



华中师范大学伍伦贡联合研究院
Central China Normal University Wollongong Joint Institute



UNIVERSITY
OF WOLLONGONG
AUSTRALIA

CSIT884

Web Development

Lecture 09B – DOM

Objectives

- Understand Document Object Model of a website
- Use JavaScript to generate dynamic content on a website



DOM

Document Object Model (DOM):

- represents the structure of a webpage as a tree in memory
- allows access to the tree via programming language such as JavaScript

We can use JavaScript to change structure, style and content of a website via its DOM.



DOM tree

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>My website</title>
```

```
  </head>
```

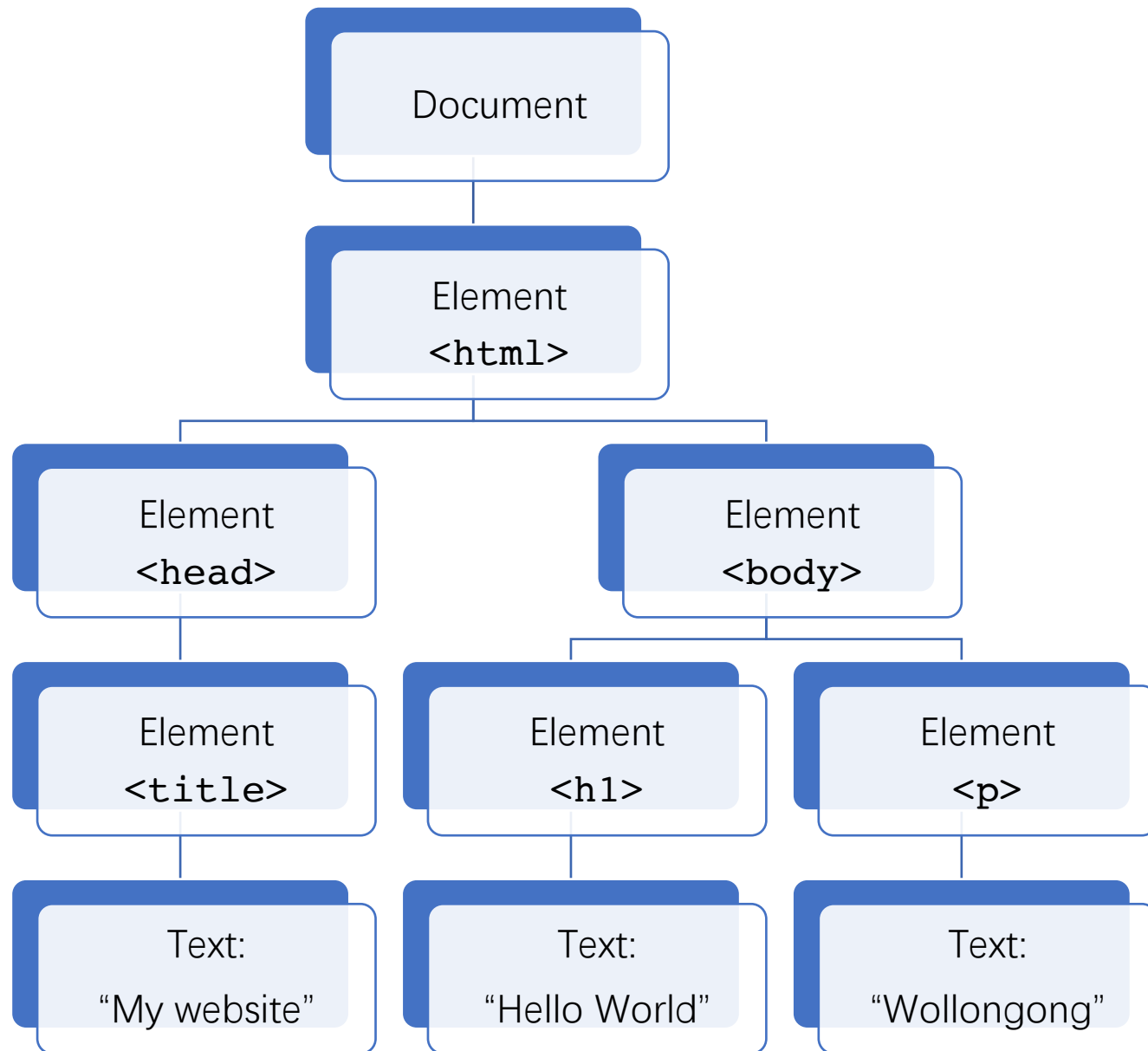
```
  <body>
```

```
    <h1>Hello World</h1>
```

```
    <p>Wollongong</p>
```

```
  </body>
```

```
</html>
```



Manipulate DOM tree

We can use DOM to create dynamic web page:

- add new HTML elements and attributes
- change HTML elements
- change HTML attributes
- change CSS styles
- remove existing HTML elements and attributes
- create new HTML events



document object

The document object represents a website.

- Select an element by id:

```
document.getElementById(id)
```

- Select elements by tag name:

```
document.getElementsByTagName(tagName)
```

- Select elements by class name:

```
document.getElementsByClassName(className)
```

Create an element

```
var element = document.createElement( tagName );
```

- `tagName` is a string specifying the type of element to be created, such as `h1`, `li`, `p`, `button`, `div`, `span`, ...
- `element` is the constructed element

Example:

```
var h1 = document.createElement( "h1" );
```

```
var para = document.createElement( "p" );
```

```
var span = document.createElement( "span" );
```



Create a text node

```
var textNode = document.createTextNode(data);
```

- data is a string containing the data to be put in a text node
- textNode is the constructed Text Node

After a Text Node is created, we can insert it into an element on the web page by calling:

```
element.appendChild(textNode)
```

```
element.insertBefore(textNode, existingChild)
```


Delete/change an element

- Remove an HTML element:

```
document.removeChild(element);
```

- Replace an HTML element:

```
document.replaceChild(new, old);
```

Manipulating element's attributes

- Accessing an attribute value

```
element.getAttribute(attrName);
```

```
element.setAttribute(attrName, attrValue);
```



Example: add dependant

Let the user enter dependants information and display them in an **unordered list**.

First name:

Last name:

Relationship:

- James Smith, son
- Mary Smith, daughter

Example: add dependant

```
<input type="text" id="firstNameInput" />
```

```
<input type="text" id="lastNameInput" />
```

```
<input type="text" id="relationInput" />
```

```
<button onClick="addDependant()">Add dependant</button>
```

```
<ul id="personList">
```

```
</ul>
```

First name:

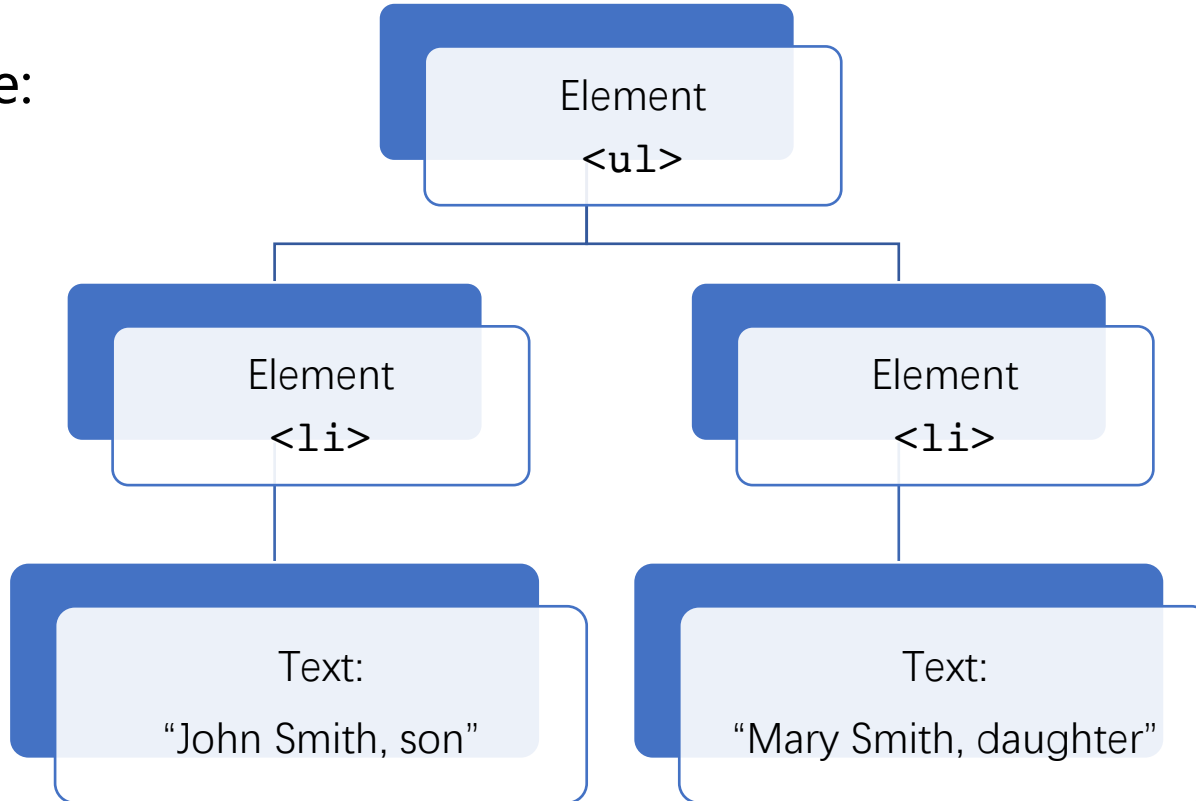
Last name:

Relationship:

- James Smith, son
- Mary Smith, daughter

Example: add dependant

DOM tree:



```
<ul id="personList">
```

```
</ul>
```

First name:

Last name:

Relationship:

- James Smith, son
- Mary Smith, daughter

Example: add dependant

```
function addDependant() {  
  
    // get person info  
  
    // create a new list item holding the person info  
  
    // add the new list item element to the unordered list  
  
}
```

First name:	<input type="text" value="Mary"/>
Last name:	<input type="text" value="Smith"/>
Relationship:	<input type="text" value="daughter"/>
<input type="button" value="Add dependant"/>	
<ul style="list-style-type: none">• James Smith, son• Mary Smith, daughter	

Example: add dependant

```
function addDependant(){  
    // get person info  
    var firstNameTf = document.getElementById("firstNameInput");  
    var firstName = firstNameTf.value;  
    var lastNameTf = document.getElementById("lastNameInput");  
    var lastName = lastNameTf.value;  
    var relationTf = document.getElementById("relationInput");  
    var relation = relationTf.value;  
    var personInfo = firstName + " " + lastName + ", " + relation;
```

First name:

Last name:

Relationship:

- James Smith, son
- Mary Smith, daughter

Example: add dependant

```
function addDependant(){  
    // get person info  
    ...  
    var personInfo = firstName + " " + lastName + ", " + relation;  
  
    // create a new list item holding the person info  
    var li = document.createElement("li");  
    var personText = document.createTextNode(personInfo);  
    li.appendChild(personText);  
  
    // add the new list item element to the unordered list  
    var personUL = document.getElementById("personList");  
    personUL.appendChild(li);  
}
```

First name:

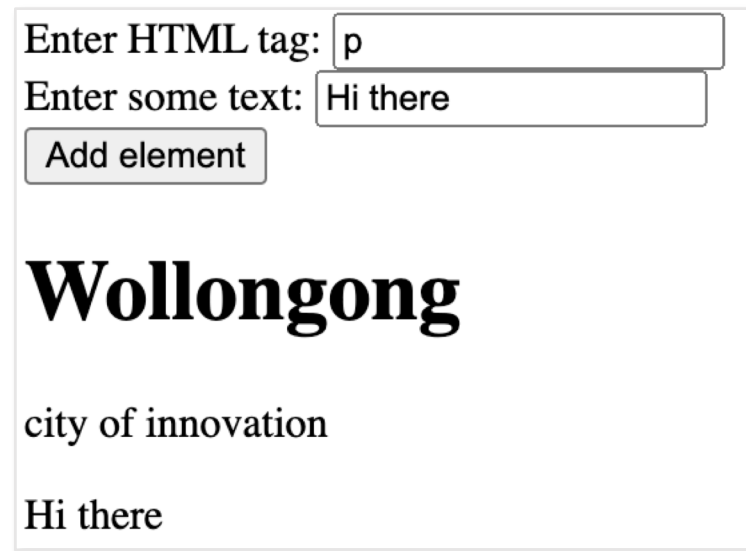
Last name:

Relationship:

- James Smith, son
- Mary Smith, daughter

Example: add dynamic elements

Let the user enter HTML tag name and some text, then create an element with the specified tag name holding the text, and display the created element on the webpage.



The screenshot shows a web interface with two input fields and a button. The first input field is labeled "Enter HTML tag:" and contains the text "p". The second input field is labeled "Enter some text:" and contains the text "Hi there". Below these fields is a button labeled "Add element". Below the button, the output of the form is displayed: a large, bold, black serif font "Wollongong", followed by the text "city of innovation" in a smaller, regular serif font, and finally "Hi there" in the same regular serif font.

Example: add dynamic elements

