CS2204 Fundamentals of Internet Applications Development

Lecture 7 JavaScript – Part 2

Computer Science, City University of Hong Kong

Semester B 2024-25

Announcement: Midterm

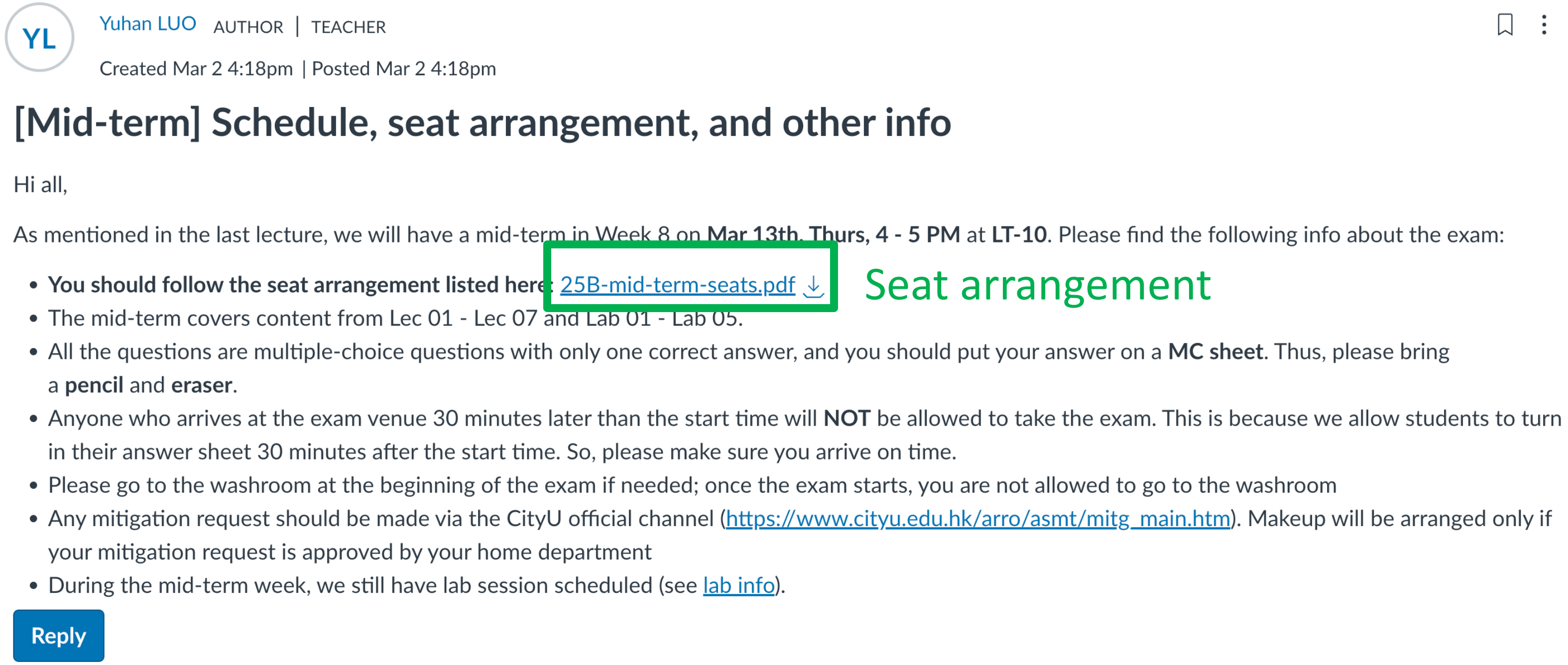
Mid-term is scheduled in one week, on **March 13th**, **Thursday** from **4:00 PM to 5:00 PM** (60 minutes)

Venue: LT-10
Coverage: Lec01 – Lec07, Lab01– Lab05

Form: **40** ~~45~~ multiple choice questions (each question has only one current answer); close book; no calculator allowed

There are two versions of exam papers. In Q41, you should choose the versions of your exam paper based on the color of the papers.

Announcement: Midterm



YL Yuhan LUO AUTHOR | TEACHER

Created Mar 2 4:18pm | Posted Mar 2 4:18pm

[Mid-term] Schedule, seat arrangement, and other info

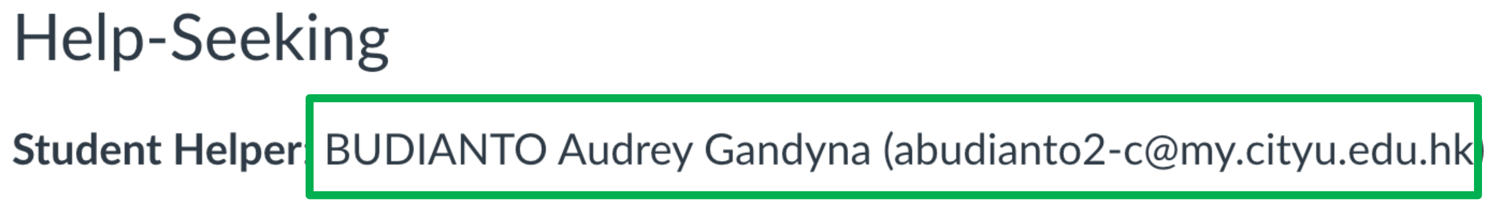
Hi all,

As mentioned in the last lecture, we will have a mid-term in Week 8 on **Mar 13th, Thurs, 4 - 5 PM** at **LT-10**. Please find the following info about the exam:

- **You should follow the seat arrangement listed here**: 25B-mid-term-seats.pdf
- The mid-term covers content from Lec 01 - Lec 07 and Lab 01 - Lab 05.
- All the questions are multiple-choice questions with only one correct answer, and you should put your answer on a **MC sheet**. Thus, please bring a **pencil** and **eraser**.
- Anyone who arrives at the exam venue 30 minutes later than the start time will **NOT** be allowed to take the exam. This is because we allow students to turn in their answer sheet 30 minutes after the start time. So, please make sure you arrive on time.
- Please go to the washroom at the beginning of the exam if needed; once the exam starts, you are not allowed to go to the washroom
- Any mitigation request should be made via the CityU official channel (https://www.cityu.edu.hk/arro/asmt/mitg_main.htm). Makeup will be arranged only if your mitigation request is approved by your home department
- During the mid-term week, we still have lab session scheduled (see lab info).

About bonus points

For questions related to bonus points, please contact the student helper first



Help-Seeking

Student Helper: BUDIANTO Audrey Gandyna (abudianto2-c@my.cityu.edu.hk)

Post-lab Quiz 3 Review

To provide multi-browser support, which of the followings are correct? Enter "true" or "false" for each of the following statements. Keep your answers in lower case.

a. Provide multiple source formats one by one.
b. Provide a fallback code with the attribute 'onerror'.
c. Add a static cover image for the video.
d. Extend audio description.

Post-lab Quiz 3 Review

In an HTML form (i.e., enclosed by <form> ... </form>), create three input elements by filling out the blank space below. When you need to use quotation marks, please use double-quotation mark " rather than single quotation mark '. Also, do not use extra blank space when it is not necessary.

