SNS COLLEGE OF ENGINEERING

An Autonomous Institution

Coimbatore-107

19TS601-FULL STACK DEVELOPMENT

UNIT-1

JAVASCRIPT AND BASICS OF MERN STACK

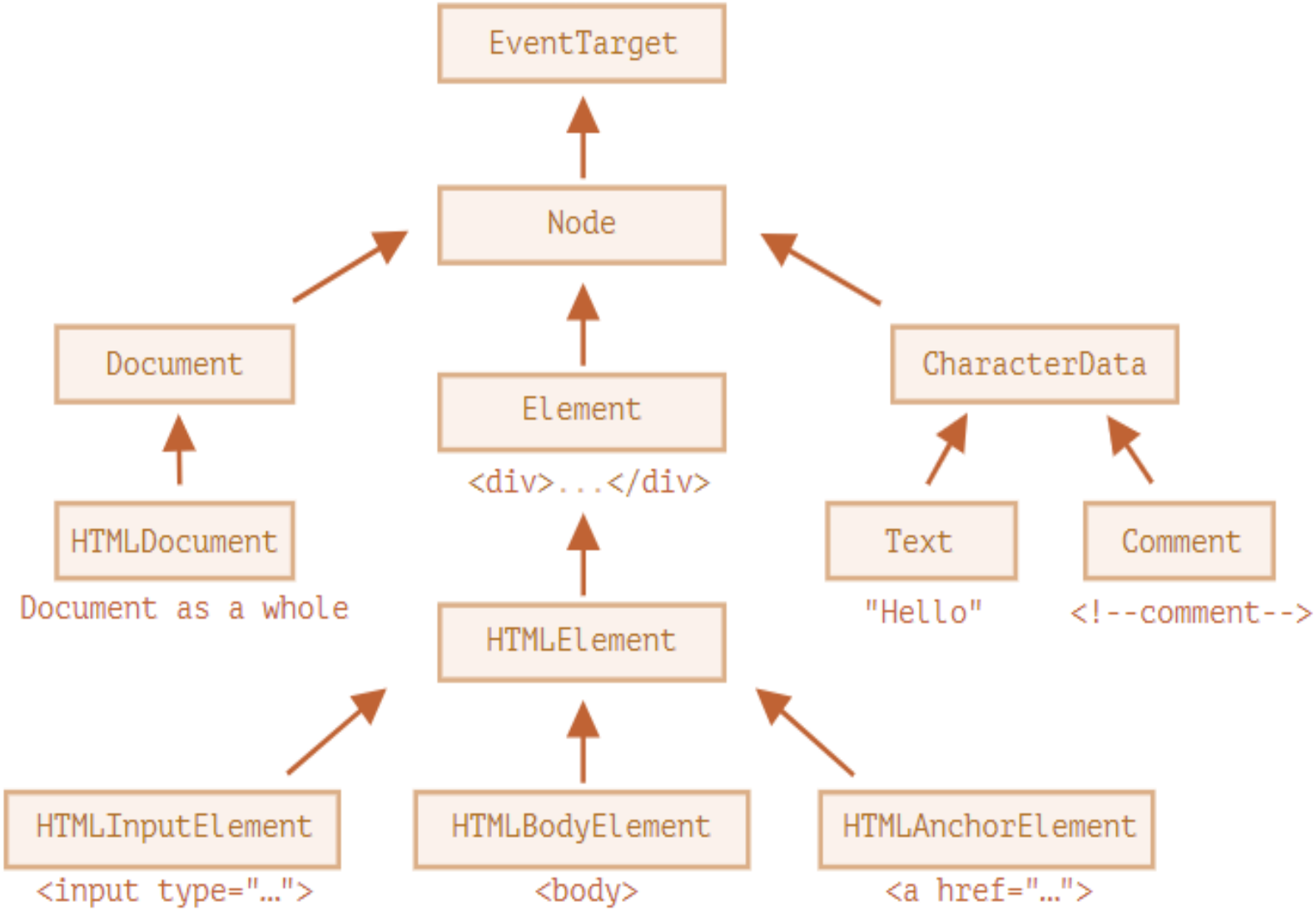
Node properties - browser events - Event delegation

Node properties: type, tag and contents

DOM node classes

- Different DOM nodes may have different properties.
- For instance, an element node corresponding to tag <a> has link-related properties, and the one corresponding to <input> has input-related properties and so on.
- Text nodes are not the same as element nodes.
- But there are also common properties and methods between all of them, because all classes of DOM nodes form a single hierarchy.

- Each DOM node belongs to the corresponding built-in class.
- The root of the hierarchy is EventTarget, that is inherited by Node, and other DOM nodes inherit from it.
- Here's the picture, explanations to follow:


EventTarget
Node
Document
Element
CharacterData
...
HTMLDocument
Text
Comment
Document as a whole
"Hello"
<!--comment-->
HTMLElement
HTMLInputElement
HTMLBodyElement
HTMLAnchorElement
<input type="...">
<body>
<a href="...">
CS8651-Internet Programming
4

- The classes are:
- EventTarget – is the root “abstract” class for everything. Objects of that class are never created. It serves as a base, so that all DOM nodes support so-called “events”.
- Node – is also an “abstract” class, serving as a base for DOM nodes.It provides the core tree functionality: parentNode, nextSibling, childNodes and so on (they are getters). Objects of Node class are never created. But there are other classes that inherit from it (and so inherit the Node functionality).

- Document, for historical reasons often inherited by HTMLDocument (though the latest spec doesn't dictate it) – is a document as a whole.
- The document global object belongs exactly to this class. It serves as an entry point to the DOM.

- CharacterData – an “abstract” class, inherited by:
- Text – the class corresponding to a text inside elements, e.g. Hello in Hello .
- Comment – the class for comments. They are not shown, but each comment becomes a member of DOM.

- Element – is the base class for DOM elements.
- It provides element-level navigation like nextElementSibling, children and searching methods like getElementsByTagName, querySelector.
- A browser supports not only HTML, but also XML and SVG. So the Element class serves as a base for more specific classes: SVGElement, XMLElement (we don't need them here) and HTMLElement.

- Finally, HTMLElement is the basic class for all HTML elements.
- It is inherited by concrete HTML elements:
- HTMLInputElement – the class for <input> elements,
- HTMLBodyElement – the class for <body> elements,
- HTMLAnchorElement – the class for <a> elements,…and so on.

Browser Events

- Events are considered occurrences or actions, happening in the system you are developing that the latter informs you about so that you can respond to them. In brief, it is a signal that something has happened in your system.
- DOM Events
- Mouse Events:
  - click– when a click is made on an element by the mouse (touchscreen devices are capable of generating it on tap).
  - contextmenu – the right-click of the mouse on an element.
  - mouseover/mouseout –when the cursor of the mouse moves onto or leaves an element.
  - mousedown/mouseup - while the mouse button is pressed or is over an element.
  - mousemove- the mouse is moved.

Browser Events

- Form Element Events:
  - submit - the event of submitting a <form> by the visitor.
  - focus - focusing on an element by the visitor.
- Keyboard Events:
  - keydown - when the visitor presses the button.
  - keyup - when any key is released by the visitor.
- Document Events:
  - DOMContentLoaded - the state of loading and proceeding of HTML, the DOM is completely built.

Browser Events

- CSS Events:
  - transitionend - completion of a CSS-animation.
- There are more events, but the ones above are the most essential.

