

Get better WordPress performance with Cloudways managed hosting. Start with \$100, free →

We're
hiring

Blog

Docs

Get
Support

Contact
Sales



Tutorials

Questions

Learning Paths

For Businesses

For Builders

Social Impact



CONTENTS

Setup

Root Nodes

Parent Nodes

Children Nodes

Sibling Nodes

Conclusion

RELATED

CodeIgniter: Getting Started With a Simple Example

[View](#)

How To Install Express, a Node.js Framework, and Set Up Socket.io on a VPS

[View](#)

Tutorial Series: Understanding the DOM — Document Object Model



This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

MANAGE CHOICES

AGREE & PROCEED

// Tutorial //

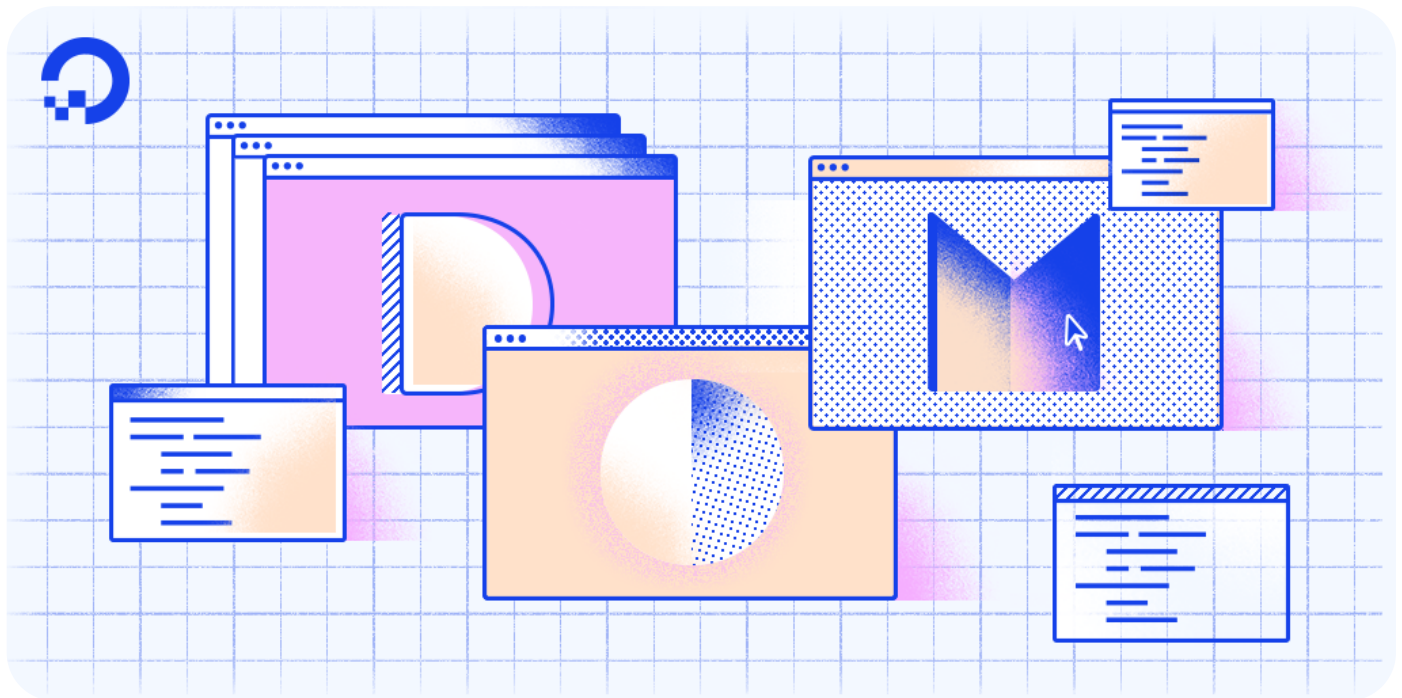
How To Traverse the DOM

Published on December 4, 2017

JavaScript Development



By [Tania Rascia](#)



Introduction

The previous tutorial in this series, [How to Access Elements in the DOM](#), covers how to use the built-in methods of the `document` object to access HTML elements by ID, class, tag name, and query selectors. We know that the DOM is structured as a [tree of nodes](#) with the `document` node at the root and every other node (including elements, comments, and text nodes) as the various branches.

