香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

FOR LOOPS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll be able to create different kinds of `for` loop

WE WILL LOOK AT

`for`

`for … in`

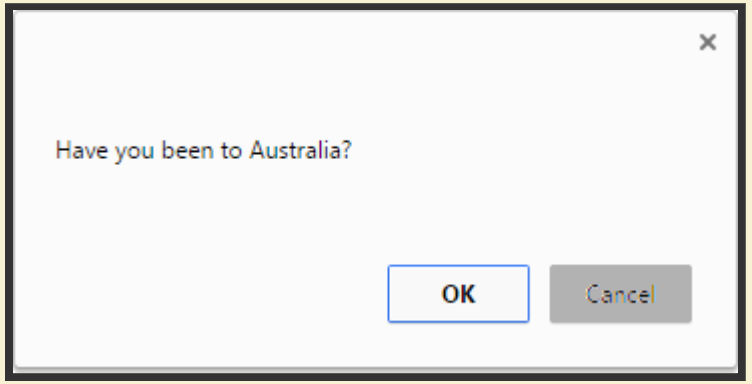
`for … of`

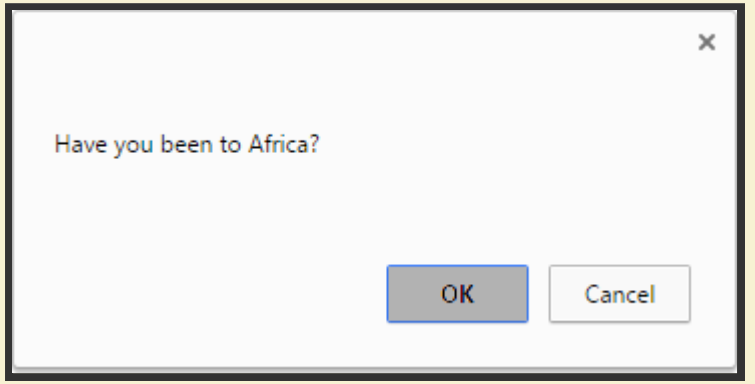
FOR LOOPS

- `for` clearly shows the start and end values
- `for` is especially good for handling a series of data

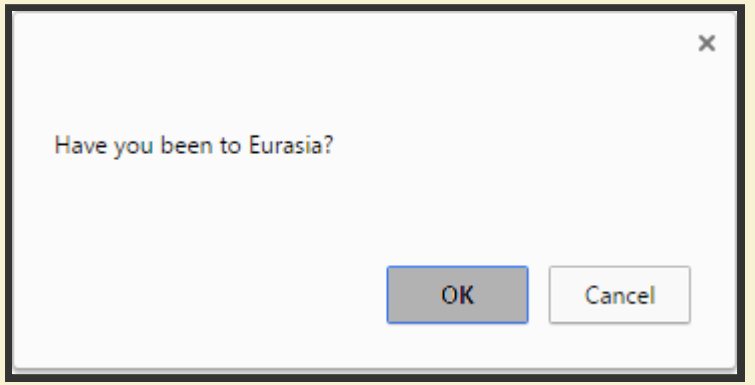
- (*`data_structure`*`.length`
  tells you how many items *data_structure* contains)

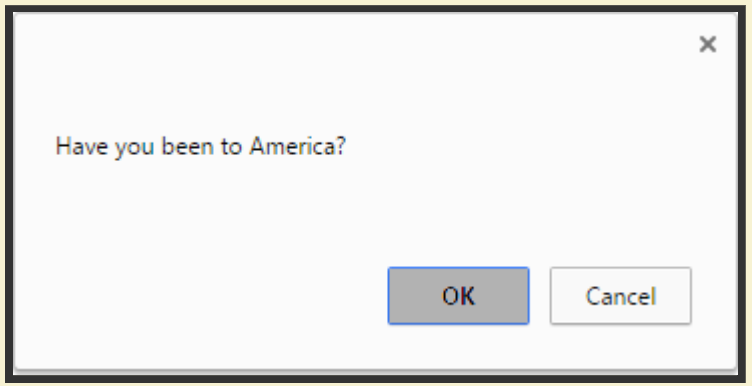
```
<html><head>
  <script>
  var continents=["Australia", "Africa",
      "Antarctica", "Eurasia", "America"];
  var response, count=0;
    for (var index=0; index < continents.length;
                      index++) {
      response = confirm("Have you been to " +
                         continents[index] + "?");
      if (response) count++;
    }
    alert("You have been to " + count +
          " continents!");
  </script>
</head></html>
```

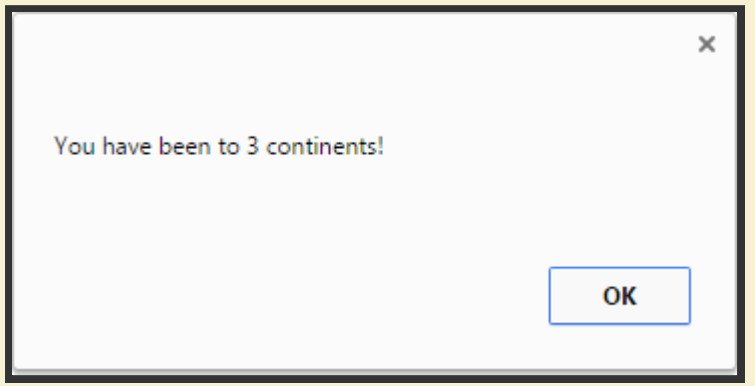

Have you been to Australia?
OK
Cancel


Have you been to Africa?
OK
Cancel


Have you been to Antarctica?
OK
Cancel


Have you been to Eurasia?
OK
Cancel


Have you been to America?
OK
Cancel


You have been to 3 continents!
OK

FOR ... IN LOOPS

- `for ... in` gives you the index of each item

```
<!doctype html>
<html><head>
  <script>
    var continents=["Australia", "Africa",
      "Antarctica", "Eurasia", "America"];
    var response, count=0;
    for (var index in continents) {
      response=confirm("Have you been to "
                + continents[index] + "?");
      if (response) count++;
    }
    alert("You have been to " + count +
          " continents!");
  </script>
</head></html>
```

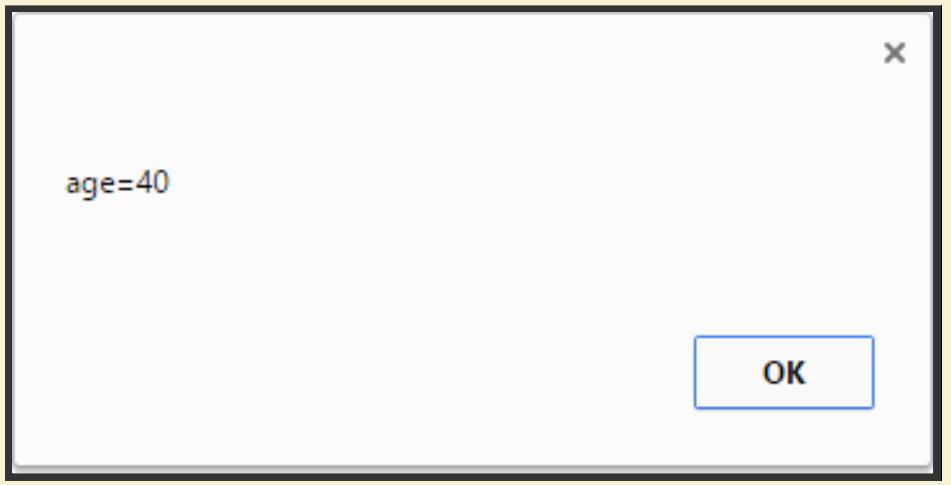
FOR ... IN LOOPS

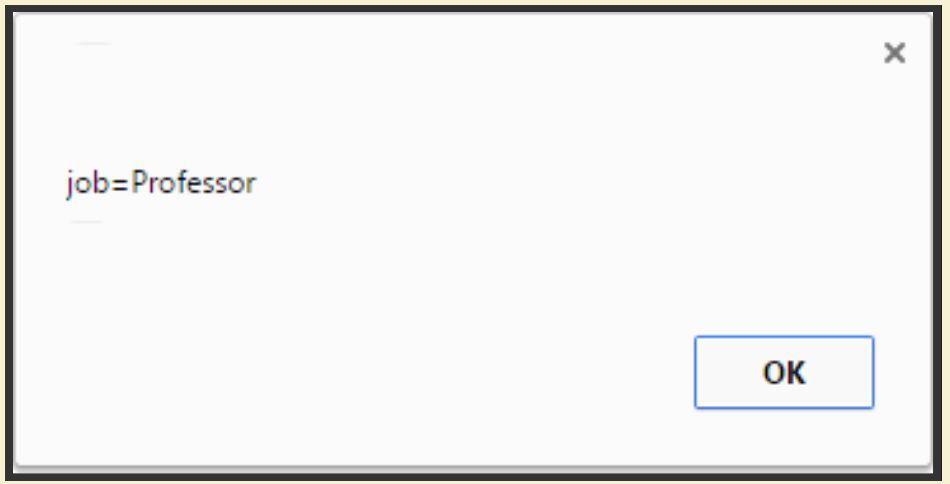
- This example shows how `for ... in` can be used to access the content of a data structure

```
<!doctype html>
<html><head>
  <title>Example of for in</title>
  <script>
  var response, count=0;
  var onePerson = { initials:"DR", age:40,
                    job:"Professor" };

  for (var property in onePerson) {
    alert(property + "=" + onePerson[property]);
  }
  </script>
</head></html>
```


initials=DR
OK


age=40
OK


job=Professor
OK

FOR ... OF LOOPS

- `for ... of` gives you each item

```
<!doctype html>
<html><head>
  <title>Example of for of</title>
  <script>
  var continents=["Australia", "Africa",
      "Antarctica", "Eurasia", "America"];
  var response, count=0;
  for (var continent of continents) {
    response = confirm("Have you been to " +
                       continent + "?");
    if (response) count++;
  }
  alert("You have been to " + count + " continents!");
  </script>
</head></html>
```

OMITTING PARTS

- The 3 parts of the `for` can be omitted
- E.g. this will make an infinite loop:

```
for ( ; ; ) {
    alert("Welcome!");
}
```

This is OK:

```
var number=1;
for ( ; number <= 12; number++ ) {
    alert(number + " times 9 = ", number * 9);
}
```

So is this:

```
for (var rabbits=2, generation=1;
    generation<=12;
    generation++, rabbits *= 2 ) {
        alert("gen: " + generation +
              " total:" + rabbits);
}
```

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

LOOP CONTROL

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll be able to control loops in two new ways

WE WILL LOOK AT

`break`

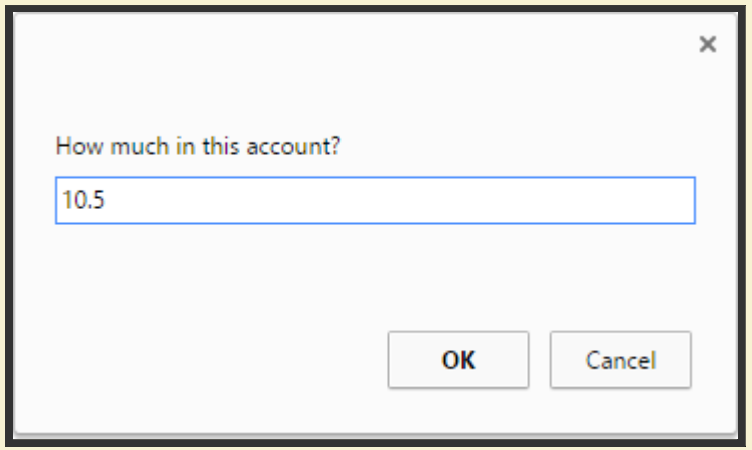
`continue`

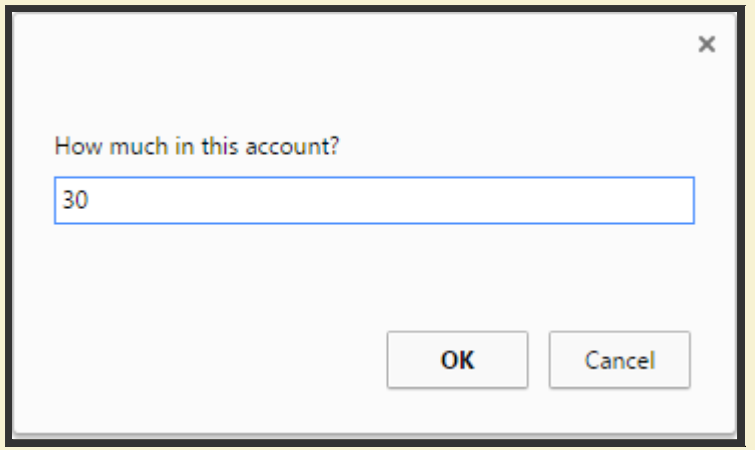
LOOP CONTROL

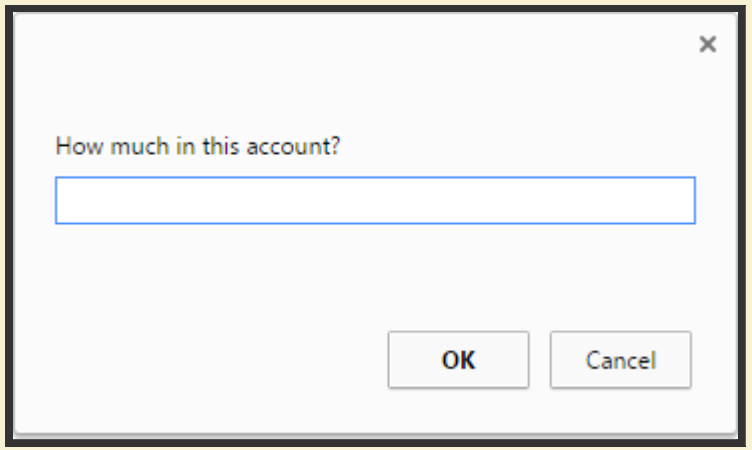
- `break` totally stops the loop
- `continue` stops the current iteration

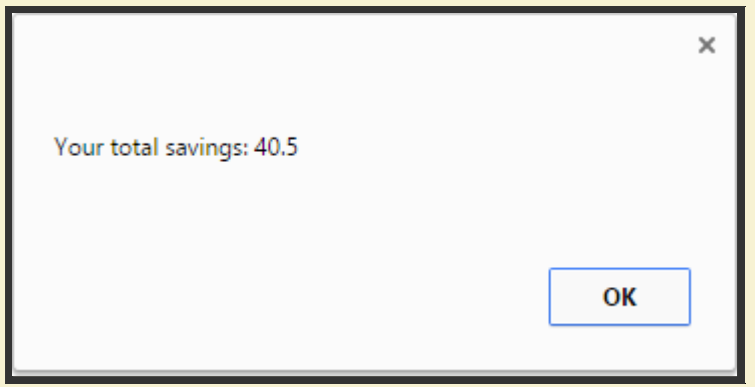
- These apply to both types of loop

```
<!doctype html>
<html><head><script>
  var total_amount=0;
  while (true) {
  this_amount=prompt("How much in this account?");
  this_amount=parseFloat(this_amount);
  if (this_amount>0)
    total_amount+=this_amount;
  else
    break;
  }
  alert("Your total savings: " + total_amount);
</script></head></html>
```


