SEC201.2 Web-Based Programming

JavaScript: Advanced Topics - II

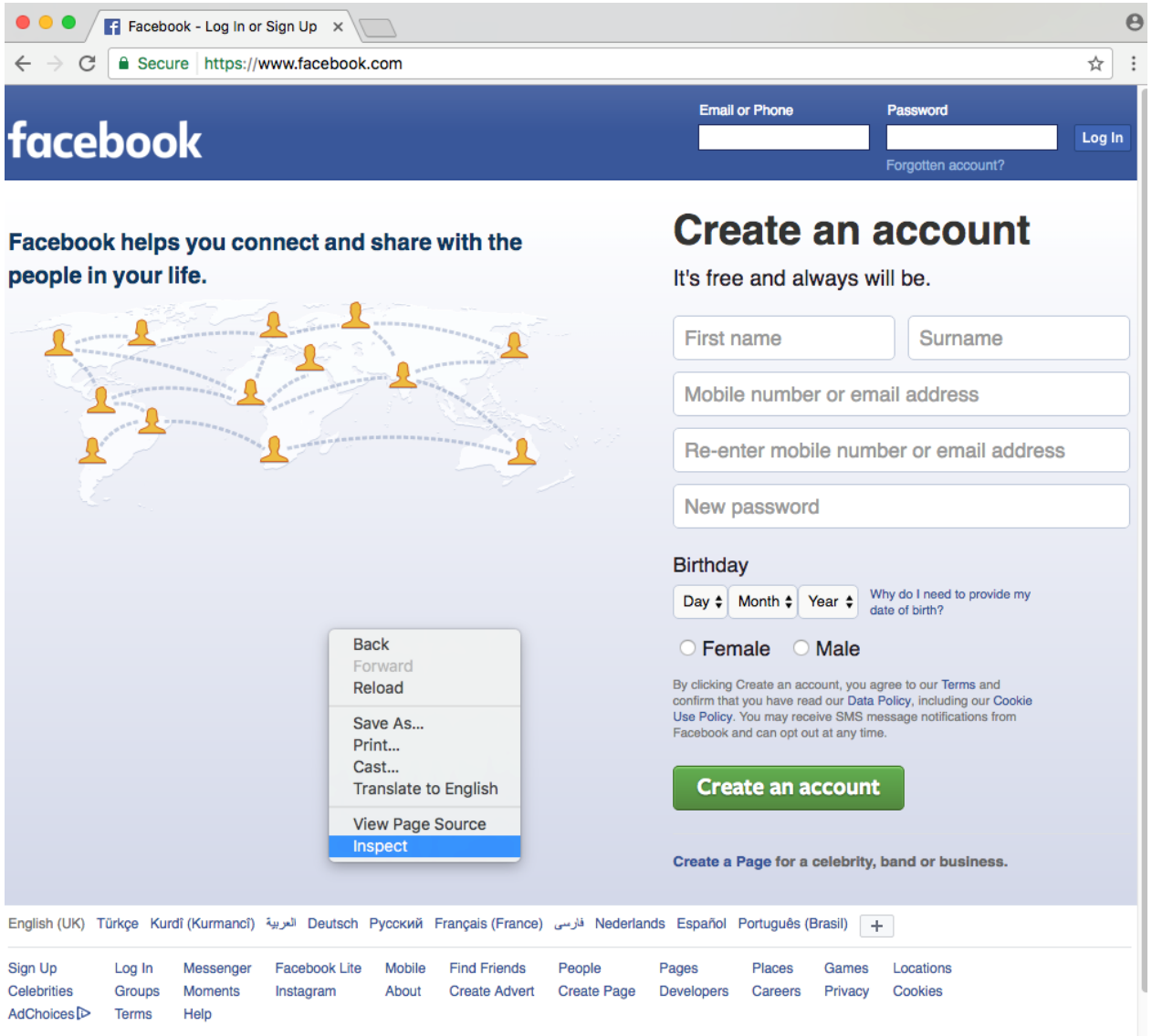
Outline

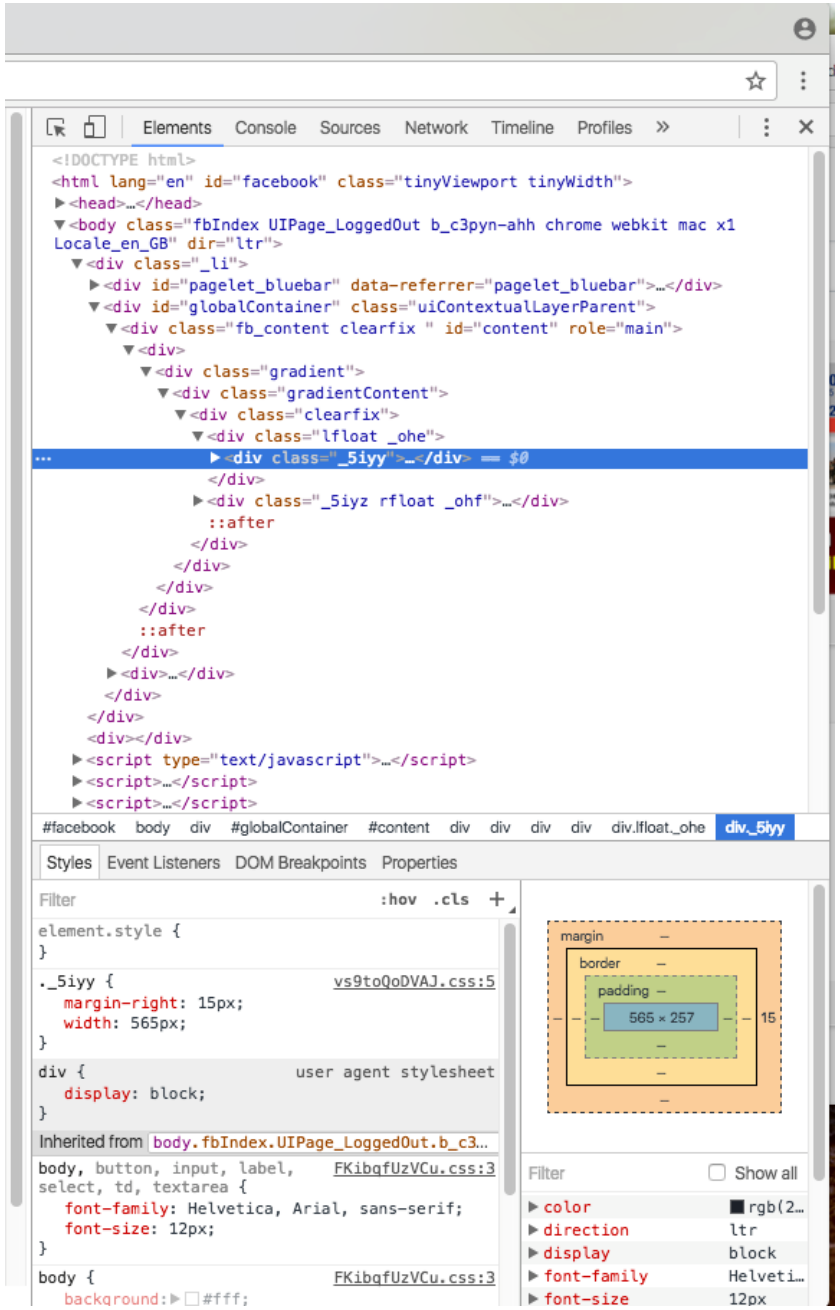
- The DOM
  - Basic Concepts
  - Node Relationships
  - Locating Nodes
  - Creating and Adding Nodes
  - Deleting Nodes
  - Cloning Nodes

The DOM

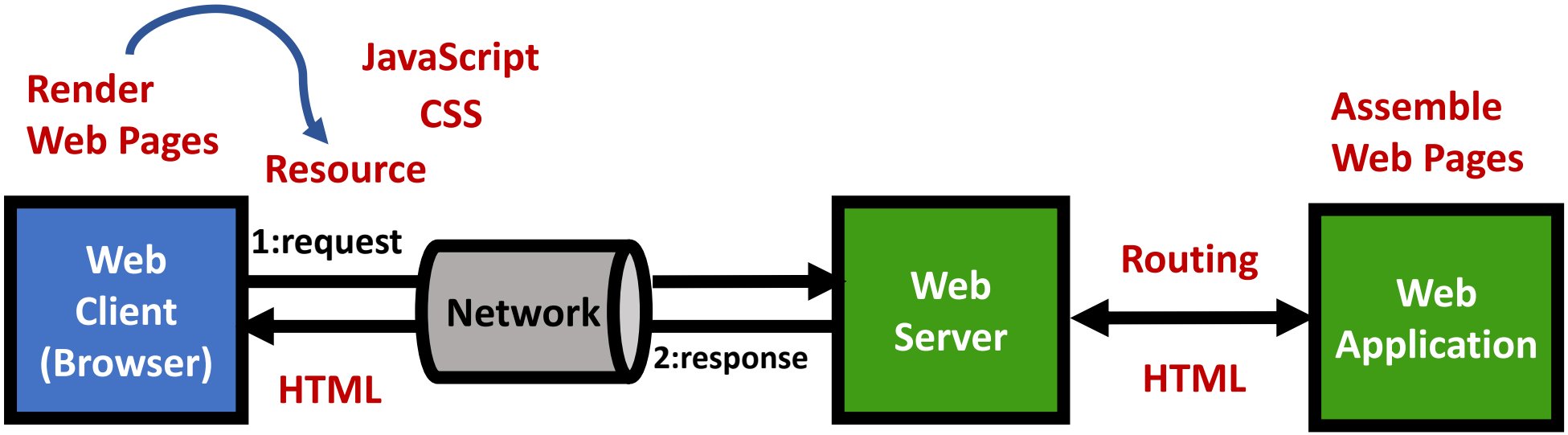
- DOM means ***Document Object Model***
- When you load something into a browser, it is converted into a DOM structure
  - Whatever you load
    - You load HTML → it gets converted into this DOM
    - You load XML → it gets converted into this DOM

Inspect the DOM with Google Chrome

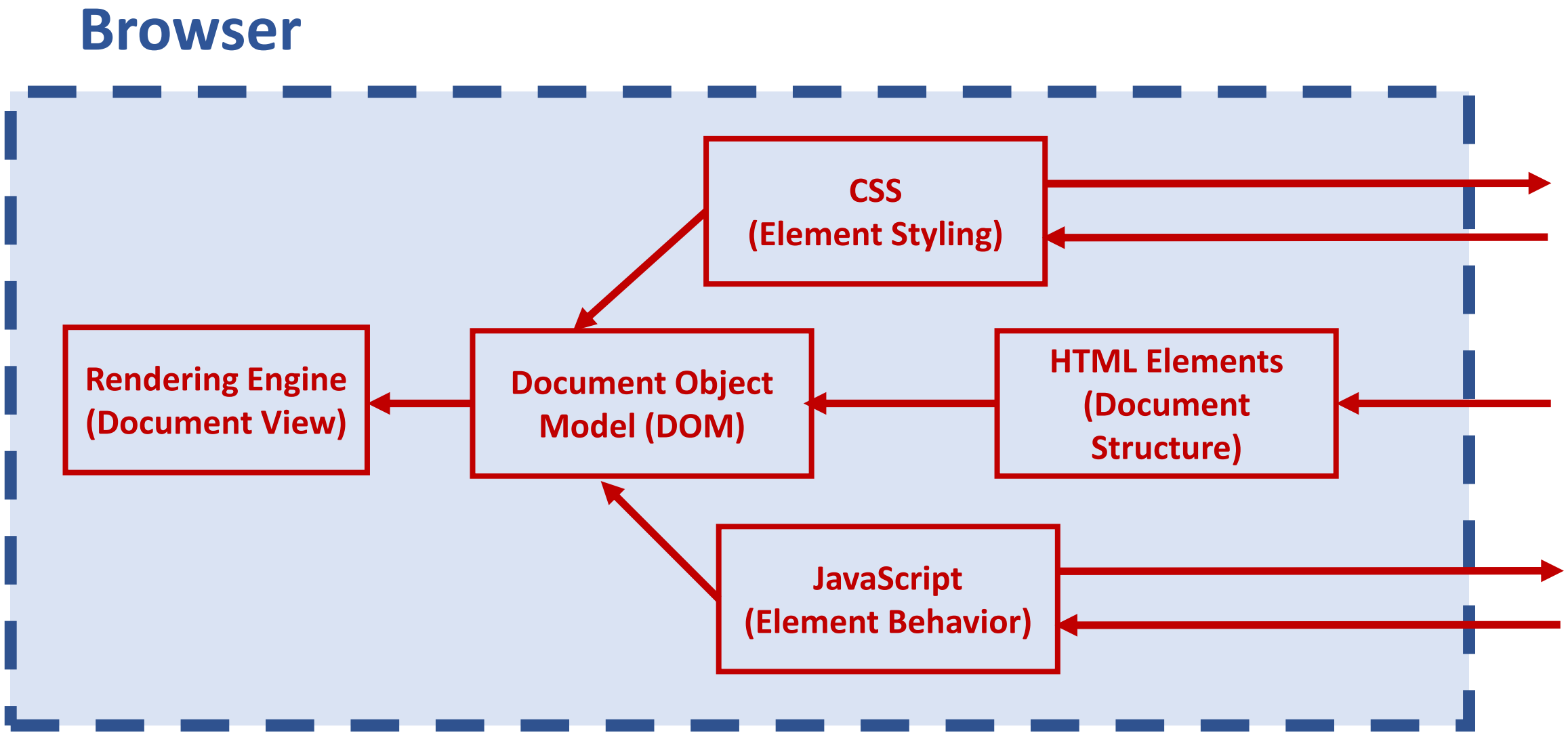




How HTML, CSS, and JavaScript work together in order to provide a webpage that can be rendered in a user's browser?



