COMP4021
Internet Computing

# Handling the DOM

David Rossiter & Gibson Lam

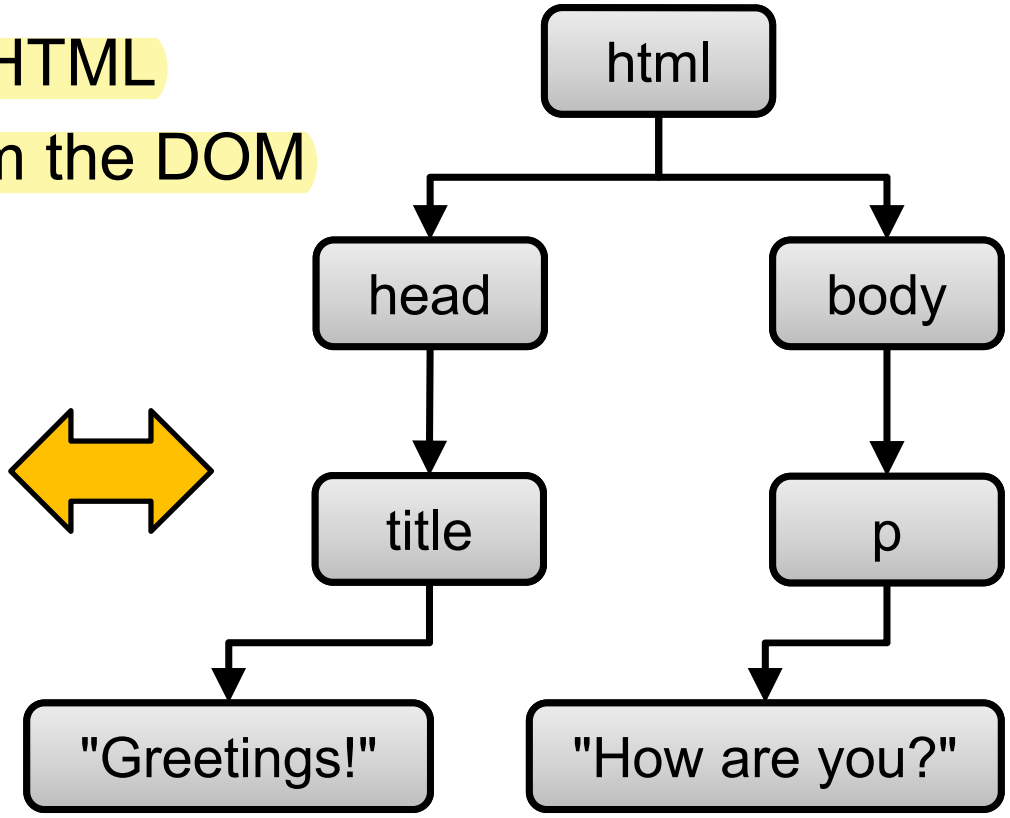
# The Document Object Model

- When you load something into a browser it is stored in the browser memory using a tree structure
- That structure is the DOM (Document Object Model)
- This happens with HTML and also other languages such as XML and SVG, not covered here
- You can use JavaScript to add, delete or change anything in the DOM structure at any time

# An Example DOM Structure

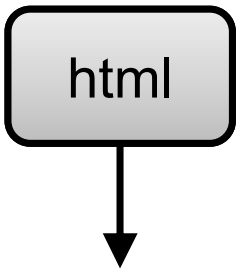
- The DOM is created from the HTML
- The HTML can be created from the DOM

```
<!DOCTYPE html>
<html>
<head>
    <title>Greetings!</title>
</head>
<body>
    <p>How are you?</p>
</body>
</html>
```



for each node in the tree structure, you can ask its type

# The Root Element



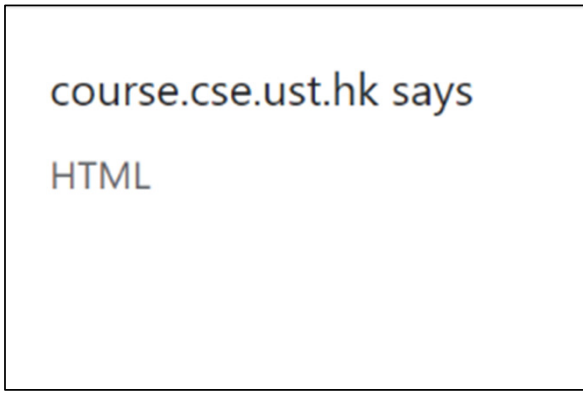

- The *root element* means the top of the tree
- The root element of an HTML document is the `<html>` element
- You can use this code to refer to it:

  `document.documentElement`

- For example,

  `alert(document.documentElement.nodeName);`

  shows this



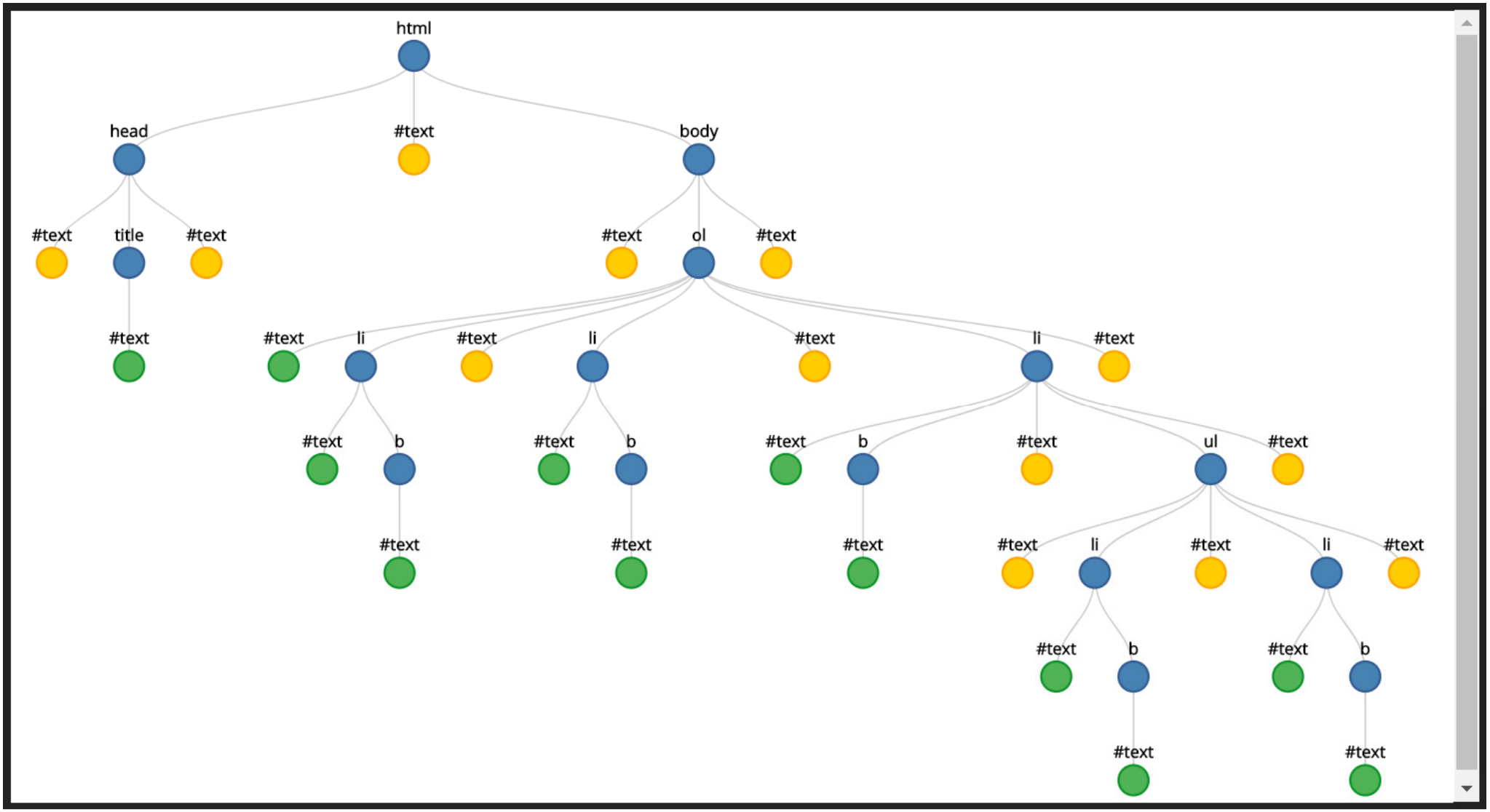
# A DOM Visualizer

- Sometimes it is useful to see the DOM structure
- You can use a DOM visualizer here:

http://bioub.github.io/dom-visualizer/

- Note that this visualizer is best for small files
- If you try to visualize a large file you get a messy result

# An Example DOM Visualization



# DOM Nodes

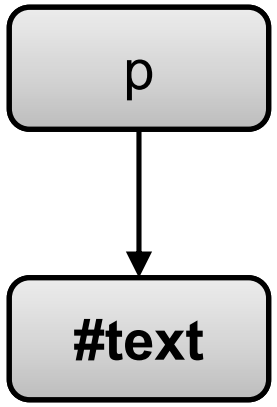
- Every ‘box’ in the DOM tree is called a DOM node
- We will look at two types of DOM nodes:
  - Element nodes e.g. <p>
    which store the information of HTML elements
  - Text nodes
    which store text which is usually inside the first type
- There are some other types of node but these are the most common

