

Lab 6: DOM, HTML, JavaScript

Task1:

Exercise:

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

Correct! 

Hint

Edit This Code:

Check Your Code »

Result:

Show Answer

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

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

</body>
</html>
```

[Visit our HTML tutorial.](http://www.w3schools.com)

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 »

Result:

Show Answer

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



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

</body>
</html>
```



Fig 1.2 test image

Exercise:

Correct! 

Change the link below to a local link.

Hint

Edit This Code:

[HTML Images](/html/html_images.asp)

Result:

Show Answer

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

<a href="/html/html_images.asp">HTML Images</a>

</body>
</html>
```

[HTML Images](#)

Fig 1.3 test local image

Exercise:

Correct! 

Change the link below to open in a new window.

Hint

Edit This Code:

[HTML Images](html_images.asp)

Result:

Show Answer

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

<a href="html_images.asp" target="_blank">HTML
Images</a>

</body>
</html>
```

[HTML Images](#)

Fig 1.4 test target="_blank" in link

Exercise:

Correct! 

Remove the underline from the link below.

Hint

Edit This Code:

[HTML Images](html_images.asp)

Result:

Show Answer

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

<a href="html_images.asp" target="_blank"
style="text-decoration:none">HTML Images</a>

</body>
</html>
```

[HTML Images](#)

Fig 1.5 test style attribute

Exercise:

Correct! 

Add a link to the image below (make it go "www.w3schools.com" if you click on it).

Hint

Edit This Code:

Check Your Code »

Result:

Show Answer

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

</a>
</body>
</html>
```



Fig 1.6 test link of image

Exercise:

Correct! 

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">
</iframe>
</body>
</html>
```



w3schools.com



Fig 1.7 test iframe

Exercise:

Correct! 

Remove the border from the iframe.

Hint

Edit This Code:

Check Your Code »

Result:

Show Answer

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

<iframe src="http://www.w3schools.com"
style="border:none;"></iframe>

</body>
</html>
```



w3schools.com



Fig 1.8 test style in iframe

Exercise:

Correct! 

Set the width of the iframe to 500 pixels.

Hint

Edit This Code:

Check Your Code »

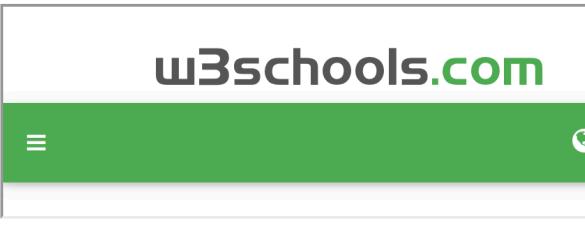
Result:

Show Answer

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

<iframe src="http://www.w3schools.com" width="500">
</iframe>

</body>
</html>
```



w3schools.com



Fig 1.9 test width attribute in iframe

Exercise:

Correct! 

Change the color of the iframe's border to red.

Hint

Edit This Code:

[Check Your Code »](#)

Result:

[Show Answer](#)

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

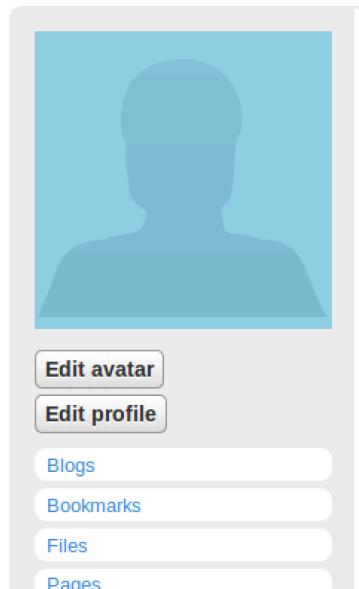
<iframe src="http://www.w3schools.com"
style="border:medium solid red"></iframe>

</body>
</html>
```



