

Werkgroep Blockchain

# Making in-game items real with blockchain

# Menno Bruijn

- Test consultant bij Capgemini
- Gamer, motorrijder en klimmer
- Bezig met blockchain sinds 2014

Ga alvast naar [Menti.com](https://menti.com)  
Voer de code in: 55 92 50







Ga alvast naar [Menti.com](https://www.menti.com)  
Voer de code in: 55 92 50





Ga alvast naar Menti.com  
Voer de code in: 55 92 50

# In de afgelopen jaren

- Markt van in-game items is gegroeid
- Groei van het aantal free to play games
- Groei van het aantal gamers



FEATURED 2D 14H 15M

MULTIPLAYER 6v6 MW

BATTLE ROYALE AUTO-FILL

SOLO DUOS TRIOS QUADS

PLUNDER AUTO-FILL

SOLO DUOS TRIOS QUADS

OTHER

WARZONE RUMBLE 50v50 DEATHMATCH

PRACTICE TRIALS x24



Mission - High Roller

[2/7] - Buy 1 Loadout Drop(s)

Daily Challenges

Start 1 Contract(s) at the Goregard Lumberyard

Collect your loadout 2 time(s)

Get 15 kills with a common weapon

View Challenges



READY



CHALLENGES		
Play 2 games	2/2	1,000
Play 12 games	5/12	3,000
Get 2 knockdowns as Bangalore	2/2	3,000
Outlive 250 opponents	181/250	3,000
SEE ALL CHALLENGES		

TIER 1 0 / 10

OUT OF SEASON

SEASON ENDS: 69 Days

PAGE 1 / 13

1	2	3	4	5	6	7

PURCHASE You can purchase the Battle Pass to claim all the rewards you've earned.



Roostertail Frills (5)



Crown Crate Experience Scroll



Crown Lesson: Riding Speed



Crown Tri-Restoration Potion (5)



# Hoe zet je game items op een blockchain?

- Tokenisation
- Non-fungible tokens
- Tokenstandaarden



# Benodigdheden voor tokenisation







```
Cardownership.sol  Cardgenerator.sol X  ownable.sol  ERC721.sol  Migrations.sol

card-game > contracts > Cardgenerator.sol
1  pragma solidity ^0.4.19;
2
3  import "./ownable.sol";
4
5  contract Cardgenerator is Ownable {
6
7      //Event for new card creation
8      event NewCard(uint cardId, uint dna);
9
10     //Parameters to define amount of digits, the modulus and the amount of cards in a start
11     address public owner;
12     uint dnaDigits = 16;
13     uint dnaModulus = 10 ** dnaDigits;
14     uint public startingDeck = 10;
15     uint public cardPackSize = 5;
16     uint public cardPackCost;
17
18     uint public globalCardCount;
19
20     //Card characteristics
21     struct Card {
22         uint dna;
23     }
24
25     //Array of all cards created
26     Card[] public cards;
27
28     mapping (uint => address) public cardToOwner;
29     mapping (address => uint ) public ownerCardCount;
30
31     //Constructor for setting initial card pack price
32     function Cardgenerator() public {
33         cardPackCost = 3 ether;
34         owner = msg.sender;
35     }
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



```
Cardownership.sol  Cardgenerator.sol X  ownable.sol  ERC721.sol  Migrations.sol
card-game > contracts > Cardgenerator.sol

