Manipulating the DOM

Web Development I

What is the “DOM”?

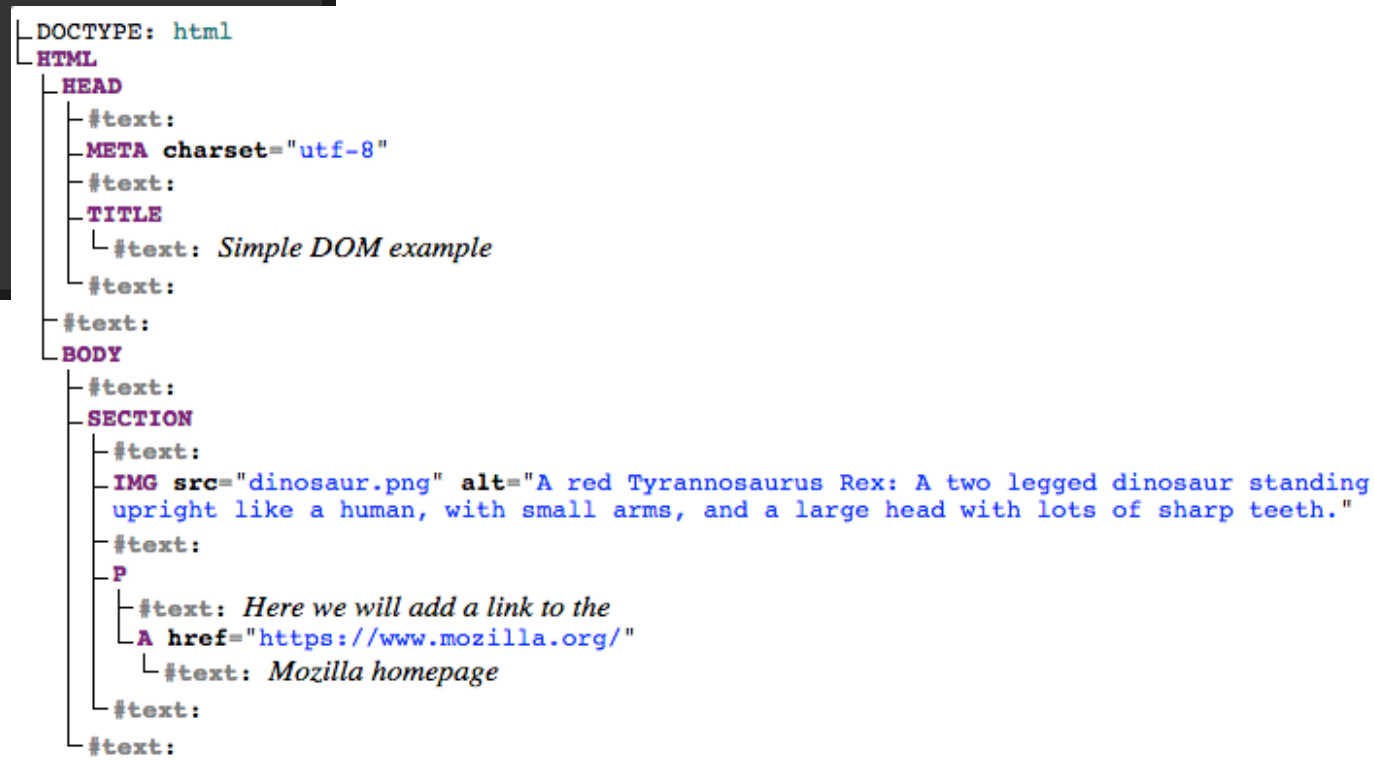
- Abbreviation for the Document Object Model
- Each web page is a “document”
- The DOM is an in memory, tree structure representation, of that document (or web page)
- Enables the HTML structure to be easily accessed by programming languages
- The browser itself uses it to apply styling and other information to the correct elements as it renders a page
- Developers like you can manipulate the DOM with JavaScript AFTER the page has been rendered.

HTML

```html
<!doctype html>
<html lang="en-US">
  <head>
    <meta charset="utf-8" />
    <title>Simple DOM example</title>
  </head>
  <body>
    <section>
      <img
        src="dinosaur.png"
        alt="A red Tyrannosaurus Rex: A two legged dinosaur standing upright like a
human, with small arms, and a large head with lots of sharp teeth." />
      <p>
        Here we will add a link to the
        <a href="https://www.mozilla.org/">Mozilla homepage</a>
      </p>
    </section>
  </body>
</html>
```

← The HTML

The DOM →



https://software.hixie.ch/utilities/js/live-dom-viewer/

Live DOM Viewer

Markup to test (permalink, save, upload, download, hide):

```
<!DOCTYPE html>
...
```

DOM view (hide, refresh):

```
├─DOCTYPE: html
└─HTML
  ├─HEAD
  └─BODY
    └─#text: ...
```

Rendered view: (hide):

...

innerHTML view: (show, refresh):

