thirdweb Solidity SDK

Provides the tools needed to build custom smart contracts efficiently by offering a set of prebuilt base contracts and a set of reusable components, or extensions, that can be integrated into your own smart contracts.

Open Source Library

The contracts library is open-source. You can view the source code and contribute to it on GitHub.

View on GitHub

Base Contracts

Base contracts are prebuilt smart contracts that you can build on top of or modify, such as the ERC721Base contract. These contracts work out of the box and do not require any functions to be implemented. They are built using extensions and thus unlock intelligent features in the SDKs and custom sections on the Dashboard to easily interact with your smart contract.

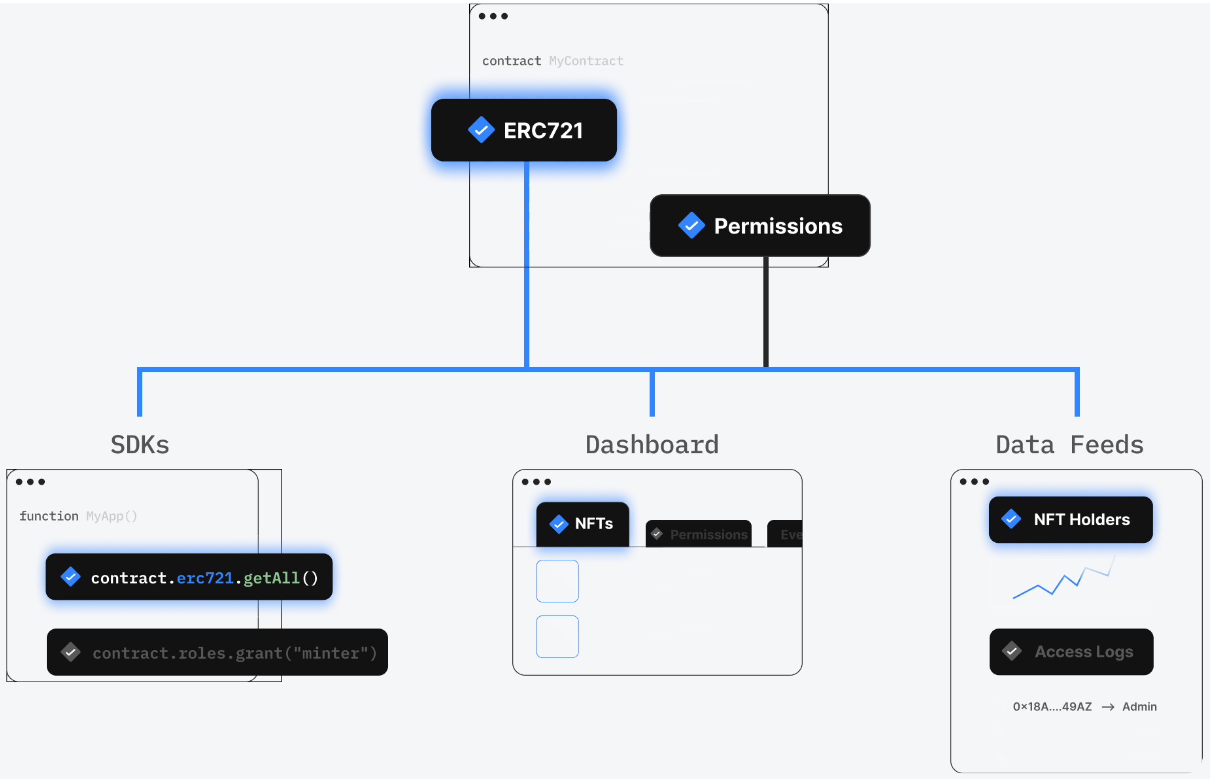
Extensions

Extensions are Solidity interfaces that are recognizable by the Dashboard and unlock functionality in the SDKs. Common EIPs are also recognizable as extensions as our interfaces conform to industry standards. They are composable pieces of logic that can be added to smart contracts easily via inheritance.