w3schools.com

Fig 1.10 test style in iframe



Alice

```
je elgg-page-default">
  -page-messages">
  -page-topbar">
  -page-header">
  -page-body">
  elgg-inner">
  :s="elgg-layout elgg-layout-one-column
  >
  class="elgg-body elgg-main">
  div class="elgg-widget-add-control">
  div id="widgets-add-panel" class="elgg-
  widgets-add-panel hidden clearfix">
  div class="profile elgg-col-2of3">
    <div class="elgg-inner clearfix">
      <div id="profile-owner-block">
        <div class="elgg-avatar elgg-avatar-
        large">
          <a>
            <img class="" style="background:
              url(http://www.xsslabelgg.com/_graphics
              /icons/user/defaultlarge.gif)
              no-
              c='
              /html/body/div/div[4]/div/div/div[3]/div/div/a/
              /spacer.gif">
          </a>
        </div>
      </div>
    </div>
  </div>
  <div class="elgg-body elgg-main">
    <div class="elgg-widget-add-control">
      <div id="widgets-add-panel" class="elgg-
      widgets-add-panel hidden clearfix">
        <div class="profile elgg-col-2of3">
          <div class="elgg-inner clearfix">
            <div id="profile-owner-block">
              <div class="elgg-avatar elgg-avatar-
              large">
                <a>
                  <img class="" style="background:
                    url(http://www.xsslabelgg.com/_graphics
                    /icons/user/defaultlarge.gif)
                    no-
                    c='
                    /html/body/div/div[4]/div/div/div[3]/div/div/a/
                    /spacer.gif">
                </a>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Fig 1.11 test firebug inspect feature

Edit This Code:

[See Result »](#)

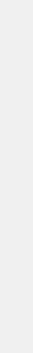
Result:

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

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

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

</body>
</html>
```



Hello World!

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 »

Result:

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

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

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

</body>
</html>
```

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 »

Result:

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

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

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

</body>
</html>
```

We are "Vikings".

Fig 1.14 test \ character usage of string output

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

Hint

Edit This Code:

```
<!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>
```

See Result »

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:

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

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

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

</body>
</html>
```

See Result »

Result:

6

Fig 1.16 test string method indexOf()

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

Edit This Code:

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

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

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

</body>
</html>
```

See Result »

Result:

Banana,Kiwi

Fig 1.17 test slice method usage in string

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

Edit This Code:

```
<!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>
```

See Result »

Result:

Hello Universe

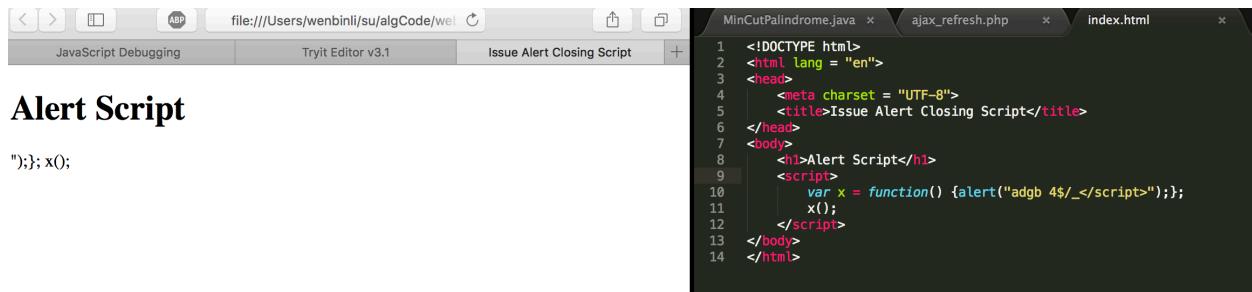
Fig 1.18 test string replace method

Observation and Explanation:

1. Like fig 1.1, link_name is link structure.
2. Like fig 1.2 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 specific string object, like fig 1.18.

