

WEEK 12 – JAVASCRIPT FUNCTIONS

WEN-BIN JIAN

DEPARTMENT OF ELECTROPHYSICS, NATIONAL CHIAO TUNG UNIVERSITY

OUTLINE

- 1. Functions**
- 2. Value or Reference**
- 3. Variable & Functions**
- 4. Arguments & variable number of arguments of a Function**
- 5. No overloading in JavaScript**
6. Copying values or direct reference to a variable
7. Scope
8. Garbage collection
9. Skills for using functions

FUNCTIONS

- For other languages, you sometimes have two types of function calls, one is the “**subroutine**” and the other is “**function**”. For JavaScript, they are the same.
 - Subroutine: `function func(){ ... return;}`
 - Function: `function func(){ ... return value;}`
- Function definition is declared by
 - `function functionName(arg0, arg1, arg2, ...){`
 - `...}`
- After the definition, you can call the function by using “`functionName(par1, par2, ...)`”.
- The “**return**” in the function defines the leaving statement thus the statements after “**return**” will not be executed.

FUNCTIONS

- Without input parameters: `function func() { ... return value;}`
- With input parameters: `function func(par1, par2, ...) { ... use value of par1, par2, ... return;}`
- Parameters: value or reference?
 - values for common-used objects of Booleans, Numbers, and Strings
 - Reference for Array created by: `var arr = new Array(3);`
 - Reference for Objects created by: `var obj_name = new Object()`
- Nested functions: `function fmain(){ function fsub1 (){} function fsub2(){} }`

FUNCTIONS

- Variable functions:

- `var f = function(x){`
- `if (typeof(x)=="number") return (x*x+2*x+1);`
- `else return 0;`
- `};`

call it using "f(3)"

```
function func(a, b){return (a%b);}
var f = func;
f(11,3) → 2
```

- Array functions:

- `var arra = new Array(3);` here arra.length is 3
- `arra[0] = function(x){`
- `if (typeof(x)=="number") return (x*x+2*x+1);`
- `else return 0;`
- `};`

call it using "arra[0](5)"

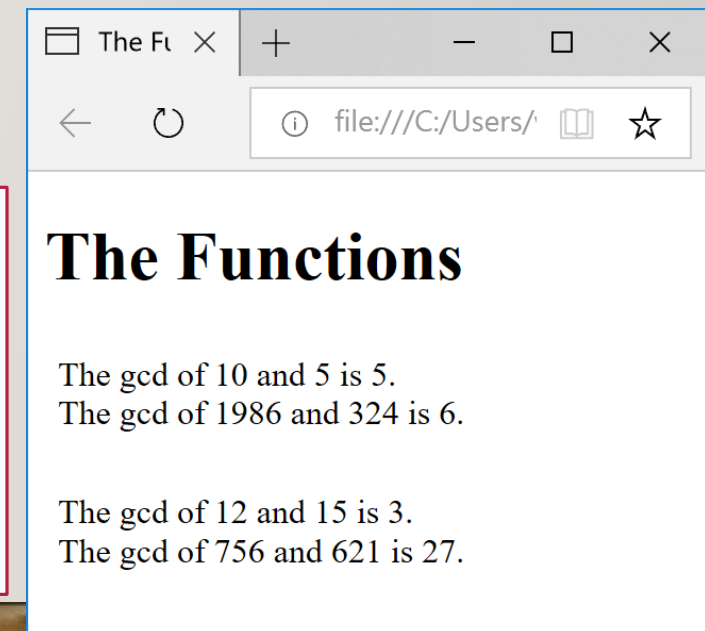
```
arra[1] = func;
arra[1](11,3) → 2
```

FUNCTIONS

- When do you use functions? When you write the same code for many times, you shall use the function call.
- For example, we want to solve similar problems for many times.
 - `a = 12; b = 15;`
 - `while (a % b != 0){`
 - `let c = a % b; a = b; b = c;}`
- You can collect them to a function.
 - `function gcd(a, b){ while (a % b != 0){`
 - `let c = a % b; a = b; b = c;}`

'let' usage:
let a = 20;
Declare a variable in a function. It's a local variable and disappears outside the function.

