

FULL STACK



Automation Testing

Document Object Model (DOM)



A Day in the Life of an Automation Test Engineer

Anna is already familiar with the fundamentals of JavaScript.

She must now add some additional features to the project, such as manipulating and styling, in order to make it attractive.

To achieve the above scenario, she will learn the Document Object Model (DOM), DOM methods, DOM elements, and DOM forms.



Learning Objectives

By the end of this lesson, you will be able to:

- 👁️ Analyze how to change the content of HTML components
- 👁️ Find HTML elements by ID, class name, and CSS selectors.
- 👁️ Describe node relationships to navigate
- 👁️ Analyze how to add and remove HTML elements

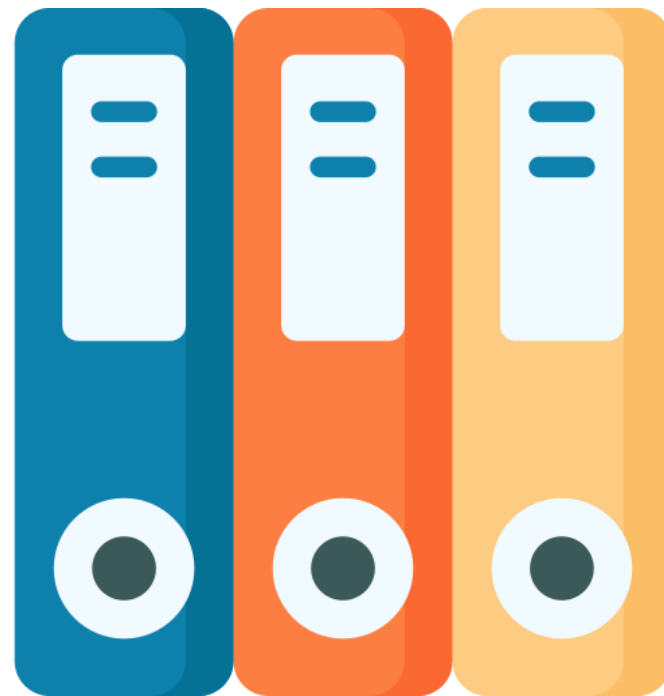


FULL STACK

What Is DOM?

What Is DOM?

The document object model (DOM) is a programming interface for web documents.

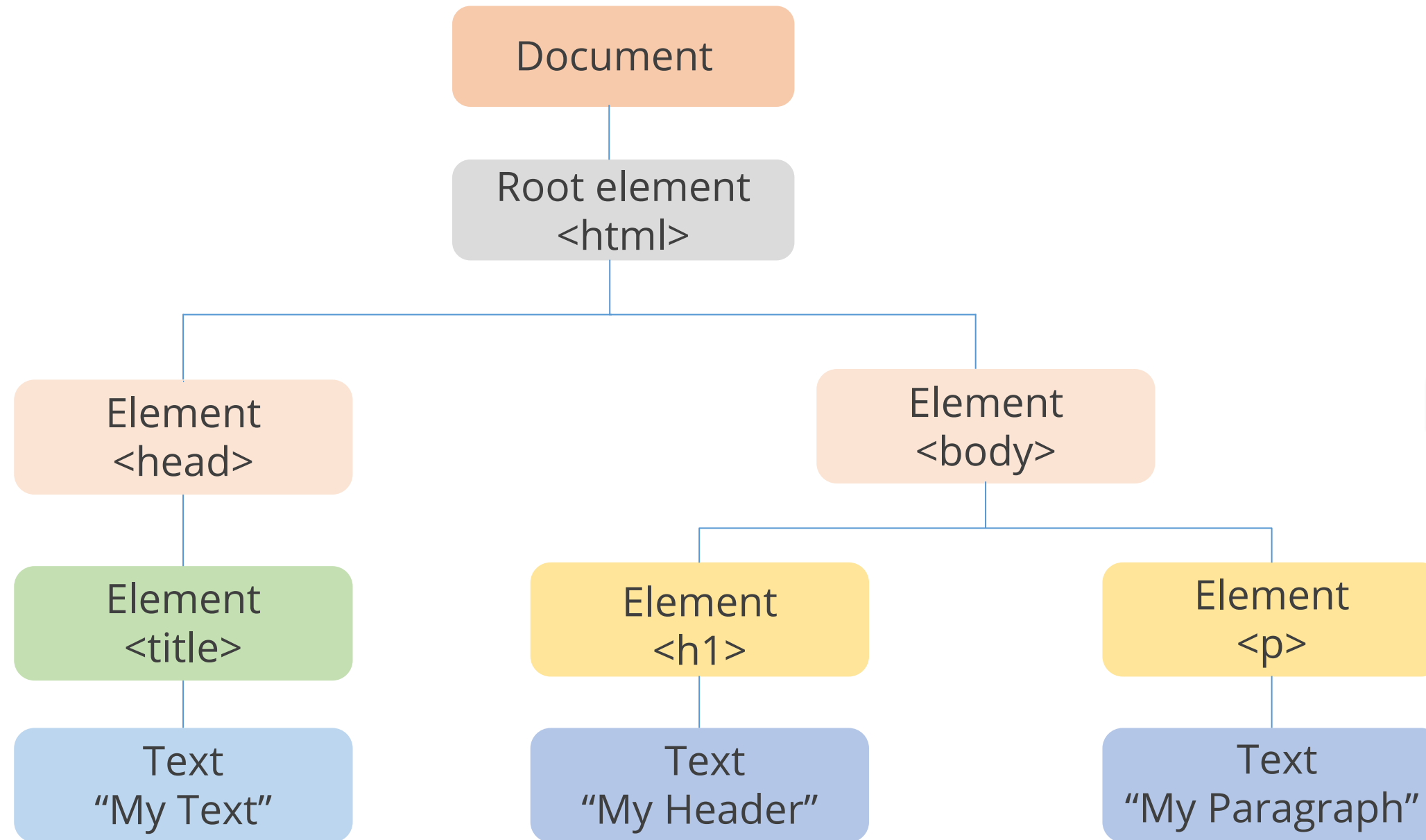


Nodes and objects are used to represent the document in the DOM.



