LESSON D12_EN. HOW TO USE E-LEARNING TOOLS AND TECHNOLOGIES. HTML+TIME and SMIL.

Parent institution: UE-B, Bucharest, Romania,
22 Franceza St., Postal Code 030104

Authors: Cristian-Teodor Păun, Member of Teaching Staff,
Fax: + 4021 315 77 30; e-mail: cristianpaun@ueb.ro

Accessible for consultations daily from 9.00 to 10.00.

After the learning this lesson you will be more reach with the following knowledge:
- How to create timing documents in WEB
- What are tag that may be used in (X)HTML
- Using the new specification of SMIL
- What are the similarities and differences between HTML+TIME and SMIL

<u>LEARNING OBJECTIVES:</u>
After the learning this lesson you will accomplish the ability to:
- Transform the old HTML documents in new interactive (X)HTML documents
- To use the new specification of SMIL and HTML+TIME languages
- To create new WEB pages with slide-show presentations
- To use the advantage HTML+TIME and SMIL

Part I: Introduction to HTML+TIME and SMIL

<u>CONTENT OF LESSON Part I - Introduction to HTML+TIME and SMIL:</u>

HTML+TIME (**T**imed **I**nteractive **M**ultimedia **E**xtensions) is a very powerful tool that is implemented for the first time Microsoft Internet Explorer 5. You can add timing and media synchronization support to HTML pages sing HTML+TIME. You can add images, video, and sounds to an HTML page, and also, synchronize them with HTML text elements over a specified amount of time using a few Extensible Markup Language (XML)-based elements and attributes. For more information, please go to http://www.w3.org/TR/REC-xml-names/#ns-decl
For the beginning, you can create a HTML page like a slide-show, you know from MS PowerPoint presentation, but these pages will be "slide-show-Web style" presentations, witch may contain synchronized text, images, audio, video, and streaming media.

You must remember that these presentations will be timed-style or interactive-style, or your own combination of both styles.

SMIL (**S**ynchronized **M**ultimedia **I**ntegration **L**anguage) is pronounced "smile" and is a language for describing audiovisual presentations. Like HTML+TIME is simple to learn and to understanding. SMIL is written in XML and like HTML language, very simple. You shall write your own programs using a simple text-editor (like Notepad in Windows environments or like "vi" in Linux environments)

You must know that in IT field any year could be history for everybody and for any software product. For example, we bring for you from W3C Web site (the official site of World-Wide Web Consortium where are published the recommendations for WWW activities, including WWW markup languages), a little information reproduced here: "***SMIL 2.0** is the W3C successor to SMIL 1.0. HTML+TIME 2.0 are the successor to HTML+TIME 1.0.* **Note** *HTML+TIME 2.0 is available in minimal installations of Internet Explorer 5.5 and Internet Explorer 6.*"

In this lesson we suppose that you are study in Windows environments.

1. Let see that are “Ease of Use”!

If you are interested in this and you already have the knowledge about HTML and XML, and the tags from markup language are familiars, let start to inspect this field.

So, in short, to add timing to your HTML document, all you have to do is to add some new attributes to existing HTML elements.

The HTML+TIME attributes specify when an element appears on a page, how long it remains displayed. To make your job easier, some new XML-based elements have been created to simplify incorporating media into your Web pages.

For example, you can compare the following two examples.

```
<!-- document sample001_begin.html -->
<html>
<head>
<title> My simple HTML page </title>
<body>
        <h5>    You must following this step to add timing to your HTML pages:
        </h5>
        <pre>
                1. create a XML Namespace
                2. make a reference to the time2 Behavior
                3. specify Beginning and Ending Times
                4. include one or more Action
        </pre>


</body>
</html>
```

Fig.1 Sample how to start

The first example is a simple HTML page that gave to other persons the information how to change that simple HTML page in some attractive HTML page. The second example is the same thing, but we use some new attributes in the HTML elements. We may copy this example and try to display on your screen with Internet Explorer v5.5 or a greater version.

```
<!-- document sample002_begin.html -->
<html xmlns:t="urn:schemas-microsoft-com:time">
<head>
<title> My first HTML page with TIME </title>
<style type="text/css">
.time
    {behavior: url (#default#time2) ;
    }
</style>
<?import namespace="t" implementation="#default#time2">
</head>
<body>
        <h1> Attribute "begin" simple example</h1>
        <h3 id="blinking" class="time" begin="2 ; blinking.end+2" dur="1">
        This is my first HTML+TIME page
        </h3>
        <p class="time" begin="2” text="bold">
                You must following this step to add timing to your HTML pages:
        </p>
        <p class="time" begin="3s" dur="3s" repeatCount="10">
                1. create a XML Namespace
        </p>
        <p class="time" begin="6s" dur="3s" repeatCount="10">
                2. make a reference to the time2 Behavior
        </p>
        <p class="time" begin="9s" dur="3s" repeatCount="10">
                3. specify Beging and Ending Times
        </p>
        <p class="time" begin="12s" dur="3s" repeatCount="10">
                4. include one or more Action
        </p>
</body>
</html>
```

Fig.2 Your first timing page

2. Authoring HTML+TIME

Our first example said to use the following steps to add timing to an HTML element:

- Create an XML Namespace
- Reference the time2 Behavior
- Specify Beginning and Ending Times
- Include an Action

So, let see what means to follow these steps.

3. Create an XML Namespace

When you are using any of the HTML+TIME elements, witch may be the **t:EXCL**, **t:SEQ** or **t:PAR** elements, you must declare the XML namespace t: in the HTML tag, as shown in the following line of code:

```
<HTML XMLNS:t ="urn:schemas-microsoft-com:time">
```

You can observe that we made this in the second line of our second example. Also, to use the namespace you must preface HTML+TIME elements with **"t:"** (this string identifies the HTML+TIME elements as qualified XML namespace extensions)

To establish **t:** as the namespace, you must import the **time2** behavior into the namespace as shown in the following line of code, witch you can find at the 10th line from the same example:

```
<?IMPORT namespace="t" implementation="#default#time2">
```

4. Reference the time2 Behavior

You must associate the element with the **time2** behavior, so that the element is affected by the document timeline. To accomplish this, you can add the inline STYLE attribute to the HTML element as follows:

```
<P STYLE="behavior:url(#default#time2)" ...>
```

We create a STYLE for entire file "sample002_begin.html" in the description part of html file:

```
<style type="text/css">
.time {behavior: url (#default#time2)  }
</style>
```

In our examples we show you a few element that can make your page timing like when the line begin to appaire on the screen, how long time it will stay there, we made a line to blink on the screen in header 3 tag, and so on. In the following section, we will take care to teach you slowly, step by step, the "secrets" of the WEB art design.

