Open Source Software Project Development

Dr. T.Y. Wong

Weeks 5, 6, 8

JavaScript Programming (1)

- A interpreter language with the browser as its interpreter.

JavaScript, it starts everything…

`JavaScript ≠ JAVA`

- Initially, the language is called "**`LiveScript`**".
  - A scripting language developed at Netscape.
  - In1995, JAVA was believed to be the **next big thing**.
  - SUN and Netscape together worked on marketing the new scripting language and **renamed it as JavaScript**.

- Interesting fact #1: JavaScript is a trademark of SUN.

- Interesting fact #2: JavaScript programs are automatically open-source.

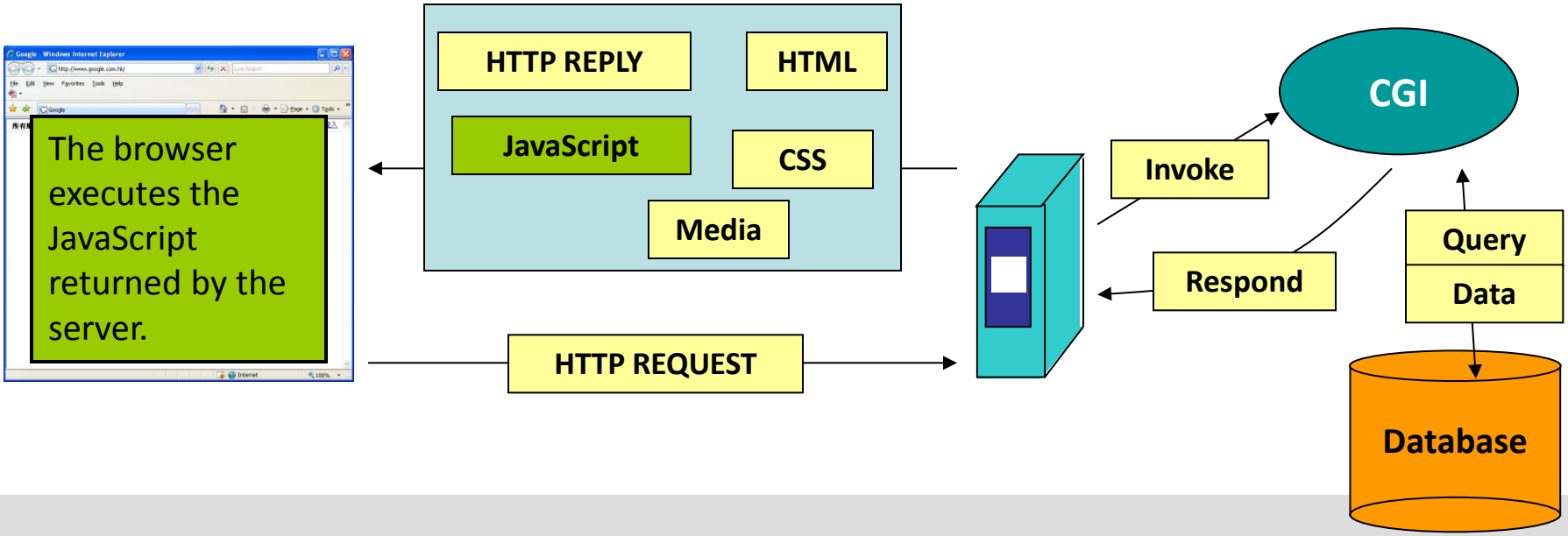
JAVA VS JavaScript

- Encore: **JavaScript ≠ JAVA**

	JAVA	JavaScript
Basic Syntax	C-like	C-like
OO?	Yes	No. It is called prototype-based programming language.
Typing	Strong typing	Dynamic typing
Variable types	The set of variables in JAVA is just a subset of that of JavaScript.	
	E.g., A variable in JavaScript can be of type "`function`" (?!)	

JavaScript VS CGI

- They are totally different, but are usually work together.
  - JavaScript is also known as the **client-side scripting language**.
  - It is about **automating** the client and makes the client **programmable**!



JavaScript at a Glance…

- To give powers to web authors to…
  - Handle events, e.g., mouse, keyboard, window resize.
  - Manipulate the browser window(s);
  - Handle HTML forms;
  - Make (not arbitrary) HTTP requests;
  - Manipulate HTML content through the Document Object Model (DOM);

Program codes for 06_asyn_demo

ajax/	hidden_frame/	---	---

http://demo4140-tywong.rhcloud.com/06_asyn_demo/

A Preview on Asynchronous Page Design

What is AJAX?

- AJAX means…

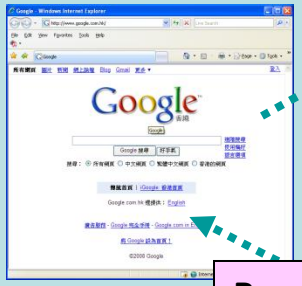
Asynchronous

JavaScript

And

XML

Asynchronous VS Synchronous



Request

Response

Synchronous

After the request has been sent, the browser displays the HTML carried by the response.

Asynchronous

That means after the request is sent, the browser can still do other things while waiting for the response.

Such an asynchronous feature is controlled by **JavaScript**.

But, what is AJAX?

- AJAX means…

Asynchronous

JavaScript

And

XML

XML or not?

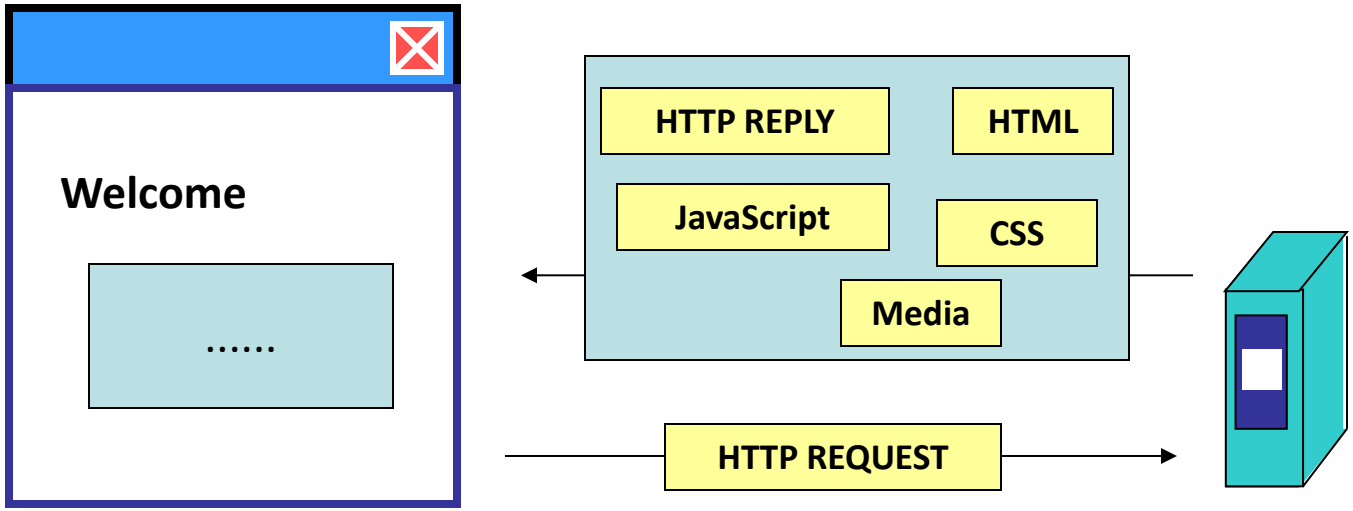
HTML
The usual response generated by synchronous mechanism.

XML
The **suggested data format** of the response for the asynchronous mechanism is XML. It is because XML is well-structured nature and is very suitable for simple data transfer.

Remember, it is not a must.
E.g., the lecturer likes to use the **plain-text format**.
Later, we will also cover the use of **JSON – JavaScript Object Notation**.

Before Asynchronous Control…

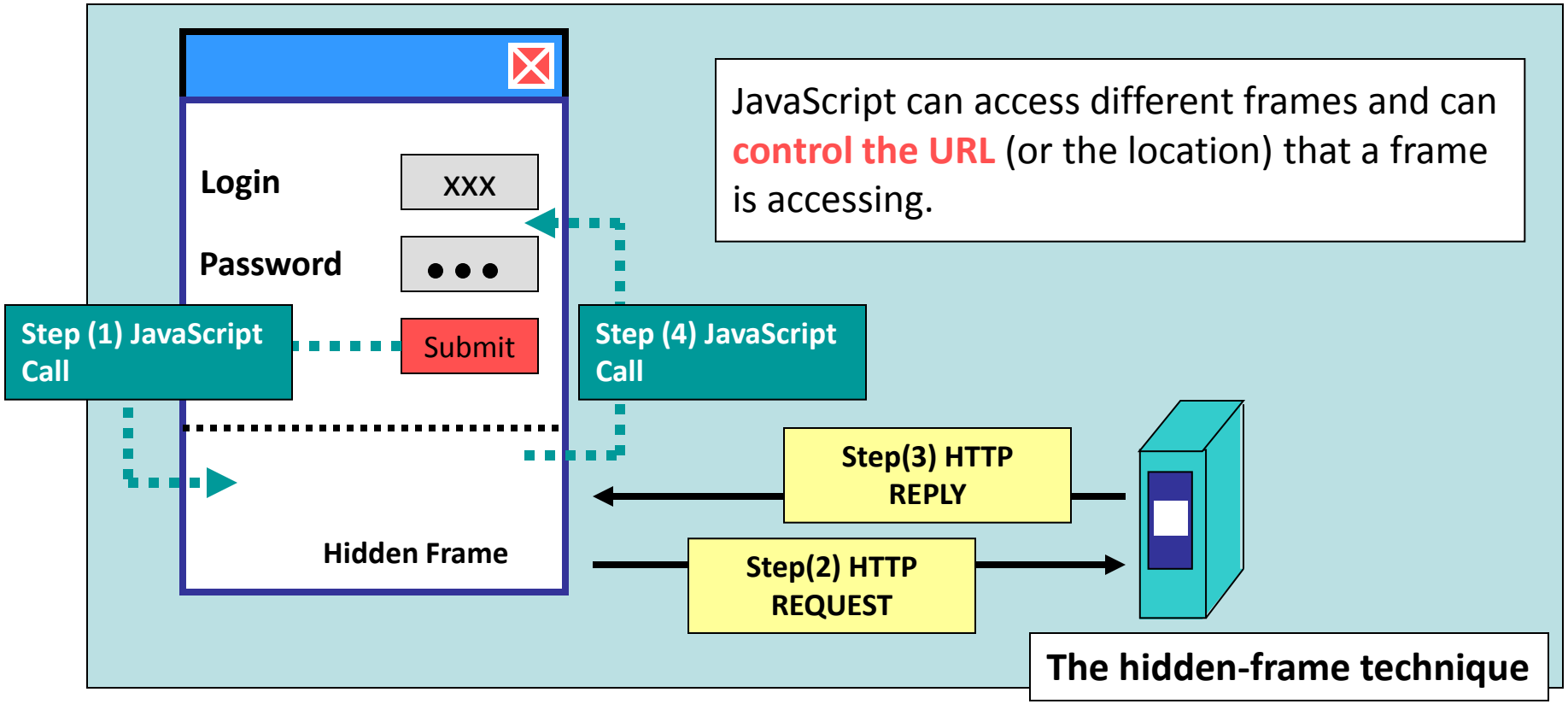
- Let's take a look at what traditional web systems (Weeks 1 – 4) are …



The browser displays the HTML stated inside the HTTP reply and discards what it is previously displaying.

