Front-end Development

Document Object Model

4

Fundamental Web Programming

Asst. Prof. Manop Phankokkruad, Ph.D.
School of Information Technology
King Mongkut's Institute of Technology Ladkrabang

KMITL

Outline

1. What is the Document Object Model ?
2. HTML DOM
3. DOM Programming Interface

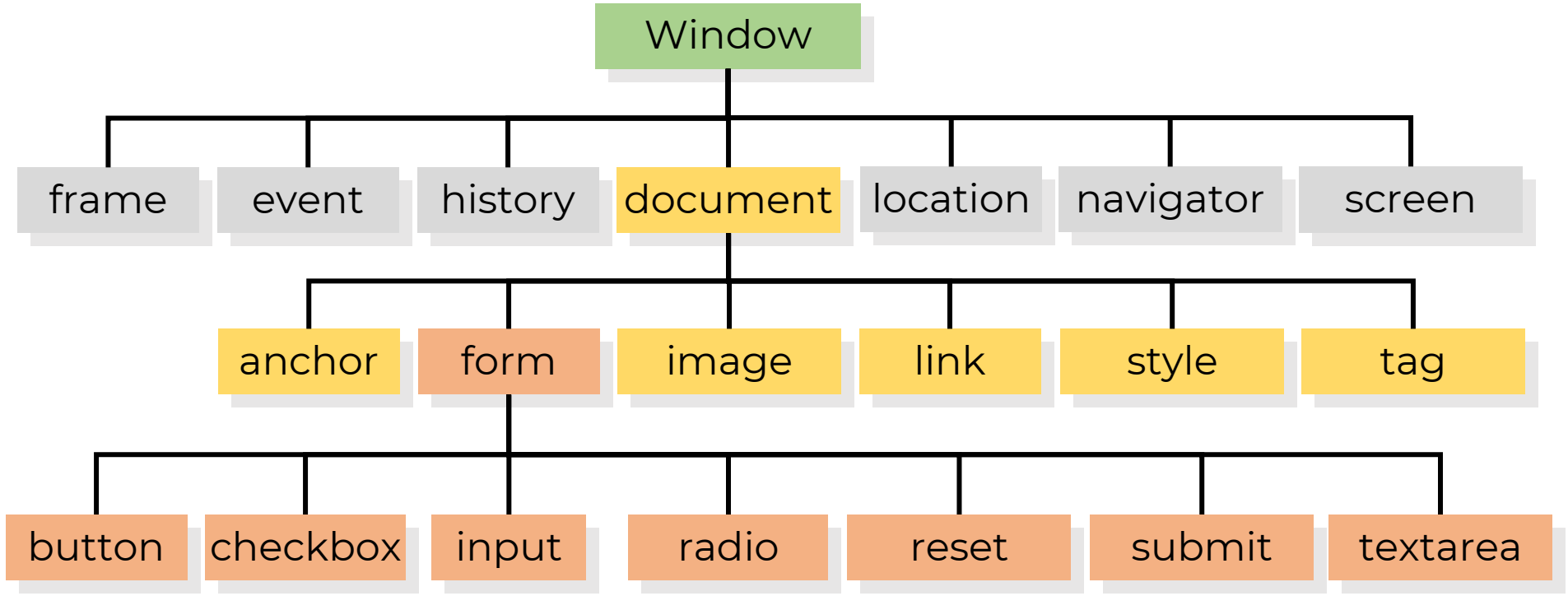
What is the DOM?

Document Object Model (DOM) is a cross-platform and language-neutral interface that treats a document as a tree structure wherein each node is an object representing a part of the document. The DOM represents a document with a logical tree. The DOM is a W3C standard. The W3C DOM standard is separated into 3 different parts:

- **Core DOM** - standard model for all document types.
- **XML DOM** - standard model for XML
- **HTML DOM** - standard model for HTML

