Document Object Model (DOM)

by Sophia

WHAT'S COVERED

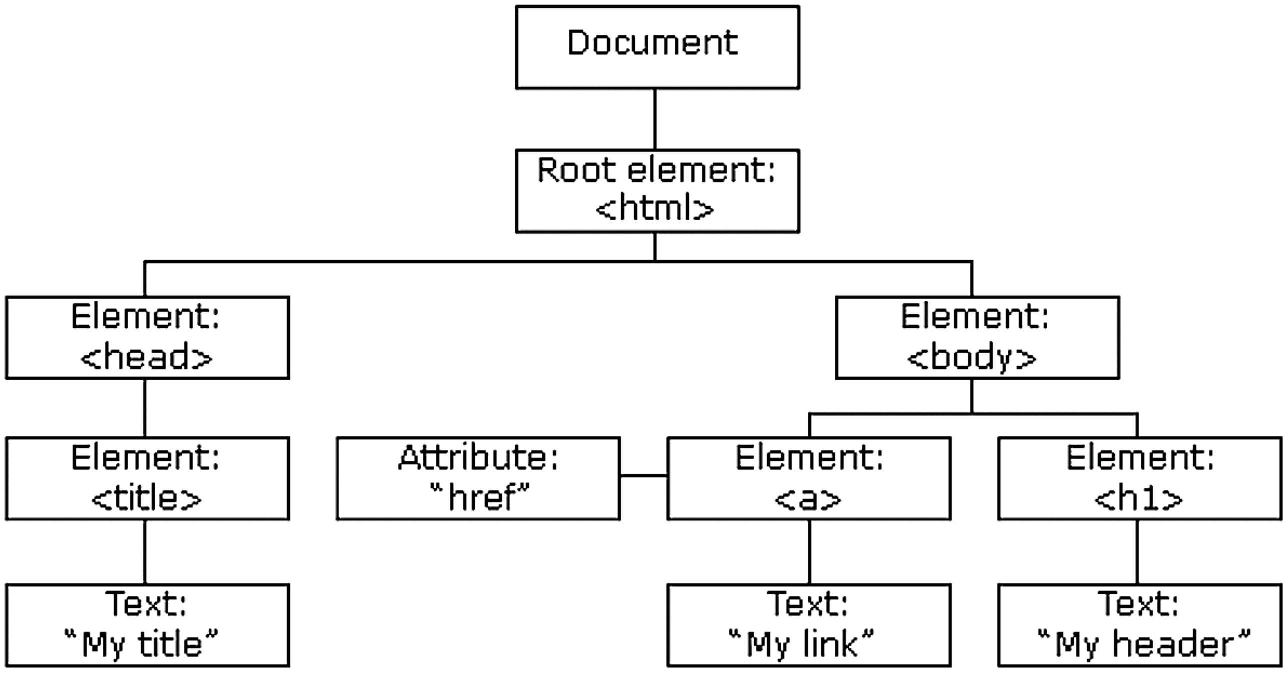
In this lesson, you will learn about the DOM (Document Object Model), its purpose, and its function from the perspective of a web developer. Additionally, you will learn about the relationship between the DOM and the JavaScript scripting language.

Specifically, this lesson will cover the following:

