document, set via its src attribute.

Syntax (Container tag):

```
<iframe
    src="URL"
    name="frame-name"
    frameborder="0|1"
    scrolling="yes|no|auto">
      ...alternative-text...
</iframe>
```

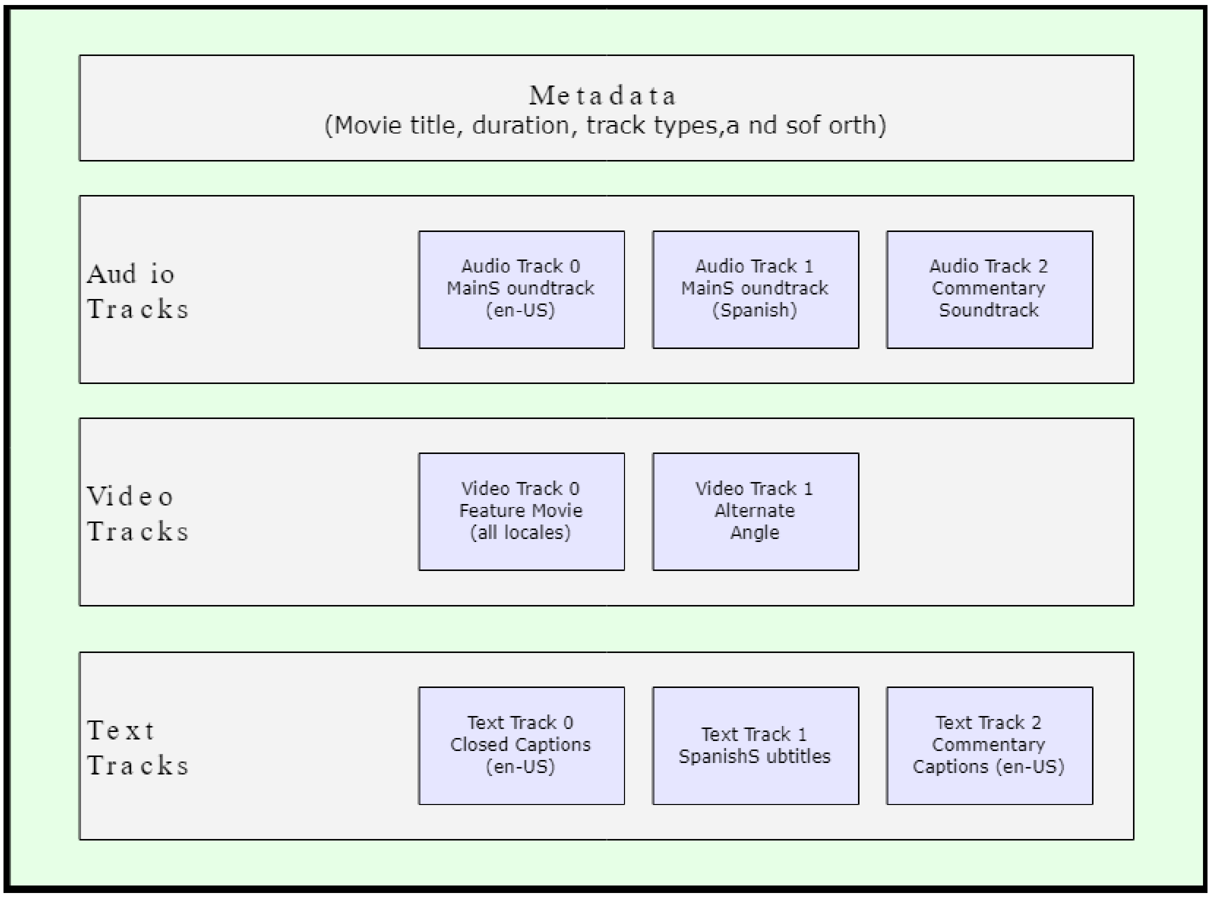
You could use CSS to position and style the iframe. Iframe is used extensively by JavaScript and Ajax.

