JavaScript 1

Objectives

- To understand the basic concept of JavaScript

Contents

- What is JavaScript?
- How to use JavaScript?
- JavaScript Variables
- JavaScript Data Types and Operators

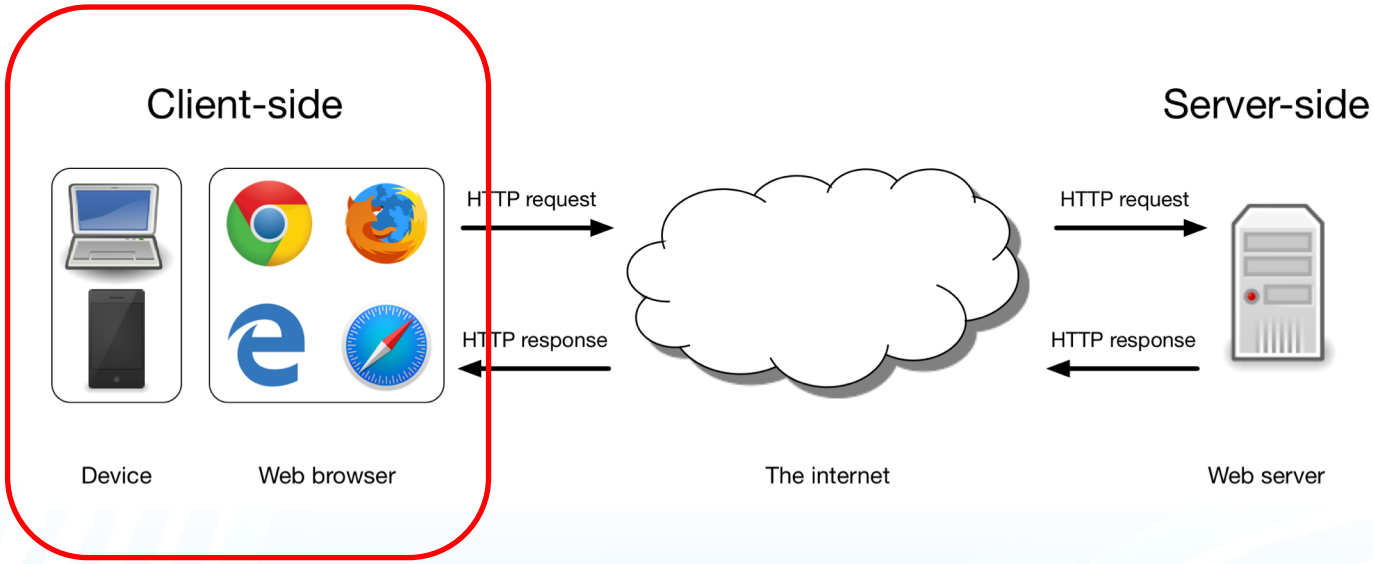
Contents

- What is JavaScript?
- How to use JavaScript?
- JavaScript Variables
- JavaScript Data Types and Operators

What is Java Script?

- Web application components:
  - A single **HTML file** describing the content and structure of the page
  - Zero or more **CSS files** that describe the presentation or style of page elements
  - Zero or more **JavaScript files** that specify dynamic, interactive behaviour
  - Zero or more **media files**, e.g., images, audio or video

What is Java Script? (cont.)



What is Java Script? (cont.)

- JavaScript was designed to add **interactivity** to HTML pages
- JavaScript is a scripting language, a **lightweight** programming language
- A JavaScript is usually **embedded directly** into HTML pages
- JavaScript is an **interpreted** language (means that scripts execute without preliminary compilation)
- JavaScript is accompanied by a vast array of prewritten libraries of code called APIs (Application Programming Interfaces), see https://developer.mozilla.org/en/docs/Web/API.

What is JavaScript? (cont.)

- JavaScript can put **dynamic text** into an HTML page
- JavaScript can **react to events**
- JavaScript can **read and write HTML elements**
- JavaScript can be used to **validate data**
- JavaScript can be used to **detect the visitor's browser**
- JavaScript can be used to **create cookies**
- Why JavaScript?

