Solidity Continued

FinTech

Lesson 20.3

Class Objectives

By the end of this lesson, you will be able to:

Summarize the requirements needed to deploy a smart contract by using the JavaScript VM.

Identify the five steps for creating a deployable smart contract.

Use the Remix IDE to deploy and run transactions.

Develop a smart contract containing a function to send ether.

Provide an example of when to use the Solidity `msg` object.

Test the JavaScript VM in Solidity using functions to withdraw and deposit ether.

Discuss the history of real-world blockchain applications.


RECAP

Recap

So far, you have:

Written smart contracts with functions.

Used getters and setters.

Learned about memory and gas-fee structure.

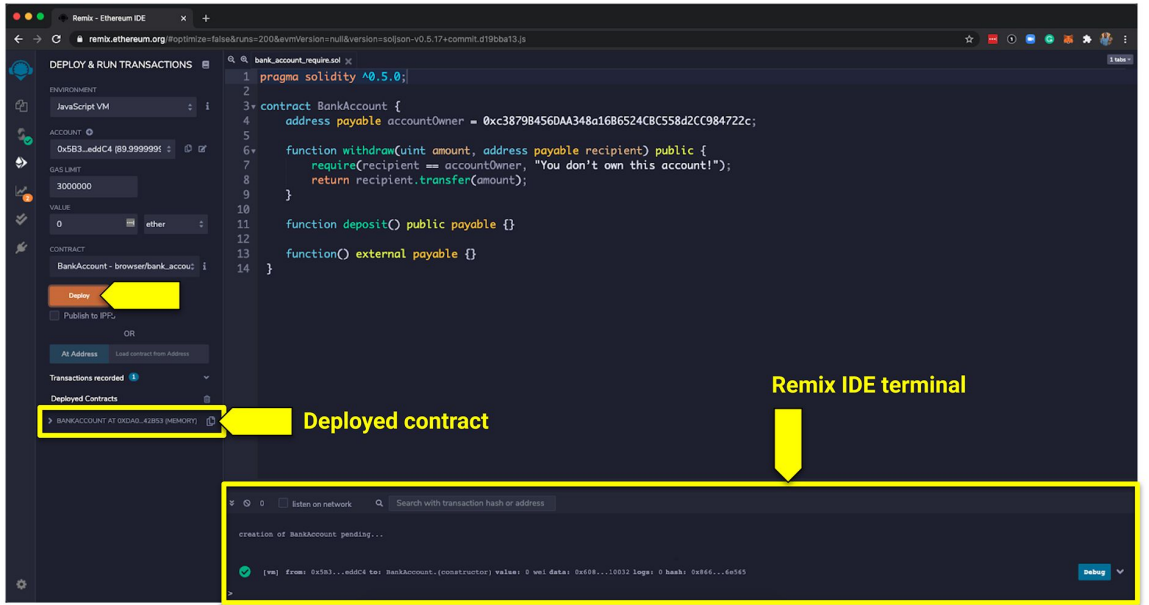
Explored variable types in Solidity.

Recap

You have also learned how to write code in Solidity and create smart contracts by using the Remix IDE.

Today, you will learn the final step of creating a working smart contract:

deploying a smart contract by using the Javascript VM and the Remix IDE.



You have **not yet** deployed any contracts you have written to the Ethereum blockchain.

There are two ways in which contracts can be deployed:

In an operating Ethereum blockchain network

In a sandboxed blockchain network

A **sandbox** is an isolated development environment.

Sandbox

A sandbox environment exists on a network and mimics the end-user operating environment.

- With a sandbox, programmers can develop and test new code in a low-risk environment that’s identical to the production environment.
- In this case, the sandboxed environment mimics a live node that’s running the EVM.

Lesson Goals

In this lesson, you will:

Learn about

the requirements for deploying and running smart contracts, and observe the smart contracts in action.

Deploy and test

contracts by using the Remix IDE and a sandboxed EVM—without spending real ether.

Homework

You'll complete the entire life cycle of a smart contract—from coding and compiling to successfully deploying on the EVM that the Remix IDE provides.

Deploying a Smart Contract in Ethereum: Requirements

Deploying a Smart Contract in Ethereum: Requirements

It’s crucial to understand the requirements for deployment.

01

Requirement 1:

Have a compiled smart contract.

02

Requirement 2:

Choose an execution environment.

03

Requirement 3:

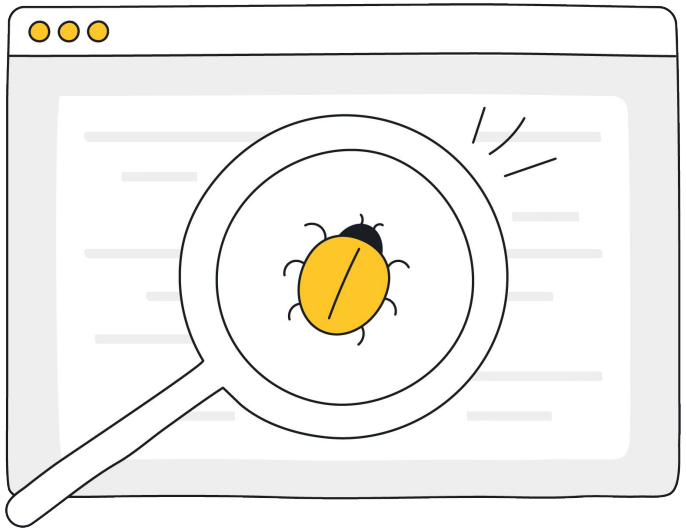
Identify the amount of gas that you need.