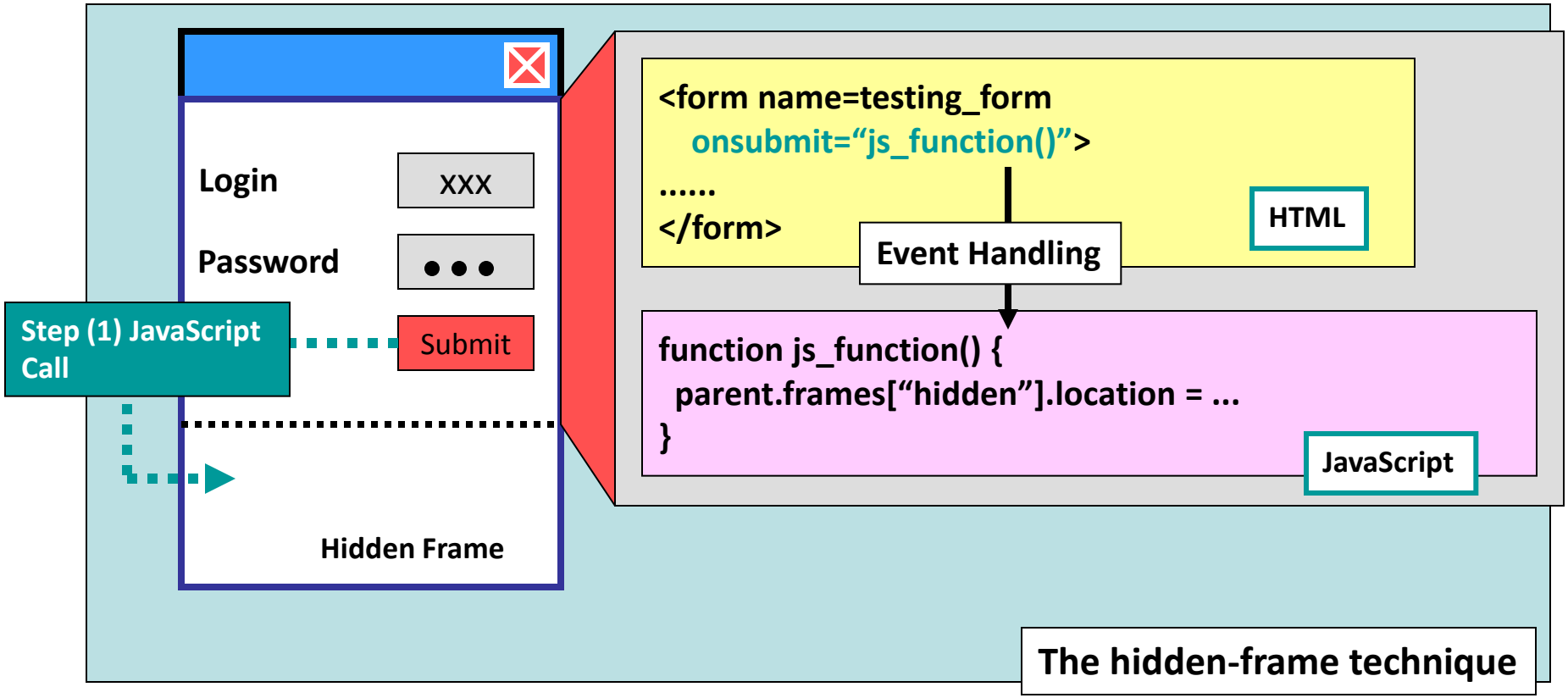
Approaching Asynchronous Control……

- Before AJAX, the asynchronous-like request/response model is already there…



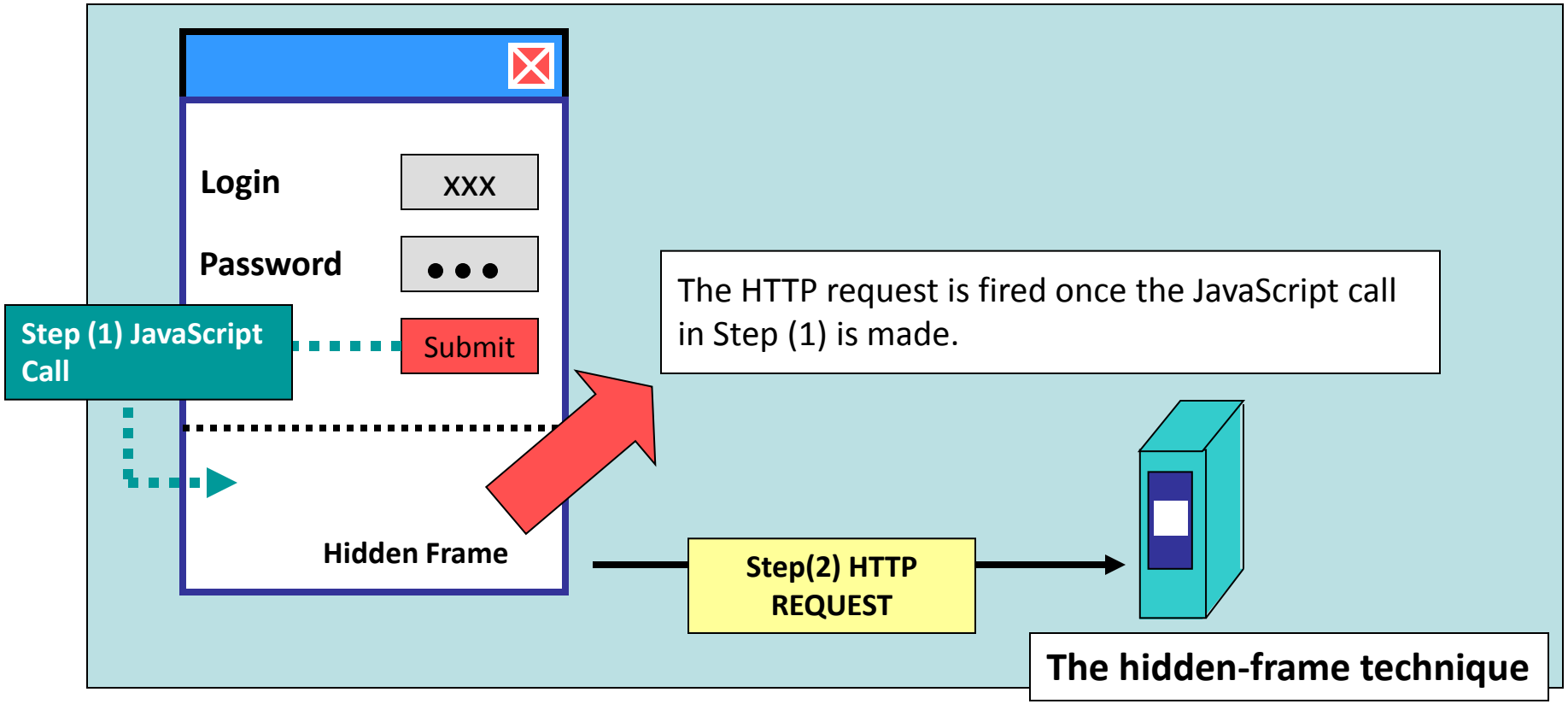
Approaching Asynchronous Control……

- Before AJAX, the asynchronous-like request/response model is already there…



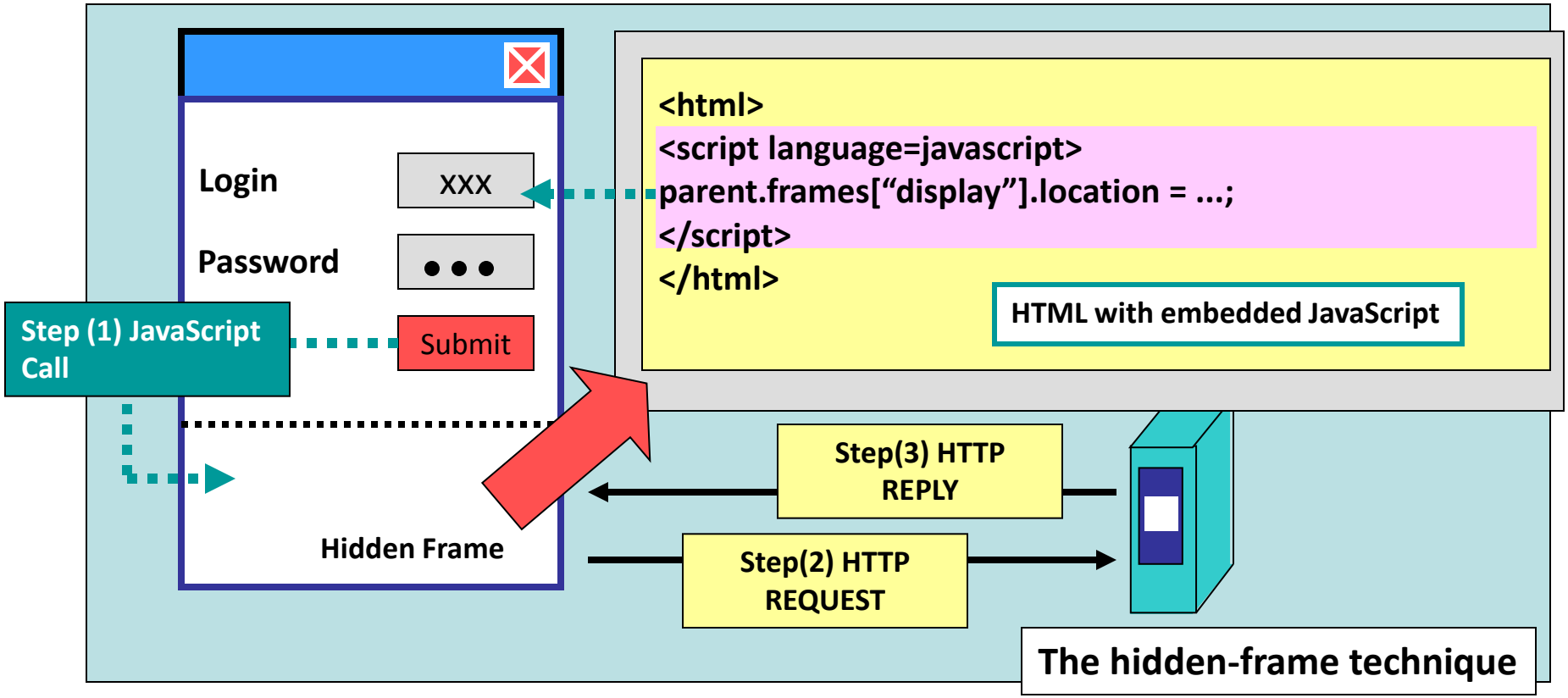
Approaching Asynchronous Control……

- Before AJAX, the asynchronous-like request/response model is already there…



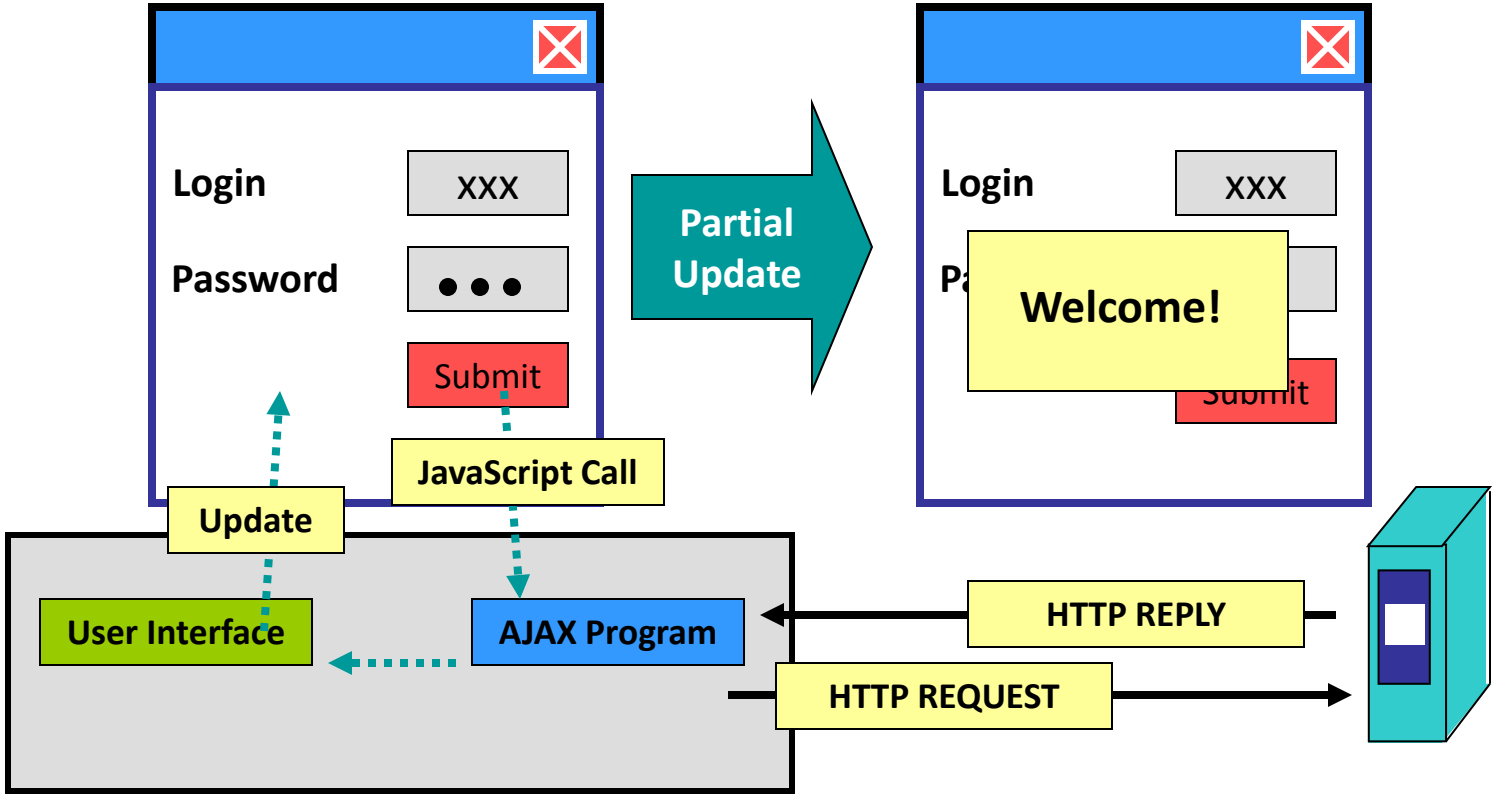
Approaching Asynchronous Control……

- Before AJAX, the asynchronous-like request/response model is already there…



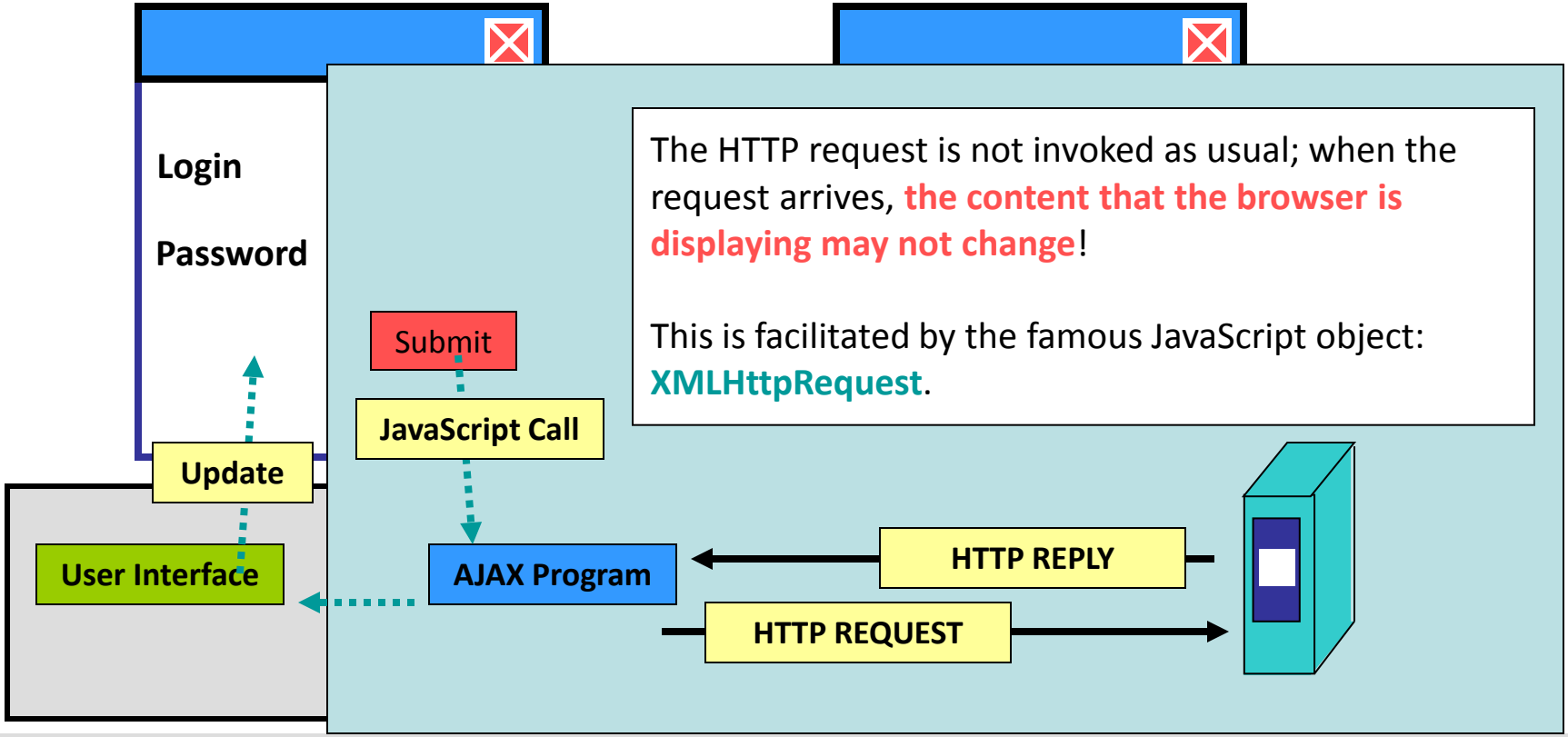
With Asynchronous Control……

- Let's look at what AJAX-enabled systems are…



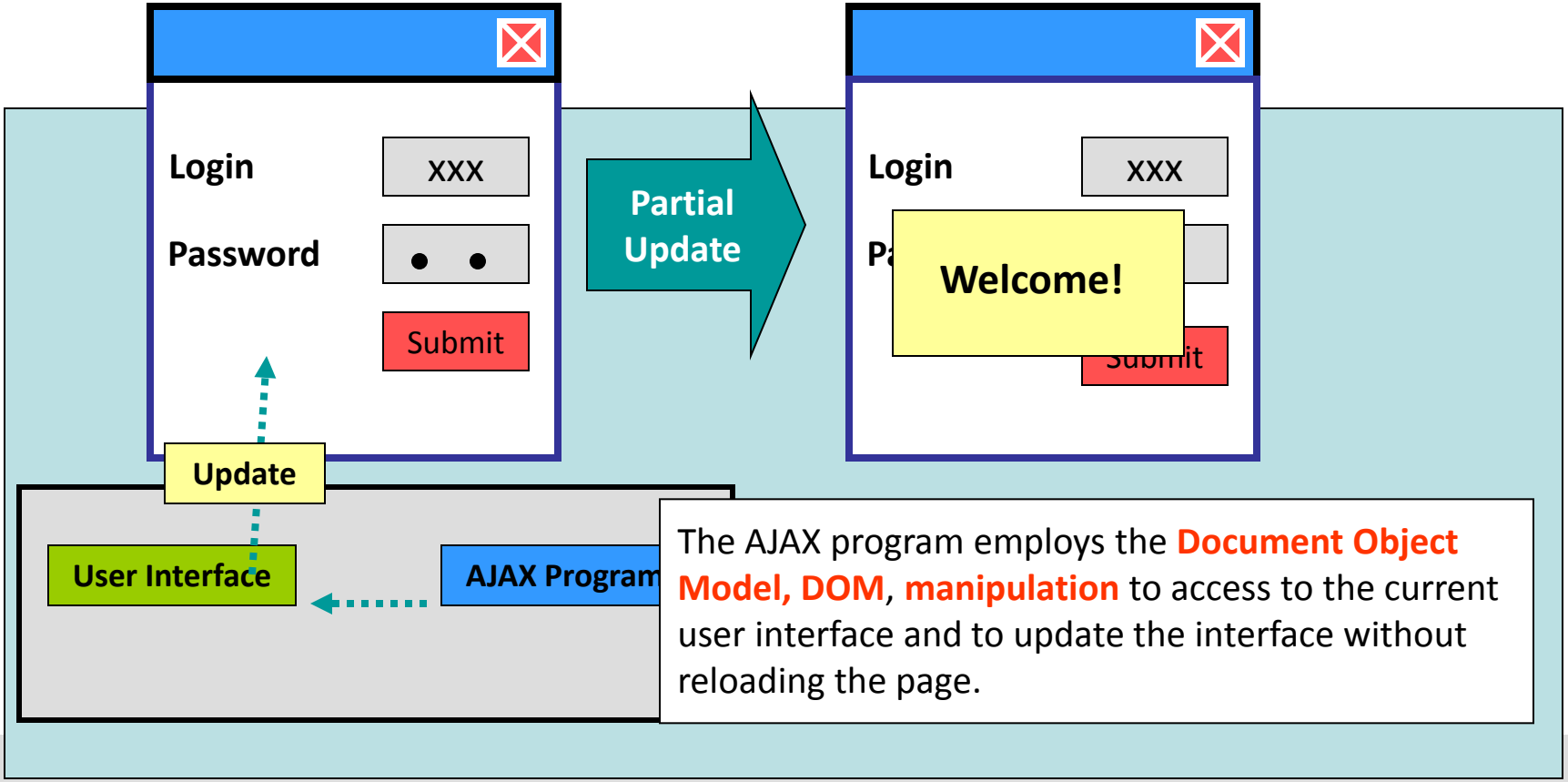
With Asynchronous Control……

- Let's look at what AJAX-enabled systems are…

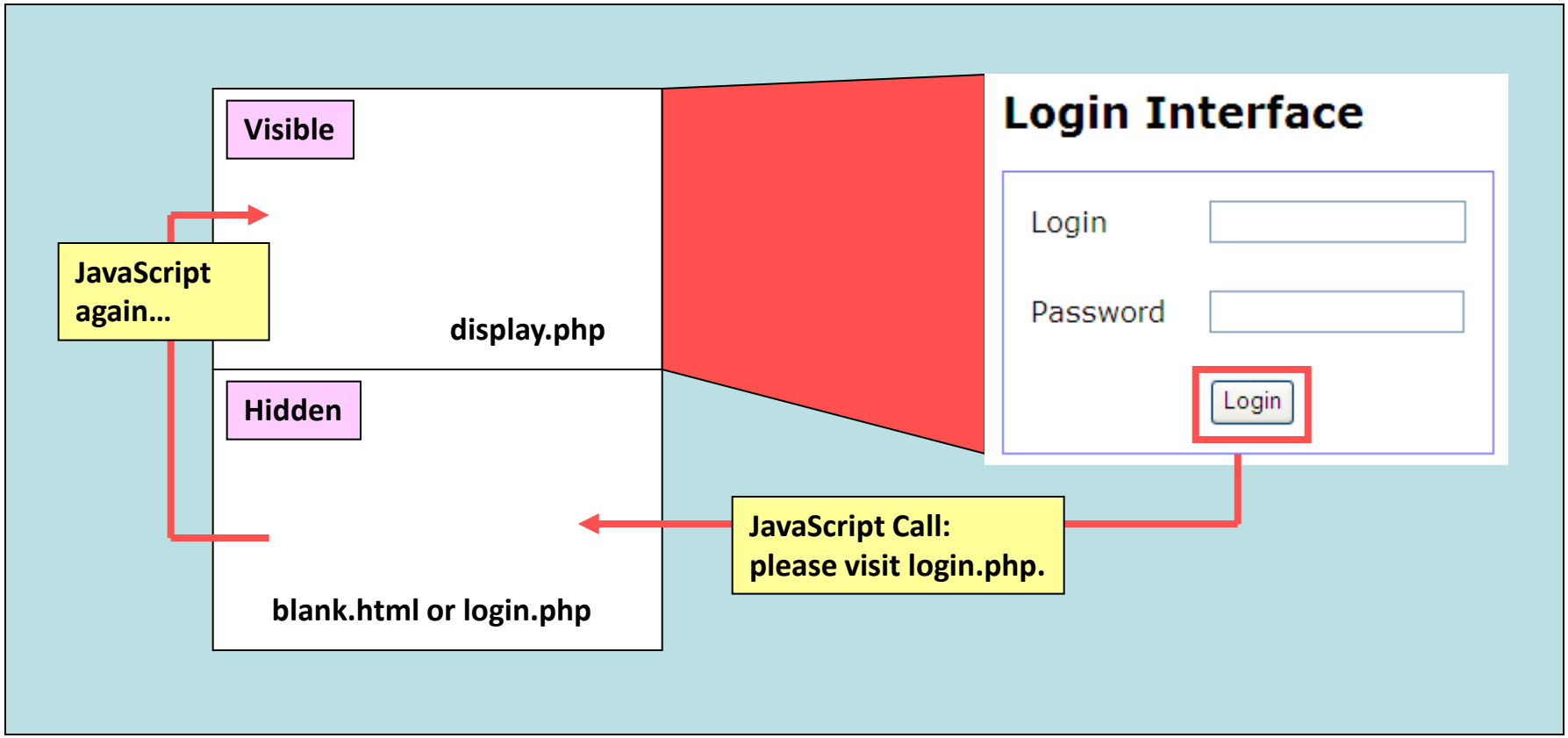


With Asynchronous Control……