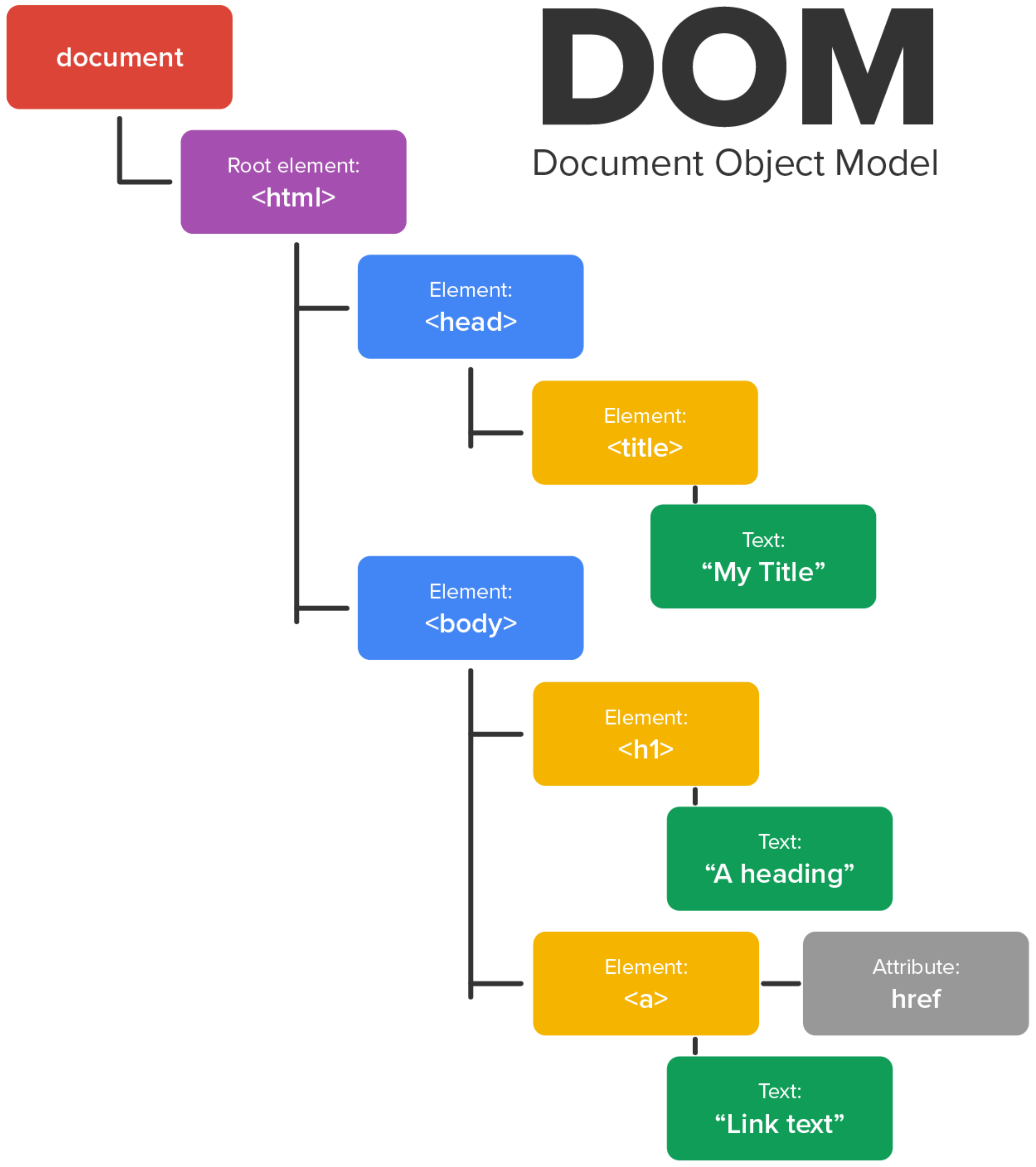
1. The DOM (Document Object Model)

Web pages are complex organizations of HTML structures in a hierarchical form. Furthermore, those HTML structures contain multiple aspects such as their attributes, events, and their internal content. In order to keep track of and manage all the details of a webpage, browsers create what is called the **Document Object Model** or the DOM. The DOM is a hierarchical and programmatic representation of an HTML webpage and its content, styles, and behaviors. The DOM was originally developed as a standard by the W3C, when after 2004 the DOM development was taken over by the WHATWG (Web Hypertext Application Technology Working Group). While the WHATWG still manages the DOM standard as a living document, the W3C periodically publishes a snapshot of the WHATWG standard documents.

EXAMPLE



Source: w3schools.com



WATCH

In this video, you will learn more about DOM.

TRY IT

Now that you have reviewed the Document Object Model, try drawing the diagram from memory to test your understanding.

Directions:

1. Get a piece of paper and something to write with.
2. Draw out the Document Object Model (DOM) diagram.
3. Ensure you have labeled each element properly.

