BIT 2nd Year

Semester 3

IT 3505

Web Application Development II

Fundamentals of Asynchronous JavaScript and XML (AJAX) – Part 1

Introdcution to DOM

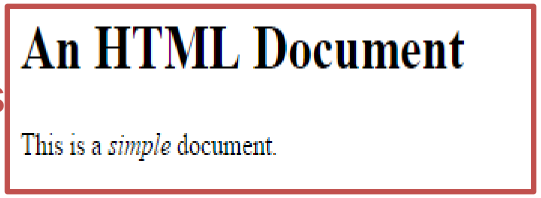
What is DOM?

- Document Object Model
  - The web browser builds a *model* of the web page (the *document*). This model is called the **Document Object Model(DOM)** and includes all the *objects* in the page (tags, text, etc.)
  - The developers can access objects in a DOM through the scripting language JavaScript.

This is what the browser reads

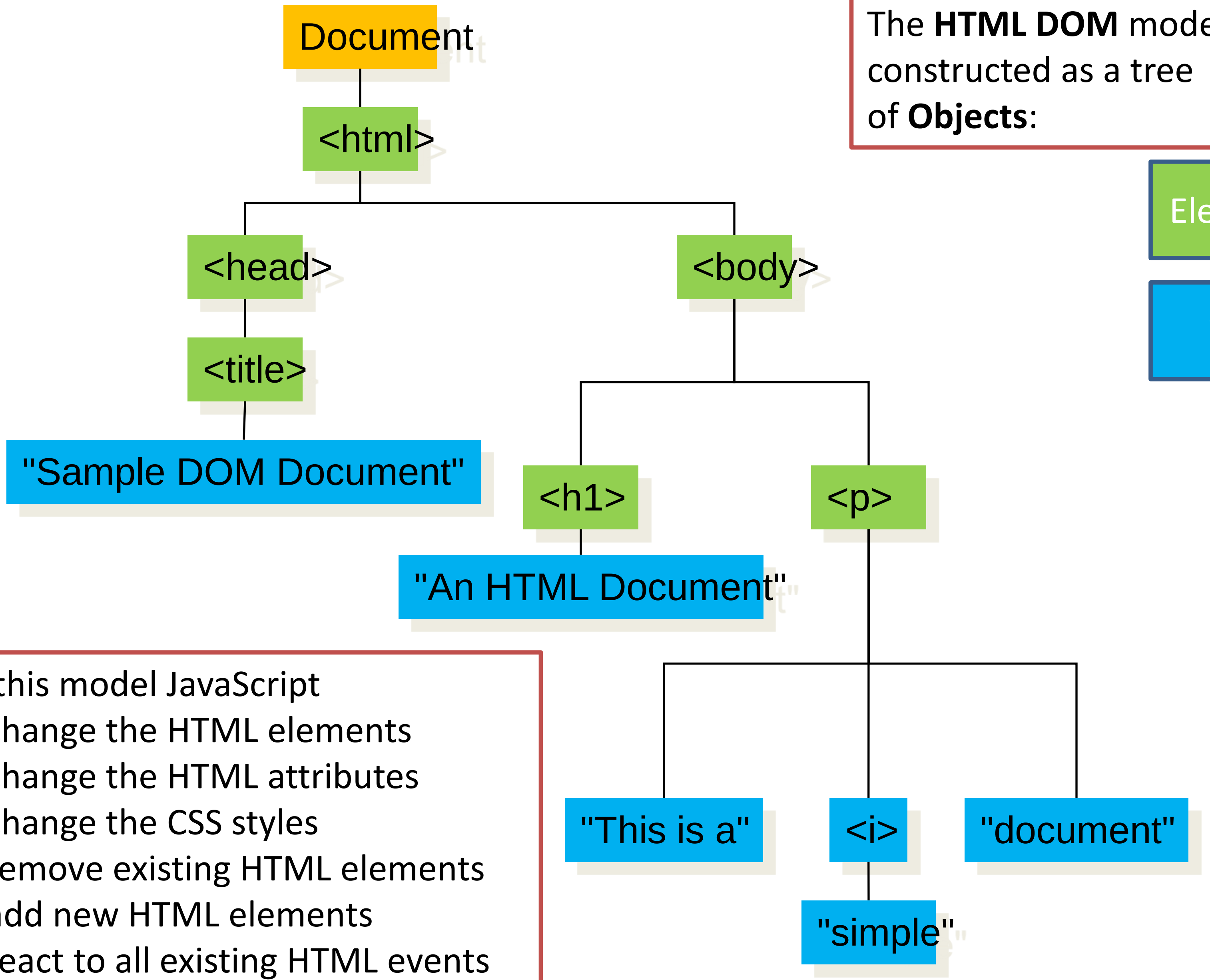
```
<html>
  <head>
    <title>Sample DOM Document</title>
  </head>
  <body>
    <h1>An HTML Document</h1>
    <p>This is a <i>simple</i> document.
  </body>
</html>
```

This is what the browser displays



An HTML Document

This is a *simple* document.



The **HTML DOM** model is constructed as a tree of **Objects**:

With this model JavaScript
•can change the HTML elements
•can change the HTML attributes
•can change the CSS styles
•can remove existing HTML elements
•can add new HTML elements
•can react to all existing HTML events
•can create new HTML events

Getting vs. Setting

Getting

```
var oldvalue =
  document.getElementById(“myID").value;

document.getElementById(“myID").value =
  "new value";
```

Setting

Getting vs. Setting

- The getElementById Method
  - This is the most common JavaScript method to

```
<html>
<body>

<p id=“myID"></p>

<script>
      document.getElementById("myID").innerHTML = “Let’s
Learn DOM”;
</script>

</body>
</html>
```

gets the content of an element

The getElementById Method

- One can even set the value of an element by using **getElementById** method

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

<p id="title">Let's Learn DOM </p>

<p id="myID"></p>

<script>
var x = document.getElementById("title");
document.getElementById("myID").innerHTML ="The value of
element is " + x.innerHTML;
</script>

</body>
</html>
```

DOM EventListener

- This method is used to attache an event handler to the specified element.
- Usage :

```
element.addEventListener(event, function, useCapture);
```

Type of the event

function need to be called

Optional : Boolean value

DOM EventListener

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

<button id="myBtn">click</button>

<script>
document.getElementById("myBtn").addEventListener("cli
ck", hello);

function hello() {
    alert ("Hello World!");
}
</script>

</body>
</html>
```

DOM EventListener

- You can add multiple eventlisteners of different types or the same type to a single element

```
element.addEventListener("mouseover",
FunctionOne);
element.addEventListener("click",
FunctionTwo);
element.addEventListener("click",
FunctionThree);
element.addEventListener("mouseout",
FunctionFour);
```

document.write()

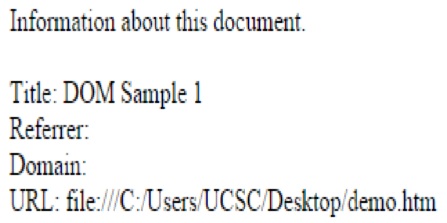
- This can be used to write directly to the HTML output stream

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

<script>
       document.write(Date());
</script>

</body>
</html>
```

document.write()

```
<html>
  <head>
    <title>DOM Sample 1</title>
  </head>
  <body>
    Information about this document.<br>
    <script type="text/javascript">
    document.write("<br>Title: ",document.title);
    document.write("<br>Referrer: ",document.referrer);
    document.write("<br>Domain: ",document.domain);
    document.write("<br>URL: ",document.URL);
    </script>
  </body>
</html>
```

$\rightarrow$



```
Information about this document.

Title: DOM Sample 1
Referrer:
Domain:
URL: file:///C:/Users/UCSC/Desktop/demo.htm
```

A complete set of DOM events can be found here

DOM Events

- By using JavaScript function can be developed to be invoked
  - When a user clicks the mouse
  - When a web page has loaded
  - When the mouse moves over an element
  - When an input field is changed
  - When an HTML form is submitted

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

<p onclick="this.innerHTML='changed'">Click
on this </p>

</body>
</html>
```

This can be any element

Some information about elements

```
<html>
  <head>
    <title>DOM Sample</title>
    <script type="text/javascript">
    function showInfo() {
      var element = document.getElementById("opener");
      var buffer = element.id + " tag is " + element.tagName;
      alert(buffer);
      element = document.getElementById("actionItem");
      buffer = element.id + " tag is " + element.tagName;
      buffer += ", type is "+element.type;
      alert(buffer);
    }
    </script>
  </head>
  <body>
    <p id="opener">The id attribute is very helpful.</p>
    <p id="closer">This is the closing paragraph.</p>
    <form>
    <button id="actionItem" type="button" onclick="showInfo()">Show Info</button>
    </form>
  </body>
</html>
```

Element doesn't refer to the value! Just the element. We have to explicitly ask for the value or the attribute values

Creating elements through Nodes

- Steps
  1. Create the element (element node)
  2. Append it to an existing element

```
<div id="myID">
<p id="p1">This is a paragraph.</p>
</div>

<script>
var p = document.createElement("p"); //creates a new <p>
element
var node = document.createTextNode("Newly,created"); // To add
text to the <p> element
p.appendChild(node); //append the text node to the <p> element
var element = document.getElementById("myID");
element.appendChild(p); // append the new element to an
existing element
</script>
```

Moving forward

- The HTML Document Object Model is a standard for structuring data on a web page
  - The field is advancing rapidly as people recognize the benefits of standardized structure and access
  - The DOM is steadily improving to cover general purpose data structuring requirements
- XML (Extendible Markup Language) also uses the Core DOM to specify its structured data
  - similar to HTML but more carefully defined