CS 116 – Action Script Strings

Elie Abi Chahine

Introduction

- In the previous chapter, we covered object oriented programming and gave examples on objects found in the Actionscript language.
- In this chapter we are going to cover the String object (class).
- The String class contains methods that let you work with text strings.
- Strings are sequences of characters.
- ActionScript 3.0 supports ASCII and Unicode characters.

Introduction

- In programming a string is a text value.

- The value can be a sequence of letters, numbers, or other characters strung together into a single value.

- For instance, this line of code creates a variable with the data type String and assigns a literal string value to that variable:

```
var sAlbumName:String = “3 for the money";
```

Creating a string

- You have multiple ways to create a string:

```
var str1:String = "hello";

var str2:String = 'hello';
```

- Now using the constructor:

```
var str1:String = new String("hello");

var str2:String = new String(str1);

var str3:String = new String(); /* creates an empty
                                   string “” */
```

Some Escape sequence characters

Escape Sequence	Character
\n	Newline
\r	Carriage return
\t	Tab
\unnnn	The Unicode character with the character code specified by the hexadecimal number *nnnn*; for example, \u263a is the smiley character.
\’	Single quotation mark
\”	Double quotation mark
\\	Single backslash character

Some Escape sequence characters

Example:

```
var str1:String = "Hello\nWorld";
trace(str1); /* Hello
             World */

var str2:String = "Hello\tWorld";
trace(str2); /* Hello   World */

var str3:String = "Hello \u263a";
trace(str3); /* Hello☺ */

var str4:String = "That's \"A-OK\"";
trace(str4); /* That’s “A-OK” */

var str5:String = 'That\'s "A-OK"';
trace(str5); /* That’s “A-OK” */
```

Properties

- As we mentioned in the OOP chapter, every object has attribute (properties and methods).

- The property that the String object has is ***length:int*** which is equal to the number of characters in the string

```
var str:String = "Hello";
trace(str.length); /* 5 */
```

- An empty string and a null string both have a length of 0, as the following example shows:

```
var str1:String = new String();
trace(str1.length); /* 0 */

var str2:String = "";
trace(str2.length); /* 0 */
```

Methods

- Also, a lot of methods are found inside the ***String*** object
- Using those methods wisely can make your life way easier
- Important things to do before using the method (function):
    - Read it’s description in the help
    - Know what kind of arguments you need to send to the function
    - Know what the function returns to you
    - Know if the function changes in the original object or returns a new object for you (sometimes it does both)
    - Go through the examples given in the help

Accessing characters

- You can examine individual characters in various positions in a string using the **charAt()** method and the **charCodeAt()** method, as in this example:

```
var str:String = "hello!";
for (var i:int = 0; i<str.length; i++)
{
      trace(str.charAt(i), "-", str.charCodeAt(i));
}
```

```
Output:   h - 104
          e - 101
          l - 108
          l - 108
          o - 111
          ! - 33
```

Concatenating Strings

(taking two strings and joining them sequentially into one)

- The String class includes a ***concat()*** method, which can be used as follows:

```
var str1:String = "Bonjour";

var str2:String = "from";

var str3:String = "Paris";

var str4:String = str1.concat(" ", str2, " ", str3);

trace(str4); /* Bonjour from Paris */
```

Concatenating Strings

(taking two strings and joining them sequentially into one)

- Concatenation can also happen using the **"+"** or **"+="** operators

```
var str1:String = "green";
var str2:String = "ish";
var str3:String = str1 + str2;
trace(str3); /* greenish */

var str4:String = "green";
str4 += "ish";
trace(str4); /* greenish */
```

Concatenating Strings

(taking two strings and joining them sequentially into one)

- If you use the + operator (or the += operator) with a String object and an object that is not a ***string***, ActionScript automatically converts the nonstring object to a String object in order to evaluate the expression, as shown in this example:

```
varstr:String = "Random = ";
vararea:Number = Math.random();
str = str + area;
trace(str); /* Random = 0.7126771118491888 */

var str2:String = "Average = ";
str2 += (5+7)/2;
trace(str2);  /* Average = 6 */
```

Uppercase & Lowercase

- The ***toLowerCase()*** method and the ***toUpperCase()*** method convert alphabetical characters in the string to lowercase and uppercase, respectively:

```
var str:String = "Dr. Bob Roberts, #9."
var  str2:String = str.toLowerCase();
trace(str2);  /* dr. bob roberts, #9. */
trace(str.toUpperCase()); /* DR. BOB ROBERTS, #9. */
```

- After these methods are executed, the source string remains unchanged. To transform the source string, use the following code:

```
str = str.toUpperCase();
```

Finding Substrings

- One of the most important thing is being able to find certain characters or substrings inside a string variable in order to use it or replace it by other values.

