CPSC 216 - JavaScript

JavaScript is a full-featured, object-oriented programming language that allows us to implement a LOT of new and cool features in our webpages / websites.

It can make our webpages dynamic, as opposed to static. It enables a much higher degree of interaction and engagement with users - this is a good thing!

Review of Basic Programming:

1. Variables ✓
2. Arithmetic ✓
3. Conditionals ✓
4. Loops ✓
5. Functions ✓

You should

6. Scope / Globals ✓

7. Arrays ← } complex data types

8. Objects ← }

9. Maps ←

10. Data Types — some (primitive)

11. Exception Handling

} have seen most/all of this in CPSC 140/150 in Python!

→ functionality is the same in most programming languages... just slightly different syntax!

e.g. French vs. Italian vs. German vs. English

See index-review.html in Chapter 6, and the comments there!!

Loops:

Principle Idea:

"If you know how many times you want to loop, use a for loop!! If you don't, use a while loop!!"

① Loop variable → we typically have some variable that is used to (a) decide when the loop is done, and (b) serve as an index to arrays, etc. within the loop

```
for ( var i = 0 ; i < MAX ; i++ ) {
}
```

① definition and starting value
② termination condition
③ increment

```
① var i = 0 ;
while ( i < imax ) {   ②
}
```

≡

```
    ...
    i++;
}
```

Functions

```
function myFunction (p1, p2, ....) {
    ...
}
```

→ p1, p2, ... are parameters that are "passed" to the function, from "somewhere else", i.e, the browser, the server, another part of the script, some other function, ... etc.

→ the names of the parameters

in the function need not (should not!)
be the same as whoever calls the
function!!!

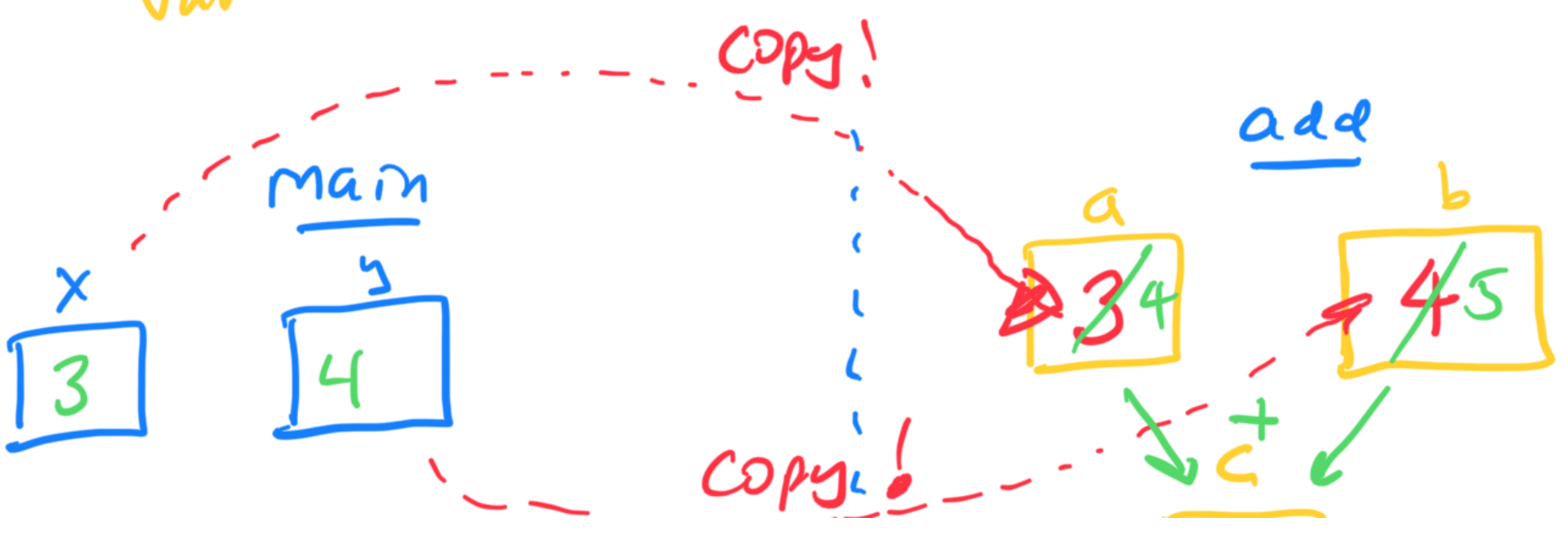
Examples:

```
function add (a, b) {
    a = a + 1;
    b = b + 1;
    c = a + b;
    return c;
}
.....

var r1 = add (3, 4);

var x = 3, y = 4;

var r2 = add (x, y);
```



→ The point here is that a, b are Copies of x, y. Changing a, b within the function does not change x, y !!!!!

This is called "pass by value".

Default Values :

```
function add ( a=0 , b=1 ) {
    a = a+1 ;
    b = b+1 ;
    c = a+b ;
    return c ;
}
```

```
console.log(add(3,4));  → 9
console.log(add());     → 3
```

Functions + Loops + If-Else Example:

(zyLab 6.17)

Guessing Game (Simple Stupid Version)

① A function gets called with TWO parameters
   ① A number to be guessed
   ② Max. # of guesses.

② The user can then guess the number, up to MAX times.

③ with each guess a "higher/lower" hint is given (unless they guess the number)

Step 1 : Simulate performance.

(37, 10)

↑ # ↑ # of guesses.

50 → "lower"

25 → "higher"

35 → "higher"

40 → "lower"

37 → "correct!" → 5 guesses.

Step 2 : Pseudo Code

while (total guesses < MAX) {

  input guess
  if guess > #
    - "lower"

  else

if guess < ...
- "higher"
else
- "correct" ... exit loop!

}

You got it → # of guesses

or

You did not get it.

Step 3: Actual code, maybe -

→ lots of "print" statements to help debug

→ lots of comments to organize thoughts.

→ focus on basic functionality, first.

Step 4: Sanity checking

→ think about bad / stupid user input (enters nothing, enters string instead of number, etc.)

→ "edge cases" ... think about input values that could cause problems

√res , $\overline{\quad}$, 0 , etc...

"bob smith" (spaces)...

upper case vs. lower case ...