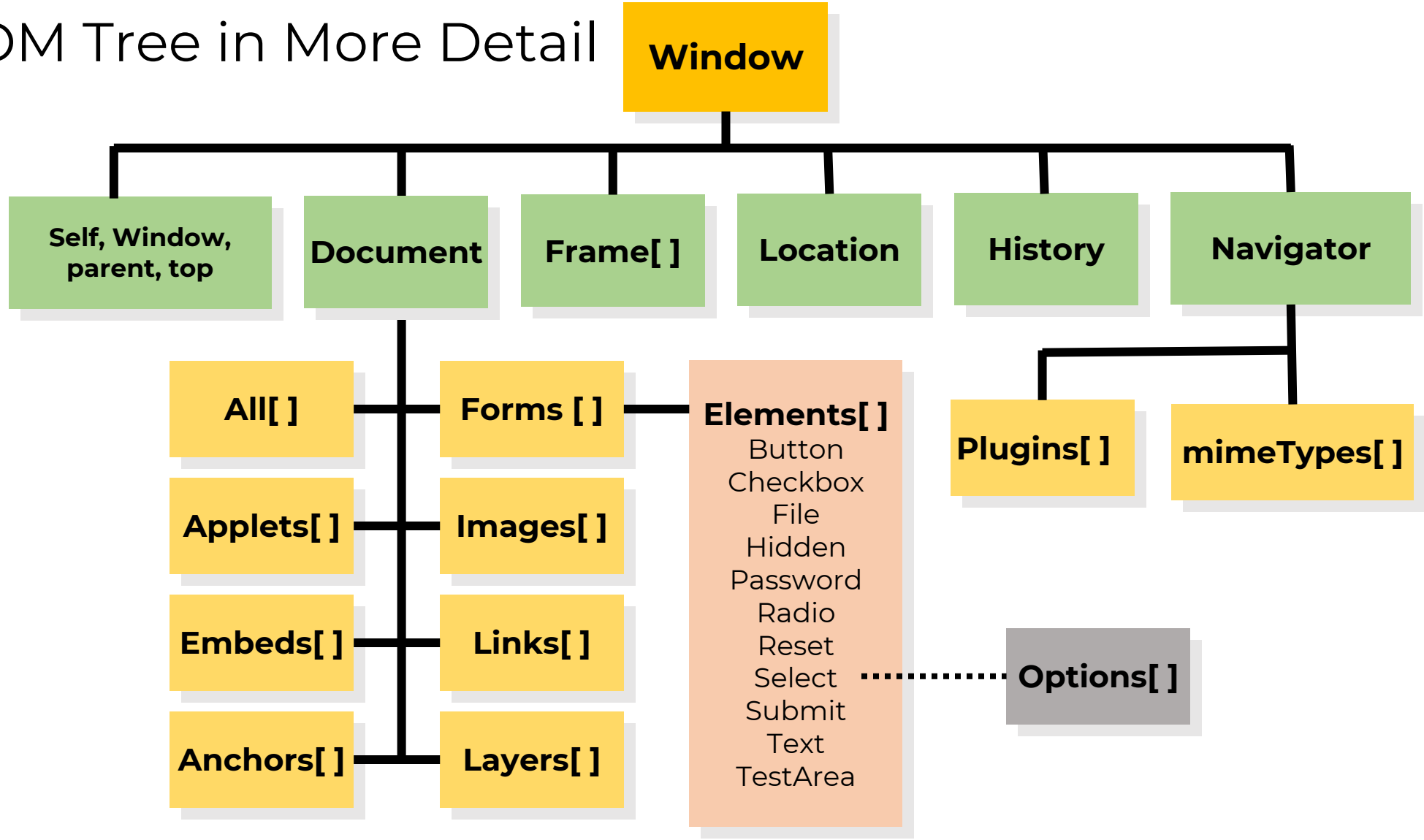
DOM hierarchy

DOM defines the logical structure of documents and the way a document is accessed and manipulated.



DOM hierarchy

DOM Tree in More Detail



DOM Levels

- **DOM Level 1** provided a complete model for an entire HTML or XML document, including the means to change any portion of the document.
- **DOM Level 2** introduced the getElementById function as well as an event model and support for XML namespaces and CSS.
- **DOM Level 3** added support for XPath and keyboard event handling, as well as an interface for serializing documents as XML.
- **DOM Level 4** is a snapshot of the WHATWG living standard.

HTML DOM

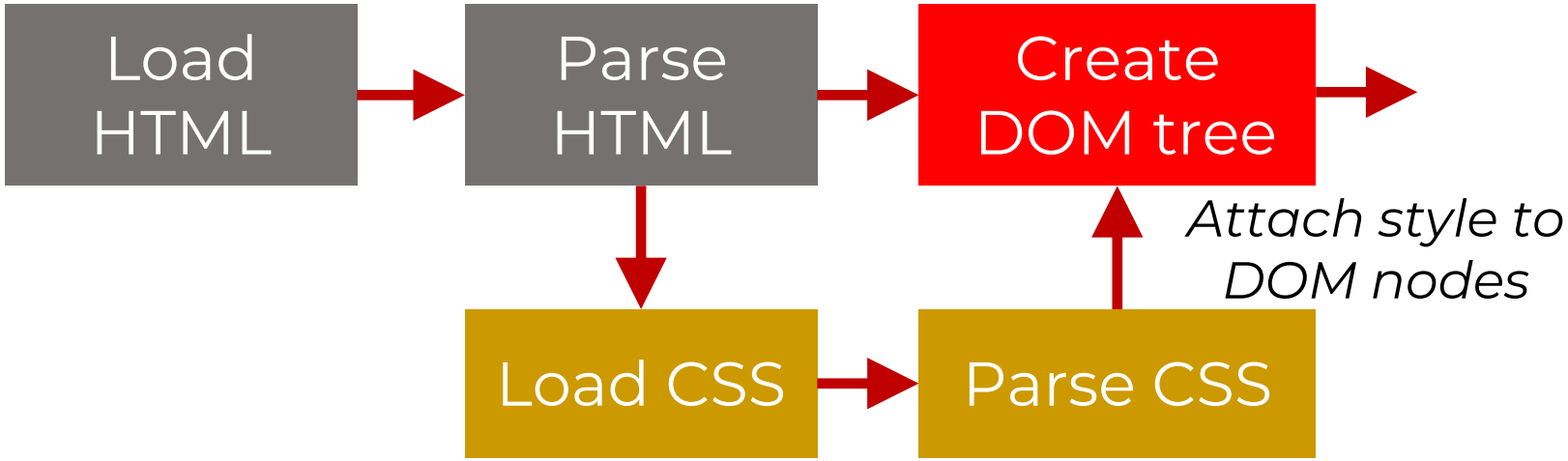
The **HTML DOM** is a standard object model and programming interface for HTML. It defines:

A. The HTML elements as objects
B. The attributes as properties
C. Assign the methods to access all HTML elements
D. Define the events for all HTML elements

- Every element on an HTML page is accessible in JavaScript through the DOM.
- The DOM is the tree of nodes corresponding to HTML elements on a page.

Browser and DOM

When a browser displays a document, it must combine the document's content with its style information. It processes the document in two stages.



1. The browser converts HTML and CSS into the DOM.
2. The browser displays the contents of the DOM.

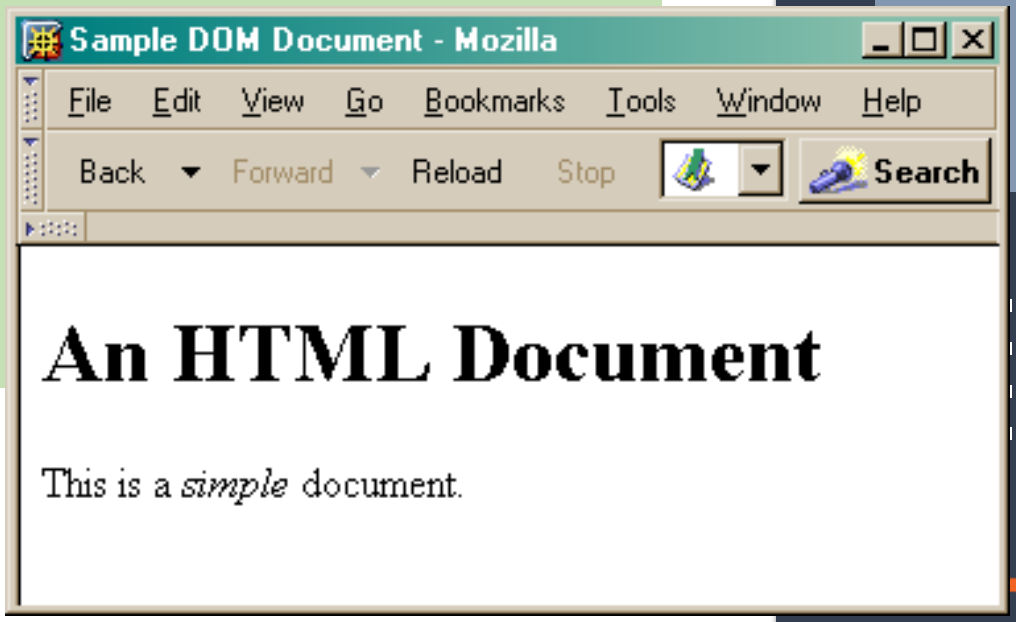
Browser and DOM

This is what the browser reads

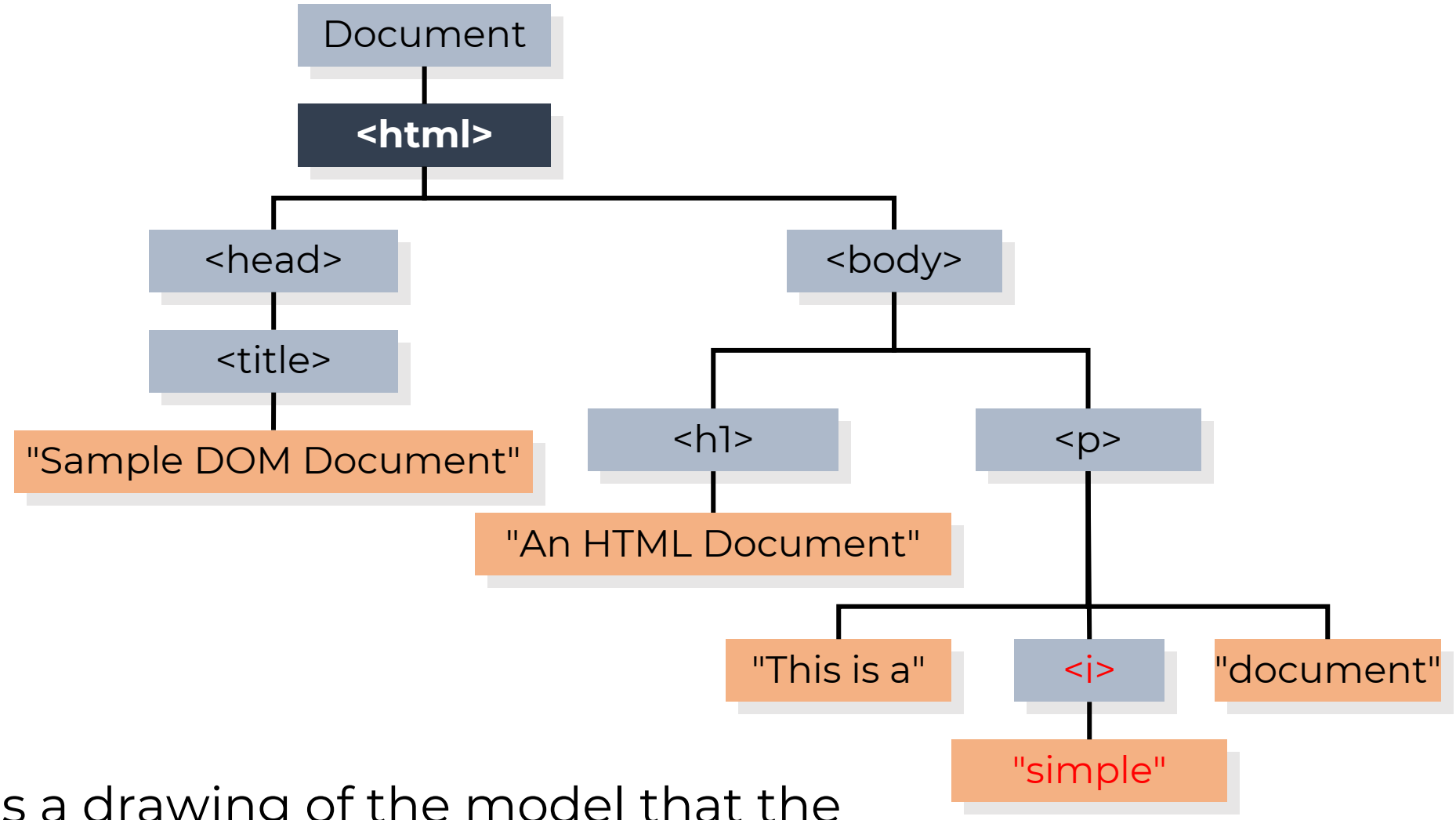
HTML

```
<html>
 <head>
  <title>Sample DOM Document</title>
 </head>
 <body>
  <h1>An HTML Document</h1>
  <p>This is a <i>simple</i> document.</p>
 </body>
</html>
```

This is what the browser displays on screen.



Browser and DOM



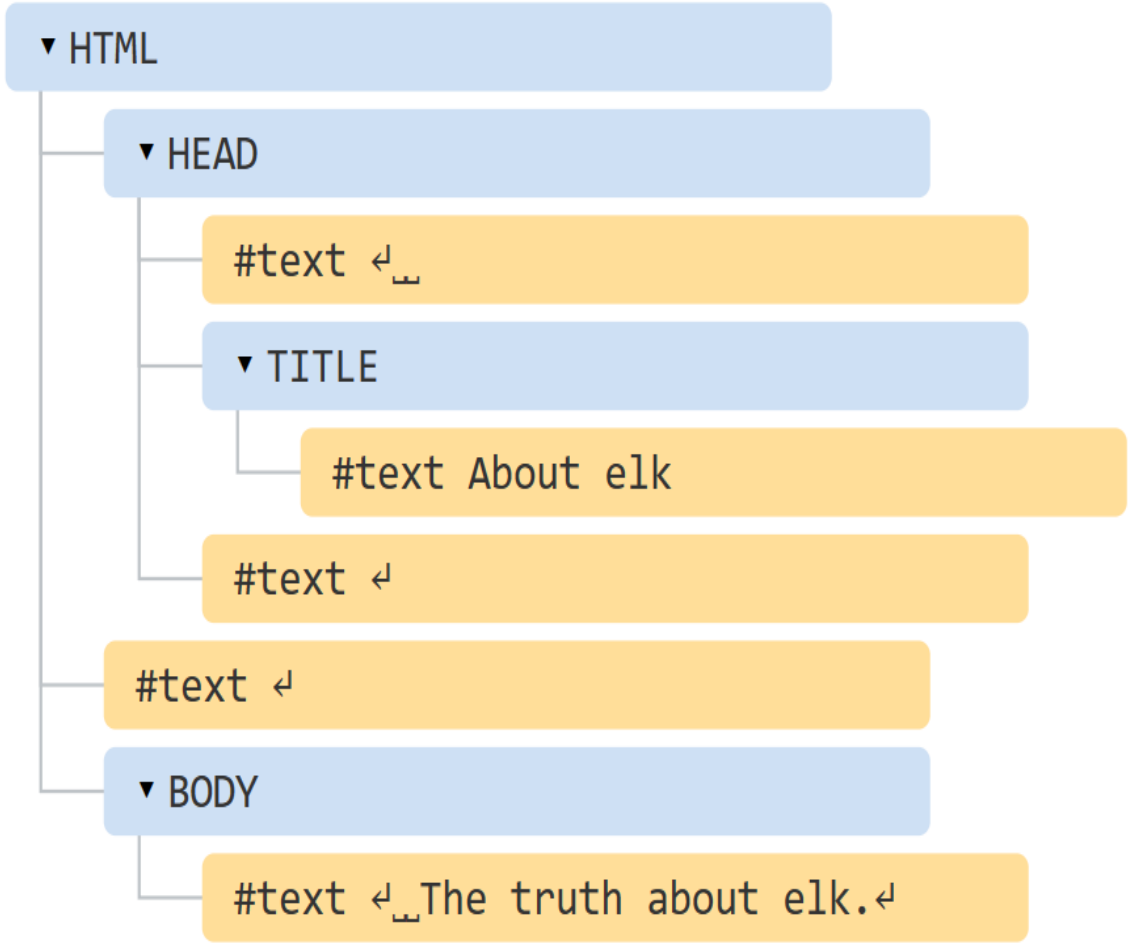
This is a drawing of the model that the browser is working with for the page.