Task2:



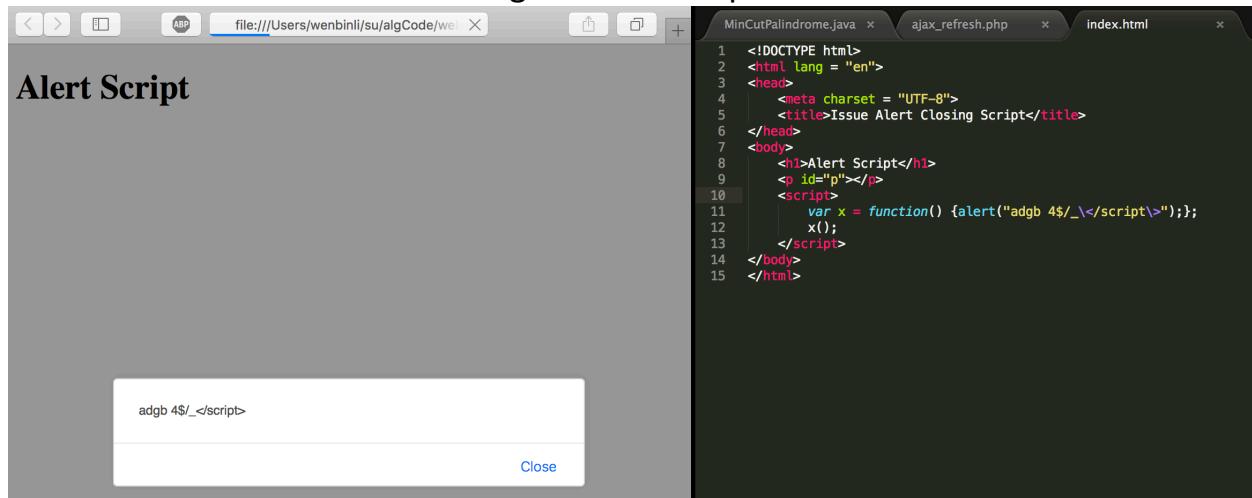
The screenshot shows a browser window with the title "Issue Alert Closing Script". The page content is an HTML file containing a script that attempts to close an alert window. The code is as follows:

```

<!DOCTYPE html>
<html lang = "en">
<head>
<meta charset = "UTF-8">
<title>Issue Alert Closing Script</title>
</head>
<body>
<h1>Alert Script</h1>
<script>
    var x = function() {alert("adgb 4$/_</script>");};
    x();
</script>
</body>
</html>

```

Fig 2.1 error output



The screenshot shows a browser window with the title "Issue Alert Closing Script". The page content is the same as in Fig 2.1, but the output is different. A modal dialog box is displayed with the text "adgb 4\$/_.</script>". Below the dialog is a "Close" button. The browser status bar shows the URL "file:///Users/wenbinli/su/algCode/webscripts/index.html".

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 »

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>
```

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 »

Result:

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

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

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

</body>
</html>
```

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 »

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>
```

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:

```
<!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>
```

See Result »

Result:

Click Me
Hello World

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

Run »

```
<!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>
```

Result Size: 582 >

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

Try it

Cookies associated with this document:
 ASPSESSIONIDSQCCTTTA=BEDMBDCDJJKJEKFJGCGGH;
 __gads=ID=d5a809cedb36695d:T=1477358797:S=ALNI_MYmmkt6CO91Zbh-BXUImmmFJt7kWQ; _ga=GA1.2.763069848.1466467053; _gat=1

Fig 3.5 HTML DOM Cookie Property

