# Web Programming

## JavaScript – Programming Basics (1)

School of Computing, Gachon University

# Roadmap: Common Programming Basics

- Variables and Data Types

- Expressions and Operators
- Conditionals

# JavaScript Variables

- **Using variables**
  - Represent numbers, strings, …
  - Declare and assign a value
    - Keyword `var` to declare the variable
    - Assignment operators to assign the variable a value
    - If you declare a variable without assigning a value, the default value of variable is "undefined".
  - Keyword `var` is not mandatory
- **Example:**
  - `var score;`
  - `var lastName = “Doe”;`
  - `age = 21;`

# JavaScript Identifiers

- **Variable names**
  - Use a-z, A-Z, 0-9, $, _
  - Spaces, symbols are not allowed
  - The first letter of a variable can be a-z, A-Z, $, _
    - Numbers are not allowed for the first letter
  - Variables are case-sensitive
    - e.g., Count, count, and COUNT are all different variables

Correct expression:

myVariable
My_variable
My_1st_variable
$my_variable
_my_variable

Wrong expression:

1my_example (✗)
@my_variable (✗)
~my_variable (✗)
++my_variable (✗)

# Data Types (1/2)

- **Data types**
  - JavaScript allows a variable to have different data types without declaration
    - (different from C, Java, etc.)
  - The data type of a variable is defined when a value is assigned to it

```
var score;
score = 66.8;
score = “high”;
```

# Data Types (2/2)

- **Primitive data types**
  - **Number**: integer, floating-point numbers
  - **Boolean**: logical values "true" or "false"
  - **String**: a sequence of alphanumeric characters
- **Composite data types (or complex data types)**
  - **Array**: a sequence of values
  - **Object**: a named collection of data
- **Special data types**
  - **Null**: empty variable. assigned by a value null
  - **Undefined**: the variable has been created, but not yet assigned a value

# Primitive Data Types (1/3)

- **Numeric types**
  - Integer
    - Positive or negative number with no decimal point
    - Ranged from $-2^{53}$ to $2^{53}$
  - Floating-point number
    - Usually written in exponential notation
    - 3.14159, 2.5e10 ($2.5X10^{10}$)

# Example: Numeric types

## ex10-1.html

```
1   <!Doctype html>
2   <html>
3       <head>
4           <meta charset="utf-8">
5           <title>JavaScript</title>
6       </head>
7       <body>
8           <h3> Numeric types</h3>
9           <script>
10              var x1 = 34;
11              var x2 = 34.00;
12              var x3 = 123e5;
13              var x4 = 123e-5;
14              document.write("<p>x1 = " + x1 + "</p>");
15              document.write("<p>x2 = " + x2 + "</p>");
16              document.write("<p>x3 = " + x3 + "</p>");
17              document.write("<p>x4 = " + x4 + "</p>");
18          </script>
19      </body>
20  </html>
```

**Numeric types**

x1 = 34

x2 = 34

x3 = 12300000

x4 = 0.00123

# Primitive Data Types (2/3)

- **Boolean type**
  - A Boolean value is a logical value of either **true** or **false** (yes/no, on/off)
  - In JavaScript, you can use the words “true” and “false” directly to indicate a Boolean value

# Example: Boolean type

## ex10-2.html

```
1   <!Doctype html>
2   <html>
3       <head>
4           <meta charset="utf-8">
5           <title>JavaScript</title>
6       </head>
7       <body>
8           <h3> Boolean types</h3>
9           <script>
10              document.write("<p>52 > 273 : " + (52 > 273) + "</p>" );
11              document.write("<p>52 < 273 : " + (52 < 273) + "</p>");
12          </script>
13      </body>
14  </html>
```

**Boolean types**

52 > 273 : false

52 < 273 : true

# Primitive Data Types (3/3)

- **String type**
  - A string variable can store a sequence of alphanumeric characters, spaces, and special characters
  - String can be enclosed in either single quotation marks (‘’) or double quotation marks (“”)
  - Unlike Java and C, JavaScript does not have a single character (char) data type

# Example: Data type

## ex10-3.html

```
1   <!Doctype html>
2   <html>
3       <head>
4           <meta charset="utf-8">
5           <title>JavaScript</title>
6       </head>
7       <body>
8           <h3> String types</h3>
9           <script>
10              var str1 = "He is called 'Johnny'.";
11              var str2 = 'She is called "Alice".';
12              document.write(str1 + "<br>");
13              document.write(str2);
14          </script>
15      </body>
16  </html>
```

**String types**

He is called 'Johnny'.
She is called "Alice".

