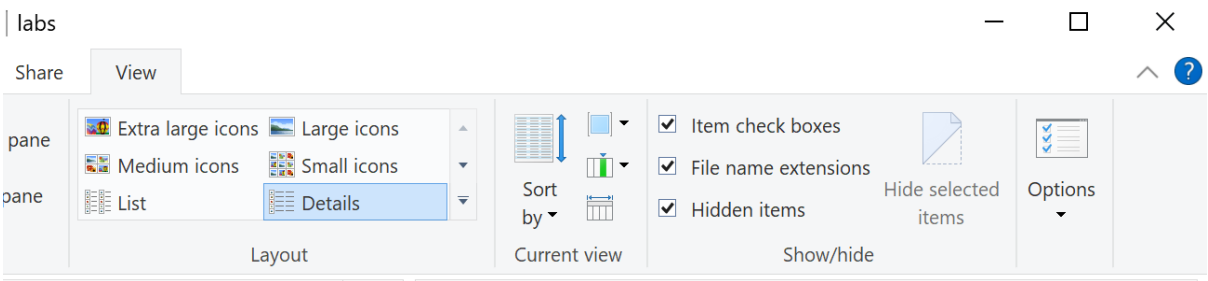
JavaScript
Intro for Web Development
Lab 1

1 Lab Setup

You should have received installation instructions on setting up Node as well as a suitable IDE (VS Code / Notepad++). You should also have received the source code for the lab sessions, which should have been downloaded from the course LMS or a GitHub repo.

We will need to be able to view file name extensions directly in order to be able to directly manipulate JavaScript source code files. If you are using Windows, make sure that you have checked the File name extensions in your File Explorer in order for us to be able to directly manipulate the file extensions when creating or modifying files.



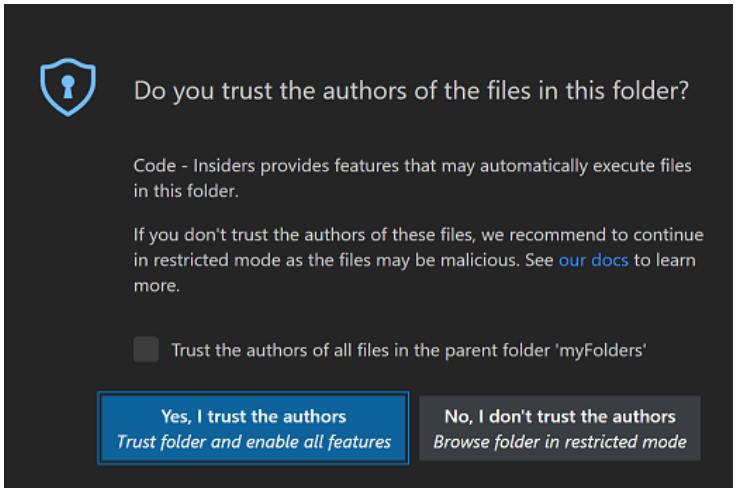
For MacOs:
[Show or hide file name extensions - Article 1]

Show or hide file name extensions - Article 2

2 Viewing JavaScript source code files

We will create and edit the source code files using Visual Studio Code (VS Code), which is an example of a popular and widely used IDE to work with JavaScript programs.

Start up VS Code. From the main menu, select File -> Open Folder, and then navigate to the `labcode/js` folder and open it. You may obtain a message similar to the following:



Click in the checkbox next to the sentence `Trust the authors of all files in the ……` and then click on the button `Yes, I trust the authors`.

You will see numerous files with the extension `.js` in the Explorer pane on the left. This are the JavaScript source code files.

You can then click on the any of the files with the extension `.js` in the explorer pane on the left. This will open up that file for editing in the main editor view. You can practice opening and closing these files in VS Code.

DON'T EDIT any of the files now ! If you want to write JavaScript code, please create a new program of your own: this will be demonstrated in an upcoming topic.

3 Executing a JavaScript program

There are a multitude of ways to execute a JavaScript program.

1. Using the Node.js runtime environment to directly execute a JavaScript program
2. Including JavaScript into a HTML document
3. Running JavaScript in Console panel of Chrome Developer Tools
4. Using an online JavaScript playground (25 JavaScript playgrounds)

We will use approach 1 for now and examine the others later on.