Log: (hide):

```
rendering mode: CSS1Compat
document has no title
```

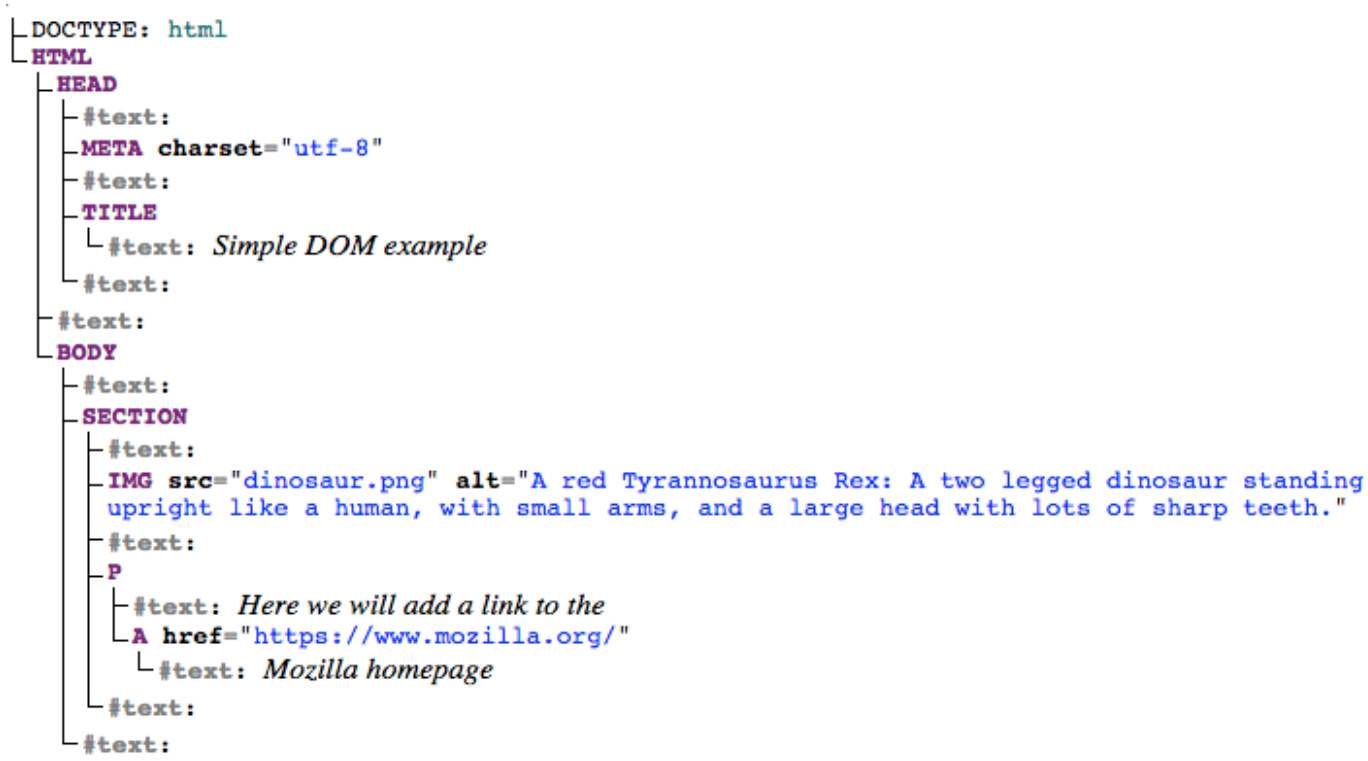
This script puts a function `w(s)` into the global scope of the test page, where *s* is a string or object to output to the log.

▶ Some files are available for testing purposes, notably "image" is an image.

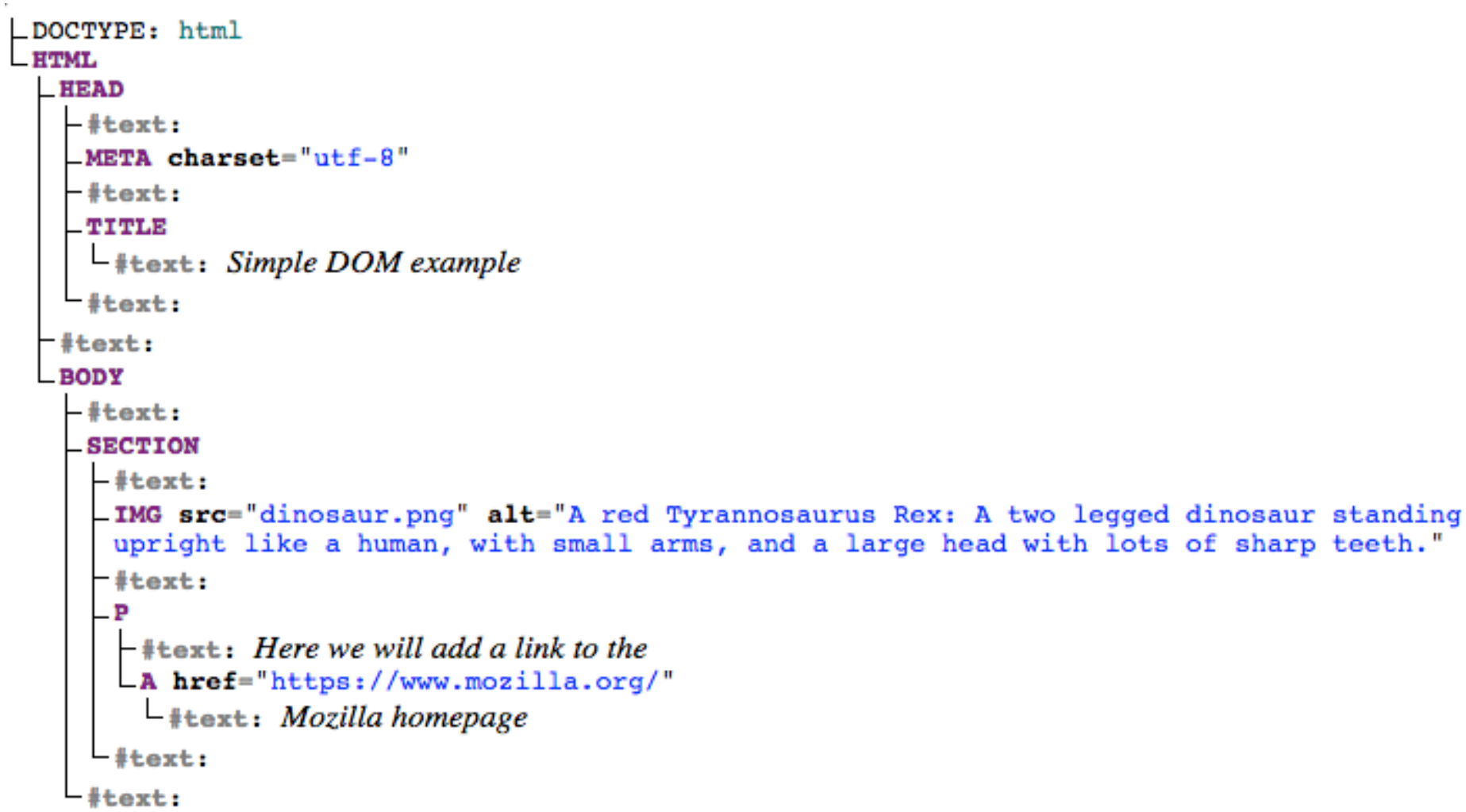
Each entry in the tree is called a node.

