How does Google Docs put such amazing functionality into a Web application? They leverage Web 2.0 technologies, which provide robust functionality with relatively simple code. In this article, learn how to build a Web application to create slideshow presentations using Asynchronous JavaScript and XML (Ajax) technology.

Introduction

Web 2.0 technologies are starting to show their potential behind corporate firewalls, and more companies want to create real business value with Web 2.0. We were shocked to see Google Docs for the first time. It's amazing that such a complex application can be done so beautifully through the Web. In this three-part series, we'll show you how to build a Web application like Google Docs using Ajax technology. There are three types of common office documents: text, spreadsheet, and presentation. This series focuses on the presentation format.

In Part 1, we'll look at user interface (UI) design, which is one of the most important aspects of online applications. Users tend to prefer software that they are familiar with, so making a cool and familiar presentation GUI on the Web greatly affects users' perception of our tool. Within this article, we'll focus on the key elements of a good interface and get one working.

Part 2 will focus on adding editing features to our presentation application. We'll change font styles, drag and drop pictures in the presentation, and modify our text content. All of these features will be covered in Part 2.

Finally, in Part 3, we'll look at persistence. Our philosophy is, a Web application should be compatible with existing ones. We'll export or import other presentation files into our system, keeping in mind these questions:

- How do we transform one type of file to another?
- How do we save current works for user later use?
• How do we share presentations with other people?

Right now, let's start developing the user interface.

Developing the Web UI

Our design is similar to the screen shown in Figure 1 (see larger version). The LeftFrame is the miniFrame, which is used to show all the slides in the presentation, just like Power Point or OpenOffice.org. The middle window is the mainFrame, where users can see and edit the content of slide. The right frame indicates the preferences of the selected element in the mainFrame.

Figure 1. Design layout

The DIV object and our application framework

In this project, we will use the DHTML object div to help format our views. It not only has the capability to execute commands that conveniently change text styles, but also features zooming the content in and out. It is the most important element in our presentation, and nearly 50% of our code deals with div objects.

There are four main div objects in our framework, as you can see in Figure 2 (view larger image). Because ToolBar and LeftFrame are comparatively easy to implement, we'll focus on the miniFrame and mainFrame implementation in this article.
Figure 2. Interface zones

The miniFrame

MiniFrame contains all of the slides for the current presentation. This window has two major functions. First, it must manage all slides, including adding and deleting slides. Second, when users change the contents of the mainFrame, corresponding modifications must also be indicated in miniFrame. People have come to expect this in their desktop applications, so we want to deliver the same functionality on the Web.

Managing slides - the design of miniFrame

As miniFrame is a container, we believe that the DIV object is the best choice, because the DIV object has the ability to contain any other elements of DHTML. Additionally, each slide is also presented by a DIV object. Figure 3 shows the relationships between the objects in miniFrame and the object in mainFrame. We'll refer to miniFrame's large container of slides as Outside-DIV. Within that container will be a number of different small DIV slides, which we refer to as Inside-DIV.
Listing 1 shows the code that creates miniFrame.

**Listning 1. DIV definition of miniFrame**

```html
<DIV id=miniFrame SCROLLING=NO>
  <div id='slide1' style="POSITION: absolute; SCROLLING="NO">
  </div>
  <div id='slide2' style="POSITION: absolute; SCROLLING="NO">
  </div>
</DIV>
```

**Managing slides - adding functionality**

Obviously, to manage slides you need to be able to create them. So, next we'll create functionality for adding and deleting slides. To do this, we need to manipulate Outside-DIV and reassign its contents. For the Add operation, we can append new slide code to Outside-DIV using DHTML's `innerHTML` function. To guarantee that existing slides can maintain their state, each slide has its own unique ID and order indicator. We'll establish a variable called `slideIndex` to indicate different slides. The code is shown below in Listing 2.

**Listing 2. NewSlide()**

```javascript
function NewSlide(SlideIndex) {
  window.document.getElementById("Minileft").innerHTML =
  window.document.getElementById("Minileft").innerHTML +
  "<div " + " style='position:absolute; top: "+(SlideIndex)*175+" px;'
+ "id='slide"+SlideIndex+
"</div>";
}
```

For the Delete operation, we need a little more complexity. We don't know which slide a user will want to delete, so we'll use the variable `delSlideIndex` to cover this. When we delete a slide, we will rearrange each slide's position. The code is shown in Listing 3.
Listing 3. DelSlide()

```javascript
function DelSlide(delSlideIndex) {
    var slides=window.document.getElementById("Minileft").all.tags('DIV');
    slides[delSlideIndex].removeNode(true);
    for(var no=delSlideIndex+1;no<slides .length;no+){
        slides[no].style.top=parseInt(slides[no].style.top)- 175;
    }
}
```

Dynamic preview

As in PowerPoint or other presentation applications, when users change the content in `mainFrame`, the related mini-slide should show the modification simultaneously. This is the way it is shown in desktop versions, so we want to match that functionality to keep the users comfortable. You can see an example of this effect in Figure 4 below.