1) Add a radio button with name "choice1", id "radio1", value "Agree", and checked by default

Most of the students lost the marks due to typos (e.g., “Argee”) or missing quotation marks when assigning id, name, etc

If you believe you did it correctly but did not receive the marks, please contact the student helper BUDIANTO Audrey to manually check the answers for you

Question Time: CSS3

1. How to support special CSS3 effects, such as gradient color, across different browsers?

Question Time: CSS3

3. What’s the difference between transform and transition in CSS?

Question Time: CSS3

5. Compared with basic transition, what are the advantages of animation?

Question Time: JavaScript

7. JavaScript is a programming language that provide instructions for a browser to _______ and _________

Agenda

JavaScript Basic Logics

- Expression
- Statement
- Operation

JavaScript Control Flow

- Conditional
- Loop

Mid-term brief review

Agenda

JavaScript Basics

- Expression
- Statement
- Operation

JavaScript Control Flow

- Conditional
- Loop

Mid-term brief review

JavaScript Expressions

An ***expression*** is a combination of constants, variables, and function calls that **evaluate to a result**

Examples:

4 is a constant

x is a variable

sqrt() is function call, x*3.0 is the parameter we pass into the function

```
x = 3.0*4.0;

y = 2.0 + x;

z = 5.0 + x/y - sqrt(x*3.0);
```

JavaScript Statement (1)

Each instruction is a JavaScript **statement**

- each statement ends with a semicolon ; For example,

```
var x;
sqrt(3);
```

- a single statement may span multiple lines
- multiple statements may occur on a single line if each statement is separated by a semicolon (;) - NOT a good practice though

Critical thinking: What’s the difference between statement and expression?

JavaScript Statement (2)

Differing from regular program consisting of all instructions as a whole, JavaScript:

- may spread out as different **fragments**
- each fragment enclosed by `<script> ... </script>` in a HTML file or in a separate file (depending on whether embedded, inline and/or external scripts are used)

JavaScript Operators

An operator specifies an operation to be performed on some values , which are called the **operands** of the operator

e.g., in `x = a + b;` `a` and `b` are operands; `+` and `=` are the operators

Common JavaScript **Operators**

- Assignment Operator (i.e., `=`)
- Arithmetic Operators (e.g., `+, -, *, /`)
- Comparison Operators (e.g., `>, <, ==, !=`)
- Logical Operators (e.g., `&&, ||, !`)
- String Operator (e.g., `+`)
- Conditional Operator (e.g., `?:`)

Assignment Operator =

An assignment operator assigns a value to its **left operand** based on the value of its right operand. Generic form is

```
variable = expression;
```

= is an assignment operator that is different from the **mathematical equality** (which is **==** in JavaScript)

```
x + 10 = y;

2=x;

var a, b, c;
a = (b = 2) + (c = 3);

var a, b, c;
b = 2;
c = 3;
a = b + c;
```

Critical thinking:
Are the following expressions with an assignment operator valid?

Arithmetic Operators

Four basic operators: `+, -, *, /`

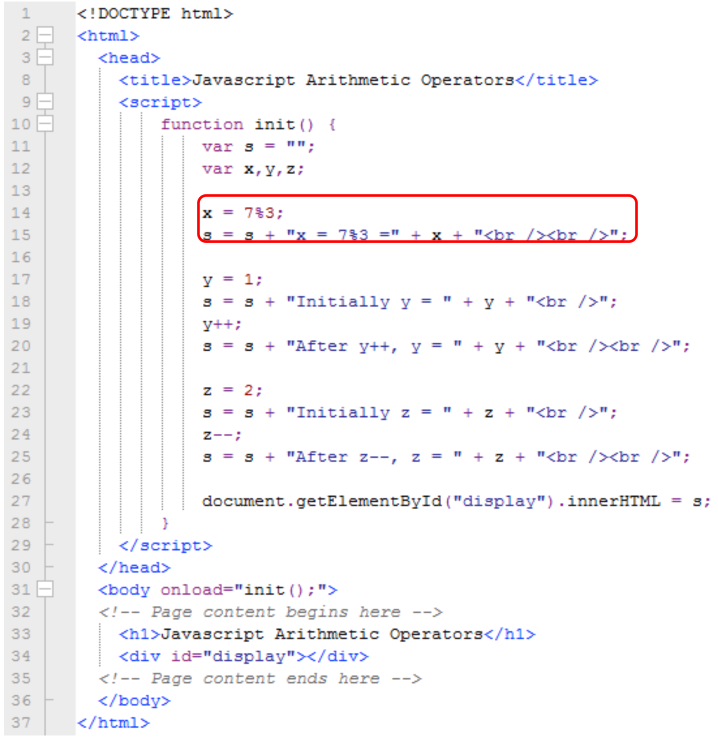
Modulus operator: `%`, returns the division remainder

```
 x =  5 % 2; /* x is 1 */
```

Increment operator: `++`, e.g., `x++` will increase the value of the variable `x` by `1`

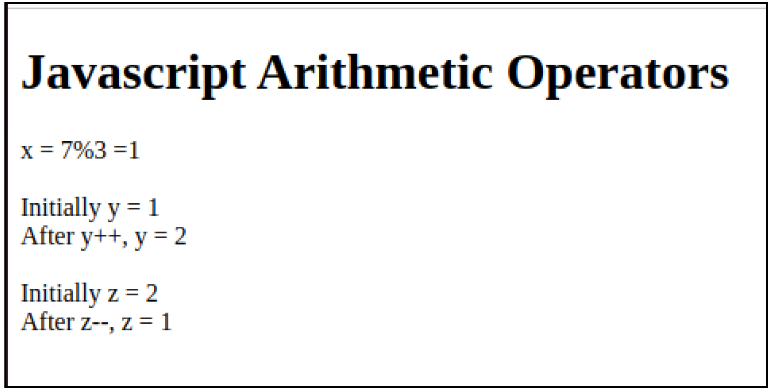
Decrement operator: `--` , e.g., `x--` will decrease the value of the variable x by `1`

Arithmetic Operators: example



```
<!DOCTYPE html>
<html>
  <head>
    <title>Javascript Arithmetic Operators</title>
    <script>
        function init() {
            var s = "";
            var x,y,z;

            x = 7%3;
            s = s + "x = 7%3 =" + x + "<br /><br />";

            y = 1;
            s = s + "Initially y = " + y + "<br />";
            y++;
            s = s + "After y++, y = " + y + "<br /><br />";

            z = 2;
            s = s + "Initially z = " + z + "<br />";
            z--;
            s = s + "After z--, z = " + z + "<br /><br />";

            document.getElementById("display").innerHTML = s;
        }
    </script>
  </head>
  <body onload="init();">
  <!-- Page content begins here -->
    <h1>Javascript Arithmetic Operators</h1>
    <div id="display"></div>
  <!-- Page content ends here -->
  </body>
</html>
```

Code Example: lec07-01-JS-arithmetic-operators.html



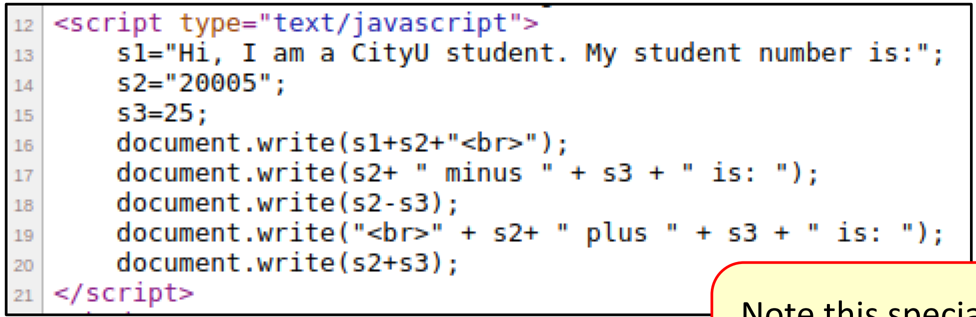
Javascript Arithmetic Operators

```
x = 7%3 =1

Initially y = 1
After y++, y = 2

Initially z = 2
After z--, z = 1
```

