

A Dynamic Page Editor

Using DHTML

By

JOONGSEOK PARK

Bachelor of Science

The Ohio State University

Columbus, Ohio

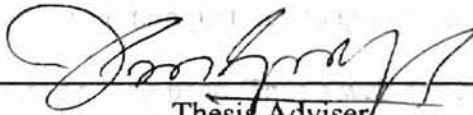
1994

Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
July, 1999

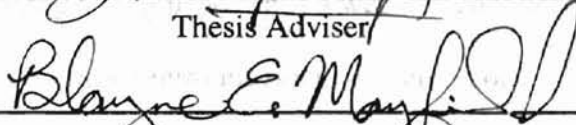
## A Dynamic Page Editor

### Using DHTML

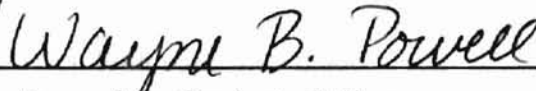
Thesis Approved:



Thesis Adviser







Dean of the Graduate College

## VII PREFACE

Hypertext systems provide an efficient way to manage and browse information. Frequently, users want to create new functions to satisfy their special need or to access different documents under the standard system. One of the solutions to this problem is to extend the functionality of Web components that include Web Server, Browser, HTTP, and HTML.

When users want to print a certain part in a long web page, standard HTML can't provide a method of breaking a long page. Thus a web designer needs an efficient way of managing this problem. This thesis introduces a new method of dividing a long page into multiple logical pages using dynamic HTML (DHTML). A new tool called the "Dynamic Page Editor (DPE)" is developed. By using this new approach, users can generate separate logical pages from a given long page they made.

DHTML cooperates with CSS, CSSP, and JavaScript to generate dynamic pages. The DPE allows users to create web pages that use different styles in different parts of the page, as well as to show and print a certain page that users want to print. Advantages of the DPE include removing redundant frame sets and creating logical pages from a long web page.

## AKNOWLEDGMENTS

I would like to express my sincere appreciation and gratitude to my thesis advisor, Dr. K. M. George for his inspiration, guidance, encouragement and help for the completion of my thesis work. His patience and constructive ideas helped me make this thesis work an enjoyable and memorable experience. I would also like to express my deep sense of gratitude and appreciation to Dr. John P. Chandler and Dr. Blayne Mayfield for serving on my graduate committee and for providing valuable suggestion and ideas.

I want to my greatest appreciation, thanks, and love to my parents Jaemoon Park, and Kihwa Kim for encouragement and supports. And during the study, research and completion of this thesis I have experienced mush joy and pleasure because of my wife Jiyoun Lee and our new daughter Lauren Park.

<i>Chapter</i>	<i>Page</i>
1.1	21

## TABLE OF CONTENTS

<b><i>Chapter</i></b>	<b><i>Page</i></b>
Chapter 1: INTRODUCTION .....	1
Chapter 2: WEB COMPONENTS .....	6
2.1 Server Side Components .....	8
2.1.1 CGI .....	8
2.1.2 Server API .....	8
2.1.3 Server Side Includes .....	8
2.1.4 HTTP Cookies .....	9
2.2 Browser Side Components .....	9
2.2.1 HotJava Browser .....	9
2.2.2 Netscape Navigator .....	9
2.2.3 Internet Explorer .....	10
2.3 Web Protocols .....	10
2.3.1 Resource Addressing .....	10
2.3.2 Data Transfer with HTTP .....	11
2.4 HTML .....	12
Chapter 3: WWW DEVELOPMENT TOOLS .....	14
3.1 Style Sheets .....	14
3.1.1 CSS .....	14
3.1.2 CSSP .....	15
3.2 A Script Language .....	18
3.2.1 Core JavaScript .....	18
3.2.2 VBScript .....	21

<b>Chapter</b>	<b>Page</b>
3.3 A Document Object Model .....	21
3.4 Global View of Development Tools .....	22
<b>LIST OF TABLES</b>	
Chapter 4: DESIGN AND IMPLEMENTATION OF DPE .....	24
<i>Table</i>	
4.1 Dynamic Page Editor(DPE) .....	24
4.2 Event Sequence in DPE .....	26
4.3 Properties of DPE .....	28
4.4 Implementation Scheme of DPE using DHTML .....	31
Chapter 5: TEST .....	39
Chapter 6: CONCLUSIONS .....	45
APPENDIXES .....	46
A.1 DPE with Internet Explorer .....	46
A.2 Error Messages .....	47
A.3 Basic HTML Tag reference .....	48
A.4 List of Events.....	49
A.5 HTML code for page break.....	50
REFERENCE .....	53

## LIST OF TABLES

<i>Table</i>	<i>Page</i>
I. The Properties of CSSP.....	16
II. Summary of the Development tools.....	23
III. The Navigator Object's Browser Identification Properties.....	28
IV. The Differences between Netscape and Internet Explorer.....	30

<i>Figure</i>		<i>Page</i>
20b	HTTP	43
21	HTTP	44
<b>LIST OF FIGURES</b>		
		44

<i>Figure</i>	<i>Page</i>
1. Broad and Shallow Hierarchy .....	2
2. Relationships between the sheet and page margins .....	3
3. Page Partition using DPE .....	4
4. Web Browser Domains .....	7
5. URL Structure .....	10
6. HTTP Process .....	12
7. Example of CSS code.....	17
8. Core JavaScript .....	19
9. JavaScript DOM.....	20
10. JavaScript Object Model accesses a small subject of web documents .....	22
11. DHTML accesses all objects.....	26
12. User Action and Browser's Response.....	27
13. The CheckBrowser() function .....	29
14. The Initial DPE window .....	31
15. "Page Setting" for page breaks function .....	33
16. The Areas for page breaks .....	35
17. Web Page without page break .....	40
18a. Display of first page in the editor window .....	41
18b. The pop-up window for the first page.....	41
19a. JPEG files in DPE .....	42
19b. The pop-up window for the second page .....	42
20a. <TABLE> tags in DPE.....	43



<b>Figure</b>	<i>Chapter 1 Introduction</i>	<b>Page</b>
20b. Table displayed in pop-up window .....		43
21a. Standard HTML tags in DPE .....		44
21b. The pop-up window for the fourth page.....		44

## **Chapter 1 Introduction**

Generally, the current browsers send a message requesting a web page via a URL to a server. A server receives the request and retrieves the requested web page, which is composed in HTML, and sends it to the requesting browser. When the browser receives the page, it displays the HTML, which includes text and graphics. The browser accomplishes this by interpreting the HTML document. Until recently, Web browsers have rendered HTML documents as a single column scrolling window and allowed users to create, move and resize document frames. Users can then view and change properties of these frames that include the background color, the border style, and so on.

Many web pages rely on a scrollbar to allow users to scroll through material too large to fit into a frame, however this is not comfortable for users who want to see only certain parts of a long web page. Sometimes, users also may encounter the problem of a long Web page that can't be broken up into smaller pages. Hence it is not possible to print a specific portion of a web page as a print page. The current browsers do not support the page break function to generate pages [30]. That means the current browsers can't split a server-supplied page into one or more pages and print the original long web page in meaningful segments. However, the current browsers suggest one option to generate pages. This technique scatters `<A NAME = "anchorName"> </A>` tags throughout the page. The tags provide links to segments of documents. Usually these tags appear at the top of the page. The tags at the top of page will link to the page position of those named anchors, e.g. `<A HREF= "anchorName"> linkText </A>`. The major problem with this technique is that once users navigate to a page segment, they have to scroll back up or down to find the links to continue to another part of the page. Many times designers add

GO TO TOP links to take users back to the navigation links. This method does not generate print pages from the existing long web page because the current browsers do not support page breaks in web documents.

Figure 1 illustrates the standard web document's hierarchy. Breadth refers to the number of options at each level of hierarchy: Depth refers to number of levels in the hierarchy. If a hierarchy is too broad and shallow, users are faced with too many links on the menu at the top of the page. Many times users have to visit the top of a page when they want to find what they are looking for. And it is impossible to print a certain portion of a long page that looks like page A (see figure 1).

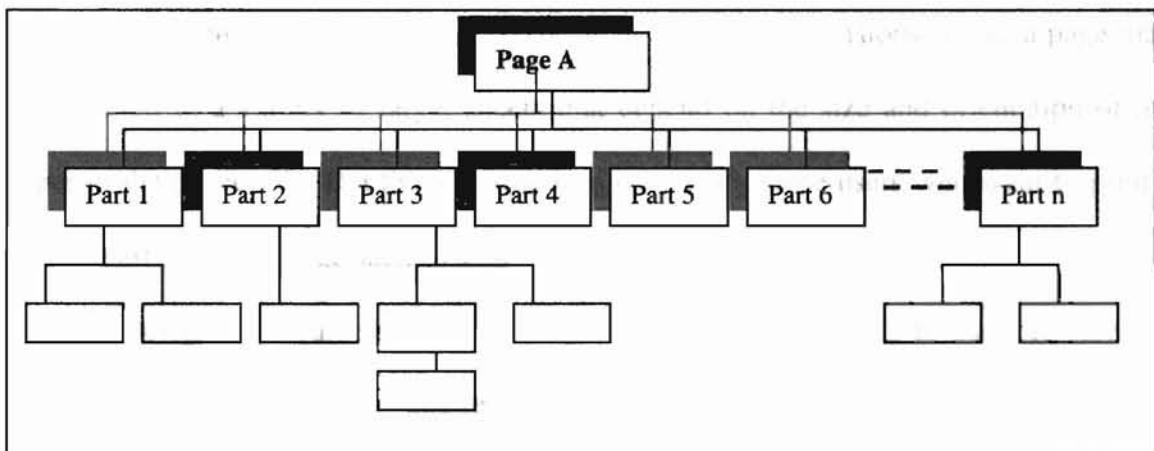


Figure 1. Broad and Shallow Hierarchy.

The current browser can display different pages by using margin or font properties. According to change in the properties of margin and font, the browser window shows different layout of the content of document. The margin properties ('margin-top,' 'margin-right,' 'margin-bottom,' 'margin-left,' and 'margin-right') apply within the page content. For example, the size of the content will be changed by the properties of margin and font. If a user assigns a large size margin or font, then the size of the content in each physical page will be decreased.

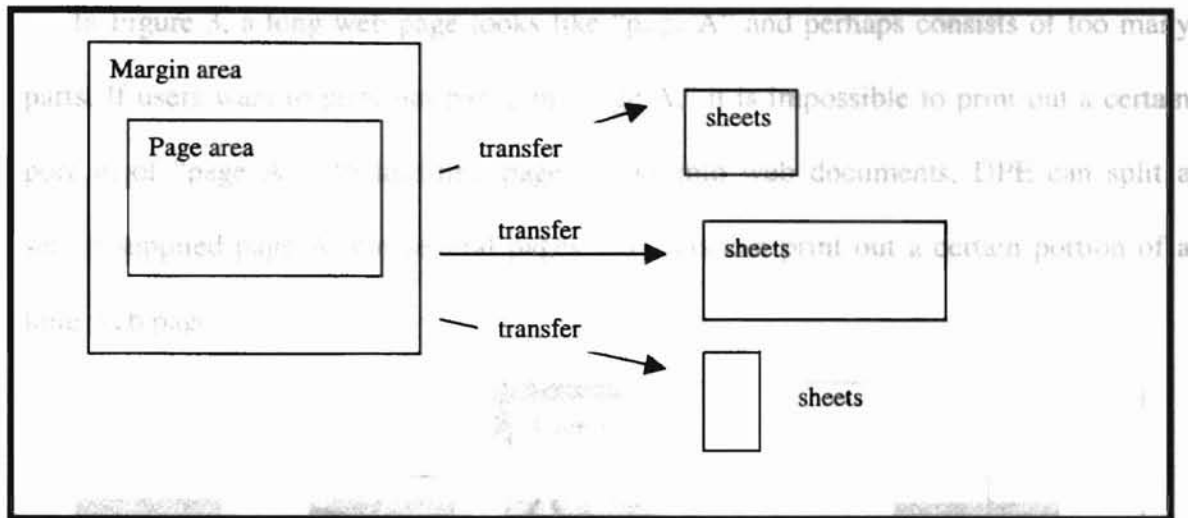


Figure 2. Relationships between the sheet and page margins.

The box in the left in Figure 2 represents a page, and three arrows (each labeled "transfer") are shown pointing to three sheets of different dimensions. Thus, a page may be rendered to a variety of target sheets that depend on the size and orientation of the paper in the printer [2]. But this method does not satisfy those users who want to print a certain part of a long page such as a table.

Web designers should have the facility to include page breaks to support viewers' needs. Therefore, the topic of this research is to develop a method of incorporating page breaks into a web document.

To solve the above-mentioned problem, a new tool called Dynamic Page Editor (DPE) is designed to generate page breaks in web documents. DPE uses DHTML that cooperates with CSS, CSSP, and JavaScript to generate dynamic pages. By inserting page breaks in web documents, users can keep the same long page, but display only the page in the browser window that users want to print. Also DPE allows users to change the properties of the frames including the background color, the border style and font size for text.

In Figure 3, a long web page looks like “page A” and perhaps consists of too many parts. If users want to print out part 2 in “page A,” it is impossible to print out a certain portion of “page A.” By inserting page breaks into web documents, DPE can split a server-supplied page A into several pages. So users can print out a certain portion of a long web page.