B. Video/Audio embedding

Contents of a media file

First, let's go through the terminology quickly. Formats like MP3, MP4 and WebM are called **container formats**. They define a structure in which the audio and/or video tracks that make up the media are stored, along with metadata describing the media, what codecs are used to encode its channels, and so forth.

A WebM file containing a movie which has a main video track and one alternate angle track, plus audio for both English and Spanish, in addition to audio for an English commentary track can be conceptualized as shown in the diagram below. Also included are text tracks containing closed captions for the feature film, Spanish subtitles for the film, and English captions for the commentary.


Metadata
(Movie title, duration, track types,a nd sof orth)
Aud io Tracks
Audio Track 0 MainS oundtrack (en-US)
Audio Track 1 MainS oundtrack (Spanish)
Audio Track 2 Commentary Soundtrack
Video Tracks
Video Track 0 Feature Movie (all locales)
Video Track 1 Alternate Angle
Text Tracks
Text Track 0 Closed Captions (en-US)
Text Track 1 SpanishS ubtitles
Text Track 2 Commentary Captions (en-US)

The audio and video tracks within the container hold data in the appropriate format for the codec used to encode that media. Different formats are used for audio tracks versus video tracks. Each audio track is encoded using an audio codec, while video tracks are encoded using (as you probably have guessed) a video codec. As we talked about before, different browsers support different video and audio formats, and different container formats (like MP3, MP4, and WebM, which in turn can contain different types of video and audio).

For example:

- A WebM container typically packages Vorbis or Opus audio with VP8/VP9 video. This is supported in all modern browsers, though older versions may not work.
- An MP4 container often packages AAC or MP3 audio with H.264 video. This is also supported in all modern browsers, as well as Internet Explorer.
- The Ogg container tends to use Vorbis audio and Theora video. This is best supported in Firefox and Chrome, but has basically been superseded by the better quality WebM format.

There are some special cases. For example, for some types of audio, a codec's data is often stored without a container, or with a simplified container. One such instance is the FLAC codec, which is stored most commonly in FLAC files, which are just raw FLAC tracks.

Another such situation is the always-popular MP3 file. An "MP3 file" is actually an MPEG-1 Audio Layer III (MP3) audio track stored within an MPEG or MPEG-2 container. This is especially interesting since while most browsers don't support using MPEG media in the `<video>` and `<audio>` elements, they may still support MP3 due to its popularity.

An audio player will tend to play an audio track directly, e.g. an MP3 or Ogg file. These don't need containers.

Media file support in browsers

Why do we have this problem? It turns out that several popular formats, such as MP3 and MP4/H.264, are excellent but are encumbered by patents; that is, there are patents covering some or all of the technology that they're based upon. In the United States, patents covered MP3 until 2017, and H.264 is encumbered by patents through at least 2027.

Because of those patents, browsers that wish to implement support for those codecs must pay typically enormous license fees. In addition, some people prefer to avoid restricted software and prefer to use only open formats. Due to these legal and preferential reasons, web developers find themselves having to support multiple formats to capture their entire audience.

The codecs described in the previous section exist to compress video and audio into manageable files, since raw audio and video are both exceedingly large. Each web browser supports an assortment of **codecs**, like Vorbis or H.264, which are used to convert the compressed audio and video into binary data and back. Each codec offers its own advantages and drawbacks, and each container may also offer its own positive and negative features affecting your decisions about which to use.

Things become slightly more complicated because not only does each browser support a different set of container file formats, they also each support a different selection of codecs. In order to maximize the likelihood that your web site or app will work on a user's browser, you may need to provide each media file you use in multiple formats. If your site and the user's browser don't share a media format in common, your media won't play.

Due to the intricacies of ensuring your app's media is viewable across every combination of browsers, platforms, and devices you wish to reach, choosing the best combination of codecs and container can be a complicated task. See Choosing the right container in Media container formats (file types) for help selecting the container file format best suited for your needs; similarly, see Choosing a video codec in Web video codec guide and Choosing an audio codec in Web audio codec guide for help selecting the first media codecs to use for your content and your target audience.