REFLECT

Keep in mind the structure and overall organization of the nodes within the DOM. This model does not change and is how you will eventually make sense of using JavaScript to manipulate the HTML and CSS content.

When discussing the DOM, each element within the DOM structure is referred to as a node. The <html> element, the <h1> element, even the text contained within the <a> element are all nodes. A DOM starts with the “document” object which is the main container or **root node** of the DOM. This object contains not only the page structure but also any attributes and characteristics of the page itself. Next is the **root element** of the page, the **<html> node**. This node represents the starting HTML tag in an HTML webpage and everything within. Next, you see that the <html> node also contains two **child nodes**, head and body. These represent the <head> and <body> sections of the webpage.

Within the <body> element, you can see that there are two child elements, an <a> anchor node and an <h1> node. The <a> node contains an attribute “href” which would hold a URL as its value. Also, the <a> node has a Text child node which represents the actual clickable text that was placed between the starting and end <a> tags.

Additionally, there are collection objects within the document object itself that contain references to all of the same element type, such as links, images, forms, etc. The document.images collection is an array containing information about all of the images in the page and makes it easier to manage just those resources.

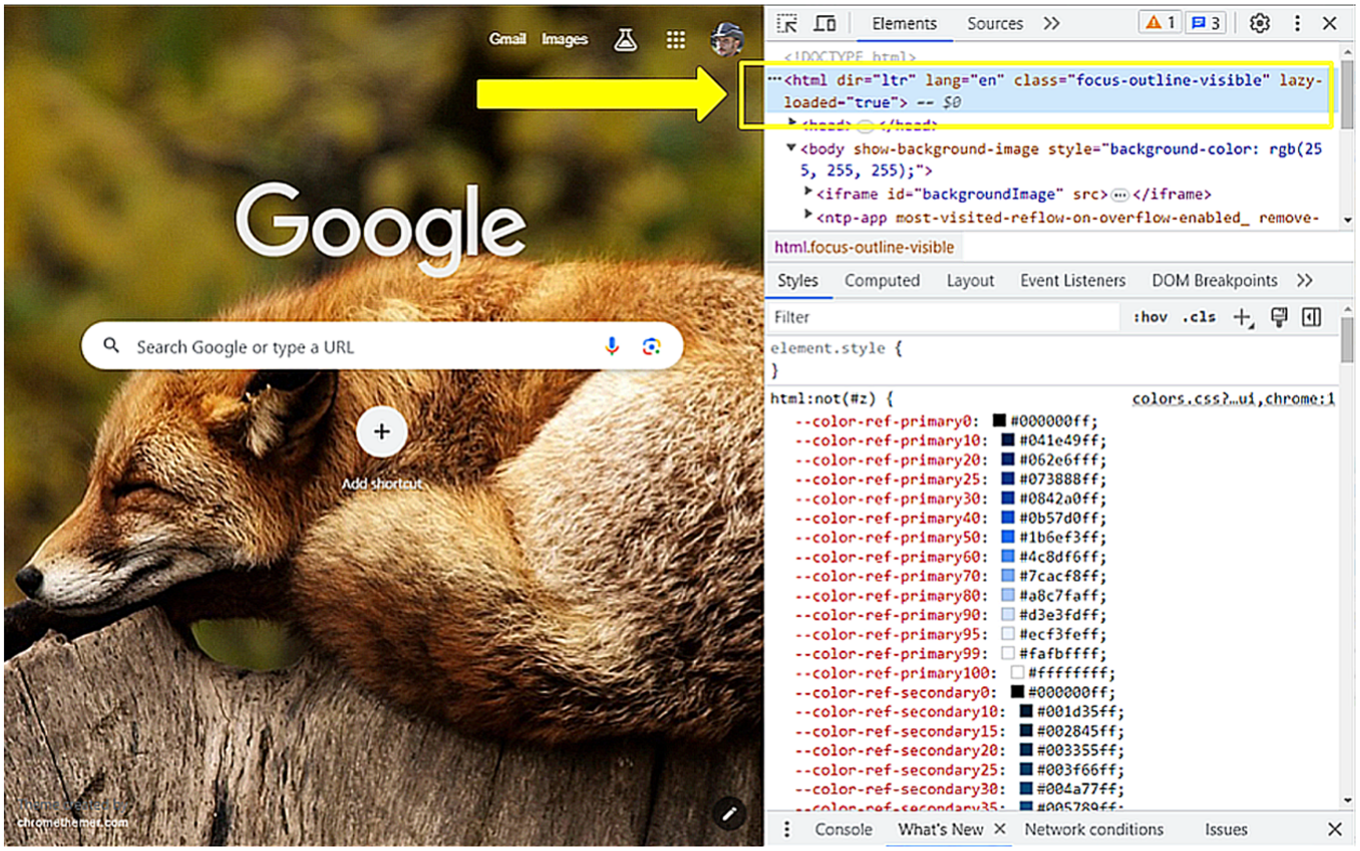
As you can see, every “thing” on a webpage is represented within the DOM structure as a node and is accessible through that structure. Understanding the intricate details of the DOM is not critical to being a web developer, however it does help tremendously when you get involved with scripting using JavaScript and other scripting or programming languages. The DOM serves as a guide as to how we can access elements, their parents, their children, and their attributes. We can even access and manipulate an element’s style properties using JavaScript as each node in the DOM contains its own STYLE attribute.