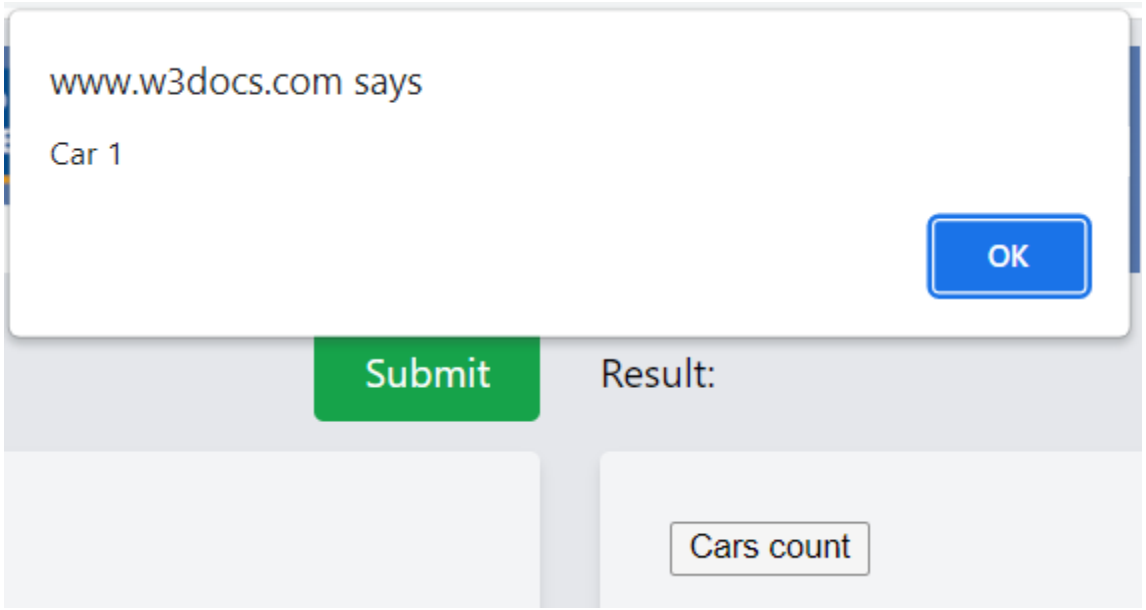
Event Handlers

- A handler is assigned for reacting to events: it is a function, running in case of an event.
- Handlers are a means of running JavaScript code in case of user actions.

Browser Events

```
<html>
 <head>
  <title>Title of the Document</title>
 </head>
 <body>
  <script>
   function carsCount() {
    for(let i = 1; i <= 5; i++) {
     alert("Car " + i);
    }
   }
  </script>
  <input type="button" onclick="carsCount()" value="Cars count">
 </body>
</html>
```

Browser Events



Event Handlers via DOM Property

- A handler can also be assigned with a DOM property on<event>.
- Let's see elem.onclick in the example below:

```
<html>
 <head>
  <title>Title of the Document</title>
 </head>
 <body>
  <input id="inputId" type="button" value="Click">
  <script>
   inputId.onclick = function() {
    alert('Welcome to W3Docs');
   };
  </script>
 </body>
</html>
```

Browser Events-Event Listener (addEventListener)

- Event listener allow to add multiple handlers to a single event whereas Event handler doesn't allow multiple handler for single event.

Event Delegations

- Event Delegation is basically a pattern to handle events efficiently.
- Instead of adding an event listener to each and every similar element, we can add an event listener to a parent element and call an event on a particular target using the .target property of the event object.
- Allows you to avoid adding event listeners to specific nodes; the event listener is added to one parent.

```
<html>
  <head>
    <title>Title of the Document</title>
  </head>
  <body>
    <ul id="parent-list">
      <li id="post_1">Post 1</li>
      <li id="post_2">Post 2</li>
      <li id="post_3">Post 3</li>
      <li id="post_4">Post 4</li>
      <li id="post_5">Post 5</li>
      <li id="post_6">Post 6</li>
    </ul>
  </body>
</html>
```

- When each of the child elements is clicked, something needs to happen.
- It is possible to add a separate event listener to every LI element.
- But if the elements are frequently added or removed, adding or removing event listeners would become a nightmare.
- The most elegant solution is adding an event listener to the parent UL element.

