Introduction to Document Object Model (DOM)

Instructor: Sukhbir Singh Tatla
Email: sukhbir.tatla@sheridancollege.ca
Course: Web Programming (SYST10199)

DOM (Document Object Model)

- The DOM is a **W3C (World Wide Web Consortium)** standard.
- The DOM defines a standard for accessing documents like HTML and XML.

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

- The DOM defines the objects and properties of all document elements, and the methods to access them.

DOM (Document Object Model)

- Every JavaScript program running in a web browser has access to a `document` object.
- This object holds the browser's internal representation of the Document Object Model (DOM), and *contains all the information the browser retrieves from the HTML tags and attributes, CSS style rules, images, and other components that make up the source code of the page*.
- Understanding the DOM is key to becoming an effective JavaScript programmer.

DOM (Document Object Model)

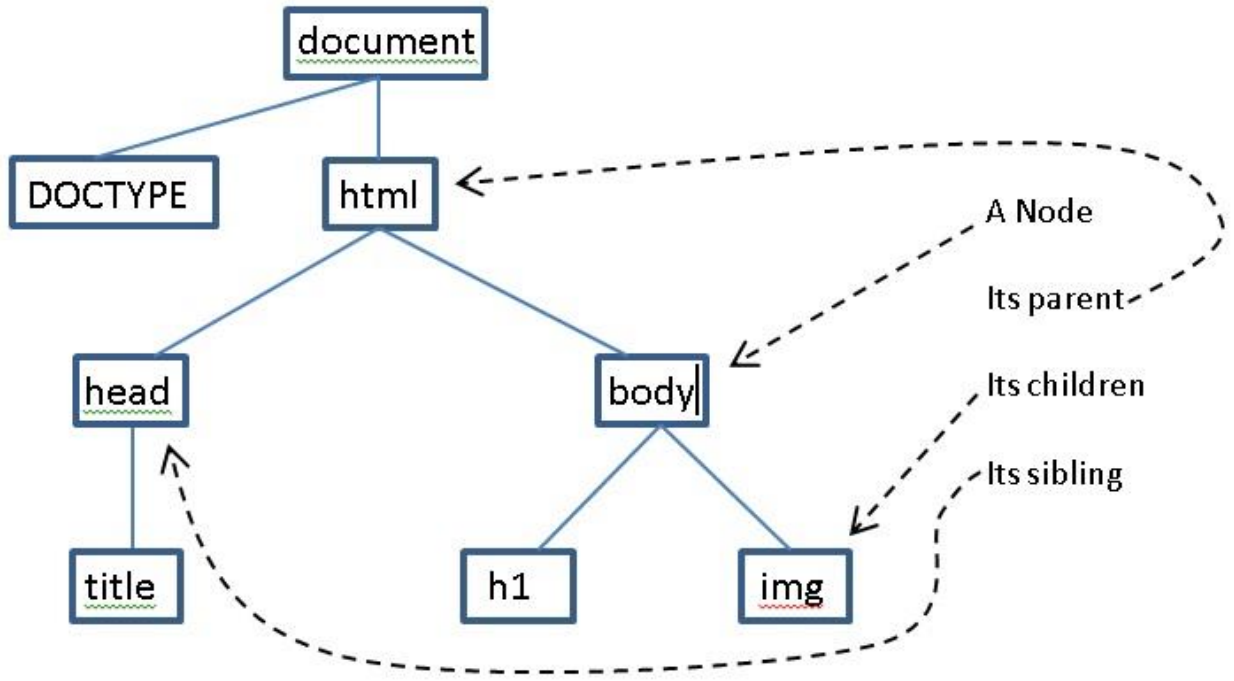
- HTML page can be viewed as a hierarchical family tree of elements containing other elements.
- When the browser reads an HTML source page,
  - it constructs an object for each element, and
  - links it to the elements it contains (the "children") and also links it to the element that contains it (the "parent").

```
<!DOCTYPE html>
<html>
   <head>
      <title>Hello World</title>
   </head>
   <body>
      <h1 id='message' class='heading'>
                        Hello, World!</h1>

      <img src='images/smiley.jpg'
                        alt='smile image'>
   </body>
</html>
```