These extensions require varying levels of logic implementation* in the inheriting smart contract. The three different types of extensions in this SDK are:

- **Extension contracts**: These extensions are fully complete and do not require any logic to be implemented. By inheriting from these contracts, you add the functionality and therefore the logic to your inheriting smart contract.
- **Abstract contracts**: These extensions require some of the logic to be implemented. Some of the functions on the extension depend on the use case and therefore require custom logic that is specific to your inheriting smart contract.
- **Interfaces**: These extensions do not have an implementation. They provide the guidelines to write your own logic by giving you the functions, parameters and return types. This allows you to ensure that your contract conforms to this particular extension so that you can unlock features in the SDKs and Dashboard.

*implement = write the logic for the function with a matching function signature (matching name, parameters, visibility and return type)



Why use the Solidity SDK?

- Effortlessly integrate common smart contract features, including popular EIPs such as ERC20, ERC721, ERC1155, permissions, contract metadata, and more.
- Simplify the development workflow, allowing you to focus on more intricate systems.
- Using the base contracts gives you a gas-optimized and audited foundation to build your projects.
- Each extension you inherit unlocks smarter SDKs, custom admin Dashboards and tailored data feeds.

For example, if you deploy the ERC721Base contract which inherits from multiple extensions including ERC721Mintable, you'll unlock the `mint` button in the dashboard and can use the mint function in the SDK; which automatically uploads and pins your metadata to IPFS!

Edit this page

Was this page helpful? 👍 Yes 👎 No

Next
Getting Started »

Build A Web3 App

Use the CLI to create a web application with the React and JavaScript SDKs pre-configured:

```
npx thirdweb create app --evm
```

This will kick off an interactive series of questions to help you get started:

- Give your project a name
- Select `Create React App` as the framework
- Select `JavaScript` or `TypeScript` as the language

Exploring The Project

The `create` command generates a new directory with your project name. Open this directory in your text editor.

Inside the `index.js` file, you'll find the `ThirdwebProvider` wrapping the entire application.

This wrapper allows us to use all of the React SDK's hooks and UI Components throughout the application, as well as configure an `activeChain`; which declares which chain our smart contracts are deployed to.

Since we deployed our smart contract to the `Goerli` network, we'll set the `activeChain` to `"goerli"`:

Interact With Contracts

To connect to your smart contract in the application, provide your smart contract address (*which you can get from the dashboard*) to the `useContract` hook like so:

src/App.js

```
import { useContract } from "@thirdweb-dev/react";

export default function Home() {
  const { contract } = useContract("<CONTRACT_ADDRESS>");

  // Now you can use the contract in the rest of the component!
}
```

You can now call any function on your smart contract with `useContractRead` and `useContractWrite` hooks. Each extension you implemented in your smart contract also unlocks a new functionality for you to use in the SDK.

Let's look at how this works by reading our smart contract's NFTs and then minting new ones.

Reading Data

The NFT Collection contract we built earlier implements the ERC721 and ERC721Supply extensions.

By doing so, we can use the "View All" functionality in our application! In the React SDK, that is available through the `useNFTs` hook, which we can use by providing the `contract` object to the hook like so:

src/App.js

```
import { useContract, useNFTs } from "@thirdweb-dev/react";

export default function Home() {
  const { contract } = useContract("<CONTRACT_ADDRESS>");
  const { data: nfts, isLoading: isReadingNfts } = useNFTs(contract);
}
```

Displaying Data

Use the NFT Media Renderer component to render each of the NFTs loaded from the `useNFTs` hook, making use of the loading state in a conditional statement:

src/App.js

```
import { useContract, useNFTs, ThirdwebNftMedia } from "@thirdweb-dev/rea

export default function Home() {
  const { contract } = useContract("<CONTRACT_ADDRESS>");
  const { data: nfts, isLoading: isReadingNfts } = useNFTs(contract);

  return (
    <>
      {isReadingNfts ? (
        <p>Loading...</p>
      ) : (
        <div>
          {nfts?.map((nft) => (
            <ThirdwebNftMedia metadata={nft.metadata} key={nft.metadata.i
          ))}
        </div>
      )}
    </>
  );
}
```

Writing Transactions

Since our smart contract implements the `ERC721Mintable` extension, the `mint` feature is available to us in the SDK.

