

# *Client-side Technologies*

*Dr. Niveen Nasr El-Den*  
*iTi*



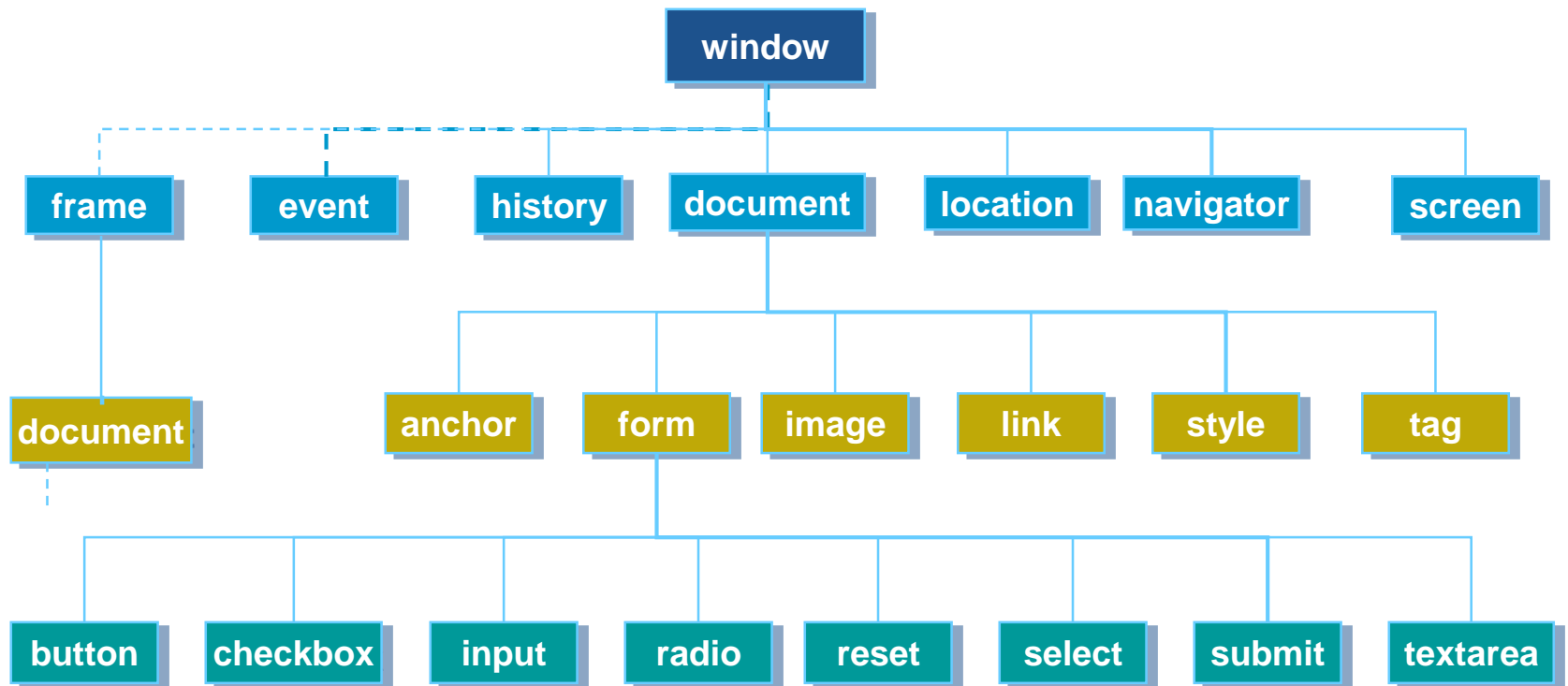
*Day 7*



# *JavaScript Fundamentals* cont.

# Model Hierarchy

BOM is a larger representation of everything provided by the browser including any other functionality the browser may expose to JavaScript.



# Document Object

- The *document* object represents the entire HTML document and can be used to access all elements in a page.
- A *page* is what appears within the browser window.
- So, every window is associated with a document object.
- Document Object has its own set of Properties, Collections, Methods & Event handlers.

# DOM & document Object

- The **document** object in the **BOM** is the top level of the **DOM** hierarchy.
- DOM is a representation of the whole document as nodes and attributes.
- You can access each of these nodes and attributes and change or remove them.
- You can also create new ones or add attributes to existing ones.

