

JS Events and Objects

CSCI2720 2020–21 Term 2
Building Web Applications



香港中文大學
The Chinese University of Hong Kong

Dr. Chuck-jee Chau
chuckjee@cse.cuhk.edu.hk

Outline

- ▶ The DOM Tree
- ▶ Accessing HTML elements
- ▶ Navigation
- ▶ Events
- ▶ Objects
- ▶ The keyword `this`
- ▶ JSON
- ▶ The jQuery legacy

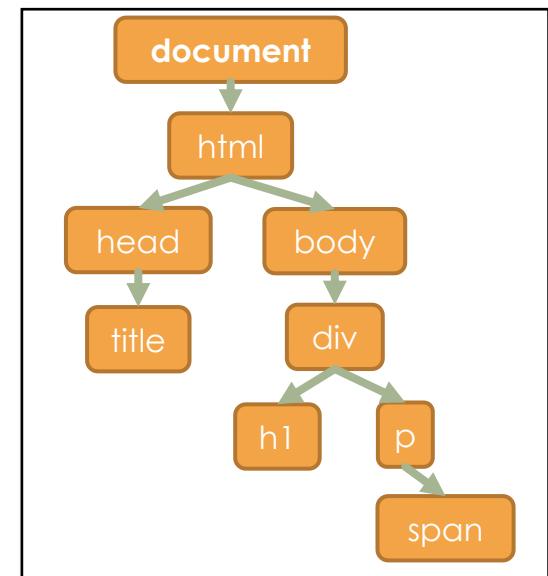
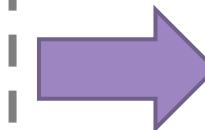
The DOM Tree

- ▶ To render an HTML document, the browser builds a tree data structure as the **window.document** object
 - ▶ The **window** object represents the browser window
 - ▶ All global variables and functions are by default *members* of **window** without explicitly mentioning
 - ▶ **document** refers to **window.document**
- ▶ The tree is built when the page is loaded, only *once!*

The DOM Tree

- ▶ 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 methods — can search for children within one element, as a list:
`getElementsByName()`, `getElementsByTagName()`
- ▶ Object collections
 - ▶ E.g. `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.setAttribute` — setting HTML attributes directly (e.g. `class`)
 - ▶ `element.style.property` — setting CSS property directly
 - ▶ CSS property in *camelCase*, e.g. `border-left` → `borderLeft`

Accessing HTML elements

```
<h2 id="head">CSCI2720</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>
```

<https://codepen.io/chuckjee/pen/MWbgobd>

CSCI2720

Something about **DOM**

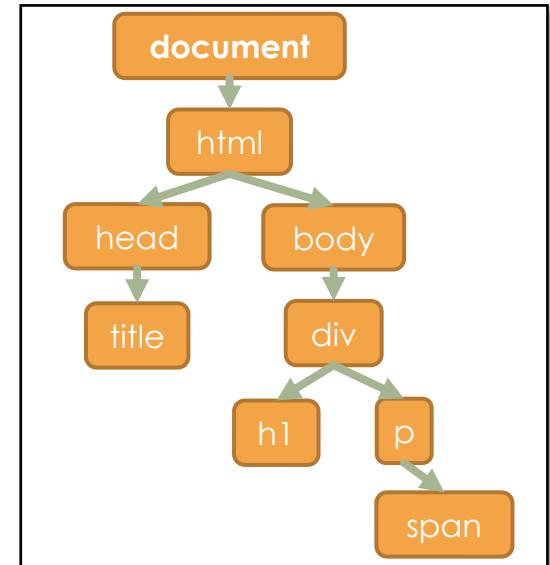
Another paragraph...

Hello

World

Navigation

- ▶ Navigation between nodes
 - ▶ parentNode
 - ▶ childNodes[*node#*]
 - ▶ firstChild
 - ▶ lastChild
 - ▶ nextSibling
 - ▶ previousSibling
- ▶ Node editing
 - ▶ createElement()
 - ▶ createTextNode()
 - ▶ appendChild()
 - ▶ insertBefore()
 - ▶ remove() / removeChild()
 - ▶ replaceChild()



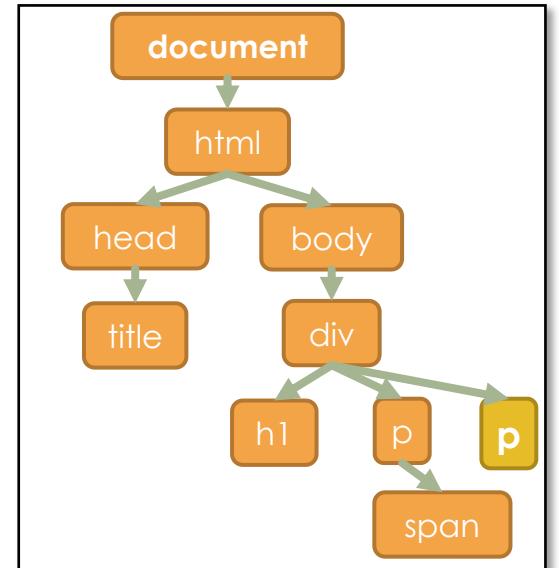
Navigation

```
<div>
  <h1>Nice try</h1>
  <p>DOM is <span>fun</span>!</p>
</div>

<script>
// these are document methods!
let para = document.createElement("p");
let node = document.createTextNode("Are you sure?");
para.appendChild(node);

let child = document.querySelector("div").lastChild;
document.querySelector("div").insertBefore(para, child);
</script>
```