# Composite Data Types (1/3)

- **Object** **(We will cover this later)**
  - Everything in JavaScript is an object
  - In the Web browser, the browser window, forms, buttons, text boxes, etc. are also objects
  - **Methods** are things that objects can do.
    - Window object can alert the user by “alert()”
  - All objects have **properties**
    - Browser has a name and the version number

# Composite Data Types (2/3)

- **Array object**
  - An array contains a set of data represented by a single variable name
  - An array in JavaScript are represented by an **Array** object; "new Array(n)" to construct this object
    - "new" means you are creating an object
  - The first element of an array is "Array[0]"; the last one Array[n-1]
    - myArray = new Array(5);  // We have myArray[0] ~ myArray[4]
  - Can also declare arrays without a variable length; allows automatic extension of the length
    - Car = new Array();
    - Car[9] = "Ford"; Car[99] = "Honda";

# Example: Array object

## ex10-4.html

```
1   <!Doctype html>
2   <html>
3       <head>
4           <meta charset="utf-8">
5           <title>JavaScript</title>
6       </head>
7       <body>
8           <h3> Array object</h3>
9           <script>
10              var list = new Array(1, true, "JavaScript");
11              document.write(list + "<hr>");
12              var city = new Array();
13              city[0] = "Seoul";
14              city[2] = "London";
15              for(var i=0; i<city.length; i++) {
16                  document.write("city[" + i +"] = " + city[i] + "<br>");
17              }
18          </script>
19      </body>
20  </html>
```

**Array object**

1,true,JavaScript

city[0] = Seoul
city[1] = undefined
city[2] = London

# Composite Data Types (3/3)

- **Array literal**
  - Can also declare an array by assigning elements
  - Using an array literal is the easiest way to create a JavaScript Array.
    - arrayName = [ element1, element2, …];
  - m x n array elements are accessed by:
    - arrayName[i][j] …

# Example: Array literal

ex10-5.html

```
1   <!Doctype html>
2   <html>
3       <head>
4           <meta charset="utf-8">
5           <title>JavaScript</title>
6       </head>
7       <body>
8           <h3> Array literal</h3>
9           <script>
10              var score = [90, 70, 60, 50, 40];
11              var sum = 0;
12              for(var i=0; i<score.length; i++) {
13                  sum += score[i];
14              }
15              document.write("sum = " + sum );
16          </script>
17      </body>
18  </html>
```

**Array literal**

sum = 310

# Special Data Types

- **Null**
  - Refers to “nothing”
- **Undefined**
  - A value “undefined” is returned when you attempt to use a variable whose type and value are not provided
- **You can set a variable as “null” if you want absolutely nothing in it; but you just don’t want it to be “undefined”**

# Special Characters

| Character | Meaning |
|---|---|
| \b | Backspace |
| \t | Horizontal tab |
| \n | New line |
| \\ | Backslash |
| \’ | Single quote |
| \” | Double quote |

- In JavaScript you can add special characters to a text string by using the backslash sign.
- The backslash (\) is used to insert apostrophes, new lines, quotes, and other special characters into a text string.

# Roadmap: Common Programming Basics

- Variables and Data Types
- Expressions and Operators
- Conditionals

# Operator

- **typeof operator**
  - A unary operator
  - Returns: number, string, Boolean, object, function, undefined, null

# Example: typeof operator

## ex10-6.html

```
1   <!Doctype html>
2   <html>
3       <head>
4           <meta charset="utf-8">
5           <title>JavaScript</title>
6       </head>
7       <body>
8           <h3> typeof operator</h3>
9           <script>
10              var a=254, b="hello", c = 3>2, d=[52, 23], x=null, y;
11              document.write("<br> Type of a: " + typeof(a));
12              document.write("<br> Type of b: " + typeof(b));
13              document.write("<br> Type of c: " + typeof(c));
14              document.write("<br> Type of d: " + typeof(d));
15              document.write("<br> Type of x: " + typeof(x));
16              document.write("<br> Type of y: " + typeof(y));
17          </script>
18      </body>
19  </html>
```