```
>>> document.getElementsByTagName("script")
[<script> jquery-1.6.4.min.js, <script> jquery-u...6.min.js, <script> elgg.1410864370.js, <script>]
>>> document.getElementsByTagName("div")
[<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, <div> elgg-widgets-add-panel, <div> profile, <div> elgg-inner, <div> profile-owner-block, <div> elgg-avatar, <div> profile-details, <div> elgg-body, <div> elgg-widget-col-1, <div> elgg-col-lof3, <div> elgg-widget-col-2, <div> elgg-col-lof3, <div> elgg-widget-col-3, <div> elgg-col-lof3, <div> elgg-widget-loader, <div> elgg-ajax-loader, <div> elgg-page-footer, <div> elgg-inner, <div> mts]
```

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

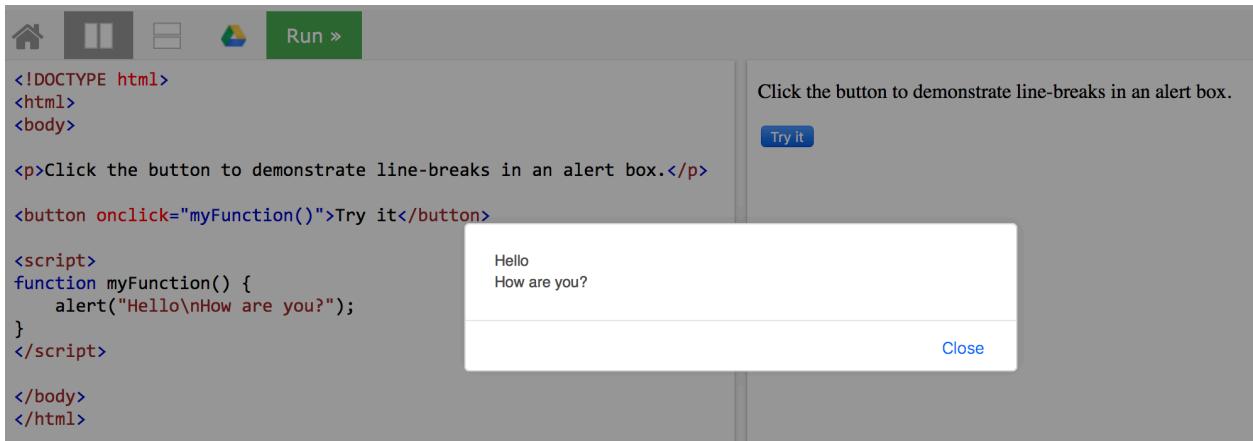
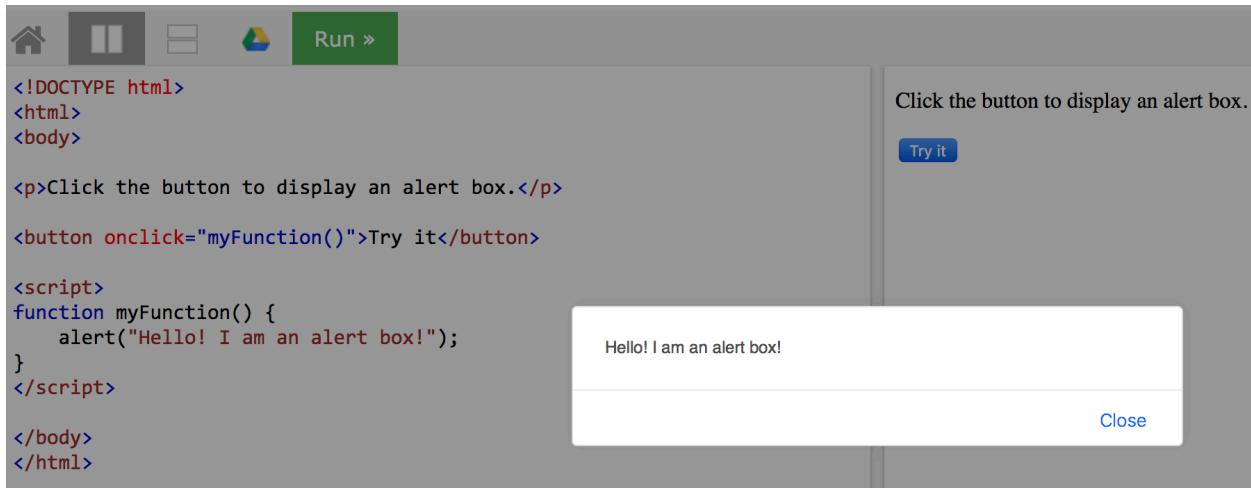


Fig 3.7 how the message in browser

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 browsers. '\n' to make a break line. Alert is a BOM method.

Task 4:



The screenshot shows a browser interface with a code editor on the left and a preview area on the right. The code editor contains the following HTML and JavaScript:

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

<p>Click the button to display an alert box.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
    alert("Hello! I am an alert box!");
}
</script>

</body>
</html>
```

The preview area displays the text "Click the button to display an alert box." and a "Try it" button. A modal alert box is open, showing the message "Hello! I am an alert box!" with a "Close" button.

