**Writing in-browser JavaScript and Developer Console | Web Development Tutorials #47**

In this tutorial, we will see How to write JavaScript and what are the methods to write it. We will make a new file as *tut46.html* and add an instant boilerplate to get the basic HTML code to get started. Then give the title as **JavaScript Tutorial** under the <title> tag.

The next step is to add the HTML code in the body. We will add the container and a simple paragraph as shown below, just to understand the concept of JavaScript.

<body>

<div class="container">

<div class="row">

<p>

This is a row in this container

</p>

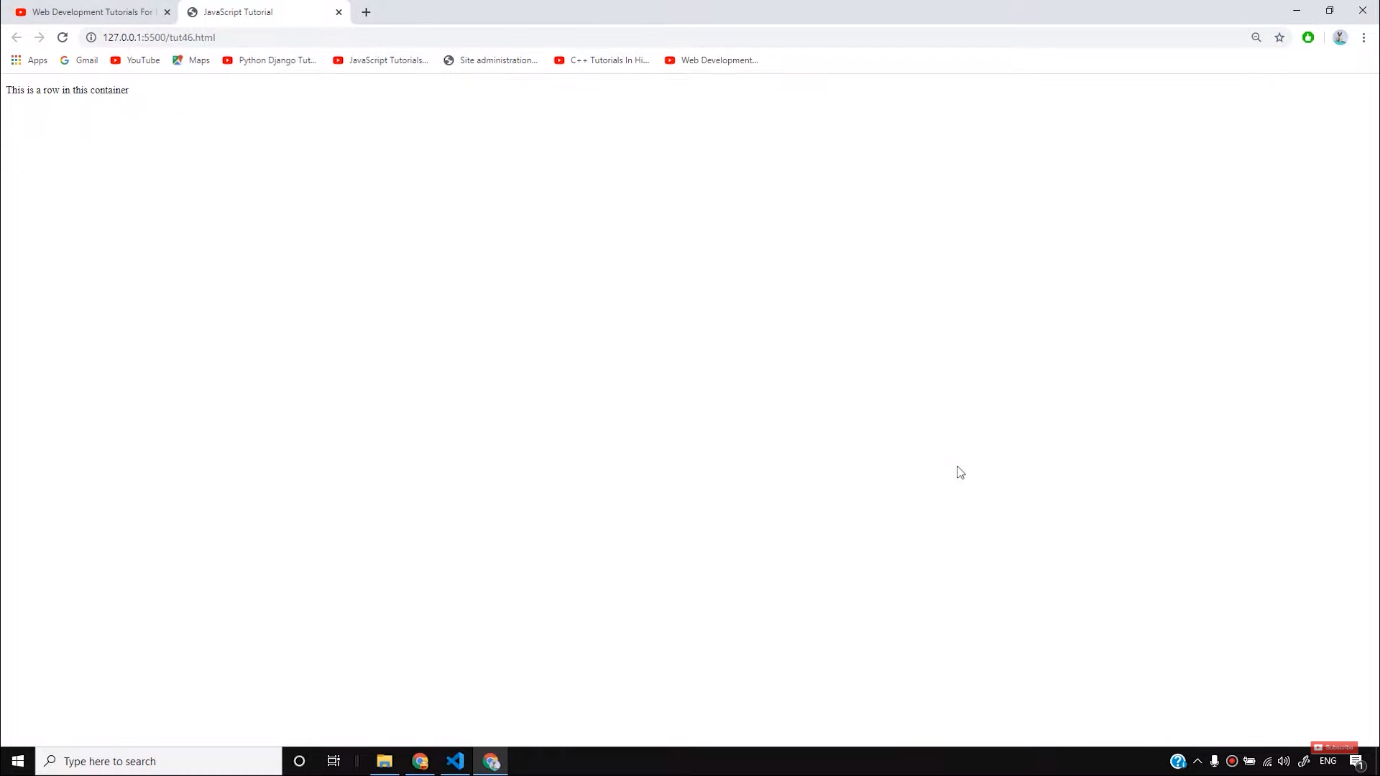
</div>

</div>

</body>

Copy

The result of the above code will look as follows-



However, our main focus in this tutorial is to understand JavaScript. Therefore, we will not do any styling or add CSS here.

