Document Object Model

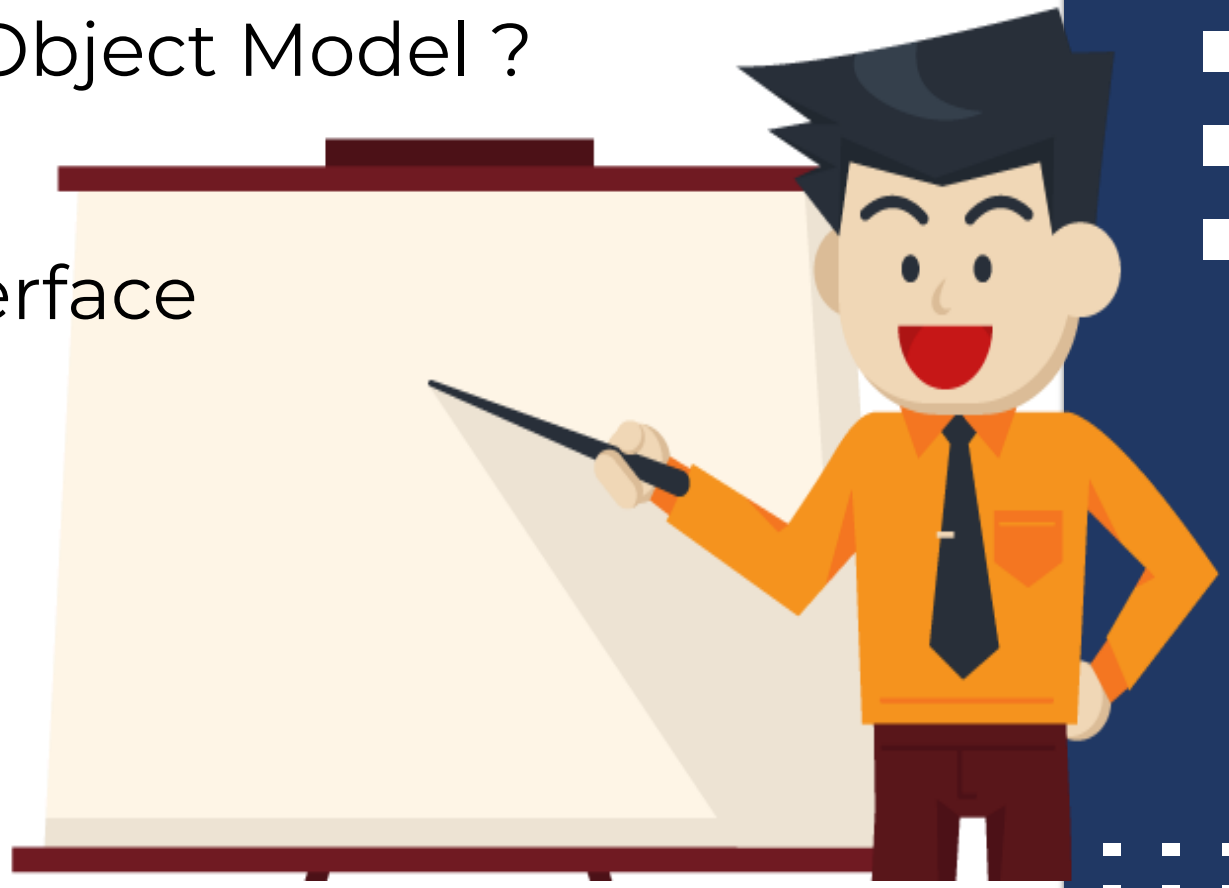
5

Web Technology

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

Outline

- What is the Document Object Model ?
- HTML DOM
- DOM Programming Interface

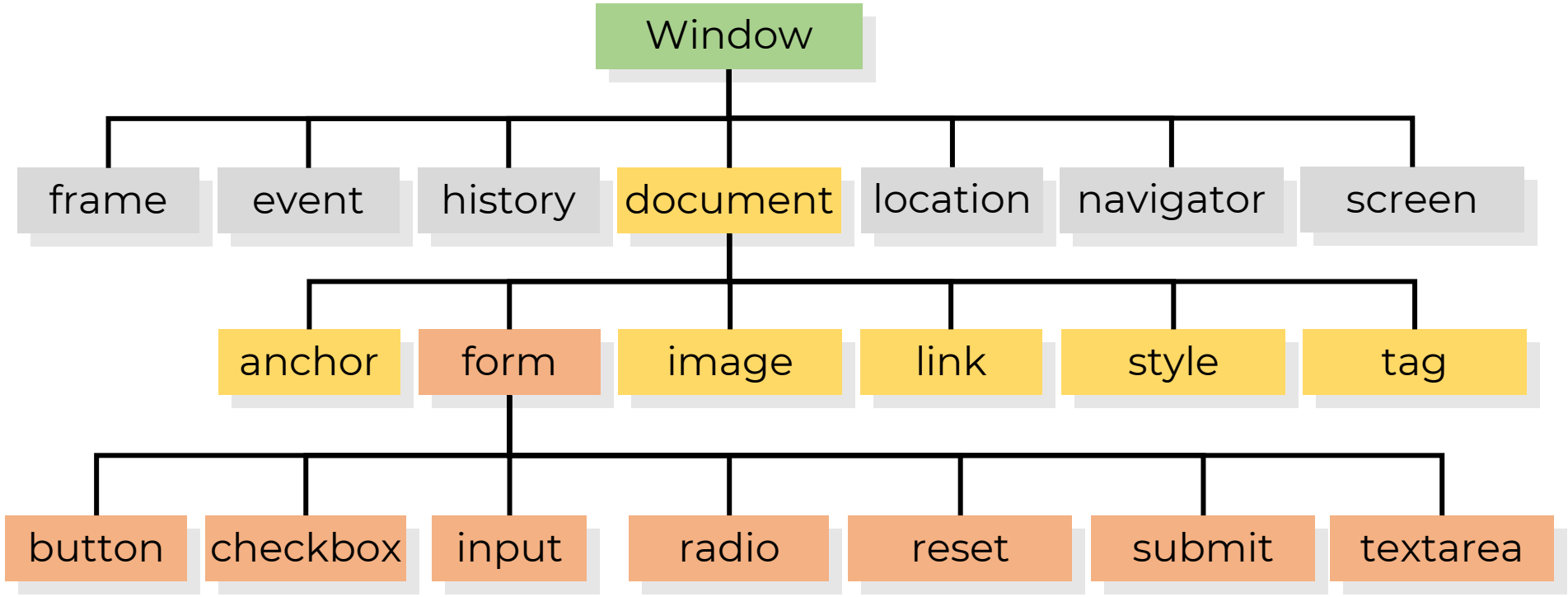


What is the DOM?

- **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.



What is the DOM?

DOM gives we access to all the elements on a web page. Using JavaScript, we can create, modify and remove elements in the page dynamically. With the DOM,

- HTML elements can be treated as *objects*
- Many *attributes of HTML elements* can be treated as *properties* of those objects.
- Then, objects can be scripted (through their id attributes) with JavaScript to achieve dynamic effects.

What is the DOM?