IC_WI201.html IC_WI201r.html



The screenshot shows a web browser window with the following content:

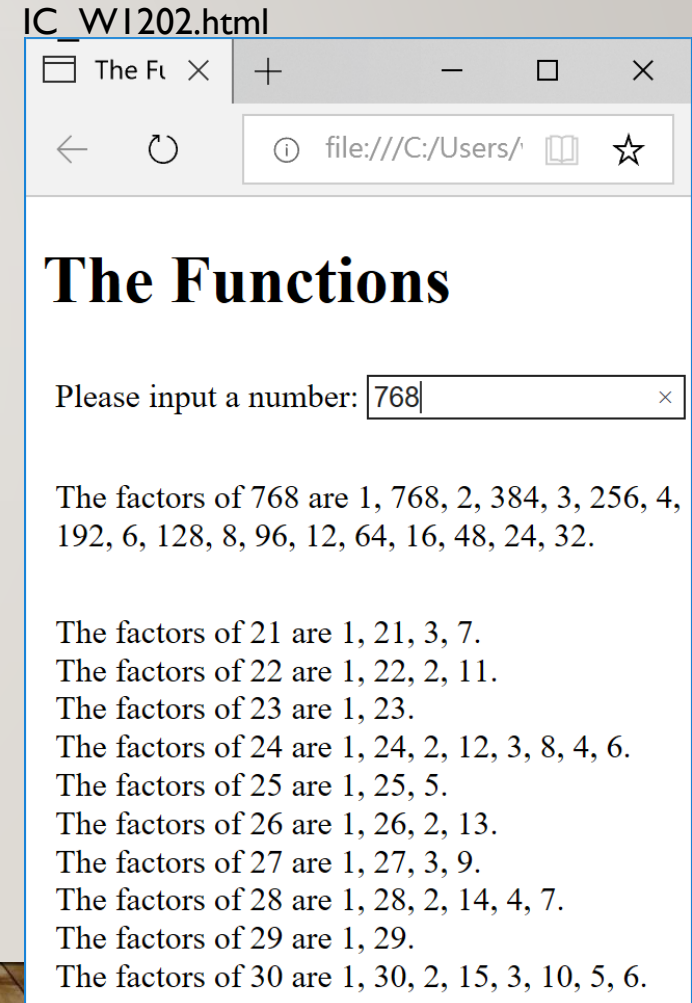
The Functions

The gcd of 10 and 5 is 5.
The gcd of 1986 and 324 is 6.

The gcd of 12 and 15 is 3.
The gcd of 756 and 621 is 27.

FUNCTIONS

- You can write your functions to solve your problems.
- For example, we want to find all the factors of a number.
- We may write a function and set up the user interface.
- Once we get the user's request, we can give an answer to him.



