

GEM 1501 Problem Solving With Computers

Lecture 6:

Selected Topics of Java Script

Frank Stephan

# Summary of Previous Lecture

- Correctness of algorithms
  - Kinds of errors
  - Sources of errors
  - Correctness
- Efficiency of algorithms
  - Example
  - Big-O
  - Complexity of Algorithms and Problems

# Correctness of Algorithms

- Syntactical and Semantical Errors

```
function square(x);  
  { var y = x+x;  
    return(y); }
```

first line: semicolon after function declaration is syntax-error;  
second line: addition instead of multiplication is semantical error.

- Sources of errors:
  - Programmers make coding-errors;
  - Programmers misunderstand task;
  - Programmers misunderstand functionality of library-functions;
  - Used compilers or software is defective;
  - Used hardware defective.

# Efficiency of Algorithms

- Nontermination versus slowness

```
function fibonacci(n)
  { if (n<2) { return(n); }
    else { return(fibonacci(n-2)+fibonacci(n-1)); } }
```

- This programme terminates for all natural numbers  $n$  but it is very slow. Slowness and nontermination is not the same, but for the user similarly undesirable.
- Let  $\text{fibotime}(n)$  be the time needed by above function to compute  $\text{fibonacci}(n)$  measured by counting the number of times the function is called. This gives the bounds

$$2^{n/2} \leq \text{fibotime}(n) \leq 2^n.$$

So  $1125899906842624 \leq \text{fibotime}(100) \leq 1125899906842624^2$ .

- A better underlying algorithm would have  $\text{fibotime}(n) \in O(n^2)$ .

# Overview of Today's Lecture

- 12.00-12.30h: Midterm Test
- Summary of Previous Lecture
- Java Script Arrays
- Input and Output in Java Script
- Optimizing Speed

# Creating and Accessing Elements

```
var alpha = new Array(5);
    // makes array of five empty fields
var beta  = new Array(5,4,3,2);
    // makes array of four fields containing 5,4,3,2
var gamma = new Array(1); gamma[0] = 18;
    // makes array of one field containing 18
var delta = new Array(10); var k; for (k=0;k<10;k=k+1) { delta[k]=k; }
    // makes an array of ten fields cntng 0,1,2,3,4,5,6,7,8,9
delta[2] = delta[0]+2*delta[1];
    // accesses members of delta and updates delta[2] as above
function arrsum(arr)
{ var m = 0; var n = 0;
  while (n < arr.length) { m = m+arr[n]; n = n+1; }
  return(m); }
    // function summing up all members of the array
document.write("The sum of members of delta is "+arrsum(delta)+".");
    // sums the members of delta
```

# Moving Array Elements Around

Command	x	y	
var x = 0;	0	-	
var y = new Array(1,2,3);	0	(1,2,3)	
x = y[1];	2	(1,2,3)	<< y[0],y[1],y[2]
y.push(4,5,6,7,8);	2	(1,2,3,4,5,6,7,8)	
y.push(9);	2	(1,2,3,4,5,6,7,8,9)	
x = y.pop();	9	(1,2,3,4,5,6,7,8)	
x = y.shift();	1	(2,3,4,5,6,7,8)	
y.unshift(0,x);	1	(0,1,2,3,4,5,6,7,8)	
x = y.slice(2,4);	(2,3)	(0,1,2,3,4,5,6,7,8)	<< NOTE THE
x = y.splice(2,4);	(2,3,4,5)	(0,1,6,7,8)	<< DIFFERENCE
var z = x.concat(y);	x,y as above,	z is (2,3,4,5,0,1,6,7,8)	

Do not use "+" for the concatenation of arrays, "+" is only for strings  
After "z = x+y;", z is "2,3,4,50,1,6,7,8" So z is x.join()+y.join()

# Java Script Methods Save Work

- Print Array in Form "(2,5,6)"
- Self-programmed

```
function content(a)
{
  var u = "("; var k = 0;
  while (k < a.length)
  {
    u = u+a[k];
    if (k < a.length) { u += ","; }
    k++; }
  u = u+")";
  return(u); }

```

- Using join-method

```
function content(a)
{
  var u = a.join();
  return("(" + u + ")"); }

