Client-side Technologies

Eng. Niween Nasr El-Den
SD & Gaming CoE
iTi

Day 6

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
onmousedown	when pressing any of the mouse buttons.
onmousemove	when the user moves the mouse pointer within an element.
onmouseout	when moving the mouse pointer out of an element.
onmouseup	when the user releases any mouse button pressed
onmouseover	when the user moves the mouse pointer over an element.
onclick	when clicking the left mouse button on an element.
ondblclick	when Double-clicking the left mouse button on an element.
ondragstart	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, shft, ..etc.)
onkeyup	When User releases the pressed a key

Other Events

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

Other Events

Event handler	Description
onreset	When the user clicks the form reset button
onsubmit	When the user clicks the form 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
onstart	When a marquee element loop begins
onfinish	When a marquee object finishes looping
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 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>
```

Example!

Event Object

Event Object

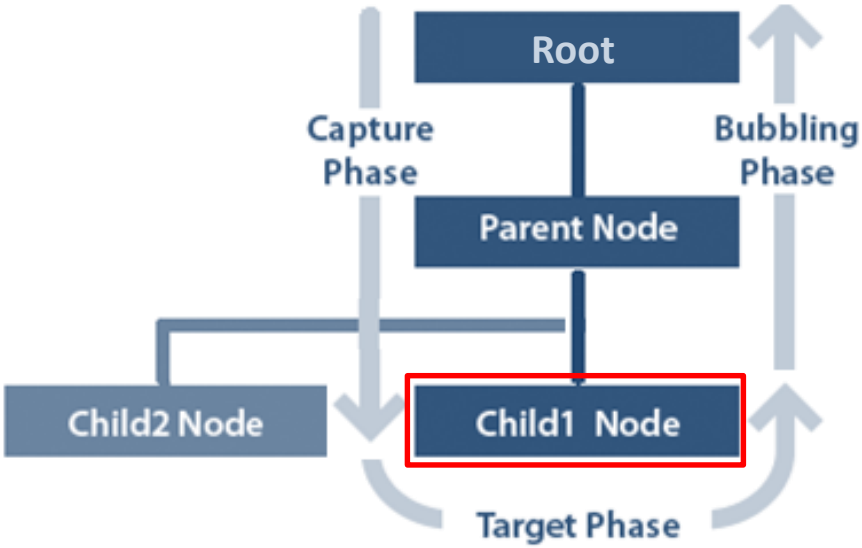
- The event object gives information about an event that has occurred.
- When an event occurs, an **event** object is initialized automatically and passed to the event handlers.
- We can create event object via its constructor

  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:**

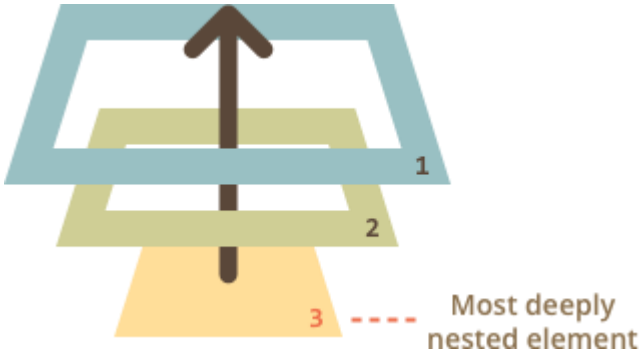
  [window.]event

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 detect events
  - **Event Capture (Phase1)**
    - **Event goes down**
  - **Event Bubbling (Phase2)**
    - **Event goes up**



Event Object



- 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 target	The element that fired the event
type	String representing the type of event.
returnValue	Determines whether the event is cancelled.
clientX (layerX)	Mouse pointer X coordinate at the time of the event occurs relative to upper-left corner of the window.
clientY (layerY)	Mouse pointer Y coordinate at the time of the event occurs relative to upper-left corner of the window.
offsetX	Mouse pointer X coordinate relative to element that fired the event.
offsetY	Mouse pointer Y coordinate relative to element that fired the event.

Event Object Properties

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
keyCode	Returns UniCode value of key pressed
which	
button	Any mouse buttons that are pressed
cancelBubble	Can cancel an event bubble

Example!

Event Object Properties

Event Object Property	Description
eventPhase	Any mouse buttons that are pressed
cancelBubble	Can cancel an event bubble

event.eventPhase value	Constant	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	Event.AT_TARGET	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!

Useful Methods for Event

Methods	Description
addEventListener()	Registers an event handler function for the specified event on the current object.
removeEventListener()	method to remove an event listener that has been registered with the addEventListener method.
dispatchEvent()	Initializes an event object created by the createEvent method or Event Constructor
event.stopPropagation()	Disables the propagation of the current event in the DOM hierarchy. (IE8 = cancelBubble)
event.preventDefault()	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)

Using Event Constructor

- 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
    - bubbles: indicating whether the event bubbles. The default is false
    - cancelable: indicating whether the event can be canceled. The default is false.

- To fire the event programmatically use dispatchEvent() on a specific element
  elem.dispatchEvent(myEvent)

Document Object Model

DOM

DOM

- 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.
- 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

DOM

- 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 Relationships

Scripting HTML

HTML DOM

- The HTML DOM is a standard for how to get, change, add, or delete HTML elements.
- It is a hierarchy of data types for HTML documents, links, forms, comments, and everything else that can be represented in HTML code.
- The general data type for objects in the DOM are *Nodes*. They have *attributes*, and some nodes can contain other nodes.
- There are several node types, which represent more specific data types for HTML elements. Node types are represented by numeric constants.

HTML DOM

- It allows code running in a browser to access and interact with every node in the document.
- Nodes can be created, moved and changed.
- Event listeners can be added to nodes and triggered on occurrence of a given event.

HTML DOM

- According to the DOM, everything in an HTML document is a node.
- The DOM says:
  - The entire document is a document node
  - Every HTML element is an element node
  - The text in the HTML elements are text nodes
  - Every HTML attribute is an attribute node
  - Comments are comment nodes
- JavaScript is powerful DOM Manipulation

Simple Example!