How much in this account?
10.5
OK
Cancel


How much in this account?
30
OK
Cancel


How much in this account?
OK
Cancel

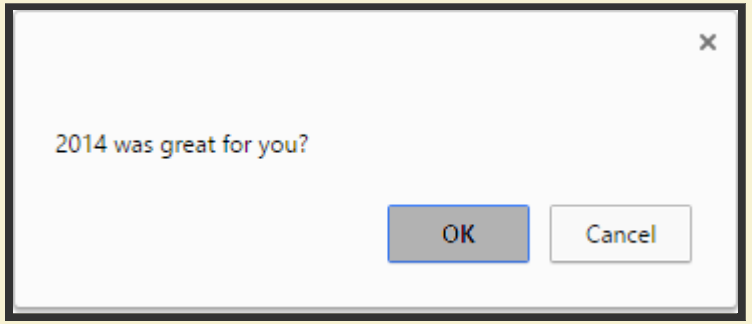

Your total savings: 40.5
OK

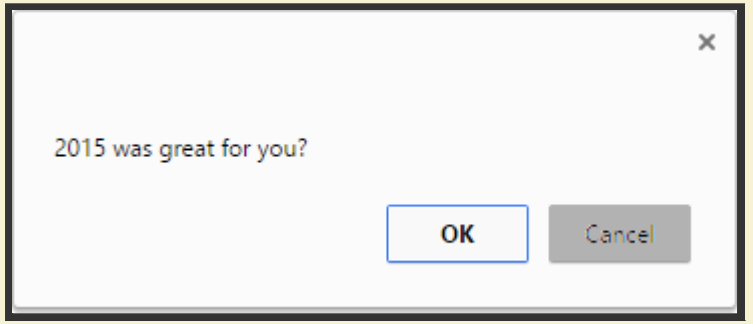
CONTINUE

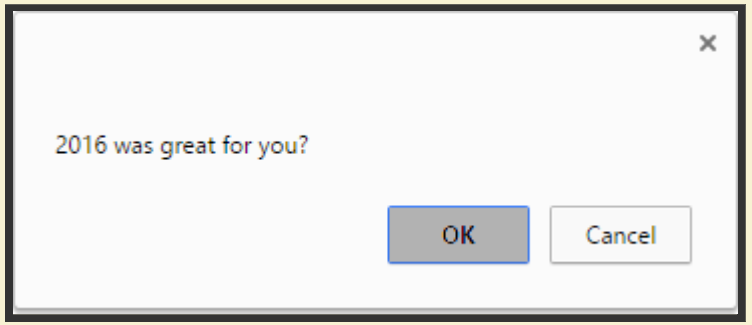
- `continue` skips the current iteration of the loop

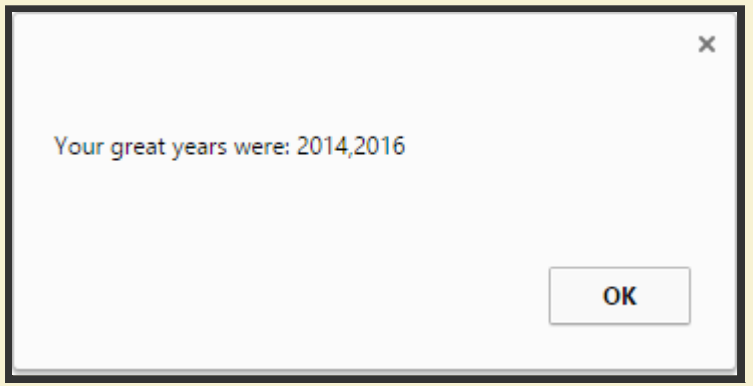
- (*`array`*`.push()` adds an item to the end of *array*)

```
<!doctype html>
<html><head>
  <script>
    var year, great_years = [];
    for (year = 2014; year <= 2016; year++) {
      correct=confirm(year + " was great for you?")
      if (!correct) continue;
      great_years.push(year)
    }
    alert("Your great years were: " + great_years);
  </script>
</head></html>
```


2014 was great for you?
OK
Cancel


2015 was great for you?
OK
Cancel


2016 was great for you?
OK
Cancel


Your great years were: 2014,2016
OK

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

MORE ON ARRAYS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll learn some advanced array functions

ADVANCED ARRAY FUNCTIONS

`sort()`	`indexOf()`	`slice()`
`reverse()`	`lastIndexOf()`	`splice()`

SORTING

- *array*.`sort()` sorts the elements in *array*:

```
var pets = ["Dog", "Cat", "Rabbit", "Hamster"];
pets.sort();
// Now pets is ["Cat", "Dog", "Hamster", "Rabbit"]
```

REVERSE

- *array*.`reverse()` reverses *array*
- The first element becomes the last;
  The last element becomes the first

```
var pets = ["Dog", "Cat", "Rabbit", "Hamster"];
pets.reverse();
// pets is ["Hamster", "Rabbit", "Cat", "Dog"]
```

DESCENDING ORDER

- By combining `sort()` and `reverse()`, you can sort things in descending order:

```
var pets = ["Dog", "Cat", "Rabbit", "Hamster"];
pets.sort().reverse();
// pets is ["Rabbit", "Hamster", "Dog", "Cat"]
```

FINDING AN ELEMENT

- Use *array*.`indexOf(`*target*`)` to find the index of the first occurence of *target* in *array*:

```
var pets = ["Dog", "Cat", "Rabbit", "Hamster"];
alert(pets.indexOf("Rabbit")); // This shows 2
```

- If *target* is not in *array*, `indexOf()` will return **-1**

MORE ON FINDING AN ELEMENT

- Pass a second value to `indexOf()` to control where to start the search

```
array.indexOf(target, startPosition)
```

```
<html><body><script>
  var pets = ["Dog", "Cats", "Rabbit", "Hamster",
          "Rabbit", "Rabbit", "Dog", "Cat",
          "Hamster", "Hamster", "Rabbit"];
  var rabbitPositions = [], startSearchAt = 0;
  do {
    foundAt = pets.indexOf("Rabbit", startSearchAt);
    if(foundAt != -1) {
      rabbitPositions.push(foundAt);
      startSearchAt = foundAt + 1;
    }
  } while(foundAt != -1);
  alert(rabbitPositions); // This shows [2, 4, 5, 10]
</script></body></html>
```

FINDING ELEMENT BACKWARDS

- Use *array*.`lastIndexOf(`*target*`)` to find *target* in *array*, starting from the last element in *array*:

```
var pets = ["Rabbit", "Dog", "Cat",
            "Rabbit", "Hamster"];
alert(pets.lastIndexOf("Rabbit")); // This shows 3
```

SLICE()

- Extract part of an array by *array*.`slice`(*startPosition*):

```
var pets = ["Dog", "Cat", "Rabbit", "Hamster"];
var result = pets.slice(1);
// result is ["Cat", "Rabbit", "Hamster"]
```

- You can also set where to stop, by *array*.`slice`(*startPosition*, *endPosition*):

```
var pets = ["Dog", "Cat", "Rabbit", "Hamster"];
var result = pets.slice(1, 3);
// result is ["Cat", "Rabbit"]
```

REMOVE SOMETHING ANYWHERE IN AN ARRAY

- `splice()` is used when you want to remove element(s) anywhere from an array
- To remove element(s) anywhere from an array, use *array*.`splice`(*position*, *quantity*)

```
var pets = ["Dog", "Cat", "Rabbit", "Hamster"];
var result = pets.splice(1, 1);
// Now pets is ["Dog", "Rabbit", "Hamster"]
// and result is ["Cat"]
```

- `splice()` returns the removed element(s)

ADD SOMETHING ANYWHERE IN AN ARRAY

- `splice()` can also be used when you want to add element(s) anywhere to an array
- To add an element anywhere to an array, use *array*.`splice`(*position*, *0*, *element*)

```
var pets = ["Dog", "Cat", "Hamster"];
var result = pets.splice(2, 0, "Rabbit");
// Now pets is ["Dog", "Cat", "Rabbit", "Hamster"]
// and result is []
```

- Because nothing is removed from *pets*, *result* is [ ]

REPLACE SOMETHING ANYWHERE IN AN ARRAY

- To replace element(s) anywhere in an array, use *array*.`splice`(*position*, *quantity*, *element(s)*)

```
var pets = ["Dog", "Cat", "Hamster"];
var result = pets.splice(1, 1, "Rabbit", "Fish");
// Now pets is ["Dog", "Rabbit", "Fish", "Hamster"]
// and result is ["Cat"]
```

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

ARRAY FUNCTIONS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll learn some more advanced array functions

WE'LL LOOK AT

`forEach()`

`map()`

FOREACH()

- You can go through every element using loop (`for` / `while`)

```
var pets = ["Dog", "Cat", "Hamster"];
for(var i = 0; i < pets.length; i++) {
    alert(pets[i]);
}
```

- You can also use *array*.`forEach`(*function*):

```
var pets = ["Dog", "Cat", "Rabbit"];
pets.forEach(alert);
// This shows 3 separate alerts
```

MORE ON FOREACH()

- You can think of `forEach()` in this way:

```
function forEach(theArray, fn) {
  for(var i = 0; i < theArray.length; i++) {
    fn(theArray[i], i, theArray);
  }
}
```

- So, your function should look like this,
  if you need all of the 3 things:

```
function yourFunction(element, index, array) {}
```

```
<!doctype html>
<html>
<body>
   <script>
      var numbers = [1, 2, 3, 4, 5];
      numbers.forEach( function(elem, idx, arr) {
        arr[idx] = elem * elem;
      });
      alert(numbers); // This shows [1,4,9,16,25];
   </script>
</body>
</html>
```

MAP()

- `map(`*function*`)` stores the result of each execution of *function* into an array it returns.
  You can think of `map( )` in this way:

```
function map(theArray, fn) {
  var results = [];
  for(var i = 0; i < theArray.length; i++) {
    results.push(fn(theArray[i], i, theArray));
  }
  return results;
}
```

```
<!doctype html>
<html>
<body>
   <script>
      var square = function(el) { return el * el; }
      var numbers = [1, 2, 3, 4, 5];
      var results = numbers.map(square);
      alert(results); // This shows [1,4,9,16,25];
   </script>
</body>
</html>
```

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

THE DOM - BASIC CONCEPTS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll appreciate the concept of a DOM
- You'll understand the role of text nodes
- You'll appreciate the concept of whitespace and how it is stored