Use the `Web3Button` component to perform a `mint` transaction; which ensures the user has their wallet connected and is on the right network before calling the function:

src/App.js

```
// Place this import at the top of the file
import { Web3Button } from "@thirdweb-dev/react";

// ...

// Place this beneath your existing NFT display logic:
<Web3Button
  contractAddress={"<CONTRACT_ADDRESS>"}
  action={(contract) =>
    contract.erc721.mint({
      name: "Hello world!",
      // Place any URL or file here!
      image: "<your-image-url-here>",
    })
  }
>
  Mint NFT
</Web3Button>;
```

Deploy Your App

To host your application on IPFS and share it with the world, run the following command:

```
yarn deploy
```

That's it! 🥳 This command uses Storage to:

- Create a production build of your application
- Upload the build to IPFS
- Generate a URL where your app is permanently hosted.

Share your app with us on Twitter!

What's Next?

Congratulations on making it this far! 🎉

Now you've learned the basics, take your skills to the next level by building on top of our templates.

Manage Your Contract

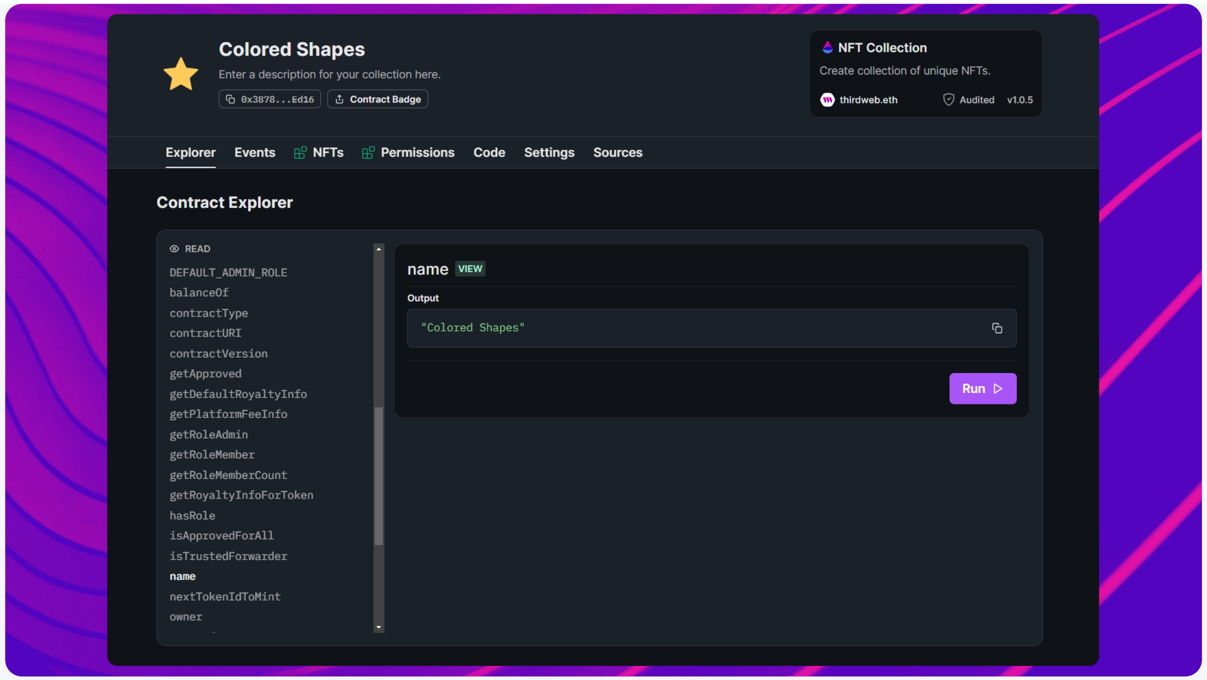
The dashboard is a central location where you and your team can manage and analyze your smart contract, with tools that allow you to perform common admin operations, with features such as:

- **Contract Explorer**: Execute any function on your contract, and see the results in real-time.
- **Events**: View A live-updating feed of events emitted by your contract.
- **Settings**: Update contract metadata, royalty and platform fees, and more.
- **Sources**: View all of your contract's source code, and verify your contract on Etherscan.
- **Code**: Learn how to use the SDK to interact with your contract.

