

Forum:[Hack Forums](#)

Thread: [\[Antagonism/Infamous\] Javascript - Loop Escaping, EObjects, Events & Funcs \[Tutorial\]](#)

Keeper - May 25, 2013, 1:45 pm

Introduction:

> This is a continuation of my first tutorial on Javascript [Antagonism | Javascript Basics \[Tutorial\]](#). In this one I'll go through break and continue 'mechanisms' for escaping loops, embedded objects, functions and events in Javascript. I'll give out examples for every explanation and part of the tutorial so that you may keep up with the concept.

Loop Escaping:

Of course, Javascript enables us to terminate the current looping and proceed with the next iteration of the cycle. break casts the control outside the loop. and continue casts to the next iteration. Let's take the following modification of the function forExample():

```
Code function forExample() {  
    for (i = 0; i < 10; i++) {  
        if (i % 2 == 0) { continue }  
        document.forms[0].elements[0].value =  
        document.forms[0].elements[0].value + "\n" +  
        "the value of 'i' is: " + i;  
    }  
}
```

The if() cycle tests the remaining of the diving of i by 2. If the remaining is 0, continue proceeds to the next iteration of the for() cycle.

The idea is that only the odd numbers of the value of i are being visualized.

Embedded Objects:

In Javascript, the objects String, Data and Math are embedded into the language itself. Each one of them enables the usage of some pretty well-known methods. Some of the most common ones are:

anchor()

- Places the string in an HTML tag for an anchor. For example "

bold()

- Makes the font bold

italics()

- Makes the font italics

substring()

- Returns a substring with a predefined beginning or end

toLowerCase()

- Lowercases the string

toUpperCase()

- Uppercases the string

All objects contain the following three methods:

eval

- converts string into Javascript code

toString

- converts the object into a string

valueOf

- converts the object into a primitive value

Let's see an example with some methods using a function. First initialize a new string in a separate file:

Code `var s = new String('Antagonism/Infamous');`

Now let's write the function that accepts one argument (stringName). It will take a string value:

Code `function stringDemo(stringName) {`

Now we'll call the function stringDemo with the argument s and use document.write() to visualize the output:

Code `document.write(stringName + '<br />');`

The next three lines will be outputting the formatted and italicized font, the second one the bold text and the third line will show how we can combine them.

Code `document.write(stringName.italics() + '<br />');`

`document.write(stringName.bold() + '<br />');`

`document.write(stringName.bold().italics().fontSize(25) + '<br />');`

Now let's convert the string into an array. The next line cuts off the places where there is a space and visualizes the segments as elements of an array:

Code `document.write(stringName.split(" ") + '<br />');`

We'll now sort the elements of that array with sort():

Code `document.write(stringName.split(" ").sort() + '<br />');`

And finally let's actually end our function till here:

Code `}`

`</script>`

Append the following HTML to output the examples we've written and explained so far:

Code `..`

`<body onload = stringDemo(s)>`

`..`

The calling of the function stringDemo() in the body tag serves for the activation of it whenever the page is loaded/accessed. The function is called with the argument s. Here we end up with the final 'compiled' code, combining all the examples we've given:

Code `<script>`

`var s = new String('Antagonism/Infamous');`

`function stringDemo(stringName) {`

`document.write(stringName + '<br />');`

`document.write(stringName.italics() + '<br />');`

`document.write(stringName.bold() + '<br />');`

`document.write(stringName.bold().italics().fontSize(25) + '<br />');`

`document.write(stringName.split(" ") + '<br />');`

`document.write(stringName.split(" ").sort() + '<br />');`

```
}  
</script>  
Functions:
```

Functions are used to perform specific tasks and are being called by undefined sources.

```
Code function name( [arguments/parameters] ) {  
[code to be executed]  
}
```

Therefore, a basic example of that would be the following:

```
Code <script>  
function Keeper() {  
document.write('Devoted to Antagonism & Infamous');  
}  
</script>
```

```
<button onclick="Keeper()">Visualize the example</button>
```

Let's take an example with a function that accepts two arguments and returns, let's say the multiplication of two integers:

```
Code function Keeper(first,second) {  
return first * second;  
}
```

```
document.getElementById('example').innerHTML=Keeper(5,3);
```

Obviously the result will be 15. Of course, we can force the arguments to be user supplied data and make a sort of a calculator with no more than 10 LOC.

Javascript Events:

Another common thing in Javascript are the events. They react accordingly to the user's interaction with the script/page, performing specific construction or calling other function from within the Javascript code. They ensure extra information for the user, reacting to his actions over the page. They are of course declared by the developer. For example with the functions `onMouseOver()` and `onMouseOut()`, we can include code for altering the content of a certain element in our mark-up. Take the following example:

```
Code <a href='http://example.com/' onMouseOver="window.status='Check the link';">A Link</a>  
Upon hovering the link, you'll see the following status of the windows:
```

You can also use the switch construction if you want to alter an image's information or simply switch two different image files:

```
Code function switcher(place) {  
swtich (place){  
    case(1): document.[element].src="image2.jpg";  
        window.status='Second image';  
        break;  
    case(2): document.[element].src="image1.jpg";  
        window.status='First image';
```

```
        break;
    }
}
```

Now let's talk a bit about cookies and sessions. Suppose you want to save some information about a session (for example ones that are for browsing within the website). Let's start off with the first script that has a function for the local date and time and then initialize the cookie from the function `onLoad()`:

```
Code <script>
function Cookie() {
var obtained = new Date();
document.cookie = obtained.toLocaleString();
}
</script>
```

```
<body onLoad = 'Cookie()'
```

The function for event handling starts the function we just made (`Cookie()`) which generates and appends a value to the variable `obtained`. Things stand the same for the leaving of the page. Say the user was to close it:

```
Code function Leave() {
var timeVisit = new Date();
    var tempTime = timeVisit.toLocaleString();
    alert("You visited that page at: " + document.cookie + " and now it is " + tempTime);
}
</script>
```

```
<body onLoad = 'Cookie()' onUnload = 'Leave()'
```

Conclusion:

This was the second part of my series on Javascript. Can't say I included everything I initially listed on my table of contents but let there be room for the future tutorials. If anybody has any questions or issues related to the example and explanations included in this part, leave a reply and I will try to troubleshoot. Thanks for reading!

-?Divine- - May 25, 2013, 1:46 pm

Very nice tutorial Keeper. Good job.

h?cker22 - May 25, 2013, 7:15 pm

Great job Keeper, you probably hear that a lot, but I mean it.

Thank you.