THE DOM

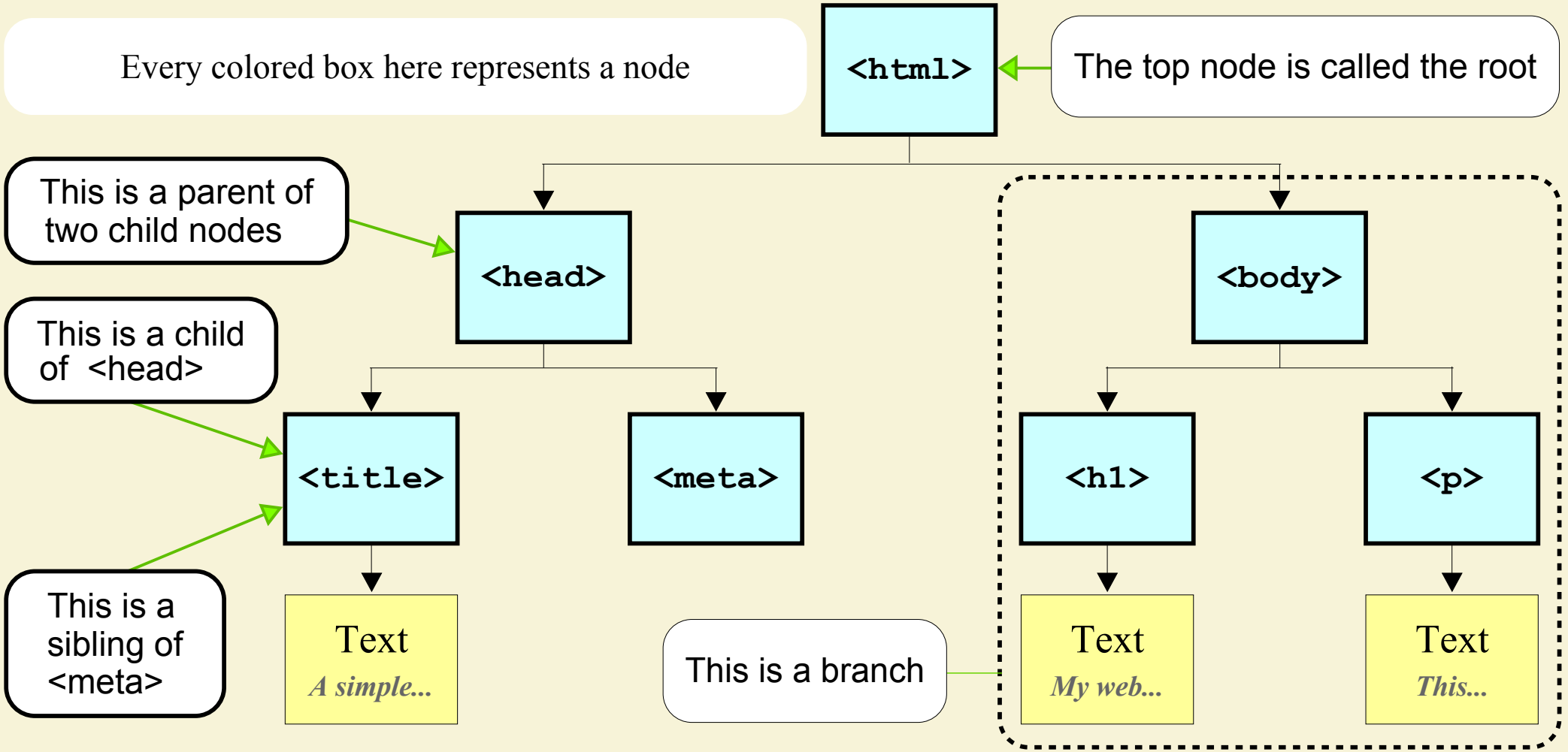
DOM means *Document Object Model*

When you load something into a browser,
it is converted into a DOM structure

CODE WE HAVE SEEN

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

WHAT THE PAGE LOOKS LIKE

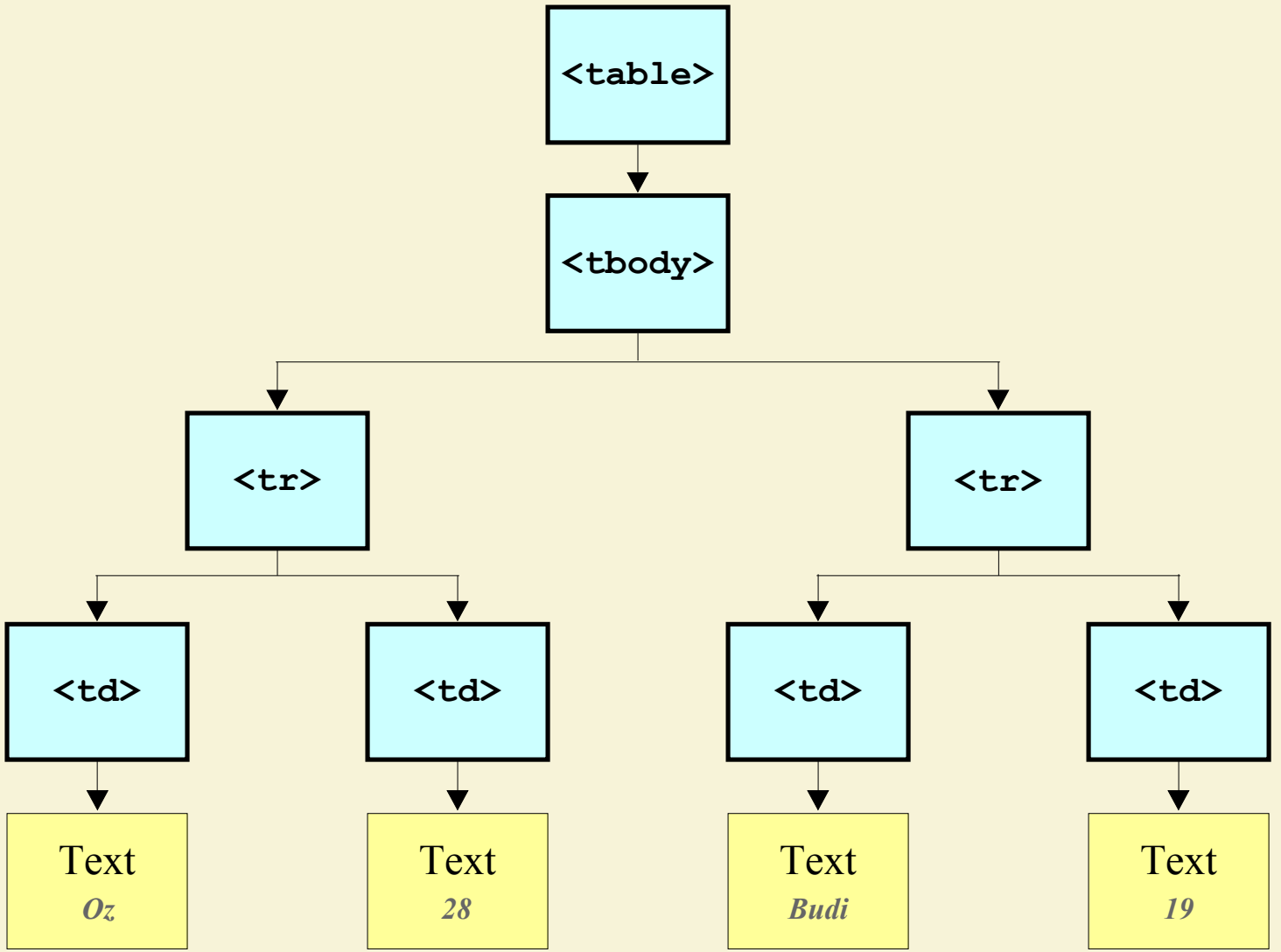

Every colored box here represents a node
<html>
The top node is called the root
This is a parent of two child nodes
<head>
<body>
This is a child of <head>
<title>
<meta>
<h1>
<p>
This is a sibling of <meta>
Text
A simple...
This is a branch
Text
My web...
Text
This...

```
<!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>
```

WHAT THE PAGE LOOKS LIKE

Oz 28

Budi 19


<table>
<tbody>
<tr>
<tr>
<td>
<td>
<td>
<td>
Text
Oz
Text
28
Text
Budi
Text
19

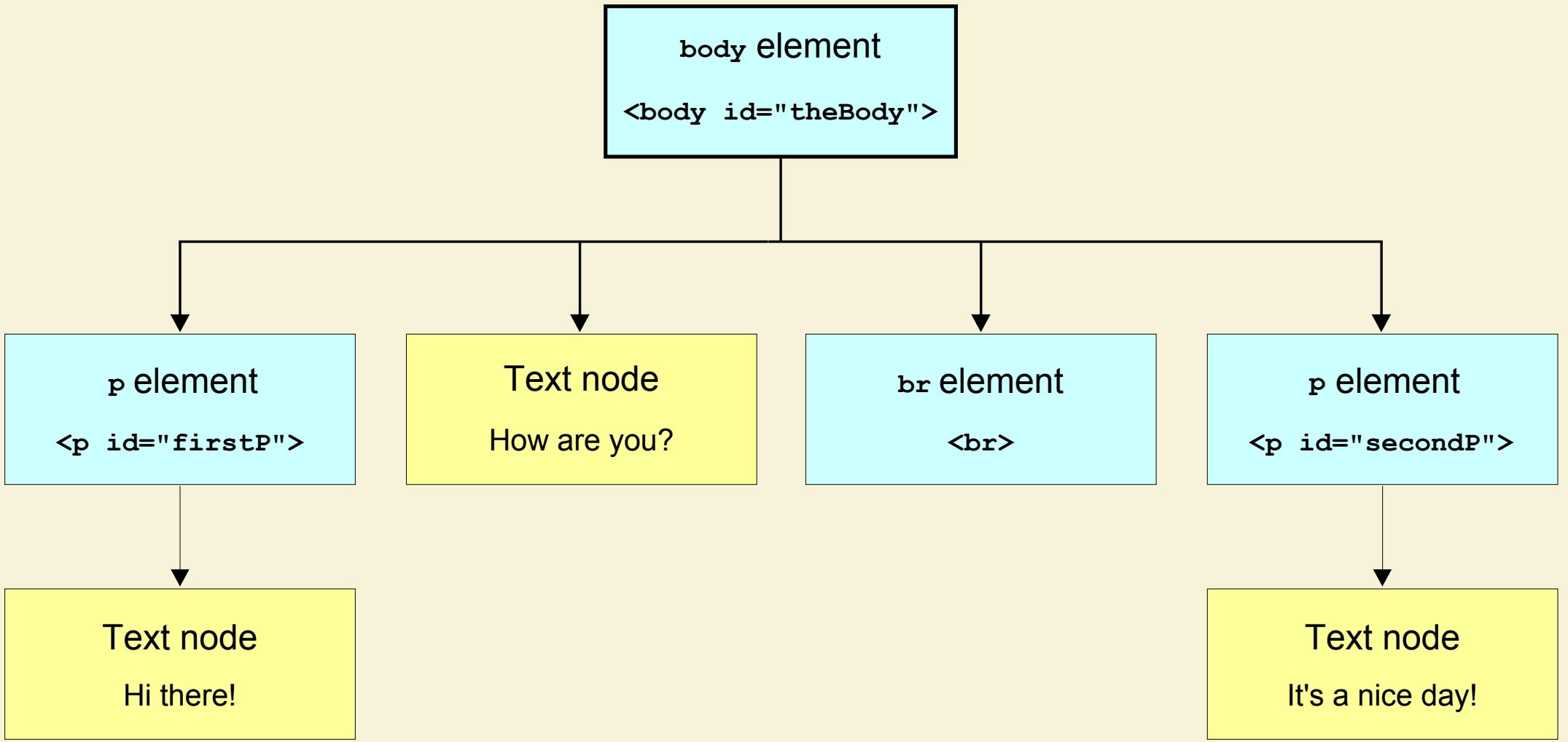
```
<!DOCTYPE html>
<html>
<body id="theBody"><p id="firstP">
Hi there!
</p>
How are you?
<br>
<p id="secondP">
It's a nice day!
</p>
</body>
</html>
```

WHAT THE PAGE LOOKS LIKE

Hi there!

How are you?

It's a nice day!


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!

WHITESPACE NODES

- Whitespace is anything you can't see i.e. spacing
- There may be a text node which contains only whitespace
- These is called a 'whitespace node'
- These are sometimes troublesome

A COMPARISON

```
<body><p>Hello.</p>
```

does not have a whitespace node between <body> and <p>

```
<body>
<p>Hello.</p>
```

does have a whitespace node between <body> and <p>

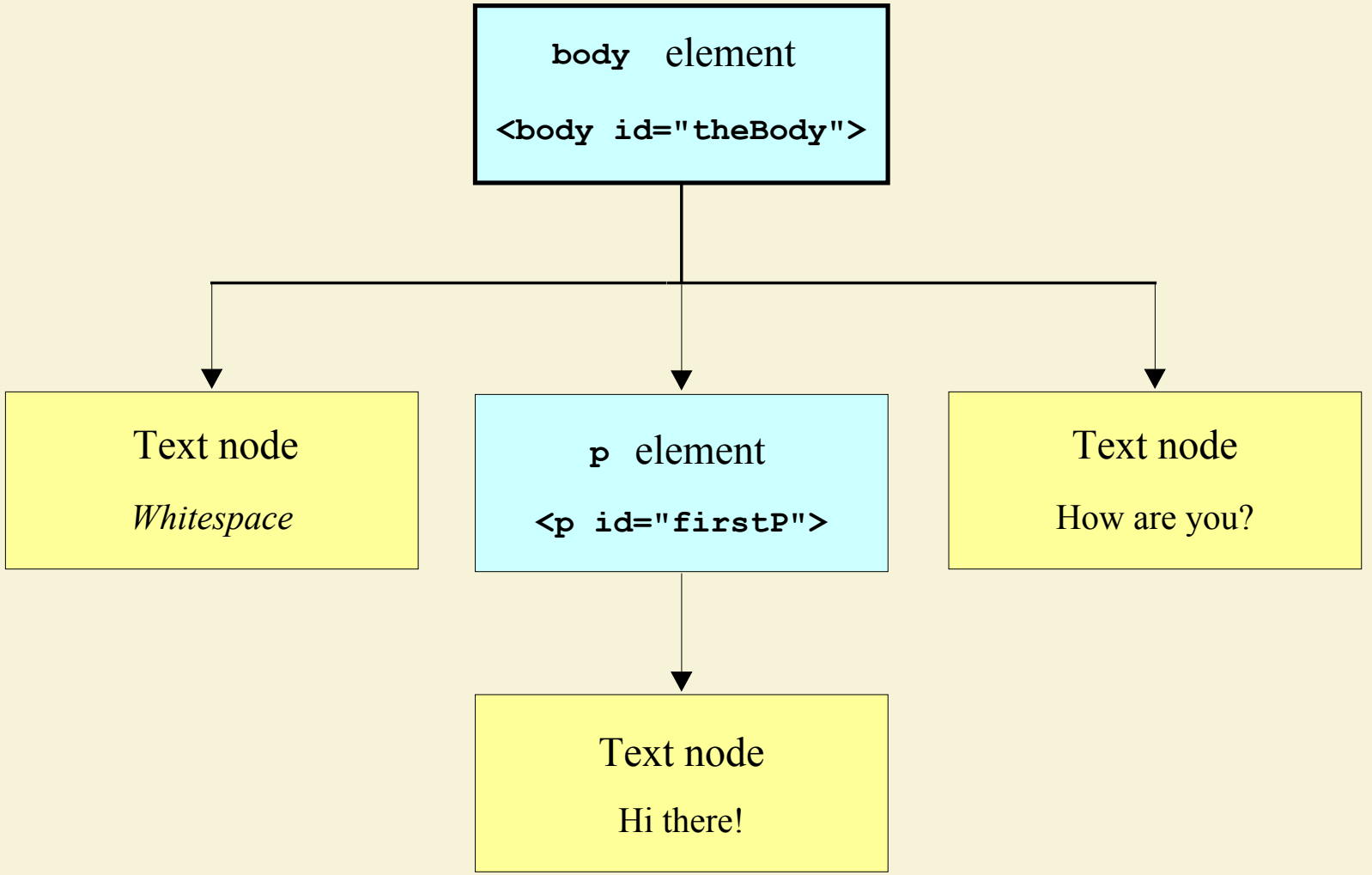
EXAMPLE OF A WHITESPACE NODE

```
<!DOCTYPE html>
<html>
<body id="theBody">
<p id="firstP">
Hi there!
</p>
How are you?
</body></html>
```

WHAT THE PAGE LOOKS LIKE

Hi there!

How are you?


body element
<body id="theBody">
Text node
Whitespace
p element
<p id="firstP">
Text node
How are you?
Text node
Hi there!

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

NODE RELATIONSHIPS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll understand the relationship between nodes
- You'll be able to visualize the path for a node
- You'll appreciate the use of event handlers

WE WILL LOOK AT

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

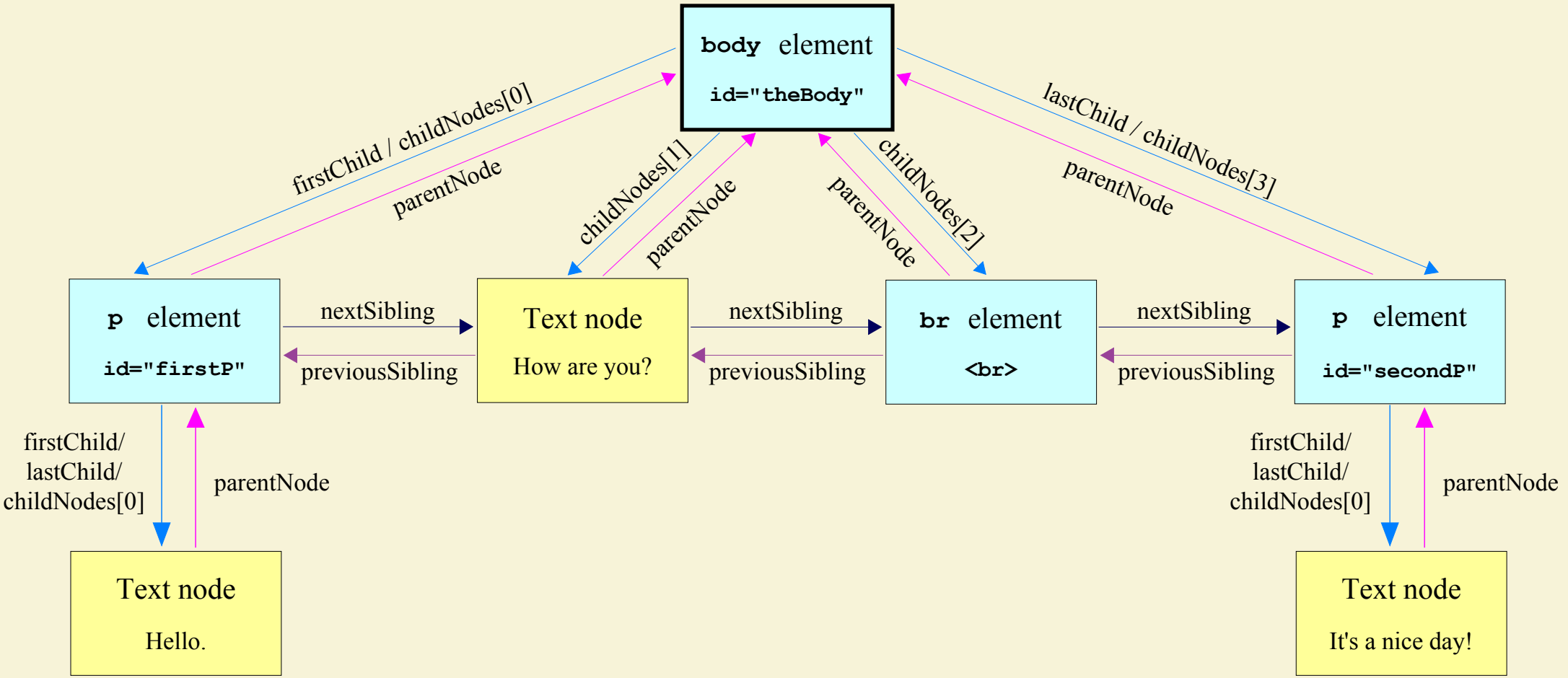
NODE RELATIONSHIPS

How is one node related to another?

Specific code lets you use such relationship

NODE RELATIONSHIPS

- Your code can access all of these:
  - `parentNode`
  - `childNodes[]`, `firstChild`, `lastChild`
  - `previousSibling`, `nextSibling`


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?