# An Example

- You might think this simple HTML is stored using one node:

```
<p>Hello</p>
```

- However, 2 nodes are used!
- One node is for the HTML tag
- The other node is for the text inside the tag

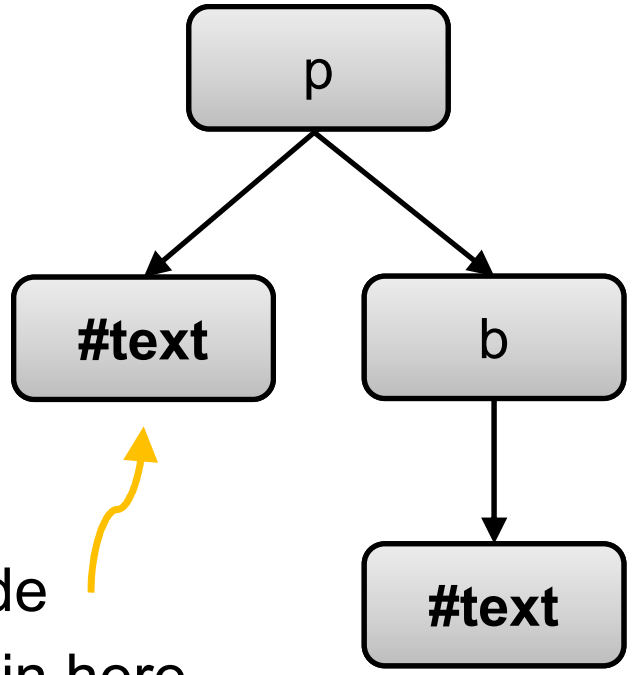



- This is an element node `<p>…</p>`
- This is a text node
- `Hello` is stored in here

# Another Example

```
<p>Hello <b>Dave</b></p>
```

- 4 nodes are used to store this HTML



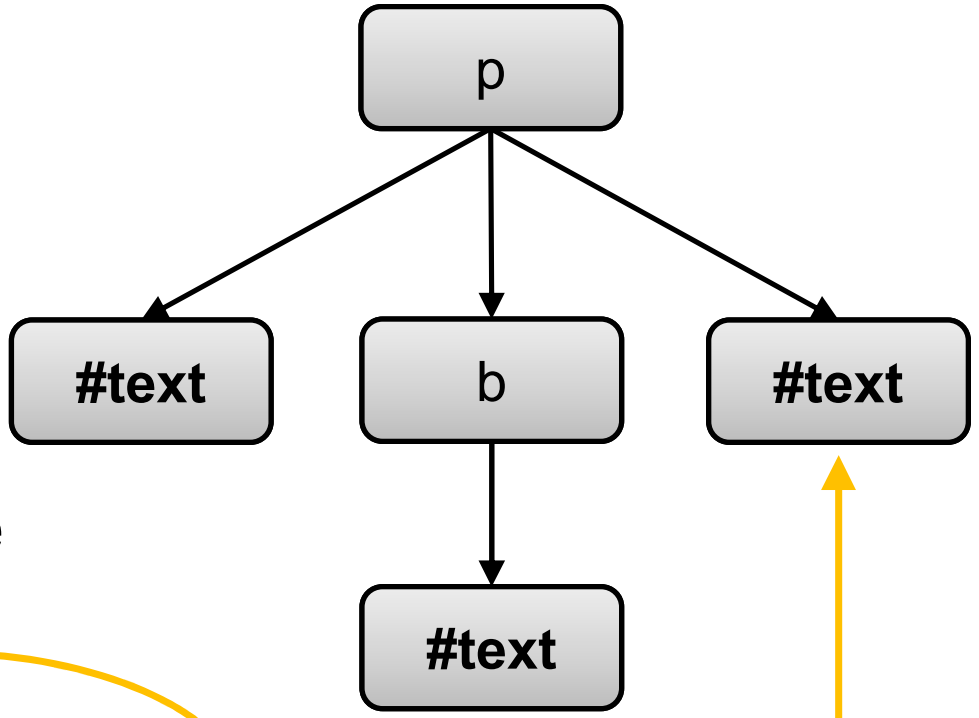

- This is a text node
- `Hello` is stored in here

- This is a text node
- `Dave` is stored in here

# Whitespace

- Whitespace means ‘things you can’t see’ in the web page
- Whitespace is also stored in the DOM like everything else

```
<p>Hello <b>Dave</b>
</p>
```




- This is a text node
- The ‘go to next line’ character is stored here
- We often don’t show whitespace nodes because there are too many

# Finding an Element in the DOM

- First, you find the thing in the DOM you want to change
- There are several ways to do that
- One way is to look for something is by using its `id`

```
<p id="main_text">Let's do a BBQ!</p>
```