**DOM** is a **subset** of **BOM**.

In other word: **the document is yours!**

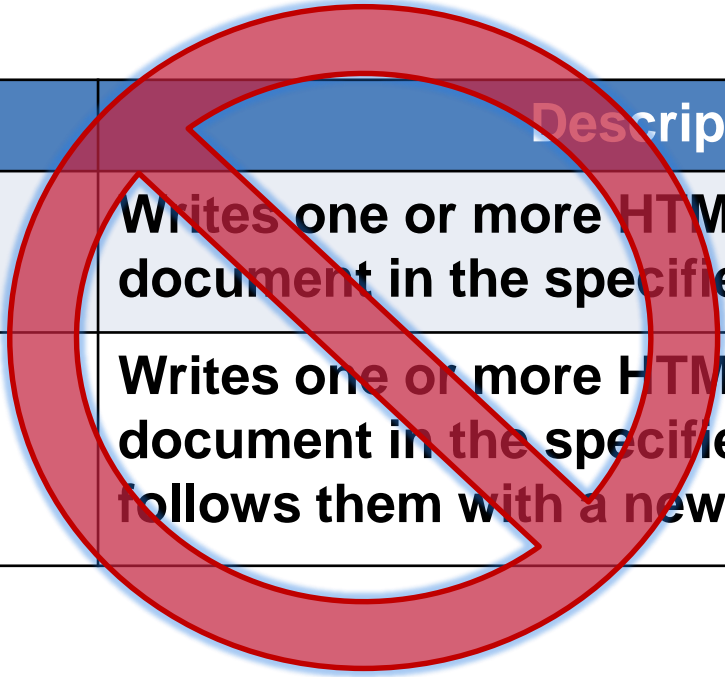
# DOM

- DOM Stands for Document Object Model.
- W3C standard.  
[https://developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model)
- Its an API that interact with documents like HTML, XML.. etc.
- Defines a standard way to access and manipulate HTML documents.
- Platform independent.
- Language independent

# Document Methods

**BOM**

Method	Description
<b>write()</b>	<b>Writes one or more HTML expressions to a document in the specified window.</b>
<b>writeln()</b>	<b>Writes one or more HTML expressions to a document in the specified window and follows them with a new line character.</b>