IC_WI202.html

The Factors of a Number

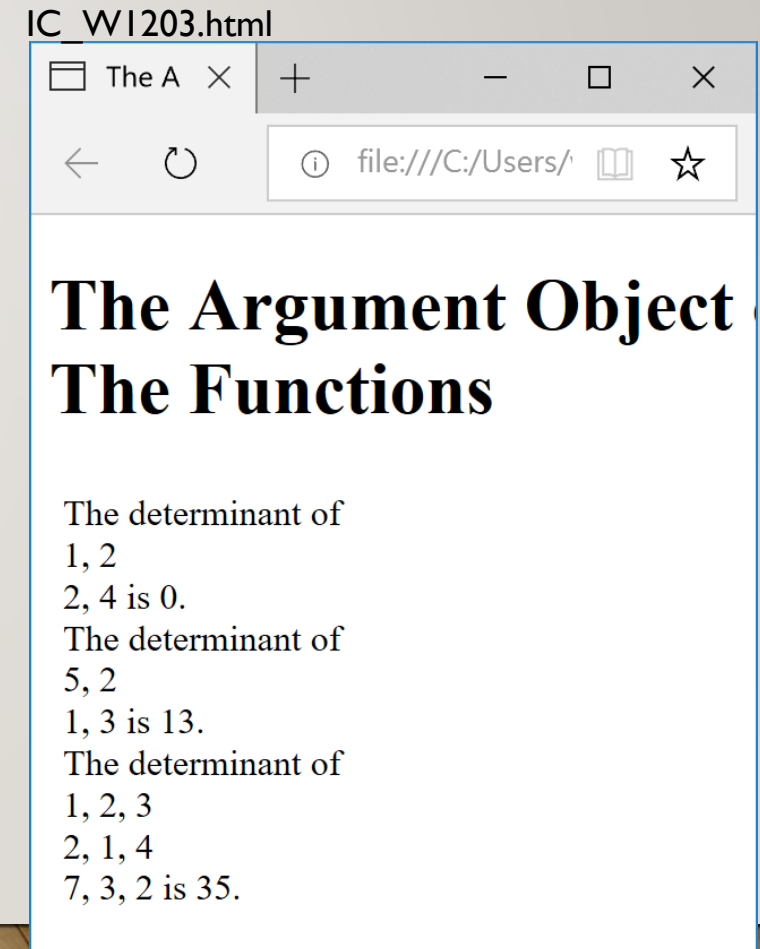
Please input a number:

The factors of 768 are 1, 768, 2, 384, 3, 256, 4, 192, 6, 128, 8, 96, 12, 64, 16, 48, 24, 32.

The factors of 21 are 1, 21, 3, 7.
The factors of 22 are 1, 22, 2, 11.
The factors of 23 are 1, 23.
The factors of 24 are 1, 24, 2, 12, 3, 8, 4, 6.
The factors of 25 are 1, 25, 5.
The factors of 26 are 1, 26, 2, 13.
The factors of 27 are 1, 27, 3, 9.
The factors of 28 are 1, 28, 2, 14, 4, 7.
The factors of 29 are 1, 29.
The factors of 30 are 1, 30, 2, 15, 3, 10, 5, 6.

ARGUMENTS

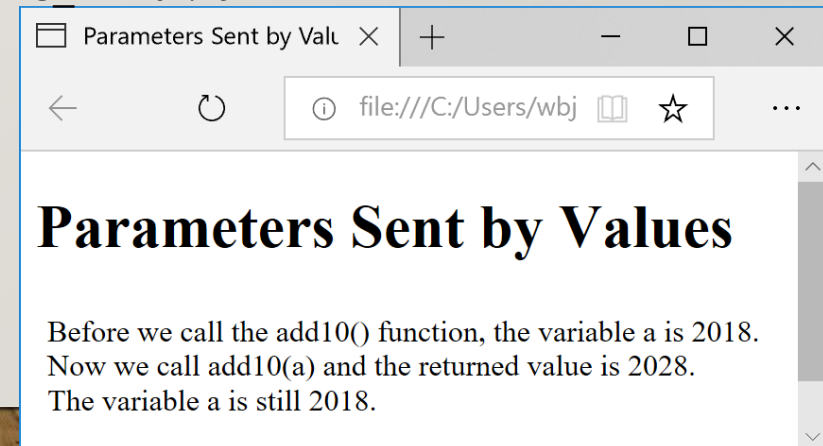
- The JavaScript processes the function arguments by using the **arguments** object.
- The **arguments** object is similar to array.
- You could send either one three parameters to a function even if you define the function to accept two arguments.
- The first parameter is **arguments[0]** and the second one is **arguments[1]**.
- The number of parameters sent to the function is recorded by **arguments.length**.



ARGUMENTS

- The parameters of nulls, Booleans, Numbers, and Strings are passed to functions through copying by values.
- The named arguments can be used in connection with the **arguments** object.
- You can change the value of the **arguments** object.
- For a function of two named arguments, if only one argument is passed to the function, the change to **arguments[1]** will not be reflected in the named argument.
- The named arguments and the **arguments** object do not use the same memory. They are kept in synchrony.