30
31 //Constructor for setting initial card pack price
32 function Cardgenerator() public {
33     cardPackCost = 3 ether;
34     owner = msg.sender;
35 }
36
37 //Create a new card by passing dna, assign to msg.sender and trigger New Card event
38 //Card id = position-1 in card array
39 function _createCard(uint _dna) private {
40     uint id = cards.push(Card(_dna)) - 1;
41     cardToOwner[id] = msg.sender;
42     ownerCardCount[msg.sender]++;
43     emit NewCard(id, _dna);
44     globalCardCount++;
45 }
46
47 //Generate (semi)random dna from a string using keccak256 and revert to needed amount
48 function _generateRandomDna(uint _time, uint _nonce) private view returns (uint) {
49     uint hashInput = _time * _nonce;
50     uint rand = uint(keccak256(hashInput));
51     return rand % dnaModulus;
52 }
53
54 //Create a random card when account has no cards
55 //add randomly generated name based upon cardtype
56 function createStartingDeck() public {
57     require(ownerCardCount[msg.sender] == 0);
58     uint _nonce = 1;
59     for (uint i = 0; i < startingDeck; i++) {
60         uint randDna = _generateRandomDna(block.timestamp, _nonce);
61         _createCard(randDna);
62         _nonce++;
63     }
64 }
65
66 //Buy cardpacks. Takes (cardPackCost * _amount) in ETH.
```



# Interpretatie van DNA

- Random hash
- Pak de laatste 10 cijfers
- Pas interpretatie kenmerken toe





```
Cardownership.sol X Cardgenerator.sol ownable.sol ERC721.sol Migrations.sol
card-game > contracts > Cardownership.sol
1  pragma solidity ^0.4.19;
2
3  import "../Cardgenerator.sol";
4
5  contract Cardownership is Cardgenerator {
6
7      event Transfer(address indexed _from, address indexed _to, uint256 _tokenId);
8      event Approval(address indexed _owner, address indexed _approved, uint256 _tokenId);
9      event ApprovalForAll(address indexed _owner, address indexed _operator, bool _approved);
10
11     mapping (uint => address) cardApprovals;
12
13     modifier onlyOwnerOf(uint _cardId) {
14         require(msg.sender == cardToOwner[_cardId]);
15         _;
16     }
17
18     function balanceOf(address _owner) external view returns (uint256 _balance) {
19         return ownerCardCount[_owner];
20     }
21
22     function ownerOf(uint256 _tokenId) external view returns (address _owner) {
23         return cardToOwner[_tokenId];
24     }
25     // Ga hier verder met ERC721 implementatie
26     function _transfer(address _from, address _to, uint256 _tokenId) private {
27         ownerCardCount[_to]++;
28         ownerCardCount[_from]--;
29         cardToOwner[_tokenId] = _to;
30         emit Transfer(_from, _to, _tokenId);
31     }
32
33     function transfer(address _to, uint256 _tokenId) public onlyOwnerOf(_tokenId) {
34         _transfer(msg.sender, _to, _tokenId);
35     }
36
```



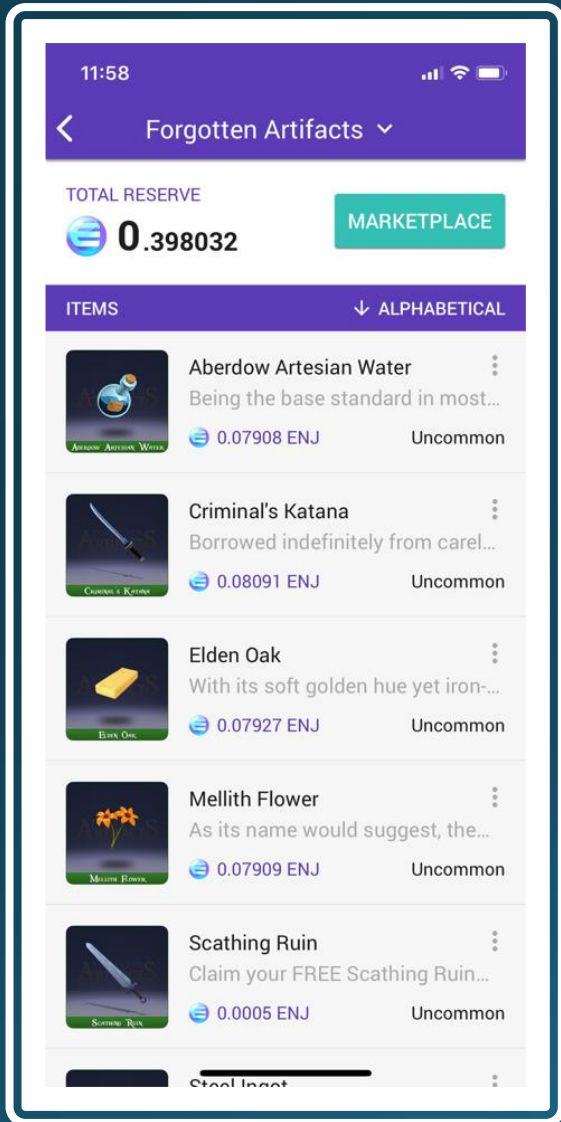