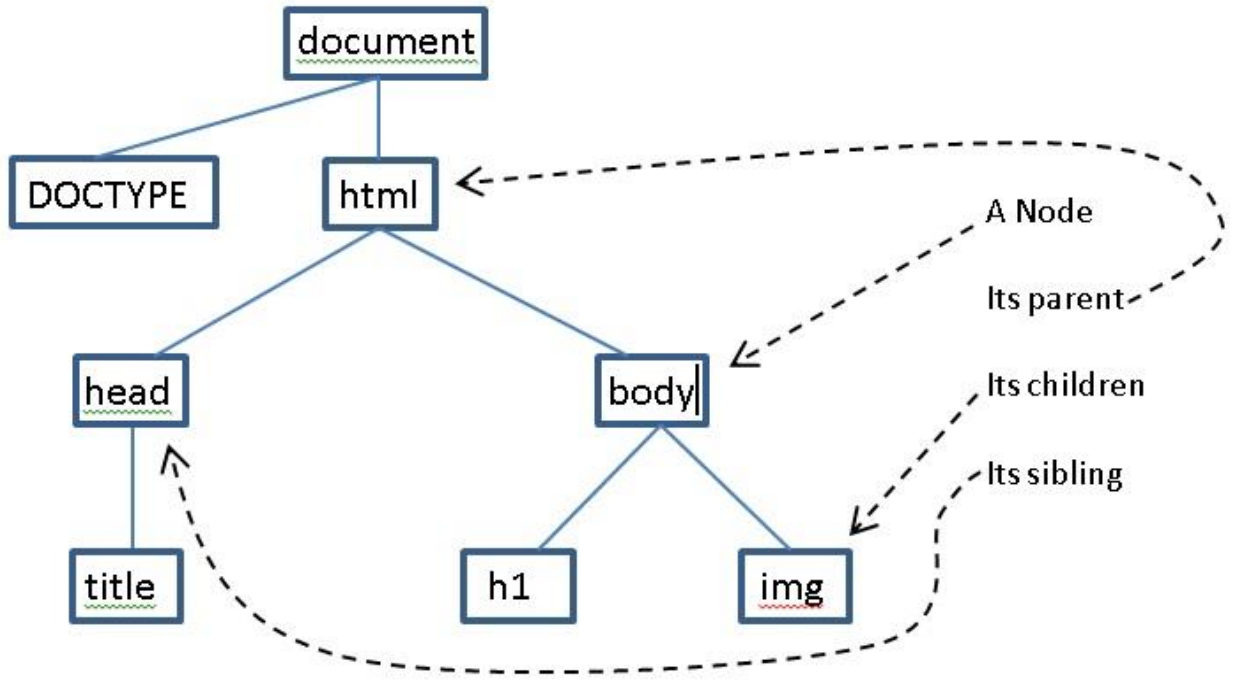
DOM (Document Object Model)

- In the previous example, the document consists of a `<!DOCTYPE>` element and an `<html>` element.
- The `<html>` element contains a `<head>` and a `<body>`.
- The `<head>` contains a `<title>` element and the `<body>` contains an `<h1>` element and an `<img>` element.
- This set of relationships can be displayed in a tree diagram like the one shown.



DOM (Document Object Model)

- The items in the boxes are referred to as ***nodes***.
- Each node is a JavaScript object that has been constructed by the browser to represent the corresponding HTML element.
- Each node object contains information about the attributes, CSS style and contents of the element it represents.
- A node can have one parent (above it), any number of children directly below it, and any number of siblings (nodes with the same parent).
- When you are using JavaScript in a web page, the built-in global variable, `document`, holds the root node of this tree.



The HTML DOM

- With this object model, JavaScript gets all the power it needs to create dynamic HTML:
  - JavaScript can change all the HTML elements in the page.
  - JavaScript can change all the HTML attributes in the page.
  - JavaScript can change all the CSS styles in the page.
  - JavaScript can react to all the events in the page.

Finding HTML Elements

- Often, with JavaScript, you want to manipulate HTML elements.

- To do so, you have to find the elements first.
- There are few of ways to do this:
  - Finding HTML elements by `id`
  - Finding HTML elements by `class`
  - Finding HTML elements by `name` attribute
  - Finding HTML elements by `tag`

Finding HTML Elements by `id`

- The easiest way to find HTML elements in the DOM, is by using the element `id`.
- This example finds the element with `id="intro"`:

```
<!DOCTYPE html>
<html>
<body>
        <p id="intro">Hello World!</p>

        <script>
                var e = document.getElementById("intro");
                document.write("<p>The text from the intro paragraph: " + e.innerHTML + "</p>");
        </script>
</body>
</html>
```

- If the element is found, the method will return the element as an object (as `e`).
- If the element is not found, `e` will contain `null`.

