Client-side Technologies

Eng. Niveen Nasr El-Den
iTi

Day 8

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

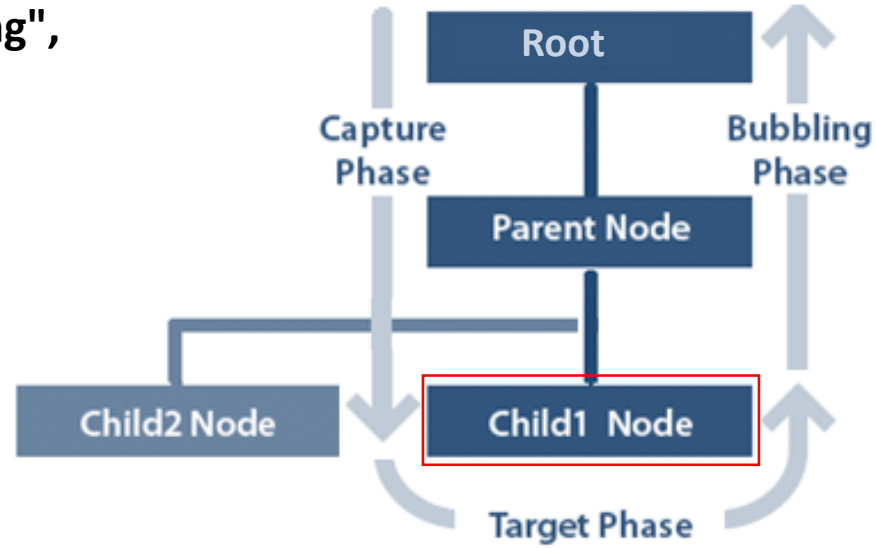
  var evt= new Event()

- 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:**

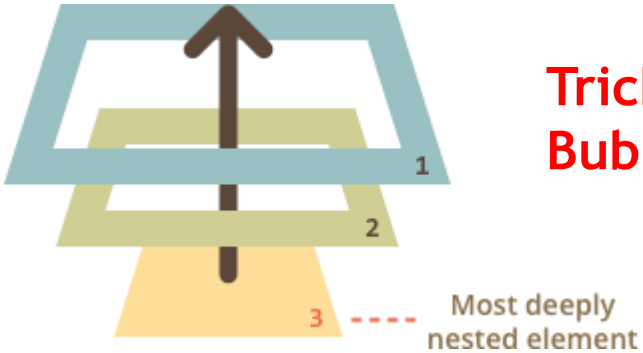
  [window.]event

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 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
button	Any mouse buttons that are pressed
keyCode	Returns UniCode value of key pressed
which	
code	Represents a physical key on the keyboard
cancelBubble	Can cancel an event bubble

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

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 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	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!

Event Object Methods

Methods	Description
event.stopPropagation()	**Disables the propagation of the current event in the DOM hierarchy.** **(IE8 = cancelBubble)**
event.stopImmediatePropagation()	prevents other listeners are attached to the same element for the same event from being called, no remaining listeners will be called.
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)**
event.composedPath()	**Returns the event's path**