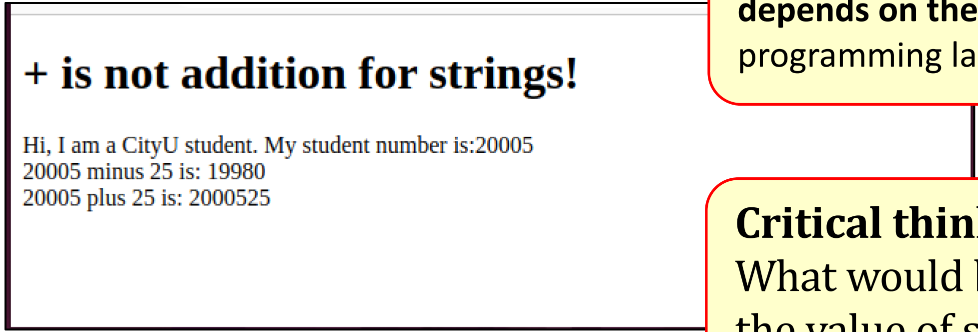
7%3 is the remainder when 7 is divided by 3 so it is equal to 1 because $7 = 2 \times 3 + 1$

Arithmetic Operators on strings

When "+" meets string



```
<script type="text/javascript">
    s1="Hi, I am a CityU student. My student number is:";
    s2="20005";
    s3=25;
    document.write(s1+s2+"<br>");
    document.write(s2+ " minus " + s3 + " is: ");
    document.write(s2-s3);
    document.write("<br>" + s2+ " plus " + s3 + " is: ");
    document.write(s2+s3);
</script>
```



+ is not addition for strings!

Hi, I am a CityU student. My student number is:20005
20005 minus 25 is: 19980
20005 plus 25 is: 2000525

Note this special case - **meaning of operator depends on the operands**. Most other programming languages don't have this behavior

Critical thinking
What would be the output if we change the value of s2 to "a"?

Code Example: lec07-02-JS-string-operator.html

JavaScript: `NaN` (1)

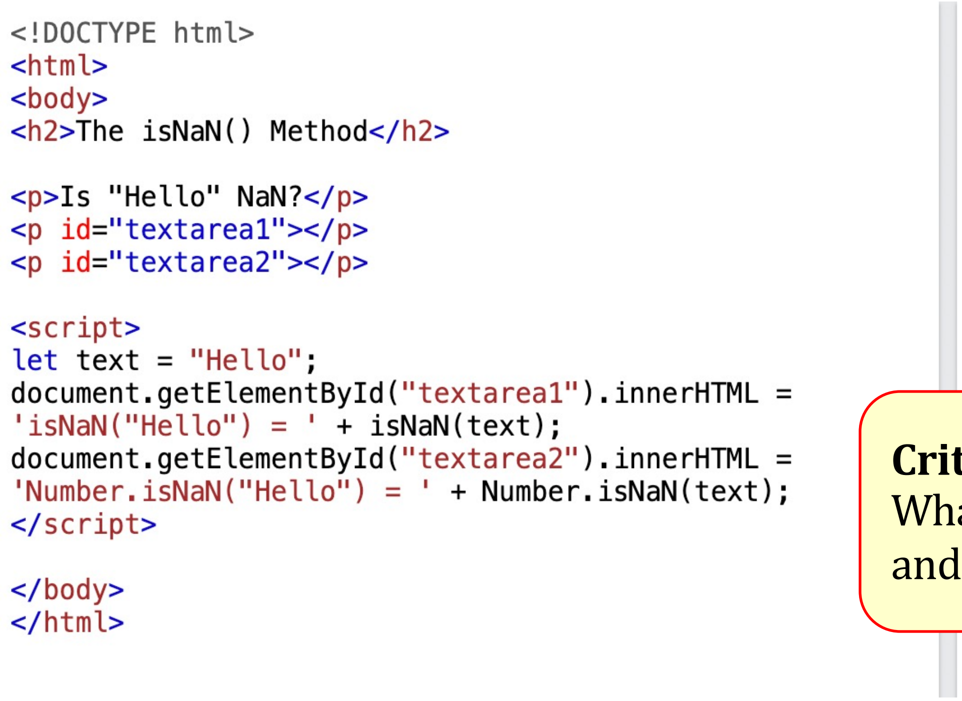
Short for "Not-a-Number"

`isNaN()` method returns true if **a value** is Not-a-Number.

`Number.isNaN()` method returns true *if the value is `NaN`* **AND** *the type is Number* (it means that the type of value `NaN` is a number)

Critical thinking
Then what's the difference between `isNaN()` and `Number.isNaN()`?

JavaScript: `NaN` (2)



```html
<!DOCTYPE html>
<html>
<body>
<h2>The isNaN() Method</h2>

<p>Is "Hello" NaN?</p>
<p id="textarea1"></p>
<p id="textarea2"></p>

<script>
let text = "Hello";
document.getElementById("textarea1").innerHTML =
'isNaN("Hello") = ' + isNaN(text);
document.getElementById("textarea2").innerHTML =
'Number.isNaN("Hello") = ' + Number.isNaN(text);
</script>

</body>
</html>
```

Critical thinking
What's the output in `textarea1` and `textarea2` respectively?

Code Example: lec07-03-JS-nan.html

JavaScript: `NaN` (3)

```
console.log(Number.isNaN(NaN));

console.log(Number.isNaN(Number.NaN));

console.log(Number.isNaN(0 / 0));

console.log(Number.isNaN(10 / "abc"));

console.log(Number.isNaN("Hello"));

console.log(Number.isNaN(42));

console.log(Number.isNaN(true));

console.log(Number.isNaN(null));

console.log(Number.isNaN(undefined));

console.log(Number.isNaN({}));
```

In JS, when **arithmetic operations** are performed, **the operands are coerced to numbers** if they aren't already.

Code Example: lec07-04-JS-more-nan.html

JavaScript `try` and `catch`

`try` statement: a block of code to be tested for errors while being executed

`catch` statement: a block of code to be executed, if an error occurs in the try block.

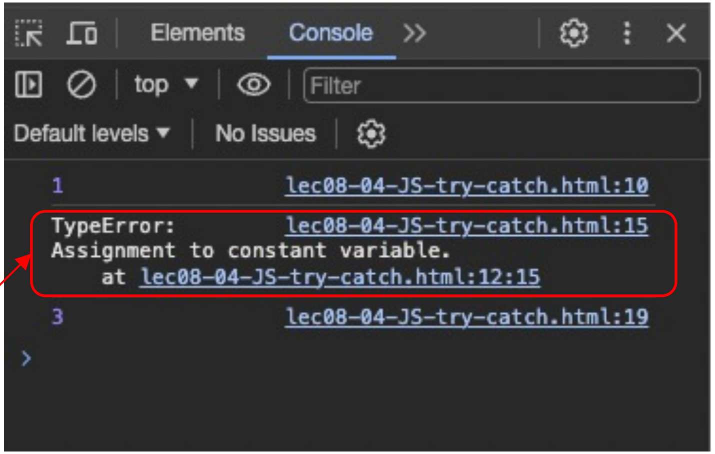
```
try {
  Block of code to try
}
catch(err) {
  Block of code to handle errors
}
```

The exception (`e`) is caught by the catch statement and an error message can be accessed by `e.message`

JavaScript `try` and `catch` example

`try` and `catch` statement is often used to prevent vulnerable statements from blocking the entire code

```
<script>
        const x = 1;
        console.log(x);
        try{
            x = 2;
            console.log(x);
        }catch(e){
            console.log(e);
        }

        y = 3;
        console.log(y);
 </script>
```



```
1                              lec08-04-JS-try-catch.html:10
TypeError:                     lec08-04-JS-try-catch.html:15
Assignment to constant variable.
    at lec08-04-JS-try-catch.html:12:15
3                              lec08-04-JS-try-catch.html:19
>
```

What if we remove the `try {…}` and `catch (e) {…}` statement here?

