JavaScript Note

Releases:

ES6: biggest update, from then on, the language is called modern JavaScript with new updates every year

Linking:

Inline Script: **<script>** in the HTML file contains the JS contents

External Script: a separate file with .js as extension, **<script src=”path”>** links the file

Declaring Variable:

Variables declared by **let** are only available inside the block where they're defined

Variables declared by **var** are available throughout the function in which they're declared

let var1 = value1;	assign **value1** to the variable **var1**
var1 = value2;	reassign **var1** with **value2**
const var1 = value2;	declare a constant **var1** with **value2** as value
var var1 = value1;	declare a variable **var1** with **value1** as value
console.log(value1/expr1, …);	output the value to the weblog

the variable name must abide by certain syntax

can only contain letters, numbers, underscore, and the $ sign

not the same as reserved keywords

Data Types:

the values in JS are either an object or a primitive value

Dynamic Typing: data types are determined automatically by JS (note that the values have type, rather than variables)

Primitive:

1. Number: only floating-point numbers **Number()**
2. String: identified by single/double quotes **String()**
3. Boolean: **true** and **false**
4. Undefined: a variable defined but not assigned with value (**let var1;**)
5. Null: means empty value
6. Symbol: value that is unique (since ES2015)
7. BigInt: larger integers than the number type can hold (since ES2020)

typeof value returns the type of the **value**

Type Coersion:

let n = ‘1’ + 1; converts the latter **1** into a string, the plus sign is interpreted as the concatenation operator

Comment: **//comment** or **/*comment*/**

Operator Precedence:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence

Template String:

denoted by backticks (``), able to concatenate strings using the syntax

con_str = `str1 ${var1}`, the placeholder **${expr1}** would be replaced by the value of **expr1**

Decision Making: **if (condition) { command1 } else if { command2 } else { command3 }**

Falsy Values: values that would become false when being converted: **0, ‘’, undefined, null, NaN**

other values are called the truthy values

Equality Operator: === strict, true only if the 2 values are identical, no type coercion

in parallel, the strict inequality operator is **!==**

Logical Operator: and **&&**, or ||, not **!var1**

Expression: pieces of code that produce value

Statement: pieces of code that do not produce value

switch (var1) { without the **break** statement, the code carries on

case 'case1': command1;

break;

case 'case2': command2;

default: command3;}

Conditional Operator: **condition ? command1: command2;**

if the condition before the question mark returns **true**, then **command1** would be executed, otherwise **command2**

Strict Mode:

'use strict'; in the first line activates the strict mode

Function:

Function Declaration: can be called before the function is defined

function func_name(parameters) {commands; returns;}

Function Expression: can only be called after the function is defined

const func_name = function(parameters) {commands; returns;}

Arrow Function: a special form of function, the return is implicitly included

const func_name = (parameters) => {commands; returns;}

Array: denoted by square brackets **[ ]**, the elements are separated by commas

can be defined using **array_name = new Array(param1, …);**

indexing is also with square brackets, **array_name[index];**

.length: returns the number of elements contained in the array

.push(param): add the parameter **param** to the end of the array, returns the new length of the array

.unshift(param): add the parameter **param** to the beginning of the array

.pop(): remove the last element from the array, returns the popped element

.shift(): remove the first element from the array, returns the popped element

.indexOf(param): return the index of the **param** that is inside the array

.includes(param): return **true** if **param** is inside the array, otherwise **false**

Objects: indicated by curly braces, defining key-value pairs, each key is also called a property

obj = {key: pair, …}

indexing: **obj.key** or **obj['key']** returns the pair corresponding to **key**

updating: **obj.key = pair1;**

appending: **obj['key'] = pair;**

Method: any function attached to an object

func_name: function (params) { commands; returns;}

inside the object, keyword **this** can be used to refer to the object itself

Loop:

for (let rep = init_value; rep cond expr; rep++) {command;}

continue and **break** is equal as they are in C

while (condition) {commands;}

DOM: documented object model, structured representation of HTML documents, allowing being accessed by JS

document.querySelector('identifier') select the element named **identifier**, the same as is located in CSS

.textContent get the text of the selected element