In the next example we 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 handleClick(event) {
  event.stopPropagation();

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

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

  alert(thisPath);
}
```

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

Two examples follow: HTML and SVG

They use the same code

Event handlers are added 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)
    attachHandler(node.childNodes[i]);
}
```

Click on the link or form elements to see the DOM path to that node

(The html that you see below is just some 'random' html which helps to demonstrate the technique).

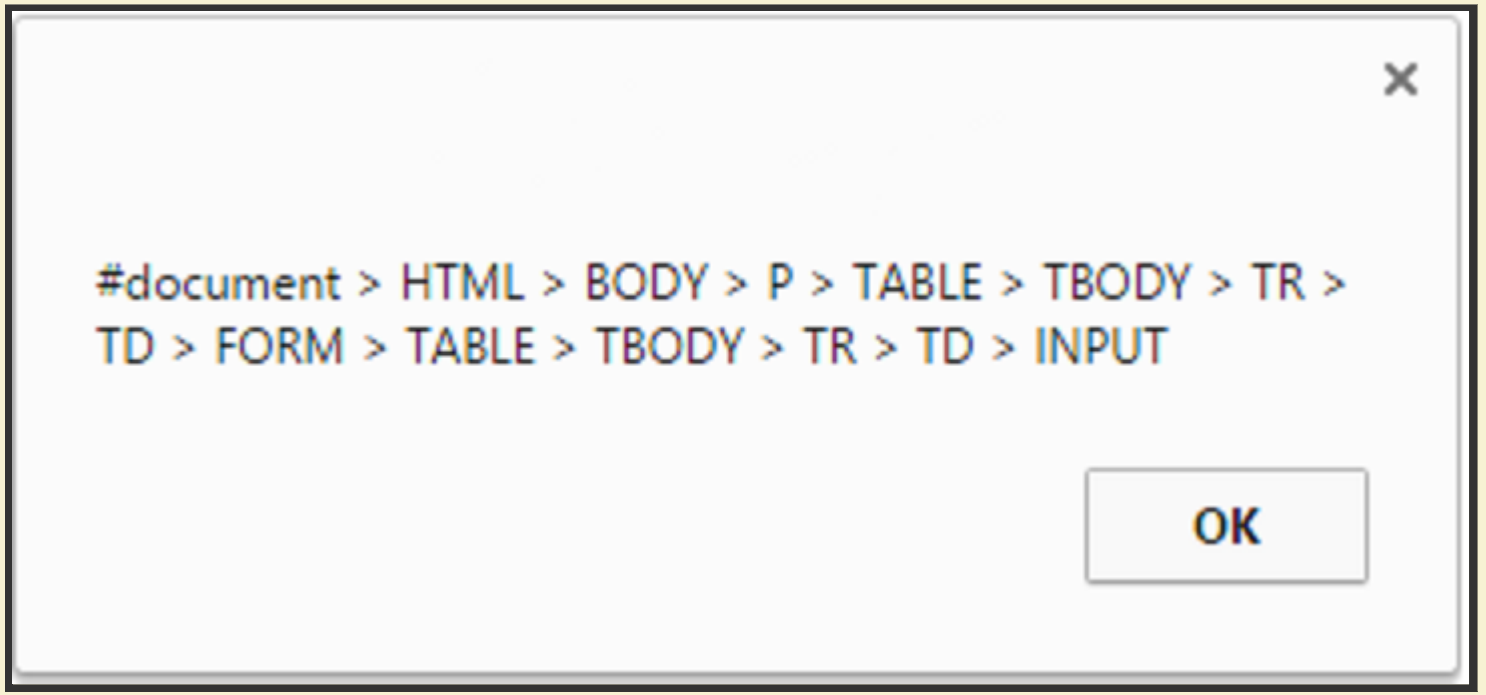
LISTS

- List 1
- List 2
- List 3
- List 4
    1. Order List 1
    2. Order List 2
    3. Order List 3
    4. Order List 4

TABLES

Cell 1	Cell 2	Cell 3	Cell 4
		Click!	

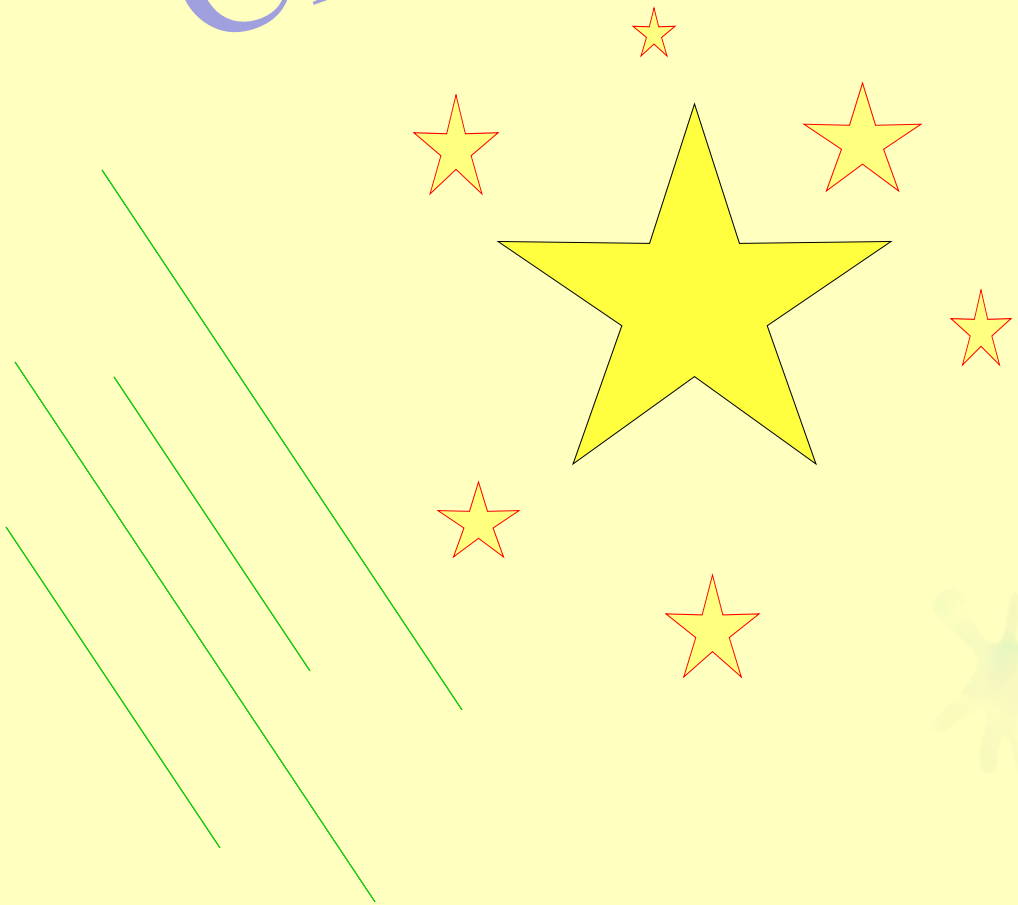
HTML RESULT

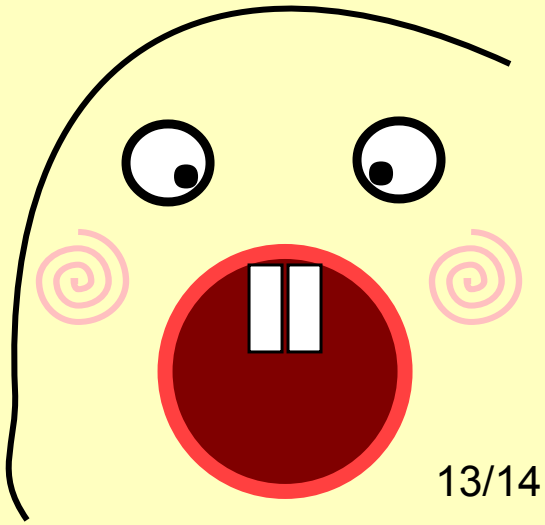

#document > HTML > BODY > P > TABLE > TBODY > TR >
TD > FORM > TABLE > TBODY > TR > TD > INPUT
OK

Click on any object to see the DOM path to that node

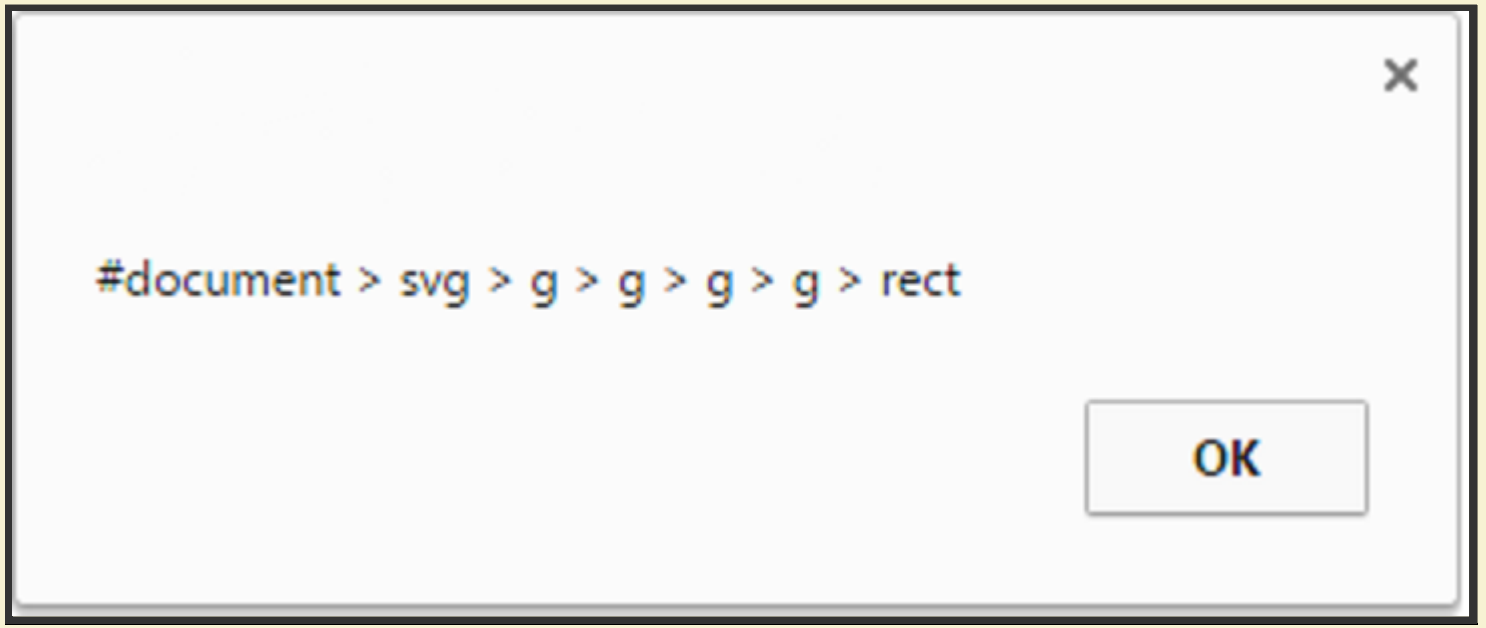
Click Me!

Please click ME!





SVG RESULT



香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

LOCATING NODES

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll appreciate how to find a node in the DOM
- You'll appreciate how to set the attribute of a node

WE WILL LOOK AT

`getElementsByTagName()`

`getElementById()`

`setAttribute()`

THE DOM

Everything is in the DOM

We can add/delete/copy/change anything

To do this, we need to understand how to access things

USING THE EXACT PATH

- Method 1: Use the exact DOM path
  - Sometimes hard to work out the exact path
  - Easy to make mistakes
  - In another browser the DOM may be slightly different - code fails!

USING THE TYPE

- Method 2: Use `getElementsByTagName()`
  - 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

USING THE NAME

- Method 3: Use `getElementById()`
  - If you give a node a unique name e.g.

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

    then this method is the easiest way

```
<body>
<h2 style="color:black" id="cute_text">
Click on a button to change the colour
</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

```
<!DOCTYPE html>
<html> <head>
<script>
function change_color1(){
  document.childNodes[0].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>
```

Click on a button to change the colour of this text

Change using method 1 Change using method 2

Change using method 3

Click here to open the example

SETATTRIBUTE()

This is a common way to change something

For example:

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