To run Node.js, we need to open a command prompt shell in Windows (or terminal in MacOs) and navigate to the folder containing the JavaScript source code files (`labcode`)

Working with MacOS
Opening terminal on Mac
Opening and using terminal on Mac
Navigating files and folders in Terminal
Video on navigating files and folders in Terminal

Working with Windows
Opening a command prompt shell directly from File Explorer
Different ways of opening the command prompt in Windows 10
Navigating to different directories in command prompt in Windows 10 - Article 1
Navigating to different directories in command prompt in Windows 10 - Article 2

In the `labcode` folder in the shell terminal, you can compile and execute the JavaScript source code file by simply typing:

```
node name-of-file
```

You will see the output from the program appear in the terminal immediately below the prompt.

You can choose to include the extension `*.js` as well if you wish to explicit. Node will automatically assume that extension if none is included.

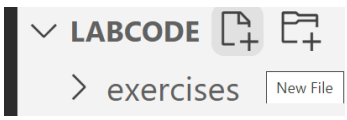
Experiment by selecting random source code files and running them in the way just demonstrated.

Some useful tips when working from the command prompt in Windows:

1. You can use the Tab key to auto complete the name of the files that you are typing.
2. You can use the up and down arrow keys to scroll through the history of commands that you have typed in the terminal.
3. Use the command `cls` to clear the screen when it becomes too cluttered with text

4 Creating and running a JavaScript program

Select the Add New File button in the Explorer Pane of VS Code.



and type in the name: `first-program.js`

The editor view in VS Code opens with a new tab showing a currently empty view. You will type in the source code for your program here.

Type (or copy and paste) the following statement below to anywhere inside the JavaScript program

```
console.log("Hello there !");
```

Save the file (File -> Save from main menu or use Ctrl-S as shortcut).

Compile and execute this program by typing into the terminal with:

```
node first-program
```

You should see the corresponding output in the terminal.

You can also create a new empty source code file directly in File Explorer (Windows) or Finder app (MacOS).

For Windows, right click in any empty space in the `labcode` folder, and select New -> Text Document.

Give the newly created file a suitable name (for e.g. `second-program`, `third-program`, `etc`) and **MAKE SURE** you include the extension *.js. You should be able to see the new file appear in the Explorer Pane in VS Code. Click on the file name to open it.

Type the same single statement shown earlier and save and run it with Node, for e.g.

```
node second-program
```

5 Basic output in JavaScript

You can provide text output with the `console.log` statement. The text content must be placed within either double quotes (for e.g. `"I like JavaScript"`) or single quotes (for e.g. `'I like JavaScript'`)

Add the following statements to the editor view for `first-program.js`

```
console.log('I like JavaScript');

console.log("I also like StarBucks");
```

Notice that both statements are terminated with a semicolon. This is not compulsory, but highly recommended in JavaScript programs.

Save the file and compile and execute in the usual way.

If you need to use a single quote within the text output, then the text should be enclosed with double quotes. Similarly, if you need to use double quotes within the text output, then the text should be enclosed with single quotes.

Add the following statements to the editor view for `first-program.js`

```
console.log("It's going to be a nice day today");

console.log('I think that "flying hard" is a funny phrase');
```

Save the file and compile and execute in the usual way.

If you need to add a line break so that some of the text output appears on a new line, you need to use the \ followed by a special escape character (in this case, \n).

Add the following statements to the editor view for `first-program.js`

```
console.log("StarBucks is a great company \n I want to work there forever");
```

Escape characters are used to tell JavaScript to output text in a certain way, or to include characters that are normally not allowed in a text output because of the way that JavaScript interprets them.

For e.g. if you want to add tabs between words, use `\t`

Add the following statements to the editor view for `first-program.js`

```
console.log("There \t is \t a \t tab \t between all those words");
```

If you want to use single quote within a text output that is also enclosed with single quotes you need to precede the single quote with a \. Similar comments apply as well to double quotes.

Add the following statements to the editor view for `first-program.js`

```
console.log('It\'s going to be a lousy day today');

console.log("I think that \"flying soft\" is a funny phrase");
```

If you do need to use single or double quotes in your text, the best is to enclose the text with quotes of the other type as explained earlier.

Save the file and compile and execute in the usual way.

