

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

Creating New Nodes

Inserting Nodes into the DOM

Removing Nodes from the DOM

Conclusion

RELATED

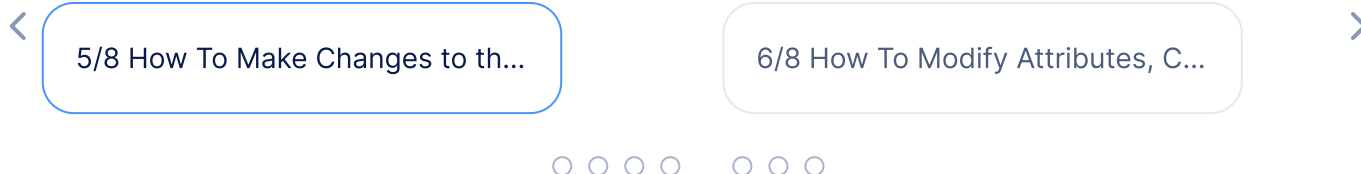
Codelgniter: Getting Started With a Simple Example

[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



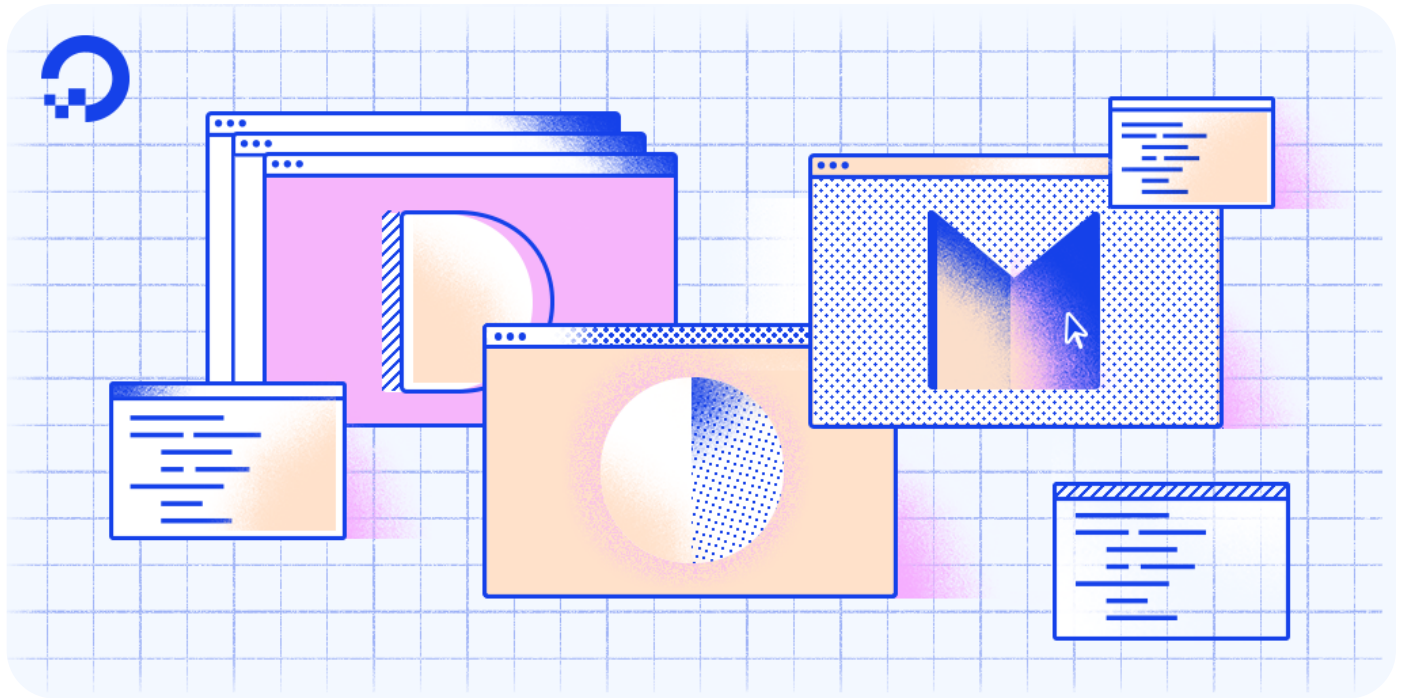
This site uses cookies and related technologies, as described in our privacy policy, for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences.

MANAGE CHOICES

AGREE & PROCEED

Published on December 26, 2017

JavaScript Development

By [Tania Rascia](#)

Introduction

In the previous two installments of the [Understanding the DOM](#) series, we learned [How To Access Elements in the DOM](#) and [How To Traverse the DOM](#). Using this knowledge, a developer can use classes, tags, ids, and selectors to find any node in the DOM, and use parent, child, and sibling properties to find relative nodes.

The next step to becoming more fully proficient with the DOM is to learn how to add, change, replace, and remove nodes. A to-do list application is one practical example of a JavaScript program in which you would need to be able to create, modify, and remove 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.

