7.2 Document Object Model (DOM)

DOM structure

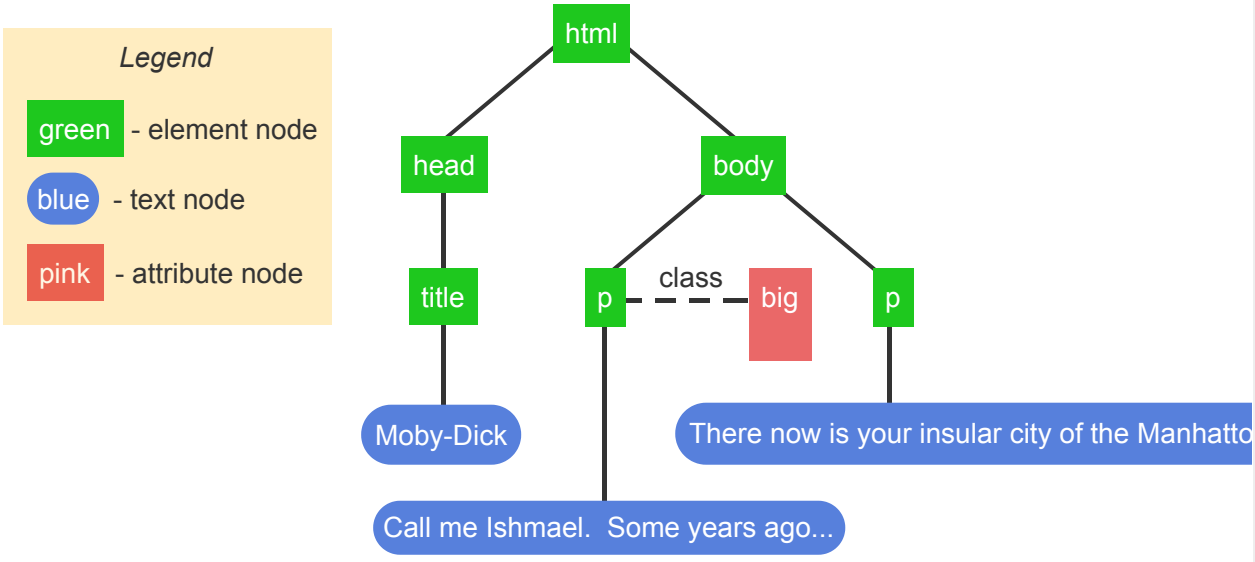
The Document Object Model (DOM) is a data structure corresponding to the HTML document displayed in a web browser. A DOM tree is a visualization of the DOM data structure. A ***node*** is an individual object in the DOM tree. Nodes are created for each element, the text between an element's tags, and the element's attributes.

- The ***root node*** is the node at the top of the DOM.
- A ***child node*** is the node directly under another node. A node can have zero, one, or more child nodes (children).
- A ***parent node*** is the node directly above another node. All nodes, except the root node, have one parent node.

PARTICIPATION ACTIVITY 7.2.1: Creating the DOM from HTML.

Start 2x speed

```
<html>
  <title>Moby-Dick</title>
  <body>
    <p class="big">Call me Ishmael.  Some years ago...</p>
    <p>There now is your insular city of the Manhattoes...</p>
  </body>
</html>
```