IC_WI204.html



The screenshot shows a web browser window with the title 'Parameters Sent by Values'. The address bar displays 'file:///C:/Users/wbj'. The main content area contains the following text:

Parameters Sent by Values

Before we call the add10() function, the variable a is 2018.
Now we call add10(a) and the returned value is 2028.
The variable a is still 2018.

NO OVERLOADING

- Functions can be overloaded in computer languages like C++ and Java but they cannot be overloaded in the JavaScript language.
- If two functions are defined with the same function name, the later one will be called.
- The C language typically use overloading to solve the call with different parameters.
 - `int abs (int n);`
 - `long int abs (long int n);`
- The JavaScript use the **arguments** object thus it already support the function called with different number and types of parameters.

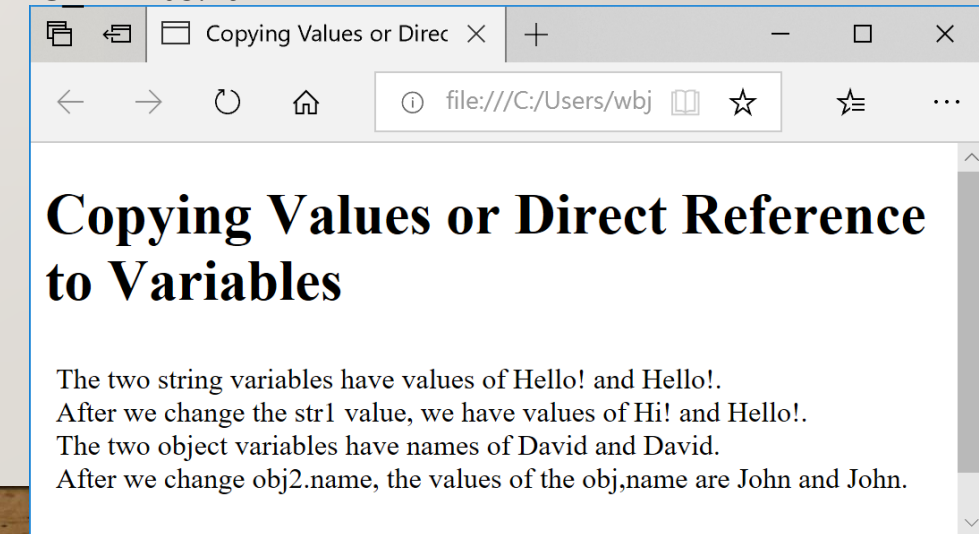
OUTLINE

1. Functions
2. Value or Reference
3. Variable & Functions
4. Arguments & variable number of arguments of a Function
5. No overloading in JavaScript
6. **Copying values or direct reference to a variable**
7. **Scope**
8. **Garbage collection**
9. **Skills for using functions**

COPYING VALUES OR DIRECT REFERENCE TO VARIABLES

- For these basic types including undefined, null, boolean, number, and string, the assignment requests the **copy of values**.
 - `var num1 = 5; var num2 = num1;`
 - `var nobj1 = null; var nobj2 = nobj1;`
- The above codes show the copying of values. The `num1` and `num2` are two different variables having different references (memory addresses).
- For most other object types, the assignment gives direct reference to variables.
 - `var obj1 = new Object(); var obj2 = obj1;`
 - `obj1.name = "John"; → obj2.name ?`