Contents

- What is JavaScript?
- How to use JavaScript?
- JavaScript Variables
- JavaScript Data Types and Operators

How to use JavaScript?

- We can **insert** JavaScript in **between parts of the HTML code**.
- We inform browsers which parts are client side script by using
  - **<script>**: Starting of JavaScript
  - **</script>**: End of JavaScript
- Script tag can be inserted in <head>, <body>, or both

How to use JavaScript?

- We can import JavaScript from **.js file** using <script>

<script type="text/javascript" src="javascript.js"></script>

Type: Specifying the scripting language, for instance: type="text/javascript" or type="text/vbscript"

src: gives a URL for an external script.

Note: - in xxx.js, no need to put <script>, attributes of script tag, and </script> tag
- .js is JavaScript file

Contents

- What is JavaScript?
- How to use JavaScript?
- JavaScript Variables
- JavaScript Data Types and Operators

JavaScript Variables

- Modify the first program
- Variable declaration
- Assignment statement
- Displaying Output

JavaScript Variables

```
<script type="application/javascript">
    //1. First program
    var firstName = "Wudhichart"; // Declaration and initial assignment
    var lastName, fullinfo;
    var ID;

    ID = "9999999"
    lastName = "Sawangphol";

    fullinfo = ID + " " + firstName + " " + lastName;

    console.log(fullinfo);
</script>
```

Variables

- A **variable** is a way for a program to temporarily store a piece of information and subsequently refer it it by a name

```
var firstName = "Wudhichart";
```

Variable declarations

```
var firstName = "Wudhichart";
```

- This statement is a variable declaration
- The keyword **var** is used to indicate that a variable is to be declared.
- Declaring a variable requires:
  - a **variable name** and
  - **optionally** an **initial value** for the variable.

Variable Name Rules

- They **must start** with **a letter**
- Containing only **a mix of letters and digits**, no spaces.
- Variable names **cannot be a JavaScript keyword** (like var).

Assignment statement

```
var firstName = "Wudhichart"; // Declaration and initial assignment
```

- Unlike mathematics, = does not represent an equality sign in JavaScript. It is instead an assignment operator

More declarations and assignments

```
var lastName, fullinfo;
```

```
lastName = "Sawangphol";
```

```
fullinfo = ID + " " + firstName + " " + lastName;
```

Displaying Output

```
console.log(fullinfo);
```

- There are two to display an output:
  - On console
  - On webpage
- **console.log(…)** instructs JavaScript to log (write out) to the console the value of the expression (remember it evaluates to a single value)
- Open Console (Note that only Chrome has this panel):
  - Windows: F12
  - macOS: option+cmd+J

Check understanding

- Create a program:
  - Declare one variable named *number1*
  - Output *number1*
  - Assign *number1* with 10
  - Output *number1*
  - Increase the value of *number1* by 10
  - Output *number1*

Contents

- What is JavaScript?
- How to use JavaScript?
- JavaScript Variables
- JavaScript Data Types and Operators

JavaScript Data Types

- Boolean
  - Boolean represents a logical entity and can have two values: true, and false.
- Number
  - The value of Number is between $-(2^{53} -1)$ and $2^{53} -1$
- String
  - It is used to represent textual data.
- Null
  - Null represents the intentional absence of any object value. The single value of the Null data type (null) is a bit like the undefined value except it's within our control.
- Undefined
  - A variable that has not been assigned a value has the value undefined.
- Objects

JavaScript Data Types

```
<script type="application/javascript">
    //3. Data-types
    // Getting started with data types
    var outMessage = "";
    var myNum = 10;
    var myStr = "ten";
    var MyBool = true;

    outMessage = "The value of myNum is " + myNum + " and type is " + (typeof myNum);
    console.log(outMessage);

    outMessage = "The value of myStr is " + myStr + " and type is " + (typeof myStr);
    console.log(outMessage);

    outMessage = "The value of MyBool is " + MyBool + " and type is " + (typeof MyBool);
    console.log(outMessage);

    myNum = "Hello World!!"; // Is this Possible?

    outMessage = "The value of myNum is " + myNum + " and type is " + (typeof myNum);
    console.log(outMessage);
</script>
```

What is the output?

JavaScript is weakly typed. It is perfectly normal for a variable to hold different values of the **same** data type during program execution.

JavaScript Data Types: Objects

- User-defined object
- Built-in Standard objects
  - Arrays
  - Dates
  - Number
  - Math
  - String

Objects: User-defined object

- In the real world, objects can be described by many data items. A person has a name, a gender, an age, etc.

Objects: User-defined object

```
<script type="application/javascript">
    //3. User-defined object
    var wudhichart = {
        firstName: "Wudhichart",
        lastName: "sawangphol",
        age: 30,
        university: "Mahidol",
        feesPaid: true
    };

    var aCandidate, aProperty;

    aCandidate = wudhichart;

    wudhichart.age = 40;
    console.log("Wudhichart is " + wudhichart.age + " years old.");
    console.log("Candidate is " + aCandidate.age + " years old.");

    wudhichart["feesPaid"] = false;
    console.log("Candidate up-to-date with fees: " + aCandidate["feesPaid"]);

    aCandidate["parking permit"] = "blue";
    console.log("Wudhichart's parking permit is " + wudhichart["parking permit"]);

    aProperty = "parking permit";
    console.log("Wudhichart's parking permit is " + wudhichart[aProperty]);
    aProperty = "age";
    console.log("Wudhichart is " + wudhichart[aProperty] + " years old.");
</script>
```

What is the output?

Objects: Built-in Standard objects

- Array
  - **Array** is a global object that is used in the construction of arrays; which are high-level, list-like objects.
- Date
  - **Date** represents a single moment in time. Date objects are based on a time value that is the number of milliseconds since 1 January, 1970 UTC.
- Number
  - **Number** object is a wrapper object allowing you to work with numerical values.
- Math
  - **Math** is a built-in object that has properties and methods for mathematical constants and functions.
- String
  - **String** object is a constructor for strings, or a sequence of characters.

Built-in Standard objects: Array

```
var Instructors = ['Wudhichart', 'Pawitra'];
var aArray = new Array(5);
```

```
console.log("The first instructor in array is " + Instructors[0]);
```

- Create array
- Access array
- Array Properties
  - Array.length
    - Reflects the number of elements in an array
- Methods
  - Array.isArray()
- Array instance methods
  - Arrayobj.pop()
    - Removes the last element from an array and returns that element
  - Arrayobj.push()
    - Adds one or more elements to the end of an array and returns the new length of the array
  - Arrayobj.indexOf()
    - Returns the first (least) index of an element within the array equal to the specified value

Built-in Standard objects: Date

- Create Date
  - var now = new Date();
    - This is the current date and time
  - var aDate1 = new Date(value);
    - Value is UNIX time.
  - var aDate2 = new Date(dateString);
    - dateString need to be in the format of ISO 8601, e.g., 2017-08-13T10:20:30 (**YYYY-MM-DDTHH:mm:ss.sss*Z***)
- Methods
  - Date.parse() , etc.
- Date instance methods
  - Dateobj.getDate()
  - Dateobj.getDay()
  - Dateobj.getFullYear() , etc.

Built-in Standard objects: Number

- Create Number
  - var aNum1 = new Number("10");
  - var aNum2 = Number("20");
- Methods
  - Number.isNaN()
  - Number.isInteger()
  - Number.parseInt()
  - Number.parseFloat() , etc.

- Number instance methods
  - Numberobj.toExponential()
  - Numberobj.toString()
  - Numberobj.valueOf() , etc.

Built-in Standard objects: Math

- There is no constructor for Math
- Properties
  - Math.PI
  - Math.SQRT2, etc.
- Methods
  - Math.pow(x, y)
  - Math.round(x)
  - Math.abs(x) , etc.