- Let's look at what AJAX-enabled systems are…

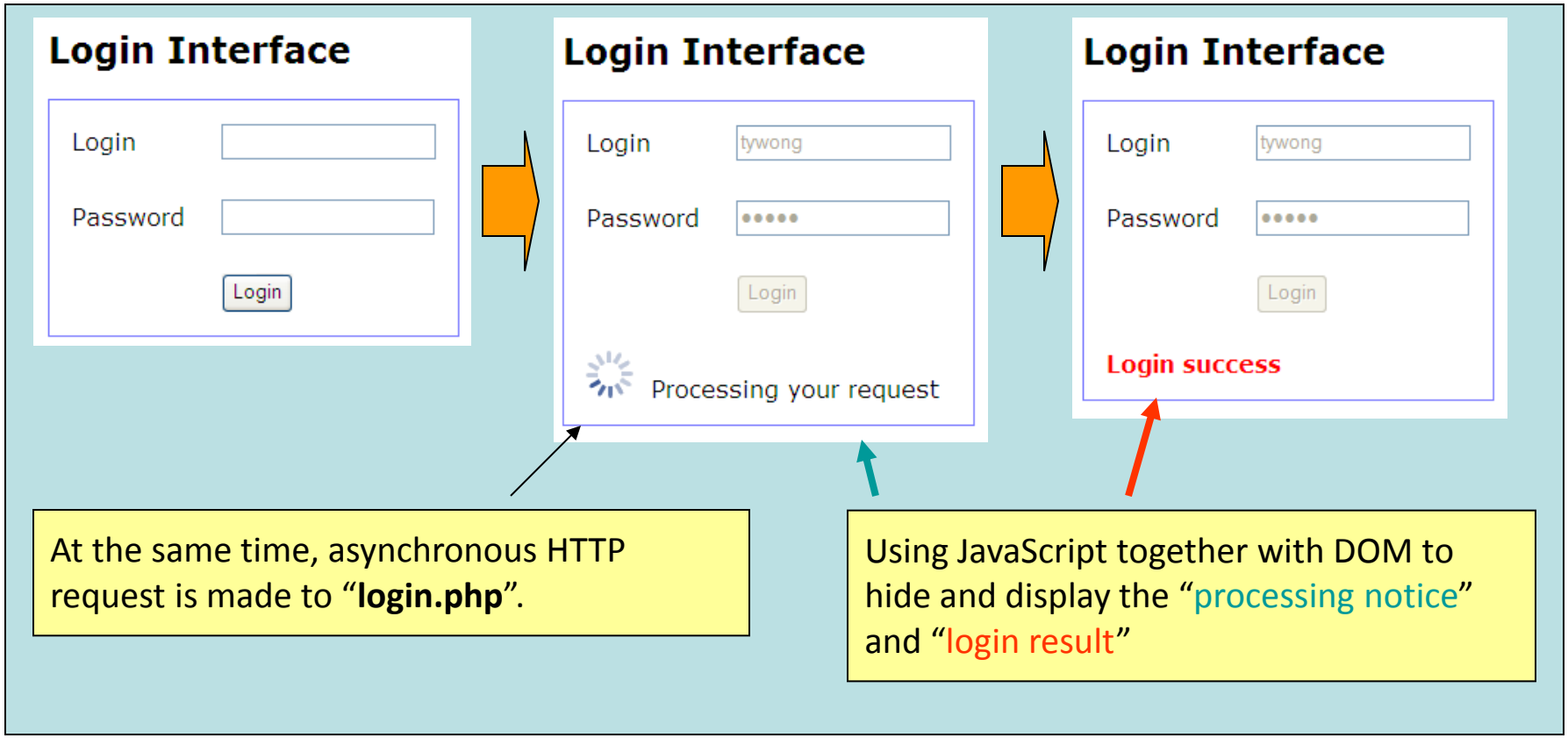


Examples – the hidden-frame toy



See "hidden_frame/"

AJAX Examples – the toys



See "ajax/"

AJAX Examples – real ones

- Let's have an easy-to-understand real example first.

Using JavaScript to **listen to the event** that a user is typing.

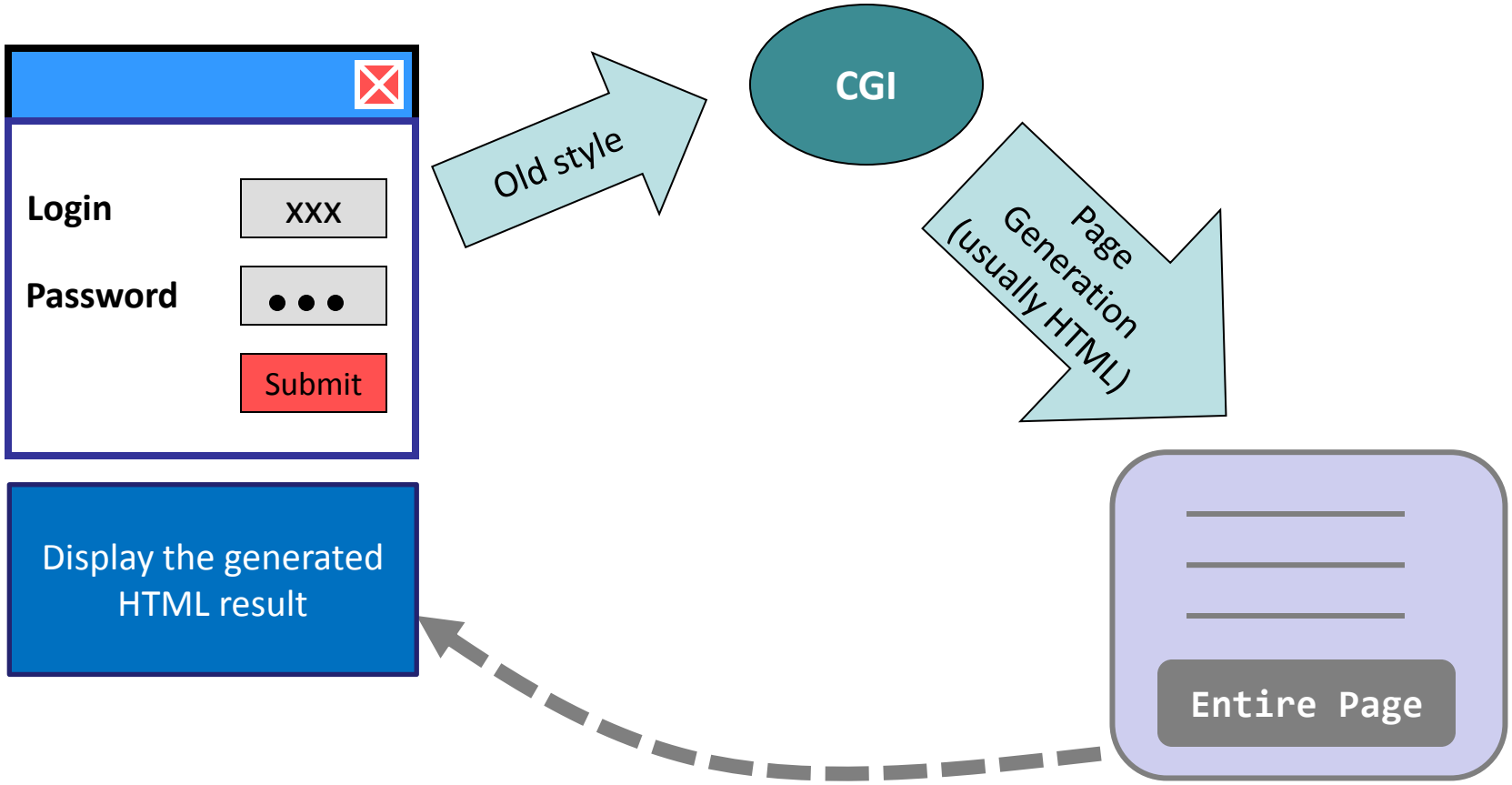
Using **XMLHttpRequest** object to retrieve the list of suggestions.

Displaying the retrieved list, not by refreshing the entire page.

http://www.google.com.hk/complete/search?q=AJAX&client=hp

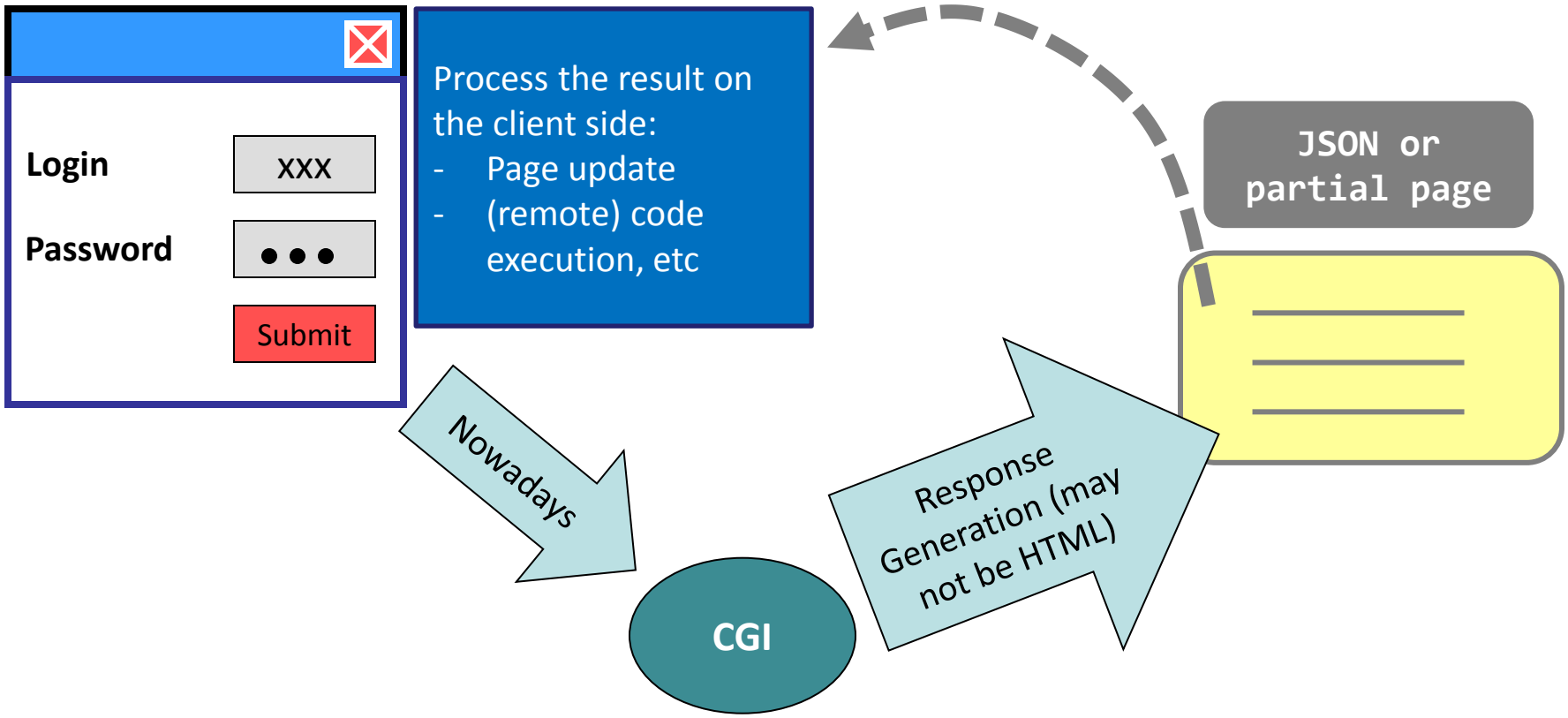
Summary

- From **Web Site** to **Web App**…



Summary

- From **Web Site** to **Web App**...



JavaScript…brain-damaging programming

- How brain-damaging is it?
  - Most of time, the codes for IE and Chrome are **not the same**.
  - It is just because different browsers have different JavaScript implementations!