**typeof operator**

Type of a: number
Type of b: string
Type of c: boolean
Type of d: object
Type of x: object
Type of y: undefined

# Expression

- **A set of literals, variables, operators that evaluate to a single value**
  - Binary: **left_operand operator right_operand**
  - Unary: **operator operand**
- **Different operators for different types of expressions**
  - Arithmetic
  - Logical
  - String
  - Conditional

# Arithmetic Operators (1/2)

- Arithmetic operators - binary

| Operator | Name | Description | Example |
|---|---|---|---|
| + | addition | adds the operands | 3 + 5 |
| - | subtraction | subtracts the right operand from the left operand | 5 - 3 |
| * | multiplication | multiplies the operands | 3 * 5 |
| / | division | divides the left operand by the right operand | 30 / 5 |
| % | modulus | calculates the remainder | 20 % 5 |

# Arithmetic Operators (2/2)

- **Arithmetic operators – unary (++, --)**
  - Prefix operator
    - Placed before the operand; value returned after operation
    - e.g., `++count;  --count;`
  - Postfix operator
    - Placed after the operand; value returned before operation
    - e.g., `count++; count --;`

# Assignment Operators

| Operator | Description | Example |
|---|---|---|
| = | assigns the value of the right operand to the left operand | A = 2 |
| += | add the operands and assigns the result to the left operand | A += 5 |
| -= | subtracts the operands and assigns the result to the left operand | A -= 5 |
| *= | multiplies the operands and assigns the result to the left operand | A *= 5 |
| /= | divides the left operands by the right operand and assigns the result to the left operand | A /= 5 |
| %= | assigns the remainder to the left operand | A %= 5 |

# Example: Operators

## ex10-7.html

```
1   <!Doctype html>
2   <html>
3       <head>
4           <meta charset="utf-8">
5       </head>
6       <body>
7           <h3> Operator</h3>
8           <script>
9               var x=32, y=20;
10              y *= x;
11              var div = x / 10;
12              var mod = x % 2;
13              document.write("<br> x = " + x);
14              document.write("<br> y = " + y);
15              document.write("<br> div = " + div);
16              document.write("<br> mod = " + mod);
17              document.write("<br> ++x = " + ++x);
18              document.write("<br> x++ = " + x++);
19              document.write("<br> x = " + x);
20          </script>
21      </body>
22  </html>
```

**Operator**

x = 32
y = 640
div = 3.2
mod = 0
++x = 33
x++ = 33
x = 34

# Comparison Operators

| Operator | Name | Description | Example |
|---|---|---|---|
| == | equal | type conversion before checking equality | “5” == 5 |
| === | strictly equal | no type conversion before testing | “5” === 5 |
| != | not equal | “true” when operands are not equal | “4” != 2 |
| !== | strictly not equal | no type conversion before testing inequality | 5 !== “5” |
| > | greater than | “true” if left operand is greater than right operand | 5 > 2 |
| < | less than | “true” if left operand is less than right operand | 2 < 5 |
| >= | greater than or equal | “true” if left operand is greater than or equal to right operand | 5 >= 2 |
| <= | less than or equal | “true” if left operand is less than or equal to right operand | 5 <= 2 |

# Example: Comparison Operators

ex10-8.html

```
1   <!Doctype html>
2   <html>
3       <head>
4           <meta charset="utf-8">
5       </head>
6       <body>
7           <h3>Comparison Operator</h3>
8           <script>
9               var a=2022;
10              var b="2022";
11              var c=2012;
12              document.write("a="+ a + ", b=" + b + ", c=" + c);
13              document.write("<br> a==b : " + (a==b));
14              document.write("<br> a === b : " + (a===b));
15              document.write("<br> a > b : " + (a>b));
16              document.write("<br> a > c : " + (a>c));
17          </script>
18      </body>
19  </html>
```

**Comparison Operator**

a=2022, b=2022, c=2012
a==b : true
a === b : false
a > b : false
a > c : true

# Logical Operators

○ **Perform Boolean operations on Boolean operands**

| Operator | Name | Description | Example |
|---|---|---|---|
| **&&** | logical AND | evaluate to “true” when both operands are true | 3>2 && 5<2 |
| **\|\|** | logical OR | evaluate to “true” when either operand is true | 3>1 \|\| 2>5 |
| **!** | logical NOT | evaluate to “true” when the operand is false | !(5==3) |