```

# Mixed Arrays

```
var a = new Array("The","number","is",2006,0);  
a[4] = new Array("This","is","a","subarray");
```

Resulting structure:

```
("The","number","is",2006,("This","is","a","subarray"))  
a.length is 5, a[4].length is 4  
a[4].push(4);  
a.join() is "The,number,is,2006,This,is,a,subarray,4"
```

The following operation produces a cycle:

```
a[4][4] = a;
```

This is possible but bad style.

```
a[4][4][4][4][3] is defined and has the value 2006.  
a.join() is "The,number,is,2006,This,is,a,subarray,"
```

# Sorting in Java Script

- Some sorting algorithm is implemented in Java Script.
- An element  $x$  is before  $y$  if, in the lexicographic order, the text `"" + x` comes before the text `"" + y`.
- The types of the array elements are preserved, only their ordering inside the array is adjusted.
- `a: (" an array" ," begin" ,2,4,5,8,(99,3),999,22," end" )`  
`a.sort(): (2,22,4,5,8,(99,3),999," an array" ," begin" ," end" )`

# Sorting Example

```
var a = new Array(6);  
a[0] = new Array("Carter", "Jimmy", 39);  
a[1] = new Array("Reagan", "Ronald", 40);  
a[2] = new Array("Bush", "George H W", 41);  
a[3] = new Array("Clinton", "Bill", 42);  
a[4] = new Array("Bush", "George W", 43);  
a[5] = new Array("Obama", "Barack", 44);  
var b = a.sort();
```

Variable a

```
(("Carter", "Jimmy", 39),  
 ("Reagan", "Ronald", 40),  
 ("Bush", "George H W", 41),  
 ("Clinton", "Bill", 42),  
 ("Bush", "George W", 43),  
 ("Obama", "Barack", 44))
```

Variable b

```
(("Bush", "George H W", 41),  
 ("Bush", "George W", 43),  
 ("Carter", "Jimmy", 39),  
 ("Clinton", "Bill", 42),  
 ("Obama", "Barack", 44),  
 ("Reagan", "Ronald", 40))
```

# Input and Output in Java Script

- Usage of current window (document)
- Opening of Additional windows
- Preprogrammed windows
- Forms instead of Writing and Prompting

# document.write

- Data can be converted into text and be written into the main window.

```
document.write("The variable n has the value "+n+".");
```

If n is not a variable but an array, the method "n.join()" will automatically be invoked to write the values. So the line

```
document.write("The variable n has the value "+n.join()+".");
```

gives the same output for arrays, but an error-message for normal variables.

- Problem: There is no "document.read" for getting values.
- Using html-commands permit to format texts.

# Using HTML commands to format texts

<code>&lt;br&gt;</code>	new line
<code>&lt;br&gt; &lt;br&gt;</code>	empty line plus new line (IE needs the space, Mozilla not)
<code>&lt;i&gt;10&lt;/i&gt;</code>	The number "10" will be written in italic
<code>&lt;b&gt;Downing&lt;/b&gt;</code>	The name "Downing" will be written in bold face
<code>&lt;u&gt;Street&lt;/u&gt;</code>	The word "Street" will be underlined
<code>&lt;pre&gt;...&lt;/pre&gt;</code>	Used for printing all characters with same space
<code>&amp;lt;br&amp;gt;</code>	Used to print " " instead of a new line

```
document.write("<h1>document.write</h1><ul><li>Data can ...  
    window.<dl><dd>document.write(...);</dl>If n ... the line<dl><dd>  
    document.write(...);</dl>gives the .... <li>Problem: There ...  
    <li>Using html-commands permits ... </ul>GEM 1501 ...");  
produces an output similar to the last slide.
```

<http://www.temple.edu/cs/web/codes.html>

# Modifying and Opening Windows

```
window.status = "If you think 'Done' is boring then change status";
    // Displays some text at the bottom line of the window below
    // the "document-part" of the window
window.location = "http://www.abc.net.au/news/offbeat/";
    // loads a funny webpage and aborts the current program.
ftext=window.open("http://www.sg","ftext");
    // Opens page "http://www.sg" in a new window with name "ftext"
ttext=window.open("", "ttext");
    // Empty file-name indicates that no file is used
ttext.document.write("This is a new window.<br>");
    // Writes something into this window
```

In general: window refers to the current window,

    window.document to the main part of it (all except bottom line)  
document.write(x); is the same as window.document.write(x);  
window.open("", "ttext"); opens a window whose parent is the current  
    window, therefore the "window." before the "open".