Requirement 1

Have a Compiled Smart Contract

Remember, a compiled smart contract can still be vulnerable to attack by bad actors.

- When writing Solidity code, business logic errors might exist due to incorrect code implementation.
- Having a successfully compiled contract ensures that the EVM can run it, and that we can use it in any Ethereum-based network.
- **Compile the bank account contract in Remix before moving on to Requirement 2.**



Requirement 2

Choose an Execution Environment

An **execution environment** is a valid blockchain network where the contract will reside and run. The Remix IDE offers the following execution environments:

JavaScript VM

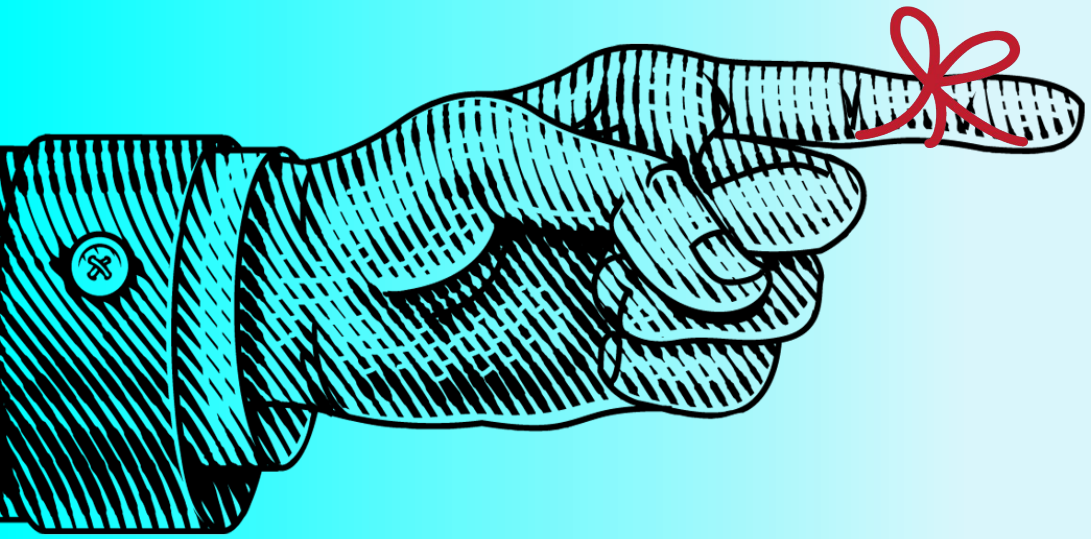
- This sandboxed blockchain is implemented in JavaScript.
- It runs in the browser to emulate a real Ethereum blockchain.
- This environment has no associated costs.
- It's intended for testing smart contracts in a sandboxed environment before deploying them to a real blockchain.
- This is the environment we'll use to deploy our smart contracts in this lesson.

Injected Web3

- From this environment, we can deploy our smart contracts in private or public Ethereum networks by using external tools, such as MetaMask or Mist.
- We'll cover these tools in later units.

Web3 provider

- From this environment, we can deploy our smart contracts directly in any private or public Ethereum network.
- We do so by using a remote node of the network—without any external tool in the middle.
- We just need to provide the URL address of the provider that we want.



Remember:

- In the previous unit, you used the Web3 provider that's built into the Web3 Python library as an Ethereum tester.
- Later, we'll use the full Web3 provider environment and other blockchain development tools to create a blockchain web application.