# Example: Logical Operators

## ex10-9.html

```
1  <!Doctype html>
2  <html>
3      <head>
4          <meta charset="utf-8">
5      </head>
6      <body>
7          <h3>Logical Operator</h3>
8          <script>
9              document.write("<br> (312 > 302) && (399 < 400) : "+ ((312>302) && (399<400)));
10             document.write("<br> (44 == -44) || (-1 < 0): " + ((44 == -44) || (-1<0)));
11             document.write("<br> !(-100 < 0) : " + (!(-100<0)));
12         </script>
13     </body>
14 </html>
```

**Logical Operator**

(312 > 302) && (399 < 400) : true
(44 == -44) || (-1 < 0): true
!(-100 < 0) : false

# Operator Precedence

○ Expressions are evaluated on a left-to-right basis with the highest priority precedence evaluated first

| Precedence | Operator |
|---|---|
| 1 | parentheses, function calls |
| 2 | ~, -, ++, --, new, void, delete |
| 3 | *, /, % |
| 4 | +, - |
| 5 | <<, >>, >>> |
| 6 | <, <=, >, >= |
| 7 | ==, !=, ===, !== |
| 8 | & |

| Precedence | Operator |
|---|---|
| 9 | ^ |
| 10 | \| |
| 11 | && |
| 12 | \|\| |
| 13 | ?: |
| 14 | =, +=, -=, *=, … |
| 15 | comma(,) operator |

# String Objects

- **Literal strings and string variables are represented by a string object**
- **The string object contains methods/properties for manipulating text strings**
  - length - returns the number of characters in a string
  - substr() - extracts a substring

# String Operators

- **String concatenation operator (+)**

```
var oneString = “one”;
var anotherString = oneString + “, two, three, …”;
// We get anotherString = “one, two, three, …”
```

  - If you add a number and a string, the result will be a string!

    var y = “2" + 8;

    → 28

- **String assignment operator (+=)**

```
var oneString = “one”;
oneString += “, two, three, …”;
// We get oneString = “one, two, three, …”
```

# Example: String Operators

## ex10-10.html

```
1   <!Doctype html>
2   <html>
3       <head>
4           <meta charset="utf-8">
5           <title>JavaScript</title>
6       </head>
7       <body>
8           <h3>String Operators</h3>
9           <script>
10              var greeting = "Hello";
11              document.write(greeting += "Luna" + "<br>");
12              document.write("Web " + "Programming <br>");
13              document.write(23 + "HTML <br>");
14              document.write(23 + 35 + "<br>");
15              document.write(23 + 35 + "CSS <br>");
16              document.write("JavaScript" + 23 + 35 + "<br>");
17          </script>
18      </body>
19  </html>
```

**String Operators**

HelloLuna
Web Programming
23HTML
58
58CSS
JavaScript2335

# Roadmap: Common Programming Basics

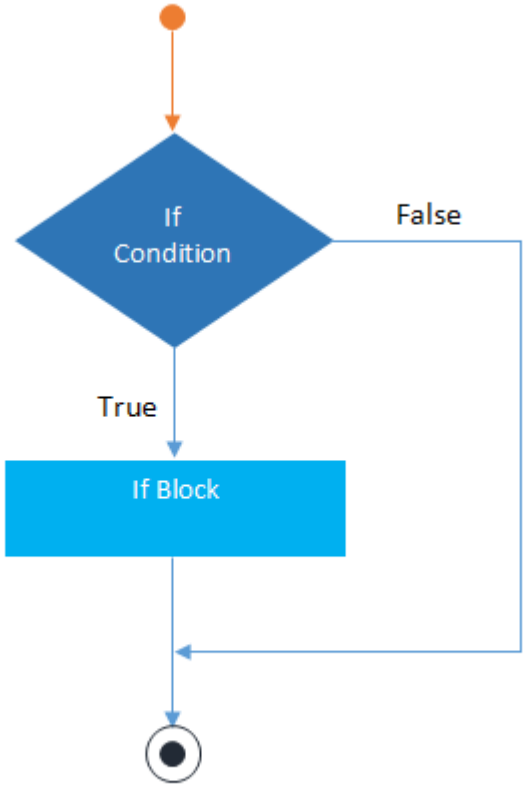
- Variables and Data Types
- Expressions and Operators
- Conditionals