# Special Purpose Windows

```
window.alert("The value is "+u);  
    // sends a message-window to the user informing about some value.  
  
var a = window.prompt("Please enter the value for a");  
    // sends an input window and returns the value to variable a  
  
while (window.confirm("Run the loop? ")) { funcone(); functwo(); }  
    // Each time before running the body of the loop the user has to  
    // confirm this. The window.confirm-function returns false or true.  
  
http://www.comp.nus.edu.sg/~gem1501/buttonwindow.html  
    // Sample webpage with buttons and windows
```

# A Button

In the html-part of the page (outside scripts)

```
<input type="button"  
  value = "I am a button, please click me"  
  onClick = "window.alert('You succeeded to click');">
```

The `onClick` field contains a short Java Script program, may be just one line, which is activated when the user clicks on it.

The `value` is the text on the button shown to the user.

# Modifying Button Values

Button values can be modified, for example when used for games.

For this the button has to be placed into a form, forms and buttons are numbered and there can be several buttons per form.

Forms can also have other input modes.

```
<form>
<input type="button"
  value = "I am a button"
  onMouseOver = "document.forms[0].elements[0].value='Please click me';"
  onClick = "window.alert('You succeeded to click');">
</form>
```

Use different types of quotes for the quotes around parameter values and for the quotes to mark text inside these parameters.

Attention: forms and document.write have problems with each other.

# Example: Buttons and Windows

```
<html><head><script language="JavaScript"><!--
function buttonclicked()
    { if (window.confirm("Are you sure that you are ok?"))
        { document.forms[0].elements[0].value="Good"; }
      else { document.forms[0].elements[0].value="Too Bad"; } }
function square(x)
    { window.alert("The square of "+x+" is "+x*x); }
//--> </script> </head>

<body> <b>Click the buttons</b> <form>
<input type="button" value="Feeling" style="height:23;width:180;"
onClick="buttonclicked();"><br>
<input type="button" value="Squarenumber" style="height:23;width:180;"
onClick="square(window.prompt('Input a number',0));"><br>
<input type="button" value="Bottom Line" style="height:23;width:180;"
onClick="window.status='Last button has been clicked';">
</form></body></html>
```

# Other Types of Input with Forms

```
function telephonenumber()  
  { var t;  
    if (document.forms[4].elements[0].checked)  
      { t=document.forms[4].elements[1].value+" telephone number is ";  
        t=t+document.forms[4].elements[2].value; }  
    else  
      { t="if you do not want, no information is displayed"; }  
    window.alert(t); }
```

```
<form>
```

```
<input type="checkbox"> I like to display the phone number.<br>
```

```
<select> <option selected value="work">work telephone number</option>
```

```
<option value="home">home telephone number</option>
```

```
<option value="hand">hand telephone number</option> </select><br>
```

```
<input type="text" value="+65-6516-2759"><br>
```

```
<input type="button" value="Submit" onClick="telephonenumber();">
```

```
</form>
```

# Optimizing Speed

Certain commands can be used to abbreviate longer ones. This gives faster code, although it is less readable for those who do not know the abbreviations. Here some examples.

## Abbreviated Version

```
x += 8; y -= 3;
x *= 2+u; y /= 3+u; z %= 4;
v++;
w--;
```

## Long Version

```
x = x+8; y = y-3;
x = x*(2+u); y = y/(3+u); z %= 4;
v = v+1;
w = w-1;
```

```
var x; var y=0; var z=2500000;
x=1; while(x<=z) { y+=x++; }
x=0; while(x<z) { y+=++x; }
x=z; while(x>0) { y+=x--; }
```

```
var x; var y; var z; y=0; z=2500000;
x=1; while(x<=z) { y=y+x; x=x+1; }
x=0; while(x<z) { x=x+1; y=y+x; }
faster version of the above two loops
```

Less frequent access of variable makes browser faster.

<http://www.comp.nus.edu.sg/~gem1501/speed.html>

# In Two Weeks

- Growth of functions
- Computing versus Verifying Solutions
- NP-complete problems
- Is  $P$  equal  $NP$ ?
- Even worse problems