Each extension you implement also unlocks additional features in the dashboard. For example, the contract you deployed in the previous step has the `NFTs` and `Permissions` tabs unlocked.

Contract Explorer

Connect your wallet and execute any available function on your smart contract directly from the dashboard.


Colored Shapes

Enter a description for your collection here.

0x3878...Ed16 · Contract Badge

NFT Collection

Create collection of unique NFTs.

thirdweb.eth · Audited · v1.0.5

Explorer · Events · NFTs · Permissions · Code · Settings · Sources

Contract Explorer

READ

```
DEFAULT_ADMIN_ROLE
balanceOf
contractType
contractURI
contractVersion
getApproved
getDefaultRoyaltyInfo
getPlatformFeeInfo
getRoleAdmin
getRoleMember
getRoleMemberCount
getRoyaltyInfoForToken
hasRole
isApprovedForAll
isTrustedForwarder
name
nextTokenIdToMint
owner
```

name VIEW

Output

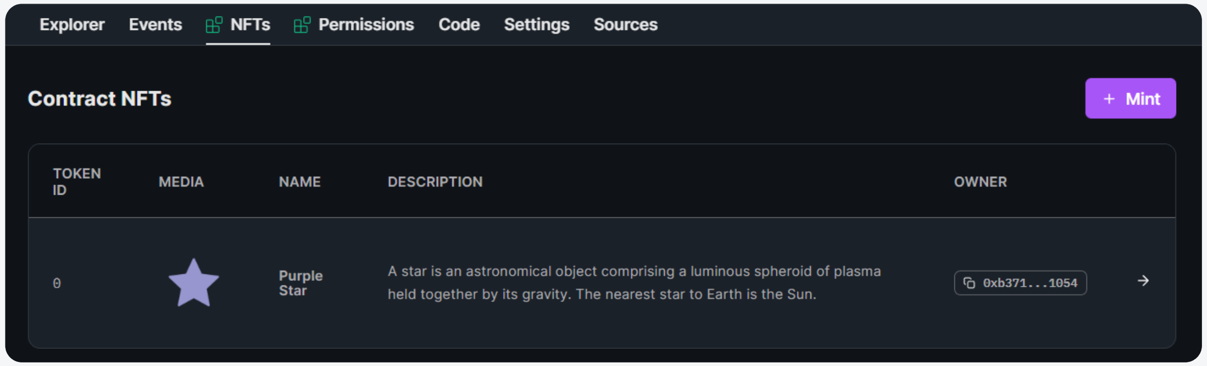
```
"Colored Shapes"
```

Run

NFTs

Since our contract implements the ERC721 standard, we unlock the `NFTs` tab in the dashboard.

Here, we can see all of the NFTs minted by our contract, and even mint new ones.