This site uses cookies and related technologies, as described in our [privacy policy](#), for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

In this tutorial, we will go over how to traverse the DOM (also known as walking or navigating the DOM) with parent, child, and sibling properties.

Setup

To begin, we will create a new file called `nodes.html` comprised of the following code.

nodes.html

Copy

```
<!DOCTYPE html>
<html>

<head>
  <title>Learning About Nodes</title>

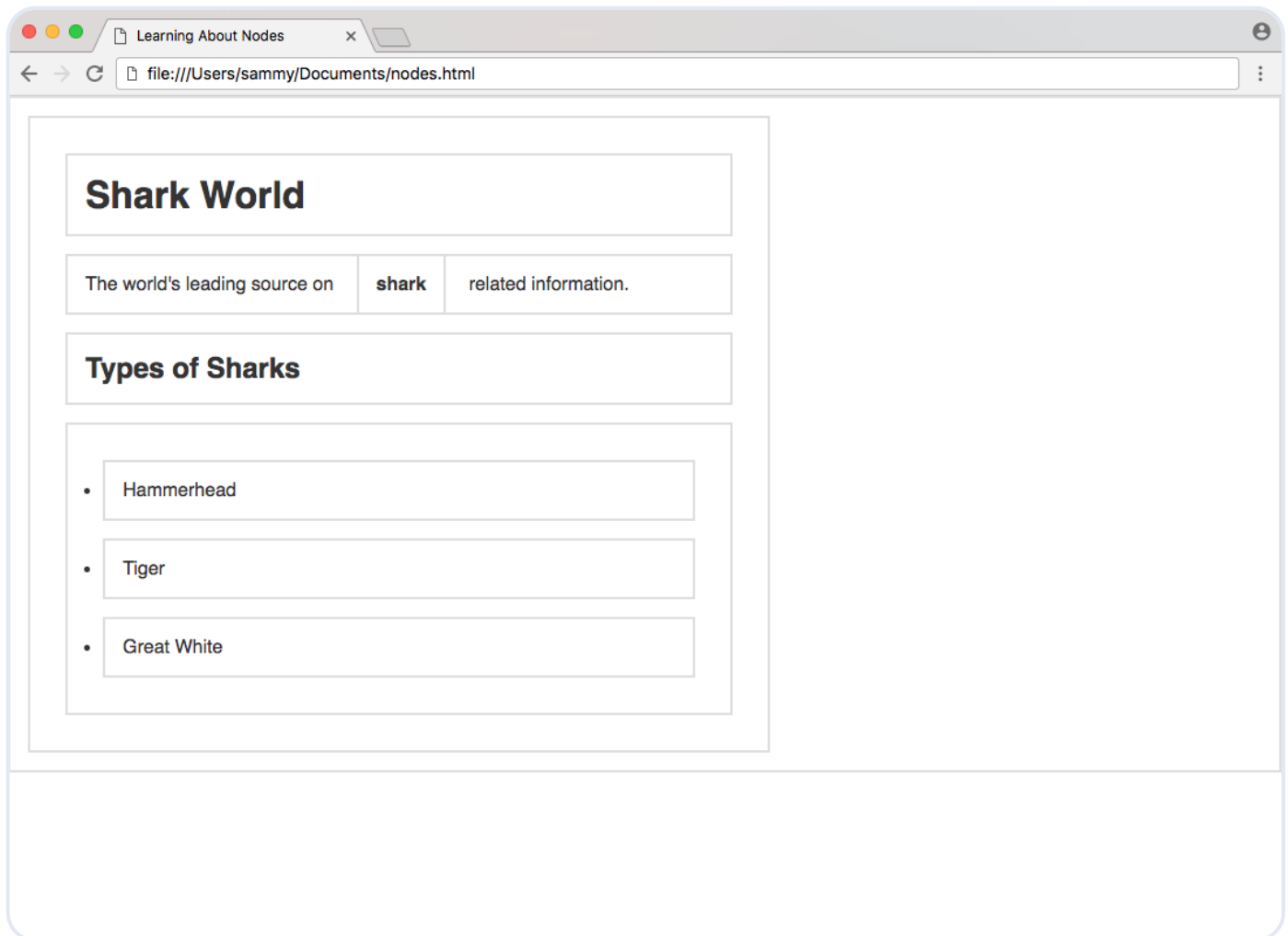
  <style>
    * { border: 2px solid #dedede; padding: 15px; margin: 15px; }
    html { margin: 0; padding: 0; }
    body { max-width: 600px; font-family: sans-serif; color: #333; }
  </style>
</head>

<body>
  <h1>Shark World</h1>
  <p>The world's leading source on <strong>shark</strong> related information.</p>
  <h2>Types of Sharks</h2>
  <ul>
    <li>Hammerhead</li>
    <li>Tiger</li>
    <li>Great White</li>
  </ul>
</body>

<script>
  const h1 = document.getElementsByTagName('h1')[0];
  const p = document.getElementsByTagName('p')[0];
  const ul = document.getElementsByTagName('ul')[0];
</script>

</html>
```

