DANGERO.US

PREMIUM SITE TEMPLATES

“TypeWriter”

Premium Script By iDangero.us

Documentation

Thank you for purchasing this item. If you have any questions that are beyond the scope of this help file, please feel free to contact us via support ticket form here.

Table of Contents

1. About “TypeWriter”

TypeWriter is a premium JavaScript which allows you to print any phrases. It can be used in any purpose, such as printing welcome message, FAQs, conversation, advertisement, AutoFill etc. (see included demo site).

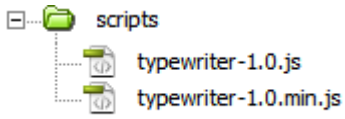
This is a standalone script, and therefore it does not require any JavaScript library such as jQuery or MooTools.

2. Typewriter Files Structure

In the downloaded archive you’ll find two folders – **demo-site** and **scripts**.

Demo site located in **demo-site** folder is intended to help you to learn how the typewriter works on the real examples.

Scripts folder contains two version of TypeWriter script – minified and full versions:



3. Installation

To install **TypeWriter** script you need to copy one of typewriter files (minified or full) to your website (for example into the **scripts** folder), and to link this JavaScript (js) file to your website using the <script> tag.

For example, your website URL is **www.example.com**, and you have copied minified version of TypeWriter into the **scripts** root folder. You’ll need to add this code in the <head> section of your HTML file:

```
<head>
-----------
<script src="http://www.example.com/scripts/typewriter-1.0.min.js" type="text/javascript"></script>
-----------
</head>
```

4. TypeWriter Usage

After the TypeWriter is installed (linked to website) then the typeWriter() JavaScript function will be available for usage. Let's look at this function and its parameters more closely to know how to use it:

Function

```
typeWriter({twID:integer, text:string, id:string, delay:integer }, action)
```

or

```
typeWriter({twID:integer, text:string, id:string, time:integer }, action)
```

Function Parameters

twID – unique integer number. Required only if you are using several TypeWriters on one page (like on the Demo site). Default value is **1**.

text – string value. Text which is must be printed by TypeWriter. Required

id – string value. Value of the ID attribute of the container where text will be printed. Required

delay – integer number. Delay in ms for each character printing. Not required, default value is **50** ms (if not specified).

time – integer number. Time in ms - this is the time for which will be printed the entire text. Not required.

If delay or time parameters are not specified, than the TypeWriter will be executed with a delay parameter equal to 50 ms.

action – string value. Action value is the special command which is tells to TypeWriter what to do with a container, specified with **id** parameter. Not required. It could take one of three values:

- **clear** – Default value. If specified, than when the container (specified with **id** parameter) content will be removed before printing.

- **stop** – If specified, then printing will stop (for the TypeWriter with the same **twID** parameter)
- **add** – if specified, then the printing text will be added to the container (specified with **id** parameter) content.

TypeWriter Examples

Of course there is no sense to use TypeWriter without events (such as onclick, onmouseover, onmouseout, etc.). Let's look at the usage examples with button (button with a "print-button" id attribute), and when we'll click on it - the some message will be printed to some container (div with a "print-here" id attribute). We also need **stop** and **add** buttons. Here is the simple HTML formatting:

```
<div id="print-here"></div>
<input type="button" id="print-button" />
<input type="button" id="stop-button" />
<input type="button" id="add-button" />
```

Example 1. We are using only one TypeWriter on the page. We want to print the "Hello World!" phrase. We need to add some script to page which will control our TypeWriter, and we should use the following formatting:

```
<script type="text/javascript">

document.getElementById('print-button').onclick = function() {
    typeWriter({text:'Hello World!',id:'print-here',delay:100})
}
document.getElementById('stop-button').onclick = function() {
    typeWriter({},"stop")
}
document.getElementById('add-button').onclick = function() {
    typeWriter({text:"Some Additional Text",id:'print-here'},"add")
}

</script>
```

Every character will be printed with a delay in 100 ms. As you can see the **action** parameter is not specified, in this case it is equal to the default value - "**clear**"

As you can see the **delay** parameter is not specified, in this case it is equal to the default value - "**50**"

Or by using the onclick attributes for buttons:

```
<input type="button" id="print-button" onclick="typeWriter({text:'Hello World!',id:'print-here',delay:100})" />
<input type="button" id="stop-button" onclick="typeWriter({},'stop')" />
<input type="button" id="add-button" onclick="typeWriter({text:'Some Additional Text',id:'print-here'},'add')" />
```

Of course it is not required to use STOP and ADD buttons every time. Only if need.