Lecturer's experience:

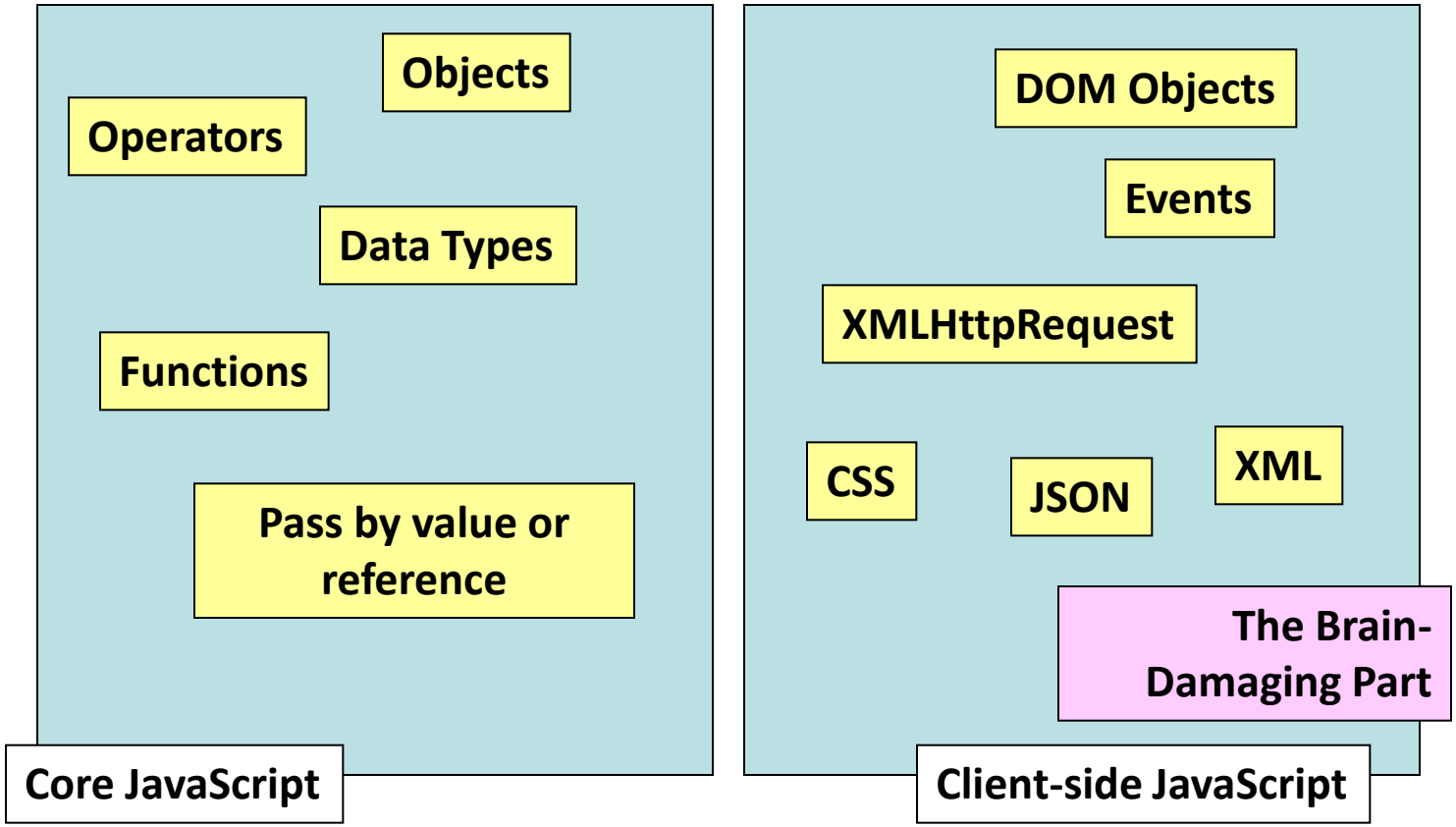
Chrome always catches up the up-to-date standard.
While IE is always **FOLLOWING ITS OWN STANDARD**!

You have to find out which parts are not following the up-to-date standard!!

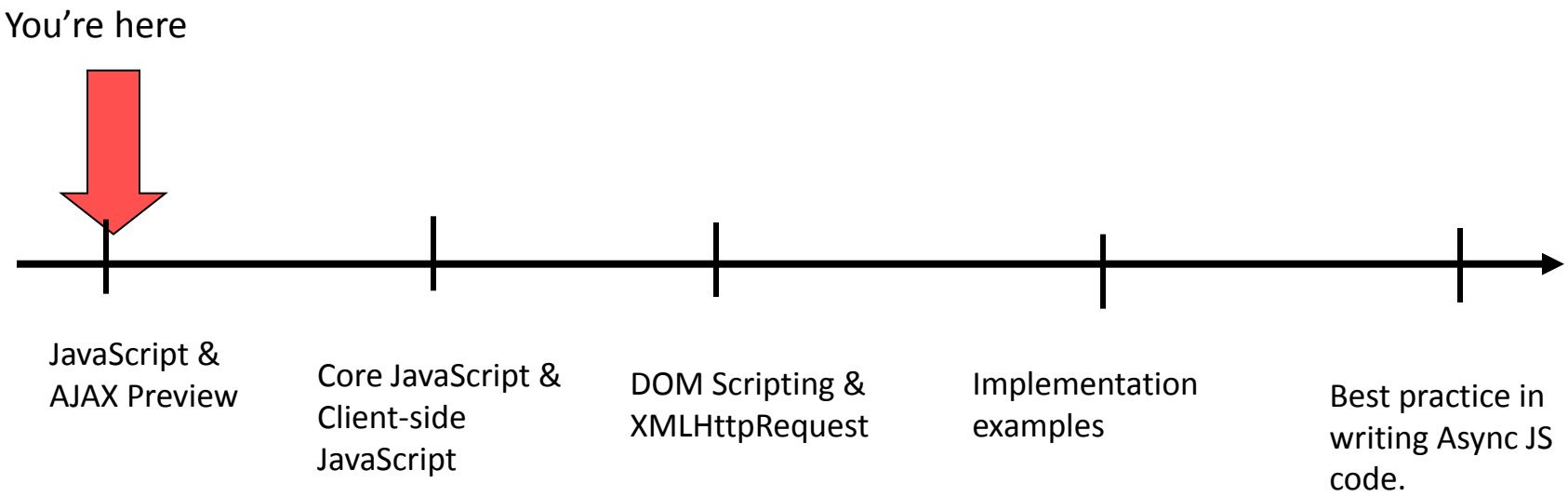
The brain-damaging part

JavaScript…two sides of a story

- There are two sides about JavaScript.



Colorful Topics are coming...



Program codes for js_core

all_files.zip	array_compare.html	array_content.html	array_function.html	array_index.html
array_resize.html	comparison.html	crazy_example.html	embed.html	embed.js
error.html	object_example1.html	object_example2.html	parseInt.html	pass_by_xx.html
protocol.html	scoping.html	scoping2.html	string_compare.html	test.html
try_and_catch.html	typing.html	undefined.html	---	---

Fall 2011, CSCI4140, Department of Computer Science and Engineering, The Chinese University of Hong Kong.

http://demo4140-tywong.rhcloud.com/07_js_core/

Core JavaScript

- a crash course...hope that you won't crash...

Prerequisite: Embedding scripts...

```
<html>
<script>
    alert("hello world");
</script>
</html>
```

Script starts

Script ends

Inline Script

Any scripts between **<script>** & **</script>** will be ignored.

```
<html>
<script src="[url to javascript file]"></script>
</html>
```

External Script

```
<html>
<a href="javascript: alert('hello world');">
click here</a>
</html>
```

The special "javascript:" protocol.

javascript:

See "embedded.html", "embed.js", "protocol.html"

Prerequisite: Embedding scripts...

```
<html>
<script>
    alert("hello world");
</script>
</html>
```

```
<script language="JavaScript">
```

Deprecated!

```
<script type="text/javascript">
```

Up-to-date!

http://www.w3.org/TR/REC-html40/interact/scripts.html

Prerequisite: Embedding scripts...

```
<html>
<script>
    alert("hello world");
</script>
</html>
```

Old browsers did not recognize the <script> tag....

As a result, the content will become the text and be displayed...[so sad]

```
<html>
<script><!--
    alert("hello world");
//-->
</script>
</html>
```

The workaround is to change all the scripts into **HTML comments**!

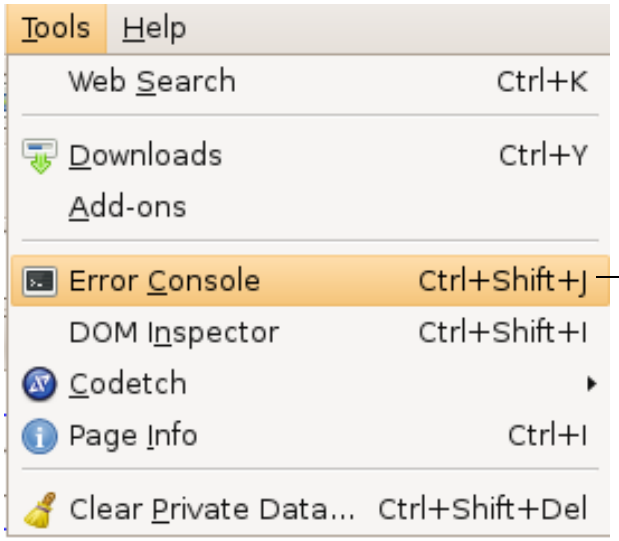
You know, HTML comment block begins with "**<!--**" and ends with "**-->**".

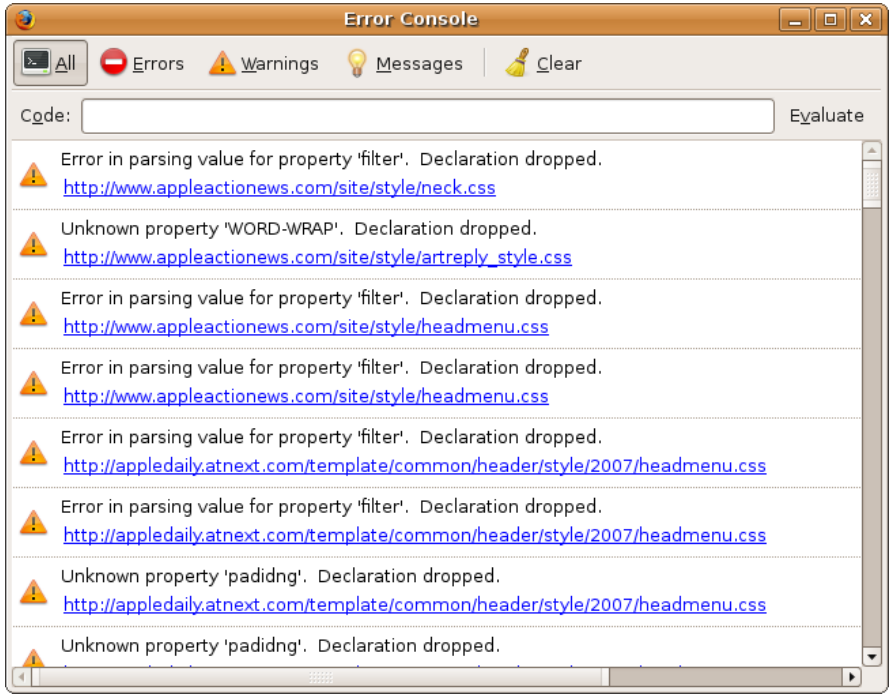
JavaScript's comment. This is to avoid the JavaScript engine treating "**-->**" as a part of the script.

You know, in addition to the "**//**" one-line comment, JavaScript also support the "**/* */**" comment block!

Prerequisite: Debugging…

Firefox users





Chrome users: Using "Developer Tool" (Devtool)!

Prerequisite: Debugging…

```
<html>
<script>
    function test_func(input) {
        if(input == 0)
            return;
        else
            return true;
    }
    var x = Math.random() * 10;
    alert (test_func(Math.round(x)));
</script>
</html>
```

Can you spot the problem?

Firefox users:
Go to…`about:config`

Switch the following field to true.

`javascript.options.strict`

Reload and check the error console again…

Reference: http://kb.mozillazine.org/Javascript.options.strict

See "error.html"

Prerequisite: Debugging…

```
<html>
<script>
    try {
        i = j + k;
    }
    catch( e ) {
        document.write(e);
    }
</script>
</html>
```

The try-and-catch block can save you a day.

Yet, this means the error is handled *properly*, so the error console will be show the corresponding error message..

See “try_and_catch.html”

JavaScript Data Type

- Basically, it is JAVA…but
  - with dynamic typing!

Keyword
var

```
<html>
<script>
    var name;
</script>
</html>
```

A variable can the one of the following types:

- number
- string
- object
- function
- boolean

JavaScript Data Type

- Undefined VS Undeclared?

Operator
typeof [var]
Return String; The type of the input variable.

```
<html>
<script>
    var string = "";

  /*  Case #1  */
    var i;
    string = typeof i + "; " + i + "<br>\n";
    document.write(string);

    string = typeof j + "; " + j + "<br>\n";
    document.write(string);
</script>
</html>
```

'+' is overloaded for concatenating strings.

undeclared!
This will result in error!

IMO, this should be called "**value undefined**"!

```
undefined; undefined
```

See "undefined.html"

JavaScript Data Type

- "**`null`**" is an object?

```
<html>
<script>
    var string = "";

  /*  Case #2  */
    var i = null;
    string += typeof i + "; " + i + "<br>\n";

    document.write(string);
</script>
</html>
```

Operator
typeof [var]
Return String; The type of the input variable.

"null" means **null object reference**.

```
object; null
```

See "typing.html"

JavaScript Data Type

- "number" includes both integer and floating point number.

Operator
typeof [var]
Return String; The type of the input variable.

```
<html>
<script>
    var string = "";

  /*  Case #3  */
    var i = null;
    i += 2.2;
    string += typeof i + "; " + i + "<br>\n";

    document.write(string);
</script>
</html>
```

Interesting! Object reference + number = number!

```
number; 2.2
```

No float, no int.
Just number.

See "typing.html"

JavaScript Data Type

- “string” is more superior than “number”.

Operator
typeof [var]
Return String; The type of the input variable.

```
<html>
<script>
    var string = "";

  /*  Case #4  */
    var i = 2.2;
    i += “ 100 hello”;
    string += typeof i + "; " + i + "<br>\n";

    document.write(string);
</script>
</html>
```

JavaScript loves strings more than numbers…

```
string;2.2 100 hello
```

See “typing.html”

JavaScript Data Type

- Well..."function" is a data type!

```
<html>
<script>
    var string = "";

  /*  Case #5  */
    var i = function() {
        alert("hello world");
    };
    string += typeof i + "; " + i + "<br>\n";

    document.write(string);
    i();
</script>
</html>
```

Operator
typeof [var]
Return String; The type of the input variable.

Although this is crazy, this is allowed and is called a "**function literal**".

```
function; function()
{ alert("hello world"); }
```

See "typing.html"

JavaScript – Functions

- Scoping .... is ... a ... mess ... (it is called the **function scope**)

```
<script>
function test() {
    var i = 0;
    if(true) {
        var j = 0;
        document.write(j);
        document.write(k);
        for(var k = 0; k < 10; k++) {
            document.write(k);
        }
        document.write(k);
    }
    document.write(j);
}
test();
document.write(i);
</script>
```

undefined, with error.

Because JavaScript has the function scope.

We can see the values!!

Because no block scope in JavaScript.

The value is yet defined.

It won't result in error.