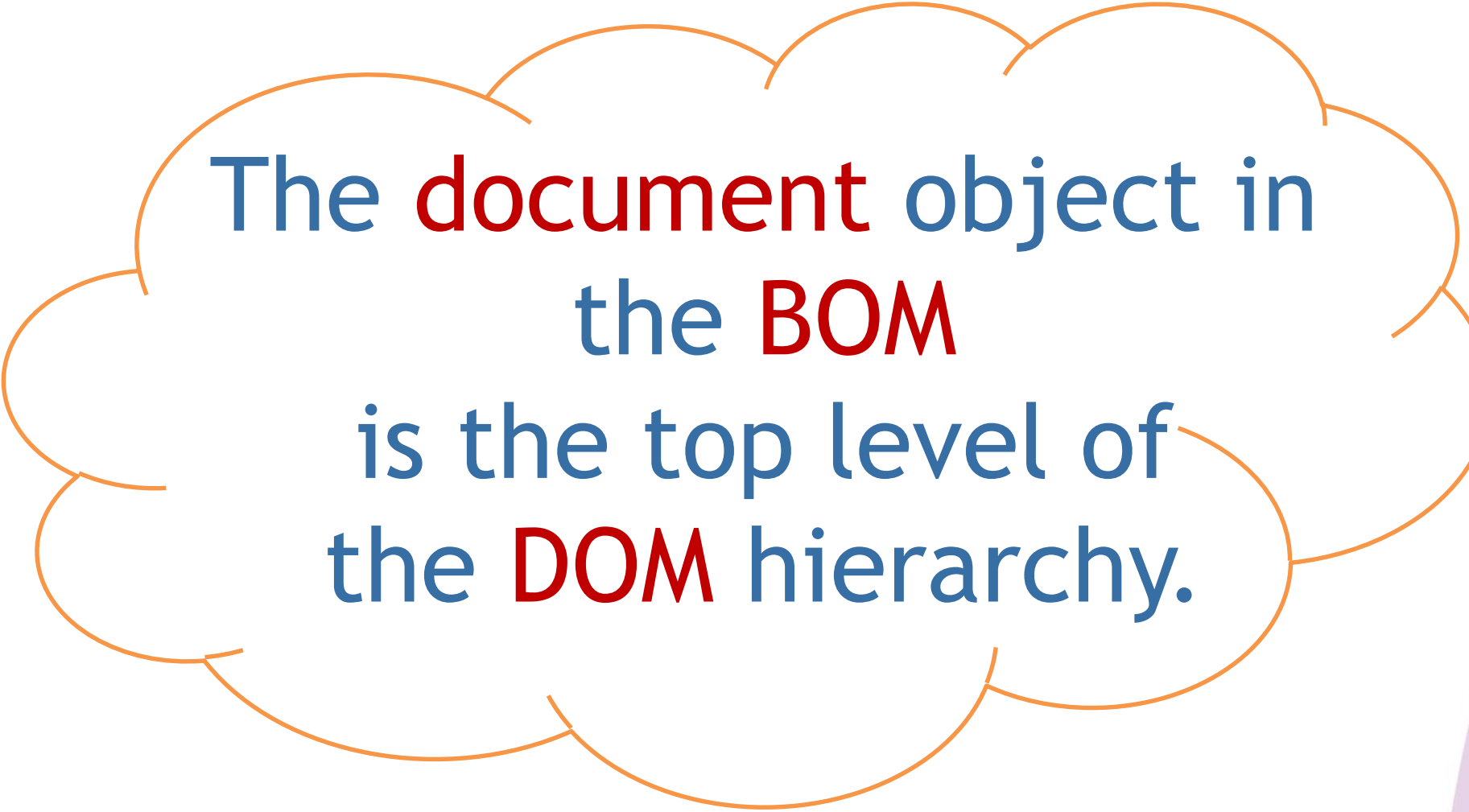


**Example!**



# Document Properties

Property	Description
<b>bgColor</b>	A string that specifies the background color.
<b>fgColor</b>	A string that specifies the text color.
<b>linkColor</b>	The color of text for a link that the user has not yet visited.
<b>vlinkColor</b>	The color of text for a link that the user has already visited.
<b>alinkColor</b>	The color of text for a link that the user clicks.
<b>title</b>	A string that specifies the contents of the TITLE tag.
<b>cookie</b>	A string containing the name/value pair of cookies in the document.



The **document** object in  
the **BOM**  
is the top level of  
the **DOM** hierarchy.

# DOM

methods and properties, allows  
accessing any element on the page,  
modify, delete or remove elements,  
or add new ones using  
document object

# Document Methods

W3C

*for accessing document elements*

Method	Description
getElementById("id")	For accessing any element on the page via its ID attribute
getElementsByName("name")	Returns a collection of objects with the specified name
getElementsByTagName("tagName")	Returns a collection of objects with the specified tagname

**Example!**

# New HTML5 Selectors

- In HTML5 we can select elements by ClassName

```
var elements = document.getElementsByClassName('entry');
```

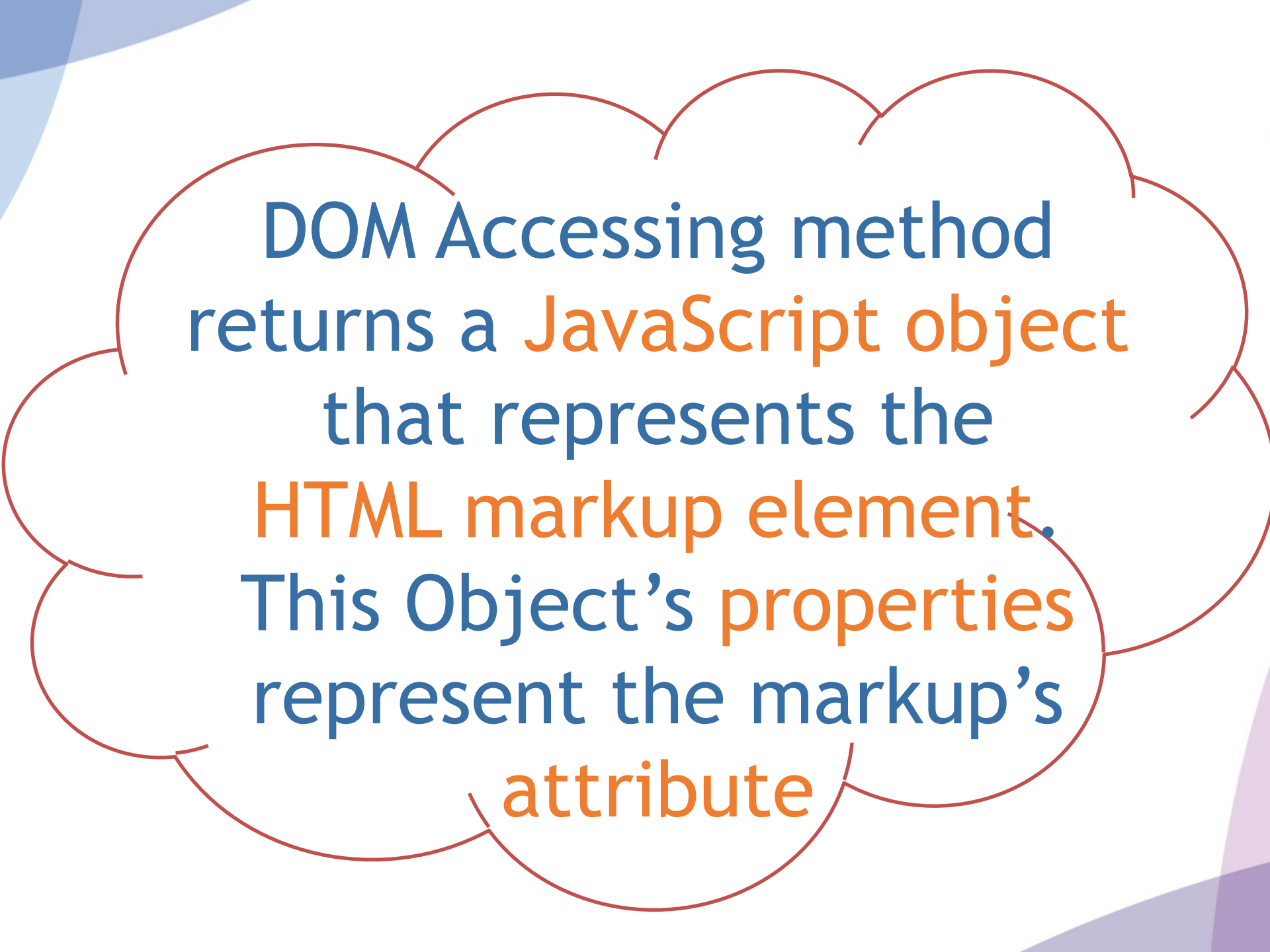
- Moreover there's now possibility to fetch elements that match provided CSS syntax

```
var elements = document.querySelectorAll(".someClasses");
```

```
var elements = document.querySelectorAll("div,p");
```

```
var elements = document.querySelector("#someID");
```

```
var first_td = document.querySelector("span");
```



