

1NOV 2023  
SONAM KUMARI\_13201172022

The screenshot displays the Alchemy University course interface. The top section shows the course 'Blockchain Cryptography' with a progress bar at 33%. The course is divided into several modules, with 'Blockchain and Crypto' selected. The main content area shows a video player with the title 'Blockchain & Crypto' and a play button. Below the video, there is a task titled 'Brute Force Hashing' under the heading 'CRYPTOGRAPHIC HASHES 1: Find Favorite Color'. The task description explains that cryptographic hash functions like SHA256 are one-way functions, and it provides a challenge to find the input for a given hash. The task includes a code editor with a JavaScript solution and a test output section showing three tests passed: 'should work for red', 'should work for green', and 'should work for blue'.

**Blockchain Cryptography** (33%)

- Bootcamp Introduction (✓)
- The First Primitive (✓)
- Blockchain and Crypto** (✓)
- Cryptographic Hashes (✓)
- Digital Signatures (0%)
- Proof of Work (0%)
- Blockchain Network (0%)
- Week 1 Project (0%)

## Blockchain & Crypto

University's Ethereum Developer Bootcamp! In this course we will take you need to know to build, deploy, test and interact with programs on the Ethereum of this course we'll refer to these programs as **smart contracts** and you'll soon First things first, let's dive into what purpose blockchains serve and discuss our primitive: **the cryptographic hash function** 📌

### Brute Force Hashing

Cryptographic Hash Functions like **SHA256** are one-way functions. This means that if you have the input, it's relatively trivial to find the output. On the other hand, if you have the output, it is infeasible to find the input.

However, if you knew the hashes of some common inputs, then you could **brute-force** guess at the output or create a **Rainbow Table** to determine what that input is.

It's easy to find that the SHA256 hash of "apples" is `0xf5903f...0f74d9`. If this was a likely input, a hacker could search for it specifically and know that the input was "apples"! 🕵️

⚠️ For security purposes, it's important to remember to use a random **salt** which you can add to your input to make it unquessable via the

```
1 const { sha256 } = require("ethereum-cryptography/sha256");
2 const { toHex, utf8ToBytes } = require("ethereum-cryptography/utis");
3
4 // the possible colors that the hash could represent
5 const COLORS = ['red', 'green', 'blue', 'yellow', 'pink', 'orange'];
6
7 // given a hash, return the color that created the hash
8 function findColor(hash) {
9   return COLORS.find(x => toHex(sha256(utf8ToBytes(x))) === toHex(hash));
10 }
11
12 module.exports = findColor;
```

**Test Output** Run Tests

**FINDCOLOR**

- ✓ should work for red Test Passed
- ✓ should work for green Test Passed
- ✓ should work for blue Test Passed