The diagram shows that some nodes represent elements (identified as HTML, HEAD, META and so on) and others represent text (identified as #text).

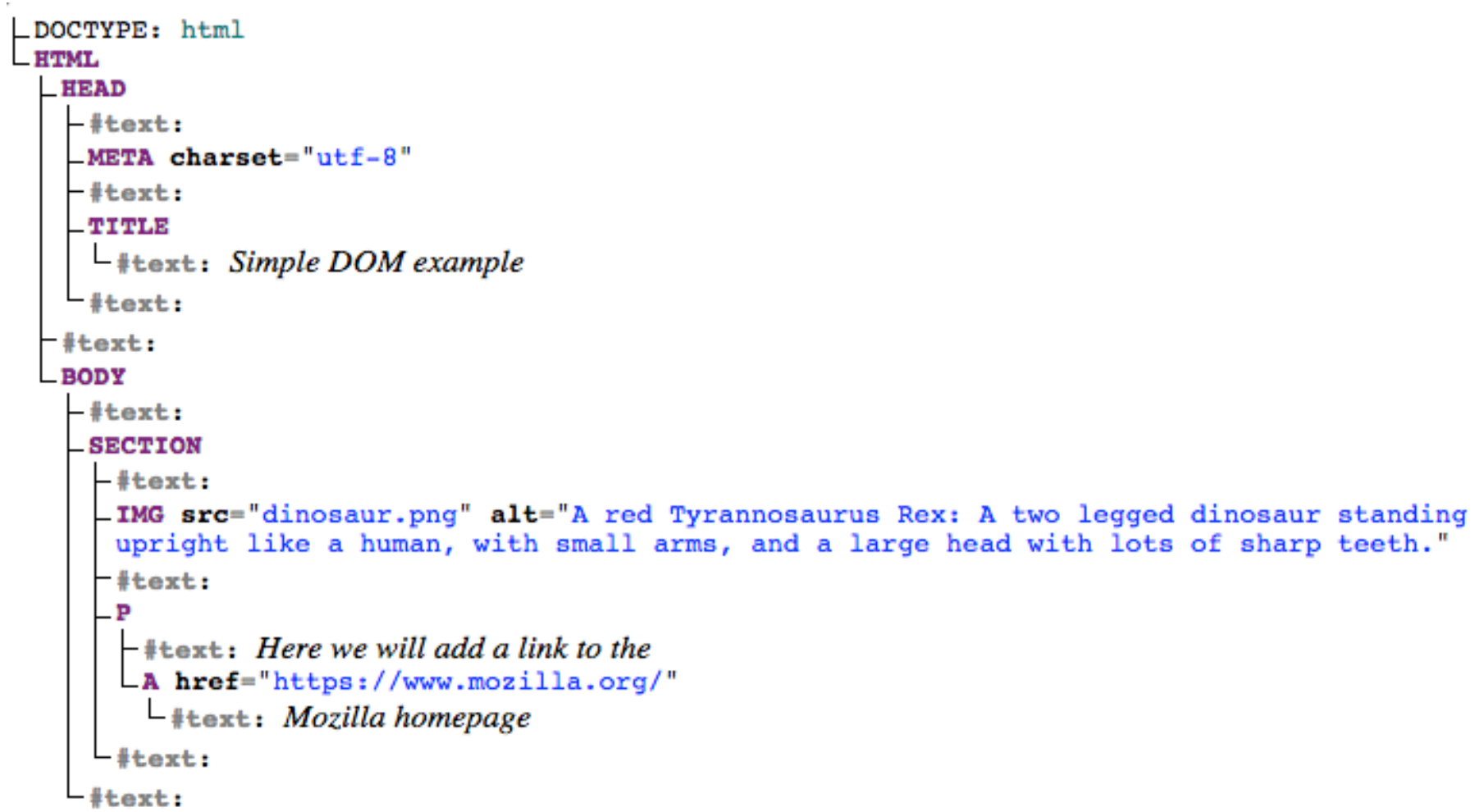
There are other types of nodes as well, but these are the main ones you'll encounter.



- Nodes are also referred to by their position in the tree relative to other nodes:
- **Root node**: The top node in the tree, which in the case of HTML is always the HTML node (other markup vocabularies like SVG and custom XML will have different root elements).
- **Child node**: A node directly inside another node. For example, IMG is a child of SECTION in the example.