In a static website, elements are added to the page by directly writing HTML in an `.html` file. In a dynamic web app, elements and text are often added with JavaScript. The `createElement()` and `createTextNode()` methods are used to create new nodes in the DOM.

Property/Method	Description
<u><code>createElement()</code></u>	Create a new element node
<u><code>createTextNode()</code></u>	Create a new text node
<u><code>node.textContent</code></u>	Get or set the text content of an element node
<u><code>node.innerHTML</code></u>	Get or set the HTML content of an element

To begin, let's create an `index.html` file and save it in a new project directory.

index.html

```
<!DOCTYPE html>
<html lang="en">

  <head>
    <title>Learning the DOM</title>
  </head>

  <body>
    <h1>Document Object Model</h1>
  </body>

</html>
```

Copy

Right click anywhere on the page and select "Inspect" to open up Developer Tools, then

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.

We've created a new `p` element, which we can test out in the *Console*.

```
> console.log(paragraph)
```

[Copy](#)

Output

```
<p></p>
```

The `paragraph` variable outputs an empty `p` element, which is not very useful without any text. In order to add text to the element, we'll set the `textContent` property.

```
> paragraph.textContent = "I'm a brand new paragraph.";
> console.log(paragraph)
```

[Copy](#)

Output

```
<p>I'm a brand new paragraph.</p>
```

A combination of `createElement()` and `textContent` creates a complete element node.

An alternate method of setting the content of the element is with the `innerHTML` property, which allows you to add HTML as well as text to an element.

```
> paragraph.innerHTML = "I'm a paragraph with <strong>bold</strong> text. Copy
```

Note: While this will work and is a common method of adding content to an element, there is a possible [cross-site scripting \(XSS\)](#) risk associated with using the `innerHTML` method, as inline JavaScript can be added to an element. Therefore, it is recommended to use `textContent` instead, which will strip out HTML tags.

It is also possible to create a text node with the `createTextNode()` method.

```
> const text = document.createTextNode("I'm a new text node.");
> console.log(text)
```

[Copy](#)

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.

With these methods, we've created new elements and text nodes, but they are not visible on the front end of a website until they've been inserted into the document.

Inserting Nodes into the DOM

In order to see the new text nodes and elements we create on the front end, we will need to insert them into the `document`. The methods `appendChild()` and `insertBefore()` are used to add items to the beginning, middle, or end of a parent element, and `replaceChild()` is used to replace an old node with a new node.

Property/Method	Description
<code>node.appendChild()</code>	Add a node as the last child of a parent element
<code>node.insertBefore()</code>	Insert a node into the parent element before a specified sibling node
<code>node.replaceChild()</code>	Replace an existing node with a new node

To practice these methods, let's create a to-do list in HTML:

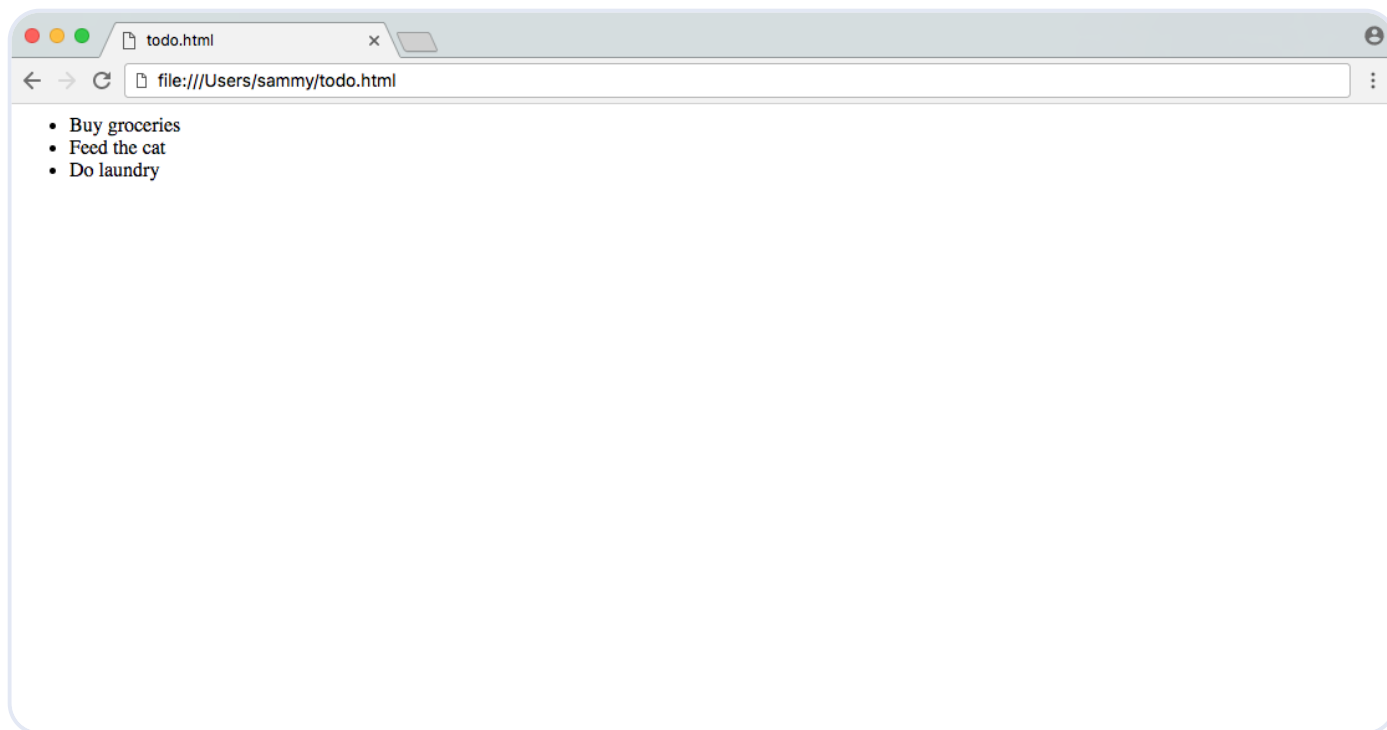
todo.html

```
<ul>
  <li>Buy groceries</li>
  <li>Feed the cat</li>
  <li>Do laundry</li>
</ul>
```

