Web Pages that Reproduce Themselves by JavaScript

Yasusi Kanada

Central Research Laboratory, Hitachi Ltd.
Higashi-Koigakubo 1-280, Kokubunji, Tokyo 185, Japan

A JavaScript program in a Web page can clear the page content including the program itself and generate new content. The program can generate exactly the same content including the program itself. This means that a Web page can reproduce itself by JavaScript program that is included in the page. Although exact reproduction is useless, inexact reproduction, which transform part of the content, is usable for more practical purpose. For example, Web pages that change its view from outline mode to detail mode by clicking a button in the page can be implemented using this method. This method can also applicable to other types of documents, such as SGML or XML, if the document may contain self-reproductive program. Another method for reproducing Web pages without reproducing programs is also mentioned. Reproductive Web pages partially but really work on Netscape Navigator.

§1. Introduction

Self-reproduction or self-printing of programs is an old topic in programming. Self-reproductive program outputs the same source program as the original. For example, the following C program, which was programmed by Minoru Uehara, is a self-reproductive program [Ari 94].

```
char c,*x,*y,*z;main(){x="char c,*x,*y,*z;main(){x=";c='"';y="%s%c%s%c;c='%c';y=%c%s%c;\
z=%c%s%c%s";z=";printf(y,x,c,x,c,c,c,y,c,c,z,c,z);}";printf(y,x,c,x,c,c,c,y,c,c,z,c,z);}
```

Self-reproduction of programs is probably useless in practical sense because nothing is changed by the reproduction.

This method can be applied to document reproduction, if a document can contain a program and it can reproduce a document that contains a program. An HTML page can contain a JavaScript program, and this program can clear the page content and generate new content when invoked. Thus, Web pages can be reproduced. The self-reproduction of a Web page is outlined in **Figure 1**. Whether a JavaScript program generated by the program works or not is not specified in JavaScript specification [Net 97a]. However, a self-reproductive page and the generated program partially but really works at least on Netscape Navigator 3.0 and 4.0! The mechanism of exact reproduction is explained more in **Section 2**.

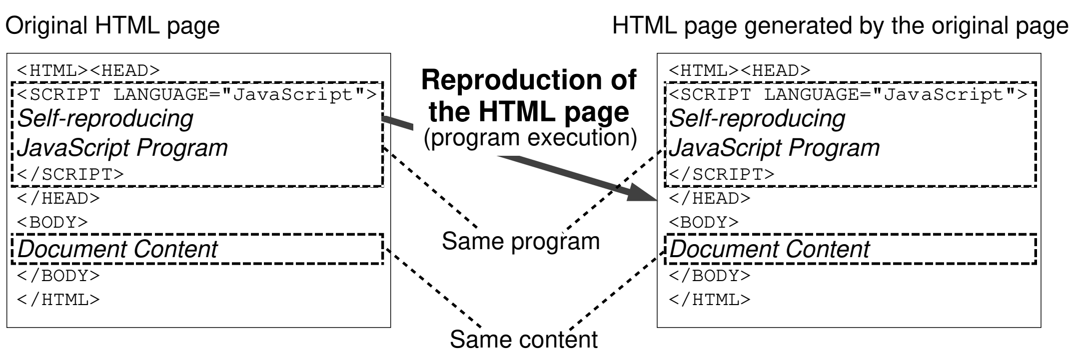

Figure 1. Outline of self-reproduction of Web pages

Exact reproduction of Web pages is probably useless in practical sense too, because of the same reason as the program reproduction. However, *in*exact reproduction may be useful because it can simulate modification or transformation of the document and the transformation may realize useful purposes. For example, inexact reproduction makes changing document view according to user's intention possible. The mechanism of document transformation using self-reproduction is explained in **Section 3**. Several examples, including a practical one, are shown in **Section 4**. The reason why whole reproduction of a Web page, instead of real partial rewriting, is necessary for document modification is that rewriting of original or previously generated text is not allowed in JavaScript. Rewriting is inhibited probably not because of an ad hoc reason, but because it causes serious semantic problem. There are other methods for changing document views, of course. However, the method using self-reproduction has several advantages, which are explained in Section 4.

We can understand a JavaScript program is reproduced and works on Navigators because its behavior is mostly as expected. However, it is difficult to look at the generated program itself using Navigators. A method for viewing generated program or method of debugging self-reproductive programs is explained in **Section 5**. Self-reproduction of programs seems to be tricky. A less tricky method for reproducing documents is overviewed in **Section 6**.