```
Cardownership.sol Cardgenerator.sol ownable.sol ERC721.sol X Migrations.sol
card-game > contracts > ERC721.sol
1  pragma solidity ^0.4.19;
2
3  contract ERC721 {
4      event Transfer(address indexed _from, address indexed _to, uint256 _tokenId);
5      event Approval(address indexed _owner, address indexed _approved, uint256 _tokenId);
6
7      function balanceOf(address _owner) public view returns (uint256 _balance);
8      function ownerOf(uint256 _tokenId) public view returns (address _owner);
9      function transfer(address _to, uint256 _tokenId) public;
10     function approve(address _to, uint256 _tokenId) public;
11     function takeOwnership(uint256 _tokenId) public;
12 }
```

Hoe ziet dit eruit met een front end?











0.277777	ETH
----------	-----

Name	Item ID ↓	Type	Reserve	Circ	Supply	Others
 <a href="#">In-game currency</a>	7000000000000bca	FT	0	1500	Infinity	Transferable
 <a href="#">Capgemini Value #1</a>	188000000000037e	NFT	0	3	10 (Fixed)	Transferable, 100000000
ROI						

Ethereum Kovan

0 / 5,000

Api Hits

1 / 5

Team Members

General Information

Assets

Team

Plan Details

Wallet Settings

Wallet

0x1F2DD...36D99

600

KENJ

0.277777

ETH

## Asset Details

Asset Name

In-game valuta (Gems) 



25 / 255

Maximum Total Supply

1000



ENJ Value Per Asset

0.5



The ENJ Value field must be 0.0031622 or more

% ENJ Returned on Asset Melt

50



Asset Type

Fungible Token (FT) 



Supply Model

Fixed 



Transferable

Always Transferable 



Transfer Fee Type

None 



## Starting Supply

To create assets on the blockchain, you will need to define a starting supply and have enough ENJ in your linked wallet to cover that amount. You can stake the value of any amount of items between 1 and the specified maximum total supply. Whatever amount of ENJ you choose to stake now can be used when minting items.

Starting Supply

1000



Total ENJ required for starting supply:



500 KENJ



## Platform Assets

Search transactions, addresses, blocks, coins, assets, and platforms



INFO

ASSETS

HOLDERS



DETAILS

MARKET

TXNS



Age of Rust

33 (29,399,214) Assets



Canto's Private Keys

FT

Age of Rust

20,000 ENJ



Arjen's Holo drive

FT

Age of Rust

12,000 ENJ



Ankh Artifact

FT

Age of Rust

999,9999 ENJ



Origin

FT

Age of Rust

650 ENJ



Isolationists Mask

FT

Age of Rust

200 ENJ



Mech Source Code

FT

Age of Rust

200 ENJ



Ununpentium

FT

Age of Rust

100 ENJ



Lastlight

FT

Age of Rust

100 ENJ



Portable Fold Gate

FT

Age of Rust

50 ENJ



Resistance Radio

FT

Age of Rust

50 ENJ





## Toekomstmogelijkheden

- Bewijsbaar unieke game-items
- Dezelfde game-items interpreteerbaar door verschillende games
- Items buiten games verhandelbaar
- Spelers worden écht eigenaar