Requirement 3

Identify the Amount of Gas That You Need

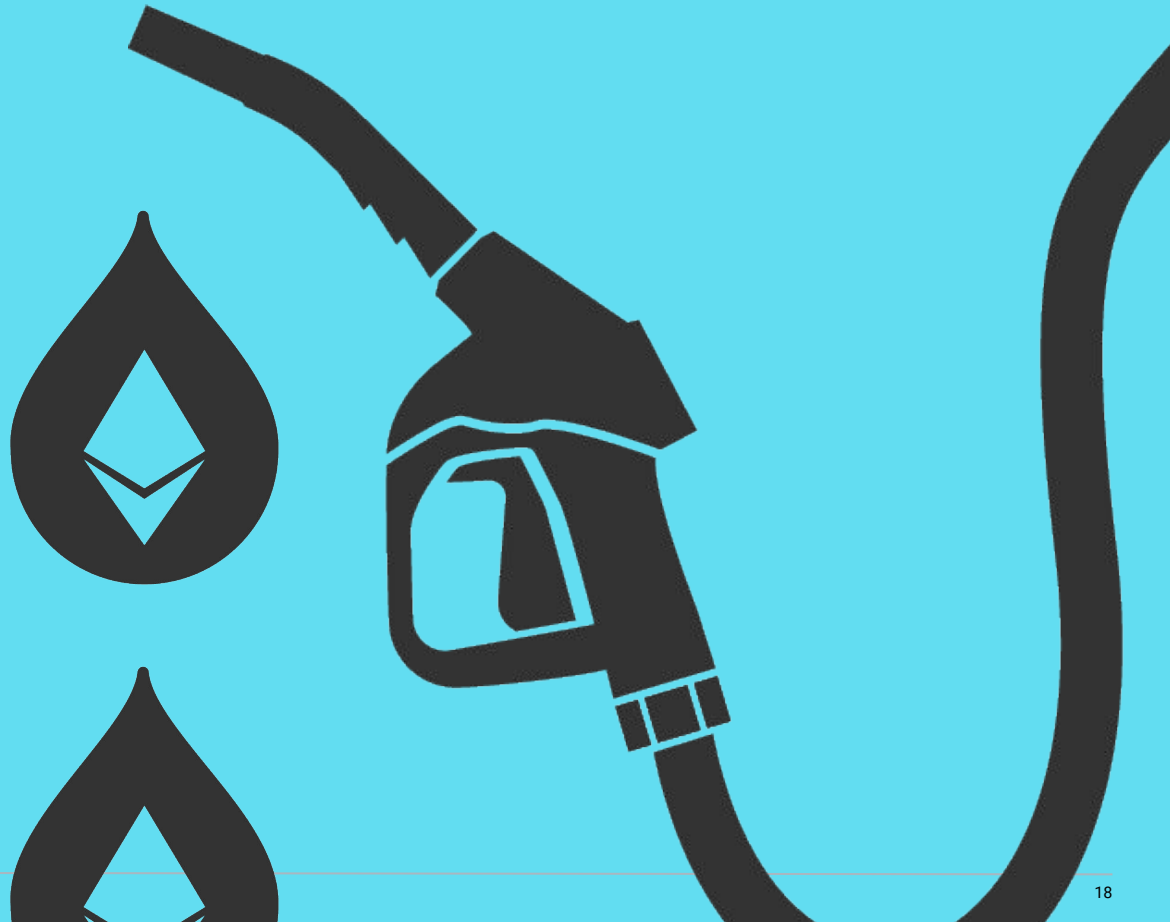
The final requirement for deploying a smart contract involves the payment that we make to the network where we'll deploy it.

- Every time that we deploy a smart contract, we need to have ether available to pay for the gas that's required to deploy and run the smart contract.
- Because we'll use the JavaScript VM in this lesson, we won't incur any costs. However, we need to learn how to identify the appropriate amount of gas by analyzing the execution log of the contract.
- The execution log records all the activity that relates to the contract. We can thus confirm the operations status of the contract and observe the gas expense for each operation that the smart contract made.

Deploying a Smart Contract in Ethereum: Requirements

As a fintech professional, it's important to know that deploying a smart contract requires more gas than what a simple ether transfer pays.

The amount of gas depends on the complexity of the contract and the optimization of the created code.



Class Slack Channel:

Good sources for learning about gas optimization:

[Under-Optimized Smart Contracts Devour Your Money]

[Presentation from DappCon]

The three requirements for deploying a smart contract are not complex.

Most of the complexity involves understanding the business needs and then writing the contract.

Once we successfully compile a smart contract and choose an execution environment, we can deploy the contract in seconds.

Deploying a Smart Contract in Ethereum: Requirements

We want to:

- Provide a checklist to assist newcomers with Solidity programming.
- Describe the steps that they should take to create and deploy a smart contract.

What will this list consist of?

Deploying a Smart Contract in Ethereum: Requirements

Taking into account all the skills that we've learned so far, the checklist might be:

01 Define the business scenario where we want to use a smart contract.

02 Architect the functionality of the smart contract, and define the functions.

03 Create the code that implements the solution that we want to build.

04 Compile the smart contract in the Remix IDE. If any errors occur, fix them.

05 Deploy the smart contract in the Ethereum network.

Items 1–3 are generic. We use them to create a software application in Python or in Solidity.

Instructor Demonstration

Deploy and Run a Smart Contract in the JavaScript VM

Questions?

Time to Code

Sending Remittances via Smart Contracts

Suggested Time:

30 minutes

Time's Up! Let's Review.

Questions?

Instructor Demonstration

The Solidity msg Object

The `msg` (message) object represents the transaction call (originated by an Ethereum address) or the message call (originated by a contract's address) that triggers a contract execution.

Attributes of the msg Object

`msg.sender`	Represents the Ethereum address that initiated the contract call. It can be an Ethereum address or a contract's address.
`msg.value`	Represents the value of ether that is sent in the transaction (expressed in wei).
`msg.data`	Represents the data payload of the call into our contract (expressed in bytes).
`msg.sig`	Represents the first four bytes of the data payload.

Questions?


Break

Time to Code

Personal Savings Smart Contract

Suggested Time:

30 minutes

Time's Up! Let's Review.

Questions?

Real-World Smart Contract Applications

Smart contracts are computer programs that allow credible transactions of digital assets under certain conditions without third parties.

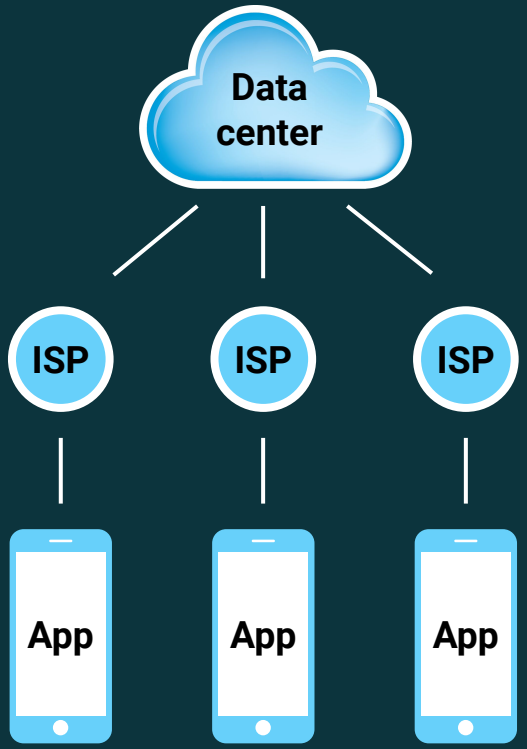
How can we connect smart contracts to the real world?