- As a simple example, the user enters his full address then we find in it the street address, area, country …

- The String object(class) contains methods for finding patterns in strings and for replacing found matches with replacement substrings.

- These methods will described in the following slides.

Substr & Substring

- The ***substr()*** and ***substring()*** methods are slightly different.

- Both:
  - Return a substring of a string.
  - Take two parameters.
  - Take as first parameter the position of the starting character in the given string.

- However, in the ***substr()*** method, the second parameter is the length of the substring to return, and in the ***substring()*** method, the second parameter is the position of the character at the end of the substring (which is not included in the returned string).

Substr & Substring

- This example shows the difference between these two methods:

```
var str:String = "Hello from Paris, Texas!!!";
trace(str.substr(11,15)); /* Paris, Texas!!! */
trace(str.substring(11,15)); /* Pari */
```

***Note:* It is up to you guys to do some tests.**

Example: What will happen if I send a position outside my array?

What will happen if I send a negative position?

Slice

- The ***slice()*** method functions similarly to the ***substring()*** method. When given two non-negative integers as parameters, it works exactly the same. However, the ***slice()*** method can take negative integers as parameters, in which case the character position is taken from the end of the string, as shown in the following example:

```
var str:String = "Hello from Paris, Texas!!!";
trace(str.slice(11,15)); /* output: Pari */
trace(str.slice(-3,-1)); /* output: !! */
trace(str.slice(-3,26)); /* output: !!! */
trace(str.slice(-3,str.length)); /* output: !!! */
trace(str.slice(-8,-3)); /* output: Texas */
```

***Note:* Do extra tests on it!!!!!**

IndexOf & LastIndexOf

- You can use the ***indexOf()*** and ***lastIndexOf()*** methods to locate matching substrings within a string, as the following example shows:

```
var str:String = "The moon, the stars, the sea, the land";
    trace(str.indexOf("the")); /* 10 */
```

***Note:* It is case sensitive**

- You can specify a second parameter to indicate the index position in the string from which to start the search, as follows:

```
var str:String = "The moon, the stars, the sea, the land“
trace(str.indexOf("the", 11)); /* 21 */
```

LastIndexOf

- The ***lastIndexOf()*** method finds the last occurrence of a substring in the string:

```
var str:String = “The moon, the stars, the sea, the land”;
trace(str.lastIndexOf("the")); /* 30 */
```

Note: Also case sensitive

- If you include a second parameter with the ***lastIndexOf()*** method, the search is conducted from that index position in the string working backward (from right to left):

```
var str:String = “The moon, the stars, the sea, the land”;
trace(str.lastIndexOf("the", 29));  /* 21 */
```

split

- You can use the ***split()*** method to create an array of substrings, which is divided based on a delimiter. For example, you can segment a comma-delimited or tab-delimited string into multiple strings.

- The first parameter is the delimiter you are looking for

- The second parameter, which is optional, defines the maximum size of the array that is returned.

```
var sQueryStr:String = "first=john&last=do&title=manager";
var aParams:Array = sQueryStr.split("&", 2);
trace(aParams[0]); /* first=john */
trace(aParams[1]); /* last=do */
```

Match , Search & Replace

- The String class includes the following methods for working with patterns in strings:
  - Use the ***match()*** and ***search()*** methods to locate substrings that match a pattern.
  - Use the ***replace()*** method to find substrings that match a pattern and replace them with a specified substring.
- At this point, you know how things are working and where to find explanations so, I’m going to leave it up to you to research and learn how to use them properly.