*The id of this paragraph*

# Example of Changing Paragraph Text

- We can change the text content of the paragraph in the previous slide like this:

`innerHTML` *can be used to change the content*

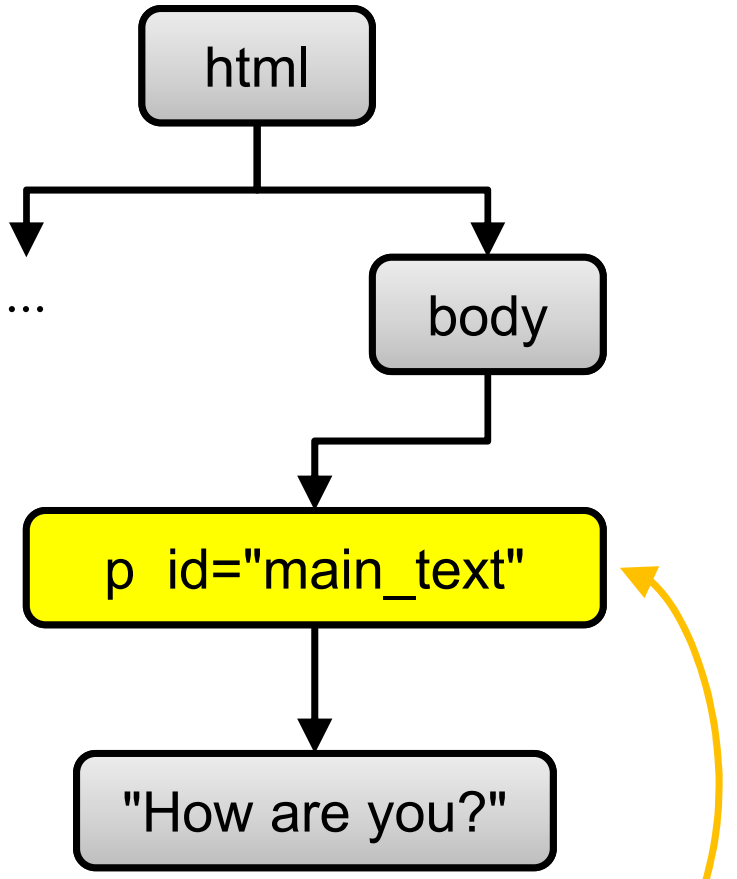
```
document.getElementById("main_text").innerHTML =
                                    "I want to fly a kite!";
```

- The above line of code is like typing this:

```
<p id="main_text">I want to fly a kite!</p>
```

# Using getElementById

```
...
<body>
    <p id="main_text">
    Let's do a BBQ!
    </p>
    ...
</body>
</html>
```




- After this code:

```
x=document.getElementById("main_text")
```

`x` points to the element called `main_text` (but doesn't change it)

# Changing the Style Attribute

- One useful thing you can do for HTML elements is to change their `style` attribute
- For example, this is one way to change the background colour of an element:

```
x.style.backgroundColor = "red";
```

- `x` is from the last slide

# Changing Style

- If the CSS name has a hyphen ( `-` ) then you need to remove the hyphen and capitalize the following letter
- For example:
  - `background-color` becomes `backgroundColor`
  - `font-family` becomes `fontFamily`
  - and so on


```
...
<p style="font-family: Helvetica;
          font-size: 60px">
   Big Big Text</p>
...
```

```
<p id="fun">
Big Big Text</p>
...
x=document.getElementById("fun");
x.style.fontFamily="Helvetica";
x.style.fontSize="60px";
...
```