§2. Exact self-reproduction of documents

A most simple example of exact self-reproduction of an HTML page is shown in **Figure 2**. This HTML page contains a button as its only visible content. The appearance of this page by Navigator is shown in **Figure 3**. The JavaScript program in Figure 2 is invoked when the user clicks the button shown in the window in Figure 3. Then exactly the same HTML page is generated by the program and displayed on the window. Because the generated page is exactly the same, the appearance is the same as shown in Figure 3, and the user can repeat the reproduction again and again. This page is available at http://www.st.rim.or.jp/~kanada/reproduction/reproduction-b.html.

```
<html><head><script LANGUAGE="JavaScript">
function reproduce() { var i; var q = "'"; document.clear();
var A = new Array("", "document.writeln(A[i], "+q); var B = new Array("", q+", B[i]);");
var C = "}; document.writeln(C); for (i = 1; i >= 0; i--) {";
var Q = new Array(q, q+"+Q[i]+"+q); var S = new Array("/", q+"+S[i]+"+q);
for (i = 0; i <= 1; i++) {
document.writeln(A[i], '<html><head><script LANGUAGE="JavaScript">', B[i]);
document.writeln(A[i], 'function reproduce() { var i; var q = "'+Q[i]+'"; document.clear();', B[i]);
document.writeln(A[i], 'var A = new Array("", "document.writeln(A[i], "+q); var B = new Array("", q+", B[i]);");', B[i]);
document.writeln(A[i], 'var C = "}; document.writeln(C); for (i = 1; i >= 0; i--) {";', B[i]);
document.writeln(A[i], 'var Q = new Array(q, q+"+Q[i]+"+q); var S = new Array("/", q+"+S[i]+"+q);', B[i]);
document.writeln(A[i], 'for (i = 0; i <= 1; i++) {', B[i]);
}; document.writeln(C); for (i = 1; i >= 0; i--) {
document.writeln(A[i], '}} <'+S[i]+'script><'+S[i]+'head><body>', B[i]);
document.writeln(A[i], '<form><input TYPE="SUBMIT" VALUE="+" onClick="reproduce()"><'+S[i]+'form>', B[i]);
document.writeln(A[i], '<'+S[i]+'body><'+S[i]+'html>', B[i]);
}} </script></head><body>
<form><input TYPE="SUBMIT" VALUE="+" onClick="reproduce()"></form>
</body></html>
```

JavaScript program

Content (a button)

Figure 2. An exact self-reproductive page in HTML

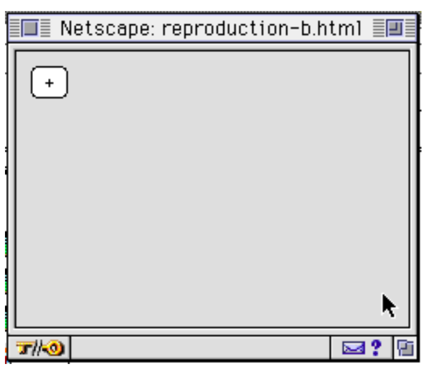

Figure 3. The appearance of the exact self-reproductive page[1]

The self-reproduction of this page really works on Navigator 3.0 or later versions, although not completely. It is not complete because, if the user repeats the reproduction process more than once, uses "go back" menu to back up to a reproduced page and clicks the button, then the button does not work as expected. It does not work because the program has gone! Microsoft Internet Explorer does not seem to reproduce a JavaScript program.

[1] Netscape Navigator on Apple Macintosh is used for showing the appearance of HTML pages thoughout this paper.

