COMP 307

Principles of Web Development

Lecture 6

Unit 2 – Frontend Internet Languages

JavaScript 2

Contents

Class Outline

- Document Object Model
- DOM manipulation

Readings

- Internet and World Wide Web textbook
  - Chapters: 11 to 13
- Internet Resources
  - https://www.w3schools.com/js/
  - https://www.tutorialspoint.com/javascript/index.htm
  - 10 JS Projects in 10 Hours https://www.youtube.com/watch?v=dtKciwk_si4

Contents

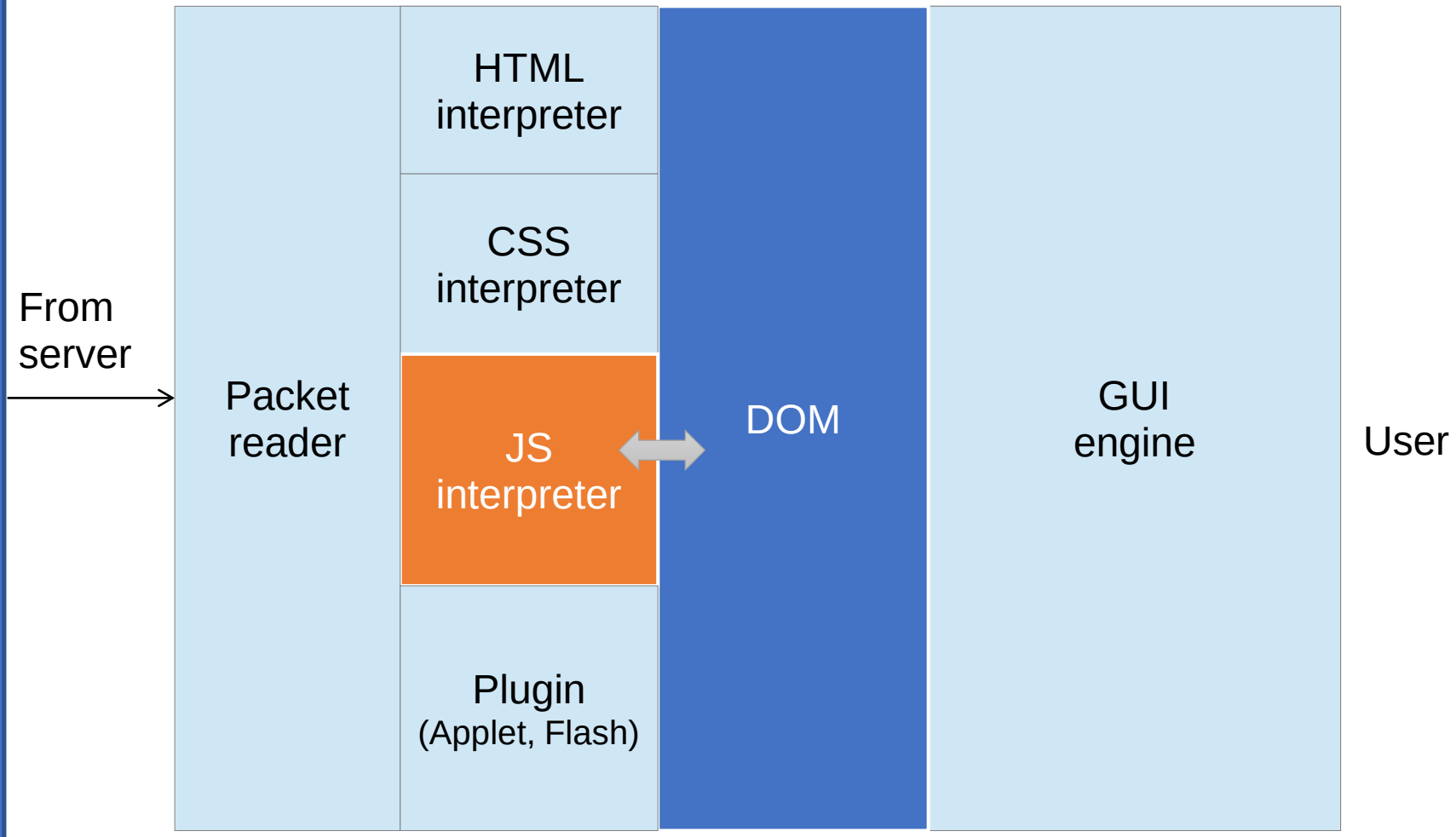
JD & DOM

Document Object Model

Java Script 2

Contents

The Browser System



What is DOM

- Document Object Model
- A data structure that defines what should be displayed on the computer screen
- Since many languages interact with the GUI a common internal language is needed to make things easier for the GUI algorithms.
- It is a Tree
  - Each node is an element
  - Pointers point in the direction the elements need to be rendered on the screen

