

22

USING JAVASCRIPT and the Document Object Model

OVERVIEW

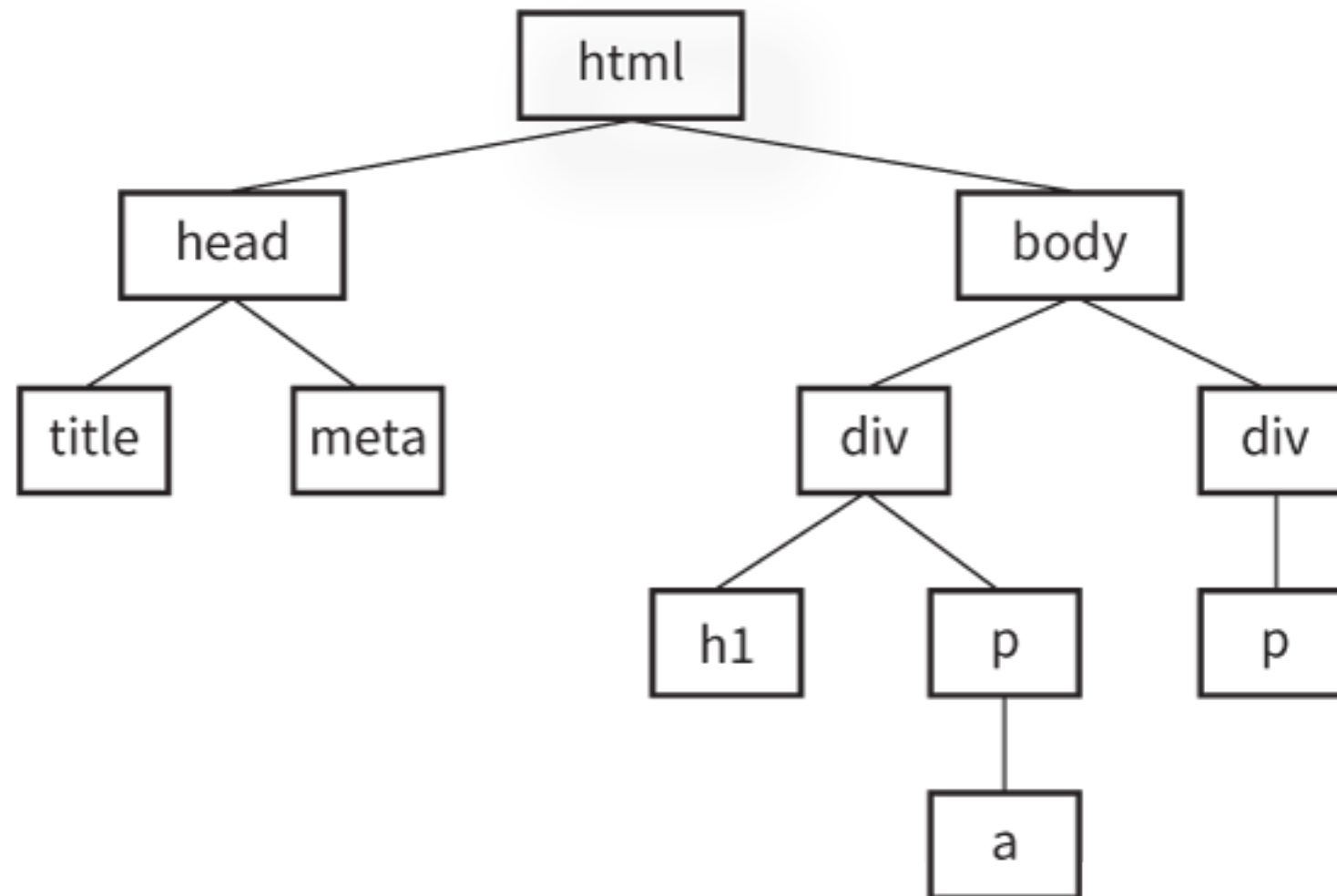
- **What the DOM is**
- **Accessing and changing elements, attributes, and contents**
- **Polyfills**
- **JavaScript libraries**

Intro to the DOM

- The **Document Object Model (DOM)** is a **programming interface** that provides a way to access and manipulate the contents of a document.
- It provides a structured **map of the document** and a set of **methods** for interacting with them.
- It can be used with other XML languages and it can be accessed by other programming languages (like PHP, Ruby, etc.).