DOM Accessing method  
returns a JavaScript object  
that represents the  
HTML markup element.  
This Object's properties  
represent the markup's  
attribute



We can access any  
markup content via  
`innerHTML`, `innerText`, or  
`textContent`  
properties

# Document Collection

---

- links, anchors, images, forms are collection/array containing all occurrences of those objects within the document.
- Since they are treated as arrays, they have the *length* property which specifies the number of entries in the collection/array.
- To access a specific entry in any of these collections, we can use either their index or name.



# Document Collection

Collection	Description
<b>forms[ ]</b>	<b>An array containing an entry for each form in the document</b>
<b>images[ ]</b>	<b>An array containing an entry for each image in the document</b>
<b>anchors[ ]</b>	<b>An array containing an entry for each anchor in the document.</b>
<b>links[ ]</b>	<b>An array containing an entry for each link in the document.</b>

- An item from an object collection can be referenced in one of the following ways:
  1. `collection[i]`
  2. `collection.item(i)`
  3. `collection.namedItem(id)`
  4. `collection["name"]`
  5. `collection["id"]`
  6. `collection.name`
- Example:
  - `document.forms[0]`
  - `document.forms["myForm"]`
  - `document.forms["frmId"]`
  - `document.myForm`

# Document Event Handler

---

<b>onclick</b>
<b>ondblclick</b>
<b>onkeydown</b>
<b>onkeypress</b>
<b>onkeyup</b>
<b>onmousedown</b>
<b>onmouseup</b>

# Image

---

- The Image object is an image on an HTML form, created by using the HTML '**IMG**' tag.
- Any images created in a document are then stored in an array in the **document.images** property.
- Image Object has its own set of Properties, Methods & Event handlers.

# Image

- Object Model Reference:

[window.]document.imageName  
[window.]document.imageID  
[window.]document.images[i]  
[window.]document.images[imageName]

document.img1.src="img1.jpg"  
document.images[0].src="img1.jpg"

document.img2.src="img2.jpg"  
document.images[1].src="img2.jpg"

```
<html>
<body>
  ....
  
  
  ...
</body>
</html>
```

# Image Properties & Event handlers

## Properties

<b>name</b>	<b>id</b>	<b>src</b>	<b>height</b>	<b>width</b>
-------------	-----------	------------	---------------	--------------

## Event handlers

<b>onabort</b>	<b>onload</b>	<b>onerror</b>	<b>onclick</b>	<b>ondblclick</b>	<b>onmouseover</b>
----------------	---------------	----------------	----------------	-------------------	--------------------

**Example!**

# Form

- By using the form you have at your disposal; information about the elements in a form and their values.
- A separate instance of the form object is created for each form in a document.
- Objects within a form can be referred to by a numeric index or be referred to by name.
- **Object Model Reference:**
  - [window.]document.formname
  - [window.]document.forms[i]
  - [window.]document.forms["formNAME"]
  - [window.]document.forms["formID"]

# Form

Properties

```
<form  
  [name="formName"]  
  [target="frameName or windowName"]  
  [onsubmit="handlerText Or Function"]  
  [onreset="handlerText Or Function"]  
>  
</form>
```

Events



# Form Properties

Property	Description
<b>elements[ ]</b>	<b>An array containing all of the elements of the form. Use it to loop through form easily.</b>
<b>length</b>	<b>The number of elements in the form.</b>
<b>name</b>	<b>The name of the form.</b>
<b>id</b>	<b>The id of the form.</b>
<b>target</b>	<b>The name of the target frame or window form is to be submitted to.</b>

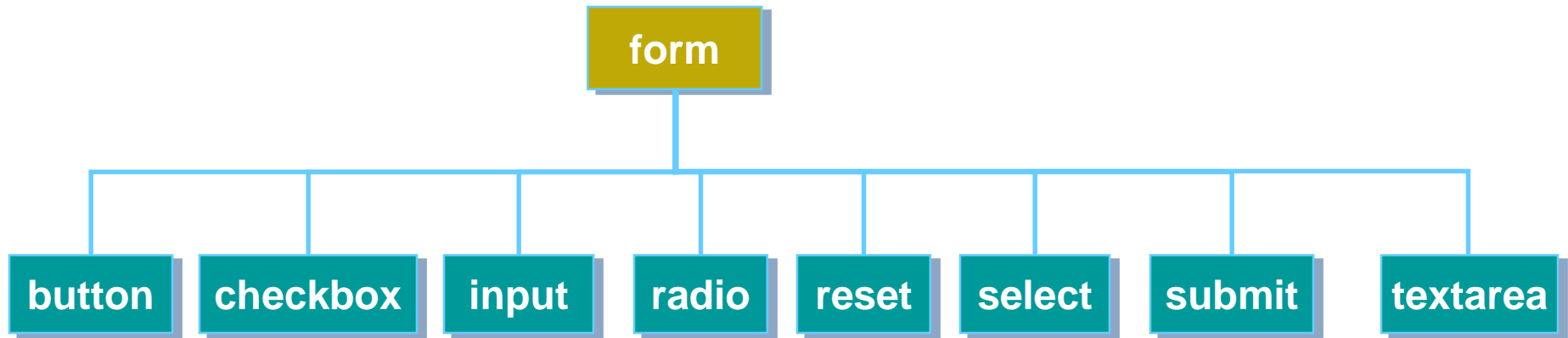
# Form Methods

Method	Description
<b>reset()</b>	<b>Resets the form.</b> <b>Clicking the reset button clears all contents that the user has made..</b>
<b>submit()</b>	<b>Submits a form.</b> <b>Clicking the submit button submits the content of the form to the server</b>

# Form Event Handler

Event	Description
<b>onreset</b>	<b>Code is executed when the form is reset (by clicking on "reset" button)</b>
<b>onsubmit</b>	<b>Code is executed when form is submitted</b>

# Form



# Text

```
<input type="text"  
      id="id"  
      value="string"  
      maxlength="n"  
      size="x"  
>
```

# Text Event Handler

Event	Event Handler	Description
<b>focus</b>	<b>onfocus</b>	<b>Fires when the field gains focus when the user tabs into or clicks inside the control.</b>
<b>blur</b>	<b>onblur</b>	<b>Fires when the field loses focus when the user tabs from or clicks outside the control.</b>
<b>change</b>	<b>onchange</b>	<b>Fires after changing the field value and losing being focused.</b>
<b>input</b>	<b>oninput</b>	<b>Fires every time the field value changes.</b>
<b>select</b>	<b>onselect</b>	<b>Fires when the content of the field being selected.</b>

# Text Methods

Method	Description
<b>blur( )</b>	<b>Removes focus from the field.</b>
<b>focus( )</b>	<b>Assigns focus to the field; places the cursor in the control.</b>
<b>select( )</b>	<b>Selects, or highlights, the content of the control.</b>

**Example!**

# Drop-down lists

```
<select  
  id="id"  
  multiple  
  size="n"  
>  
  <option value="string" selected > label </option>  
  <option value="string2" > label2 </option>  
  ...  
</select>
```



# Drop-down lists Properties

Property	Description
length	The number of options in the list.
selectedIndex	The index number, beginning with 0, of the selected option.
options[ ]	An array of the options in the list. Used to reference properties associated with the options; e.g., options[1].value or options[2].text.
selected	A <b>true</b> or <b>false</b> value indicating whether an option is chosen.
value	The <b>value</b> associated with an option
text   label	The <b>text</b> label associated with an option.

# Drop-down lists Event Handler

Event Handler	Description
<b>onfocus</b>	<b>The control gains focus.</b>
<b>onblur</b>	<b>The control loses focus.</b>
<b>onchange</b>	<b>A different option from the one currently displayed is chosen.</b>

**Example!**

# Radio Button

```
<input type="radio"  
  id="id"  
  name="name"  
  value="string"  
  checked  
>
```

# Radio Button Properties

Property	Description
<b>length</b>	The number of radio buttons with the same name.
<b>checked</b>	A true or false value indicating whether a button is checked.
<b>value</b>	The value attribute coded for a button. A checked button with no assigned value is given a value of "on".

# Radio Button Event Handler

Event Handler	Description
<b>onfocus</b>	<b>The control gains focus.</b>
<b>onblur</b>	<b>The control loses focus.</b>
<b>onclick</b>	<b>The button is clicked.</b>

**Example!**

# Checkbox

```
<input type="checkbox"  
  id="id"  
  name="name"  
  value="string"  
  checked  
>
```

# Button

---

```
<input type="button"  
  id="id"  
  value="string"  
>
```

# Button Event Handler

Event Handler	Description
<b>onclick</b>	<b>The mouse is clicked and released with the cursor positioned over the button.</b>
<b>ondblclick</b>	<b>The mouse is double-clicked with the cursor positioned over the button.</b>
<b>onmousedown</b>	<b>The mouse is pressed down with the cursor positioned over the button.</b>
<b>onmouseout</b>	<b>The mouse cursor is moved off the button.</b>
<b>onmouseover</b>	<b>The mouse cursor is moved on top of the button.</b>
<b>onmouseup</b>	<b>The mouse button is released with the cursor positioned over the button.</b>



# Reminder DOM References

- Use **this** keyword is used to refer to the current object.
  - e.g. the calling object in a method.

- Self reference to the object is used :

```
<input type="text"  
      onfocus = "this.value='You are in!'" />
```

- Passing current Object as a function parameter:

```
function myFunction(myObject) {  
    myObject.value = "In the function!!"  
}  
  
<input type="text" onclick="myFunction(this)" />
```



# *Events & Event Handlers*

# Events

- We have the ability to create dynamic web pages by using *events*.
- Events are **actions** that **respond** to user's specific actions.
- Events are controlled in JavaScript using **event handlers** that indicate what **actions** the browser takes in **response** to an event.
- Examples for different events:
  - A mouse click
  - A web page loading
  - Taking mouse over an element
  - Submitting an HTML form
  - A keystroke on your keyboard

# Events

- Event handlers are created as **attributes** added to the HTML tags in which the event is triggered. (first way of binding an event handler)
- An Event handler adopts the event name and appends the word **“on”** in front of it.  
**< tag onevent = “JavaScript commands;” >**
- Thus the **“click”** event becomes the **onclick** event handler.

# Mouse Events

Event handler	Description
<b>onmousedown</b>	when pressing any of the mouse buttons.
<b>onmousemove</b>	when the user moves the mouse pointer within an element.
<b>onmouseout</b>	when moving the mouse pointer out of an element.
<b>onmouseup</b>	when the user releases any mouse button pressed
<b>onmouseover</b>	when the user moves the mouse pointer over an element.
<b>onclick</b>	when clicking the left mouse button on an element.
<b>ondblclick</b>	when Double-clicking the left mouse button on an element.
<b>ondragstart</b>	When the user has begun to select an element

# Keyboard Events

Event handler	Description
onkeydown	When User presses a key
onkeypress	When User presses a key other than Modifiers (ctrl, shift, ..etc.)
onkeyup	When User releases the pressed a key

# Other Events

Event handler	Description
<b>onabort</b>	The User interrupted the transfer of an image
<b>onblur</b>	when loosing focus
<b>onfocus</b>	when setting focus
<b>onchange</b>	when the element has lost the focus and the content of the element has changed
<b>onload</b>	a document or other external element has completed downloading all the data into the browser
<b>onunload</b>	a document is about to be unloaded from the window
<b>onerror</b>	When an error has occurred in a script.
<b>onmove</b>	when moving the browser window

# Other Events

Event handler	Description
onreset	When the user clicks the <b>form</b> reset button
onsubmit	When the user clicks the <b>form</b> submit button
onscroll	When the user adjusts an element's scrollbar
onresize	When the user resizes a browser window
onhelp	When the user presses the F1 key
onselect	When selecting text in an input or a textarea element
onselectstart	When the user is beginning to select an element



# Binding Events

---

- **Binding Event Handlers to Elements can be:**
  - 1.Event handlers as tag attribute**
  - 2.Event handlers as object property**

# Event handlers as tag attribute

```
<input type=button value="click me" name=b1  
  onclick="alert('you have made a click')">
```

OR

```
<script>  
function showmsg(){  
    alert("you have made a click")  
}  
</script>
```

```
<input type=button value="click me"  
  onclick="showmsg()" />
```

**Example!**

# Event handlers as object property

```
<body>
  <form>
    <input type=button name='b1' value="Click ME" />
  </form>
</body>
```

```
<script>
function showAlert (){
  alert("you have clicked me")
}
document.forms[0].b1.onclick=showAlert
</script>
```

**Note: there are no parentheses**



# Event handlers as object property

```
<body>
  <form>
    <input type=button name='b1' value="Click ME" />
  </form>
</body>
```

```
<script>
document.forms[0].b1.onclick=function showAlert (){
    alert("you have clicked me")
}
</script>
```

# Event handlers return value

```
<a href="1.htm" onclick="myFunc(); return false">
```

This will make the browser ignore the action of href

- Another way that can also make the browser ignore the action of href is:

```
<a href="javascript:void(0)" onclick="alert('hi')" >
```

click me

```
</a>
```

To avoid refresh action  
if href is empty

**Example!**

# void Operator

- **void** Operator
  - A unary operator used to explicitly return undefined.
  - It can be used as shown

```
void expression;  
void(expression);
```

- Example:

```
var val= void "javascript";  
typeof val;    //undefined
```



# *Event Object*

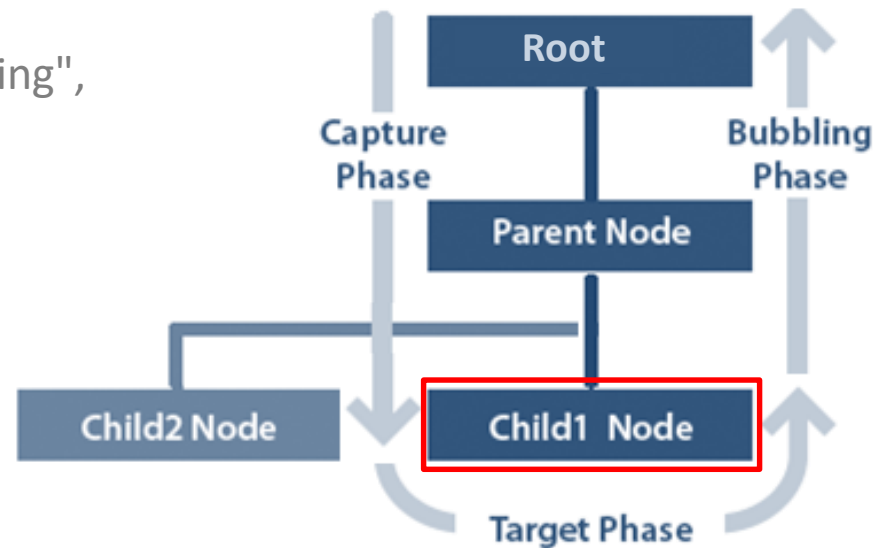
# Event Object

- The event object gives information about an event that has occurred.
- When an event occurs, an **event** object is initialized automatically and passed to the event handlers.
- We can create event object via its constructor  
`var evt= new Event()`
- The Event object represents the state of an event, such as the element in which the event occurred, the state of the keyboard keys, the location of the mouse, and the state of the mouse buttons.
- **Object Model reference:**  
`[window.]event`

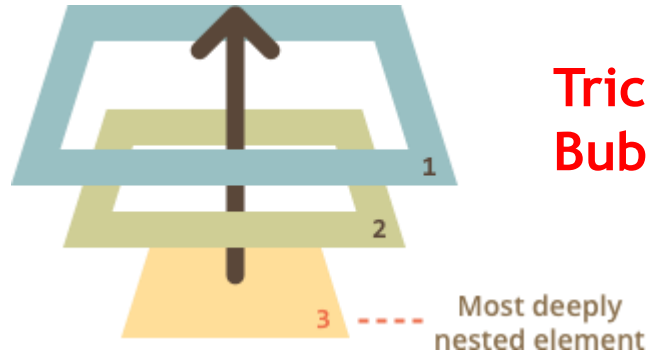


# Event Object

- Events always propagate from the root
- When an event occurs, it is dispatched to the target element first.
- 2 ways for objects to handle fired events
  - Event Capture (Phase1)
    - Capturing is also called "trickling",
    - Event goes down,
  - Event Bubbling (Phase3)
    - Event goes up