- The web browser builds a *model* of the web page (the *document*) that includes all the *objects* in the page (tags, text, etc).
- All of the *properties*, *methods*, and *events* available to the web developer for manipulating and creating web pages are organized into objects.
- Those objects are accessible via scripting languages in modern web browsers.

DOM Levels

DOM Level 1

Interfaces for representing an XML and HTML document.

1. Document
2. Node
3. Attribute
4. Element
5. Text interfaces.

DOM Levels

DOM Level 2

It contains six different specifications:

1. DOM2 Core
2. Views
3. Events
4. Style
5. Traversal and Range
6. DOM2 HTML.

DOM Level 3

It contains five different specifications:

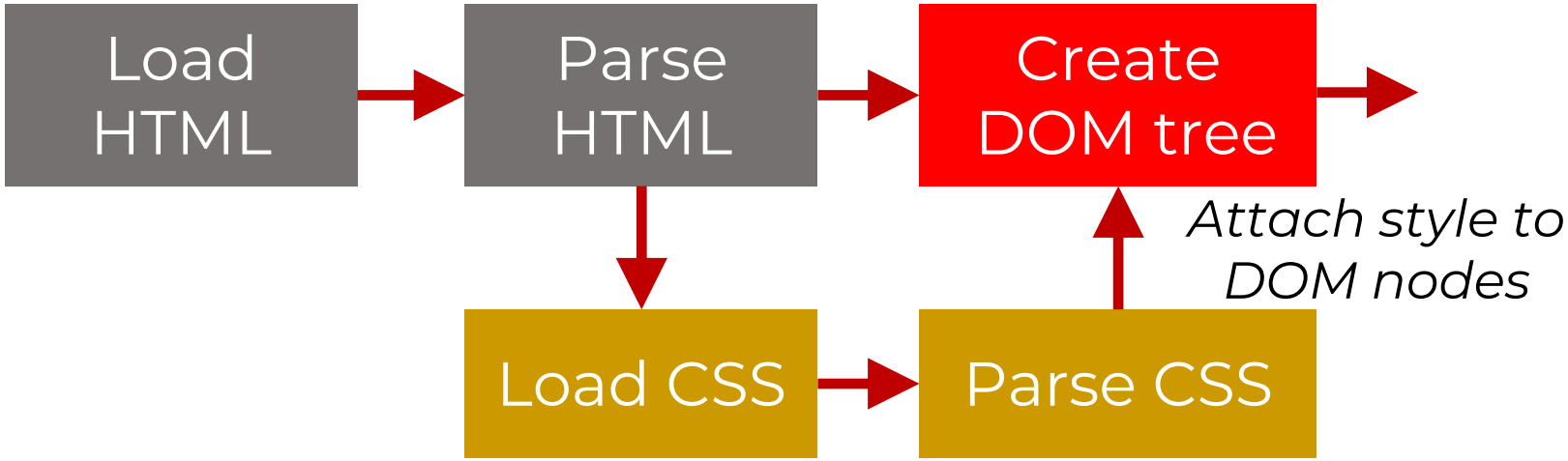
1. DOM3 Core
2. Load and Save
3. Validation
4. Events
5. XPath

HTML DOM

- The HTML DOM is a standard object model and programming interface for HTML. It defines:
  - The HTML elements as objects
  - The properties of all HTML elements
  - The methods to access all HTML elements
  - 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.

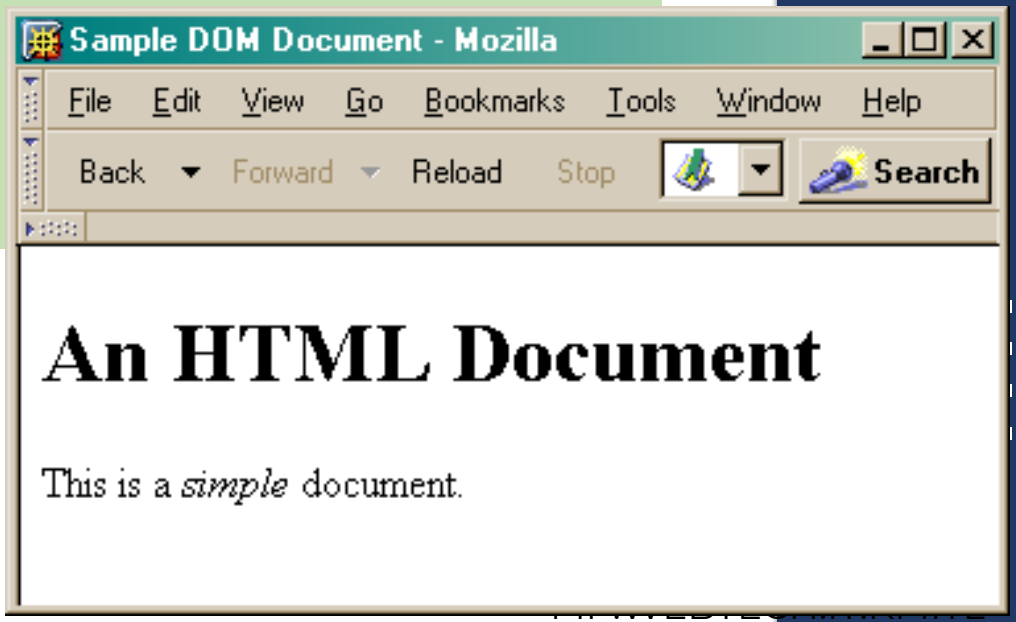
Browser and DOM

This is what the browser reads

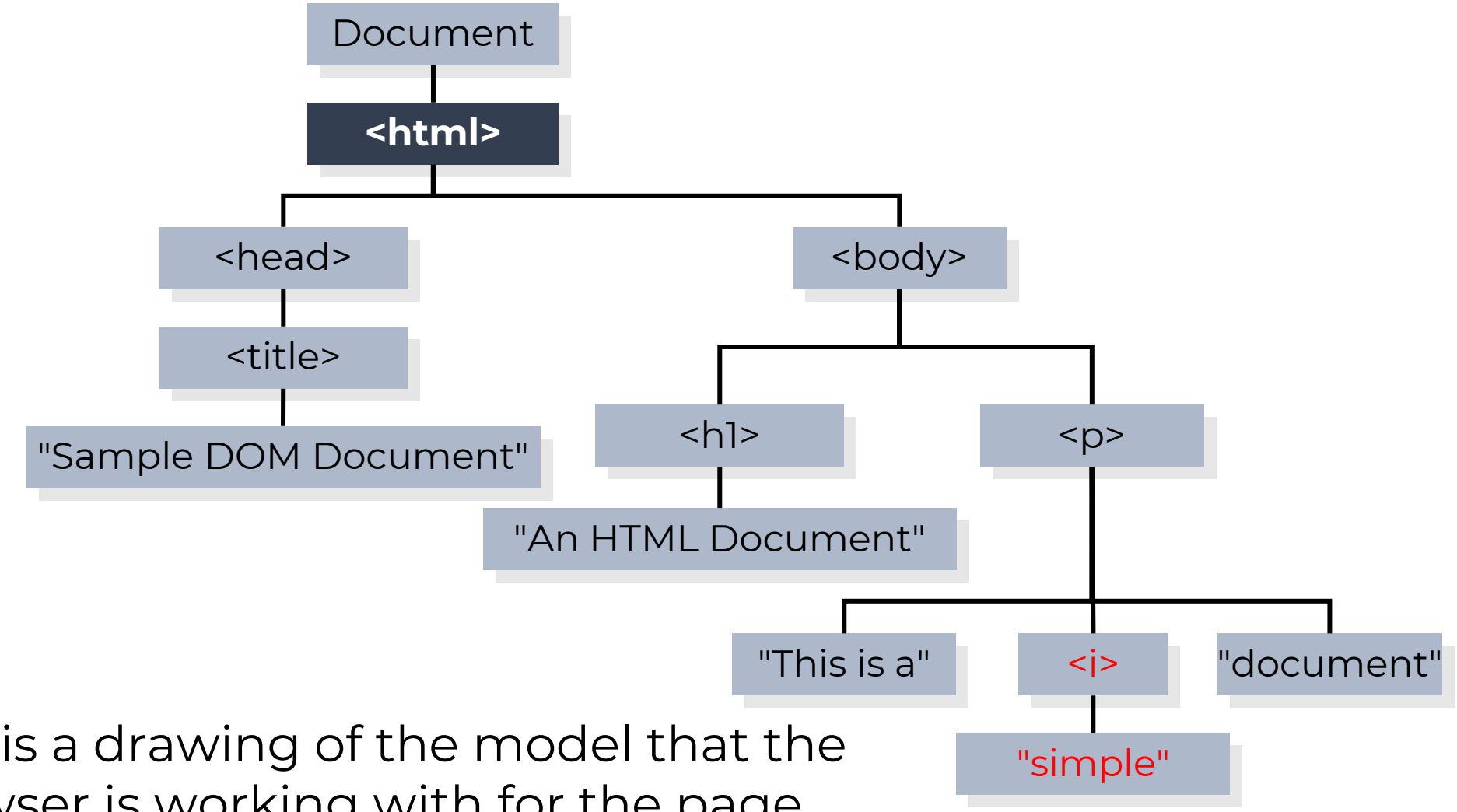
HTML

```