# Conditional Statements

- `if`
- `if/else`
- `if/else if`
- `switch/case`

# if

- **Syntax**
  - `if` (*condition*) { *statements* }
  - "if" in lowercase; using "IF" will result error!
  - *Condition* yields a logical *true* or *false* value
  - If the condition is true, *statements* are executed




If statement

# Example: if

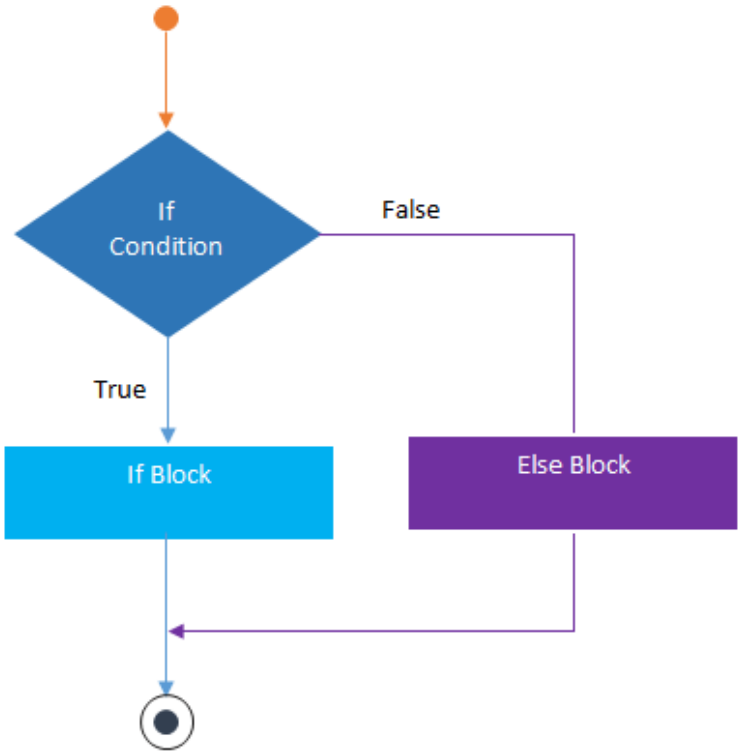
## ex10-11.html

```
1   <!Doctype html>
2   <html>
3       <head>
4           <meta charset="utf-8">
5       </head>
6       <body>
7           <h3>if statement</h3>
8           <script>
9               var age = prompt('Enter your age: ');
10              if(age >= 18) {
11                  document.write("You can sign up.");
12              }
13          </script>
14      </body>
15  </html>
```

# if / else

- **if/else statement**
  - `if` (*condition*) { *statements-1* }

    `else` { *statements-2* }
  - If the *condition* is false, execute statements-2




If-Else statement

# Example: if/else

## ex10-12.html

```
1   <!Doctype html>
2   <html>
3       <head>
4           <meta charset="utf-8">
5       </head>
6       <body>
7           <h3>if/else statement</h3>
8           <script>
9               var age = prompt('Enter your age: ');
10              if(age >= 18) {
11                  document.write("You can sign up.");
12              }
13              else {
14                  document.write("You must be at least 18 to sign up.")
15              }
16          </script>
17      </body>
18  </html>
```