One additional thing to keep in mind: mobile browsers may support additional formats not supported by their desktop equivalents, just like they may not support all the same formats the desktop version does. On top of that, both desktop and mobile browsers *may* be designed to offload handling of media playback (either for all media or only for specific types it can't handle internally). This means media support is partly dependent on what software the user has installed.

The <video> element

The <video> element allows you to embed a video very easily. A really simple example looks like this:

```
<video src="rabbit320.webm" controls>

  <p>Your browser doesn't support HTML5 video. Here is a <a href="rabbit320.webm">link to the video</a> instead.</p>

</video>
```

The features of note are:

src

In the same way as for the <img> element, the src (source) attribute contains a path to the video you want to embed. It works in exactly the same way.

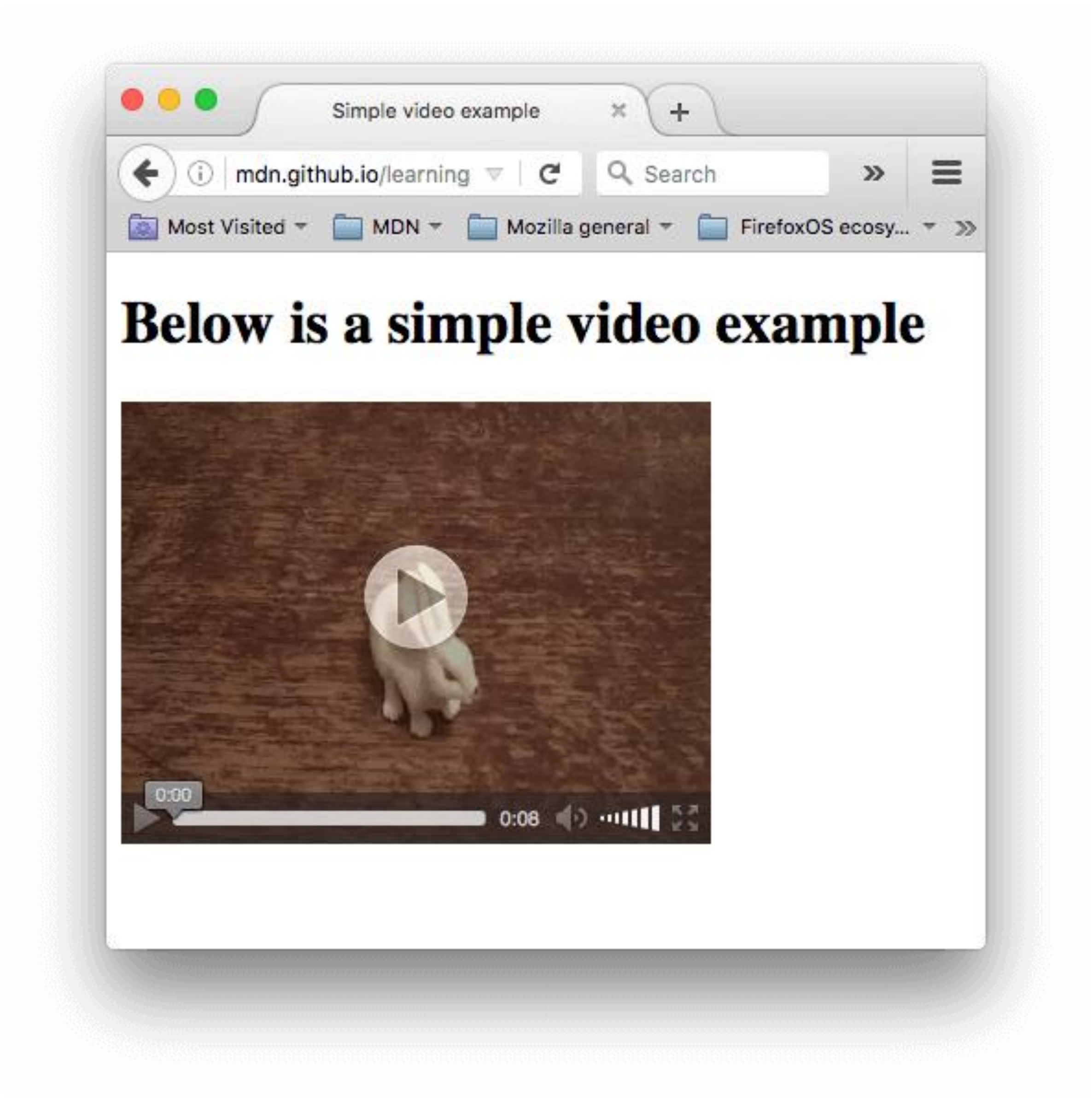
controls

Users must be able to control video and audio playback (it's especially critical for people who have epilepsy.) You must either use the controls attribute to include the browser's own control interface, or build your interface using the appropriate JavaScript API. At a minimum, the interface must include a way to start and stop the media, and to adjust the volume.

The paragraph inside the <video> tags