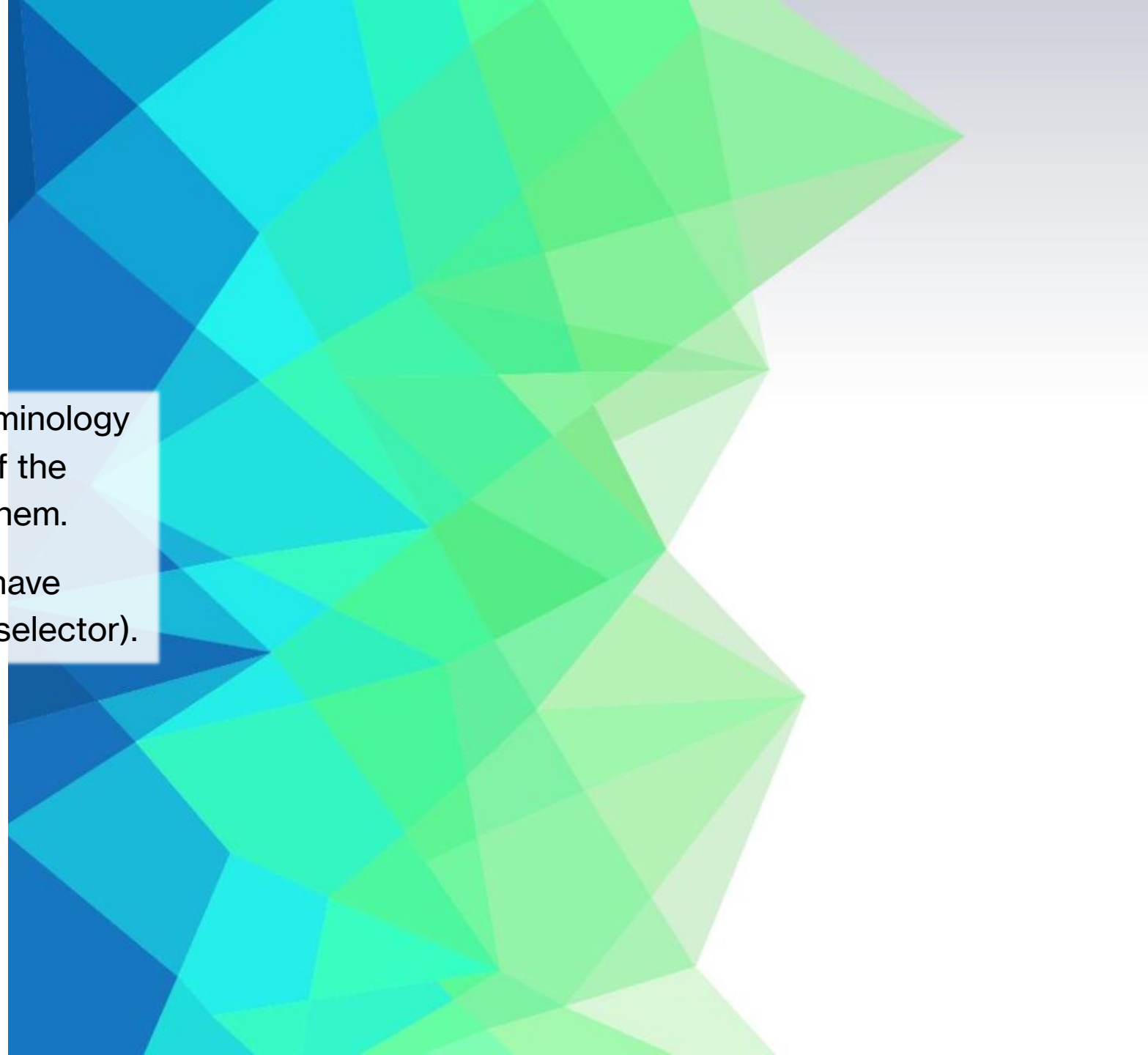
- **Descendant node**: A node anywhere inside another node. For example, IMG is a child of SECTION in the example, and it is also a descendant. IMG is not a child of BODY, as it is two levels below it in the tree, but it is a descendant of BODY.
- **Parent node**: A node which has another node inside it. For example, BODY is the parent node of SECTION in the example.
- **Sibling nodes**: Nodes that sit on the same level in the DOM tree. For example, IMG and P are siblings in the example.



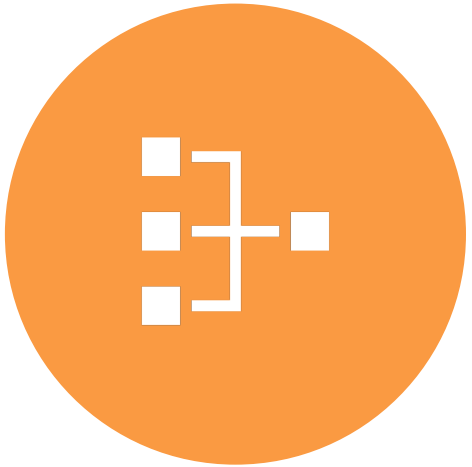
Top Tip

It is useful to familiarize yourself with this terminology before working with the DOM, as a number of the code terms you'll come across make use of them.

You may have also come across them if you have studied CSS (e.g. descendant selector, child selector).



Nodes and Elements



THE DOM **HAS MANY** NODES,
EVERYTHING IS A NODE.

AN ELEMENT **IS A** NODE

Confusingly

- Element ATTRIBUTES are also Nodes but
  - *Attr objects inherit the Node interface, but since they are not actually child nodes of the element they describe, the DOM does not consider them part of the document tree.*
  - *Thus, the Node attributes parentNode, previousSibling, and nextSibling have a null value for Attr objects.*
    - *https://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html#ID-637646024*

But.....

