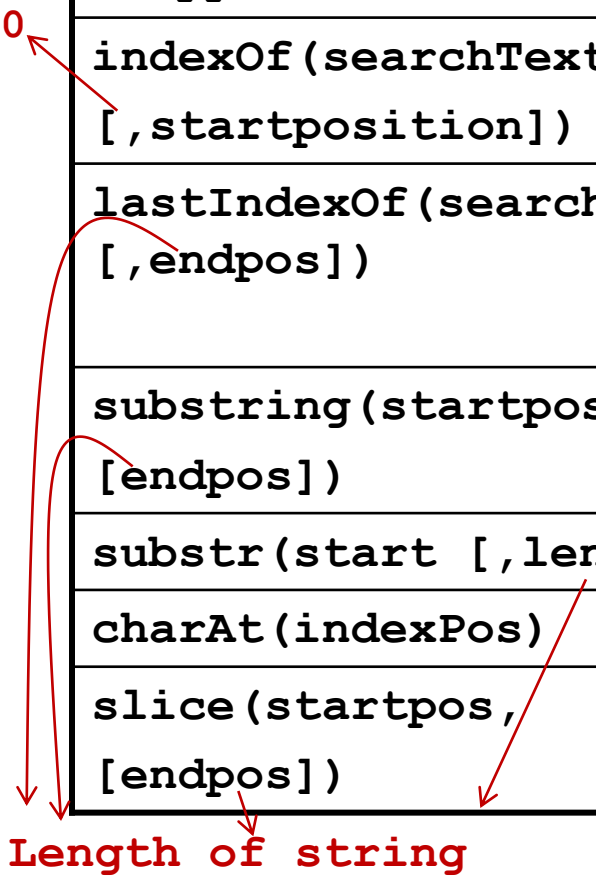
Predefined JavaScript classes

String

- Creating strings
  - `var str="abc";`
  - or `var str= new String("abc");`
- `String` class has several members like to get length of the string, search for a pattern etc.
- Regular expression can be used with string to find if a pattern matches.

String members

Methods	Examples	Results
length	"hi".length	2
toLowerCase()	"Hi".toLowerCase()	hi
toUpperCase()	"hi".toUpperCase()	HI
indexOf(searchText [,startposition])	"hello".indexOf("e",0) or "hello".indexOf("e")	1
lastIndexOf(searchstring [,endpos])	"hello".lastIndexOf("l","hello".length) or "hello".lastIndexOf("l")	3
substring(startpos, [endpos])	"hello".substring(1,3)	el
substr(start [,length])	"hello".substr(1,3)	ell
charAt(indexPos)	"hello".charAt(4)	o
slice(startpos, [endpos])	"hello".slice(3) or "hello".slice(3,5)	lo



Methods	Examples	Results
charCodeAt()	'A'.charCodeAt()	65
fromCharCode(n1,n2,..,nX)	String.fromCharCode(72,69,76,76,79)	HELLO
match(regexp)	"hello".match(/ll/)	ll
replace(regexp/substr, newstring)	"hello".replace(/ell/,"ipp")	hippo
search(regexp)	"hello".search(/ll/)	2
split(separator [, limit])	"hello".split("") "red:green:blue".split(":") "red:green:blue".split(":", 2)	h,e,l,l,o red,green,blue red,green

How can you increment characters?

More on regular expression coming up

Members that wrap HTML tags

Method	Example	HTML
anchor(aname)	"Part2".anchor("p2")	Part2
big()	"Welcome".big()	<BIG>Welcome</BIG>
blink()	"Highlights".blink()	<BLINK>Highlights</BLINK>
bold()	"Hello".bold()	<B>Hello</B>
italics()	"sky".italics()	<I>Sky</I>
link(url)	Yahoo.link(www.yahoo.com)	 Yahoo
small()	"Rights reserved".small()	Rights reserver
strike()	"strike".strike()	<strike>strike</strike>
sub()	"h"+"2".sub()+ "o"	h ₂o
sup()	"E=MC"+"2".sup()	E=MC²

Escape Sequence

- Used to insert Special Characters
- Suppose you want to print
  - This is very "special" mode
- And so you write

```
var txt="This is very "special" mode ";
document.write(txt);
```

  It throws an error!
- To solve this problem, you must place a backslash (\) before each double quote in

```
var txt="This is very \"special\" mode ";
document.write(txt);
```

HCL

Code	Outputs
\'	single quote
\"	double quote
\&	ampersand
\\	backslash
\n	new line
\r	carriage return
\t	tab
\b	backspace
\f	form feed

Break up a Code Line

- You can break up a code line within a text string with a backslash. The example below will be displayed properly:

- ```
  alert("red:green\
      :blue".split(":",2));
  ```
  ```

Regular Expression

- A regular expression (abbreviated to "regex”) is a set of pattern matching rules encoded in a string according to certain syntax rules.
- The syntax is complex but very powerful and allows lots of useful pattern matching than say simple wildcards *.

Creating regular expression

- **`RegExp`** class can be used to create regular expression strings
- Regular expression can also be created by putting them between **`/ /`**
  - **`var x=/ll/; or`**
  - **`var reg=new RegExp("ll");`**

    `alert("hello".match(reg));`

Both return **`ll`**

Patterns

- `\d` is to match any digit
- `\s` is to match any whitespace character
- `\w` is to match any word character (letters, digits, or "`_`" (underscore))
- `.` Means any character
- `[]`: If we need a match to be any one of the characters among a list. Range such as a-z can also be specified here
- `{}`: character{n} where n is an integer
- `[^]` Typing a caret after the opening square bracket will negate the pattern.
- Quantifiers:
  - `*` : Zero or more occurrences
  - `?` : Zero or one occurrence
  - `+`: One or more occurrences

Examples

1. Octal
   1. `"07679".match(/0[0-7]/)` → 07
   2. `"07679".match(/0[0-7]+/)` → 0767

2. Protocol
   1. `"1.2.3.6.7".match(/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/)` →1.2.3.6
   2. `"1.2.3.".match(/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/)` → null

3. Number between 1 to 999
   1. `"04867".match(/[1-9]\d{0,2}/;` →486
   2. `"4807".match(/[0-9]{0,2}/);` →480

More Examples

- Match an integer

```
var _x= prompt("enter a no","1");
if(_x.match(/[+-]? \d+/)==_x)
alert("ok");
else alert("not ok");
```

- Verifying validity of JavaScript variable name:

```
var _x= prompt("enter var","");
if(_x.match(/[a-zA-Z$_][a-zA-
  Z$_\d]*/)==_x)
alert("ok");
else
alert("not ok");
```

HCL