Browser and DOM

This is what the browser reads

HTML

```
<!DOCTYPE HTML>
<html>
<head>
  <title>About elk</title>
</head>
<body>
  The truth about elk.
</body>
</html>
```



Types of DOM nodes

In the HTML DOM, everything is a node. The DOM represents documents as a hierarchy of Node objects. The main DOM node types are:

1. **Document node**
   - the start of the tree
2. **Element Node**
   - contains an HTML tag
   - can have element, text, and attribute child nodes.
3. **Attribute node**
   - Represents attribute of Element node.

KMITL

Types of DOM nodes

4. **Text Node**
   - contains text / textual content of an element.
   - cannot have child nodes or attributes.
   - contained within *Element Nodes*.
5. **Comment**
   - an HTML comment
6. **DocumentType**
   - the Doctype declaration

KMITL

Browser and DOM

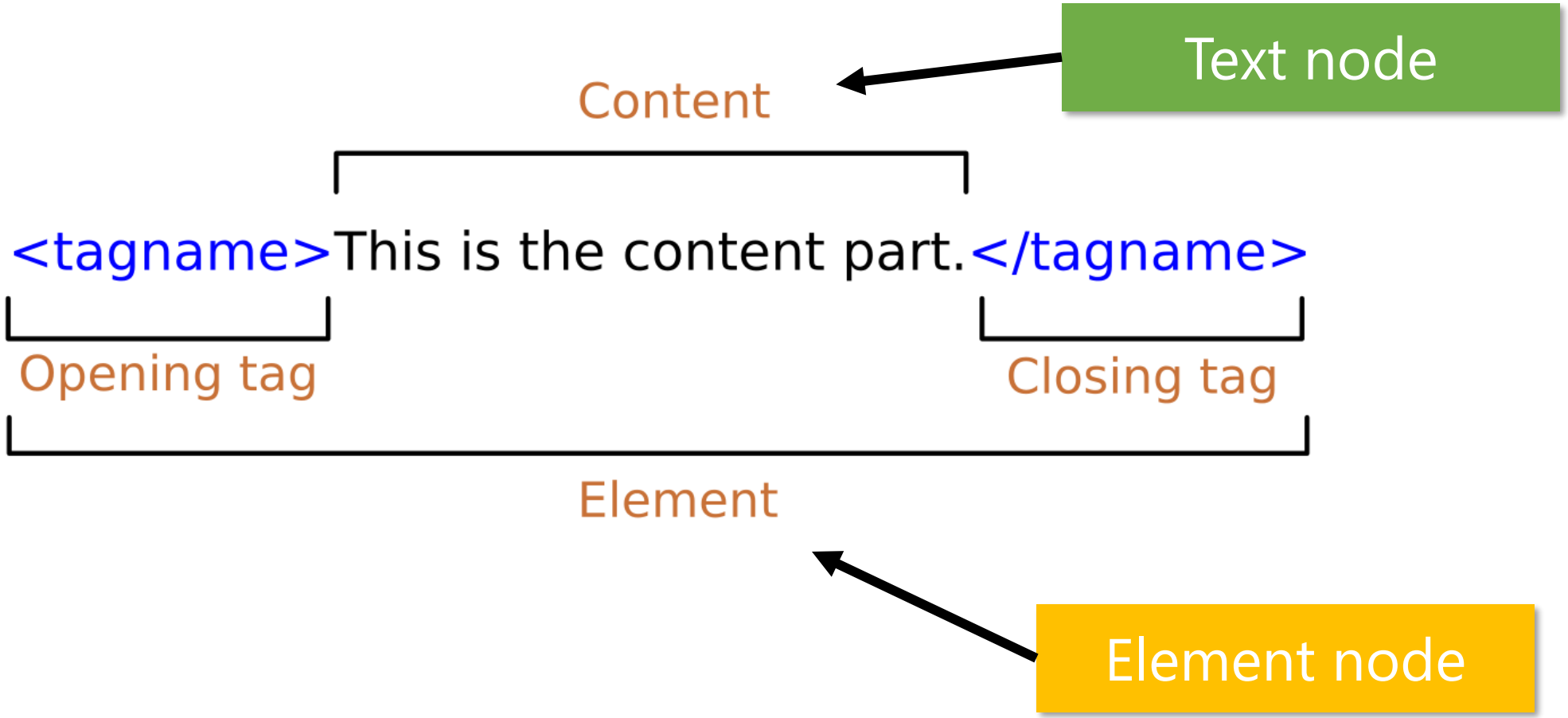
the browser reads DOM

HTML

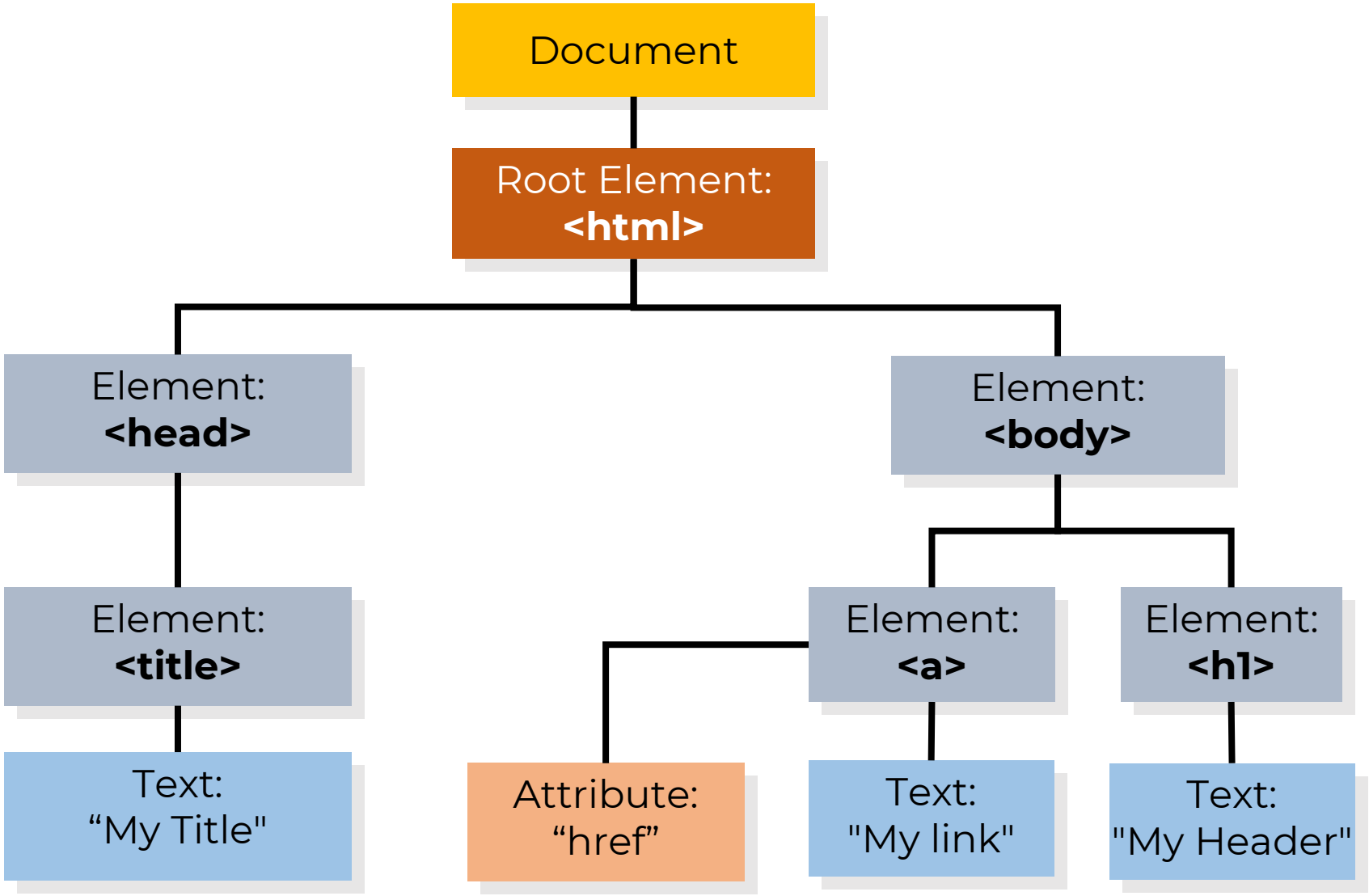
```
<!DOCTYPE HTML>
<html>
<body>
  The truth about elk.
  <ol>
    <li>An elk is a smart</li>
    <!-- comment -->
    <li>...and cunning animal!</li>
  </ol>
</body>
</html>
```