For now, just remember that:

The DOM is made up of Elements

These Elements are Nodes in a Tree

If you want to dig deeper:

- https://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html#ID-637646024
- https://dom.spec.whatwg.org/#introduction-to-the-dom
- https://medium.com/front-end-weekly/what-is-the-difference-between-element-and-node-in-dom-56182fce7af1.
- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction#fundamental_data_types

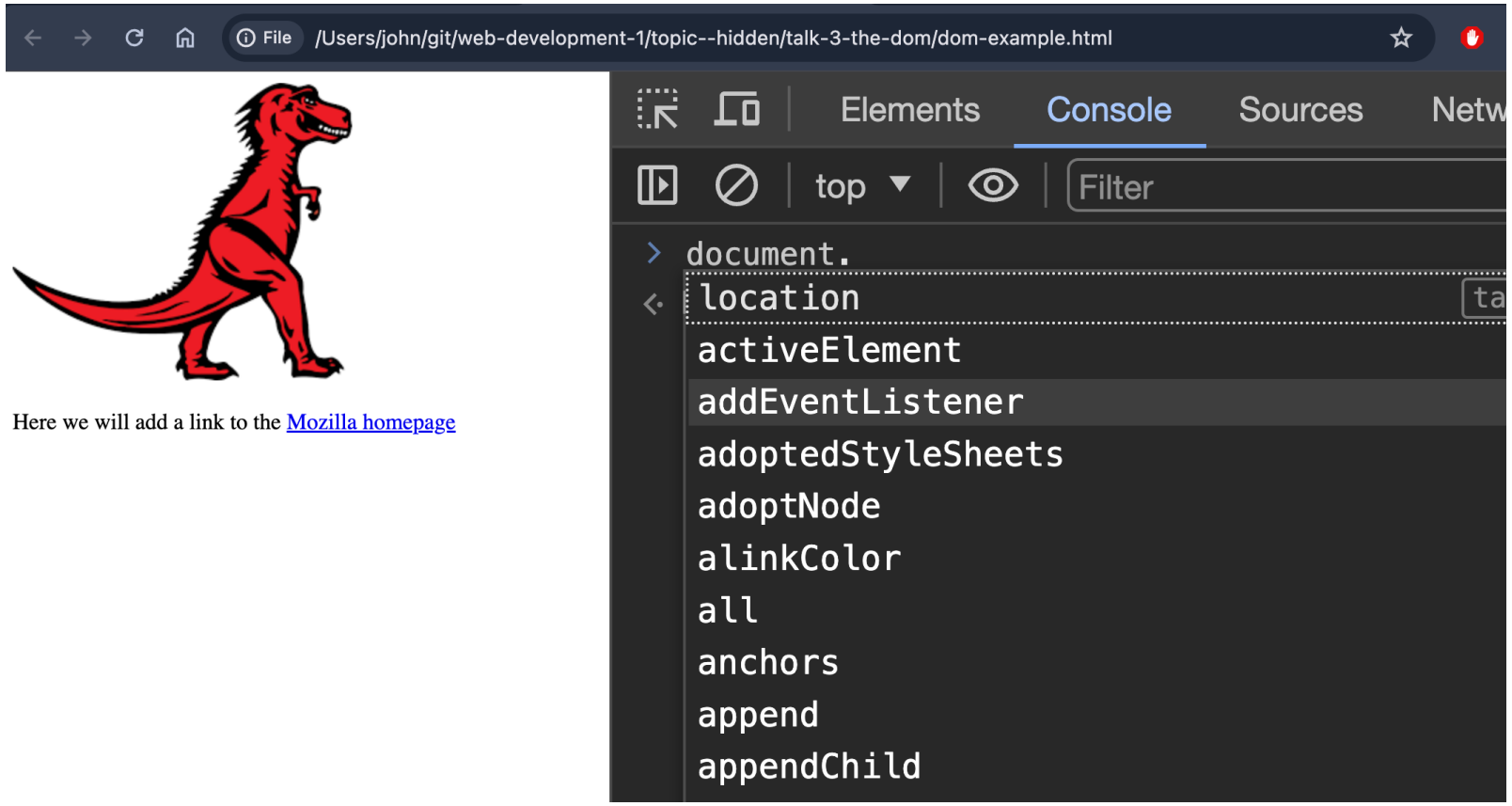
File /Users/john/git/web-development-1/topic--hidden/talk-3-the-dom/dom-example.html

Here we will add a link to the Mozilla homepage

Elements Console Sources

```
<!DOCTYPE html>
<html lang="en-US">
▶ <head> ··· </head>
▼ <body>
  ▼ <section>
      <img src="dinosaur.png" alt="A red T
      nosaur standing upright like a humar
      ad with lots of sharp teeth.">
  ··· ▼ <p> == $0
        "Here we will add a link to the "
        <a href="https://www.mozilla.org/"
      </p>
    </section>
  </body>
</html>
```

html body section p

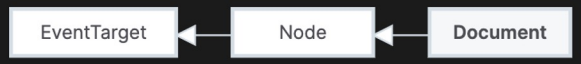


- The DOM is available via the “document” constant
- The “document” constant is available anywhere when running a script in the browser
- It is available in the “Global Scope”

Document

The `Document` interface represents any web page loaded in the browser and serves as an entry point into the web page's content, which is the DOM tree.

The DOM tree includes elements such as `<body>` and `<table>`, among many others. It provides functionality globally to the document, like how to obtain the page's URL and create new elements in the document.



The `Document` interface describes the common properties and methods for any kind of document. Depending on the document's type (e.g. HTML, XML, SVG, ...), a larger API is available: HTML documents, served with the `"text/html"` content type, also implement the `HTMLDocument` interface, whereas XML and SVG documents implement the `XMLDocument` interface.

Constructor

`Document()`

Creates a new `Document` object.

https://developer.mozilla.org/en-US/docs/Web/API/Document

Instance properties

This interface also inherits from the `Node` and `EventTarget` interfaces.

`Document.activeElement` Read only

Returns the `Element` that currently has focus.

`Document.adoptedStyleSheets`

Add an array of constructed stylesheets to be used by the document. These stylesheets may also be shared with shadow DOM subtrees of the same document.

`Document.body`

Returns the `<body>` or `<frameset>` node of the current document.

`Document.characterSet` Read only

Returns the character set being used by the document.