In the upcoming units,
you will learn the skills required to develop
a blockchain application.

A **dApp** is a software application that has a decentralized operation and uses decentralized storage or a database.

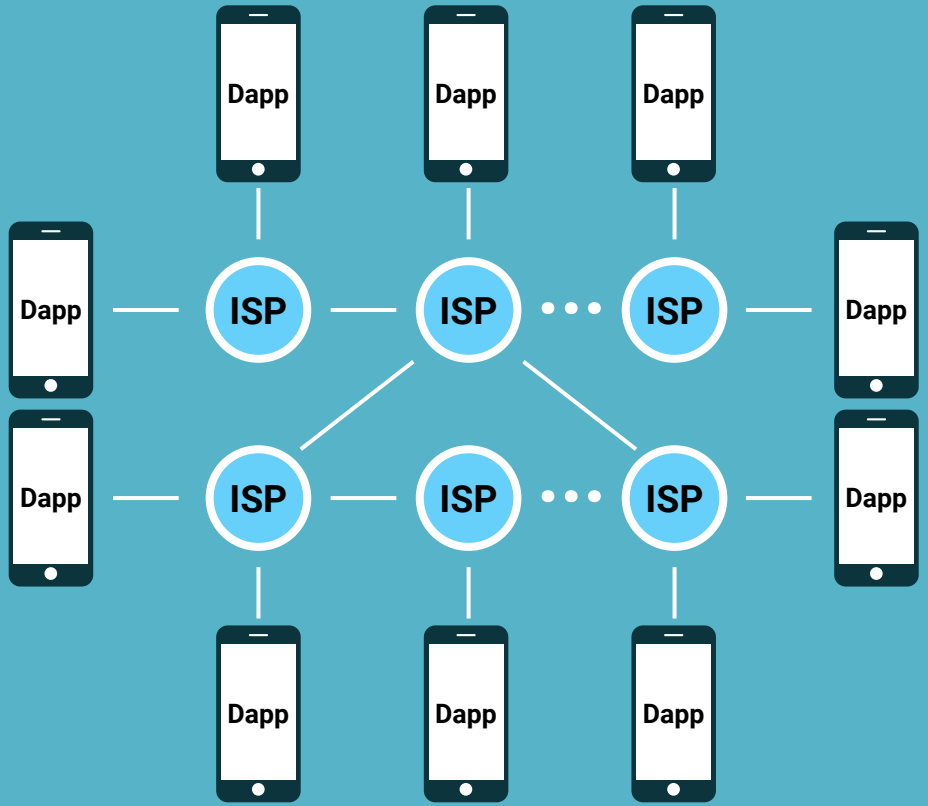
Apps

Run on a centralized server
and use centralized storage.