DOM Model

The DOM model is built as an object tree, as shown below:



DOM and JavaScript

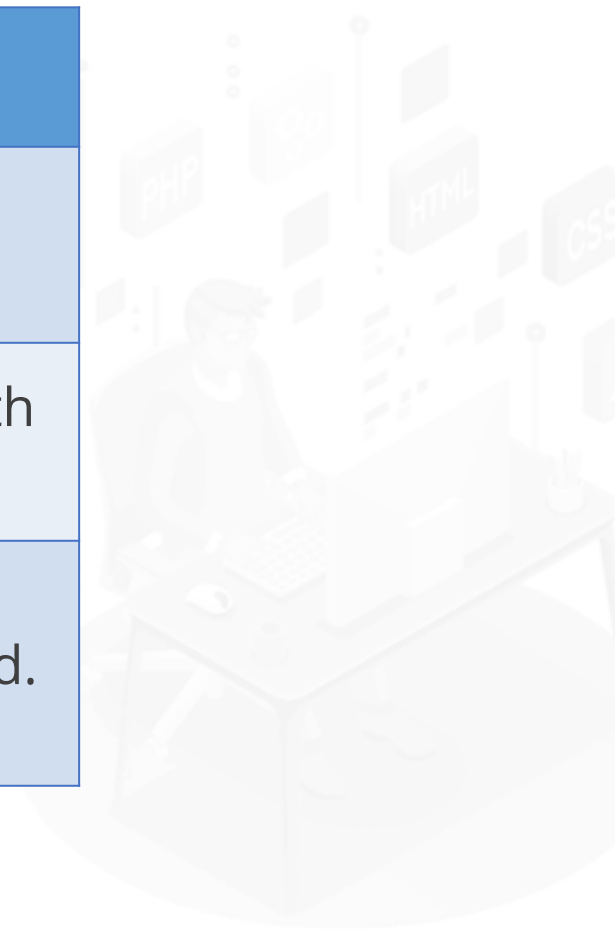
Although DOM is not a programming language, it is necessary for JavaScript to be able to have a model or notion of web pages, HTML documents, and SVG documents.



Methods of Document Object

The important methods of document object are as follows:

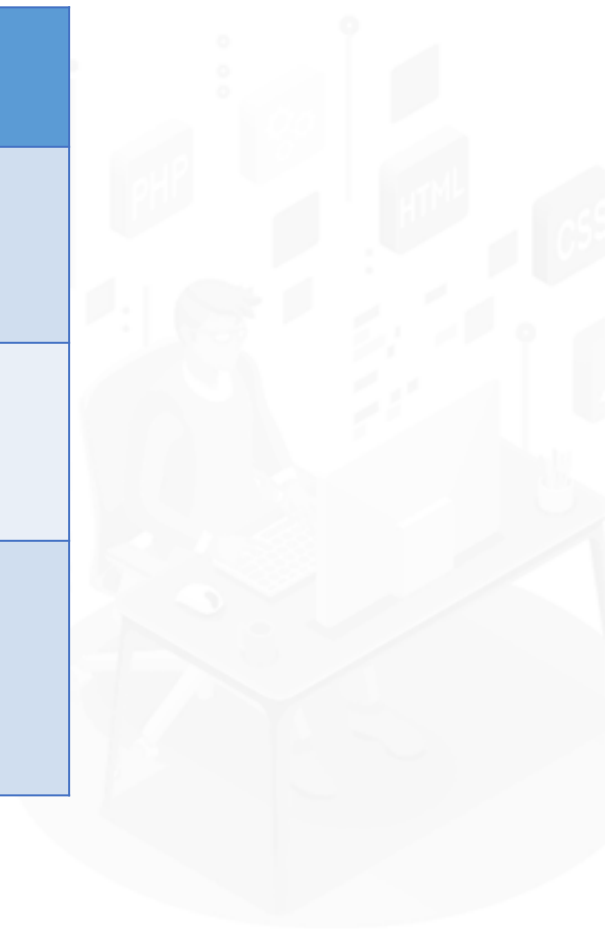
Method	Description
<code>write("string")</code>	The string is written to the document.
<code>writeln("string")</code>	The given string is written to the document with a newline character at the end.
<code>getElementById()</code>	The element with the given ID value is returned.



Methods of Document Object

The important methods of document object are as follows:

Method	Description
<code>getElementsByName()</code>	All elements with the specified name value are returned.
<code>getElementsByTagName()</code>	All elements with the specified tag name are returned.
RegExp	Regular expression is represented.



DOM Programming Interface

All other objects in a web page are owned by the HTML DOM document object.



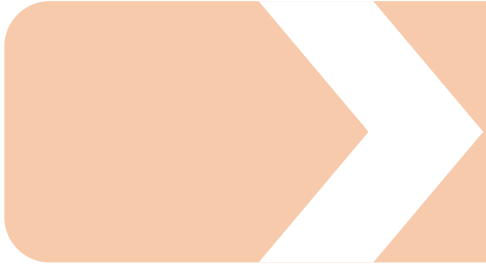
JavaScript can access the HTML document object model.



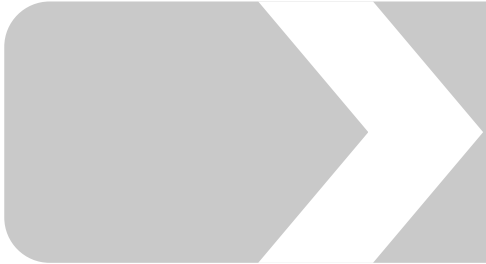
All HTML elements are defined as objects in the DOM.




DOM Programming Interface



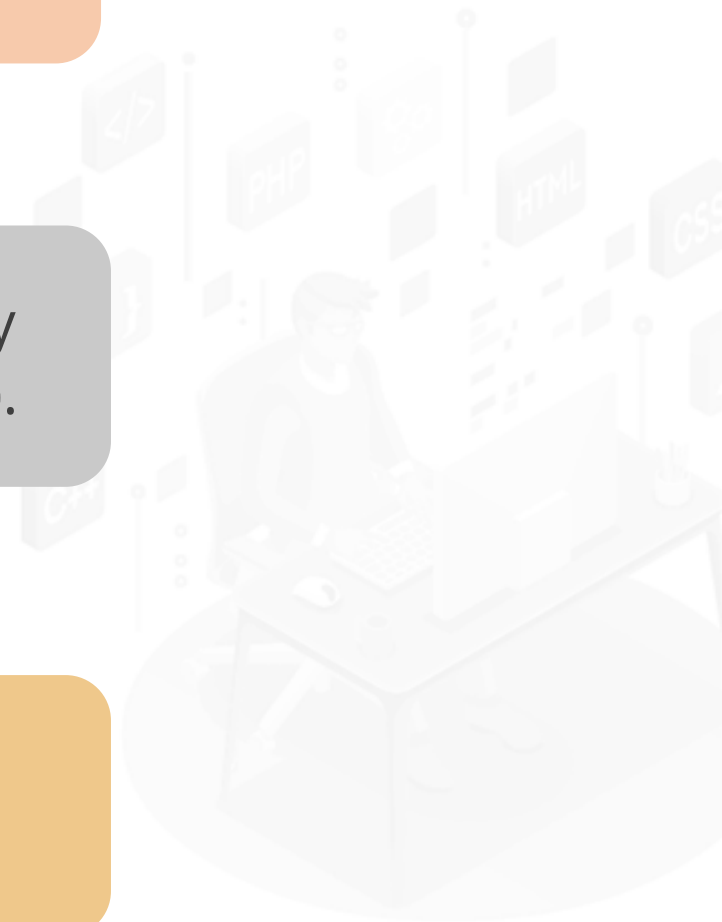
Each object's properties and methods make up its programming interface.



A property is a value that can be obtained or changed by the user (like changing the content of an HTML element).