Types of DOM nodes

The fundamental of element tag



DOM Tree Structure

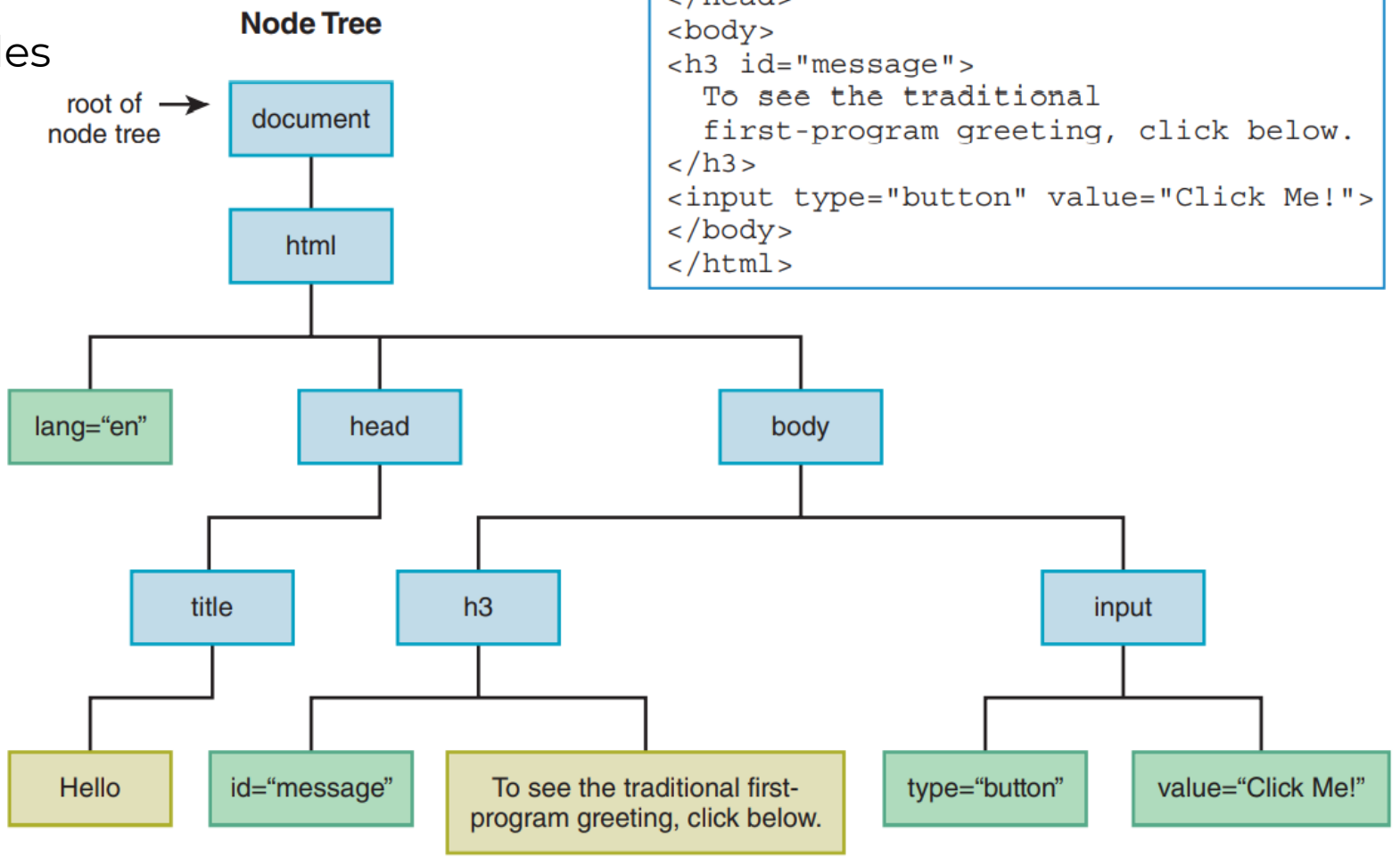


Browser and DOM

FIGURE: Node tree for web page

blue: element nodes
yellow: text nodes
green: attribute nodes

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Hello</title>
</head>
<body>
<h3 id="message">
  To see the traditional
  first-program greeting, click below.
</h3>
<input type="button" value="Click Me!">
</body>
</html>
```



Relationship among Nodes

- **Root node** : The topmost node of the tree is the root node. As it is topmost, so there is no parent of this root node.
- **Parent node** can have one or more than one *child nodes*.
- **Child node** : a node extending from another node.
- **Leaf** : The leaf nodes are the nodes which have no child node.
- **Siblings** : The nodes which have same parent are the siblings of each other.
- Every node has exactly one parent node (except root).

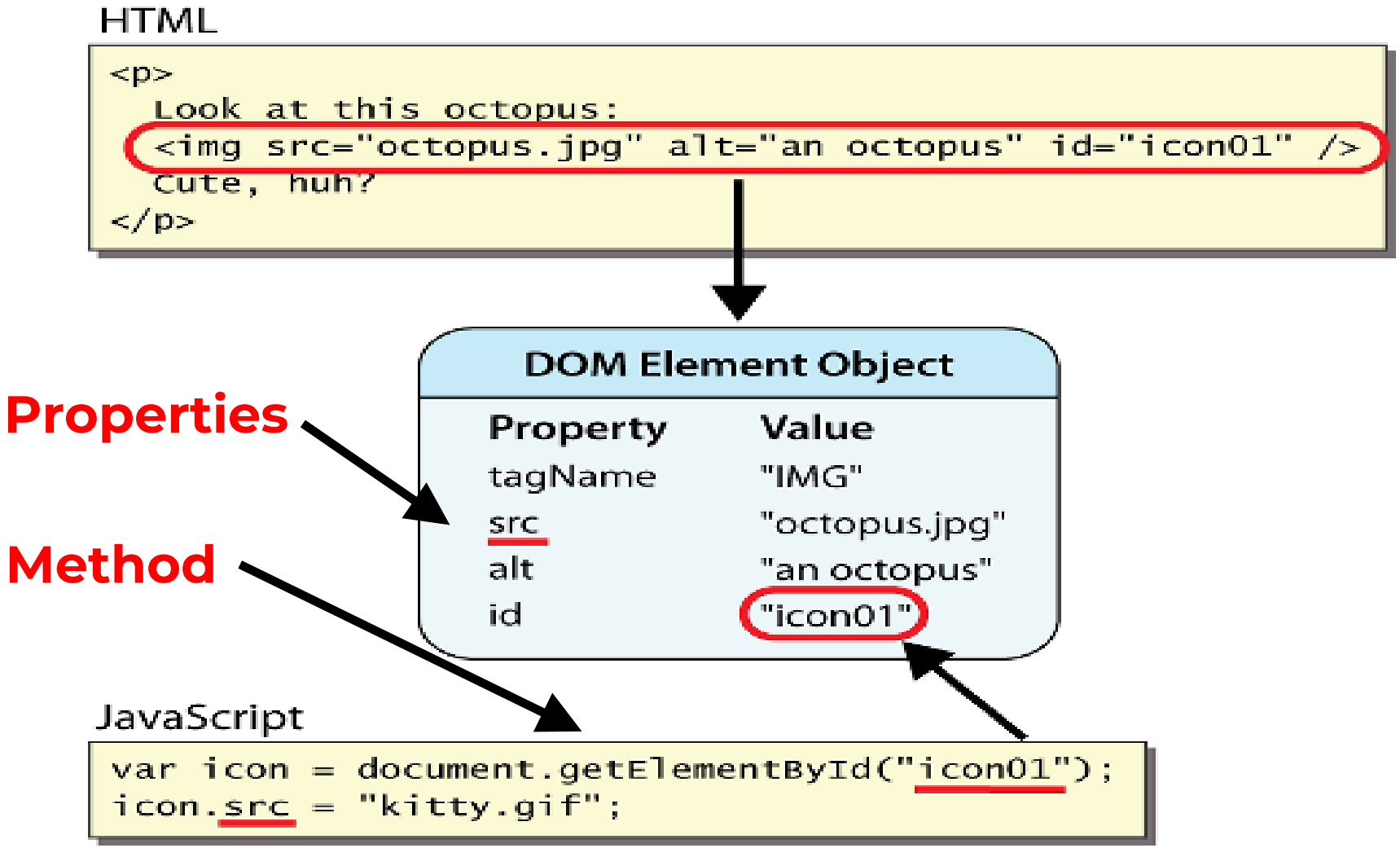
DOM Programming Interface

In the HTML DOM, all HTML elements are defined as objects. The HTML DOM can be accessed with JavaScript and other programming languages.

- The programming interface is the properties and methods of each object.
- A **property** is a value that one can get or set (like changing the content of an HTML element).
- A **method** is an action one can do (like adding or deleting an HTML element).

DOM Programming Interface

DOM Element Objects



Programming Interface

Some commonly used HTML DOM Methods:

- **getElementById**() - get the node with a specified id.
- **getElementByClassname**() - get the node with a specified classname.
- **getElementByName**() - get the node with a specified name.
- **getElementByTagName**() - get the node with a specified Tag name.
- **appendChild**() - insert a new child node.
- **removeChild**() - remove a child node.

KMITL

The Node object

Properties:

- **className** - list of CSS classes of element
- **innerHTML** – text content inside element, including Tags.
- **parentNode** - the parent node of a node
- **firstChild** - first child of node
- **childNodes** - the child nodes of a node
- **attributes** - the attributes nodes of a node

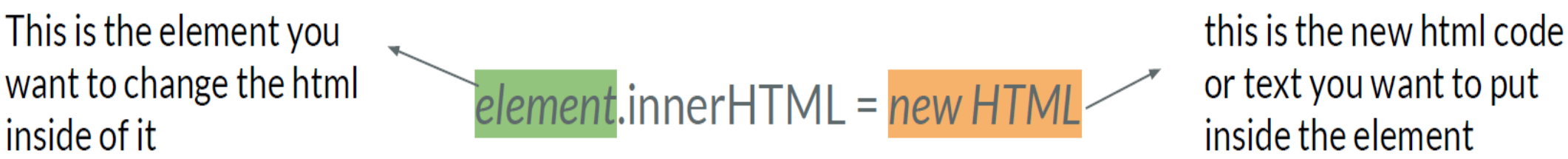
many more, some depending on type of node. These properties can be accessed and changed using JavaScript.

KMITL

Programming Interface

Changing HTML Elements

- The easiest way to modify the content of an HTML element is by using the innerHTML property.
- To change the content of an HTML element, use this syntax:




```
let header = document. getElementById("heading");
header.innerHTML = "My new heading";
```

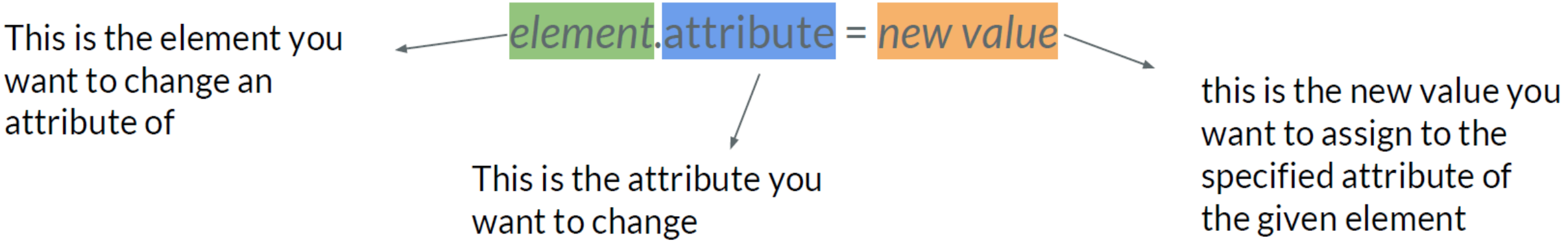
JavaScript

KMITL

Programming Interface

Changing HTML Elements

To change change the value of an HTML attribute.




```
let myLink = document. getElementById("myLink");
myLink.href = "http://www.newwebsite.com";
```

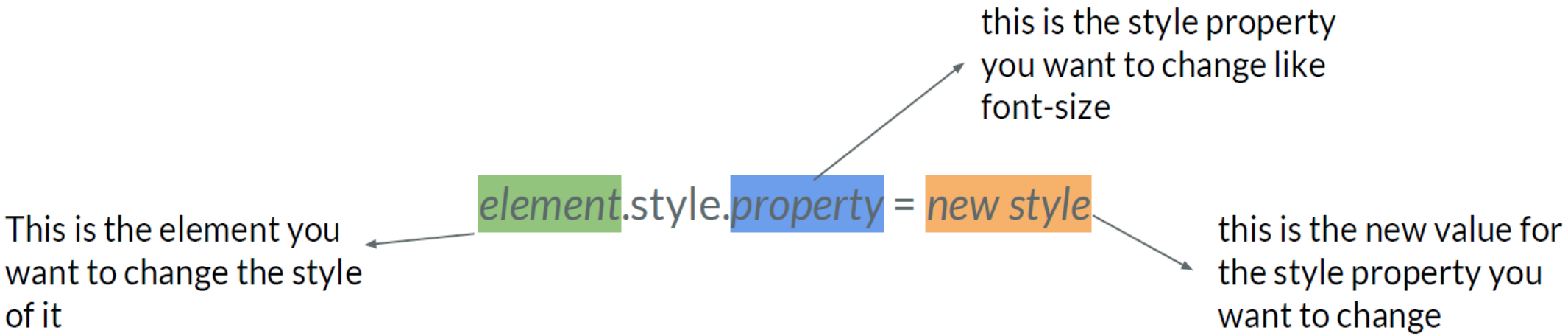
JavaScript

KMITL

Programming Interface

Changing CSS properties

To change the style of an HTML element, use this syntax:




```
let pars = document. getElementById("p1");
pars.style.fontSize = "2em";
```

JavaScript

Programming Interface

Adding HTML Elements

To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.

This creates the text that can go inside an html element. e.g. some text inside a <p> or <h1>

This is the name of the element you want to create e.g. "p"

```
document.createElement(element);
document.createTextNode(some text);
parentElement.appendChild(childElement);
```

This is the element you want to append the child element to

This is the child element you want to nest inside the parent element

KMITL

Programming Interface

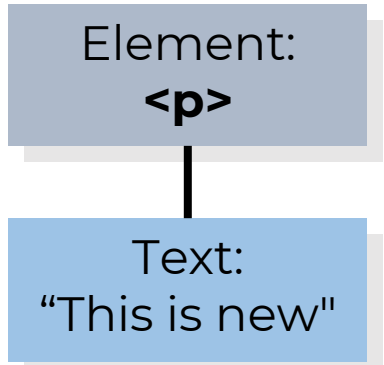
Adding a new HTML elements

HTML

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
<p>This is new.</p>      ← add this element
</div>
```

JavaScript

```
let para = document.createElement("p");
let tnode = document.createTextNode("This is new.");
para.appendChild(tnode);
let parentEle = document.getElementById("div1");
parentEle.appendChild(para);
```



Programming Interface

Inserting a new HTML elements

To use the **insertBefore()** method:

```
parentElement.insertBefore(newElement, existingElement)
```

This is the parent element you want to insert the new element inside it

This is the new element you want to insert inside the parent element and before the existing element

This is the existing element inside parent element, for which you want to insert the new element before it

KMITL

Programming Interface

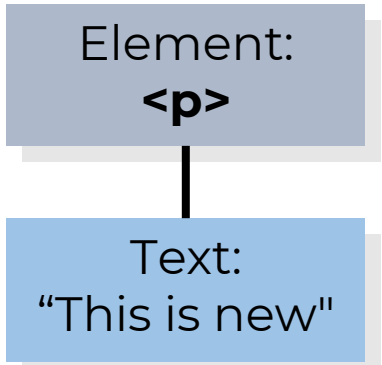
Inserting HTML Elements

HTML

```
<div id="div1">
<p>This is new.</p>
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>
```

```
let para = document.createElement("p");
let tnode = document.createTextNode("This is new.");
para.appendChild(tnode);
let parentEle = document.getElementById("div1");
let fchild = document.getElementById("p1");
parentEle.insertBefore(para, fchild);
```