See "scoping.html"

JavaScript – Functions

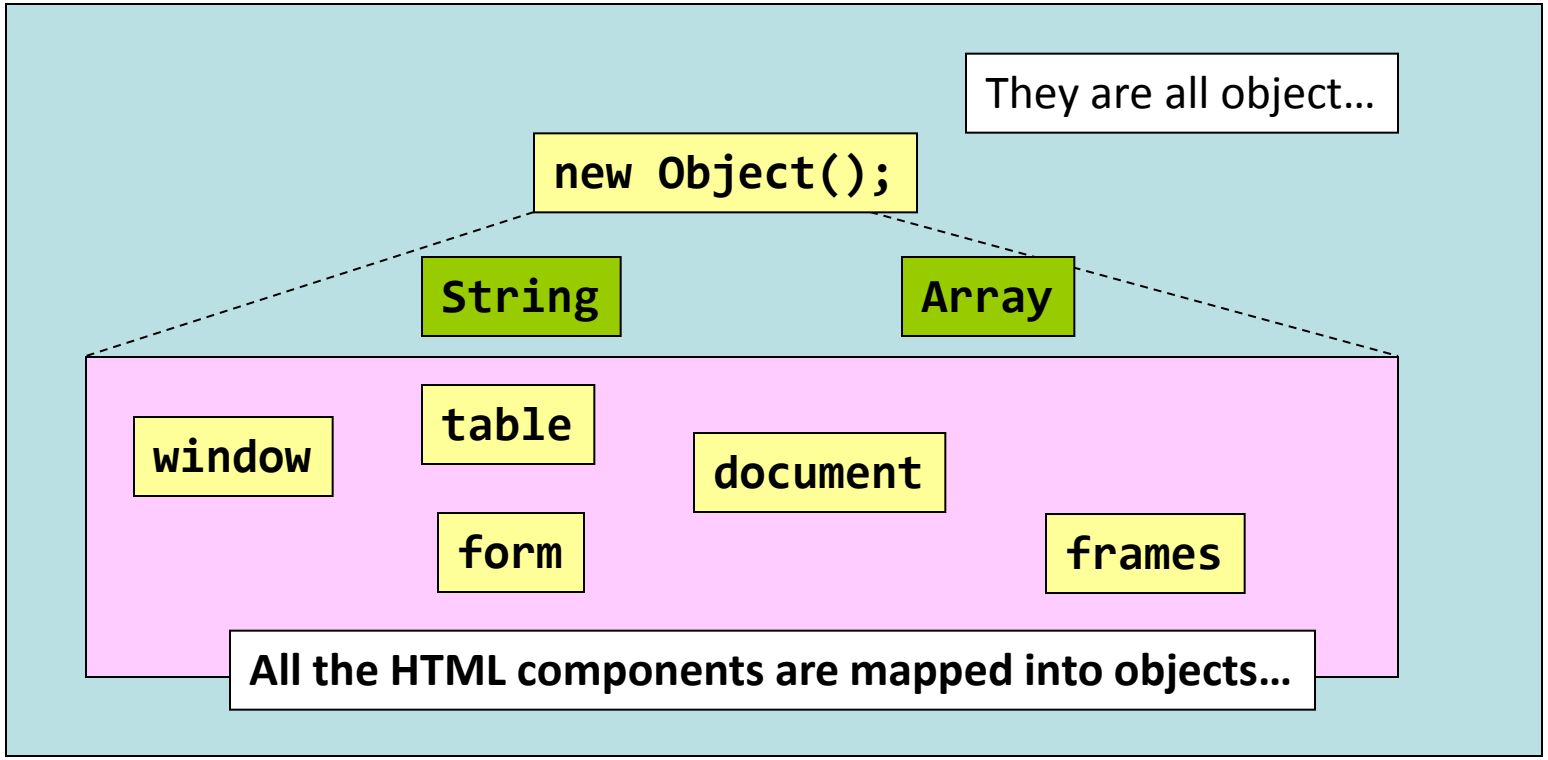
- Passing by Value / Reference?
  - It follows JAVA.

Types	Copied By	Passed By
Number	value	value
Boolean	value	value
Object	reference	reference

See "pass_by_xx.html"

JavaScript - Object

- JAVA is full of objects.
  - how about JavaScript?



JavaScript – Array

- Again, JavaScript follows JAVA...an array is an object!

Instantiation
new Array()
[...]

```
<script>
    var a = [1,2,3];
    var b = new Array();
    b[0] = 1;
    b[1] = 2;
    b[2] = 3;

    var result = false;
    if(a.length == b.length) {
        result = true;
        for(var i = 0; i < a.length ; i++)
            if(a[i] != b[i])
                result = false;
    }
    document.write(result);
</script>
```

"**length**" is an instance variable that stores the number of elements in the array.

See "array_compare.html"

JavaScript – Array

```
<script>
    var a = [0,1,2];
    var result;

    a[5] = 5;
    result = new String();
    for(i = 0; i < a.length; i++)
         result += a[i] + "; ";
    document.write(result + "<br><br>");

    a.length = 1;
    result = new String();
    for(i = 0; i < a.length; i++)
         result += a[i] + "; ";
    document.write(result + "<br><br>");

    a.length = 10;
    result = new String();
    for(i = 0; i < a.length; i++)
         result += a[i] + "; ";
    document.write(result + "<br><br>");
</script>
```

What?! Skipping index?!

Well...“**length**” is not a read-only instance variable...

Let’s shrink the array!

Let’s grow the array! (Please...stop scaring me...)

See “array_resize.html”

JavaScript – Array

```
<script>
    var a = new Array();

    a[0] = true;
    a[1] = "string";
    a[2] = 1.0;

    for(var i = 0; i < a.length; i++) {
        document.write(
            typeof a[i]
            + "; " + a[i] + "<br>\n");
    }
</script>
```

Well...an array is just a container...there is no fixed type for an array.

See “array_content.html”

JavaScript – Array

```
<script>
    var array = new Array();

    array[0] = "I";
    array[1] = "know";
    array["this"] = "THIS";
    array["program"] = "PROGRAM";
    array["is"] = "IS";
    array["crazy"] = "CRAZY";

    for(var i in array) {
        document.write(
            "array[" + i + "] = "
            + array[i] + "<br>\n"
        );
    }
</script>
```

Look at the indices!!
Look at "**`array.length`**", too!!

This special for-loop gives you all the indices of an array.

See "array_index.html"

JavaScript – Array

```
<script>
    var a = new Array();

    a.push("this");
    a.push("class");
    a.push("is");
    a.push("boring");
    document.write(a.join(' ') + "<br><br>\n");

    var b = (a.join(' ')).split(' ');

    b.sort();
    document.write(b.join(' ') + "<br><br>\n");

    b.reverse();
    document.write(b.join(' ') + "<br><br>\n");
</script>
```

Useful Method
push()/pop()
join()/split()
sort()/reverse()

There are familiar faces …

See “array_function.html”

JavaScript – String

- You know every scripting language treats string specially.

```
<html>
<script>

    var a = new String();
    var b = "";

    document.write(a == b);
    document.write(a === b);

</script>
</html>
```

Both give you an empty string.

Compared by values.

Compared if they are the same object.

See “string_compare.html”

JavaScript – String

- Since the operator '+' is overloaded…

```
<html>
<script>

    var a = "222";
    var b = "111";

    document.write(a + b);
    document.write("<br>");
    document.write(parseInt(a) + parseInt(b));

</script>
</html>
```

When either one side of '+' is a string, then both sides will become strings.

See "parseInt.html"

JavaScript – Object (1)

```
<html>
<script>

    var rectangle = new Object();

    rectangle.width = 2.0;          // adding an instance variable
    rectangle.height = 3.0;         // adding an instance variable
    rectangle.area =                // adding an instance method...my god...
      function() {
           return (this.width * this.height);
      };

    /* output the result */
         document.write("<h1>Area = " + rectangle.area() + "</h1>");

</script>
</html>
```

Let's have a customized object.

<h1>Area = 6</h1>

See "object_example1.html"

JavaScript – Object (2)

```
<html>
<script>

    function Rectangle(w, h)
    {
         this.width = w;
         this.height = h;
         this.area = function() {
             return (this.width * this.height);
         };
    }  /** no return value is needed. **/

  /* output the result */

    var rectangle = new Rectangle(2, 4.5);
    document.write("<h1>Area = " + rectangle.area() + "</h1>");

</script>
</html>
```

There is no Class in JavaScript. This function is just acting as a constructor, without a Class.

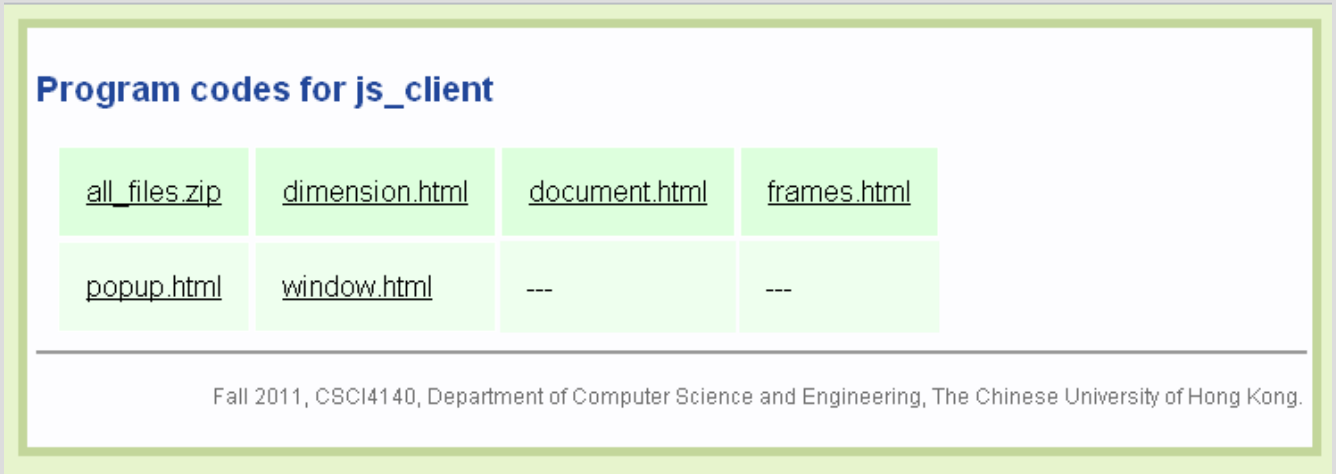
<h1>Area = 9</h1>

See "object_example2.html"

- Further Readings:
  - Want more weird things?

http://net.tutsplus.com/tutorials/javascript-ajax/top-10-things-that-javascript-got-wrong/

See “crazy_example.html”



Program codes for js_client

all_files.zip	dimension.html	document.html	frames.html
popup.html	window.html	---	---

Fall 2011, CSCI4140, Department of Computer Science and Engineering, The Chinese University of Hong Kong.

http://appsrv.cse.cuhk.edu.hk/~csci4140/cgi-bin/js_client/

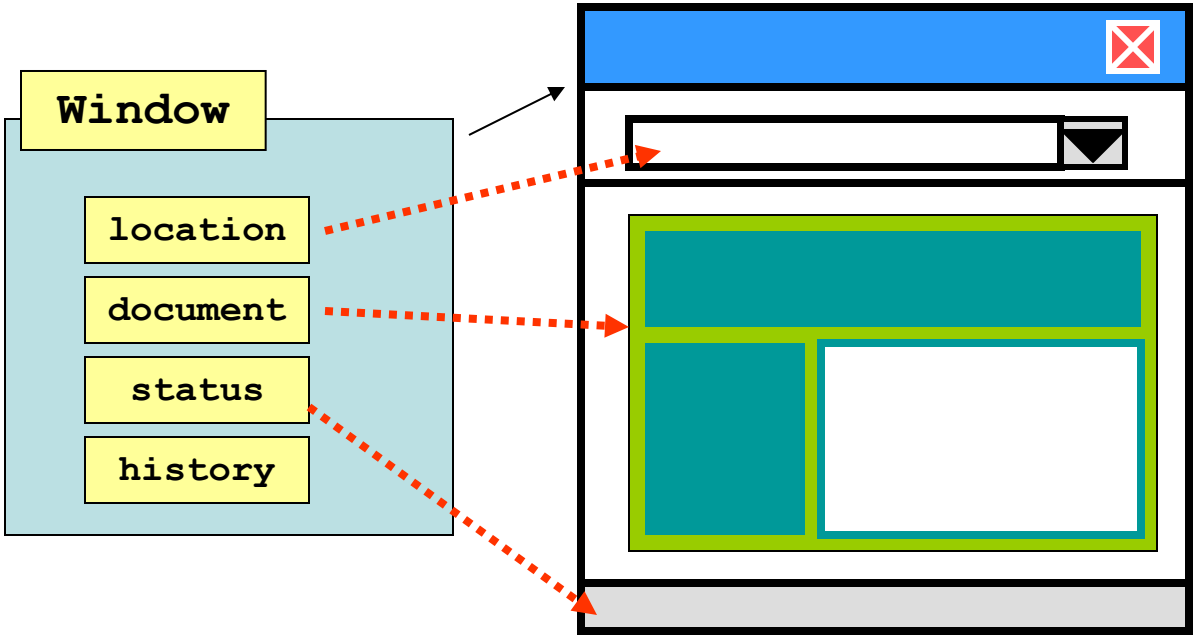
Client-side JavaScript

- The starting point...Window & its members

What is Client-Side JavaScript?

- To use JavaScript to control (nearly) everything in a browser.
  - Client-side JavaScript models a browser in an OO way…

E.g., "**window**" is **a default object**; it has some famous member objects (as instance variables).



What is Client-Side JavaScript?

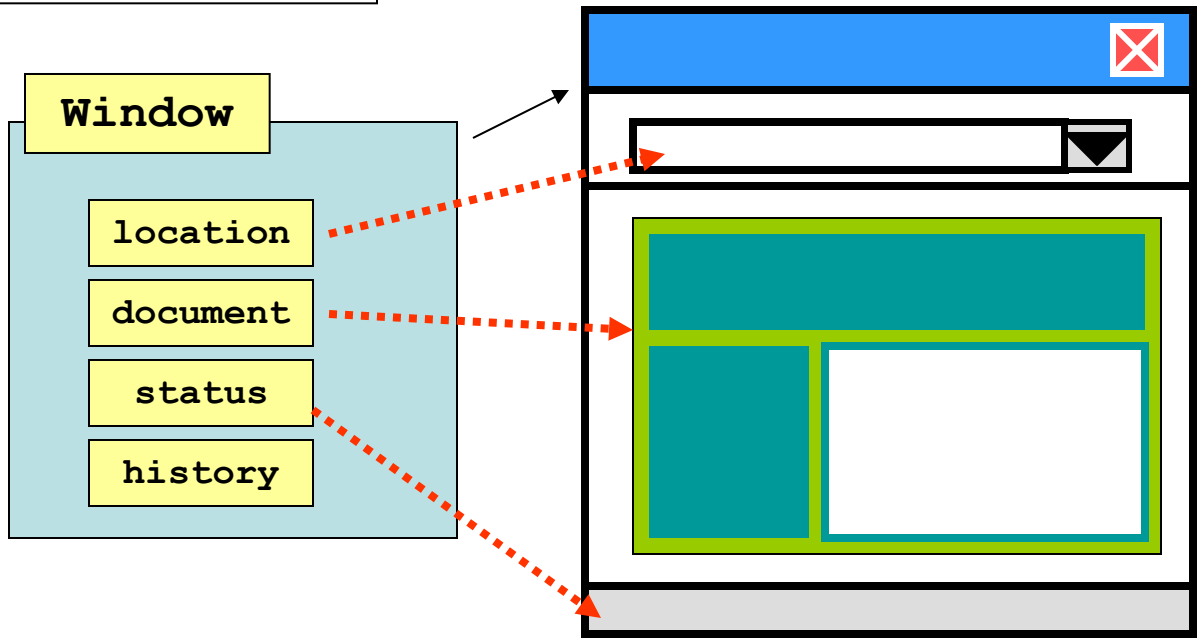
- Two questions:

Question 1

How could I can possibly memorize all the members, not to mention the instance methods, of "**window**"?!

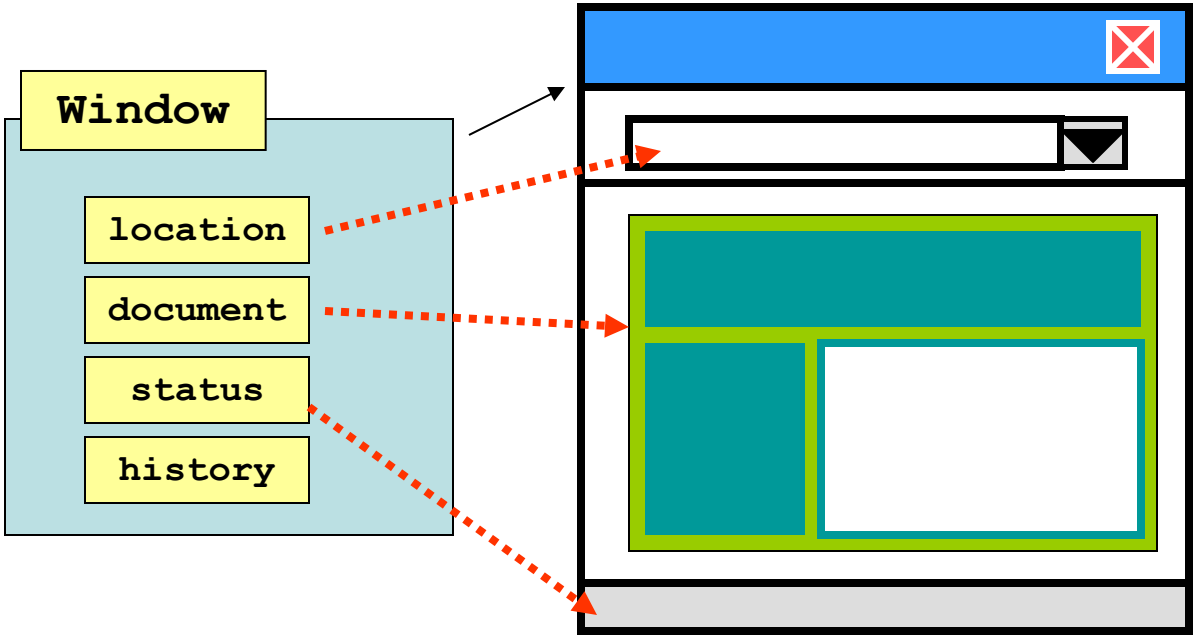
Question 2

Who defines the members? I guess it is the browser.....so **MS could change the fields so as to discourage people to write for Firefox?!**



What is Client-Side JavaScript?

