```
var config = {
    baseBet: { type: "number", value: 10 },
    baseTarget: { type: "multiplier", value: 1.11 },
    targetIncrease: { type: "multiplier", value: 1.09 },
    stopBalance: { type: "number", value: 1000 },
    maxLoss: { type: "number", value: 100 },
    maxTarget: { type: "number", value: 50},
    div: { type: "number", value: 2 },
    mult: { type: "number", value: 2 },
    state: {
        value: 'cont', type: 'radio', label: 'On run',
        options: {
            cont: { type: 'noop', label: 'Continue play on hit' },
            stop: { type: 'noop', label: 'Stop play on hit' },
        }
    },
};
```

# Nigerian Prince v1

*// an experiment from the Ghostnipple Laboratory*

*// if this script should escape the laboratory, under no circumstances should you run it, as it will empty your bank roll faster than a Nigerian Prince.*

```
var baseBet        = config.baseBet.value
var baseTarget     = config.baseTarget.value
var targetIncrease = config.targetIncrease.value
var maxTarget      = config.maxTarget.value
var stopBalance    = config.stopBalance.value
var maxLoss        = config.maxLoss.value
var maxDiv         = config.div.value
var maxMult        = config.mult.value
var currentBet  = baseBet
var cashOut     = baseTarget
var startBR     = balance
var currentBR   = balance
var sessionBR   = balance
var lossCount   = 0


    while (true) {
        if (stopBalance > this.balance / 100  || lossCount >= maxLoss) {
            this.stop();
        }
        let lastGame = await this.bet(Math.round(currentBet) * 100, cashOut);
        let lastCrash = lastGame.multiplier;
        console.log(`${lastGame.multiplier }`);
        if (lastCrash < cashOut) {
            cashOut += targetIncrease
        if (cashOut > maxTarget) {
            currentBet *= maxMult
            cashOut /= maxDiv
                lossCount++ }
        } else {
        currentBet = baseBet
        cashOut = baseTarget
        lossCount = 0
        }

        this.clearLog()
        this.log(`Betting ${Math.round(currentBet)} bits @ ${Math.round(cashOut * 100) / 100}x`);
        this.log(`Current profit/losses: ${Math.round(this.balance - startBR) / 100}`)
        this.log(`Session profit/losses: ${Math.round(this.balance - sessionBR) / 100}`)
        if (lastGame.balance > startBR) { startBR = lastGame.balance; }
    }
```