Code Example: lec07-05-JS-nan.html

Efficient Arithmetic Operators & Assignments

Operator	Example	Is the Same As
`=`	`x = y`	`x = y`
`+=`	`x += y`	`x = x + y`
`-=`	`x -= y`	`x = x - y`
`*=`	`x *= y`	`x = x * y`
`/=`	`x /= y`	`x = x / y`
`%=`	`x %= y`	`x = x % y`

Advantages

- Conciseness: reduce the code length
- Efficiency/in-place modification: optimize code execution without creating new variables

Increment/decrement Operators

- **Post-**increment and post-decrement: `k++` and `k--`
- **Pre-**increment and pre-decrement : `++k` and `--k`

```
var k=0, j, z;
j=k++;
z=++k;
```

Critical thinking
What are the results of **`k,j`** and **`z`** after the execution of the above code?

Code Example: lec07-06-JS-increment.html

Comparison and Logical Operators (1)

Comparison operators accept **two** operands and compare them

The result is a Boolean value, i.e., `true` or `false`

Relational operators	Syntax	Example
Less than	`<`	`x<y`
Greater than	`>`	`z>1`
Less than or equal to	`<=`	`b<=1`
Greater than or equal to	`>=`	`c>=2`

Equality operators	Syntax	Example
Equal to	`==`	`a==b`
Not equal to	`!=`	`b!=3`

Comparison and Logical Operators (2)

Logical operators are used for combining **Boolean** (or **logical) values** to create **new logical values**

Logical AND (`&&`)

- return **true** if **both** operands are **true**, false otherwise (e.g., **a>1**&&**b<1**)

Logical OR (`||`)

- return **false** if **both** operands are **false**, true otherwise

Logical NOT (`!`)

- invert the Boolean value of the operand

x	**y**	**x&&y**
true	true	true
true	false	false
false	true	false
false	false	false

x	**y**	**x||y**
true	true	true
true	false	true
false	true	true
false	false	false

x	**!x**
true	false
false	true

Comparison and Logical Operators (3)

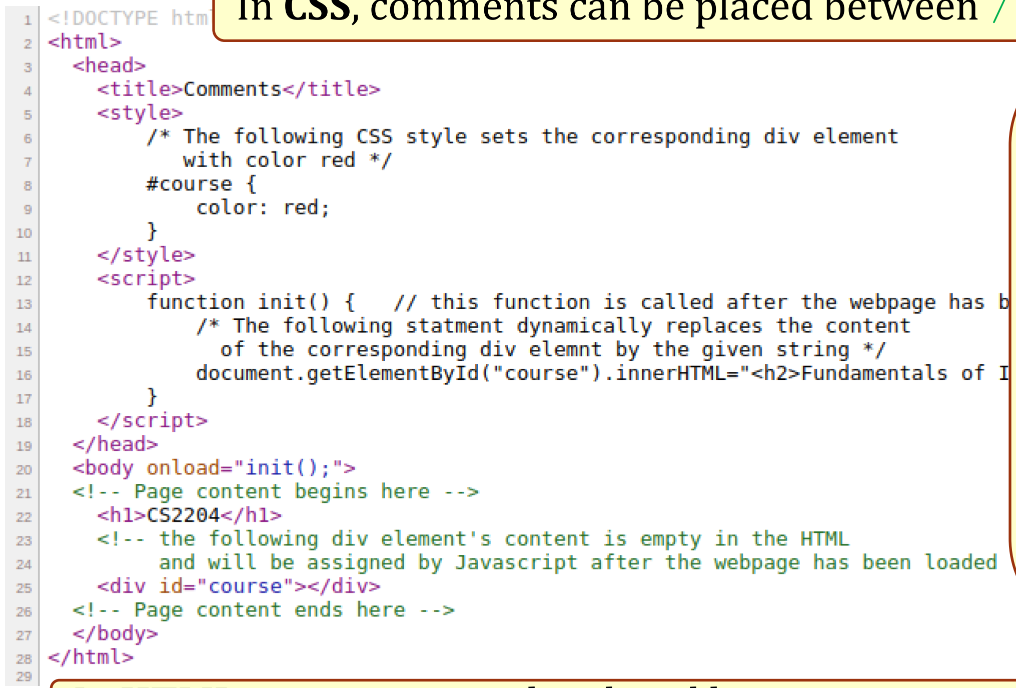
Logical expressions can be `true` or `false` only

In JavaScript, if the value of an expression is one of the followings, this value can be treated as `false`; otherwise, the value is treated as `true`

- `false`
- `0`
- `“”`
- `undefined`
- `NaN`
- `null`

JavaScript Comments

In **CSS**, comments can be placed between /* … */ and can **span multiple lines**



```
<!DOCTYPE htm
<html>
  <head>
    <title>Comments</title>
    <style>
        /* The following CSS style sets the corresponding div element
           with color red */
        #course {
            color: red;
        }
    </style>
    <script>
        function init() {   // this function is called after the webpage has b
            /* The following statment dynamically replaces the content
              of the corresponding div elemnt by the given string */
            document.getElementById("course").innerHTML="<h2>Fundamentals of I
        }
    </script>
  </head>
  <body onload="init();">
  <!-- Page content begins here -->
    <h1>CS2204</h1>
    <!-- the following div element's content is empty in the HTML
         and will be assigned by Javascript after the webpage has been loaded
    <div id="course"></div>
  <!-- Page content ends here -->
  </body>
</html>
```

In **JavaScript**, there are 2 ways to add comments:

1)comments can be placed between /* … */ and can **span multiple lines**

2) placed after // until the end of line so this **is a single line** comment. Note that **CSS does not support this single line** comment style

In **HTML**, comments can be placed between <!-- … --> and can **span multiple lines**

Code Example: lec07-07-comment.html

Agenda

JavaScript Basics

- Expression
- Statement
- Operation

JavaScript Control Flow

- Conditional
- Loop

Mid-term brief review

Flow Control Statements

Common Flow Control Statements

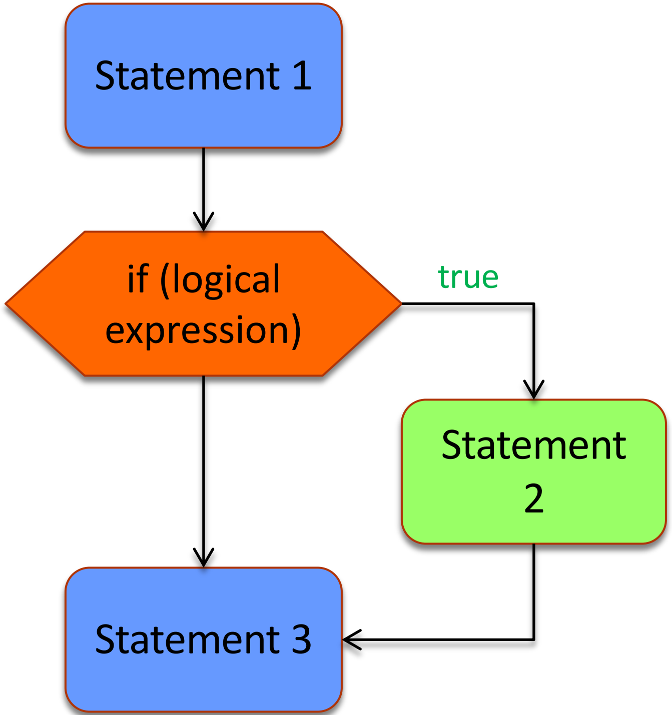
- **if-else** statement
- **switch** statement
- Loops
  - **for** statement
  - **while** statement
  - **do-while** statement
- **break** statement
- **continue** statement
- **return** statement
- **block** statement

if-statement: One-Way Conditional (1)