Captions

1. The web browser reads the HTML and creates the DOM's root node from the html element.

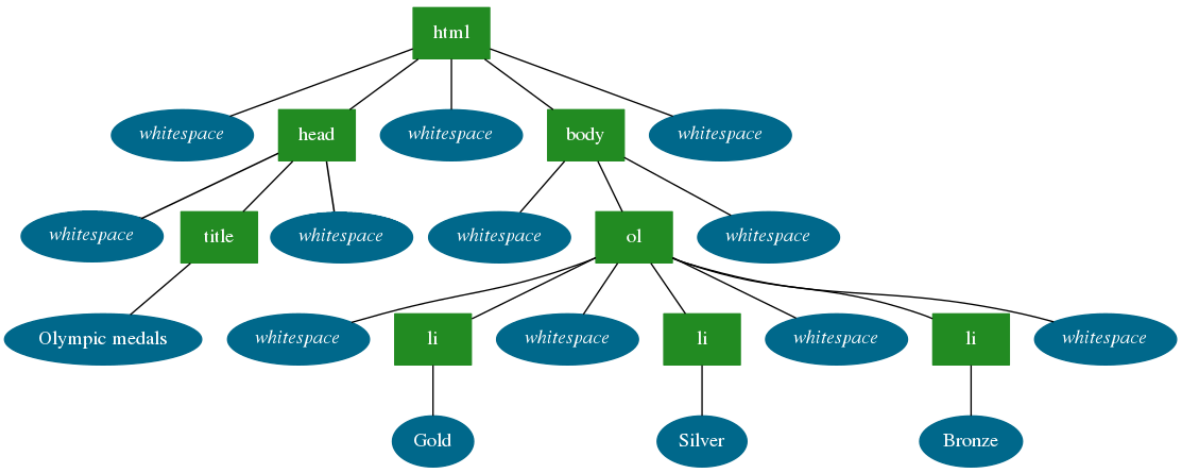
2. Although no head element exists in the HTML, a head node is created as a child of the html node. The title node is added as a child of the head node.
3. A text node is created for the title element's text content.
4. The body node is a child of the root node. The p element is contained within the body element, so the p node is a child of the body node.
5. An attribute node is created for the p element's class attribute. Attribute nodes are always connected to element nodes and are not considered children.
6. The browser continues reading the HTML and creating DOM nodes.

Feedback?

An idealized representation of the DOM tree excludes text nodes that only contain whitespace. However, a web developer occasionally needs to know the complete DOM tree, which includes whitespace as shown in the example below.

Figure 7.2.1: Complete DOM tree visualization with whitespace text nodes.

```
<!DOCTYPE html>
<html>
   <head>
      <title>Olympic medals</title>
   </head>
   <body>
      <ol>
         <li>Gold</li>
         <li>Silver</li>
         <li>Bronze</li>
      </ol>
   </body>
</html>
```

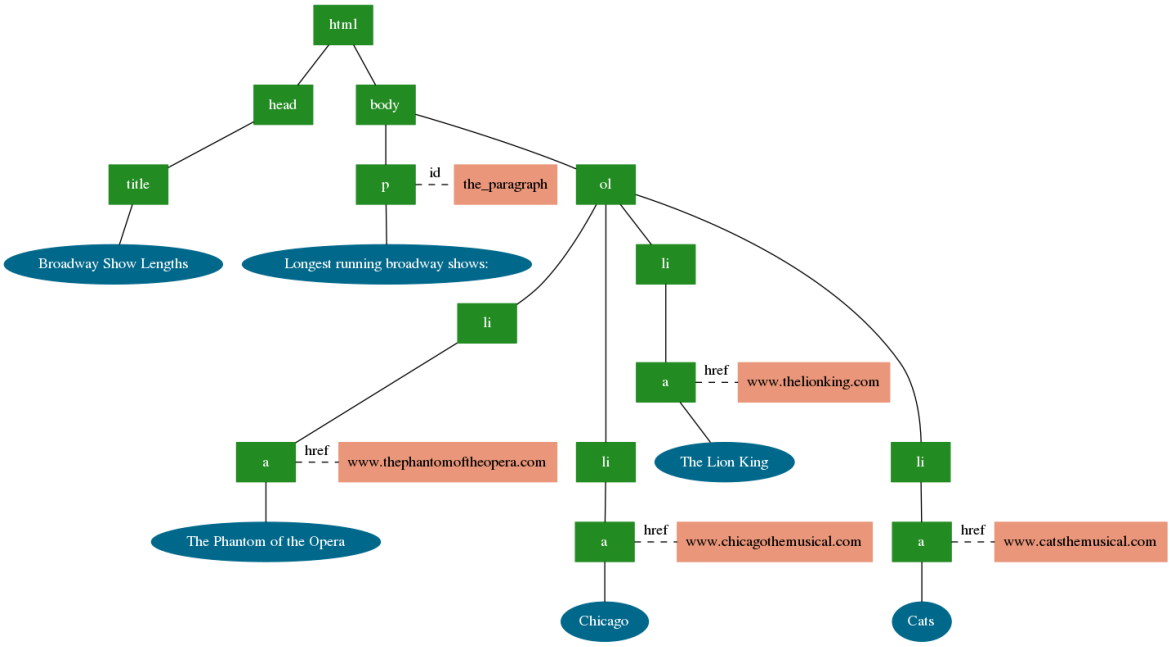


Feedback?