# Event Object



Trickle down,  
Bubble up

- If the event propagates up, then it will be dispatched to the ancestor elements of the target element in the DOM hierarchy.
- The propagation can be stopped with the **stopPropagation()** method and/or the **cancelBubble** property.

# Event Object Properties

Event Object Property	Description
srcElement	The element that fired the event
target	
currentTarget	identifies the current target for the event, as the event traverses the DOM
type	String representing the type of event.
clientX (layerX)	Mouse pointer X coordinate at the time of the event occurs <b>relative</b> to <b>upper-left</b> corner of the window.
clientY (layerY)	Mouse pointer Y coordinate at the time of the event occurs <b>relative</b> to <b>upper-left</b> corner of the window.
offsetX	Mouse pointer X coordinate <b>relative</b> to <b>element</b> that fired the event.
offsetY	Mouse pointer Y coordinate <b>relative</b> to <b>element</b> that fired the event.

# Event Object Properties

For alt,ctrl,shft keys

- keypress event don't fire when any of them is pressed
- Their properties is set to **true** only on **onkeydown** event

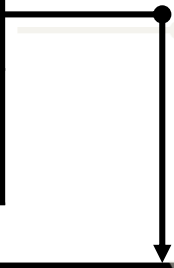
Event Object Property	Description
altKey	True if the alt key was also pressed
ctrlKey	True if the alt key was also pressed
shiftKey	True if the alt key was also pressed
button	Any mouse buttons that are pressed
keyCode( <b>deprecated</b> )	Returns UniCode value of key pressed use <b>code</b> property instead
which( <b>deprecated</b> )	
key	Represents the value of the key pressed (e.g. n)
code	Represents a physical key on the keyboard (e.g. keyN)

event.button value	Description
1	Left Mouse Button
<b>2</b>	Right Mouse Button
4	Middle Mouse Button

**Example!**

# Event Object Properties

Event Object Property	Description
eventPhase	Any mouse buttons that are pressed
cancelBubble ( <b>deprecated</b> )	Can cancel an event bubble



event.eventPhase value	Constant Property	Description
0	Event.NONE	No event is being processed at this time.
1	Event.CAPTURING_PHASE	The event is being propagated through the target's ancestor objects
2	<b>Event.AT_TARGET</b>	The event has arrived at target
3	Event.BUBBLING_PHASE	The event is propagating back up through the target's ancestors in reverse order

**Example!**

# Event Object Methods

Methods	Description
<b>event.stopPropagation()</b>	Disables the propagation of the current event in the DOM hierarchy. (IE8 = cancelBubble)
<b>event.stopImmediatePropagation()</b>	prevents other listeners are attached to the same element for the same event from being called, no remaining listeners will be called.
<b>event.preventDefault()</b>	To cancel the event if it is cancelable, meaning that any default action normally taken by the implementation as a result of the event will not occur. (IE8 = returnValue)
<b>event.composedPath()</b>	Returns the event's path

# Other Useful Methods

Methods	Description
<b>elem.addEventListener()</b>	Registers an event handler function for the specified event on the current object.
<b>elem.removeEventListener()</b>	method to remove an event listener that has been registered with the addEventListener method.
<b>elem.dispatchEvent()</b>	Initializes an event object created by the Event Constructor

# Synthetic Events

- To create custom event use Event constructor  
`var myEvent= new Event(p1,p2)`
  - p1: the name of the custom event type
  - p2: an object with the following Optional properties with `false` as default value
    - bubbles: indicating whether the event bubbles.
    - cancelable: indicating whether the event can be canceled.
    - ~~• composed: indicating whether the event will trigger listeners outside of a shadow root.~~
- To fire the event programmatically use `dispatchEvent()` on a specific element  
`elem.dispatchEvent(myEvent)`



# Synthetic Events

- To create custom event use CustomEvent constructor  
`var evt= new CustomEvent(p1,p2)`
  - p1: the name of the custom event type
  - p2: is object with details property to add more data to the event object
- To fire the event programmatically use dispatchEvent()  
on the element registering the event  
`elem.dispatchEvent(evt)`



# *Assignments*