Execute a statement (or a block of statements) if a specified condition is true

```
statement1;
if (condition)
    statement2;
statement3;
```

```
statement1;
if (condition){
    statement2;
    statement22;
    …
}
statement3;
```



if-statement: One-Way Conditional (2)

Execute a statement (or a block of statements) if a specified condition is `true`

The variable `s` is initialized to be `"This is the end of summer"`

The variable `input` is assigned as a string (text) from the output of the **prompt function**. The expression `Number(input)` converts the string `input` to its numerical value so that it can be manipulated as a number

```
<script>
    function init() {
      var s = "This is the end of summer";
      var input = prompt("Enter a month");
      var month = Number(input);

      if(month>1 && month<=5) {
        s = "This month is in Semester B\n";
      }
      if(month ==12 || month ==1 ){
        s = "This is a winter month.\n";
      }
      if(month>=9) {
        s = "This month is in Semester A.\n";
      }
      if(month!=8) {
        s = "This month is the middle of summer.\n";
      }

      alert(s);
    }
</script>
```

The expression (month>1 && month <=5) is true if month > 1 AND month <=5, i.e., when month is equal to 2, 3, 4, 5

The expression (month==12 || month ==1) is true if month is equal to 12 OR 1, i.e., when month is equal to 12, 1

The expression (month>=9) **will be executed** if month is 9, 10, 11

The expression (month!=8) **will be executed** if month is 6, 7

Code Example: lec07-08-JS-if.html

if-else: Two-Way Conditional (1)

Execute a statement (or a block of statements) if a specified condition is true. Otherwise, another statement (or a block of statements) will be executed

```
if (condition)
     statement1;
else
     statement2;
```

```
if (condition){
     statement1;
     statement2;
     …
} else {
     statement3;
     statement4;
     …
}
```

if-else: Two-Way Conditional (2)

Execute a statement (or a block of statements) if a specified condition is `true`. Otherwise, another statement (or a block of statements) will be executed

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Javascript Two-Way Conditional</title>
    <script>
      function init() {
            var p, s;
            s = "";
            p = prompt("Enter a positive integer: ");
            if(Number(p)>0) {
              s = p +" is a positive integer\n";
            } else {
              s = p + " is NOT a positive integer\n";
            }
            s = s + "Two-way conditional example.";
            alert(s);

        }
    </script>
  </head>
  <body onload="init();">
    <!-- Page content begins here -->
    <h1>Two-Way Conditional</h1>
    <!-- Page content ends here -->
  </body>
</html>
```

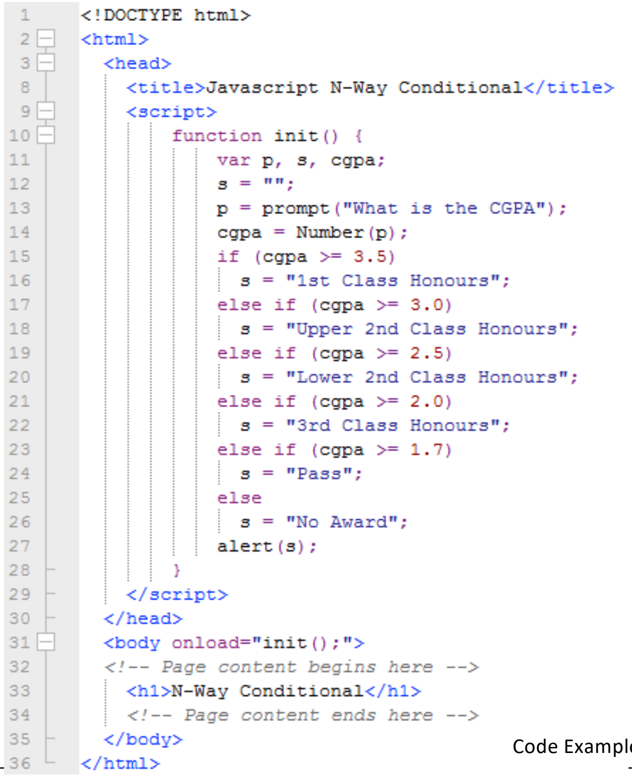
Code Example: lec07-09-JS-if-else.html

```
if (condition){
     statement1;
     statement2;
     …
} else {
     statement3;
     statement4;
     …
}
```

`\n` specifies that a line break should be added such that the subsequent text will be displayed in a new line when shown by the `alert()` function

Multiple else-if (N-Way Conditional)

You can have as many nested “else if” statements as you want.



```
<!DOCTYPE html>
<html>
  <head>
    <title>Javascript N-Way Conditional</title>
    <script>
        function init() {
            var p, s, cgpa;
            s = "";
            p = prompt("What is the CGPA");
            cgpa = Number(p);
            if (cgpa >= 3.5)
              s = "1st Class Honours";
            else if (cgpa >= 3.0)
              s = "Upper 2nd Class Honours";
            else if (cgpa >= 2.5)
              s = "Lower 2nd Class Honours";
            else if (cgpa >= 2.0)
              s = "3rd Class Honours";
            else if (cgpa >= 1.7)
              s = "Pass";
            else
              s = "No Award";
            alert(s);
        }
    </script>
  </head>
  <body onload="init();">
  <!-- Page content begins here -->
    <h1>N-Way Conditional</h1>
    <!-- Page content ends here -->
  </body>
</html>
```

```
if (logical expression 1)
          //action for true
          statement a;
else if (logical expression 2)
          //action for true
          statement b;
else if (logical expression 3)
          //action for true
          statement c;
… …

else
          //action for false
          statement;
```

CGPA	Boolean Expression	Award Classification
3.5 or above	CGPA>=3.5	1st Class Honours
3.0-3.49	CGPA>=3.0 AND CPGA<3.5	Upper 2nd Class Honours
2.5-2.99	CGPA>=2.5 AND CPGA<3.0	Lower2nd Class Honours
2.0-2.49	CGPA>=2.0 AND CPGA<2.5	3rd Class Honours
1.7-1.99	CGPA>=1.7 AND CPGA<2.0	Pass
<1.7	CPGA<1.7	No Award

Code Example: lec07-10-JS-N-way-conditional.html

Conditional operator : ?

Another convenient way for if-else statement