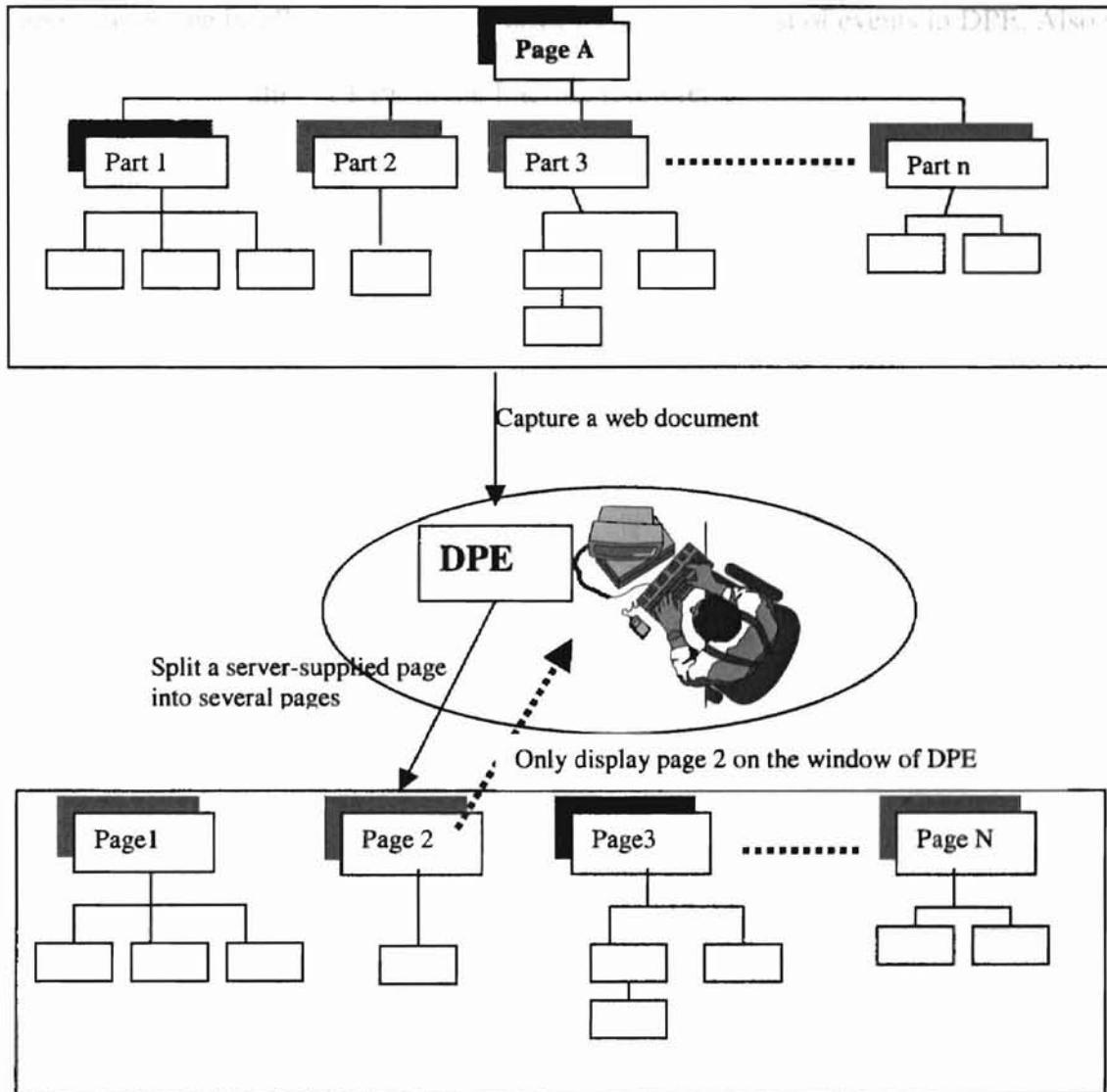


Figure 3. Page Partition using DPE.

The thesis is structured as follows. Chapter 2 briefly reviews and summarizes the current web components. In chapter 3, we discuss WWW development tools to generate page breaks, namely Cascade Style Sheet (CSS), CSSP (Cascade Style Sheets Position), and JavaScript. Chapter 4 introduces DPE and describes its implementation method within current web browser. In chapter 5, we demonstrate the functionality of DPE. In appendixes, we briefly describe some error messages and list of events in DPE. Also we show the functionality of DPE in the Internet Explorer.

## ***Chapter 2 Web Components***

This chapter briefly reviews the web components. This thesis focuses on the client side browser and HTML parts in the web components. The Web is a vast collection of documents on the Internet that are linked together via hyperlinks [23]. The Internet consists of millions of computers worldwide that communicate electronically. A hyperlink is a predefined linkage between two documents. The hyperlinks allow a user to access documents on various Web Servers without concern for their location. Users gain access to the Web through a browser. A browser fetches documents from Web servers and displays them to the user.

In building a web Browser, the web designers need to understand a number of distinctly different areas, called domains. A domain contains a distinct set of objects that behave according to rules and policies characteristic of the domain. For example, browser domains deal with installation, navigation, bookmarks, and so on, while a graphical user interface domain is concerned with menus, buttons, windows, and so on [27].

Figure 4 shows a domain model of a web browser. There are three major domains namely, application domain, service domain, and implementation domain.

The application domains are the subject matter of the system from the user's perspective such as Client side browser. The service domains produce a generic mechanism and utility functions required supporting the application domains. The implementation domains have entities of the system (operating system, network, programming language). The client browser uses the services of a Graphical Users Interface (GUI), the Common Client Interface (CCI) and handlers for different protocols, data formats, and language.

## 2.1 Server Side Components

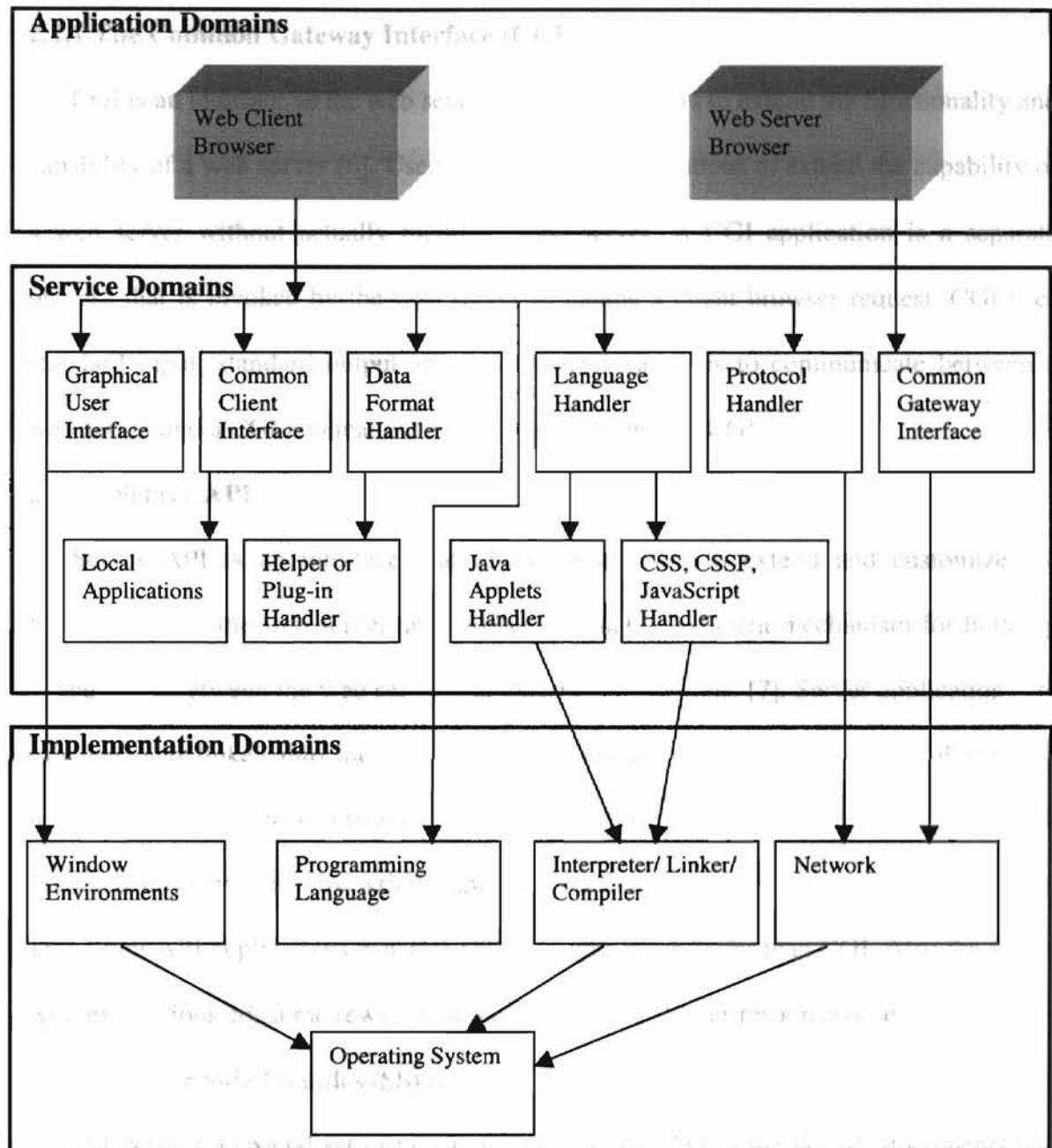


Figure 4. Web Browser Domains (adopted from [29]).



## **2.1 Server Side Components**

### **2.1.1 The Common Gateway Interface (CGI):**

CGI is an interface to the web server that enables users to extend the functionality and capability of a web server [6]. Users can use CGI applications to extend the capability of a web server without actually modifying the server. A CGI application is a separate process that is invoked by the web server to handle a client browser request. CGI uses standard input, standard output and environment variables to communicate between a web server and a CGI application. All web servers support CGI.

### **2.1.2 Server API:**

Server API is an interface that allows developers to extend and customize the functionality of the web server and provide a scalable, efficient mechanism for building connections between the web server and external applications [7]. Server applications are executed and linked into the web server dynamically as if they were part of the web server itself. If a user understands the API of a server, the server's functionality can be extended in many ways by writing applications that process the input in various ways. The server API applications generate HTML pages much faster than CGI. Also the server API applications consume fewer resources and have a shorter response time.

### **2.1.3 Server Side Includes (SSI):**

SSI defines a special set of tags that can be embedded in the HTML documents and preprocessed by the web server before they are sent to a browser. SSI does not require any external interface and is generally easier to be implemented than CGI or Server APIs.

#### **2.1.4 HTTP Cookies:**

Cookies are an HTTP mechanism proposed by Netscape Communications and are supported by several other browsers as well. Cookies are designed to communicate state information to the browser from the web server. When a browser accesses a CGI application containing a cookie header, that cookie type is transferred back to the browser along with the CGI response [8].

### **2.2 Browser Side Components**

There are several browsers currently available. Some of the popular browsers are listed in this section.

#### **2.2.1 HotJava Browser**

HotJava is the first browser to offer full support for the Java language [31]. In addition to allowing clients to download the normal components such as images, documents, sounds, etc., distributed Java applets can also be downloaded. Applets are (usually) small executable programs written in the Java programming language and included in HTML pages, similar to commonly embedded images.

While HotJava is the first browser to implement the applet tag, Netscape and Internet Explorer have followed. They also provide more features than HotJava.

#### **2.2.2 Netscape Navigator**

Navigator deals with unknown file types using helper applications and plug-ins. The plug-in API includes functionality for windowless, transparent plug-ins. Navigator supports JavaScript and Java applets, and also allows users to customize interfaces slightly but not to change menus.

### 2.2.3 Internet Explorer

Internet Explorer supports shell integration, which largely dissolves the dividing line between the operating system and the browser. In addition, the Internet Explorer supports helper applications and plug-ins, along with supporting Java applets and ActiveX (Jscript and VBScript)[31].

## 2.3 Web Protocols

The WWW (World Wide Web) consists of a body of information protocols, standards, and conventions that govern their use. These information protocols can differ greatly from one another but they all allow clients and servers to communicate. Some of the most common WWW protocols and utilities are Resource Addressing, and Data Transfer.

### 2.3.1 Resource Addressing

URL (Uniform Resource Locators) are a standard for identifying objects in the WWW. Every document has a URL that serves as its network-wide address. Documents can form links to others by including their URLs. A URL is a string of characters that uniquely identifies an object in the WWW. The URL describes objects anywhere on the Internet and these objects are accessed using different protocols.

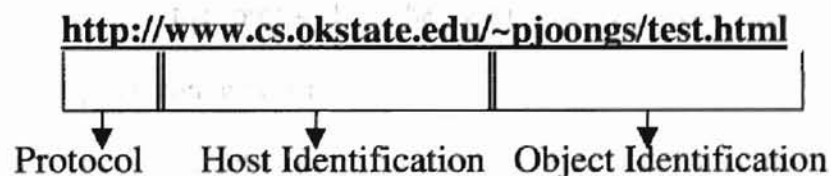


Figure 5. URL Structure.

Figure 5 shows the structure of a URL. It has three parts, namely protocol, host identification, and object identification. The protocol portion specifies the protocol (e.g. http, ftp) to be used by the browser to communicate with the WWW server. The host

identification portion of the URL identifies the server where the required information resides. Generally, the host ID consists of a cluster name, a domain name and a port number. The object identification portion of the URL has two different types of identifiers. One of them refers to HTML files and the other refers to executable programs.

### 2.3.2 Data Transfer with HTTP

HTTP is a "request-response" type protocol specifying that a client will open a connection to a server and then sends a request using a very specific format. The server will then respond and close the connection [9].

HTTP is an open standard, a connectionless protocol, which allows many quick connections without having to hold the port open. After the server has responded to the client's request, the connection between the client and server is closed. There is no "memory" between client connections. Figure 6 shows the HTTP process.

All HTTP transactions take place over a TCP/IP connection and an HTTP transaction consists of four stages.

1. Connection: The browser (client) attempts to connect with the server. The status of the connection is displayed on the status line on most browsers.
2. Request: After a connection is established, the client sends a request to the WWW server specifying the protocol to be used, the required document, and the document types it can understand.
3. Response: The server either sends an error message if it cannot fulfill the request or responds with the document if it can fulfill the request. The browser might display a "reading response" message or a "transferring" message on the status line.

4. Close: after the server sends the response, either the client, the server, or both close the connection.

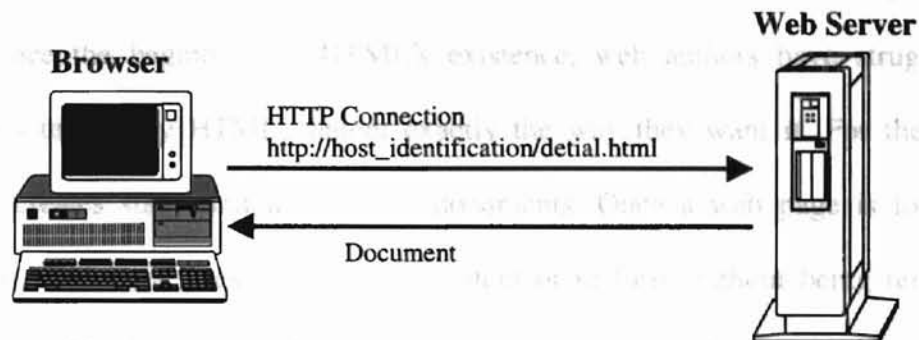


Figure 6. HTTP Process (adapted from [9]).

## 2.4 HyperText Markup Language (HTML)

Web documents are developed using the language HTML. HTML allows browsers to display documents based on the logical layout of the document. HTML was designed to be an application of the Standard Generalized Markup Language (SGML,[10]), that is, a class of documents conforming to an SGML Document Type Definition (DTD) defining “HTML documents.” It does not require exact formatting since different screens would display things differently. Instead, HTML allows users to describe what the document is and allows the browser the flexibility to display the text however it looks best on the screen.

HTML is a good tool for publishing lightweight (small file size) documents over the Internet. However, HTML in its traditional form is not powerful enough to create the interactive and multimedia-rich documents that today's commercial web sites demand [16].

Control over page layout is one of HTML's most obvious limitations. Even with today's third-generation HTML specifications, web authors do not have the ability to lay out pages with the pixel-level accuracy available to the traditional desktop publishing author

[17]. A standard HTML document cannot specify that text and images be located at exact coordinates, on top of each other, or even that the text be displayed in a particular font size. Since the beginning of HTML's existence, web authors have struggled to get browsers to display HTML content exactly the way they want it. For the most part, HTML creates static and unanimated documents. Once a web page is loaded into a browser window, it doesn't change in content or in form without being reloaded. This truly limits HTML's potential as an interactive multimedia format. To make the web a more compelling medium, HTML must provide web authors with the ability to dynamically update content, change the appearance of content, and hide, show, and animate content [18].

## **Chapter 3 WWW Development Tools**

DHTML (Dynamic HTML), CSS (Cascade Style sheet), CSSP (Cascade Style Sheet Position), and JavaScript are the tools used in the development of the DPE. Dynamic HTML changes the way a designer designs. Web designers will not have to rethink their design because HTML is unable to handle a specific task. If a user can imagine it, DHTML will allow designs to do it. DHTML is not a new version of HTML, nor is it just a set of a few new tags. DHTML provides a new way of thinking about Web page design [3]. DHTML consists of style sheets (CSS), a scripting language (JavaScript or VBScript) and a Document Object Model. Explanation of these three components follows:

### **3.1 Style Sheets**

A Style Sheet allows users to describe precisely how web pages should look and exactly where and how things should appear.

#### **3.1.1 CSS**

CSS is applied to elements, blocks, classes of elements, page layout (margin, border padding), font size and style (bold, italic.), and text properties (case, underline, color, alignment, indent etc.) [11]. CSS can support the separation of format and content and makes maintenance easier and faster. The benefit of a style sheet is to maintain consistency across organization and simplify the exchange of information. Also the style sheets can reformat documents for different audiences.

CSS is a way to centrally control the formatting of a document. CSS is a convenient, quick way to apply sophisticated formatting, and is especially useful when dealing with a lengthy document [11]. CSS will improve the printing of Web documents. Paper has

different properties from a computer screen and the differences can be accounted for in a style sheet. Web authors should be confident that their documents will look as good on paper as they do on a computer screen [12].

CSS allows a user to separate the style and layout of HTML files from their informational content. To generate page breaks in a long web page, a unique identifier is needed for each page. The ID can serve as the page number.

The value of an ID attribute must be unique throughout the document. That is, every element inside `<body> ...</body>` tags can have an ID attribute, but the values must all be different. This makes the ID attribute useful for setting style rules on individual element [12]. A selector that includes an ID attribute is called an ID selector. The general form of an ID selector is shown below:

ID Selector	Declaration	element
#page-number	{ set the properties }	What to do with a

### 3.1.2 CSSP

CSSP consists of positions, stacks, and hides HTML elements. Designers want to explicitly control the position of HTML elements to produce rich static HTML documents. They also want powerful layout control to enable dynamic, animated HTML-based content. CSSP provides two methods of positioning HTML elements using its new “position” property as follows: 1) “Relative” positioning allows elements to be offset relative to their natural position in the document's flow. The “relative” positioning is most useful in scripting environments, where animation effects can be created through relative positioning. 2) “Absolute” positioning allows elements to be removed from the document's flow and positioned arbitrarily. This means, an absolute positioned element is



laid out independently of both its parent and child elements, without regard to their dimensions or position. For that matter, the layout of each 'absolute' positioned element is independent of any other. Dynamic aspects of managing positioned elements, such as hiding, displaying, and movement can only be performed only by using an external scripting language. Table I shows the properties of CSSP.

Table I: The Properties of CSSP (adapted from [14])

Property	Value	Description
<b>Position</b>	Absolute   relative   static	Whether the element will accept further positioning properties or position itself as an ordinary HTML element
<b>Left</b>	<length>   <percentage>   auto	Distance from the left margin to the element
<b>Top</b>	<length>   <percentage>   auto	Distance from the top margin to the element
<b>Width</b>	<length>   <percentage>   auto	The width of the element
<b>Height</b>	<length>   <percentage>   auto	The height of the element
<b>Clip</b>	<shape>   auto	Show the bounding box that defines the element.
<b>Overflow</b>	Auto   scroll   visible   hidden	What to do with any part of the element that does not fit into the specified height and width
<b>Visibility</b>	Inherit   visible   hidden	Whether the element will be shown at load time
<b>z-index</b>	Auto   <integer>	The element's position in the stacking order

Netscape and Microsoft are adopting different versions of some important technologies. For example Netscape developed layers in response to the need for dynamic positioning of elements. With style sheet positioning, any block elements and contents, such as <DIV>, can be positioned on the browser window. On the other hand, Layers in Netscape require the special <LAYER> element. The user can specify the position and content of a layer of HTML in the body of the page. It can be defined using the <LAYER> tag. The style for a positioned block of HTML content always includes

the property "position." But Microsoft's Internet Explorer exposes all HTML elements including the <DIV> tag in DOM [14].

The position value of CSSP can be either absolute, which indicates a layer with an absolute position in its containing layer, or relative, which indicates a layer with a position relative to the current position in the document. It is possible to specify the top and left properties to indicate the horizontal indentation from the containing layers for an absolutely positioned layer, or the current position in the document for a relatively positioned layer.

If a document contains one or more layers with absolute positions, these layers are unlikely to share styles, since each one will need its own specific value for top and left to indicate its position. The use of individual named styles can be valuable for defining layers, since a named style for each layer (a named style is the same as a style with a unique ID) can be defined. An example is shown in Figure 7.

```

<STYLE TYPE="text/css">
<!--
#page1 {position:absolute;
top:20px; left:5px;
visibility: show;
width:200px;
}
#page2 {position:absolute;
top:20px; left:5px;
visibility: hide;
width:200px;
}
-->
</STYLE>

```

Figure 7. Example of CSS code.

The example code shows only "page1" on the screen that is positioned 20 pixels from the top of the page and 5 pixels from the left on the screen and the user can not see page2.

## 3.2 A Scripting Language

A scripting language such as JavaScript or VBScript gives web pages the power to follow and react to mouse clicks and movements anywhere in web documents. There are two scripting languages (JavaScript and VBScript) widely in use. Core JavaScript is described below.

### 3.2.1 Core JavaScript

JavaScript is an object-oriented scripting language for client and server applications. JavaScript lets Web developers create applications that run over the Internet. Client applications run in a browser, such as Netscape Navigator, or Internet Explorer. Server applications run on a server. Core JavaScript encompasses all of the statements, operators, objects, and functions that make up the basic JavaScript language. Core JavaScript can be extended for a variety of purposes by supplementing it with additional objects. Using Core JavaScript, a designer can create DHTML pages that process user input and maintain persistent data using special objects, files, and relational databases. Through JavaScript's LiveConnect functionality, applications can access Java and CORBA distributed-object applications [13].

Figure 8 shows that Server-side and client-side JavaScript share the same core language. The core language contains a set of core objects, such as the array object. It also defines other language features such as its expressions, statements, and operators. Server-side and client-side JavaScript use the same core functionality.

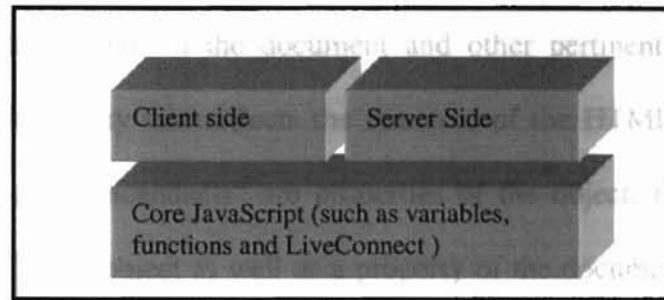


Figure 8. Core JavaScript.

Client-side JavaScript extends the core language by supplying objects to control a browser (Navigator or another web browser) and its Document Object Model (DOM). Client-side extensions allow an application to place elements on an HTML form and respond to user events and page navigation and then often do their job faster than a Java applet equivalent [13].

Server-side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server. Server-side extensions allow an application to communicate with a relational database, provide continuity of information from one invocation to another of the application, or perform file manipulations on a server [13].

JavaScript 1.2 introduced a powerful event model into the browser, permitting JavaScript programs in the browser to control a large amount of the user interface, to take over the whole screen, mimic full-scale applications, and interact more closely with the user.

More facilities in JavaScript 1.2 make JavaScript even more powerful as both a general-purpose and a web-oriented scripting language. JavaScript allows dynamic web Sites and DHTML.

Figure 9 describes JavaScript objects in Navigator and how to use them. These client-side JavaScript objects are sometimes referred to as Navigator objects. When documents

are loaded in the Navigator, it creates a number of JavaScript objects with property values based on the HTML in the document and other pertinent information. These objects exist in a hierarchy that reflects the structure of the HTML page itself. In this hierarchy, an object's "descendants" are properties of the object. For example, a form named "showpage" is an object as well as a property of the document, and is referred to as "document.showpage [14]."

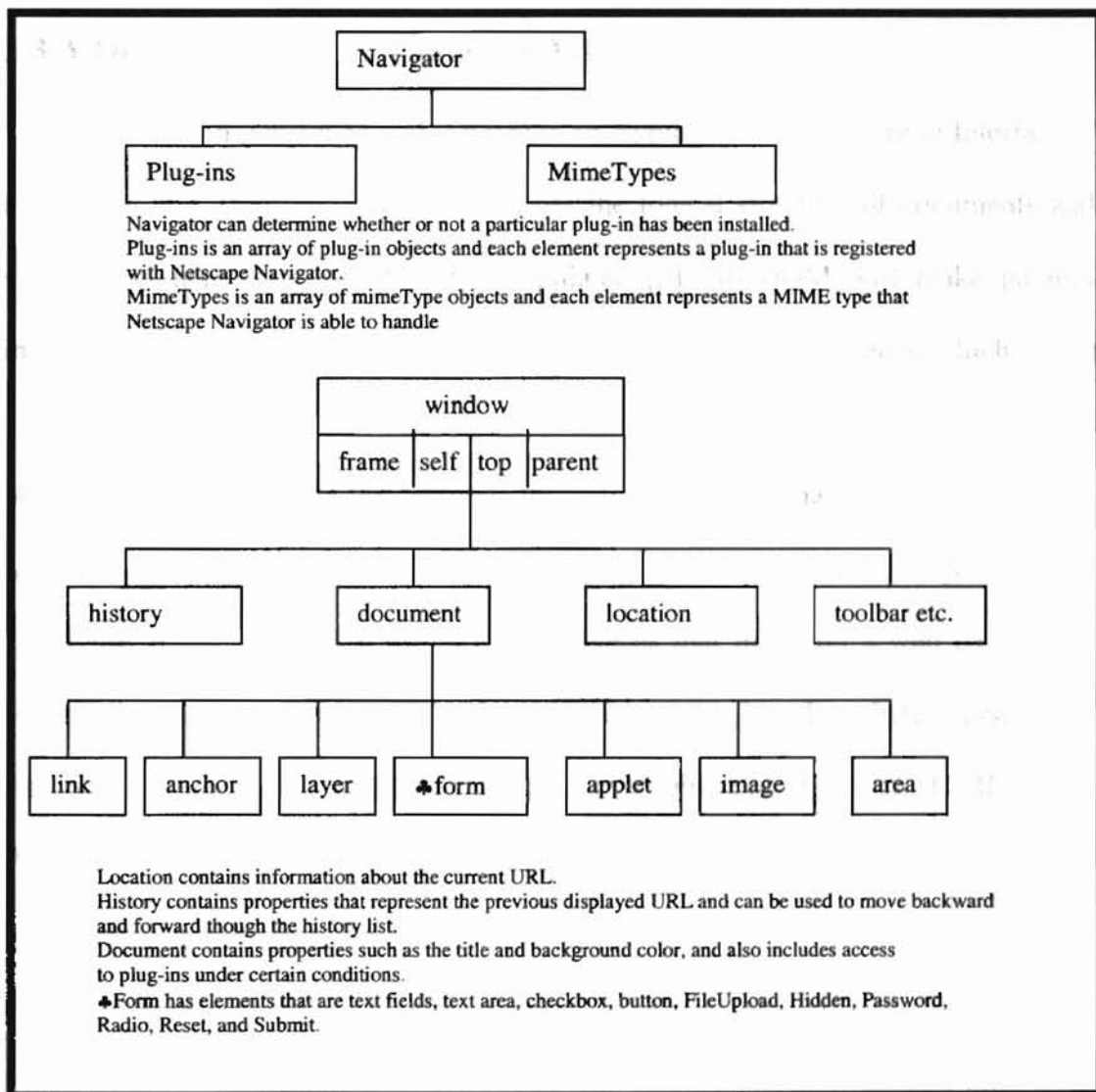


Figure 9. JavaScript DOM (adapted from [35]).

### 3.2.2 VBScript

Microsoft Visual Basic Scripting Edition belongs to the Visual Basic family of programming languages. It brings active scripting to a wide variety of environments, including web client scripting in Microsoft Internet Explorer and web server scripting in Microsoft Internet Information Server [26]. VBScript is a fast, portable, lightweight interpreter for use in World Wide Web browsers and other applications that use Microsoft ActiveX Controls, automation servers, and Java applets.

### 3.3 A Document Object Model (DOM)

The Document Object Model (DOM) is an Application Programming Interface (API) for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated [3]. So DOM will make paragraphs, images, and every single character in a document. DOM will determine which properties can be accessed or changed and which events can be intercepted for that object. The object model also defined the methods that can be used to change or access the object. Any element that is delimited by a tag an HTML page is considered an object.

Figure 10 illustrates that the individual pieces that make up a web page are small objects. A model is the connection of those objects to make other objects, and ultimately the document itself [3]. The DOM can access every piece of objects in HTML document. When users create web page using a scripting language, such as JavaScript, to change form elements or images, the JavaScript Object Model assigns the object status to only a small subset of web document. For example, the text inside <H1> or <P> tags is not an object.

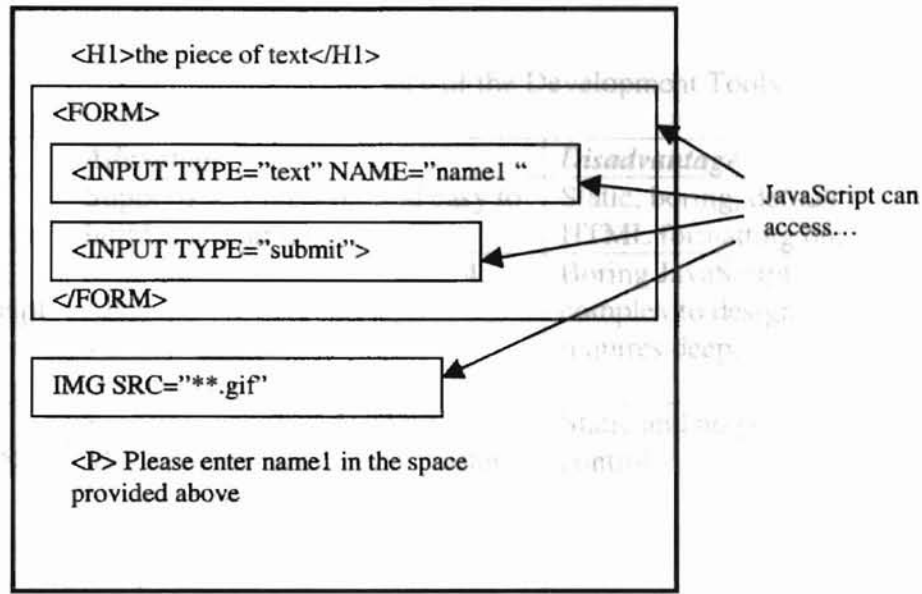


Figure 10. JavaScript Object Model accesses a small subset of web documents (adapted from [3]).

