

1. DOM innerHTML:

Exercise v1.3 - Mozilla Firefox

www.w3schools.com/js/exercise.asp?filename=exercise_dom_change1

Welcome new customers with an ad on Google. Get Going Today Google

Exercise:

Use the innerHTML property to change the content of the <p> element to "New text!".

Hint

Edit This Code: See Result »

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Display the result here.</p>

<script>
document.getElementById("demo").innerHTML = "New Text!";
</script>
```

Result: Show Answer

New Text!

Exercise - © w3schools.com

15/10/15 - 10:21:22 am

By using `getElementById()` we can select any DOM element from HTML file using its specific ID. Here we can replace contents of html file or contents under any of its DOM elements by using `.innerHTML()` command on the respective element/node of HTML document. We are using `.innerHTML` on <p> element with ID = "demo" to replace contents under <p> tag in given HTML file.

Changing attribute value:

Exercise v1.3 - Mozilla Firefox

www.w3schools.com/js/exercise.asp?filename=exercise_dom_change4

Reach the right customers at the right time with AdWords. Get Going Today Google

Exercise:

Use HTML DOM to change the value of the input's value attribute to "Goodbye".

Hint

Edit This Code: See Result »

```
<!DOCTYPE html>
<html>
<body>

<input type="text" id="myText" value="Hello">

<script>
document.getElementById("myText").value = "Goodbye";
</script>
```

Result: Show Answer

Goodbye

Exercise - © w3schools.com

15/10/15 - 10:22:36 am

Here, by using `.value = "Goodbye"` (syntax: `DOMElement/variable.attribute_name = value_of_attribute`) command we are changing value attribute of <input> element specified with ID = "myText" to "Goodbye".

Onlick event:

Before clicking button:

The screenshot shows a web browser window titled "Exercise v1.3 - Mozilla Firefox". The address bar shows the URL "www.w3schools.com/js/exercise.asp?filename=exercise_dom_events2". The page content includes a Pinpointe advertisement for "DRIP MARKETING CAMPAIGNS" with a "TRY IT FREE" button. Below the ad, the heading "Exercise:" is followed by the instruction: "Use the DOM to assign an onclick event to the <button> element. Clicking the button should trigger displayDate().". A "Hint" button is visible. The "Edit This Code:" section contains the following code:

```
<button id="myBtn">Try it</button>
<p id="demo"></p>
<script>
document.getElementById("myBtn").onclick = displayDate;
function displayDate() {
  document.getElementById("demo").innerHTML = Date();
}
```

The "Result:" section shows a "Try it" button. The "Show Answer" button is also visible. The bottom of the browser window shows the Windows taskbar with the date and time "15/10/15 - 10:25:27 am".

Here by using JavaScript .onclick command on element <button> with ID "myBtn" we are setting an onclick event on <button> to call function displayDate() in javascript. DisplayDate() uses .innerHTML command to replace contents of <p> tag with current Date.

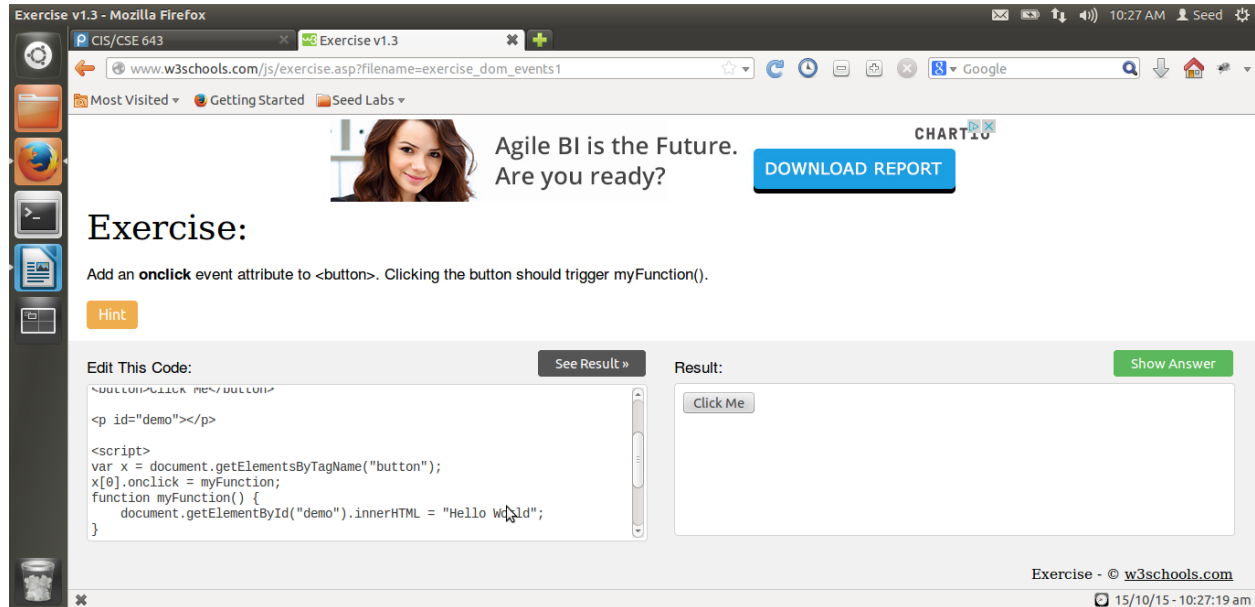
After clicking event:

The screenshot shows the same web browser window as before, but after the "Try it" button has been clicked. The "Result:" section now displays the current date and time: "Thu Oct 15 2015 10:25:42 GMT-0700 (PDT)". The "Try it" button is still visible, but it is now disabled. The "Show Answer" button is also visible. The bottom of the browser window shows the Windows taskbar with the date and time "15/10/15 - 10:25:44 am".

Here, we can see that after clicking on try it button, an onclick event gets invoked and our code makes call to displayDate() function to display current-date in element with id “demo” of HTML.

Event on tag button:

Before clicking button:



The screenshot shows a web browser window titled "Exercise v1.3 - Mozilla Firefox" with the URL "www.w3schools.com/js/exercise.asp?filename=exercise_dom_events1". The page features a banner for "Agile BI is the Future. Are you ready?" with a "DOWNLOAD REPORT" button. Below the banner, the heading "Exercise:" is followed by the instruction: "Add an **onclick** event attribute to <button>. Clicking the button should trigger myFunction()." A "Hint" button is visible. The "Edit This Code:" section contains the following code:

```
<button>Click Me</button>

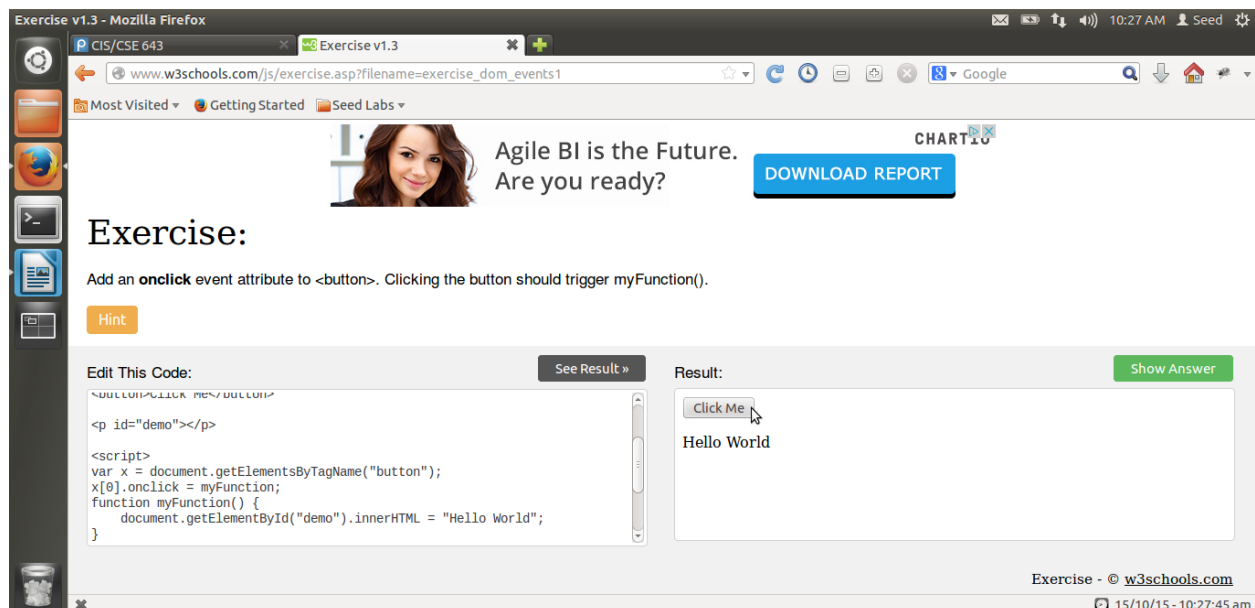
<p id="demo"></p>

<script>
var x = document.getElementsByTagName("button");
x[0].onclick = myFunction;
function myFunction() {
    document.getElementById("demo").innerHTML = "Hello World";
}
```

The "Result:" section shows a "Click Me" button. A "Show Answer" button is located at the top right of the code editor area. The footer indicates "Exercise - © w3schools.com" and the time "15/10/15 - 10:27:19 am".

Here, we are using `getElementsByTagName("button")` command to get list of all elements with tag `<button>` and storing it in variable `x`. Now, by using `x[0]` we select first button element of our html and apply `onclick` event on that using `.onclick` command.

After clicking button:



The screenshot shows the same web browser window as before, but the "Result:" section now displays "Hello World" below the "Click Me" button. The code in the "Edit This Code:" section remains the same. The footer indicates "Exercise - © w3schools.com" and the time "15/10/15 - 10:27:45 am".

Here, we can see that after clicking on the button onclick event got invoked and javascript made call to myFunction() which displays “Hello World” in element specified by ID “demo”.

Running cookie code:



The screenshot shows a web browser interface with two main sections: "Edit This Code:" and "Result:". The "Edit This Code:" section contains the following code:

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to get the cookies associated with the current document.</p>
<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var x = document.cookie;
  document.getElementById("demo").innerHTML = "Cookies associated with this document: " + x;
}
</script>
</body>
</html>
```

The "Result:" section shows the output of the code. It contains a button labeled "Try it" and the following text:

Cookies associated with this document: ASPSESSIONIDCATABBCD=DNLLJOJDMGBHANPCGPBJNAOG; _ga=GA1.2.310225197.1444929250; _gads=ID=303f117d7fe6c390:T=1444929249:S=ALNI_MZTz4PeoUSa_1CHqIXQhIDD0QUKsA; _gat=1

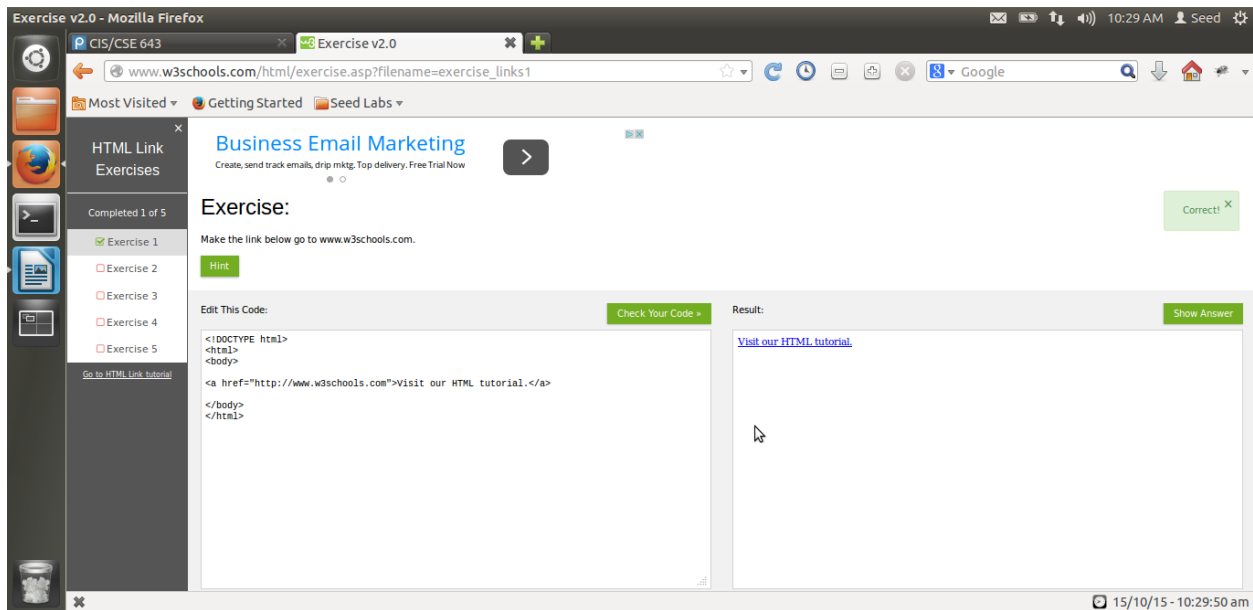
Here, we are getting document cookies using document.cookie command and store them in variable x. Then using .innerHTML tag on element with ID “demo” we display cookies specified with this session.