```
var age = getAge(); // get age value from the getAge() function
var voteable = (age < 18) ? "Too young":"Old enough";
/*create variable voteable and assign it a value depending on the
variable age; if age < 18, voteable will by assigned with "Too young"
…*/
```

The above code is the same as

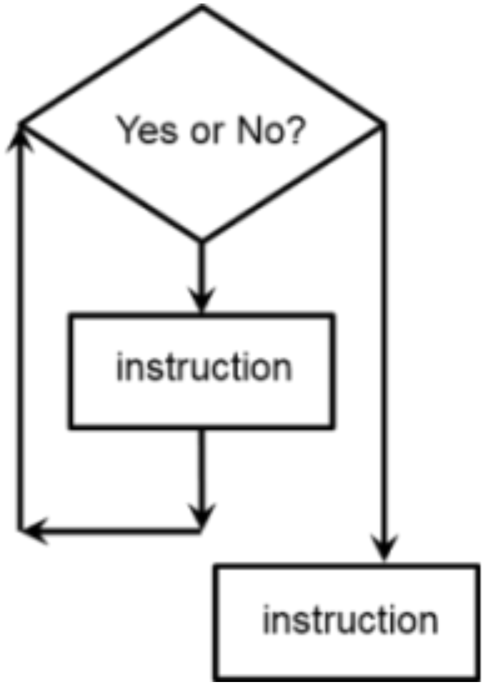
```
var voteable;
if (age < 18)
 voteable = "Too young";
else
 voteable = "Old enough";
```

Loop

A **Loop** is a construct that allows you to **repeat a block of code multiple times.**

e.g., a countdown timer

A loop often constitutes of a set of instructions that the computer follows over and over until a certain condition is met.



switch

A **multi-branch flow** control is easier to follow that multiple (nested) statements

- Execute statements associated with the case where its label matches the expression's value; if no matching label is found, the default case will be executed

Break statement ensures the program breaks out of switch once the matched statement is executed

- If there is no break statement, execution "*falls through*" to the next statement in the succeeding case

```
switch (expression) {
  case label1:


  case label2:


  ...
  case labelN:


  default:


}
```

Code Example: lec07-11-JS-switch.html

For-Loop (1)

Example: calculate summation from 1 to 10

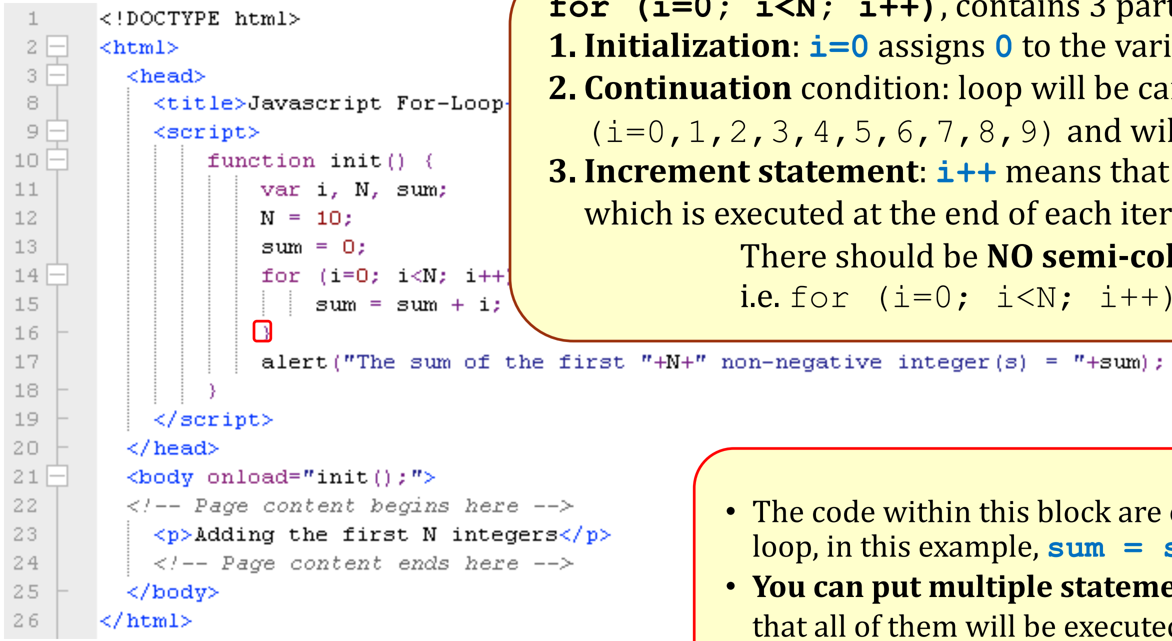
```
var sum, i;
sum = 0;

for (i=1; i<11; i++)
{
    sum = sum + i;
} /* sum = sum + i ; where i ranges from 1 to 10 */
```

The above code is equivalent to

```
var sum = 0;
sum = sum + 1;
sum = sum + 2;
 …
sum = sum + 10;
```

For-Loop (2)

The for-loop is often used to carry out a task for a finite number of times



```
<!DOCTYPE html>
<html>
  <head>
    <title>Javascript For-Loop
    <script>
      function init() {
        var i, N, sum;
        N = 10;
        sum = 0;
        for (i=0; i<N; i++
          sum = sum + i;
        }
        alert("The sum of the first "+N+" non-negative integer(s) = "+sum);
      }
    </script>
  </head>
  <body onload="init();">
  <!-- Page content begins here -->
    <p>Adding the first N integers</p>
    <!-- Page content ends here -->
  </body>
</html>
```

`for (i=0; i<N; i++)`, contains 3 parts inside the parentheses:

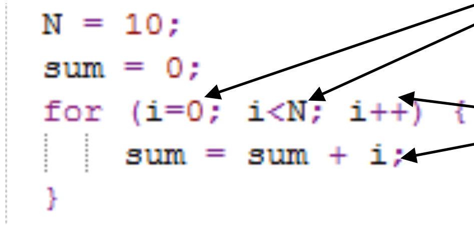
1. **Initialization**: **`i=0`** assigns **`0`** to the variable **`i`** at the beginning of the loop
2. **Continuation** condition: loop will be carried out **`i<10`** is **true** (`i=0,1,2,3,4,5,6,7,8,9`) and will **stop** when **`i<10`** is **false** (`i=10`)
3. **Increment statement**: **`i++`** means that the variable **`i`** is increased by **`1`**, which is executed at the end of each iteration after **`sum = sum + i;`**

There should be **NO semi-colon after the parenthesis**, i.e. `for (i=0; i<N; i++);` is wrong

- The code within this block are executed at each iteration of the for-loop, in this example, **`sum = sum + i;`**
- **You can put multiple statements inside the curly brackets** such that all of them will be executed at each iteration of the loop

Code Example: lec07-12-JS-for-loop.html

For-Loop (3)



```
N = 10;
sum = 0;
for (i=0; i<N; i++) {
    sum = sum + i;
}
```

The code on the left is executed according to the sequence of operations

```
N=10
sum=0
i=0
```

1st iteration

```
i<N ⇔ 0<10=true so the loop will continue to run
sum=sum+i ⇔ sum=0+0=0
i++ ⇔ i=0+1=1
```

2nd iteration

```
i<N ⇔ 1<10=true so the loop will continue to run
sum=sum+i ⇔ sum=0+1=1
i++ ⇔ i=1+1=2
```

3rd iteration

```
i<N ⇔ 2<10=true so the loop will continue to run
sum=sum+i ⇔ sum=1+2=3
i++ ⇔ i=2+1=3
```

4th iteration

```
i<N ⇔ 3<10=true so the loop will continue to run
sum=sum+i ⇔ sum=3+3=6
i++ ⇔ i=3+1=4
```

5th iteration

```
i<N ⇔ 4<10=true so the loop will continue to run
sum=sum+i ⇔ sum=6+4=10
i++ ⇔ i=4+1=5
```

6th iteration

```
i<N ⇔ 5<10=true so the loop will continue to run
sum=sum+i ⇔ sum=10+5=15
i++ ⇔ i=5+1=6
```

7th iteration

```
i<N ⇔ 6<10=true so the loop will continue to run
sum=sum+i ⇔ sum=15+6=21
i++ ⇔ i=6+1=7
```

8th iteration