TRY IT

Directions: Now that you know more about the DOM structure, it is time to take a look at a real-world example.

1. Open your Google Chrome browser and navigate to the Google homepage or any other public webpage.

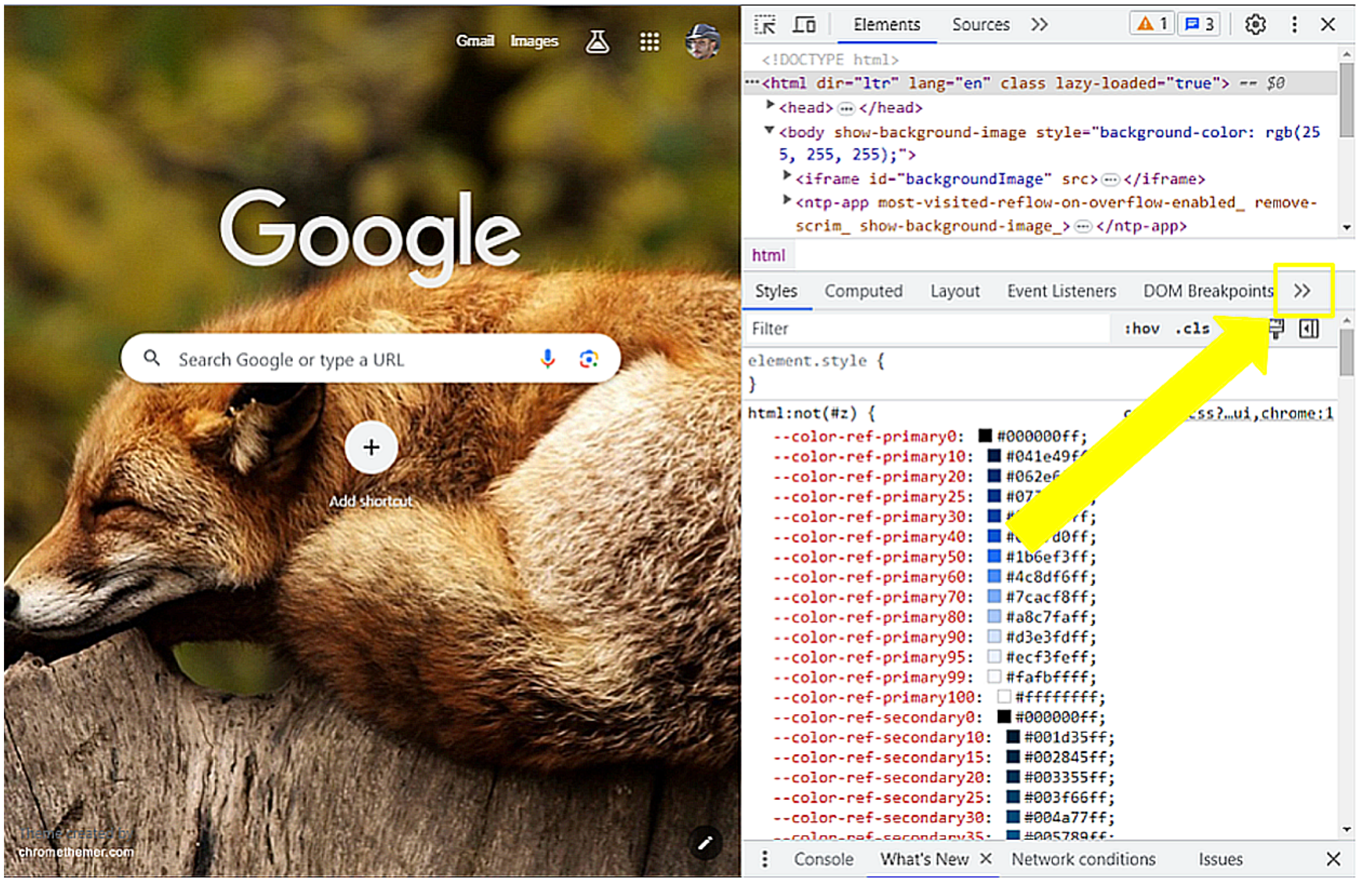
2. Open the developer tools in the Google Chrome browser by pressing F12. Click on the HTML tag at the top of the document in the “Elements” tab.



```
<!DOCTYPE html>
<html dir="ltr" lang="en" class="focus-outline-visible" lazy-
loaded="true"> == $0
  <head> ⋯ </head>
  <body show-background-image style="background-color: rgb(25
  5, 255, 255);">
    <iframe id="backgroundImage" src>⋯</iframe>
    <ntp-app most-visited-reflow-on-overflow-enabled_ remove-
```

```
html.focus-outline-visible
Styles  Computed  Layout  Event Listeners  DOM Breakpoints
Filter                                   :hov .cls
element.style {
}
html:not(#z) {                           colors.css?…ui,chrome:1
  --color-ref-primary0: #000000ff;
  --color-ref-primary10: #041e49ff;
  --color-ref-primary20: #062e6fff;
  --color-ref-primary25: #073888ff;
  --color-ref-primary30: #0842a0ff;
  --color-ref-primary40: #0b57d0ff;
  --color-ref-primary50: #1b6ef3ff;
  --color-ref-primary60: #4c8df6ff;
  --color-ref-primary70: #7cacf8ff;
  --color-ref-primary80: #a8c7faff;
  --color-ref-primary90: #d3e3fdff;
  --color-ref-primary95: #ecf3feff;
  --color-ref-primary99: #fafbffff;
  --color-ref-primary100: #ffffffff;
  --color-ref-secondary0: #000000ff;
  --color-ref-secondary10: #001d35ff;