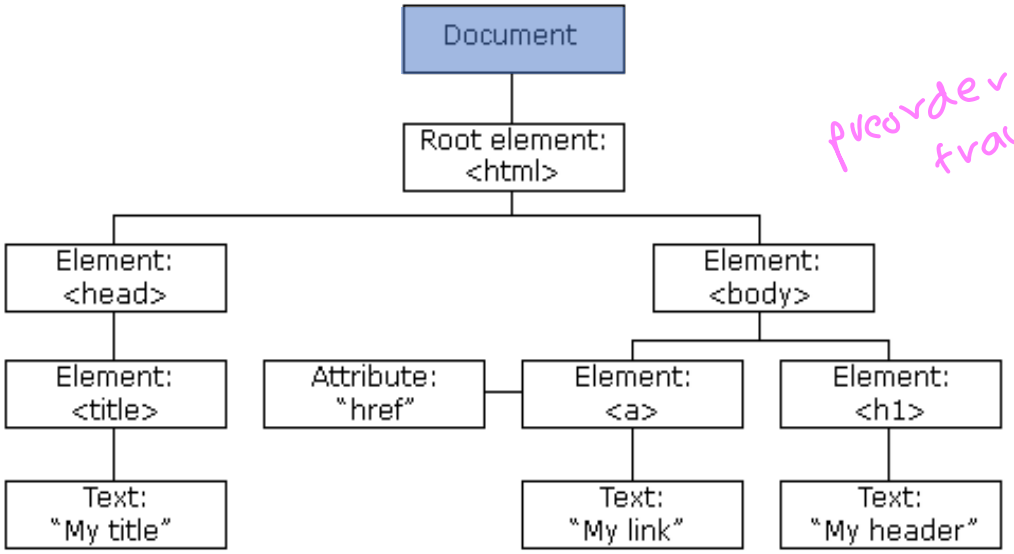
Contents

JD & DOM

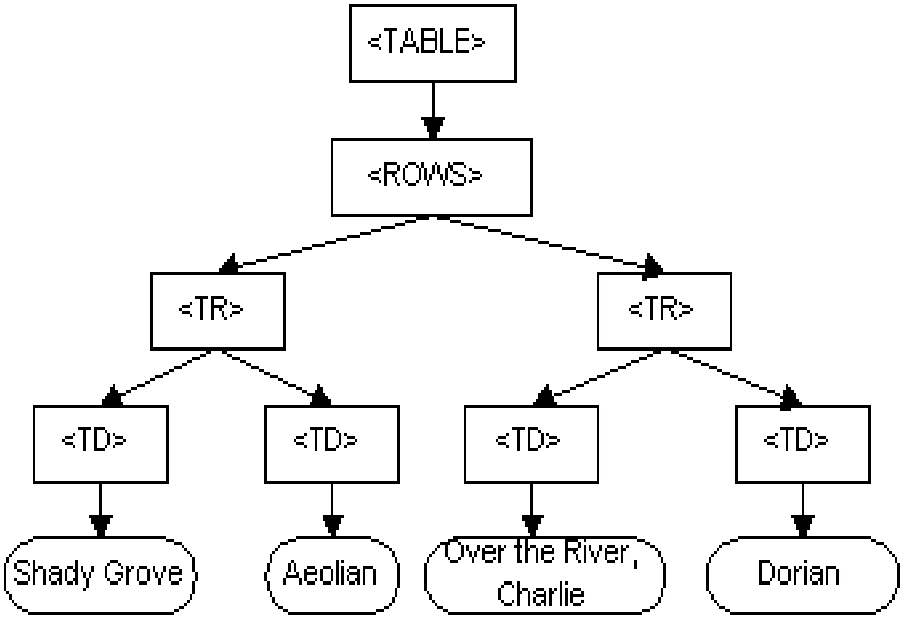
COMP 307
Principles of Web Development

Contents

What is DOM



provide traversal



inconsistency between browsers can be due to differences or renderers

Some browsers have implemented extra tags

HTML + CSS + JS + DOM

Ptr to DOM

replace

search

Auto loaded

p refer to p

p is an object

JS

CSS

```
<html> <head>
        <script>
            function myFunction() {
                document.getElementById("demo").innerHTML =
                "Cookies associated with this document: " +
                            document.cookie;
            }
        </script>

        <style>
                body {background-color: lightblue;}
                h1 {color: white;text-align: center;}
                p {font-family: verdana;font-size: 20px;}
        </style>

        </head>
        <body>
                <p id="demo">Click the button to display the cookies
                associated with this document.</p>

                <button onclick="myFunction()">Try it</button>
        </body>
</html>
```

Dynamic Programming

- Since DOM is a Tree
  - Nodes can be deleted or changed
  - Pointers can be moved
- HTML **creates** DOM nodes
- CSS **modifies** DOM node properties
- JS can **interact** with the user to create/delete nodes and modify node properties

can do all of the above

- HTML **Call Backs**
  - HTML created DOM nodes can reference back to JS code/functions

Callback

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to display the date.</p>

<button onclick="displayDate()">The time is?</button>

<script>
   function displayDate() {
      document.getElementById("demo").innerHTML = Date();
   }
</script>

<p id="demo"></p>

</body>
</html>
```