<https://codepen.io/chuckjee/pen/qBqWjvj>



Events

- ▶ Events have a vital role for web interaction, e.g.
 - ▶ onclick
 - ▶ onload / onunload
 - ▶ onchange
 - ▶ onmouseover / onmouseout
 - ▶ onfocus
- ▶ Either specify event to object, or use the addEventListener() method

Events

```
--> <p id="p1">Click me</p>
--> <p id="p2">Mouseover me</p>
--> <input type="text" value="Change me">

--> <script>
--> // show an alert box on click
--> document.querySelector("#p1").onclick = function() { alert("Clicked!"); }

--> // change CSS style on mouse over/out
--> document.querySelector("#p2").onmouseover =
-->   function() { document.querySelector("#p2").style.color = "red"; }
--> document.querySelector("#p2").onmouseout = () =>
-->   document.querySelector("#p2").style.color = "blue"; // arrow function

--> // print to console when input box is changed
--> document.querySelector("input").onchange = () => console.log(document.querySelector("input").value);
--> </script>
```

<https://codepen.io/chuckjee/pen/MwbgvYy>

Objects

- ▶ JavaScript 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 a 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 all right, meaning `obj['x']`
- ▶ The *value* can be anything, even another object or `null`

Creating objects

- ▶ Many ways to create an object

- ▶ Define properties directly

```
var stu1 = new Object();
stu1.name = "John"; // equivalent to student['name']
stu1.age = 18;      // equivalent to student['age']
```

- ▶ Use *literal notation*

```
var stu1 = { name:"John", age:18 }
```

- ▶ Use a constructor

- ▶

```
function Student(name, age) {      // this is slightly different
    this.name = name;                // from the examples above
    this.age = age;
}
var stu1 = new Student("John", 18); // instantiate Student
var stu2 = new Student("Jim", 19);  // i.e. make new instance
```

The keyword `this`

```
var obj = {  
    x: 5,  
    y: function () {  
        console.log( this.x );  
    }  
};  
  
z = obj.y;          // Copy the function by reference  
  
// What is the output? (note: z === obj.y)  
obj.y();  
z();
```

- ▶ In most cases, the keyword **this** in a function refers to the object through which the function is being called
 - ▶ See: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/this>
 - ▶ Note: It is special for arrow functions!
See <https://www.codementor.io/@dariogarciamoya/understanding-this-in-javascript-with-arrow-functions-gcpjwfyuc>

The keyword `this`

```
// continued from the previous slide ...

obj.y();           // Output: 5

// z() is evaluated as window.z() in a browser, so during
// this function call, "this" refers to the window object
z();               // Output: undefined

var obj2 = { x: 999 };
obj2.z = obj.y;

// During this function call, "this" refers to obj2
obj2.z();          // Output: 999
```

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 objects, etc.
- ▶ For the detailed syntax of JSON, see: <http://json.org>

JSON

- ▶ A collection of name/value pairs — Object literal
 - ▶ In various 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 most languages, this is 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 into JS objects

```
let myJSONtext = '{"name": "John", "age": 18}'; // check quotes!
```

- ▶ Decode JSON encoded data

```
let myData = JSON.parse(myJSONtext);  
// giving a String, Number, Boolean, Null, Object, or Array
```

- ▶ Encode data

```
let myJSONText = JSON.stringify(myData); // giving a string
```

The jQuery legacy

- ▶ *jQuery* has been around in the web for over 10 years, yet fading out now because of the improvement in JS...
- ▶ jQuery is a JavaScript library built *on top of* DOM
 - ▶ Performance: DOM performs faster than jQuery
 - ▶ Ease-of-use: jQuery is convenient!
 - ▶ less code to write, uniform interface, functionalities, selectors, *etc.*
 - ▶ Cross-browser compatibility: jQuery is better perhaps
- ▶ Which one to use? What is the difference?
 - ▶ <http://youmightnotneedjquery.com>

Read further...

- ▶ w3schools.com JavaScript HTML DOM Tutorial
 - ▶ 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: Working with JSON
 - ▶ <https://developer.mozilla.org/en-us/docs/Learn/JavaScript/Objects/JSON>

Checkpoint

How to navigate the DOM tree?

Can I add a text node to become a child of one sibling somewhere?

What CSS styles can be modified using DOM methods?