The process of self-reproduction is explained below. The same page as shown in Figure 2 is divided into five parts in **Figure 4**. The program in the first part, which is labeled “Part F1,” erases the page content shown in Figure 4 (`document.clear();`), and initializes arrays. Although the source of the running program is also erased, the running program itself, which is probably in compiled pseudo code, is (must be) preserved. The second part, which is labeled “Part F2,” is part of the program, and it runs twice because it is in a for-statement. The same text as Part F1 is generated in the first run, and the same text as Part F2 is generated in the second run. The third part, Part C, runs once, and it generates the same text as Part C. The fourth part, Part L2, again runs twice. The same text as Part L2 is generated in the first run, and the same text as the last part, Part L1, is generated in the second run. Thus, the execution of whole program generates exactly the same text as the original HTML page.

```
<html><head><script LANGUAGE="JavaScript">
function reproduce() { var i; var q = "'"; document.clear();
var A = new Array("", "document.writeln(A[i], "+q); var B = new Array("", q+", B[i]);");      Part F1
var C = "}; document.writeln(C); for (i = 1; i >= 0; i--) {";
var Q = new Array(q, q+"+Q[i]+"+q); var S = new Array("/", q+"+S[i]+"+q);
for (i = 0; i <= 1; i++) {
document.writeln(A[i], '<html><head><script LANGUAGE="JavaScript">', B[i]);
document.writeln(A[i], 'function reproduce() { var i; var q = "'+Q[i]+'"; document.clear();', B[i]);      Part F2
document.writeln(A[i], 'var A = new Array("", "document.writeln(A[i], "+q); var B = new Array("", q+", B[i]);");', B[i]);
document.writeln(A[i], 'var C = "}; document.writeln(C); for (i = 1; i >= 0; i--) {";', B[i]);
document.writeln(A[i], 'var Q = new Array(q, q+"+Q[i]+"+q); var S = new Array("/", q+"+S[i]+"+q);', B[i]);
document.writeln(A[i], 'for (i = 0; i <= 1; i++) {', B[i]);
}; document.writeln(C); for (i = 1; i >= 0; i--) {      Part C
document.writeln(A[i], '}} <'+S[i]+'script><'+S[i]+'head><body>', B[i]);
document.writeln(A[i], '<form><input TYPE="SUBMIT" VALUE="+" onClick="reproduce()"><'+S[i]+'form>', B[i]);      Part L2
document.writeln(A[i], '<'+S[i]+'body><'+S[i]+'html>', B[i]);
}} </script></head><body>
<form><input TYPE="SUBMIT" VALUE="+" onClick="reproduce()"></form>      Part L1
</body></html>
```

Figure 4. Divisions of the exact self-reproduction page

Arrays `A` and `B` are used for writing Part F1 and F2 in generated text differently. Array `C` is used for writing Part C. Arrays `Q` and `S` are used for writing single quote and slash characters. A quote or slash character must be written using a special technique because it requires an escape character when it is in a string.

§3. Partial transformation of documents by inexact self-reproduction

Partial document transformation, i.e., generation of slightly different document page from the original, can be realized using a modified version of the reproductive page shown in Figure 2. A simple example, a page with a two-mode switch in a HTML page, is shown in **Figure 5**. The macroscopic structure of this page content is the same as the exact reproduction page. However, there are several small differences. This page contains a button labeled “+.” If the user clicks this button, a page that contains a button labeled “–” is generated. If the user clicks this new button, a page exactly the same as the original, i.e., the page that contains a button labeled “+,” is generated. The user can repeat switching the button forever. The appearances of these two states are shown in **Figure 6**.

```
<html><head><script LANGUAGE="JavaScript">
function reproduce() { var i; var q = "'"; document.clear();
var A = new Array("", "document.writeln(A[i], "+q); var B = new Array("", q+", B[i]);");
var C = "}; document.writeln(C); for (i = 1; i >= 0; i--) {";
var Q = new Array(q, q+"+Q[i]+"+q); var S = new Array("/", q+"+S[i]+"+q);
var T = new Array("+"=="+"?"-":"+", q+"+T[i]+"+q);
for (i = 0; i <= 1; i++) { (1)
document.writeln(A[i], '<html><head><script LANGUAGE="JavaScript">', B[i]);
document.writeln(A[i], 'function reproduce() { var i; var q = "'+Q[i]+'"; document.clear();', B[i]);
document.writeln(A[i], 'var A = new Array("", "document.writeln(A[i], "+q); var B = new Array("", q+", B[i]);");', B[i]);
document.writeln(A[i], 'var C = "}; document.writeln(C); for (i = 1; i >= 0; i--) {";', B[i]);
document.writeln(A[i], 'var Q = new Array(q, q+"+Q[i]+"+q); var S = new Array("/", q+"+S[i]+"+q);', B[i]);
document.writeln(A[i], 'var T = new Array("'+T[i]+'"=="+"?"-":"+", q+"+T[i]+"+q);', B[i]);
document.writeln(A[i], 'for (i = 0; i <= 1; i++) {', B[i]);      (2)
}; document.writeln(C); for (i = 1; i >= 0; i--) {
document.writeln(A[i], '}} <'+S[i]+'script><'+S[i]+'head><body>', B[i]);
document.writeln(A[i], '<form><input TYPE="SUBMIT" VALUE="'+T[i]+'" onClick="reproduce()"><'+S[i]+'form>', B[i]);
document.writeln(A[i], '<'+S[i]+'body><'+S[i]+'html>', B[i]); (3)
}} </script></head><body>
<form><input TYPE="SUBMIT" VALUE="+" onClick="reproduce()"></form>
</body></html>        (4)
```

Figure 5. An inexact self-reproduction: a page with a switch

Switching is realized by the following mechanism. The first “+” in the expression labeled “(1)” in Figure 5 and “+” in the part labeled “(4)” are the switch value, and are changed to “–” in the reproduced page. The value is