We have two options to place the JavaScript in the code. One is in the <*head>*section and the other is after the *<body>* tag. But if we view other professional websites, they place their JavaScript at the end of the <body> tag so that the DOM does not get affected. The basic code of JavaScript is as follows-

<script>

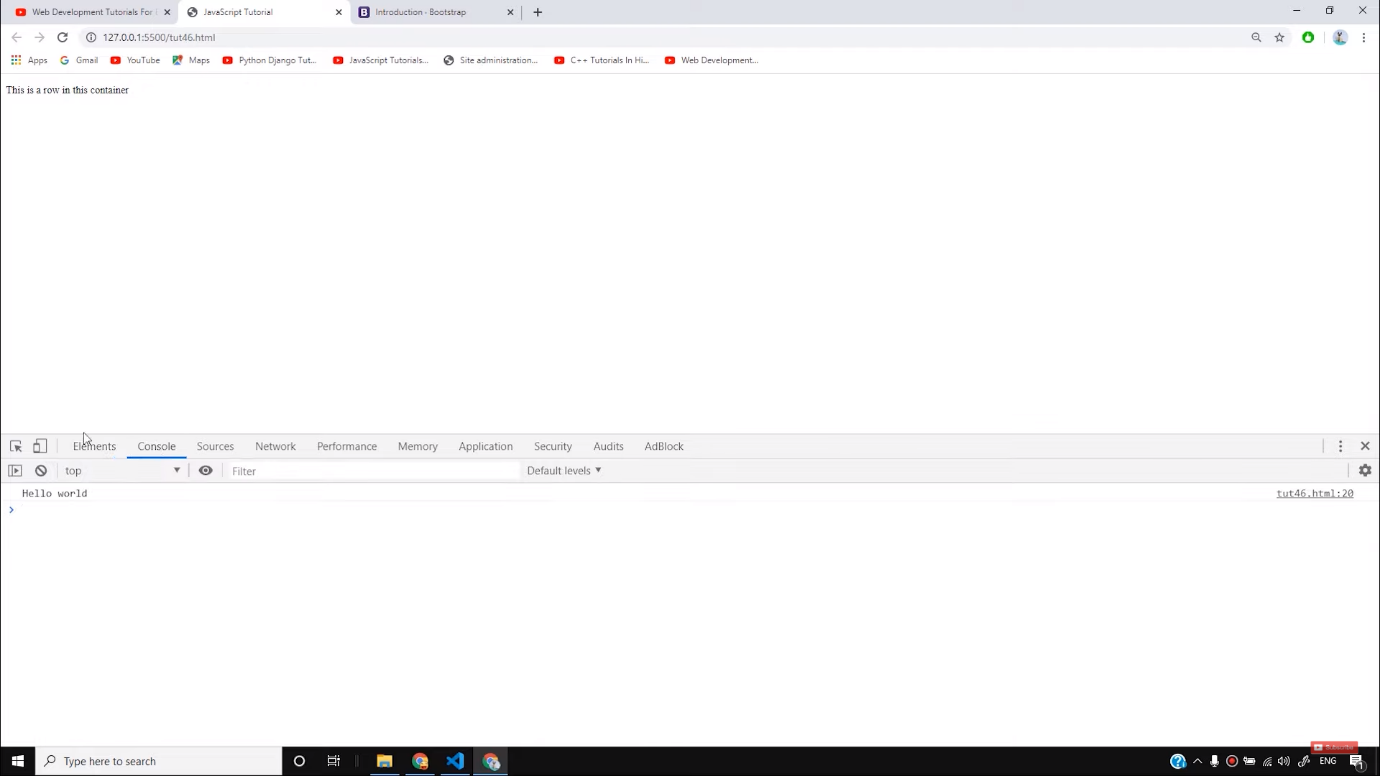
//Write your js here

console.log('Hello world');

</script>

Copy

If we open the **console** section of the browser, we can see the output of the above code as shown below-



Console is the only place, where you will find all the errors made in the code. If you have locked any request or have applied AJAX in the code, the errors regarding all these things will be shown in the console tab itself. Also, we can see the values of different variables if written in the code. Although there may be a chance that if you are using any other browser apart from Chrome like Firefox or Safari, you may find a different console tab, the high-level ideas of all the browsers will be the same. Therefore, it is recommended to use the Chrome browser due to its high developer tools.

So I believe you must have got some basic ideas about JavaScript and the developer console provided by the browser. Furthermore, in the tutorials, we will see some advanced features of JavaScripts and how to make a professional website with a *hamburger* icon on it. To learn more about JavaScript stay with the tutorials.