Explorer · Events · NFTs · Permissions · Code · Settings · Sources

Contract NFTs

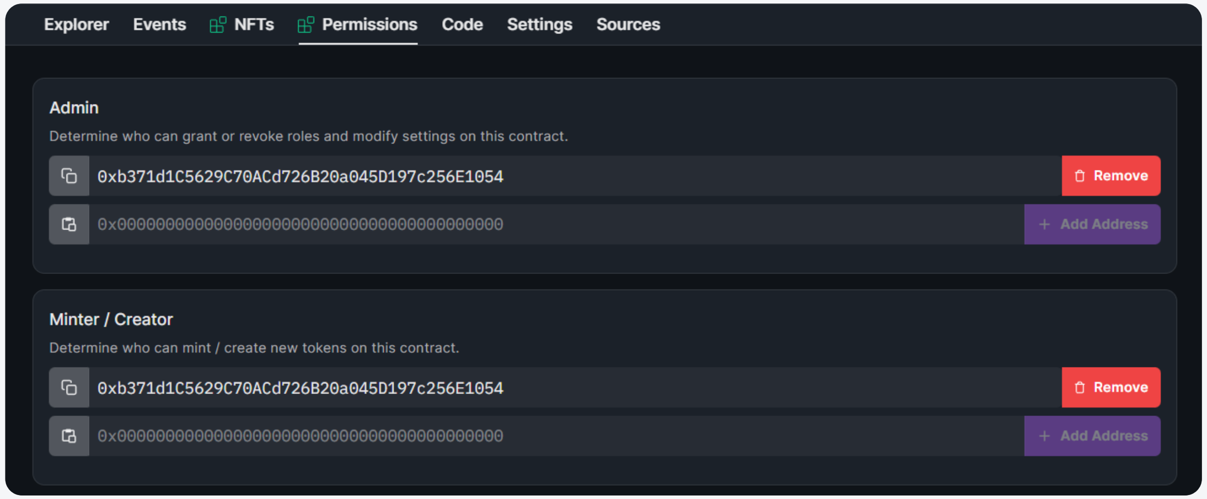
+ Mint

TOKEN ID	MEDIA	NAME	DESCRIPTION	OWNER
0	★	Purple Star	A star is an astronomical object comprising a luminous spheroid of plasma held together by its gravity. The nearest star to Earth is the Sun.	0xb371...1054

Permissions

As our contract also implements the Permissions extension, we unlock the `Permissions` tab in the dashboard.

Easily invite members of your team with tiered permission levels, and manage their access to your contract.



Admin

Determine who can grant or revoke roles and modify settings on this contract.

0xb371d1C5629C70ACd726B20a045D197c256E1054

0x0000000000000000000000000000000000000000

Minter / Creator

Determine who can mint / create new tokens on this contract.

0xb371d1C5629C70ACd726B20a045D197c256E1054

0x0000000000000000000000000000000000000000

Code

The Code tab provides the steps to creating a new application with our SDK, and interacting with your contract.

Let's take a look at how we can use the SDK to build a web application next:

Next Up: Build An Application with the SDK

Edit this page

Was this page helpful? 👍 Yes 👎 No

Previous
« Deploy A Smart Contract

Next
Build A Web3 App »

Deploy A Smart Contract

You may want to build a smart contract from the ground up, or deploy a prebuilt solution that meets your needs; we provide solutions for creating and deploying *any* smart contract.

Begin Your Journey

Select which card best describes your contract needs below.

Explore

Secure, gas-optimized, and audited contracts that are ready to be deployed with one-click.

Build your own

Create custom contracts that are specific to your use case using the Solidity SDK.

Deploy from source

Already have a contract ready to deploy? Learn how to use our interactive CLI to ship it.

Edit this page

Was this page helpful? 👍 Yes 👎 No

Previous
« Introduction

Next
Manage Your Contract »

Overview

thirdweb is a comprehensive development framework that empowers you to seamlessly build, launch, and manage web3 applications and games across any EVM-compatible blockchain.

Architecture

Whether you're developing decentralized applications from scratch or enhancing existing ones, Our tool suite accelerates your development workflow across the smart contract, application, and infrastructure layers.

Smart Contracts

Smart contracts serve as the foundational building blocks for any decentralized application. We conduct audits on our pre-built contracts to ensure the highest quality and security level. thirdweb's ContractKit extensions follow dynamic contract patterns by default. We also optimize contracts for gas efficiency, minimizing transaction costs. By prioritizing these technical considerations, developers can focus on creating exceptional web3 applications with confidence and ease.

thirdweb is designed to integrate with any contract on any blockchain seamlessly. With **ContractKit**, you can quickly build and customize smart contracts while enabling extensions to unlock more functionality. Whether you're launching your smart contract with the **Deploy**, importing an already deployed contract, or exploring pre-built contracts through **Explore**, it streamlines the first steps of decentralized development.

Upon deployment, using **Dashboard**, you can easily manage your smart contract and track its performance through **Data Feeds**.

Applications

thirdweb offers software development kits (SDKs) that allow seamless integration with your smart contract when developing applications or games using popular programming languages such as React, React Native, TypeScript, Python, Go, and Unity.

Our SDKs enable wallet connectivity and provide specialized sets of functionality for common Ethereum patterns and standards (i.e., ERC-20, ERC-721, ERC-1155) based on your contract, remote procedure call (RPC) URLs, and an IPFS Gateway. These features optimize performance and enhance developer experience when building decentralized applications.