This is a callback

Search by ID

Reading & Writing DOM

```
<!DOCTYPE html>
<html><head>
<script>
   function getOption() {
      var obj = document.getElementById("mySelect");
      document.getElementById("demo").innerHTML = obj.options[obj.selectedIndex].text;
   }
</script></head>
<body>

<form>
   Select your favorite fruit:
   <select id="mySelect">
      <option>Apple</option>
      <option>Orange</option>
      <option>Pineapple</option>
      <option>Banana</option>
   </select>
   <br><br>
   <input type="button" onclick="getOption()" value="Click Me!">
</form>

<p id="demo"></p>

</body>
</html>
```

We will see more about this next lecture.

Example 2

```
<!DOCTYPE html>
<html>
 <body>

 <form id="frm1">
  First name: <input type="text" name="fname" value="Donald"><br>
  Last name: <input type="text" name="lname" value="Duck"><br><br>
 </form>

 <p>Click "Try it" to display the value of each element in the form.</p>

 <button onclick="myFunction()">Try it</button>

 <p id="demo"></p>

 <script>
   function myFunction() {
     var x = document.getElementById("frm1");
     var text = "";
     var i;
     for (i = 0; i < x.length ;i++) {
       text += x.elements[i].value + "<br>";
     }
     document.getElementById("demo").innerHTML = text;
   }
 </script>

 </body>
</html>
```

Contents

JD & DOM

DOM Manipulation

- getElementByID
  - `x = document.getElementByID("name").value;`
  - `ref = document.getElementByID("name");`
- setAttribute & getAttribute
  - `ref.setAttribute("attribute","value");`
- insertBefore & appendChild
  - insert before an element or add and element

```
var newNode = createNewNode(
   document.getElementById( "ins" ).value );
currentNode.parentNode.insertBefore( newNode, currentNode );
```

.insertBefore and appendChild