# if / else if

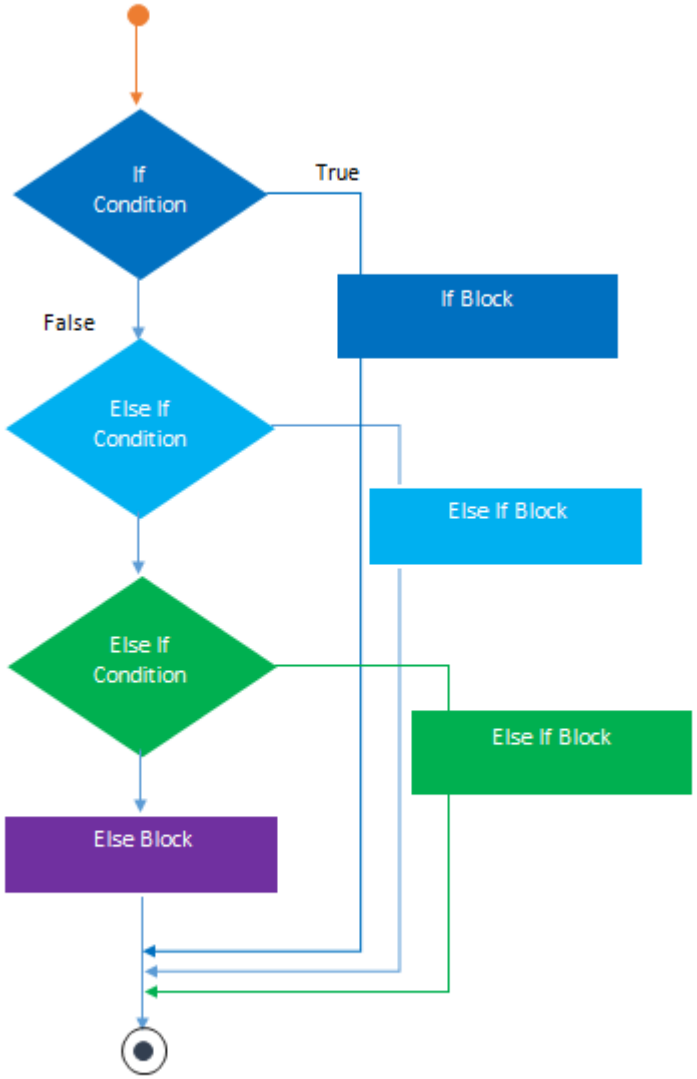
- **if/else if statement**
  - `if` (*condition-1*) { *statements-1* }

    `else if` (*condition-2*) { *statements-2* }

    `else` { *statements-3* }
  - If the *condition-1* is false,

    *condition-2* is evaluated to execute

    either statements-2 or statements-3




If-Else-If Ladder

# Example: if / else if

ex10-13.html

```
<!Doctype html>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <h3>if/else if statement</h3>
        <script>
            var score = prompt('Enter your score: ');
            score = parseInt(score);
            if(score >= 90) {
                document.write(score+" is A.");
            }
            else if(score >= 80) {
                document.write(score+" is B.");
            }
            else if(score >= 70) {
                document.write(score+" is C.");
            }
            else if(score >= 60) {
                document.write(score+" is D.");
            }
            else{
                document.write(score+" is F.");
            }
        </script>
    </body>
</html>
```

# Conditional Ternary Operator

- **Another way for if/else**
  - Syntax:
    - (*condition*) **?** (*statement-1*) **:** (*statement-2*)
    - (*condition*) **?** (*value-1*) **:** (*value-2*)
  - Substitutes for a simple "if/else" statement
  - e.g.,
    - **`if`** (3 > 2) { alert("true"); } **`else`** { alert("false"); }
    - (3 > 2) **?** alert("true") **:** alert("false");

# Example: Conditional Ternary Operator

## ex10-14.html

```
<!Doctype html>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <h3>Ternary Operator</h3>
        <script>
            var age = prompt("Enter your age: ");
            var result = (age>=18)?"You can sign up.":"You must be at least 18 to sign up.";
            document.write(result);
        </script>
    </body>
</html>
```

# switch / case (1/2)

- **Syntax:**
  - **`switch`** (*expression*) {

    **`case`** *label1*:

    *statements-1*; **`break`**;