- Note the changes

use JS to make the same effect

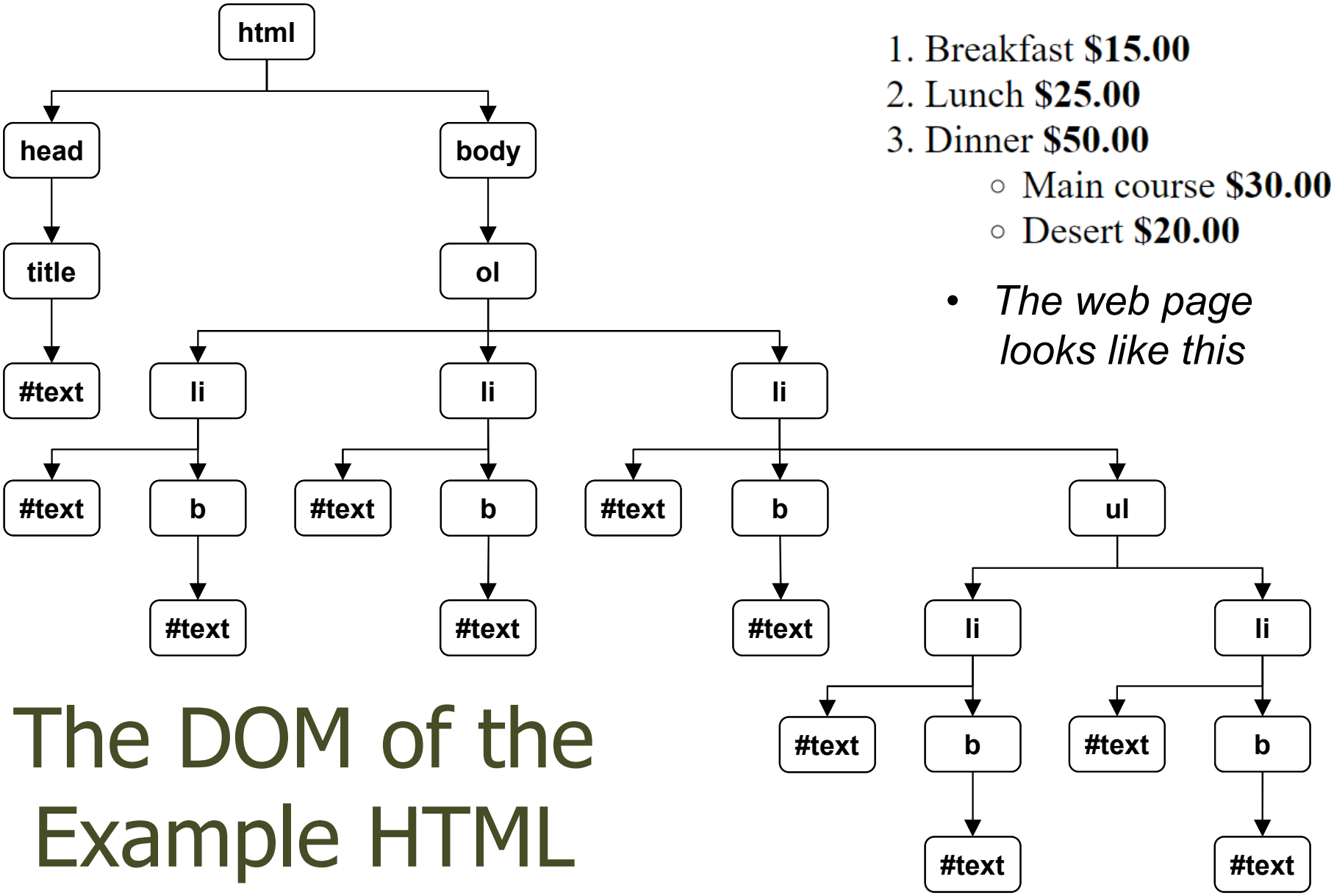
# Example HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>The Example
           Document</title>
</head>
<body>
    <ol>
        <li>Breakfast <b>$15.00</b></li>
        <li>Lunch <b>$25.00</b></li>
        <li>Dinner <b>$50.00</b>
            <ul>
                <li>Main course <b>$30.00</b></li>
                <li>Desert <b>$20.00</b></li>
            </ul>
        </li>
    </ol>
</body>
</html>
```

1. Breakfast **$15.00**
2. Lunch **$25.00**
3. Dinner **$50.00**
    - Main course **$30.00**
    - Desert **$20.00**

- *The web page looks like this*

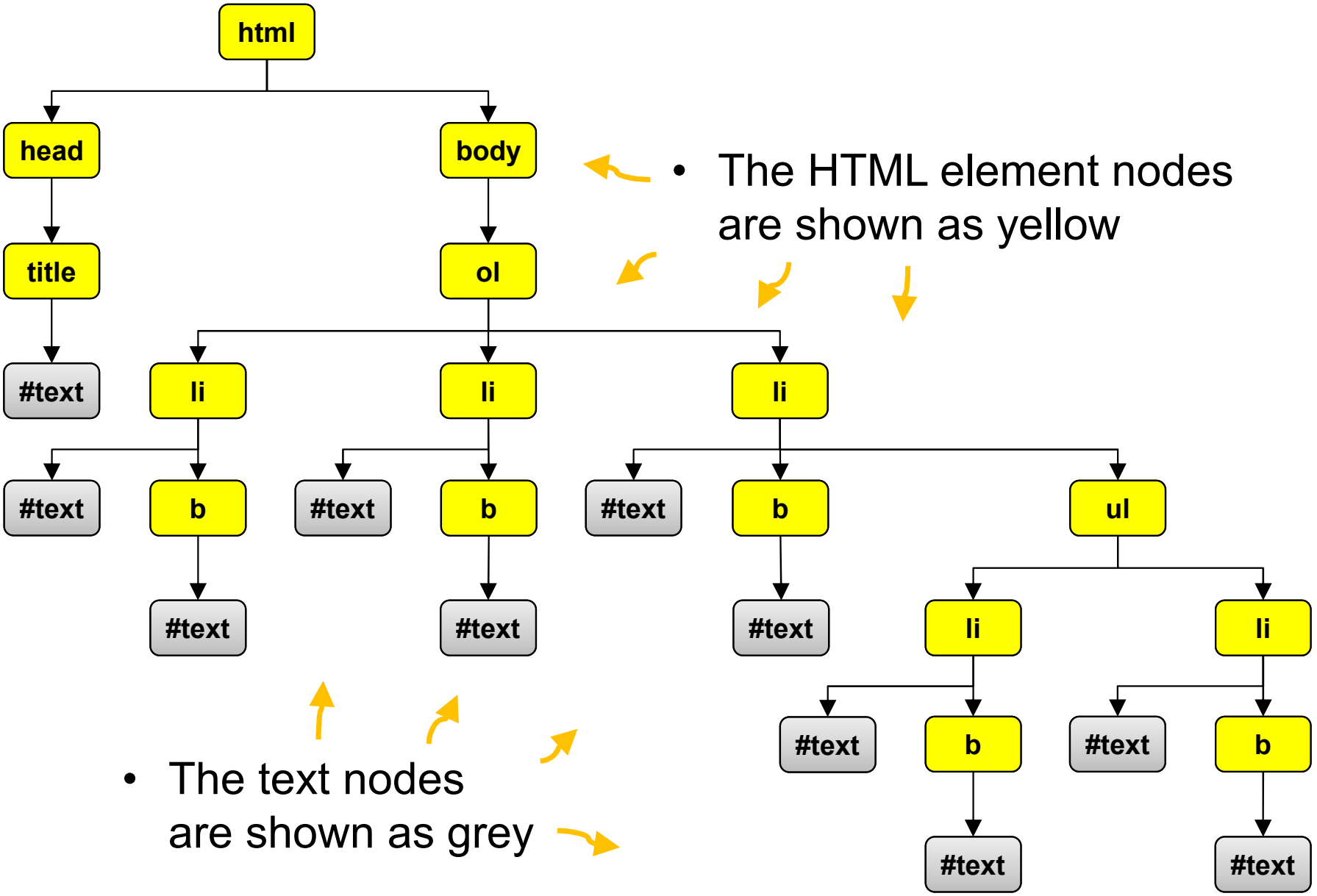
- We will use this as an example in the next few slides




1. Breakfast **$15.00**
2. Lunch **$25.00**
3. Dinner **$50.00**
    - Main course **$30.00**
    - Desert **$20.00**

- *The web page looks like this*

# The DOM of the Example HTML

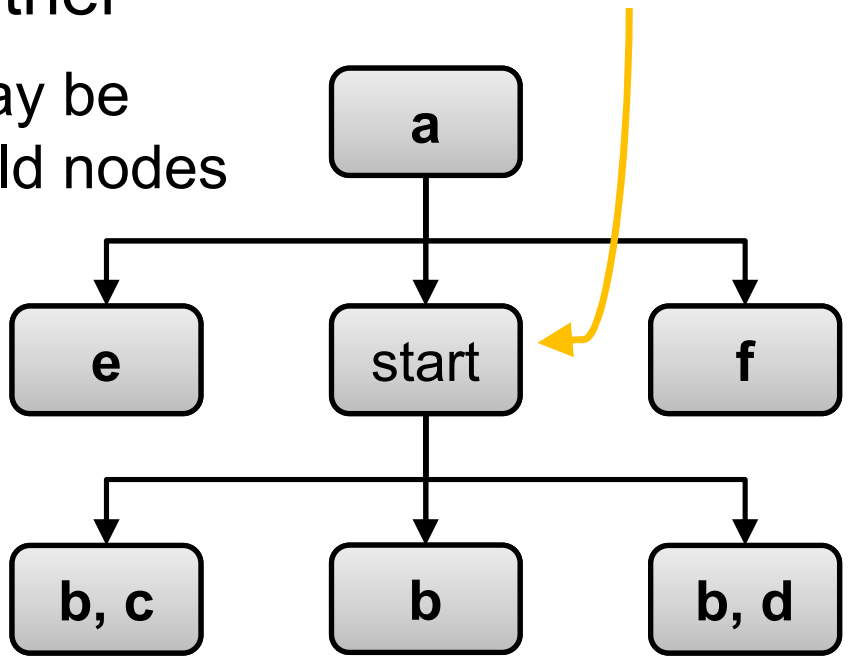

html
head
body
title
ol
#text
li
li
li
#text
b
#text
b
#text
b
ul
#text
#text
#text
li
li
#text
b
#text
b
#text
#text
• The HTML element nodes are shown as yellow
• The text nodes are shown as grey

# Traversing the DOM

- You can move around (=traverse) the DOM tree from one node to another
- These are useful:
  a) `start.parentNode`
  b) `start.childNodes[]`
  c) `start.firstChild`
  d) `start.lastChild`
  e) `start.previousSibling`
  f) `start.nextSibling`

- There may be many child nodes

- A node can have any id, `start` is just an example id