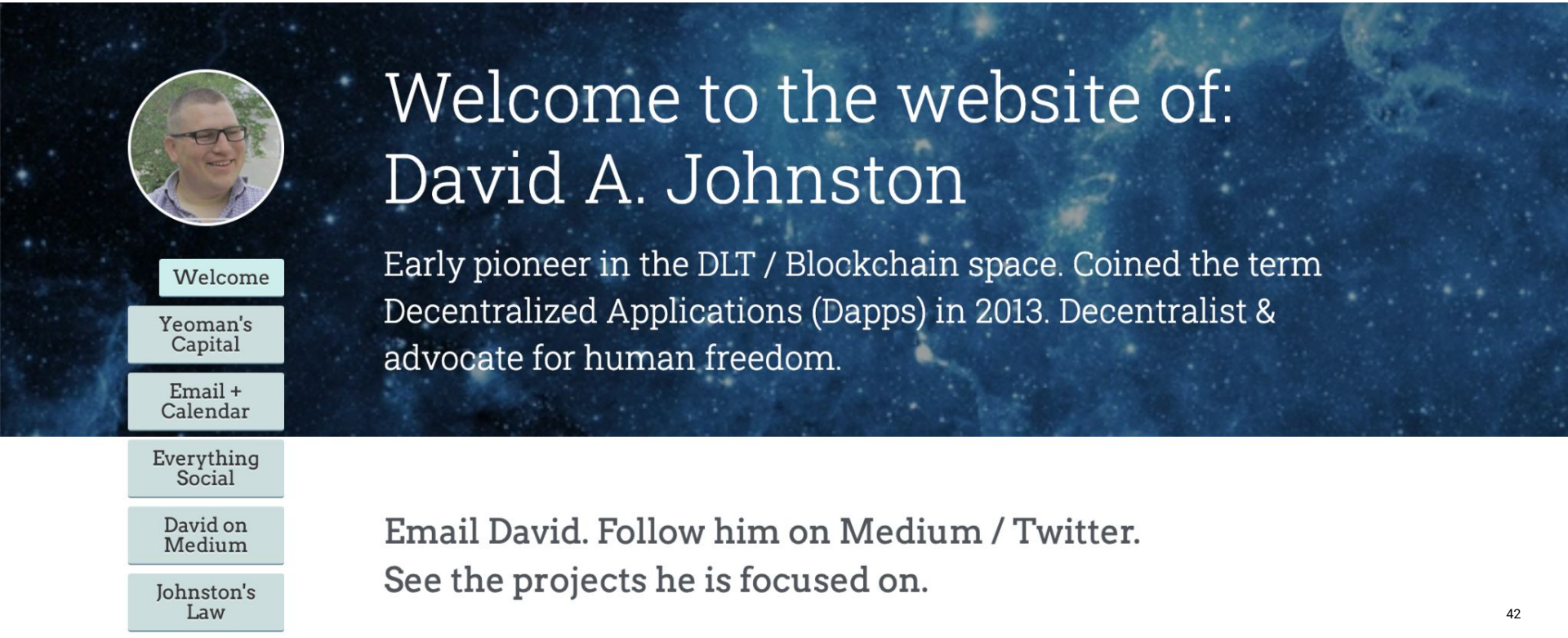
dApps

Run in a decentralized environment
that the blockchain nodes provide.



dApps

The concept of dApps was formally presented in 2015 by David Johnston, et al. in a white paper entitled [“The General Theory of Decentralized Applications, Dapps”](#).



dApps

David Johnston, et al. defined the criteria with which dApps should comply:

Decentralized operation	Open-source code, autonomous operation, consensus of its users.
Decentralized storage	Storage on a blockchain, cryptographically sealed.
Cryptographic cryptocurrency	Use of tokens for access and value contribution.
Token generation	Tokens as proof of value, generated through a cryptographic algorithm.

dApps

Using dApps, we can develop software applications not only for the financial sector but also for the following:

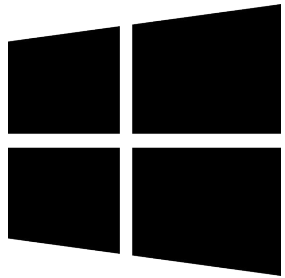
Web browsers

Cloud storage

Instant messaging

Social networks

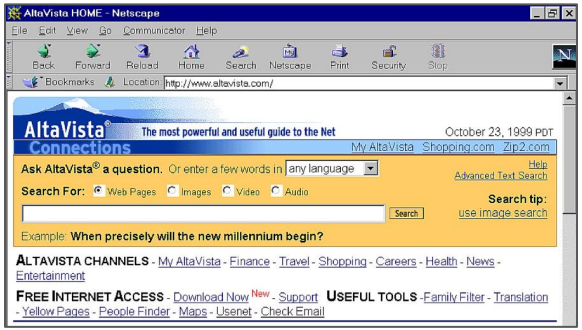
Operating systems



Evolution of the Internet

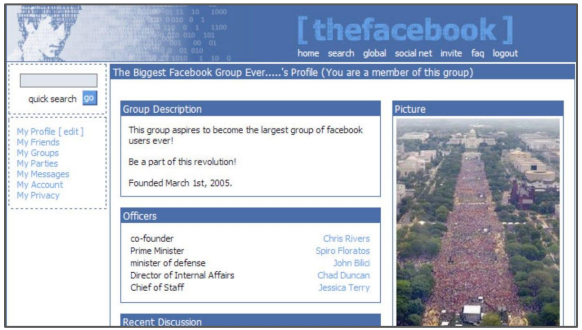
This new paradigm of software development is also known as version 3 of the Internet, or Web 3.0.

Web 1.0



The beginning of the Internet. Websites were very simple with limited interactivity, and the basic tools and protocols were defined.

Web 2.0



Marked by the rise of social networks and cloud computers.