Built-in Standard objects: String

- Create String
  - Var str = “Hello World”;
  - Var str_obj = new String(“Hello World”);
- String concatination “+”
- Properties
  - String.length
- String instance methods
  - String.charAt()
  - String.split()
  - String.substring(), etc.

JavaScript Operators

- Arithmetic operators
- Comparison operators
- Conditional (ternary) operators
- Assignment operators
- Boolean (Logical) Operators
- Operator precedence

JavaScript Operators: Arithmetic operators

- The basic arithmetic operators (+, −, *, /, and %) are binary operators, because they each operate on two operands
- JavaScript provides the remainder operator, %, which yields the remainder after division
- Parentheses can be used to group expressions as in algebra.
- Operators in arithmetic expressions are applied in a precise sequence determined by the rules of precedence order:
  - Multiplication, division and remainder operations are applied first.
  - Addition and subtraction operations are applied next.
- When we say that operators are applied from left to right, we are referring to the associativity of the operators. Some operators associate from right to left.

JavaScript Operators: Arithmetic operators

- Summary of Arithmetic operators

JavaScript Operators	Arithmetic Operators	JavaScript Expression
Addition	+	x + y
Subtraction	-	x - y
Division	/	x / y
Multiplication	*	x * y
Remainder	%	x % y
Exponentiation	**	x ** y
Increment	++	++x or x++
Decrement	--	--x or x--
Unary negation	-	-x
Unary plus	+	+x

JavaScript Operators: Comparison operators

- JavaScript has both strict and type–converting comparisons.
- A **strict** comparison is only true if the type and content of operands are the same (e.g., ===)
- A **type–converting** comparison (commonly-used abstract comparison) convert the type of operands before comparing (e.g., ==)

JavaScript Operators: Comparison operators

- Summary of Comparison operators

JavaScript Operators	Comparison Operators	JavaScript Expression
Equality	==	x == y
Inequality	!=	x != y
Strict equality	===	x === y
Strict inequality	!==	x !== y
Greater than	>	x > y
Greater than or equal	>=	x >= y
Less than	<	x < y
Less than or equal	<=	x <= y

JavaScript Operators: Conditional (ternary) Operator

- This is operator is used as a shortcut for the if statement
- Syntax: *condition* ? *expr1* : *expr2*
- Example

```
var num1 = 1;
var num2 = 3;
var biggerNumber = num1 > num2 ? num1 : num2;
console.log("biggerNumber is " + biggerNumber);
```

What is the output?

JavaScript Operators: Assignment operators

- An **assignment operator** assigns a value of its right operand to its left operand.

JavaScript Operators	Assignment Operators	JavaScript Expression	Meaning
Assignment	=	x = y	x = y
Addition Assignment	+=	x += y	x = x + y
Subtraction Assignment	-=	x -= y	x = x - y
Division Assignment	/=	x /= y	x = x / y
Multiplication Assignment	*=	x *= y	x = x * y
Remainder Assignment	%=	x %= y	x = x % y
Exponentiation Assignment	**=	x **= y	x = x ** y

JavaScript Operators: Logical operators

- **Logical operators** are normally used with Boolean value

JavaScript Operators	Logical Operators	JavaScript Expression
Logical AND	&&	expr1 && expr2
Logical OR	||	expr1 || expr2
Logical NOT	!	! expr

JavaScript Operators: Logical operators

- Truth Table

		&&	||
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

	!
T	F
F	T

JavaScript Operators: Operator precedence

- Order as follows:
  - Grouping ()
  - Postfix increment (decrement)
  - Logical NOT, Unary Plus (Negation), Prefix increment (decrement) ordered from right-to-left
  - Exponentiation
  - Multiplication, Division, Remainder ordered from left-to-right
  - Addition, Subtraction ordered from left-to-right
  - Less Than (or Equal), Greater Than (or Equal) ordered from left-to-right
  - (Strict) Equality, (Strict) Inequality ordered from left-to-right
  - Logical AND
  - Logical OR
  - Assignment

Debugging JavaScript

- Follow the demo in the class

References

- http://developer.mozilla.org/en-US/