



Web Crawler Practice

Web Design III - JavaScript Basic



Dr. Chun-Hsiang Chan

Department of Geography,
National Taiwan Normal University



Outline

- Why JavaScript?
- JavaScript Syntax
- JavaScript in HTML
- JavaScript Output
- JavaScript Variables
- JavaScript Operators
- JavaScript Data Types
- JavaScript Objects
- JavaScript Events
- JavaScript Strings
- JavaScript Numbers
- JavaScript Arrays
- JavaScript Math
- JavaScript Type Conversion
- JavaScript Flow Control
- Assignment

What is JavaScript?

- **JavaScript** is an essential programming language for web development, primarily because of its ability to add interactivity and dynamic behavior to websites. Here are some reasons why we need JavaScript:
- **Client-Side Interactivity:** JavaScript allows developers to create interactive elements on web pages. This includes features like form validation, sliders, accordions, pop-up dialogs, and more, enhancing user experience.
- **Dynamic Content:** With JavaScript, you can dynamically update content on a web page without needing to reload the entire page. This enables features like live updates, real-time data display (such as stock tickers or chat applications), and interactive maps.
- **Browser Compatibility:** JavaScript is supported by all major web browsers, making it a reliable choice for creating cross-browser compatible web applications.

What is JavaScript?

- **Enhanced User Experience:** By adding JavaScript, developers can create smoother and more engaging user experiences. Animations, transitions, and effects can be implemented to make interactions more visually appealing.
- **Cross-platform Development:** With the rise of frameworks like React Native, developers can use JavaScript to build not only web applications but also mobile applications for iOS and Android platforms, leveraging the same codebase.
- **Community and Libraries:** JavaScript has a vast ecosystem of libraries and frameworks (such as React, Angular, and Vue.js) that streamline development and provide pre-built solutions for common tasks. This accelerates development and enables developers to build complex applications more efficiently.

JS Syntax

- A computer program is a list of "instructions" to be "executed" by a computer. In a programming language, these programming instructions are called statements. A JavaScript program is a list of programming statements.
- Semicolons separate JavaScript statements.
- For best readability, programmers often like to avoid code lines longer than 80 characters.
- JavaScript statements can be grouped together in code blocks, inside curly brackets {...}. The purpose of code blocks is to define statements to be executed together.

JS Syntax

Demo: [w0301_initialize.html](#)

```
<p id="demo">JavaScript can change the style of an HTML element.</p>
<button type="button" onclick="document.getElementById('demo').innerHTML = '
  Hello JavaScript!'">JS say hi!</button>
<button type="button" onclick="document.getElementById('demo').style.fontSize='
  35px'">Bigger!</button>
<button type="button" onclick="document.getElementById('demo').style.display='
  none'">Disappear!</button>
<button type="button" onclick="document.getElementById('demo').style.display='
  block'">Show!</button>
```

In the browser, ...

0 JavaScript can change the style of an HTML element.

JS say hi! Bigger! Disappear! Show!

2 Hello JavaScript!

4 JS say hi! Bigger! Disappear! Show!

Hello JavaScript!

JS say hi! Bigger! Disappear! Show!

3 JS say hi! Bigger! Disappear! Show!

JavaScript in HTML

- JavaScript code is inserted between `<script>` and `</script>` tags in HTML.
- A JavaScript function is a block of JavaScript code that can be executed when "called" for. For example, a function can be called when an **event** occurs, like when the user clicks a button.
- Locations of JavaScript:
 - Head
 - Body
 - External JavaScript file

JavaScript in HTML

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>w0302 JS location</title>
  <script src="w0302_external_js.js"></script>
  <script type="text/javascript">
    function myFunction() {
      document.getElementById("demo2").innerHTML = "Wawawawa.";
    }
  </script>
</head>