Figure 4. Synchronizing the mini-slide with the edit window

To implement this feature, we must detect the content-change event in `mainFrame`. In DHTML, there are two possible events you can use to do this: `onkeyup()` and `onmouseup()`. We initiate the two event-catching functions in `onPageLoad()`. You can see this code in Listing 4.

Listing 4, onPageLoad()

```javascript
function onPageLoad() {
    slideFrame.document.onmouseup = updataMainFrame;;
    slideWindow.document.onkeyup = updataMainFrame;
}
```

The function of `updataMainFrame()` just copies the current content of `mainFrame` to the corresponding mini-slide. The variable `currentSlide` indicates which one has the focus. See this code in Listing 5.
Listing 5. updateMainFrame()

```javascript
function updateMainFrame(CurrentSlide) {
    window.document.getElementById("slide"+CurrentSlide).innerHTML=
    window.document.getElementById("MainFrame").innerHTML;
}
```

The mainFrame

The middle window is the mainFrame that we use to show the editable content of your presentation. Users can edit pictures and text objects in this window. We implement it with another DIV element because this object can contain all other objects including itself. Because the DIV object has an editable mode, when we turn that on, all objects in the DIV can be altered. This will save us a lot of work in implementing our online presentation. Figure 5 highlights the mainFrame for clarity.

Figure 5. The mainFrame

To create a DIV that can be editable, we simply add the following code to our HTML:

```html
<div id="mainFrame" contentEditable='true' SCROLLING=NO
    style="overflow:hidden;" frameborder="0" ></div>
```

Setting the contentEditable attribute to true is the key. Figure 6 shows that when a DIV element in mainFrame has contentEditable set to true, clicking an object will show it in edit mode.
Figure 6. A DIV element in edit mode

The slide show

You may think that implementing a slide show on a Web site would be very difficult. In reality, DHMTL gives you powerful tools to accomplish this. For our project, we'll need to consider effects for each slide and display them in a large size for viewing.

To achieve the slide show effects, we must first set a time to control our show procedure. Using Javascript, you can do this easily with the code in Listing 6.

Listing 6. Setting slide show effects

```javascript
function SetTime()
{
    TimeOutID = window.setInterval('doIt();',1000);
}
```

`setInterval()` is the key function. `TimeOutID` is the return value used to terminate the timer. `doIt()` is the procedure we need to trigger after each interval. The value 1000 sets the interval to 1000ms.

Next, we create a full window to show every slide and fetch associated contents into it. In Listing 7, we use variable `EndIndex` to judge whether the slide reaches the last one. `ShowIndex` records the slide currently showing. To make the view full screen, we just set our `mainFrame` to a suitable size. All this is done in the `doIt()` function.
Listing 7. Creating the full screen view.

```javascript
function doIt()
{
    if(ShowIndex==EndIndex){
        ShowIndex=0;
        clearInterval(TimeOutID);
        window.document.getElementById("MainFrame").style.zoom="100%";
        return;
    }

    MainFrame.document.body.innerHTML=
        window.document.getElementById("slide"+ShowIndex).contentWindow.document.body.innerHTML;
    if(ShowIndex==0)
      {
        window.document.getElementById("MainFrame").style.zoom="150%";
      }
    ShowIndex++;
}
```

Users generally want to control their slides manually while viewing in Slide Show mode. This requires a control bar, which should meet the following requirements:

- Appear only when in Slide Show mode
- Indicate each slide's index
- Provide a clear, controls

Our example is shown in Figure 7 (see larger image).

Figure 7. Slide view controls

Listing 8 shows the HTML code required to create this control panel. Listing 9 shows the associated JavaScript code.

Listing 8. HTML code for Slide Show control panel

```html
<div id=controlbar style="color: blue; position:absolute; width:1300;height:500; left:7px; top:595px;z-index: 5; visibility:hidden;">
<table border=0 width="1045" cellspacing="0" bordercolor="#EEEEEE" cellpadding="0" bordercolor="#EEEEEE" bgcolor="#C0C0C0" height="34">
<tr>
    <td width="100%" height="34">
        <table border=0 width="550" cellspacing="0" bordercolor="#EEEEEE" cellpadding="0" >
            <tr>
                <td width="97">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n```
Listing 9. JavaScript code for Slide Show control panel

```javascript
function myforword()
{
  slideindex--;  
  doIt();
}

function myauto()
{
  window.document.getElementById("B4").disabled=false; 
  window.document.getElementById("B2").disabled=true; 
  TimeOutID = window.setInterval('autodoit();', 1000); 
}

function mycancel()
{
  slideindex=-1;  
  doIt();
}

function myback()
{
  slideindex++;  
  doIt();
}

function autodoit()
{
  slideindex++;  
  doIt();
}
```

**Conclusion**

Using DHTML and the `div` object with Cascading StyleSheets (CSS), it is relatively easy to make a presentation-like application for the Web. The sample code provides some additional demonstrations of how to use JavaScript DOM to handle CSS. We'll expand on all of this in Part 2 as we add additional functions.