All this descriptors are necessary to include in your HTML pages the timing attributes and the browser to serve like on your display. If you see a static page is necessary to allow the ActiveX code to be executing for this time on your computer. Is possible to be not allowed for faster loading and display the page.

Now you must compare the following example. In the left side you have a HTML+TIME page, in the central side is the same page write in SMIL and in the right side is a counter lines to compare more simple the two "programs". In the firs moment you shall say "Oh, but SMIL is simpler!" The answer is "NO". These programs are the same. Of course, in HTML page we must write more characters but a lot of them you may store in html-pattern file and you use this file to write a new page.

<html xmlns:t="urn:schemas-microsoft-com:time">	<smil>	1
<head>		2
<?import namespace="t" implementation="#default#time2">		3
<style>.t {behavior: url(#default#time2)}</style>		4
</head>		5
<body>	<body>	6
<t:seq repeatCount="indefinite">	<seq repeatCount="indefinite">	7
<img class="t" src="rose1.jpg" dur="3s" />	<img src="Rose1.jpg" dur="3s" />	8
<img class="t" src="rose2.jpg" dur="3s" />	<img src="Rose2.jpg" dur="3s" />	9
<img class="t" src="small rose.jpg" dur="3s" />	<img src="Small rose.jpg" dur="3s" />	10
</t:seq>	</seq>	11
</body>	</body>	12
</html>	</smil>	13

Fig.3 Compare between HTML+TIME and SMIL

5. HOMEWORK

Pick up three picture on your working folder, copy these programs in Notepad, save on your working hard disk with different name. Don't forget to save first program with "html" extension and the second with "smil" extension. Finally open the html file with Internet Explorer v5 or greater and the smil file with RealPlayer.

6. SOME THEORY

If you don't like it jump to "**The Simplest case: the beginning moment**" in the next section. You shall read this later to increase you own skills

7. Timing and Interaction Support

We show you in the simplest way that the TIME is powerful tool that don't force your minds to memories a lot of new rules or abbreviations or names or anything else. Timing support is based upon a simple model of adding attributes to HTML elements.

HTML elements can be set to have a begin time and a duration or, more, elements can be made to be repeat.

You can "decorating" the HTML tags with additional natural attributes and is very important that is need not learn an entirely new syntax or document structure to add timing to pages.
Is important to know the rules for applying the timing and these are fairly simple:

- For all style tags the time behavior will remove the style before and after the defined begin and end times.
- For all displayed elements, the time behavior will control either the `visibility` attribute or the `display` attribute of the local style.
- By default, the `visibility` attribute will be used, and the document will not reflow in response to timing. An additional attribute allows the author to specify that the `display` attribute be used instead. This will cause a document reflow in response to timing events.

HTML Tag Attributes

For all HTML elements that support timing, the following simple timing attributes are supported:

begin
This defines the offset (i.e. the delay) in seconds at which the element should be displayed. By default this is an offset from the time the page is displayed. Legal values are signed clock values.

beginWith
The referenced element will start at the time base defines by the argument, and is relative to other elements and time sequence (plus or minus any begin value). No more than one of beginWith, beginAfter or beginEvent should be specified.

beginAfter
The referenced element will start at the time base defines by the argument, and is relative to other previous elements and may never be active or displayed. Legal values are element IDs within the current time scope. No more than one of beginWith, beginAfter or beginEvent should be specified.

beginEvent
Defines the time base to be relative and synchronizes to the referenced event.

dur
This defines the duration in seconds for which the element should remain active or displayed. Legal values are clock values greater than 0.

end
This defines the end time for the element. Legal values are signed clock values.

endWith
This supports end-point sync between elements. The current element will end at the same time as the referenced element (plus or minus any end value). Legal values are element IDs within the current time scope.

endEvent
Defines the end-time to be relative to the referenced event.

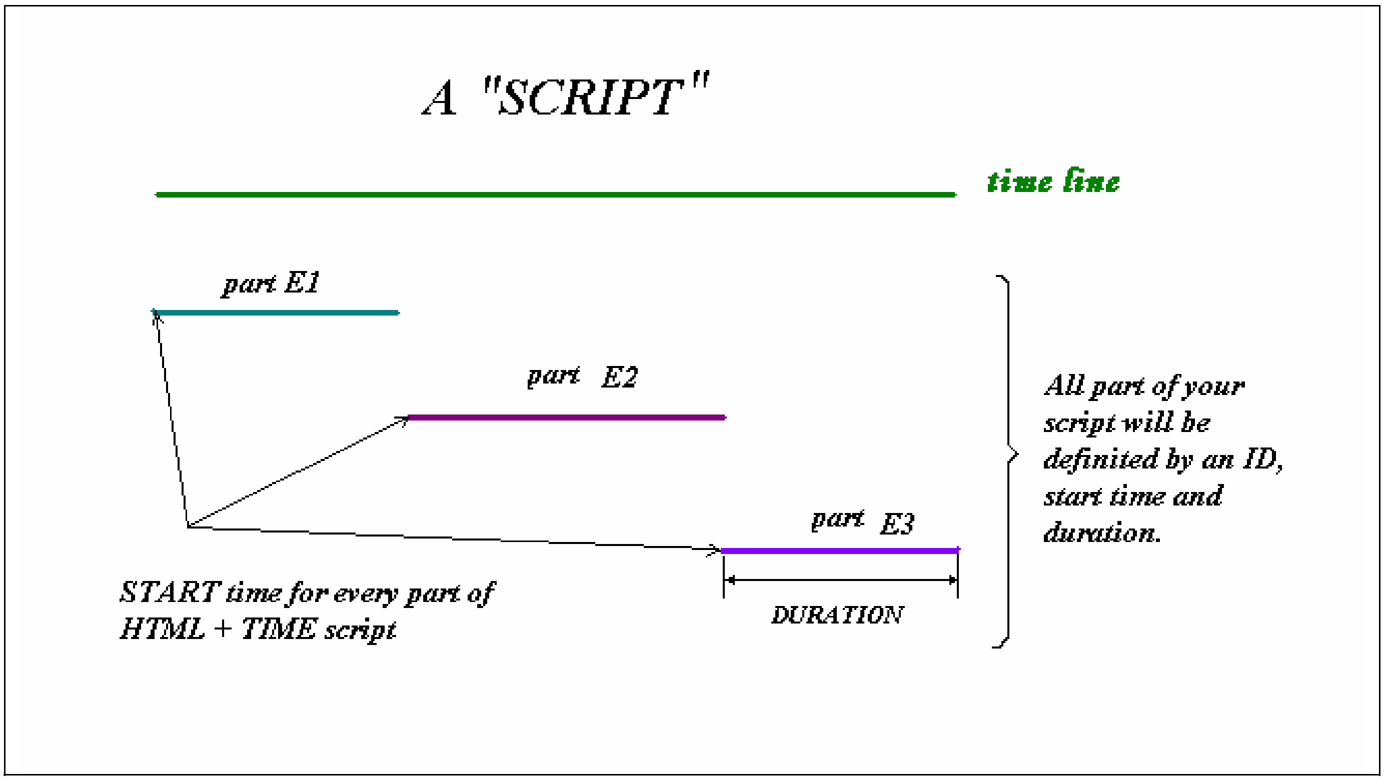
repeat
This causes the element to play repeatedly (loop) for the specified number of times. Each repeat iteration lasts for the duration defined by the dur or end attributes. This attribute has no affect if the duration is not defined or is indefinite. Legal values are integer or fractional iterations, greater than 0.

repeatDur
This causes the element to play repeatedly (loop) for the specified duration (in seconds). Legal values are clock values greater than 0.

skip-content
This attribute is introduced for future extensibility of HTML+TIME (and SMIL).

timeAction
This defines the attribute to be controlled over time as the element begins and ends on the local timeline. This attribute only applies to element tags.