Example 2. We are using three TypeWriters on the page. For each TypeWriter we need to print different text and three control buttons. HTML formatting:

```
<div id="print-here1"></div>
<input type="button" id="print-button1" />
<input type="button" id="stop-button1" />
<input type="button" id="add-button1" />
<div id="print-here2"></div>
<input type="button" id="print-button2" />
<input type="button" id="stop-button2" />
<input type="button" id="add-button2" />
<div id="print-here3"></div>
<input type="button" id="print-button3" />
<input type="button" id="stop-button1" />
<input type="button" id="add-button3" />
```

When we are using several TypeWriters it is required to specify twID parameter. It must be integer number and unique for every TypeWriter. Here is script:

```
<script type="text/javascript">
//First TypeWriter
document.getElementById('print-button1').onclick = function() {
    typeWriter({twID:1,text:'Text for first TW',id:'print-here1'})
}
document.getElementById('stop-button1').onclick = function() {
    typeWriter({twID:1},"stop")
}
document.getElementById('add-button1').onclick = function() {
    typeWriter({twID:1,text:"Additional Text for first TW",id:'print-here1'},"add")
}
//Second TypeWriter
document.getElementById('print-button2').onclick = function() {
    typeWriter({twID:2,text:'Text for second TW',id:'print-here2',delay:30})
}
document.getElementById('stop-button2').onclick = function() {
    typeWriter({twID:2},"stop")
}
document.getElementById('add-button2').onclick = function() {
    typeWriter({twID:2,text:"Additional Text for second TW",id:'print-here2',delay:20},"add")
}
//Third TypeWriter
document.getElementById('print-button3').onclick = function() {
    typeWriter({twID:3,text:'Text for third TW',id:'print-here3',time:400})
}
document.getElementById('stop-button3').onclick = function() {
    typeWriter({twID:3},"stop")
}
document.getElementById('add-button3').onclick = function() {
    typeWriter({twID:3,text:"Additional Text for third TW",id:'print-here3',delay:100},"add")
}
</script>
```

As you can see it is very easy to use TypeWriter.

5. Text Formatting

In the previous examples our TypeWriter prints plain text without any formatting. Now let's look at the special text formatting acceptable for TypeWriter on the real example:

```
<div id="print-here"></div>
<input type="button" id="print-button" />

<script type="text/javascript">

var $textToPrint = "Hello! My name is TypeWriter!\n";
    $textToPrint += "I'm a premuim [#color:#777;font-weight:bold]JavaScript[/#] created by \0\0iDangero.us\n";
    $textToPrint += "I'm very\b\b\b\b [&cool]cool[/&] and [&funny]funny[/&]!\n";
    $textToPrint += "\tTypeWriter";

document.getElementById('print-button').onclick = function() {
    typeWriter({text:$textToPrint,id:'print-here',delay:100})
}

</script>
```

According to the example above TypeWriter will print text from $textToPrint variable to the container with a "print-here" id attribute. As you can see $textToPrint contains special formatting characters. Let's look at their values:

\n - line feed special character. Will print the **
** HTML tag.

\0 – extra delay for printing. Print nothing, but printing process will be delayed on the value equal to double delay value (specified in the typewriter() parameters). In the example above printing will be delayed for 400ms – 2 extra delay characters *(2*delay)

\b – one symbol will remove one character. On the example above the "very" word will be removed character by character.

\t - horizontal tabulation. Will print the "**   **"

[#some_css_style] … *Styled Phrase* … **[/#]** – additional styling for phrases or characters. Every character in the Styled Phrase will be wrapped with a …character…**** tags. In the example above the "JavaScript" phrase will be printed like that:

```
<span style="color:#777;font-weight:bold">J</span>
<span style="color:#777;font-weight:bold">a</span>
<span style="color:#777;font-weight:bold">v</span>
<span style="color:#777;font-weight:bold">a</span>
<span style="color:#777;font-weight:bold">S</span>
<span style="color:#777;font-weight:bold">c</span>
<span style="color:#777;font-weight:bold">r</span>
<span style="color:#777;font-weight:bold">i</span>
<span style="color:#777;font-weight:bold">p</span>
<span style="color:#777;font-weight:bold">t</span>
```

[&additional_class] *… Styled Phrase …* **[/&]** – additional class for phrases or characters. Every character in the Styled Phrase will be wrapped with a …character… tags. In the example above the “cool” and “funny” phrases will be printed like that:

```
<span class="cool">c</span>
<span class="cool">o</span>
<span class="cool">o</span>
<span class="cool">l</span>

<span class="funny">f</span>
<span class="funny">u</span>
<span class="funny">n</span>
<span class="funny">n</span>
<span class="funny">y</span>
```