- Document Object Model, DOM.
  - There is a standard that solves the mentioned two problems.
  - Take a look: http://www.w3schools.com/jsref/default.asp



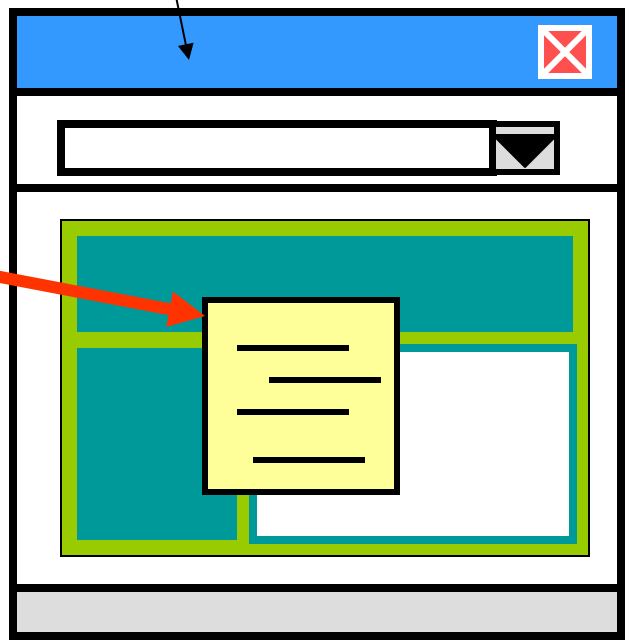
Client-side JS - Window

- The “window” is the global execution context...

```
<script>
document.write( this + “<br>”);
document.write(this === window);

for ( var i in this ) {
    document.write(
        "this." + i + " = "
        + this[i]
        + "<br><br>"
    );
}
```

THIS!



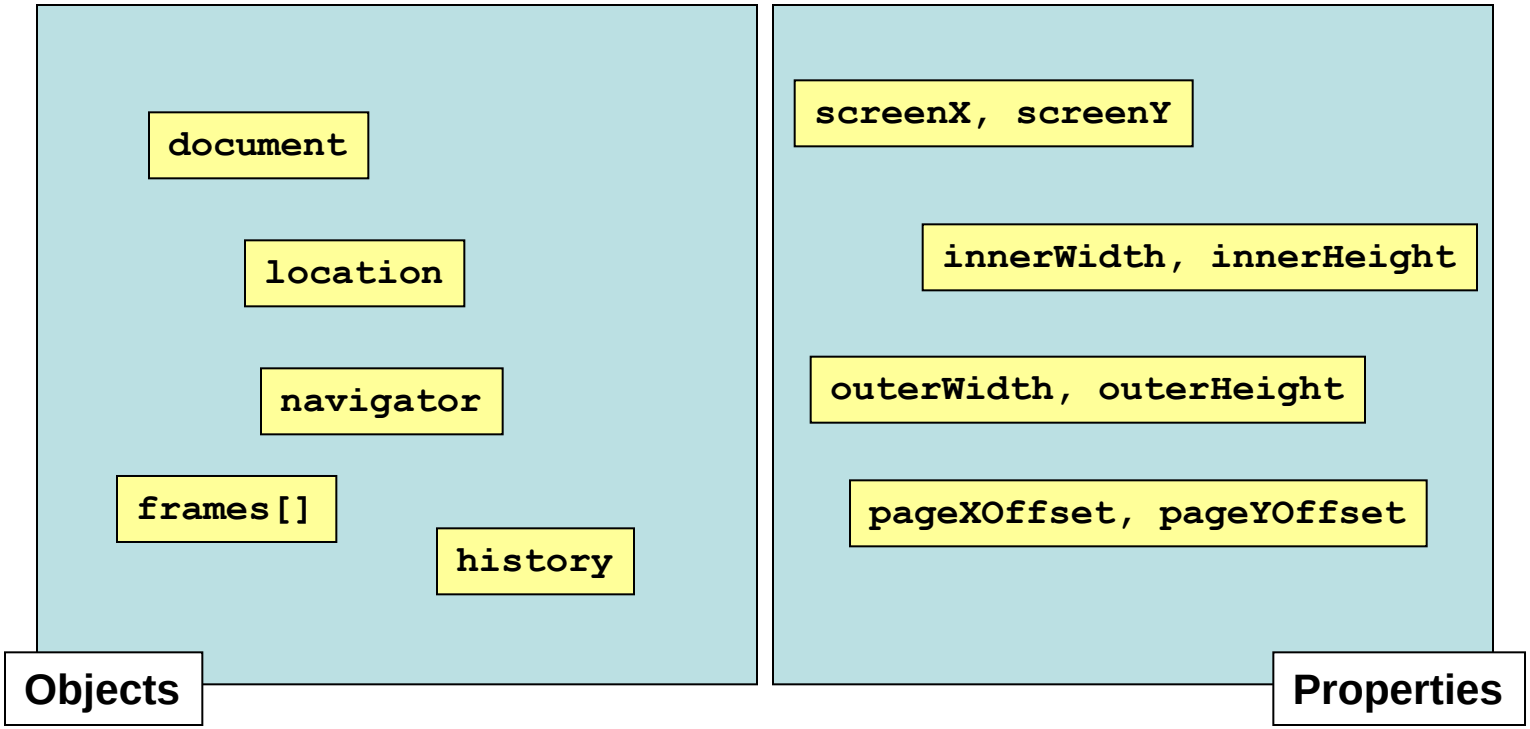
array == object...with one difference:

- array will have numeric indices;
- object would never have members with numeric names.

See “window.html”

Window's objects & properties

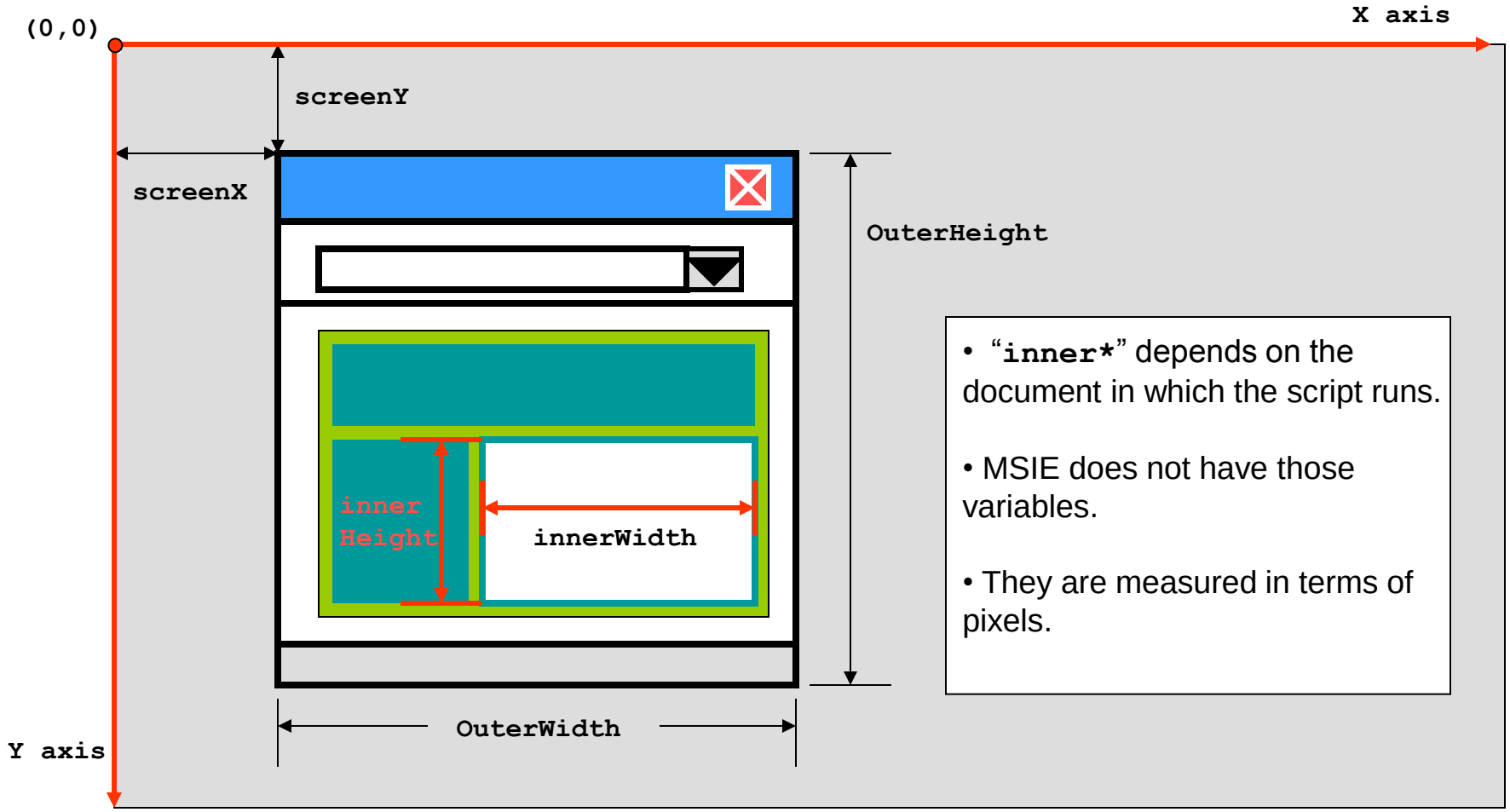
- There are tons of them…



See "document.html", "frames.html"

Window's objects & properties

- There are tons of them…

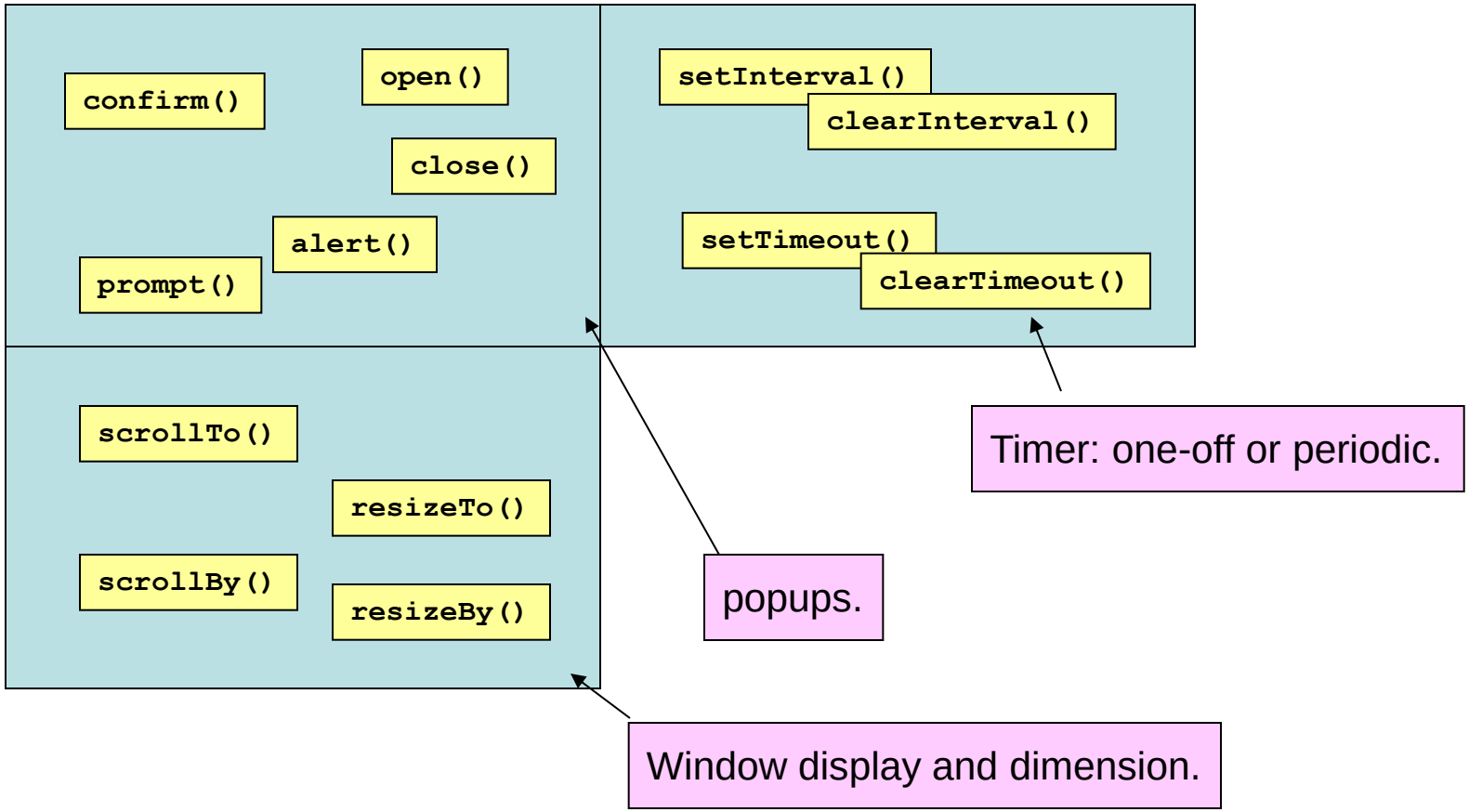


- "**inner***" depends on the document in which the script runs.
- MSIE does not have those variables.
- They are measured in terms of pixels.

See "dimension.html"

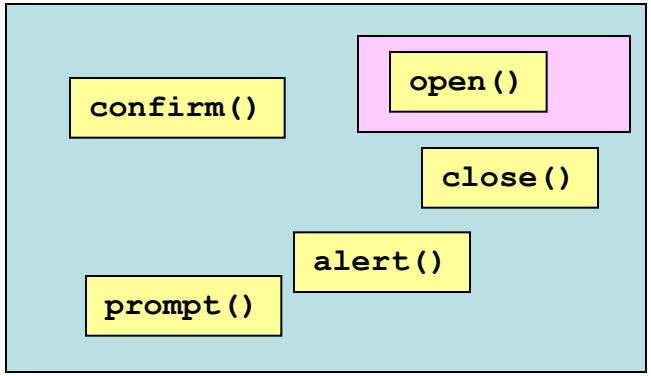
Window's methods

- There are tons of them...



Window's methods

- There are tons of them…



By the way, do you know when will the popup blocker work?

Only when you are not explicitly asking for the popup!

```
<html>
<script>
function popup() {
    window.open(
      "http://www.cse.cuhk.edu.hk");
}
popup();
</script>
</html>
```

```
<html>
<script>
function popup() {
    window.open(
      "http://www.cse.cuhk.edu.hk");
}
</script>

<a href="javascript: popup();">
click me</a>

</html>
```

So, which one will trigger the blocker?

