Jerry Cain and Eric Roberts
CS 106J

Handout #15A
April 19, 2017

Mechanics of Functions (Addendum)

Mechanics of Functions
(Addendum)

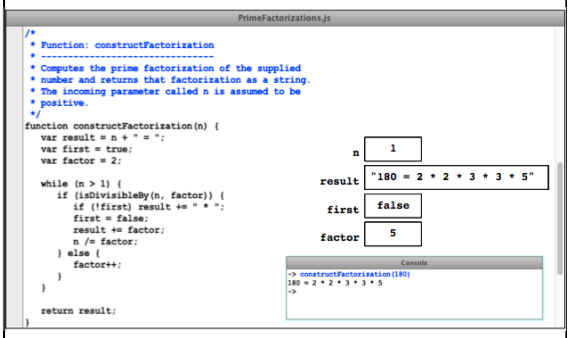
Jerry Cain and Eric Roberts
CS 106J
April 19, 2017

Exercise: Generating Prime Factorizations

- A more computationally intense problem is to generate the prime factorization of a positive integer *n*.
- An integer is prime if it's greater than 1 and has no positive integer divisors other than 1 and itself.
  - ✓ 5 is prime: it's divisible only by 1 and 5.
  - ✓ 6 is not prime: it's divisible by 1, 2, 3, and itself.
- Some prime factorizations:

```
Console
-> PrimeFactorizations(501, 512)
501 = 3 * 167
502 = 2 * 251
503 = 503
504 = 2 * 2 * 2 * 3 * 3 * 7
505 = 5 * 101
506 = 2 * 11 * 23
507 = 3 * 13 * 13
508 = 2 * 2 * 127
509 = 509
510 = 2 * 3 * 5 * 17
511 = 7 * 73
512 = 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2
->
```

PrimeFactorizations.js



PrimeFactorizations.js

Some thought questions and exercises:

- The solution relies on a single Boolean called **first**. What problem is **first** solving for us?
- During our trace of **constructFactorization(180)**, **factor** assumed the values of 2, 3, 4, and 5. 2, 3, and 5 are prime numbers and therefore qualified to appear in a factorization? How does the implementation guarantee 4 will never make an appearance in the returned factorization?
- What is returned by **constructFactorization(1)**? How could you have changed the implementation to return **"1 = 1"** as a special case return value?
- Trace through the execution of **constructFactorization(363)** as we did for **constructFactorization(180)**.
- Our implementation relies on a parameter named **n** to accept a value from the caller, and then proceeds to destroy **n** by repeatedly dividing it down to 1. Does this destruction of **n** confuse **PrimeFactorizations**'s **for** loop? Note that its counting variable is also named **n**.

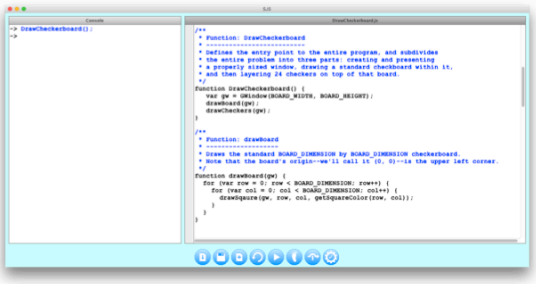
Exercise: Drawing A Checkerboard

- For the rest of lecture, we'll collectively design and decompose (and to the extent we have time, implement) a graphics program that draws the initial configuration for a game of checkers.

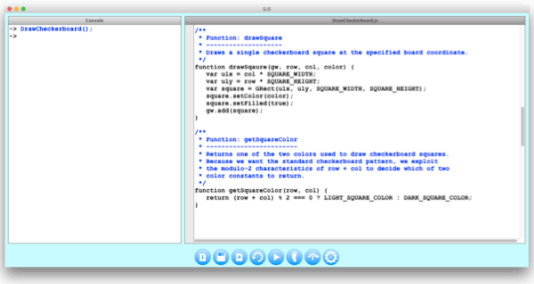
```
/**
 * Function: DrawCheckerboard
 * --------------------------
 * Defines the entry point to the entire program, and subdivides
 * the entire problem into three parts: creating and presenting
 * a properly sized window, drawing a standard checkboard within it,
 * and then layering 24 checkers on top of that board.
 */
function DrawCheckerboard() {
   var gw = GWindow(BOARD_WIDTH, BOARD_HEIGHT);
   drawBoard(gw);
   drawCheckers(gw);
}
```

DrawCheckerboard.js

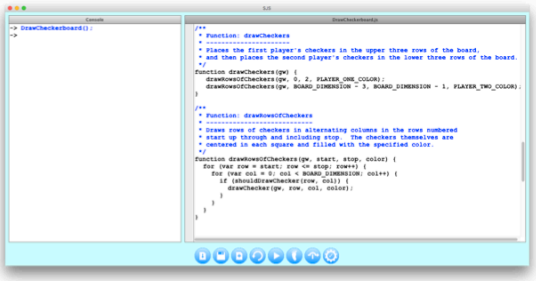
DrawCheckerboard.js



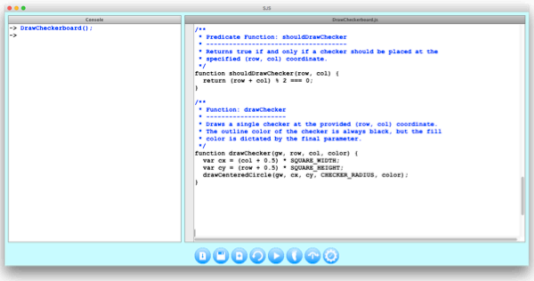
DrawCheckerboard.js



DrawCheckerboard.js



DrawCheckerboard.js



DrawCheckerboard.js