<body>
  <p id="demo1">What!</p>
  <script>
    document.getElementById("demo1").innerHTML = "Hello JavaScript!";
  </script>
  <p id="demo2">[head] See changes here!</p>
  <button type="button" onclick="myFunction()">Click me!</button>
  <p id="demo3">[body] See here also!</p>
  <button type="button" onclick="myFunction2()">Click me also!</button>
  <script>
    function myFunction2() {
      document.getElementById("demo3").innerHTML = "Wahahaha!";
    }
  </script>
  <p id="demo4">[external] See here too!</p>
  <button type="button" onclick="myFunction3()">from External!</button>
</body>

function myFunction3() {
  document.getElementById("demo4").innerHTML = "external JS!";
}
```

Demo: [w0302_js_location.html](#) & [w0302_external_js.js](#)

Before

Hello JavaScript!

[head] See changes here!

Click me!

[body] See here also!

Click me also!

[external] See here too!

from External!

After

Hello JavaScript!

Wawawawa.

Click me!

Wahahaha!

Click me also!

external JS!

from External!

JavaScript Output

JavaScript can "display" data in different ways:

- Writing into an HTML element, using **innerHTML**.
To access an HTML element, JavaScript can use the `document.getElementById(id)` method. The `id` attribute defines the HTML element.
- Writing into the HTML output using **document.write()**.
For testing purposes, it is convenient to use `document.write()`.
Using `document.write()` after an HTML document is loaded, will delete all existing HTML.
The `document.write()` method should only be used for testing.

JavaScript Output

JavaScript can "display" data in different ways:

- Writing into an alert box, using `window.alert()`.
In JavaScript, the window object is the global scope object. This means that variables, properties, and methods by default belong to the window object.
- Writing into the browser console, using `console.log()`.
For debugging purposes, you can call the `console.log()` method in the browser to display data.

JavaScript Output

Demo: [w0303_output.html](#)

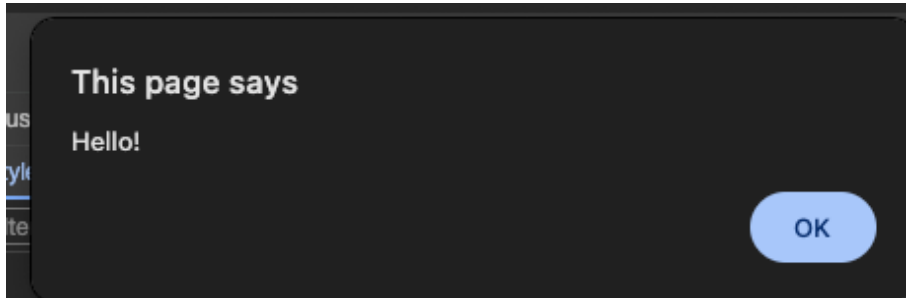
```
<div>Get your answer!<p id="demo1"></p>
</div>
<div>Refresh the whole page!<p id="demo2"></p>
</div>
<script>
  document.getElementById('demo1').innerHTML = 12 * 2;