See "popup.html"

Program codes for js_dom

DOM.html	all_files.zip	append_innerHTML.html	append_paragraph.html
append_text.html	compatible.html	dom_tree.html	dom_tree.js
dom_tree_top.html	highlight.html	highlight_script.html	selector.html
test_browser.html	---	---	---

Fall 2011, CSCI4140, Department of Computer Science and Engineering, The Chinese University of Hong Kong.

http://demo4140-tywong.rhcloud.com/09_js_dom

Client-side JavaScript

- DOM Scripting

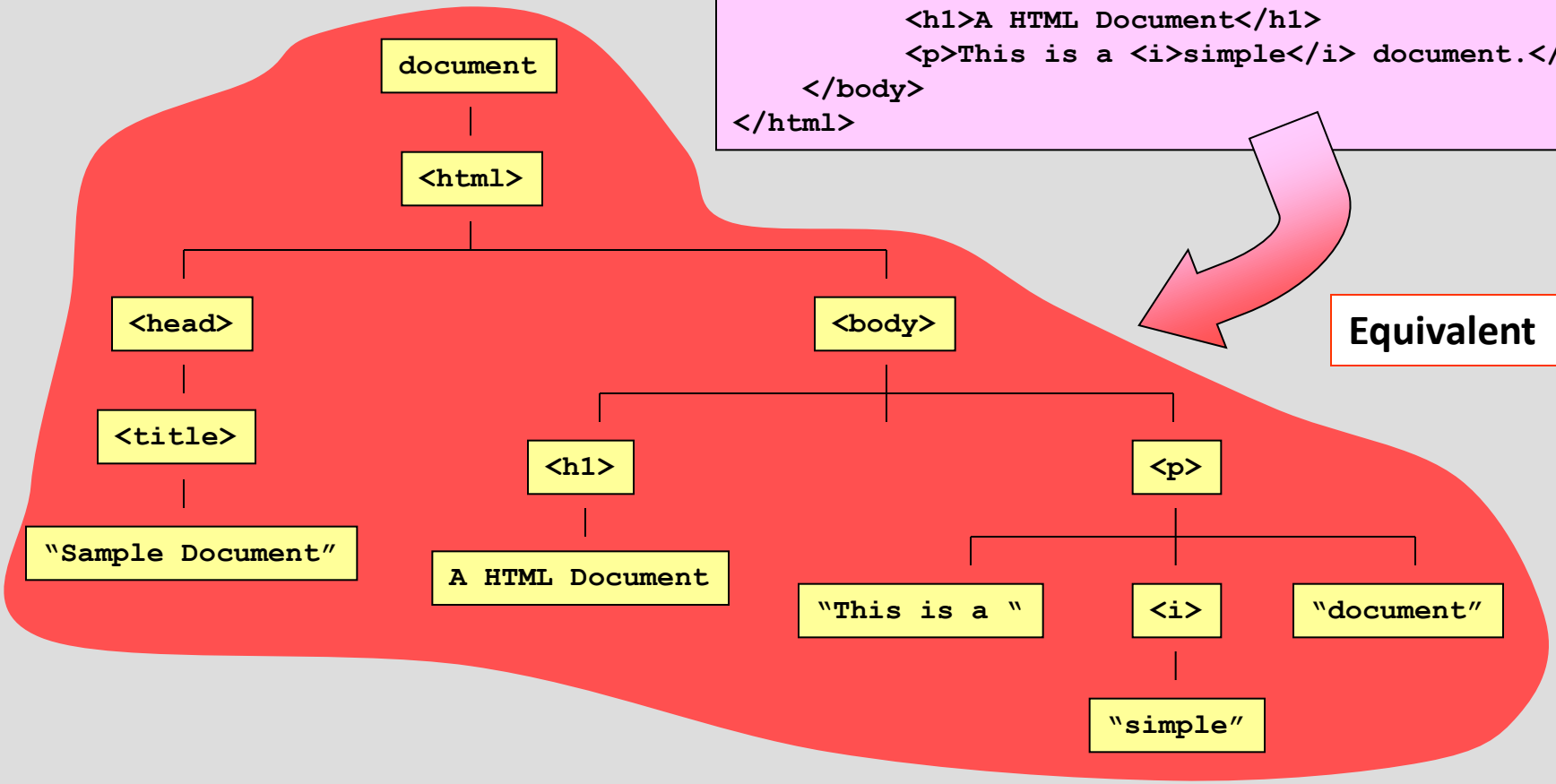
What is DOM Scripting?

- DOM scripting is about the structure of a HTML page…
  - Using JavaScript to **manipulate the tree structure of a HTML page**.
  - Using JavaScript to **build a (partial) HTML page**, instead of using "**`document.write()`**".

HTML document as a tree

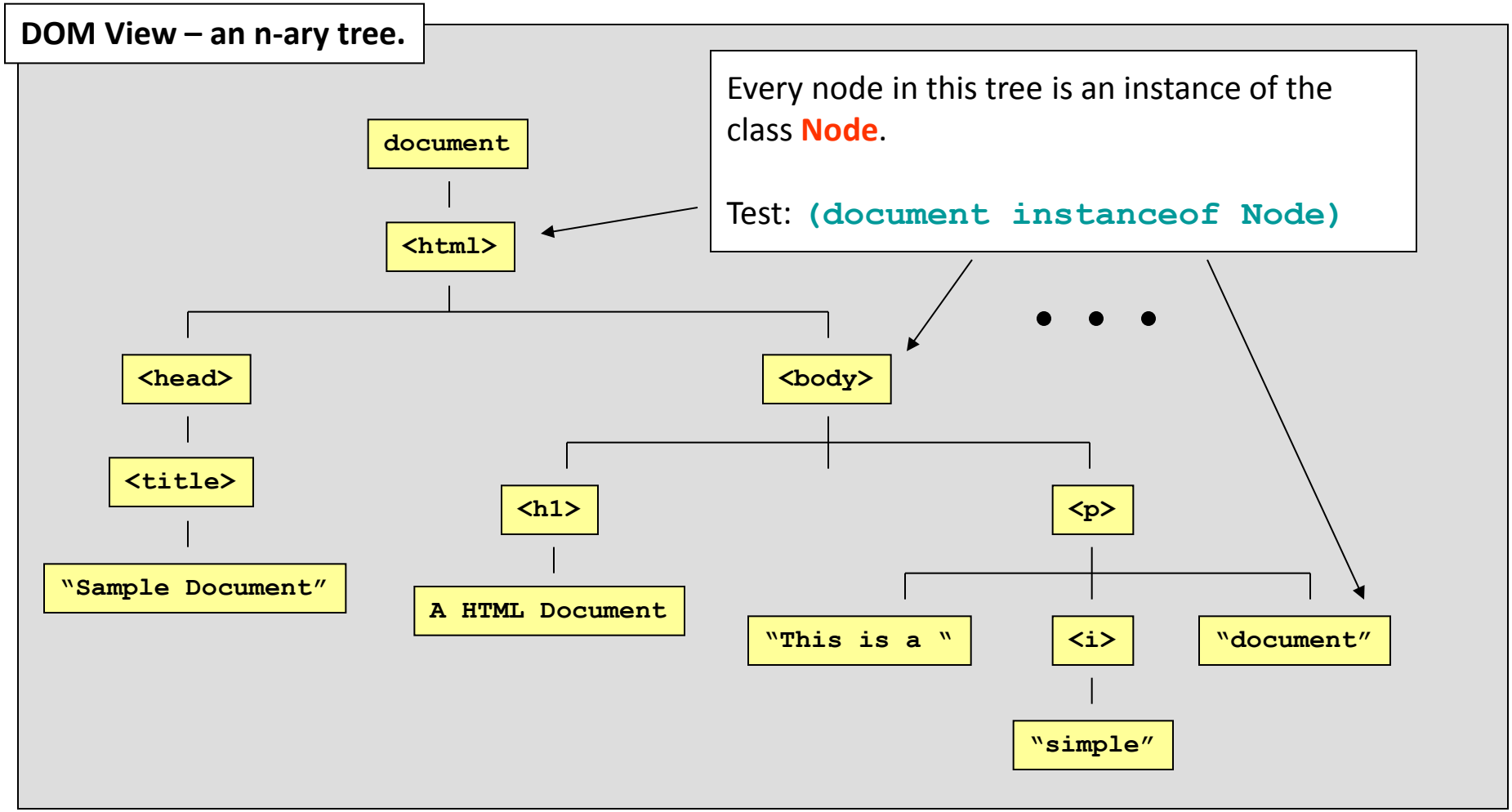
```
<html>
    <head>
        <title>Sample Document</title>
    </head>
    <body>
        <h1>A HTML Document</h1>
        <p>This is a <i>simple</i> document.</p>
    </body>
</html>
```

DOM View – an n-ary tree.



Equivalent

HTML document as a tree



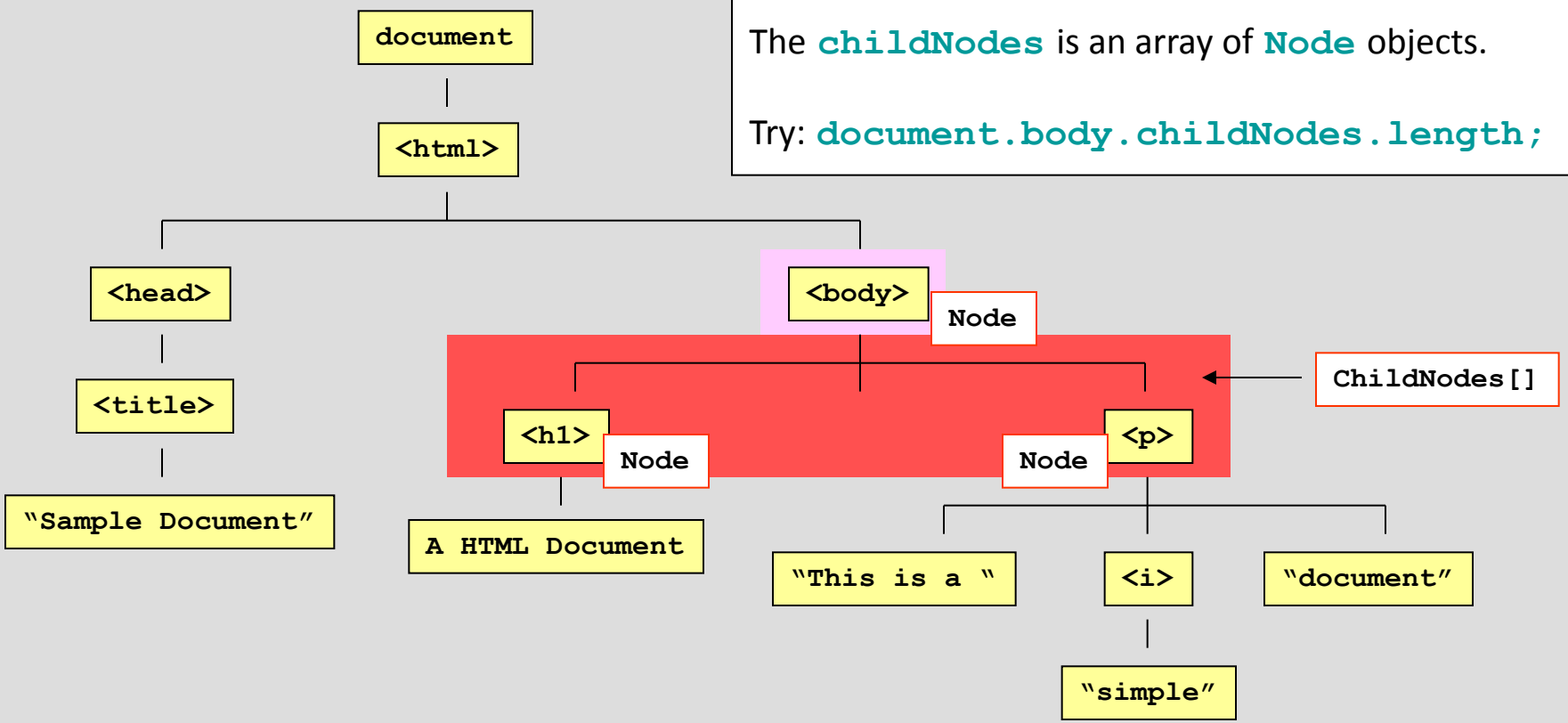
HTML document as a tree

DOM View – an n-ary tree.

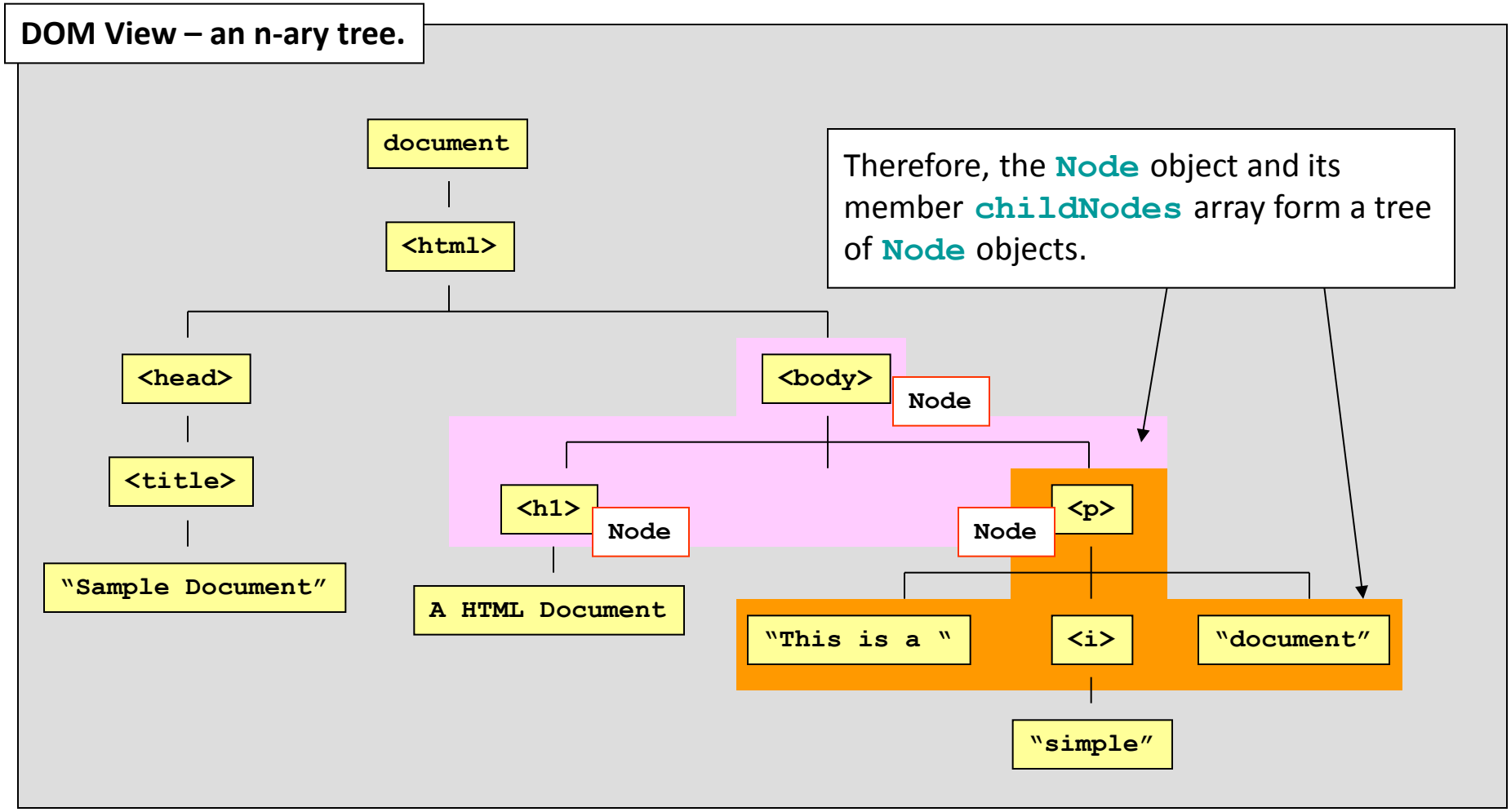
Every node is the root of the n-ary tree specified by the **instance array variable**: **childNodes**.

The `childNodes` is an array of `Node` objects.

