

JavaScript

4b Functions

Systems and Web Development Workshop
2025 Spring

Dr. Jefferson Fong



JavaScript Based on C

- Many parts of **JavaScript** are **based on C**,
 - Such as functions, variables, conditions, loops and arrays.
 - The basic syntax is very similar to what you've learned in C.
- We will go through these slides quickly.
 - We will not repeat the Structure Programming in C course.
- If you need more info, you can study the slides and see sample codes
 - In iSpace, under folder “4To6 JS_Samples” \ “5 FuncVarCondi_samples”
 - 5aXXXX.html are sample codes for Functions.
 - Or in https://www.w3schools.com/js/js_functions.asp
https://www.w3schools.cn/js/js_functions.asp



Function Sample Code

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Functions</h2>

<p>This example calls a function which performs a
calculation, and returns the result:</p>

<p id="demo"></p>

<script>
function myFunction(p1, p2) {
    return p1 * p2;
}
document.getElementById("demo").innerHTML =
myFunction(4, 3);
</script>

</body>
</html>
```

Try it Yourself

[https://www.w3schools.com/js/js_functions.a
sp](https://www.w3schools.com/js/js_functions.asp)

JavaScript Functions

This example calls a function which performs a calculation, and returns the result:

12



JavaScript Function Syntax

Just like C

Skim

- A JavaScript function is defined with the
 - function keyword,
 - followed by a name,
 - followed by parentheses ().

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



- Function names can contain
 - letters, digits, underscores, and dollar signs (same rules as variables).



JavaScript Function Syntax

Skim

Just like C

- The code executed by the function is placed inside curly brackets { }.
- Function **parameters** are listed inside the parentheses () in the function definition.

function add(parameter1, parameter2, parameter3)

- Function **arguments** are the **values** received by the function when it is invoked, i.e. called.
- Inside the function, the arguments (the **parameters**) behave like **local variables**.



Parameters and Arguments

Skim

Just like C

- The parentheses may include parameter names separated by commas:

(parameter1, parameter2, ...)

(num1, num2, num3 ...)

(abc, xyz, some ...)

- Function **arguments** are the **values** received by the function when it is invoked.

- *add(3, 4) invokes function add(x, y)*



- parameter 1 and parameter2 behave as local variables.

(parameter1, parameter2, ...)



Function Return

Just like C

Skim

- When JavaScript reaches a **return statement**, the function will stop executing.
- Functions often computes a **return value** and returns value

```
var x = myFunction(4, 3);
```



myFunction is called, and the returned value will be stored in x.

```
function myFunction(a, b) {  
    return a * b;  
}
```





Function Sample Code

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Functions</h2>

<p>This example calls a function which performs a
calculation, and returns the result:</p>

<p id="demo"></p>

<script>
function myFunction(p1, p2) {
    return p1 * p2;
}
document.getElementById("demo").innerHTML =
myFunction(4, 3);
</script>

</body>
</html>
```

Try it Yourself

https://www.w3schools.com/js/js_functions.asp

JavaScript Functions

This example calls a function which performs a calculation, and returns the result:

12

Function_Sample.html



JavaScript Function Scope

- **Just like C**, the scope determines the accessibility (visibility) of variables inside and outside a function.
 - Think of the boundary of a functions as **one-way mirrors**
 - From inside, you can see outside.
 - From outside, you cannot see inside.
 - Variables defined inside a function are not accessible (visible) from outside the function.
 - Variables defined outside a function are accessible (visible) from inside the function.



Local JavaScript Variables

Skim

Just like C

- Variables declared within a JavaScript function, become LOCAL to the function.
- **Local variables** have local scope: They can **only be accessed within the function**.

```
// code here can not use carName
```

```
function myFunction() {  
    var carName = "Volvo";
```

```
    // code here can use carName
```

```
}
```



Local JavaScript Variables

<p>The local variable carName cannot be accessed from code outside the function:</p>

```
<p id="demo"></p>
<p id="demoFunction"></p>
```

```
<script>
myFunction(); //this invokes the function - calls myFunction
```

```
document.getElementById("demo").innerHTML =
"The type of carName is " + typeof carName;
```

```
function myFunction() {
  var carName = "Volvo";
  document.getElementById("demoFunction").innerHTML =
    "The type of carName is " + carName + "(inside the function)";
}
</script>
```

Try it Yourself

https://www.w3schools.com/js/js_functions.asp

- JS Assignment
- JS Data Types
- JS Functions
- JS Objects
- JS Events
- JS Strings
- JS String Methods
- JS String Search
- JS String Templates
- JS Numbers

Local Variables

Variables declared within a JavaScript function.

Local variables can only be accessed from within the function.

Example

The local variable carName cannot be accessed from code outside the function:

The type of carName is undefined

The type of carName is Volvo (inside the function)




Global JavaScript Variables

Skim

Just like C

- A variable declared outside a function, becomes **GLOBAL**.
- A global variable has **global scope**:
 - All scripts and functions on a web page can access it.

```
var carName = "Volvo";  
  
// code here can use carName  
  
function myFunction() {  
    // code here can use carName  
}
```





Global JavaScript Variables

```
<!DOCTYPE html>
<html>
<body>

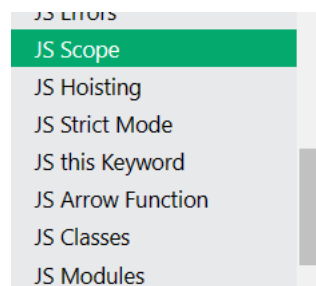
<p>A GLOBAL variable can be accessed from any script or function.</p>

<p id="demo"></p>

<script>
var carName = "Volvo";
myFunction();

function myFunction() {
    document.getElementById("demo").innerHTML =
        "I can display " + carName;
}
</script>

</body>
</html>
```



Global JavaScript Variables

A variable declared outside a function, becomes **GLOBAL**.

Example

A GLOBAL variable can be accessed from any script or function.

I can display Volvo

globalDemo.html



Skim

Why Functions?

Just like C

- code:
 - Define the code once, and use it many times.
- Use same code many times
 - with different arguments to produce different results.
- functions can be used “as variables”,
 - in all types of formulas, assignments, and calculations.



Functions Used as Variable Values

- Instead of using a variable to store the return value of a function like this

```
var x = toCelsius(77);  
var text = "The temperature is " + x + " Celsius";
```

- You can use the function directly, as a variable value:

```
var text = "The temperature is " + toCelsius(77) + " Celsius";
```



Functions Used as Variable Values

```
<h2>JavaScript Functions</h2>

<p>This example calls a function to convert from Fahrenheit to
Celsius:</p>
<p id="demo"></p>

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

- JS Data Types
- JS Functions
- JS Objects
- JS Events
- JS Strings
- JS String Methods
- JS String Search
- JS String Templates
- JS Numbers
- JS BigInt
- JS Number Methods
- JS Number Properties
- JS Arrays
- JS Array Methods

Why Functions?

You can reuse code: Define the code once, and use it many times.

You can use the same code many times with different arguments, to produce different results.

Example

Convert Fahrenheit to Celsius:

```
function toCelsius(fahrenheit) {
    return (5/9) * (fahrenheit-32);
}
```

JavaScript Functions

This example calls a function to convert from Fahrenheit to Celsius:

25

ToCelcius.html



Events and Functions, Text to Number

```
<body>
  <P>Top Number: <br/><input type="number" id="topValue" /></p>
  <P>Bottom Number: <br/><input type="number" id="bottomValue" /></p>
  <P><button type="button" onclick="add()">Add</button></p>
  <input type="number" id="result" disabled placeholder="Result"/>
</body>
<script>
  function add() {
    var topNum = Number(document.getElementById("topValue").value);
    var bottomNum = Number(document.getElementById("bottomValue").value);
    var result = (topNum + bottomNum);
    document.getElementById("result").value = result;
  }
</script>
```

Top Number:

Bottom Number:

- The value from the input box is a text value.
- Use the built-in function `Number()` to **convert** the **text value** into a **number**.



Task 2: Class Exercise – Calculator

My Calculator

Number One:

Number Two:

+	-	x	/
---	---	---	---

result

Change the code from last slide to be this calculator.



Submission

- Put your code for Task 1 clubRegister.html and Task2 myCalculator.html into the folder Lab4.
- Put the Lab 4 folder in stuweb.
- Zip the folder Lab4 and submit it to iSpace.
 - In the iSpace Online text, please put the correct links to stuweb for both tasks, so the TA can see your html pages by using that link in a browser. E.g.,

Online text

