

CS2204 Fundamentals of Internet Applications Development

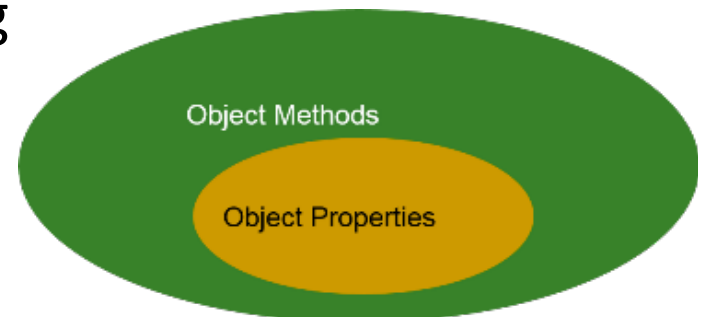
Lecture 10 JavaScript – Part3

Computer Science, City University of Hong Kong

Semester A 2023-24

How Does An Object Look Like?

- An object contains **two main** parts:
- **Properties**
 - **values** associated with an object, such as length, and width; styles and events are also properties
 - can **get/change** their values by JS
- **Methods**
 - **actions** that can be preformed on objects, such as write() of the document object, i.e., document.write()
 - use them in JS to do something



Built-In Objects In JavaScript

- The following objects are **built-in** JavaScript:
 - Boolean
 - Math
 - Date
 - Array
 - String

Boolean

- **Boolean** is a **primitive data type** but can also be viewed as **object**.

Create Boolean by:

```
var myboolean = true (or false), or  
var myboolean = new Boolean (value)
```

- **Value**

- specifies **the initial value** of the Boolean object
- the value is converted to a Boolean value, if necessary
 - if value is omitted or is 0, -0, null, false, NaN, undefined, or the empty string (""), the object has an initial value of **false**
 - all other values, including any object or the string "false", create an object with an initial value of **true**

- **Print Boolean value**

- *All these give a false Boolean*

```
❑ var myBoolean=new Boolean()  
❑ var myBoolean=new Boolean(0)  
❑ var myBoolean=new Boolean(null)  
❑ var myBoolean=new Boolean("")  
❑ var myBoolean=new Boolean(false)  
❑ var myBoolean=new Boolean(NaN)
```

- *All these give a true Boolean*

```
❑ var myBoolean=new Boolean(true)  
❑ var myBoolean=new Boolean("true")  
❑ var myBoolean=new Boolean("false")  
❑ var myBoolean=new Boolean("Richard")
```

Code Example: lec10-01-JS-boolean.html

Math Object

- Math object is **not** a function object (**NO** need to “new”)
 - can access it without using a constructor
- Math object contains:
 - **properties** - mathematical **constants**
 - mathematical constant examples: Math.PI, Math.LN2, and Math.LN10
 - **methods** - mathematical **functions**
 - mathematical function examples: Math.max(number), and Math.round(number)
 - Math.random(): return a random value in [0, 1), where 1 is **exclusive**

Code Example: lec10-02-JS-math.html

Date Object

- The Date object is used to work with **dates** and **times**. Create Date object by using **new**:

```
var mydate = new Date ()
```

```
new Date (dateString)
```

```
new Date (yr_num, mo_num, day_num [, hr_num, min_num, sec_num, ms_num])
```

no argument	the constructor creates a Date object for today's date and time according to local time
milliseconds	an integer value , representing the number of milliseconds since 1 January 1970 00:00:00 UTC
dateString	a string value , representing a date the string should be in a format recognized by the parse method
yr_num, mo_num, day_num	integer values, representing year, month, and day month is representing by 0 to 11 with 0=January, and 11=December
hr_num, min_num, sec_num, ms_num	integer values representing hours, minutes, seconds, and milliseconds

Date Object

- The most **common error** is not creating the date object and use the methods

Remember to use Date constructor

```
today = new Date();
```

- Some useful **methods** of the date object
 - today.getDate() - returns 1-31
 - today.getDay() - returns 0-6
 - today.getMonth() - returns 0-11
 - today.getFullYear() - returns the current year
 - today.getHours(), etc.