Copy

When you load your page in the browser, it will look like this:

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 order to add a new item to the end of the to-do list, we have to create the element and add text to it first, as we did in the “Creating New Nodes” section above.

```
> // To-do list ul element
> const todoList = document.querySelector('ul');
>
> // Create new to-do
> const newTodo = document.createElement('li');
> newTodo.textContent = 'Do homework';
```

[Copy](#)

Now that we have a complete element for our new to-do, we can add it to the end of the list with `appendChild()`.

```
> // Add new todo to the end of the list
> todoList.appendChild(newTodo);
```

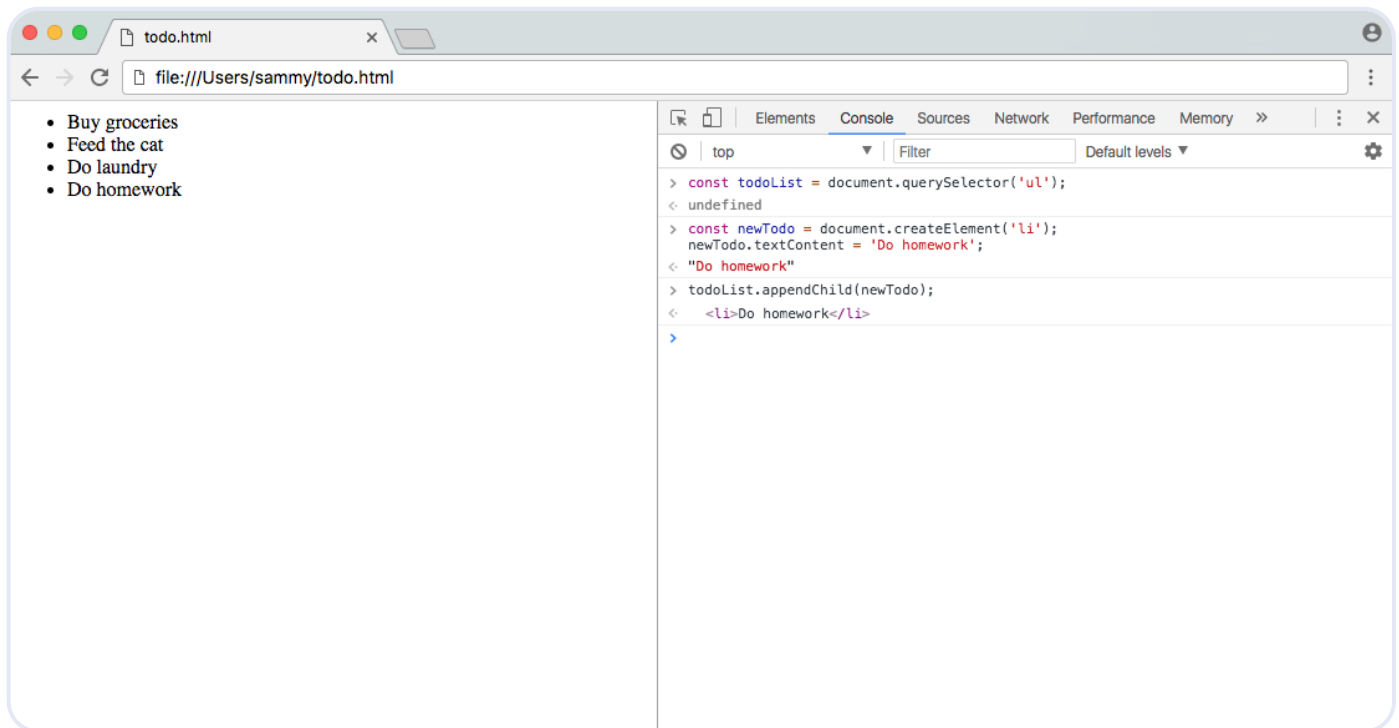
[Copy](#)