### 3.4 Global View of Development Tools

HTML was intended to be a simple, rather limited language for describing primitive information layouts in World Wide Web pages. The first version of HTML specification emphasized simplicity. The browser controlled the actual page appearance, determining the appearance of headers, paragraphs, and other primitive layout elements. In recent years, however, complex features have been added to CSS, CSSP, and JavaScript have been added into HTML. These extensions have served designers well, giving them more control over the appearance of their pages. Table II shows a global view of development tools. N represents the Netscape Navigator and IE represents the Internet Explorer in the Table II

Table II: Summary of the Development Tools

	<i>Advantage</i>	<i>Disadvantage</i>
<b>HTML Only</b>	Supports any browser and easy to build and maintain.	Static, boring, default HTML formatting only
<b>HTML, JavaScript</b>	Runs on N 2 / 3/ 4 and IE 3 / 4. Adds intelligence to pages Version and vendor issues minimized	Boring JavaScript, complex to design and requires deep.
<b>HTML and CSS1</b>	Separate format and content. Make maintenance easier, faster. Maintain consistency across organization. Simplify exchange of information.	Static and no positioning control
<b>HTML, CSS1 and CSSP</b>	Positioning control	Static and tricky to degrade gracefully.
<b>HTML, JavaScript and style sheet</b>	1.Access CSSP and CSS1 properties from JavaScript. 2.Enable JavaScript scripts to control page format. 3. Enable JavaScript scripts to control page format Enable dynamic control of elements which are given a unique ID and declared to be position	



## **Chapter 4 Deign and Implementation of DPE**

### **4.1 Dynamic Page Editor (DPE)**

Dynamic HTML allows a web page to change after it is loaded into the browser. It is not necessary to communicate with the web server for an update. Implementation of DPE makes use DHTML. DHTML presents richly formatted pages and lets users interact with the content on those pages without having to download additional informative contents from the server. This means that a page can respond immediately to user actions, such as a mouse click, without having to retrieve an entire new page from the server.

If the content on a page is to be changed via CGI, then this requires a server to perform the changes to the page and re-serve the entire page, back to the client. But the processing time of CGI is quite slow, for it places a burden on both network traffic and server processing time. With long delays between a user's action and an on-screen response, building effective Web-based applications is quite constricting using CGI. This suggests the use of the DHTML method to change the web page, because DHTML works on the client-side so it can avoid the disadvantages of CGI. This means that page modifications can appear immediately following a trigger, such as a user selection. This is based on the DHTML objective that any aspect of the currently loaded page-text styles, swapped images, context-sensitive forms and tables, and even the on-screen data itself may be modified [32]. Therefore, DHTML is used in this thesis.

CSS and CSSP are static and hard-coded, so it is not possible to change a page dynamically. But when using JavaScript, users can change the layout of the browsing

page dynamically, and can modify the page in a variety of ways after being opened. Users also can generate the page that they want to print, and can make content vanish or appear, even changing the color of individual parts of the page if desired.

DHTML creates new HTML tags and optional attributes, as well as style sheets and programming options that encapsulate the script in reusable components. This lets developers design web pages that are much more animated and interactive than pages built with standard HTML. With HTML 3.2, dynamism is limited to embedding applets in a page. DHTML brings Web pages alive with a user interface, elements that change color when the cursor passes over them, and images that users can drag-and-drop from one location to another [32]. In addition, DHTML permits the precise positioning of elements and style sheets that make formatting of large documents easier. In other words, DHTML would radically change pages and the way designers work with them [20].

Microsoft and Netscape agree on several important aspects of DHTML [27]. Designers can view elements of each page as an object they can name, give attributes to, and address in a script that is part of the page. According to using CSS, CSS syntax defines items whose position, style, and appearance can change after the document is loaded. CSS lets a designer separate page content and structure from its presentation [27]. Also DHTML pages can be built to pre-load sets of images on multiple layers. Mouse movements can then hide or reveal various layers.

Figure 11 shows that DPE using DHTML can access every piece of HTML documents. Every tag, image, and each single piece of text would be recognized as an object [3].

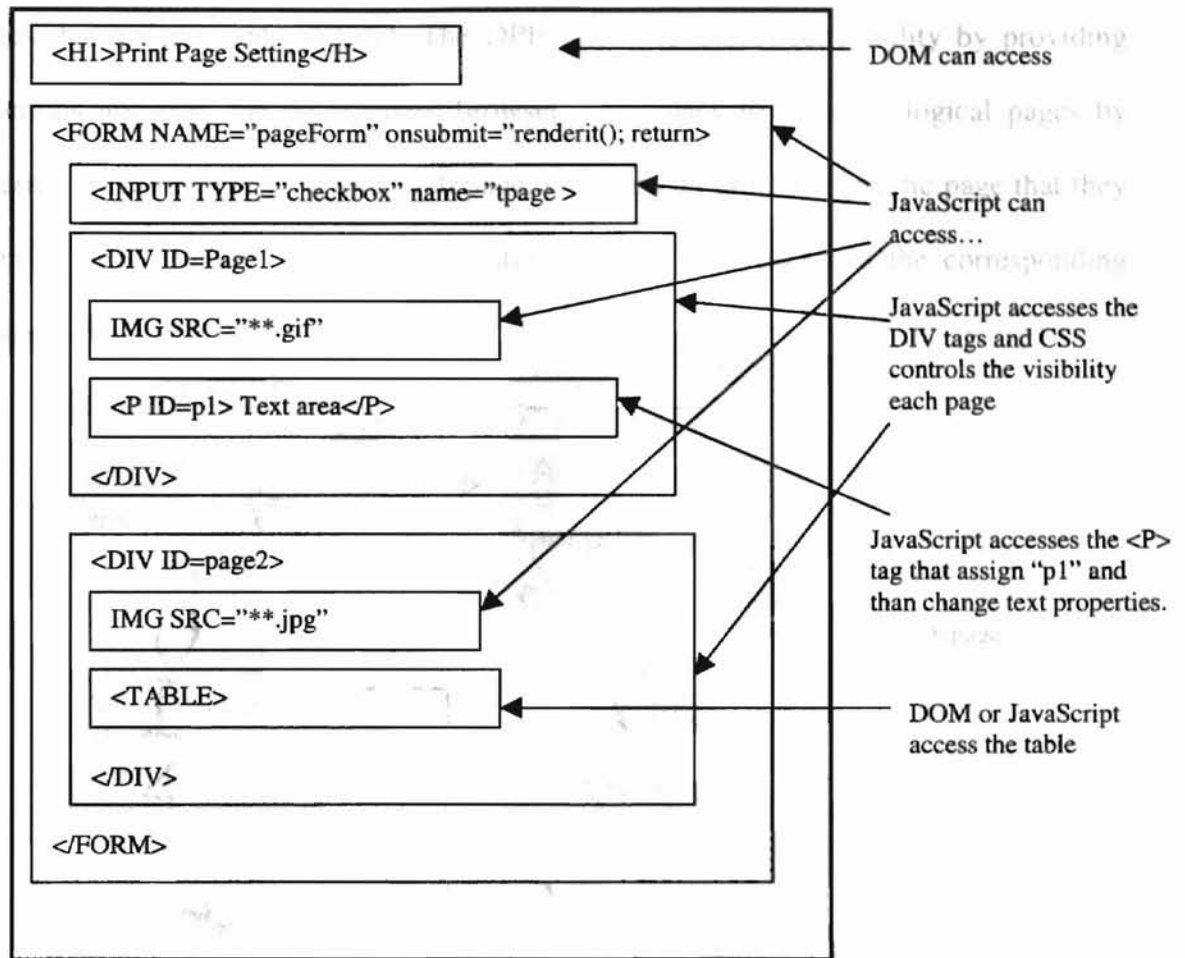


Figure 11. DHTML accesses all objects  
 □ represents an object (adapted from [3]).

With the new object model, JavaScript can access those <DIV> tags shown in figure 11 in response to mouse movement or click. Finally, the style sheet controls the visibility of each page. According to the unique ID assigned in the <P> tag, JavaScript accesses the text and can change the text inside <P>...</P> tags, even after the document has been displayed.

## 4.2 Event Sequence in DPE

The DPE using CSS, CSSP, and JavaScript together allows users to create web pages that use different styles in different parts of the page as well as show and print selected

pages that a user wants to print. The DPE improves a browser's utility by providing functions not available on standard browsers. It is easy to generate logical pages by inserting page breaks into a long web page, so users can print the specific page that they want to print. Figure 12 shows those steps taken by the user and the corresponding actions of the browser.

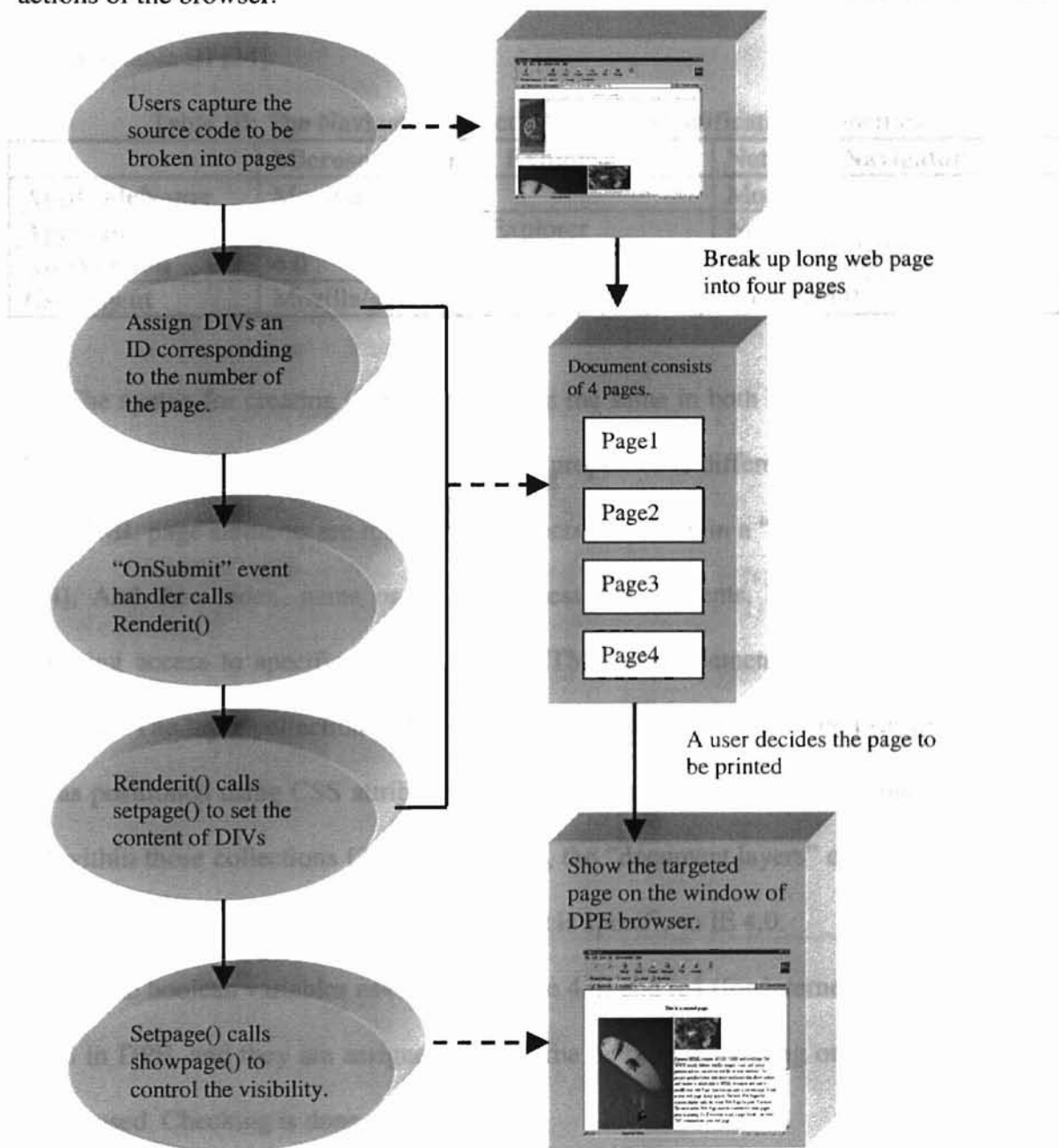


Figure 12. User Action and Browser's Response.

### 4.3 Properties of DPE

DPE uses JavaScript to access and change the properties of the CSSP element.

Also DPE works in both Netscape Communicator and Internet Explorer. However, there are the syntax differences between Netscape 4.0 (N4) and Internet Explorer 4.0 (IE4). The four properties that identify Netscape 4.0 (N4) and Internet Explorer 4.0 (IE4) are given in table III [34].

Table III: The Navigator Object's Browser Identification Properties

	Microsoft Internet Explorer	Netscape Navigator
<b>AppCodeName</b>	Mozilla	Mozilla
<b>AppName</b>	Microsoft Internet Explorer	Netscape
<b>AppVersion</b>	4.0	4.0
<b>UserAgent</b>	Mozilla/4.0	Mozilla/4.0

The syntax for creating CSS documents is the same in both Navigator 4 and Internet Explorer, but the syntax for controlling CSS properties is different. In Internet Explorer, all HTML page elements are reflected as objects contained in a "document.all" collection [24]. And then index, name or ID can access the elements. But in Netscape model, scripting access to specific collections of HTML page elements is supported, such as "layers." The layer collection in Navigator includes areas bounded by a <layer> tag and areas positioned using CSS attributes [34]. Elements can be accessed by index, name or ID within those collections [34]. That means, the "document.layers" object is specific to Netscape 4.0, while the "document.all" object is specific to IE 4.0.

So the boolean variables ns4 (for Netscape 4.0) and ie4 (for Internet Explorer 4.0) are used in DPE, and they are assigned values true or false depending on which browser is being used. Checking is done as follows:

Both `ns4 = (document.layers)? true:false`

Netscape `ie4 = (document.all)? true:false`

The DPE works on version 4 of both Netscape and Internet Explorer, however the DPE must determine which browser is in use. The Figure 11 shows how it checks the browsers.

```
function checkBrowser() {
    if (navigator.appName == 'Microsoft Internet Explorer') {
        IE = true;
        if (navigator.appVersion.indexOf('3.') > -1) {
            ie3 = true;
            alert('Please upgrade your browser to 4.x or greater.');
```

Figure 13. The Checkbrowser() function.