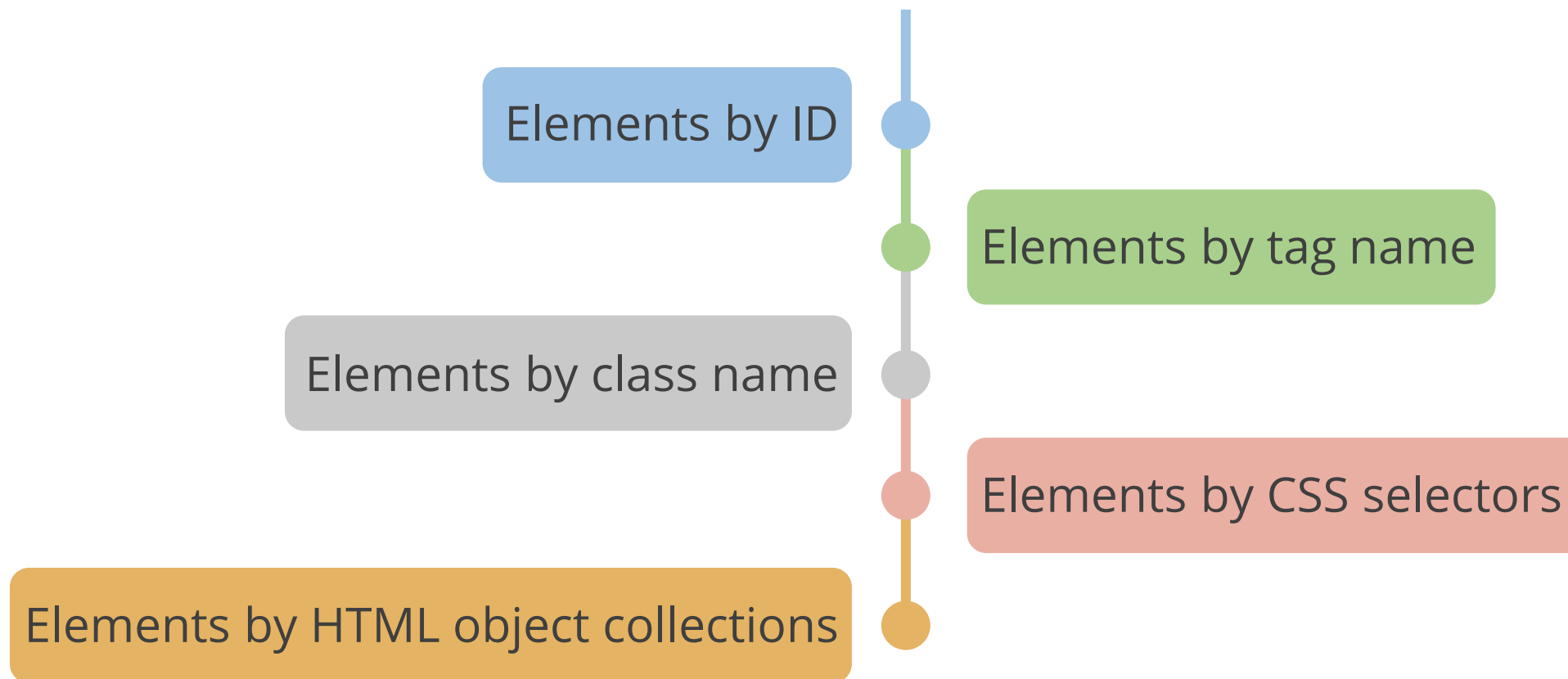


A method is an action that a user can perform (like adding or deleting an HTML element).



DOM Elements

DOM elements are similar to DIV, HTML, and BODY elements. All of these can be given classes with CSS or interacted with JavaScript.



Finding HTML Elements by ID

The `getElementById()` returns an Element object that represents the element whose id property matches the specified string.

Example:

```
const element = document.getElementById("intro");
```



Finding HTML Elements by Tag Name

The `getElementsByName()` method of the element class returns a live HTML collection of elements with the specified tag name.

Example:

```
const element = document.getElementsByTagName ("p") ;
```



Finding HTML Elements by Class Name

The `getElementsByClassName()` method returns a list of child elements that have the specified class name.

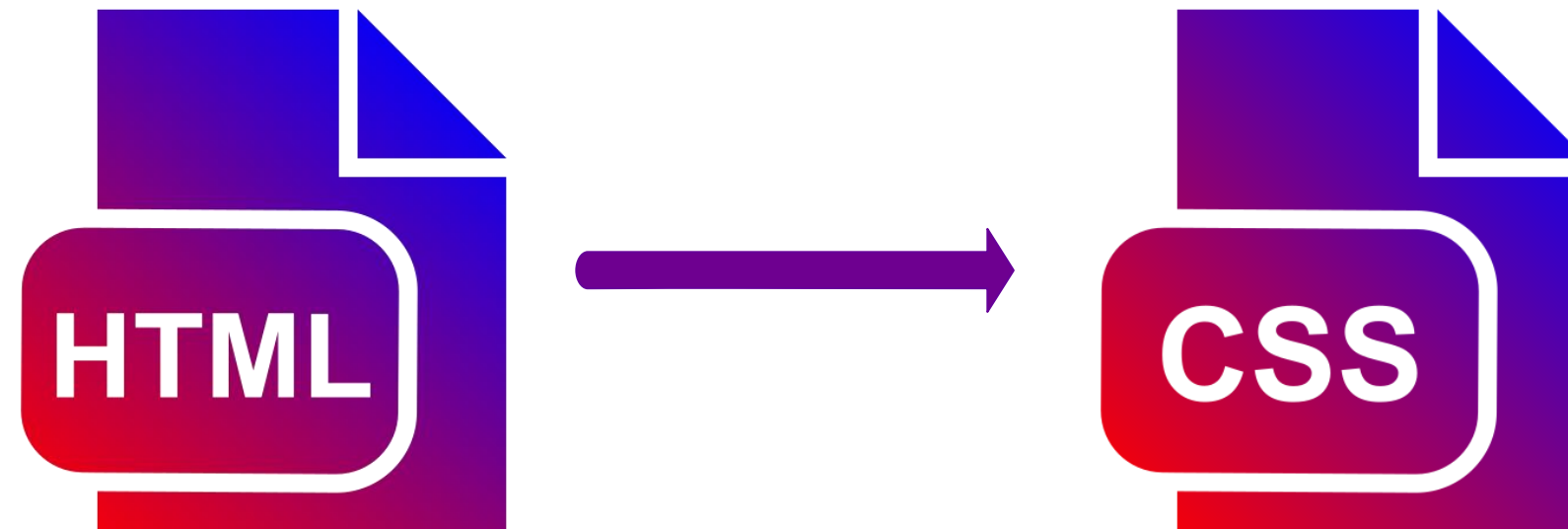
Example:

```
const x = document.getElementsByClassName("intro");
```



Finding HTML Elements by CSS Selectors

In order to make changes to HTML elements in a JavaScript program, the user may need to use CSS selectors.



The **querySelector()** and **querySelectorAll()** JavaScript methods can help users with this.

Finding HTML Elements by CSS Selectors

The `querySelector()` method of the Element Interface can be used to find elements using CSS selectors.