You can see the new `li` element has been appended to the end of the `ul`.

todo.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.

```
<li>Do homework</li>
</ul>
```



Maybe we have a higher priority task to do, and we want to add it to the beginning of the list. We'll have to create another element, as `createElement()` only creates one element and cannot be reused.

```
> // Create new to-do
> const anotherTodo = document.createElement('li');
> anotherTodo.textContent = 'Pay bills';
```

Copy

We can add it to the beginning of the list using `insertBefore()`. This method takes two arguments — the first is the new child node to be added, and the second is the sibling node that will immediately follow the new node. In other words, you're inserting the new node before the next sibling node. This will look similar to the following pseudocode:

```
parentNode.insertBefore(newNode, nextSibling);
```

Copy

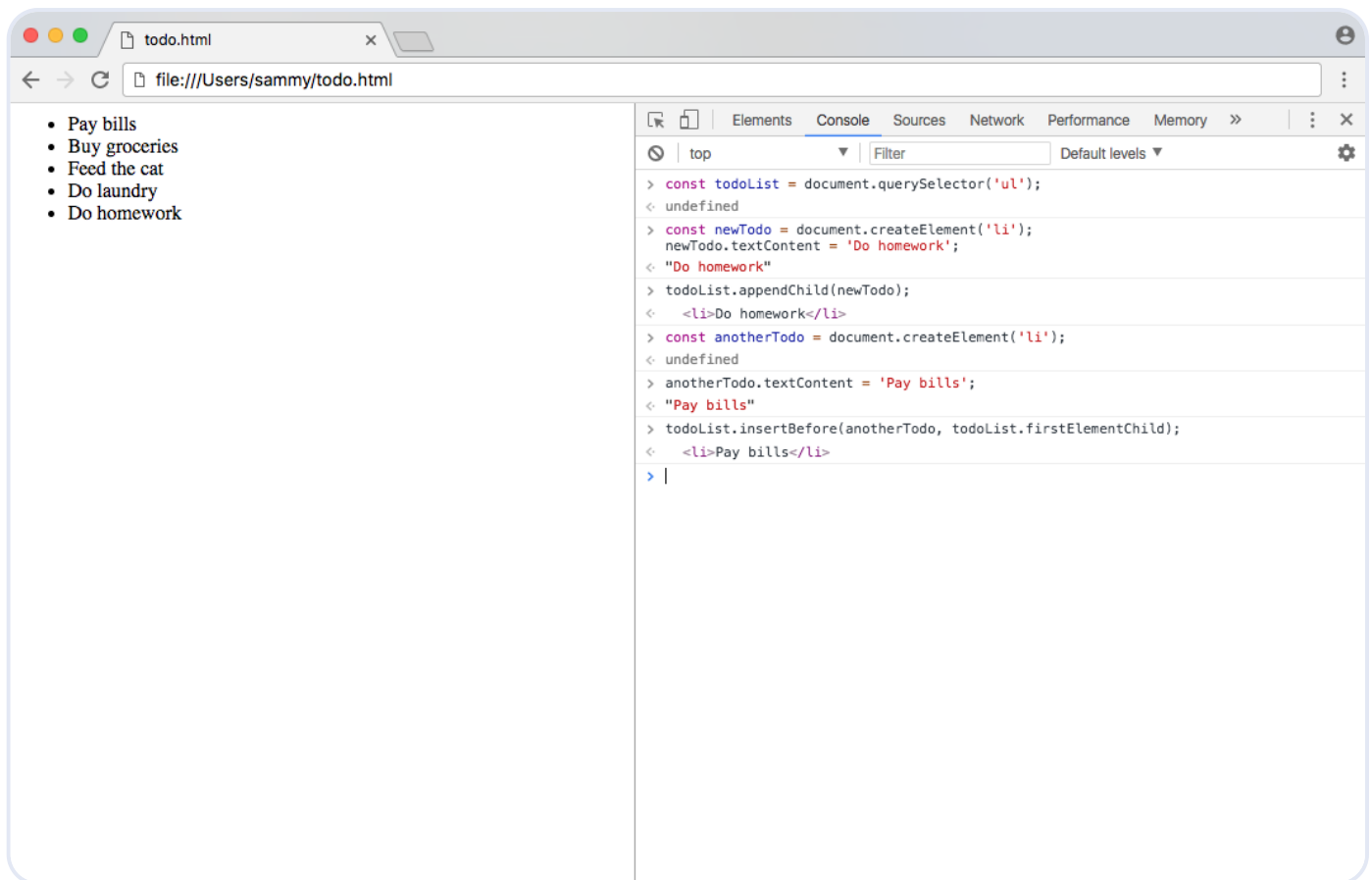
For our to-do list example, we'll add the new `anotherTodo` element before the first element child of the list, which is currently the `buy groceries` list item.