**Code as described/written in the video**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript Tutorial</title>

</head>

<body>

<div class="container">

<div class="row">

<p>

This is a row in this container

</p>

</div>

</div>

<script>

//Write your js here

console.log('Hello world');

</script>

</body>

</html>

# JavaScript Introduction

[❮ Previous](https://www.w3schools.com/js/default.asp)[Next ❯](https://www.w3schools.com/js/js_whereto.asp)

This page contains some examples of what JavaScript can do.

## **JavaScript Can Change HTML Content**

One of many JavaScript HTML methods is getElementById().

The example below "finds" an HTML element (with id="demo"), and changes the element content (innerHTML) to "Hello JavaScript":

### **Example**

document.getElementById("demo").innerHTML = "Hello JavaScript";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_inner_html)

JavaScript accepts both double and single quotes:

### **Example**

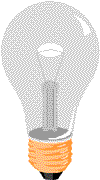
document.getElementById('demo').innerHTML = 'Hello JavaScript';

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_inner_html_quotes)

## **JavaScript Can Change HTML Attribute Values**

In this example JavaScript changes the value of the src (source) attribute of an <img> tag:

### **The Light Bulb**

Turn on the light  Turn off the light

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_lightbulb)

ADVERTISEMENT

## **JavaScript Can Change HTML Styles (CSS)**

Changing the style of an HTML element, is a variant of changing an HTML attribute:

### **Example**

document.getElementById("demo").style.fontSize = "35px";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_style)

## **JavaScript Can Hide HTML Elements**

Hiding HTML elements can be done by changing the display style:

### **Example**

document.getElementById("demo").style.display = "none";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_hide)

## **JavaScript Can Show HTML Elements**

Showing hidden HTML elements can also be done by changing the display style:

### **Example**

document.getElementById("demo").style.display = "block";

# JavaScript Where To

[❮ Previous](https://www.w3schools.com/js/js_intro.asp)[Next ❯](https://www.w3schools.com/js/js_output.asp)

## **The <script> Tag**

In HTML, JavaScript code is inserted between <script> and </script> tags.

### **Example**

<script>  
document.getElementById("demo").innerHTML = "My First JavaScript";  
</script>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_whereto)

Old JavaScript examples may use a type attribute: <script type="text/javascript">.  
The type attribute is not required. JavaScript is the default scripting language in HTML.

## **JavaScript Functions and Events**

A JavaScript function is a block of JavaScript code, that can be executed when "called" for.

For example, a function can be called when an **event** occurs, like when the user clicks a button.

You will learn much more about functions and events in later chapters.

## **JavaScript in <head> or <body>**

You can place any number of scripts in an HTML document.

Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

## **JavaScript in <head>**

In this example, a JavaScript function is placed in the <head> section of an HTML page.

The function is invoked (called) when a button is clicked:

### **Example**

<!DOCTYPE html>  
<html>  
<head>  
<script>  
function myFunction() {  
  document.getElementById("demo").innerHTML = "Paragraph changed.";  
}  
</script>  
</head>  
<body>

<h2>Demo JavaScript in Head</h2>  
  
<p id="demo">A Paragraph</p>  
<button type="button" onclick="myFunction()">Try it</button>

</body>  
</html>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_whereto_head)

ADVERTISEMENT

## **JavaScript in <body>**

In this example, a JavaScript function is placed in the <body> section of an HTML page.

The function is invoked (called) when a button is clicked:

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<h2>Demo JavaScript in Body</h2>  
  
<p id="demo">A Paragraph</p>  
  
<button type="button" onclick="myFunction()">Try it</button>  
  
<script>  
function myFunction() {  
  document.getElementById("demo").innerHTML = "Paragraph changed.";  
}  
</script>  
  
</body>  
</html>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_whereto_body)

Placing scripts at the bottom of the <body> element improves the display speed, because script interpretation slows down the display.

## **External JavaScript**

Scripts can also be placed in external files:

### **External file: myScript.js**

function myFunction() {  
  document.getElementById("demo").innerHTML = "Paragraph changed.";  
}

External scripts are practical when the same code is used in many different web pages.

JavaScript files have the file extension **.js**.

To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

### **Example**

<script src="myScript.js"></script>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_whereto_external)

You can place an external script reference in <head> or <body> as you like.

The script will behave as if it was located exactly where the <script> tag is located.