Try: `document.body.childNodes.length;`



HTML document as a tree



HTML document as a tree

- A simple recursive algorithm can help traversing the entire tree.

```
function do_show_tree(node_obj) {
    document.write(node_obj.toString());
    for(var i = 0; i < node_obj.childNodes.length; i++)
        do_show_tree(node_obj.childNodes[i]);
}
```

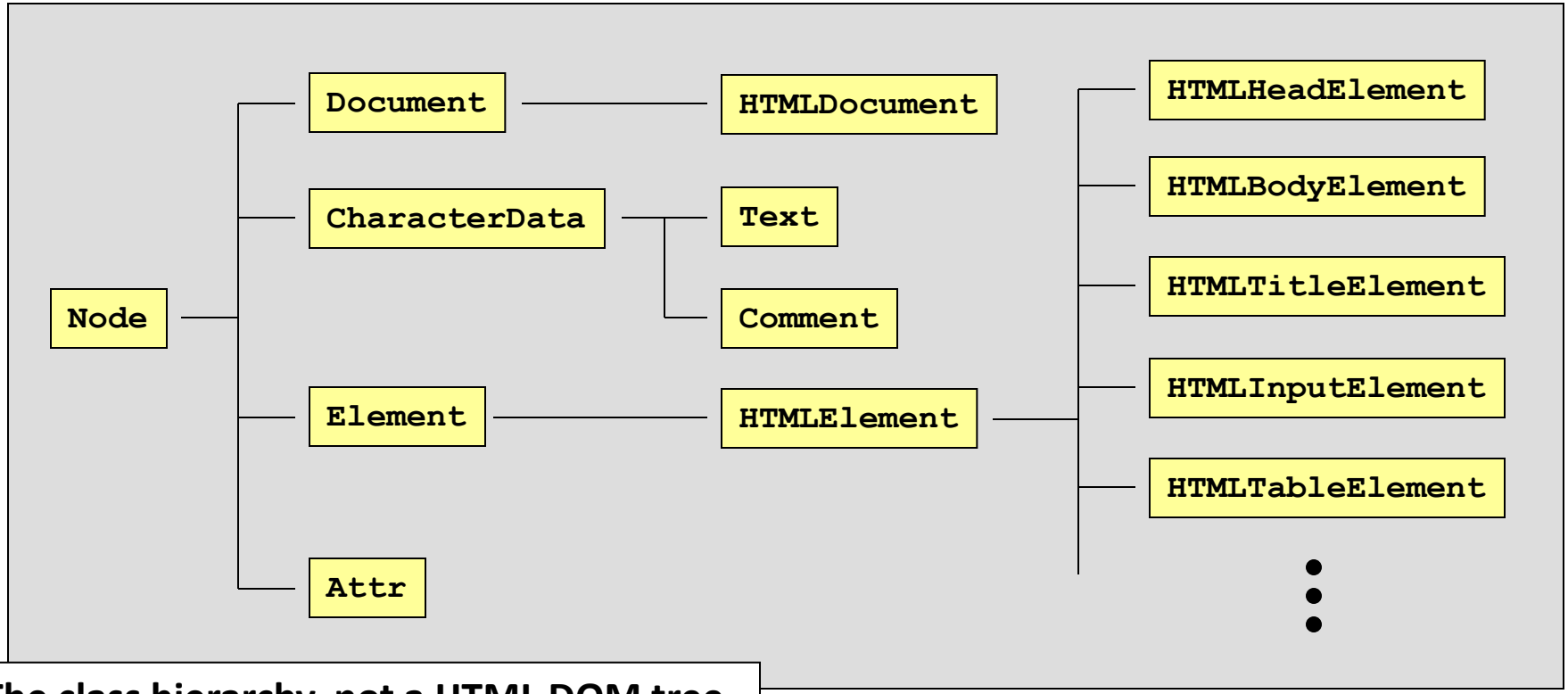
There are chances that the Node objects are **leaf nodes**., i.e., nodes without any children. But, who are they?

`Text object` `Comment object`

See "dom_tree.html", "dom_tree.js"

The Node Hierarchy

- The Node class is actually a parent class.



The class hierarchy, not a HTML DOM tree.

The Node Hierarchy

- The Node class is actually a parent class.

By the latest DOM standard, DOM Level 2, every "**`HTML*Element`**" class maps to a HTML tag:

<head>	HTMLHead Element
<body>	HTMLBodyElement
<title>	HTMLTitleElement
<input>	HTMLInputElement

** **	HTMLDivElement
<table>	HTMLTableElement
<tr>	HTMLTableRowElement
……	……

`HTMLHeadElement`

`HTMLBodyElement`

`HTMLTitleElement`

`HTMLInputElement`

`HTMLTableElement`

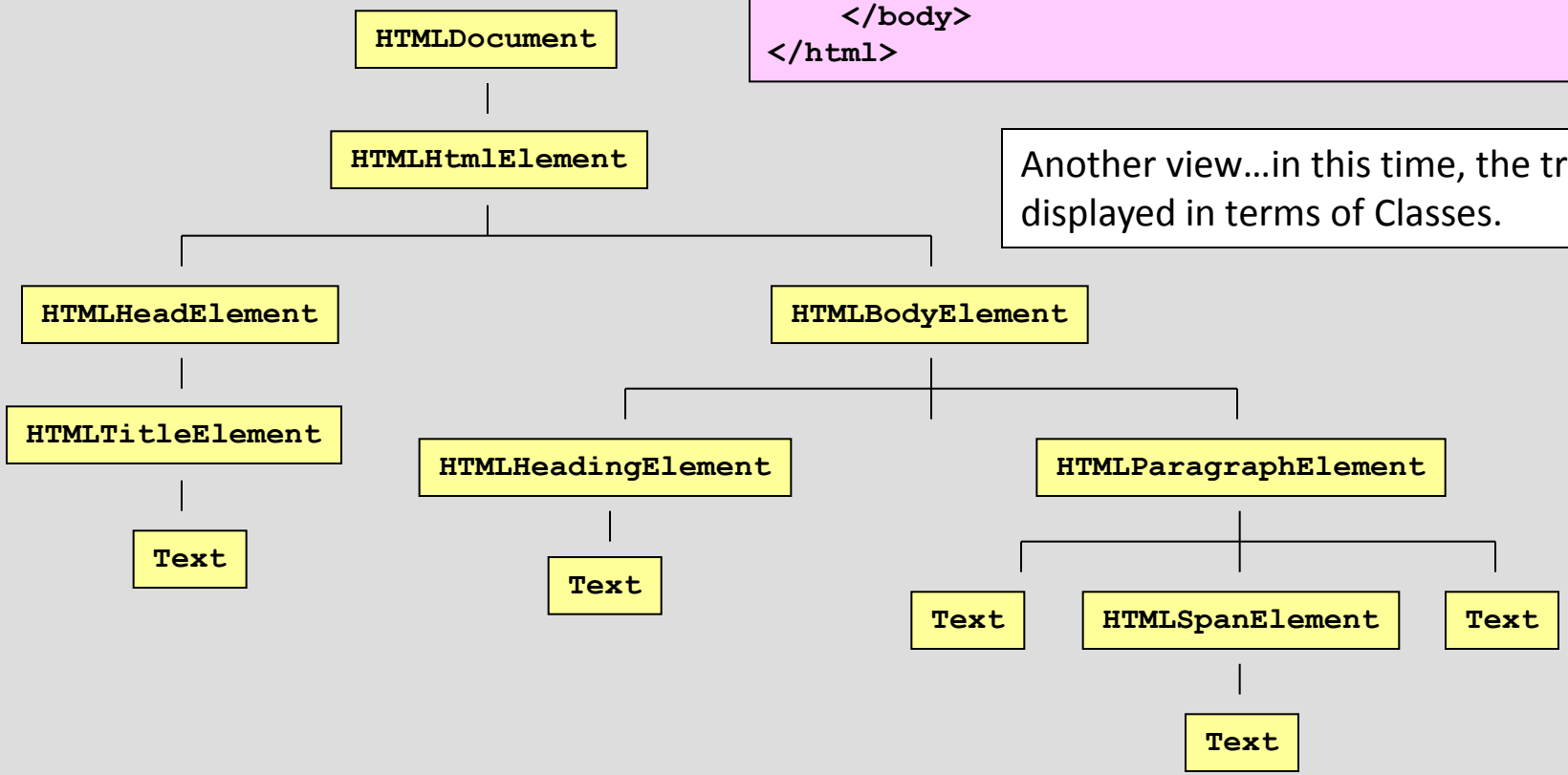
⋮

The class hierarchy, not a HTML DOM tree.

HTML*Element VS HTML Tag

```
<html>
    <head>
        <title>Sample Document</title>
    </head>
    <body>
        <h1>A HTML Document</h1>
        <p>This is a <i>simple</i> document.</p>
    </body>
</html>
```

DOM View – an n-ary tree.

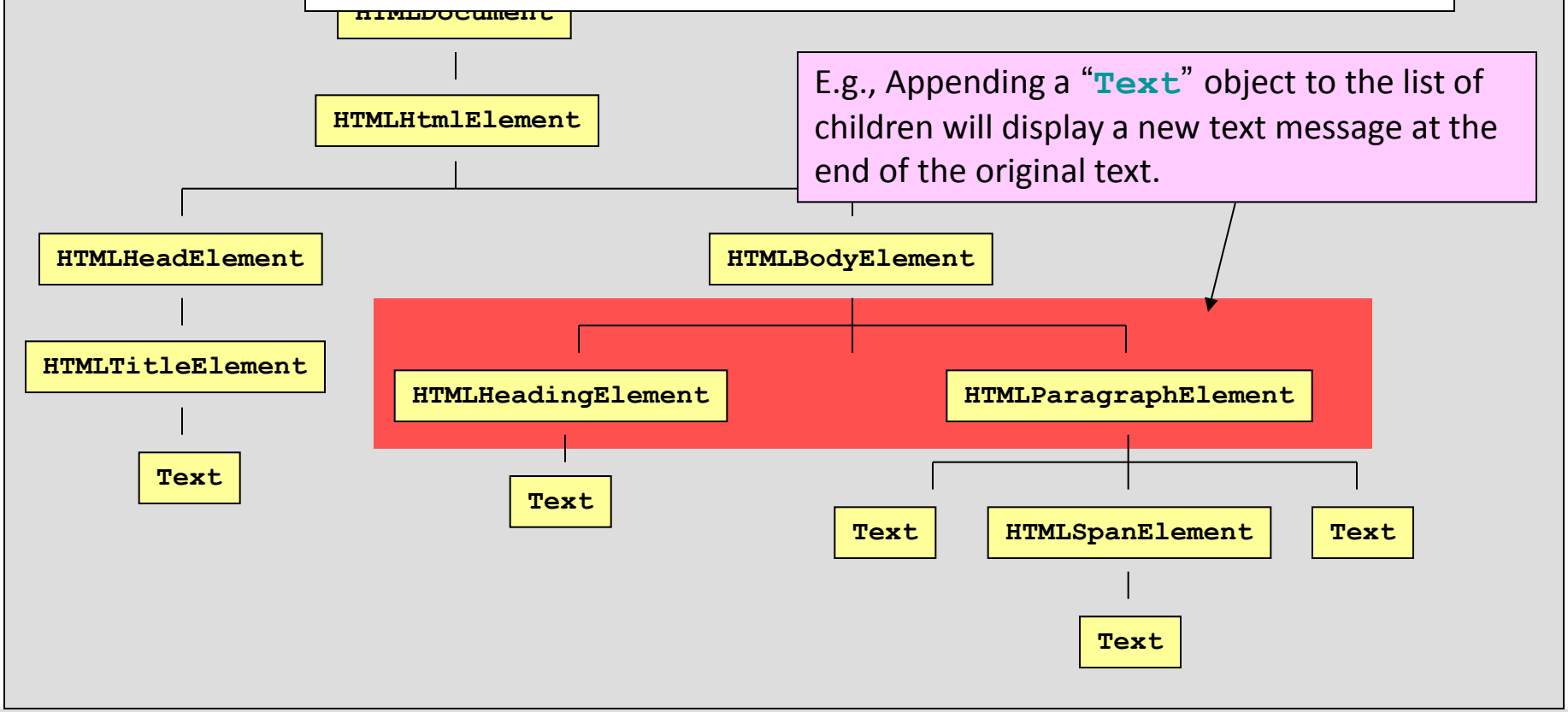


Another view…in this time, the tree is displayed in terms of Classes.

HTML*Element VS HTML Tag

Here comes the important aspect about DOM scripting:

You are free to insert any HTML*Element objects into a Node element.



Construct HTML page using DOM

```html
<html>
    <head>
        <title>Sample Document</title>
    </head>
    <body>
        <h1>A HTML Document</h1>
        <p>This is a <i>simple</i> document.</p>
    </body>
</html>
```

```
<script>
    function append_text() {
        var text_element = document.createTextNode
                                ("This is a new String!!");

        var body_element = document.getElementsByTagName("body")[0];
        body_element.appendChild(text_element);
    }

    window.addEventListener( "load",
                function (e) { setTimeout(append_text, 1000); },
                false );
</script>
```

Append the “Text” object into the tree spanning from the “body” object

Delaying the appearance of the new “Text” object by 1 second, or 1,000 milliseconds.

See “append_text.html”

Construct HTML page using DOM

```
<script>
    function append_text() {
        var p_element = document.createElement("p");
        var text_element = document.createTextNode(
                                "This is in paragraph: " +
                                p_element.toString() );
        p_element.appendChild(text_element);

        var body_element = document.getElementsByTagName("body")[0];
        body_element.appendChild(p_element);
    }

    window.addEventListener("load",
                function (e) { setTimeout(append_text, 1000); },
                false);
</script>
```

Do you know what it is doing?

See "append_paragraph.html"

Construct HTML page using DOM

```
<script>
    function append_text() {
        var p_element = document.createElement("p");
        p_element.innerHTML = 
                          "This is in paragraph: " +
                          p_element.toString();


        var body_element = document.getElementsByTagName("body")[0];
        body_element.appendChild(p_element);
    }

    window.addEventListener("load",
                 function (e) { setTimeout(append_text, 1000); },
                 false);
</script>
```

You can **access the HTML codes** of a specific element.

An alternative…

See "append_innerHTML.html"

Construct HTML page using DOM

- See? DOM scripting allows the **dynamic creation of components** in a page!

- This is essential in AJAX-based system because…
  - Old components may vanish or be replaced.
  - New components may be created and be displayed.

```
appendChild(new_child)
replaceChild(new_child, old_child)
insertBefore(new_child, ref_child)
removeChild(old_child)
```

Add/remove Nodes in a DOM tree.

DOM Conformance

- Did you find that your IE can run the programs?
  - No.
  - It is because IE does not work well with new standard…

This tests whether "`addEventListener`" is undefined or not.

```
<script>

if(window.addEventListener)
        document.write("This is a good browser!");
else
        document.write("Your browser is defective by design!");

</script>
```

DOM Conformance

- E.g., "**`addEventListener`**" is valid only in DOM Level 2 standard.
  - IE 9 "*promises*" to support a partial set of the DOM Level 2 standard.
  - But...it is still using "**`attachEvent`**" ...

This tests whether "`addEventListener`" is undefined or not.

```
<script>

if(window.addEventListener)
        document.write("This is a good browser!");
else
        document.write("Your browser is defective by design!");

</script>
```

See "test_browser.html"

DOM Conformance

- To be a "*responsible*" program(mer),
  - Embed **conformance coding** as shown in the following example…
  - Or, stick with the old standards…but the up-to-date standard will be adopted eventually.

- That's the **most brain-damaging part** about client-side JavaScript programming…

```
if(window.screenX == undefined) {
  // IE...
    document.write("IE's position = ("
        + window.screenLeft + "px, " + window.screenTop + "px)<br>");
}
else {
    document.write("Window's position = ("
         + window.screenX + "px, " + window.screenY + "px)<br>");
}
```

See "compatiable.html"

DOM Conformance

- References:

  http://www.quirksmode.org/compatibility.html

  http://www.webdevout.net/

  - and, the best thing to do is to do experiments by yourself!
  - or, using JavaScript library such as jQuery.

Locating DOM Objects

- The best thing about the DOM standard is for scripts to **locate objects in a programmable way**.

```
<a href="http://www.cse.cuhk.edu.hk" id=dept name=dept>Dept</a>
```

Node.tagName

Except "**Text**" and "**Comment**", every node inside a document has a **tagName**.

Node.id

"**id**" plays a vital role in DOM scripting.

Because an ID is required to be unique!

Node.name

You have met "**name**" before...when you are writing "**form**".

Locating DOM Objects

`document.getElementById(String)`

Very helpful

- It gives you an object that is having the "**`id`**" specified by the input string.

- If no object is found, the method returns "**`null`**".

`document.getElementsByTagName(String)`

Quite helpful

- It gives you an array of objects that are having the "**`tagName`**" specified by the input string.

- If no object is found, the output array has 0 element.

`document.getElementsByName(String)`

Not helpful

- It gives you an array of objects that are having the "**`name`**" specified by the input string.

- If no object is found, the output array has 0 element.

See "highlight.html"

Locating DOM Objects

Sorry....

This example contains tons of **event handling stuffs**:

- dynamically installing and removing event handler;

- mouse click event;

- select object's change event;

We'll go over them later.

See "highlight.html"

Locating DOM Objects – in HTML5

`document.querySelector(selector);`

Extremely helpful

- It gives you the first object that matches the "**`selector`**".

- If no object is found, the method returns "**`null`**".

- The array version: **`document.querySelectorAll(selector);`**

How jQuery reacts: http://blog.jquery.com/2010/10/16/jquery-143-released/

All selectors: http://www.w3.org/TR/css3-selectors/#selectors

See "selector.html"