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.

todo.html

Copy

```
<ul>
  <li>Pay bills</li>
  <li>Buy groceries</li>
  <li>Feed the cat</li>
  <li>Do laundry</li>
  <li>Do homework</li>
</ul>
```



The new node has successfully been added at the beginning of the list. Now we know how to add a node to a parent element. The next thing we may want to do is replace an existing node with a new node.

We'll modify an existing to-do to demonstrate how to replace a node. The first step of creating a new element remains the same.

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.


```
parentNode.replaceChild(newNode, oldNode);
```

Copy

We will replace the third element child of the list with the modified to-do.

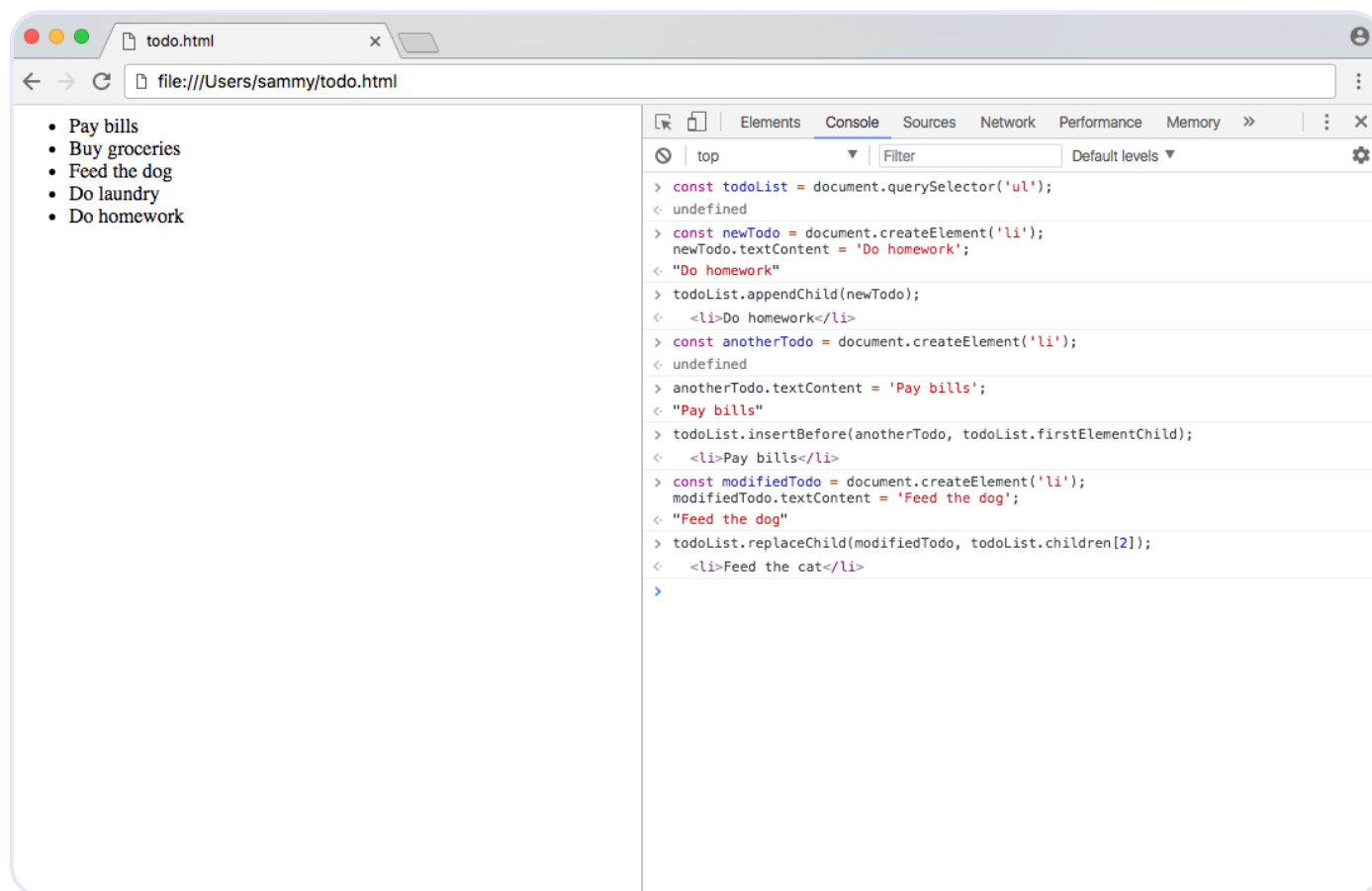
```
> // Replace existing to-do with modified to-do  
> todoList.replaceChild(modifiedTodo, todoList.children[2]);
```

Copy

todo.html

```
<ul>  
  <li>Pay bills</li>  
  <li>Buy groceries</li>  
  <li>Feed the dog</li>  
  <li>Do laundry</li>  
  <li>Do homework</li>  
</ul>
```

Copy



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.