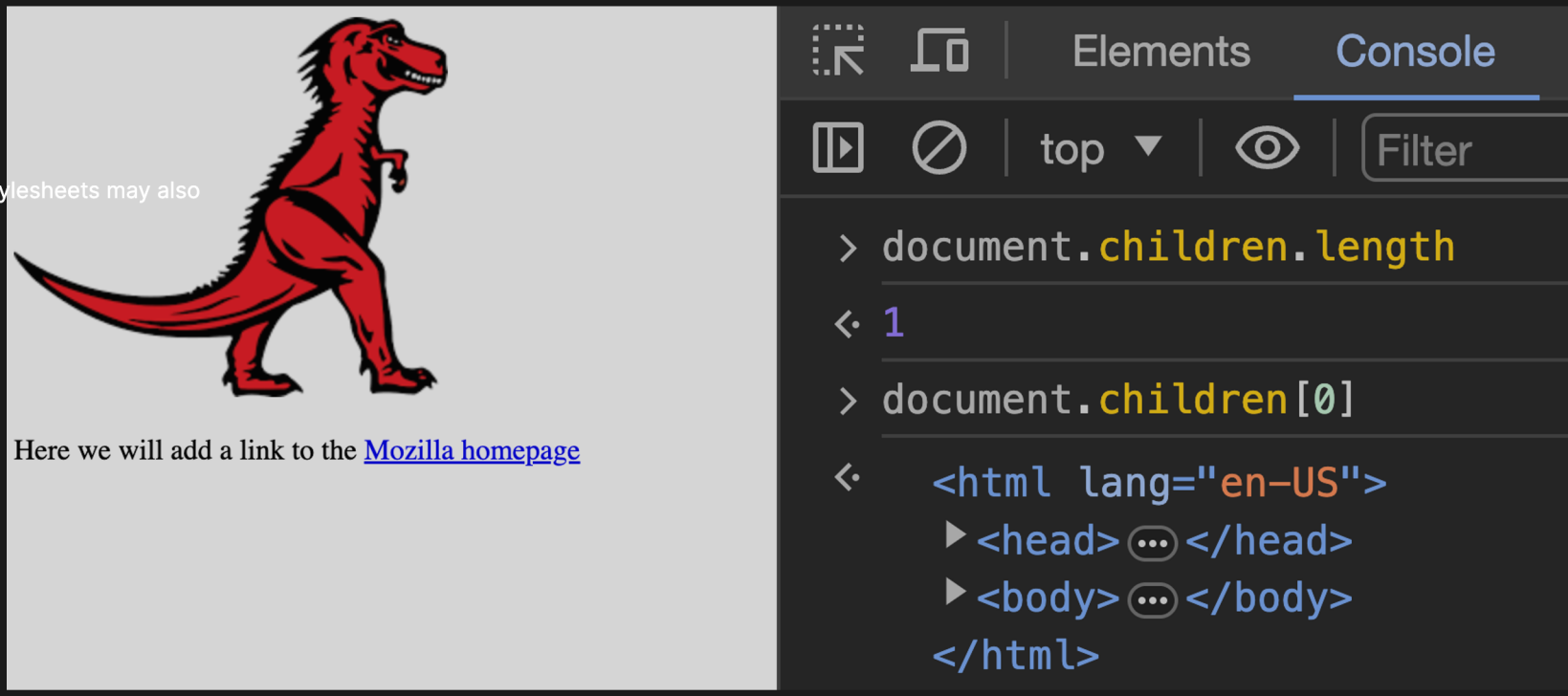
`Document.childElementCount` Read only

Returns the number of child elements of the current document.

`Document.children` Read only

Returns the child elements of the current document.

`Document.compatMode` Read only



https://developer.mozilla.org/en-US/docs/Web/API/Document

Here we will add a link to the Mozilla Developer Network

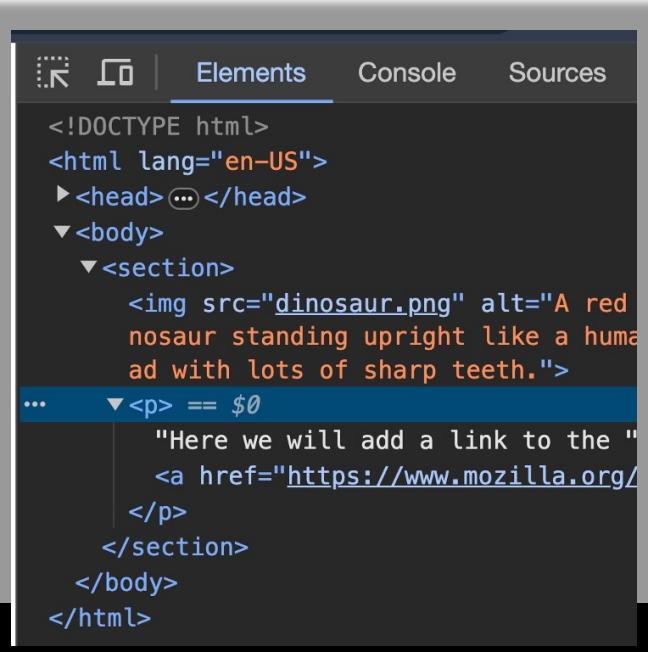
Elements Console Sources Network

top Filter Default levels

```
> const link = document.querySelector("a");
< undefined
> link
<   <a href="https://www.mozilla.org/">Mozilla homepage</a>
> link.textContent = "Mozilla Developer Network";
< 'Mozilla Developer Network'
> link
<   <a href="https://www.mozilla.org/">Mozilla Developer Netwo
> link.href = "https://developer.mozilla.org";
< 'https://developer.mozilla.org'
> link
<   <a href="https://developer.mozilla.org">Mozilla Developer N
>
```

Here we will add a link to the Mozilla homepage

We hope you enjoyed.