Example:

```
Const x = document.querySelector("p");
```

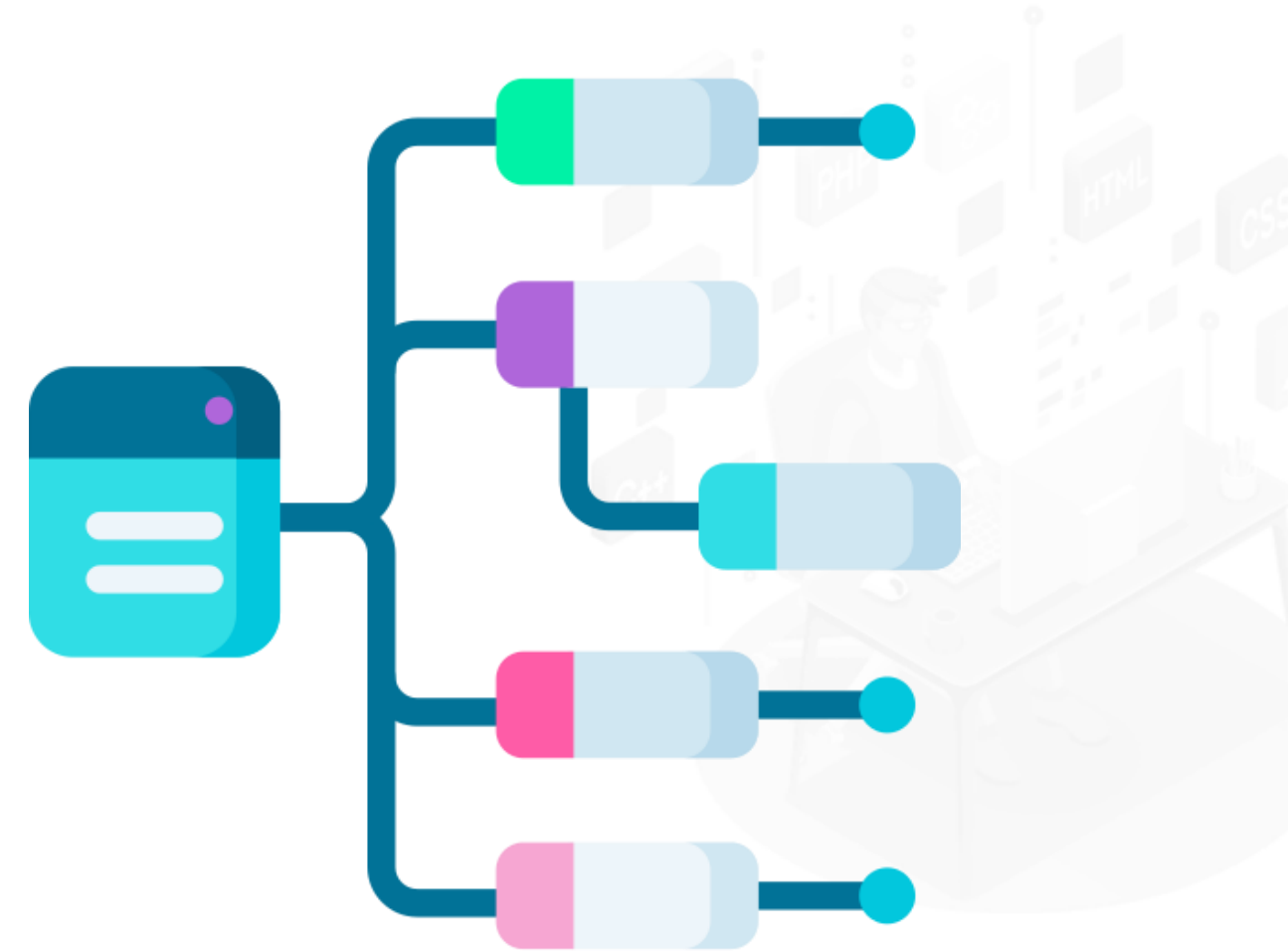


Finding HTML Elements by CSS Selectors

The `querySelectorAll()` method returns a static node list object with a collection of child elements that match the CSS selectors.

Example:

```
Const x = document.querySelectorAll("p.intro");
```



Finding HTML Elements by HTML Object Collections

HTML collection is a list of HTML elements that looks like an array. The length Property returns the collection's element count.

Example:

```
const x = document.forms["frm1"];
let text = "" ;
for (let i = 0; i < x.length; i++)
{
    text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML =
text;
```

The length Property returns the collection's element count.



JavaScript HTML DOM - Changing HTML

The innerHTML property is the simplest way to change the content of an HTML element.

Example:

```
<html>
<body>
<p id="p1">Hello World!</p>
<script>
document.getElementById("p1").innerHTML = "New text!";
</script>
</body>
</html>
```



JavaScript HTML DOM - Changing HTML

Changing the value of an HTML elements attribute:

Example:

```
<!DOCTYPE html>
<html>
<body>

<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>
</body>
</html>
```



JavaScript HTML DOM - Changing HTML

JavaScript can create dynamic HTML content:

Example:

```
<!DOCTYPE html>
<html>
<body>
<script>
document.getElementById("demo").innerHTML = "Date : " +
Date();
</script>
</body>
</html>
```



DOM Forms

If a form field (name) is left blank, the function displays an error message and returns false, preventing the form from being submitted.

JavaScript Example:

```
function validateForm() {  
    let x = document.forms["myForm"]["fname"].value;  
    if (x == "") {  
        alert("Name must be filled out");  
        return false;  
    }  
}
```



DOM Forms

The function can be called after the form has been submitted.

HTML Example:

```
<form name="myForm" action="/action_page.php"
onsubmit="return validateForm()" method="post">
Name: <input type="text" name="fname">
      <input type="submit" value="Submit">
</form>
```



Automatic HTML Form Validation

The required attribute prevents the form from being submitted if a form field (name) is left empty.