<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.

Types of DOM nodes

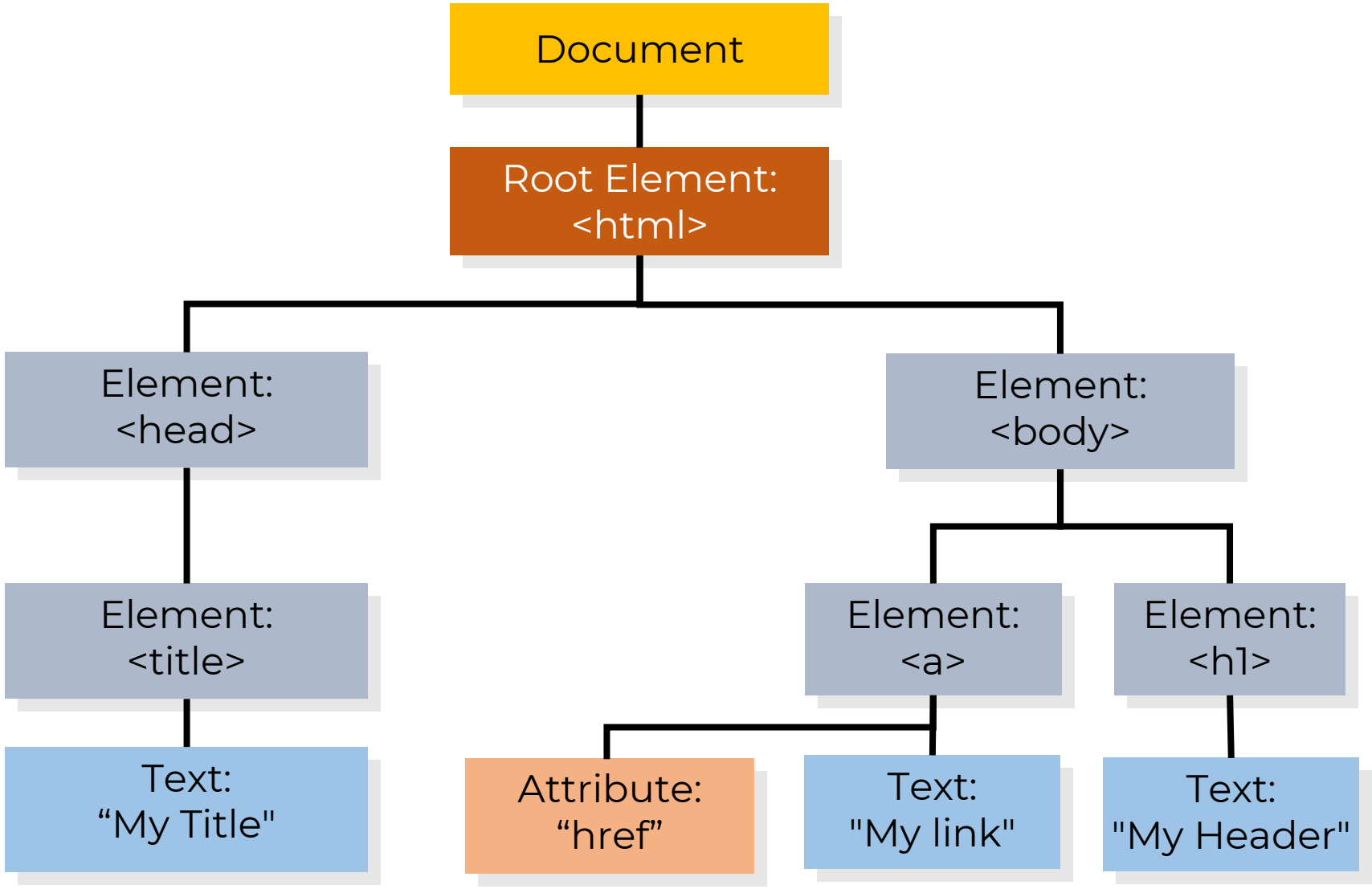
In the HTML DOM, everything is a node. The DOM represents documents as a hierarchy of Node objects. The main DOM node types are:

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.

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

DOM Tree Structure



Relationship among Nodes

- Every node has exactly one parent node (except root).