6. TypeWriter JavaScript Explanation

Let's open typewriter-1.0.js file and look at the JavaScript code of it.

```
1  typeWriterContainer = [];
2  typeWriterTimeouts = [];
3  function typeWriter(a,action){
4      a['twID']==null?a['twID']=1:a['twID']=a['twID'];
5      action==null?action="clear":action=action;
6      if (!a['text'] && action=="clear") throw new Error("TypeWriter argument 'text' is undefined!");
7      if (!a['id'] && action=="clear") throw new Error("TypeWriter argument 'id' is undefined!");
8      if(!typeWriterTimeouts[a['twID']]) typeWriterTimeouts[a['twID']] = new Array();
9      if(action == "clear") {
10         clearTwTimeouts();
11         document.getElementById(a['id']).innerHTML="";
12     }
13     if(action == "stop") {
14         clearTwTimeouts();
15         return
16     }
17     if(action == "add") {
18         clearTwTimeouts();
19     }
20     a['delay'] = a['time']?a['time']/a['text'].length:(a['delay'] || 50);
21     a['id'] = a['id'].indexOf("#")>=0?a['id'] = a['id'].substring(1,a['id'].length):a['id'];
22     typeWriterContainer[a['twID']] = document.getElementById(a['id']);
23     var newType = new Object();
24     for (var $i=0,$j=0,$k=0;$i<a['text'].length;) {
25       var specialChar = a['text'].substr($i+1,1)
26       if (a['text'].substr($i,1)!="[" && (specialChar!="#" || specialChar!="&")) {
27          if(!newType['text'+$j]) {
28              newType['text'+$j]={};
29              newType['text'+$j]['phrase']="";
30          }
31          newType['text'+$j]['phrase'] += a['text'].substr($i,1);
32          numberOfPhrases = $j;
33          $i++
34       }
```

These two arrays are used to store information about TypeWriters containers and Timeouts intervals for each typewriter.

If **twID** parameter is not specified it will be set equal to "1".

If **action** parameter is not specified it will be set equal to "clear".

These two lines generate Errors if **text** and **id** parameters are not specified.

If **action** parameter is equal to "**clear**" then the **clearTwTimeouts**() function will be executed and the container content will be removed

If **action** parameter is equal to "**stop**" then the **clearTwTimeouts**() function will be executed and the typewriter() function will be stopped.

If **action** parameter is equal to "**add**" then the **clearTwTimeouts**() function will be executed.

If the **time** parameter is specified then the **delay** parameter will be equal to time/text-length, otherwise it will be equal to **delay** parameter (if it is not specified then it will be equal to **50**

newType – is the object which will contain all text parts from the formatted text

This **FOR** loop used to search all characters in the formatted text and to divide it into new objects (text parts) with appropriate parameters. First **IF** statement used to get together simple characters without any style formatting. And after the new object will be created inside **newType** object with the appropriate property – **phrase.**

```
35        else {
36           if (newType['text'+$j]) $j++;
37           var styledPhrase = a['text'].substr($i).split('[/'+specialChar+']')[0].split(']')[1]
38           var phraseStyleClass = a['text'].substr($i).split(styledPhrase+'[/'+specialChar+']')[0].split('['+specialChar+'')[1].split(']
')[0];
39           newType['text'+$j] = {};
40           newType['text'+$j]['phrase'] = styledPhrase;
41           var addionalProperty = specialChar=="#"?'style':'class';
42           newType['text'+$j][addionalProperty] = phraseStyleClass;
43           var styledLenght = ('['+specialChar+''+phraseStyleClass+']'+styledPhrase+'[/'+specialChar+']').length;
44           $i+=styledLenght;
45           numberOfPhrases = $j;
46           $j++;
47        }
48     }
49     var charDelay = 0,newChar,charExtraDelay = 0
50     $j=0;
51     for (textPart in newType) {
52        for ($i=0; $i< newType['text'+$j]['phrase'].length;$i++) {
53           newChar = newType['text'+$j]['phrase'].substr($i,1)
54           charDelay +=(a['delay']+charExtraDelay);
55           charExtraDelay =0;
56           switch (newChar){
57            case "\n": typeWriterTimeouts[a['twID']].push(setTimeout('typeWriterContainer['+a['twID']+'].innerHTML+="<br/>"',
charDelay));
58            case "\0": {
59               charExtraDelay=a['delay']*2;
60            }
61            break;
62            case "\b": {
63               function removeChar() {
64                  if (typeWriterContainer[a['twID']].lastChild.nodeValue == null) {
65                     typeWriterContainer[a['twID']].removeChild(typeWriterContainer[a['twID']].lastChild);
66                  }
67                  else typeWriterContainer[a['twID']].innerHTML=typeWriterContainer[a['twID']].innerHTML.substr(0,
typeWriterContainer[a['twID']].innerHTML.length-1);
68               }
69               typeWriterTimeouts[a['twID']].push(setTimeout(removeChar,charDelay));
70
71           }
72           break;
73           case " ":  typeWriterTimeouts[a['twID']].push(setTimeout('typeWriterContainer['+a['twID']+'].innerHTML+=" "',charDelay));
74           break;
75           case "\t": {
76               typeWriterTimeouts[a['twID']].push(setTimeout('typeWriterContainer['+a['twID']+'].innerHTML+="   "',
charDelay));
77           }
78           break;
```