This attributes are sufficient to implement in a HTML script (or SMIL) how to be “played” the acts.


A "SCRIPT"
time line
part E1
part E2
part E3
All part of your script will be definited by an ID, start time and duration.
START time for every part of HTML + TIME script
DURATION

Legal values are:

display
This causes the style.display property to be used to make elements appear and disappear on the timeline.

visibility
This causes the style.visibility property to be used to make elements appear and disappear on the timeline and this is the default.

style
This causes the inline style attribute to be removed and applied to the element over time. This allows for easier control over complex time-based styles.

onOff
This causes an element "on" property to be toggled between true and false over time. If there is no "on" property defined for the element, this does nothing. The "on" property enables and disables the intrinsic behavior of the element.

Clock value syntax

The grammar for clock values may be written in the following syntax:

Signed-Clock-value	("+" | "-")? Clock-value ; default is "+"
Clock-value	HMS-value | Timecount-value
HMS-value	(Hours ":")? Minutes ":" Seconds ("." Fraction)?
Timecount-value	Timecount ("." Fraction)? ("h" |"min" |"s" |"ms")? ; default is "s"
Hours	DIGIT+
Minutes	2DIGIT ; range from 00 to 59
Seconds	2DIGIT ; range from 00 to 59
Fraction	DIGIT+
Timecount	DIGIT+
2DIGIT	DIGIT DIGIT
DIGIT	[0-9]

Either other words, the minutes and seconds fields have value in the range 00 to 59 but you must specify the leading zeros between ":" and "." delimiters. The hours can have any integer value and you don’t must specify any leading zeros. You can specify fractional seconds in both HMS-values and Seconds-values with arbitrary precision, but nothing greater than millisecond accuracy is guaranteed.

Let see the following examples for legal clock values:

Clock values:	
02:30:03	this means 2 hours, 30 minutes and 3 seconds on the clock
2:33	this means 2 minutes and 33 seconds (!)
Timecount values:	
3h	this means 3 hours period
45min	this means 45 minutes period
30s	this means 30 seconds period
5ms	this means 5 milliseconds period
Signed clock values:	
+2:10	plus 2 minutes and 10 seconds
+300ms	Plus 300 milliseconds
-10.3	minus 10.3 seconds (10,300 milliseconds)

8. The Simplest case: the beginning moment

You already know that you can add attributes to any object in HTML to add timing. By default, all timed elements are relative to a document “root context”, and so exist in a single time scope for the page. You will see that with advanced timing support you can use more powerful constructs if is necessary and when you need them. In this moment, you begin to use “TIME” in yours HTML documents, you do not have to understand anything about a timing structure or hierarchy to do simple things. Also, all the offsets are in the same “timespace” (in the simple, default case), making it easy to align elements in time that are laid out all over the page.

In the “sample003_begin.html” you will make appear some paragraphs over time, tick-tack like a clock…

```
<!-- document sample003_begin.html -->
<html xmlns:t="urn:schemas-microsoft-com:time">
<head>
<title> My first HTML pages with TIME (3) </title>
<style type="text/css">
.time
     {behavior: url (#default#time2) ;
      }
</style>
<?import namespace="t" implementation="#default#time2">
</head>
<body>
  <t:seq>
         <p id="p1" class="time" begin="1s" dur="1s">
                  <!-- This is a paragraph of text that appears after 1 second -->
                  One …
         </p>
         <p id="p2" class="time" beginAfter="p1" dur="1s">
                  <!-- This is a paragraph of text that appears after 2 seconds -->
                  Two …
         </p>
         <p class="time" beginAfter="p2">
                  <!-- This is a paragraph of text that appears after 3 seconds -->
                  Three…
         </p>
   </t:seq>
</body>
</html>
```

The page count for you from one to three. But you can do more.

9. Using durations and relative timing

If you want yours HTML pages to support greater flexibility in terms of relative timing, you must use new timing attributes for the duration and time base parameters. From this moment you must to thing a time diagram like a scene graph. You must establish a hierarchy of entities relative timing. The easiest way to start is to see our example “sample003_begin.html” where you will see how to define relative time among element. For more information you must turn back to the “theory’ and read carefully and try to test the knowledge in the same example changing the HTML attributes.

10. How to repeat your content

You can make any given element or timeline play repeatedly using the “repeat” attributes. Also, you can specify either a number of times to play the simple duration, or a total time for which to repeat the element timeline.

```
<!-- document sample004_sequence.html -->
<html xmlns:t="urn:schemas-microsoft-com:time">
<head>
  <?import namespace="t"
        implementation="#default#time2">
  <style>.t
        {behavior: url(#default#time2)}
        </style>
</head>
<body>
        <t:seq repeatCount="indefinite">
                <img class="t" src="rose1.jpg" dur="1s" width=200 height=150 />
                 <img class="t" src="rose2.jpg" dur="1s" width=200 height=150 />
                <img class="t" src="small rose.jpg" dur="1s" width=200 height=150/>
        </t:seq>
</body>
</html>
```

In this example we set the image presentation to repeat indefinitely. You shall see the “rose2” after “rose1”, the “small rose” after “rose2”, and so on. In every second the browser will display the next rose from the sequence.

11. CONCLUSION (part I)

In this part of the lesson we introduce you in timing attributes. In this moment you know to timing your HTML pages using the new timing attributes in the old tags. So, you transform the old pages in life WEB presentations where you can repeat different entities like text or image. Also, you can establish the moment to begin an act, how long period the entity may be display by the browser.