# Finding HTML Tags

- As you know, you can use `document.getElementById()` to find a particular element in the DOM
- Alternatively, you can use `document.getElementsByTagName()` to find all HTML elements (there might be zero, 1, or >1) which have the same tag e.g. `<h2>` `<p>` etc
- For `getElementsByTagName` the results will be in a list - see the next slide

# An Example Using Tag Name

- The second child of every second list item is changed

- Some JavaScript:

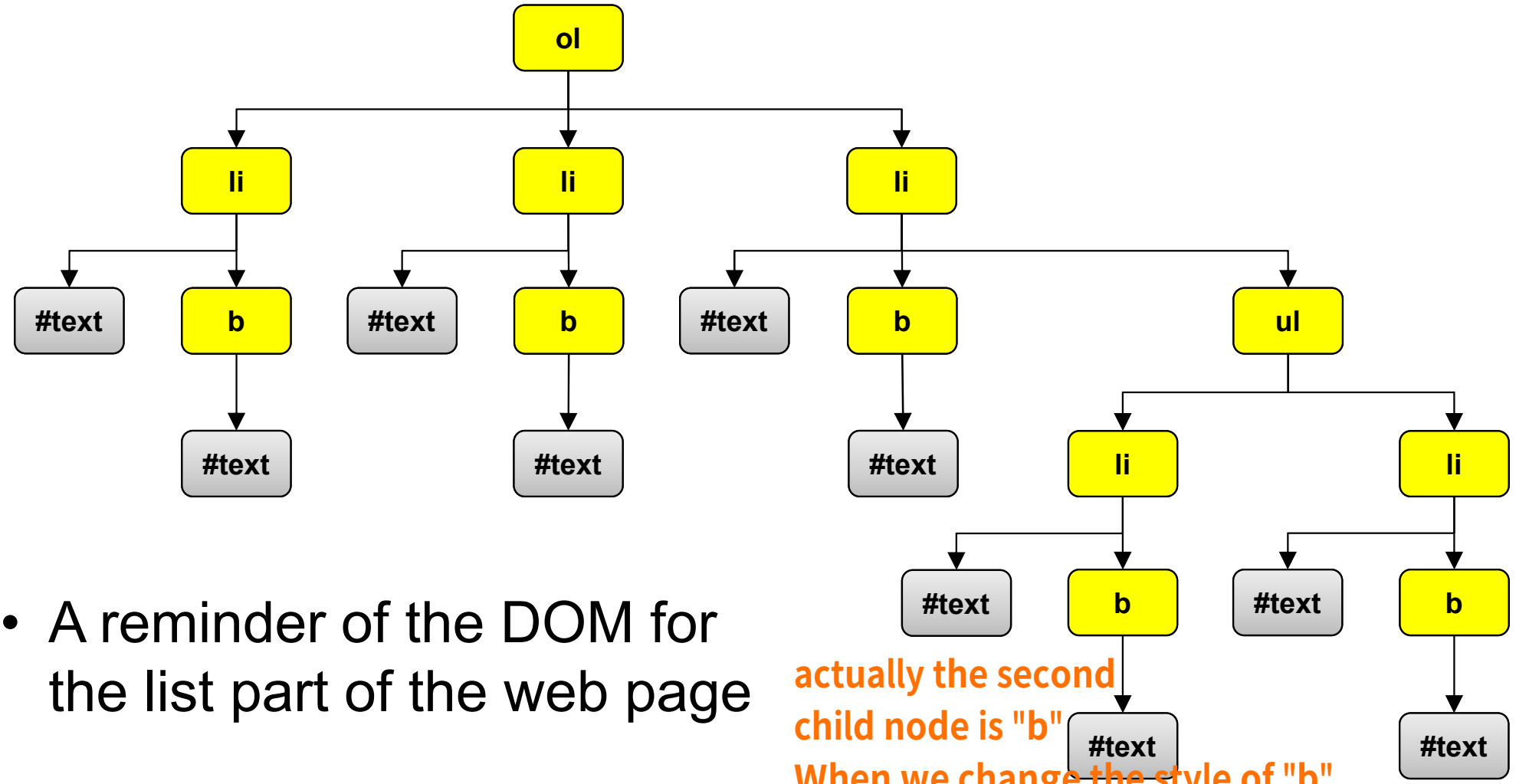
```
var all_li = document.getElementsByTagName("li");
for (var i = 0; i < all_li.length; i += 2) {
    all_li[i].childNodes[1].style.color = "red";
}
```

the second child

- After this code has finished, three out of the five <b> elements have been changed to red:

1. Breakfast **$15.00** ⇦
2. Lunch **$25.00**
3. Dinner **$50.00** ⇦
    - Main course **$30.00**
    - Desert **$20.00** ⇦

The second list is also included




- A reminder of the DOM for the list part of the web page

actually the second child node is "b"

When we change the style of "b", everything underneath it will inherit the style

# Adding a New Element

- You can use JavaScript to add new elements to the DOM
- It involves two steps:
  1. Create the DOM node you want to add
     - After this stage the node is not actually in the DOM tree
  2. Insert the newly created node into an appropriate place in the DOM

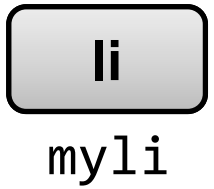
# Creating a New DOM Element

- Here's an example of creating a new DOM element:

*This could be any variable name*