```
<!DOCTYPE html>
<html>
  <body>

  <ul id="myList">
    <li>Coffee</li>
    <li>Tea</li>
  </ul>

  <p>Click the button to insert an item to the list.</p>

  <button onclick="myFunction()">Try it</button>

  <script>
    function myFunction() {
      var newItem = document.createElement("LI");              // create an <li> tag
      var textnode = document.createTextNode("Water");         // create a string
      newItem.appendChild(textnode);                           // <li>Water</li>

      var list = document.getElementById("myList");
      list.insertBefore(newItem, list.childNodes[0]);          // insert newItem before node 0
    }
  </script>

 </body>
</html>
```

creates the tag

Moving nodes

```
<!DOCTYPE html>
<html>
<body>

<ul id="myList1"><li>Coffee</li><li>Tea</li></ul>

<ul id="myList2"><li>Water</li><li>Milk</li></ul>

<p>Click the button to move an item from one list to another</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  var node = document.getElementById("myList2").lastChild;
  var list = document.getElementById("myList1");
  list.insertBefore(node, list.childNodes[0]);
}
</script>

</body>
</html>
```

move last element of list 2 to top of list 1

Contents

JD & DOM

Replacing nodes

```
<!DOCTYPE html>
<html>
<body>

<ul id="myList"><li>Coffee</li><li>Tea</li><li>Milk</li></ul>

<p>Click the button to replace the first item in the the list.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  var textnode = document.createTextNode("Water");
  var item = document.getElementById("myList").childNodes[0];
  item.replaceChild(textnode, item.childNodes[0]);
}
</script>

</body>
</html>
```

Contents

JD & DOM

Remove nodes

```
<!DOCTYPE html>
<html>
<body>

<ul id="myList"><li>Coffee</li><li>Tea</li><li>Milk</li></ul>

<p>Click the button to remove the first item from the list.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  var list = document.getElementById("myList");
  list.removeChild(list.childNodes[0]);
}
</script>

</body>
</html>
```

DOM Collections

- The Document Object Model contains several collections, which are groups of related objects on a page. DOM collections are accessed as properties of DOM objects such as the document object or a DOM node.
  - Images collection
  - Links collection
  - Forms collection
  - Anchors collection

groups of related objects

Contents

JD & DOM

Collection Example

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript HTML DOM</h2>

<p>Hello World!</p>

<p>Hello Norway!</p>

<p>Click the button to change the color of all p elements.</p>

Hello

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  const myCollection = document.getElementsByTagName("p");          // gather the collection by tag name
  for (let i = 0; i < myCollection.length; i++) {                    // iterate through the collection
    myCollection[i].style.color = "red";
  }
}
</script>

</body>
</html>
```

for all <p>'s

Header traversal of tree

Document Links example (1)

```
1  <!DOCTYPE html>
2
3  <!-- Fig. 12.13: collections.html -->
4  <!-- Using the links collection. -->
5  <html>
6     <head>
7        <meta charset="utf-8">
8        <title>Using Links Collection</title>
9        <link rel = "stylesheet" type = "text/css" href = "style.css">
10       <script src = "collections.js"></script>
11    </head>
12    <body>
13       <h1>Deitel Resource Centers</h1>
14       <p><a href = "http://www.deitel.com/">Deitel's website</a>
15          contains a growing
16          <a href = "http://www.deitel.com/ResourceCenters.html">list
17          of Resource Centers</a> on a wide range of topics. Many
18          Resource centers related to topics covered in this book,
19          <a href = "http://www.deitel.com/books/iw3htp5">Internet &
20          World Wide Web How to Program, 5th Edition</a>. We have
21          Resource Centers on
22          <a href = "http://www.deitel.com/Web2.0">Web 2.0</a>,
23          <a href = "http://www.deitel.com/Firefox">Firefox</a> and
24          <a href = "http://www.deitel.com/IE9">Internet Explorer 9</a>,
25          <a href = "http://www.deitel.com/HTML5">HTML5</a>, and
26          <a href = "http://www.deitel.com/JavaScript">JavaScript</a>.
27          Watch for related new Resource Centers.</p>
28       <p>Links in this page:</p>
29       <div id = "links"></div>
30    </body>
31 </html>
```