Both Netscape and IE can open the DPE. But there are some differences between Netscape (N4) and Internet Explorer (IE4). For examples, To perform math on the left and top property of the style object, Internet Explorer uses pixel property such as "50px." In Netscape the possible values of the visibility are show, hide and inherit. But the possible values of visibility in Internet Explorer are visible, hidden, and inherit. Also those two browsers use different property names for background color, but use the same values. The table IV shows the differences between Netscape and Internet Explorer.

Table IV: The Differences between Netscape and Internet Explorer.

<b>N4 Property</b>	<b>IE4 Property</b>
left	PixelLeft
top	pixelTop
visibility	visibility
bgcolor	backgroundColor

### 4.3 The Implementation Scheme of DPE using DHTML.

DPE consists of four frames named frame #1, frame #2, frame #3, and frame #4:

1. The first frame (frame #1) on the top left segments is called "editor.html" and is use to set the properties of HTML and mark the page that is targeted by a user (Figure 14, section 1).
2. The second frame (frame #2) on the bottom left segment is called "submit.html" and is to submit the setting to the other two frames ("code.html" and "show.html") (Figure 14, section2).
3. The third frame (frame #3) on the top right segment is called "code.html" and is to show the source code (Figure14, section 3).
4. The last frame (frame #4) on the bottom right segment is called "show.html" and is to display a specific page (Figure 14, section 4).

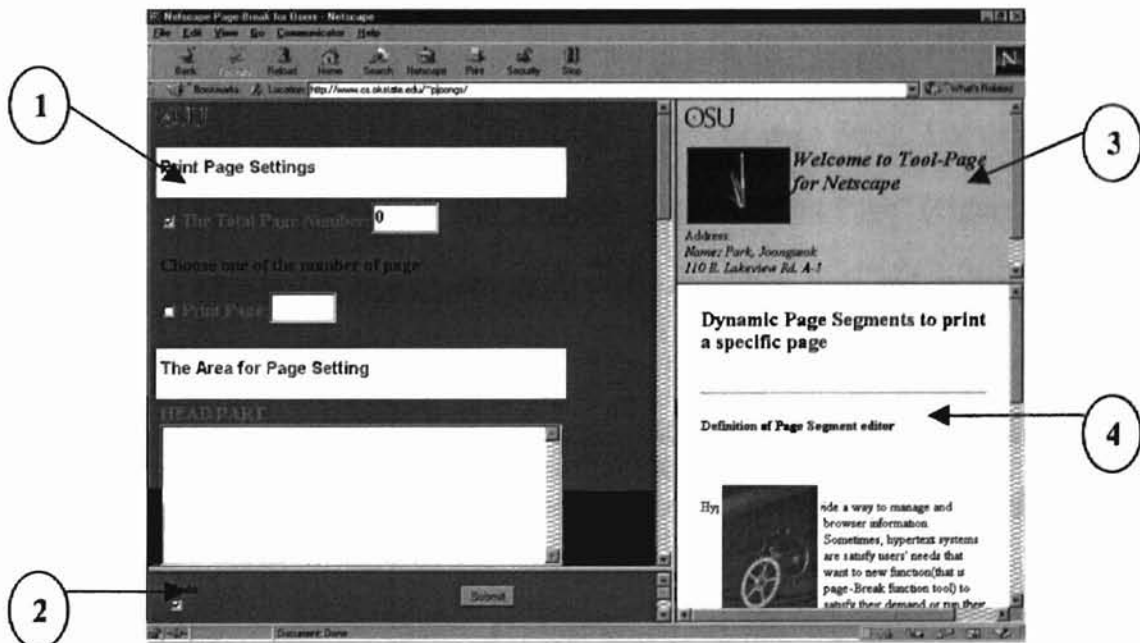


Figure 14. The Initial DPE window.



The following are the capabilities of the editor and how they are used:

First, users can enter HTML code in the text-area that will be displayed in frame #1. They may also capture the source code from outside the page editor and insert it into the Textarea. If a user wants to add a page, then the facilities provided in the editor.html enters a page number that they want to create. DPE will automatically update the total page count and write a correct tag to generate page-breaks. Also if a user enters an existing page number then an error message is displayed. Users can choose the page to print in a printer or to show in frame #4. When all desired properties of the web page are set, they need to click on the submit button in frame #2. Then the page that users want to print will appear in frame #4 and also in a pop-up window.

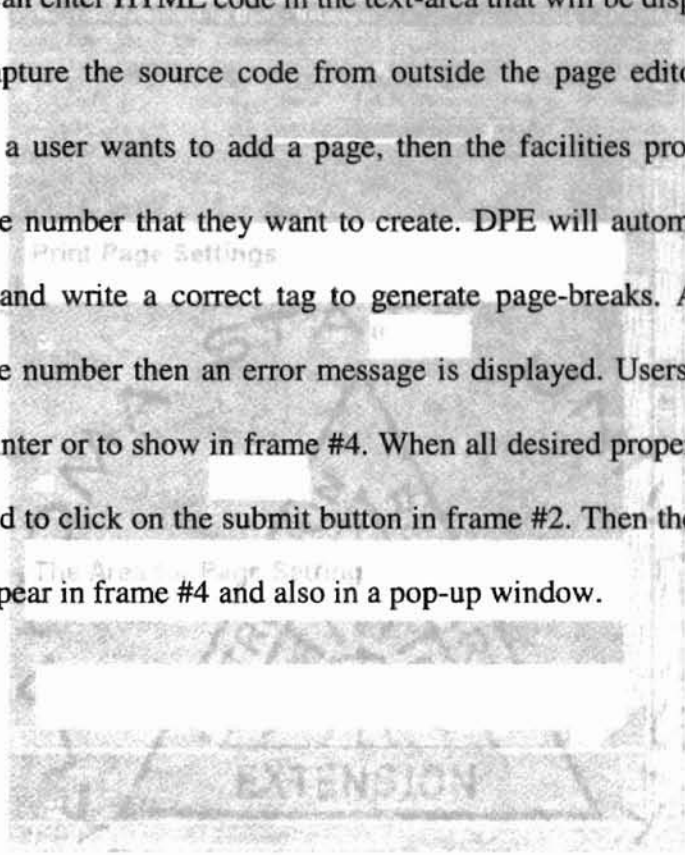


Figure 1. The screenshot of the page editor. The screenshot shows the editor interface with various frames and a dialog box. The dialog box is titled 'Print Page Settings' and contains several options for printing the page. The main editor area shows a text area for entering HTML code and a 'Submit' button. The rendered page is visible in frame #4, showing a section titled 'EXTENSION'.

Figure 2. The screenshot of the page editor showing the rendered page.

Figure 3. The screenshot of the page editor showing the rendered page.

Figure 4. The screenshot of the page editor showing the rendered page.

that is Definition of Print page) in the Navigator status bar. The  
window

There are  
Total Page  
page that  
The  
of  
the

The  
of  
the

The  
of  
the

The  
of  
the

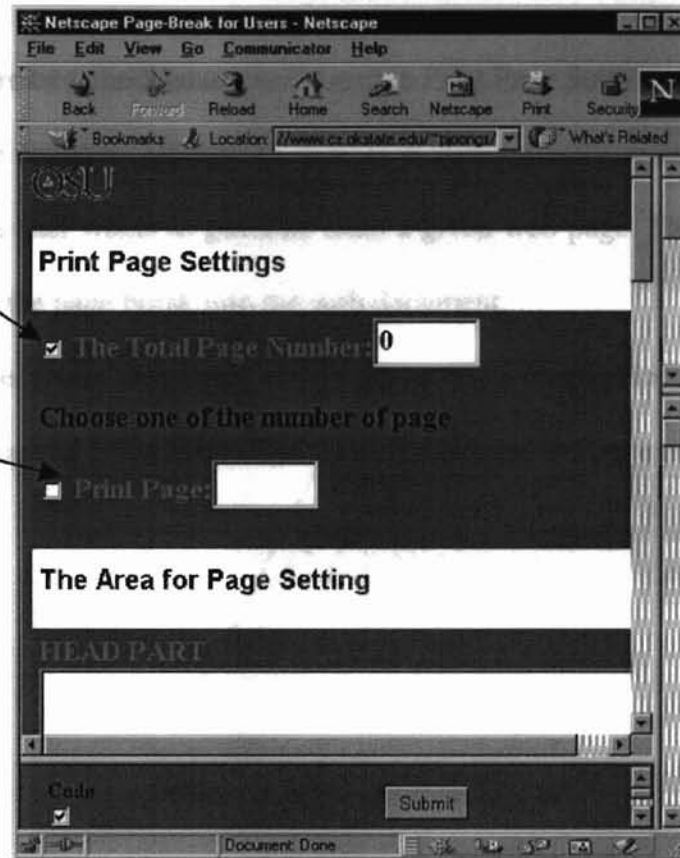


Figure 15. "Page Setting" for page break function.

DPE uses "checkbox" object to set the properties of page break. For example, DPE reads the page number that users want to print when the "Print Page" (Figure 15 box1) is checked by users. The following example shows a checkbox code and creates a hypertext link to an anchor in a frame.

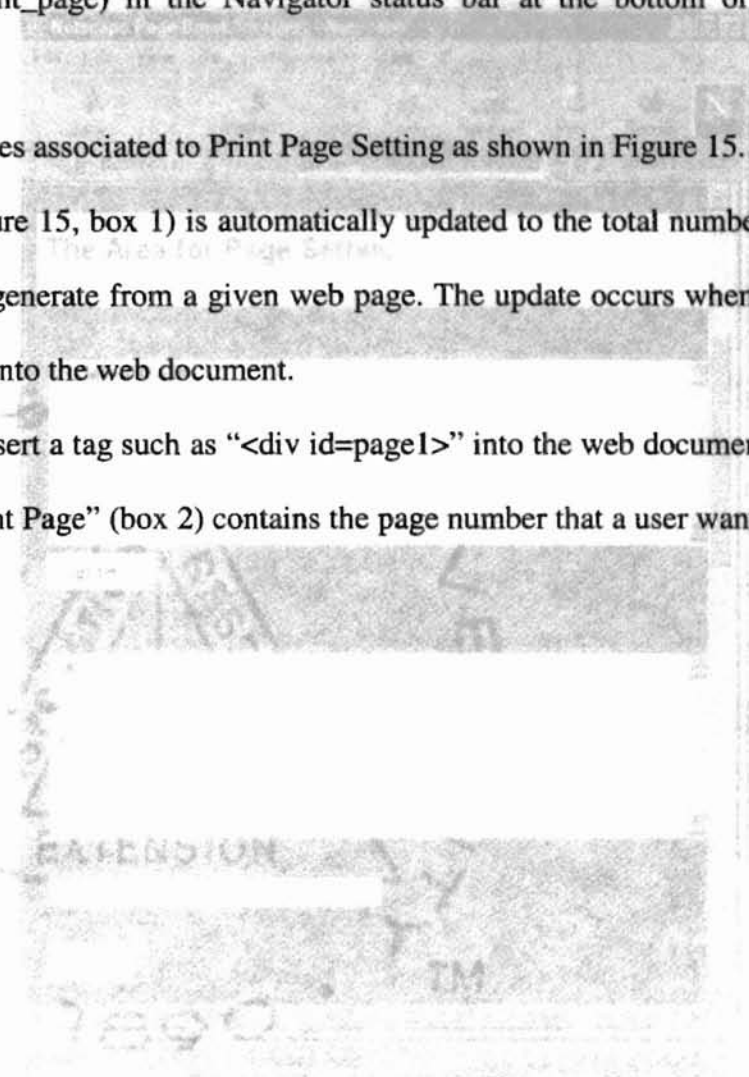
```
<input type="checkbox" name="page-break"> <a href="defs.html#print_page"
target="showFrame" onMouseOver="window.status='Definition of
print_page';return true">Print Page:</a><input type="text" name="PagePrint"
value="" size=3>
```

The link displays the anchor named "print\_page" in the file defs.html in the frame named 'ShowFrame' that must be within the current frames. As a result, the DPE loads pages into showFrame and "status" in the "onMouseOver" event handler displays messages

(that is "Definition of Print\_page) in the Navigator status bar at the bottom of the window.

There are two checkboxes associated to Print Page Setting as shown in Figure 15. The "Total Page Number" (Figure 15, box 1) is automatically updated to the total number of pages that a user wants to generate from a given web page. The update occurs when the user inserts the page break into the web document.

For example, a user may insert a tag such as "<div id=page1>" into the web document to generate "page1." "The Print Page" (box 2) contains the page number that a user wants to print.



Print Page  
Total Page Number: 10  
The Print Page: 10  
Print

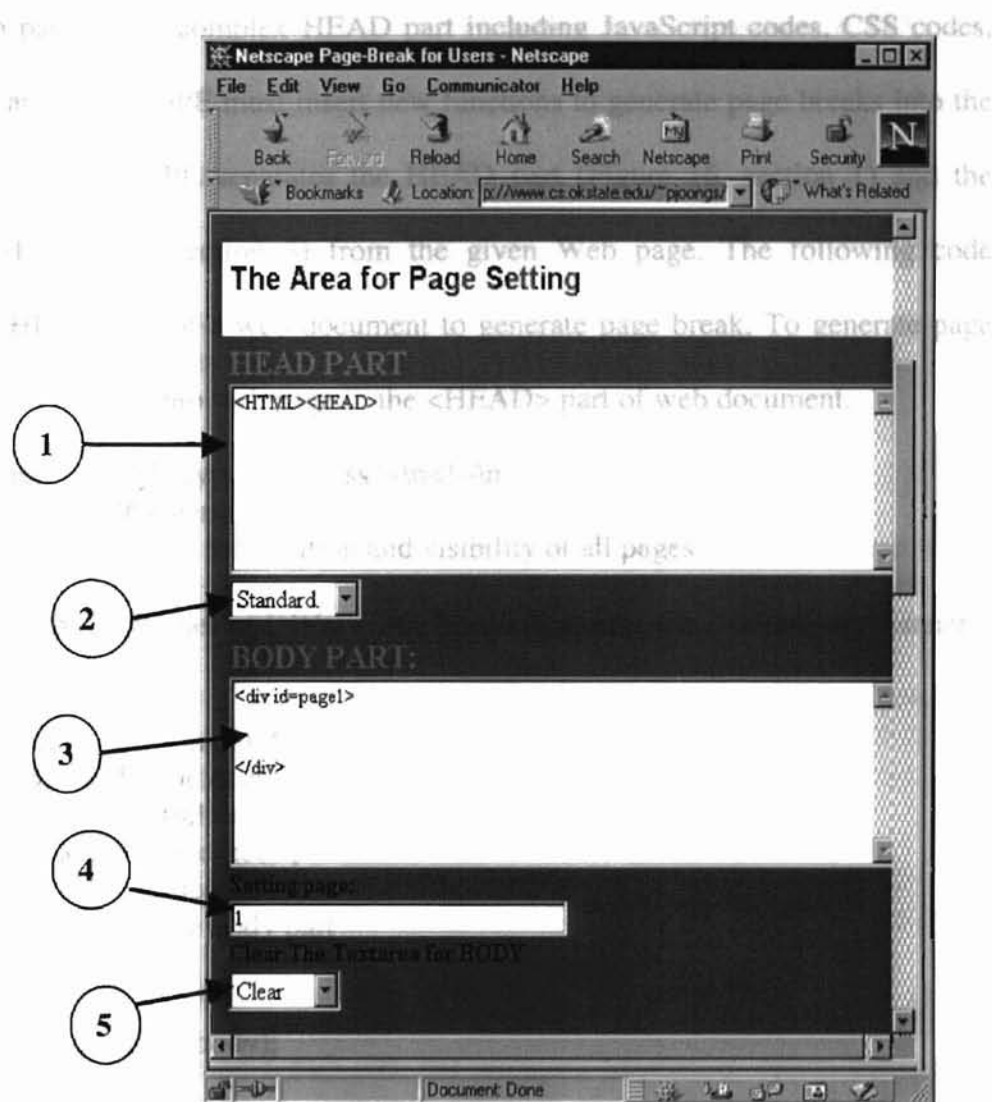


Figure 16. The Areas for page breaks.