JavaScript

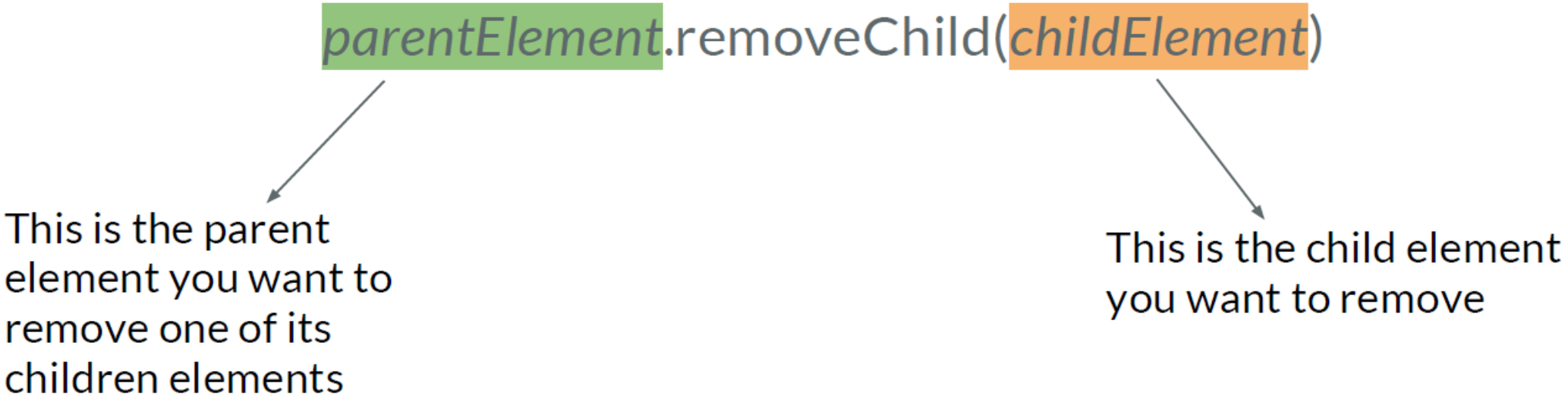


KMITL

Programming Interface

Removing the existing HTML elements

- To remove an HTML element, you must know the parent of the element
- Then you can use this syntax to remove the element you want:



Programming Interface

Removing the existing HTML Elements

HTML

```
<div id="div1">
<p id="p1">This is a paragraph.</p>  ← remove this element
<p id="p2">This is another paragraph.</p>
</div>
```

JavaScript

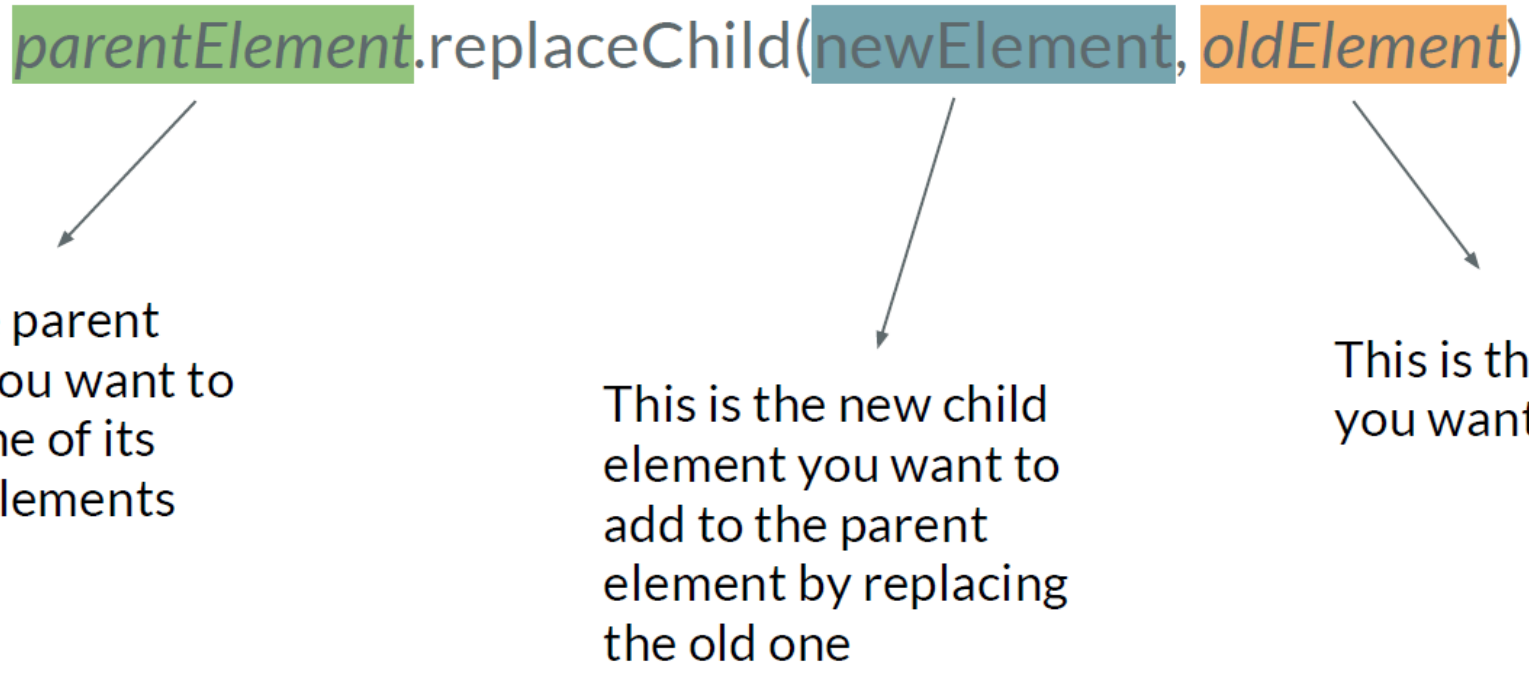
```
let parent = document.getElementById("div1");
let child = document. getElementById ("p1");
parent.removeChild(child);
```

KMITL

Programming Interface

Replacing HTML Elements

To replace an element, use the replaceChild() method:



Programming Interface

Replacing the existing HTML Elements

HTML

```
<div id="div1">
<p id="p1">First paragraph</p>  ← replace this element
<p id="p2">Second Paragraph</p>
</div>
```

JavaScript

```
let newPar = document.createElement("p");
let node = document.createTextNode("This is new.");
newPar.appendChild(node);
let parent = document. getElementById("div1");
let oldPar = document. getElementById("p1");
parent.replaceChild(newPar, oldPar);
```

KMITL

More Information

- JavaScript Tutorial
  https://www.w3schools.com/js/default.asp
- XML DOM Tutorial
  https://www.w3schools.com/xml/dom_intro.asp

- XML DOM Tutorial
  https://www.tutorialspoint.com/dom/