Wenbin Li SUID: 687150735

Lab 6: DOM, HTML,JavaScript

Task1:

Exercise:

Correct! ×

Make the link below go to www.w3schools.com.

Hint

Edit This Code: Check Your Code »

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

<a href="http://www.w3schools.com">Visit our HTML
tutorial.</a>

</body>
</html>
```

Result: Show Answer

Visit our HTML tutorial.

Fig 1.1 test link

Exercise:

Correct! ×

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

Hint

Edit This Code: Check Your Code »

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

<img id="image" src="smiley.gif" width="304"
height="228">

<script>
// Add code here
document.getElementById('image').src =
'pic_mountain.jpg';
</script>

</body>
</html>
```

Result: Show Answer

Fig 1.2 test image

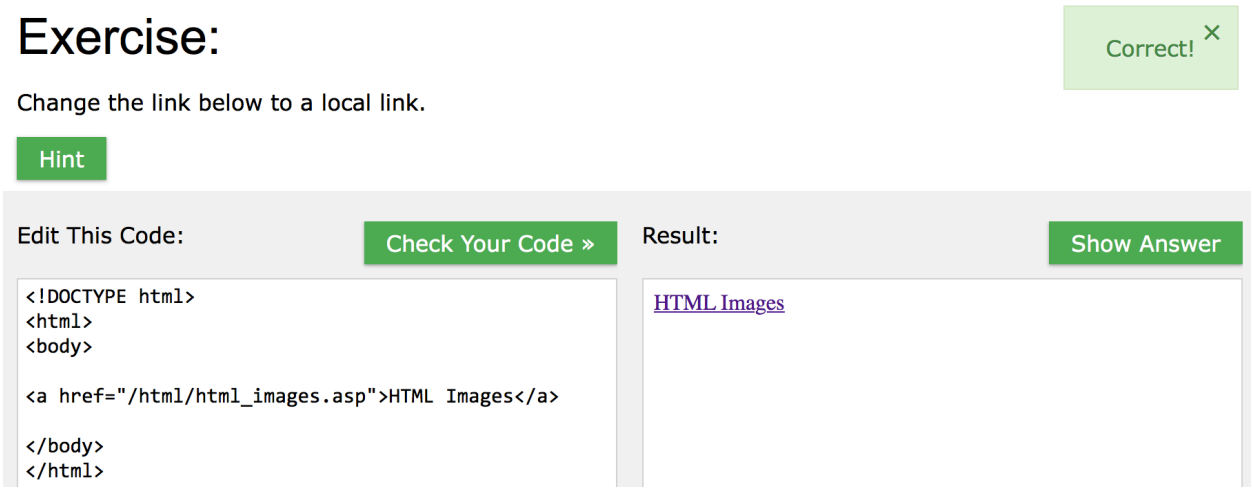

Fig 1.3 test local image

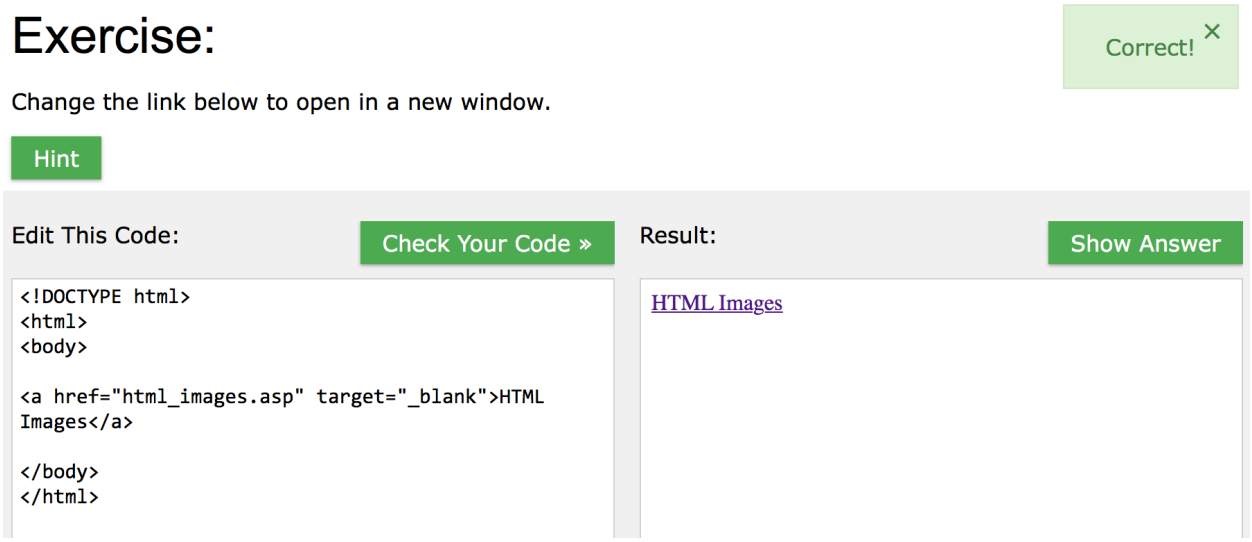

Fig 1.4 test target="_blank" in link

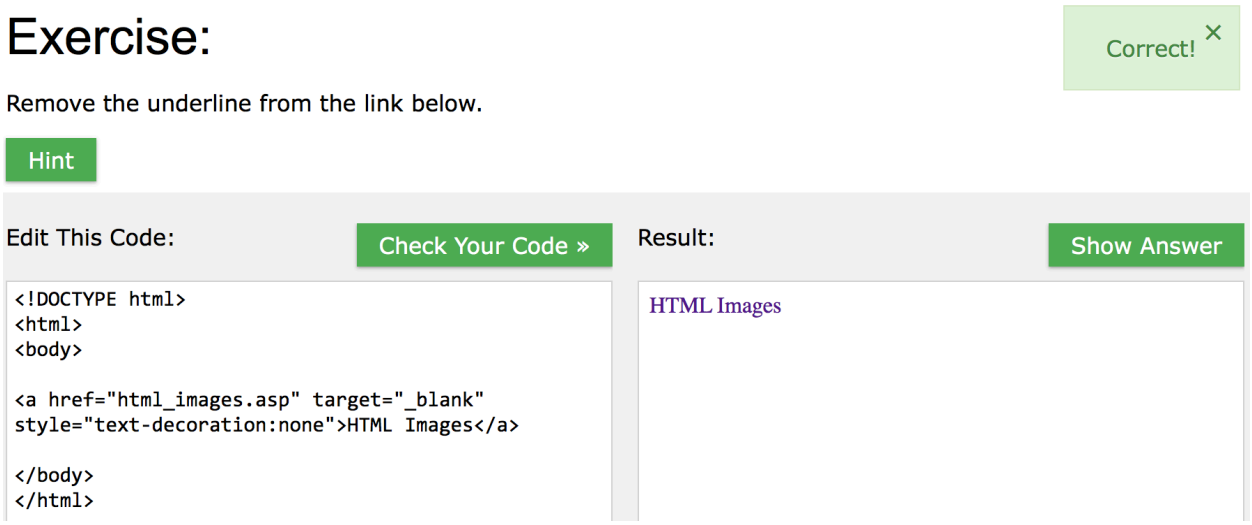

Fig 1.5 test style attribute

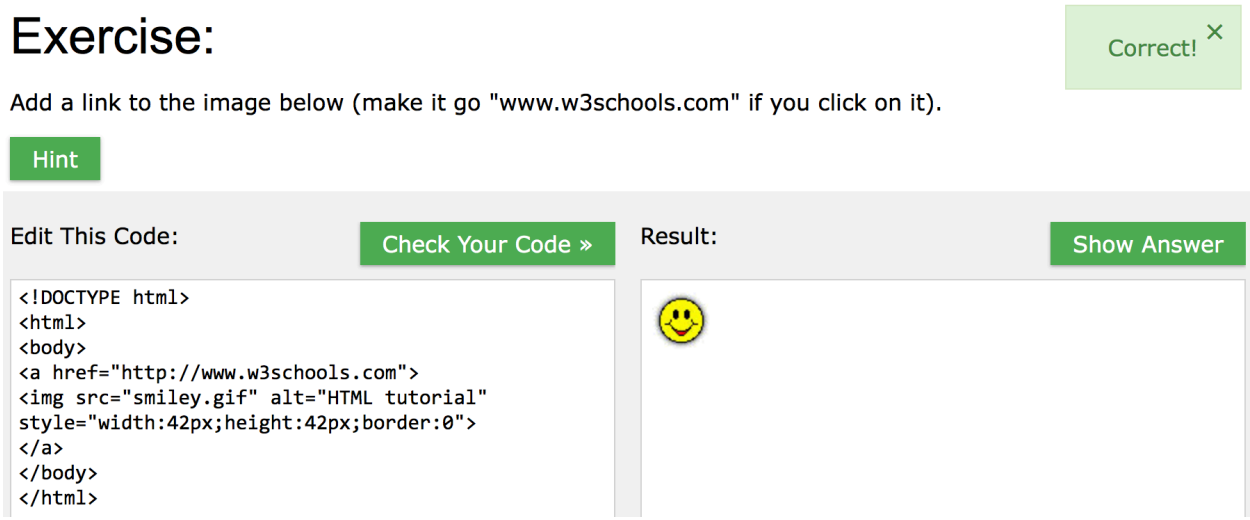

Fig 1.6 test link of image

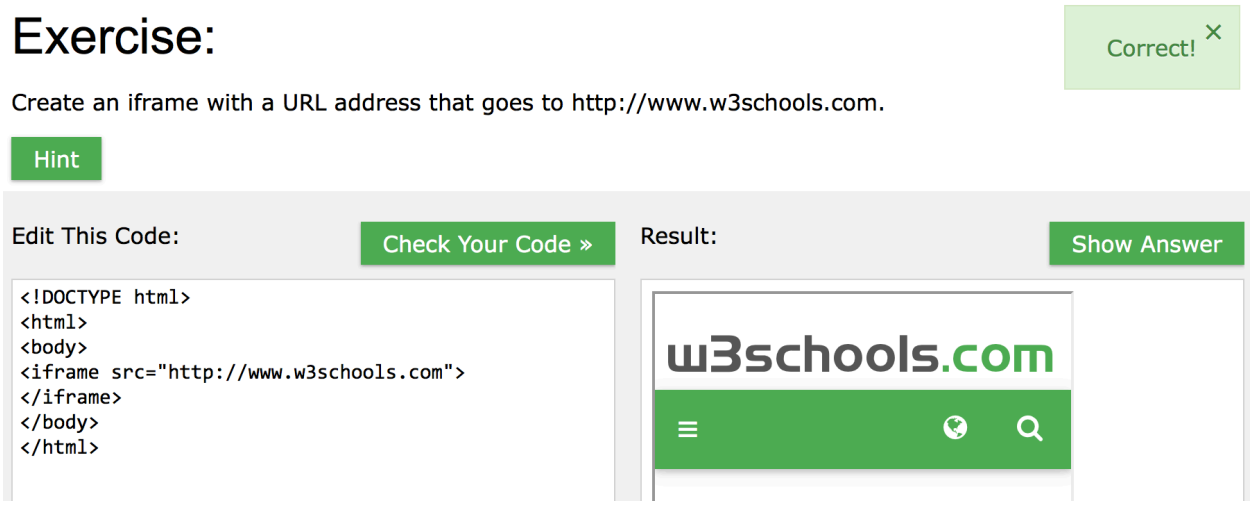

Fig 1.7 test iframe

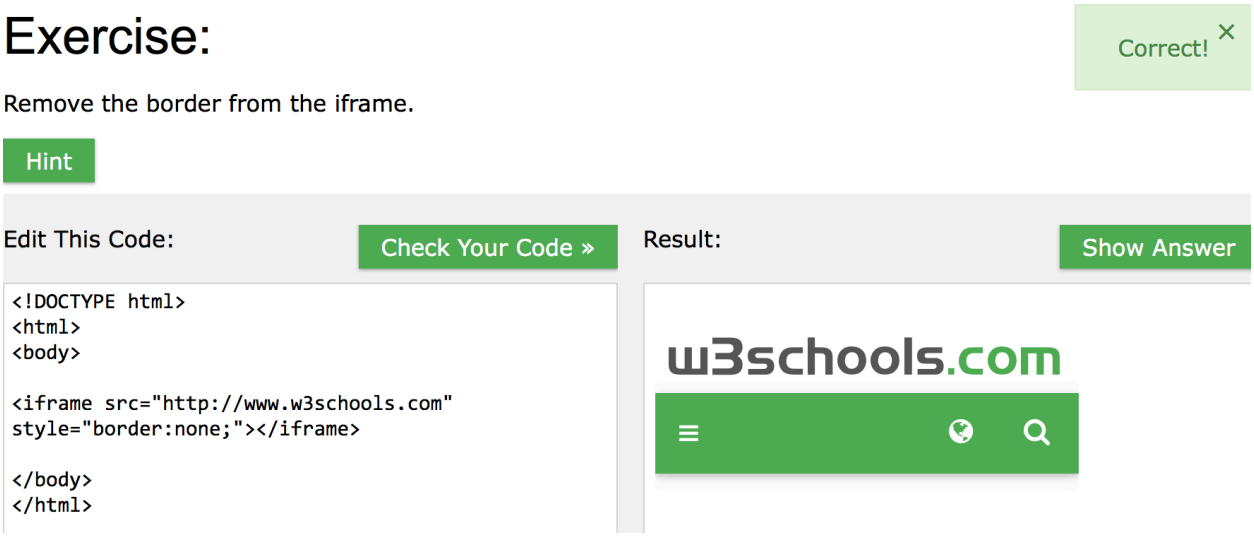

Fig 1.8 test style in iframe

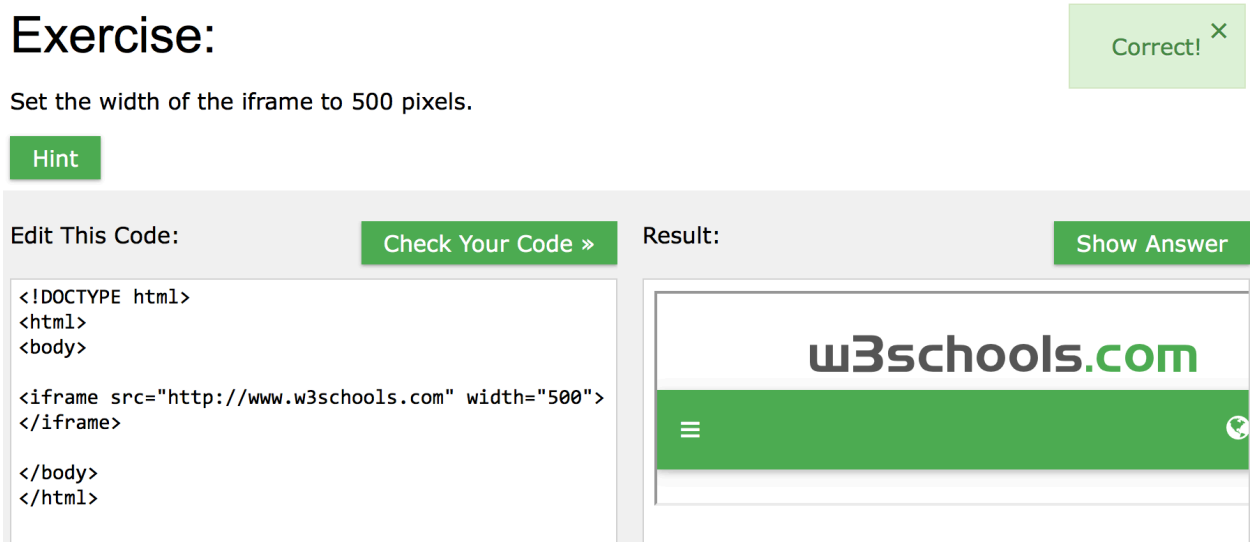

Fig 1.9 test width attribute in iframe