The text box consists of two parts. One (Figure 16, section 1) is for the “HEAD” part of the HTML code and the other one (Figure 16, section 3) is to enter the “BODY” part of the HTML code. The first Select box (Figure 16, section 2) has two options. When a user selects the “standard.” The DPE automatically writes the HTML tag and the HEAD tag into the text-area (Figure 16 section 1) as shown. When user selects the “clear” option, the DPE removes all text in the text-area (Figure 16, section 1).

Most web pages have complex HEAD part including JavaScript codes, CSS codes, Applet codes and so on. DPE must insert new functions to generate page breaks into the given HEAD part. So DPE separates the HEAD part (Figure 16, section 1) and the BODY part (Figure 16, section 3) from the given Web page. The following code describes the HEAD part of a web document to generate page break. To generate page break, this code is added into web page in the <HEAD> part of web document.

```

Stylesheet = '<style type="text/css">\n<!--\n';
Stylesheet += "#pagenumber{ ";
// JavaScript controls the position and visibility of all pages
•
// setting the properties of DIVs(width, border, padding, etc..) with same manner
•
•
Stylesheet += '\n-->\n</style>\n';
with (top.showFrame.document) {
    head = '<html>\n\n<head>\n';
    head += 'Stylesheet';
    head += '</head>'
    // describe the body part
    •
    •
    write(head + body);
    close();
}

```

When users assign a page number in the “Setting Page box” (Figure 16, section 4), the DPE automatically writes “page-break” tag into the text-area. To generate page-breaks, DPE needs a page-break tag (such as “<div id=Page-number>”). For example, a user assigns page number in the “Setting Page Box”, and then DPE automatically writes “page-break” tag into text-area (Figure 16, section 3) if the page number is valid.

To better benefit from page break, users may use positional elements (e.g. DIVs and Ids) to generate pages so as to avoid breaking inside a block, table, or a float element. The content between consecutive <DIV>...</DIV> tags is generated as a page. Each web document is divided into pages as follows:

```
<DIV ID ="page-number">
  HTML Content
</DIV>
```

JavaScript script will find all DIVs on its own later when the page is displayed.

The setting parameters in the editor.html (Figure 14 section 1) are submitted to the “renderit()” function when a user clicks “submit” button. Because the “onSubmit” event handler in the “submitFrame (Figure 14 section 2)” calls the “renderit()” function. The following code shows the “onSubmit” event handler in the submitFrame.

```
<form name="submitForm" onSubmit="top.Fame.renderit(); return false">
```

A JavaScript browser will execute the renderit() function to set the properties of the page (border, background-color, padding and so on) if some checkboxes in the editor.html are checked. The renderit() calls the setpage() function to decide the page to be shown and to set the properties of the contents.

```
Function renderit();
//setting the properties of page
if (back-groundCheck.checked) {
  •
}
•
•
if (page-break.checked) {
  •
  setpage(page-number);
  •
}
```

The string argument passed to `setpage()` is only the number of the page. The `setpage()` function evaluates page number and attaches the prefix “page” and assigns to “pshow” which is the page-to-be-shown, because the ID consists of “page” and “number” (e.g. page1). The page currently visible, “cshow”, is hidden by calling its `showpage()` method. The new page “pshow” is made visible. This new visible segment is assigned to “cshow” as shown below:

```
function setpage(page-number){
    pshow = eval(page + page-number)
    cshow.showpage(false);
    pshow.showpage(true);
    cshow = pshow;
}
```

Finally, the script passes one argument to the `showpage()` function to control the visibility.

```
function showIt(set) {
    with(this){
        visibility = (set) ? "show" : "hide"
    }
}
```

The DPE automatically adds the properties of visibility of each page into the `<HEAD>` tag of web document and then merges it with the `<BODY>` part.

It first creates the CSS STYLE rules that define the positioned elements for declared pages that users want to print. Since all pages are defined by positional elements, the first two declarations are for position and visibility, before the user sees them. All pages are initially defined “hide” for visibility and defined “absolute” for position to allow for their overlapping positioning by the “AllHidePage()” function that hides all pages.

## *Chapter 5 Test*

Each web page relies on a scrollbar to allow users to browse through material too large to fit into a frame. It is not possible for users to see a selected portion of a page. Also it is impossible to print only a certain part of the given web page. However some users need to have the flexibility of dividing a web page into meaningful pages and print them.

To solve the above problem, the DPE uses DHTML because DHTML can create new HTML tags and optional attributes to generate page breaks. Also DHTML will create style sheets and programming options that encapsulate script in reusable components. By inserting page breaks into a document, the DPE can keep the same web page, but display only the desired page on the browser window that users want to print out. Also the DPE can allow users to change the properties of frames including the font size, color, and so on.

In this chapter, we demonstrate the functionality of DPE. Figure 17 shows a web page in the Netscape browser without page breaks.



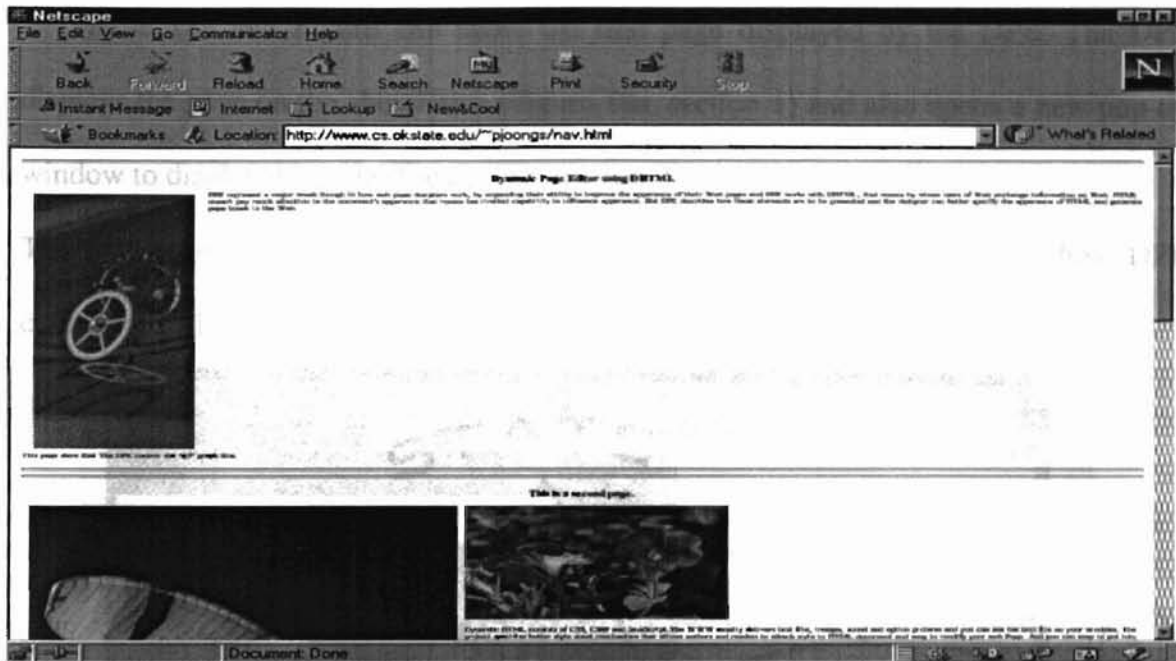


Figure 17. Web Page without page break.

Figures 18-21 demonstrate how DPE transforms a web page into four pages.

Users need to perform the following four steps when using DPE to generate pages:

Step1: Write HTML code for the HEAD part or capture the HEAD part from an existing page and paste it in the Textbox (Figure 16, section 1) for the HEAD part in the DPE window.

Step2: Assign page number to be generated using the “Setting Page” box (Figure 16, section 4).

Step3: Write or capture HTML code for BODY part between consecutive <DIV>...</DIV> tag that is generated as a page.

Step4: If a user wants to add more pages, and then repeats step2 and step3.

Step5: Assign a page number in the “Print Page” box (Figure 16, section 2) the user desires to print out.

Step6: Click on the submit button in the “submit” frame.

Figure 18a and Figure 18b show the first page displayed by the DPE. The DPE displays the page on the showFrame (Figure 18a, section 1) and also opens a new pop-up window to display the page (Figure 18b, section 2).

The first page has one "gif" graph file and one paragraph. This illustrates how DPE controls gif file and text file.

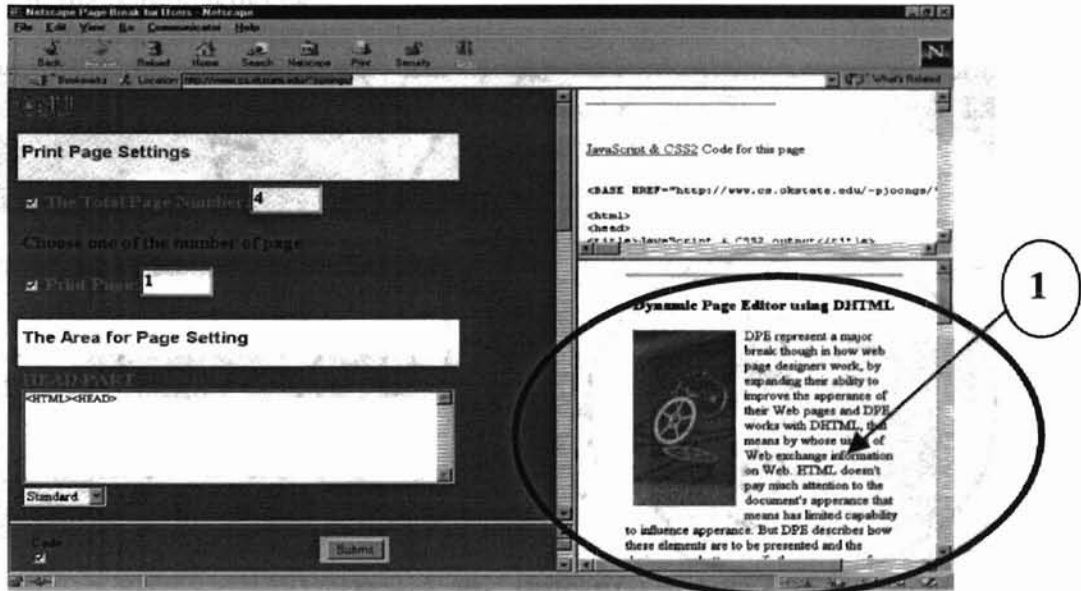
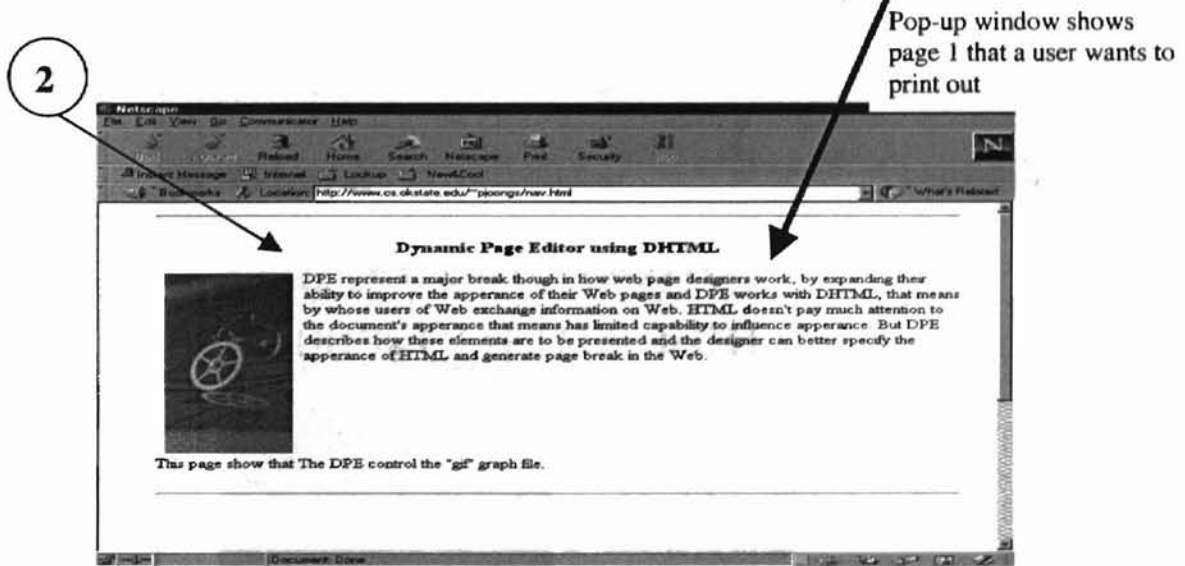


Figure 18a. Display of first page in editor window.



Pop-up window shows page 1 that a user wants to print out

Figure 18b. The pop-up window for the first page.

Figures 19a and 19b illustrate how the DPE displays the second page on the showFrame and pop-up window. The second page consists of two "jpg" files and one paragraph. Figures 19a and 19b verify that DPE can handle "jpg" image files. The page is also display in the pop-up window.