```
<html>
  <head>
    <title> Simple Example! </title>
  </head>
  <body>
    <h1>Heading</h1>
    <p id="my">A paragraph</p>
    <!--and that's all-->
  </body>
</html>
```

Root Node
➔ document.documentElement

Element Node

Text Node

document.body

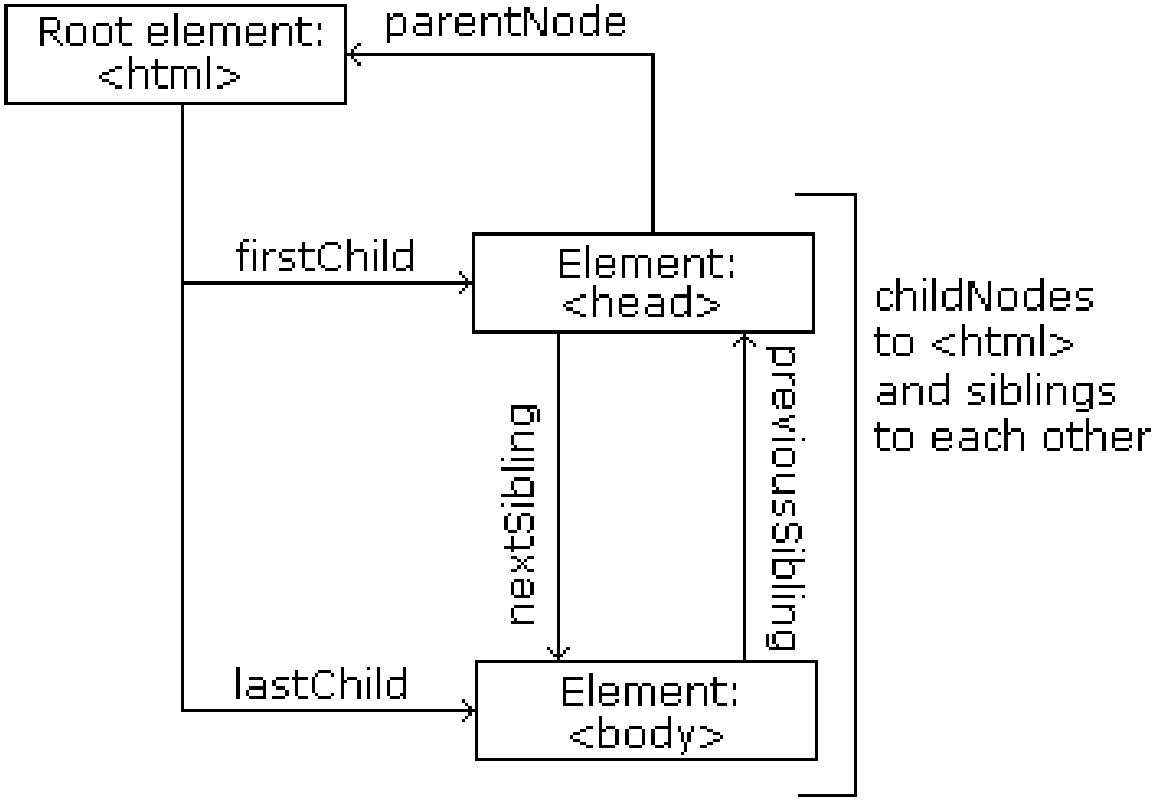
Document Node

Attribute Node

Comment Node

Node Tree

- The HTML DOM views HTML document as a node-tree.
- All the nodes in the tree have relationships to each other.
  - Parent
    - parentNode
  - Children
    - firstChild
    - lastChild
  - Sibling
    - nextSibling
    - previousSibling

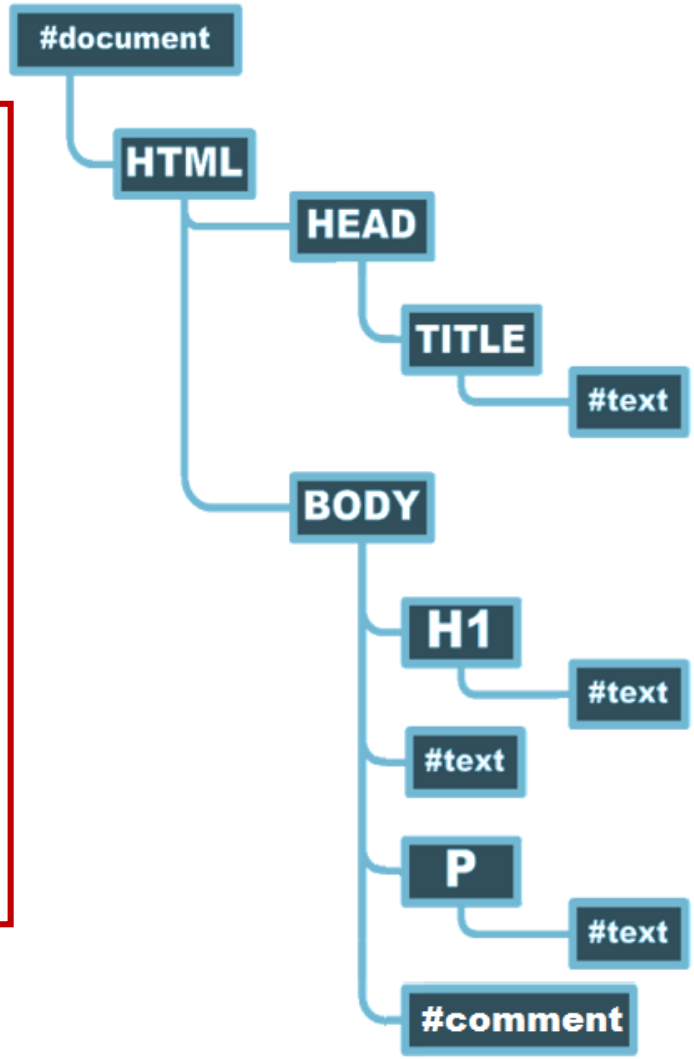


Nodes Relationships

- The terms **parent**, **child**, and **sibling** are used to describe the relationships.
  - Parent nodes have children.
  - Children on the same level are called siblings (brothers or sisters).
- Attribute nodes are not child nodes of the element they belong to, and have no parent or sibling nodes
- In a node tree, the top node is called the root
- Every node, except the root, has exactly one parent node
- A node can have any number of children
- A leaf is a node with no children
- Siblings are nodes with the same parent

Simple Example!