Comparing Strings

- You can use the following operators to compare strings:

< <= != == => >

- These operators can be used with conditional statements, such as if and while, as the following example shows:

```
var str1:String = "Apple";
var str2:String = "apple";
if (str1 < str2)
{
        trace("A < a");
}
```

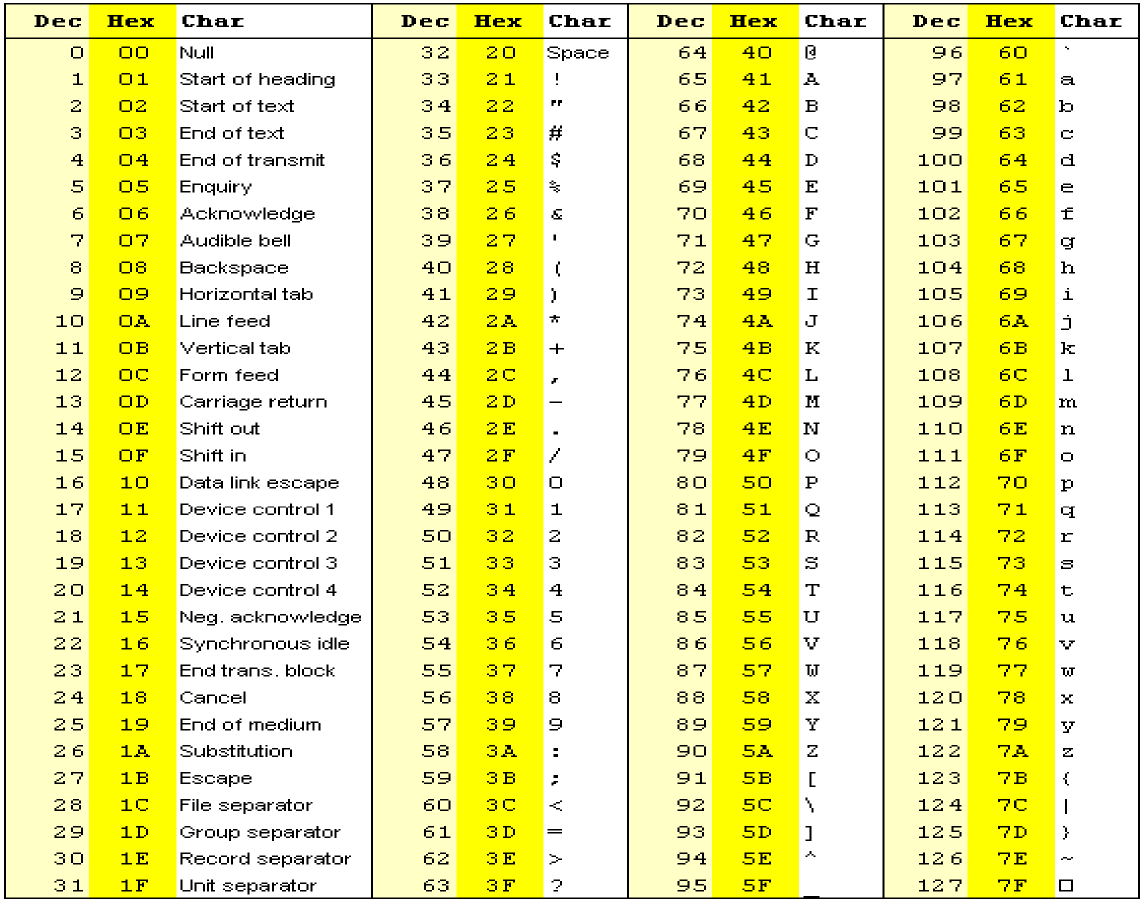
Comapring Strings

- When using these operators with strings, ActionScript considers the character code value of each character in the string, comparing characters from left to right, as in the following:

```
trace("A" < "B"); /* true */

trace("A" < "a"); /* true */

trace("Ab" < "az"); /* true */

trace("abc" < "abza"); /* true */

trace("abc" < "abcd"); /* true */
```

Comparing Strings

- Use the following table to know the Number associated with each ASCII character



Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Comparing Strings

- Use the **==** and **!=** operators to compare strings with each other and to compare strings with other types of objects, as the following example shows:

```
var str1:String = "1";
var str2:String = "1";
var str3:String = "2";

trace(str1 == str2); /* true */

trace(str1 == str3); /* false */

var total:uint = 1;
trace(str1 == total); /* true */
```

The End ☺