IC_WI205.html



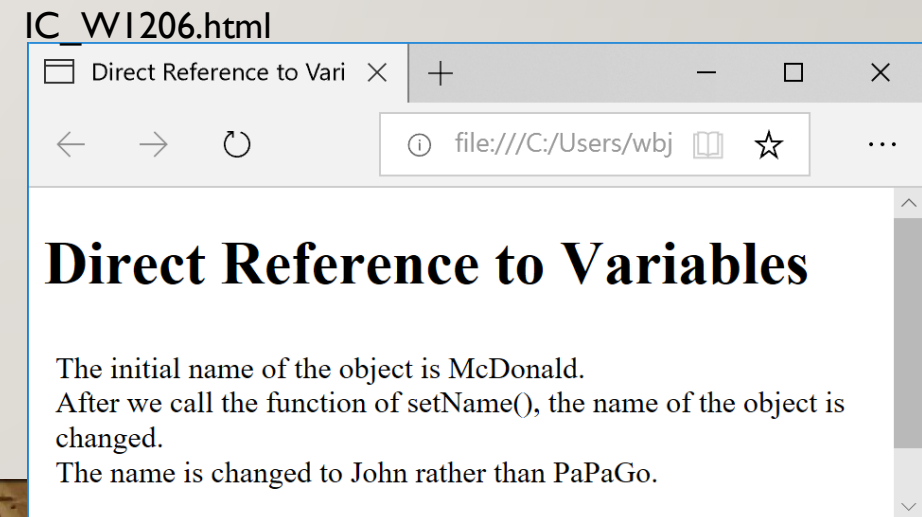
The screenshot shows a web browser window with the title 'IC_WI205.html'. The address bar shows the file path 'file:///C:/Users/wbj'. The main content area displays the title 'Copying Values or Direct Reference to Variables' in a large, bold font. Below the title, there is a paragraph of text explaining the difference between copying values and direct reference for primitive and object types.

Copying Values or Direct Reference to Variables

The two string variables have values of Hello! and Hello!.
After we change the str1 value, we have values of Hi! and Hello!.
The two object variables have names of David and David.
After we change obj2.name, the values of the obj.name are John and John.

COPYING VALUES OR DIRECT REFERENCE TO VARIABLES

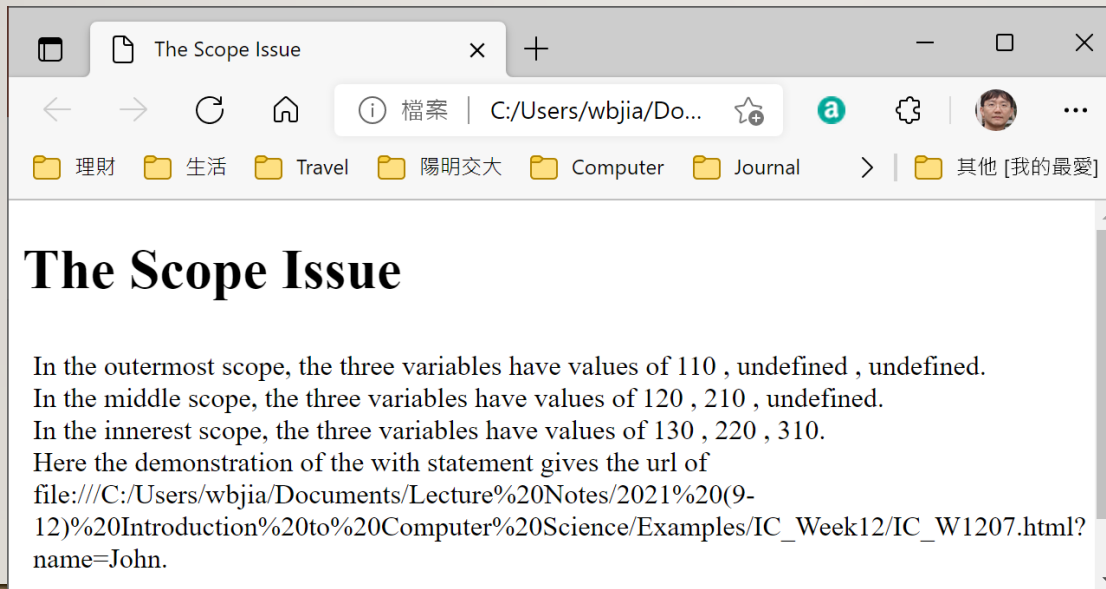
- If we send parameters to functions by using **basic types**, we copy values. The variable values outside the functions can not be changed inside the function.
- If you want to change the variable value inside the function, you have to use direct reference to variables. The way to use direct reference is to create an **object**.
- When a new object is created, it will be accompanied with a block of memory and a memory address.