Fig 4.1 Alert Information



The screenshot shows a browser interface with a code editor on the left and a preview area on the right. The code editor contains the following HTML, CSS, and JavaScript:

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

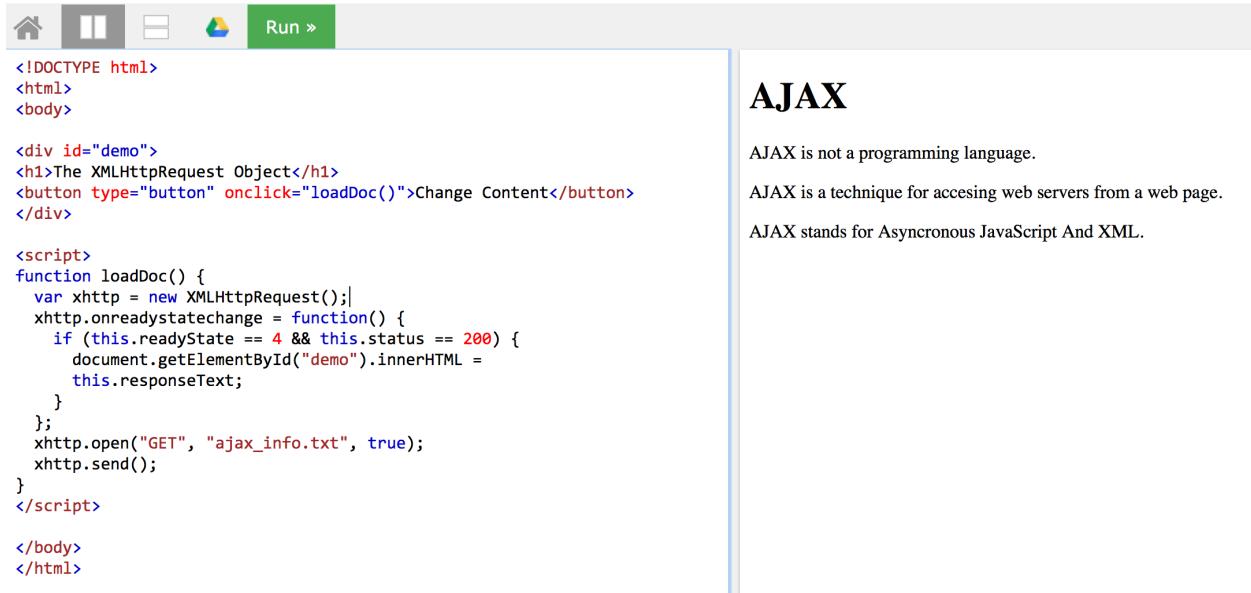
<div id="demo">
<h1>The XMLHttpRequest Object</h1>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<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", "ajax_info.txt", true);
    xhttp.send();
}
</script>

</body>
</html>
```

The preview area has a title "The XMLHttpRequest Object" and a "Change Content" button. Below the button is a text input field containing the text "Hello! I am an alert box!".

Fig 4.2 How to use Ajax to send the http request



```

<!DOCTYPE html>
<html>
<body>

<div id="demo">
<h1>The XMLHttpRequest Object</h1>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<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", "ajax_info.txt", true);
    xhttp.send();
}
</script>