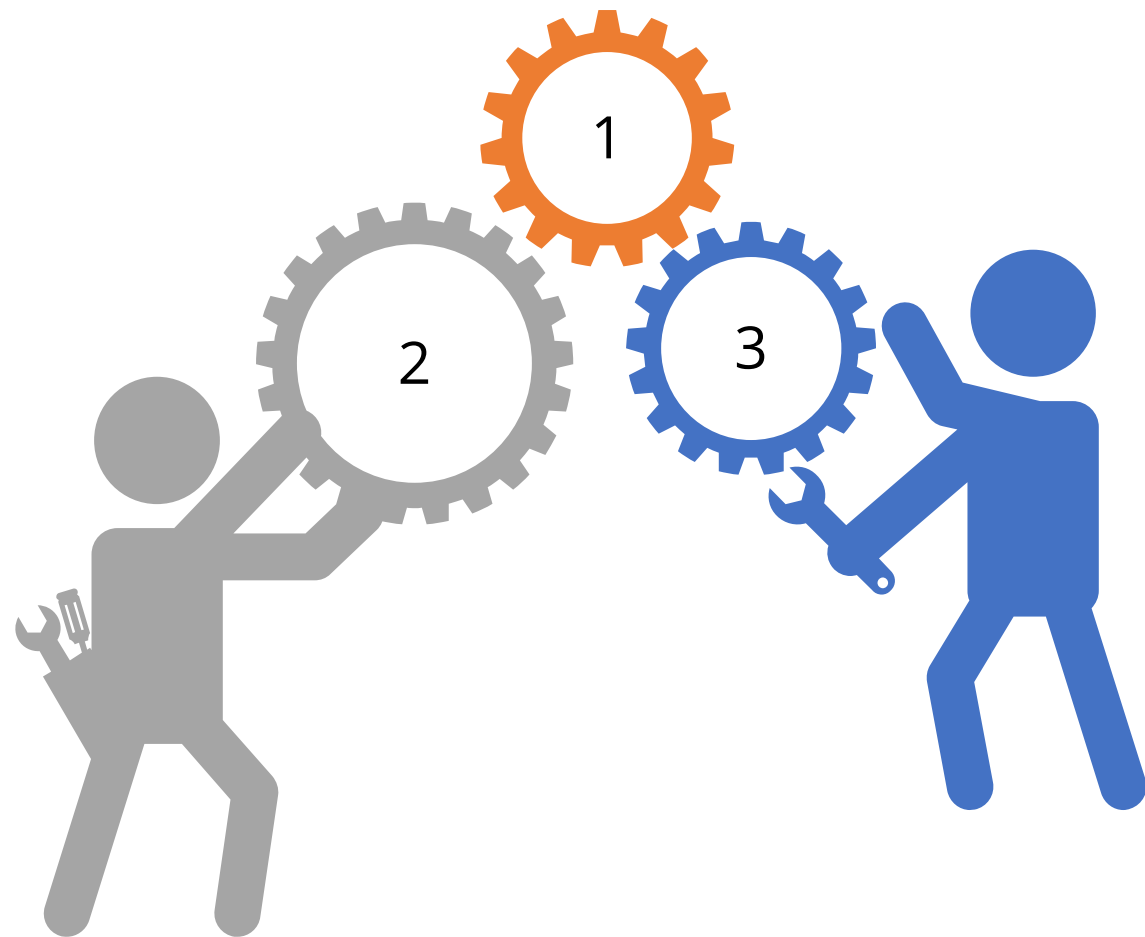
HTML Example:

```
<form action="/action_page.php" method="post">  
  <input type="text" name="fname" required>  
  <input type="submit" value="Submit">  
</form>
```



Data Validation

Data validation is the process of verifying the user information is accurate, complete, and valuable.



All required fields were filled out by the user.

The user provided a correct date.

The user typed anything in a number box.



Data Validation

Validation can be defined in a variety of ways and implemented in a variety of ways:



Client-side validation:

Before input is sent to a web server, it is processed by a web browser.

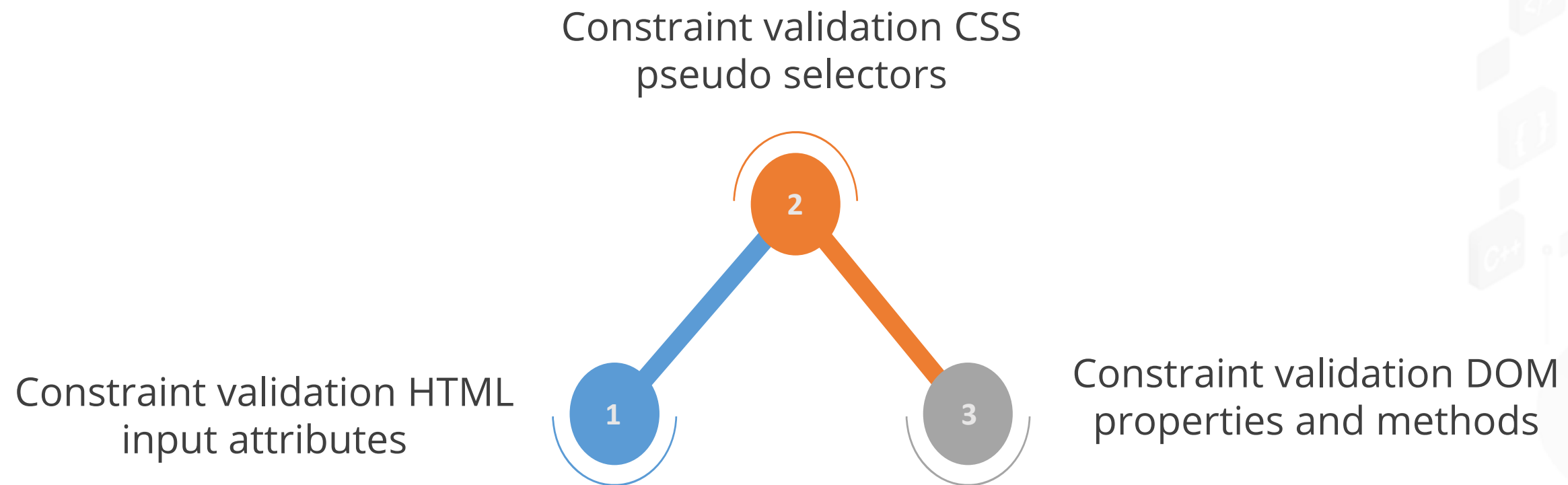


Server-side validation:

After input has been sent to the server, a web server performs it.

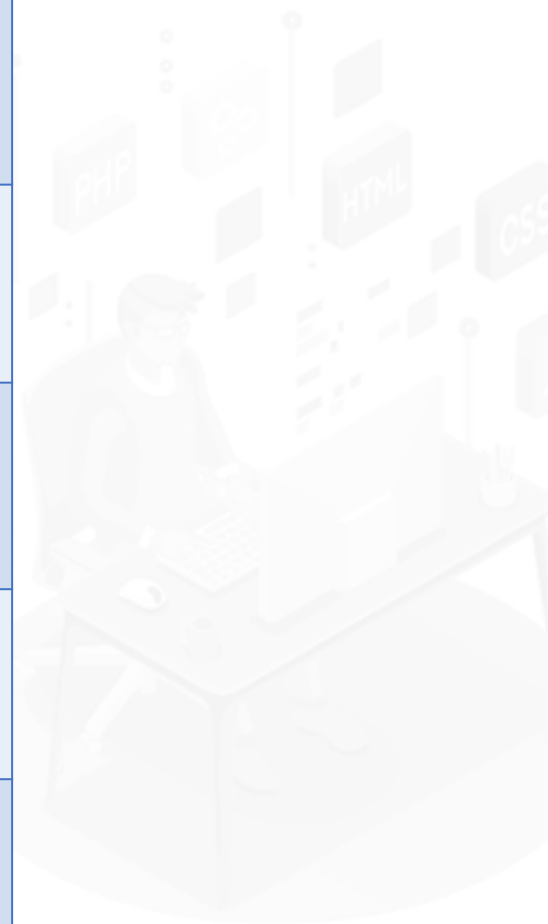
HTML Constraint Validation

The Constraint Validation API allows users to validate the values that they enter into form controls before sending to the server.