List of escape characters in JavaScript

6 Comments in JavaScript

JavaScript supports single-line and block comments. A single-line comment starts with two forward-slash characters (//). A single-line comment makes all the text following the // on the same line into a comment.

At any empty space between all the `console.log` statements in the editor view for `first-program.js`, insert the following comments

```
// This is a great programming language

// This is a great sentence
```

Save the file and compile and execute in the usual way. Notice that the sentences after the // do not appear in the output.

Remove the // before the comment. Notice that the editor puts red wavy lines below the sentence words indicating a syntax error has been detected.

Save the file and compile and execute in the usual way. You will now get a syntax error flagged by the compiler with an output that looks similar to the following:

```
first-program.js:7
 This is a great sentence
      ^^
SyntaxError: Unexpected identifier
←[90m    at Object.compileFunction (node:vm:352:18)←[39m
```

Add back the // before the sentence, save the file and compile and execute in the usual way. This time the program should execute and produce output normally.

Another type of comment is a block comment. In this case, the comment itself can span multiple lines. At any empty space between all the `console.log` statements in the editor view for `first-program.js`, insert the following block comment:

```
/* This is a really really long comment
I wonder why I have so much time
to write a long comment like this
Must be not enough work to do at home */
```

Save the file and compile and execute in the usual way. Notice that the sentences between the /* and */ do not appear in the output, similar to the case of single line comment.
Remove the /* and */ . Once again, notice that the IDE flags a syntax error. Add the /* and */ back in again and save.

You can use VS Code to help add comments via Edit -> Toggle Line Comment, Toggle Block Comment

7 Declaring variables and storing values in them

File to use: `variables-basic.js`
Exercise: `variables-question.js`
Answer: `variables-answer.js`

Variables are named via identifiers. There are specific rules regarding how identifiers are created in JavaScript. Keywords are reserved words that have special meaning in the JavaScript programming language. They cannot be used for identifiers.

There are some basic rules regarding how variable names should be properly created in JavaScript. In particular, variable names should be descriptive and use camelCase.

8 Data types

File to use: `datatypes-basic.js`
Exercise: `datatypes-question.js`
Answer: `datatypes-answer.js`

Data types are used to classify data that a computer program processes. In general, there are 3 main data types:

1) String - represents all text content, human readable or otherwise, that is not meant to be used in mathematical operations
2) Number - represents all numbers, whole and with fractional digits. Also includes numbers from other base systems: octal, hexadecimal, binary, etc
3) Boolean - represents the binary nature of machine language. It only has two possible values: true (1) or false (0).

Data types in JavaScript

There are 7 primitive data types and 1 complex data type in JavaScript. The 3 most commonly used primitive data types is String, Boolean and number. The other data types are occasionally encountered, so you need to know what they mean as well.

9 Mathematical expressions with arithmetic operators

File to use: `expressions-basic.js`
Exercise: `expressions-question.js`
Answer: `expressions-answer.js`

You can perform standard numeric operations using the basic arithmetic operators in JavaScript

The general format of a mathematical expression is as follows:

variable = mathematical expression

The = is known as the assignment operator.

The result of the mathematical expression on the right hand side of the = operator is first evaluated. Then this result is stored into the variable on the left hand side of the = operator.

You can use the remainder (%) operator to get the remainder from an integer division. This is useful in helping to determining whether a given number is fully divisible by another number.

You can also perform concatenation operations on strings using the + operator

10 Unary and assignment operators

Files to use:
`unary-operators.js`
`assignment-operators.js`

Exercise: `unary-assignment-question.js`
Answer: `unary-assignment-answer.js`

Unary operators are frequently used as shortcut to perform specific mathematical expressions. In particular, the prefix and postfix increment and decrement operators are widely used.

Assignment operators are frequently used as shortcut for a standard arithmetic operator and assignment operator used together.

11 Comparison and logical operators

File to use: `comparison-logical.js`
Exercise: `comparison-logical-question.js`
Answer: `comparison-logical-answer.js`

We can use comparison and logical operators in expressions that produce Boolean values (true or false) as their result. This can be later used in conditional statements.

Each character in a string is mapped to a numeric code which is then translated into a binary sequence and stored in a memory location. This numeric code is known as the ASCII code or character set.
When using logical operators >, <, >=, <= with strings, we will compare the two strings character by character using the ASCII code for their respective matching character positions.