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.



In this example website, we have an HTML document with a few elements. Some basic CSS has been added in a `style` tag to make each element obviously visible, and a few variables have been created in the `script` for ease of access of a few elements. Since there is only one of each `h1`, `p`, and `ul`, we can access the first index on each respective `getElementsByTagName` property.

Root Nodes

The `document` object is the root of every node in the DOM. This object is actually a property of the `window` object, which is the global, top-level object representing a tab in the browser. The `window` object has access to such information as the toolbar, height and width of the window, prompts, and alerts. The `document` consists of what is inside of the `innerWindow`.

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

Property	Node	Node Type
<code>document</code>	<code>#document</code>	<code>DOCUMENT_NODE</code>
<code>document.documentElement</code>	<code>html</code>	<code>ELEMENT_NODE</code>
<code>document.head</code>	<code>head</code>	<code>ELEMENT_NODE</code>
<code>document.body</code>	<code>body</code>	<code>ELEMENT_NODE</code>

Since the `html`, `head`, and `body` elements are so common, they have their own properties on the `document`.

Open the *Console* in DevTools and test each of these four properties by submitting them and viewing the output. You can also test `h1`, `p`, and `ul` which will return the elements due to the variables we added in the `script` tag.

Parent Nodes

The nodes in the DOM are referred to as parents, children, and siblings, depending on their relation to other nodes. The **parent** of any node is the node that is one level above it, or closer to the `document` in the DOM hierarchy. There are two properties to get the parent — `parentNode` and `parentElement`.

Property	Gets
<u><code>parentNode</code></u>	Parent Node
<code>parentElement</code>	Parent Element Node

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

- `body` is the parent of `h1`, `h2`, `p` and `u1`, but not `li`, since `li` is two levels down from `body`.

We can test what the parent of our `p` element is with the `parentNode` property. This `p` variable comes from our custom `document.getElementsByTagName('p')[0]` declaration.

```
> p.parentNode;
```

[Copy](#)

Output

```
► <body>...</body>
```

The parent of `p` is `body`, but how can we get the grandparent, which is two levels above? We can do so by chaining properties together.

```
> p.parentNode.parentNode;
```

[Copy](#)

Output

```
► <html>...</html>
```

Using `parentNode` twice, we retrieved the grandparent of `p`.

There are properties to retrieve the parent of a node, but only one small difference between them, as demonstrated in this snippet below.

```
> // Assign html object to html variable
> const html = document.documentElement;
>
> console.log(html.parentNode); // > #document
> console.log(html.parentElement); // > null
```

[Copy](#)

The parent of almost any node is an element node, as text and comments cannot be parents to other nodes. However, the parent of `html` is a document node, so `parentElement` returns `null`. Generally, `parentNode` is more commonly used when traversing the DOM.

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

Property**Gets**`childNodes`

Child Nodes

`firstChild`

First Child Node

`lastChild`

Last Child Node

`children`

Element Child Nodes

`firstElementChild`

First Child Element Node

`lastElementChild`

Last Child Element Node

The `childNodes` property will return a live list of every child of a node. You might expect the `ul` element to get three `li` elements. Let's test what it retrieves.

```
> ul.childNodes;
```

Copy

Output

```
► (7) [text, li, text, li, text, li, text]
```

In addition to the three `li` elements, it also gets four text nodes. This is because we wrote our own HTML (it was not generated by JavaScript) and the indentation between elements is counted in the DOM as text nodes. This is not intuitive, as the *Elements* tab of DevTools strips out white space nodes.

If we attempted to change the background color of the first child node using the

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

[Copy](#)

>

Output

```
Uncaught TypeError: Cannot set property 'background' of undefined
```

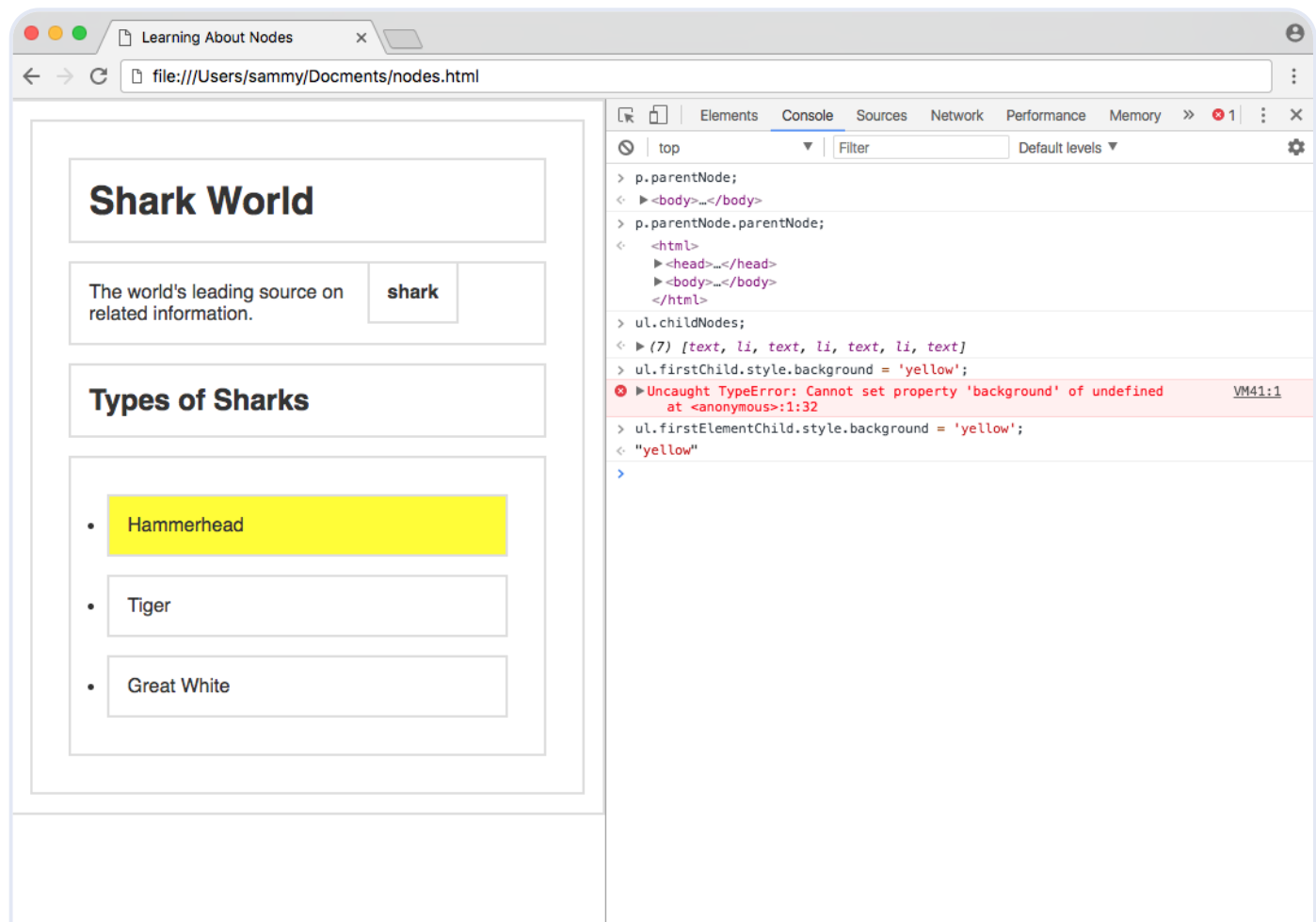
The `children`, `firstElementChild` and `lastElementChild` properties exist in these types of situations to retrieve only the element nodes. `ul.children` will only return the three `li` elements.

Using `firstElementChild`, we can change the background color of the first `li` in the `ul`.

```
> ul.firstElementChild.style.background = 'yellow';
```

[Copy](#)

When you run the code above, your webpage will be updated to modify the background color.



This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

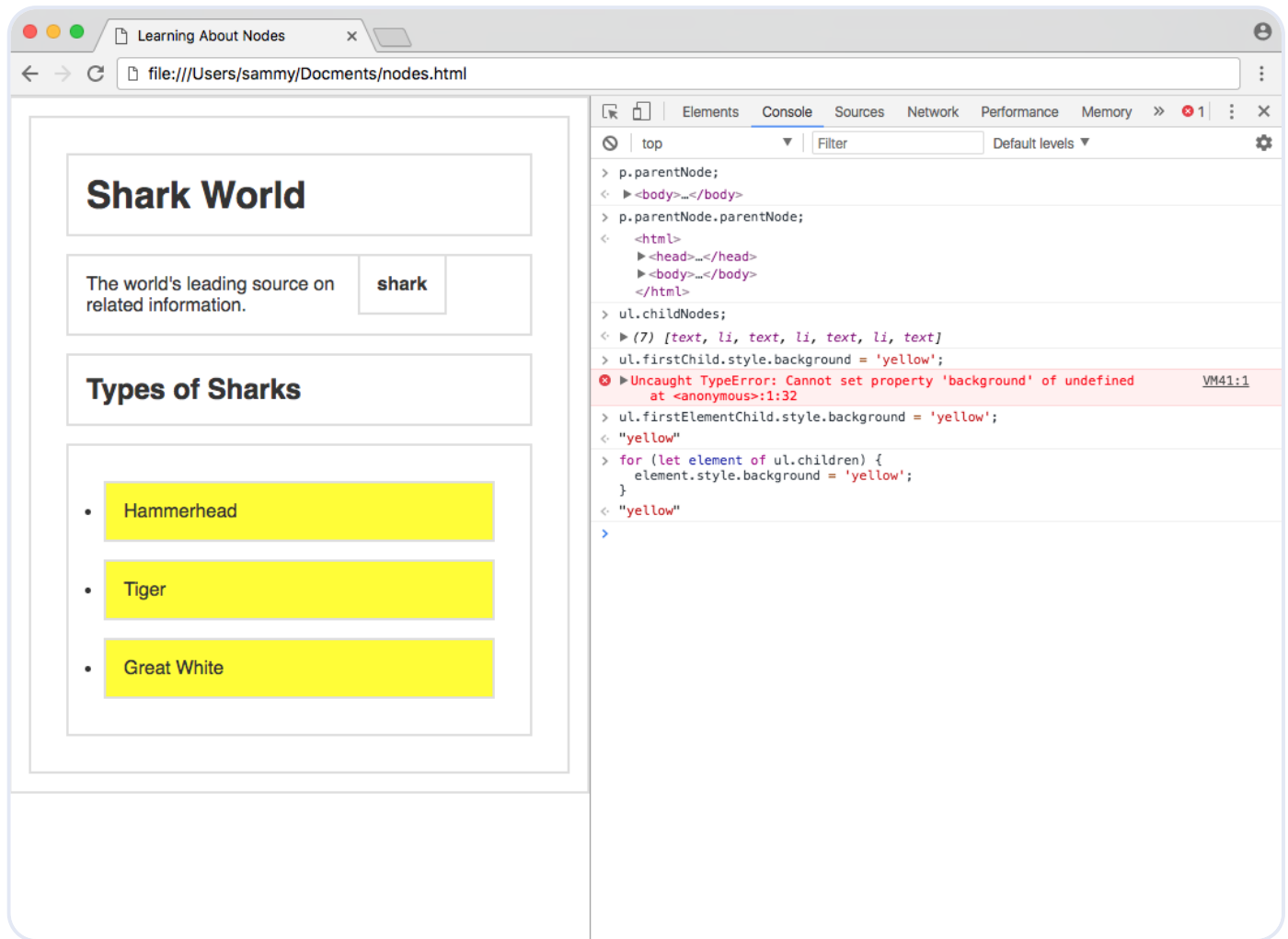
select all nodes are more likely to be used, as white-space newlines and indentation will not exist in this case.

A `for...of` loop can be used to iterate through all `children` elements.

```
> for (let element of ul.children) {  
>   element.style.background = 'yellow';  
> }
```

[Copy](#)

Now, each child element will have a yellow background.



Since our `p` element has both text and elements inside of it, the `childNodes` property is helpful for accessing that information.

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

```
"The world's leading source on "
<strong>shark</strong>
" related information."
```

`childNodes` and `children` do not return arrays with all the [Array properties and methods](#), but they appear and behave similarly to JavaScript arrays. You can access nodes by index number, or find their `length` property.

```
> document.body.children[3].lastElementChild.style.background = 'fuchsia' Copy
```

The above code will find the last element child (`li`) of the fourth child element (`ul`) of `body` and apply a style.

The screenshot shows a web browser window with the title "Learning About Nodes" and the address bar showing a local file path. The page content includes a header "Shark World", a paragraph "The world's leading source on related information." with a "shark" button, and a section "Types of Sharks" with a list of three items: "Hammerhead", "Tiger", and "Great White". The developer console on the right shows the following code and output:

```
> p.parentNode;
< > <body>...</body>
> p.parentNode.parentNode;
< <html>
  > <head>...</head>
  > <body>...</body>
  </html>
> ul.childNodes;
< > (7) [text, li, text, li, text, li, text]
> ul.firstChild.style.background = 'yellow';
< > Uncaught TypeError: Cannot set property 'background' of undefined VM41:1
> ul.firstChild.style.background = 'yellow';
< "yellow"
> for (let element of ul.children) {
  element.style.background = 'yellow';
}
< "yellow"
> for (let element of p.childNodes) {
  console.log(element);
}
< "The world's leading source on " VM47:2
  <strong>shark</strong> VM47:2
  " related information." VM47:2
< undefined
> document.body.children[3].lastElementChild.style.background = 'fuchsia';
< "fuchsia"
> |
```

Using parent and child properties, you can retrieve any node in the DOM.

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

The **siblings** of a node are any node on the same tree level in the DOM. Siblings do not have to be the same type of node - text, element, and comment nodes can all be siblings.

Property	Gets
<u>previousSibling</u>	Previous Sibling Node
<u>nextSibling</u>	Next Sibling Node
<u>previousElementSibling</u>	Previous Sibling Element Node
<u>nextElementSibling</u>	Next Sibling Element Node

Sibling properties work the same way as the children nodes, in that there is a set of properties to traverse all nodes, and a set of properties for only element nodes.

`previousSibling` and `nextSibling` will get the next node that immediately precedes or follows the specified node, and `previousElementSibling` and `nextElementSibling` will only get element nodes.

In our `nodes.html` example, let's select the middle element of `ul`.

```
> const tiger = ul.children[1];
```

[Copy](#)

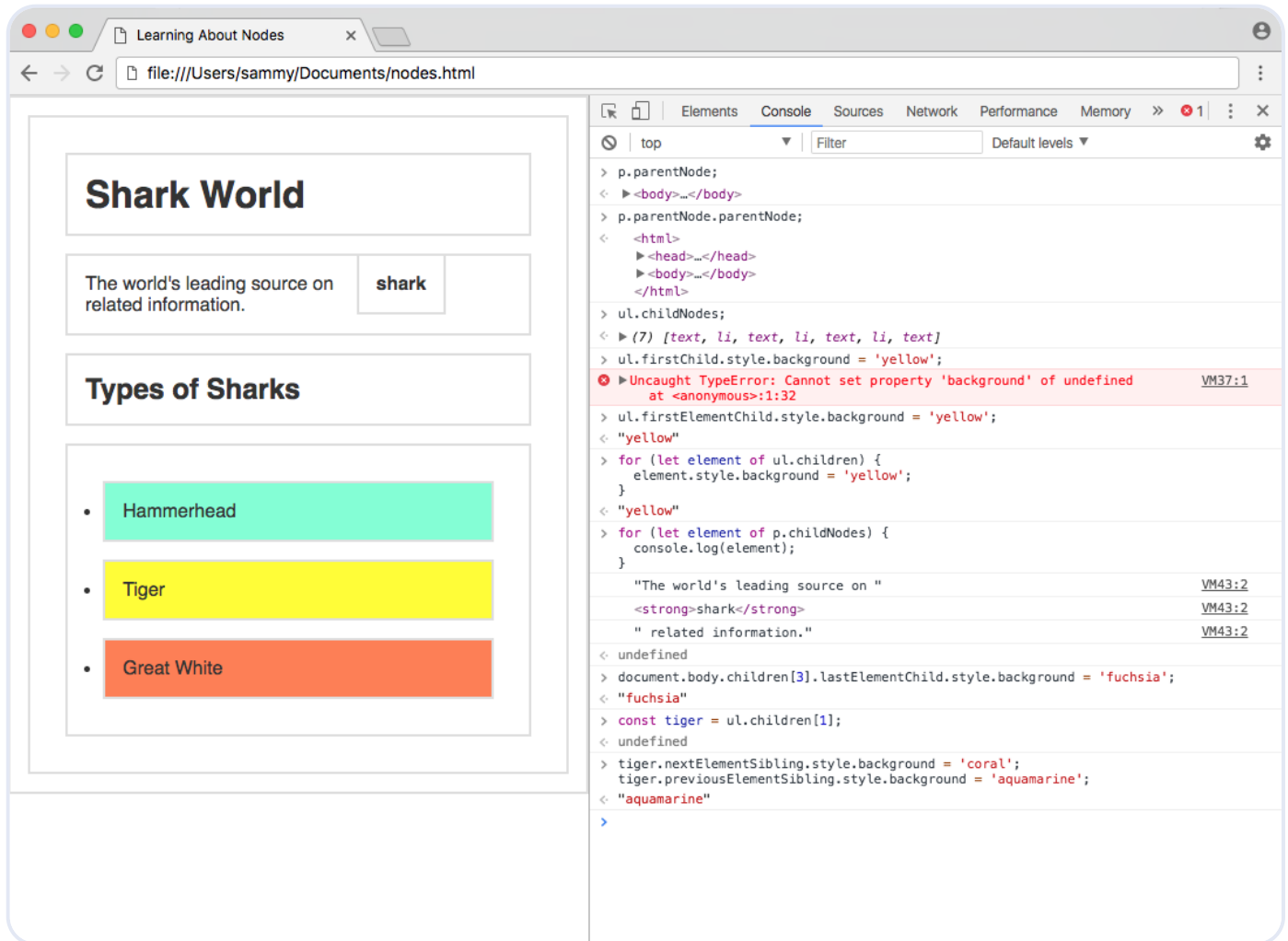
Since we created our DOM from scratch and not as a JavaScript web app, we will need to use the element sibling properties to access the previous and next element nodes, as there is white space in the DOM.

```
> tiger.nextElementSibling.style.background = 'coral';  
> tiger.previousElementSibling.style.background = 'aquamarine';
```

[Copy](#)

Running this code should have applied `coral` to the background of `Hammerhead` and

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.



Sibling properties can be chained together, just like parent and node properties.

Conclusion

In this tutorial, we covered how to access the root nodes of every HTML document and how to walk the DOM tree through parent, child, and sibling properties.

With what you learned in [How to Access Elements in the DOM](#) and this tutorial, you should be able to confidently access any node in the DOM of any website.

Thanks for learning with the DigitalOcean Community. Check out our offerings for

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

Next in series: [How To Make Changes to the DOM](#) →

Want to learn more? Join the DigitalOcean Community!

Join our DigitalOcean community of over a million developers for free! Get help and share knowledge in our Questions & Answers section, find tutorials and tools that will help you grow as a developer and scale your project or business, and subscribe to topics of interest.

[Sign up now](#) →

Tutorial Series: Understanding the DOM – Document Object Model

The Document Object Model, usually referred to as the DOM, is an essential part of making websites interactive. It is an interface that allows a programming language to manipulate the content, structure, and style of a website. JavaScript is the client-side scripting language that connects to the DOM in an internet browser.

Subscribe

JavaScript Development

Browse Series: 8 articles

[1/8 Introduction to the DOM](#)

[2/8 Understanding the DOM Tree and Nodes](#)

[3/8 How To Access Elements in the DOM](#)

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

About the authors



[Tania Rascia](#) Author



[Lisa Tagliaferri](#) Editor

Still looking for an answer?

Ask a question

Search for more help

Was this helpful?

Yes

No



Comments

1 Comments

B *I* U



Leave a comment

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

You can type `!ref` in this text area to quickly search our full set of tutorials, documentation & marketplace offerings and insert the link!

[Sign In](#) or [Sign Up](#) to Comment

[Sivaraj V](#) • February 24, 2019



Running this code should have applied coral to the background of Hammerhead and aquamarine to the background of Great White.

I think you guys has written wrongly in color.

Example image was correct.

[Reply](#)



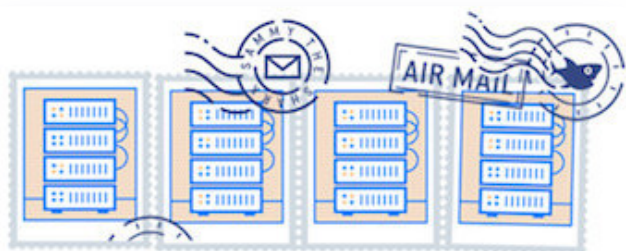
This work is licensed under a Creative Commons Attribution-NonCommercial- ShareAlike 4.0 International License.

Try DigitalOcean for free

Click below to sign up and get **\$200 of credit** to try our products over 60 days!

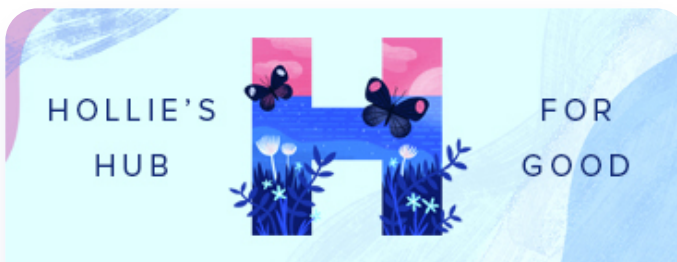
Sign up →

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

[JavaScript](#)[Python](#)[MySQL](#)[Docker](#)[Kubernetes](#)[All tutorials →](#)[Free Managed Hosting →](#)

Get our biweekly newsletter

Sign up for Infrastructure as a Newsletter.

[Sign up →](#)

Hollie's Hub for Good

Working on improving health and education, reducing inequality, and spurring economic growth? We'd like to help.

[Learn more →](#)

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

Become a contributor

You get paid; we donate to tech nonprofits.

[Learn more](#) →

Featured on Community

[Kubernetes Course](#) [Learn Python 3](#) [Machine Learning in Python](#)
[Getting started with Go](#) [Intro to Kubernetes](#)

DigitalOcean Products

[Cloudways](#) [Virtual Machines](#) [Managed Databases](#) [Managed Kubernetes](#)
[Block Storage](#) [Object Storage](#) [Marketplace](#) [VPC](#) [Load Balancers](#)

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the cloud and scale up as you grow – whether you're running one virtual machine or ten thousand.

[Learn more](#) →

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

Company	Products	Community	Solutions	Contact
About	Products	Tutorials	Website Hosting	Support
Leadership	Overview	Q&A	VPS Hosting	Sales
Blog	Droplets	CSS-Tricks	Web & Mobile	Report Abuse
Careers	Kubernetes	Write for	Apps	System Status
Customers	App Platform	DO donations	Game Development	Share your ideas
Partners	Functions	Currents	Streaming	
Channel Partners	Cloudways	Research	VPN	
Referral Program	Managed Databases	Hatch Startup Program	SaaS Platforms	
Affiliate Program	Spaces	deploy by DigitalOcean	Cloud Hosting for Blockchain	
Press	Marketplace	Shop Swag	Startup Resources	
Legal	Load Balancers	Research Program		
Security	Block Storage	Open Source		
Investor Relations	Tools & Integrations	Code of Conduct		
DO Impact	API	Newsletter		
	Pricing	Signup		
	Documentation	Meetups		
	Release Notes			
	Uptime			

© 2023 DigitalOcean, LLC. All rights reserved

This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.