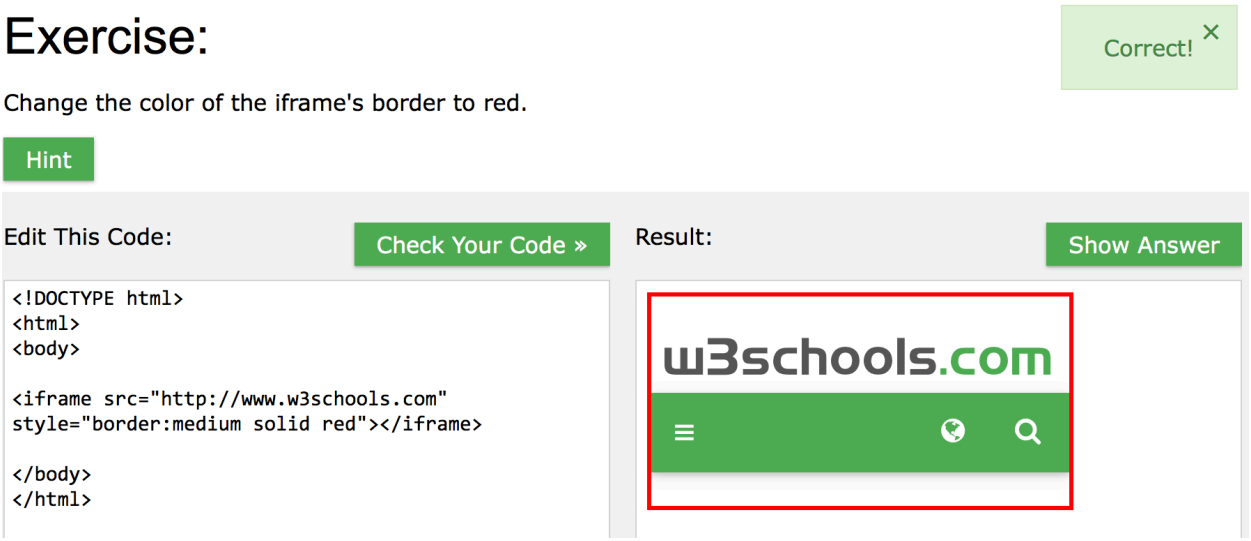

Fig 1.10 test style in iframe

Fig 1.11 test firebug inspect feature

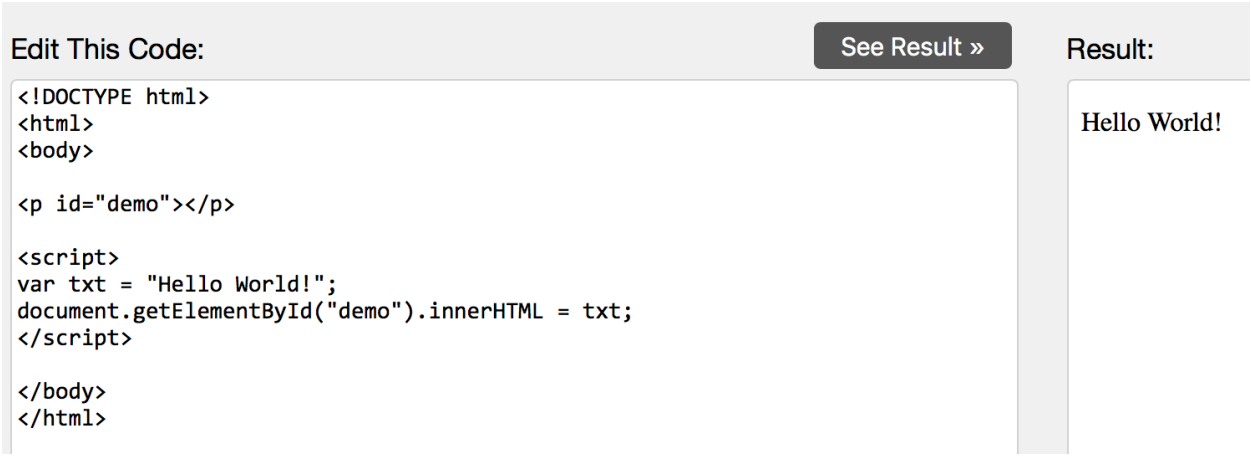

Fig 1.12 test JS output

Use the length property to display the length of the txt variable's value.

Edit This Code: See Result »

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

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

<script>
var txt = "Hello World!";
document.getElementById("demo").innerHTML = txt.length;
</script>

</body>
</html>
```

Result:

12

Fig 1.13 test String length attribute

The string below is broken - use escape characters to display the text correctly.

Hint

Edit This Code: See Result »

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

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

<script>
document.getElementById("demo").innerHTML = "We are \"Vikings\".";
</script>

</body>
</html>
```

Result:

We are "Vikings".

Fig 1.14 test \ character usage of string output

Concatenate the two strings to display "Hello World!".

Hint

Edit This Code: See Result »

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

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

<script>
var str1 = "Hello ";
var str2 = "World!";
document.getElementById("demo").innerHTML = str1 + " " + str2;
</script>

</body>
</html>
```

Result:

Hello World!

Fig 1.15 test string concat

Display the position of the first occurrence of "World" in the variable txt.

Hint

Edit This Code: See Result »

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

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

<script>
var txt = "Hello World";
document.getElementById("demo").innerHTML = txt.indexOf("World");
</script>

</body>
</html>
```

Result:

6

Fig 1.16 test string method indexOf()

Use the slice() method to display only "Banana,Kiwi".

Edit This Code: See Result »

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

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

<script>
var str = "Apple,Banana,Kiwi";
document.getElementById("demo").innerHTML = str.slice(6);
</script>

</body>
</html>
```

Result:

Banana,Kiwi

Fig 1.17 test slice method usage in string

Use the replace() method to replace "World" with "Universe".

Edit This Code: See Result »

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

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

<script>
var txt = "Hello World";
document.getElementById("demo").innerHTML = txt.replace(/World/g,
"Universe");
</script>

</body>
</html>
```

Result:

Hello Universe

Fig 1.18 test string replace method

Observation and Explanation:

1. Like fig 1.1, link_name is link structure.
2. Like fig 1.2 <img src=”target”> is image structure. The image id = “image”, so I could use document.getElementById(“image”).src=”somepic” to change iamge src to “somepic”.
3. As link, href could be used as local relative path.
4. The attribute target=”_blank” could make link target opened in a new window, like fig 1.4.
5. The style attribute could be modified for link, like fig 1.5.
6. Link could modify img object like fig 1.6.
7. <iframe src=”url”> is structure of iframe like fig 1.7.
8. like fig 1.8, 1.9, 1.10, iframe could modify with style.
9. Fig 1.11 displays that I used firebug to inspect alice’s profile image.

10. Like fig 1.12, I used document.getElementById("demo").innerHTML = text to change contents for paragraph with id = "demo".
11. Fig 1.13 used txt.length attribute.
12. Fig 1.14 used '\' as escape character.
13. indexOf() is a method for string to find the 1st target position, like fig 1.16.
14. String::slice() is a method to slice the String obj like fig 1.17.
15. String method replace () could replace characters in specifc string object, like fig 1.18.

Task2:

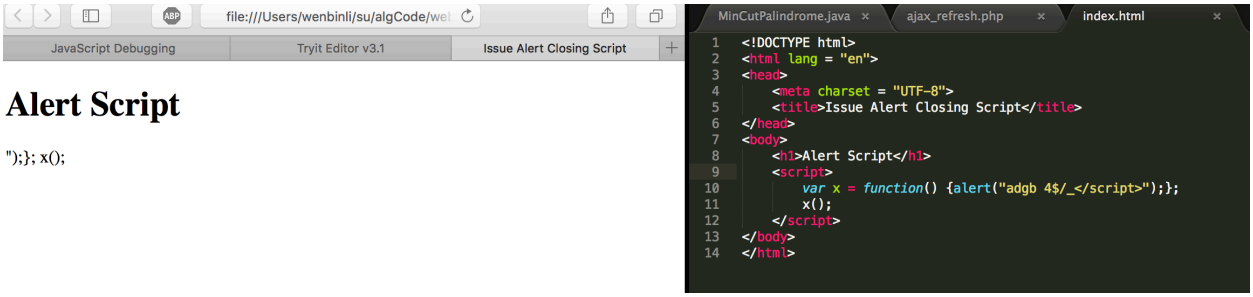

Fig 2.1 error output

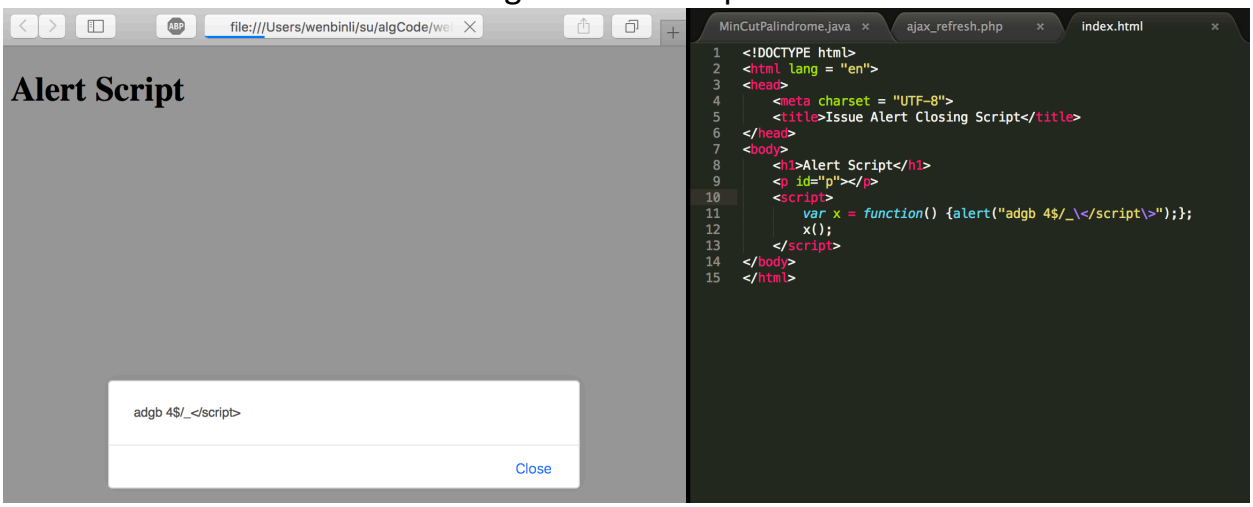

Fig 2.2 fix error

Observation:

1. As fig 2.1, there was no alert window coming out and extra output ');}; x();' was printed out in browser.
2. As fig 2.2, output was what I wanted.

Explanation:

1. Like fig 2.1, when render engine parsed the string of "</script>", the string

would be recognized as closing tag of JavaScript. So that alert sentence was broken up, and it would not alert anything. After </script> ');}; x();' would be rendered as a part of body content, then these characters were printed out in browser.

2. As Fig 2.2, I added '\' in '</script>' to make '\</script\> '. Then render would parse '\</script\> ' as normal characters ''</script>'' rather than JavaScript closing tag.

Task 3:

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>
// Add code here
document.getElementById("demo").innerHTML =  "New text!";
</script>

</body>
</html>
```

Result:

New text!

Fig 3.1 Use the innerHTML property to change the content of the <p> element to "New text!"

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>
// Add code here
document.getElementById("myText").value = "Goodbye";
</script>

</body>
</html>
```

Result:

Goodbye

Fig 3.2 Use HTML DOM to change the value of the input's value attribute to "Goodbye".

Use the DOM to assign an onclick event to the <button> element. Clicking the button should trigger displayDate().

Hint

Edit This Code: See Result »

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

<button id="myBtn">Try it</button>

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

<script>
// Add code here
document.getElementById("myBtn").onclick = displayDate;
function displayDate() {
    document.getElementById("demo").innerHTML = Date();
}
</script>

</body>
</html>
```

Result:

Try it

Tue Oct 25 2016 01:11:52 GMT-0400 (EDT)

Fig 3.3 Use the DOM to assign an onclick event to the <button> element. Clicking the button should trigger displayDate().

Add an **onclick** event attribute to <button>. Clicking the button should trigger myFunction().

Hint

Edit This Code: See Result »

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

<button onclick="myFunction()">Click Me</button>

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

<script>
function myFunction() {
    document.getElementById("demo").innerHTML = "Hello World";
}
</script>

</body>
</html>
```

Result:

Click Me

Hello World

Fig 3.4 Add an onclick event attribute to <button>. Clicking the button should trigger myFunction()

Run » Result Size: 582 ›

```
<!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>
```

Click the button to get the cookies associated with the current document.

Try it

Cookies associated with this document:
ASPSESSIONIDSQCCTTTA=BEDMBDCDJJDJJKJEKFJGCGGH; __gads=ID=d5a809cedb36695d:T=1477358797:S=ALNI_MYmmkt6CO91ZbH-BXUlmmmFJt7kWQ; _ga=GA1.2.763069848.1466467053; _gat=1

Fig 3.5 HTML DOM Cookie Property

```
>>> document.getElementsByTagName("script")
HTMLCollection[ script jquery-1.6.4.min.js, script jquery-u...6.min.js, script elgg.1410864370.js, script ]
>>> document.getElementsByTagName("div")
HTMLCollection[ div.elgg-page, div.elgg-page-messages, div.elgg-page-topbar, div.elgg-inner, div.elgg-page-header, div.elgg-inner, div.elgg-page-body, div.elgg-inner, div.elgg-layout, div.elgg-body, div.elgg-widget-add-control, div#widgets-add-panel.elgg-widgets-add-panel, div.profile, div.elgg-inner, div#profile-owner-block, div.elgg-avatar, div#profile-details.elgg-body, div#elgg-widget-col-1.elgg-col-1of3, div#elgg-widget-col-2.elgg-col-1of3, div#elgg-widget-col-3.elgg-col-1of3, div#elgg-widget-loader.elgg-ajax-loader, div.elgg-page-footer, div.elgg-inner, div.mts ]
>>>
```

Fig 3.6 Print all the script, div tags in firebug console

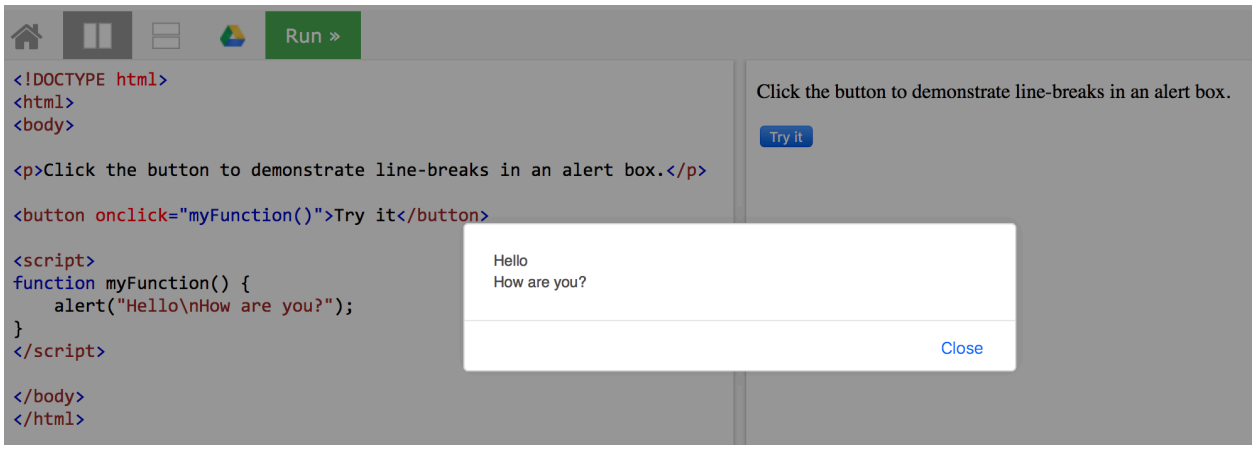

Fig 3.7 how the message in browser

Observation and Explanation:

1. As fig 3.1, document.getElementById() could get specific node for the DOM, and innerHTML is contents of this node. So that I could change contents of displaying.
2. As fig 3.2, I could use document.getElementById() to get node, and then specify the value of input.
3. As fig 3.3 and fig 3.4, I specified onclick attribute with method document.getElementById() to trigger a function.
4. As fig 3.5, I could use document.cookie to get cookie contents of this DOM.
5. Divs = document.getElementsByTagName("div") could return Divs array as fig 3.6, and the same as document.getElementsByTagName("script").
6. As fig 3.7, alert() method is used for alert message in browers. '\' to make a break line. Alert is a BOM method.

