COMP 302 Lecture 3 09 September 2016

Objects

var x = new Object(); *or* x = {};
x.foo = "hello";
x.foo2 = 7;
x.foo3 = new Object();

Foo	"hello"
Foo2	7
Foo3	Object

x = {foo: "hello", foo2: 7, foo3: {}};
x = {"!*%++-": 7};
delete x.foo2; //some properties we can't delete from some things that are preset

Arrays

x = new Array();
x = []; //Arrays are special Objects where the keys are indices are integers starting at 0
x[0] = 7; x[1] = "hello";
x.length;
x[x.length] : "more"
x.foo = "hello";

see 3c302.txt

x.push(7);
x.pop(); //stack
or x.shift(7);
x.unshift(7); //queue
x.splice(…) //to remove things at arbitrary places

Undefined
This is what is displayed when we give an array that exists but without value.
Null
Defined, but has a null value.

JS has *null* and *undefined*.

Null is technically speaking an object type. The difference is that something that is null is still defined. Both map to false.

Equality
== "loose equality" or "value equality", will try to convert things on either side to a same type. Will try to see if values are the same, without really thinking whether they're 'binarily' the same.

0 == false? "" == 0?

see 3c302-1.txt

To avoid that, we can use
=== !== “strict equality”
“abc” === “abc”
“abc” !== “abc”

Control Flow
if (…) {}
else {}

see 3c302-2.txt

Switch
see 3c302-3.txt

Loops
while, do, for
can declare var inside
for (var i = 0; i < 11; i++) {} //i is declared inside the function, not the for only
for (var x in y) {document.write(y[x]);

Exception-handling
try
{
}
catch (e) //catches everything, unlike Java
{
}
finally
{
}
//throw exists as well

Functions

Functional languages

- More privileged than C/Java
- Define – most languages
- Store functions in variables, pass them as parameters, return from functions
  - Define them dynamically
- Convert to from/string → reflection

Ways to define functions
1. Function doubler (x) { return x + x;} //see 3c302-4.txt
2. var doubler = function(x) { return x + x; }
3. var doubler = new Function(“x”, “return x + x;”);
   doubler.toStr()