Part II:

CONTENT OF LESSON - PART II:

Introduction

Now, you know how to make your HTML pages more attractive and you understood that you may create a slide show to better explain the ideas of contents from HTML pages. Is time to learn more about the actions that can be done with your HTML page? We shall learn together step by step new attributes of HTML+TIME and SMIL. In the first part, you see that HTML+TIME and SMIL are very close. You learn that HTML pages are simple display by Internet Explorer v5.5 or greater and for SMIL, you already use a special player like RealPlayer. In this part you shall learn about the time synchronization of different part of your presentations. You shall know about sequence executions of blocks and the parallel executions of block. We use the "block" term to describe a part of a HTML page where you make a distinct action with one or more HTML line.

1. Using the timeAction attribute

With timeAction attribute, you can make an element to be *active* and not on a timeline. Let see a little example to see what means to be "active" element, or not. For the purposes of control in HTML+TIME, all HTML elements are grouped into categories. By default, all elements categorized as *content* will be controlled with the visibility property, like in any programming language. While the element is active on the timeline (from the defined begin until the defined end), the visibility property will be set to "visible". Again, after the end time, the visibility property will be set to "hidden". All *style* elements will be controlled by removing the effect of the element intrinsic behavior.

```
<!-- document sample005_playing.html -->
<html xmlns:t="urn:schemas-microsoft-com:time">
 <head>
 <title> My first HTML pages with TIME </title>
 <style type="text/css">.t {behavior: url (#default#time2)       }
 </style>
 <?import namespace="t" implementation="#default#time2">
</head>
<body>
 <table border="0" cellpadding="1" cellspacing="5" width="640">
  <thead>
   <tr>
    <th width="160"> First column text </th>
    <th width="160"> Second column text </th>
    <th width="160"> Thrid column text </th>
    <th width="160"> Fourth column text </th>
   </tr>
  </thead>
  <tbody>
   <tr>
    <td width="160">
      <span class="t" begin="10s" dur="20s">
       This is some text that appears after ten seconds and remains
        visible for 20 seconds.
       <br>
       <img class="t" width="120" height="90" src="Buket.jpg" />
      </span>
    </td>
    <td width="160">
     <b class="t" begin="2s" dur="8s">
```

```
      This is some text will appear normally at first.
         But it is in a bold text tag
         then when the timeline is active is displayed bold for 8 seconds
         and then revert to normal display again
         when the timeline is inactive.
   </b>
  </td>
  <td width="160">
    <span class="t" style="text-decoration:line-through; color:blue"
     begin="10s" dur="20s" timeAction="style">
     This is some text will appear normally at first, then
     be displayed in red strikethrough for 10 seconds,
     and then revert to normal display again.
    </span>
  </td>
  <td width="160">
    <span class="t" begin="10s" dur="20s" timeAction="display">
     This is some text that appears after ten seconds, and remains
     visible for 20 seconds. When it becomes visible and again when
     it is hidden, the document will reflow.
    </span>
  </td>
 </tr>
 </tbody>
</table>
</body>
</html>
```

We gave to you the above example to show the default behavior for “timeActions” attribute. We include in a table with four columns four different type of formatting text. You must observe that span tag will be hidden when it is inactive and the bold element will be visible, but not bold, when it is inactive. In third column the information is the same for all timing period. But the author may be interested to display supplementary information in the fourth column and we timed both columns. We cut the text from the third column with a lime through the text and display for 20 seconds the supplementary information. All of these may used by a scenario for your further HTML pages. Please copy them and test the time behavior in Internet Explore.

2. Advanced Timeline support

From the above example you learn the following:

- the display value is useful when the author wants the document to reflow over time. You can use it for image sequences, where only the active image should affect the layout of the document;
- the style value helps you to make easier the control of complex set of styles over time. Any style control that can be defined using the inline style attribute can be animated over time using this timeAction setting.

Until now, you use simple time attributes that provide a very easy to use mechanism to add simple timing to your HTML pages. The greatest advantage is a very simple syntax.

Starting now, any time you can introduce animation and timing in your pages. There will be cases in which you want to build up more complex timing structures, and to easily manipulate them.
In HTML+TIME you can use a useful “par” attribute to structure timed elements. Using this attribute, you can manipulate different blocks independently, each block will have its internal synchronization and it will be a parent timing for integral document. This name comes from SMIL (Synchronized Multimedia Integrated Language), and is short for "parallel".
Also, in this part of lesson we shall learn more about SMIL and we shall compare “HTML+TIME” and SMIL pages. There may be a potential conflict with the notion of "paragraphs" by introduction of <par> tag that can be set up “true” or “false” for a block of paragraphs. The par attribute defines a relative timeline which can be manipulated as a unit, moved in time, looped, cut and pasted, etc. and let see how we can do this.

3. Timeline Attribute Syntax “*par*”

With this new tag you can define a new timing entity. We shall use this create a new relative timeline context with the scope of “the HTML container”. For the purposes of the timing model, this makes the container element equivalent to a timeline element. Timed elements within the container scope will be relative to this new timing context. It may be an exception when we use elements defined with “beginEvent”. The timeline container itself is relative to the document root time scope or to an

enclosing timeline container. If a begin delay and/or duration is defined for the sequence container, the entire container will be affected.
<t:par>

It defines a new relative timeline context. All timed elements within the timeline scope will be relative (perhaps indirectly in the case of beginWith/beginAfter usage) to this new timing context. The timeline element itself is relative to the document root time scope (or to an enclosing timeline element).
You may skip the following paragraphs that contain explanations about the attributes of “par” element and go directly to **EXAMPLE**.

4. Timeline Element Attributes

We present to you “some theory” above. In this moment if you already understood the example above we ensure you that you are on the right way. You can define the timeline containers in the same manner as any other element. You can use an offset value and/or a time base for the entire local context, and shift the time of everything within the timeline scope. For more explanation you shall access the information of the site http:\www…..
Here is a list of the timeline “par” element attributes to keep closely in our mind:

Begin
beginWith
beginAfter
beginEvent
	Dur	end
		endWith
		endSync

All this attributes was defined above, but is important to remember that the referenced element must not be within the scope of the timeline. If such a circular reference is detected at runtime, the attribute will be ignored, and the default time base will be used instead.

This is particularly useful when the contained elements include media that has finite but unknown duration (e.g. mpeg movies). The end will be computed depending on the value of the attribute. Legal values are:

- **first** - the container should end with the first (earliest) end of a contained, timed element. With other words, it finished as soon as any timed element contained ends, the container also ends;
- **last** - the container should end with the last (latest) end of a contained, timed element. With other words, it finished after all timed elements contained ends, the container also ends.
- **[id-ref]** - The container should end with the end of the referenced element. Is important the referenced element must be contained within the current timeline container.

Where, all the description refers to "contained" elements. It should be understood that this refers to time containment, and not simply HTML containment. A div element that is a timeline may contain other div tags that are not timelines; all HTML descendents of a timeline container are considered to be time-children (i.e. contained by the timeline div), if there is no intervening timeline container in the hierarchy. The timing hierarchy need not be tied directly to the HTML hierarchy. The default value of "endSync" is "last".

