COMP 302 Lecture 4 12 September 2016

In JS, functions are values. They're not "special, magical things". Means you can use a function anywhere you can define a value.

see 4c302

We can assign and pass functions in variables.

see 4c302-1

Inside JS, everything is call-by-value.
Objects are passed by the value of their reference (similar to Java where ints as arguments aren't modified outside the function, but Arrays and Objects are).

Some extra functional issues → ECMA-versions.

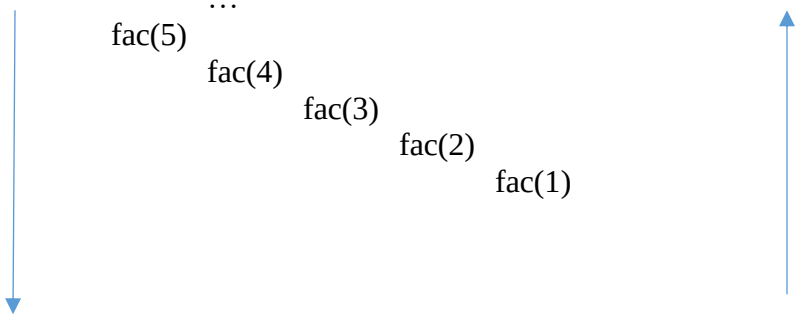
- Default parameters : function foo(a, b = 3) {}
- "Rest" parameters : function foo(a, b, … others) {}
- "Arrow" form : instead of function(a,b) {},
  (a,b) => {}
- Generator functions : function* foo() {yield <value>}

JS as a Functional Language

- Functions are "1st class citizens/values"
  - Means you have as much freedom with functions as you have with other values. Functions are just like integers, or Strings, or… Can create, store, pass, return (…) functions just like other primitive data types.
  - Data → rely on argument-parameter binding
    - Immutable once create/binded

see 4c302-2

fac(5)
= fac(4) * 5
= fac(3) * 4 * 5
= fac(2) * 3 * 4 * 5
= fac(1) * 2 * 3 * 4 * 5
…



fac(0)

recursion

computation

see 4c302-3

"tail" recursion can be more efficient

see 4c302-4

Notice that prefixSum and fac look similar

see 4c302-5

<name>	prefixSum	fac
<limit>	0	0
<base>	0	1
<combine>	+ (or plus)	*
<reduce>	-1 (or decrement)	-1