```
<!DOCTYPE html>
<html> <head>
<script>
function change_color1(){
  document.childNodes[0].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>
```

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

CREATING AND ADDING NODES

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll be able to create different types of nodes
- You'll be able to add nodes to a DOM

WE WILL LOOK AT

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

ADDING TO THE WEB PAGE

First, create whatever you want to add

- whatever you create is not yet in the DOM

Second, add it at the desired place

CREATING NODES

Use `createElement()` e.g.

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

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

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

ADDING NODES - INSERTBEFORE()

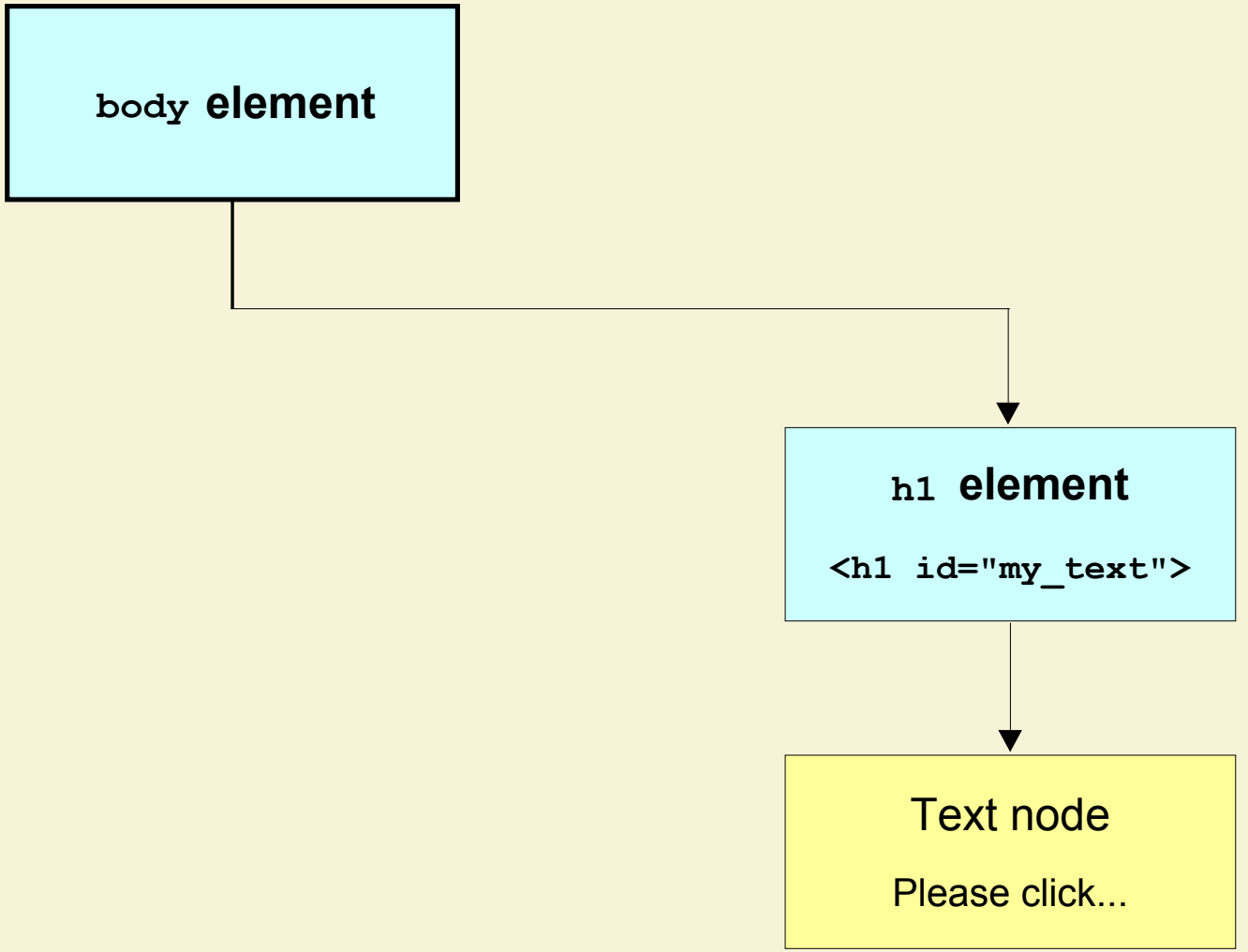
For example:

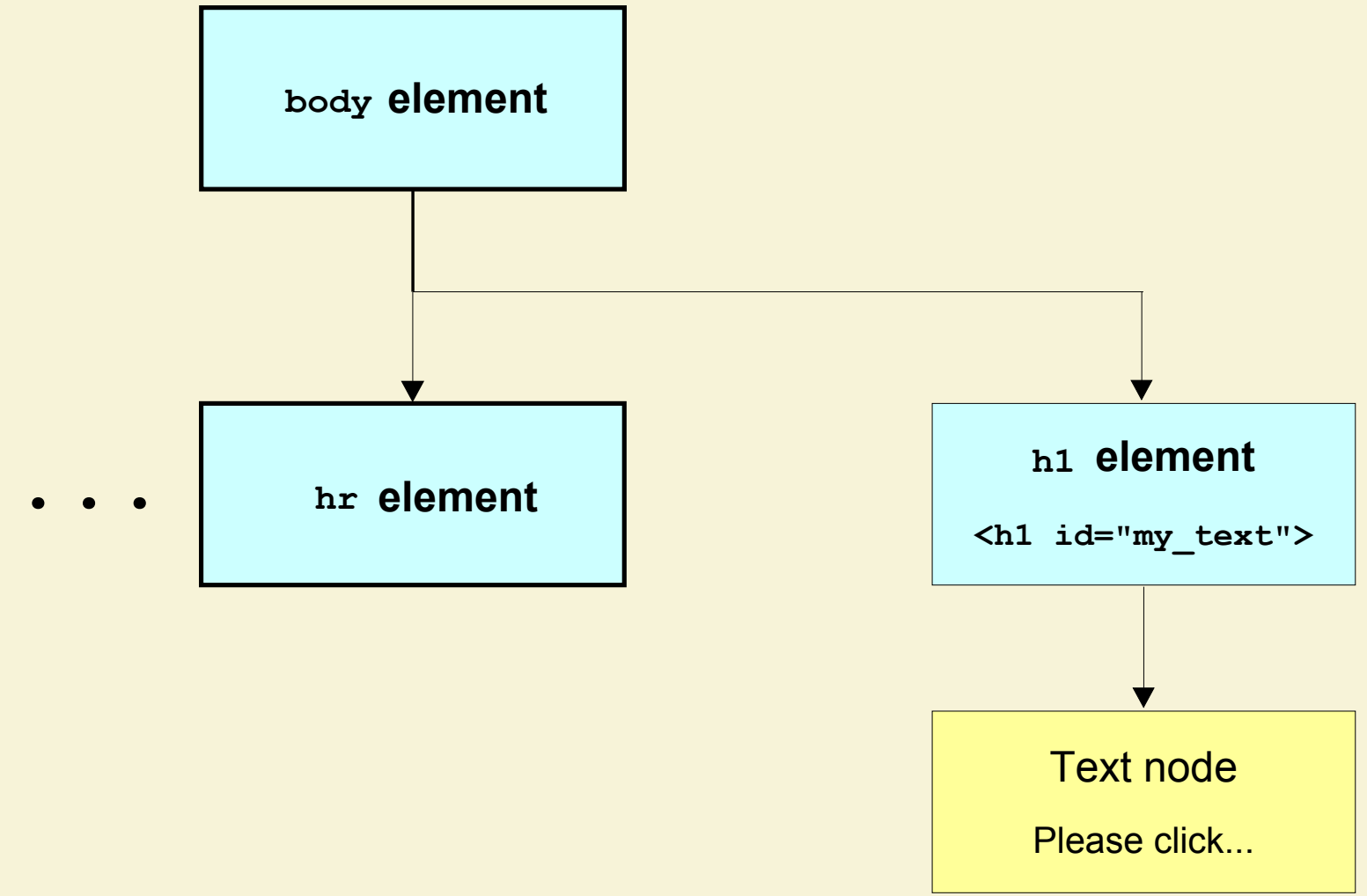
```
newItem = document.createElement("hr");

destParent = document
             .getElementsByTagName("body")[0];
destParent.insertBefore(
             newItem, destParent.firstChild);
```

Please click on the page

Click here to open the example


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...

```
<!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>
```

ADDING NODES - APPENDCHILD()

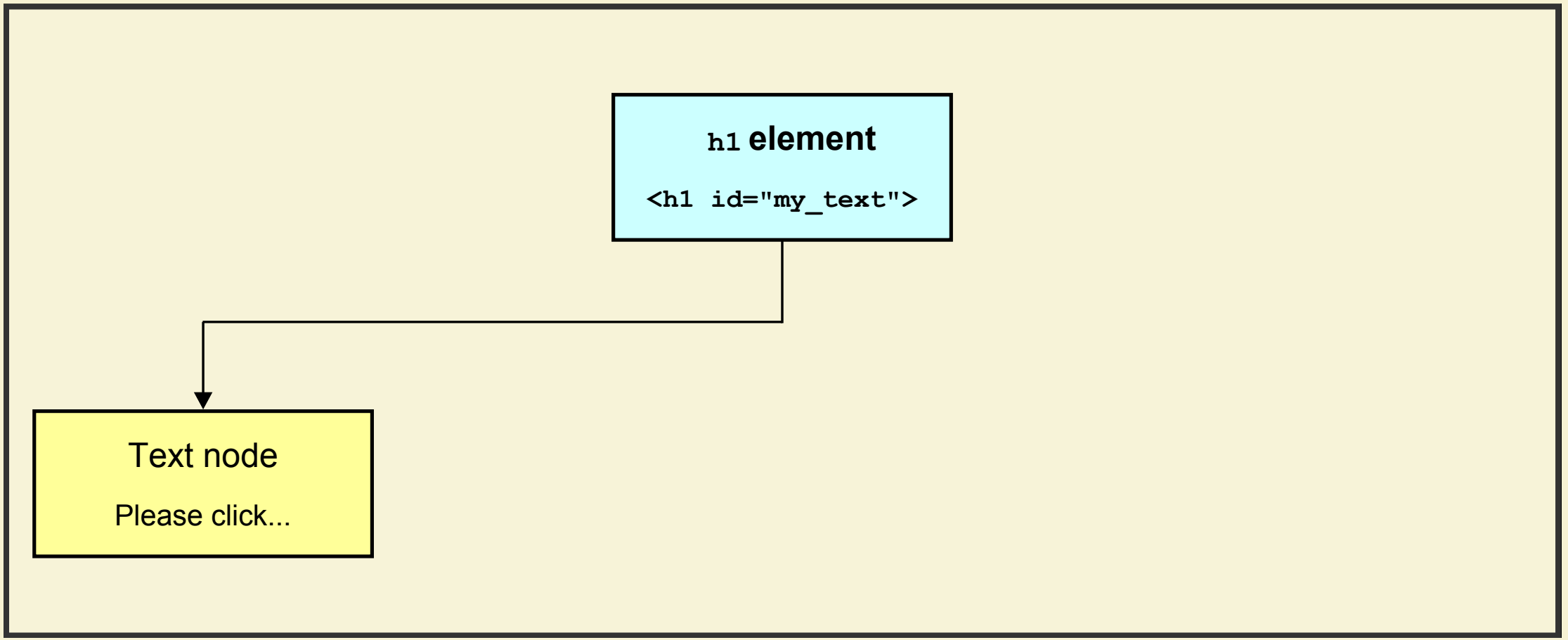
For example:

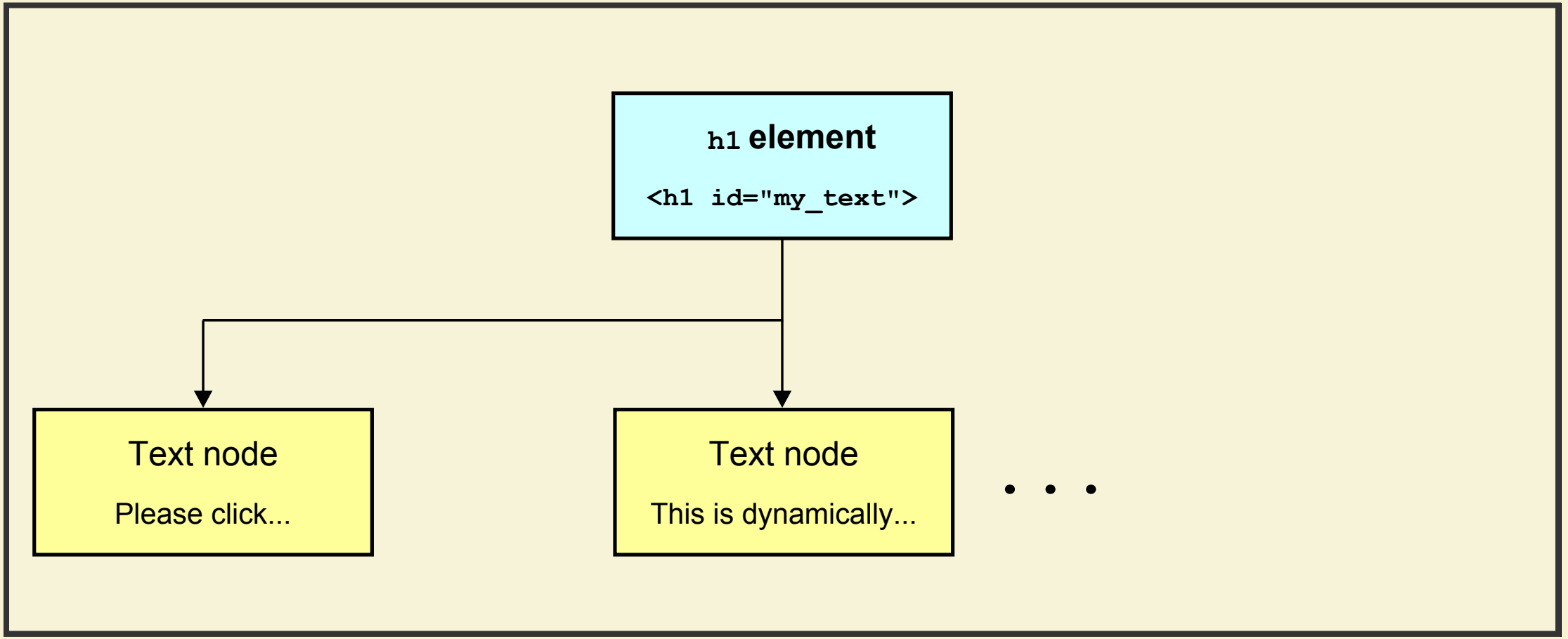
```
result=document.createTextNode(
            "This is dynamically added Text!");

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

Please click on the page

Click here to open the example


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...
• • •

```
<!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>
```

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

DELETING NODES

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll be able to delete node from a DOM

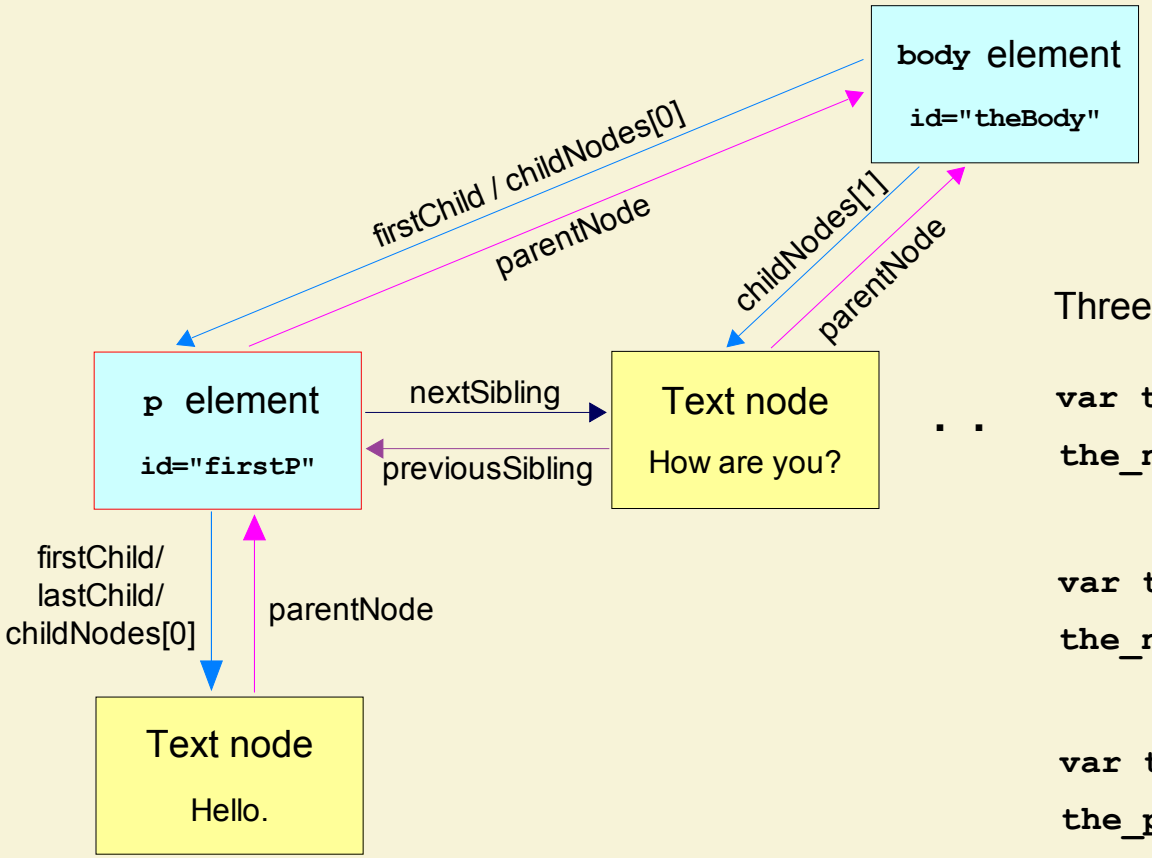
WE WILL LOOK AT

Removing a node `removeChild()`

DELETING A NODE

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

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



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);
```

Hello.

How are you?

It's a nice day!

Example 1 Example 2 Example 3

Reload the page to reset the DOM.

Click here to open the example

```
<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" onclick="delete3()">Example 3</button>
<p>
Reload the page to reset the DOM.
</p>
</body>
</html>
```