5. PARALLEL TIMING EXAMPLE

```
<!-- document sample006_playing.html -->
<html xmlns:t="urn:schemas-microsoft-com:time">
<head>
<title> My first HTML pages with TIME </title>
<style type="text/css">
.t
     {behavior: url (#default#time2) ;
     }
</style>
<?import namespace="t" implementation="#default#time2">
</head>
<body>
<t:par>
<span class="t" id="TL1" dur="10s">
   <!-- This begins right away, and lasts for 10 seconds -->
   <p>
      This is some text that appears immediately
   </p>
   <p class="t" begin="2s" dur="2s">
      This is some text that appears after two seconds
```

```
    </p>
    <p class="t" begin="3s">
        This is some text that appears after three seconds
    </p>
</span>
 <div class="t" beginAfter="TL1"  begin="4s">
    <p>
        This is some text from "div" tag that wait four seconds
           after the first timeline (the tree lines above) and begin.
           You may put here the initial text ...that will still display!
    </p>
    <p class="t" begin="3s" dur="4s">
        This is some more exciting text that appears tree
        second later into the second timeline, which should be ... visible 4 seconds?
    </p>
</div>
</t:par>
</body>
</html>
```

So, after you copy this HTML lines please observe how it works. In tree step it will display the contents of span tag in the upper part of the browser page. After two seconds the browser will display the content div tag. Please note that in this page we have two entities and so we have two referenced base time H. In the moment H+0s start the span tag and in the H+2s start the div tag. Inside of these the timelines also are described with their "local" time.
Observe the next HTML line that are the same thing with "sample006_playing.html" but we do not use the "par" tag. Please comment the different between these two programs. The comments are in the conclusions of this part. Is better to note your own comments and then read the author comments.

```
<!-- document sample006_without_par_attribut.html -->
<html xmlns:t="urn:schemas-microsoft-com:time">
<head>
<title> My first HTML pages with TIME </title>
<style type="text/css">
.t
      {behavior: url (#default#time2) ;
       }
</style>
<?import namespace="t" implementation="#default#time2">
</head>
<body>

<span class="t" id="TL1" dur="10s">
    <!-- This begins right away, and lasts for 10 seconds -->
    <p>
        This is some text that appears immediately
    </p>
    <p class="t" begin="2s" dur="2s">
        This is some text that appears after two seconds
    </p>
    <p class="t" begin="3s">
        This is some text that appears after three seconds
    </p>
</span>
 <div class="t" beginAfter="TL1" begin="2s">
    <!-- This begins slightly after the first chunk is done,
        at 10.2 seconds
     -->
    <p>
        This is some text from "div" tag that wait four seconds
           after the first timeline (the tree lines above) and begin.
           You may put here the initial text ...that will still display!
    </p>
    <p class="t" begin="3s" dur="4s">
        This is some more exciting text that appears tree
        second later into the second timeline, which should be ... visible 4 seconds?
    </p>
</div>

</body>
</html
```

6. Sequence support

When you want to have a series of images, a series of sounds and so is recommended to use a very simple declarative syntax is indicated to support in this specific case. The <t:seq> tag is provided for this purpose. You must note that this element *can* be used for other cases as well as the simple sequence, but this is not recommended. A sequence is not a good general purpose means of declaring timing structure when the document is being edited - changing from a sequence declared with the <t:seq> tag to another timing relationship requires much more work than changing timing attributes associated with individual elements.

7. The syntax of "sequence" elements

<t:seq>

This element defines a new relative timeline context. You must known the following rules for using <t:seq> element:

- All timed elements within the sequence block are timed by default as though they had the beginAfter attribute set to the timed lexical predecessor
- Any time base attributes (beginWith or beginAfter) will be ignored
- All timed elements within the sequence scope will be (at least indirectly) relative to this new timing context
- The sequence element itself is relative to the document time scope or to an enclosing timeline element.

You must note in case of using "beginAfter" timing, if any timed element does not have a specified duration, all following elements may (is possible) never begin.

8. The attributes of "sequence" elements

You can use the "seq" (sequence) element in the same manner as any other element to timing your HTML pages. An offset value and/or a time base will offset the entire local context, and shift the time of everything within the local timeline scope.

Here is a list of the timeline "seq" element attributes to keep closely in our mind:

Begin
beginWith
beginAfter
beginEvent
	Dur
	repeat
	repeatDur
	timeAction
		end
		endWith
		endSync

Contained (child) tag Attributes

Children of a sequence element can take most of the time attributes, excepting the time base. So, in the following list you can see what the accepted tags are.

Begin	Dur repeat repeatDur timeAction	End endEvent

In the follow example we show to you how to use sequence attributes in your HTML pages.

```
<!-- document sample008_sequence.html -->
<html xmlns:t="urn:schemas-microsoft-com:time">
<head>
<title> My first HTML pages with TIME </title>
<style type="text/css">
.time
     {behavior: url (#default#time2) ;
      }
</style>
<?import namespace="t" implementation="#default#time2">
</head>
<body>
```

```
<span class=time repeatDur="indefinite" timeAction="style">
  <t:seq>
    <!-- This will sequence the three styles, repeating indefinitely.  -->
    <FONT style="text-decoration:line-through; color:red"   t:dur="2">
    <FONT style="text-decoration:line-through; color:green" t:dur="2">
    <FONT style="text-decoration:line-through; color:blue"  t:dur="2">
    Here is some text that will get a really gaudy color treatment.
    </FONT>
    </FONT>
    </FONT>
  </t:seq>
</span>
<div>
  <t:seq repeatDur="indefinite">
    <!-- This will sequence the three images, repeating indefinitely.
        Any timebase parameters will be ignored, as the time base is
        implicit via the sequence block. Offsets are legal.
        Durations are recommended, as the default is to remain
        visible indefinitely, which means that nothing after that
        will ever show up.
    -->
    <img class=time src="rose1.jpg" dur="2s" timeAction="display" width="200" height="200" />
    <img class=time src="rose2.jpg" dur="2s" timeAction="display" width="200" height="200"/>
    <img class=time src="small rose.jpg" dur="2s" timeAction="display" width="200" height="200" />
  </t:seq>
</div>
</body>
</html>
```

Note that this ignores all aspects of layout. It is up to the HTML author to describe the desired layout. With the timeAction attribute set to "display", the document will reflow over time.

9. CONCLUSION (part II)

You must remember that the definition refers to "all timed elements". Also, you must know that a "timed" element is an HTML element that has any TIME attribute specified (as well as the TIME elements described in this document). Elements with no timing attributes will be ignored by the timing mechanism, and will function in a traditional static manner. If a begin delay and/or duration is defined for the sequence container, the entire container will be affected (e.g. hidden) before the begin time and after the duration completes.
For conclude what we present to you we make comments about the two examples "sample007…". If you try to "play" these examples you already observe that:

- In H+1s appear the first sentence
- Starting from the H+2s, in case the "par" element is not used the timeline appear in parallel and "beginAfter="TL1" begin="2s"" attribute was ignored
- All timed elements have the same base time and with advance time elements we can to establish different base time for different context block.
- We have attributes to synchronize the element and so we can stop an element at the desired time.

