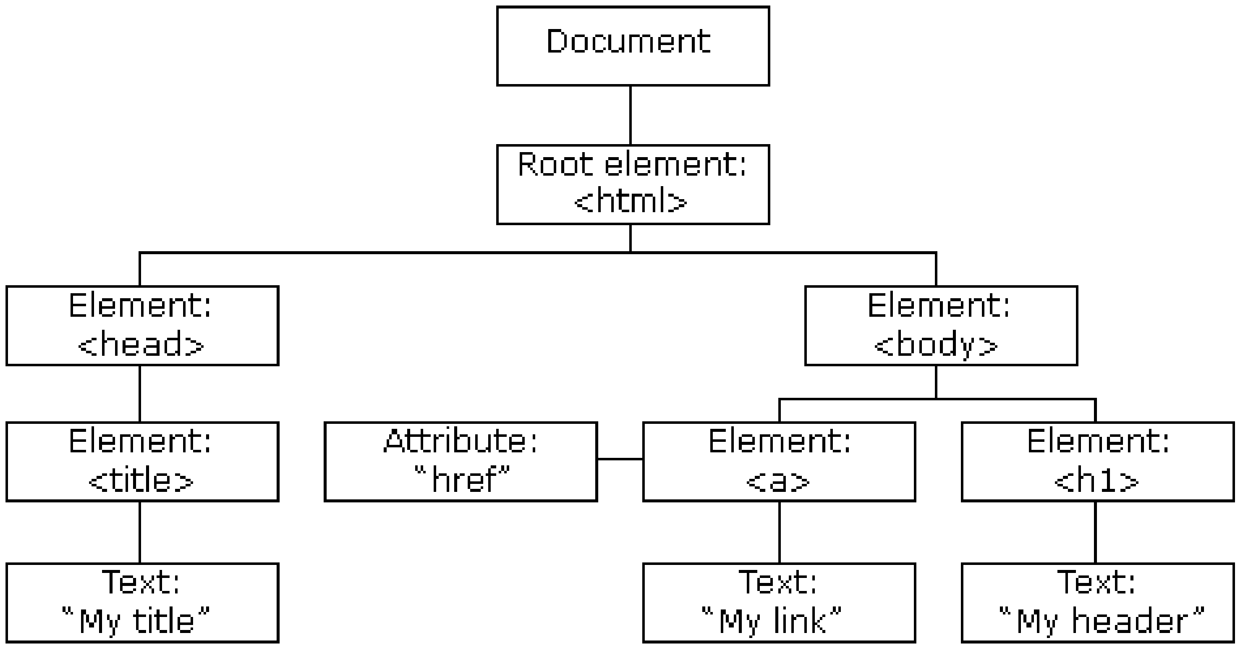
Java Script

HTML DOM


Document
Root element: <html>
Element: <head>
Element: <body>
Element: <title>
Attribute: "href"
Element: <a>
Element: <h1>
Text: "My title"
Text: "My link"
Text: "My header"

JavaScript gets all the power it needs to create dynamic HTML

- JavaScript can change all the HTML elements
- JavaScript can change all the HTML attributes
- JavaScript can change all the CSS styles
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What is the HTML DOM?

- The HTML DOM is a standard object model and programming interface for HTML. It defines:
  - The HTML elements as objects
  - The properties of all HTML elements
  - The methods to access all HTML elements
  - The events for all HTML elements

HTML DOM

- HTML DOM methods are actions you can perform (on HTML Elements).
- HTML DOM properties are values (of HTML Elements) that you can set or change.

The DOM Programming Interface

- The HTML DOM can be accessed with JavaScript (and with other programming languages).
- In the DOM, all HTML elements are defined as objects.
- The programming interface is the properties and methods of each object.
- A property is a value that you can get or set (like changing the content of an HTML element).
- A method is an action you can do (like add or deleting an HTML element).

The getElementById Method

- The getElementById Method
- The most common way to access an HTML element is to use the id of the element.
- In the example above the getElementById method used id="demo" to find the element.

The innerHTML Property

- The easiest way to get the content of an element is by using the innerHTML property.
- The innerHTML property is useful for getting or replacing the content of HTML elements.
- The innerHTML property can be used to get or change any HTML element, including <html> and <body>.

Changing HTML Elements

Property	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element
Method	Description
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element

Adding Events Handlers

- Method   Description
- document.getElementById(id).onclick = function(){
-         Code
- }   Adding event handler code to an onclick event

Finding HTML Objects

Property	Description
document.anchors	Returns all <a> elements that have a name attribute
document.applets	Returns all <applet> elements (Deprecated in HTML5)
document.baseURI	Returns the absolute base URI of the document
document.body	Returns the <body> element
document.cookie	Returns the document's cookie

Finding HTML Elements by Class Name

- If you want to find all HTML elements with the same class name, use getElementsByClassName().
- This example returns a list of all elements with class="intro".

Finding HTML Elements by HTML Object Collections

- This example finds the form element with id="frm1", in the forms collection, and displays all element values:

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

Changing the HTML Output Stream

- In JavaScript, document.write() can be used to write directly to the HTML output stream:

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

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

</body>
</html>
```

innerHTML

- document.getElementById(id).innerHTML = new HTML
- This example changes the content of a <p> element:

- Example
- <html>
- <body>

- <p id="p1">Hello World!</p>

- <script>
- document.getElementById("p1").innerHTML = "New text!";
- </script>

- </body>
- </html>

Changing HTML Style

- To change the style of an HTML element, use this syntax:
  - document.getElementById(id).style.property = new style

Changing HTML Style

```
<html>
<body>

<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color = "blue";
</script>

<p>The paragraph above was changed by a script.</p>

</body>
</html>
```

Creating nodes

```
<p>The <img src="img/cat.png" alt="Cat"> in the
  <img src="img/hat.png" alt="Hat">.</p>

<p><button onclick="replaceImages()">Replace</button></p>

<script>
  function replaceImages() {
    let images = document.body.getElementsByTagName("img");
    for (let i = images.length - 1; i >= 0; i--) {
      let image = images[i];
      if (image.alt) {
        let text = document.createTextNode(image.alt);
        image.parentNode.replaceChild(text, image);
      }
    }
  }
</script>
```

Events

- Registering event listeners
- There are 3 ways to register event handlers

EventTarget.addEventListener

-
  ```
  // Assuming myButton is a button element
  myButton.addEventListener('click', greet, false);
  function greet(event){
      // print and have a look at the event object
      // always print arguments in case of overlooking any
  other arguments
      console.log('greet:', arguments);
      alert('hello world');
  }
  ```

Event : HTML attribute

- <button onclick="alert('Hello world!’)»>
- This way should be avoided. This makes the markup bigger and less readable.
- Concerns of content/structure and behavior are not well-separated, making a bug harder to find.

Event: DOM element properties

- function print(evt) {
-   // the evt parameter is automatically assigned the event object
-   // take care of the differences between console.log & alert
-   console.log('print:', evt);
-   alert(evt);
- }
- // any function should have a appropriate name, that's what called semantic
- table_el.onclick = print;

References

- https://www.w3schools.com/js/js_htmldom_methods.asp
- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Traversing_an_HTML_table_with_JavaScript_and_DOM_Interfaces