Here, we can see that in the cookies we have first cookie named ASPSESSIONID... = //something

This is a session ID value for this particular established session with server.

2. HTML tags:

Links:



Here using href attribute in <a> tag element in HTML we can set URL which the link should follow to when clicked.

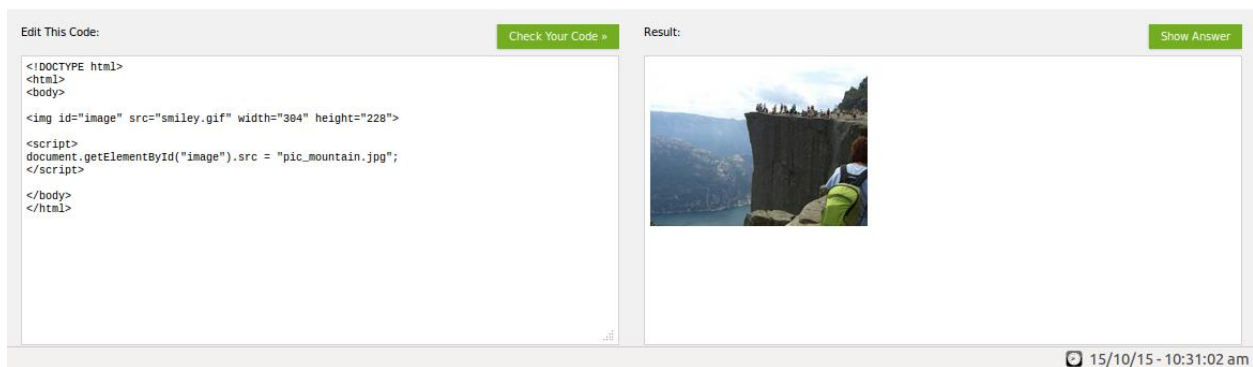
Image:

Changing image source:

Exercise:

Use JavaScript to change the image (the src attribute) to a new image called "pic_mountain.jpg"

Hint



Here using .src source-attribute setting method on img element with "image" ID in javascript we are changing src (source) value of tag to "pic_mountains.jpg".

Iframe:

Exercise:

Correct! X

Create an iframe with a URL address that goes to <http://www.w3schools.com>.

Hint

Edit This Code:

Check Your Code >

Result:

Show Answer

```
<!DOCTYPE html>
<html>
<body>
<iframe src="http://www.w3schools.com"/>
</body>
</html>
```