This is called **fallback content** — this will be displayed if the browser accessing the page doesn't support the <video> element, allowing us to provide a fallback for older browsers. This can be anything you like; in this case, we've provided a direct link to the video file, so the user can at least access it some way regardless of what browser they are using.

The embedded video will look something like this:



Below is a simple video example

```
<video controls>

  <source src="rabbit320.mp4" type="video/mp4">

  <source src="rabbit320.webm" type="video/webm">

  <p>Your browser doesn't support HTML5 video. Here is a <a href="rabbit320.mp4">link to
the video</a> instead.</p>

</video>
```

Here we've taken the src attribute out of the actual <video> tag, and instead included separate <source> elements that point to their own sources. In this case the browser will go through the <source> elements and play the first one that it has the codec to support. Including WebM and MP4 sources should be enough to play your video on most platforms and browsers these days.

Each <source> element also has a type attribute. This is optional, but it is advised that you include it. The type attribute contains the MIME type of the file specified by the <source>, and browsers can use the type to immediately skip videos they don't understand. Iftype isn't included, browsers will load and try to play each file until they find one that works, which obviously takes time and is an unnecessary use of resources.

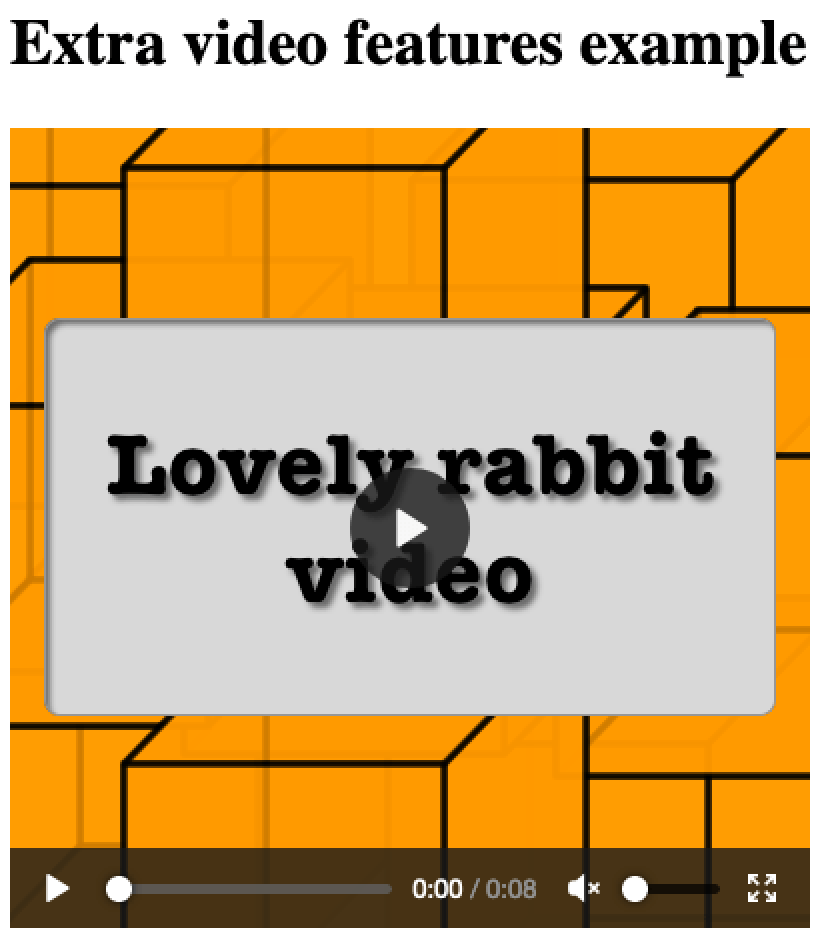
Refer to our guide to media types and formats for help selecting the best containers and codecs for your needs, as well as to look up the right MIME types to specify for each.