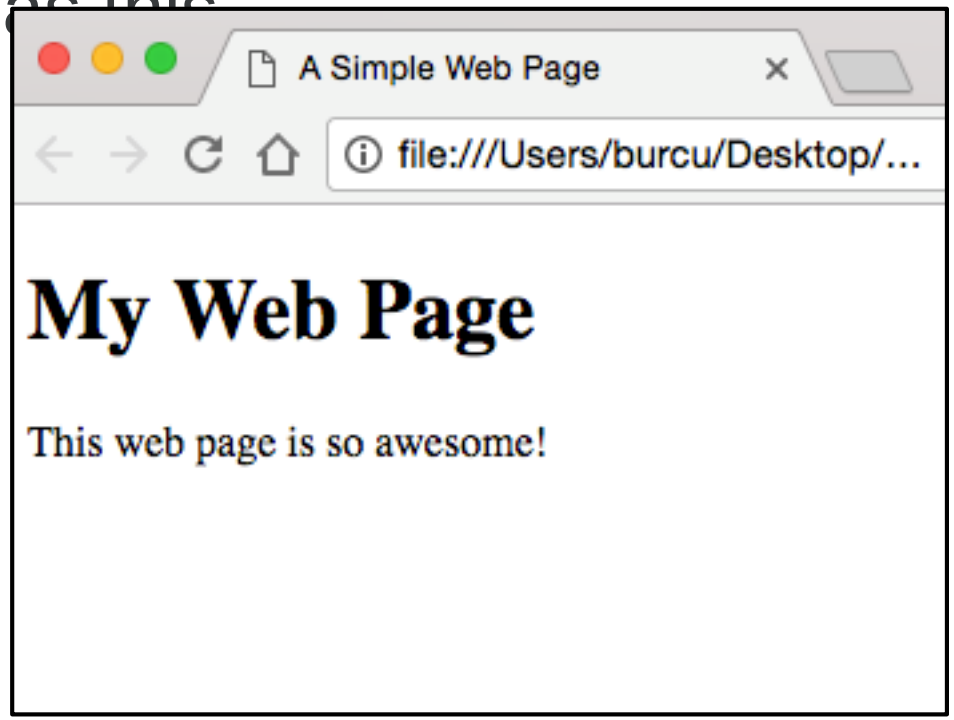
What happens inside the Browser?



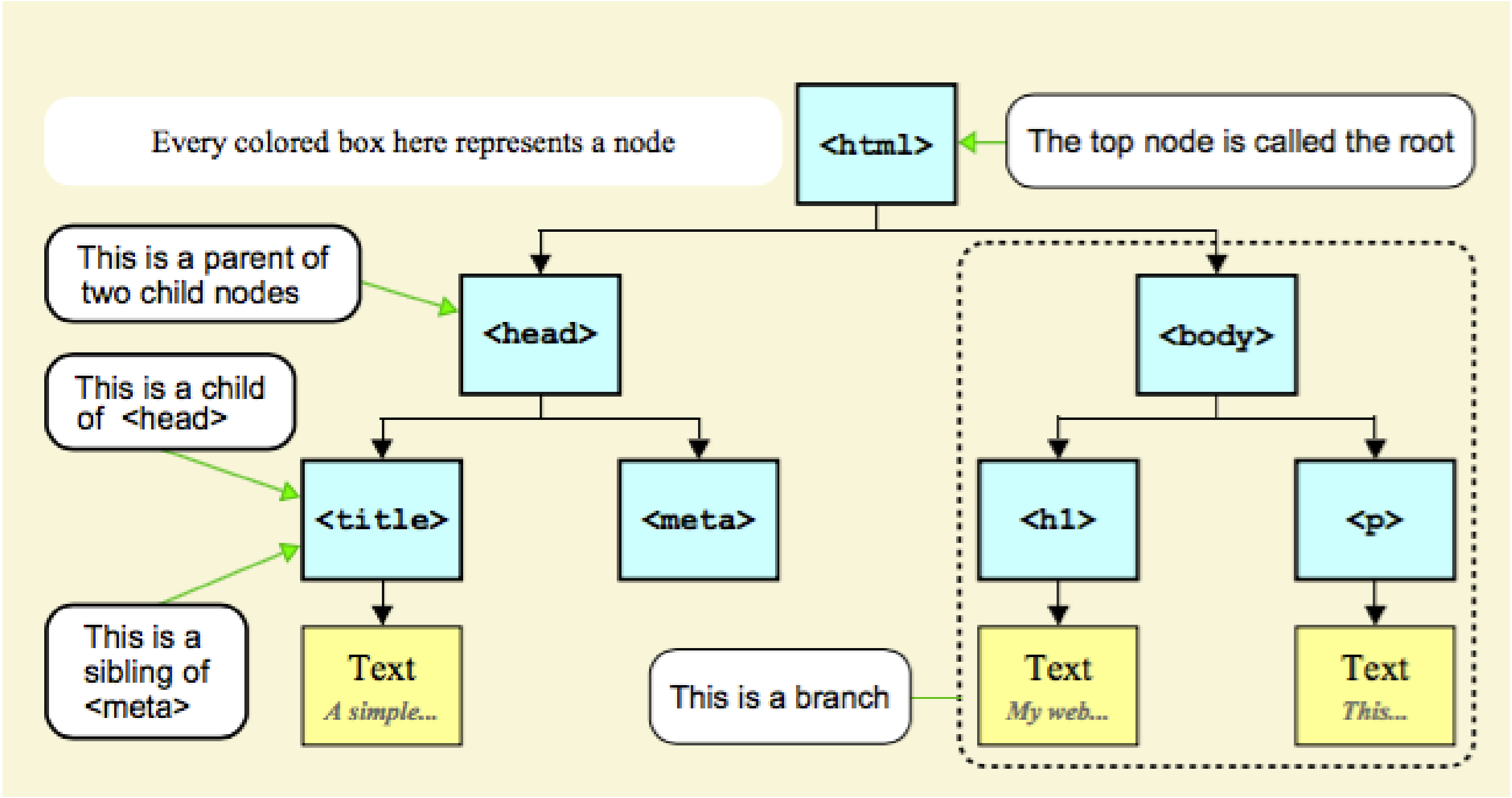
Example1: DOM Structure of a Simple HTML Code

```
<!DOCTYPE html>
<html>
    <head>
        <title>A Simple Web Page</title>
        <meta name="author" content="Burcu">
    </head>
    <body>
        <h1>My Web Page</h1>
        <p>This web page is so awesome!</p>
    </body>
</html>
```

If you load that into a browser, it gets **displayed** as this

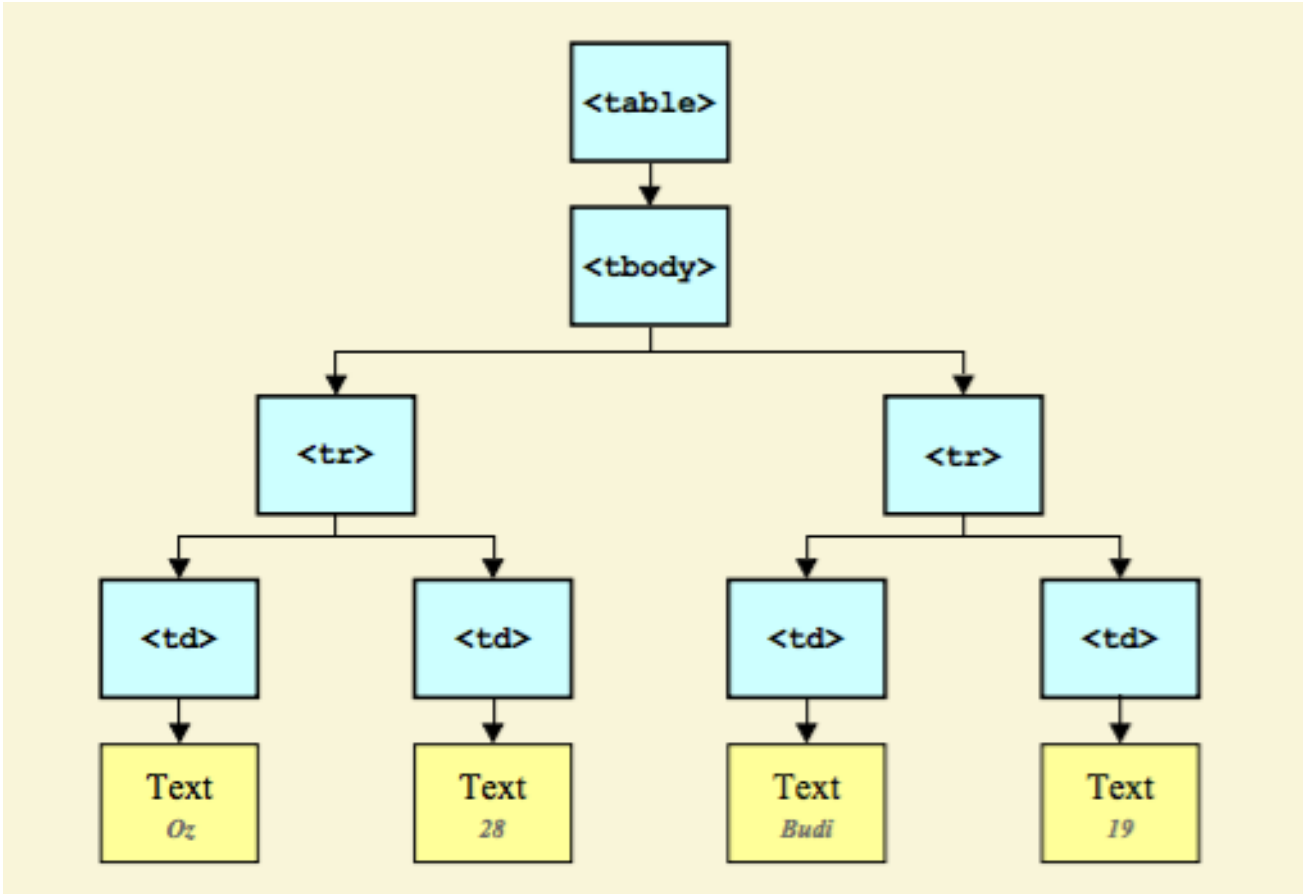


But now we're not focusing on how it gets displayed.
We're focusing on how it gets **stored**!!!


Every colored box here represents a node
<html>
The top node is called the root
This is a parent of two child nodes
<head>
<body>
This is a child of <head>
<title>
<meta>
<h1>
<p>
This is a sibling of <meta>
Text
A simple...
This is a branch
Text
My web...
Text
This...

Example2: DOM Structure of an other HTML Code

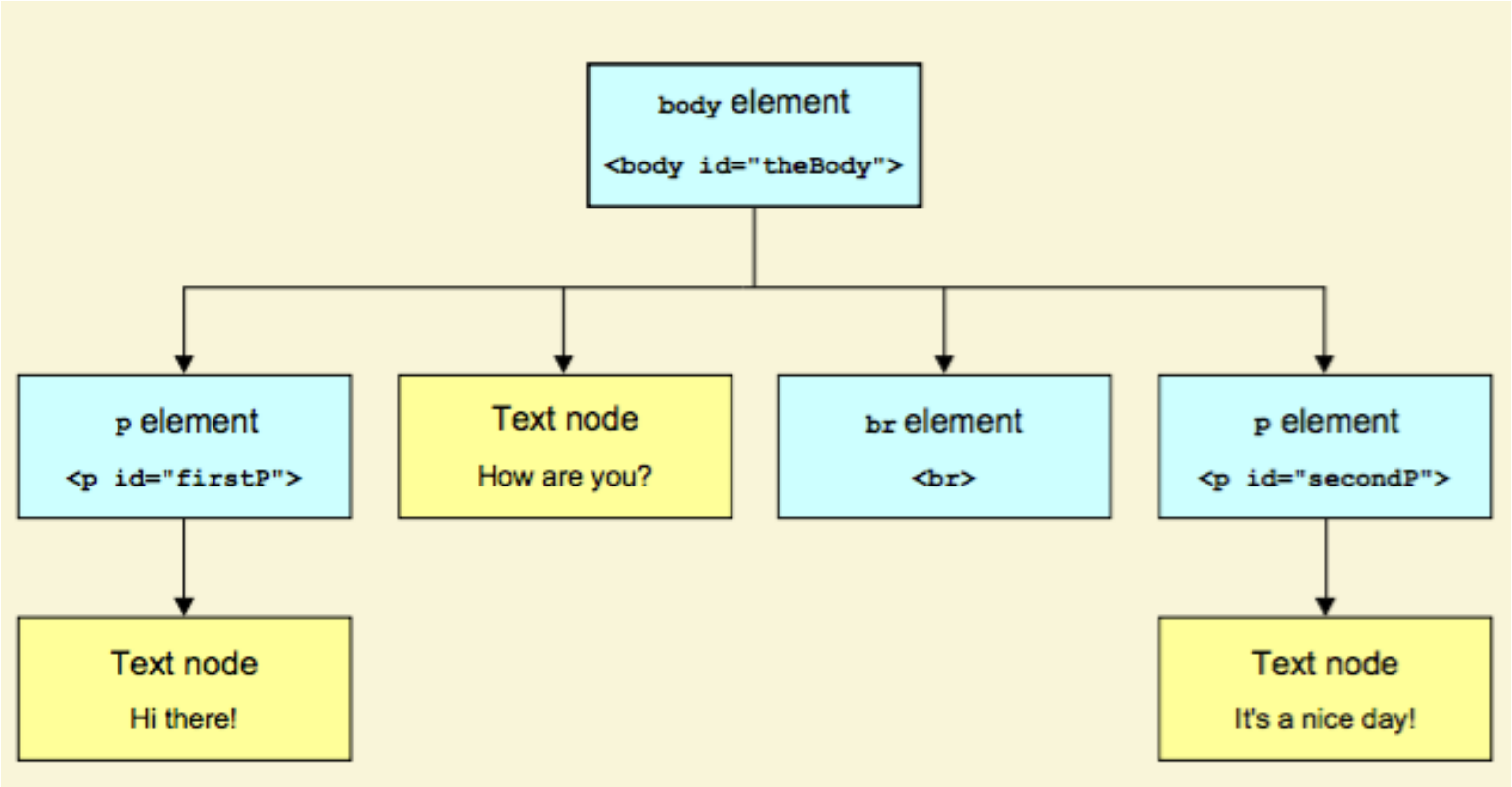
```
<!DOCTYPE html>
<html>
    <body>
        <table>
            <tbody>
                <tr>
                    <td>Oz</td>
                    <td>28</td>
                </tr>
                <tr>
                    <td>Budi</td>
                    <td>19</td>
                </tr>
            </tbody>
        </table>
    </body>
</html>
```

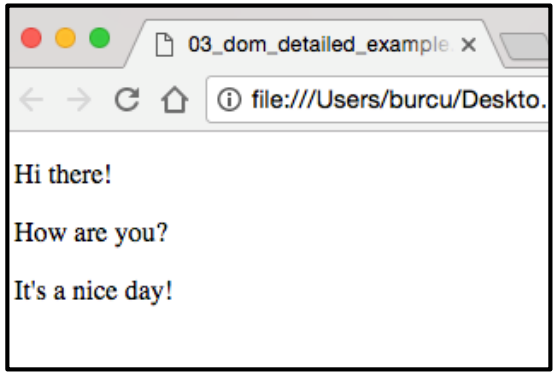


If you load that into a browser, it gets **displayed** as

Example3: DOM Structure of an other HTML Code

```
<!DOCTYPE html>
<html>
    <body id="theBody"><p id="firstP">
        Hi there!
        </p>
        How are you?
        <br>
        <p id="secondP">
            It's a nice day!
        </p>
    </body>
</html>
```