```
<input type="text" id="tagInput" />
```

```
<input type="text" id="textInput" />
```

```
<button onClick="addElement()">Add element</button>
```

```
<div id="elementHolder">
```

```
</div>
```

Enter HTML tag:

Enter some text:

Add element

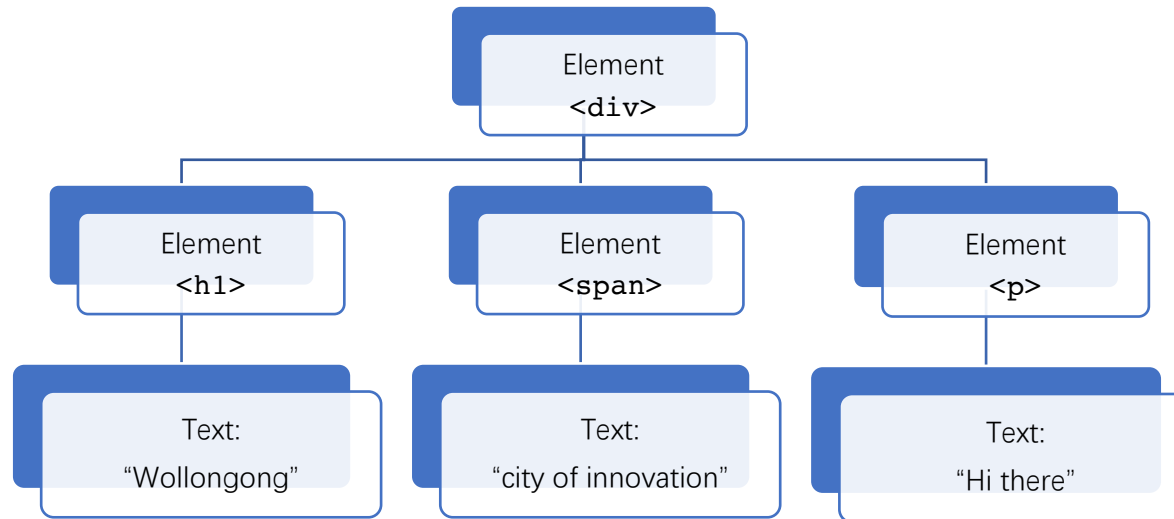
Wollongong

city of innovation

Hi there

Example: add dynamic elements

DOM tree:



Enter HTML tag:

Enter some text:

Add element

Wollongong
city of innovation
Hi there

```
<div id="elementHolder">
```

```
</div>
```

Example: add dynamic elements

```
function addElement(){  
  
    // get tag name and text from user  
  
    // create a new element holding the text  
  
    // add the new element to the div  
  
}
```

Enter HTML tag:

Enter some text:

Add element

Wollongong

city of innovation

Hi there

Example: add dynamic elements

```
function addElement(){  
  
    // get tag name and text from user  
  
    var tagTf = document.getElementById("tagInput");  
  
    var tag = tagTf.value;  
  
    var textTf = document.getElementById("textInput");  
  
    var text = textTf.value;
```

Enter HTML tag:

Enter some text:

Wollongong

city of innovation

Hi there

Example: add dynamic elements

```
function addElement(){  
  
    // get tag name and text from user  
  
    ...  
  
    // create a new element holding the text  
  
    var element = document.createElement(tag);  
  
    var textNode = document.createTextNode(text);  
  
    element.appendChild(textNode);  
}
```

Enter HTML tag:

Enter some text:

Wollongong

city of innovation

Hi there

Example: add dynamic elements

```
function addElement(){  
    // get tag name and text from user  
    ...  
    // create a new element holding the text  
    ...  
    // add the new element to the div  
  
    var holderDiv = document.getElementById("elementHolder");  
  
    holderDiv.appendChild(element);  
  
}
```

Enter HTML tag:

Enter some text:

Add element

Wollongong

city of innovation

Hi there

References

- Jennifer Niederst Robbins, Learning Web Design - A Beginner's guide to HTML, CSS, JavaScript and Web Graphics, 5th edition, O'Reilly Media, 2018.
- Eric T. Freeman and Elisabeth Robson, Head First JavaScript Programming, O'Reilly Media, 2014.
- <http://www.w3schools.com/js>
- <http://developer.mozilla.org/en-US/docs/Web/JavaScript>
- https://developer.mozilla.org/enUS/docs/Web/API/Document_Object_Model

