# CS2204 Fundamentals of Internet Applications Development

## Lecture 9 JavaScript – Part4

*Computer Science, City University of Hong Kong*

*Semester B 2024-25*

# Lab schedule

**There is only one lab session left in this semester; please attend it on time**

| | | | | |
|---|---|---|---|---|
| **Week 10** | Mar 28th | **Lab 08** | **JavaScript event handlers** | Lab8-handout.pdf |
| **Week 11** | Mar 31st | | | |

https://canvas.cityu.edu.hk/courses/62237/pages/lab-information

# CW1 grading is out

If you have any questions about your CW1 grades, please contact the TA who graded your assignment.

If your points were deducted, the TA should have left a comment and pointed out which parts of the rubric do not meet the requirements, so you can email them via Canvas.

If the TA could not address your concerns, then you may contact the instructor directly.

# CW2 will be out after this lecture

You will be asked to continue work on your CW1 by better positioning the web elements, adding CSS3 effects, and applying JavaScript functions to enhance the interactivity of the website.

CW2 takes **13%** of the course assessment and will be due in three weeks on **April 18th, 23:59 PM**

Please **start as soon as possible** as they are **6 tasks** in this assignment!

CW2: Advanced CSS Effects and Interactivity for the Hong Kong Disneyland Website

Published | Assign To | Edit

In this assignment, you are supposed to continue working on what you did in CW1 to better position the web elements and enhance the interactivity of the Hong Kong Disneyland Website with JavaScript.

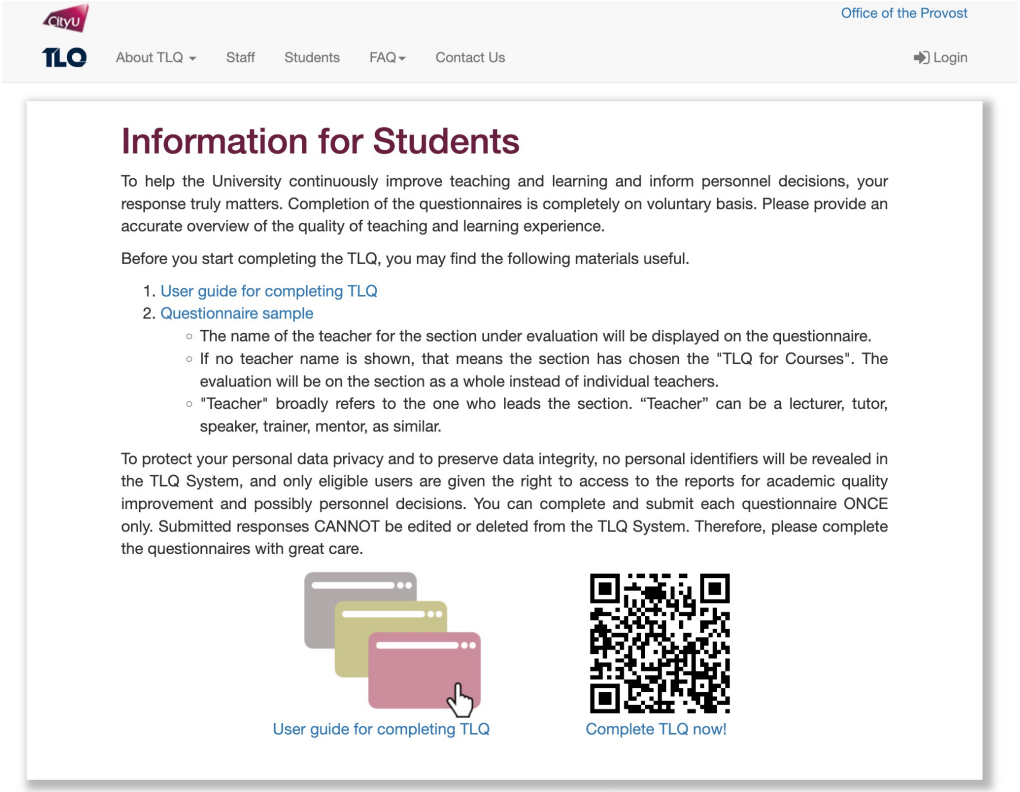
The detailed instructions for this assignment can be found here: CS2204-2425B-CW2.pdf or under file > Assignment > CW2.

**Important notes**:

- **We provided an external js function at: https://personal.cs.cityu.edu.hk/~cs2204/cs2204cw3.js, which you should link to the home page instead of making a copy to your js file.**
- You have three weeks to complete this assignment but we suggest you to start as earlier as possible because there are many tasks you need to do.
- Image sources needed in this assignment can be found under files > Assignments > CW1
- For videos, please link them directly to https://personal.cs.cityu.edu.hk/~cs2204/video/. Please **DO NOT** include them in your submission.

| | |
|---|---|
| **Points** | 13 |
| **Submitting** | a file upload |
| **File Types** | zip |

# LOQ Reminder




https://onlinesurvey.cityu.edu.hk/loq/

# Post-lab Quiz 6 Review

Create a background color transition effect (linearly changes the color from red to blue for 3 seconds) on the element with an id "trans" when mouse is over it

A.
```
#trans {transition: background-color 3s linear}
#trans:hover {background-color: blue}
```

B.
```
#trans {transition: background-color 3s linear}
#trans:hover {background-color: red to blue}
```

# Post-lab Quiz 6 Review

Which of the following is a CSS property that is used to style repeated color patterns

A. `background`
B. `background-image`

# Post-lab Quiz 6 Review

How to create a pop-up an alert window saying "Welcome!”

A. `Alert (“welcome!”)`

B. `alert (“welcome!”)`

# Question time: JS Functions (1)

1. What are the output of the following two JS program? What does `return y;` statement do?

**Program 1**

```
function f1(x) {
  var y = x * x * x;
  return y;
}

var w = f1(2);
console.log(w);
```

**Program 2**

```
function f1(x) {
  var y = x * x * x;
}

var w = f1(2);
console.log(w);
```

# Question time: JS Functions (2)

2. What will be displayed in the alert window in the following JS program?

```
var a = 0;
const b = 2;
function f1(){
  var b = 5;
}
function f2(){
  a = 6;
}
f1();
f2();
alert(a + b);
```

# Question time: JS Objects (1)

1. What are the four main objects in JS?

2. What are the two main parts of a JS object?

# Question time: JS Objects (2)

3. Given the following JS code, what will be the the output?

```
let person = {
        name: "John",
        age: 30,
        hobbies: ["reading", "hiking", "cooking"],
        address: {
                street: "Main St",
                city: "New York",
                state: "NY",
                zip: 10001 }
        };

        console.log(person.name);
        console.log(person.hobbies[1]);
        console.log(person.city);
```

# Question time: JS Objects (3)

4. Given the JS object `today = new Date ();` What's the output of `today.getMonth()`?

# Agenda

Review: JS Objects from Lec 08

DOM and Events

Dynamic Content

Multimedia: Video and Audio

# Agenda

Review: JS Objects from Lec 08

DOM and Events

Dynamic Content

Multimedia: Video and Audio

# Review: JS Objects (1)

JS is an **object-oriented programming language**: everything can be regarded as objects, including the primitive data types and functions

a **`car`** can be represented as an object with *properties* such as `color`, `brand`, `model`, and *behaviors* (i.e., functions) such as `drive()`, `stop()`

a **`person`** can be represented as an object with *properties* such as `name`, `gender`, `age`, and *behaviors* (i.e., functions) such as `walk()`, `sleep()`

# Review: JS objects (2)

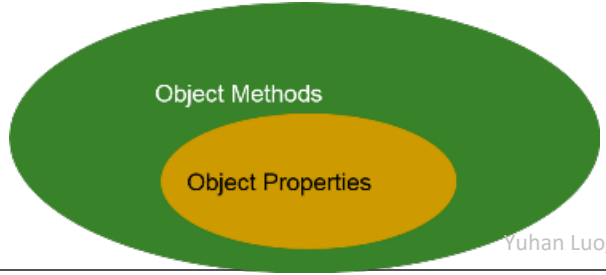
An object contains **two main** parts:

**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 (i.e., functions)**

- actions that can be preformed on objects, such as `write()` of the document object, i.e., `document.write()`



# Review: JS objects (3)

In JS, we can **self-define** objects or using **built-in** objects

To self-define objects, we can use:

- Literal

```
const person = {
firstName: "John",
lastName: "Doe",
fullName: function() {
  return this.firstName + " " + this.lastName;
    }
};

console.log(person);
console.log(person.fullName());
```

What is **this**?

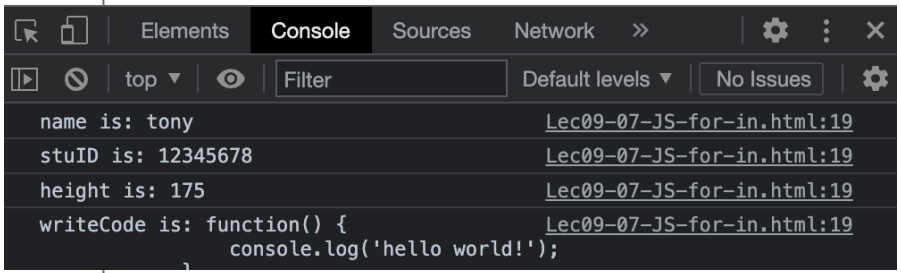
- `new Object ();` statement

```
var obj = new Object();
obj.name = 'tony';
obj.studentID = '12345678';
obj.height = 170;

obj.writeCode = function() {
    console.log('hello world!');
}

console.log(obj.name);
console.log(obj['height']);
obj.writeCode();
```

# Review: JS objects (4)

We can also iterate all the object properties and methods, as we iterate an array, where **`k`** is the index (and property name) and **`object[k]`** is the value of kth element (corresponding property value or method body)

```
var myStudent = {
    name: 'tony',
    stuID: '12345678',
    height: 175,
    writeCode: function() {
        console.log('hello world!');
    }
};

for (var k in myStudent) {
    console.log(k + ' is: ' + myStudent[k]);
}
```



# Review: Built-in JS object – Math (1)

Math object is **not** a function object (NO need to use "`new`")

- can access it without using a constructor

Math object contains:

- **properties** - mathematical constants
  - e.g., Math.PI, Math.LN2, and Math.LN10
- **methods** - mathematical functions
  - e.g., Math.max(number), and Math.round(number)

```
console.log(Math.PI);
console.log(Math.SQRT2);

console.log(Math.min(3, 6, 9));
console.log(Math.min(3, 6, 9, 'cs2204'));
console.log(Math.min());

console.log(Math.floor(1.2));
console.log(Math.floor(1.9));
console.log(Math.ceil(1.2));
console.log(Math.ceil(1.8));
console.log(Math.round(1.4));
console.log(Math.round(1.5));
```

Math.SQRT2 is commonly used, but note that there is **NO** Math.SQRT3 or Math.SQRT4, …

Code Example: lec09-01-JS-math.html

# Review: Built-in JS object – Math (2)

Math object is **not** a function object (NO need to use "`new`")

- can access it without using a constructor

Math object contains:

- **properties** - mathematical constants
  - e.g., Math.PI, Math.LN2, and Math.LN10
- **methods** - mathematical functions
  - e.g., Math.max(number), and Math.round(number)

```
console.log(Math.PI);
console.log(Math.SQRT2);

console.log(Math.min(3, 6, 9));
console.log(Math.min(3, 6, 9, 'cs2204'));
console.log(Math.min());

console.log(Math.floor(1.2));
console.log(Math.floor(1.9));
console.log(Math.ceil(1.2));
console.log(Math.ceil(1.8));
console.log(Math.round(1.4));
console.log(Math.round(1.5));
```

*Math.min()* returns infinity, how about *Math.max()?*

Code Example: lec09-01-JS-math.html

# Review: Built-in JS object – Math (3)

**`Math.random()`:** return a random value in [0, 1), where 1 is exclusive

What's the range of the random variable created by the following code create?v

```
Math.floor((Math.random()*10 + 1))
```

Given the following loop, what will be the range of the output?

```
res = '';
for (var i=0; i<10; i++) {
   res += Math.floor((Math.random()*10 + i)) + ' ';
  }
console.log(res);
```

**Critical thinking**: suppose that X and Y are two integers, how to create a random variable ranging from X – Y (including both X and Y)?



Code Example: lec09-01-JS-math.html

# Review: Built-in JS object – Date (1)

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

```
var mydate = new Date ();
new Date (milliseconds);
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 based on **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<br>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 |

Code Example: lec09-02-JS-date.html

# Review: Built-in JS object – Math (2)

To create a date object, remember to use Date constructor

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

Some useful methods (functions) 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() – returns `0-23`

Except that **`getDate()`** returns values starting from 1, `getMonth()`, `getHours()`, `getDay()` all returns values starting from **0**

# Review: Built-in JS object – Array (1)

An array is a data structure that stores a collection of values that can be of any data type. We can create an array using literal

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

Create an array using the new operator

```
var myarray = new Array(); /* can add elements later using
loops or depending on the development needs; recommended */
var myarray = new Array (element0, element1, … , elementN);
var myarray = new Array (arrayLength);
```



Code Example: lec09-02-JS-array.html

# Review: Built-in JS object – Array (2)

**Array length**

- Specifies the length of the array
- Can be accessed using the `length` property

**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

```
<script>
    var arr = [1, 2, 3];
    console.log("arr: " + arr);

    var arr2 = new Array();
    console.log("arr2: " + arr2);
    var arr3 = new Array(1, 2, 3, 4);
    console.log("arr3: " + arr3);
    console.log("\n");

    var arr4 = new Array(10);

    for (var i=0; i<arr4.length; i++) {
        arr4[i] = i + 1;
    }
    console.log("arr4: " + arr4);
    console.log("\n");

    var arr5 = [1, 2, 3];
    arr5.pop();
    console.log("arr5: " + arr5);
    arr5.shift();
    console.log("arr5: " + arr5);

    console.log("\n");

    var arr6 = [2, 3, 4];
    arr6.push(5, 6, 7);
    console.log("arr6: " + arr6);
    arr6.unshift(0, 1);
    console.log("arr6: " + arr6);

</script>
```

Code Example: lec09-03-JS-array.html

# Review: Built-in JS object – String

Similar to Boolean, string is a primitive type and can be viewed as an object.

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

**Properties**:

- length

**Methods**:

- indexOf() and lastIndexOf()
- charAt()
- substr(start, length)
- replace(pattern, replacement)
- split(separator)

```
var str1 = 'hello world!';
console.log("str1: " + str1);
var str2 = new String('hello world!');
console.log("str2: " + str2);


console.log("str2.indexOf('lo'): " + str2.indexOf('lo'));
console.log("str2.indexOf('l'): " + str2.lastIndexOf('l'));


console.log("st2.charAt(1): " + str2.charAt(1));


while (str2.indexOf('l') != -1) {
    str2 = str2.replace('l', '&');
}
console.log("str2: " + str2);


var str3 = 'hello&world&cs&2204';
var arr = str3.split('&');
for (var i=0; i<arr.length; i++) {
    console.log("arr[" + i + "]: "+ arr[i]);
}
```

Code Example: lec09-04-JS-string.html

# Agenda

Review: JS Objects from Lec 08

DOM and Events

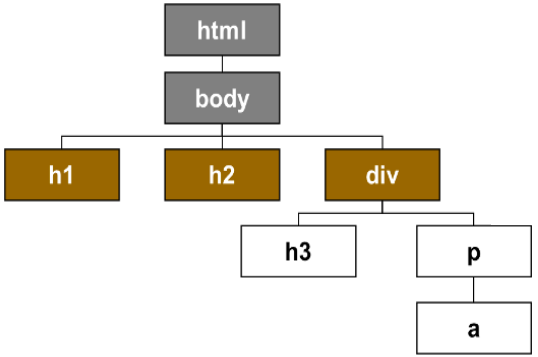
Dynamic Content

Multimedia: Video and Audio

# DOM

The DOM (Document Object Model) is the main bridge between the Web page and JavaScript

- Document
- Element: tag
- Node: element, attribute, etc.
- 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 HTML element in JS? (1)

By **ID**

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

To get the text content, we should call the property **innerHTML**

```
<div id="block1">
    this is a block
</div>

<div id="block2">
    <h1>heading1</h1>
    <h1>heading2</h1>
    <h1>heading3</h1>
    <p>this is a pargraph</p>
</div>

<ol>
    <li id="list1">The first item</li>
    <li id="list2">The first item</li>
</ol>
```

```
<script>
    var myDiv = document.getElementById('block1').innerHTML;
    console.log(myDiv); // output the text value of the object
    console.dir(myDiv); // output all the properties of the object

    var hs = document.getElementsByTagName('h1');
    console.log(hs);
    var ps = document.getElementsByTagName('p');
    console.log(ps);

    var divhs = document.getElementById('block2').getElementsByTagName('h1');
    console.log(divhs);

    var lis = document.getElementsByTagName('ol').getElementById('list1');
    console.log(lis);
</script>
```



Code Example: lec09-05-JS-byID.html

# How to select an HTML element in JS? (2)

By **tag name**

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

```
<div id="block2">
    <h1>heading1</h1>
    <h1>heading2</h1>
    <h1>heading3</h1>
    <p>this is a pargraph</p>
</div>
```

Code Example: lec09-05-JS-byID.html

**Critical thinking:** Will the following code show the HTML content? of h1?

```
var hs = document.getElementsByTagName('h1').innerHTML;
```

If no, how to make it show the HTML content?

# How to select an HTML element in JS? (3)

By **combination**

```
<div id="block2">
    <h1>heading1</h1>
    <h1>heading2</h1>
    <h1>heading3</h1>
    <p>this is a pargraph</p>
</div>
```

Code Example: lec09-05-JS-byID.html

The following code selects all the h1 elements in "block 2"

```
var divhs = document.getElementById('block2').getElementsByTagName('h1');
console.log(divhs);
```

Given the html, does the following code selects the element "list1"? Why or why not?

```
<ol>
    <li id="list1">The first item</li>
    <li id="list2">The first item</li>
</ol>
```

```
var lis = document.getElementsByTagName('ol').getElementById('list1');
console.log(lis);
```

# How to select an HTML element in JS? (4)

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
the first element object */
```

Select all

```
document.querySelectorAll('CSS selector')
```

get an array of all selected elements objects



Code Example: lec09-06-JS-query.html

# How to select an HTML element in JS? (5)

## By **CSS selector: Combination** also works

```
<div id="block1">this is a block</div>

<ol>
    <li class="list">item1</li>
    <li class="list">item2</li>
    <li class="list">item3</li>
</ol>

<div id="block2">
    <p id="ps1">this is the first paragraph another block</p>
    <p id="ps2">this is the second paragraph another block</p>
</div>
```

```
var divp = document.querySelectorAll('div > p');
console.log(divp);

var pp = document.querySelector('p ~ p');
console.log(pp);

var divpli = document.querySelectorAll('div, #ps2, .list');
console.log(divpli);
```



Code Example: lec09-06-JS-query.html

# How does JS Works With Web 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()` or `window.alert()` in JS contexts where the object is window
- *`document.write()`* - write to the document object, i.e., the Web page
- `document.querySelector("video").play()`

Often times, the interaction is trigger by **event handlers**

- **Functions** "attached" to objects and used to trap events happening to their owning object
- Example events: *`onclick`*, *`onmouseover`*, *`onchange`*, etc.

# What is an event?

An event occurs when the **user** or **browser** manipulate the page

| 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

# Three important aspects of an event

1) **Where** will the event happen?
   - Source of the event: a button, link, or an input field?

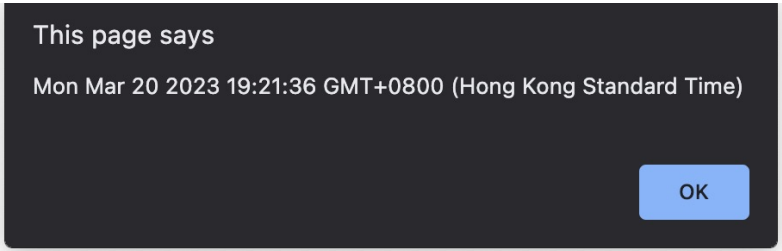
2) **Which type** of the event to be handled
   - An user action (e.g., *onclick*)?
   - Or a browser setting (e.g., *setTimeout()*)?

3) **How** to handle the event?
   - Event handler, which is usually a function

# Refer to an event (1)

## 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: Lec09-07-JS-event-attribute.html

# Refer to an event (2)

## 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: Lec09-07-JS-event-attribute.html

## An object's property

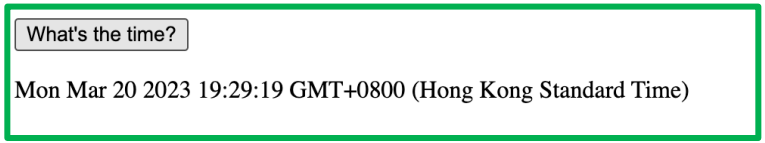
```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>Document</title>
6   </head>
7   <body>
8       <button id="btn">
9           This time is?
10      </button>
11      <script>
12          var btn = document.querySelector("#btn");
13          btn.onclick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

Code Example: Lec09-08-JS-event-object.html

# Event Handler

A piece of JavaScript codes, usually a **function** tells 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: Lec09-09-JS-event-handler.html

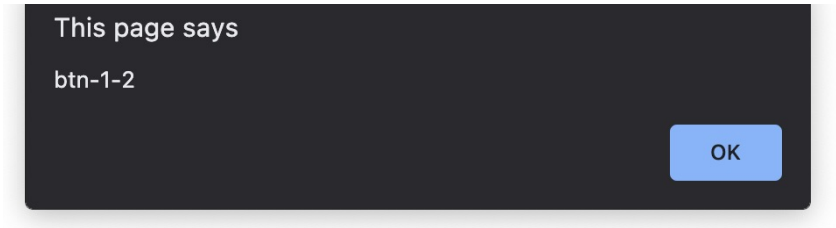
# Event Listener

A function that is registered to listen for a specific event on an element

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

```
<button id="btn1">Button 1</button>
<button id="btn2">Button 2</button>
```




When Button 1 is clicked

How about when the **Button 2** is clicked?

```
var btn1 = document.getElementById('btn1');
btn1.onclick = f1;

function f1() {
    alert('btn-1-1');
}
btn1.onclick = f2;

function f2() {
    alert('btn-1-2');
}

var btn2 = document.getElementById('btn2');
btn2.addEventListener('click', f3);
function f3() {
    alert('btn-2-1');
}
btn2.addEventListener('click', f4);
function f4() {
    alert('btn-2-2');
}
```

Code Example: Lec09-10-JS-event-listener.html

# Event Handler vs. Listener

**Question time:**

- What is the key difference between event handler and event listener?
- Which one is better?

# Example: Steps to Set Up Event

```
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               var btn = document.getElementById("btn");
10
11              btn.onclick = myEventHandler;
12          }
13
14          function myEventHandler() {
15              alert("This is an alert.")
16          }
17      </script>
18  </head>
19  <body>
20      <button id="btn">
21          Click Me
22      </button>
23
24  </body>
25  </html>
```

**Step 1**: find the object that you want to set up event for

**Step 3**: assign the event handler to the object. This can be replaced by `btn.addEventListener("click", myEventHandler);`

**Step 2**: define the event handler for it



Lec09-11-JS-event.html

# Object `this`

**`this`** can be used in the event handler to refer to the assigned object

- Advantages: same event handler may be used for many similar objects

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        window.onload = initAll;
        function initAll() {
            buttons = document.querySelectorAll("button")
            for (i=0; i < buttons.length; i++) {
                buttons[i].onclick = myEventHandler;
            }
        }
        function myEventHandler() {
            alert(this.id);
        }
    </script>
</head>
<body>
    <button id="first">
        First Button
    </button>
    <button id="second">
        Second Button
    </button>
    <button id="third">
        Third Button
    </button>

</body>
</html>
```

Code Example: Lec09-12-JS-object-this.html

Refer to the object assigned with this event handler

# Event Cancelling: null

After event handling, the event may not need to go on (e.g., on a campaign website, after pressing the "vote" button, one person shouldn't vote again)

If an event is added as follows

- `element.onclick = eventHandler;`
- we can cancel it by `element.onclick = null;`

If an event is added using `addEventListener`

- we can cancel it by `removeEventListener`

```
element.removeEventListener(type, listener[, useCapture])
```

```
<script>
    var btns = document.querySelectorAll('button');

    btns[0].onclick = f0;

    function f0() {
        alert(11111111);
        btns[0].onclick = null;
    }


    btns[1].addEventListener('click', f1);

    function f1() {
        alert(2222222);
        btns[1].removeEventListener('click', f1);
    }

</script>
```



Code Example: Lec-09-13-JS-delete.html

# Event Cancelling: onreset (1)

In JS, **`onreset`** is an event handler attribute that is used to specify a function to be executed when a form is reset

# Event Cancelling: onreset (2)

**`confirm()`** is a built-in function that displays a dialog box with a message and two buttons: "OK" and "Cancel".

Return a Boolean from the event handling function

- **False** ("Cancel"): cancel the event
- **True** ("OK"): the event continues

```
10    <script>
11      function check() {
12        if (confirm("Are you sure to clear all data?")) {
13          document.querySelector("input[name='firstname']").focus();
14          return true;
15        }
16        return false;
17      }
18    </script>
```

Confirm whether continue

If continue
  return true
else
  return false

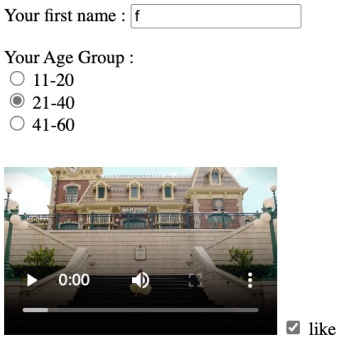
```
24    <hr>
25    <form action="#" method="get" onreset="return check();">
26      <p>
27      <label>Your first name : </label>
```

Receive return value on the form object

Code example: Lec09-14-JS-form-reset.html

# Event submission (1)

What if we want to display the user's selection upon submitting the form?




First Name: f, Age Group: 21-40, Nature: Likes nature videos, Wonders: Does not like wonders videos

Use `addEventListener()` !

```
document.getElementById('surveyForm').add
EventListener('submit', function(event) {

  … // get the form value

});
```

```
const formData = new
FormData(event.target);
// get the entire form
```

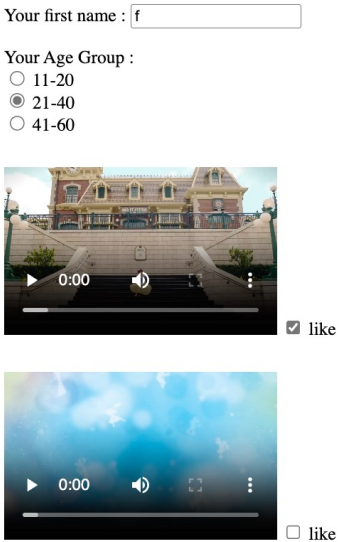
```
const firstname = formData.get('firstname');//
get the value from the input named 'firstname'
```

```
const likesNature = formData.has('nature') ?
'Likes nature videos' : 'Does not like nature
videos'; //get whether the user checks the box
named 'nature'
```

Code example: Lec09-14-JS-form-reset.html

# Event submission (2)

## The full script




```
<div id="results"></div>
<script>
document.getElementById('surveyForm').addEventListener('submit',
function(event) {
    event.preventDefault(); // Prevents form from submitting to the server
    const formData = new FormData(event.target);
    const firstname = formData.get('firstname');
    const agegroup = formData.get('agegroup');
    const likesNature = formData.has('nature') ? 'Likes nature videos' :
'Does not like nature videos';
    const likesWonders = formData.has('wonders') ? 'Likes wonders videos'
: 'Does not like wonders videos';

const results = `First Name: ${firstname}, Age Group: ${agegroup}, Nature:
${likesNature}, Wonders: ${likesWonders}`;

    document.getElementById('results').innerText = results;
  });
  </script>
```

First Name: f, Age Group: 21-40, Nature: Likes nature videos, Wonders: Does not like wonders videos

**`${}` can be used to wrap variable values in a string. e.g., `x + y = ${x+y}`; is the same as 'x + y = '+ x+y;**

Code example: Lec09-14-JS-form-reset.html

# Agenda

Review: JS Objects from Lec 08

DOM and Events

Dynamic Content

Multimedia: Video and Audio

# Dynamic Content

Common ways to **dynamically control and manipulate contents**:

Add/delete elements

- createElement
- innerHTML property
- inline script

Show or hide

- display
- visibility
- z-index
- Positioning

`setInterval()` and `clearInterval()`

# createElement (1)

Create an element node

```
document.createElement('elementName');
```

Add an element

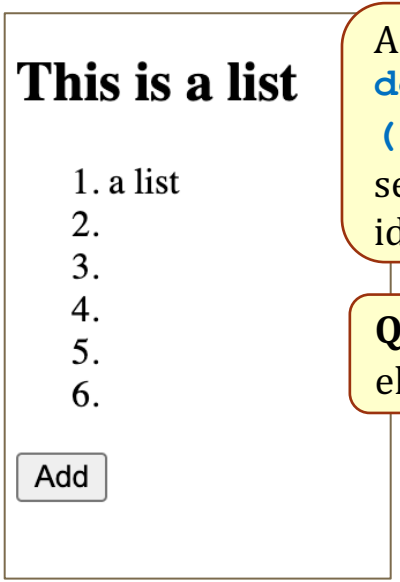
- Add a child node at the end of a parent node
  - `parentNode.appendChild('childNode');`
- Add a child node before another specified child node
  - `parentNode.insertBefore('childNode', 'specifiedNode');`
  - e.g., `specifiedNode` can be `parentNode.children[0]`

Two steps

- 1) create an element
- 2) add this element

# createElement (2)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <h2>This is a list</h2>
    <ol>
        <li>a list</li>
    </ol>

    <button>Add</button>

    <script>
        var btn = document.querySelector('button');
        btn.onclick = handler;


        function handler() {
            var li = document.createElement('li');
            var ol = document.querySelector('ol');

            ol.appendChild(li);
        }

        function handler2() {
            var li = document.createElement('li');
            var ol = document.querySelector('ol');

            ol.insertBefore(li, ol.children[0]);

        }


    </script>

</body>
</html>
```




As mentioned earlier, `document.querySelector('CSS selector')` is used to select elements with specified tags, ids, or class

**Question time**: how to insert `<li>` elements with text content?

Code Example: Lec09-15-JS-create.html

# innerHTML

Allows access of the content of an element (or actual HTML) as a string

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, in which the 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>`**

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Javascript Dynamic Content</title>
    <script>
        function showDynamicContent() {
          document.getElementById('welcome').innerHTML =
          '<h2>Welcome!</h2>';
        }
    </script>
  </head>

  <body onload="showDynamicContent();">

    <h1>CS2204</h1>

    <div id="welcome"></div>

  </body>
</html>
```

**CS2204**

**Welcome!**

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

Code Example: lec09-16-JS-innerHTML.html

# Question time

Every time when the button is clicked, can we add `<li>Item 1</li>`, `<li>Item 2</li>`, … `<li>Item 10</li>` … into `<ol>` respectively ?

**This is a list**

1. a list
2. Item 1
3. Item 2
4. Item 3
5. Item 4
6. Item 5

Add

# InnerHTML vs createElement (1)

Difference is the **efficiency**

To estimate the efficiency of a program, we usually focus on the time cost of executing it, which is the difference between the time starting the execution and the time finishing the execution.

```
function handler1() {
    var t1 = new Date();
    for (var i=0; i<500; i++) {
        var ls = document.createElement('li');
        o1.appendChild(ls);
    }
    var t2 = new Date();
    var time = t2 - t1;
    console.log(time);
    btn1.innerHTML = time;
}
```

The time before executing the loop

The time after executing the loop

The time cost to execute the loop



Code Example: lec09-18-JS-innerHTML3.html

# InnerHTML vs createElement (2)

## Comparing the efficiency of three functions

```
function handler1() {
    var t1 = new Date();
    for (var i=0; i<500; i++) {
        var ls = document.createElement('li');
        o1.appendChild(ls);
    }
    var t2 = new Date();
    var time = t2 - t1;
    console.log(time);
    btn1.innerHTML = time;
}
```

Creating each of the 500 elements `<li>` one by one

`createElement('li')` is **faster** because it directly manipulates DOM without the need to locating and modifying existing elements

```
var btn2 = document.querySelector('#btn2');
var o2 = document.querySelector('#o2');
btn2.onclick = handler2;

function handler2() {
    var t1 = new Date();
    for (var i=0; i<500; i++) {
        o2.innerHTML += '<li></li>';
    }
    var t2 = new Date();
    var time = t2 - t1;
    console.log(time);
    btn2.innerHTML = time;

}
```

Inserting and displaying 500 `<li>` one by one

```
var btn3 = document.querySelector('#btn3');
var o3 = document.querySelector('#o3');
btn3.onclick = handler3;

function handler3() {
    var t1 = new Date();

    var arr = [];
    for (var i=0; i<500; i++) {
        arr.push('<li></li>');
    }
    o3.innerHTML = arr.join('');

    var t2 = new Date();
    var time = t2 - t1;
    console.log(time);
    btn3.innerHTML = time;
```

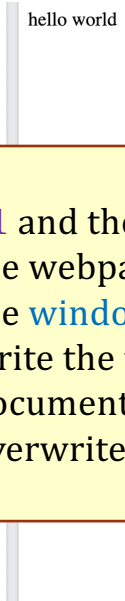
Adding 500 `<li>` to an array one by one and **display them all at once together**

Code Example: lec09-19-JS-innerHTML3.html

# Review: Inline Script

It will cause the page to re-render if the page is fully loaded, e.g., `onclick`, `window.onload`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <h1>
        Inline Script Example
    </h1>

    <button>Click</button>

    <script>
        var btn = document.querySelector('button');
        btn.onclick = function() {
            document.write('cs2204');
        }

        window.onload = function() {
            document.write('hello world');
        }
    </script>

</body>
</html>
```




h1 and the button “Click” is not shown on the webpage because the window.onload event listener is used to write the text "hello world" to the document after it has loaded, which overwrite the entire document

Code Example: Lec09-20-JS-inline-script2.html

# Hide & Show (1)

JS allows you to change the CSS properties of an element

- You can make objects appear or disappear by changing the display property, e.g., display
- There are other properties that we can change, e.g., backgroundcolor, etc.

```
63  <!-- Dynamic content one -->
64      <div id="sweet">
65      <p>
66          <img src="../images/1Nail.jpg" alt="">
67          <img src="../images/2Nail.jpg" alt="">
68          <img src="../images/3Nail.jpg" alt="">
69      </p>
70      </div>
71  <!-- Dynamic content two -->
72      <div id="sour">
73      <p> <img src="../images/4Nail.jpg" alt="">
74          <img src="../images/5Nail.jpg" alt="">
75      </p>
76      </div>
```

```
36  <script type="text/javascript">
37  function show(index) {
38      if (index == 1) {
39          document.getElementById("sweet").style.display="block";
40          document.getElementById("sour").style.display="none";
41      }
42      else {
43          document.getElementById("sweet").style.display="none";
44          document.getElementById("sour").style.display="block";
45      }
46  }
47  </script>
```

- Show as a block element

Code Example: Lec09-21-JS-dynamic-content.html



Display different contents when selecting a different link

# Hide & Show (2)

## Other methods

**Visibility property**

```
object.style.visibility="hidden"
object.style.visibility="visible"
```

**`Z-index`** - prepare all elements and stack them up with overlapping; change z-index to show either one by putting it on top.


```
<!DOCTYPE html>
<html>
<head>
<style>
#img1 {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1
}
</style>
</head>
<body>

<h1>This is a Heading</h1>

<img id="img1" src="book.png" width="100" height="180">

<button type="button" onclick="changeStack()">Change stack order</button>

<p>Default z-index is 0. Z-index -1 has lower priority.</p>

<p id="myP">This is a p element.</p>

<button type="button" onclick="hide()">Hide img</button>
<button type="button" onclick="show()">Display img</button>

<script>
function hide() {
  document.getElementById("img1").style.visibility = "hidden";
}

function show() {
  document.getElementById("img1").style.visibility = "visible";
}

function changeStack() {
  if (document.getElementById("img1").style.zIndex != "1")
    document.getElementById("img1").style.zIndex = "1";
  else
    document.getElementById("img1").style.zIndex = "-1";
}
</script>

</body>
</html>
```

Code Example: Lec09-22-JS-dynamic-content.html

# Call a function at specified intervals: `setInterval()`

The `setInterval()` method calls a function at specified intervals (in milliseconds).

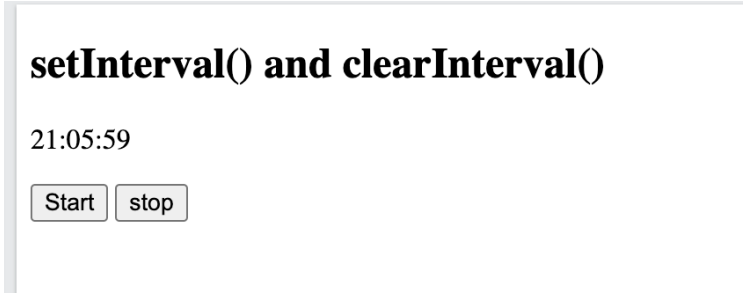
```
myInterval = setInterval(function, milliseconds);
```

It continues calling the function until `clearInterval()` is called, or the window is closed.

```
clearInterval(myInterval);
```

# `setInterval()` Example

```
<script>
var myInterval;

function startTimer() {
   myInterval = setInterval(myTimer, 1000);
}


function myTimer() {
  const date = new Date();
  document.getElementById("demo").innerHTML =
  date.toLocaleTimeString();
}

function stopTimer(){
  clearInterval(myInterval);
}
</script>
```




```
<!DOCTYPE html>
<html>
<body>

<h2>setInterval() and clearInterval()</h2>

<p id="demo">Time will show up here</p>
<button onclick="startTimer()">Start</button>
<button onclick="stopTimer()">stop</button>
```

Code Example: Lec09-23-JS-interval.html

# Agenda

Review: JS Objects from Lec 08

DOM and Events

Dynamic Content

Multimedia: Video and Audio

# Multimedia Programming

Before HTML5, although video and audio are supported with the `<object>`tag, there was no scripting ability

- Image is the only media that can be scripted

In HTML5, **video & audio scripting APIs** are provided in addition to the usual dynamic content techniques

# Video

Video & audio are timed media: have to be played over time

HTML5 provides many useful video properties &

```
.duration
.control
.oncanplay
.onended
```

And methods:

```
load()
play()
pause() // preferred over stop() as a W3C standard
```

# Video example

**Oncanplay** attribute of video element execute a JavaScript code when a video is ready to start playing

```
<body onload="init();">
    <div id="container">
        <video id="v" oncanplay="initButton();">
            <source src="https://personal.cs.cityu.edu.hk/~cs2204/video/Castle.mp4" type="video/mp4">
            <source src="https://personal.cs.cityu.edu.hk/~cs2204/video/Castle.ogg" type="video/ogg">
        </video>
        <div id="b">
            <button onclick="vplay();">play</button>
            <button onclick="vstop();">stop</button>
            <button onclick="vpause();">pause</button>
            <button onclick="vff();">fast forward</button>
            <button onclick="vfb();">fast backward</button>
```

```
<script>
    var v;

    function init() {
        v = document.getElementById("v");
    }

    function initButton() {
        document.getElementById('b').style.visibility = 'visible';
    }

    function vplay() {
        v.play();
    }

    function vpause() {
        v.pause();
    }

    function vstop() {
        v.currentTime = 0;
        v.pause();
    }

    function vff() {
        v.currentTime += v.duration / 10;
        if (v.currentTime >= v.duration) {
            v.currentTime = 0;
        }

    }

    function vfb() {
        v.currentTime -= v.duration / 10;
        if (v.currentTime < 0) {
            v.currentTime = 0;
        }
    }
</script>
```

use methods to play and pause the video

stop is to set the play time to the beginning and pause

fast forward is add video duration/10 to current play time, reset to 0 if larger than duration

fast backward: minus current play time


Code Example: Lec09-23-JS-script-video.html

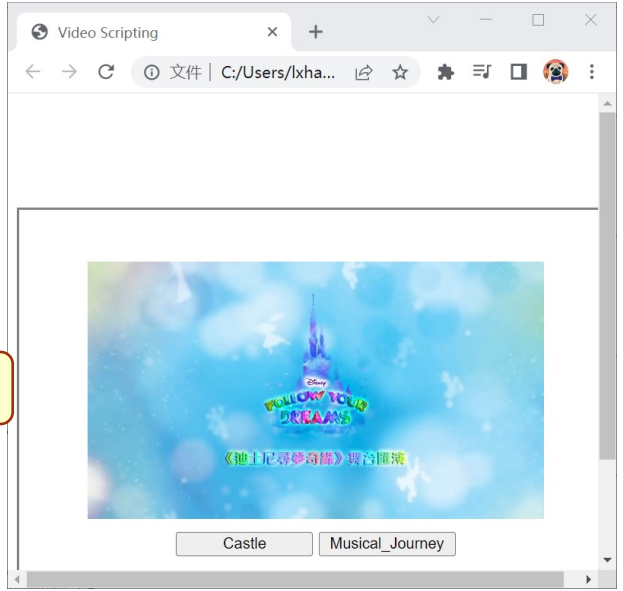
# Video example (Cont.)

How to **switch** videos by JS?

- by manipulating different HTML and their properties/attributes

```
        function init() {
            v = document.getElementById("v");
        }
        function initButton() {
            document.getElementById("b").style.visibility = "visible";
        }

        function switchVideo(n) {
            if (n%2 == 0) {
                document.getElementById('vdiv').innerHTML = '<video id="v"
oncanplay="initButton();" autoplay>' + '<source
src="https://personal.cs.cityu.edu.hk/~cs2204/video/Musical_Journey.mp4"
type="video/mp4">' + '<source
src="https://personal.cs.cityu.edu.hk/~cs2204/video/Musical_Journey.ogg"
type="video/ogg">' + '</video>';
            } else {
                document.getElementById('vdiv').innerHTML = '<video id="v"
oncanplay="initButton();" autoplay>' + '<source
src="https://personal.cs.cityu.edu.hk/~cs2204/video/Castle.mp4" type="video/mp4">' +
'<source src="https://personal.cs.cityu.edu.hk/~cs2204/video/Castle.ogg"
type="video/ogg">' + '</video>';
            }
        }
    </script>
</head>

<body onload="init();">
    <div id="container">
        <div id="vdiv">
            <video id="v" oncanplay="initButton();" autoplay>
                <source src="https://personal.cs.cityu.edu.hk/~cs2204/video/Castle.mp4"
type="video/mp4">
                <source src="https://personal.cs.cityu.edu.hk/~cs2204/video/Castle.ogg"
type="video/ogg">
            </video>
        </div>
        <div id="b">
            <button onclick="switchVideo(1);">Castle</button>
            <button onclick="switchVideo(2);">Musical_Journey</button>
        </div>
    </div>
</body>
```

Change the <video> tag by .innerHTML of container




Code Example: Lec09-24-JS-switch-video.html

# Audio

**Audio** is similar to video, except that it does not have a visual component

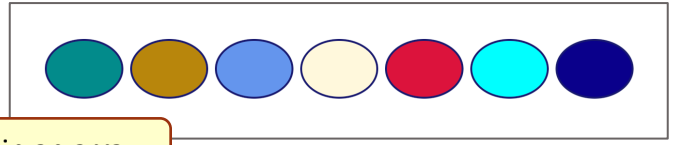
- audio can exist as a JS object **without** the `<audio>` tag
  - e.g., `var a= new Audio();`
  - especially useful for storing multiple audios for different events

```
41    <script type="text/javascript">
42    var audio=new Array();
43    var audioFile= new Array("a.wav","b.wav","c.wav","d.wav","e.wav","f.wav","g.wav");
44    //pre-load audio files
45    for (i=0; i<audioFile.length; i++) {
46      audio[i]= new Audio("../audio/"+audioFile[i]);
47      audio[i].load();
48    }
49    function init() {
50      //play a round of notes
51      for (i=0; i<audioFile.length; i++) {
52        audio[i].play();
53      }
54    }
55    function playAudio(m) {
56      audio[m].play();
57    }
58    </script>
```

storing many audio objects in an array

creating the audio object with a URL

Pre-loading the audio files so they could be played faster



```
60    <body onload="init();">
61      <div id="container">
62        <div id="a" onmouseover="playAudio(0)" class="note">
63        </div>
64        <div id="b" onmouseover="playAudio(1)" class="note">
65        </div>
66        <div id="c" onmouseover="playAudio(2)" class="note">
67        </div>
68        <div id="d" onmouseover="playAudio(3)" class="note">
69        </div>
70        <div id="e" onmouseover="playAudio(4)" class="note">
71        </div>
72        <div id="f" onmouseover="playAudio(5)" class="note">
73        </div>
74        <div id="g" onmouseover="playAudio(6)" class="note">
75        </div>
76      </div>
77    </body>
78  </html>
```

Code Example: Lec09-25-JS-switch-audio.html

# Lecture summary (1)

JavaScript is an object-oriented programming languages. Objects can be the primitive data types or complex data structures.

Each object has its properties and methods (i.e., functions). Likewise, there are also built-in objects and self-created objects.

`this` refers to an object depending on how the object is being invoked

JavaScript interacts with the HTML elements through DOM (Document Object Model) by assigning/changing its properties and call its methods. We can select HTML elements in JavaScript by ID, tag names, CSS queries, and combinations of these methods

EventHandlers are a special type of functions that tells the web object how to react when an event occurs.

# Lecture summary (2)

JavaScript can dynamically create, edit, remove, and change the style and display of HTML elements on a website, either through directly manipulating the DOM or modifying the innerHTML property of the elements

To modify an element, JS first need to select the element. This can be be done by `getElementById()or` CSS queries. Both can be combined to select multiple elements

To execute a function at regular intervals, we can use bulit-in JS function `setInterval()` and cancel it by `clearInterval()`

In HTML5, videos and audios can be directly manipulated by JS