Node Tree

The DOM treats the structure of a document like a tree with branches:

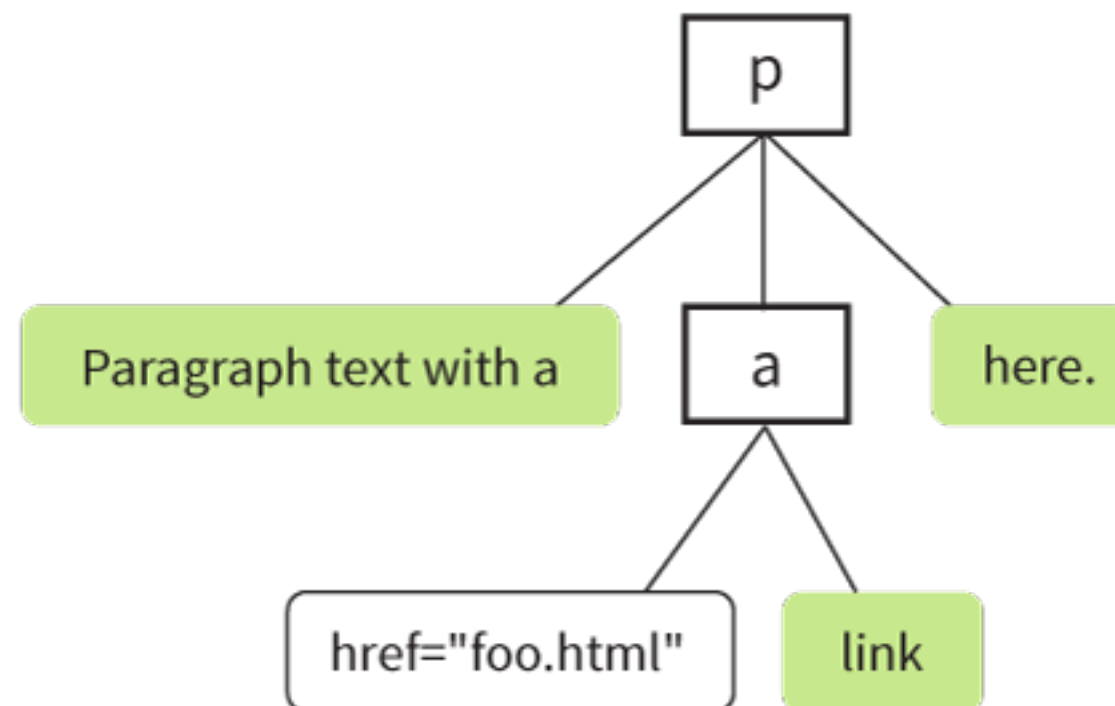


Node Tree (cont'd)

Every element, attribute, and piece of content is a node on the tree and can be accessed for scripting:

The nodes within
a `p` element

```
<p>Paragraph text with a <a href="foo.html">link</a> here.</p>
```



Accessing Nodes

To point to nodes, list them separated by periods (.).

In this example, the variable `foo` is set to the HTML content of an element with `id="beginner"`:

```
var foo = document.getElementById("beginner").innerHTML;
```

- The **document** object points to the page itself.
- **getElementById** specifies an element with the `id` "beginner".
- **innerHTML** stands for the HTML content within that element.

Accessing Nodes (cont'd)

Methods for accessing nodes in the document:

getElementsByTagName()

Accesses all elements with the given tag name

Example: `document.getElementsByTagName("p");`

getElementById()

Accesses a single element by the value of its `id` attribute

Example: `document.getElementById("special");`

getElementsByClassName()

Access elements by the value of a class attribute

Example: `document.getElementsByClassName("product");`

Accessing Nodes (cont'd.)

`querySelectorAll()`

Accesses nodes based on a CSS selector

Example: `document.querySelector(".sidebar p");`

`getAttribute()`

Accesses the value of a given attribute

Example: `getAttribute("src")`

Manipulating Nodes

There are several built-in methods for manipulating nodes:

setAttribute()

Sets the value of a given attribute:

```
bigImage.setAttribute("src", "newimage.jpg");
```

innerHTML

Specifies the content inside an element (including markup if needed):

```
introDiv.innerHTML = "<p>This is the intro text.</p>";
```

style

Applies a style using CSS properties:

```
document.getElementById("intro").style.backgroundColor = "#000;";
```

Adding and Removing Elements

The DOM allows developers to change the document structure by adding and removing nodes:

- **`createElement()`**
- **`createTextNode()`**
- **`appendChild()`**
- **`insertBefore()`**
- **`replaceChild()`**
- **`removeChild()`**

Polyfills

A **polyfill** uses JavaScript to make new features work in browsers that don't natively support them.

- **Picturefill**: Enables support for `picture`, `srcset`, and `sizes`
- **Selectivizr**^{*}: Allows IE 6–8 to support CSS3 selectors
- **HTML5 shiv**^{*}: Allows IE6–8 to recognize HTML5 elements

^{*}If you don't need to support IE 8 and earlier, you don't need these polyfills.

JavaScript Libraries

- A **JavaScript library** is a collection of prewritten functions and methods that you can use in your scripts to accomplish common tasks or simplify complex ones.
- Some are large frameworks for building complex applications.
- Some are targeted to specific tasks, such as forms or math.
- The most popular library is **jQuery**.
- Try searching “JavaScript library for _____” to see if there are pre-made scripts you can use or adapt to your needs.