Other <video> features

There are a number of other features you can include when displaying an HTML video. Take a look at our next example:

```
<video controls width="400" height="400"

      autoplay loop muted preload="auto"

      poster="poster.png">

  <source src="rabbit320.mp4" type="video/mp4">

  <source src="rabbit320.webm" type="video/webm">

  <p>Your browser doesn't support HTML video. Here is a <a href="rabbit320.mp4">link to
the video</a> instead.</p>

</video>
```

The resulting UI looks something like this:



The new features are:

width and height

You can control the video size either with these attributes or with CSS. In both cases, videos maintain their native width-height ratio — known as the **aspect ratio**. If the aspect ratio is not maintained by the sizes you set, the video will grow to fill the space horizontally, and the unfilled space will just be given a solid background color by default.

autoplay

Makes the audio or video start playing right away, while the rest of the page is loading. You are advised not to use autoplaying video (or audio) on your sites, because users can find it really annoying.

loop

Makes the video (or audio) start playing again whenever it finishes. This can also be annoying, so only use if really necessary.

muted

Causes the media to play with the sound turned off by default.

poster

The URL of an image which will be displayed before the video is played. It is intended to be used for a splash screen or advertising screen.

preload

Used for buffering large files; it can take one of three values:

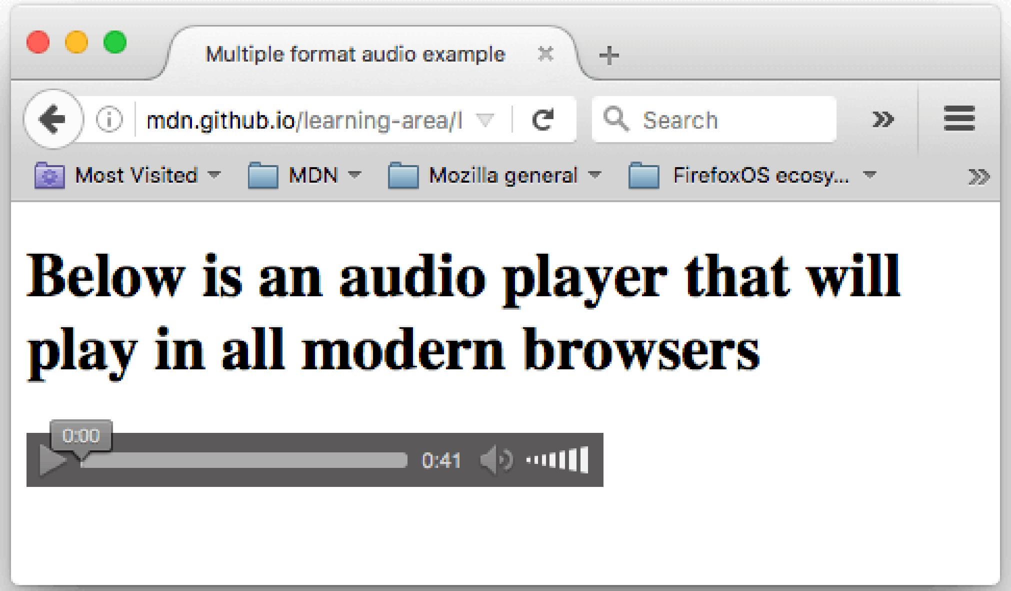
- "none" does not buffer the file
- "auto" buffers the media file
- "metadata" buffers only the metadata for the file