Array

- An array is a list of values associated with a variable name
- Create an array using **literal**

```
var myarray = [ ];
```

- Creating **an array object** MUST use the **new** operator

```
var myarray = new Array();
```

```
var myarray = new Array (element0, element1, ... , elementN);
```

```
var myarray = new Array (arrayLength);
```

- Array length
 - it specifies the **length** of the array
 - you can access this value using the **length property**

Code Example: lec10-04-JS-array.html

Array

- **Delete** an element
 - `pop()`; delete the **last** element
 - `shift()`; delete the **first** element
- **Add** element(s)
 - `push(values)`; add values to the **end** of the array
 - `unshift(values)`; add values to the **beginning** of the array
- `join()` **combines** array's elements as a string (does not change the original array)
 - `arr.join('separator')`

String

- Similar to Boolean, string is a **primitive type** and can be viewed as an object. To create:

```
var str = 'string';
```

```
var str = new String('string');
```

- Properties:
 - length
- Methods:
 - indexOf() and lastIndexOf()
 - charAt()
 - substr(start, length)
 - replace(pattern, replacement)
 - split(separator)
 - concat(string1, string2)

Example

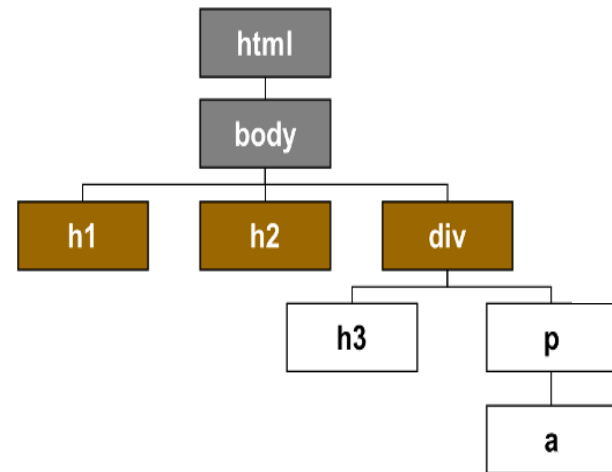
- Given a string, find the character that **appears most time** in the string and **remove it** from the string

```
9   var str = 'fjkdojfdkajofdjkaokakfda';
10  var obj = new Object();
11
12  for (var i=0; i<str.length; i++) {
13      var ch = str.charAt(i);
14      if (obj[ch]) {
15          obj[ch] += 1;
16      } else {
17          obj[ch] = 1;
18      }
19  }
20
21  var ch = '';
22  var vMax = 0;
23  for (var k in obj) {
24      if (obj[k] > vMax) {
25          vMax = obj[k];
26          ch = k;
27      }
28  }
29  console.log(ch + ' ' + vMax);
```

```
48  var newStr = '';
49  for (var i=0; i<str.length; i++) {
50      if (str.charAt(i) !== ch) {
51          newStr += str.charAt(i);
52      }
53  }
54
55  console.log(newStr);
```

DOM

- The **DOM (Document Object Model)** is the main bridge between the Web page and JavaScript
 - **Document**
 - **Element**: tag
 - **Node**: element, attribute, text
 - Each node can be viewed as an **object**



- JavaScript would **find/select** an object and then **changes** its properties/content or call its methods

How to Select an Element?

- **By ID**

- `document.getElementById('id')`
- get the corresponding **element object**

- **By tag name**

- `document.getElementsByTagName('tagName')`
- get **an array** of **elements objects** of this tag

- **By combination**

- e.g., `document.getElementById('id').getElementsByTagName('p')`
- Critical thinking

Code Example: lec10-07-JS-byID.html

How to Select an Element?

- By **CSS selector**
- Select one each time
 - `document.querySelector('CSS selector')`
 - `document.querySelector('#id')`
 - `document.querySelector('.className')`
 - `document.querySelector('tagName')`
 - `document.querySelector('#id1, #id2')` – return either one found first
 - get the first **element object**
- Select all
 - `document.querySelectorAll('CSS selector')`
 - get **an array** of all selected **elements objects**

How JavaScript Works With Objects

- **Properties**

- the properties of an object, such as font properties, color properties, and box properties can be **read** or **changed** with JS

- **Methods** - use them to do something, such as:

- *alert()* - it should be `window.alert()`, the object is window, to alert something to the window
- *document.write()* - write something to the document object, i.e., the Web page
- `document.querySelector("video").play()`

- **Event Handlers**

- they are functions are "**attached**" to objects and used to trap events happening to their owning object
- example events: *onclick*, *onmouseover*, *onchange*, etc.

Event

- What is an **Event**?
 - The time at which something happens
 - Many different events, such as:

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

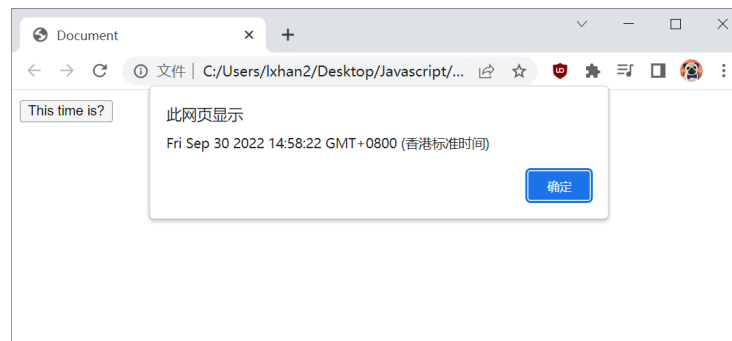
For full list, you can refer to: https://www.w3schools.com/jsref/dom_obj_event.asp

Two Ways to Refer an Event

HTML element's attribute

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <title>Document</title>
6  </head>
7  <body>
8    <button onclick="eventHandler();">
9      This time is?
10   </button>
11   <script>
12     function eventHandler() {
13       alert(Date());
14     }
15   </script>
16 </body>
17 </html>
18
```

Code Example: lec10-09-JS-event-attribute.html



Event

- Three **important** aspects
 - 1) **Where** will the event happen?
 - **Source** of the event
 - 2) **Which type** of the event to be handled
 - Refer to one **event example**, e.g., onclick
 - 3) **How** to handle the event?
 - Event handler, which is usually a **function**

Two ways to refer an Event

HTML element's attribute

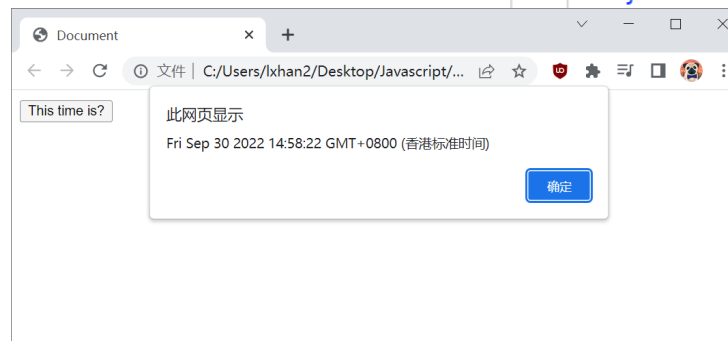
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6 </head>
7 <body>
8   <button onclick="eventHandler();">
9     This time is?
10  </button>
11  <script>
12    function eventHandler() {
13      alert(Date());
14    }
15  </script>
16 </body>
17 </html>
18
```

Code Example: lec10-09-JS-event-attribute.html

An object's property

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6 </head>
7 <body>
8   <button id="btn">
9     This time is?
10  </button>
11  <script>
12    var btn = document.querySelector("#btn");
13    btn.onclick = eventHandler;
14
15    function eventHandler() {
16      alert(Date());
17    }
18
```

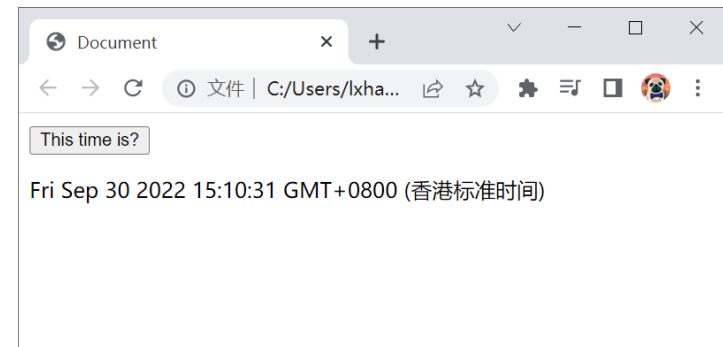
Code Example: lec10-10-JS-event-object.html



Event Handler

- A piece of JavaScript codes, usually a **function**
 - Tell the object how to react when that event occurs

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6 </head>
7 <body>
8   <button id="btn">
9     This time is?
10  </button>
11  <p id="output"></p>
12  <script>
13    var btn = document.querySelector("#btn");
14    btn.onclick = eventHandler;
15
16    function eventHandler() {
17      document.getElementById('output').innerHTML=Date();
18    }
19  </script>
20 </body>
21 </html>
```



eventHandler replaces the content of `<p>` whose id is "output" by the current time.

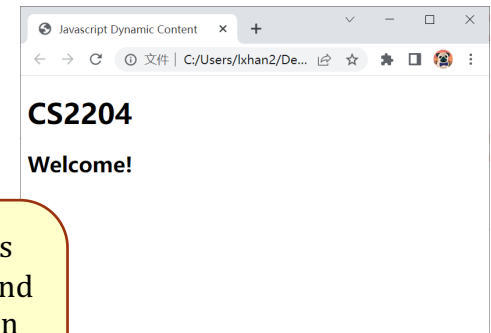
Code Example: lec10-11-JS-event-handler.html

Change Element's Content

- **innerHTML** allows access of **the content** of an element (or actual HTML) as a string, e.g.,

1. Originally the “welcome” div has no content: `<div id="welcome"></div>`
2. After the page has finished loading, the `onload` function is invoked which will call the `showDynamicContent()` function
3. The following statement
`document.getElementById("welcome").innerHTML = "<h2>Welcome!</h2>";`
replaces the current content of the “welcome” div with the string “`<h2>Welcome!</h2>`” such that the webpage will be displayed as if the html of the “welcome” div is
`<div id="welcome"><h2>Welcome!</h2></div>`

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Javascript Dynamic Content</title>
6     <script>
7       function showDynamicContent() {
8         document.getElementById("welcome").innerHTML="<h2>Welcome!</h2>";
9       }
10    </script>
11  </head>
12  <body onload="showDynamicContent();">
13    <!-- Page content begins here -->
14    <h1>CS2204</h1>
15    <div id="welcome"></div>
16    <!-- Page content ends here -->
17  </body>
18 </html>
19
```



Usually the `onload` **event handler** is added as an attribute in the body tag and is used to call some JavaScript function to carry out some tasks right after the webpage has finished loading to initialize some settings

Change Element's Attribute

- Set a new value to an **attribute**

```
element.setAttribute('attributeName', 'value');
```

- e.g., `element.setAttribute('disabled', 'true');`
`element.setAttribute('style', 'background-color: red;');`
- it adds it as an **inline attribute** to the element
- Example: After clicking a button, the user can input a number, and the number is shown in the box
 - If number is **odd**, the background color of box is **red**
 - Otherwise, the background color is **sky blue**
 - After click, the button is **disabled**

Click

Message

Change Element's Attribute

```
6  <style>
7    div {
8      width: 300px;
9      height: 300px;
10     background-color: gray;
11     margin-top: 20px;
12
13     text-align: center;
14     line-height: 300px;
15   }
16 </style>
```

```
19  <button id="btn">Click</button>
20  <div>Message</div>
21
22  <script>
23    var btn = document.getElementById('btn');
24
25    btn.onclick = function() {
26      var num = prompt('Enter a number');
27      var box = document.querySelector('div');
28      box.innerHTML = num;
29      if (num%2 == 0) {
30        box.setAttribute('style', 'background-color: skyblue;');
31      } else {
32        box.setAttribute('style', 'background-color: red;');
33      }
34      btn.setAttribute('disabled', 'true');
35    }
36  </script>
```

Change Element's Attribute

- For `style`, it is more recommended to define all new styles **in a new CSS class**, e.g., `.newClass {}`, and use `setAttribute('class', 'newClass')`

```
16 .even {  
17     background-color: skyblue;  
18 }  
19 .odd {  
20     background-color: red;  
21 }
```

```
35 if (num%2 == 0) {  
36     // box.setAttribute('style', 'background-color: skyblue;');  
37     box.setAttribute('class', 'even');  
38 } else {  
39     // box.setAttribute('style', 'background-color: red;');  
40     box.setAttribute('class', 'odd');  
41 }  
42
```

Code Example: lec10-13-JS-attribute.html

Operate on <input>

- The content of `<input type="text">` is obtained by using `value`, not `innerHTML`
- The attribute `type` can be changed using `setAttribute('type', 'anotherType')`
- Example: in a registration form,
 - For the `username`, if the `username length` is **not** within [5, 10], when the mouse moves out, an error message is shown; when the user types again, the error message disappears
 - For `password`, when the small box is clicked, the `plaintext` of the password can be shown
 - Two more events
 - `element.onfocus`
 - `element.onblur`


UName: Input 5~10 characters

PWord: 

Operate on <input>

[illegible]

UName: Input 5~10 characters

PWord: 

```
<script>
    var mName = document.querySelector('#uname');
    var mMsg = document.querySelector('.msg');
    var mPwd = document.querySelector('#pwd');
    var mBox = document.querySelector('.box');

    var flag = 0;
    mBox.onclick = function() {
        if (flag == 0) {
            mPwd.setAttribute('type', 'text');
            flag = 1;
        } else {
            mPwd.setAttribute('type', 'password');
            flag = 0;
        }
    }

    mName.onblur = function() {
        var str = mName.value;
        if (str.length < 5 || str.length > 10) {
            mMsg.setAttribute('style', 'color: red;');
            mMsg.innerHTML = 'Invalid user name!';
        }
    }

    mName.onfocus = function() {
        mMsg.setAttribute('style', 'color: #ccc;');
        mMsg.innerHTML = 'Input 5~10 characters';
    }
}
</script>
```

Select Based on Node Relationship

- Select the **parent** node of an element or node
 - `element.parentNode;`
- Select the **children elements**
 - `element.children`

```
8 <div class="mCon">
9   <div class="nav">
10     <ul>
11       <li id="l1">1</li>
12       <li>2</li>
13       <li>3</li>
14     </ul>
15   </div>
16
17   <div class="content">
18     <ul>
19       <li>7</li>
20       <li>8</li>
21       <li>9</li>
22     </ul>
23   </div>
24 </div>
```

```
26 <script>
27   var item = document.getElementById('l1');
28   var parent = item.parentNode;
29   console.log(parent);
30
31   var myUl = document.querySelector('.content > ul');
32   var lis = myUl.children;
33
34   for (var i=0; i<lis.length; i++) {
35     console.log(lis[i].innerHTML);
36   }
37 </script>
```

Code Example: lec10-15-JS-relation.html

Event Handler

- The second way to **add** an event handler
 - Why we need it?
- Syntax

```
event.addEventListener(type, listener[, useCapture])
```

 - **type**: a **string** to represent an event (**no** prefix *on*)
 - e.g., 'click', 'mouseover', etc.
 - **listener**: **function** to handle this event

Code Example: lec10-16-JS-event-listener.html

Object This

- **this** can be used in the **event handler** to refer to **the assigned object**
 - Benefit: same event handler may be used for many similar objects

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Document</title>
6  <script>
7      window.onload = initAll;
8      function initAll() {
9          buttons = document.querySelectorAll("button")
10         for (i=0; i < buttons.length; i++) {
11             buttons[i].onclick = myEventHandler;
12         }
13     }
14     function myEventHandler() {
15         alert(this.id);
16     }
```

```
17     </script>
18 </head>
19 <body>
20     <button id="first">
21         First Button
22     </button>
23     <button id="second">
24         Second Button
25     </button>
26     <button id="third">
27         Third Button
28     </button>
29
30 </body>
31 </html>
```

Refer to the object assigned with
this event handler