Finding HTML Elements by `id`

- Any element you retrieve using `document.getElementById` will be an object of type **Node**.
- Node objects contain a number of fields, each containing information associated with the corresponding HTML element.
- These fields can be used to make changes to the DOM even after the page has been loaded.
- Here are some of the useful node fields:
  - `innerHTML`: a string representing the contents of the node as HTML
  - `style`: the CSS style information associated with the node
  - `className`: the HTML class attribute
  - plus… there will be one field for every attribute specified in the corresponding tag in the original HTML source code (e.g. `id`, `src`, `href`, `value`, `type`, etc.)

🛠 Do It Yourself!

- Open helloworld.html example from SLATE -> Content -> Course Material -> Week 2 -> do-it-yourself.zip, go to the console and try the following commands, noting the return values in each case.

```
var node = document.getElementById("message");
node;
node.innerHTML;
node.style;
node.className;
```

- Now, add an `id` attribute to the `<img>` element by changing the source code.
- Now, use JavaScript statements similar to the ones above to retrieve the `src` and `alt` attributes of the image.

Changing HTML Content

- The easiest way to modify the content of an HTML element is by using the `innerHTML` property.
- To change the content of an HTML element, use this syntax:

```
var e = document.getElementById(id);
e.innerHTML = new HTML
```

```
<!DOCTYPE html>
<html>
<body>
        <p id="p1">This is a paragraph.</p>

        <script>
                alert("Press OK to see the paragraph change.");
                var e = document.getElementById("p1");
                e.innerHTML = "New text in paragraph.";
        </script>
</body>
</html>
```

🛠 Do It Yourself!

1. Modify innerHTMLExample.html so that it prompts the user for some text, and then changes the contents of the `<p>` to match what the user typed.
   - Why do you have to create the script after the `<p>` element? What happens if you place it earlier in the file?

2. Get addition.html from the In-class Source Codes. Read the comment header in the page source and follow the instructions to complete the program.

Changing HTML Style

- Changing an element's CSS information can be done most easily by accessing the `style` field of the corresponding DOM node.
- This `style` object contains all the CSS information for the `node`, where each field of the object corresponds to a CSS property.
- To change the style of an HTML element, use this syntax:

```
var e = document.getElementById(id);
e.style.property = new style
```

Changing HTML Style

- The only problem in accessing the `style` object is that you can get into problems with the hyphenated CSS style properties like `background-color`.
- The problem is that the JavaScript expression `e.style.background-color` looks like we are subtracting the variable `color` from the field `e.style.background`.
- The solution implemented in most browsers is to convert `hyphenated-property-names` to `camelCasePropertyNames`
  - e.g. `border-radius-top-left` becomes `borderRadiusTopLeft`.

Changing HTML Attribute

- You can also access, **change or add any other attributes** of an element by accessing the fields for those attributes.
- For example, if you want to change an image, you can modify its `src` attribute by accessing the `src` field of the corresponding node.
- Or if you want to change a link, you can modify the `href` attribute of the corresponding node.

🛠 Do It Yourself!

- 1. Load innerHTMLExample.html and press OK when prompted. Now hit F12 to open the console and type the following:

```
var e = document.getElementById("heading");
e.style.color = "red";
```

- Now try changing other CSS styles in the same way. Add a border, change the font, font-size, border-radius etc.
- 2. Get the file mood.html and follow the instructions in the comments at the top.

Bonus Assignment

- Fill in your Profile on SLATE
  - Fill in Profile information before Mid-Term Exams:
  - Contact Info: Email – Sheridan email id.
  - Personal Information: Fill in any 7 fields.
  - Upload a recognizable picture of yourself (remember, this is not Facebook though, so please keep it tasteful and professional)

- Why should you do it?
  - To help me get to know you faster
  - Get to know your classmates faster
  - Get a 1% bonus on your final mark which could mean the difference between a C+ and a *B* or between an *A* and an *A+*

Assignment #1

- Assignment #1 posted on SLATE.
- Submission Due date: Feb 01 2021.
- Special instructions for submission are specified in SLATE → Assessments → Assignments.
- Late submissions will be penalized 10% per day for 3 days.
- After 3 days, the assignment WILL NOT BE ACCEPTED.
- All assignments must be completed as individual efforts unless stated otherwise. Please refer to the Academic Dishonesty Policy.
- NOTE: NO makeup assignments or bonus assignments will be given at the end of the term. So be sure to complete the work when assigned.
- NOTE: "Originality Check" is implied, marks will be deduced if the percentage is greater than 50%.


"That's all Folks!"