# DOM Basics

In this lesson, we meet the DOM again, in detail.
Let's begin!

By now, you already understand how HTML, CSS, and JavaScript connect to each other to provide fully functional web pages. The structure of the page is built on **HTML** markup that provides the content.

The typography and visual appearance of the page is made complete by **CSS** that assigns style to the structure. **JavaScript** brings interactivity, motion, and dynamism to the page.

In the exercises and listings of the previous chapters you often see HTML markup mixed with styles and scripts, so you probably got used to the phenomenon of these three technologies working together.

*Besides the role-based connection among HTML, CSS, and JavaScript, there is another point of gravity that attracts and binds them: the **Document Object Model**, or shortly, DOM.*

In this chapter, you will learn why it is important in creating your web pages.

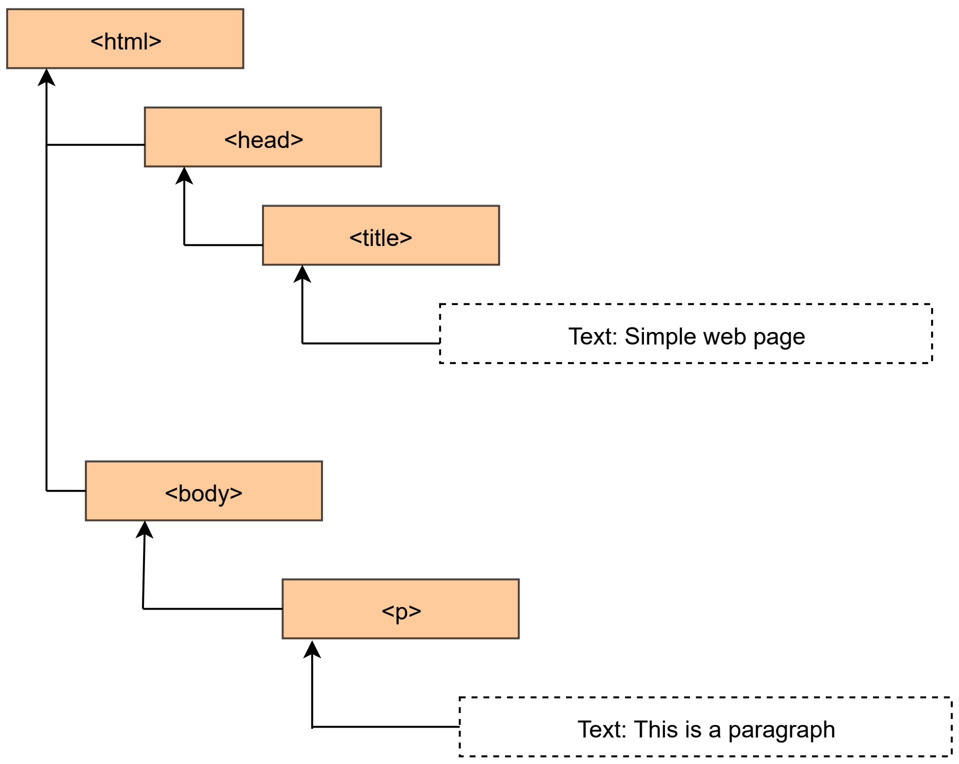
## The basics of the DOM #

The structure of HTML markup provides a **hierarchy of elements**, similar to any programming language providing a **syntax tree** and a **semantic tree**.

For example, take a look at this simple markup:

```
<html>
  <head>
    <title>Simple web page</title>
  </head>
  <body>
    <p>This is a paragraph</p>
  </body>
</html>
```

The natural hierarchy of the components of the page is easy to understand. The `<html>` element directly embeds two other elements, `<head>`, and `<body>`, and each of them nests another element, `<title>`, and `<p>`, as shown in the image below:



The structure of a simple HTML markup

As you can see from the above image, texts enclosed by `<title>` and `<p>` are also part of this structure.

*If you try to think as the rendering engine of the browser does, this object model is indispensable to display the page.*

Right before showing the page, the browser must build up the structure of the

Right before showing the page, the browser must build up the structure of the page. This step generally starts by parsing the HTML source and ends with a

structure of the page represented as a tree structure in the memory.

Different browsers may use separate implementations to express this structure, but all representations must use the same semantics; otherwise they would render different pages and page structures.

After having prepared this model in the memory, rendering engines utilize it to display the page according to the HTML standard.

---

In the *next lesson*, we'll study the ubiquitous language of page representation.