Elements Console Sources Network

top Filter Default leve

```
> const section = document.querySelector("section");
< undefined
> section
< ▼<section>
    <img src="dinosaur.png" alt="A red Tyrannosaurus Re
    nosaur standing upright like a human, with small ar
    ad with lots of sharp teeth.">
  ▶<p> … </p>
  </section>
> const paragraph = document.createElement("p");
  paragraph.textContent = "We hope you enjoyed.";
< 'We hope you enjoyed.'
> section.appendChild(paragraph)
<   <p>We hope you enjoyed.</p>
```

Element

p 414 × 18.5

Here we will add a link to the Mozilla homepage

We hope you enjoyed.

Elements | Console | Sources

top | Filter

```
> paragraph
<p>We hope you enjoyed.</p>
>
```

developer.mozilla.org/en-US/docs/Web/API/Element

Get real-time assistance with your coding queries. Try AI Help now!

mdn web docs | References | Guides | Plus | Blog | Play | AI Help BETA | Theme

References > Web APIs > Element

Filter

Document Object Model

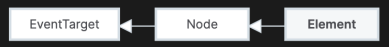
- Element
- ▼ Instance properties
  - ariaAtomic
  - ariaAutoComplete
  - ariaBusy
  - ariaChecked
  - ariaColCount
  - ariaColIndex
  - ariaColIndexText
  - ariaColSpan
  - ariaCurrent
  - ariaDescription
  - ariaDisabled
  - ariaExpanded
  - ariaHasPopup
  - ariaHidden

Element

`Element` is the most general base class from which all element objects (i.e. objects that represent elements) in a `Document` inherit. It only has methods and properties common to all kinds of elements. More specific classes inherit from `Element`.

For example, the `HTMLElement` interface is the base interface for HTML elements. Similarly, the `SVGElement` interface is the basis for all SVG elements, and the `MathMLElement` interface is the base interface for MathML elements. Most functionality is specified further down the class hierarchy.

Languages outside the realm of the Web platform, like XUL through the `XULElement` interface, also implement `Element`.



Instance properties

`Element` inherits properties from its parent interface, `Node`, and by extension that interface's parent, `EventTarget`.

`Element.assignedSlot` Read only

Instance properties

`Element` inherits properties from its parent interface, `Node`, and by extension that interface's parent, `EventTarget`.

`Element.assignedSlot` Read only

Returns a `HTMLSlotElement` representing the `<slot>` the node is inserted in.

`Element.attributes` Read only

Returns a `NamedNodeMap` object containing the assigned attributes of the corresponding HTML element.

`Element.childElementCount` Read only

Returns the number of child elements of this element.

`Element.children` Read only

Returns the child elements of this element.

mdn web docs References Guides Plus Blog Play AI Help BETA Theme

References > Web APIs > Element > remove()

Filter

- prepend()
- querySelector()
- querySelectorAll()
- releasePointerCapture()
- remove()
- removeAttribute()
- removeAttributeNode()
- removeAttributeNS()
- replaceChildren()
- replaceWith()
- requestFullscreen()
- requestPointerLock()
- scroll()

Element: remove() method

The `Element.remove()` method removes the element from the DOM.

Syntax

JS

```
remove()
```

Parameters

None.

Return value

None (`undefined`).

Here we will add a link to the Mozilla homepage

We hope you enjoyed.

Elements Console Sources Network

top Filter Default

```
> paragraph
    <p>We hope you enjoyed.</p>
> paragraph.rem
ƒ
```

remove tab
removeAttribute
removeAttributeNode
removeAttributeNS
removeChild
removeEventListener

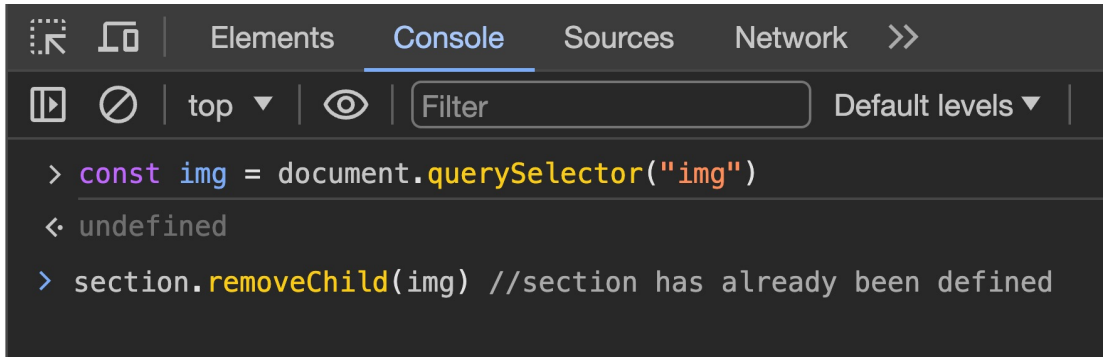
Here we will add a link to the Mozilla homepage