```
var myli = document.createElement("li");
```

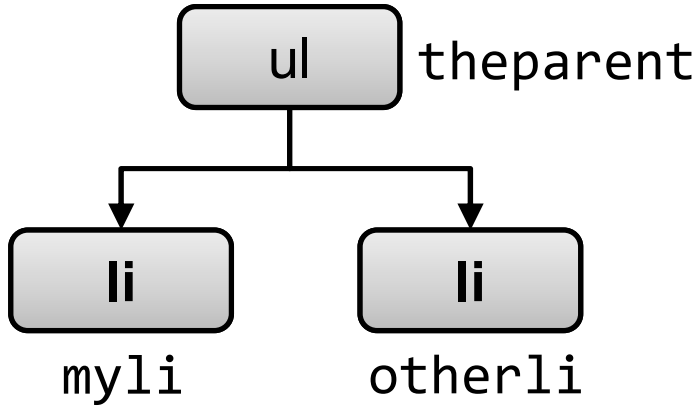
- After this line of code, the element is in 'floating' in memory by itself, and isn't attached to the DOM



# Putting an Element into the DOM

- Once you have a new element, you can insert it into the DOM in the place you want
  - If you want to you can insert the new element before another element i.e.

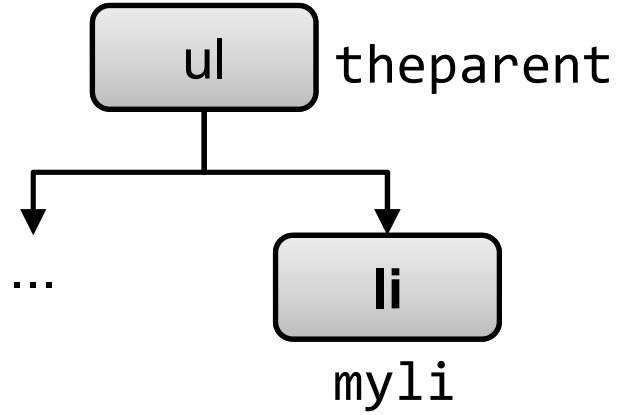
```
theparent.insertBefore(myli, otherli);
```




This is the `myli` created in the last slide

  - Or you can insert the new element at the end of all children under a parent, i.e.