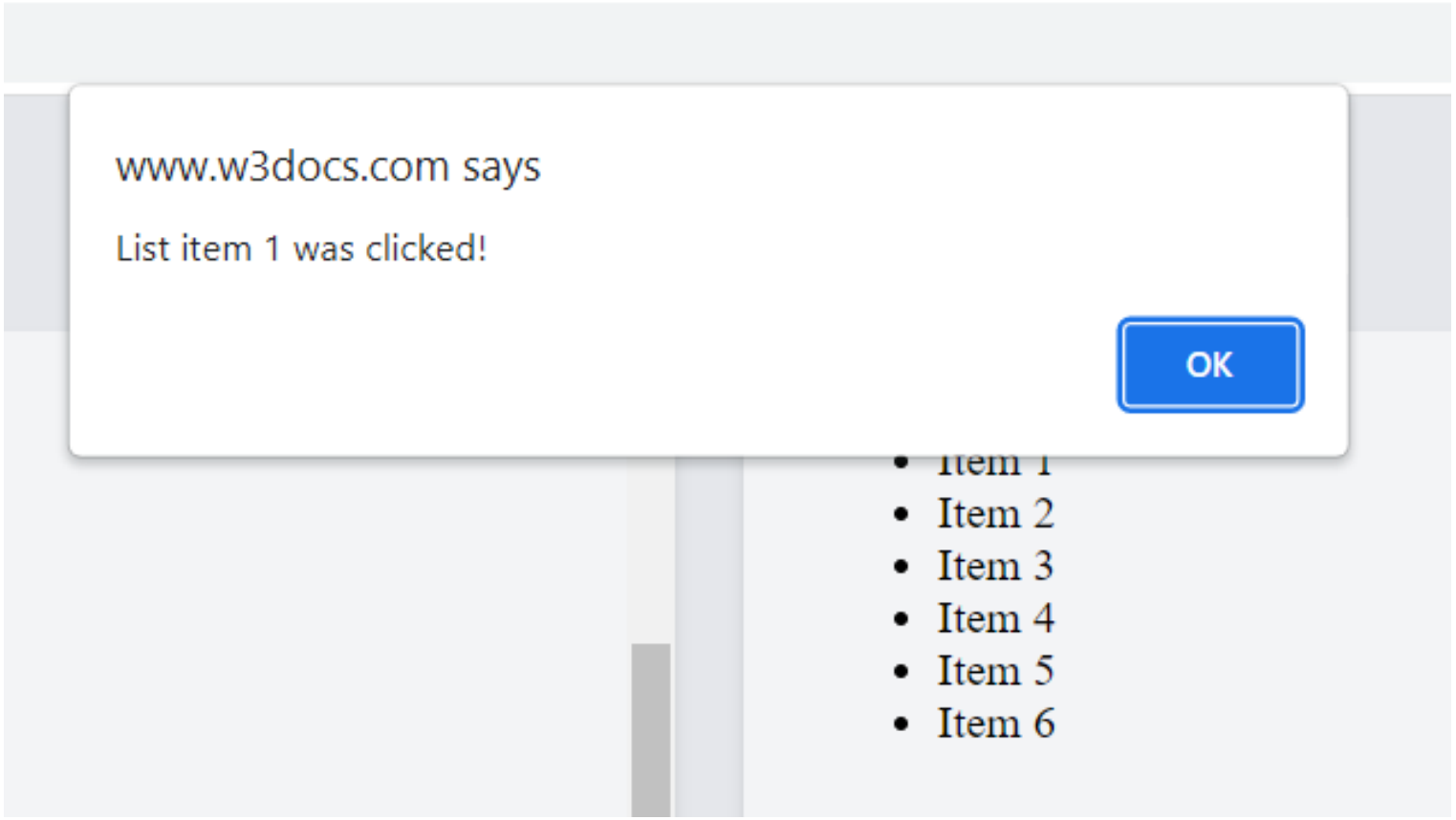
Illustration of the basic delegation

```
<html>
  <head>
    <title>Title of the Document</title>
  </head>
  <body>
    <ul id="parent-list">
      <li id="item_1">Item 1</li>
      <li id="item_2">Item 2</li>
      <li id="item_3">Item 3</li>
      <li id="item_4">Item 4</li>
      <li id="item_5">Item 5</li>
      <li id="item_6">Item 6</li>
    </ul>
```

Illustration of the basic delegation

```
<script>
 // Get the element, add a click listener...
 document.getElementById("parent-list").addEventListener("click", function(e) {
    // e.target is the clicked element!
    // If it was a list item
    if(e.target && e.target.nodeName == "LI") {
      // List item found!  Output the ID!
      alert("List item " + e.target.id.replace("item_", "") + " was clicked!");
    }
  });
</script>
</body>
</html>
```

Output


www.w3docs.com says
List item 1 was clicked!
OK
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6

Thank You