</body>
</html>

```

AJAX

AJAX is not a programming language.

AJAX is a technique for accessing web servers from a web page.

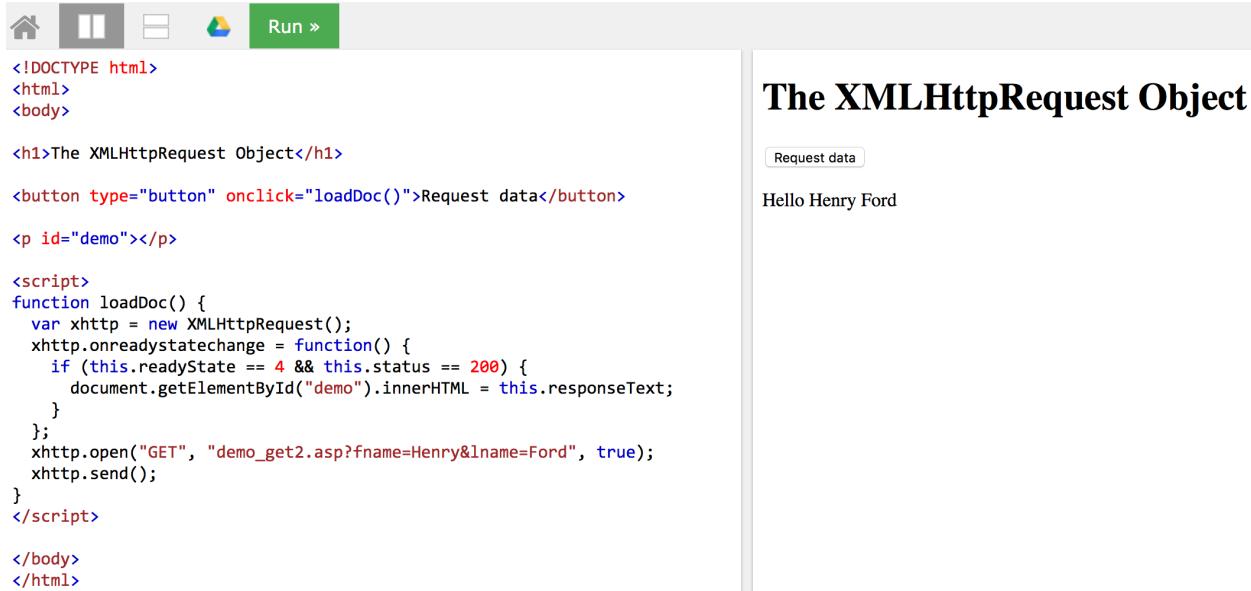
AJAX stands for Asynchronous JavaScript And XML.

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:



The screenshot shows a browser window with the title "The XMLHttpRequest Object". The page content includes an "h1" header "The XMLHttpRequest Object", a "button" with the "onclick" event set to "loadDoc()", and a "p" element with id="demo". Below the button is a "script" block containing JavaScript code. The code defines a "loadDoc" function that creates a new XMLHttpRequest object, sets its "onreadystatechange" event handler to update the "innerHTML" of the "demo" paragraph when readyState is 4 and status is 200, opens a GET request to "demo_get2.asp?fname=Henry&lname=Ford", and sends the request. The browser's toolbar at the top includes icons for home, back, forward, and run.

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