```
<!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>
```

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

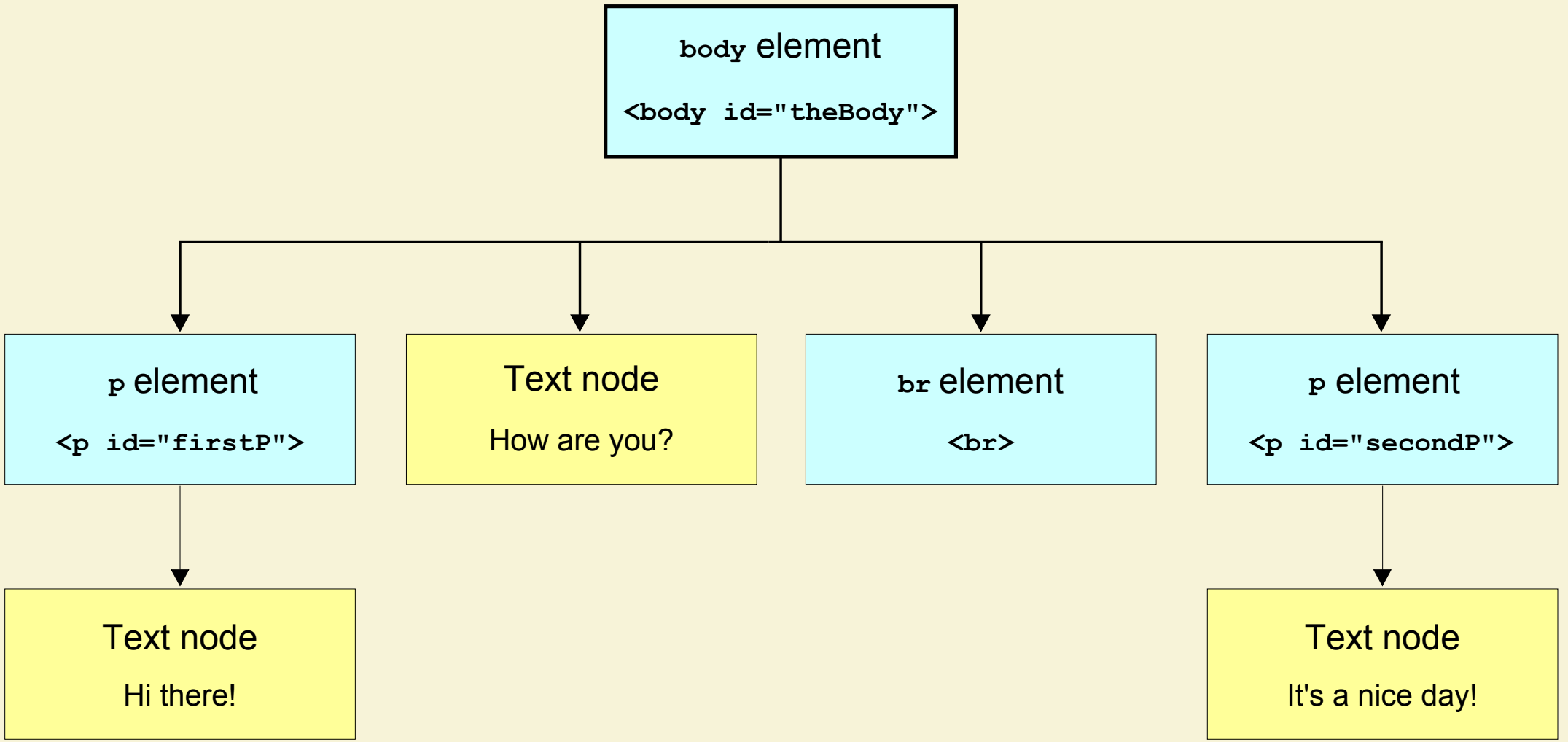
Hello.

How are you?

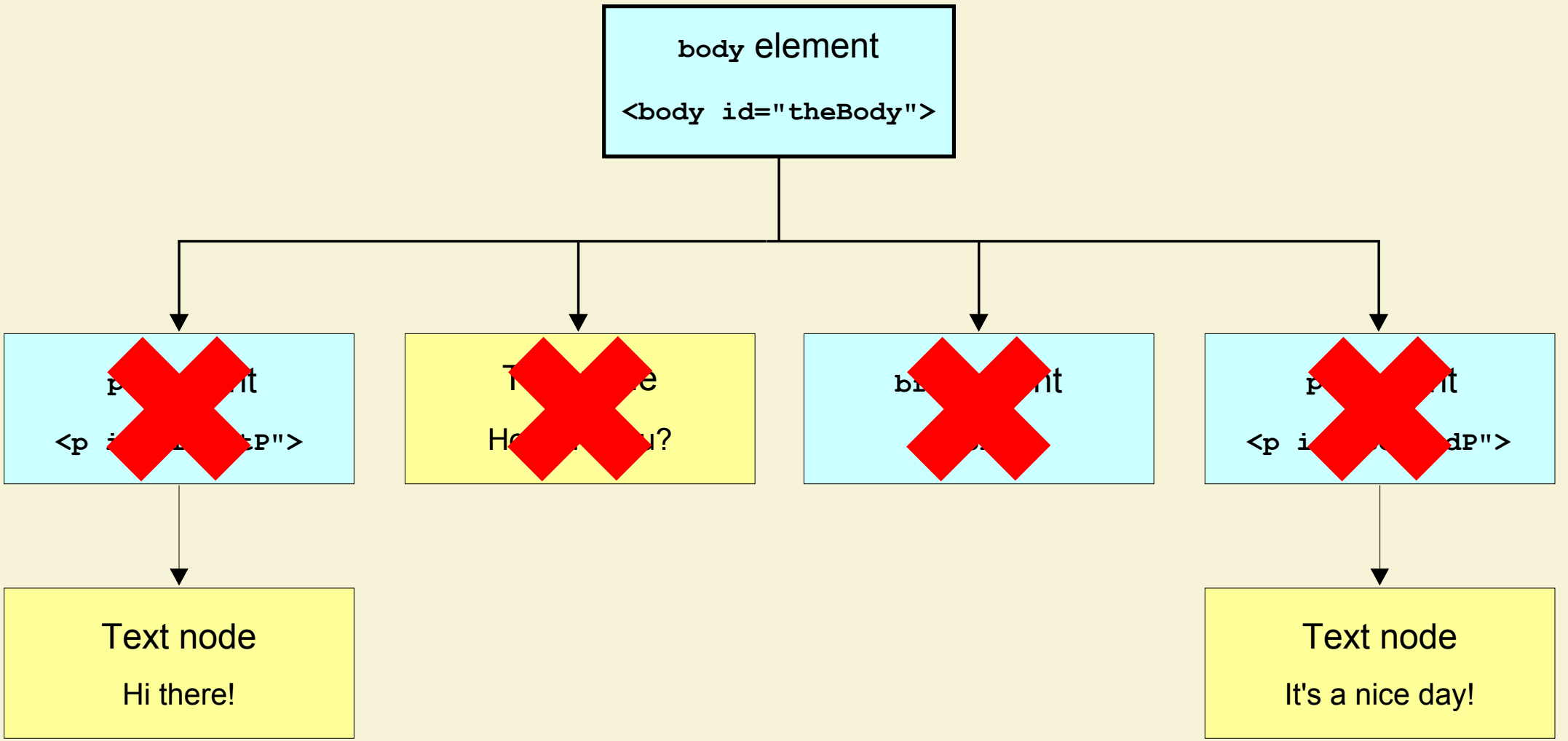
It's a nice day!

Delete children

Click here to open the example


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">
```

DELETING ALL CHILDREN

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

while (theNode.firstChild) // While there is a child
    theNode.removeChild(theNode.firstChild);
```

```
<!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>
```

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

CLONING NODES

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

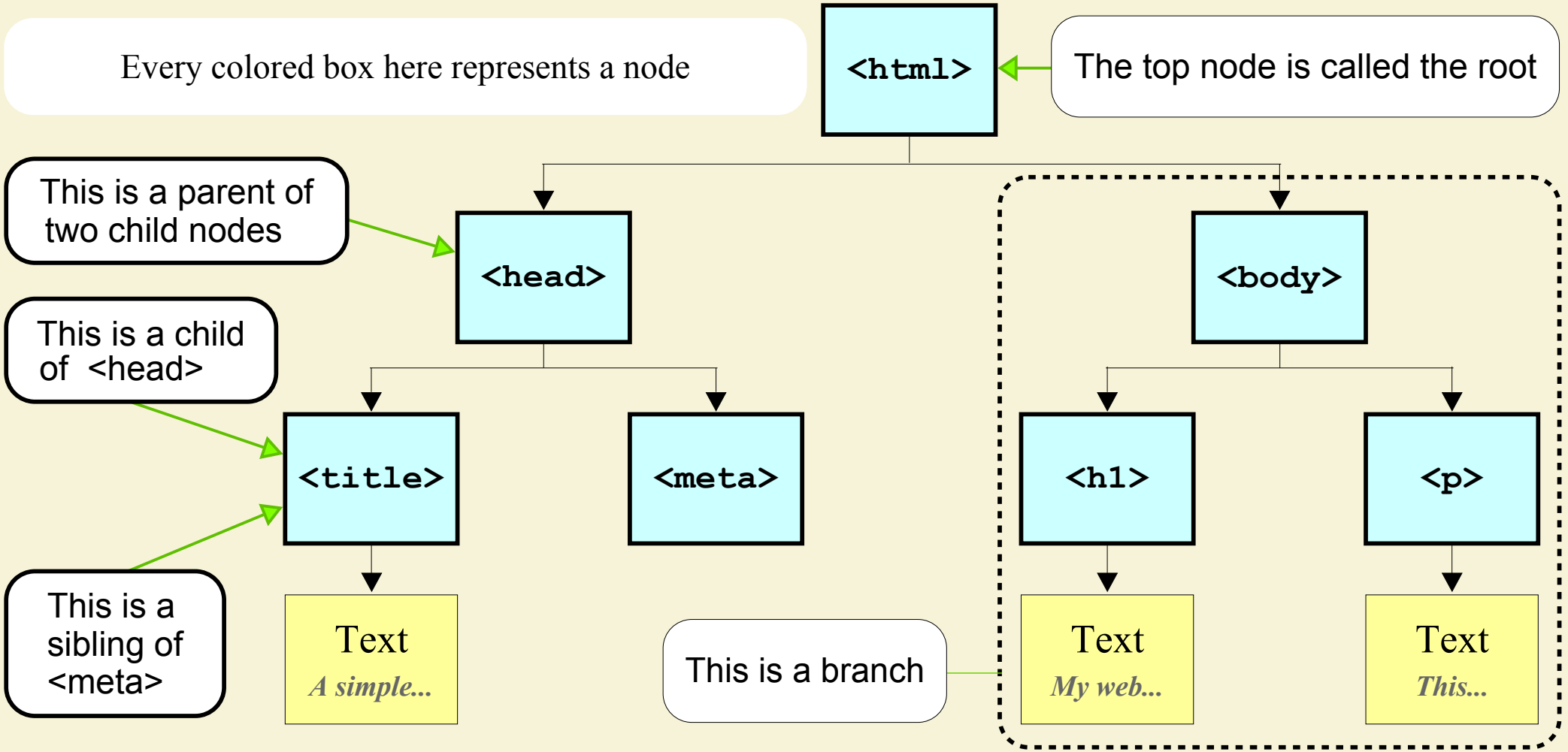
- You'll be able to copy (clone) a node
- You'll be able to copy a branch of nodes

WE WILL LOOK AT

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

THE BASIC IDEA

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


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...

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

- Good morning
- Hello

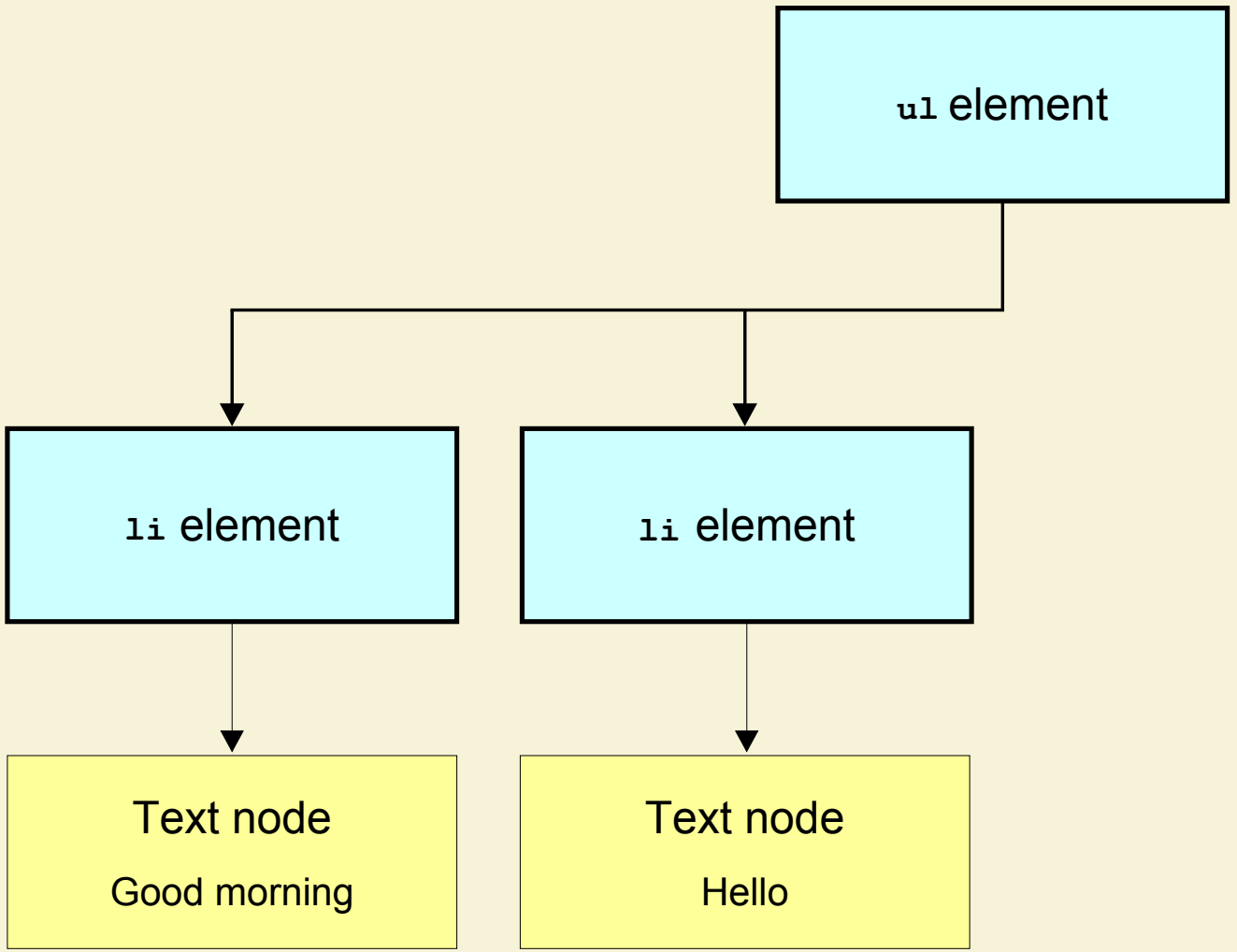
Click on the button to cloneNode()

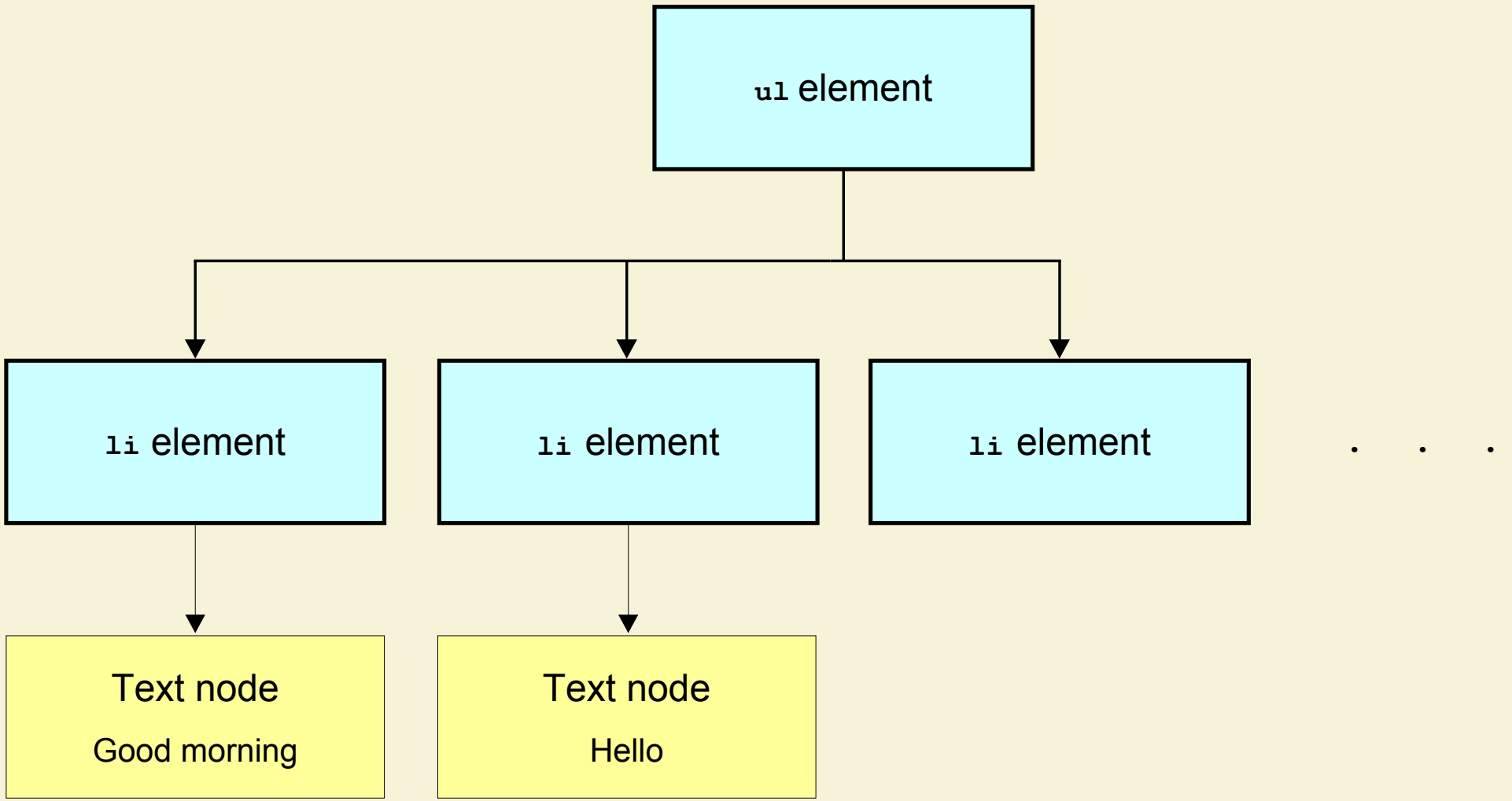
Copy it!

Click here to open the example