If you load that into a browser, it gets **displayed** as

Node Relationships

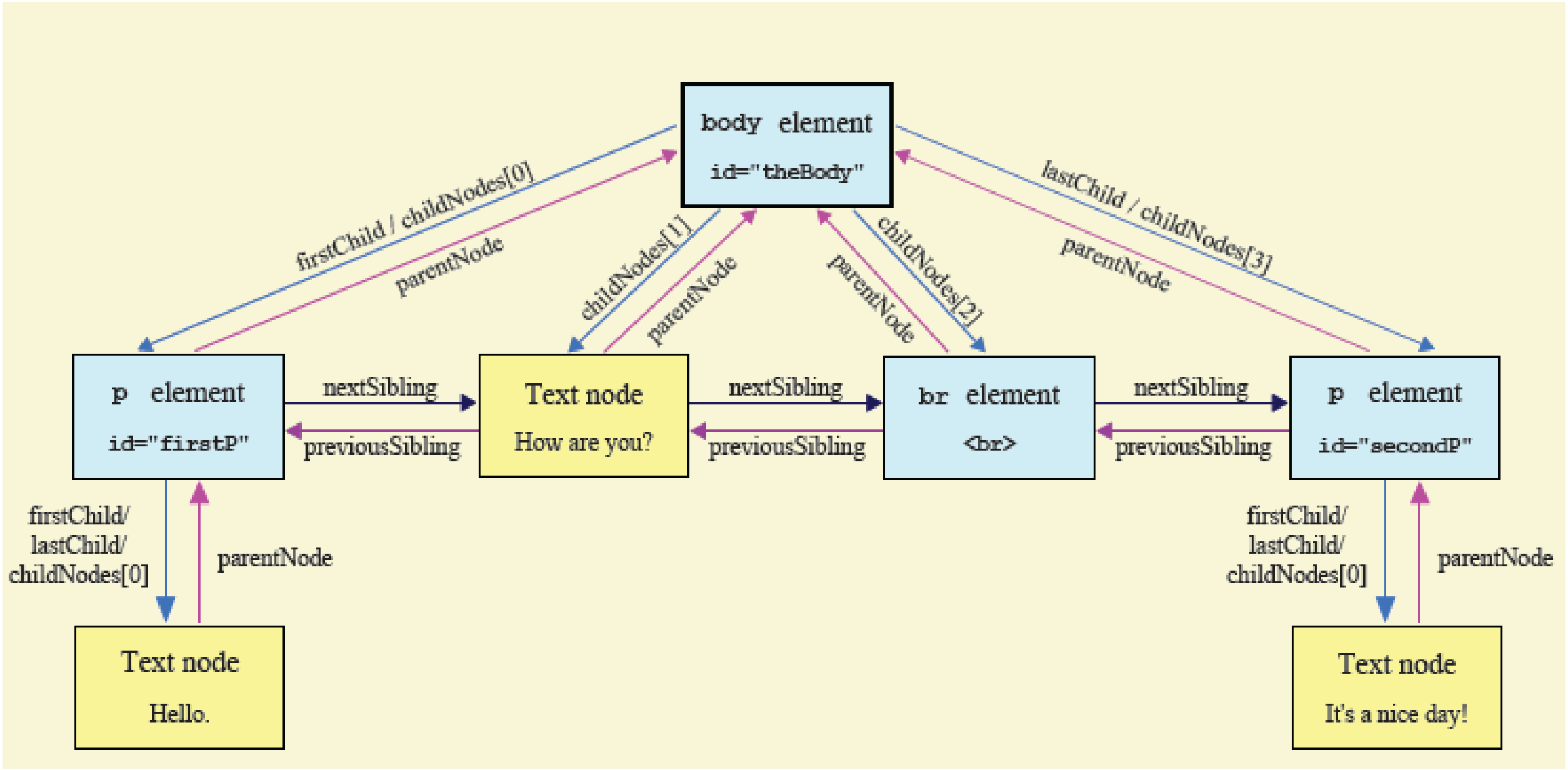
- Anything loaded into the browser memory gets converted to a DOM structure
- How are all these nodes related to one another?
  - We can use specific code which let's us explore that relationship
  - This applies to many different programming languages
    - Don't think this is just JavaScript
    - PHP or Java
  - So these are the keywords which can work in multiple languages

Node Relationships

- Your code can access all of these

Handling the **parent**	parentNode
Handling **child nodes**	childNodes[], firstChild, lastChild
Handling **siblings**	previousSibling, nextSibling

These are keywords which can work on multiple languages!


body element
id="theBody"
firstChild / childNodes[0]
parentNode
childNodes[1]
parentNode
parentNode
childNodes[2]
lastChild / childNodes[3]
parentNode
p element
id="firstP"
nextSibling
previousSibling
Text node
How are you?
nextSibling
previousSibling
br element

nextSibling
previousSibling
p element
id="secondP"
firstChild/
lastChild/
childNodes[0]
parentNode
Text node
Hello.
firstChild/
lastChild/
childNodes[0]
parentNode
Text node
It's a nice day!

How to Find the Path?

Let's say we want to find a particular node in the DOM structure

Ex: Click on a node and then it shows the path to that node

Show the path to a node

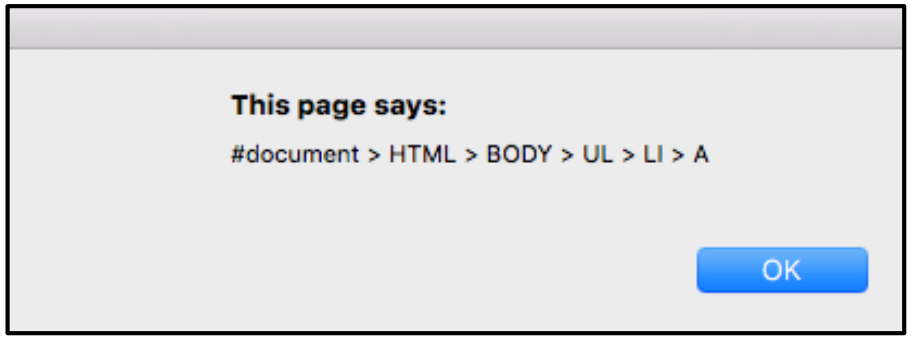
1. The function starts with a node
2. The type of the node is added to a string
3. The code moves to the parent of the node
4. If the node has a parent, repeat (2) and (3)

Function Handling Click Event

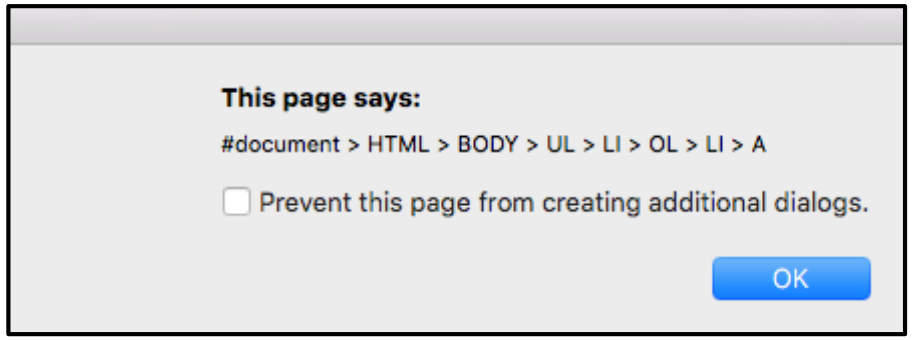
```
// Repeatedly go up the DOM structure
function handleClick(event) {
    event.stopPropagation();

    var node = event.target
    var thisPath = node.nodeName;

    while (node.parentNode) {
        node = node.parentNode;
        thisPath = node.nodeName + " > " + thisPath;
    }
    alert(thisPath);
}
```

Output

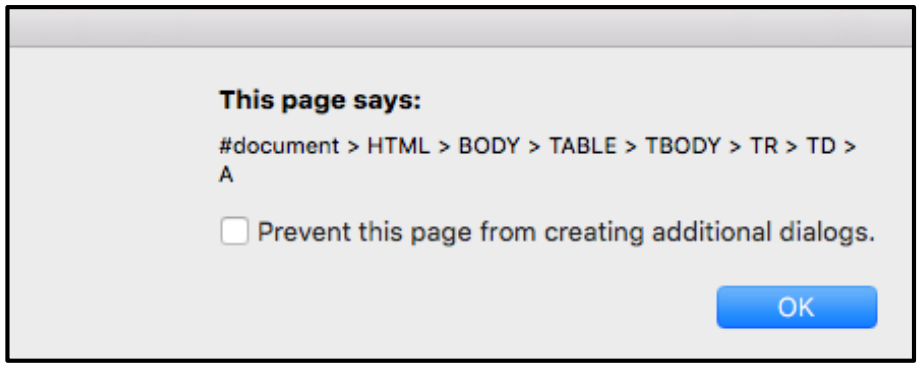
- Click on “**List 1**”



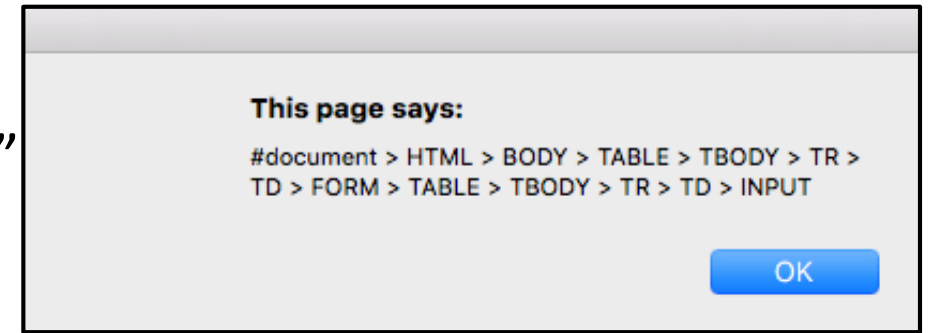
- Click on “**Order List 4**”

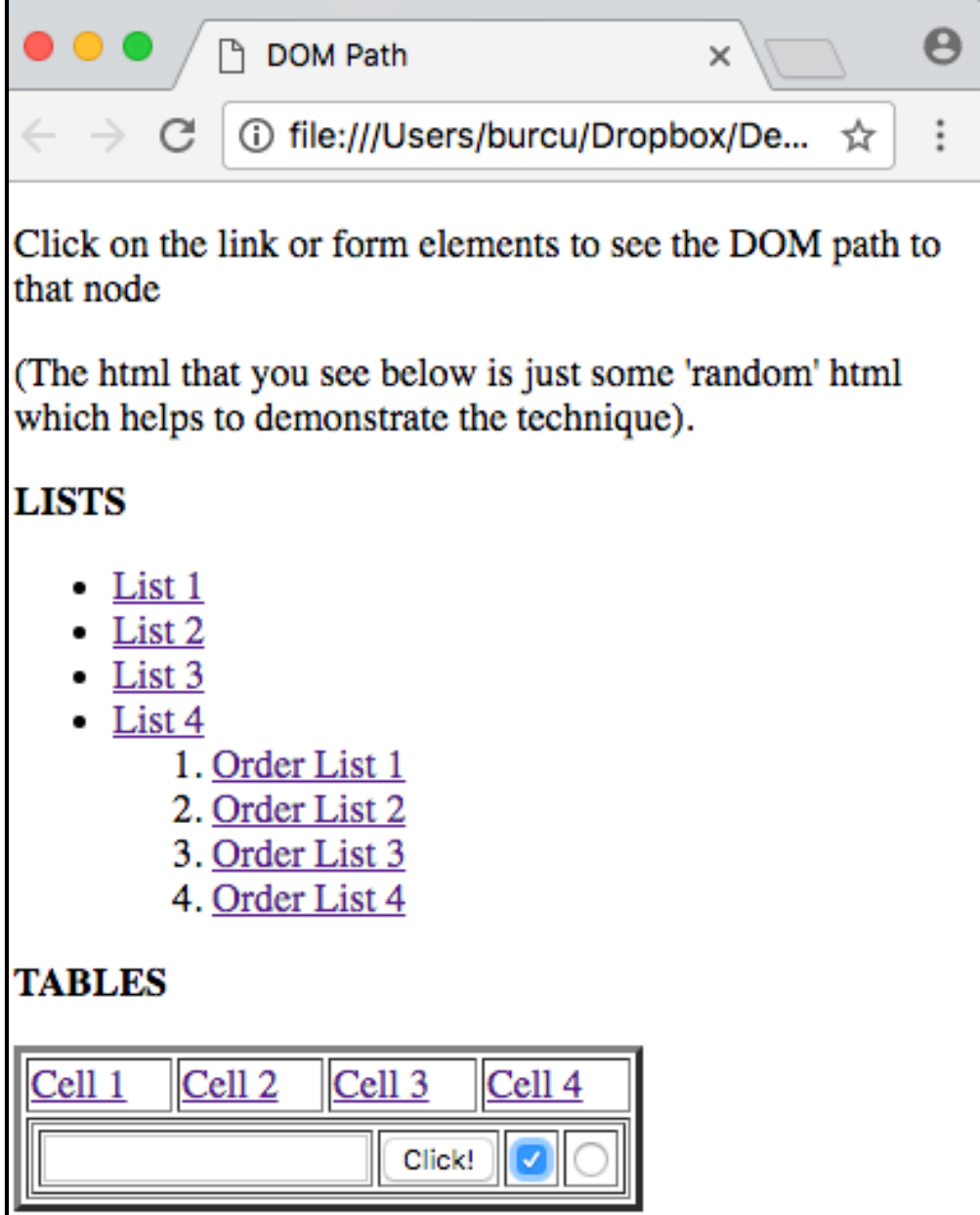


- Click on “**Cell 1**”



- Click on “**Rectangle**”





Function Handling Click Event

```
// Repeatedly go up the DOM structure
function handleClick(event) {
    event.stopPropagation();

    var node = event.target
    var thisPath = node.nodeName;

    while (node.parentNode) {
        node = node.parentNode;
        thisPath = node.nodeName + " > " + thisPath;
    }
    alert(thisPath);
}
```

→ **event.stopPropagation**

A little instruction which just makes sure that this code is only executed once and it's not executed every time

Ex: let's say we click on an input element in a table. We don't want the input element to trigger the function and the table also to trigger the function. So we say, make sure it only happens once by stopping the event after its been handled once.

How to Trigger the Code?