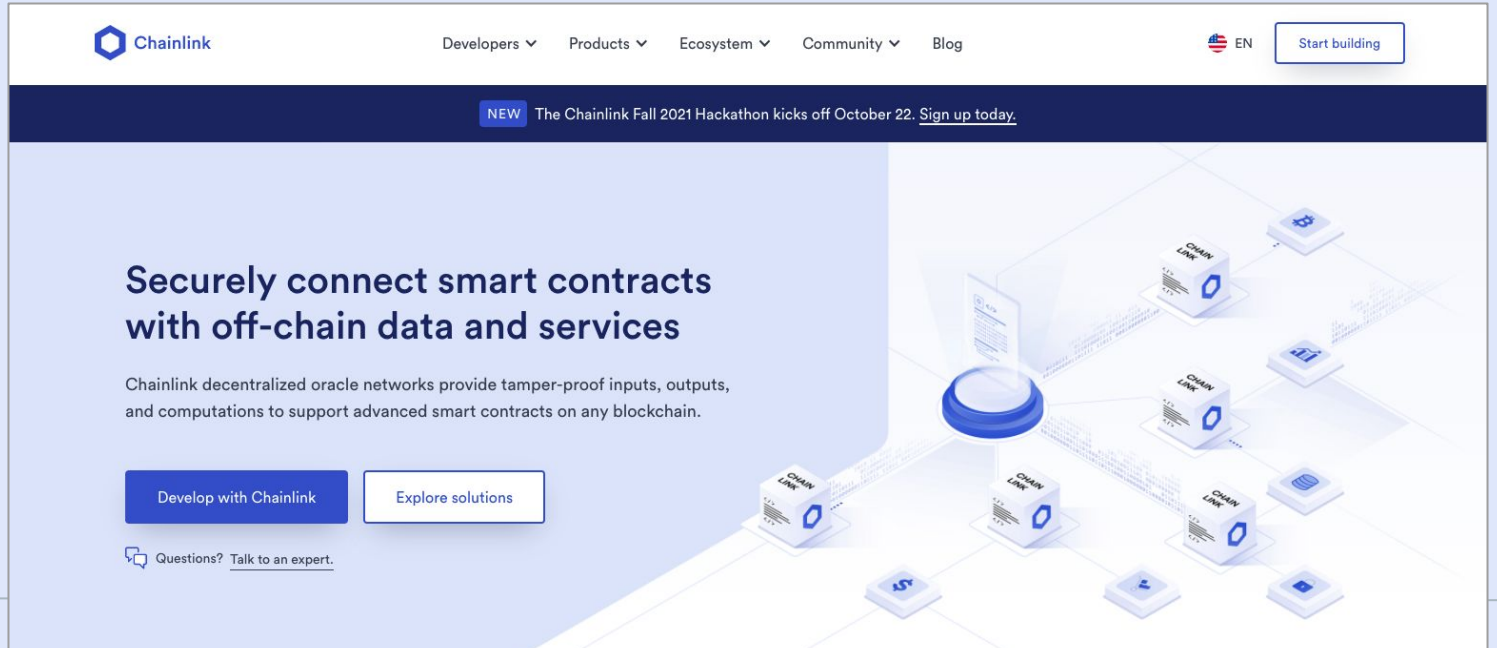
Web 3.0

The evolution of the Internet to a decentralized environment powered by blockchain technologies.

https://www.techradar.com/ /www.businessinsider.com

dApps

In order to develop a dApp, it's common to use additional frameworks, such as Chainlink, an open-source blockchain network that provides access to real-world data, events, payments, and more without sacrificing the security and reliability guarantees inherent to blockchain technology.

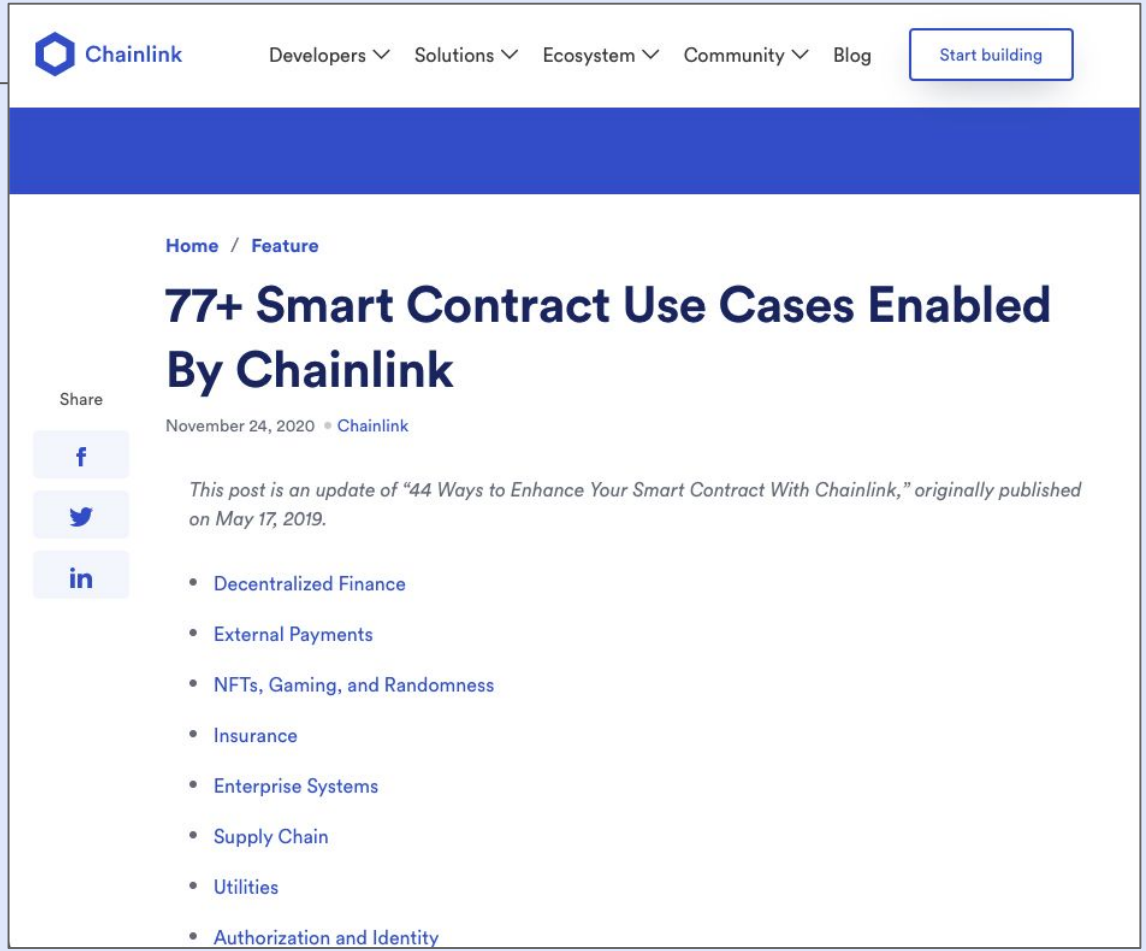



Time for a Quick Video
What Is Chainlink?

dApps

There are many ways that blockchain technology can be applied in the real world.