```
<html>
        <head>
            <title>Simple Example!</title>
        </head>
        <body>
            <h1>Greeting</h1>
             Welcome All
            <p>A paragraph</p>
            <!-- and that's all-->
        </body>
 </html>
```



Node Properties

- All nodes have three main properties

Property	Description
nodeName	Returns HTML Tag name in uppercase display
tagname	
nodeType	returns a numeric constant to determine node type. There are 12 node types.
nodeValue	returns null for all node types except for text and comment nodes.

Using nodeName
If node is text it returns #text
For comment it returns #comment
For document it returns #document

Value	Description
1	Element Node
2	Attribute Node
3	Text Node
8	Comment Node
9	Document Node

To get the Root Element:
document.documentElement.

Node Collections

- Node Collections have One Property
  - **length : gives the length of the Collection.**
    - **e.g. childNodes.length:** returns number of elements inside the collection
- We can check if there is child collection using
  - **hasChildNodes():** Tells if a node has any children
- We can check if there is attribute collection using
  - **hasAttributes():** Tells if a node has any attributes

Collection	Description	Accessing
childNodes	Collection of element's children	childNodes[ ] childNodes.item()
attributes	Returns collection of the attributes of an element	attributes[ ] attributes.item()

Dealing With Nodes

- Dealing with nodes fall into four main categories:
  - Accessing Node
  - Modifying Node’s content
  - Adding New Node
  - Remove Node from tree

Accessing DOM Nodes

- You can access a node in **5** main ways:
  - [window.]document.getElementById("id")
  - [window.]document.getElementsByName("name")
  - [window.]document.getElementsByTagName("tagname")
  - By navigating the node tree, using the node relationships
  - New HTML5 Selectors.

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

New HTML5 Selectors

- Selecting the first div met