PARTICIPATION ACTIVITY 7.2.2: DOM structure.

Refer to the HTML and resulting DOM below.

```
<!DOCTYPE html>
<html>
   <head>
      <title>Broadway Show Lengths</title>
   </head>
   <body>
      <p id="the_paragraph">Longest running broadway shows:</p>
      <ol>
         <li><a href="http://www.thephantomoftheopera.com/">The
Phantom of the Opera</a></li>
         <li><a href="http://www.chicagothemusical.com/">Chicago</a>
</li>
         <li><a href="http://www.thelionking.com/">The Lion King</a>
</li>
         <li><a href="http://www.catsthemusical.com/">Cats</a></li>
      </ol>
   </body>
</html>
```



1) Which node is the root of the DOM tree?

- (•) html
- ( ) a
- ( ) body

Correct

An HTML document's DOM tree always has an html root node. The root node is the only node with no parent.

2) Which of the following DOM nodes is never a child node?

- Element node
- Attribute node
- Text node

Correct

Element attributes become attribute nodes. Attribute nodes are attached to element nodes and are not considered to be children.

3) How many child nodes does the p element have?

- 0
- 1
- 2

Correct

The p element has one text node child, which contains the paragraph's text. The pink box represents the paragraph's `id` attribute node and is not considered a child node.

4) How many child nodes does the ol element have?

- 0
- 2
- 4

Correct

All li elements between `<ol>` and `</ol>` tags are children of the ol element.

5) Which DOM nodes never have child nodes?