SCOPE

- The scope of variables are used in functions. Thus, there are global variables and local variables when you use functions.
- Local variables in a function will be deleted when the execution of the function is finished.
- The scope is varied when using the statements of “with” or “catch” in a “try-catch” statement.

IC_W1207.html



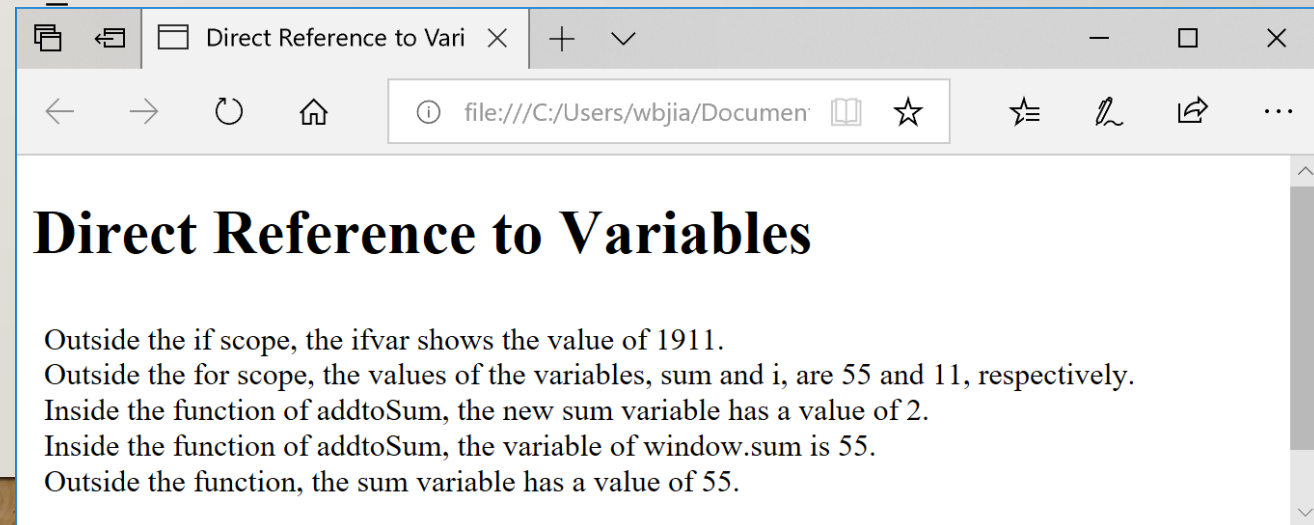
The screenshot shows a web browser window with the title 'The Scope Issue'. The address bar shows the file path 'C:/Users/wbjia/Do...'. The browser interface includes navigation buttons, a search bar, and a folder navigation bar with folders like '理財', '生活', 'Travel', '陽明交大', 'Computer', 'Journal', and '其他 [我的最愛]'. The main content area displays the title 'The Scope Issue' and the following text:

In the outermost scope, the three variables have values of 110 , undefined , undefined.
In the middle scope, the three variables have values of 120 , 210 , undefined.
In the innerest scope, the three variables have values of 130 , 220 , 310.
Here the demonstration of the with statement gives the url of
file:///C:/Users/wbjia/Documents/Lecture%20Notes/2021%20(9-12)%20Introduction%20to%20Computer%20Science/Examples/IC_Week12/IC_W1207.html?
name=John.

SCOPE

- The if and for statements do not have block-level scopes.
- If you forgot to declare variable in function, the variable will be **promoted to a global variable**.
- If you define a variable with the same name in a function, you will block your access to the global variable of the same name.
- You can call `window.variable_name` to access global variables inside your functions.