The <audio> element

The <audio> element works just like the <video> element, with a few small differences as outlined below. A typical example might look like so:

```
<audio controls>

  <source src="viper.mp3" type="audio/mp3">

  <source src="viper.ogg" type="audio/ogg">

  <p>Your browser doesn't support HTML5 audio. Here is a <a href="viper.mp3">link to the
audio</a> instead.</p>

</audio>
```

This produces something like the following in a browser:



Below is an audio player that will play in all modern browsers

This takes up less space than a video player, as there is no visual component — you just need to display controls to play the audio. Other differences from HTML video are as follows:

- The <audio> element doesn't support the width/height attributes — again, there is no visual component, so there is nothing to assign a width or height to.
- It also doesn't support the poster attribute — again, no visual component.

Other than this, <audio> supports all the same features as <video> — review the above sections for more information about them.

Displaying video text tracks

Now we'll discuss a slightly more advanced concept that is really useful to know about. Many people can't or don't want to hear the audio/video content they find on the Web, at least at certain times. For example:

- Many people have auditory impairments (such as being hard of hearing or deaf) so can't hear the audio clearly if at all.

- Others may not be able to hear the audio because they are in noisy environments (like a crowded bar when a sports game is being shown).
- Similarly, in environments where having the audio playing would be a distraction or disruption (such as in a library or when a partner is trying to sleep), having captions can be very useful.
- People who don't speak the language of the video might want a text transcript or even translation to help them understand the media content.

Wouldn't it be nice to be able to provide these people with a transcript of the words being spoken in the audio/video? Well, thanks to HTML video, you can. To do so we use the WebVTT file format and the <track> element.

Note: "Transcribe" means "to write down spoken words as text." The resulting text is a "transcript."

WebVTT is a format for writing text files containing multiple strings of text along with metadata such as the time in the video at which each text string should be displayed, and even limited styling/positioning information. These text strings are called **cues**, and there are several kinds of cues which are used for different purposes. The most common cues are:

subtitles

Translations of foreign material, for people who don't understand the words spoken in the audio.

captions

Synchronized transcriptions of dialog or descriptions of significant sounds, to let people who can't hear the audio understand what is going on.

timed descriptions

Text which should be spoken by the media player in order to describe important visuals to blind or otherwise visually impaired users.

A typical WebVTT file will look something like this:

```
WEBVTT
```

```
1

00:00:22.230 --> 00:00:24.606

This is the first subtitle.



2

00:00:30.739 --> 00:00:34.074

This is the second.



  ...
```