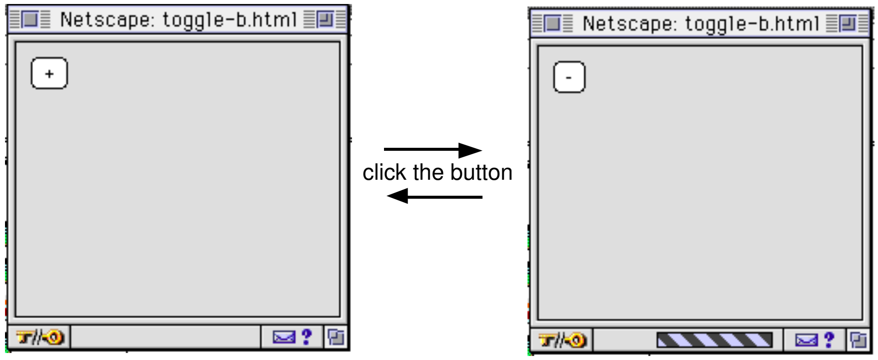

Figure 6. The appearance of two states of the page with a switch

switched by expression (1). The expression value is “–” when the original switch value is “+,” and vice versa. The value is stored into T[0], the first array element, by the assignment to T, and this value is used in the parts labeled “(2)” and “(3).” Each statement that contains these parts runs twice. Part (1) and (2) in the reproduced page are generated from part (2) in the original page. Part (2) is exactly the same as the original, but the switch value in part (1) is changed. Part (3) and (4) in the reproduced page are generated from part (3) in the original page. Part (3) is exactly the same as the original, but the switch value in part (4) is changed. This page is available at http://www.st.rim.or.jp/~kanada/reproduction/toggle-b.html.

§4. Applications of self-reproductive documents

Several simple examples and a more practical example are presented in this section. All the examples are available at http://www.st.rim.or.jp/~kanada/reproduction/examples.html.

The first example, which is shown in **Figure 7**, is a page with a counter. The button in the original page is labeled “0.” Every time the user clicks the button, the label is incremented: “1,” “2,” “3,” and so on. Thus, the content of this page is continually changing and never becomes the same as an ancestor.

```
<html><head><script LANGUAGE="JavaScript">
function reproduce() { var i; var q = "'"; document.clear(); var c = 0;
var A = new Array("", "document.writeln(A[i], "+q); var B = new Array("", q+", B[i]);");
var C = "}; document.writeln(C); for (i = 1; i >= 0; i--) {";
var Q = new Array(q, q+"+Q[i]+"+q); var S = new Array("/", q+"+S[i]+"+q);
var D = new Array(c+1, q+"+D[i]+"+q);
for (i = 0; i <= 1; i++) {
document.writeln(A[i], '<html><head><script LANGUAGE="JavaScript">', B[i]);
document.writeln(A[i], 'function reproduce() { var i; var q = "'+Q[i]+'"; document.clear(); var c = '+D[i]+';', B[i]);
document.writeln(A[i], 'var A = new Array("", "document.writeln(A[i], "+q); var B = new Array("", q+", B[i]);");', B[i]);
document.writeln(A[i], 'var C = "}; document.writeln(C); for (i = 1; i >= 0; i--) {";', B[i]);
document.writeln(A[i], 'var Q = new Array(q, q+"+Q[i]+"+q); var S = new Array("/", q+"+S[i]+"+q);', B[i]);
document.writeln(A[i], 'var D = new Array(c+1, q+"+D[i]+"+q);', B[i]);
document.writeln(A[i], 'for (i = 0; i <= 1; i++) {', B[i]);
}; document.writeln(C); for (i = 1; i >= 0; i--) {
document.writeln(A[i], '}} <'+S[i]+'script><'+S[i]+'head><body>', B[i]);
document.writeln(A[i], '<form><input TYPE="SUBMIT" VALUE="'+D[i]+'" onClick="reproduce()"><'+S[i]+'form>', B[i]);
document.writeln(A[i], '<'+S[i]+'body><'+S[i]+'html>', B[i]);
}} </script></head><body>
<form><input TYPE="SUBMIT" VALUE="0" onClick="reproduce()"></form>
</body></html>
```

Figure 7. An inexact self-reproduction: a page with a counter

The second example, which is shown in **Figure 8**, is a page with simulated radio buttons. There are three buttons in this page. Only one of them is labeled “+” and the other two are labeled “–.” The label of the button that the user clicks becomes “+,” and others become “–.”

The third example, which is shown in **Figure 9**, is more practical. There are three buttons and texts that accompany them. The appearance of this page is shown in **Figure 10**. The user can “open” or “close” the text that accompanies the button. “Closed” content may be an outline and opened content may be a more detailed content. This function is similar to the function of opening or closing folders in Explorer in Microsoft Windows 95 or in Finder in Apple Macintosh. Although the example shown in Figure 9 is still a toy, this function can be used for practical purposes if Web browsers support this function.

```
<html><head><script LANGUAGE="JavaScript">
function reproduce(N) { var i; var q = "'"; document.clear();
var A = new Array("", "document.writeln(A[i], "+q); var B = new Array("", q+", B[i]);");
var C = "}; document.writeln(C); for (i = 1; i >= 0; i--) {";
var Q = new Array(q, q+"+Q[i]+"+q); var S = new Array("/", q+"+S[i]+"+q);
var T0 = new Array(N==0?"+":"-", q+"+T0[i]+"+q);
var T1 = new Array(N==1?"+":"-", q+"+T1[i]+"+q);
var T2 = new Array(N==2?"+":"-", q+"+T2[i]+"+q);
for (i = 0; i <= 1; i++) {
document.writeln(A[i], '<html><head><script LANGUAGE="JavaScript">', B[i]);
document.writeln(A[i], 'function reproduce(N) { var i; var q = "'+Q[i]+'"; document.clear();', B[i]);
document.writeln(A[i], 'var A = new Array("", "document.writeln(A[i], "+q); var B = new Array("", q+", B[i]);");', B[i]);
document.writeln(A[i], 'var C = "}; document.writeln(C); for (i = 1; i >= 0; i--) {";', B[i]);
document.writeln(A[i], 'var Q = new Array(q, q+"+Q[i]+"+q); var S = new Array("/", q+"+S[i]+"+q);', B[i]);
document.writeln(A[i], 'var T0 = new Array(N==0?"+":"-", q+"+T0[i]+"+q);', B[i]);
document.writeln(A[i], 'var T1 = new Array(N==1?"+":"-", q+"+T1[i]+"+q);', B[i]);
document.writeln(A[i], 'var T2 = new Array(N==2?"+":"-", q+"+T2[i]+"+q);', B[i]);
document.writeln(A[i], 'for (i = 0; i <= 1; i++) {', B[i]);
}; document.writeln(C); for (i = 1; i >= 0; i--) {
document.writeln(A[i], '}} <'+S[i]+'script><'+S[i]+'head><body><form>', B[i]);
document.writeln(A[i], '<input TYPE="SUBMIT" VALUE="'+T0[i]+'" onClick="reproduce(0)">', B[i]);
document.writeln(A[i], '<input TYPE="SUBMIT" VALUE="'+T1[i]+'" onClick="reproduce(1)">', B[i]);
document.writeln(A[i], '<input TYPE="SUBMIT" VALUE="'+T2[i]+'" onClick="reproduce(2)">', B[i]);
document.writeln(A[i], '<'+S[i]+'form><'+S[i]+'body><'+S[i]+'html>', B[i]);
}} </script></head><body><form>
<input TYPE="SUBMIT" VALUE="+" onClick="reproduce(0)">
<input TYPE="SUBMIT" VALUE="-" onClick="reproduce(1)">
<input TYPE="SUBMIT" VALUE="-" onClick="reproduce(2)">
</form></body></html>
```

Figure 8. An inexact self-reproduction: a page with radio buttons

```
<html><head><script LANGUAGE="JavaScript">var q = "'";
function mkt(N, V) { var j; this.length = V.length;
for (j = 0; j < V.length; j++) { var not = new Array(0);
not["+"] = "-"; not["-"] = "+"; not["closed"] = "open"; not["open"] = "closed";
this[j] = new Array(Math.floor(j/2)==N?not[V[j]]:V[j], q+"+T["+j+"][i]+"+q); }}
function reproduce(N) { var i; document.clear();
var A = new Array("", "document.writeln(A[i], "+q); var B = new Array("", q+", B[i]);");
var C = "}; document.writeln(C); for (i = 1; i >= 0; i--) {";
var Q = new Array(q, q+"+Q[i]+"+q); var S = new Array("/", q+"+S[i]+"+q);
var T = new mkt(N, new Array("+", "closed",
"+", "closed", "+", "closed"));
for (i = 0; i <= 1; i++) {
document.writeln(A[i], '<html><head><script LANGUAGE="JavaScript">var q = "'+Q[i]+'";', B[i]);
document.writeln(A[i], 'function mkt(N, V) { var j; this.length = V.length;', B[i]);
document.writeln(A[i], 'for (j = 0; j < V.length; j++) { var not = new Array(0);', B[i]);
document.writeln(A[i], 'not["+"] = "-"; not["-"] = "+"; not["closed"] = "open"; not["open"] = "closed";', B[i]);
document.writeln(A[i], 'this[j] = new Array(Math.floor(j/2)==N?not[V[j]]:V[j], q+"+T["+j+"][i]+"+q); }}', B[i]);
document.writeln(A[i], 'function reproduce(N) { var i; document.clear();', B[i]);
document.writeln(A[i], 'var A = new Array("", "document.writeln(A[i], "+q); var B = new Array("", q+", B[i]);");', B[i]);
document.writeln(A[i], 'var C = "}; document.writeln(C); for (i = 1; i >= 0; i--) {";', B[i]);
document.writeln(A[i], 'var Q = new Array(q, q+"+Q[i]+"+q); var S = new Array("/", q+"+S[i]+"+q);', B[i]);
document.writeln(A[i], 'var T = new mkt(N, new Array("'+T[0][i]+'", "'+T[1][i]+'",', B[i]);
document.writeln(A[i], '"'+T[2][i]+'", "'+T[3][i]+'", "'+T[4][i]+'", "'+T[5][i]+'"));', B[i]);
document.writeln(A[i], 'for (i = 0; i <= 1; i++) {', B[i]);
}; document.writeln(C); for (i = 1; i >= 0; i--) {
document.writeln(A[i], '}} <'+S[i]+'script><'+S[i]+'head><body><pre><form>', B[i]);
document.writeln(A[i], '<input TYPE="SUBMIT" VALUE="'+T[0][i]+'" onClick="reproduce(0)">'+T[1][i]+'', B[i]);
document.writeln(A[i], '<input TYPE="SUBMIT" VALUE="'+T[2][i]+'" onClick="reproduce(1)">'+T[3][i]+'', B[i]);
document.writeln(A[i], '<input TYPE="SUBMIT" VALUE="'+T[4][i]+'" onClick="reproduce(2)">'+T[5][i]+'', B[i]);
document.writeln(A[i], '<'+S[i]+'form><'+S[i]+'body><'+S[i]+'html>', B[i]);
}} </script></head><body><pre><form>
<input TYPE="SUBMIT" VALUE="+" onClick="reproduce(0)">closed
<input TYPE="SUBMIT" VALUE="+" onClick="reproduce(1)">closed
<input TYPE="SUBMIT" VALUE="+" onClick="reproduce(2)">closed
</form></body></html>
```

Figure 9. An inexact self-reproduction: view change

Document view change can be implemented using other methods such as below. However, the method using self-reproduction has several advantages as explained below.

- A method using a CGI scripts
  In this method, when the user pushes a button on the page, a request is sent to a server using common gateway interface (CGI). The result is sent from the server to the client. The view can be changed very flexibly in this method and the programming a CGI script is relatively easy. However, the request and the changed page is sent through the network every time the user wants to change the view. Thus, the view change is much slower than the method using self-reproduction.

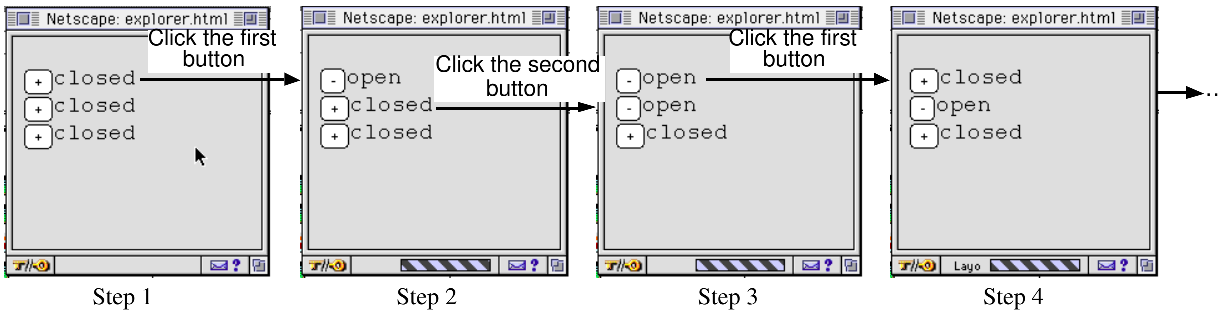

Figure 10. The appearance of the view change example page

- A method using Java or ActiveX
  The buttons are placed on a Java applet [Sun 97] or ActiveX [Mic 97] control in this method. When the user pushes a button, the applet or control changes the view. The applet or control must control everything in the view. This means that it must implement a document browser. Thus, a large number of program steps is necessary and it is hard to make the new browser compatible with the original HTML browser. On the contrary, the original browser is used for displaying a reproduced text in the method using self-reproduction.

- A method using Dynamic HTML
  View change can be implemented much easier by using Dynamic HTML [Net 97b] than using Java or ActiveX. Components of a page, which is called layers, can be added to or removed from the page dynamically. However, this method is still less flexible than the method using CGI or JavaScript, because displayed shapes of layers are limited to be rectangles, and the coordinates in the document must be specified in the page.

§5. A method for viewing generated programs

We can recognize self-reproductive page works on Navigators because its behavior is mostly as expected. However, it is difficult to see the generated program itself using Navigators 3.0 or 4.0. If you open the page source window of Navigator when a generated HTML page is displayed, you will see the original page. If you go back to a generated page from another generated page and open the page source window, you can see the generated text. However, the source does not contain the program because of a bug of Navigators. Thus, another method for viewing generated programs is needed, especially when debugging a self-reproductive program.

A method for viewing generated programs is explained here. If you replace `head` and `script` tag in the call of `document.writeln` by `body` and `pre` tags as described below, this page generates visible text that is mostly the same as the page to be reproduced.

```
<head><script LANGUAGE="JavaScript"> ==> <body><pre>
```

However, this page is not reproductive any longer. If you perform the following process, which is shown in **Figure 11**, you will see the text. You click a button in the transformed page, click any button in the newly generated page, close all the JavaScript error message windows that have been opened by the mouse clicking, back up to the generated page, and open the document source window.

The above tag replacement makes the HTML page syntactically incorrect. If you hate syntax errors very much, you have to replace more tags. However, the above modification is enough for temporary purpose, i.e., debugging.