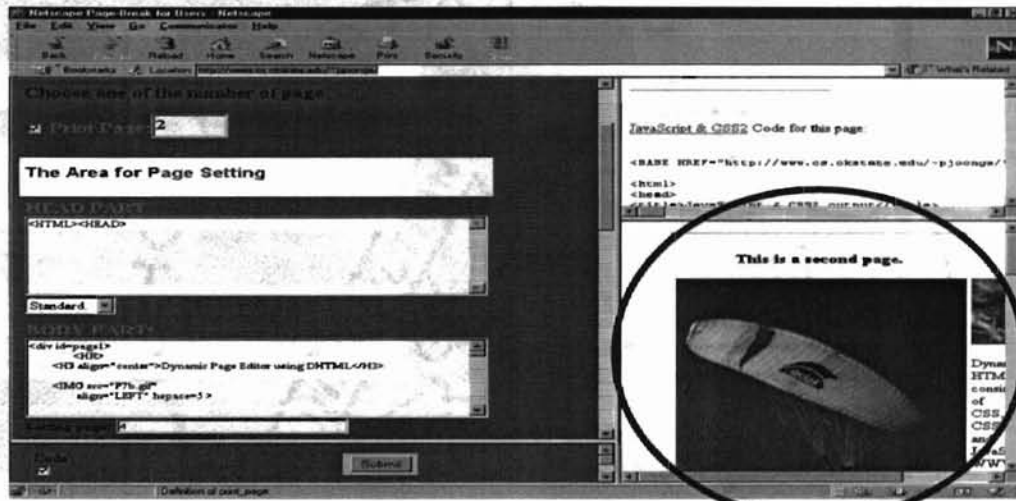


Figure 19a. JPEG files in DPE.

Pop-up window shows page2

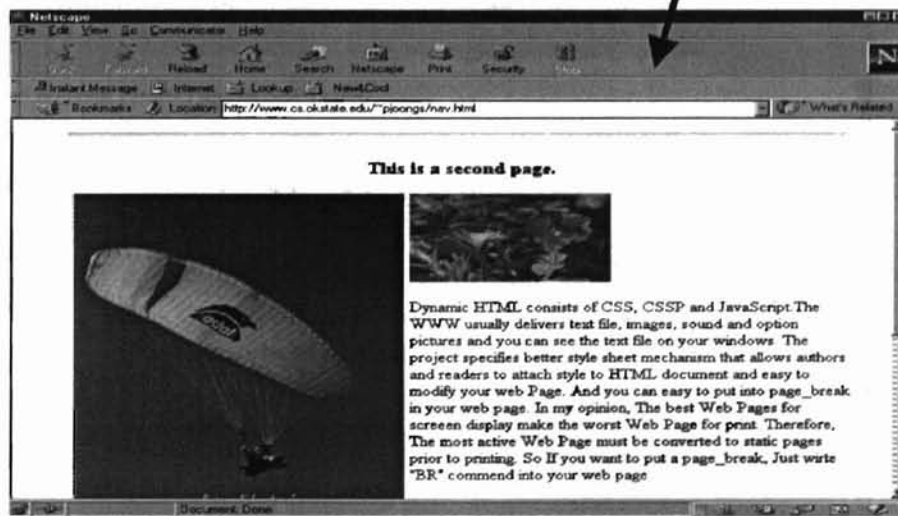


Figure 19b. The pop-up window for the second page.

Figures 20a and 20b illustrate the third page displayed by the DPE on the showFrame and pop-up window (Figure 20b). This page consists of two tables. The Figures 20a and 20b verify that DPE can handle the <TABLE> tags. The tables are also displayed in a pop-up window.

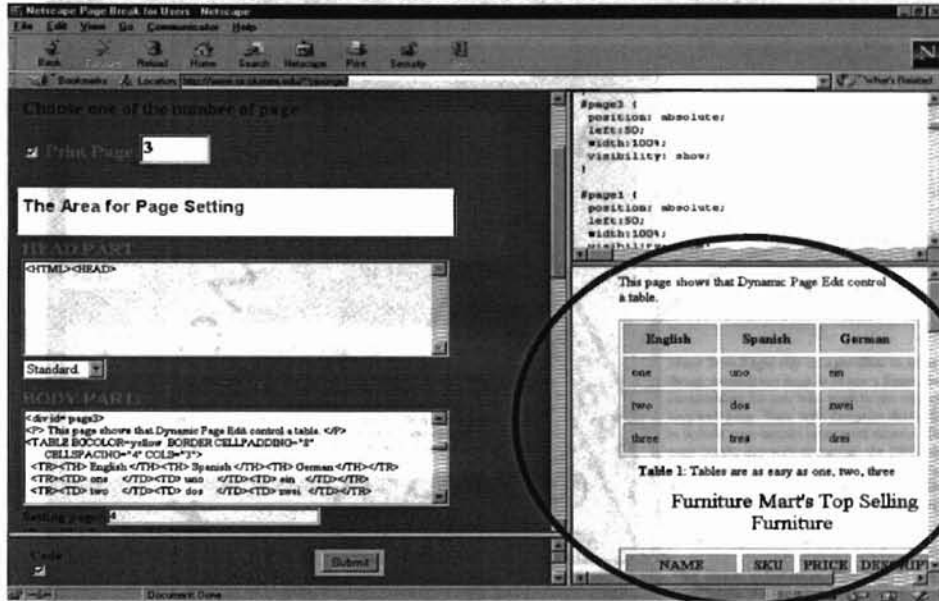


Figure 20a. <TABLE> tags in DPE.

Pop-up window shows the third page that a user wants to print out

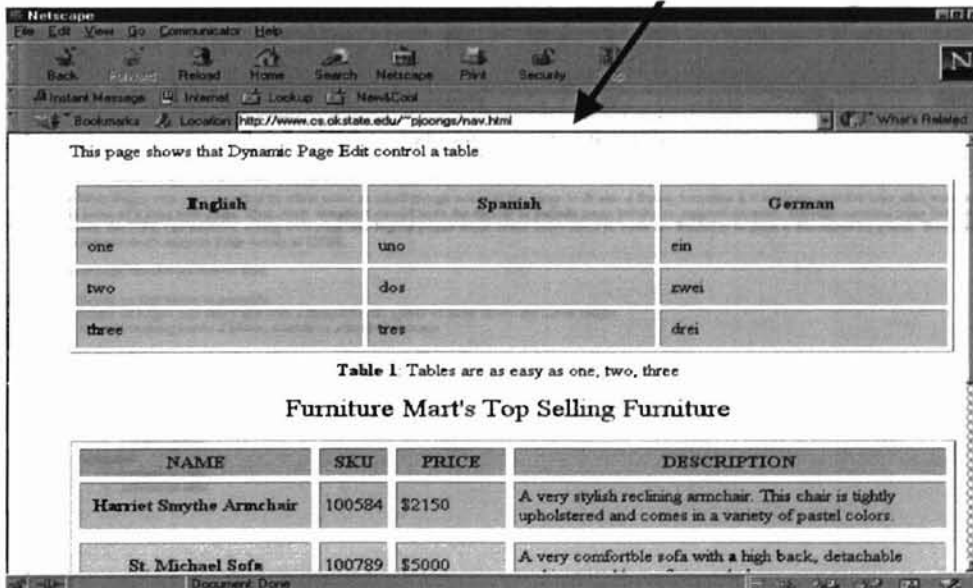


Figure 20b. Table displayed in pop-up window.

Figure 21a and 21b show the fourth page after inserting a page break into the web document.

The DPE also can control standard HTML tag.

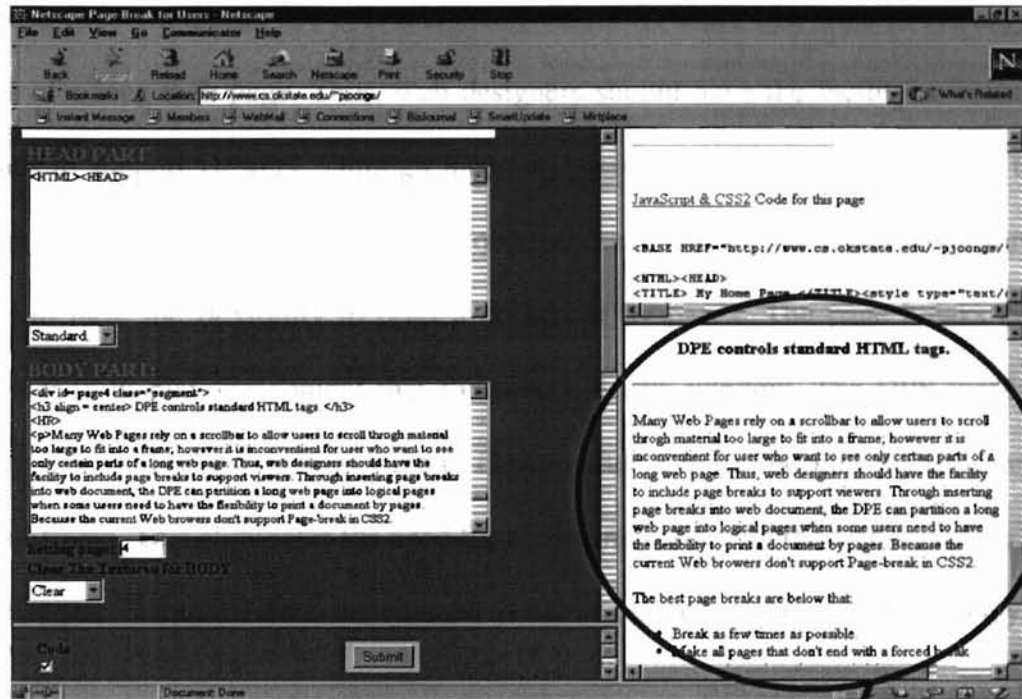


Figure 21a. Standard HTML tags in DPE.



Figure 21b. The pop-up window for the fourth page.

## *Chapter 6 Conclusions*


Many web pages rely on a scrollbar to allow users to scroll through material too large to fit into a frame; however it is inconvenient for users who want to see only certain parts of a long web page. Thus, web designers should have the facility to include page breaks to support viewers. Through inserting page breaks into web document, the DPE (Dynamic Page Editor) can partition a long web page into logical pages when some users need to have the flexibility to print a document by pages. The page breaks cause the browser to display the content that follows on a new window. The DPE works on the client side, so it allows a web page to change after it is loaded into the text-area in the editor frame. This eliminates the need for communication with the web server for updates or changes. Therefore, since the changes are done on the browser side, long delays can be avoided.

User convenience is the greatest asset for marketing good software. However, this thesis is not concerned with marketability, only utility. A menu bar is preferable to check boxes as presented herein. This enhancement DPE is considered further work.

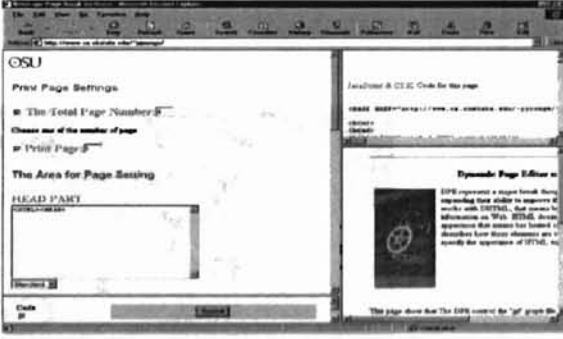
# Appendixes

## A.1 DPE with Internet Explorer

The standard Internet Explorer displays a web documents but the Internet Explorer does not have page breaks so it can't display only the desired page on the browser window that a user wants to print.



A user captures a web document into the DPE. By inserting page breaks into the web document, the DPE can be broken up into smaller pages to display only a desired page that the user want to print.

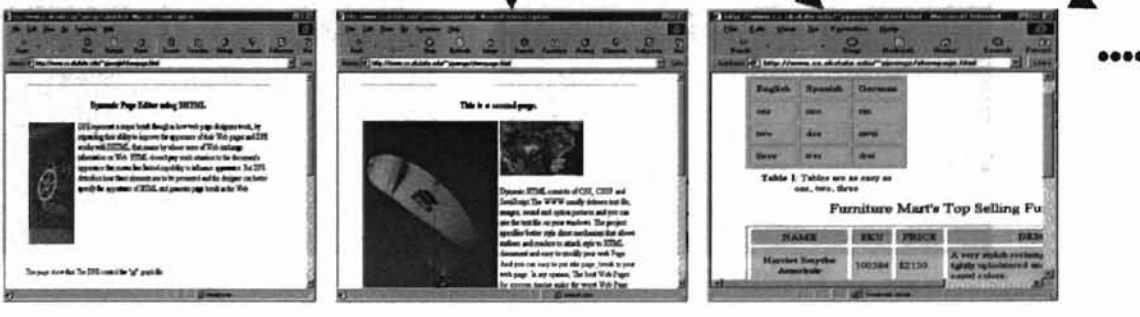


Generate page1 by DPE

Generate page2 by DPE

Generate page3 by DPE

Generate page n by DPE



NAME	SKU	PRICE	DESC
Harvey Specter Amethyst	100284	\$2,150	A very much revving light-up registered motorcycle class.

## A.2 Error Messages

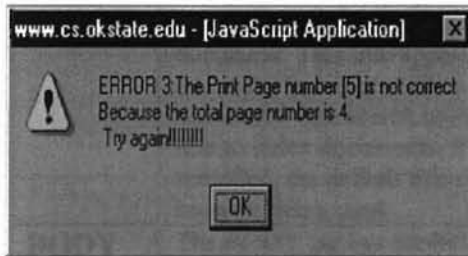
There are four major error messages when a user writes the wrong page number or downloads the DPE in a lower version of the browser.

The DPE works correctly in version 4.0 or greater browser because lower versions of browsers (such as Netscape 3.0 or Internet Explorer 3.0) do not support JavaScript 1.2 and CSS. If a user attempts to use the DPE in the lower version of browser, and then the DPE displays error messages as shown below:

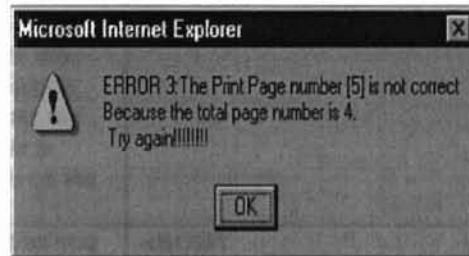
**ERROR 1: Please upgrade to your browser with 4.0 or greater**

**ERROR 2: Your browser does not support JavaScript 1.2 or CSS**

If users set a wrong page number in the checkbox of the “Print Page” then the DPE displays an ERROR 3 as shown below:

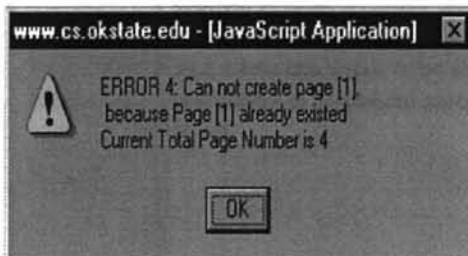


In Netscape browser

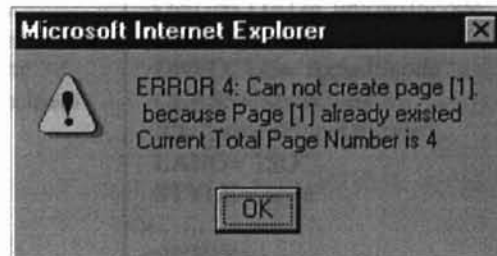


In Internet Explorer

If users set a wrong page number in the “Setting Page” area, then DPE displays ERROR 4 as shown below.



In Netscape browser



In Internet Explorer



### A.3 Basic HTML Tag reference

This table describes the tags that indicate the basic structure of a web page.

	Description	Syntax
<b>HTML</b>	The HTML tag identifies documents as an HTML documents. All documents must start with the <HTML> tag and end with </HTML> tag.	<HTML> ...The head, body, etc. goes here... </HTML>
<b>HEAD</b>	The HEAD tag defines an HTML document header. The HEAD element contains information about the current document, such as its title or keywords that may be useful to search engines and other data that is not considered document content. User agents do not generally render elements that appear in the HEAD as content. They may, however, make information in the HEAD available to users through other mechanisms. Each user should put all header information between the <HEAD> and </HEAD> tags, which should precede the BODY tag.	<HEAD>... </HEAD>
<b>TITLE</b>	The TITLE tag specifies the title of the document. This title appears in the title bar of the browser window. In addition, automated web search tools can use the title to index documents. If no title is specified, the default title depends on the browser being used.	<TITLE>...</TITLE>
<b>BODY</b>	The BODY tag has attributes that let you specify characteristics for the document. You can specify the background color or an image to use as a tiled background for the window in which the document is displayed. You can specify the default text color, active link color, unvisited link color, and visited link color. You can specify actions to occur when the document finishes loading or is unloaded, and when the window in which the document is displayed receives or loses focus.	<BODY BACKGROUND="bgURL" BGCOLOR="color" TEXT="color" LINK="color" ALINK="color" VLINK="color" ONLOAD="loadJScode" ONUNLOAD="unloadJScode" ONBLUR="blurJScode" ONFOCUS="focusJScode" CLASS="styleClass" ID="namedPlaceOrStyle" LANG="ISO" STYLE="style" > ..... </BODY>

## A.4 List of Events

Event	Event Handler	Description
Abort	OnAbort	The loading of an image was aborted
Blur	OnBlur	A form element loses focus or a window or frame loses focus
Change	OnChange	A select, text, or textarea field loses focus and its has been modified
Click	OnClick	An object on a form is clicked
Dblclick	OnDbClick	A mouse double-click was detected
Error	OnError	An error has occurred. Used to determine if an <IMG> loaded properly.
Focus	OnFocus	A window, frame, or frameset receives focus or a form element receives input focus
Mousedown	OnMouseDown	A mouse button was pressed
Mouseover	OnMouseOver	Mouse pointer has entered the area of the elements.
Mousemove	OnMouseMove	The mouse was moved.
Mouseout	OnMouseOut	The cursor leaves an area or link
Load	OnLoad	The browser finishes loading a window.
Move	OnMove	The user or script moves a window or frame.
Submit	OnSubmit	A form is about to be submitted
Keydown	OnKeyDown	The user presses a key
KeyPress	OnKeyPress	The user presses or holds down a key
Reset	Onreset	Form is about to be reset

## A.5 HTML code for page break:

The HTML code captures several web pages from WWW to show that DPE works correctly.

### <div id=page1>

```
<H3 align="center">Dynamic Page Editor using DHTML</H3>
<IMG src="P7b.gif"
  align="LEFT" hspace=5 >
<P>DPE represents a major break though in how web page designers work by expanding their ability to improve the
appearance of their Web pages and DPE works with DHTML, that means by whose users of Web exchange
information on Web. HTML doesn't pay much attention to the document's appearance that means has limited
capability to influence appearance. But DPE describes how these elements are to be presented and the designer can
better specify the appearance of HTML and generate page breaks in the Web.</P>
<BR CLEAR="ALL">
<P>This page show that The DPE control the "gif" graph file.</P>
<HR>
```

### </div>

### <div id=page2>

```
<HR>
<H3 align = center>This is a second page.</H3>
<IMG src="kki.jpg"
  align="LEFT" hspace=5 >
  <IMG src="garden3.jpg" align=middle border=0 width=180 height=100>
<P>Dynamic HTML consists of CSS, CSSP and JavaScript.The WWW usually delivers text files, images, sound and
options pictures and you can see the text file on your windows. The project specifies a better style sheet mechanism
that allows authors and readers to attach style to an HTML document and easy to modify your web Page. And you
can easy to put into page_breaks in your web page. In my opinion, The best Web Pages for screen display make the
worst Web Page for print. Therefore, The most active Web Page must be converted to static pages prior to printing.
So If you want to put a page_breaks, just write "BR" command into your web page.</P>
  <BR CLEAR="ALL">
<P>This page show that the DPE control the "jpg" graph file</P>
<HR>
```

### </div>

### <div id=page3>

```
<P> This page shows that Dynamic Page Edit control a table. </P>
<TABLE BGCOLOR=yellow BORDER CELLPADDING="8"
  CELSPACING="4" COLS="3">
  <TR><TH> English </TH><TH> Spanish </TH><TH> German </TH></TR>
  <TR><TD> one </TD><TD> uno </TD><TD> ein </TD></TR>
  <TR><TD> two </TD><TD> dos </TD><TD> zwei </TD></TR>
  <TR><TD> three </TD><TD> tres </TD><TD> drei </TD></TR>
<CAPTION ALIGN="BOTTOM"> <B>Table 1</B>: Tables are as easy as one, two, three
</CAPTION>
</TABLE>
<TABLE CELLPADDING=3 CELSPACING=6 BORDER=2>
<CAPTION ALIGN=TOP>
  <BIG><BIG>Furniture Mart's Top Selling Furniture
  </BIG></BIG>
</CAPTION>
<!-- heading row -->
<TR BGCOLOR=#CCCCFF>
  <TH>NAME</TH>
  <TH>SKU</TH>
  <TH>PRICE</TH>
  <TH>DESCRIPTION</TH>
</TR>
```

```

<!-- end of heading row -->
<!-- furniture rows -->
<TR BGCOLOR=#DDEEAA>
  <TH>Harriet Smythe Armchair</TH>
  <TD>100584</TD><TD>$2150</TD>
  <TD>A very stylish reclining armchair.
  This chair is tightly upholstered
  and comes in a variety of pastel colors.</TD>
</TR>
<TR BGCOLOR=#CCFFCC>
  <TH>St. Michael Sofa</TH>
  <TD>100789</TD><TD>$5000</TD>
  <TD>A very comfortable sofa with a high back,
  detachable cushions, and loose fitting upholstery.</TD>
</TR>
<TR BGCOLOR=#BBEECC>
  <TH>Antique Finish Bookshelf</TH>
  <TD>499216</TD><TD>$1979</TD>
  <TD>This stained pine bookshelf is 5 feet tall and 3 feet wide.
  It has been stained by skilled craftsmen and looks
  just like an antique french book case.</TD>
</TR>
<TR BGCOLOR=#AAEEFF>
  <TH>Cosmic Walnut Table</TH>
  <TD>600357</TD><TD>$966</TD>
  <TD>This four foot diameter walnut table has a
  finely polished finish, and will be the
  center piece of any room.</TD>
</TR>
<TR BGCOLOR=#BBDDFF>
  <TH>Variety of prints</TH>
  <TD></TD><TD>$100 to $5000</TD>
  <TD>We have an extensive selection of prints, ranging from
  18th Century watercolors to modern contemporary prints</TD>
</TR>
<TR BGCOLOR=#CYAN>
  <TH ALIGN=CENTER VALIGN=MIDDLE ROWSPAN=2>
  <FONT SIZE=+3>JULY SALE!!</FONT></TH>
  <TD ROWSPAN=2 COLSPAN=3 ALIGN=CENTER>
  <FONT SIZE=+1>
  Don't miss our annual July sale.
  All these prices will be slashed by 50%!!!
  But on Aug 1, they go back up, so don't be late!!
  </FONT>
  </TD>
</TR>
</TABLE>
</div>

```

### <div id= page4 >

<H3 align = center> DPE controls standard HTML tags. </H3>

<HR>

<P>Many Web Pages rely on a scrollbar to allow users to scroll through material too large to fit into a frame; however it is inconvenient for user who want to see only certain parts of a long web page. Thus, web designers should have the facility to include page breaks to support viewers. Through inserting page breaks into web document, the DPE can partition a long web page into logical pages when some users need to have the flexibility to print a document by pages.

Because the current Web browsers don't support Page-break in CSS2.

</P>

The best page breaks are below that:

<UL>

<LI>Break as few times as possible.

<LI>Make all pages that don't end with a forced break appear to have about the same height.

<LI>Avoid breaking inside a block, a table or a floated element.

</UL>

<P>You can find all the latest news from Netscape at

<A HREF="http://home.netscape.com">Netscape's Home Page</A>

<P>The directory structure is:</P>

```
<DIR>  
<LI>composer  
<DIR>  
<LI>editing.htm  
<LI>publishing.htm  
</DIR>  
<LI>navigator  
<DIR>  
<LI>userguide.htm  
<LI>javascript.htm  
</DIR>  
</DIR>  
</div>
```

## *References*

- [1] Eric Krock "The Universal HTML Presentation Template"  
<http://developer.netscape.com/tech/dynhtml/index.html>
- [2] Bert Bos, Hakon Wium Lie, Lan Jacobs "Cascading Style Sheet level 2"  
<http://www.w3.org/TR/REC-CSS2/>
- [3] Robert Mudry "The DHTML companion"  
Upper Sanddle River, N.J. : Prentice Hall PTR, c1998.
- [4] Jason Cranford Teague "DHTML for the world Wide Web"  
Berkeley, Calif. : Peachpit Press, c1998
- [5] Bert Bos, D. Raggett, H. Lie "HTML3 and Style Sheets," W3C Working Draft,  
<http://www.w3.org/pub/WWW/TR/WD-style>
- [6] Alan Richmond, "The Common Gateway Interface for Server-Side Preceding"  
<http://wdvl.com/Authoring/CGI>
- [7] Netscape Communications Netscape Server API  
<http://home.mcom.com/newsref/std/server-api.html>
- [8] Netscape Communications, "Persistent Client State: HTTP Cookies"  
[http://www.netscape.com/newsref/std/cookies\\_spec.html](http://www.netscape.com/newsref/std/cookies_spec.html)
- [9] D. W. Connolly and H. F. Nielsen, "HyperText Transfer Protocol"  
<http://wdvl.com/Internet/Protocols/HTTP/Properties.html>
- [11] Hakon Wium Lie and Bert Bos "Cascading Style Sheets"  
<http://www.awl.com/cseng/titles/0-201-41998-X/liebos/ch1/ch1.html>
- [12] H. Lie, Bert. Bos, "Cascading Style Sheets, level 1," W3C Proposed Recommendation,  
<http://www.w3.org/pub/WWW/TR/PR-CSS1>
- [13] Client-Side JavaScript Guide  
<http://developer.netscape.com/docs/manuals/js/client/jsguide/index.htm>
- [14] Eric Krock, "Positioning HTML Elements with CSS"  
[http://developer.netscape.com/docs/presentations/dynhtml/cssp\\_pres/index.htm](http://developer.netscape.com/docs/presentations/dynhtml/cssp_pres/index.htm)

- [15] Hakon Wium Lie, Robert Stevahn, "CSS Print Extensions"  
<http://www.w3.org/pub/WWW/TR/WD-print-970626>
- [16] Netscape, "Extensions to HTML 2.0,"  
[http://home.netscape.com/assist/net\\_sites/html\\_extensions.html](http://home.netscape.com/assist/net_sites/html_extensions.html)
- [17] Netscape, "Extensions to HTML 3.0,"  
[http://home.netscape.com/assist/net\\_sites/html\\_extensions\\_3.html](http://home.netscape.com/assist/net_sites/html_extensions_3.html)
- [18] D. Raggett, "HTML 3.2 Reference Specification,"  
<http://www.w3.org/pub/WWW/TR/PR-html32-961105>
- [19] R. Stevahn, "PANOSE: An Ideal Typeface Matching System for the Web," W3C Workshop on High Quality Printing from the Web,  
<http://www.w3.org/pub/WWW/Printing/stevahn.html>
- [20] Michele Petrovsky "Dynamic HTML in action"  
Berkeley, Calif. : Osborne McGraw-Hill, 1998.
- [21] D.W. Connolly, "HTML Dialects: Internet Media and SGML Document Types", W3C Working Draft,  
<http://www.w3.org/pub/WWW/TR/WD-doctypes>
- [22] P. Duval, "SGML-based dialects for WEB applications based upon content-understanding," International Workshop on WWW Design Issues '94,  
<http://www.cwi.nl/ERCIM/W4G/WS94/papers/Patrick.html>
- [23] Fabio Vitali "Extending HTML in a Principled Way with Displets"  
<http://atlanta.cs.nchu.edu.tw/www/PAPER155.html>
- [24] S. Butler, R. MacMillan, S. Waters, "HTML Extensions To Improve Layout Control for Viewing and Printing Documents," W3C Workshop on High Quality Printing from the Web.  
<http://www.w3.org/pub/WWW/Printing/fmtext.html>
- [25] Selena Sol "Introduction to Web Design"  
<http://wdvl.com/Authoring/HTML/Tutorial/>
- [26] Microsoft Corporation "JScript & VBScript Tutorials and References"  
<http://msdn.microsoft.com/scripting/default.htm?/scripting/vbscript/>
- [27] Barry Phillips "Designer: the Browser War Casualties"  
Computer Technology News (Oct. 1998), pp 14-16, 21
- [28] C. Musciano and B. Kennedy. "HTML: The Definitive Guide."  
O'Reilly & Assoc., CA, 1996.

- [29] G. K. Raghavan, K. R. Nair, T. B. Horton, "A Domain Model of WWW Browser."  
In Proceedings of IEEE 0-7803-3088-9/96, pp 436-439.
- [30] Dave Raggett "Printing and World Wide Web"  
<http://www.w3.org/Printing>
- [31] Sun microsystems "Hotjava HTML components"  
<http://www.sun.com/software/htmlcomponents>
- [32] Jose A. Ramalho "Advanced HTML 4.0 with DHTML"  
Wordware Publishing, Inc. 1999.
- [33] Louis Rosenfeld and Peter Morville "Information Architecture"  
O'Reilly & Associates, Inc. 1998.
- [34] Aaron Weiss "Introduction to Dynamic HTML"  
<http://wdyl.com/authoring/DHTML/Intro>
- [35] Robert W. Husted "All About JavaScript."  
[http://developer.netscape.com/viewsource/index\\_frame.html?content=husted\\_js/husted\\_js.html](http://developer.netscape.com/viewsource/index_frame.html?content=husted_js/husted_js.html)



VITA

Park, Joongseok

Candidate for Degree of

Master of Science

Thesis: A DYNAMIC PAGE EDITOR USING DHTML

Major Field: Computer Science

Biographical:

Personal Data: Born in Cheongju City, ChungBuk, Korea, On July 3, 1969, the son of Jae-moon Park and Gi-hwa Kim.

Education: Graduated from ChungBuk High school, Cheongju City, Korea in February 1988: Received Bachelor of Science in Computer Science from The Ohio State University, Columbus, Ohio in December 1994. Completed the requirements for the Master Science degree with a major in Computer Science at Oklahoma State University in July 1999.

Experience: Employed as teaching assistant; Oklahoma State University, Department of Computer Science, 1997 to present.