IC_WI208.html



The screenshot shows a web browser window with the title 'Direct Reference to Variables'. The address bar displays the file path 'file:///C:/Users/wbjia/Documen'. The main content area contains the following text:

Direct Reference to Variables

Outside the if scope, the ifvar shows the value of 1911.
Outside the for scope, the values of the variables, sum and i, are 55 and 11, respectively.
Inside the function of addtoSum, the new sum variable has a value of 2.
Inside the function of addtoSum, the variable of window.sum is 55.
Outside the function, the sum variable has a value of 55.

SPECIAL SKILLS FOR FUNCTIONS

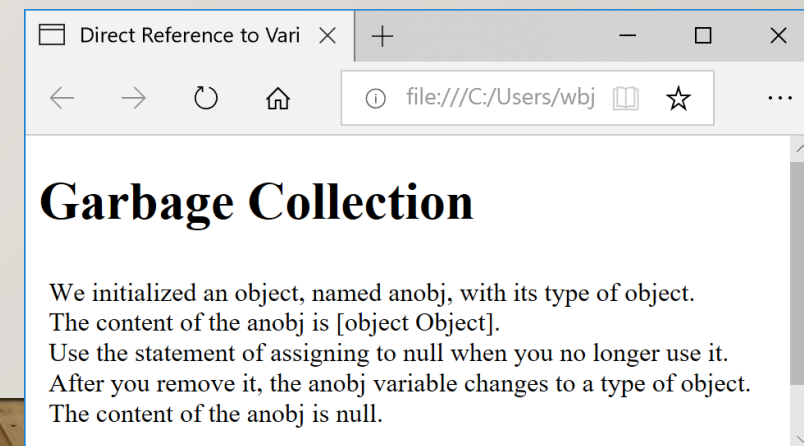
- Recursive functions – call themselves
 - `function fact(n){`
 - `if (typeof(n) != "number") return 0;`
 - `if (n == 1) return 1;`
 - `else return (n * fact(n-1));`
 - `}`

```
function fact(n){  
  if (typeof(n) != "number") return 0;  
  if (n == 1) return 1;  
  else return (n + fact(n-1));  
}
```


GARBAGE COLLECTION

- For those basic types of variables, like number and string, the garbage collections are done by the system (the browsers, like internet explorer, google chrome, ...).
- For other types of variables, where the reference are used in the copying process, the garbage collection have to be assisted by the programmers.
- Other types of variables are always **objects**. It's healthy to write a good destructor function when the programmers design a new object.
- The memory allocation happens in the process when you claim a **“new”** command for the object. The constructor function of the object will be called in the “new” process.
- For JavaScript, remember to use **“variable_name = null”** to call the destructor function of the object variables.

IC_WI209.html



The screenshot shows a web browser window with the title 'IC_WI209.html'. The address bar contains the file path 'file:///C:/Users/wbj'. The page content is as follows:

Garbage Collection

We initialized an object, named anobj, with its type of object.
The content of the anobj is [object Object].
Use the statement of assigning to null when you no longer use it.
After you remove it, the anobj variable changes to a type of object.
The content of the anobj is null.

EXERCISE

1. Please design html input fields to accept values of 2X2 matrix. Please calculate the determinant of the user's input of the 2X2 matrix. Please design 2 matrices (2X2) and calculate the multiplication of the two matrices.
2. Please design a textarea to receive the user's article and a button pressed for the calculation of the total number of characters without blanks. Calculate the repetition times of each character in the article.
3. Please design a textarea to receive the user's article and design a search button and a text field to find the position of the word specified in the searching field.

EXERCISE

- Provide two input fields for users to get an integer N ($N > 1$) as a denominator and an integer (M) for summation. Give an output of the total of $1/N + (1/N)^2 + \dots + (1/N)^M$.
- Provide one input field for users to get an integer N . Give the result of all prime numbers no bigger than N .
- Provide one input field for users to get an integer N . Print out the sum of $1/1! + 1/2! + \dots + 1/N!$.
- Provide a text field for users to get several positive floating-point numbers. Print out the arithmetic average value and the root mean square.