香 港 中 文 大 學
The Chinese University of Hong Kong

CSCI2720 - Building Web Applications

Lecture 8: Events and Objects

Dr Colin Tsang

Outline

- The DOM Tree
- Accessing HTML elements
- Navigation
- Events
- Objects
- Something about **this**
- JSON
- jQuery legacy

The COM tree

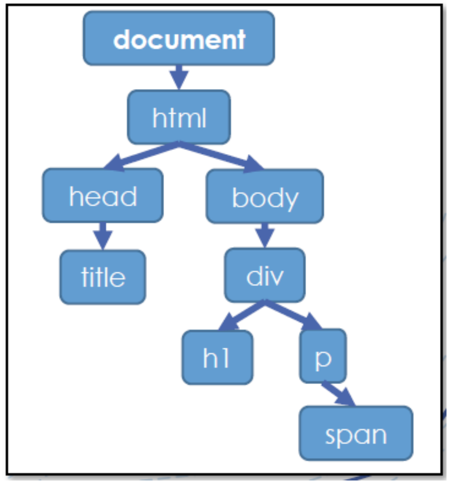
- To render an HTML document, the browser builds a tree data structure as the **window.document** object
  - This object represents the browser window
  - All global variables and functions are by default members of **window** without explicitly mentioning

- The tree is built only once when the page is loaded.

The DOM Tree

- A tree data structure is a hierarchical structure consisting of nodes connected by edges, where each node can be multiple child nodes but only one parent node.
- The tree is called the *Document Object Model* (DOM)
  - Objects of all elements
  - Properties of elements
  - Methods to access
  - Events
- DOM level 4 (2015)
  - Living standard of DOM by WHATWG

```
<html>
<head>
  <title>Hello!</title>
</head>
<body>
  <div>
  <h1>Nice try</h1>
  <p>DOM is <span>fun</span>!</p>
  </div>
</body>
</html>
```



Accessing HTML elements

- Selectors API
  - Using the same selectors as in CSS, returning only the first match: **querySelector()**
  - Similar but returning all matches as a list: **querySelectorAll()**

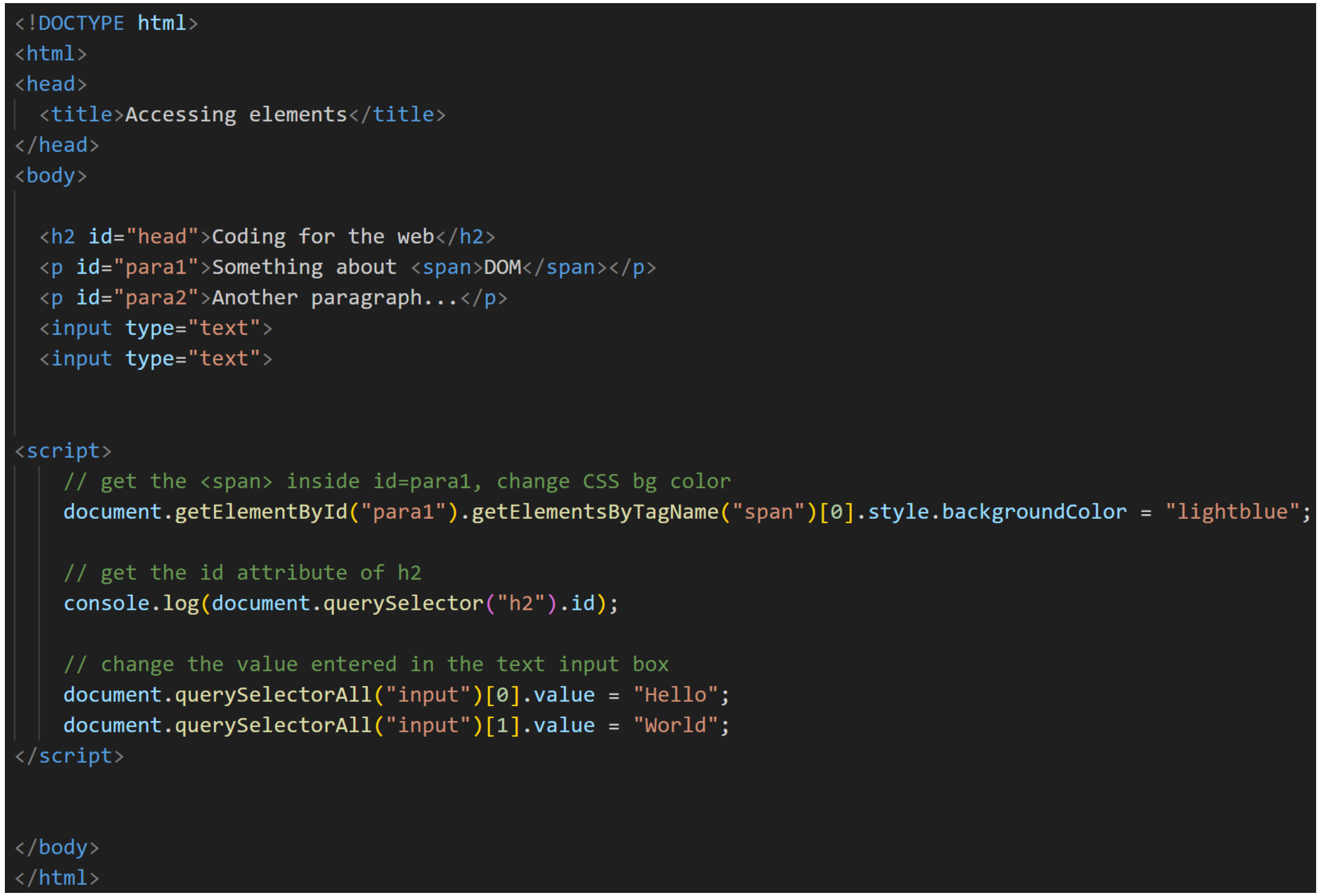
- Traditional techniques
  - Document method – specifying unique ID in document: **getElementById()**
  - Element method – can search for children within one element, as a list: **getElementsByClassName()**, **getElementsByTagName()**.