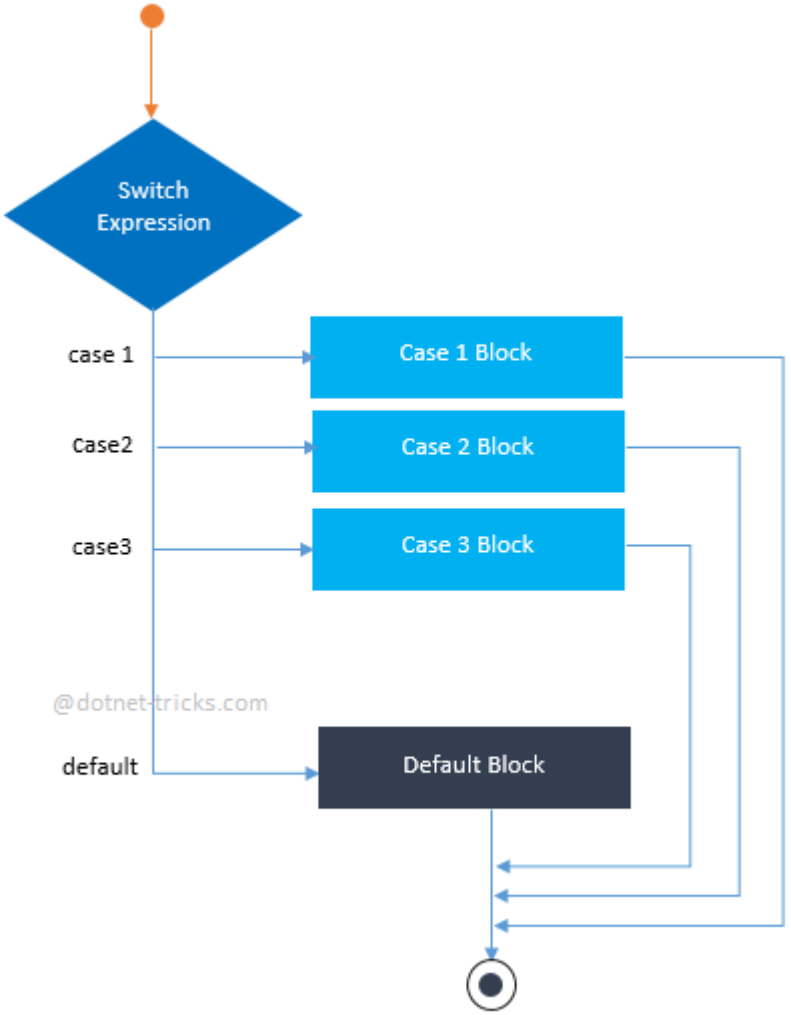
    ...

    **`default`**:

    *statements*

    }

  - Controls program flow by executing a specific *statements*, depending on the value of the *expression*




Switch Statement

# switch / case (2/2)

- **Decision making**
  - `case` labels
    - Identify specific code segments
    - Can use a variety of data types as case labels
  - `break` statement
    - Used to exit switch statements
  - `default` label
    - Contains statements that execute when the condition expression doesn't match any of the case labels

# Example: switch / case

## ex10-15.html

```
<!Doctype html>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script>
            var city;
            city = prompt("Please enter a name of the capital city : ");
            switch(city) {
                case "Seoul":
                    document.write("Seoul is the capital city of Korea.");
                    break;
                case "Beijing":
                    document.write("Beijing is the capital city of China.");
                    break;
                case "Paris":
                    document.write("Paris is the capital city of France.");
                    break;
                default:
                    document.write("Cannot find which country this city is the capital city of.");
                    break;
            }
        </script>
    </body>
</html>
```

# Exercise 1

- **Write program to check whether a number is positive, negative or zero**
  - Read an integer number(`num`) from user using `prompt()`.
  - Use `if/ else if / else`

- **Output**
  - if(num < 0), then number is negative.

```
-108 is negative.
```

  - if(num > 0), then number is positive.

```
24 is positive.
```

  - if(num == 0), then It is zero.

```
It's zero.
```

# Exercise 2

- **Uses the `switch/case` statement to check the day of the week based on a day number and output a message**
  - Declare a day variable(`day`) storing the day number and read a number from user using `prompt()`. (`day` variable is string type)
  - Uses the `switch/case` statement to assign the day of the week based on the day number. If the day is 1, the day of the week is Sunday. If the day is 2, the day of the week is Monday, and so on.
- **Output**
  - from Monday to Friday : "We are open."
  - on Saturday and Sunday: "We are closed."
  - numbers other than 1 to 7: "Invalid day."

# Exercise 3

- **Write a program to find number of days in month.**
  - Read the month number(`number`) from the user using `prompt()`.
  - Use logical OR operator preforming single task on multiple condition.
  - Print corresponding number of days in that month using the following table.

| Month | Total days |
|---|---|
| January, March, May, July, August, October, December | 31 days |
| February | 28/29 days |
| April, June, September, November | 30 days |

- **Output**
  - 1, 3, 5, 7, 8, 10, 12 : 31 days
  - 4, 6, 9, 11: 30 days
  - 2: 28 or 29 days
  - Otherwise: Invalid Month

# End of Class