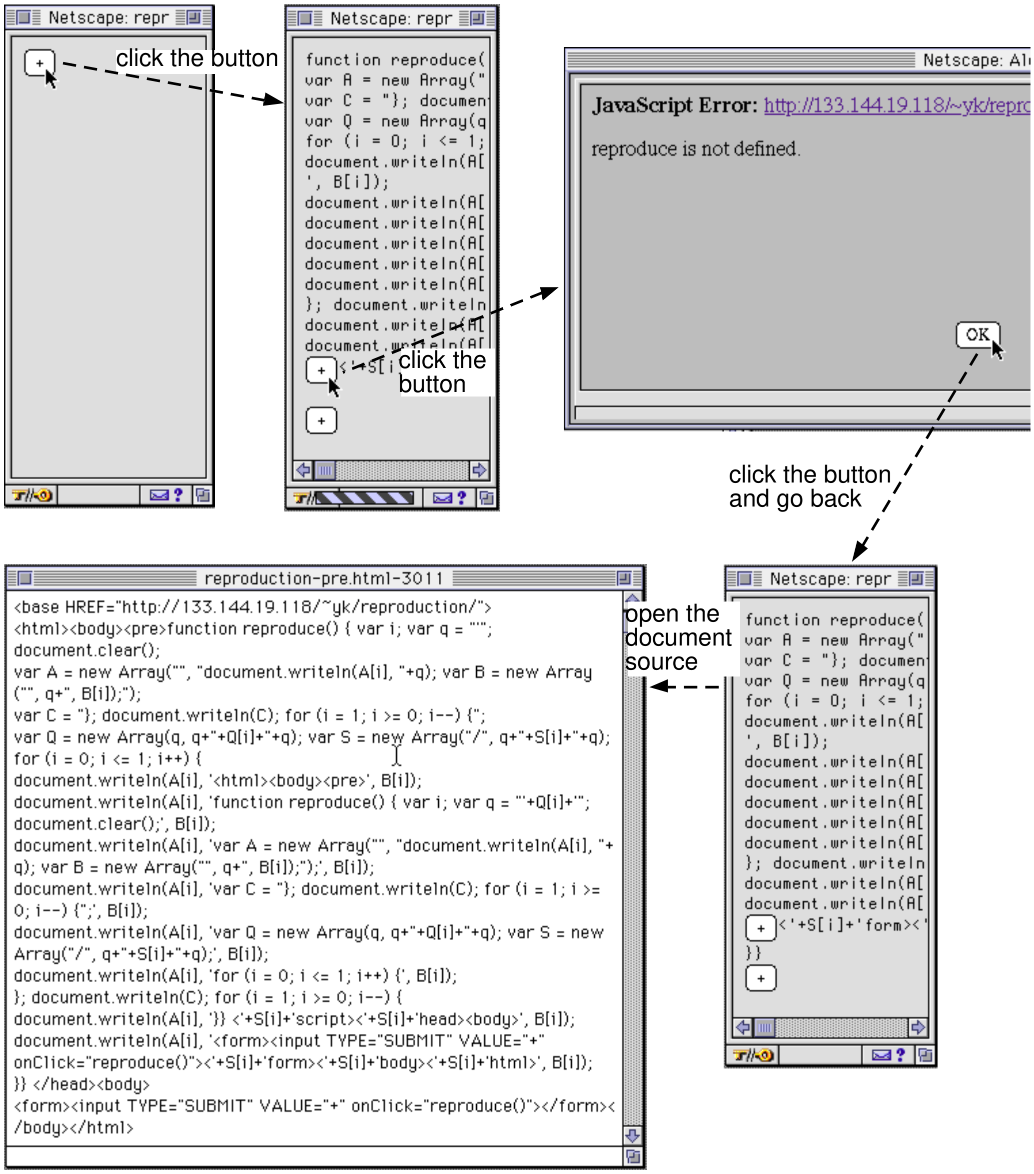

Figure 11. A method of viewing a generated page using Netscape Navigator 3.0

§6. Another method for reproducing documents

Self-reproduction of programs seems to be tricky. There is a method for reproducing documents without reproducing programs. JavaScript program can be put in a separate file (URL) for Netscape Navigators. An exact self-reproductive page using this method is shown in **Figure 12** (a), and the JavaScript program for this page is shown in Figure 12 (b). The relative URL of this program is reproduction-e.js. A `script` tag with `SRC` attribute, which is in Figure 12 (a), refers the external JavaScript program. This program reproduces the HTML page. The new page refers the same program. Inexact reproduction examples, which have been shown above, can also be rewritten easily using this method. A counter is implemented in **Figure 13**. Inexact reproduction can be realized using parameters to the JavaScript function. The function is `reproduction` and the parameter is `c` in this example.

A less tricky, clearer and, thus, more practical HTML pages and programs are obtained using this method. However, this method may be less interesting, and there are two disadvantages. One disadvantage is that this method is less powerful because the same program is used every time the page is reproduced. The other disadvantage is that programs in Figures 12 and 13 do not work on Navigator 3.0. The only current browsers, which these programs work on, is Navigator 4.0 and its preview versions.

```
<html><head><script LANGUAGE="JavaScript" SRC="reproduction-e.js">
</script></head><body>
<form><input TYPE="SUBMIT" VALUE="+" onClick="reproduce()"></form>
</body></html>
```

(a) The page content in HTML

```
function reproduce() {
document.clear();
document.writeln('<html><head><script LANGUAGE="JavaScript" SRC="reproduction-e.js">');
document.writeln('</script></head><body>');
document.writeln('<form><input TYPE="SUBMIT" VALUE="+" onClick="reproduce()"></form>');
document.writeln('</body></html>');
document.close();
}
```

(b) The JavaScript program (reproduction-e.js)

Figure 12. An exact self-reproduction without program reproduction

```
<html><head><script LANGUAGE="JavaScript" SRC="count-up-e.js"></script>
</head><body>
<form><input TYPE="SUBMIT" VALUE="0" onClick="reproduce(0)"></form>
</body></html>
```

(a) The page content in HTML

```
function reproduce(c) {
var c1 = c + 1;
document.clear();
document.writeln('<html><head><script LANGUAGE="JavaScript" SRC="count-up-e.js"></script>');
document.writeln('</head><body>');
document.writeln('<form><input TYPE="SUBMIT" VALUE="'+c1+'" onClick="reproduce('+c1+')"></form>');
document.writeln('</body></html>');
document.close();
}
```

(b) The JavaScript program (count-up-e.js)

Figure 13. An inexact self-reproduction without program reproduction: a page with a counter

§7. Conclusion

If a document can contain a program that reproduces the document, this function can be used for transforming the document. Changing the view or content of the document may be used for practical purposes, such as partially changing views from outline to detail. This function currently works partially only on Netscape Navigators only for HTML documents. If every JavaScript-ready Web browser supports reproduction function and JavaScript standard, such as ISO's, support this function, it can be used throughout the Internet for changing document views and other purposes. The function can be implemented not only for HTML documents, but also for SGML or XML documents, and documents written in other mark-up languages.

References

[Ari 94] Arisawa, M.: *Methods of Thinking in Programming*, Nikkagiren Publisher, 1994 (in Japanese)

[Mic 97] *active platform*, http://www.microsoft.com/activeplatform/default.asp, Microsoft Corporation.

[Net 97a] *JavaScript 1.1 Language Specification*, http://home.netscape.com/eng/javascript/index.html, Netscape Communications Corp.

[Net 97b] *Netscape Communicator: What's Hot*, http://home.netscape.com/comprod/products/communicator/beta_features.html, Netscape Communications Corp.

[Sun 97] *What is Java?*, http://java.sun.com/nav/whatis/index.html, Sun Microsystems, Inc.