This page presents a few of them that can be faced by using Chainlink and Solidity.



Questions?

Structured Review

Structured Review

Look how far you've come!

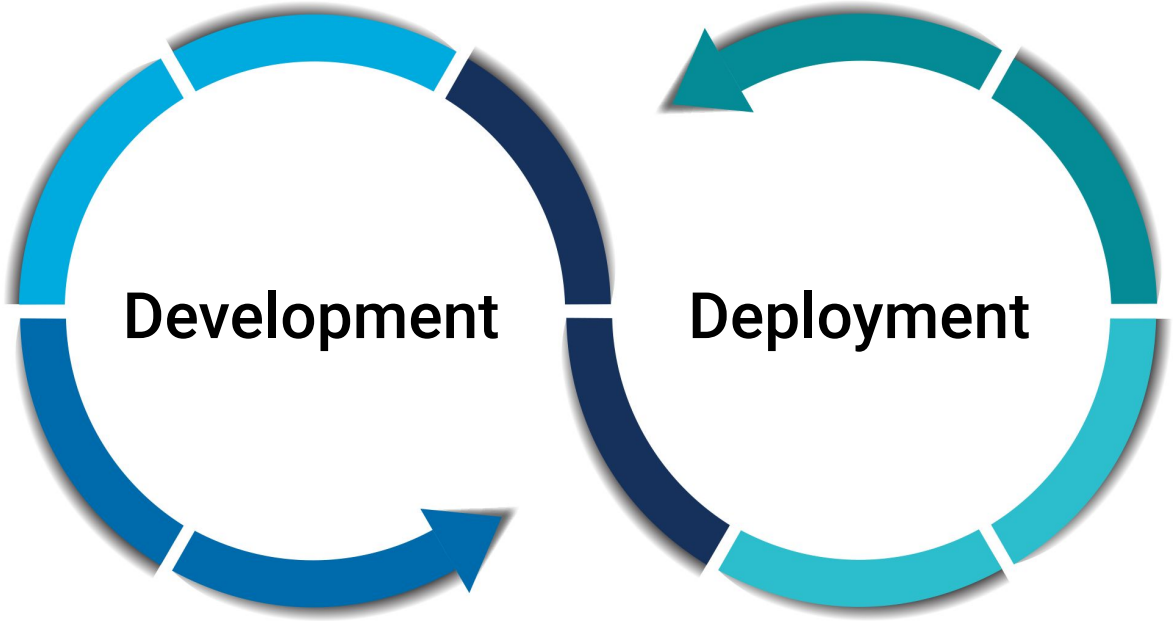
Earlier in the week,
you were just learning how to use Solidity.

Now...
you're using Solidity to deploy contracts that automate transactions on the blockchain.

How do you plan to incorporate skills from today's lesson into your resume and/or work life?