```
theparent.appendChild(myli);
```



# Easy Ways to Add a Text Node

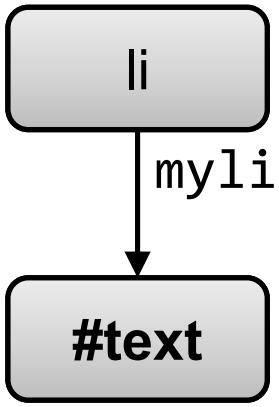
- Although you can use DOM functions to properly add text nodes ( `createTextNode` ), you can do it more easily like this:

This is the `myli` created two slides ago

```
myli.textContent = "Snack";
```

- Alternatively, you can use:

```
myli.innerHTML = "Snack";
```



# Example of Adding a New Item

- In this example, we put together the code we have shown in the previous few slides to add a new item to the example HTML

```
var myli = document.createElement("li");
```
*Create a list item*

```
myli.textContent = "Snack";
```

```
document.
    getElementsByTagName("ol")[0]
        .appendChild(myli);
```

*The first ordered list in the web page*

*This is one line of code*

*The newly added item*

1. Breakfast **$15.00**
2. Lunch **$25.00**
3. Dinner **$50.00**
    - Main course **$30.00**
    - Desert **$20.00**
4. Snack

# Removing Elements

- You can delete anything in the DOM
- To do that, you cannot ask the node to remove itself
- Instead, you have to ask its parent to remove it
- For example:

*The parent* *The child*

```
myul.removeChild(myli);
```

or

*Go up to the parent and ask it to remove this child*

```
myli.parentNode.removeChild(myli);
```