```
i<N ⇔ 7<10=true so the loop will continue to run
sum=sum+i ⇔ sum=21+7=28
i++ ⇔ i=7+1=8
```

9th iteration

```
i<N ⇔ 8<10=true so the loop will continue to run
sum=sum+i ⇔ sum=28+8=36
i++ ⇔ i=8+1=9
```

10th iteration

```
i<N ⇔ 9<10=true so the loop will continue to run
sum=sum+i ⇔ sum=36+9=45
i++ ⇔ i=9+1=10
i<N ⇔ 10<10=false so the loop will end
```

While-loop (1)

```
for(expr1; expr2; expr3)
{
  loop statements;

}
```

```
expr1;
while(expr2)
{
   loop statements;
   expr3;
}
```

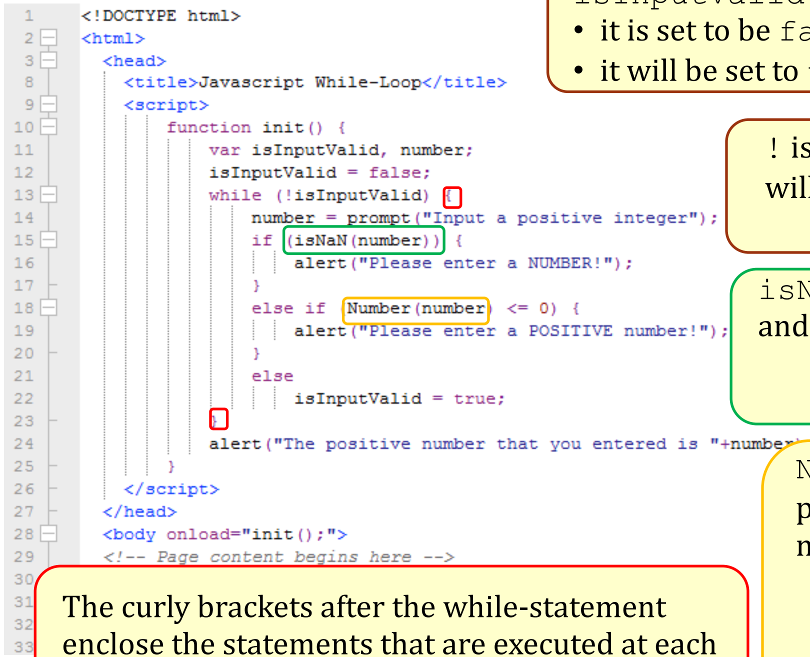
The loop statements is executed as long as **`expr2`** is true. When **`expr2`** becomes false, the loop ends.

`expr1`: Executed before entering the loop, often used for variable initialization

`expr3`: For each iteration, expr3 is executed after executing the loop body. Often used to **update** the counter variables (e.g., ***`i++`***).

While-Loop (2)

The while-loop is used to carry out a task repeatedly as long as a continuation condition is true



```
<!DOCTYPE html>
<html>
  <head>
    <title>Javascript While-Loop</title>
    <script>
        function init() {
            var isInputValid, number;
            isInputValid = false;
            while (!isInputValid) {
                number = prompt("Input a positive integer");
                if (isNaN(number)) {
                    alert("Please enter a NUMBER!");
                }
                else if (Number(number) <= 0) {
                    alert("Please enter a POSITIVE number!");
                }
                else
                    isInputValid = true;
            }
            alert("The positive number that you entered is "+number
        }
    </script>
  </head>
  <body onload="init();">
  <!-- Page content begins here -->
```

`isInputValid` is a Boolean variable which has value `true` or `false`

- it is set to be `false` initially
- it will be set to `true` if the user inputs a positive number

`!` is the **NOT** operator and will negate its subsequent Boolean expression

`isInputValid`	`!isInputValid`
true	false
false	true

`isNaN()` returns `true` if the current input is NOT a number and `false` if it is a number, e.g.,

`isNaN(234)=false`

`isNaN("abc")=true`

`Number` is JS built-in function that converts the given parameter to a number according to its value such that numeric calculations can be applied, e.g.,

`Number("123")=123`

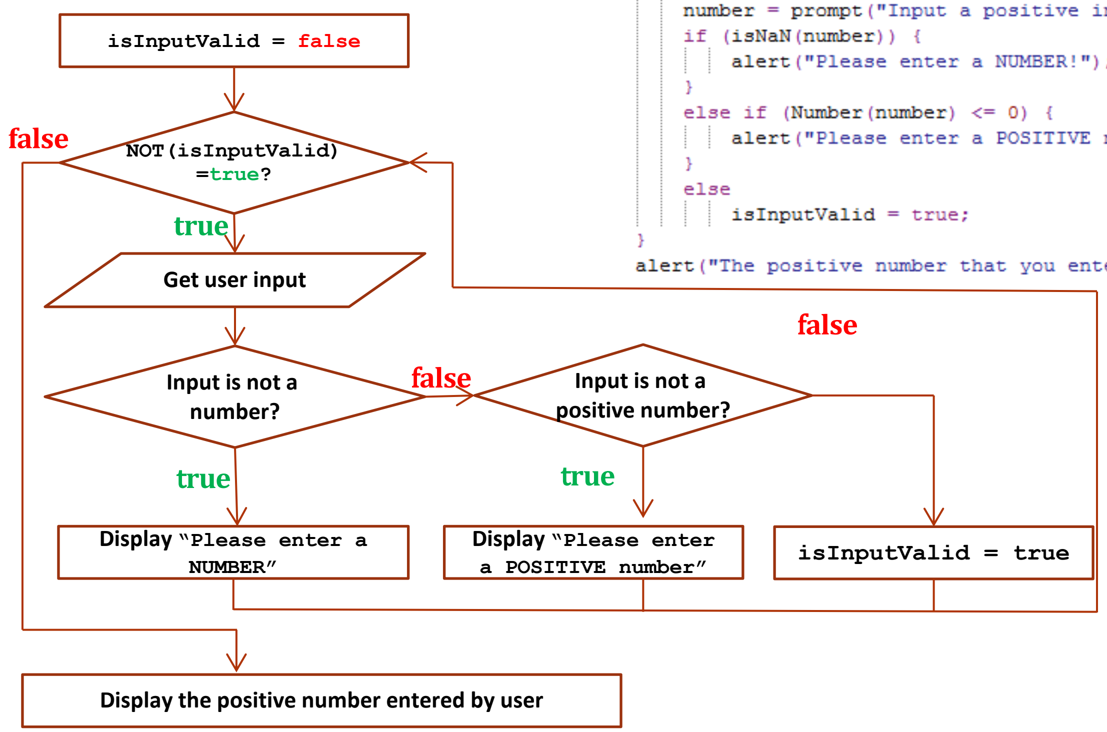
`Number("123")+1 = 124`

Critical thinking: what is the value of the expression `"123"+1 ?`

The curly brackets after the while-statement enclose the statements that are executed at each iteration of the while-loop

Code Example: lec07-13-JS-while-loop.html

While-Loop (3)


isInputValid = false
false
NOT(isInputValid) =true?
true
Get user input
Input is not a number?
false
Input is not a positive number?
false
true
true
Display "Please enter a NUMBER"
Display "Please enter a POSITIVE number"
isInputValid = true
Display the positive number entered by user

