

What the DOM Covers

In this lesson we will see what exactly the DOM covers.
Let's begin!

WE'LL COVER THE FOLLOWING



- What does the DOM API offer?
- Listing 6-1: Transforming a static web page into an interactive one

In the [DOM Basics](#) lesson, you saw the graphical representation of a web page's structure, or using the HTML terminology, the **structure of an HTML document**.

The DOM API allows full control, both querying and altering the DOM, and you have practically full support in the API to leverage all features provided by the HTML markup.

What does the DOM API offer?

The best way to understand what the DOM API covers is to look at examples, but before doing this, let's have a short overview of what this API offers to developers:

- First of all, it **allows access to a certain part of the document (the HTML markup)**. You can access a certain markup element using its identifier, or a collection of elements by their name, class type (according to the values of the element's class attribute), their tag name (for example all `<h1>` elements), and several other aspects.
- You can **query the metadata of a document** including its URL, character set, last modification date, etc.

When you obtain a collection of HTML elements, you can iterate them and carry out some operation with chosen items. If you obtain an instance of a

markup element (for example, an `element`), you can access the content of that element, including the full markup they embed, the embedded text, as well as the values of the element's attributes. Moreover, you can modify both the content and the attributes with only a few exceptions, where a certain attribute or property of an element is read-only.


Each element instance has global attributes which I have not explicitly mentioned yet; for example, the ID attribute, which has been used in many listings and exercises earlier.

There are a few common operations on every markup element that you can use. For example, you may set the focus to the specified element, or ask the browser to scroll it into view, and so on.

You already learned that JavaScript is the cornerstone of web page interactivity; when you interact with a page (or a component on the page), a script is triggered. This behavior is based on the DOM. Each markup element can respond to dozens of events that can be accessed through the DOM. You can define a response for an event, and you can trigger events as well.

At the document level, DOM provides you information that allows carrying out a number of document operations, such as opening an HTML document, closing the current document, writing content directly to the document, and so on.

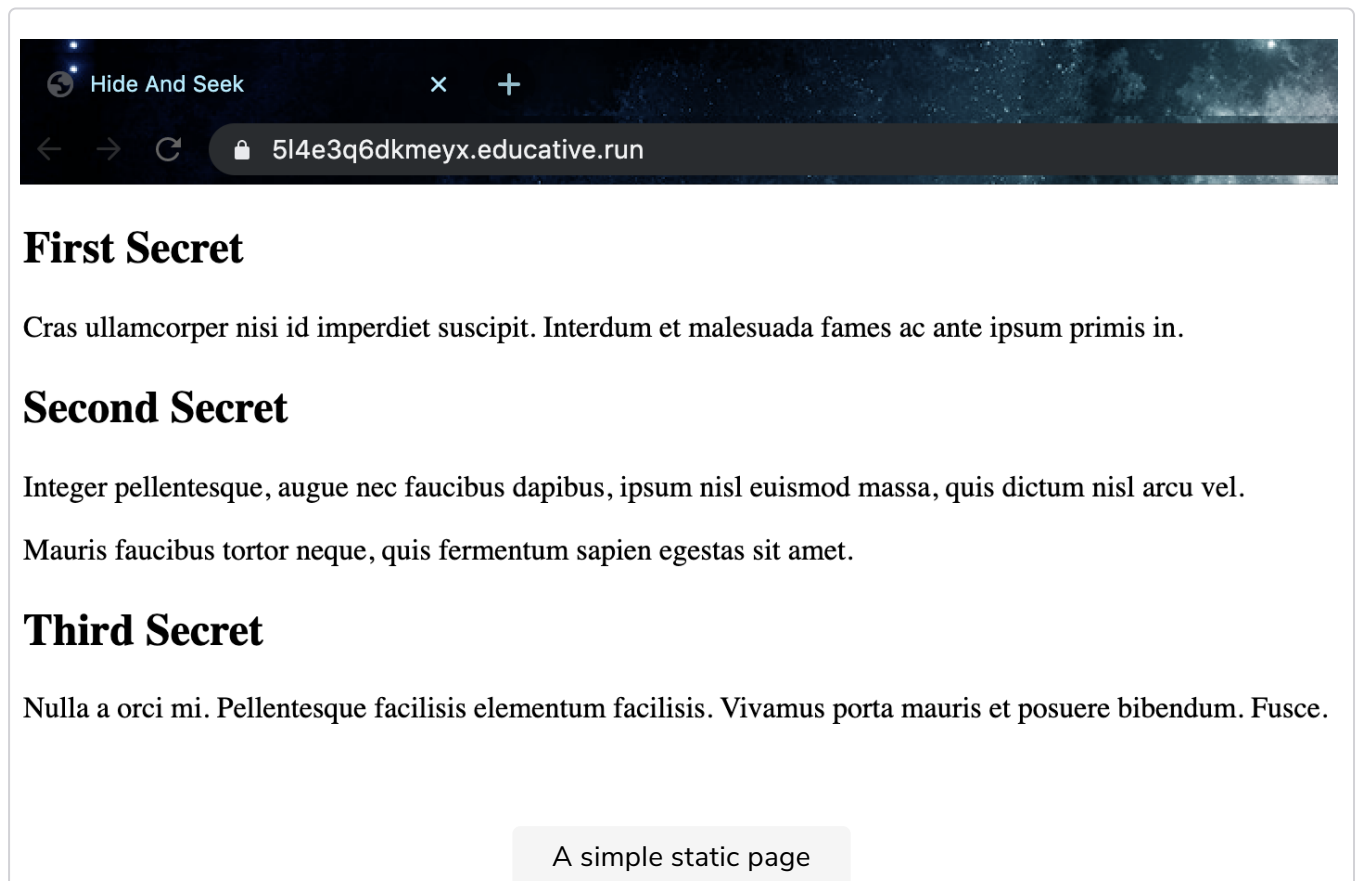
Now, you probably have a basic understanding of the DOM API, so it is time to take a closer look at several code listings and exercises.

 **NOTE:** In the next exercises you will use JavaScript to understand how the DOM works. Just like in the previous exercises, you are not assumed to know JavaScript, but you will understand all programs with the help of explanations. You do not have to wait very long to get closer to JavaScript programming; in the [next chapter](#), you will learn every nitty-gritty detail you may miss now.

In the remaining part of this section, you will transform a static web page into an interactive one. This page has this simple markup, as shown in Listing 6-1.

Listing 6-1: Transforming a static web page into an interactive one

This page displays three headings with associated content with a simple style sheet, as shown in the image below.



With a few steps, you will transform it into an interactive page that allows collapsing and expanding the content of headings, as shown below:

Output
JavaScript
HTML

+ **First Secret**

+ **Second Secret**

+ **Third Secret**



The page turned into an interactive one

To carry out this transformation, you will insert an extra `` element into each `<h2>` heading, which represents the plus (+) or minus (-) sign to expand or collapse the heading.

You will also add an event to each `<h2>` element that is triggered when the user clicks the heading, and it will expand or collapse the `<div>` element that holds the related content.

This event handler takes care to exchange the plus or minus signs accordingly.

We will see exactly how to do this in the upcoming lessons.

In the *next lesson*, we see the process of querying the document.