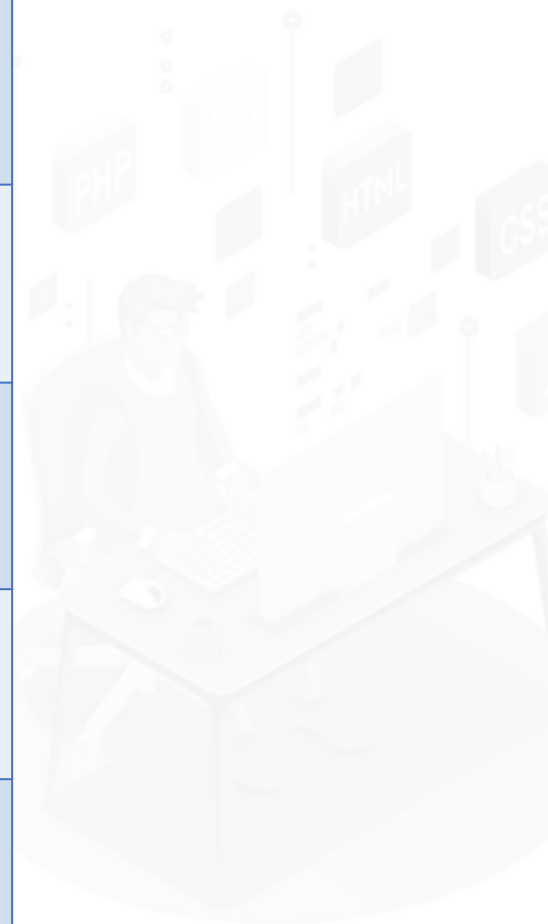
Constraint Validation HTML Input Attributes

Attribute	Description
Disabled	Specifies that the input element should be disabled
Max	Specifies the maximum value of an input element
Min	Specifies the minimum value of an input element
Pattern	Specifies the value pattern of an input element
Required	Specifies that the input field requires an element



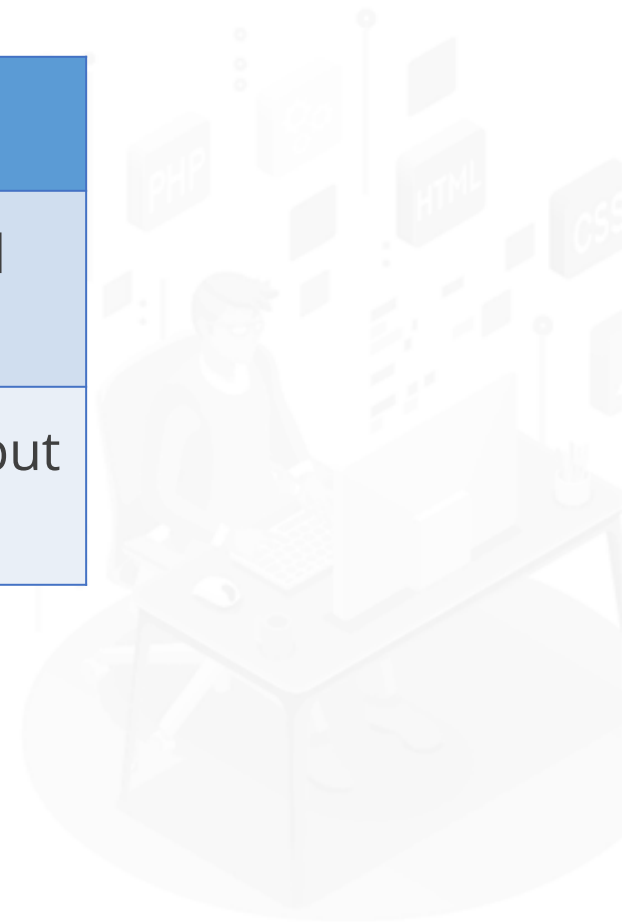
Constraint Validation CSS Pseudo Selectors

Selector	Description
:disabled	Selects input elements with the disabled attribute specified
:invalid	Selects input elements with invalid values
:optional	Selects input elements with no required attribute specified
:required	Selects input elements with the required attribute specified
:valid	Selects input elements with valid values



Constraint Validation DOM Methods

Property	Description
checkValidity()	Returns true if an input element contains valid data
setCustomValidity()	Sets the validation message property of an input element



Constraint Validation DOM Properties

Property	Description
validity	Contains boolean properties related to the validity of an input element.
validationMessage	Contains the message a browser will display when the validity is false.
willValidate	Indicates if an input element will be validated.

DOM - Changing CSS

CSS (Cascading Style Sheets) is a style sheet language for describing the appearance of a document authored in a markup language like HTML.



JavaScript can change the style of HTML elements using the HTML DOM.



DOM - Changing CSS

JavaScript methods, such as `getElementById()` and `getElementsByClassName()`, can change CSS styles such as color and font size of elements.

Example:

```
<html>
<body>
<p id="p2">Hello World!</p>
<script>
document.getElementById("p2").style.color = "blue";
</script>
</body>
</html>
```



DOM Animation

The style of an element is gradually changed in JavaScript animations.



A timer is used to initiate the changes. The animation appears to run indefinitely when the timer interval is small.



DOM Animation

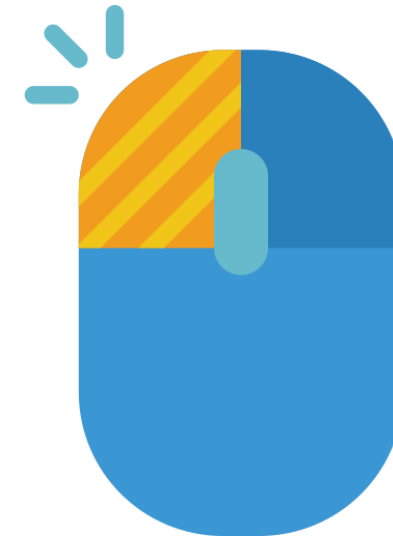
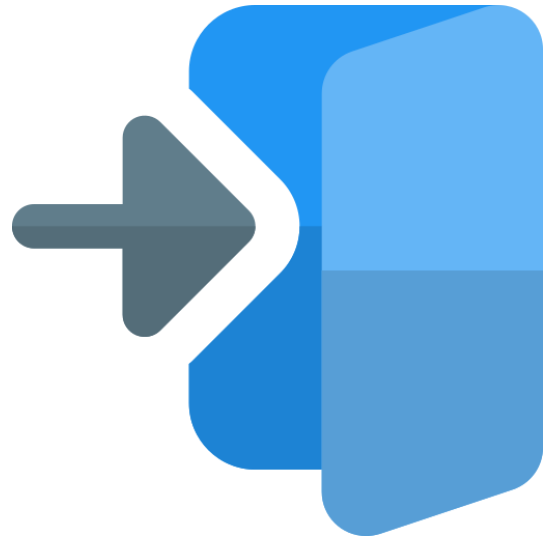
Property	Description
animationName	Describes the name of the keyframe attached to the selector
animationDuration	Describes the time how long an animation takes place
animationTimingFunction	Describes the speed of the animation
animationDelay	Describes the delay before the animation will start
animationIterationCount	Describes the number of times an animation takes place

DOM Animation

Property	Description
animationDirection	Describes if animation should play in reverse on alternate cycles
animationFillMode	Describes what values to apply when animation ends
animationPlayState	Describes whether an animation is running or paused

DOM Events

User activities or the browser can cause DOM Events, which are signals that something has happened or is happening.