Infrastructure

thirdweb offers a comprehensive suite of tools and technologies to facilitate the development of user-friendly and accessible decentralized applications. Our infrastructure includes options for fiat on-ramps, wallet integrations, and gasless relayers, providing seamless end-user experiences.

In addition, we provide built-in storage solutions and RPCs (remote procedure calls or connections to the blockchain) to optimize application performance and decentralization. Our indexer and Data Feeds enable easy integration of your contract's data into your application.

Our Auth SDK is designed to support decentralized login and seamless integration with client and server frameworks. This feature allows developers to build secure and user-friendly decentralized applications that can easily be deployed and integrated with other applications.

Solutions

We are committed to providing complete toolkits for developers across various industries, including gaming, e-commerce, finance, art, social media, and more. Our solutions toolkits incorporate a range of tools and technologies, proprietary and in partnership with industry leaders.

GamingKit, developed in partnership with Coinbase, offers a streamlined approach to building blockchain-based games. It includes pre-built contracts such as Marketplace, Multiwrap, Packs, Tokens, and more, and a Unity SDK to build your game. GamingKit also enables easy fiat on-ramp solutions via Coinbase Pay.

Similarly, CommerceKit, developed in collaboration with Shopify, enables the creation of blockchain-based storefronts that can reward customers with digital assets. The toolkit includes features for enabling token-gated commerce, loyalty programs, and easy configuration tools for digital collectible sales/royalty fees.

Our Minting solution enables creative minting mechanics and managing digital assets created on both EVM Contracts and Solana's programs. It includes tools for signature-based minting, releasing collections of unique NFTs, wrapping tokens into newly wrapped NFTs, randomized loot boxes, and more.

Edit this page

Was this page helpful? 👍 Yes 👎 No

Previous
« Home

Next
Templates »

thirdweb documentation

thirdweb is a complete web3 development framework that provides everything you need to connect your apps and games to decentralized networks.

Getting Started

Discover how to build and launch a web3 application in your preferred language.

React

React Native

TypeScript

Python

Go

Unity

Solidity

Tools

Solidity SDK

Base contracts that can be configured with extensions to meet your specific use case

Explore

Discover and deploy contracts from world-class protocols & developers in 1-click

SDK

Powerful SDKs for every stack. Support for Javascript, React, Python, Go, Node.js, Unity

UI Components

Plug and play UI components to easily add web3 functionality to your apps

Deploy

Deploy contracts on-chain with a simple deployment workflow designed for team collaboration

Publish

Publish your contracts to be discovered by our community of 70k+ web3 developers

Dashboard

Manage, analyze, and interact with all your deployed contracts conveniently from a single place

CLI

A suite of commands that let you interface with thirdweb tools through your terminal

Infrastructure

Auth

Authenticate users with their wallets. Securely verify a user's on-chain identity without relying on a centralized database

Storage

Get fast access to data stored on blockchain with a unified API that works with a storage provider of your choice

Wallet

Integrate wallet connection capabilities into web3 applications

Solutions

CommerceKit

An all-in-one toolkit to build web3 commerce apps. Add powerful web3 features to your Shopify

GamingKit

Everything you need to build web3 games. Build a stronger community around your game by

Minting

A set of tools that allow creators and developers to launch new NFT collections in a configurable, storefront enabling tokengated commerce, NFT loyalty programs, digital collectible sales, and more.

giving players ownership of in-game assets

secure and scalable way at zero cost to the creator.

Mobile

A set of developer tools and SDK's that make it easy to integrate web3 features into iOS and Android applications.

Digital Collectibles

Comprehensive set of developer tools designed to help digital collectible projects seamlessly integrate web3 features into your digital commerce and customer experience programs.

Additional Resources

Guides

Learn more about how you can use thirdweb to build your web3 app

Templates

Kickstart your project using one of our templates with a 1-line command

YouTube

Learn web3 development by building working apps with video tutorials

Events

Attend live workshops and office hours to get hands-on with code

Discord

Join our community of developers building the future of web3

Support

Troubleshooting articles and live support from customer support team

Next
Overview »