- **Parent node** can have one or more than one *child 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.
- **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.

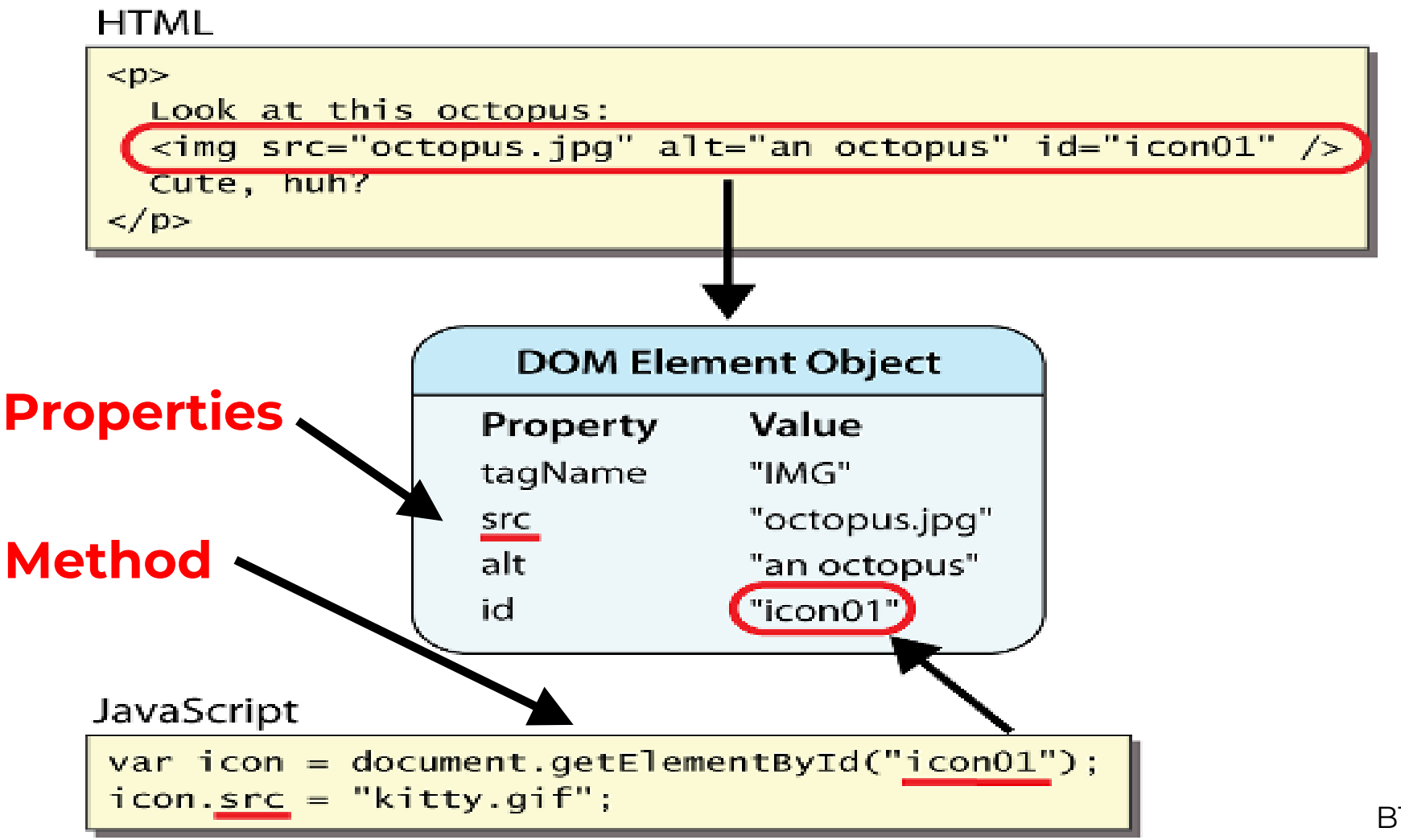
DOM Programming Interface

3

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

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.

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:

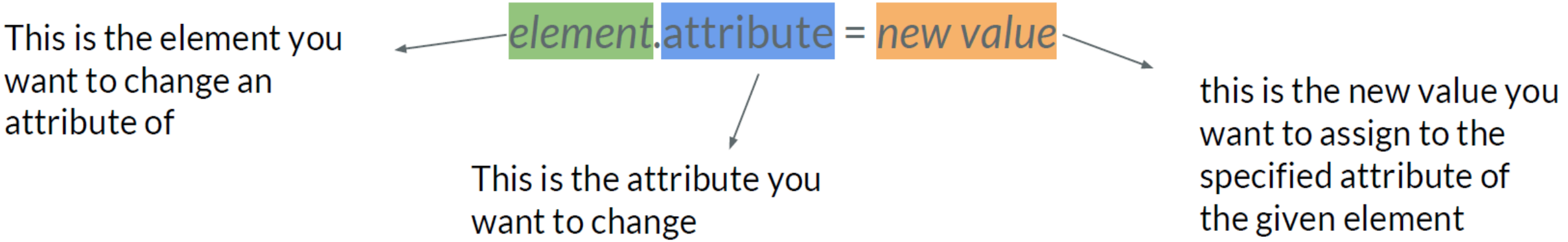