External scripts cannot contain <script> tags.

## **External JavaScript Advantages**

Placing scripts in external files has some advantages:

* It separates HTML and code
* It makes HTML and JavaScript easier to read and maintain
* Cached JavaScript files can speed up page loads

To add several script files to one page  - use several script tags:

### **Example**

<script src="myScript1.js"></script>  
<script src="myScript2.js"></script>

## **External References**

An external script can be referenced in 3 different ways:

* With a full URL (a full web address)
* With a file path (like /js/)
* Without any path

This example uses a **full URL** to link to myScript.js:

### **Example**

<script src="https://www.w3schools.com/js/myScript.js"></script>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_whereto_url)

This example uses a **file path** to link to myScript.js:

### **Example**

<script src="/js/myScript.js"></script>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_whereto_url_relative)

This example uses no path to link to myScript.js:

### **Example**

<script src="myScript.js"></script>

# JavaScript Output

[❮ Previous](https://www.w3schools.com/js/js_whereto.asp)[Next ❯](https://www.w3schools.com/js/js_statements.asp)

## **JavaScript Display Possibilities**

JavaScript can "display" data in different ways:

* Writing into an HTML element, using innerHTML.
* Writing into the HTML output using document.write().
* Writing into an alert box, using window.alert().
* Writing into the browser console, using console.log().

## **Using innerHTML**

To access an HTML element, JavaScript can use the document.getElementById(id) method.

The id attribute defines the HTML element. The innerHTML property defines the HTML content:

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Web Page</h1>  
<p>My First Paragraph</p>  
  
<p id="demo"></p>  
  
<script>  
document.getElementById("demo").innerHTML = 5 + 6;  
</script>  
  
</body>  
</html>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_dom)

Changing the innerHTML property of an HTML element is a common way to display data in HTML.

## **Using document.write()**

For testing purposes, it is convenient to use document.write():

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Web Page</h1>  
<p>My first paragraph.</p>  
  
<script>  
document.write(5 + 6);  
</script>  
  
</body>  
</html>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_write)

Using document.write() after an HTML document is loaded, will **delete all existing HTML**:

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Web Page</h1>  
<p>My first paragraph.</p>  
  
<button type="button" onclick="document.write(5 + 6)">Try it</button>  
  
</body>  
</html>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_write_over)

The document.write() method should only be used for testing.

ADVERTISEMENT

## **Using window.alert()**

You can use an alert box to display data:

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Web Page</h1>  
<p>My first paragraph.</p>  
  
<script>  
window.alert(5 + 6);  
</script>  
  
</body>  
</html>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_alert)

You can skip the window keyword.

In JavaScript, the window object is the global scope object, that means that variables, properties, and methods by default belong to the window object. This also means that specifying the window keyword is optional:

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Web Page</h1>  
<p>My first paragraph.</p>  
  
<script>  
alert(5 + 6);  
</script>  
  
</body>  
</html>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_alert2)

## **Using console.log()**

For debugging purposes, you can call the console.log() method in the browser to display data.

You will learn more about debugging in a later chapter.

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<script>  
console.log(5 + 6);  
</script>  
  
</body>  
</html>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_console)

## **JavaScript Print**

JavaScript does not have any print object or print methods.

You cannot access output devices from JavaScript.

The only exception is that you can call the window.print() method in the browser to print the content of the current window.

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<button onclick="window.print()">Print this page</button>  
  
</body>  
</html>

# JavaScript Statements

[❮ Previous](https://www.w3schools.com/js/js_output.asp)[Next ❯](https://www.w3schools.com/js/js_syntax.asp)

### **Example**

let x, y, z;    // Statement 1  
x = 5;          // Statement 2  
y = 6;          // Statement 3  
z = x + y;      // Statement 4

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_statements)

## **JavaScript Programs**

A **computer program** is a list of "instructions" to be "executed" by a computer.

In a programming language, these programming instructions are called **statements**.

A **JavaScript program** is a list of programming **statements**.

In HTML, JavaScript programs are executed by the web browser.

## **JavaScript Statements**

JavaScript statements are composed of:

Values, Operators, Expressions, Keywords, and Comments.

This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":

### **Example**

document.getElementById("demo").innerHTML = "Hello Dolly.";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_statement)

Most JavaScript programs contain many JavaScript statements.

The statements are executed, one by one, in the same order as they are written.

JavaScript programs (and JavaScript statements) are often called JavaScript code.