Now we know how to create elements, add them to the DOM, and modify existing elements. The final step is to learn to remove existing nodes from the DOM. Child nodes can be removed from a parent with `removeChild()`, and a node itself can be removed with `remove()`.

Method	Description
<code>node.removeChild()</code>	Remove child node
<code>node.remove()</code>	Remove node

Using the to-do example above, we'll want to delete items after they've been completed. If you completed your homework, you can remove the `Do homework` item, which happens to be the last child of the list, with `removeChild()`.

```
> todoList.removeChild(todoList.lastElementChild);
```

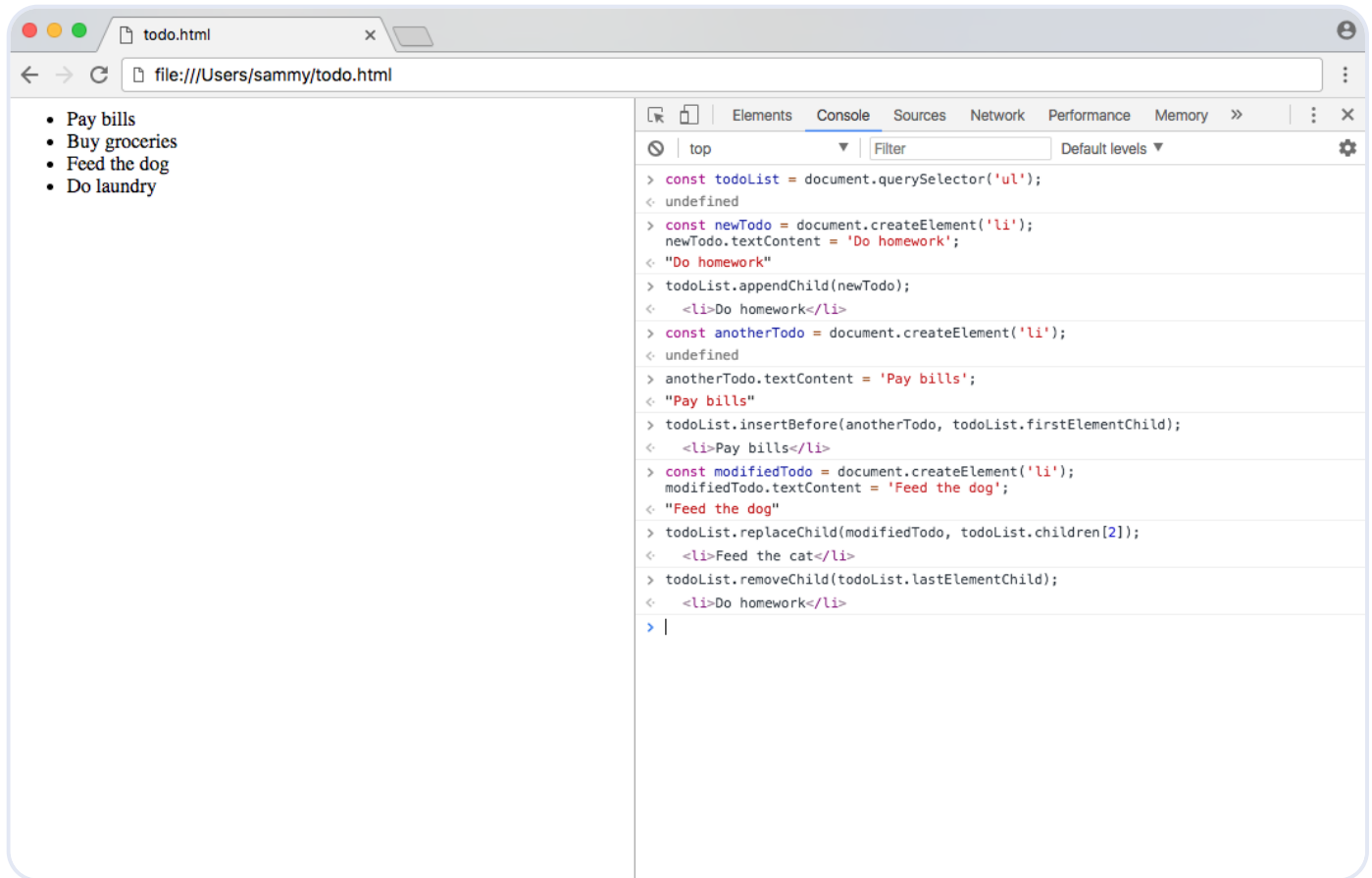
[Copy](#)

todo.html

```
<ul>
  <li>Pay bills</li>
  <li>Buy groceries</li>
  <li>Feed the dog</li>
  <li>Do laundry</li>
</ul>
```

[Copy](#)

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.