- Element nodes
- Attribute nodes only
- Text nodes and attribute nodes

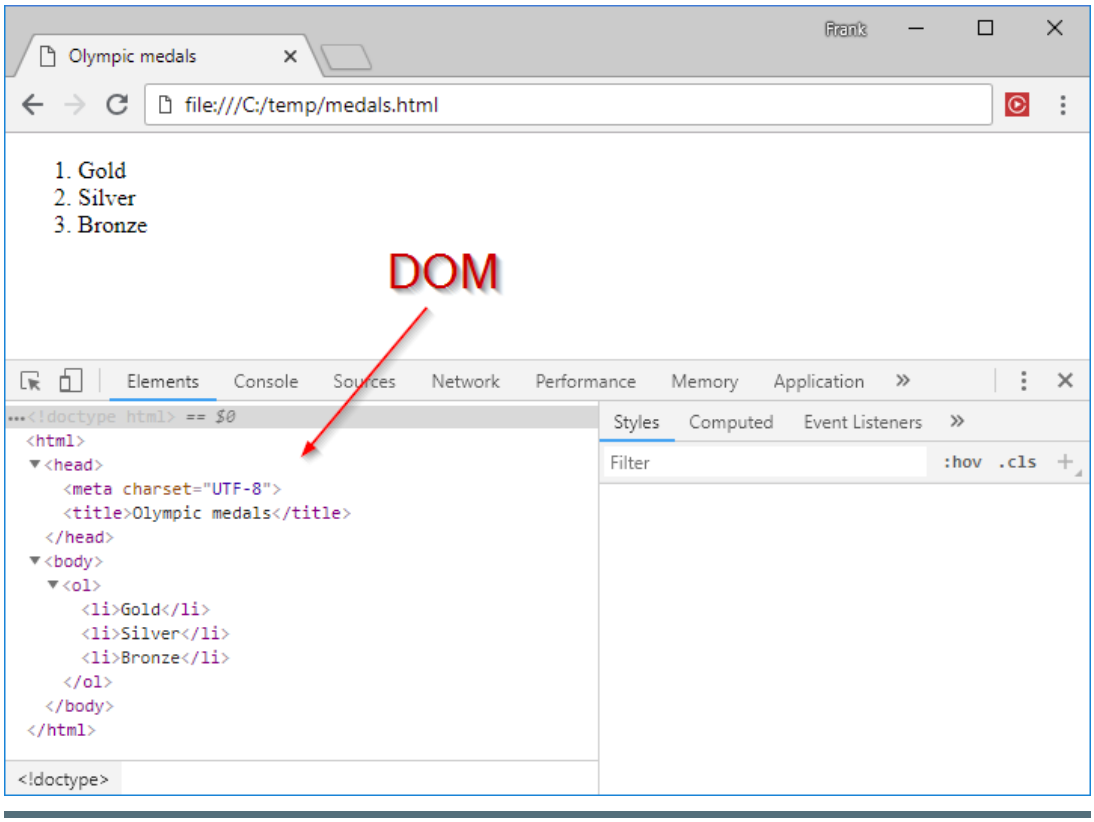
Correct

Only element nodes have children. Text nodes are always children of element nodes. Attribute nodes are not considered to be children.

Feedback?

Viewing the DOM in Chrome

The Chrome DevTools can display an HTML document's DOM by pressing Ctrl+Shift+C on Windows or Ctrl+Option+C on a Mac. The DOM may differ from the HTML. Ex: The `<head>` tag may be missing from the HTML file but is visible in the DOM below because `<meta>` and `<title>` elements are always placed in the `<head>` element.



Searching the DOM

JavaScript is commonly used to search the DOM for a specific node or set of nodes and then change the nodes' attributes or content. Ex: In an email application, the user may click a Delete button to delete an email. The JavaScript must search the DOM for the node containing the email's contents and then change the contents to read "Email deleted".

The `document` object provides five primary methods that search the DOM for specific nodes:

1. The ***document.getElementById()*** method returns the DOM node whose `id` attribute is the same as the method's parameter.
   Ex: `document.getElementById("early_languages")` returns the p node in the HTML below.
2. The ***document.getElementsByTagName()*** method returns an array of all the DOM nodes whose type is the same as the method's parameter.
   Ex: `document.getElementsByTagName("li")` returns an array containing the four li nodes from in the HTML below.
3. The ***document.getElementsByClassName()*** method returns an array containing all the DOM nodes whose `class` attribute matches the method's parameter.
   Ex: `document.getElementsByClassName("traditional")` returns an array containing the `ol` node with the `class` attribute matching the word traditional.
4. The ***document.querySelectorAll()*** method returns an array containing all the DOM nodes that match the CSS selector passed as the method's parameter.

Ex: `document.querySelectorAll("li a")` returns an array containing the two anchor nodes in the HTML below.

5. The ***document.querySelector()*** method returns the first element found in the DOM that matches the CSS selector passed as the method's parameter. `querySelector()` expects the same types of parameters as `querySelectorAll()` but only returns the first element found while navigating the DOM tree in a depth-first traversal.
   Ex: `document.querySelector("li")` returns the li node about Fortran.

A DOM search method name indicates whether the method returns one node or an array of nodes. If the method name starts with "getElements" or ends in "All", then the method returns an array, even if the array contains one node or is empty. `getElementById()` and `querySelector()` either return a single node or null if no node matches the method arguments.

PARTICIPATION ACTIVITY 7.2.3: Searching the DOM by id and by element.

Start 2x speed

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Early Programming Languages</title>
      <script src="dom.js" defer></script>
   </head>
   <body>
      <p id="early_languages">
         Early programming languages still in use:
      </p>

      <ol class="traditional">
         <li><a href="https://w.wiki/4ZAb">Fortran - 1954</a></li>
         <li><a href="https://w.wiki/4NzE">Lisp - 1958</a></li>
         <li>COBOL - 1959</li>
         <li>BASIC - 1964</li>
      </ol>
   </body>
</html>
```

dom.js

```
let para = document.getElementById("early_languages");
let listItems = document.querySelectorAll("li");
```

Captions

1. The webpage loads dom.js with the defer attribute, so the code in dom.js only executes after the webpage and DOM are finished loading.
2. dom.js calls getElementById() to find the paragraph with id "early_languages".
3. dom.js calls querySelectorAll() to find all four li elements.

Feedback?

HTMLCollection and NodeList

Technically, `getElementsByTagName()` and `getElementsByClassName()` return an HTMLCollection, and `querySelectorAll()` returns a NodeList. ***HTMLCollection*** *is an interface representing a generic collection of elements. A* ***NodeList*** *is an object with a collection of nodes. HTMLCollection and NodeList both act like an array. Both have a `length` property, and elements can be accessed with braces. Ex: `elementList[0]` is the first element in an HTMLCollection or NodeList.*

PARTICIPATION ACTIVITY 7.2.4: DOM traversal.

Refer to the HTML below.

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <title>Web development languages</title>
      <link rel="stylesheet" href="styles.css">
      <script src="dom.js" defer></script>
   </head>
   <body>
      <p class="special">
         Languages used in web development:
      </p>
      <ul id="list">
         <li id="item-1">HTML for content</li>
         <li id="item-2">CSS for presentation</li>
         <li id="item-3">JavaScript for functionality</li>
      </ul>
      <p class="special">
         <a
href="https://en.wikipedia.org/wiki/Web_development">More
information</a>
      </p>
   </body>
</html>
```

Select the element(s) or value returned by the given search method.

1) `document.getElementById("list")`

- (●) ul element
- ( ) First li element
- ( ) All li elements

Correct

The unordered list tag's `id` attribute is "list" and is returned by the `getElementById()` method.

2) `document.getElementsByTagName("li")`

Correct

- ul element
- First li element
- All li elements

All three list item tags match and are returned in an array.

3) `document.getElementsByTagName("ul")[0]`

- ul element
- First li element
- null

Correct

The ul element is the first and only element in the array returned by `getElementsByTagName("ul")`. The `[0]` selects the ul from the single element array.

4) `document.querySelectorAll("li")`

- ul element
- First li element
- All li elements

Correct

All three list item nodes match the element selector "li" and are returned in an array.

5) `document.querySelector("#list")`

- ul element
- First li element
- null

Correct

The CSS selector #list matches the `<ul>` tag's id "list".

6) `document.querySelector(".special")`

- First p element
- Last p element
- Both p elements

Correct

The first element that uses the CSS selector ".special" is the first paragraph. `querySelector()` only returns the first element found.

7) `document.querySelector("item-3")`

- ul element
- Last li element
- null

Correct

The argument "item-3" is treated as an element selector, and no element called item-3 exists. The argument should be "#item-3" to return the last `<li>` tag with id "item-3".

Feedback?

Modifying DOM node attributes

After searching the DOM for an element, JavaScript may be used to examine the element's attributes or to change the attributes. By modifying attributes, JavaScript programs can perform actions including:

- Change which image is displayed by modifying an img element's `src` attribute.

- Determine which image is currently displayed by reading the img element's `src` attribute.
- Change an element's CSS styling by modifying an element's `style` attribute.

Every attribute for an HTML element has an identically named property in the element's DOM node. Ex: `<a href="https://www.nasa.gov/" id="nasa_link">NASA</a>` has a corresponding DOM node with properties named `href` and `id`. Each attribute property name acts as both a getter and a setter.

- Getter: Using the property name to read the value allows a program to examine the attribute's value. Ex:
  `nasaUrl = document.getElementById("nasa_link").href` assigns `nasaUrl` the string `"https://www.nasa.gov/"` from the anchor element's `href` attribute.
- Setter: Writing to a property allows a program to modify the attribute, which is reflected in the rendered webpage. Ex:
  `document.getElementById("nasa_link").href = "https://www.spacex.c` changes the element's hyperlink to the SpaceX URL.

An element's attribute can be removed using the element method ***removeAttribute()***. Ex: `document.getElementById("nasa_link").removeAttribute("href")` removes the link from the anchor element so that clicking on the HTML element no longer performs an action.

PARTICIPATION ACTIVITY 7.2.5: Modifying DOM node attributes.

Start 2x speed

```
<body>
   <img src="mountain.jpg"
        alt="Mountain photo">
   <p style="color: green">
      I love the outdoors!
   </p>
</body>
```

```
let img = document.querySelector("img");
img.src = "mountain.jpg";
img.alt = "Mountain photo";

let p = document.querySelector("p");
p.style = "color: green";
```

I love the outdoors!

Captions

1. The webpage shows a photo of a lake with a sentence underneath.
2. Changing the img element's src attribute causes the webpage to replace lake.jpg with mountain.jpg.

3. Setting the p element's style attribute changes the paragraph's CSS styling, making the text green.

Feedback?

PARTICIPATION ACTIVITY 7.2.6: Reading and modifying DOM node attribute values.

Refer to the HTML below.

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <title>Orion mission</title>
      <script src="dom.js" defer></script>
   </head>
   <body>
      <h1>Orion Moon Mission Update</h1>
      <img src="inFlight.png" id="status" width="200"
           alt="Image of spaceship en route to the moon">
   </body>
</html>
```

1) Enter the missing attribute to make the image display success.png.

```
let pic =
document.getElementById("status");
pic._____ = "success.png";
```

Check Show answer

Answer

src

Setting the `src` attribute value causes the browser to display the specified image.

2) Enter the node and attribute required to read the current image name.

```
let pic =
document.getElementById("status");
if ( _____ != "success.png") {
   alert("Wrong picture!");
}
```

Check Show answer

Answer

pic.src

`pic.src` can be used as a getter to read the current value of the image's `src` attribute.

3) Enter the method name to remove the image's `width` attribute.

Answer

removeAttribute

```
let pic =
document.getElementById("status");
pic._____("width");
```

`removeAttribute()` removes an element's attribute. Removing the image's `width` attribute makes the image use the image file's actual pixel width.

Check Show answer

Feedback?

Modifying DOM node content

After searching the DOM for an element, JavaScript may be used to examine or change the element's content.

Two common properties are used to get or set an element's content:

1. The ***textContent*** property gets or sets a DOM node's text content. Ex: `document.querySelector("p").textContent = "$25.99";` changes the paragraph to `<p>$25.99</p>`.
2. The ***innerHTML*** property gets or sets a DOM node's content, including all of the node's children, using an HTML-formatted string. Ex: `document.querySelector("p").innerHTML = "<strong>$25.99</strong>"` changes the paragraph to `<p><strong>$25.99</strong></p>`.

The `innerHTML` property uses an internal parser to create any new DOM nodes. `textContent`, however, only creates or changes a single text node. For setting an element's text, `textContent` is somewhat faster than `innerHTML` because no HTML parsing is performed.

PARTICIPATION ACTIVITY 7.2.7: Modifying a webpage with textContent and innerHTML.

Start 2x speed

```
<body>
   <p>
      Most common girl names in 2020:
   </p>
   <ol>
      <li>Emma</li>
      <li>Olivia</li>
      <li>Ava</li>
   </ol>
</body>
```

Most common girl names in 2020:

1. Emma
2. Olivia
3. Ava

```
let p = document.querySelector("p");
p.textContent = "Most common girl names in 2020:";

let ol = document.querySelector("ol");
ol.innerHTML = "<li>Emma</li><li>Olivia</li><li>Ava</li>";
```

Captions

1. Assigning the textContent property replaces the paragraph's text.
2. Assigning the innerHTML property replaces the entire list.

Feedback?

Less common ways to change node content

*The **nodeValue** property gets or sets the value of text nodes. As the DOM tree represents textual content separately from HTML elements, the textual content of an HTML element is the first child node of the HTML element's node. So, to access the textual content of an HTML element within the DOM, `firstChild.nodeValue` is used to access the value of the HTML's element's first child.*

Ex:

```
document.getElementById("saleprice").firstChild.nodeValue = "$.
```

1. *Gets the DOM node for the element with id "saleprice".*
2. *Uses `firstChild` to access the textual content node for the element.*
3. *Uses `nodeValue` to update the content.*

*The **innerText** property gets or sets a DOM node's rendered content. `innerText` is similar to `textContent`, but `innerText` is aware of how text is rendered in the browser. Ex: In `<p>Testing one</p>`, `textContent` returns "Testing one" with spaces, but `innerText` returns "Testing one" with the spaces collapsed into a single space.*

PARTICIPATION ACTIVITY 7.2.8: Modify the DOM nodes.

An HTML element using the ***hidden attribute*** is not displayed by the web browser.

Add JavaScript in the `changePage()` function so that clicking on the Use Current Astronomy button does the following:

1. Uses `removeAttribute()` to remove the `hidden` attribute from the paragraph with the id `p2`, causing the paragraph to become visible.
2. Uses the `textContent` property of the span with id `lastPlanet` to change the name of the farthest planet to "Neptune".
3. Uses the `innerHTML` property of the paragraph with id `p4` to add a link:
   `<a href="https://en.wikipedia.org/wiki/Pluto">Source</a>`

Use an appropriate DOM search method like `document.getElementById()` to access the DOM nodes.

HTML | JavaScript

```
<body>
   <h1>The farthest planet</h1>

   <p id="p1">Pluto was discovered in 1930 and designated as a p
   <p id="p2" hidden>In 2006, Pluto was reclassified as a dwarf
   <p id="p3"><span id="lastPlanet">Pluto</span> is the farthes
   <p id="p4"></p>

   <input type="button" value="Use Current Astronomy">
</body>
```

Render webpage Reset code

Your webpage

The farthest planet

Pluto was discovered in 1930 and designated as a planet.

Pluto is the farthest planet from the Sun.

Use Current Astronomy

Expected webpage

The farthest planet

Pluto was discovered in 1930 and designated as a planet.

Pluto is the farthest planet from the Sun.

Use Current Astronomy

▼ View solution

Explain

```
--- START FILE: HTML ---


<body>
   <h1>The farthest planet</h1>
```

```
   <p id="p1">Pluto was discovered in 1930 and designated as
a planet.</p>
   <p id="p2" hidden>In 2006, Pluto was reclassified as a
dwarf planet.</p>
   <p id="p3"><span id="lastPlanet">Pluto</span> is the
farthest planet from the Sun.</p>
   <p id="p4"></p>

   <input type="button" value="Use Current Astronomy">
</body>


--- END FILE: HTML ---

--- START FILE: JavaScript ---

const button = document.getElementsByTagName("input")[0];
button.addEventListener("click", changePage);

function changePage() {
   let p = document.getElementById("p2");
   p.removeAttribute("hidden");

   let span = document.getElementById("lastPlanet");
   span.textContent = "Neptune";

   p = document.getElementById("p4");
   p.innerHTML = '<a
href="https://en.wikipedia.org/wiki/Pluto">Source</a>';
}


--- END FILE: JavaScript ---
```

Feedback?

PARTICIPATION ACTIVITY 7.2.9: Changing DOM node content.

Refer to the HTML and JavaScript below.

sitcom.html

```
<!DOCTYPE HTML>
<html lang="en">
   <head>
      <title>Sitcoms</title>
      <script src="sitcom.js" defer></script>
   </head>
   <body>
      <p>My <span>favorite</span> TV sitcoms:</p>
      <ol>
         <li><cite>The Office</cite></li>
         <li><cite>Community</cite></li>
         <li><cite>The Fresh Prince of Bel-Air</cite></li>
      </ol>
   </body>
</html>
```

sitcom.js

```
let span = document.querySelector("span");
span.__A__ = "least favorite";

let listItems = document.__B__("li");
for (let item of listItems) {
   item.__C__ = "<cite>Happy Hour</cite>";
}
```

Match the letters to the missing JavaScript property or method.

If unable to drag and drop, refresh the page.

B — `querySelectorAll`

`querySelectorAll()` selects all li elements.

Correct

C — `innerHTML`

The for-of loop iterates through all li elements. The `innerHTML` property changes every li element to `<li><cite>Happy Hour</cite></li>`.

Corr

A — `textContent`

The `textContent` property sets the span element's text so the paragraph displays "My least favorite TV sitcoms:".

Correct

Reset

Feedback?

CHALLENGE ACTIVITY 7.2.1: Using the Document Object Model.

530096.4000608.qx3zqy7

Start

1 2 3 4

All steps in this Challenge Activity require calling a document method to search the DOM. Write the JavaScript to assign listNodes with all elements with a class name of 'language-item'.

HTML JavaScript

```
1  <h1>Top 10 TIOBE index for March 2018:</h1> <!-- https://www.tio
2  <ol class="languages-list">
3     <li class="language-item">Java</li>
4     <li>C</li>
5     <li class="language-item">C++</li>
6     <li class="language-item">Python</li>
7     <li class="language-item">C#</li>
8     <li>Visual Basic .NET</li>
9     <li>PHP</li>
10    <li>JavaScript</li>
11    <li>Ruby</li>
12    <li class="language-item">SQL</li>
13 </ol>
```

1 2 3 4

Check Next

View your last submission

Feedback?

Exploring further:

- Document Object Model (DOM) from MDN

How was this section?

Provide section feedback