DOM Events

Examples of DOM Events:

01

Click the mouse

02

Web page has loaded

03

Image has been loaded

04

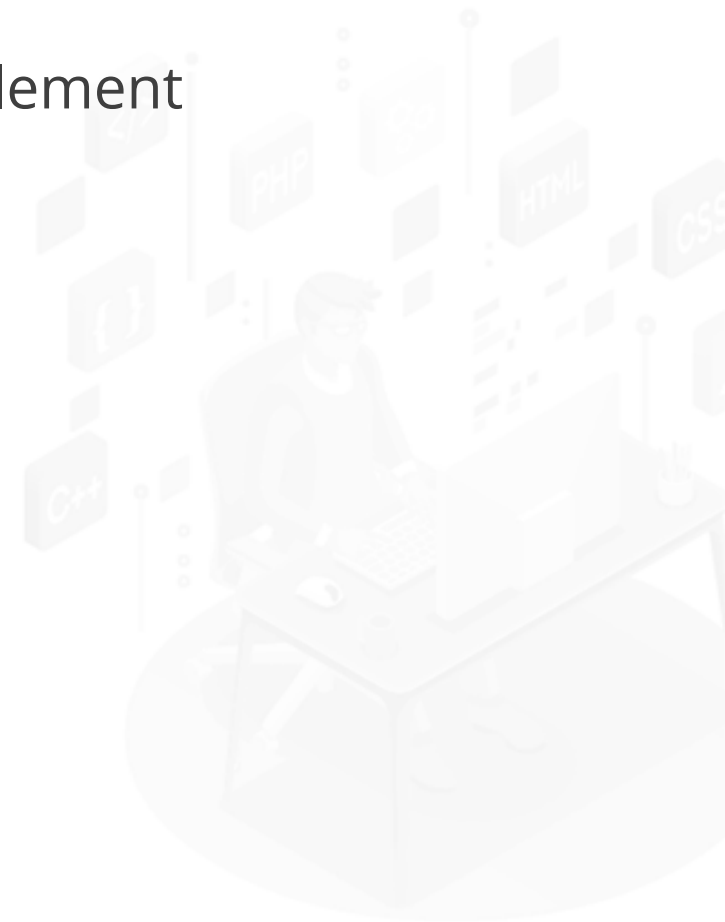
Mouse moves over an element

05

Input field is changed

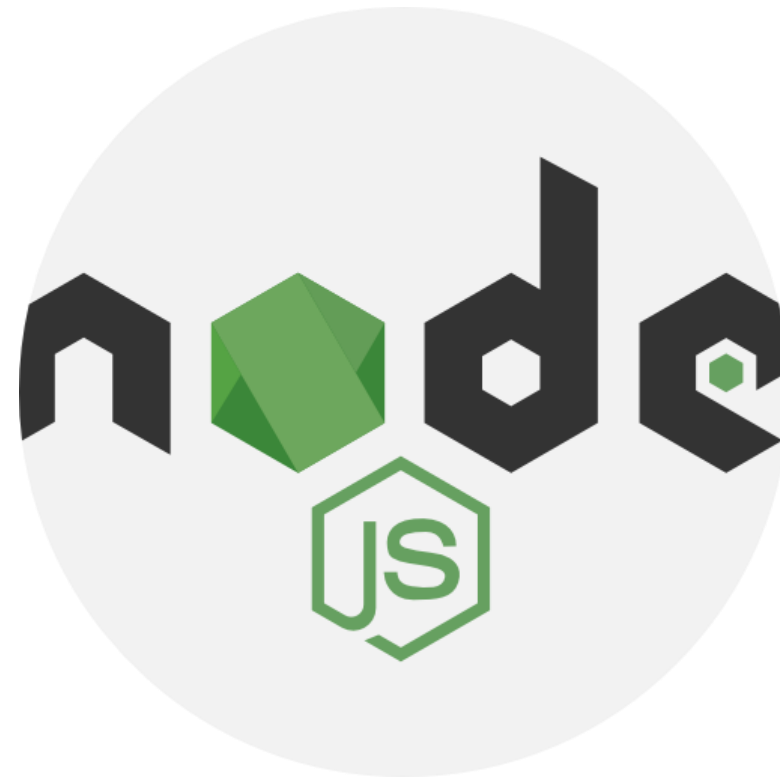
06

HTML form is submitted



DOM Nodes

All elements, attributes, text, and other aspects of the page are grouped in a hierarchical tree-like structure in the DOM.

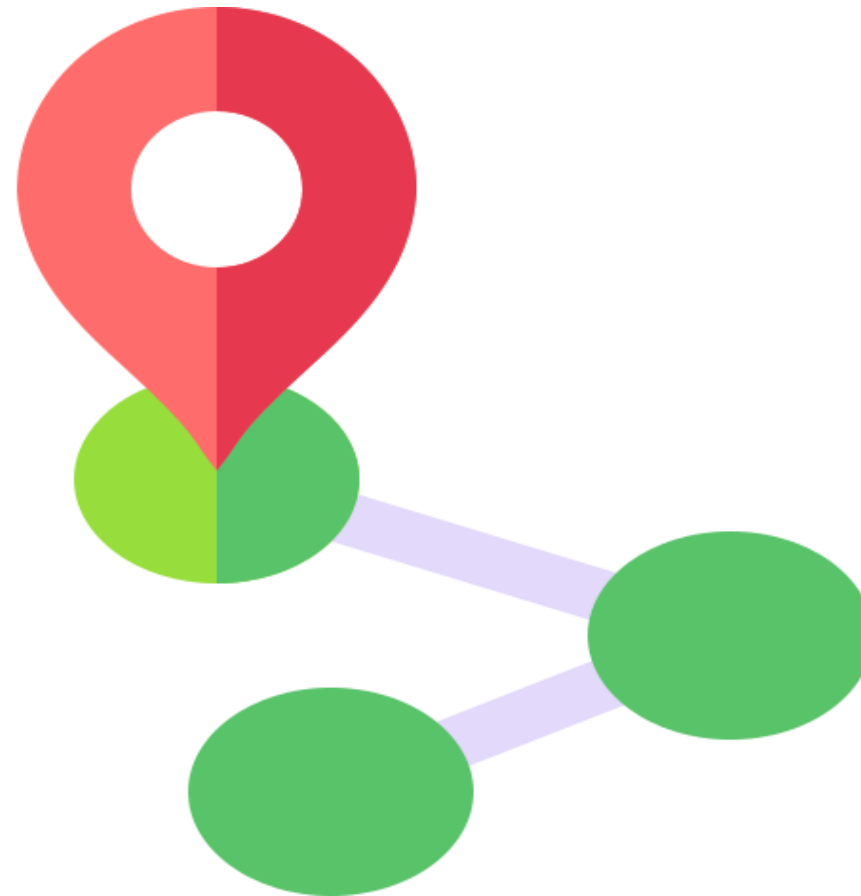


Nodes are the individual components of a document.