Task 4:

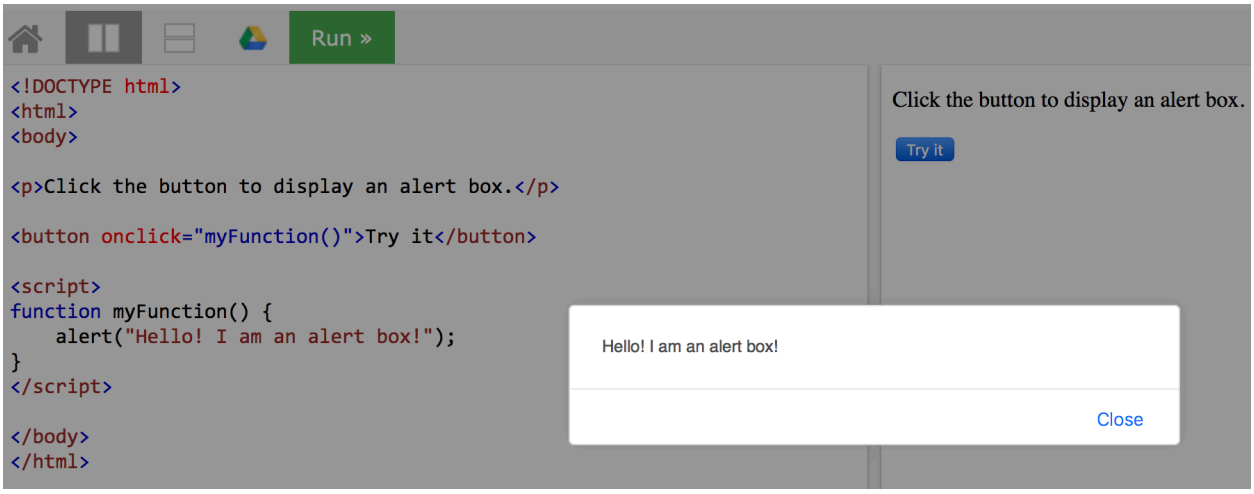

Fig 4.1 Alert Information

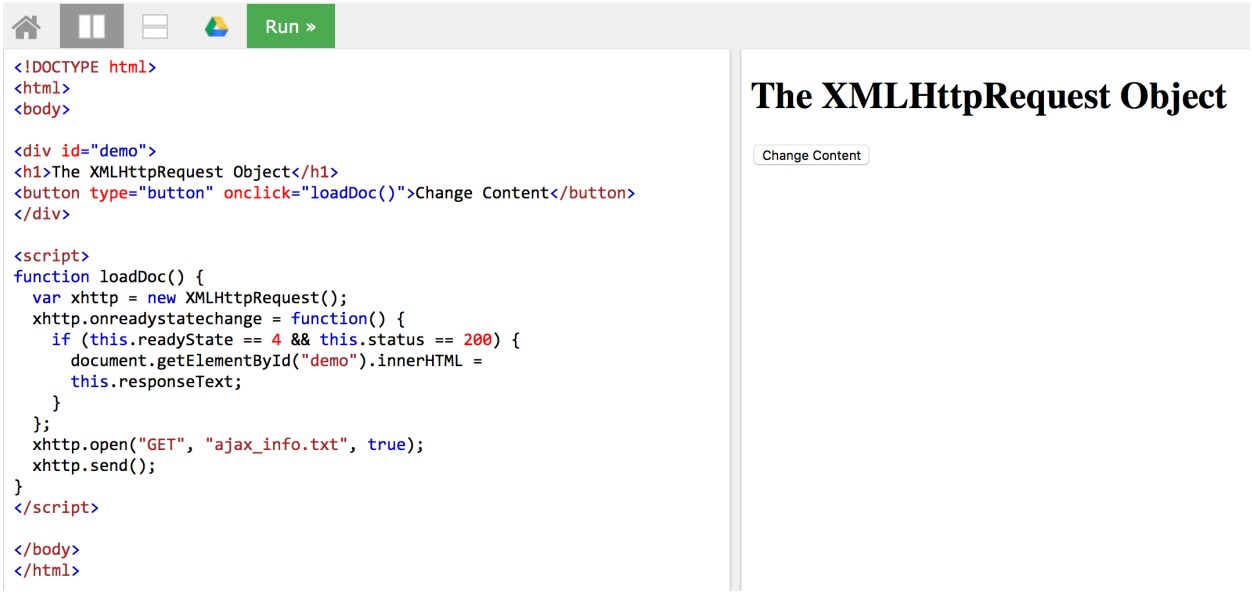

Fig 4.2 How to use Ajax to send the http request

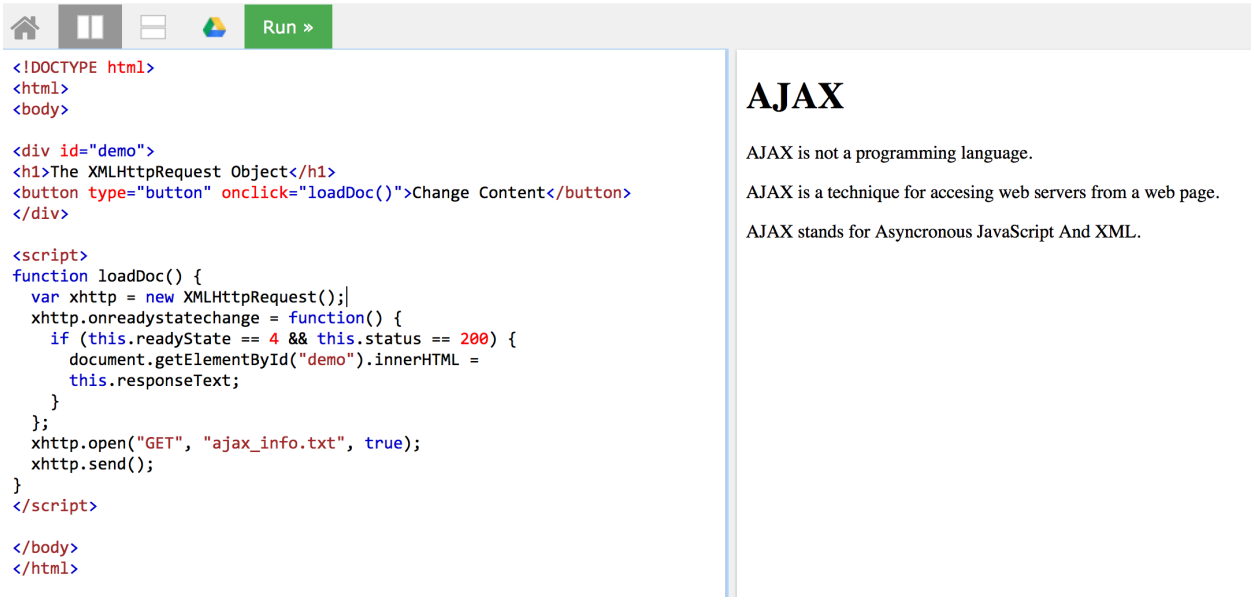

Fig 4.3 How to use Ajax to send the http request

Observation and Explanation:

1. As fig 4.1 alert method was used to alert a message in browser. Alert() is a browser API method.
2. As fig 4.2 and 4.3, this a get request method for Ajax. In JS and function loadDoc(), xhttp is created by 'new XMLHttpRequest()'. Because it is async method, xhttp.onreadystatechange = function() was defined. This.readystate == 4 means that browser has got reply. This.status == 200 means that it is a ok status, then innerHTML is changed to responseText. In xhttp.open("GET", "url", true) is to specify request type, url in server, and if it is async.

Task 5:

Run »

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

<h1>The XMLHttpRequest Object</h1>

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

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

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "demo_get2.asp?fname=Henry&lname=Ford", true);
  xhttp.send();
}
</script>

</body>
</html>
```

Fig 5.1 Ajax get request

Run »

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

<h1>The XMLHttpRequest Object</h1>

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

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

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("POST", "demo_post2.asp", true);
  xhttp.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
  xhttp.send("fname=Henry&lname=Ford&name=Good");
}
</script>

</body>
</html>
```