- Object collections
  - **document.images**, **document.links**, **document.scripts**

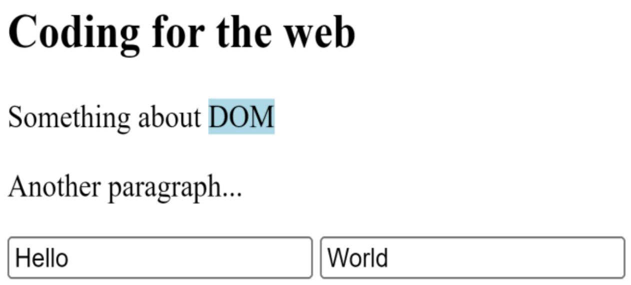
Accessing HTML elements

- Contents and properties can be fetched or modified
  - *element.innerHTML* – contents including tags
  - *element.innerText* – contents in plain text
  - *element.value* – only for form elements
  - *element.attribute* – setting HTML attribute directly (e.g., class)
  - *element.style.property* – setting CSS property directly

Accessing HTML element



```html
<!DOCTYPE html>
<html>
<head>
  <title>Accessing elements</title>
</head>
<body>

  <h2 id="head">Coding for the web</h2>
  <p id="para1">Something about <span>DOM</span></p>
  <p id="para2">Another paragraph...</p>
  <input type="text">
  <input type="text">


<script>
    // get the <span> inside id=para1, change CSS bg color
    document.getElementById("para1").getElementsByTagName("span")[0].style.backgroundColor = "lightblue";

    // get the id attribute of h2
    console.log(document.querySelector("h2").id);

    // change the value entered in the text input box
    document.querySelectorAll("input")[0].value = "Hello";
    document.querySelectorAll("input")[1].value = "World";
</script>


</body>
</html>
```



Coding for the web

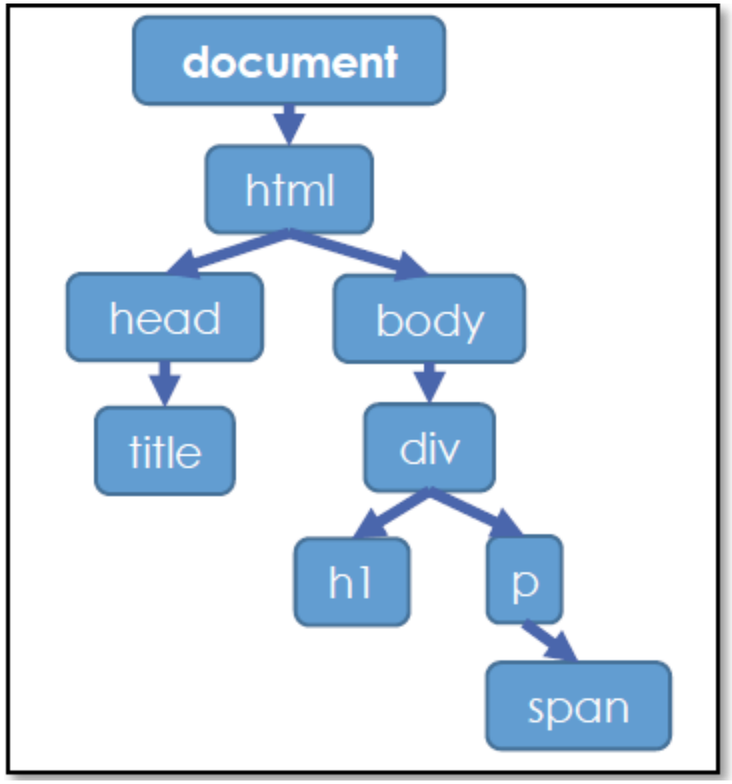
Something about DOM

Another paragraph...