```
<!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

- Good morning
- Hello

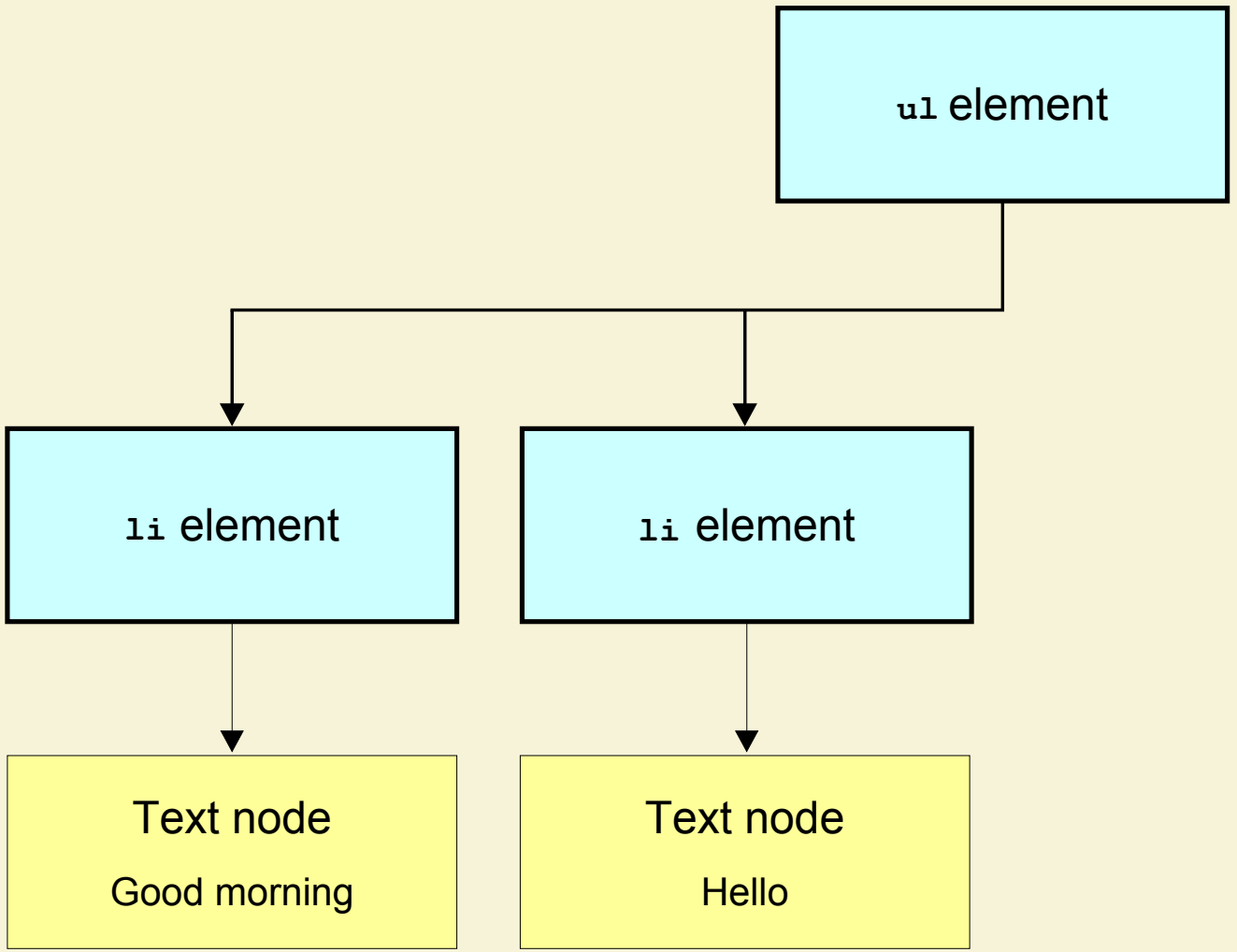
Click on the button to cloneNode(true)

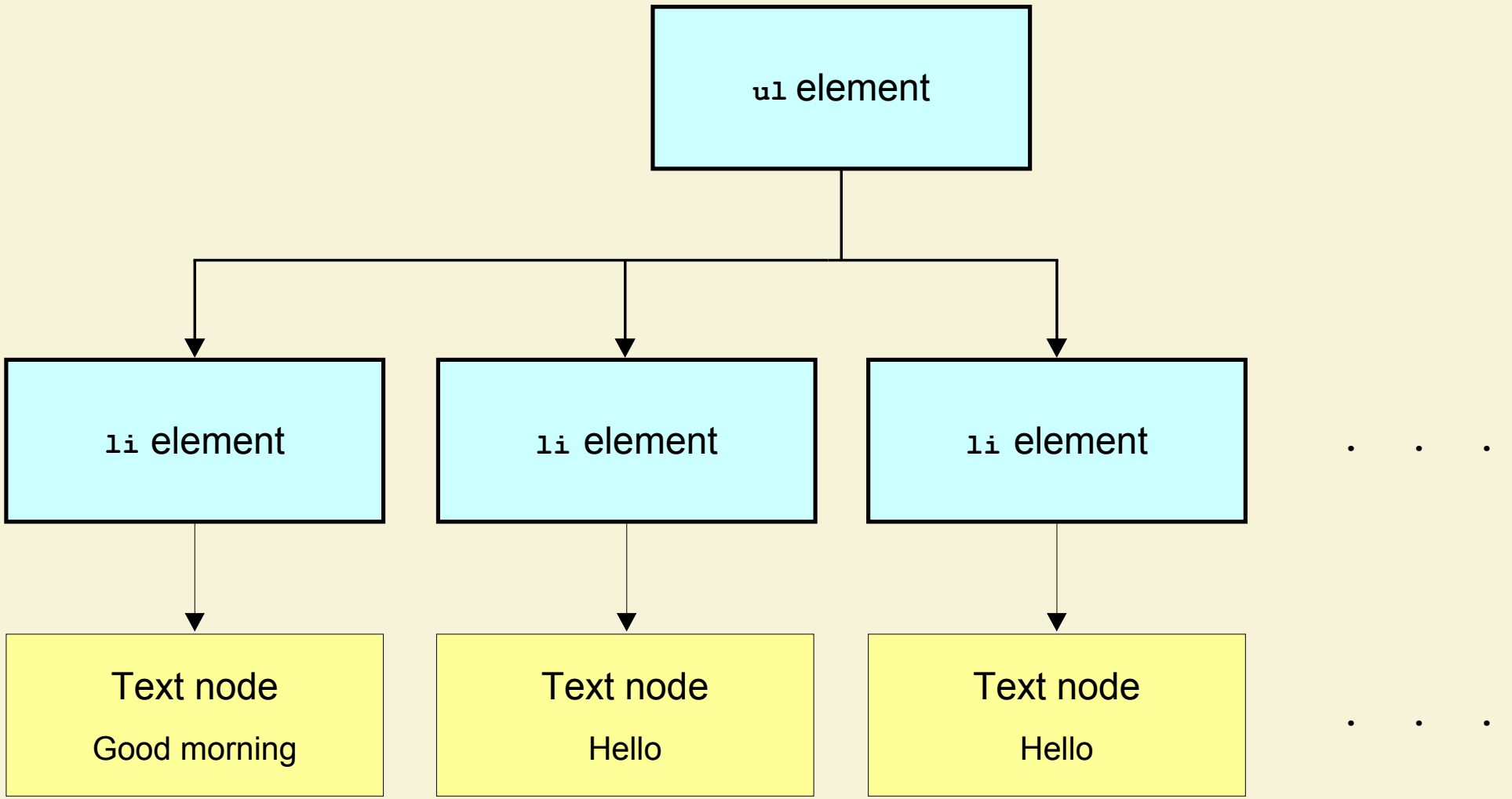
Copy it!

Click here to open the example

```
<!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
. . .

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

MOUSE EVENTS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll know how to use mouse events

WE WILL LOOK AT

- `onclick`
- `onmousedown`
- `onmouseup`
- `onmouseover`
- `onmouseout`

MOUSE EVENTS

- The most commonly used mouse events:
  - `onclick` - when the user clicks on an object
  - `onmousedown` - when the user presses down the mouse button
  - `onmouseup` - when the user lets go of the mouse button
- `onclick` = `onmousedown` followed by `onmouseup`

```
<html><body><script>
  function good_choice() { alert("Good choice!"); }
  function bad_choice() { alert("I don't agree!"); }
  </script>
  <h1>Click on the best social network...</h1>
  <img src="facebook_icon.png"
       onclick="bad_choice()">
  <img src="google_plus_icon.png"
       onclick="bad_choice()">
  <img src="twitter_icon.png"
       onclick="good_choice()">
</body></html>
```

MORE MOUSE EVENTS

- `onmouseover` - mouse is moved over an object
- `onmouseout` - mouse is moved away from an object

```
<html><body><script>
    function change_colour( new_colour ) {
      document.getElementById("myDiv")
              .style.background=new_colour;
    }
  </script>
  <div id="myDiv"
    style="position:absolute; background:yellow;
    left:300; top:100; width:300; font-size:52pt"
    onmouseover="change_colour('red');"
    onmouseout="change_colour('yellow');">
    Move your mouse over this ...
    then move it out...
  </div>
</body></html>
```

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

TIMER EVENTS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll know how to use timer events

WE WILL LOOK AT

`setTimeout`	`setInterval`
`clearTimeout`	`clearInterval`

TIMERS

- Timers are very useful for dynamic web page behaviour
- Set a timer like this:

```
var the_timer;
the_timer=setTimeout(do_something, 1000);
```

- `do_something()` will be executed 1 second later
- The value 1000 is in milliseconds, so 1000=1 second

```
<html><head><script>
  var wait_duration;
  function set_things_up() {
    wait_duration = prompt("How long do you " +
                           "want to sleep?");
    setTimeout(show_wake_up_message, wait_duration );
  }
  function show_wake_up_message() {
    alert("WAKE UP! WAKE UP! WAKE UP!!");
  }
</script></head>
  <body onload="set_things_up()">
    <h1>Alarm clock example</h1>
  </body>
</html>
```

TIMER EXAMPLE - MOVING AN IMAGE

```
<html><head><script>
  var the_timer, x_position = 0, the_image;
  function set_timer() {
    the_image=document.getElementById("stones_image");
    x_position=x_position+1;
    the_image.style.left=x_position;
    the_timer = setTimeout(set_timer, 50);
  }
  </script></head>
  <body onload="set_timer()">
    <img src="stones.png" id="stones_image"
      style="position:absolute; left:0">
</body></html>
```

STOPPING A TIMER

- If a timer is started like this:

```
var the_timer;
the_timer=setTimeout(do_something, 1000);
```

- Then stop it like this:

```
clearTimeout(the_timer);
```

```
<html><head><script>
  var the_timer, x_position = 0, the_image;
  function set_timer() {
    the_image = document.getElementById("stones_img");
    x_position = x_position + 1;
    the_image.style.left = x_position;
    the_timer = setTimeout(set_timer, 50); }
</script></head>
<body onload="set_timer()">
  <img src="stones.png" id="stones_img"
       style="position:absolute; left:0">
  <button onclick="clearTimeout(the_timer)">
      Stop!</button>
</body></html>
```

SETINTERVAL

- `setInterval()` repeatedly does something
- Start it like this:

```
var the_timer;
the_timer=setInterval(do_something, 2000);
```

- `do_something()` will be executed every 2 seconds
- To stop it:

```
clearInterval(the_timer);
```

```
<html><head><script>
  var the_timer, x_position = 0, the_image;
  function do_timer(){
    the_image = document.getElementById("stones_img");
    x_position = x_position + 1;
    the_image.style.left = x_position;
  }
</script></head>
  <body onload="the_timer=setInterval(do_timer, 50)">
    <img src="stones.png" id="stones_img"
      style="position:absolute; left:0">
    <button onclick="clearInterval(the_timer)">
        Stop!</button>
</body></html>
```

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

ADDING EVENTS USING JAVASCRIPT

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll able to set up events using JavaScript

WE'LL LOOK AT

`addEventListener()`

`removeEventListener()`

ADDING A HANDLER USING HTML

- Adding an event to an element in HTML:

```
<html><head><script>
  function do_something() {alert("Page has loaded");}
</script></head>
<body onload="do_something()"></body>
</html>
```

- `do_something()` is the *event handler* for this event

ADDING A HANDLER USING JAVASCRIPT

- We can also add an event to an element:

```
<html><body id="theBody">
  <script>
    function do_something() { alert("Page has loaded") }
    window.onload = do_something;
  </script>
</body></html>
```

ADDING A HANDLER USING JAVASCRIPT

- Another way:

```
<html><body>
  <script>
    function do_something() { alert("Page has loaded") }
    window.addEventListener("load", do_something);
  </script>
</body></html>
```

IF YOU HAVE MORE THAN ONE EVENT HANDLER

- Event handlers are stored in an array
- When an event happens, all the handlers are executed
- They are executed in the order they are added

REMOVING AN EVENT HANDLER

- To remove an event handler:

```
var theBody = document.getElementById("theBody");
theBody.removeEventListener("load", do_something);
```

```
<html><body>
  <button id="btn0" onclick=" alert('Hello!') ">
    Click Me!</button><br>
  <button id="btn1">Remove Listener</button>
  <script>
    function do_something() { alert('Clicked'); }

    var btn0 = document.getElementById("btn0");
    btn0.addEventListener("click", do_something);

    var btn1 = document.getElementById("btn1");
    btn1.addEventListener("click", function() {
      btn0.removeEventListener("click", do_something);
    });
</script></body></html>
```

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

MORE ON FUNCTIONS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll know how to assign a function to a variable
- You'll be able to pass a function to a function
- You'll be able to return a function from a function

WE WILL LOOK AT

Events	`onload`
Functions	`function`
	`return`

TWO WAYS TO DECLARE A FUNCTION

```
function functionOne() {
```

. . . code here . . .

```
}
```

This function is defined when the web page is loaded.

```
var functionTwo = function() {
```

. . . code here . . .

```
}
```

```
var functionTwo = function thisFunc() {
```

. . . code here . . .

```
}
```

Here we give a function to a variable

The function is defined when the browser reaches that point in the code

PASSING A FUNCTION TO A FUNCTION

You can pass a function to a function

```
<!doctype html>
<html>
    <head>
        <script>

function check(a, b){
        if (b!=0) return true;
        else return false;
}

function myDivide( fn, num, div ) {
        if (fn(num, div)) {
                alert("It's OK!");
                return num/div;
        } else {
                alert("Not OK!");
        }
}

result=myDivide(check, 44, 1);

        </script>
    </head>
</html>
```

RETURNING A FUNCTION FROM A FUNCTION

You can return a function from a function

```
<!doctype html>
<html>
    <head>
        <script>

            function counter() {
                var count = 0;

                return function() {
                    count++;
                    alert(count);
                }

            }

            var count = counter();
            count();
            count();
            count();
        </script>
    </head>
</html>
```

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

AN EXAMPLE DOM PROJECT

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

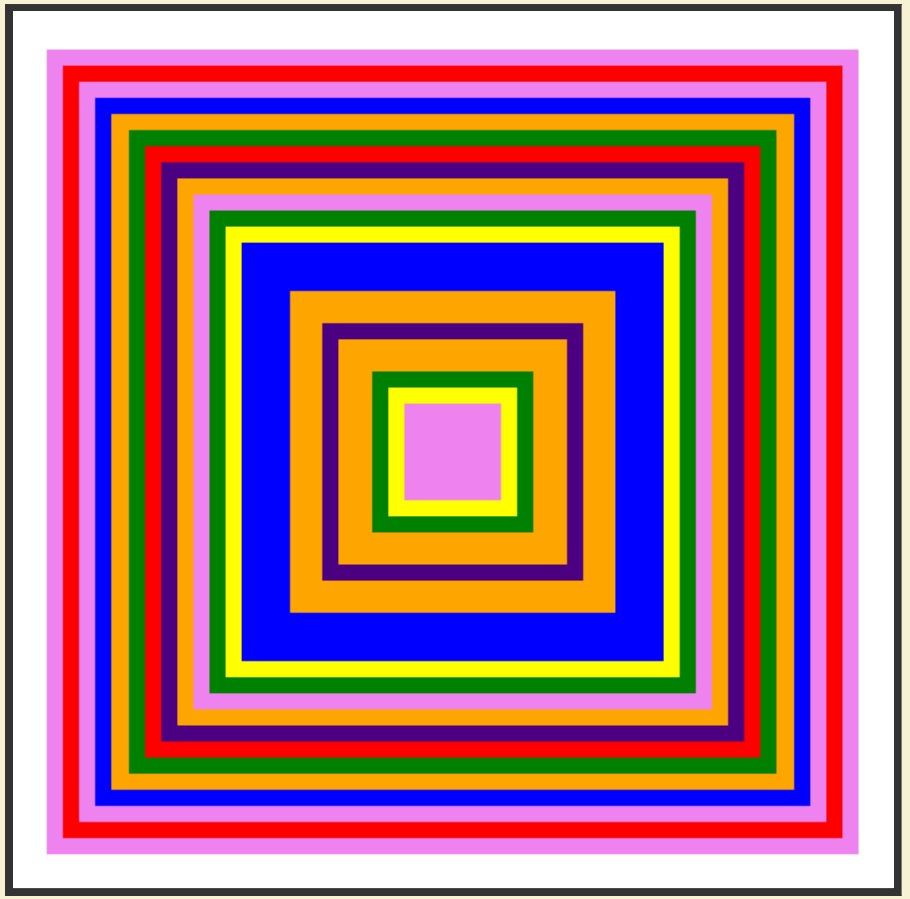
- You'll have a stronger appreciation of DOM handling
- You'll have a stronger appreciation of JavaScript

THIS PROJECT USES

`function`	`getElementById()`	`Math.random()`
`onload`	`createElement()`	`Math.floor()`
`while`	`appendChild()`	

STRENGTHENING OUR UNDERSTANDING

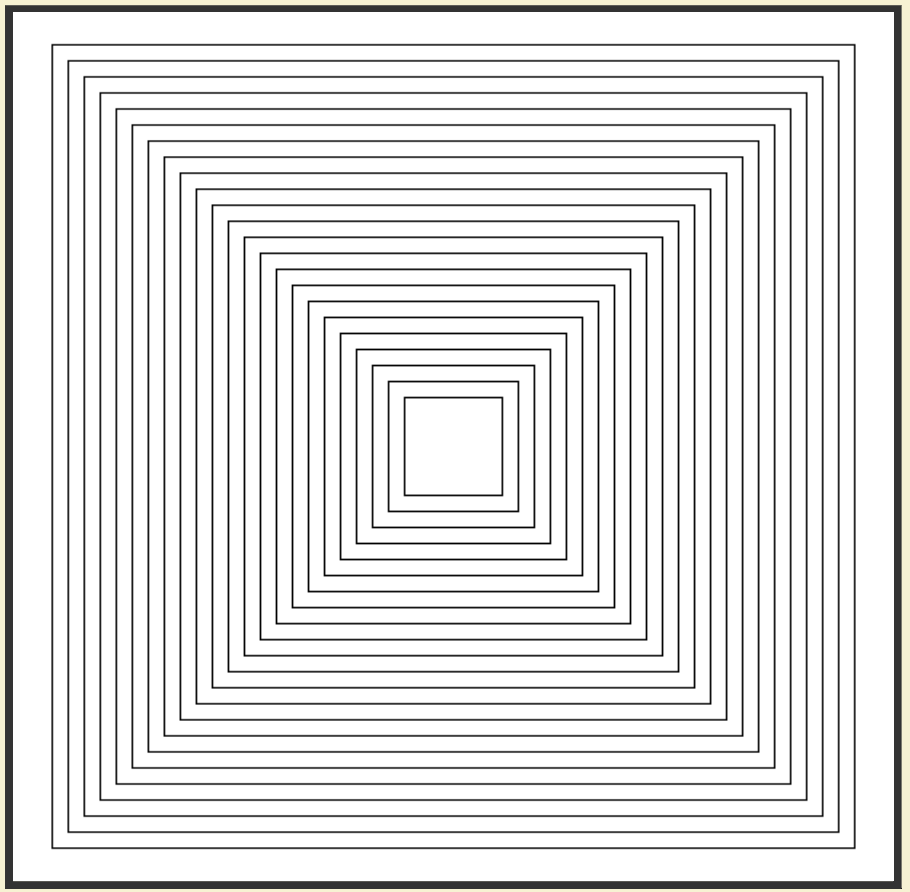
- Let's use the techniques we have learned so far
- We will generate a colourful pattern
- The pattern repeatedly re-generates itself



Click here to open the example

HOW IT WORKS

- A series of squares is generated by JavaScript
- Each square is a `div`
- Each square has a different `top`, `left`, `width` and `height`



Click here to open the example

HTML PART

```
<!doctype html>
<html><head>
   <title>An Example Project</title>
   <meta http-equiv="refresh" content="1">
   <style>
     div {position:absolute}
   </style>
   </head>
   <body id="theBody" onload="show_pattern()">
      <script src="08_dom_colorful_pattern_js.js">
      </script>
   </body>
</html>
```

- The main code is triggered when the web page is loaded:

```
<body id="theBody" onload="show_pattern()">
```

- The actual code is stored in another file:

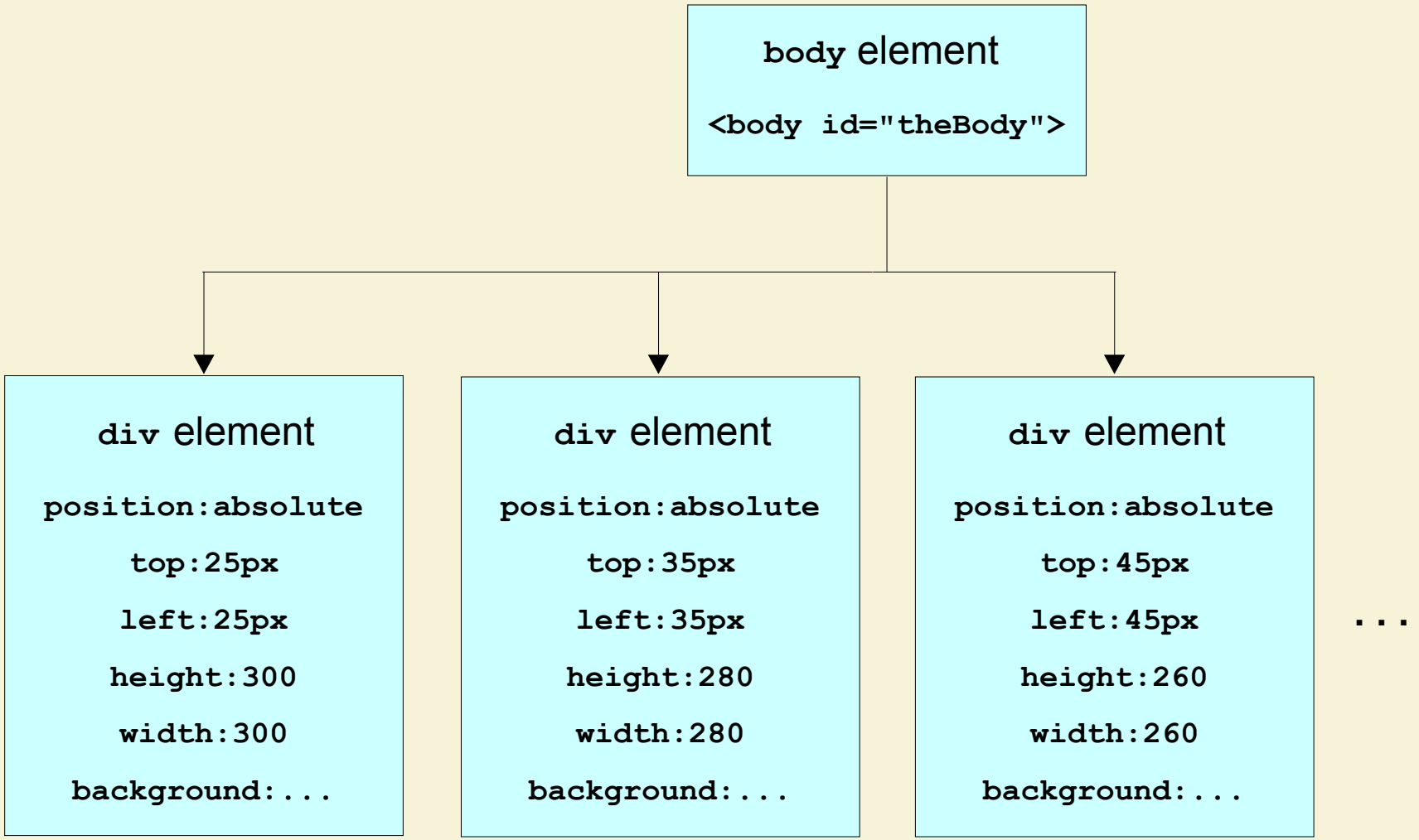
```
<script src="08_dom_colorful_pattern_js.js">
</script>
```

- This tells the browser to reload the page every second:

```
<meta http-equiv="refresh" content="1">
```

JAVASCRIPT OVERVIEW

- Set up the variables
- Inside the loop:
  1. Generate the `div` node
  2. Set the `div` node attributes
  3. Add the `div` node to the `body`
  4. Adjust variables ready for the next iteration


body element
<body id="theBody">
div element
position:absolute
top:25px
left:25px
height:300
width:300
background:...
div element
position:absolute
top:35px
left:35px
height:280
width:280
background:...
div element
position:absolute
top:45px
left:45px
height:260
width:260
background:...
...

SET UP THE VARIABLES

```
var top_position = 25, left_position = 25;
var width = 300, height = 300;
var color_list = ["red","orange","yellow",
        "green","blue","indigo","violet"];
```

WHILE LOOP STRUCTURE

```
while (width > 50) {

    // all the following code goes here

}
```

1. GENERATE THE DIV NODE

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

2. SET THE DIV NODE ATTRIBUTES

```
var random_color = Math.random() * 7;
random_color = Math.floor(random_color);
```

```
this_div.style.top = top_position + "px";
this_div.style.left = left_position + "px";
this_div.style.width = width + "px";
this_div.style.height = height + "px";
this_div.style.background =
                color_list[random_color];
```

3. ADD THE DIV NODE TO THE BODY

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

the_body.appendChild(this_div);
```

4. ADJUST VARIABLES

```
top_position += 10;
left_position += 10;
width -= 20;
height -= 20;
```

```
function show_pattern(){
    var top_position = 25, left_position = 25;
    var width = 500, height = 500;
    var color_list = ["red","orange","yellow","green","blue","indigo","violet"];
    var the_body = document.getElementById("theBody");

    while(width > 50) {
        var this_div = document.createElement("div");
        var random_color = Math.random() * 7;
        random_color = Math.floor(random_color);
        this_div.style.top = top_position + "px";
        this_div.style.left = left_position + "px";
        this_div.style.width = width + "px";
        this_div.style.height = height + "px";
        this_div.style.background = color_list[random_color];
        the_body.appendChild(this_div);
        top_position += 10; left_position += 10;
        width -= 20; height -= 20;
    }
}
```