Another method could be to remove the node itself, using the `remove()` method directly on the node.

```
> // Remove second element child from todoList
> todoList.children[1].remove();
```

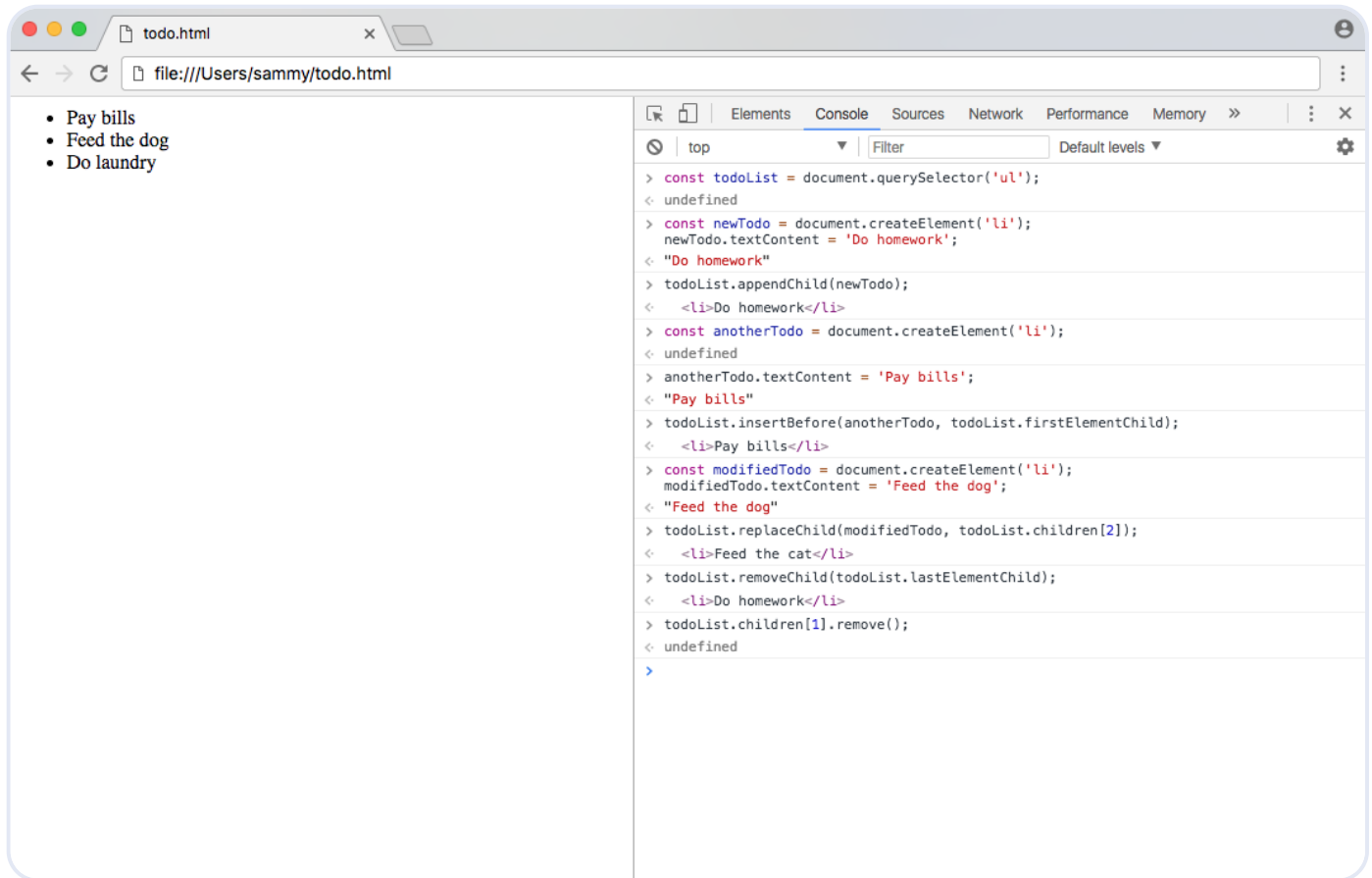
Copy

todo.html

```
<ul>
  <li>Pay bills</li>
  <li>Feed the dog</li>
  <li>Do laundry</li>
</ul>
```

Copy

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.



Between `removeChild()` and `remove()`, you can remove any node from the DOM. Another method you may see for removing child elements from the DOM is setting the `innerHTML` property of a parent element to an empty string (`" "`). This is not the preferred method because it is less explicit, but you might see it in existing code.

Conclusion

In this tutorial, we learned how to use JavaScript to create new nodes and elements and insert them into the DOM, and replace and remove existing nodes and elements.

At this point in the [Understanding the DOM series](#) you know how to access any element in the DOM, walk through any node in the DOM, and modify the DOM itself. You can now feel confident in creating basic front-end web apps with JavaScript.