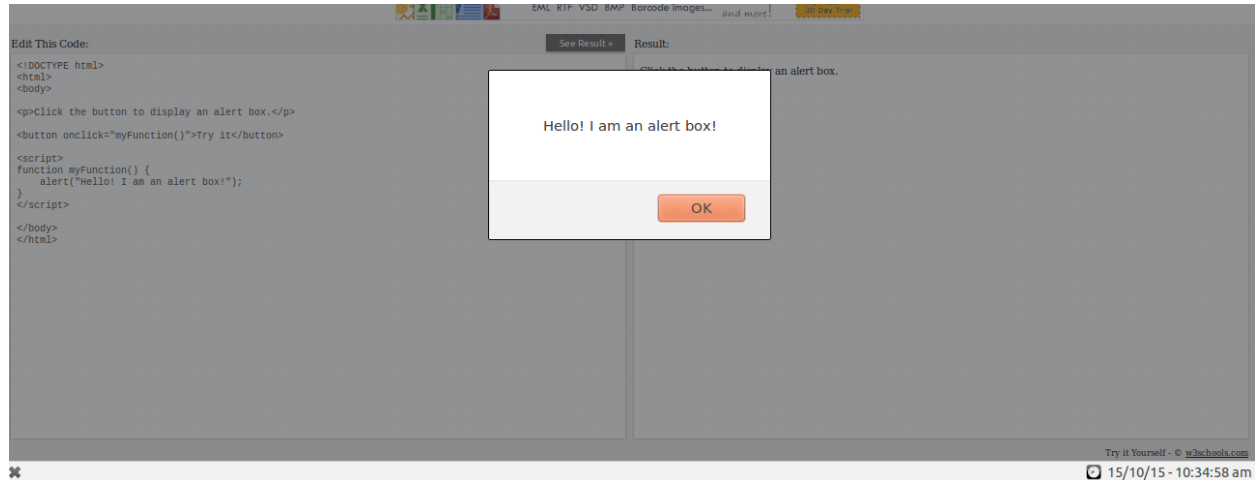


15/10/15 - 10:34:09 am

Here we are setting up the URL to be displayed in iframe element of HTML file using src attribute and URL as value for src.

3. BOM:

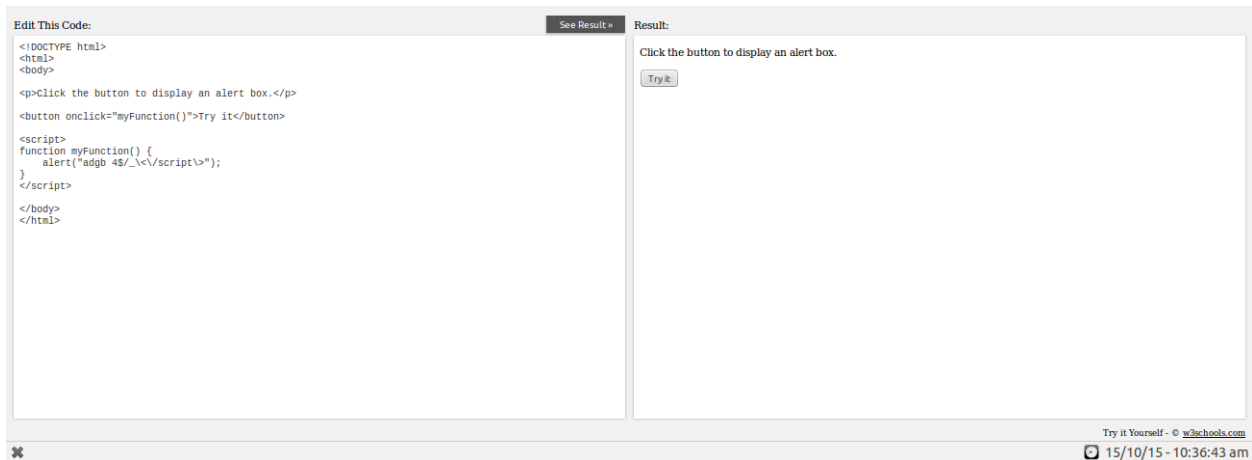
Running alert code:



Here we are using alert() method of JavaScript in function myFunction() to show an alert dialog box on browser when button “Try it” is clicked. When try it button is clicked, onclick attribute of the button gets invoked which makes call to javascript function myFunction().