This **ELSE** construction is for the case when the **&** or # character is found. And after it is found the new object will be created inside **newType** object with the appropriate properties – **phrase** and **class (or style).**

After the newType object is filled with other text objects (text parts), TypeWriter will print them according with the statements bellow. First **FOR** loop is used to search text parts in newType object, and the second inner **FOR** loop is to search characters in every text parts.

If the printed character is “/n” then the
 html tag will be added inside the TypeWriter container.

If the printed character is “/0” then the delay for all following characters will be shifted to double **delay** time.

If the printed character is “/b” then the previously printed character will be removed.

This statement used when the previous character is wrapped with **span** tag

This statement used when the previous character is simple character

If you want to get full control over the number of space characters you can change the “ “ to “ ” here:

If the printed character is “ ” (space) then the same space character will be added inside the TypeWriter container.

If the printed character is “/t” then the “   ” formatting will be added inside the TypeWriter container.

This default case is used for all other characters

If text part has the **style** parameter then the new character wrapped with **span** tag (with class attribute) will be added inside the TypeWriter container.

If text part has the **class** parameter then the new character wrapped with **span** tag (with style attribute) will be added inside the TypeWriter container.

Otherwise the simple character will be added.

This function used to clear timeouts interval of TypeWriter. If it is called then the print process will be stopped.

```
79              default:
80                if (newType['text'+$j]['style']) {
81                      typeWriterTimeouts[a['twID']].push(setTimeout('typeWriterContainer['+a['twID']+'].innerHTML+=\'<span style="'+
    newType['text'+$j]['style']+'">'+newChar+'</span>\'',charDelay));
82                }
83                else if (newType['text'+$j]['class']) {
84                      typeWriterTimeouts[a['twID']].push(setTimeout('typeWriterContainer['+a['twID']+'].innerHTML+=\'<span class="'+
    newType['text'+$j]['class']+'">'+newChar+'</span>\'',charDelay));
85                }
86                else {
87                      typeWriterTimeouts[a['twID']].push(setTimeout('typeWriterContainer['+a['twID']+'].innerHTML+="'+newChar+'"',
    charDelay));
88                }
89              break;
90             }
91         }
92          $j++;
93     }
94     function clearTwTimeouts(){
95         for (var i=0;i<typeWriterTimeouts[a['twID']].length;i++) {
96             window.clearTimeout(typeWriterTimeouts[a['twID']][i])
97         }
98     }
99 }
100 window['typeWriter'] = typeWriter;
```

7. Licensing Terms

Regular License (RL)

RL gives you as a customer non-exclusive & non-transferable right to use the product you've bought, in this case is the **TypeWriter** (further "**Item**")

1. We do not limit the number of **Item**'s copies you are going to use. Using one **Item** you can create, for example, even 100 web-sites.
2. You can use the **Item** by itself or it's also possible to apply it in other project you work at.
3. You can use the **Item** for your own purpose as well as for your clients.
4. You can use the **Item** in commercial projects.
5. You can reproduce the **Item**:

- on a web-site or as a web-site
- as a part of software you create not for sale
- as a printed variant
- in digital(electronic) format (as a presentation or an e-book)
- in video products

6. All photos used on iDangero.us for demonstration photo gallery functioning are our private property (if only there is nothing said about contrary). It's prohibited to use these photos wherever.
7. You are not allowed to sell, resell, license or give the **Item** free (any way) without our written consent. Please, do not offer to do it to any person.
8. You also do not have the right to use the **Item** in a project for selling (for example, for other templates, scripts, graphics and so on).

9. It’s prohibited to rework / redesign / reproduce the **Item** (i.e.to rename it or change graphics & so on) and after this to sell it as your own.

10. In spite of reselling limitation you could claim money for the **Item** from your client.

11. If the **Item** (the whole **Item** or its parts) is created with materials used by GNU General Public License (GPL) (or some other license) it means you should follow all the terms of the license using the **Item**.