```
isInputValid = false;
while (!isInputValid) {
    number = prompt("Input a positive integer");
    if (isNaN(number)) {
        alert("Please enter a NUMBER!");
    }
    else if (Number(number) <= 0) {
        alert("Please enter a POSITIVE number!");
    }
    else
        isInputValid = true;
}
alert("The positive number that you entered is "+number);
```

do statement

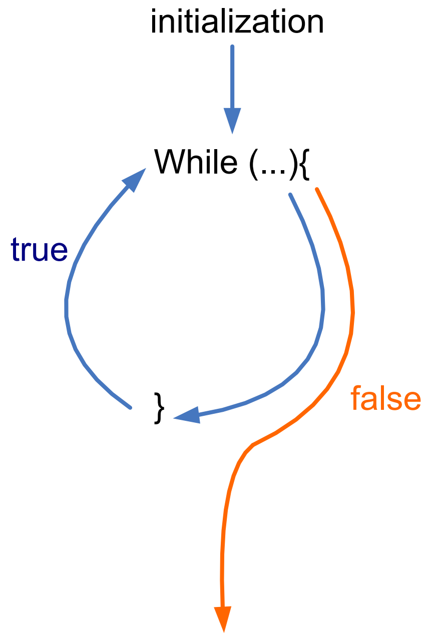
General form of do statement (*repetition* statement)

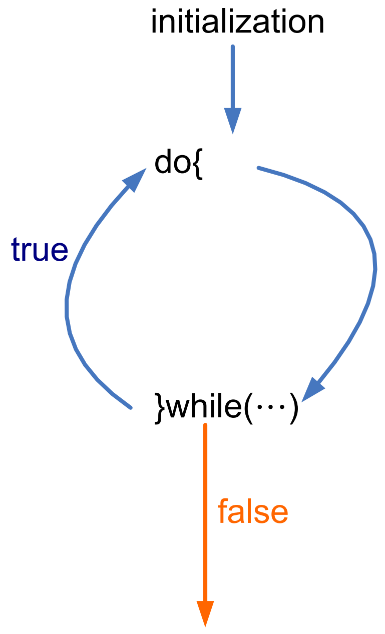
```
do
{
   statement(s);
}
while (expression);
```

Semantics

- statement is executed first; thus the **loop** body is run **at least once**
- If the value of expression is non-zero (*true*), the loop repeats; otherwise, the loop terminates

While vs Do while





Break, Continue, Return & Block

`Break;`

- terminate the current loop, switch and transfer program control to the statement following the terminated statement

`Continue;`

- terminate execution of the statements in the current iteration of the current loop and go to the next iteration and continue with the loop.

`Return;`

- used inside function, terminate the function execution and may return value to the function caller.

Block: the curly brackets **`{…}`** used to group a number of statements into a unit, not actually a statement.

Lecture summary

JavaScript is a programming language that sends browser instructions to execute a set of actions, such as popping up an alert window, dynamically generating HTML elements, etc

JavaScript has several operators, including arithmetic operators, logical operators, comparison operators.

`NaN` is a special value representing not-a-number; **`Try`** and **`catch`** are statements that prevents vulnerable codes from blocking the entire JavaScript code block

Control flow allows developers to execute a program under different conditions, repeat and terminate the program based on predefined conditions.

Agenda

JavaScript Basics

- Expression
- Statement
- Operation

JavaScript Control Flow

- Conditional
- Loop

Mid-term brief review

About Mid-term

Coverage: Lec 01 – Lec 07, Lab 1 – Lab 5

Form: 40 multiple-choice questions, each taking up 0.5 marks; in total the mid-term takes up 20% of your final assessment

Reminder:

1. Please arrive the venue (LT-10) on time
2. Please bring pencil and eraser

Lec 01: Introduction and Internet (1)

Key concepts

- Computer networks
- Network topology
- Communication channel
- IP address
- Internet Protocol / DNS
- Email sending and receiving process

Lec 01: Introduction and Internet (2)

What is **IP Address**?

What is **DNS**?

Lec 02 – Lec 03: HTML (1)

Key concepts

- What is HTML
- Commonly used tags: `<h1>`, `<p>`, `<a>`, `<img>`, `<span>`, `<form>`, `<table>`, `<ol>`, `<ul>`, `<div>`
- Basic syntax
- **File path, relative/absolute url**
- Web accessibility and multi-browser support

Lec 02 – Lec 03: HTML (2)

How do you define an image in HTML?

A. `<image src="url" alt="description">`

B. `<img href="url" alt="description">`

C. `<img src="url" alt="description">`

D. `<src="url" img="description">`

What are the key differences between the “post” and “get” methods in HTML forms?

Lec 02 – Lec 03: HTML (3)

In the following HTML code, the data sent back from the textbox cannot be processed properly, because________?

```
<form action="a.html" method="get">
    <p>
        <input type="text" id="thisbox" value="thisbox" />
        <input type="submit" value="submit" />
    </p>
</form>
```

To add a video in a webpage, how to improve user experience across platforms?

Lec 04 – Lec 06: CSS (1)

Key concepts

- What is CSS
- How to incorporate CSS in HTML pages
- Basic syntax
- Media queries
- CSS selector, document tree, media query
- Cascading and inheritance
- CSS layout and positioning, responsive design
- CSS3 effects, transition, animation

Lec 04 – Lec 06: CSS (2)

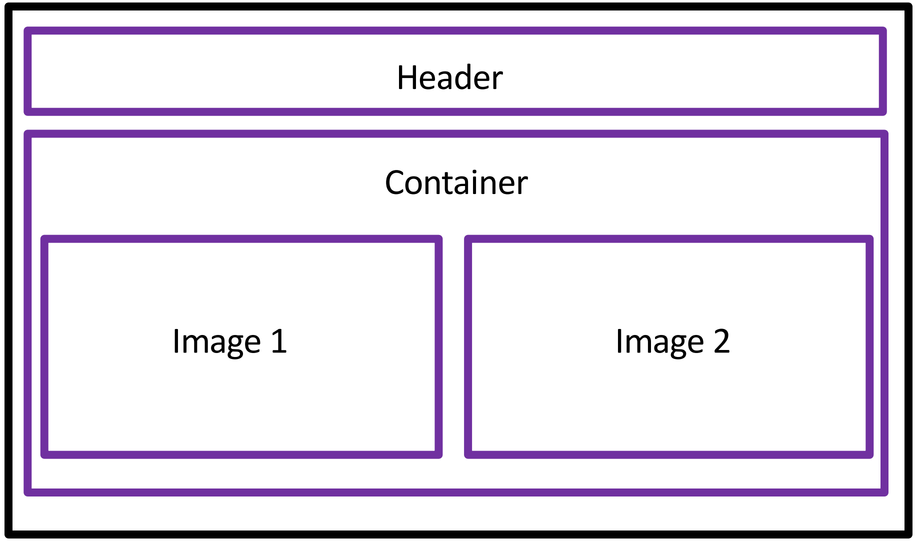
For the following styles markup, what will be the styles applied to the text ”head1” ?

```
#main { color: blue; }
body { color: black; }
#page { letter-spacing: 0.2em; }
h3 { text-transform: uppercase; }
```

```
<body>
    <div id="main">
        <h3>head1</h3>
        <div id="page">
            <h3>head2</h3>
        </div>
    </div>
</body>
```

Lec 04 – Lec 06: CSS (3)

How to apply proper CSS style to achieve a specific layout (e.g., the layout shown below)?



Lec 06 - 07: JavaScript Basic

Key concepts

- What is JavaScript (JS)
- How to incorporate JS in HTML pages
- Variable naming and scope
- Basic data types

Most parts (90%) of the mid-term questions will focus on Internet concepts, HTML, and CSS. We only have a few questions on basic JavaScript