Other Useful Methods

Methods	Description
elem.addEventListener()	Registers an event handler function for the specified event on the current object.
elem.removeEventListener()	method to remove an event listener that has been registered with the addEventListener method.
elem.dispatchEvent()	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)
```

Document Object Model

DOM

DOM

- DOM Stands for Document Object Model.
- W3C standard.

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

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!**

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

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.

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.
- Node types are represented by numeric constants.
- Event listeners can be added to nodes and triggered on occurrence of a given event.

DOM
is an API that represents and interacts with any
HTML or **XML** document.

The DOM
is a document model loaded in
the browser and
representing the document
as a node tree,
where each node represents
part of the document

The DOM
is an application programming interface “API“

a set of functions or methods used to access some functionality

The DOM
Defines the logical structure of document and the way a document is accessed and manipulated

DOM
connects web pages to scripts or programming lan guages by representing the structure of a document.

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

An **element**
is a specific type of **node**, one
that can be directly specified in
the **HTML** with
an **HTML tag**

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

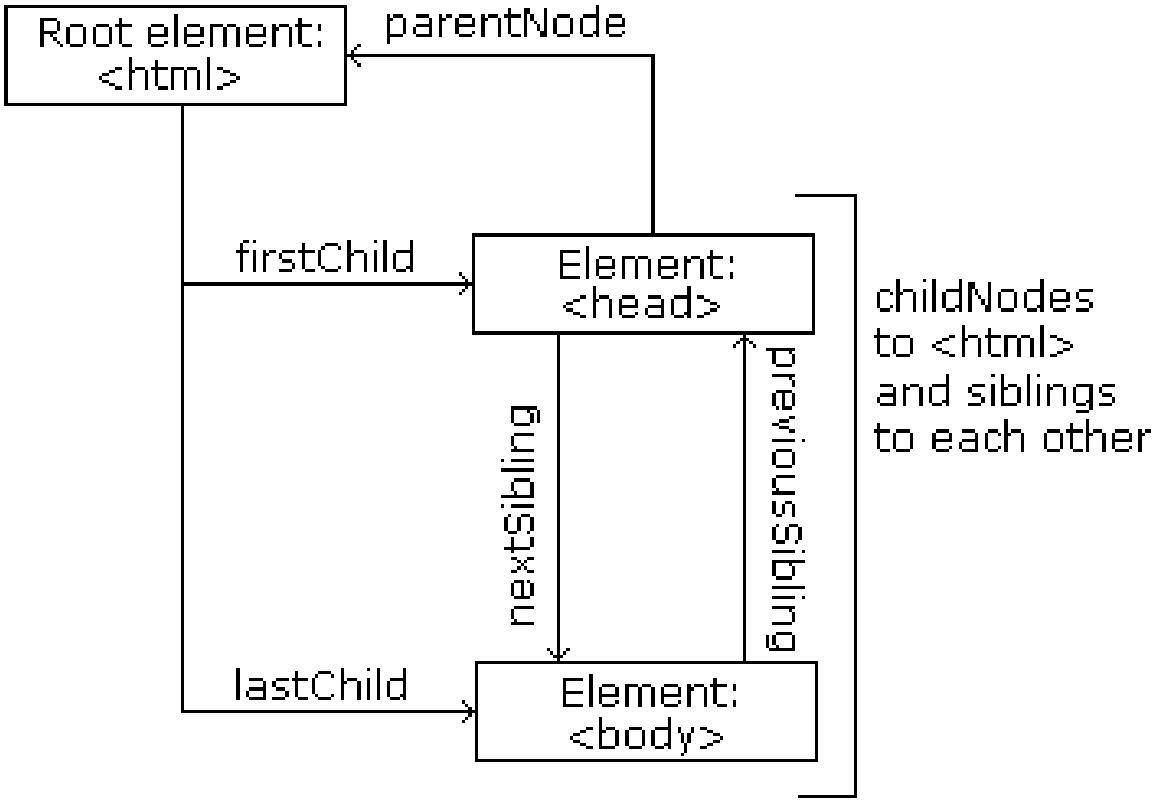
Attribute Node

Document 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

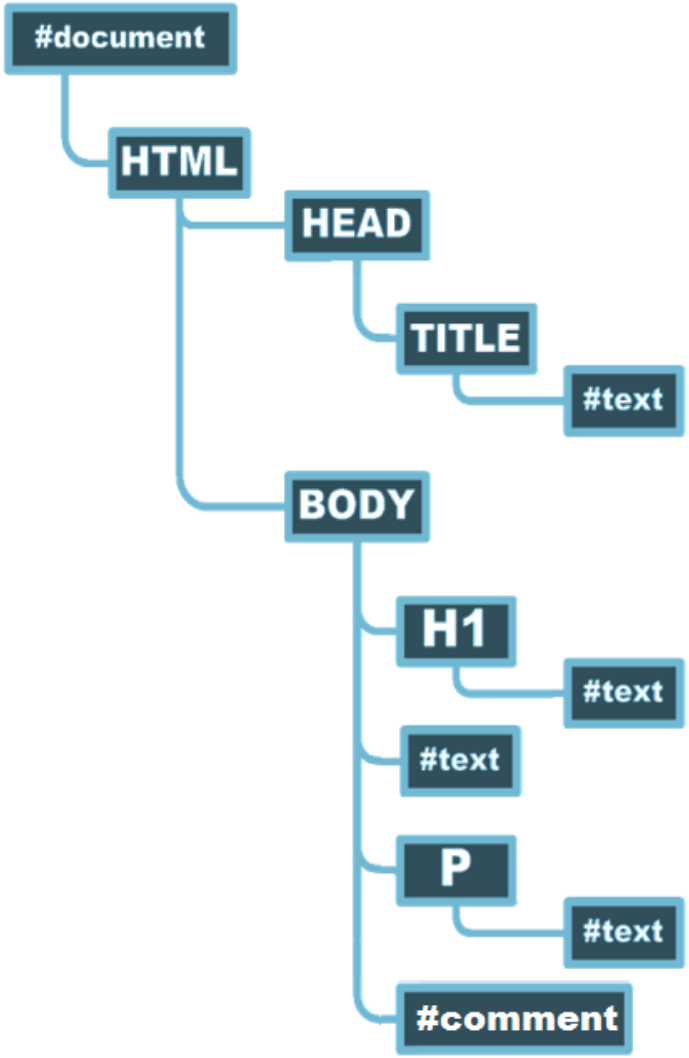


Nodes Relationships

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

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

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

Accessing DOM Nodes

Navigating the node tree, using the node relationships

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

Example!

Accessing DOM Elements

Navigating the elements nodes, using the relationships

firstElementChild	Move direct to first Element child
lastElementChild	Move direct to last Element child
parentElement	To access child's Element parent