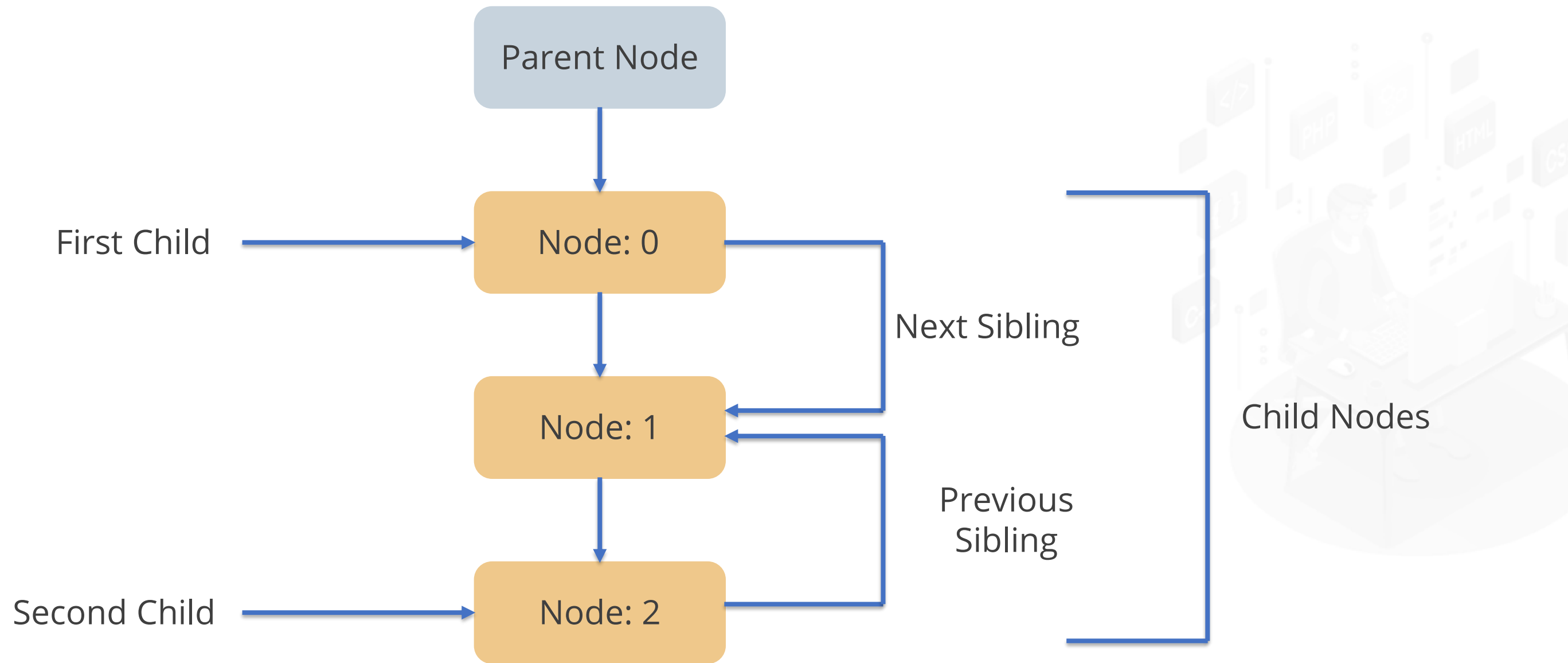
DOM Navigation

The DOM node contains a number of properties and methods which allow to navigate and modify the DOM tree structure.



DOM Navigation

JavaScript uses the following node attributes to traverse between nodes:



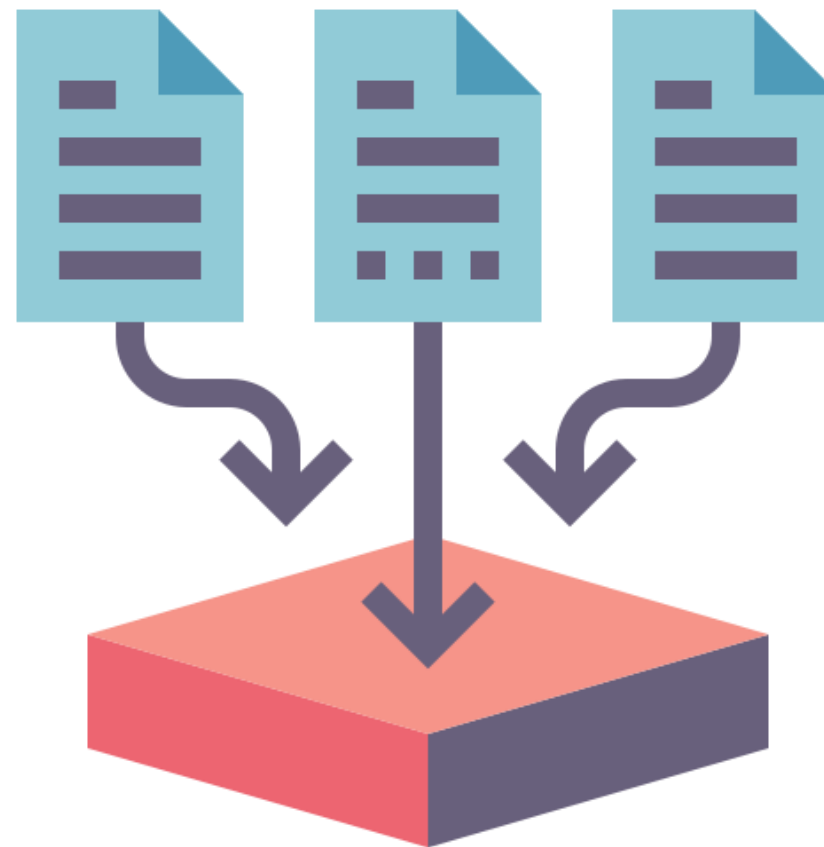
DOM Collections

Collections are groups of linked objects on a page, as defined by the document object model.



DOM Collections

The length property, item method, and named Item method are the only properties and methods that DOM collections have.



DOM Node Lists

A list (collection) of nodes derived from a document is a node list object. HTML collection object and a Node list object are nearly identical.

Example:

```
const myNodeList = document.querySelectorAll("p");
```



Working with DOM



Problem Statement:

You are asked to work with the Document Object Model.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to work with DOM are:

1. Work with DOM



Key Takeaways

- The Document Object Model (DOM) is a data representation of the objects that make up a document's structure and content.
- The document object has several properties and methods for accessing and manipulating the page's web elements.
- The document object model Events can be triggered by a user's interaction or the browser on any part of a document.
- The document object model (DOM) defines interfaces for managing XML and HTML documents.