- To trigger the code, the user clicks on a node
- To enable this, event handlers are added to the nodes

- How do we trigger that code?
  - After we've constructed that function that we saw - ***handleClick(event)***,
  - We add that function to almost every element in the DOM structure
  - How did we add it to every element?

```
// Register the click event handler for all nodes
function attachHandler(node) {
    if(node == null) return;
    node.onclick = handleClick;

    for(var i = 0; i < node.childNodes.length; ++i) {
        visit(node.childNodes[i]);
    }
}
```

Locating Nodes

- Everything is in the DOM
- We can **add**/**delete**/**copy**/**change** anything
- To do this, we need to understand **how to access things**
  - There are 3 methods for locating something in a DOM structure
    - **Method 1:** ***Using the Exact Path***
    - **Method 2:** ***Using the Type***
    - **Method 3:** ***Using the Name***

Method 1: Using the Exact Path

- Method 1: Use the exact DOM path
  - Sometimes hard to work out the exact path
    - So you go right from the start, from the root, and you go down, down, down
  - Not the best way
    - Hard to work out the exact path to the node
    - Easy to make mistakes
  - In another browser the DOM may be slightly different - code fails!
    - It's possible you build some code and it works in one browser and then you try the code in another browser and it doesn't work
    - The reason is because the DOM structure may be slightly different, especially at the very top level
    - Usually not an issue these days, but it's an issue with older browsers

Method 2: Using the Type

- Method 2: Use ***getElementsByTagName()***
  - Find something in the DOM structure by using the type
  - This requires you to know the exact tag name, e.g. is it h2 or h3?
  - There might be several nodes of that type, so you have to know exactly which one it is
    - However, there maybe five h3's or seven h2's
    - So you actually need to know which particular h1 or h2 or h3 you want to find – the first one, the second one, or the third one
    - There's a little bit of extra work with this method!

Method 3: Using the Name

- Method 3: Use ***getElementById()***
  - getElement in the DOM structure by searching for the Id
  - If you give a node a unique name, e.g.

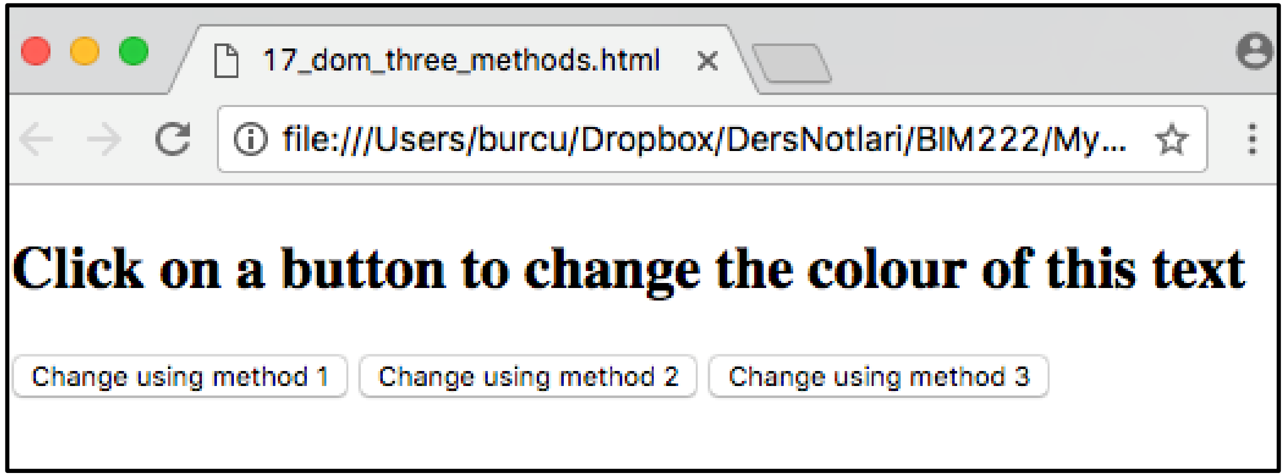
    <*element_name* id="**thing**"> . . . </*element_name*>

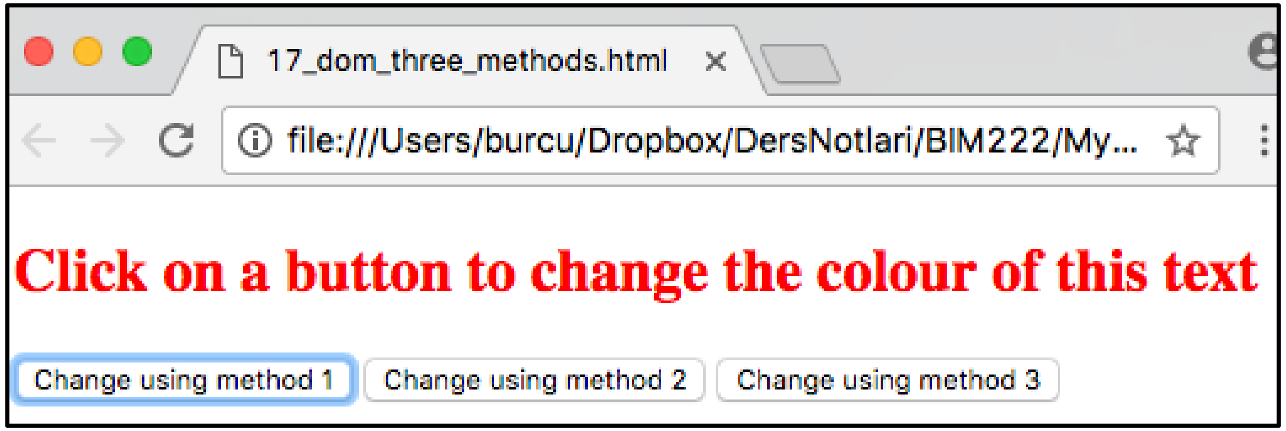
    then this method is the **easiest way**

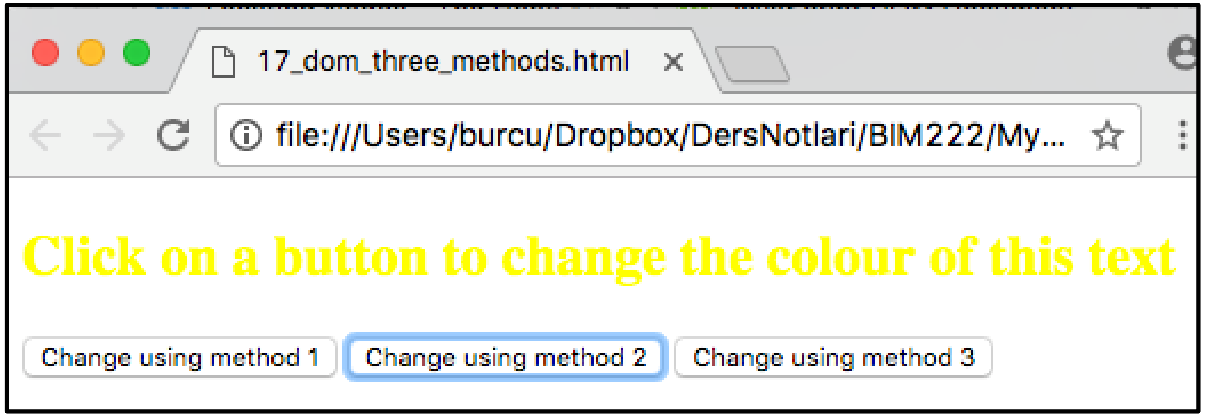
Example Code

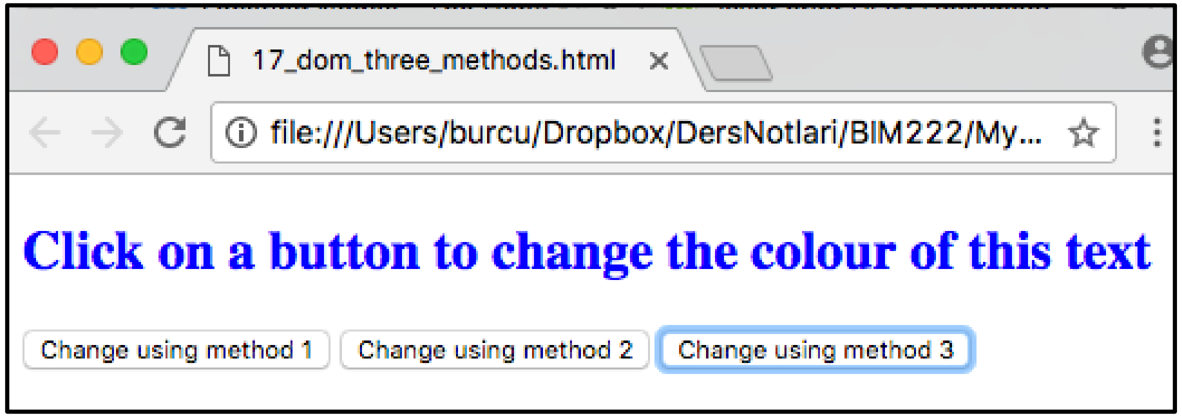
```
<!DOCTYPE html>
<html>
    <head>
        <script>
            function change_color1() {
                document.childNodes[1].childNodes[2].childNodes[1].style.color="red";
            }

            function change_color2() {
                document.getElementsByTagName("h2")[0].style.color="yellow";
            }

            function change_color3() {
                document.getElementById("cute_text").style.color="blue";
            }
        </script>
    </head>

    <body>
        <h2 style="color:black" id="cute_text">
            Click on a button to change the colour of this text
        </h2>
        <form>
            <input onclick="change_color1()" type="button" value="Change using method 1">
            <input onclick="change_color2()" type="button" value="Change using method 2">
            <input onclick="change_color3()" type="button" value="Change using method 3">
        </form>
    </body>
</html>
```

Example Code - Output









setAttribute()

A common way to change something

```
the_node = getElementById("thisNode");
the_node.setAttribute("style", "color:red");
```

Same Example Code with *setAttribute()*

```
<!DOCTYPE html>
<html>
    <head>
        <script>
            function change_color1() {
                document.childNodes[1].childNodes[2].childNodes[1].setAttribute("style", "color:red");
            }

            function change_color2() {
                document.getElementsByTagName("h2")[0].setAttribute("style", "color:yellow");
            }

            function change_color3() {
                document.getElementById("cute_text").setAttribute("style", "color:blue");
            }
        </script>
    </head>

    <body>
        <h2 style="color:black" id="cute_text">
            Click on a button to change the colour of this text
        </h2>
        <form>
            <input onclick="change_color1()" type="button" value="Change using method 1">
            <input onclick="change_color2()" type="button" value="Change using method 2">
            <input onclick="change_color3()" type="button" value="Change using method 3">
        </form>
    </body>
</html>
```

Creating and Adding Nodes

- Basic Steps
  - First, create whatever you want to add

    whatever you create is not yet in the DOM
  - Second, add it at the desired place

Creating a node	createElement(), createTextNode()
Adding a node	insertBefore(), appendChild()

Creating Nodes

Use ***createElement()***, e.g.

```
var result = document.createElement("div");
```

For text nodes, use ***createTextNode()*** e.g.

```
var result = document.createTextNode("Hello");
```

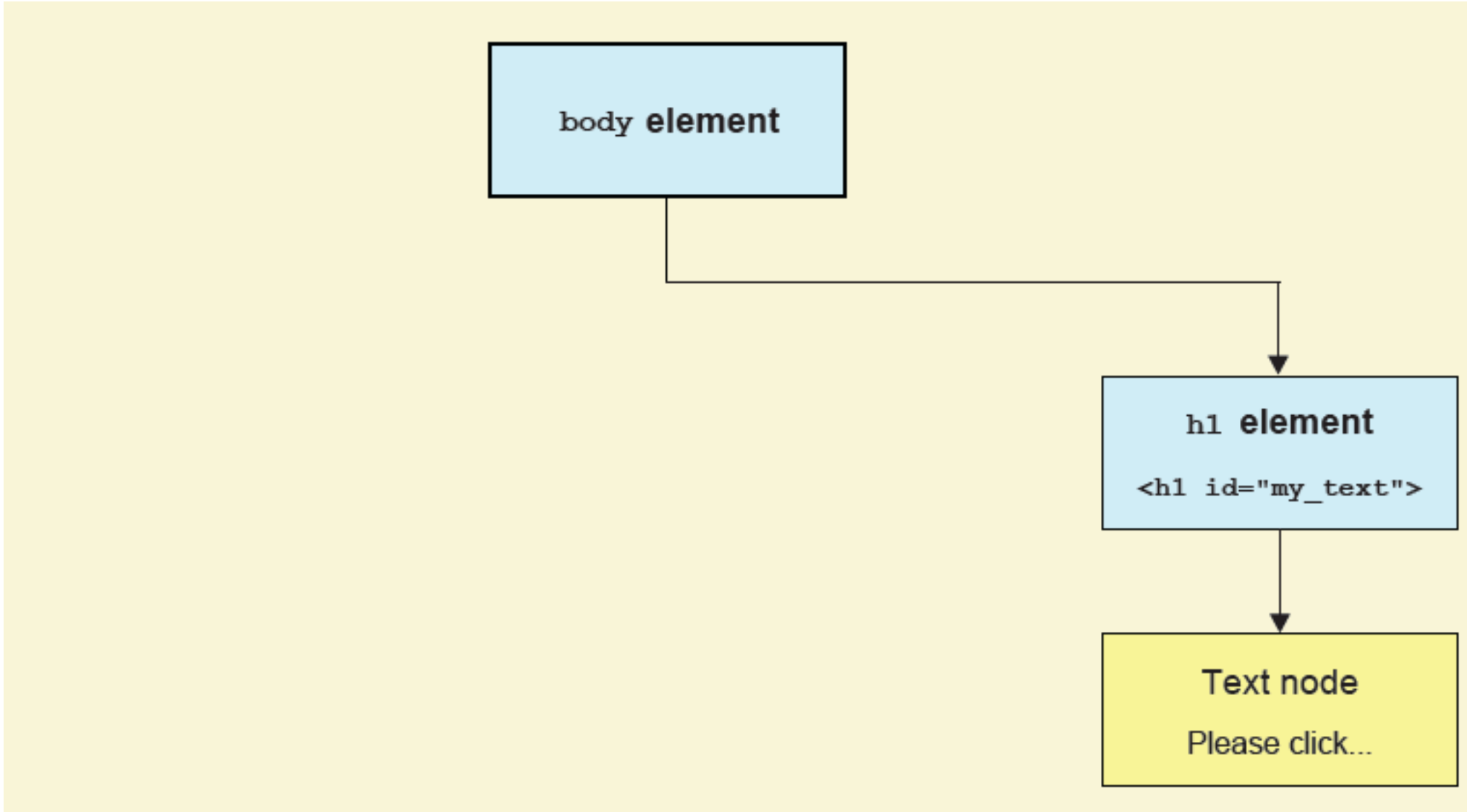
Adding Nodes – insertBefore()