nextElementSibling	Navigate down the tree to next Element
previousElementSibling	Navigate up the tree to previous Element

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	

Example!

Node's Class Attribute

- The global class attribute is get and set via className property

- The classList property returns a collection of the class attributes of the caller element, it has the following methods
  - add("classNm")
  - remove("classNm")
  - toggle("classNm")
  - replace("oldClassNm","newClassNm")

Manipulating Styles

- Modifying style properties of any HTML element is accessed using the style object
- For inline style
  - Node.style[.prop_name]
  - Node.style.cssText
- To read internal or external styling in general
  - document.styleSheets
  - document.styleSheets[i].cssRules
  - document.styleSheets[i].cssRules[idx].selectorText
  - document.styleSheets[i].cssRules[idx].cssText
- To read none inline styling applied for specific element
  - getComputedStyle(elem).prop_nm
  - getComputedStyle(elem). getPropertyValue(prop_nm)

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”.
b.append(a)	Experimental function to o add new created node “a” to DOM Tree at the end of the selected element “b”.
b.prepend(a)	Experimental function to o add new created node “a” to DOM Tree at the top of the selected element “b”.

Example!

Creating & Adding Nodes

https://developer.mozilla.org/en-US/docs/Web/API/Element

Method	Description
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)**
e.insertAdjacentElement(pos,elem)	○ e: represents the target element ○ elem: represents the element to be added ○ pos: represents the position relative to the targetElem • 'beforebegin': Before the targetElement itself. • 'afterbegin': Just inside the targetElement, before its first child. • 'beforeend': Just inside the targetElement, after its last child. • 'afterend': After the targetElement itself.

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

- To quick replace a node set its outerHTML property

```
elem.outerHTML="<div>something</div";
```

- 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()

Modeling HTML or XML documents as objects are not part of the core JavaScript language.

The DOM
Defines the logical structure of document and the way a document is accessed and manipulated

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