  --color-ref-secondary15: #002845ff;
  --color-ref-secondary20: #003355ff;
  --color-ref-secondary25: #003f66ff;
  --color-ref-secondary30: #004a77ff;
```

3. Next, locate and click on the “Properties” tab, which is often hiding in one of the chevrons.



```
<!DOCTYPE html>
<html dir="ltr" lang="en" class lazy-loaded="true"> == $0
  <head> ⋯ </head>
  <body show-background-image style="background-color: rgb(25
  5, 255, 255);">
    <iframe id="backgroundImage" src>⋯</iframe>
    <ntp-app most-visited-reflow-on-overflow-enabled_ remove-
    scrim_ show-background-image_>⋯</ntp-app>
```

```
html
Styles  Computed  Layout  Event Listeners  DOM Breakpoints
Filter                                   :hov .cls
element.style {
}
html:not(#z) {                           c     ss?…ui,chrome:1
  --color-ref-primary0: #000000ff;
  --color-ref-primary10: #041e49f
  --color-ref-primary20: #062e
  --color-ref-primary25: #07
  --color-ref-primary30: #    ff;
  --color-ref-primary40: #    d0ff;
  --color-ref-primary50: #1b6ef3ff;
  --color-ref-primary60: #4c8df6ff;
  --color-ref-primary70: #7cacf8ff;
  --color-ref-primary80: #a8c7faff;
  --color-ref-primary90: #d3e3fdff;
  --color-ref-primary95: #ecf3feff;
  --color-ref-primary99: #fafbffff;
  --color-ref-primary100: #ffffffff;
  --color-ref-secondary0: #000000ff;
  --color-ref-secondary10: #001d35ff;
  --color-ref-secondary15: #002845ff;
  --color-ref-secondary20: #003355ff;
  --color-ref-secondary25: #003f66ff;
  --color-ref-secondary30: #004a77ff;
```