Document Links example (2)

```
1  // Fig. 12.14: collections.js
2  // Script to demonstrate using the links collection.
3  function processLinks()
4  {
5     var linksList = document.links; // get the document's links
6     var contents = "<ul>";
7
8     // concatenate each link to contents
9     for ( var i = 0; i < linksList.length; ++i )
10    {
11       var currentLink = linksList[ i ];
12       contents += "<li><a href='" + currentLink.href + "'>" +
13          currentLink.innerHTML + "</li>";
14    } // end for
15
16    contents += "</ul>";
17    document.getElementById( "links" ).innerHTML = contents;
18 } // end function processLinks
19
20 window.addEventListener( "load", processLinks );
```

built-in pointer to all anchors

append li

put links into ul

https://www.w3schools.com/jsref/coll_doc_links.asp

good for reformatting for if browsers

Animating Styles

```
7   // called repeatedly to animate the book cover
8   function run()
9   {
10     count += speed;
11
12     // stop the animation when the image is large enough
13     if ( count >= 375 )
14     {
15        window.clearInterval( interval );
16        interval = null;
17     } // end if
18
19     var bigImage = document.getElementById( "imgCover" );
20     bigImage.setAttribute( "style", "width: " + (0.7656 * count + "px;") +
21        "height: " + (count + "px;") );
22  } // end function run
23
24  // inserts the proper image into the main image area and
25  // begins the animation
26  function display( imgfile )
27  {
28     if ( interval )
29        return;
30
31     var bigImage = document.getElementById( "imgCover" );
32     bigImage.setAttribute( "style", "width: 0px; height: 0px;" );
33     bigImage.setAttribute( "src", "fullsize/" + imgfile );
34     bigImage.setAttribute( "alt", "Large version of " + imgfile );
35     count = 0; // start the image at size 0
36     interval = window.setInterval( "run()", 10 ); // animate
37  } // end function display
38
39  // register event handlers
40  function start()
41  {
42     document.getElementById( "jhtp" ).addEventListener(
43        "click", function() { display( "jhtp.jpg" ); }, false );
44     document.getElementById( "iw3htp" ).addEventListener(
45        "click", function() { display( "iw3htp.jpg" ); }, false );
46     document.getElementById( "cpphtp" ).addEventListener(
47        "click", function() { display( "cpphtp.jpg" ); }, false );
48     document.getElementById( "jhtplov" ).addEventListener(
49        "click", function() { display( "jhtplov.jpg" ); }, false );
50     document.getElementById( "cpphtplov" ).addEventListener(
51        "click", function() { display( "cpphtplov.jpg" ); }, false );
52     document.getElementById( "vcsharphtp" ).addEventListener(
53        "click", function() { display( "vcsharphtp.jpg" ); }, false );
54  } // end function start
55
56  window.addEventListener( "load", start );
```

inline function

optional field

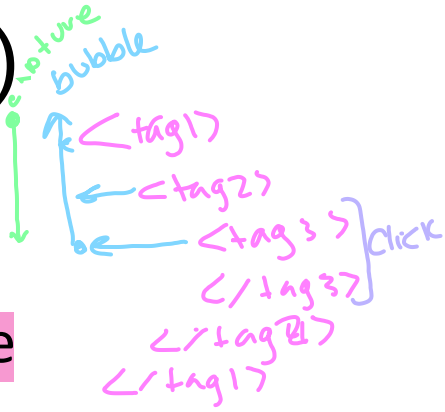
on page load

Capture vs Bubbling

Syntax:

```
document.addEventListern(event, function, capture)
```

- Event (required), event name, e.g., “click”.
- Function (required), ptr to trigger handler
- Capture (optional, default = false)
  - True = handler executed in capture phase (parent to child)
  - False = handler executed in bubbling phase (child to parent)



Prepare for next class

- Assignments
  - Mini 2 due
  - Mini 3 given out
- No labs this week
- On your own
  - Try the in-class examples

Contents

JD & DOM