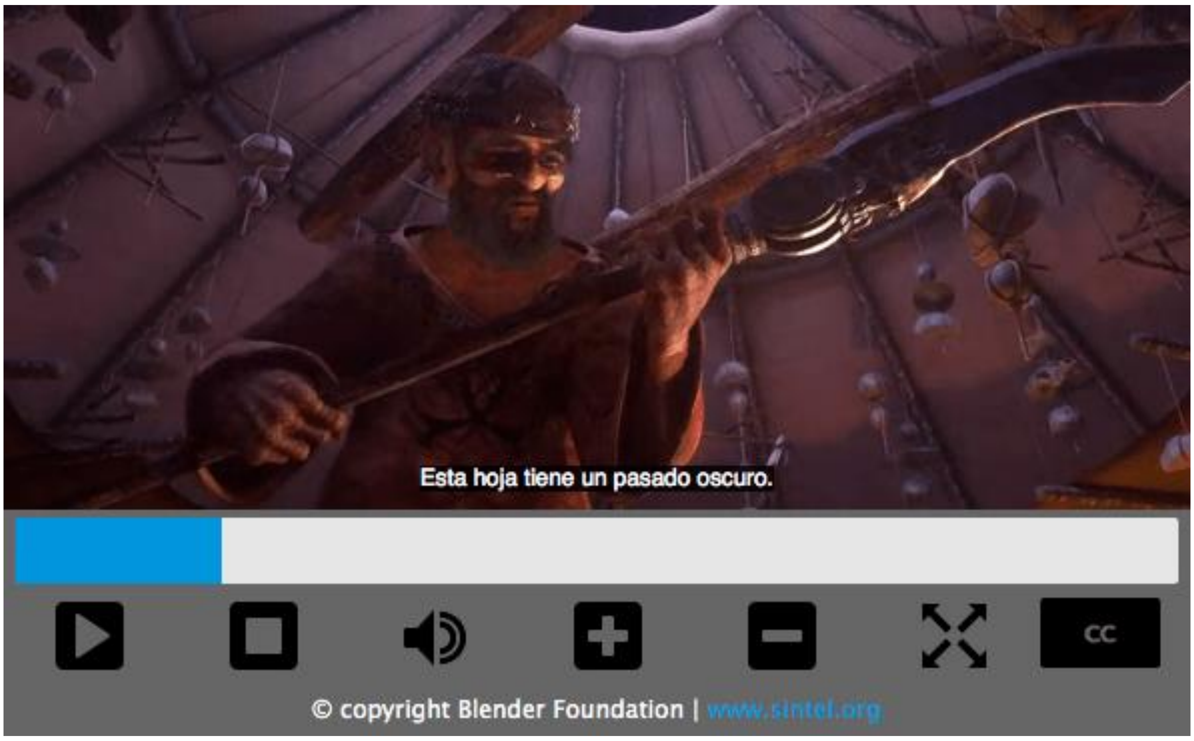
To get this displayed along with the HTML media playback, you need to:

1. Save it as a .vtt file in a sensible place.
2. Link to the .vtt file with the <track> element. <track> should be placed within <audio> or <video>, but after all <source> elements. Use the kind attribute to specify whether the cues are subtitles, captions, or descriptions. Further, use srclang to tell the browser what language you have written the subtitles in. Finally, add label to help readers identifying the language they are searching for.

Here's an example:

```
<video controls>

   <source src="example.mp4" type="video/mp4">

   <source src="example.webm" type="video/webm">

   <track kind="subtitles" src="subtitles_es.vtt" srclang="es" label="Spanish">

</video>
```

This will result in a video that has subtitles displayed, kind of like this:


Esta hoja tiene un pasado oscuro.
CC
© copyright Blender Foundation | www.sintel.org

Active learning: Embedding your own audio and video

For this active learning, we'd (ideally) like you to go out into the world and record some of your own video and audio — most phones these days allow you to record audio and video very easily and, provided you can transfer it on to your computer, you can use it. You may have to do some conversion to end up with a WebM and MP4 in the case of video, and an MP3 and Ogg in the case of audio, but there are enough programs out there to allow you to do this without too much trouble, such as Miro Video Converter and Audacity. We'd like you to have a go!

If you are unable to source any video or audio, then you can feel free to use our sample audio and video files to carry out this exercise. You can also use our sample code for reference.

I would like you to:

1. Save your audio and video files in a new directory on your computer.
2. Create a new HTML file in the same directory, called `index.html`.
3. Add `<audio>` and `<video>` elements to the page; make them display the default browser controls.
4. Give both of them `<source>` elements so that browsers will find the audio format they support best and load it. These should include `type` attributes.

5. Give the <video> element a poster that will be displayed before the video starts to be played. Have fun creating your own poster graphic.

For an added bonus, you could try researching text tracks, and work out how to add some captions to your video.

