IT 1030SEF Introduction to Internet Application Development

Tutorial 2 : JavaScript – The Beginning

Kelsey SHI
s1305223@live.hkmu.edu.hk

Exercise 1-6

1 How Does JavaScript Work?

2 Condition and Variables

3 Condition Statements

PART - 01

How does JavaScript work?

How does JavaScript work?

- JavaScript codes are placed in

```
<script>
[JavaScript codes]
</script>
```

tag at <head> section, but not limited to.

- Alternatively, JavaScript can be saved separately by saving the codes in a ".js" file. Then call below code :

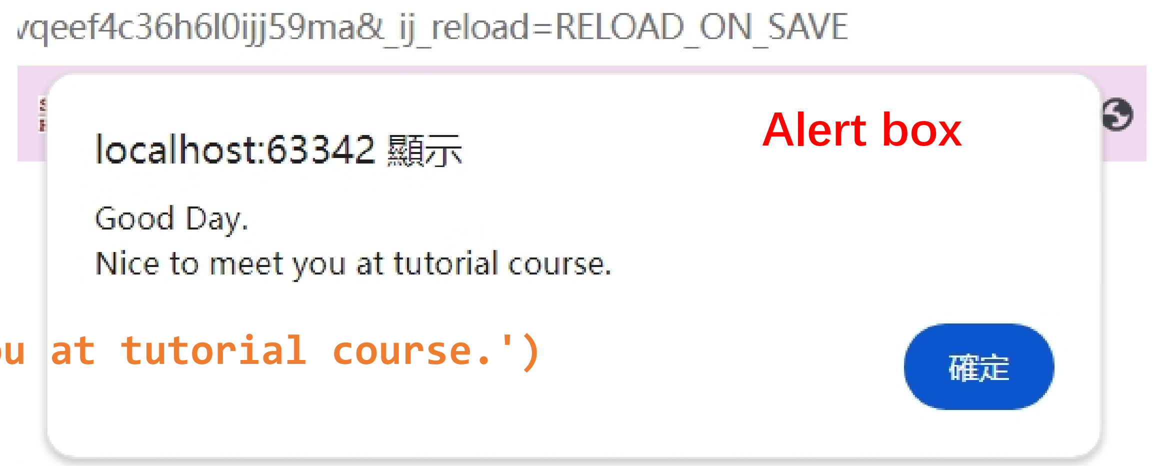
```
<script src="jsfile1.js"></script>
```

where "jsfile1" is the name of the file to import JavaScript codes to HTML statements.

Short notes about the usage of break line command (\n and <br>)

- How can we differentiate the usage of \n and <br> in JavaScript and HTML.
- \n is a special command in JavaScript program, generally control the break line for the components that handles at backend.
  - E.g. window.alert('Good Day.\nNice to meet you at tutorial course.');
  - The statement above is going to display two sentences but those are not print to the HTML page. We use "\n" to break line.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<script>
  window.alert('Good Day.\nNice to meet you at tutorial course.')
</script>
</body>
</html>
```


vqeef4c36h6l0ijj59ma&_ij_reload=RELOAD_ON_SAVE

localhost:63342 顯示

Good Day.
Nice to meet you at tutorial course.

Short notes about the usage of break line command (\n and <br>)

- Once more about using **\n**, when we are **accessing the contents of HTML components**, like **`textarea`** (we will have a practice later on this) since we are handling the HTML component, not the HTML page, we also use **\n** for breaking lines.

- Reversely, **<br>** is HTML syntax. When we want to apply break line in displaying contents to the HTML page (front-end), we should apply HTML syntax instead, especially on **`document.write()`** function, when "document" means the current HTML page.

- So, we should apply **<br>** to perform break line (of course, the JavaScript is writing command) in HTML.

Short notes about write() and writeln()

The **write()** and **writeln()** functions are so similar:

- **write()** is print out contents.
- **writeln()** is print out contents then **open a newline** after the statement.

Why **writeln()** function does **not** happened?

- Partially supported
- We need to apply **<pre></pre>** HTML syntax to use this command.
- **Generally, we apply write() and <br> or <p> to apply break line format.**

Short notes about **write()** and **writeln()**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>titlexxx</title>
</head>
<body>
    <script>
    document.writeln("Hello,\nWorld!");
    document.writeln("<pre>Hello,\nWorld!</pre>");
    document.write("Hello,<br>World!");
    </script >
</body>
</html>
```

PART - 02

Condition and Variables

Condition commands

> - Greater than

< - Less than

>= - Greater than or equal to

<= - Less than or equal to

== - equal to : Values are the same

```
1 == '1'(true) // '1' is string
```

=== - equivalent : Not just the values, the data type must be the same

```
1 === '1'(false)
```

! – Not : Combine with the commands above to have different functions

```
!(1 == '1')(false)
```

Examples

document.write(("4" == (2*2))); - true

document.write((4 == (2*2))); - true

document.write(("4" == 4)); - true

document.write(("4" <= 4)); - true : the equal sign

document.write(("4" >= 5)); - false

document.write((4 < 4)); - false : 4 == 4 but 4 is not less than 4

document.write((4 > (2 + 2))); - false : 4 is not greater than 2+2 (4)

document.write(("4" === (2*2))); - false : data type is different

document.write(!("4" === (2*2))); - true : not false → true

Exercise 1

Determine whether the following comparisons evaluate to true or false, and print out the result use `document.write()`

- Task 1 – 1 : (4 <= 3)
- Task 1 – 2 : (16 != (4 * 4))
- Task 1 – 3 : (16 != 8)
- Task 1 – 4 : (15 < (3 * 5))
- Task 1 – 5 : (16 == (2 * 8))

Variables in JavaScript

JavaScript can **declare variables**. It will be stored at a place of memory with the “name” assigned. We call the “name” as **identifier**.

Allowed format of **identifier** are as below:

1. Names can contain letters(abcd), digits(1234), underscores (_), and dollar ($) signs.
2. Names must begin with a letter (a – z, A – Z), dollar sign “$” or underscores “_”
3. Names are case sensitive (b and B are different variables)
4. Reserved words (keywords) cannot be used as names (e.g. for, while, switch, if, else, etc.)

Variables in JavaScript

- The status of each variable will be different according to the implementation.

Example:

```
var  num_1;//the variable has been declared with null value. It will determine as “undefined”
document.write(num_1); // “undefined” will be displayed

num_1 = 3; // “3” is assigned to the num_1
document.write(‘<br>’+num_1); // “3” will be displayed
```

Exercise 2

- Suppose firstNumber and secondNumber have been declared as variables. What is the output of each of the following write statements?

```
var firstNumber, secondNumber;
firstNumber =10;
document.write(secondNumber); // Task 2 - 1
firstNumber = firstNumber + 20;
document.write(firstNumber); //  Task 2 - 2
secondNumber = firstNumber  -  15
document.write(firstNumber  -  secondNumber); // Task 2 - 3
secondNumber = firstNumber;
document.write(firstNumber  +  secondNumber); // Task 2 - 4
```

Execution of a JavaScript program (codes)

In executing a JavaScript program, it also execute from the top to the bottom, each code will be **read** **from the left to the right** (especially on condition statements) , then **execute** **from the right to the left** (especially on assignment statements)

- Example:

```
var aNumber = 2;
```

←

The above statement will be read as : The value 2 is assigned to a new variable named with "aNumber".

Exercise 3

Task 3 - 1 : Declare a variable called *myNumber.*

Task 3 - 2 : Initialize *myNumber* with the number 5.

Task 3 - 3 : Perform *myNumber* minus 2 then **store the difference** back to *myNumber.*

Task 3 - 4 : Display the new value in *myNumber*.

Exercise 4 : Build up a JavaScript program that fulfill the below requirements.

Task 4 - 1 : Create a statement that **collect a name (string)** from the user by using a **dialog box**.

Use a variable **nameVar** to store the content collected.

```
window.prompt("xx", "xx");
```

Task 4 - 2 : Create another statement that **collect the weekday** from the user by using a **new dialog box**. Use a variable **weekdayVar** to store the content collected.

```
window.prompt("xx", "xx");
```

Task 4 - 3 : Write ONE statement to print out the sentence (**use alert dialog**) collected from Task 4 - 1 and Task 4 – 2 on the HTML page:

```
window.alert("xxxx");
```

Output : Hello *<u>YOURNAME</u>*! Have a nice *<u>Monday</u>*~!

Where "*<u>YOURNAME</u>*" is the content collected from Task 4 - 1,

and "*<u>Monday</u>*" is the content collected from Task 4 - 2.

Casting

In JavaScript, there is similar to Python that all value entered by user are declared as “string” data type, even that is a number.

If we need to handle the value entered by user, we need casting JavaScript provided a function to **convert the data type from string to numeric** by using:

- **`parseFloat()`** – ***Convert string type numeric to floating points***
- **`parseInt()`** – ***Convert string type numeric to integer***

Suggestion : Better use **`parseFloat()`** since it also contains the function of parseInt().

IndexOf() and charAt()

indexOf() and **charAt()** are two functions mostly in use in string data type variables.

□indexOf()

```
“string”.indexOf("phrase", position);
```

- "phrase" means the first occurrence of the letter or word phrase within the string to be searched.
- "position" is the starting index to search for the phrase, this can be omitted then JavaScript will start to search the phrase from the beginning of the string.
- It will result the beginning index of the phrase searched.

Example:

```
                  start from 0
var result = “hkmu".indexOf("m");
document.write(result); // 2
```

IndexOf() and charAt()

charAt(index)

This command is searching the content of the string place at the position with the index.

Example:

start from 0

```
var result = "handsome".charAt(3);
document.write(result); // d
```

Exercise 5 : Consider the following piece of the JavaScript statements

```
    var firstNumber = “123”;
    var secondNumber  = “321”;
    thirdNumber = firstNumber + secondNumber;
    document.write(“The sum of firstNumber and secondNumber is ” + 
thirdNumber);
```

Task 5 - 1 : Which of the folowings has been displayed on the HTML page?

Answer: The sum of firstNumber and secondNumber is 123321

Task 5 - 2 : There are something strange in the output. What JavaScript syntax we should use to perform the correct output for Task 5-1?

Answer: parseFloat()

PART - 03

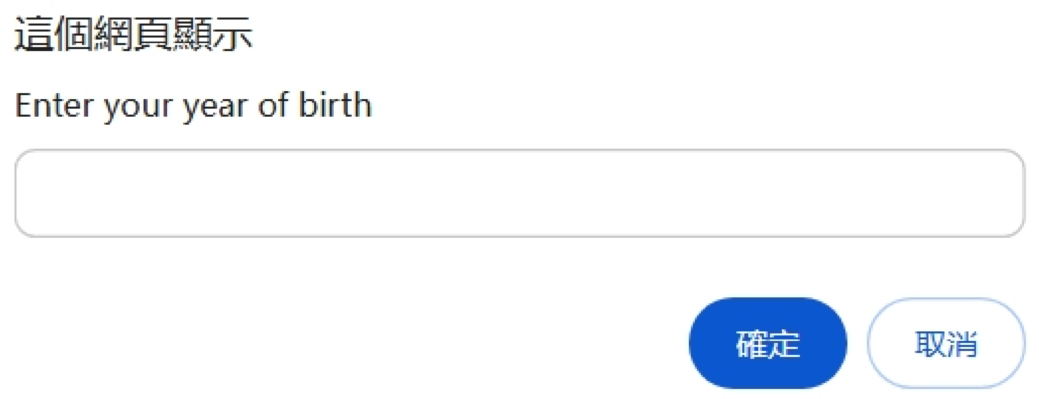
Condition Statements

Brief revision about condition statements

- We can further determine more then one conditions by using **if - else if – else** blocks.

```
if (condition 1){
statements when condition 1 is true
}
else if (condition 2){
statements when condition 2 is true
} (We can expand this else-if block for more conditions.)
else{
statements when all conditions are false
}
```

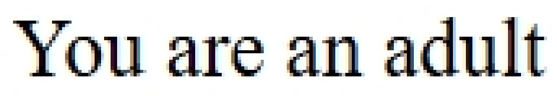
Exercise 6: Determining if an adult


這個網頁顯示

Enter your year of birth

Task 6:

1. Create a **year variable** and **prompt**.
2. Get the year of birth from **User side**.
3. Use this year (2026) minus the year variable to determine if the **age** is greater than or equal to 18.
4. Output the result.


You are an adult

You can visit w3school to study more about HTML and CSS.

There are lots of examples and practices for you to get familiar in HTML and CSS.

https://www.w3schools.com
for all students (English with translation)

https://www.w3school.com.cn
for mainland students (Simplified Chinese version)

See you next week

THANK YOU

Kelsey SHI