  document.write(18 * 2);
  window.alert('Hello!');
  console.log(1000*23);
</script>
<button type="button" onclick="document.write('ALL GONE')">Refresh</button>
<button onclick="window.print()">Print this page</button>
```

JavaScript Output

Demo: [w0303_output.html](#)

In the browser, ...

0



1

Get your answer!

24

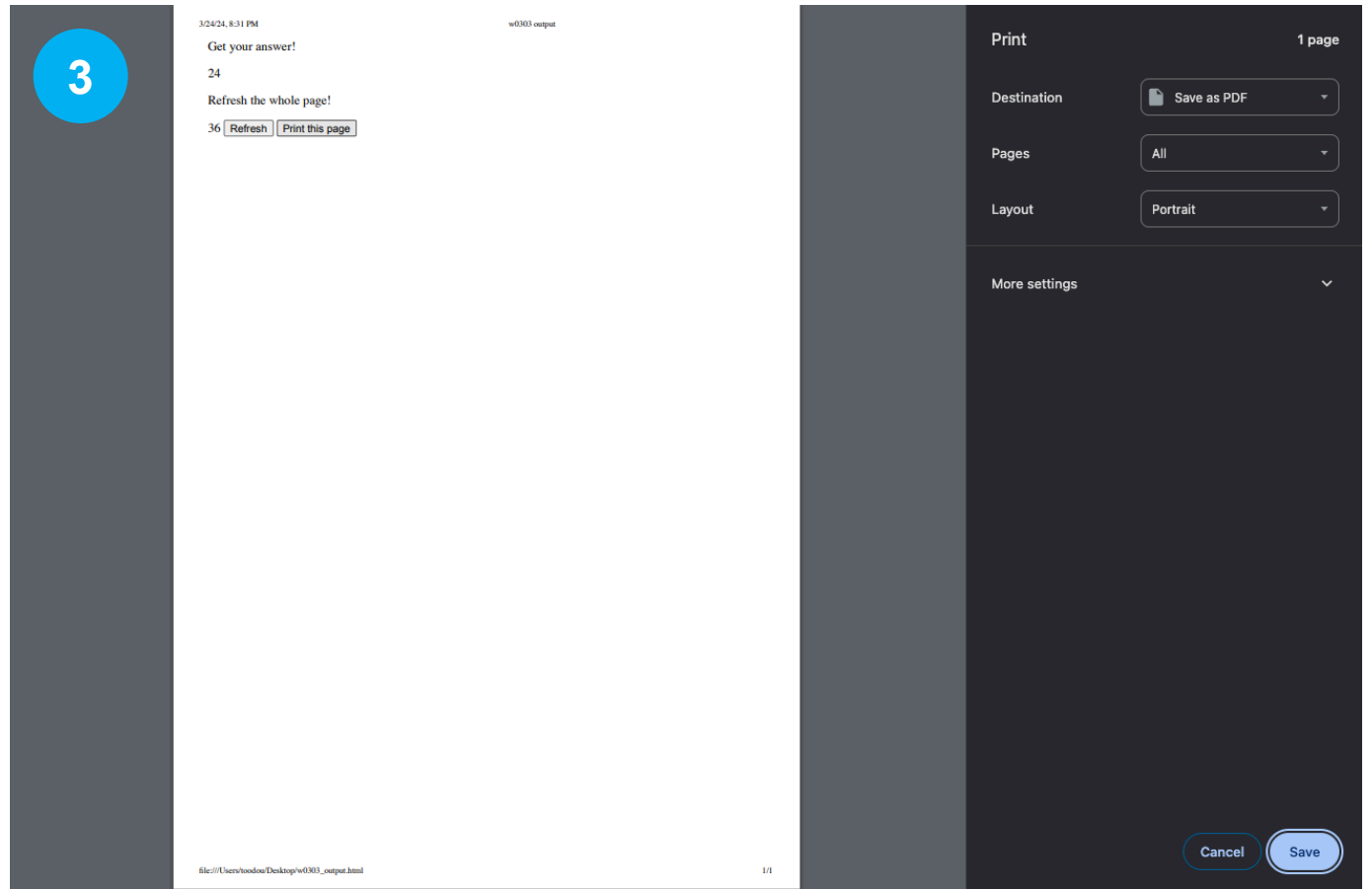
Refresh the whole page!

36 Refresh Print this page

2

~~ALL GONE~~

3



The image shows a screenshot of the Chrome DevTools interface. The top navigation bar includes tabs for Elements, Console, Sources, Network, Performance, Memory, Application, Security, Lighthouse, Recorder, and other tools. The Elements panel on the left shows the HTML structure of a page, with the `body` element selected. The Console panel in the center shows a function call `document.write('ALL GONE')` and its output, the string `23000`. The Styles panel on the right shows the default user agent styles for the `body` element, such as `display: block` and `margin: 8px`. A yellow box highlights the Console panel, and a yellow line connects it to a smaller inset of the Console panel at the bottom left, which also shows the `23000` output.

Think of an application for the console function!

JavaScript Variables

- The **var** keyword was used in all JavaScript code from 1995 to 2015.
- The **let** and **const** keywords were added to JavaScript in 2015.
- The **var** keyword should only be used in code written for older browsers.
- JavaScript Variables can be declared in 4 ways:
 - Automatically
 - Using **var**
 - Using **let**
 - Using **const**

JavaScript Variables

Demo: [w0304_variables.html](#)

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>w0304 variables</title>
  <script>
    /*declare variable*/
    b = "xyz"; // declare an automatically defined
    //var a = 'abc'; // declare a variable with 'var'
    let x, y, z; // declare block variables
    x = 5; // declare x
    y = 6; // declare z
    z = x + y; // compute x
    const m = 45; // declare a constant
  </script>
</head>

<body>
  <p id="demo"></p>
  <script>
    document.getElementById("demo").innerHTML = 'b = ' + b + '<br/>
      z = ' + z + '<br/> m = ' + m;
  </script>
</body>
```

In the browser, ...

```
b = xyz;
z = 11;
m = 45
```

JavaScript Variables

- When to use **var**, **let**, or **const**?
 1. Always declare variables
 2. Always use **const** if the value should not be changed
 3. Always use **const** if the type should not be changed (Arrays and Objects)
 4. Only use **let** if you can't use const
 5. Only use **var** if you MUST support old browsers.

JavaScript Variables

	Block Scope	Redeclare	Reassign
<code>var</code>	No	Yes	Yes
<code>let</code>	Yes	No	Yes
<code>const</code>	Yes	No	No

Lab Practice #01

Given several examples to prove the concept of the abovementioned table.

JavaScript Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

JavaScript Operators

Operator	Example	Same As
=	<code>x = y</code>	<code>x = y</code>
+=	<code>x += y</code>	<code>x = x + y</code>
-=	<code>x -= y</code>	<code>x = x - y</code>
*=	<code>x *= y</code>	<code>x = x * y</code>
/=	<code>x /= y</code>	<code>x = x / y</code>
%=	<code>x %= y</code>	<code>x = x % y</code>
**=	<code>x **= y</code>	<code>x = x ** y</code>

JavaScript Data Types

- JavaScript has 8 Datatypes
 1. String
 2. Number
 3. BigInt
 4. Boolean
 5. Undefined
 6. Null
 7. Symbol
 8. Object
- The Object Datatype
- The object data type can contain:
 1. An object
 2. An array
 3. A date

JavaScript Data Types

Demo: [w0305_data_type.html](#)

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1"
  >
  <title>w0305 data type</title>
  <script>
    // Numbers:
    let length = 16;
    let weight = 7.5;
    let a = 123e5; // 12300000
    let b = 123e-5; // 0.00123

    // Strings:
    let color = "Yellow";
    let lastName = "Johnson";

    // Booleans
    let x = true;
    let y = false;

    // Object:
    const person = { firstName: "John", lastName: "Doe" };

    // Array object:
    const cars = ["Saab", "Volvo", "BMW"];

    // Date object:
    const date = new Date("2022-03-25");
  </script>
</head>
<body>
  <div id="demo"></div>
  <script>
    document.getElementById("demo").innerHTML = date;
  </script>
</body>
```

In the browser, ...

Fri Mar 25 2022 08:00:00 GMT+0800 (Taipei Standard Time)

Using browser's console to observe other variables

```
> y
< false
> a
< 12300000
> b
< 0.00123
> person
< ▶ {firstName: 'John', lastName: 'Doe'}
> date
< Fri Mar 25 2022 08:00:00 GMT+0800 (Taipei Standard Time)
```

JavaScript Data Types

Lab Practice #02

Try some examples in the console or HTML.

For examples,

$a + b = ?$

$a + \text{date} = ?$


$a + x = ?$

$a + y = ?$

Give an explanation for the abovementioned examples.

JavaScript Objects

- In real life, a car is an **object**.

Object	Properties	Methods
	<code>car.name = Fiat</code> <code>car.model = 500</code> <code>car.weight = 850kg</code> <code>car.color = white</code>	<code>car.start()</code> <code>car.drive()</code> <code>car.brake()</code> <code>car.stop()</code>

- A car has **properties** like weight and color, and **methods** like start and stop. All cars have the same **properties**, but the property **values** differ from car to car. All cars have the same **methods**, but the methods are performed **at different times**.

JavaScript Objects

Demo: [w0306_objects.html](#)

```
<p id="demo"></p>
<script>
// Create an object:
const person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};

// Display some data from the object:
document.getElementById("demo").innerHTML =
  person.firstName + " is " + person.age + " years old.";
</script>
```

In the browser, ...

John is 50 years old.

JavaScript Events

- HTML events are "**things**" that happen to HTML elements. When JavaScript is used in HTML pages, JavaScript can "**react**" on these events. An HTML event can be something the browser does, or something a user does. Here are some examples of HTML events:
 - An HTML web page has finished loading
 - An HTML input field was changed
 - An HTML button was clicked
- Often, when events happen, you may want to do something.
- JavaScript lets you execute code when events are detected.
- HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

JavaScript Events

Demo: [w0307_events.html](#)

```
<p>Click the button to display the date.</p>
<button onclick="displayDate()">The time is?</button>
<script>
function displayDate() {
    document.getElementById("demo").innerHTML = Date();
}
</script>
<p id="demo"></p>
```

In the browser, ...

Click the button to display the date.

The time is?

Sun Mar 24 2024 22:24:27 GMT+0800 (Taipei Standard Time)

JavaScript Events

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

JavaScript Events

Lab Practice #03

Given three examples to perform your understanding of JS events.

```
<script>  
  ...;  
  ...;  
</script>
```

JavaScript Strings

- Strings are for **storing text**.
- Strings are written **with quotes**.
- Strings created with single or double quotes work the same → That is no difference between the two.
- You can use an **backslash escape character** to add quote into a string.
- The backslash escape character (`\`) turns special characters into string characters:

Code	Result	Description
<code>\'</code>	'	Single quote
<code>\"</code>	"	Double quote
<code>\\</code>	\	Backslash

JavaScript Strings

Demo: [w0308_strings.html](#)

```
<p id="demo1"></p>
<p id="demo2"></p>
<p id="demo3"></p>
<p id="demo4"></p>
<script>
let text1 = "Hello, I am a college student from National Taiwan
Normal Unviersity, which is one of the top unviersities in
Taiwan!\n The university's facilities are awesome, right?\n I
love my department -\n"Department of Geography!\n";
let text2, text3, text4;
text2 = "Hi";
text3 = "Hi";
text4 = new String("Hi");
let length = text1.length;
document.getElementById("demo1").innerHTML = (text2 == text3);
document.getElementById("demo2").innerHTML = (text3 == text4);
document.getElementById("demo3").innerHTML = (text2 === text3);
document.getElementById("demo4").innerHTML = (text3 === text4);
</script>
```

Lab Practice #04

1. Why we get these results?
2. What is the differences between “==” and “===”?

In the browser, ...

true

true

true

false

JavaScript Strings

- Javascript strings are primitive and immutable: All string methods produce a new string without altering the original string.

length	[...]	toUpperCase()	trimStart()	repeat()
charAt()	slice()	toLowerCase()	trimEnd()	replace()
charCodeAt()	substring()	concat()	padStart()	replaceAll()
at()	substr()	trim()	padEnd()	split()

JavaScript Strings

Demo: [w0309_strings2.html](#)

```
<p id="charAt">HELLO WORLD => text.charAt(0) =></p>
<p id="at">HELLO WORLD => text.at(2) =></p>
<p id="access">HELLO WORLD => text[2] =></p>
<p id="slice">Apple, Banana, Kiwi => text.slice(-12, -6) =></p>
<p id="trim">" Hello World! " => ws_txt1.trim() =></p>
<script>
let text = "HELLO WORLD";
let char = text.charAt(0);
let letter1 = text.at(2);
let letter2 = text[2];
let fruit = "Apple, Banana, Kiwi";
let part = fruit.slice(-12, -6);

let ws_txt1 = "    Hello World!    ";
let ws_txt2 = ws_txt1.trim();

document.getElementById("charAt").innerHTML = char;
document.getElementById("at").innerHTML = letter1;
document.getElementById("access").innerHTML = letter2;
document.getElementById("slice").innerHTML = part;
document.getElementById("trim").innerHTML = ws_txt2;
</script>
```

In the browser, ...

H

L

L

Banana

Hello World!

JavaScript Strings

<code>indexOf()</code>	<code>search()</code>	<code>matchAll()</code>	<code>startsWith()</code>
<code>lastIndexOf()</code>	<code>match()</code>	<code>includes()</code>	<code>endsWith()</code>

The `indexOf()` method returns the **index** (position) of the **first** occurrence of a string in a string, or it returns -1 if the string is not found.

The `lastIndexOf()` method returns the index of the last occurrence of a specified text in a string.

The `search()` method searches a string for a string (or a regular expression) and returns the position of the match.

The `match()` method returns an array containing the results of matching a string against a string (or a regular expression).

JavaScript Strings

indexOf()	search()	matchAll()	startsWith()
lastIndexOf()	match()	includes()	endsWith()

The **matchAll()** method returns an iterator containing the results of matching a string against a string (or a regular expression).

The **includes()** method returns true if a string contains a specified value. Otherwise it returns false.

The **startsWith()** method returns true if a string begins with a specified value. Otherwise it returns false:

The **endsWith()** method returns true if a string ends with a specified value. Otherwise it returns false:

JavaScript Strings

- **Template Strings** use back-ticks (``) rather than the quotes ("" or '') to define a string

Demo: [w0309_strings2.html](#)

```
<p id="demo"></p>
<script>
let firstName = "Chun-Hsiang";
let lastName = "Chan";

let welcome = `Welcome ${firstName}, ${lastName}!`;
document.getElementById("demo").innerHTML = welcome;
</script>
```

In the browser, ...

Welcome Chun-Hsiang, Chan!

JavaScript Strings

Lab Practice #05

Given a text as follows,

The last time Palestinians had the chance to vote for their national leaders was 18 years ago, and those elections sparked a civil war in Palestinian politics. That isn't a metaphor: Hamas, which won the elections, and Fatah, the biggest political faction, were shooting each other in Gaza's streets. The fighting ended with Hamas taking control of Gaza, and Fatah running the occupied West Bank through the Palestinian Authority (PA). It was a political cold war that lasted for a generation. That is where this journey starts.

Answer the following questions:

(1) How many **w** exist in the text?

(2) Where is the location of the first appearance of the **Gaza**?

JavaScript Numbers

Method	Description
toString()	Returns a number as a string
toExponential()	Returns a number written in exponential notation
toFixed()	Returns a number written with a number of decimals
toPrecision()	Returns a number written with a specified length
valueOf()	Returns a number as a number

JavaScript Numbers

Method	Description
Number()	Returns a number converted from its argument.
parseFloat()	Parses its argument and returns a floating point number
parseInt()	Parses its argument and returns a whole number

Method	Description
Number.isInteger()	Returns true if the argument is an integer
Number.isSafeInteger()	Returns true if the argument is a safe integer
Number.parseFloat()	Converts a string to a number
Number.parseInt()	Converts a string to a whole number

JavaScript Numbers

Property	Description
EPSILON	The difference between 1 and the smallest number > 1 .
MAX_VALUE	The largest number possible in JavaScript
MIN_VALUE	The smallest number possible in JavaScript
MAX_SAFE_INTEGER	The maximum safe integer ($2^{53} - 1$)
MIN_SAFE_INTEGER	The minimum safe integer $-(2^{53} - 1)$
POSITIVE_INFINITY	Infinity (returned on overflow)
NEGATIVE_INFINITY	Negative infinity (returned on overflow)
NaN	A "Not-a-Number" value

JavaScript Arrays

- An **array** is a particular variable that can hold more than one value.
- In fact, an array is a very powerful and useful data type, which is usually adopted for data analyses and storage.
- It can store strings and numbers.

JavaScript Arrays

Demo: [w0310_array.html](#)

```
<p id="demo"></p>
<script>
const cars = [
  "Saab",
  "Volvo",
  "BMW"
];
document.getElementById("demo").innerHTML = cars;
</script>
<p id="demo1"></p>
<script>
const cars1 = [];
cars1[0] = "Saab";
cars1[1] = "Volvo";
cars1[2] = "BMW";
document.getElementById("demo1").innerHTML = cars1;
</script>
<p id="demo2"></p>
<script>
const cars2 = new Array("Saab", "Volvo", "BMW");
document.getElementById("demo2").innerHTML = cars2;
</script>
```

In the browser, ...

Saab, Volvo, BMW

Saab, Volvo, BMW

Saab, Volvo, BMW

JavaScript Arrays

indexOf()	findIndex()	reverse()
lastIndexOf()	findLast()	toSorted()
includes()	findLastIndex()	toReverse()
find()	sort()	

JavaScript Math

- The JavaScript Math object allows you to perform mathematical tasks on numbers.

```
Math.E           // returns Euler's number
Math.PI          // returns PI
Math.SQRT2       // returns the square root of 2
Math.SQRT1_2     // returns the square root of 1/2
Math.LN2         // returns the natural logarithm of 2
Math.LN10        // returns the natural logarithm of 10
Math.LOG2E       // returns base 2 logarithm of E
Math.LOG10E      // returns base 10 logarithm of E
```

JavaScript Math

Math.round(x)	Returns x rounded to its nearest integer
Math.ceil(x)	Returns x rounded up to its nearest integer
Math.floor(x)	Returns x rounded down to its nearest integer
Math.trunc(x)	Returns the integer part of x
Math.sin()	Returns the sine value of input
Math.sqrt()	Returns the square root value of input
Math.abs()	Returns the absolute value of input
Math.min()	Returns the minimum value of input

JavaScript Type Conversion

Method	Description
Number()	Returns a number, converted from its argument
parseFloat()	Parses a string and returns a floating point number
parseInt()	Parses a string and returns an integer

Method	Description
toExponential()	Returns a string, with a number rounded and written using exponential notation.
toFixed()	Returns a string, with a number rounded and written with a specified number of decimals.
toPrecision()	Returns a string, with a number written with a specified length

JavaScript Type Conversion

Method	Description
<code>getDate()</code>	Get the day as a number (1-31)
<code>getDay()</code>	Get the weekday a number (0-6)
<code>getFullYear()</code>	Get the four digit year (yyyy)
<code>getHours()</code>	Get the hour (0-23)
<code>getMilliseconds()</code>	Get the milliseconds (0-999)
<code>getMinutes()</code>	Get the minutes (0-59)
<code>getMonth()</code>	Get the month (0-11)
<code>getSeconds()</code>	Get the seconds (0-59)
<code>getTime()</code>	Get the time (milliseconds since January 1, 1970)

JavaScript Functions

- A JavaScript function is a block of code designed to perform a particular task. A JavaScript function is executed when "something" invokes it (calls it).

```
function name(parameter1, parameter2, parameter3) {  
    // code to be executed  
}
```

Demo: [w0311_functions.html](#)

```
<p id="demo"></p>  
<script>  
function toCelsius(f) {  
    return (5 / 9) * (f - 32);  
}  
  
let value = toCelsius(77);  
document.getElementById("demo").innerHTML = value;  
</script>
```

JS Flow Control – Conditions

Conditional Statements

- Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this. In JavaScript we have the following conditional statements:
- Use **if** to specify a block of code to be executed, if a specified condition is true
- Use **else** to specify a block of code to be executed, if the same condition is false
- Use **else if** to specify a new condition to test, if the first condition is false
- Use **switch** to specify many alternative blocks of code to be executed

JS Flow Control – Conditions

```
// if else condition
if(expression){
    // code block
}else if{
    // code block
}else{
    // code block
}
```

```
// switch condition
switch(expression){
    case x:
        // code block
        break;
    case y:
        // code block
        break;
    default:
        // code block}
```

JS Flow Control – if else

Demo: [w0312_condition.html](#)

```
<p id="demo"></p>
<script>
const time = new Date().getHours();
let greeting;
if (time < 12) {
    greeting = "Good morning";
} else if (time < 16) {
    greeting = "Good day";
} else {
    greeting = "Good evening";
}
document.getElementById("demo").innerHTML = greeting;
</script>
```

JS Flow Control – Loops

```
// for loop
for(var i = 0; i<10; i++){
    // code block
}
```

Q: What is the difference between a for loop and a while loop?

```
// while loop
while (i < 10) {
    text += "The number is " + i;
    i++;
}
```


JS Flow Control – Loops

Demo: [w0313_loop.html](#)

```
<h3>for loop</h3>
<p id="demo"></p>
<script>
let text = "";

for (let i = 0; i < 10; i++) {
  text += "The number is " + i + "<br>";
}

document.getElementById("demo").innerHTML = text;
</script>
<h3>while loop</h3>
<p id="demo1"></p>
<script>
let text1 = "";
let j = 0;
while (j < 10) {
  text1 += "<br>The number is " + j;
  j++;
}

document.getElementById("demo1").innerHTML = text;
</script>
```

In the browser, ...

for loop

while loop

The number is 0

The number is 0

The number is 1

The number is 1

The number is 2

The number is 2

The number is 3

The number is 3

The number is 4

The number is 4

The number is 5

The number is 5

The number is 6

The number is 6

The number is 7

The number is 7

The number is 8

The number is 8

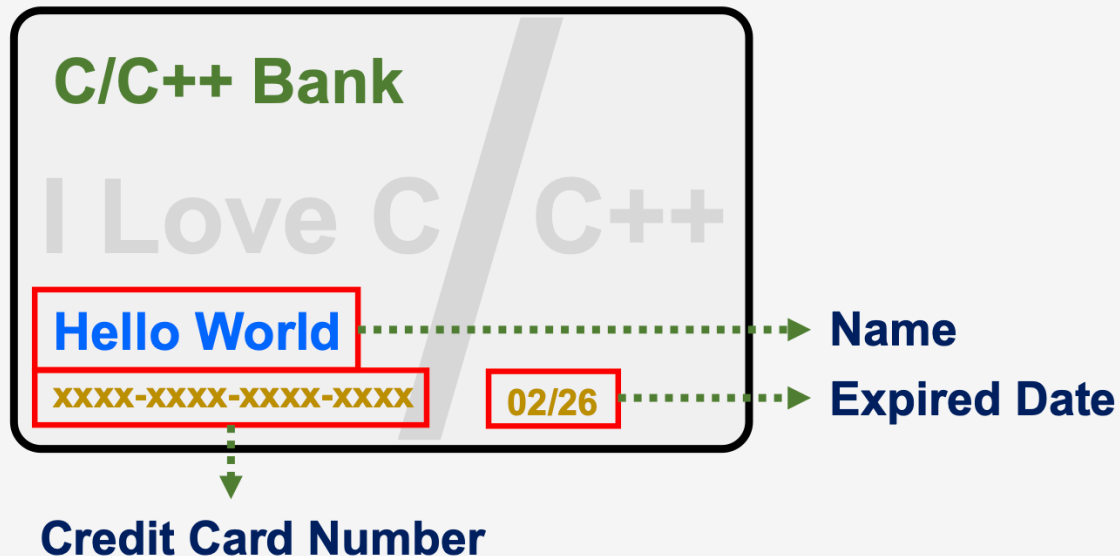
The number is 9

The number is 9

Assignment – Credit Card Validation Program

既然要知道怎麼計算出來的，首先你需要一組信用卡號碼。

Credit Card Number Generator: [LINK](#)



VISA

4024-0071-9977-6666
4485-7988-5649-8808
4556-8992-6892-5091
4024-0071-6142-5789
4716-5791-5076-2821

JCB

3088-6767-5562-6997
3337-7088-8512-1695
3112-8128-6453-8878
3088-6456-2277-6632
3096-4349-0381-0125

MasterCard

5509-8198-2415-1128
5554-8159-8205-8117
5318-7690-3591-6697
5595-6642-3010-8980
5298-4825-4483-4849

**AMERICAN
EXPRESS**

3756-1948-2665-662
3415-1949-7580-693
3406-9480-8889-710
3708-1818-3272-068
3446-6073-6603-313

Assignment – Credit Card Validation Program

1. 左邊數過來: 奇數的數字乘2
2. 如果大於9, 則可以減9或兩位數之和
3. 左邊數過來: 偶數的數字乘1
4. 全部加總除以10, 若整除則此信用卡號碼為真

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number	5	4	4	1	9	6	3	6	9	6	1	8	6	9	4	4
Weight	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
Check if > 9; then – 9	10	4	8	1	18	6	6	6	18	6	2	8	12	9	8	4
Result	1	4	8	1	9	6	6	6	9	6	2	8	3	9	8	4

Sum = 90 → $90 \div 10 = 9$ → 可整除, 故此信用卡號碼valid!

Assignment – Credit Card Validation Program

為了防止個資洩露，程式設計時要利用動態記憶體配置存取使用者之信用卡卡號，使用完畢之後，必須先將記憶體釋放，再將指標設為空指標(NULL Pointer)。

並且需要兩個基本防呆，還有一個發卡銀行判別功能：

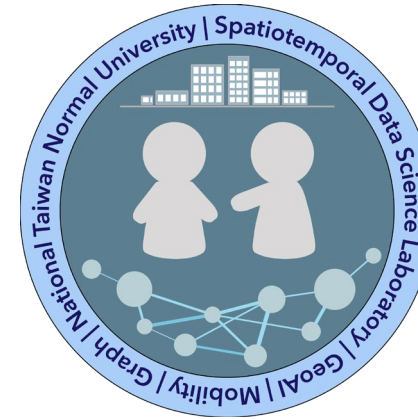
- (1) 長度必須為16
- (2) 起始碼必須為3、4或5
- (3) 依照前面所說，判定該卡由哪個發卡銀行所發行

Assignment – Credit Card Validation Program

• 本題測試資料為：

請截圖將判斷結果與依據寫出來！

- 4929-1961-5308-2660
- 2124-8732-4842-1232
- 3337-1461-8541-4447
- 3337-6130-8183-6067-1
- 4815-6246-4346-5738
- 5505-1519-3717-526
- 5461-4940-7016-1563



The End

Thank you for your attention!

Email: chchan@ntnu.edu.tw

Web: toodou.github.io