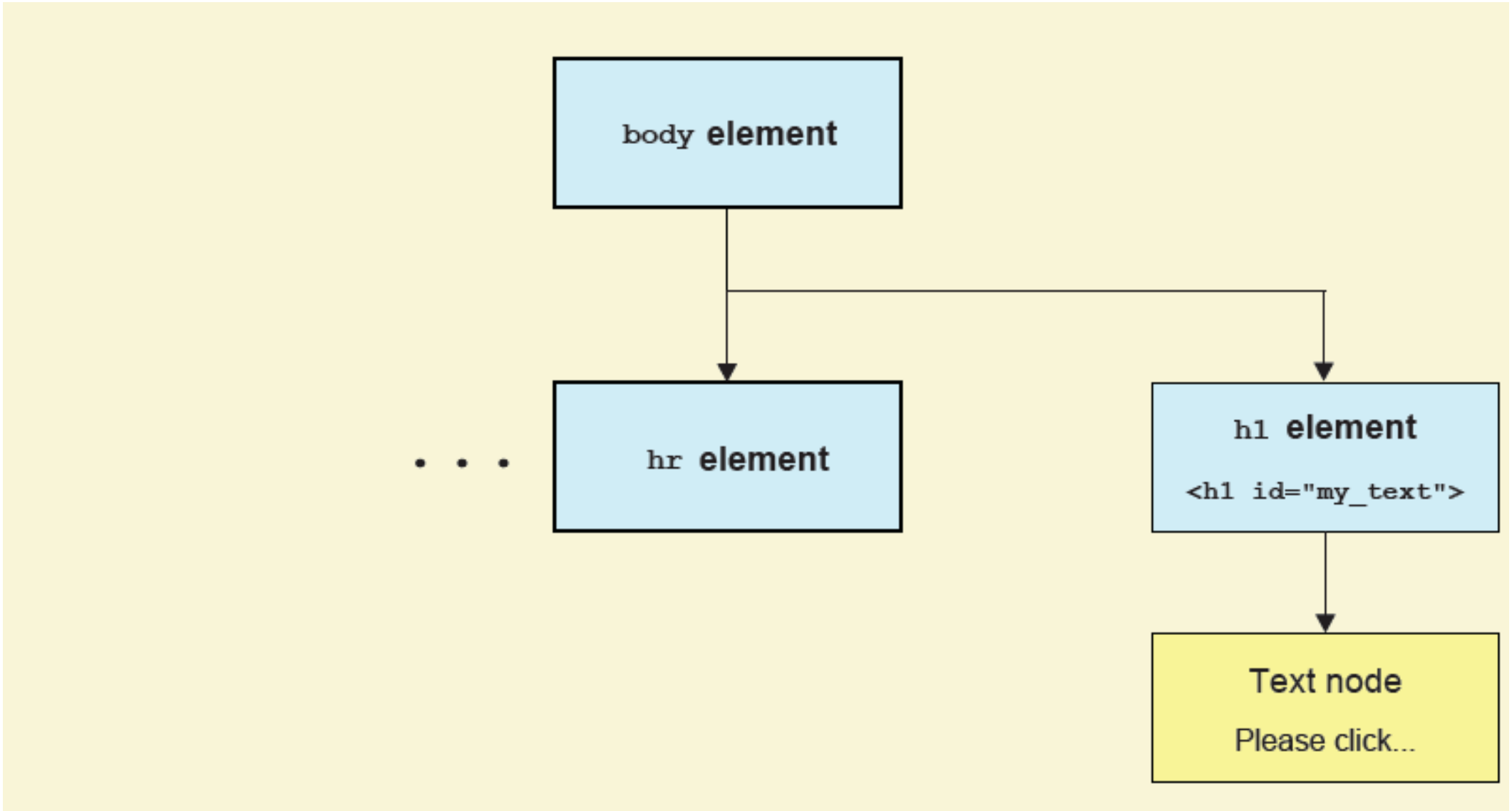
Example:

```
newItem = document.createElement("hr");

destParent = document.getElementByTagName("body")[0];

destParent.insertBefore(newItem, destParent.firstChild);
```

Example Code

```
<!DOCTYPE html>
<html>
    <head>
        <script>
            function insert_new_text() {
                var newItem = document.createElement("hr");
                var destParent = document.getElementsByTagName("body")[0];
                destParent.insertBefore(newItem, destParent.firstChild);
            }
        </script>
    </head>

    <body onclick="insert_new_text()">
        <h1 id="my_text">Please click on the page</h1>
    </body>
</html>
```


body element
h1 element
<h1 id="my_text">
Text node
Please click...

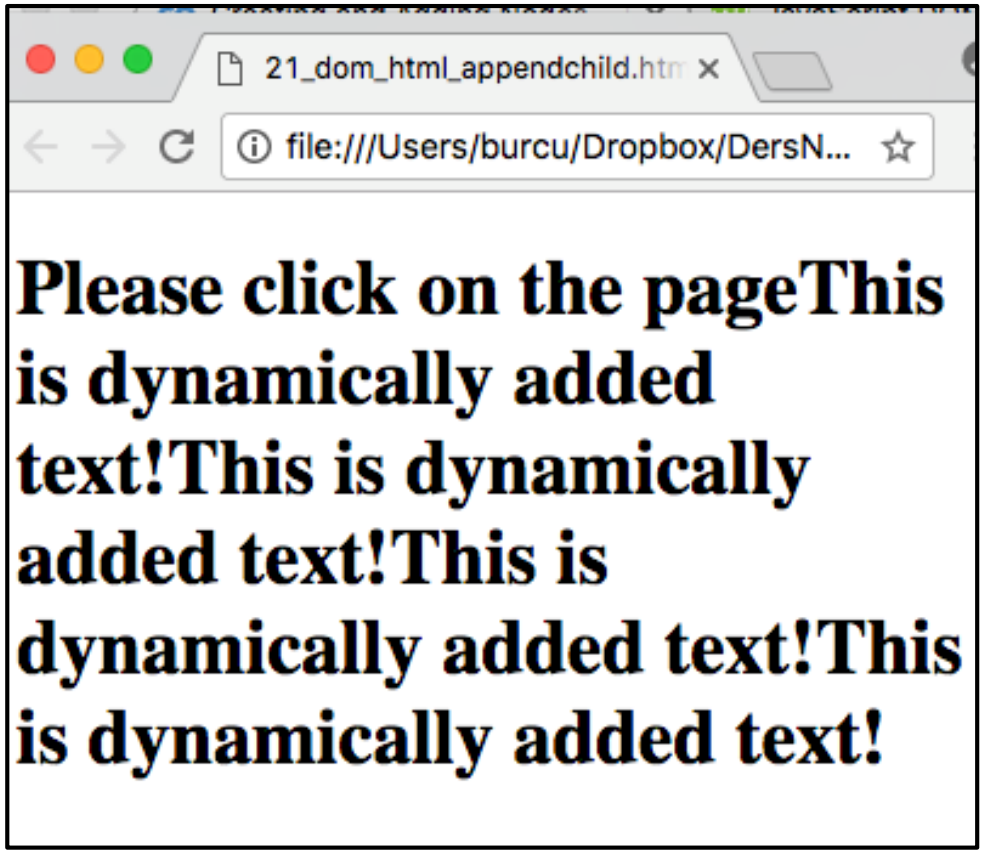

body element
hr element
h1 element
<h1 id="my_text">
Text node
Please click...

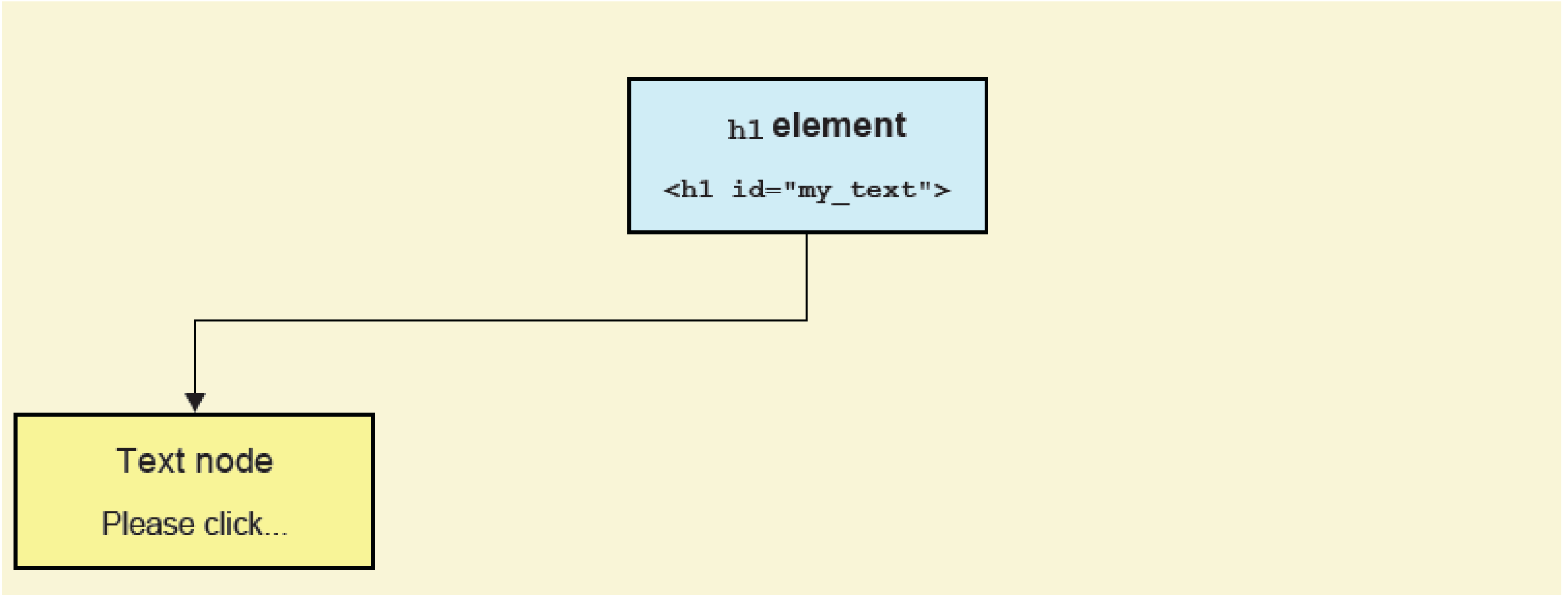
Adding Nodes – appendChild()

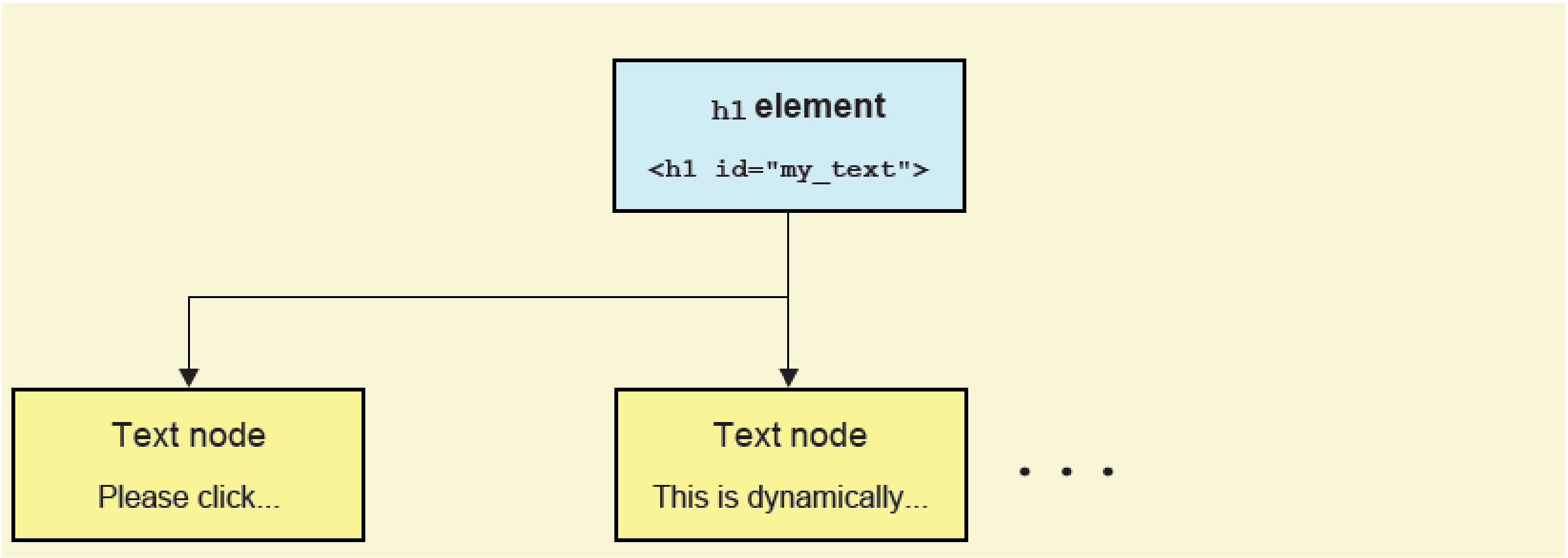
Example:

```
var result = document.createTextNode("This is dynamically added Text!");

document.getElementById("my_text").appendChild(result);
```

Example Code - Output




h1 element
<h1 id="my_text">
Text node
Please click...


h1 element
<h1 id="my_text">
Text node
Please click...
Text node
This is dynamically...
. . .

Example Code

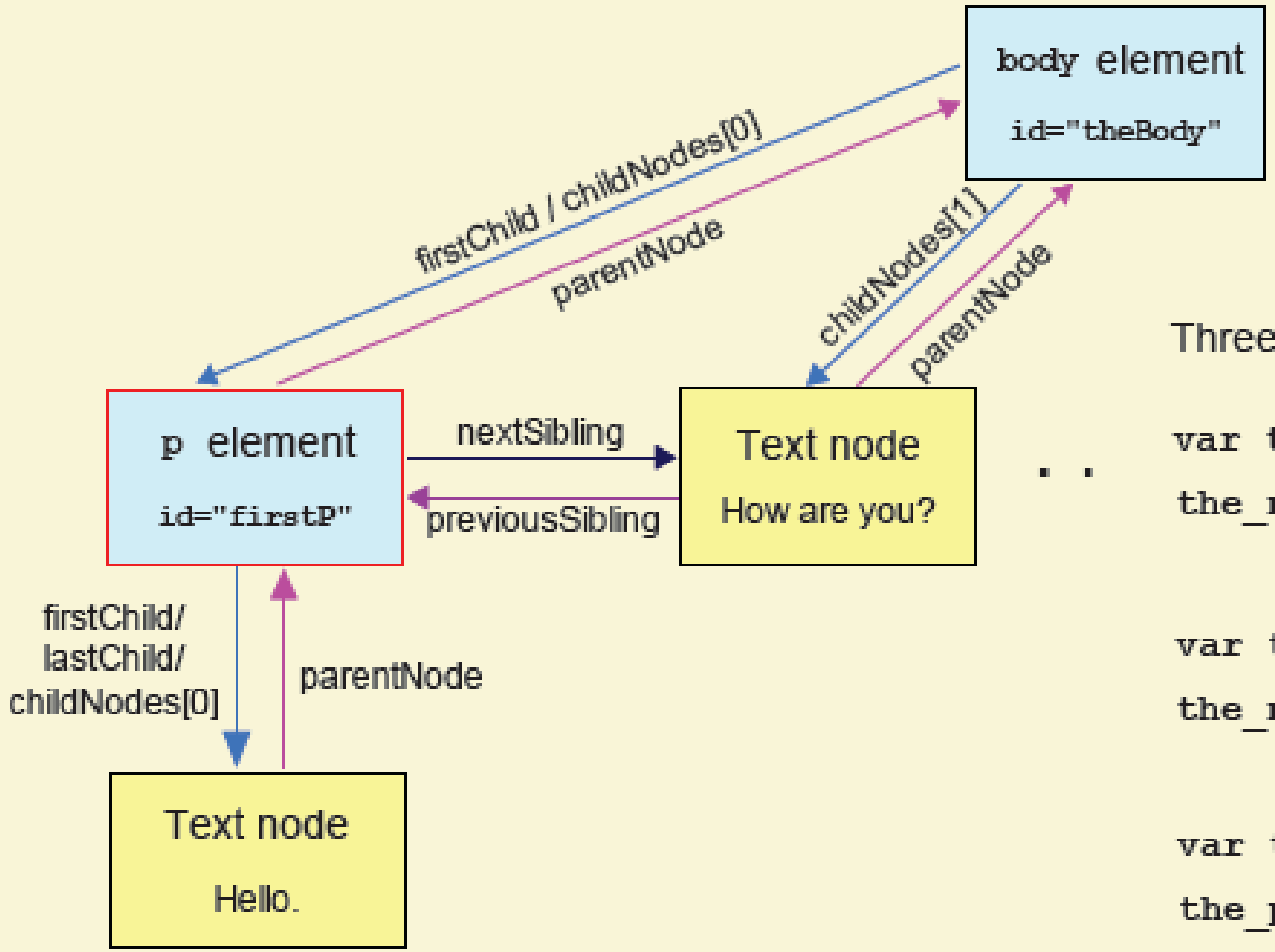
```
<!DOCTYPE html>
<html>
    <head>
        <script>
            function insert_new_text (){
                var newText = document.createTextNode(
                    "This is dynamically added text!");
                var textpart = document.getElementById("my_text");
                textpart.appendChild(newText);
            }
        </script>
    </head>

    <body onclick="insert_new_text()">
        <h1 id="my_text">Please click on the page</h1>
    </body>
</html>
```

Deleting a Node

Tell the parent of the node to delete it, e.g.

```
this_node = getElementById("myPara");
this_node.parentNode.removeChild(this_node);
```

- First we find a particular element that we want to delete, a particular node in the DOM structure
  → Here we're searching for something with the id *myPara*
- Then we have to ask the parent of that child to delete it
- Should you find the ***parentNode***?
  - Yes, unless if you are not using something like jQuery
  - The idea using ***parentNode*** for deletion is that the DOM element which you are connected to can only delete you



Three example code to delete the first paragraph in this DOM:

```
var the_node=document.getElementById("firstP");
the_node.parentNode.removeChild(the_node);
```

or

```
var the_node=document.getElementsByTagName("p")[0];
the_node.parentNode.removeChild(the_node);
```

or

```
var the_parent=document.getElementById("theBody");
the_parent.removeChild(the_parent.firstChild);
```

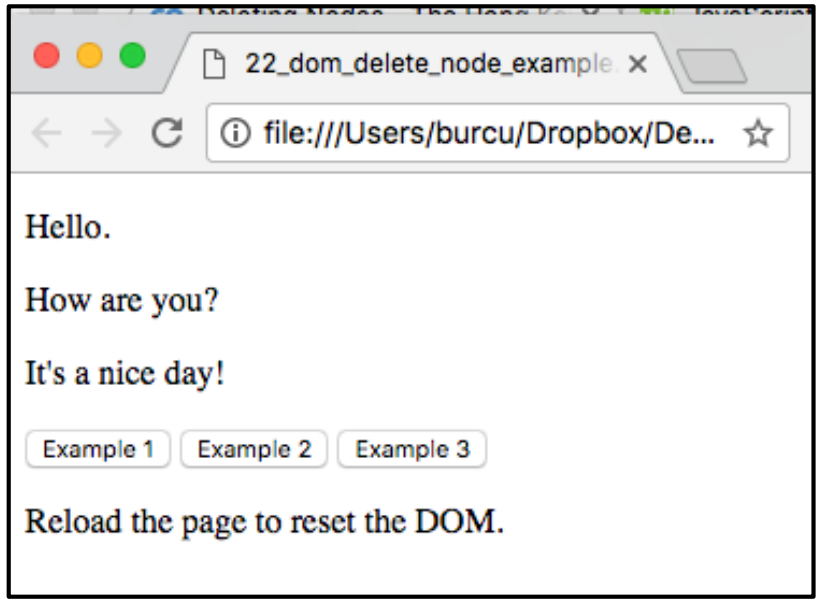
Three Example Code

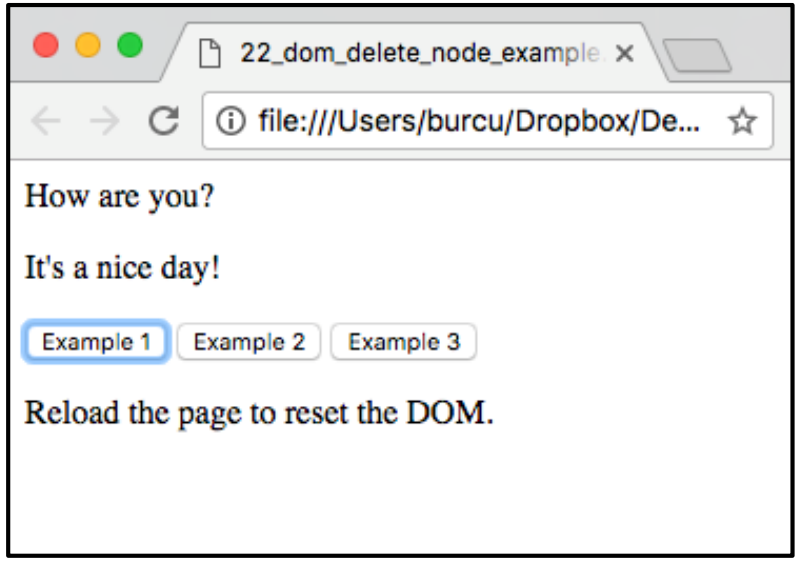
```
var the_node=document.getElementById("firstP");
the_node.parentNode.removeChild(the_node);
```

```
var the_node=document.getElementsByTagName("p")[0];
the_node.parentNode.removeChild(the_node);
```

```
var the_parent=document.getElementById("theBody");
the_parent.removeChild(the_parent.firstChild);
```

Example Code



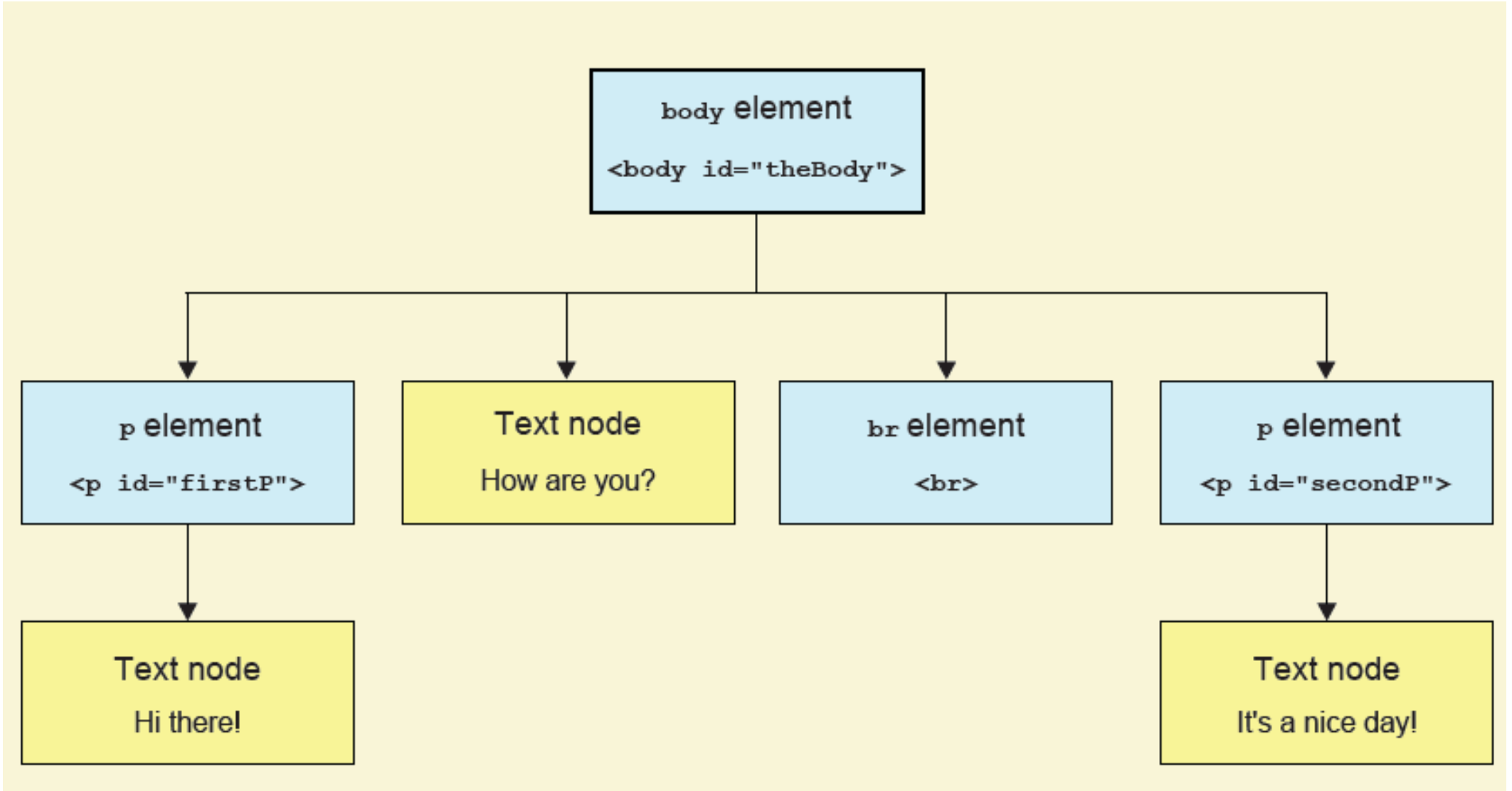



```
<!DOCTYPE html>
<html>
    <head>
        <script>
            function delete1()
            {
                var the_node=document.getElementById("firstP");
                the_node.parentNode.removeChild(the_node);
            }
            function delete2()
            {
                var the_node=document.getElementsByTagName("p")[0];
                the_node.parentNode.removeChild(the_node);
            }
            function delete3()
            {
                var the_parent=document.getElementById("theBody");
                the_parent.removeChild(the_parent.firstChild);
            }
        </script>
    </head>

    <body id="theBody"><p id="firstP">
        Hello.
        </p>
        How are you?
        <br>
        <p id="secondP">
        It's a nice day!
        </p>

        <button type="button" onclick="delete1()">Example 1</button>
        <button type="button" onclick="delete2()">Example 2</button>
        <button type="button" onmousedown="delete3()">Example 3</button>
        <p>
        Reload the page to reset the DOM.
        </p>
    </body>
</html>
```

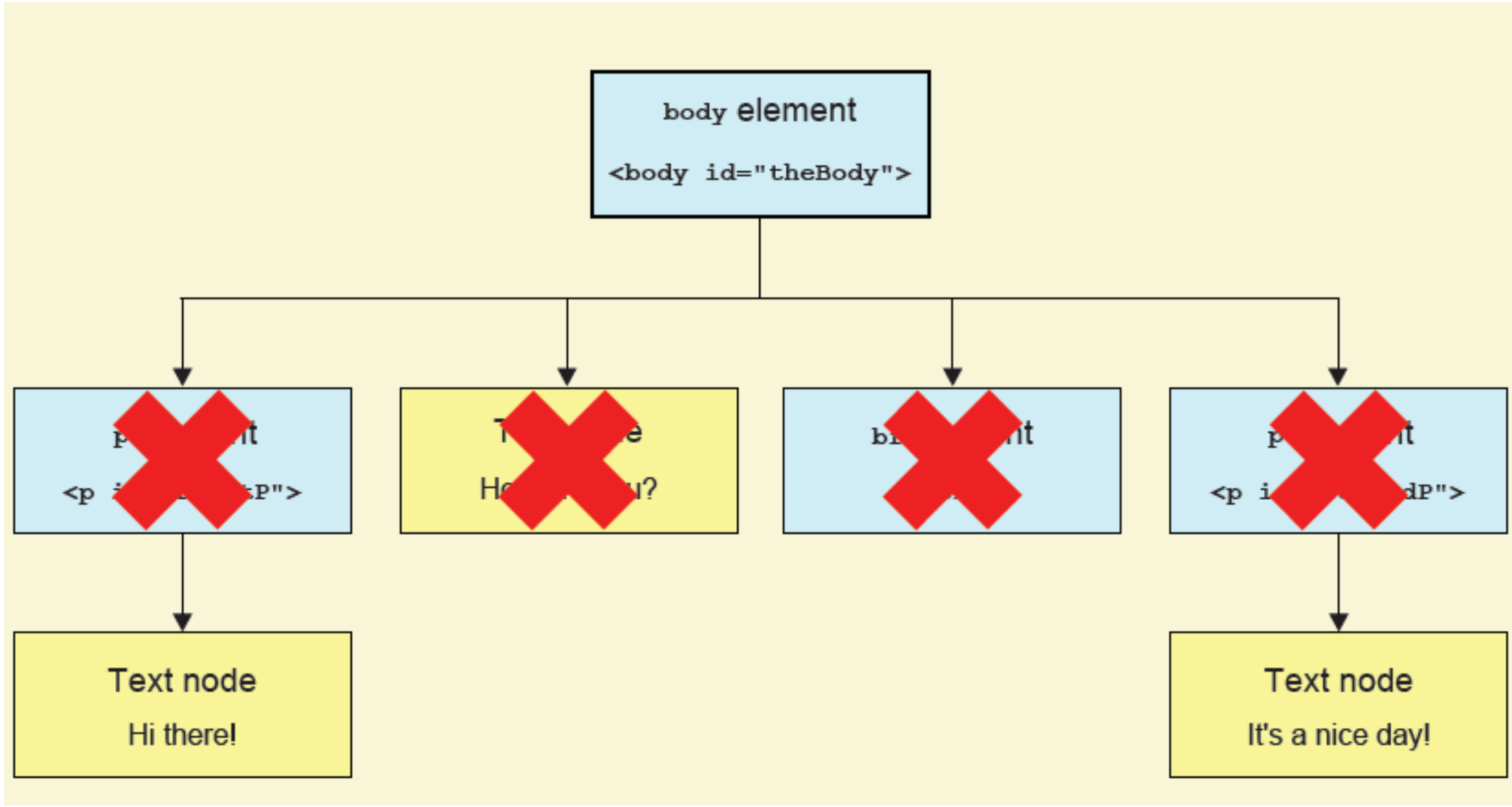
Deleting All Children

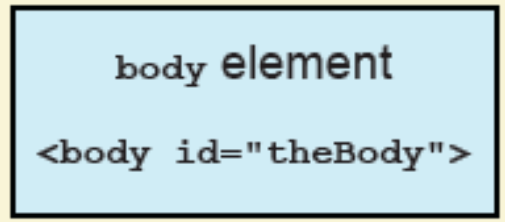
- Sometimes you want to delete everything under a node
  - For example, deleting all web page content
- One way to do that is to delete every child

```
var theNode = document.getElementById("theBody");

while (theNode.firstChild)
    theNode.removeChild(theNode.firstChild);
```


body element
<body id="theBody">
p element
<p id="firstP">
Text node
How are you?
br element

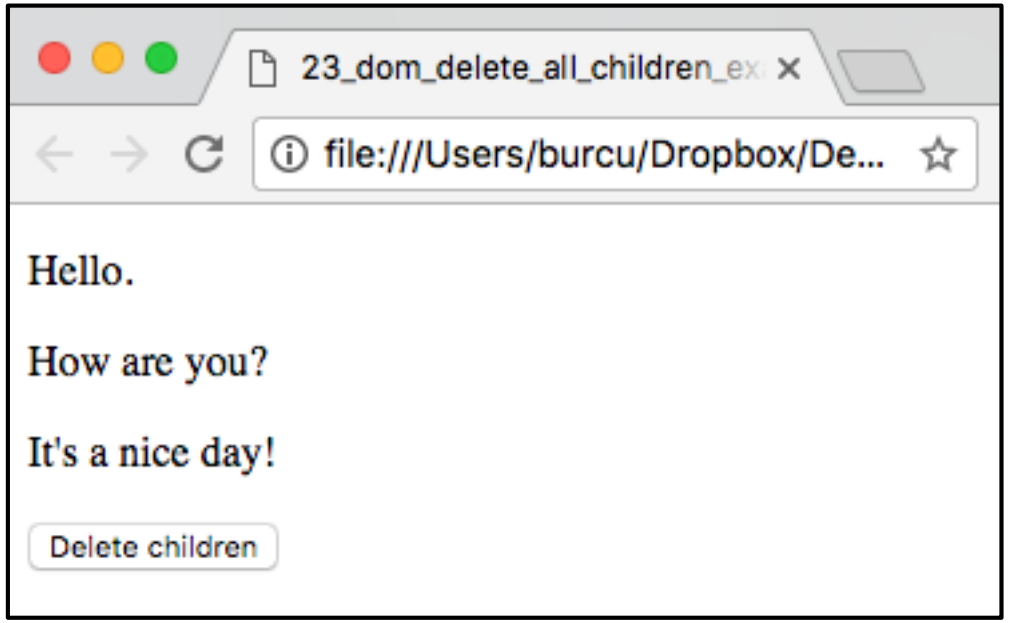
p element
<p id="secondP">
Text node
Hi there!
Text node
It's a nice day!

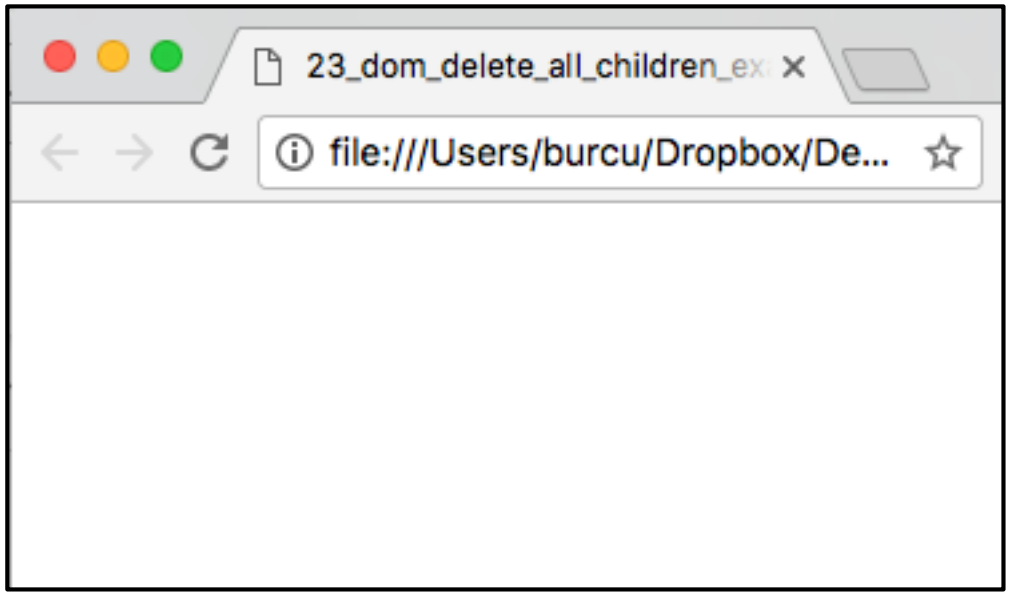

body element
<body id="theBody">
Text node
Hi there!
Text node
It's a nice day!


body element
<body id="theBody">

Example Code

```
<!DOCTYPE html>
<html>
    <head>
        <script>
            function delete_all_children() {

                var theNode = document.getElementById("theBody");

                while (theNode.firstChild)
                    theNode.removeChild(theNode.firstChild);

            }
        </script>
    </head>

    <body id="theBody">
        <p id="firstP">Hello.</p>
        How are you?
        <br>
        <p id="secondP">It's a nice day!</p>
        <button type="button" onclick="delete_all_children()">
            Delete children
        </button>
    </body>
</html>
```





Cloning Nodes

Basic Idea

1. Copy node(s) from the DOM
2. Paste the copied node(s) in the DOM

Copying a node	the_node.cloneNode()
Copying a branch	the_node.cloneNode(true)
Adding node(s)	dest.appendChild(the_node)

Cloning a Node

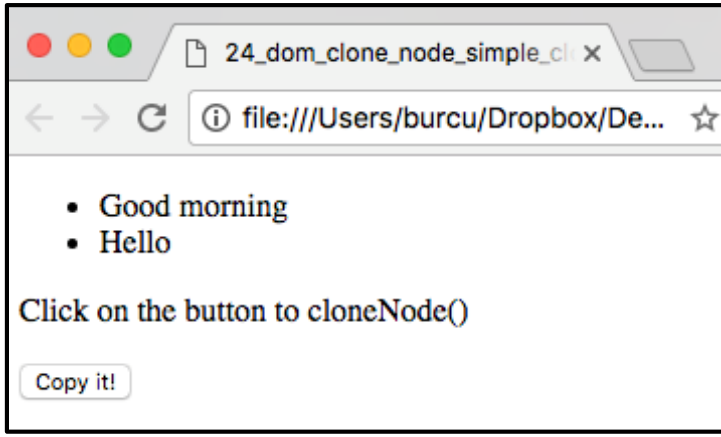
- Use ***node.cloneNode()***
- It's the same as ***node.cloneNode(false)***
- Example:

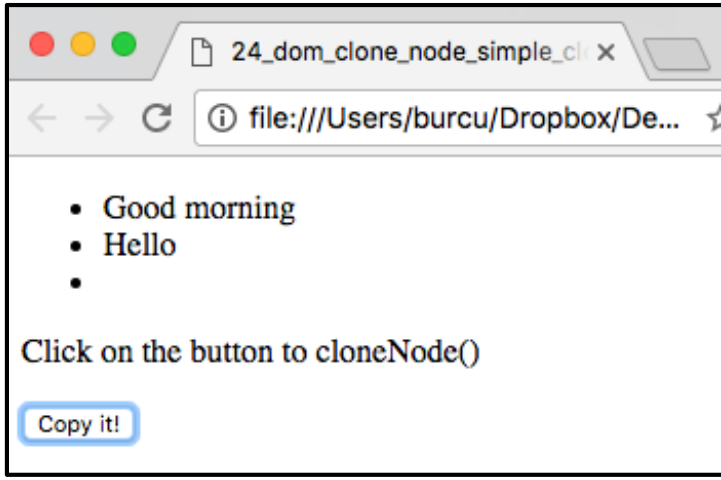
    1) A list item node <li> is copied

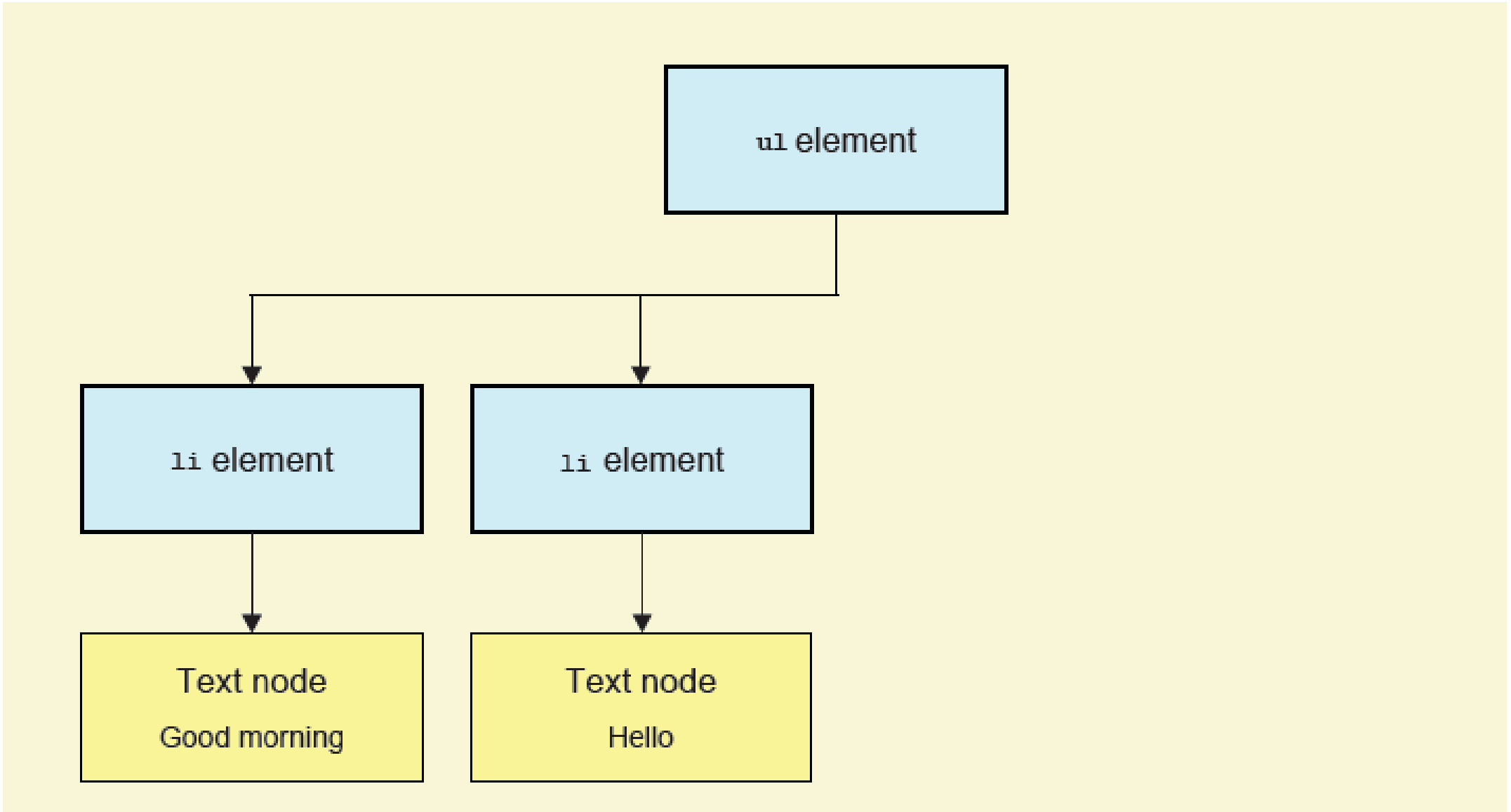
    2) The copy is then added to the end of the list

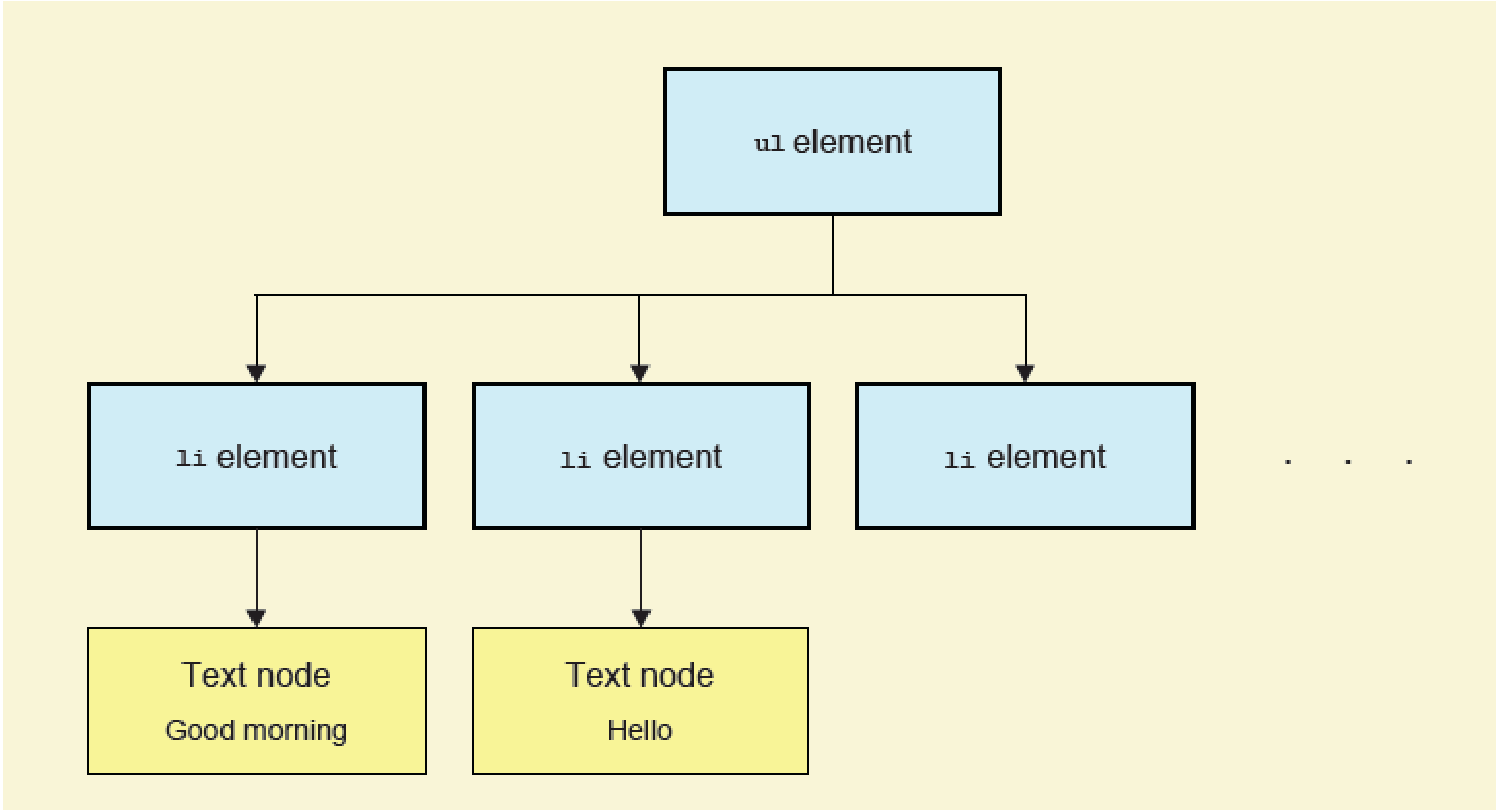
Example Code

```
<!DOCTYPE html>
<html>
    <body>
        <script>
            function myFunction() {
                var the_node=document.getElementById("myList").lastChild;
                var the_clone=the_node.cloneNode();
                document.getElementById("myList").appendChild(the_clone);
            }
        </script>
        <ul id="myList"><li>Good morning</li><li>Hello</li></ul>
        <p>Click on the button to cloneNode()</p>
        <button onclick="myFunction()">Copy it!</button>
    </body>
</html>
```






ul element
li element
li element
Text node
Good morning
Text node
Hello


ul element
li element
li element
li element
. . .
Text node
Good morning
Text node
Hello

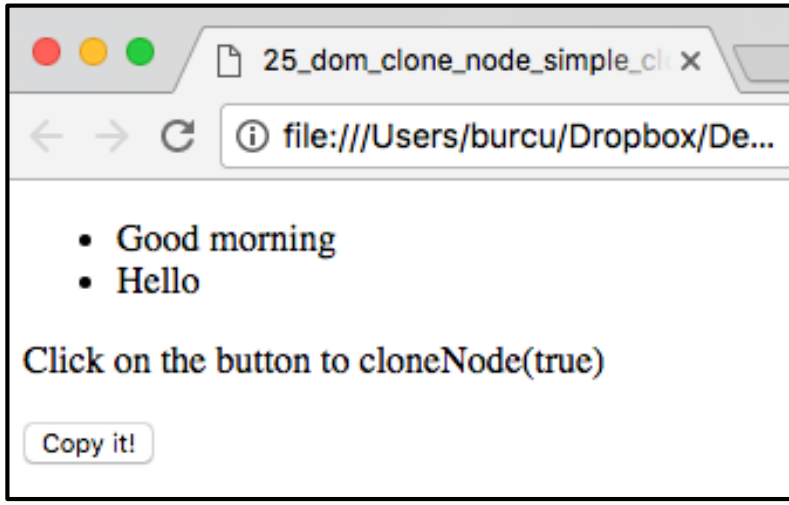
Cloning A Branch

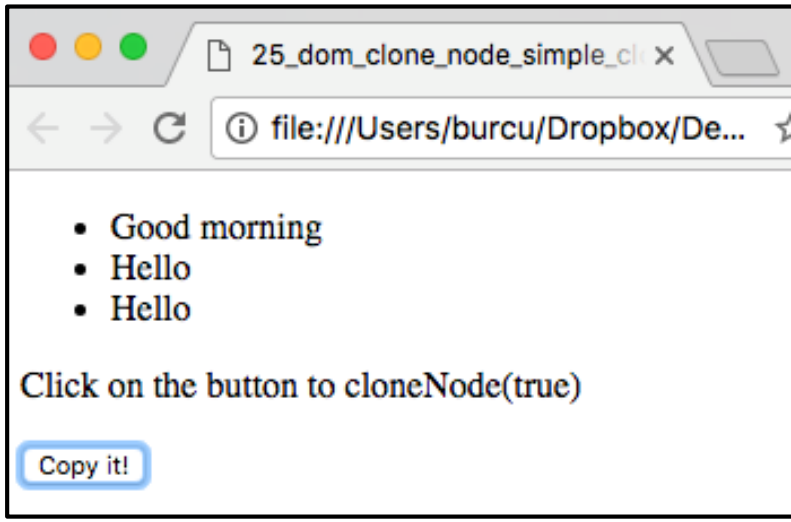
- Use ***node.cloneNode(true)***
- Example:

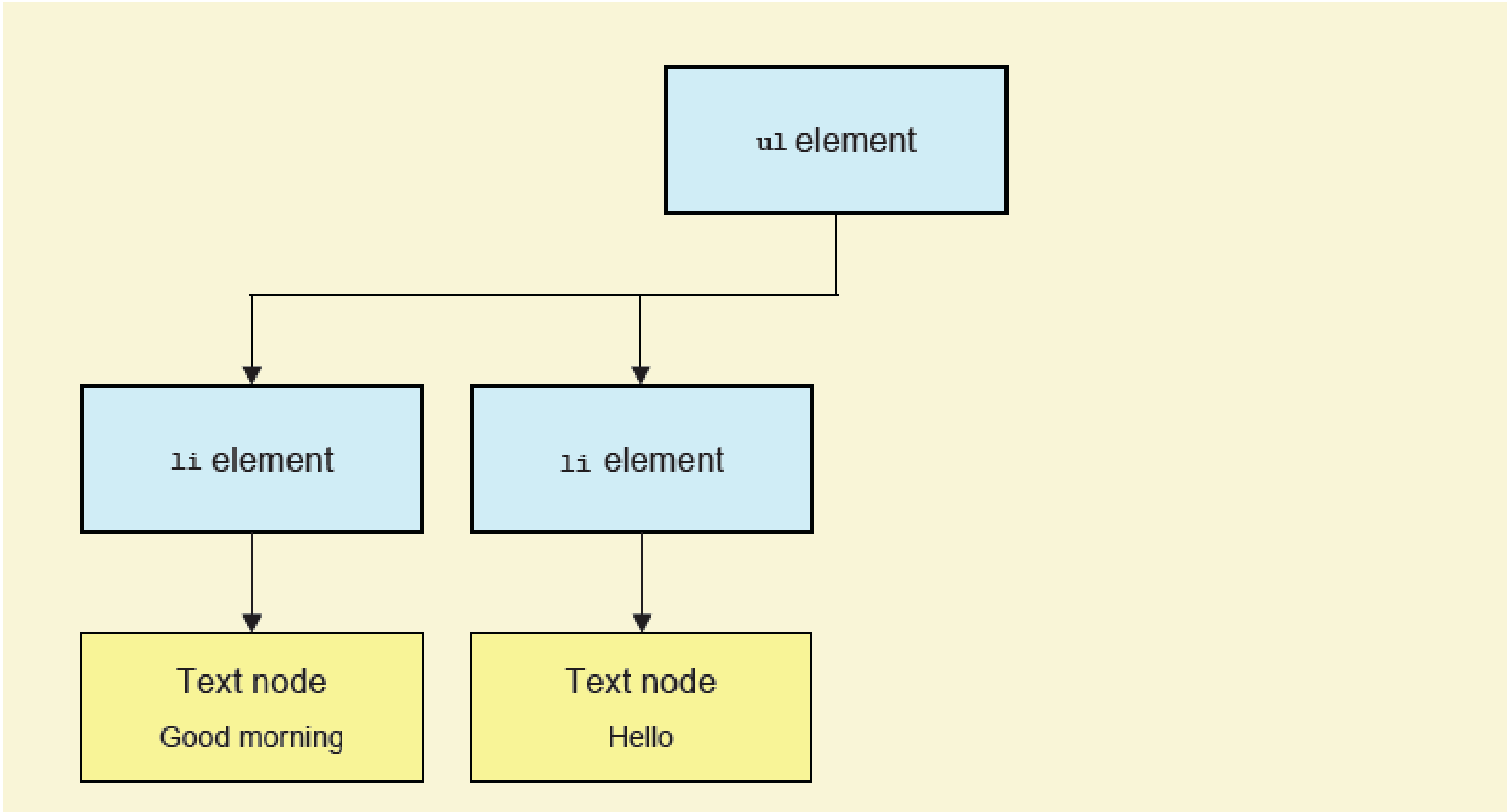
    1) A list item branch <li> with text node child are copied
    2) The copy is then added to the end of the list

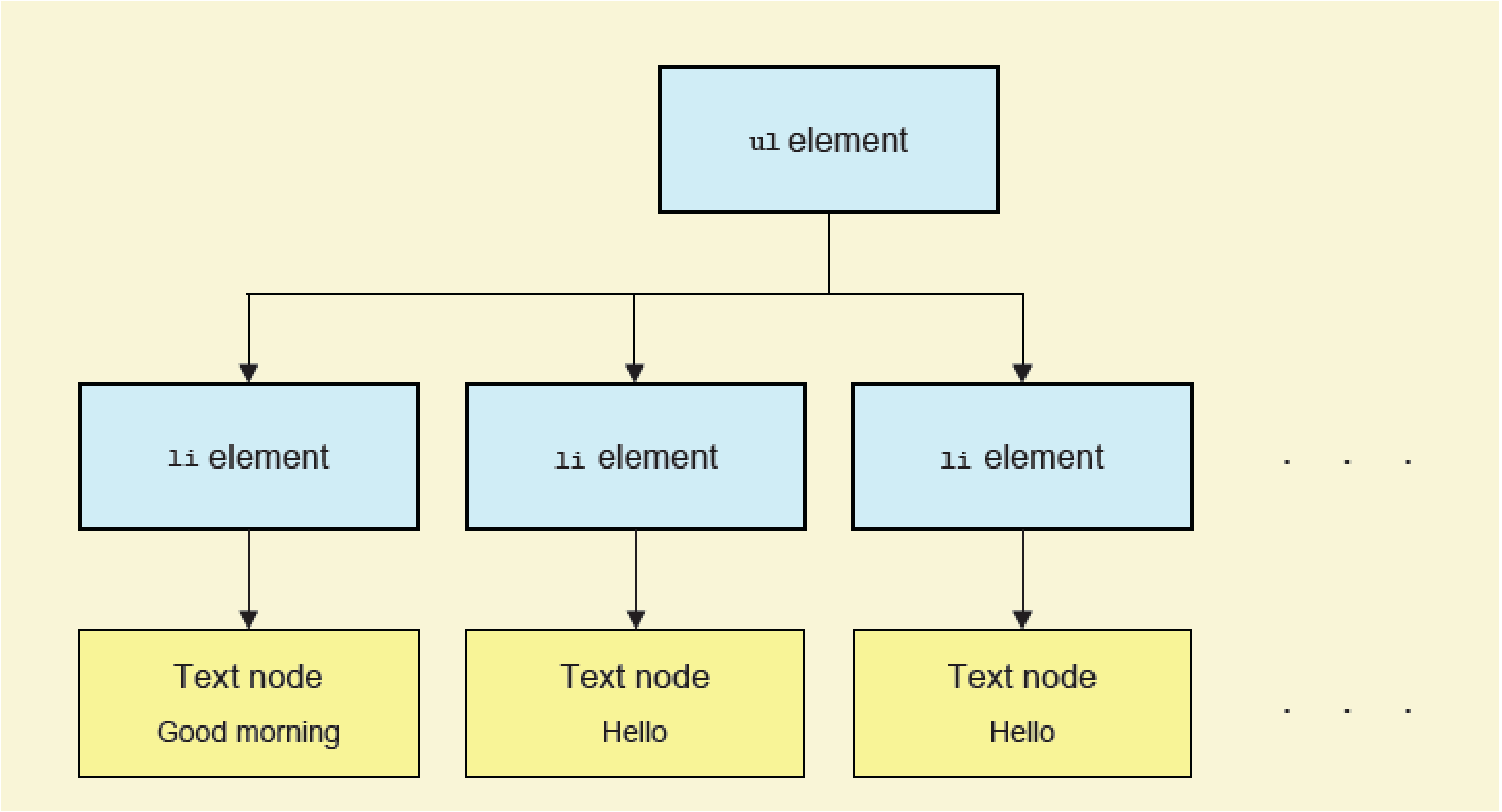
Example Code

```
<!DOCTYPE html>
<html>
    <body>
        <script>
            function myFunction() {
                var the_node = document.getElementById("myList").lastChild;
                var the_clone = the_node.cloneNode(true);
                document.getElementById("myList").appendChild(the_clone);
            }
        </script>
        <ul id="myList"><li>Good morning</li><li>Hello</li></ul>
        <p>Click on the button to cloneNode(true)</p>
        <button onclick="myFunction()">
            Copy it!
        </button>
    </body>
</html>
```






ul element
li element
li element
Text node
Good morning
Text node
Hello


ul element
li element
li element
li element
Text node
Good morning
Text node
Hello
Text node
Hello

Example Code

Copy a element, including all its attributes and child elements, and append it to the document:

```
<!DOCTYPE html>
<html>
    <body>
        <div style="border:1px solid black;background-color:pink">
          <p style="color:red;">A p element</p>
          <p style="color:green;">A p element</p>
          <p style="color:blue;">A p element</p>
        </div>

        <p>Click the button to copy the div element above, including all its
        attributes and child elements, and append it to the document.</p>

        <button onclick="myFunction()">Try it</button>

        <script>
            function myFunction() {
                var elmnt = document.getElementsByTagName("DIV")[0];
                var cln = elmnt.cloneNode(true);
                document.body.appendChild(cln);
            }
        </script>
    </body>
</html>
```

https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_node_clonenode2