4. Once you locate and open the Properties tab, you will see all of the attributes and properties of the HTML node.

THINK ABOUT IT

Notice that its “childNodes” is a collection list of 2 nodes, can you guess which two nodes? That's right, the <head> and <body>. Notice also that the firstChild is head and the lastChild is body.

5. Lastly, locate the parentNode of the HTML node and expand it. These are the properties of the “document” itself which contains the HTML element as a child node and one other child node.

THINK ABOUT IT

Which is a sibling of the <html> tag?

REFLECT

Understanding the DOM structure and seeing examples of it in action will be a valuable knowledge base for you as you continue your journey to becoming a web developer.

TERMS TO KNOW

Document Object Model (DOM)
A tree-like hierarchical data representation of the objects that are represented as nodes and that comprises the structure and content of a document on the web.

Root Node
The main container of the document object in the document object model.

Root Element
The <html> element represents the top-level element of an HTML document. All other elements must be descendants of this element. Also called, *document element*.

<HTML> Node
This node represents the starting HTML tag in an HTML webpage and everything within it.

Child Node
Any node that is a descendant of another node.

2. DOM and JavaScript

One of the key uses for the DOM is to provide a standardized framework to organize types of documents so that different types of systems can access and manipulate these resources. However, more often the DOM serves as a guide to accessing and managing elements programmatically through JavaScript.

To manage content programmatically means to use a scripting language to create, remove, and change content on the page as needed. This is where "dynamic content" comes from as content can be created, modified, and removed by JavaScript as the user interacts with the site. In fact, JavaScript can be used to add, change, or remove HTML elements as well as their attributes, CSS styles, and event handlers. With JavaScript, you can get a virtual handle to one of the nodes anywhere in the DOM and through it, access its attributes, child nodes, and parent nodes.

The DOM becomes particularly helpful when adding new content to a document using JavaScript as you will need to follow the same organizational structure as you create and add new elements to the DOM. This is particularly true when developing **single page applications (SPA)** wherein an entire web application is managed within a single webpage, wherein lots of content is dynamically created, added and removed, and made to be hidden or visible. Understanding where all of the content is located within the DOM and how to access and manipulate it is very useful knowledge and will allow you to "see" into the DOM as opposed to someone wherein the DOM is a mysterious black box.

The JavaScript functions and methods used to access and manipulate the DOM will be detailed in a later unit. Just know that JavaScript is the primary tool that we use to dynamically manage content within a DOM.

TERM TO KNOW

Single Page Applications (SPA)
An approach to development wherein a web-based application is completely contained within a single webpage.

SUMMARY

In this lesson, you learned about what **the DOM (Document Object Model)** is and how it is used by systems and developers to organize the content and structure of a document, particularly HTML documents. Additionally, you learned how **JavaScript** is generally used to access, manipulate, add, and remove content nodes from the DOM as a way to provide dynamic web content.

TERMS TO KNOW

<HTML> Node
This node represents the starting HTML tag in an HTML webpage and everything within it.

Child Node
Any node that is a descendant of another node.

Document Object Model (DOM)

A tree-like hierarchical data representation of the objects that are represented as nodes and that comprises the structure and content of a document on the web.

Root Element

The <html> element represents the top-level element of an HTML document. All other elements must be descendants of this element. Also called, *document element*.

Root Node

The main container of the document object in the document object model.

Single Page Applications (SPA)

An approach to development wherein a web-based application is completely contained within a single webpage.