<h1>The XMLHttpRequest Object</h1>

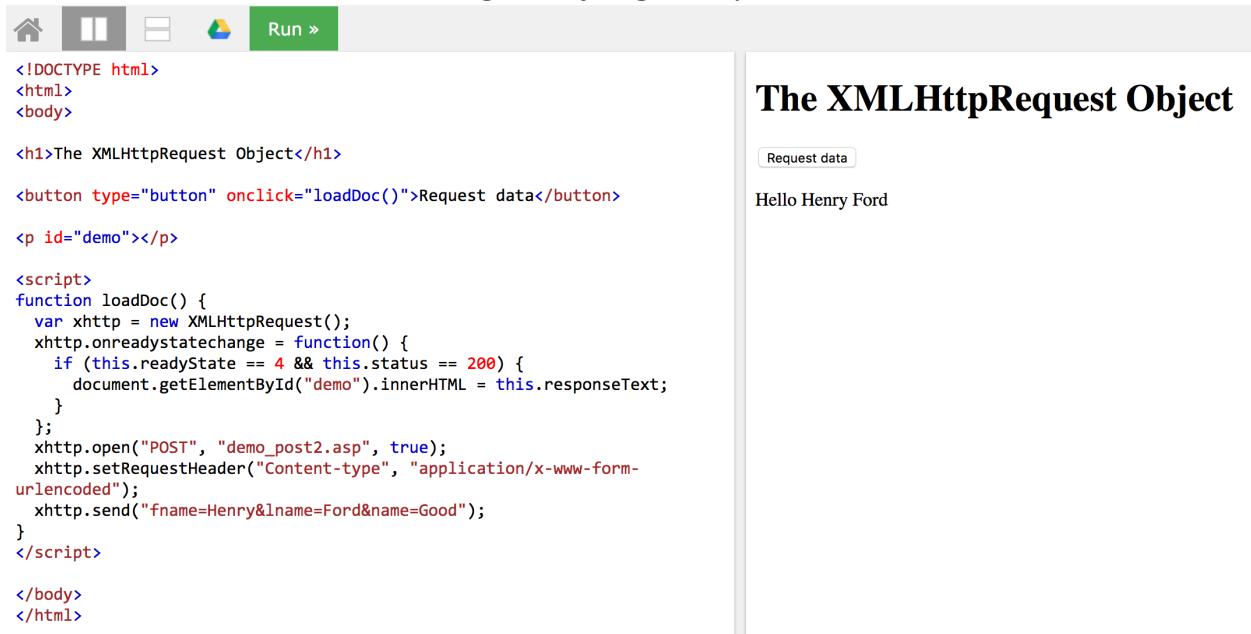
<button type="button" onclick="loadDoc()">Request data</button>

<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



The screenshot shows a browser window with the title "The XMLHttpRequest Object". The page content is identical to Fig 5.1, featuring an "h1" header, a "button" with "onclick" set to "loadDoc()", and a "p" element with id="demo". The "script" block contains the same JavaScript code as Fig 5.1, but it uses a POST request instead of a GET request. The "open" method is called with "POST" as the first argument. The browser's toolbar at the top includes icons for home, back, forward, and 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

A screenshot of a web browser window. The address bar shows 'http://www.xsslabeledgg.com/blog/all'. A floating window titled 'Live HTTP headers' is overlaid on the page. This window has tabs for Headers, Generator, Config, and About, with Headers selected. It displays the following information:

HTTP Headers
http://www.xsslabeledgg.com/blog/all

GET /blog/all HTTP/1.1
Host: www.xsslabeledgg.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
Referer: http://www.xsslabeledgg.com/blog/all
Cookie: Elgg=073d1j72coinupu79fr27ktot6
Connection: keep-alive

HTTP/1.1 200 OK
Date: Tue, 25 Oct 2016 14:37:31 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.14
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0

At the bottom of the floating window are buttons for 'Save All...', 'Replay...', 'Capture' (with a checked checkbox), 'Clear', and 'Close'.

Fig 5.4 live http headers capture get request

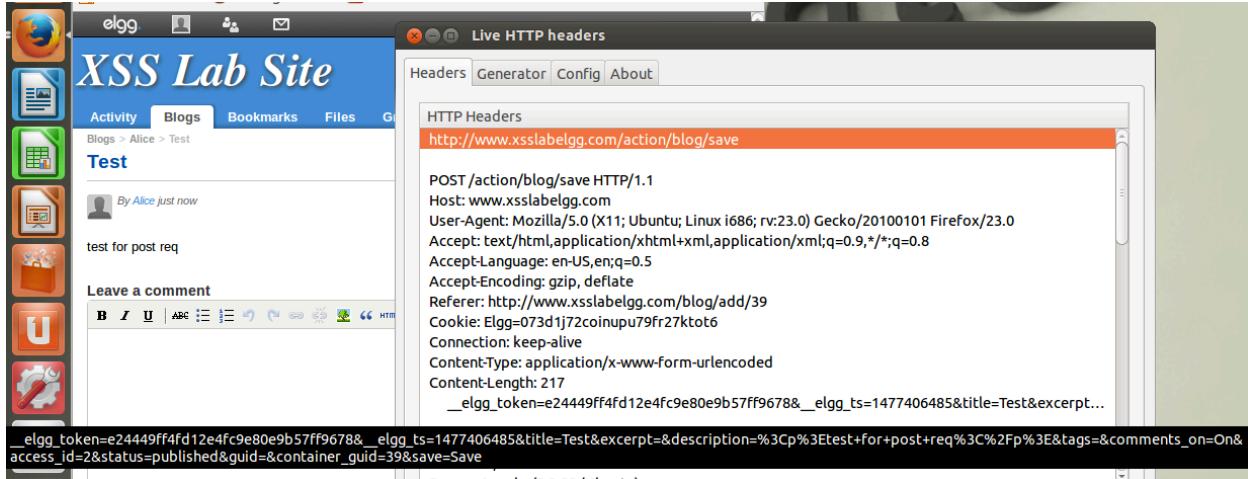


Fig 5.5 Live http headers capture post request

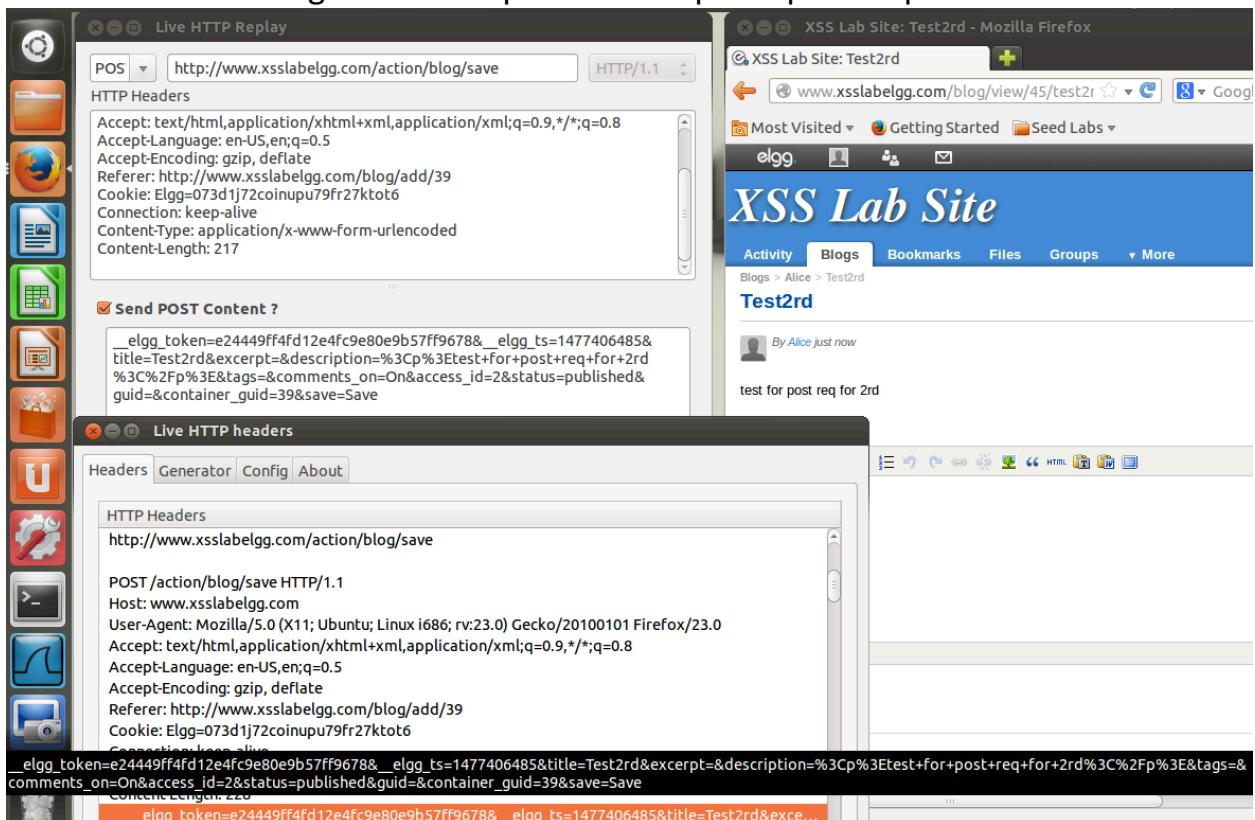


Fig 5.6 Live http headers capture replay the post request

Observation and Explanation:

- 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 type, 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.