```
let header = document.querySelector("h1");
header.innerHTML = "My new heading";
```

HTML

Programming Interface

Changing HTML Elements

- You can also change the value of an HTML attribute.



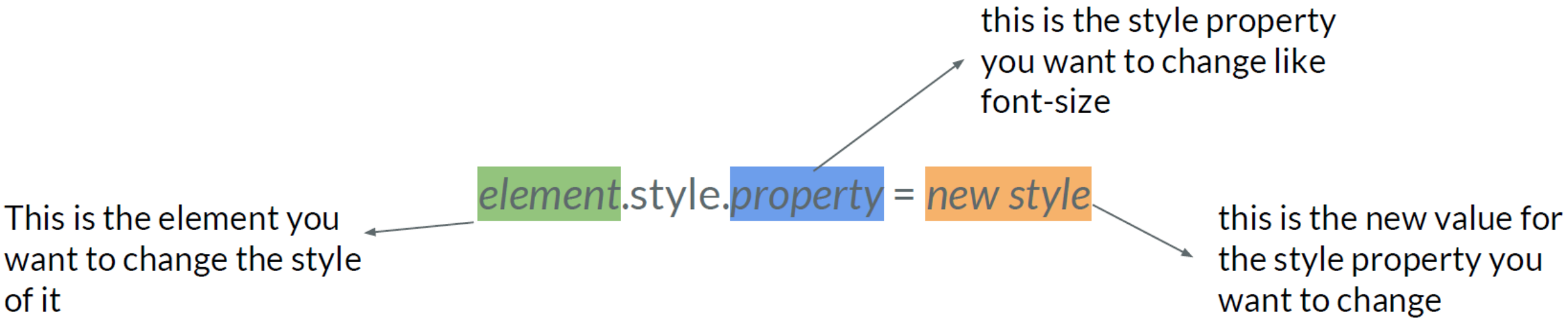

```
let myLink = document.querySelector("#myLink");
myLink.href = "http://www.newwebsite.com";
```

HTML

Programming Interface

Changing CSS properties

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




```
let pars = document.querySelectorAll("p.par");
pars[1].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

Programming Interface

Adding 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>
</div>
```

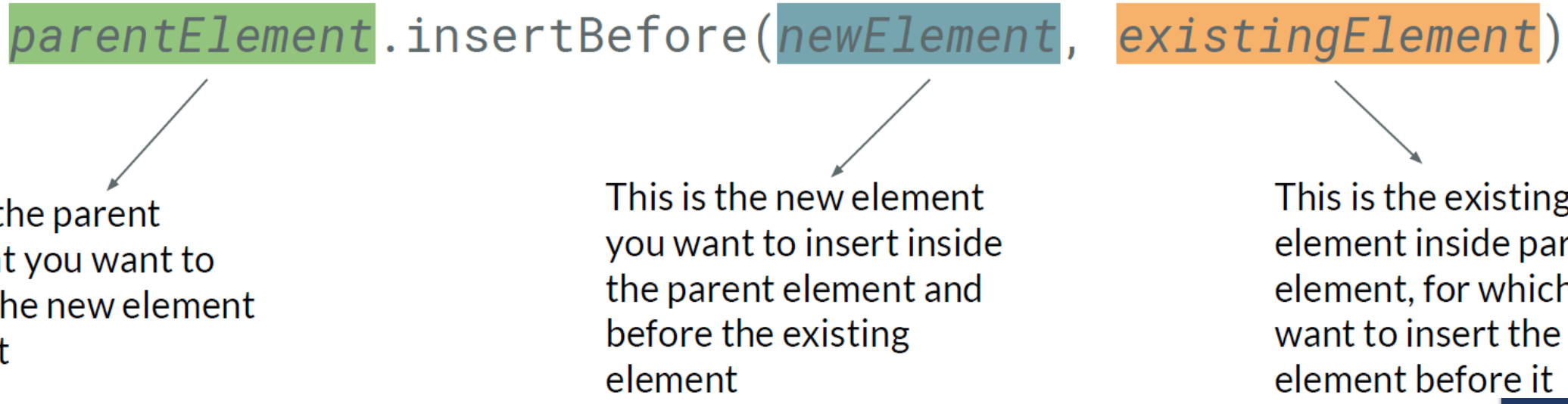
JavaScript

```
let para = document.createElement("p");
let node = document.createTextNode("This is new.");
para.appendChild(node);
let element = document.querySelector("#div1");
element.appendChild(para);
```

Programming Interface

Adding HTML Elements

- If you don't want that you can use the insertBefore() method:



Programming Interface

Adding 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>
```

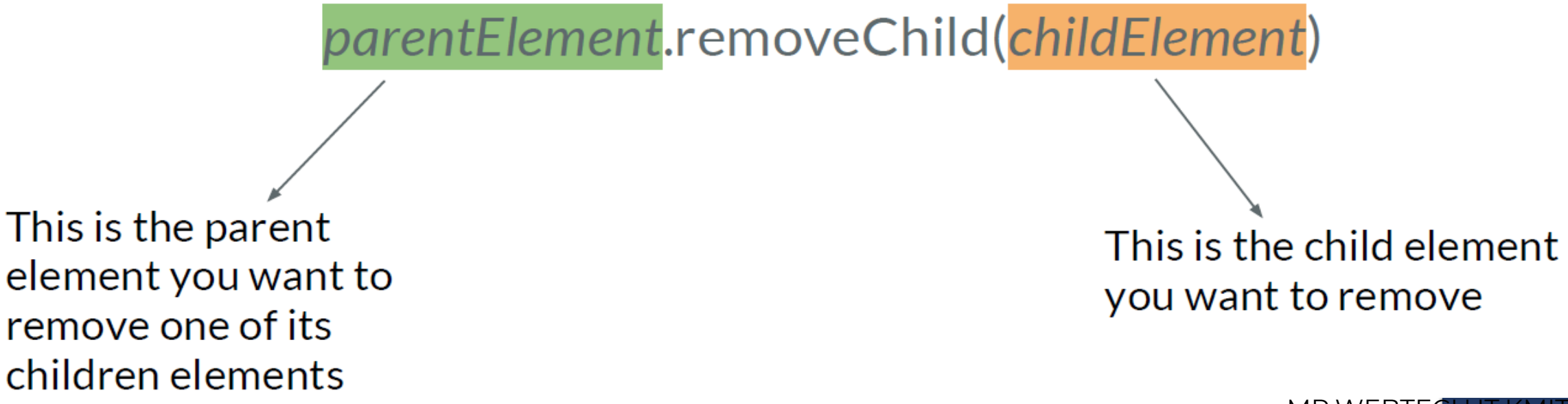
JavaScript

```
let para = document.createElement("p");
let node = document.createTextNode("This is new.");
para.appendChild(node);
let element = document.querySelector("#div1");
let child = document.querySelector("#p1");
element.insertBefore(para, child);
```

Programming Interface

Removing 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 Existing HTML Elements

HTML

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

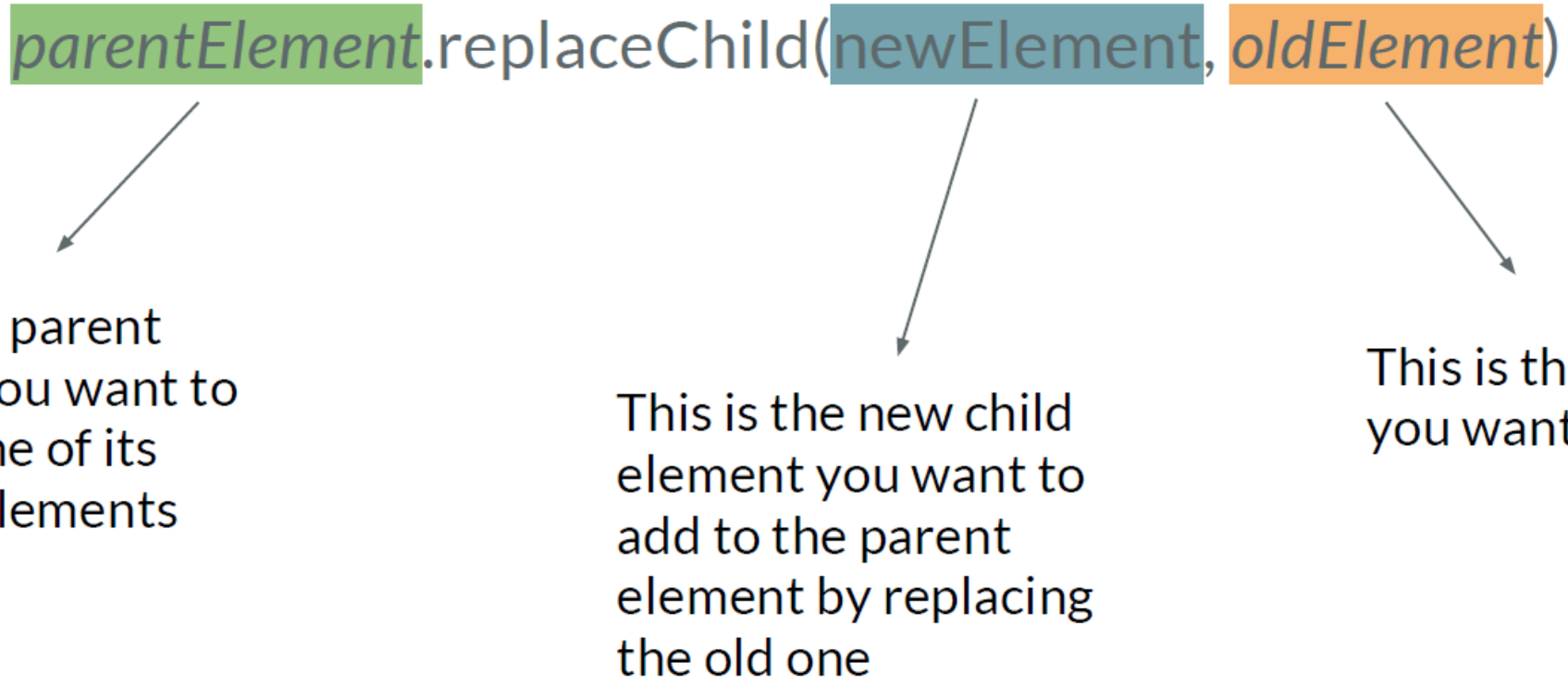
JavaScript

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

Programming Interface

Replacing HTML Elements

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



Programming Interface

Removing Existing HTML Elements

HTML

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

JavaScript

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

Programming Interface

Getting the parent

Every element has just one parent. To get it, you can use Node.parentNode or Node.parentElement.

- parentNode returns the parent of the specified node in the DOM tree.
- parentElement returns the DOM node's parent Element, or null if the node either has no parent, or its parent isn't a DOM Element.

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/