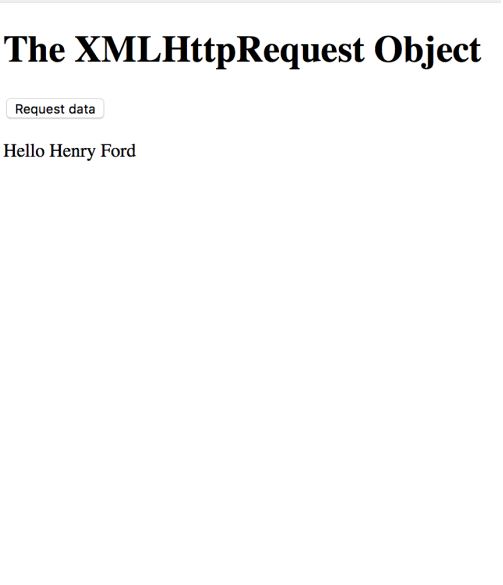

Fig 5.2 Ajax post request

Fig 5.3 login Alice's account

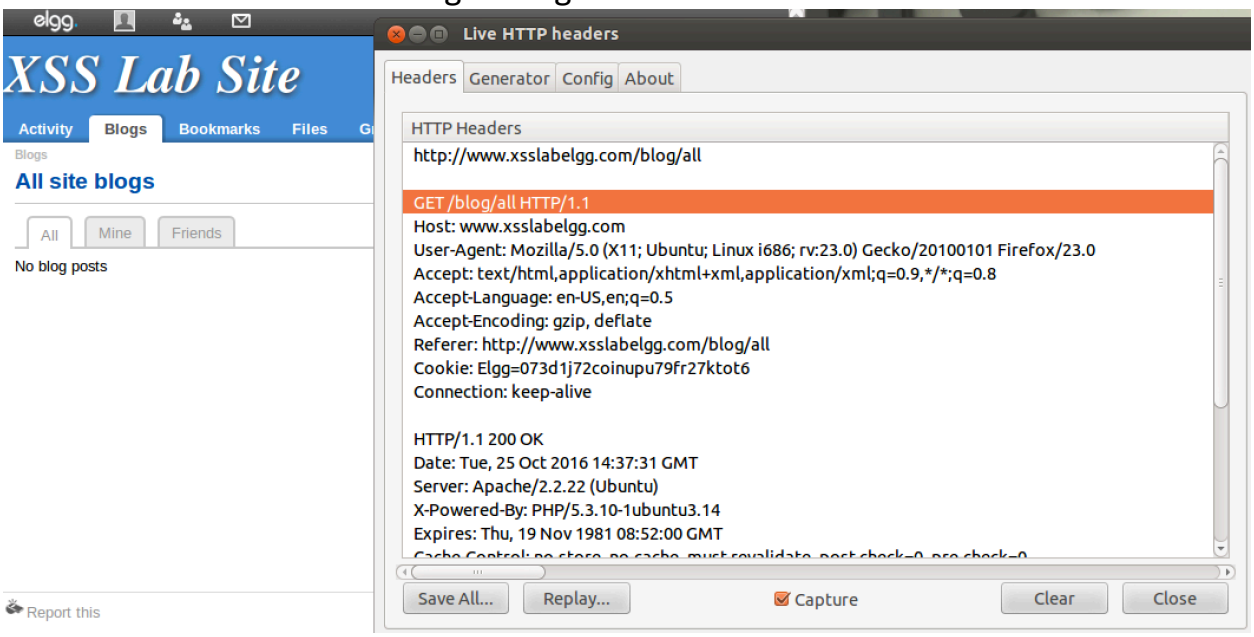

Fig 5.4 live http headers capture get request

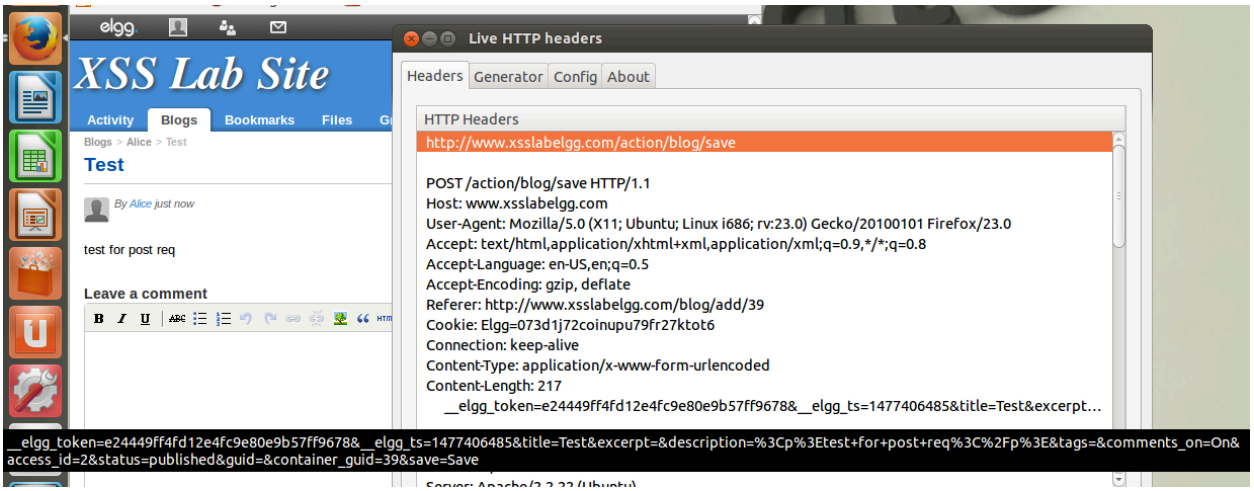

Fig 5.5 Live http headers capture post request

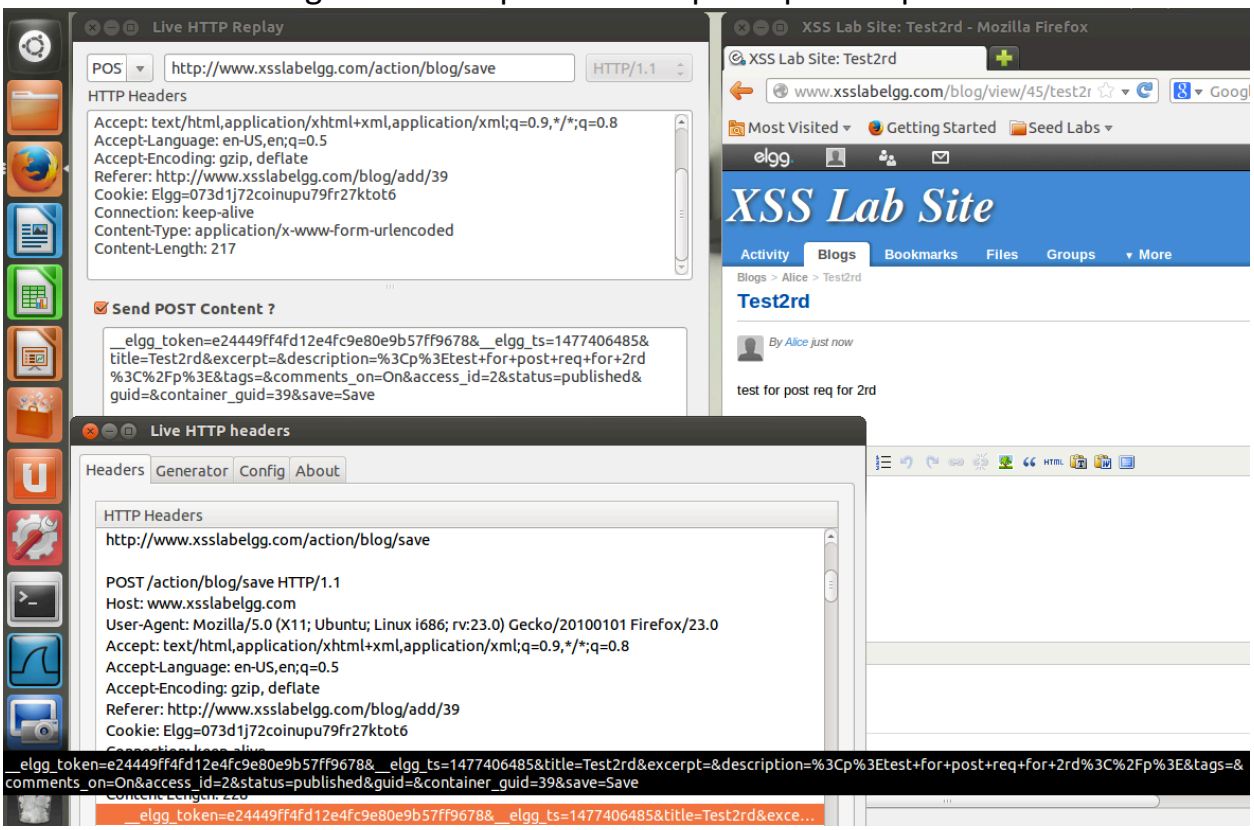

Fig 5.6 Live http headers capture replay the post request

Observation and Explanation:

1. As fig 5.1, this is a get request. In xhttp.open("GET", "url?parameters", true) is used for defining request type, url and parameters, and whether it is ascyn. In JS and function loadDoc(), xhttp is created by 'new XMLHttpRequest()'. Because it is async method, xhttp.onreadystatechange = function() was defined. This.readystate == 4 means that browser has got reply. This.status == 200 means that it is an ok status, then innerHTML is changed to responseText.

2. As fig 5.2, this is a post request. The difference between GET, like fig 5.1, and POST, is that xhttp is opened with POST tyoe, and xhttp.send() is used to send some contents including formed contents, and header attributes.
3. I logged in alice account, user name is alice, and pw is seedalice, like fig 5.3
4. Fig 5.4 shows a get request.
5. Fig 5.5 shows a POST request. The difference between fig 5.5 and fig 5.4 is that fig 5.5 sent contents. The contents are encoded, as the picture showing.
6. As fig 5.6, I used replay function of live http headers to fill a post request to post a blog ‘Test2rd’. Contents and headers is organized with encoded type.