C. Image Mapping

Image maps are "clickable" image that loads different pages depending on where (or which hot-regions) you click on the image. Each hot region of the image map can be associated with a different link. There are two type of image maps:

1. Client-side image map: browser at the client-side handles the mapping of hot-regions to links.
2. Server-side image map: server handles the mapping of hot-regions to links. The co-ordinates of the location clicked are sent to the server to be processed by a program (such as a CGI/ASP/JSP/PHP script). The mapping information is held on the server.

Client-Side Image Map

To create a client-side image map:

1. Define a image map using <map>...</map> and define the hot-regions in the map using the <area> tags. Decide what link maps to which hot-region. Hot regions can take circle, rectangle, or polygon in shape.
2. In the <img> tag, add the attribute usemap="*#mapName*" to indicate this image is used as a client-side image map.

```
<map id="myMap">
  <area shape="circle" href="url1" coords="xc, yc, r">
  <area shape="rect"   href="url2" coords="topLeftX, topLeftY, bottomRightX, bottomRightY">
  <area shape="poly"   href="url3" coords="x1, y1, x2, y2, ..., x1, y1">
</map>
```

```
<img usemap="#myMap" src="client_map.gif">
```

Example:

```
<map id="myMap">
```

```
  <area shape="rect"   coords="20,20,160,60" href="http://www.zzz.com/">
  <area shape="circle" coords="80,120,60,60" href="http://www.yyy.com/">
</map>
<img usemap="#myMap" src="banner.jpg">
```

Client-side Image Map Tags - <map> and <area>

Function: To set up a client-side image map with hot regions.

Syntax (map: Container tag, area: Standalone tag)

```
<map id|name="map-name">
  <area shape="rect|circle|poly|default"
  coords="coordinates-list"
  href="URL" nohref
  target="_blank|_parent|_self|_top|target-frame-name"
  alt="alternative-text" />
  ... more-area-declarations ...
</map>
```

<map>'s attributes:

- id|name="*map-name*": declares a unique name for the map, to be targeted in attribute usemap="#*map-name*" of the <img>. (The attribute name is used the older browsers. XHTML specifies using id instead, which automatically create a named anchor.)

<area>'s attributes:

- shape="rect|circle|poly|default": define the shape of the hot region. The "default" value for any point not part of another hot region.
- coords: list of coordinates that made up the hot region.
  - For shape="rect", coords="*topLeftX*, *topLeftY*, *bottomRightX*, *bottomRightY*" to specify the upper-left and lower-right corners.
  - For shape="circle", coords="*xc*, *yc*, *r*", where (*xc*, *yc*) is the center and *r* is the radius.
  - For shape="poly", coords="*x*1, *y*1, *x*2, *y*2, ..., *x*1, *y*1", where (*xi*, *yi*) are coordinates that made up the polygon. You should close the polygon by putting (*x*1, *y*1) as the last coordinates.
- href="*url*": gives the target URL of the hyperlink.
- nohref: deactivate the hot region, pointing to nowhere.

- If two hot regions overlap, the first takes effect.

A client-side image map can be used as a navigation bar on top of the page, instead of using individual images. This may save some transmission overhead, as each individual image triggers its own HTTP request.

Server-Side Image Map

On the <img> tag, include the "ismap" (is server map?) attribute, and wrap the <img> within an anchor <a>...</a>. For example,

```
<a href="http://www.zzz.com/search.jsp"><img ismap src="logo.jpg"></a>
Click on the image and observe the URL
```

When the image is clicked, the (x, y) position of the click is send to the server as query parameters. For example,

```
http://www.zzz.com/search.jsp?39,22
```

It is up to the server to decide on how to process the (x, y) position received via a server-side program (such as CGI/ASP/JSP/PHP/Servlet).

Client-Side vs. Server-Side Image Map

Client-side image map is much more popular (and recommended) than server-side image map because:

1. Mapping for client-side image map is processed by the browser locally with immediate response, without connecting to the server and waiting for server to response.
2. No special server-side programs needed, relies solely on the browser to process the mapping of the hot regions.