Hello | World

Navigation

- Navigation between nodes
  - **parentNode**
  - **children[node#]**
  - **firstElementChild**
  - **lastElementChild**
  - **nextElementSibling**
  - **previousElementSibling**
- Node editing
  - **createElement()**
  - **createTextNode()**
  - **appendChild()**
  - **insertBefore()**
  - **remove()**
  - **removeChild()**
  - **replaceChild()**



Events

- Events have a vital role for web interaction, e.g.
  - *onclick*
  - *onload*
  - *onunload*
  - *onchange*
  - *onmouseover*
  - *onfocus*

- Either specify event to object or use the **addEventListener()** method.

Objects

- JS is not really an object-oriented programming language, and its objects are basically:
  - A *collection of properties*: key/value pair, e.g.
    - **{name: “john”, age: 18}**

- The key must be string or symbol
  - *obj[1]* and *obj[‘1’]* are equivalent
  - *obj.1* is not possible because *1* is not a valid identifier
  - But *obj.x* is ok, meaning *obj[‘x’]*

- The value can be anything, even another object or *null*

Creating objects

- Use literal notation

```
let stu1 = {name:"john", age:18} // the most commnly seen method
```

- Define properties directly

```
let stu1 = new Object();
stu1.name = "john";
stu1.age = 18;
```

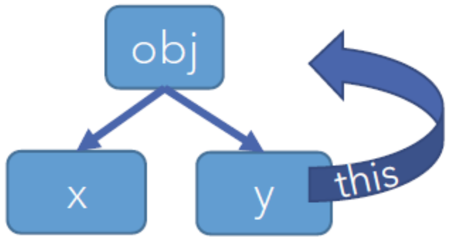
- Use a constructor (function/class)

```
function Student(name, age){
    this.name = name;
    this.age = age;
}
let stu1 = new Student("john", 18)
let stu2 = new Student("tom", 17)
```

Something about **this**

- Usually, this refers to the parent object

```
let obj = {
    x: 10,
    y: function() { console.log(this.x) }
}
obj.y(); // outputs 10
```



- What if *obj.y* is copied to another local global variable?
  - The parent of *a* and *b* is *window*

```
x = 20;
let a = obj.y;
b = obj.y;
a(); // outputs 20
b(); // outputs 20
```

Something about **this**

- In most cases, the keyword this is in a function refers to the object through which the function is being called
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/this

- Note: it is special for arrow functions.
  - No **this** for arrow functions.
  - If you use it, it retain the value of the surrounding context.
  - https://www.codementor.io/@dariogarciamoya/understanding-this-in-javascript-with-arrow-functions-gcpjwfyuc

JSON: JavaScript Object Notation

- Getting popular as a lightweight data-interchange format
    - Content type: application/json
- Closely resembles a subset of JavaScript syntax, although it is not a strict subset
- String literals within a JSON string must be enclosed by double quotes
- Support nested structures
    - e.g., objects within objects, array of object, etc.
- For the detailed syntax of JSON, see: http://json.org/

JSON

- Two ways of JSON representation:
  - A collection of name/value pairs – object literal
    - In other languages, this can be realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
      - e.g.: an object with three properties named a, b, and c
        - { “a”:1, “b”:2, “c”:3 }
  - An ordered list of values = array literal
    - In other languages, this can be realized as an array, vector, list, or sequence
      - e.g.: an array of three integers and one string value
        - [1, 2, 3, “value #4”]
- Note: JSON supports UTF-8 for non-ASCII characters

Using JSON in JavaScript

- JSON is a piece of string, but can be easily parsed in JS objects
  - `let myJSONtext1 = '{"name":"john", "age":18}'; // pay attention to quotes!`

- Decode JSON encoded data
  - `let myData = JSON.parse(myJSONtext);`

- Encode data
  - `let myJSONtext2 = JSON.stringify(myData); // return a string`

The jQuery legacy

- jQuery has been around in the web for over 10 years, yet fading out now because the improvement in JS
- jQuery is a JS library built on top of DOM
  - Performance: DOM performs faster than jQuery
  - Ease-of-use: jQuery is convenient
    - Less code to write, uniform interface, etc.
  - Cross-browser compatibility: jQuery is better perhaps
- Which one to use?
  - https://youmightnotneedjquery.com/

Further readings

- w3schools
  - https://www.w3schools.com/js/js_htmldom.asp
- MDN introduction to DOM
  - https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
- MDN introduction on JSON
  - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON