

# Homework #6

Client-Side Scripting

---

KAIST

Prof. Myoung Ho Kim

# Contents

---

- ◆ JavaScript
- ◆ jQuery
- ◆ Ajax
- ◆ Json
- ◆ Homework Assignment #6

# JavaScript

---

- ◆ JavaScript : dynamic programming language for Web application
- ◆ JavaScript in HTML
  - Using <script> tag inside HTML head to link with HTML file
  - Include javascript code inside HTML or referring javascript file
    - Embedding : <script type="text/javascript"> ...code ... </script>
    - Referring : <script type="text/javascript" src="url" ></script>

```
<html>
  <head>
    <script type = "text/javascript">
      | document.write("hello world1!");
    </script>
  </head>
  <body>
    <script type = "text/javascript">
      | document.write("hello world2!");
    </script>
  </body>>
</html>
```

# JavaScript :Basic Syntax

---

- ◆ Comment : `//` , `/* */`
- ◆ Variable : declare a variable with the `let`, `var`
  - `let` : (ex) `let myVariable = 'CS360';`  
`myVariable = 'CS360-2021';`     `//after declaring`
- ◆ Constant : declare almost everything include function, object
  - `const` : (ex) `const obj = JSON.parse(data)`  
`//after declaring`
- ◆ Datatypes : String, Numbers, Boolean, Array, Object
  - (ex) `let myArray = [1, 'You', 2, 'ME'];`  
`let myObject = document.querySelector('H1');`

# JavaScript :Basic Syntax

---

## ◆ Operator

- Basic Operator(+,-,\*,/) : (ex) 6 \* 9; 'hello' + 'world';
- Assignment(=) : (ex) let myVariable = 'CS360';
- Equality(===) : returns a true/false(Boolean) result  
(ex) let myVariable = '3';  
myVariable === 4; //return false
- Not, Does-not-equal(!, !==) :  
(ex) let myVariable = '3';  
!(myVariable === 3); myVariable !== 3;

## ◆ Conditionals (if..else)

```
(ex) let class = 'CS360';  
    if(class === 'CS360'){  
        alert('Yes! I attend CS360 class!');  
    }else{  
        alert('No! I'm not attend CS360 class!')  
    };
```

# JavaScript :Basic Syntax

---

## ◆ Functions

(ex) let myVariable = document.querySelector('h1');

alert('hello!');

```
function multiply(num1,num2) {           //multiply(5, 7);
    let result = num1 * num2;
    return result;
}
```

## ◆ Events

- Code Structures that listen for activity in the browser & run code in response
- Select <html> elements, to attach an event handler
- onclick(), keyup(), ..

(ex) document.querySelector('img').onclick = function() {  
 alert('hello world!');  
}

# jQuery

---

## ◆ jQuery

- Open source based JavaScript library that simplifies the use of JavaScript languages
- HTML, DOM tree traversal & manipulation, event handling, Ajax

## ◆ Using jQuery

- Include `<script>` tag at HTML head
- `<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>`

# jQuery : Basic expression

## jquery.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>jQuery Intro</title>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
  <script>
    $(document).ready(function(){
      $("div").click(function(){
        $(this).hide();
      });
    });
  </script>
</head>
<body>
</body>
</html>
```

Connect with jQuery library using CDN

\$(document).ready(function(){  
    //jQuery method, action  
});

\$(function(){  
    //jQuery method, action  
});

Same expression!



# jQuery : Ajax

---

## ◆ Ajax

- Asynchronous JavaScript and XML
- Allow to update only a portion of the web page without reloading the entire web page
- Exchange data with servers in the background area

## ◆ Ajax with jQuery framework

- Ajax method : `$.ajax(URL, [options])`
- URL : the address of the server where the client will send the HTTP request
- Options
  - Type : HTTP method type(GET/POST)
  - Data : data to be sent to the server with HTTP requests
  - DataType : Type of data that the server will send to client

# jQuery : Ajax

---

## ◆ Ajax example

```
$.ajax({  
    url : "/example.jsp",  
    type : "GET",  
    data : {name : 'CS360', location:'E3-1'},  
    dataType : 'json',  
    success : function(response){  
        alert('success');  
    },  
    error: function(){  
        alert('error');  
    }  
});
```

//address of the server  
//HTTP request method(GET/POST)  
//data to be sent to server  
// Type of data that the server will send to client  
//if this request&response is successful,  
it comes to success callback

# jQuery : Various Methods

---

## ◆ jQuery method

- Selector : Use CSS selector

- ID : `$('#zero')` (ex) `<input type="text" id="zero">`
- Class : `$('.zero')` (ex) `<input type="text" class="zero">`
- Attribute : `$('[value="zero"]')` (ex) `<input type="text" value="zero">`
- You can also select two CSS attributes  
(ex) `$('#id1 #id2')` //select id1 and id2

# jQuery : Various Methods

---

## ◆ jQuery method

### - Method

- .attr() : get or set attributes to element      (ex) \$('h1').attr('value');
- .val() : get or set the value of the form      (ex) \$('input#id1').val('ABC');
- .append() : append the content end of the selected element  
(ex) \$('ul').append('<li>first line</li>');
- .html() : get or change the content of the selected element  
(ex) \$('div').html('<h1> hello world! </h1>');
- .map(array, callback function) : translate all items in an array or object to new array of items.

```
(ex) var dimensions = { width: 10, height: 15, length: 20 };  
    var keys = $.map( dimensions, function( value, key ) {  
        return key;  
    });  
    // result : ["width", "height", "length"]
```

# Example 1

## ◆ jQuery-Ajax example 1 (**click** event)

- When user click the button id = 'id1', 'click' event ajax will be executed

```
<script>

    $('#id1').click(function(){    // 'click' event
        let value = $(this).attr('value');
// ajax    $.ajax({                // $(this) refers to the selected element $('#id')
        url: "/example.jsp",        you can use "this" to avoid duplication
        type: 'GET',
        data: {button : value},
        success: function(res) {
            var data = JSON.parse(res);
            $('.exClass').html('<h1> Success!</h1>');
        },
        error: function(){
            alert('error');
        }
    });

</script>
```

# jQuery UI

---

## ◆ jQuery UI

- Collection of user interfaces that collect various components and widgets that express views based on jQuery

## ◆ Using jQuery UI

- Include css & js files
  - » `<link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">`
  - » `<script src="https://code.jquery.com/jquery-1.12.4.js"></script>`
  - » `<script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>`

# jQuery UI : autocomplete

---

## ◆ Autocomplete

- Use with jQuery UI
- Options
  - Source : defines the data to use, works when typing at the input field
    - function(object request, object response(object data))
    - Data can be array types : [{label : “hi”, value : “hello”}, ... ]
      - Label : display in suggestion menu, label is always treated as text
      - Value : will be inserted into the input element when user selects an item
  - Minlength : the minimum number of character for inquiry
  - Select : occurs when selecting a corresponding field in the search list
    - Event : type of the event
    - Ui : object
      - Item : an object with label and value
- More at : <https://api.jqueryui.com/autocomplete/>

# Example 2

## ◆ jQuery-Ajax example 2 (**autocomplete** with jQuery UI)

- When user type more than 1 length in form, then it communicate with http/ajax and show the list of the value that existed in database
- Then user can select the one of the item in the list, then system autocomplete its value

```
$(function(){
    $( "#autocomplete" ).autocomplete({
        source : function( request, response ) {
            $.ajax({
                type: 'post',
                url: "/autocomplete.jsp",
                dataType: "json",
                data: { value : "hello" },
                success: function(data) {
                    response(
                        $.map(data, function(item) {
                            return {
                                label: item.data,
                                value: item.data
                            }
                        })
                    );
                }
            });
        },
        minLength: 1,
        select: function( event, ui ) {
            console.log(ui);
        }
    });
});
```

//source functioned when user type something  
//server response with json & make a list with source

**item**  
(ex) data = [{"data":"hello", "name":"CS-360"}, {"data":"hello", "name":"CS-360"}]

//response : data to suggest to user  
// label : represent on list  
// value : value that represent on input form when user select one  
**You can set more attributes, but must have "label" & "value"**

// client send request, when (length of user input) >= 1

// when user select one of the response, return ui  
ui.item => object that represent the selected one  
(ex) ui.item.label / ui.item.value



# JSON

---

## ◆ JSON

- JavaScript Object Notion
- Used for data exchange in Client & Server
- (ex) `var obj = {"name" : "kim", "age" : "20"};`

## ◆ JSON in Java(JSP)

- Using 'json' , 'json-simple' library

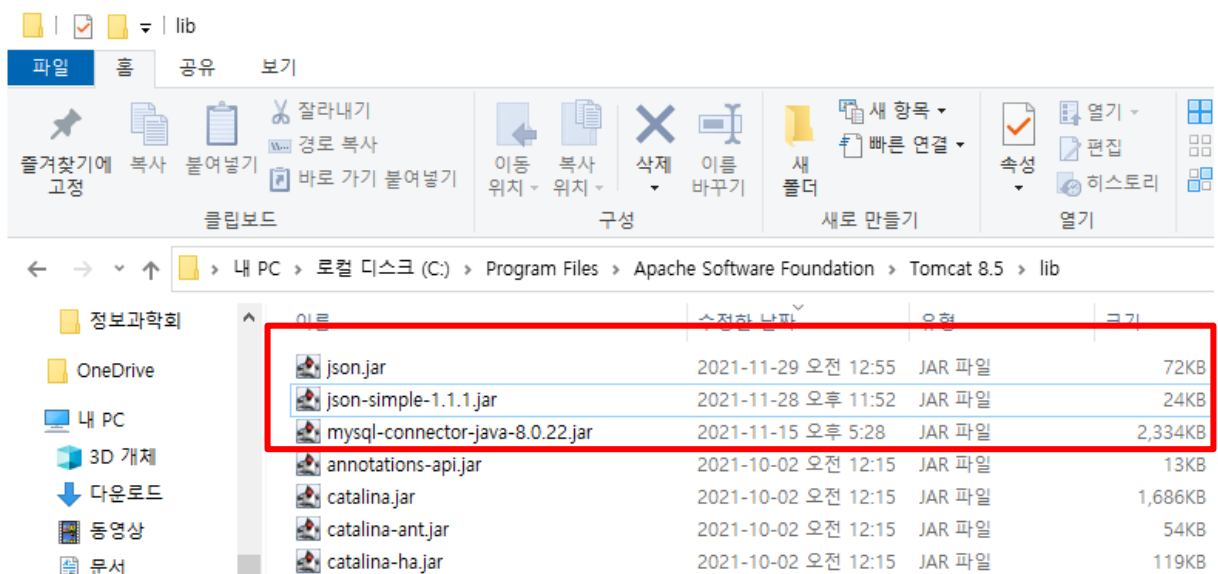
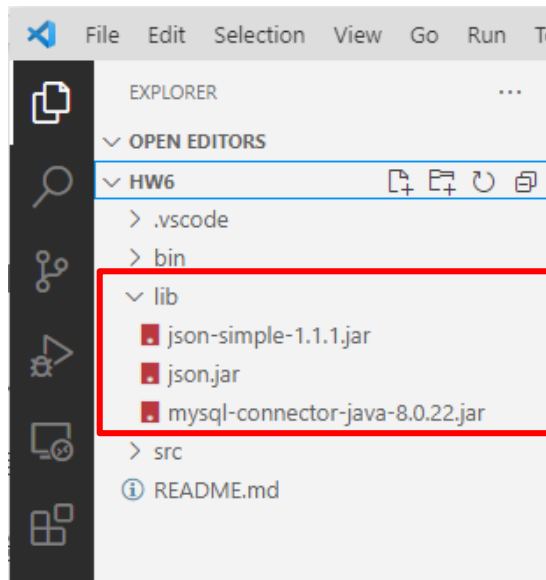
# JSON in JSP(JAVA)

- ◆ Download 'json' & 'json-simple-1.1.1.jar' in hw6.zip
- ◆ Copy 'json' & 'json-simple-1.1.1.jar', 'mysql-connector-java-8.0.22.jar' to 1)java project > lib directory & 2) tomcat > lib directory

(For Example)

1) C:\Users\junseon\Desktop\hw6\hw6\lib

2) (windows)C:\Program Files\Apache Software Foundation\Tomcat 8.5  
(macOS)\Downloads\apache-tomcat-8.5.72\lib



# JSON in JSP(JAVA)

---

## ◆ Import package

- `<%@ page import = "org.json.simple.JSONArray"%>`
- `<%@ page import = "org.json.simple.JSONObject"%>`

## ◆ JSONArray

- Make a JSON array that can include json object
- (ex) `JSONArray arr = new JSONArray();`
- Put each Json Object into Json Array with `.put();`
- (ex) `arr.put(obj);`

## ◆ JSONObject

- Make a JSON Object
- (ex) `JSONObject obj = new JSONObject();`
- Put each Json content into Json Object with `.put();`
- (ex) `obj.put("class", "CS360");`

# JSON in JSP(JAVA)

## ◆ JSON example in JSP/JavaScript

- Server send data with json format
- When client receive the data, parsing it to javascript object

### Server.jsp

```
<%  
  
...  
  
ResultSet rs = pstmt.executeQuery();  
JSONArray arr = new JSONArray();  
  
while(rs.next()){  
    JSONObject obj = new JSONObject();  
    obj.put("roomname", rs.getString("roomname"));  
    obj.put("location", rs.getString("location"));  
  
    arr.put(obj);  
}  
  
out.print(arr);  
  
%>
```

### Client.jsp(only javascript part)

```
$.ajax({  
    url : "server.jsp"  
    , type : "GET"  
    , data : {word1 : word1}  
    , success : function(res){  
        var data = JSON.parse(res);  
    }  
})
```



# Environment setup at VSCode

## ◆ Create a java project for hw6

1. Open the command palette(Ctrl+Shift+P)
2. Select 'Java: Create Java Project'
3. Select 'No build tools'
4. Browse your folder to save your project
5. Input your java project name
6. Start your 'Tomcat Brower' same as hw5

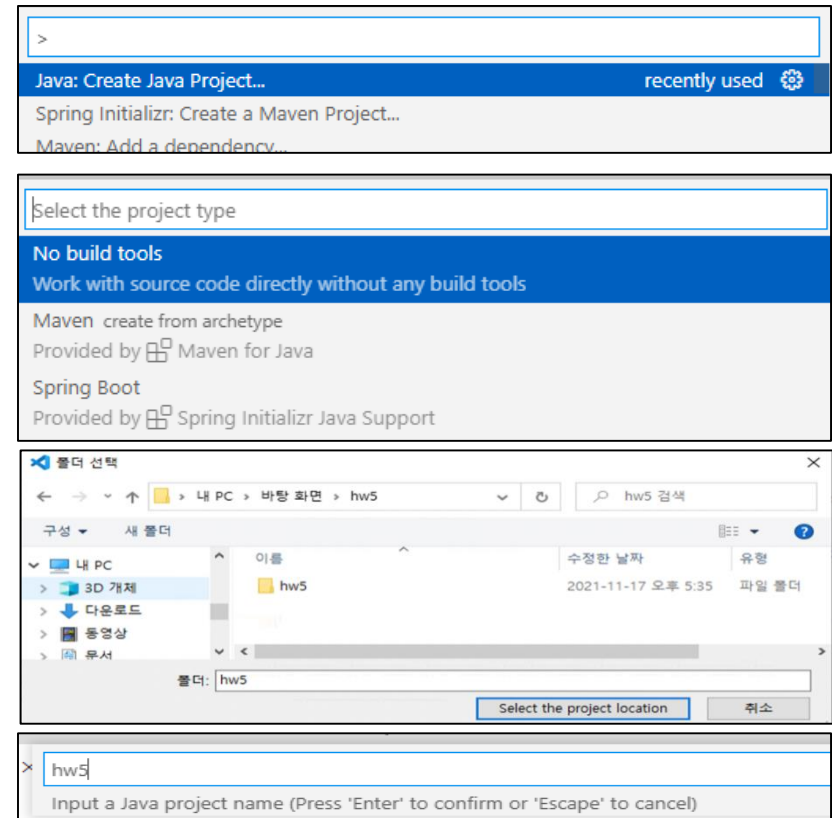
but, **change the source path & dbname**

### 1) Open Server Configuration

: change <Context docBase = [hw6 path]>

### 2) Open the 'Tomcat 8.5(apache-tomcat-8.5.72)/conf/context.xml' file

: change the url in <Resource> tag with new dbname('hw6')



# Homework Assignment #6

---

# Homework Assignment #6

---

- ◆ Make a seminar room reservation pages using JavaScript&Ajax&MySQL
  - Client : web browser(register\_ajax.jsp, reservation\_ajax.jsp, reservation\_list.jsp)
  - Server : Tomcat, executed with jsp.files
  - Data-format : client & server exchange their data with json format
- ◆ The functions of all pages are same as hw5
- ◆ Exclude changes in functional implementation, you can use the same files that you had implemented in hw5
  1. Create a table (registration, reservation)
    - (ex) Register 'room 101' as a new seminar room.
    - After 'room 101' has been registered, users can make a reservation on 'room 101'
  2. Seminar Room registration pages
    - User can see the list of registered rooms
    - User can register & delete seminar rooms

# Homework Assignment #6

---

## 3. Seminar Room reservation pages

- User can see the registered rooms
- User can search the registered rooms that meet users requirements
- User can make a reservation

## 4. Reservation list pages

- User can see the list of all reservations
- User can cancel any reservation



# 1. Create DB/table & Insert values

Make 'hw6' database & Execute the 'hw6.sql' file to make tables & insert values

- DB name should be "hw6"

1-1) '**registered**' : information of the seminar room that registered

- seminar room includes roomname, location, capacity, airconditioner, board
- roomname must be unique

1-2) '**reservation**' : information of the seminar room reservation

- reservation includes roomname, user name, purpose, start/end date, number of people
- username must be unique

<u>Room name</u>	location	capacity	Air conditioner	board
"room1"	"CS-1501"	100	"O"	"X"

<u>room name</u>	<u>User name</u>	Start date	End date	Purpose	#of people
"room 1"	Junseon	2021/11/1	2021/11/1	Group study	7

## 2. Seminar Room registration pages

---

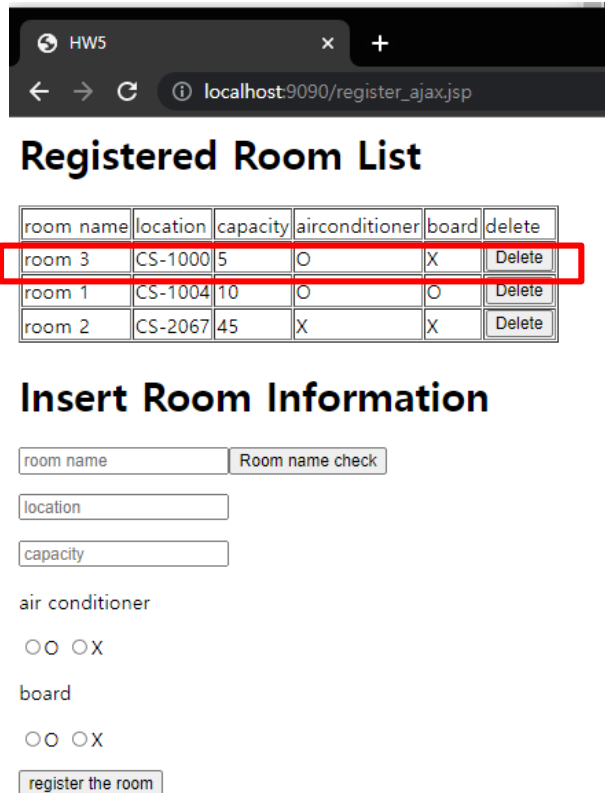
Implement '**register\_ajax.jsp**' that meets the following requirements with **ajax**.  
(you can make more than 1 pages if you need)

2-1) Implement the 'Room name duplicate value check' button, in 'Insert room information form'.

- When user put 'room name' and **click the button**, it should **alert message** whether this room name already exists in database('registered') or not.
- use ***jQuery click*** event

## 2. Seminar Room registration pages

‘register\_ajax.jsp’



**Registered Room List**

room name	location	capacity	airconditioner	board	delete
room 3	CS-1000	5	O	X	Delete
room 1	CS-1004	10	O	O	Delete
room 2	CS-2067	45	X	X	Delete

**Insert Room Information**

room name  Room name check

location

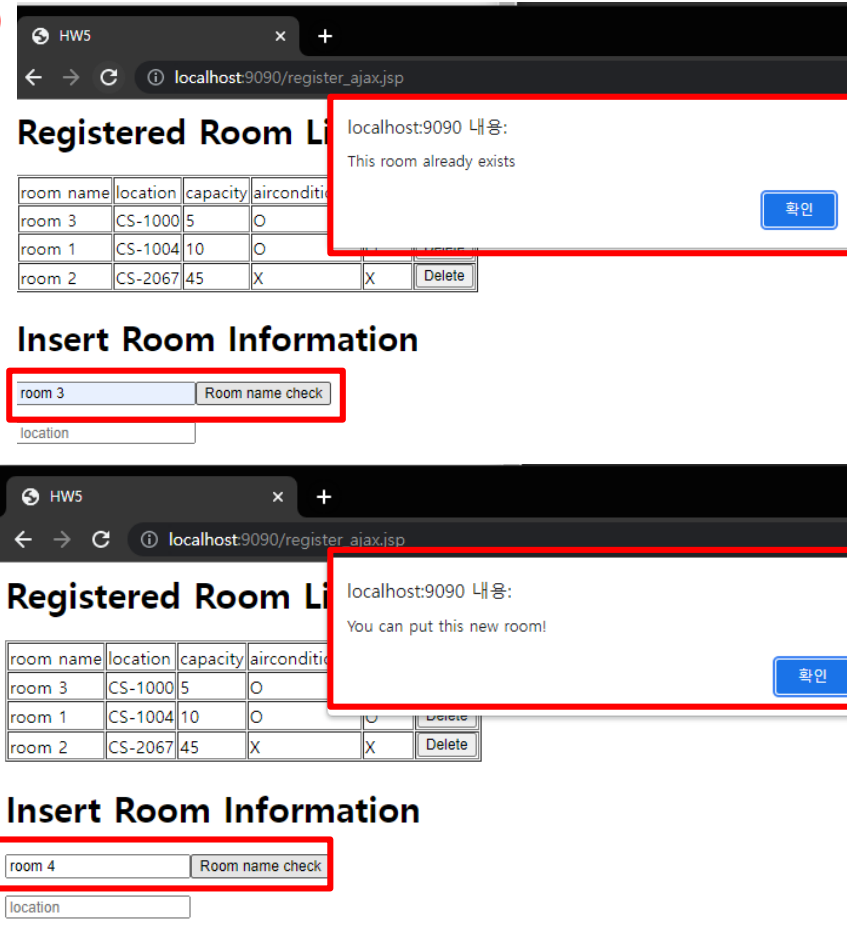
capacity

air conditioner  
☐ ☐ X

board  
☐ ☐ X

‘room 3’ => already exists in ‘registered’,  
‘room 4’ => not exists in ‘registered’

2-1)



**Registered Room List**

localhost:9090 내용:  
This room already exists

room name	location	capacity	airconditioner	board	delete
room 3	CS-1000	5	O	X	Delete
room 1	CS-1004	10	O	O	Delete
room 2	CS-2067	45	X	X	Delete

**Insert Room Information**

room 3

location

**Registered Room List**

localhost:9090 내용:  
You can put this new room!

room name	location	capacity	airconditioner	board	delete
room 3	CS-1000	5	O	X	Delete
room 1	CS-1004	10	O	O	Delete
room 2	CS-2067	45	X	X	Delete

**Insert Room Information**

room 4

location

# 3. Seminar Room reservation pages

Implement '**reservation\_ajax.jsp**' that meets the following requirements with **ajax**.  
(you can make more than 1 pages if you need)

3-1) Implement the '**arrange**' buttons, that arrange the 'registered room list' in database in **ASC** order, depending on the **value(column)** of the selected button.

- Make a three button that each button has roomname, location, capacity as value attribute
- Implement(bind) **only one** *jQuery* click event on three buttons

(*hint!* set each id on each buttons, and select three id together when binding the event!)

## 3-1) Registered Room List

room name	location	capacity	airconditioner	board
room 1	CS-1004	10	O	O
room 2	CS-2067	45	X	X
room 234	EE-123	35	X	O
room 3	CS-1000	5	O	X
room 345	EE-192	10	O	X
room 4	CS-1022	30	X	O
room 5	CS-2012	20	O	X
room 6	CS-1313	5	O	X
room 6	CS-3445	20	O	O
room 7	CS-4556	15	O	X

room name	location	capacity	airconditioner	board
room 3	CS-1000	5	O	X
room 1	CS-1004	10	O	O
room 4	CS-1022	30	X	O
room 6	CS-1313	5	O	X
room 5	CS-2012	20	O	X
room 2	CS-2067	45	X	X
room 6	CS-3445	20	O	O
room 7	CS-4556	15	O	X
room 234	EE-123	35	X	O
room 345	EE-192	10	O	X

room name	location	capacity	airconditioner	board
room 3	CS-1000	5	O	X
room 6	CS-1313	5	O	X
room 1	CS-1004	10	O	O
room 345	EE-192	10	O	X
room 7	CS-4556	15	O	X
room 5	CS-2012	20	O	X
room 6	CS-3445	20	O	O
room 4	CS-1022	30	X	O
room 234	EE-123	35	X	O
room 2	CS-2067	45	X	X

# 3. Seminar Room reservation pages

3-2) Implement the 'search function' that find the 'registered room' that meets user's conditions

- When user put the first condition('capacity') on first input form, system finds the 'registered room' in 'registered' table that meets the first condition and print out the results.
- When user put the second condition('board') on second input form, system prints out the 'registered room list' that meets **both first & second conditions**.
  - Assume that first condition is **not null**, when user input the second condition.
  - If user don't need whiteboard, system find the **both rooms with and without boards**

3-2) Put the conditions you want.

room name	location	capacity	airconditioner	board
-----------	----------	----------	----------------	-------

Condition 1 :

Condition 2 :

Put the conditions you want.

room name	location	capacity	airconditioner	board
room 4	CS-1022	30	X	O
room 5	CS-2012	20	O	X
room 2	CS-2067	45	X	X
room 6	CS-3445	20	O	O
room 7	CS-4556	15	O	X
room 234	EE-123	35	X	O

Condition 1 :

Condition 2 :

Put the conditions you want.

room name	location	capacity	airconditioner	board
room 4	CS-1022	30	X	O
room 6	CS-3445	20	O	O
room 234	EE-123	35	X	O

Condition 1 :

Condition 2 :

# 3. Seminar Room reservation pages

## 3-3) Implement the 'autocomplete' function on 'new reservation input form'

- When user put the part of the location at 'location' input form (ex) 'CS', 'EE' system finds the 'location' that **starts with this string** at database('registered' table) and represent the list of results
- When user click one of the output results, the system **autocomplete** the **location** and **roomname** according to location in 'registered' table.
- Users have to put **minimum 2 length** of the word to autocomplete
- Use **autocomplete** that **jQuery UI** provide
- Use **ajax** when get the result of the database query

### 3-3) make a new reservation

A screenshot of a web form titled 'make a new reservation'. It contains several input fields: 'location', 'room name', 'user name', two date pickers (labeled '연도-월-일'), 'purpose', 'number of the people', and a 'make a reservation!' button.

### make a new reservation

A screenshot of the same form, but the 'location' field now has a dropdown menu open. The dropdown lists several options: 'CS', 'CS-1000', 'CS-1004' (which is highlighted in blue), 'Cs-2011', and 'CS-2067'. A red arrow points to the dropdown.

### make a new reservation

A screenshot of the form where the 'location' field is filled with 'CS-1004' and the 'room name' field is filled with 'room 1'. Both fields are enclosed in a red box.

Autocomplete!

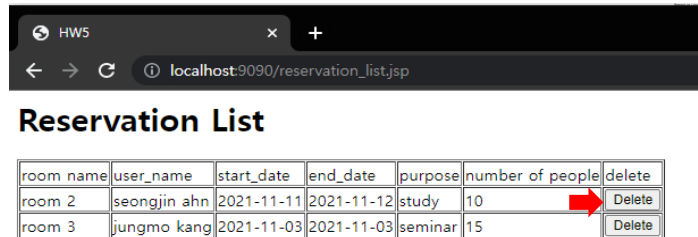
## 4. Reservation list pages

Implement '**reservation\_list.jsp**' that meets the following requirements.  
(you can make more than 1 pages if you need)

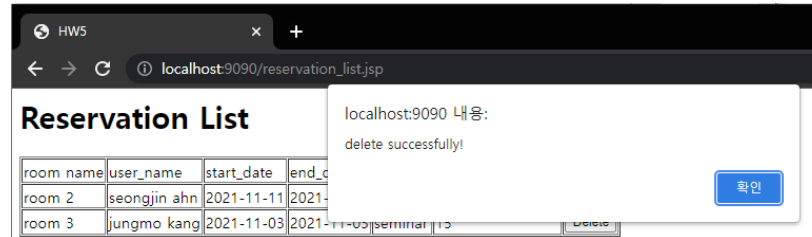
4-1) Change the 'delete' reservation function with ajax

- In hw5, you had implemented 'delete' function with form tag that using action(set the 'URL' that send the form data).
- In hw6, implement '**delete**' **button** that delete the reservation with **ajax**.
- When user **click the 'delete button**, it delete the reservation at database and get the **alert message** (you don't have to redirect this page again)
- Use **jQuery click** event on '**delete**' **button**.  
(**hint!** set same 'class' attribute on button!)

4-1)



room name	user_name	start_date	end_date	purpose	number of people	delete
room 2	seongjin ahn	2021-11-11	2021-11-12	study	10	Delete
room 3	jungmo kang	2021-11-03	2021-11-03	seminar	15	Delete



room name	user_name	start_date	end_date	purpose	number of people	delete
room 2	seongjin ahn	2021-11-11	2021-11-12	study	10	Delete
room 3	jungmo kang	2021-11-03	2021-11-03	seminar	15	Delete

# Tip for your homework!

- ◆ If you use Chrome browser, when you click F12, you can see http request & response

## Registered Room List

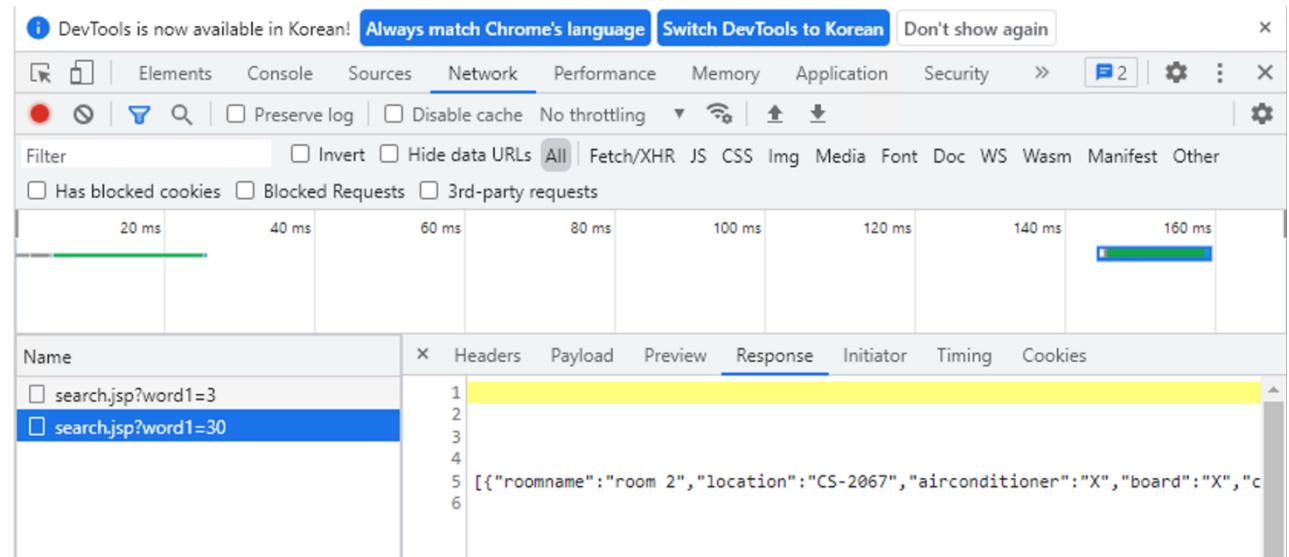
room name	location	capacity		
room name	location	capacity	airconditioner	board

Put the conditions you want.

room name	location	capacity	airconditioner	board
room 2	CS-2067	45	X	X
room 456	EE-123	50	X	O

Condition 1 :

Condition 2 :





# Submission

---

## ◆ Files to submit

- 1. JSP(\*.jsp) files that implement the homework  
(it should include 'register\_ajax.jsp', 'reservation\_ajax.jsp', 'reservation\_list.jsp')
- 2. Submit a .zip file including all the required files

## ◆ How to submit

- KLMS CS360 course pages
- File name should be “**studentNo\_NAME\_HW6.zip**”
- In a single zip file, contain the entire JSP files.

# Submission (cont'd)

---

- ◆ Due date

- December 18(Sat), 11:59 p.m.

- ◆ TA info

- Junseon Kim (email : [junseon.kim@kaist.ac.kr](mailto:junseon.kim@kaist.ac.kr))

- ◆ Notice

- No delay
  - No copy (zero score for each)

# Reference

---

## ◆ HTML

- HTML tutorial :
- <https://www.w3schools.com/html/>

## ◆ jQuery & jQuery UI

- <https://api.jquery.com/>
- <https://api.jqueryui.com/>