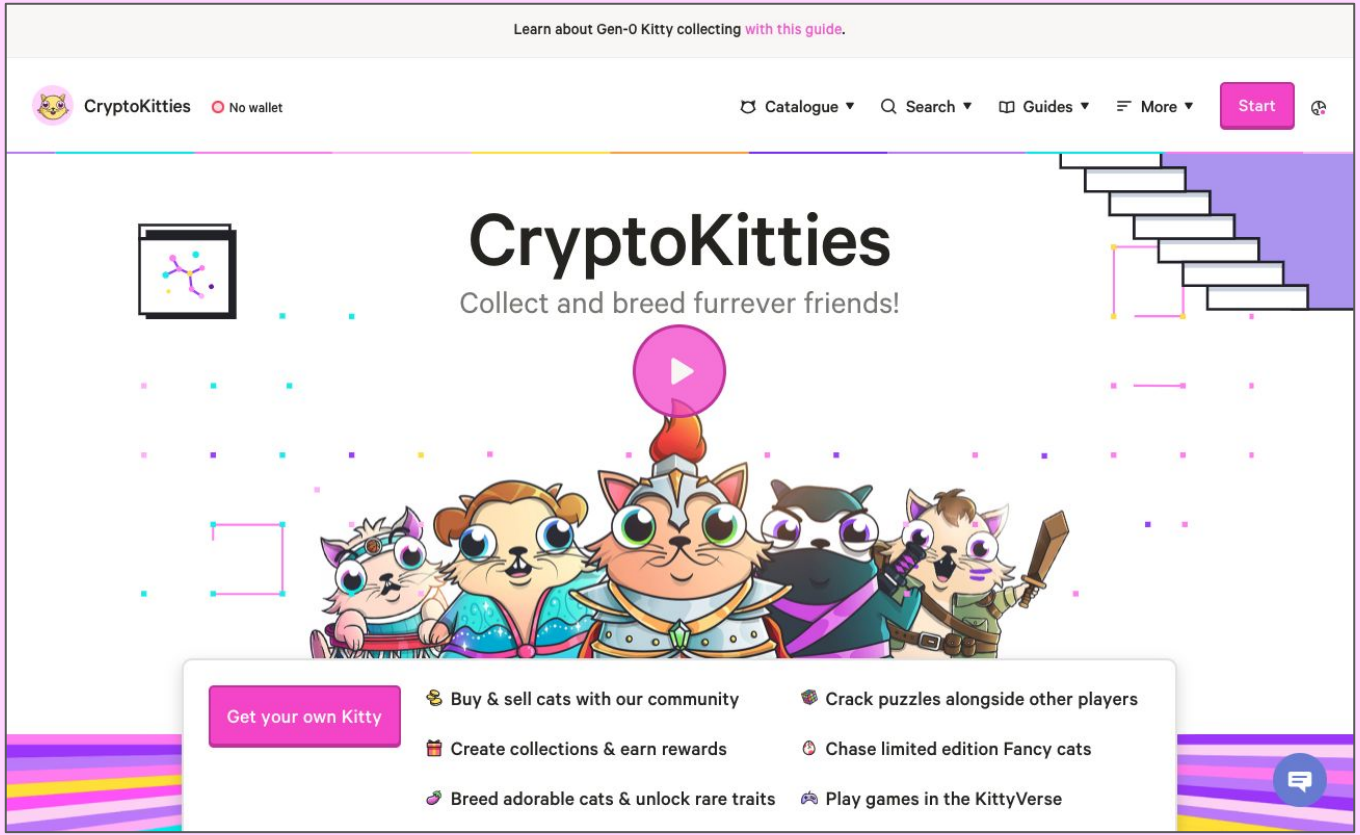
Structured Review

Throughout this unit, you have learned how to write a new programming language, test it with tools like the Remix IDE, and now deploy it. These are important skills for a blockchain developer.



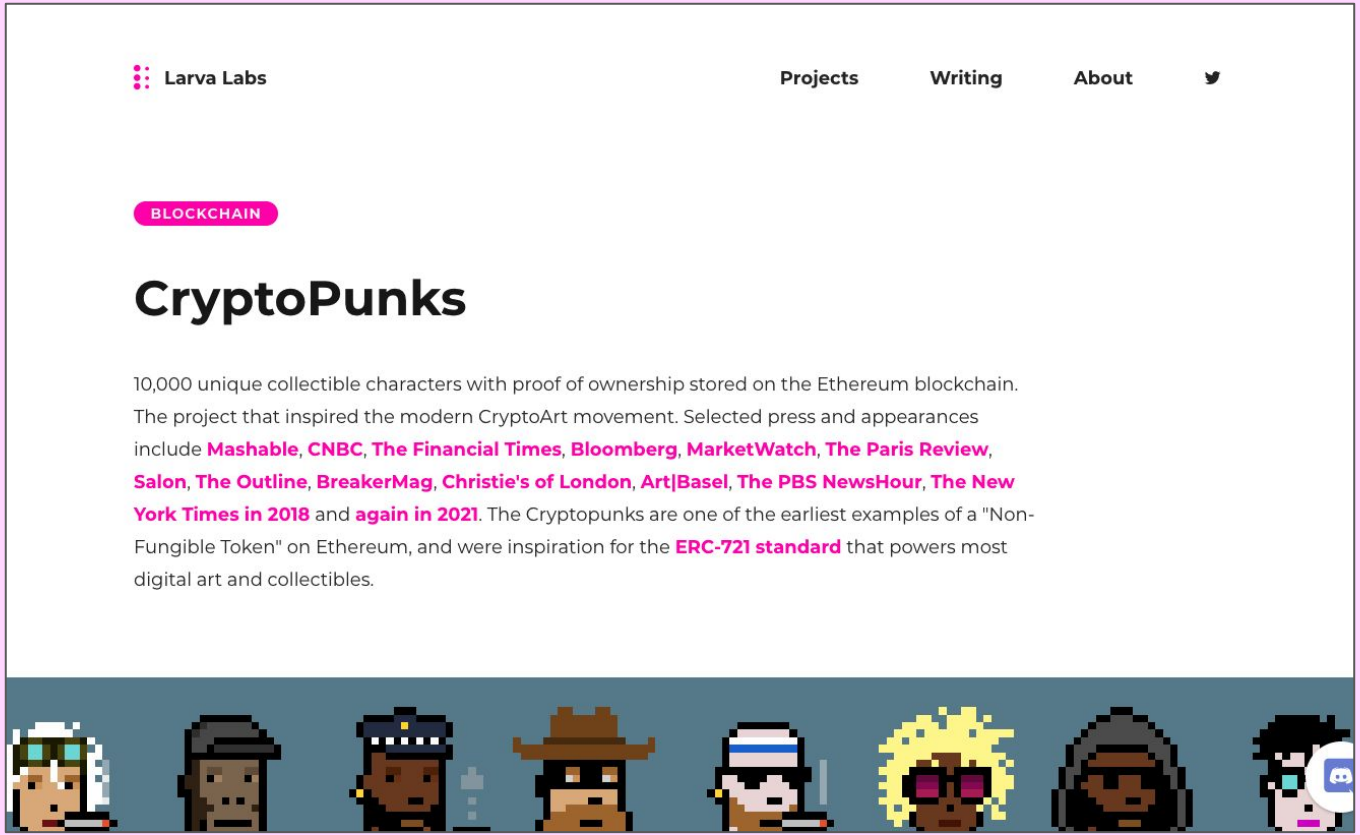
Real-World Use Cases for dApps

CryptoKitties



https://www.cryptokitties.co/

CryptoPunks



https://www.larvalabs.com

Questions?


The End