Elements Console Sources

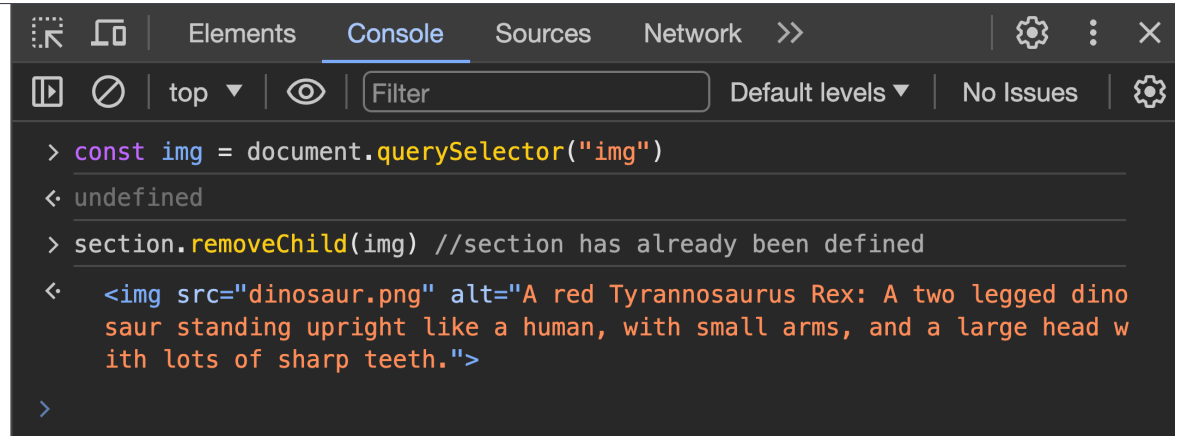
top Filter

```
> paragraph
    <p>We hope you enjoyed.</p>
> paragraph.remove()
undefined
```

Here we will add a link to the Mozilla homepage

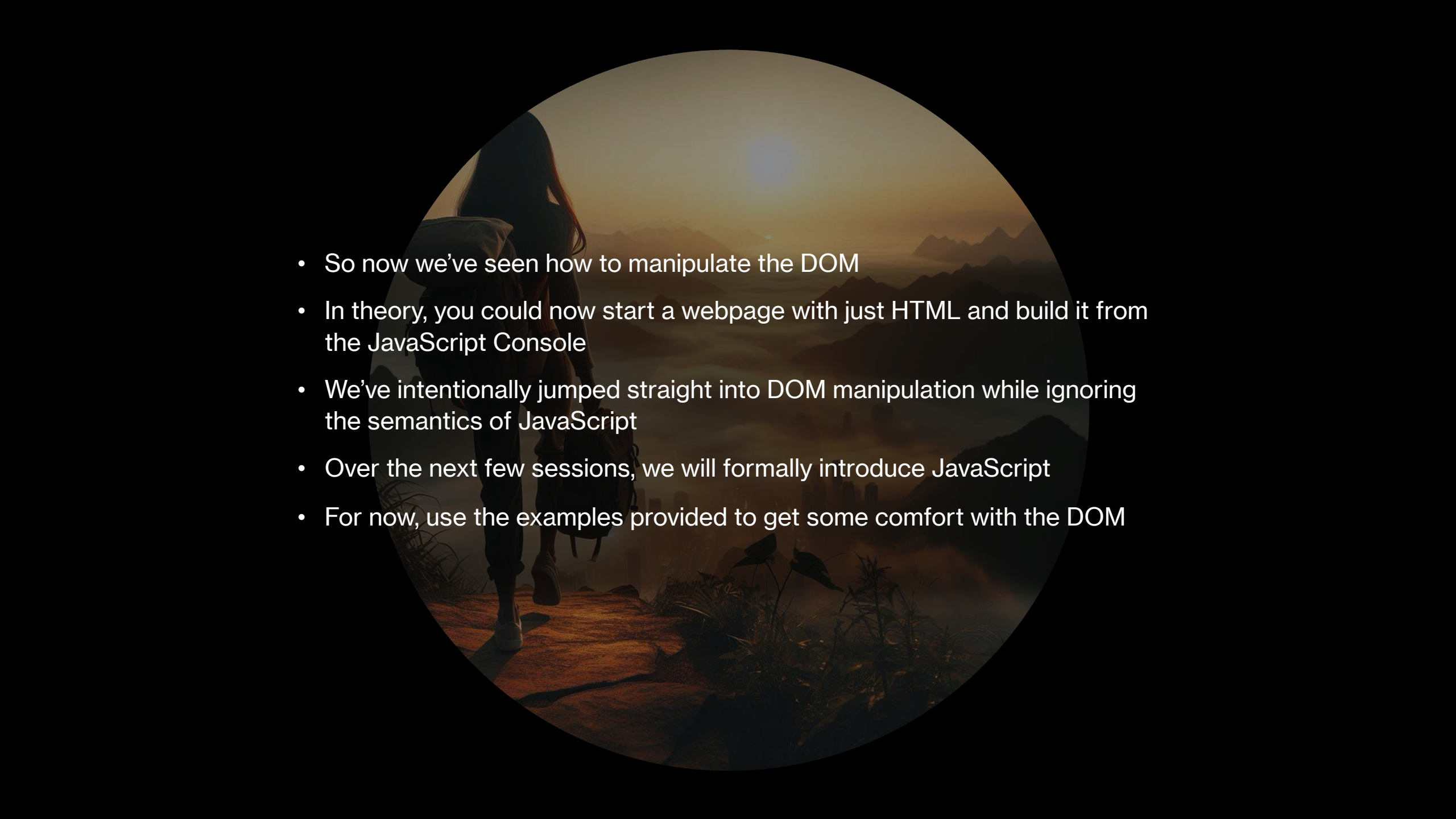


Here we will add a link to the Mozilla homepage



Recap

- We used the console to interact with the “document”
- We used “querySelector”, “querySelectorAll” and “getElementById”
- Augmented a link element via “link.href = ...”
  - *document.querySelector(“a”).href = “http://google.ie”*
- Used “document.createElement” to create an element
- Used ”element.appendChild” to add an element to a section
- Used “element.removeChild” to remove an img
  - Could also use “element.remove()” for the element to remove itself



- So now we've seen how to manipulate the DOM
- In theory, you could now start a webpage with just HTML and build it from the JavaScript Console
- We've intentionally jumped straight into DOM manipulation while ignoring the semantics of JavaScript
- Over the next few sessions, we will formally introduce JavaScript
- For now, use the examples provided to get some comfort with the DOM