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 Modify Attributes, Classes, and Styles in 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](#)

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

3 Comments

B *I* U



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

[LovableBlueShark](#) • April 7, 2022 

very good documentation. I am very glad for finding this documentation.

[Reply](#)

[rshanlon](#) • August 25, 2019 

Great tutorials! Your presentation and writing style is very clear!

I've been trying to learn more about innerHTML and thought XSS attacks were possible when using innerHTML connected to forms (i.e. user input). Otherwise, using innerHTML to just add HTML to the DOM isn't vulnerable to XSS. Is that how you understand it, too?

```
function showAnswer() {  
    const answer = "<p>The answer is <strong>Ada Lovelace</strong>!</p>";  
    document.querySelector('body').innerHTML = answer;  
}
```

I also looked into **.insertAdjacentHTML()** as a “safer” way to add HTML via JS. An easy riff for appending is using the ‘beforeend’ 1st argument + the content to append.

```
function showAnswer() {  
    const answer = "<p>The answer is <strong>Ada Lovelace</strong>!</p>";  
    document.querySelector('body').insertAdjacentHTML('beforeend', answer;  
}
```

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.

[dutchtulip](#) • October 17, 2018



These are really helpful. Thankyou!

[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 →

Popular Topics

Ubuntu

Linux Basics

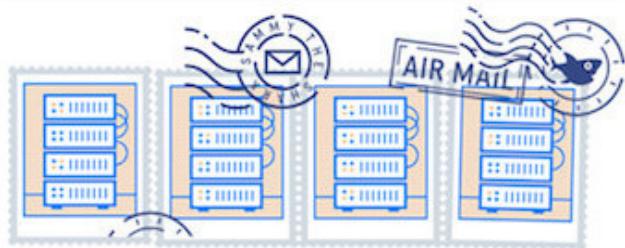
JavaScript

Python

MySQL

Docker

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.

[Free Managed Hosting →](#)

Get our biweekly newsletter

Sign up for Infrastructure as a Newsletter.

[Sign up →](#)

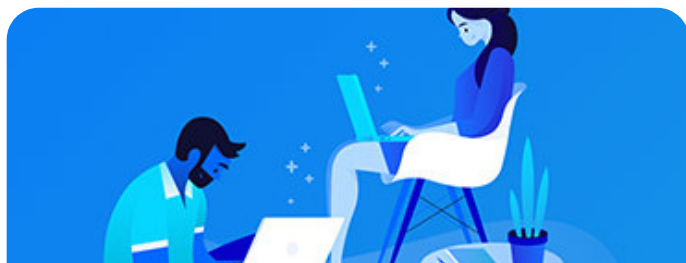
HOLLIE'S
HUB



FOR
GOOD

Hollie's Hub for Good

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

[Learn more →](#)

Become a contributor

You get paid: we donate to tech

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.

Featured on Community

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

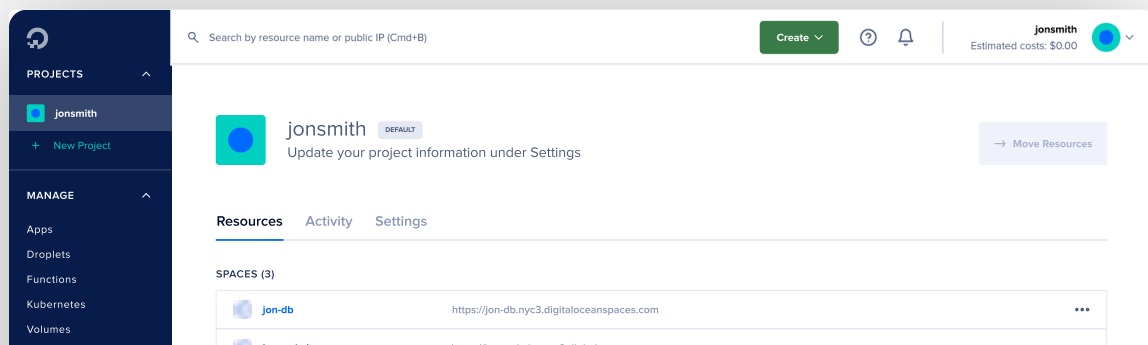
DigitalOcean Products

[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.

Blog	Droplets	CSS-Tricks	Web & Mobile Apps	Report Abuse
Careers	Kubernetes	Write for DOnations	Game Development	System Status
Customers	App Platform	Currents Research	Streaming	Share your ideas
Partners	Functions	Hatch Startup Program	VPN	
Channel Partners	Cloudways	deploy by DigitalOcean	SaaS Platforms	
Referral Program	Managed Databases	Shop Swag	Cloud Hosting for Blockchain	
Affiliate Program	Spaces	Research Program	Startup Resources	
Press	Marketplace	Open Source		
Legal	Load Balancers	Code of Conduct		
Security	Block Storage	Newsletter Signup		
Investor Relations	Tools & Integrations	Meetups		
DO Impact	API			
	Pricing			
	Documentation			
	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.