```
var elements = document.querySelector("div");
```

- Selecting the first item with class SomeClass

```
var elements = document.querySelector(".SomeClass");
```

- Selecting the first item with id someID

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

- Selecting all the divs in the current container

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

Accessing DOM Nodes

Navigating the node tree, using the node relationships

firstChild	Move direct to first child
lastChild	Move direct to last child
parentNode	To access child's parent
nextSibling	Navigate down the tree one node step
previousSibling	Navigate up the tree one node step
Using children collection → childNodes[ ]	

Example!

Modifying Node's Content

- Changing the Text Node by using

innerHTML	Sets or returns the HTML contents (+text) of an element
textContent	Equivalent to innerText.
nodeValue → with text and comment nodes only	
setAttribute()	Modify/Adds a new attribute to an element
just using attributes as object properties	

- Modifying Styles
  - Node.style

Example!

Creating & Adding Nodes

Method	Description
createElement()	To create new tag element
createTextNode()	To create new text element
createAttribute()	To creates an attribute element
createComment()	To creates an comment element

Creating & Adding Nodes

Method	Description
cloneNode(true|false)	Creating new node a copy of existing node. It takes a Boolean value true: Deep copy with all its children or false: Shallow copy only the node
b.appendChild(a)	To add new created node “a” to DOM Tree at the end of the selected element “b”.
insertBefore(a,b)	Similar to appendChild() with extra parameter, specifying before which element to insert the new node. a: the node to be inserted b: where a should be inserted before **document.body.insertBefore(a,b)**

Example!

Removing DOM Nodes

Method	Description
removeChild()	To remove node from DOM tree
parent.replaceChild(n,o)	To remove node from DOM tree and put another one in its place n: new child o: old child
removeAttribute()	Removes a specified attribute from an element

- A quick way to wipe out all the content of a subtree is to set the innerHTML to a blank string. This will remove all of the children of <body>

```
document.body.innerHTML="";
```

Example!

Summary

- Access nodes:
  - Using parent/child relationship properties parentNode, childNodes, firstChild, lastChild, nextSibling, previousSibling
  - Using getElementsById(), getElementsByTagName(), getElementsByName()
- Modify nodes:
  - Using innerHTML or innerText/textContent
  - Using nodeValue or setAttribute() or just using attributes as object properties
- Remove nodes with
  - removeChild() or replaceChild()
- And add new ones with
  - appendChild(), cloneNode(), insertBefore()

Dynamic HTML

the art of making dynamic and interactive web pages.

DHTML

- DHTML has no official definition or specification.
- DHTML stands for Dynamic HTML.
- DHTML is NOT a scripting language.
- DHTML is not w3c (i.e. not a standard).
- DHTML is a browser feature-that gives you the ability to make dynamic Web pages.
- "*Dynamic*" is defined as the ability of the browser to alter a web page's look and style *after* the document has been loaded.
- DHTML is very important in web development

DHTML

- DHTML uses a combination of:
  1. Scripting language
  2. DOM
  3. CSS

  to create HTML that can change even after a page has been loaded into a browser.

- DHTML is supported by 4.x generation browsers.

Assignment