6. JS Strings:

Code for alert:



The screenshot shows a web application interface with two main sections: "Edit This Code:" and "Result:". The "Edit This Code:" section contains the following HTML and JavaScript code:

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to display an alert box.</p>
<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  alert("adgb 4$/_</script>");
}
</script>
</body>
</html>
```

The "Result:" section shows the output of the code, which is a button labeled "Try it". Below the button, the text "Click the button to display an alert box." is displayed. The bottom of the screenshot shows the footer text "Try it Yourself - © w3schools.com" and the date "15/10/15 - 10:36:43 am".

As browser interprets `<`, `>`, `/` characters and strings of any tags (such as `<script>`, `</script>`, `<html>`, `<body>`, etc.) as special characters, we cannot directly write them in the JavaScript when we want to use them in strings. For this purpose we precede them with backslash character `"\"`, so that browser does not parse them as special characters and thinks of them as just normal string characters.

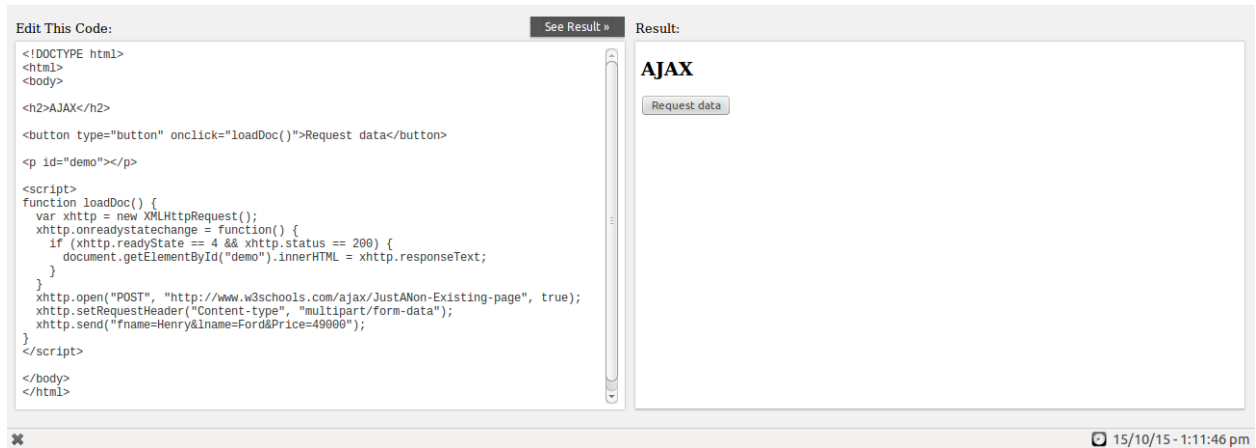
Onclick alert:



The screenshot shows the same web application interface as the previous one, but with an alert dialog box displayed in the center. The dialog box contains the text "adgb 4\$/_</script>" and an "OK" button. The background of the web application is dimmed. The bottom of the screenshot shows the footer text "Try it Yourself - © w3schools.com" and the date "15/10/15 - 10:37:08 am".

Here, we can see that after using `"\"` before all special characters such as `"<, >, /"`, which are part of HTML syntax, browser parses them and turns them into string characters. By using `alert()` command we can display a dialog box displaying that string.

Ajax:



The screenshot displays a web browser window with two main panes. The left pane, titled 'Edit This Code:', contains the following HTML and JavaScript code:

```
<!DOCTYPE html>
<html>
<body>

<h2>AJAX</h2>

<button type="button" onclick="loadDoc()">Request data</button>

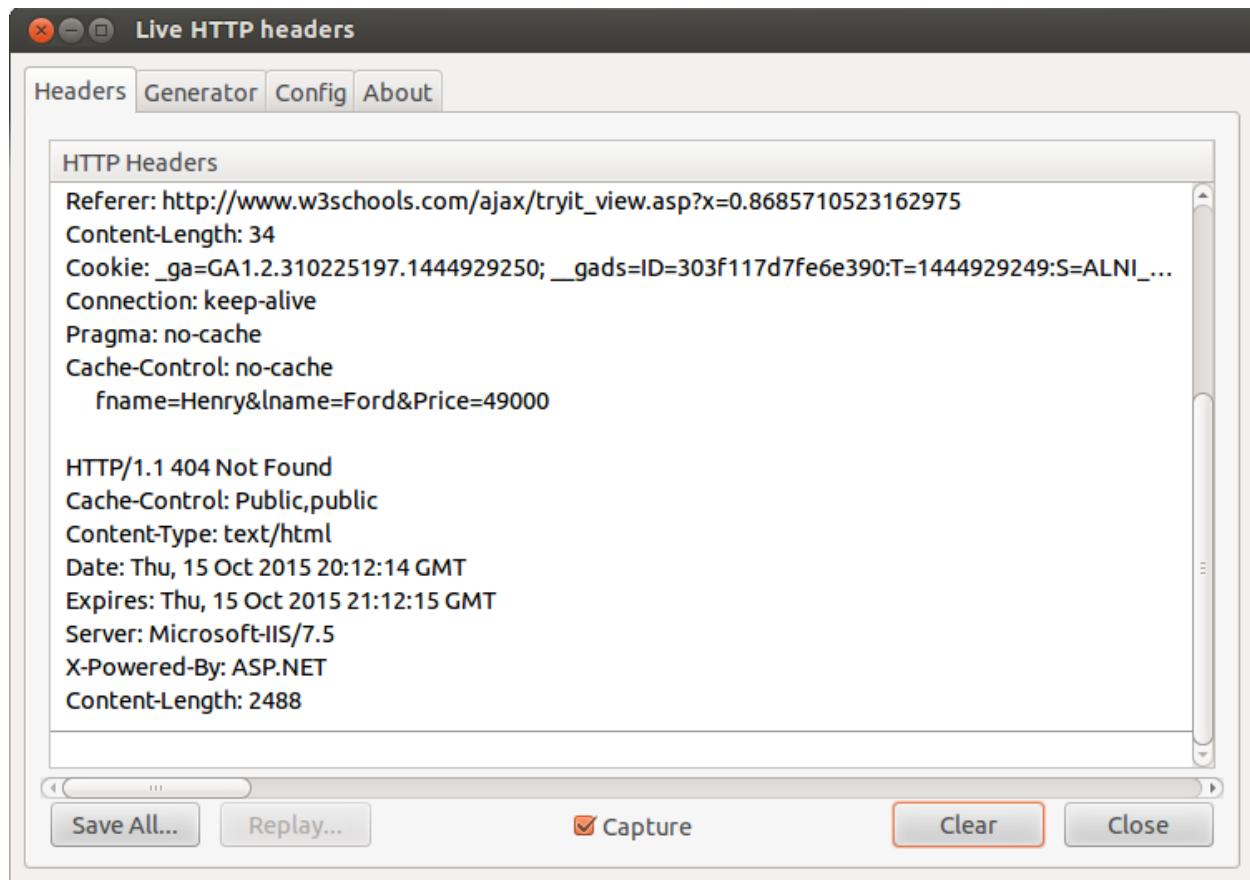
<p id="demo"></p>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
      document.getElementById("demo").innerHTML = xhttp.responseText;
    }
  };
  xhttp.open("POST", "http://www.w3schools.com/ajax/JustANon-Existing-page", true);
  xhttp.setRequestHeader("Content-type", "multipart/form-data");
  xhttp.send("fname=Henry&lname=Ford&Price=49000");
}
</script>
</body>
</html>
```

The right pane, titled 'Result:', shows the output of the Ajax request. It displays the word 'AJAX' in a large, bold font. Below it, there is a button labeled 'Request data'. The browser's status bar at the bottom indicates the date and time: '15/10/15 - 1:11:46 pm'.

Here we are creating a variable xhttp which is an XMLHttpRequest. By using open() method for this request we are providing 3 arguments which defines type of HTTPRequest i.e. GET/POST, URL where the file is located and true or false value for asynchronous flag. In this case we are sending an asynchronous POST request to URL <http://www.w3schools.com/ajax/JustANon-Existing-page>. We are setting up content type as multipart from-data in HTTP request header with command setRequestHeader() and passing content as fname=Henry, lname=Ford and Price=49000 to given URL with command send().

HTTP Live Header:



From the content of Live HTTP Headers we can confirm that we have sent an asynchronous POST request to URL <http://www.w3schools.com/ajax/JustANon-Existing-page> with content fname=Henry, lname=Ford and Price=49000. Our request sent back a HTTP 404 Error as the URL which we provided does not exist.

```
POST /ajax/JustANonExistingPage HTTP/1.1
Host: www.w3schools.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) Gecko/20100101 Firefox/23.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; charset=UTF-8
```

And from above screenshot we can confirm that the content-type for our POST request was "multipart/form-data".