Part III:

CONTENT OF LESSON - PART III:

Introduction

In the previous parts of this lesson, solutions to make the HTML pages more attractive were given. That the designer can create a multimedia presentation without interaction that behaves just like a movie is an understatement. Code should be available to make the designer able to describe interactive responses to user actions, and to define timing variants that support interaction. In this timing operation mode, interactive timing is just an option provided to the user, and thus the beginning time is indeterminate. A beginEvent can be used by the designer to define an element (or an entire timeline container) that should begin in response to some user input. If the element is not tied to a specific event (e.g. a particular button click or a stream trigger), but rather will be started by some script on the page, the element can be defined with '...beginEvent="none"...' . Such an element can be started by a script using a simple, familiar syntax.

Exposed action methods

If the user is interested creating a HTML that supports interactive control of a timed element, he can use the following methods:

beginElement()

This command starts the element on the timeline. Method "beginElement()" applies the same action, as finite timing does, when the element begin time is reached on the local timeline. In the moment the "beginElement()" method is called, the element is attached to the local timeline at the current time, plus or minus any defined delay value. If there is any time dependency, it will be notified and aligned correctly to the local timeline. One must remember that any timed elements will be synchronized by the local time and with the starting and the ending moments. Our examples will demonstrate to the students how it can be done. Another import detail to point out is that this method takes no arguments.

endElement()

The effect of the method is to stop the element on the timeline. Method "endElement()" applies the same action, as finite timing does, when the element stop time is reached on the local timeline. When "endElement()" method is called, any time dependency is notified and, as a result, it will realigned correctly to the local timeline. Note that a sequence of elements can thus be defined with no defined duration, that will sequence correctly only if each one of them is stopped interactively using the endElement() method. This method can be combined with the use of durations to create a slideshow that can be advanced with clicks, but that will also advance itself automatically after some specified duration.
It is to be specified that this method takes no arguments as well.

Example script syntax

In order to define actions on timed HTML elements, the designer can use the familiar script events (methods) like onclick(), onmouseover(), onmouseout(), etc.. The script method implementation simply references the timed element by "id", and then calls one of the action methods exposed by the element. The students should note that, in the following, several possible functional script solutions are described for demonstration.
Let us examine together the "sample009_interaction.html" example. We created a very simple HTML page where tree pictures intended to generate a slideshow are presented. In the paragraph above the picture, the person watching the show is warned about to the possibility to increase the speed of the slideshow by clicking on the picture. When the slideshow ends, only the aforementioned paragraph and also the message that the show can be reloaded by clicking on a thumbnail (that suggests the reloading process) and a simple text that invites the user to take that click remains on the screen. This is the simplest way to demonstrate to he students how easy is to use HTML+TIME.

As an exercise for you, introduce the code below with an ordinary text editor and then "play" the page with Internet Explorer. Don't forget to save the resulting file under the appropriate extension.

```
<!-- document sample009_interaction.html -->
<html xmlns:t="urn:schemas-microsoft-com:time">
<head>
<title> My first HTML pages with TIME </title>
<style type="text/css">
.time
      {behavior: url (#default#time2) ;
       }
</style>
<?import namespace="t" implementation="#default#time2">
</head>
<body>
    <b align=center>
        If it advances too slowly for you, just click on an image
        to advance it interactively.
    </b>
 <t:seq id="SLIDESHOW" t:beginEvent="none" >
         <img class=time src="img001.jpg" dur="8s" timeAction="display"
             onclick="this.endElement()" height=200 width=300>
        <img class=time src="img002.jpg" dur="8s" timeAction="display"
             onclick="this.endElement()" height=200 width=300>
        <img class=time src="img004.jpg" dur="8s" timeAction="display"
             onclick="this.endElement()" height=200 width=300>
        <img class=time src="img005.jpg" dur="8s" timeAction="display"
             onclick="this.endElement()" height=200 width=300>
        <img class=time src="img003.jpg" dur="8s" timeAction="display"
             endEvent="onclick" height=200 width=300>
 </t:seq>
 <div>
   <table>
     <tr>
       <td>
         <img class=time src="reloadP.jpg" beginAfter="SLIDESHOW"
              timeAction="display" onclick="SLIDESHOW.beginElement()" />
       </td>
       <td>
         <p onclick="SLIDESHOW.beginElement()">
              Click here to restart the slideshow.
         </p>
       </td>
     </tr>
    </table>
 </div>
</body>
</html>
```

2. SOME THEORY

If you don't like to read so much skips the next paragraph directly to FINAL EXAMPLE. Here in this last example of our lesson you shall see a presentation like PowerPoint with sound, special effects applied to the images and also some interactions with the people how are watching this.

Controlling the start of the Document timeline

The timing syntax primarily addresses relationships established among elements on a page. However, there is also a need to define, and to be able to control the start of overall *document time*. In a simple case, it will be acceptable to start document time when the document is loaded. However, especially for long HTML documents, it might be unacceptable to defer (????) the document timeline until the document has been completely loaded. Rendering of the first screen contents is performed as soon as possible in most browsers, and designer will require that time can be started to run animations, etc. near the top of the page.
The model presented here provides simple controls for authors to control the start of document time. By default, document time begins when the document is fully loaded. This covers many cases, and simplifies the model for novice authors.

Additional settings cause document time to begin either immediately (as soon as possible), or when the document is complete (the document and all associated media and objects have been loaded). Finally, there is an advanced option for authors to specify the point in the document at which time should begin.
Add attribute to doc root (on body tag) to specify alternatives: immediate (default), onDocLoad, onDocComplete, onStartTag.

3. Document timeline start control Syntax

A new attribute is supported to specify the rule for when the document root timeline starts. This attribute is only legal on the body element. Note that if document time begins before the document has fully loaded, the author must define all timing relationships such that the timing relationships are legal when document time begins, and as the rest of the document is parsed. This means that all timing references to other timed elements (e.g. using beginWith and beginAfter) must refer to elements that are defined earlier in the document. I.e. authors may not use forward references if document time begins before the document is loaded.

timeStartRule
Attribute on body element that defines the desired start time for the root timeline in the document. The available values are:

- immediate
Document timeline should start immediately upon instantiation of the document root within the DOM (i.e. as soon as the underlying code gets going). This is the default setting

- onDocLoad
Document timeline should start when the document is fully loaded, but without waiting for any document associated media, etc. This actually ties the start of the document timeline to the window.onload event.

- onDocComplete
Document timeline should start only when the document is fully loaded, and all associated media have been loaded. This actually ties the start of the document timeline to the document.readyState changing to "complete".

- onStartTag
Document timeline should start when a startDocumentTimeline element has been parsed and instantiated. Allows an author to specify how much of a document should be loaded before the document timeline should start.

A new (XML) tag is supported to control when document time starts.

<t:startDocumentTimeline/>

Defines the point at which document should start. Only used if the body tag has the timeStartRule attribute set to onStartTag.

4. HTML+TIME Usage notes

For time attributes that reference another timed element (e.g. beginWith, beginAfter), the referenced element must be timed (i.e. it must specify one of the TIME attributes or be a TIME element) and it must be in the current timing scope. That is, the referenced element must be within the HTML sub tree defined by the closest parent time container of the current tag, and must have the same parent time container (i.e. it must be a time-sibling).

In the absence of timeline attributed containers (or timeline tags), this will be the document root and all references will be in the same scope. If a timed element using a reference is within the scope of a timeline container, the scope is the local timeline block. It is illegal to reference any timed element outside of this scope. This constraint is imposed to preclude ambiguous and potentially confusing time dependency graphs.

5. Negative Offsets

The model described by HTML+TIME explicitly allows negative offsets. The common use of these is in something like a sequence, where one object should appear just before another completes. In this common case, the final computed begin time is still positive. Nevertheless, there are situations in which a negative computed begin time can obtain; this is not considered illegal.

When the computed start time for an element is negative relative to the timeline container, the element is started with the parent (it can never appear or have influence before the time container does). However, the sync relationship of the local timeline to the parent is offset: the local timeline for the element is defined to begin before it actually appears, and so it effectively begins somewhere in the middle of its timeline. This can be useful in situations where an element is set to repeat, and the author wants the first repeat iteration to begin in the middle (repeating motion-paths, scrolling, etc. are sometimes authored this way).

6. Invalid timing definition

It is possible to describe timing relationships between or among elements that are invalid. Typically, the timing is invalid because it creates circular time-dependency references. For example, if two elements are defined to begin with one another (using either beginWith or beginEvent syntax), this is invalid. Invalid timing can result through chained combinations of begin and end timing specification. Any combination that produces a circular reference is illegal.

When an invalid timing specification is detected by the implementation, an error will be generated, and one or all of the elements involved will revert to default timing.

If both duration and any end value (including clip-end) are specified for any element, the effective duration will be the minimum of the specified duration attribute and the computed duration for the specified end attribute.

7. FINAL EXAMPLE

This “final example” is a curriculum of all your knowledge accumulated by learning this lesson. You must introduce the lines and try to explain all of them. Finally you understand how timing works in the (X)HTML pages.

```
<!—sample011_presentation.html -->
<html xmlns:t="urn:schemas-microsoft-com:time">
<head>
  <?import namespace="t" implementation="#default#time2">
  <style>.t {behavior: url(#default#time2)}</style>
</head>
<body>
<t:transitionfilter targetelement="pic01" type="clockWipe"    begin="pic01.begin" dur="3s" />
<t:transitionfilter targetelement="pic02" type="ellipseWipe"  begin="pic02.begin" dur="3s" />
<t:transitionfilter targetelement="pic03" type="fade"         begin="pic03.begin" dur="3s" />
<t:transitionfilter targetelement="pic04" type="fanWipe"      begin="pic04.begin" dur="3s" />
<t:transitionfilter targetelement="pic05" type="irisWipe"     begin="pic05.begin" dur="3s" />
<t:transitionfilter targetelement="pic06" type="slideWipe"    begin="pic06.begin" dur="3s" />
<t:transitionfilter targetelement="pic07" type="slideWipe"    begin="pic07.begin" dur="3s" />
<t:transitionfilter targetelement="pic08" type="spiralWipe"   begin="pic08.begin" dur="3s" />
<t:transitionfilter targetelement="pic09" type="pushWipe"     begin="pic09.begin" dur="3s" />
<t:transitionfilter targetelement="pic10" type="snakeWipe"    begin="pic10.begin" dur="3s" />
<t:transitionfilter targetelement="pic11" type="barnDoorWipe" begin="pic11.begin" dur="3s" />
<t:transitionfilter targetelement="pic12" type="barWipe"      begin="pic12.begin" dur="3s" />
<par>
<t:audio
src="C:\Documents and Settings\Cristi\Desktop\IPA\cristian paun\SMIL vs HTML+TIME\Sripturi &
imagini\Seasons.mp3"
repeatCount="indefinite"
type="mp3" />
<table>
<tr>
<td width="200" height="200">
<t:seq repeatCount="indefinite">
  <h2 class="t" dur="5s"> I know from where come the sound</h2>
  <h2 class="t" dur="5s"> I lost the ball</h2>
  <h2 class="t" dur="5s"> I need too</h2>
  <h2 class="t" dur="5s"> Refuel</h2>
  <h2 class="t" dur="5s"> Impossible</h2>
  <h2 class="t" dur="5s"> What is this?!</h2>
  <h2 class="t" dur="5s"> I feel new smell!</h2>
  <h2 class="t" dur="5s"> Pumping in an empty head</h2>
  <h2 class="t" dur="5s"> I will change it, is too slow!</h2>
  <h2 class="t" dur="5s"> Nothing is perfect!</h2>
  <h2 class="t" dur="5s"> Schummy brain!</h2>
  <h2 class="t" dur="5s"> This is mine now</h2>
</t:seq>
</td>
<td width="200" height="200">
<t:seq repeatCount="indefinite">
  <img id="pic01" class="t" src="C:\Documents and Settings\Cristian\My Documents\My Leodardo\Poze\I know from
where come the sound.jpg" dur="5s" />
  <img id="pic02" class="t" src="C:\Documents and Settings\Cristian\My Documents\My Leodardo\Poze\I lost the
ball.jpg" dur="5s" />
  <img id="pic03" class="t" src="C:\Documents and Settings\Cristian\My Documents\My Leodardo\Poze\I need too.jpg"
dur="5s" />
  <img id="pic04" class="t" src="C:\Documents and Settings\Cristian\My Documents\My Leodardo\Poze\Carburant
natural.jpg" dur="5s" />
  <img id="pic05" class="t" src="C:\Documents and Settings\Cristian\My Documents\My Leodardo\Poze\Impossible.jpg"
dur="5s" />
  <img id="pic06" class="t" src="C:\Documents and Settings\Cristian\My Documents\My Leodardo\Poze\La muzeu.jpg"
dur="5s" />
  <img id="pic07" class="t" src="C:\Documents and Settings\Cristian\My Documents\My Leodardo\Poze\new smell.jpg"
dur="5s" />
  <img id="pic08" class="t" src="C:\Documents and Settings\Cristian\My Documents\My Leodardo\Poze\Poming in an
empty head.jpg" dur="5s" />
  <img id="pic09" class="t" src="C:\Documents and Settings\Cristian\My Documents\My Leodardo\Poze\O schimb se
misca incet.jpg" dur="5s" />
  <img id="pic10" class="t" src="C:\Documents and Settings\Cristian\My Documents\My Leodardo\Poze\Oerson to
person.jpg" dur="5s" />
  <img id="pic11" class="t" src="C:\Documents and Settings\Cristian\My Documents\My Leodardo\Poze\Schumaher
brain.jpg" dur="5s" />
  <img id="pic12" class="t" src="C:\Documents and Settings\Cristian\My Documents\My Leodardo\Poze\This is mine
now.jpg" dur="5s" />
</t:seq>
</td>
</tr>
</table>
</par>
</body>
</html>
```

In this example you have a presentation with pictures, text and also sound. The displaying of pictures will be done with special effects when you interactionate with the slide-show-Web by clicking on text or on a picture. Also, in following rows you can read an example write in SMIL to see how you can split the display frame in few little parts. You can use these parts (named “region” in language syntax).

```
<smil>
 <head>
  <layout type="text/smil-basic-layout">
        < root-layout width="700" height="560" background-color="blue"/>
        < region id="region1" top="50" left="40"
                width="260" height="195"
                background-color="red" showbackground="whenActive" />
        < region id="region2" top="350" left="40"
                width="260" height="160"
                background-color="green" showbackground="whenActive" />
        < region id="region3" top="175" left="260"
                width="260" height="180"
                background-color="green" showbackground="whenActive" />
  </layout>
 </head>
 <body>

  <par>
        <img id="pic01" region="region1" src="rose2.jpg" dur ="5s">
                <transitionfilter type="fade" begin="pic01.begin" dur="2s" />
```

```
            </img>
            <img id="pic02" region="region1" src="rose1.jpg" begin="5s" dur ="5s">
                  <transitionfilter type="clockWipe" begin="pic02.begin" dur="2s" />
            </img>
            <img region="region2" src="second.jpg"  dur ="10s" />
            <audio src="C:\Documents and Settings\Cristian\My Documents
                              \My Leodardo\HTML+TIME\Mis-Teeq - Scandalous (Radio Edit)" />
            <video region="region3" src=" C:\Documents and Settings\Cristian\My Documents
                              \My Leodardo\HTML+TIME\Filmare Breaza.avi" />
    </par>
  </body>
</smil>
```

8. Differences between SMIL and TIME

HTML+TIME are based mainly on SMIL, and so the syntax is very similar. Because of the integration with HTML and CSS, the SMIL layout syntax is not needed. However, there are some differences in the timing syntax. Some of these changes allow the syntax to match the HTML Document Object Model (DOM) in a more standard manner. In other cases, HTML+TIME extend the SMIL functionality to support more control and integration with a document time model.

As the layout is handled by existing HTML/CSS mechanisms, there is no layout section as there would be in SMIL documents. HTML+TIME effectively integrate layout and timing. This keeps the information relating to an element all together.

On a minor note, SMIL naming (using hyphens) seems to follow the CSS property naming practice, in conflict with the HTML attribute naming practice. HTML+TIME preserve the SMIL names with hyphens for compatibility, but use the HTML naming standard for new attributes.

One significant difference between HTML+TIME and SMIL is the description of an Object Model. This is appropriate in the HTML environment, and should support platform implementers as well as designers.

Key Point Summary Conclusions and Recommendations

1. For real presentation you must made scripts. On time line you must put all the actions of your presentation to see where these start and end, synchronization moments, what is happening in parallel (in same time), what could be a sequential presentation, the proprieties of any objects (image, sound, video, streaming, etc.)
2. Use the manuals when you are working and read the new documentation on Web sites, discussion lists on this subject.
3. After you read a new idea from the text try to understand and improve your knowledge by new readings.

BIBLIOGRAPHY. REFERENCES.

[1.] Mihaela Brut, Sabin Braga, *Prezentări multimedia pe WEB – Limbajele HTML+TIME şi SMIL (transl. Multimedia Presentation on Web – HTML+TIME and SMIL languages*), First Edition, Original Copyright © by Editura Polirom 2003, Printed in Romania by Editura Polirom Iaşi, B-dul Carol I nr.4, P.O.BOX 266, 700506, Romania, www.polirom.ro, ISBN: 973-681-521-8

SUPPLEMENTARY IMPORTANT BIBLIOGRAPHY. REFERENCES. (www)

[SUP.1] http://www.w3.org/TR/SMIL2/ for understanding SMIL and to know all revisions.
[SUP.2] http://www.w3.org/TR/1998/NOTE-HTMLplusTIME-19980918/ for understanding HTML+TIME and to access all revisions.
[SUP.3] http://msdn.microsoft.com/library/?url=/workshop/author/behaviors/reference/time2/htime_node_entry.asp for understanding that HTML+TIME directly from Microsoft site.
[SUP.4] http://www.w3.org/AudioVideo/ for study hardly.
[SUP.5] http://www.w3schools.com/xml/default.asp for understanding that HTML+TIME and SMIL are based on XML.
[SUP.6] http://www.w3schools.com/dom/default.asp for understanding XML DOM.
[SUP.7] http://www.w3schools.com/vbscript/default.asp for understanding how to connect HTML+TIME and SMIL knowledge with Visual Basic language.
[SUP.8] http://www.w3schools.com/js/default.asp for understanding how to connect HTML+TIME and SMIL knowledge with Java Script language.

SUPPLEMENTARY INDICATIONS ABOUT THE CONTENT OF THE LESSON

1. This lesson is an introduction in field and you must study thoroughly!
2. Use the manuals when you are working and read the new documentation.
3. After you read a new idea from the text try to understand and improve your knowledge by new readings.
4. Try to make your own presentation and step by step you shall assimilate all these new elements.
5. SMIL and HTML+TIME have a very limited set of tag within you can create intelligent presentation in e-commerce or e-business.
6. You must combine the knowledge from this lesson with JavaScript or other programming language that offer XML support.

RESPONSES TO THE QUESTIONS

1. HTML+TIME is acronym of HTML plus **T**imed **I**nteractive **M**ultimedia **E**xtensions
2. SMIL is achronym of **S**ynchronized **M**ultimedia **I**ntegration **L**anguage.

3. See fig.2, first, and after this by adding timing elements.
4. The timing element are t:excl for eclusive execution, t:par for an executions of task in parallel and t:seq for sequential execution.
5. By using HTML tag attributes like beginWith, endWith, beginAfter, end, dur, end Event and so on.

WORDS TO THE LEARNER: “Do not wait opportunities. Create them “(After Bernard Show)