## **Semicolons ;**

Semicolons separate JavaScript statements.

Add a semicolon at the end of each executable statement:

### **Examples**

let a, b, c;  // Declare 3 variables  
a = 5;        // Assign the value 5 to a  
b = 6;        // Assign the value 6 to b  
c = a + b;    // Assign the sum of a and b to c

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_statements_semicolon1)

When separated by semicolons, multiple statements on one line are allowed:

a = 5; b = 6; c = a + b;

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_statements_semicolon2)

On the web, you might see examples without semicolons.  
Ending statements with semicolon is not required, but highly recommended.

ADVERTISEMENT

## **JavaScript White Space**

JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.

The following lines are equivalent:

let person = "Hege";  
let person="Hege";

A good practice is to put spaces around operators ( = + - \* / ):

let x = y + z;

## **JavaScript Line Length and Line Breaks**

For best readability, programmers often like to avoid code lines longer than 80 characters.

If a JavaScript statement does not fit on one line, the best place to break it is after an operator:

### **Example**

document.getElementById("demo").innerHTML =  
"Hello Dolly!";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_statements_linebreak)

## **JavaScript Code Blocks**

JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.

The purpose of code blocks is to define statements to be executed together.

One place you will find statements grouped together in blocks, is in JavaScript functions:

### **Example**

function myFunction() {  
  document.getElementById("demo1").innerHTML = "Hello Dolly!";  
  document.getElementById("demo2").innerHTML = "How are you?";  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_statements_blocks)

In this tutorial we use 2 spaces of indentation for code blocks.  
You will learn more about functions later in this tutorial.

## **JavaScript Keywords**

JavaScript statements often start with a **keyword** to identify the JavaScript action to be performed.

Our [Reserved Words Reference](https://www.w3schools.com/js/js_reserved.asp) lists all JavaScript keywords.

Here is a list of some of the keywords you will learn about in this tutorial:

|  |  |
| --- | --- |
| **Keyword** | **Description** |
| var | Declares a variable |
| let | Declares a block variable |
| const | Declares a block constant |
| if | Marks a block of statements to be executed on a condition |
| switch | Marks a block of statements to be executed in different cases |
| for | Marks a block of statements to be executed in a loop |
| function | Declares a function |
| return | Exits a function |
| try | Implements error handling to a block of statements |

# JavaScript Comments

[❮ Previous](https://www.w3schools.com/js/js_syntax.asp)[Next ❯](https://www.w3schools.com/js/js_variables.asp)

JavaScript comments can be used to explain JavaScript code, and to make it more readable.

JavaScript comments can also be used to prevent execution, when testing alternative code.

## **Single Line Comments**

Single line comments start with //.

Any text between // and the end of the line will be ignored by JavaScript (will not be executed).

This example uses a single-line comment before each code line:

### **Example**

// Change heading:  
document.getElementById("myH").innerHTML = "My First Page";  
  
// Change paragraph:  
document.getElementById("myP").innerHTML = "My first paragraph.";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comments1)

This example uses a single line comment at the end of each line to explain the code:

### **Example**

let x = 5;      // Declare x, give it the value of 5  
let y = x + 2;  // Declare y, give it the value of x + 2

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comments5)

## **Multi-line Comments**

Multi-line comments start with /\* and end with \*/.

Any text between /\* and \*/ will be ignored by JavaScript.

This example uses a multi-line comment (a comment block) to explain the code:

### **Example**

/\*  
The code below will change  
the heading with id = "myH"  
and the paragraph with id = "myP"  
in my web page:  
\*/  
document.getElementById("myH").innerHTML = "My First Page";  
document.getElementById("myP").innerHTML = "My first paragraph.";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comments2)

It is most common to use single line comments.  
Block comments are often used for formal documentation.

ADVERTISEMENT

## **Using Comments to Prevent Execution**

Using comments to prevent execution of code is suitable for code testing.

Adding // in front of a code line changes the code lines from an executable line to a comment.

This example uses // to prevent execution of one of the code lines:

### **Example**

//document.getElementById("myH").innerHTML = "My First Page";  
document.getElementById("myP").innerHTML = "My first paragraph.";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comments3)

This example uses a comment block to prevent execution of multiple lines:

### **Example**

/\*  
document.getElementById("myH").innerHTML = "My First Page";  
document.getElementById("myP").innerHTML = "My first paragraph.";  
\*/