**Getting real world price data from Chainlink**

**Introduction:**

The `AggregatorV3Interface` provides a streamlined ABI for interacting with the Data Feed contract. In this lesson, we’ll explore how to retrieve pricing information from it.

**Creating a New Contract Instance:**

The `AggregatorV3Interface` includes the `latestRoundData` function, which retrieves the latest cryptocurrency price from the specified contract. We’ll start by declaring a new variable, `priceFeed`, of type `AggregatorV3Interface`. This interface requires the address of the Data Feed contract deployed on the Sepolia Network.

AggregatorV3Interface priceFeed = AggregatorV3Interface(0x1b44F3514812d835EB1BDB0acB33d3fA3351Ee43);

**The `latestRoundData` Function:**

We can call the `latestRoundData()` function on this interface to obtain several values, including the latest price.

Function latestRoundData() external view returns (uint80 roundId, int256 answer, uint256 startedAt, uint256 updatedAt, uint80 answeredInRound);

For now, we’ll focus on the `answer` value and ignore the other returned values by using commas as placeholders for the unneeded variables.

Function getLatestPrice() public view returns (int) {

(,int price,,,) = priceFeed.latestRoundData();

Return price;

}

Our `getLatestPrice()` function now retrieves the latest ETH price in USD from the `latestRoundData()` function of the Data Feed contract. The returned price is an `int256` w”th a precision of 1e8, as indicated by the `decimals` function.

**Decimals:**

* `msg.value` is a `uint256` value with 18 decimal places.
* `answer` is an `int256` value with 8 decimal places (USD-based pairs use 8 decimal places, while ETH-based pairs use 18 decimal places

This means the `price` returned from our `latestRoundData` function isn’t directly compatible with `msg.value`. To match the decimal places, we multiply `price` by 1e10:

return price \* 1e10;

**Typecasting:**

Typecasting, or type conversion, involves converting a value from one data type to another. In Solidity, not all data types can be converted due to differences in their underlying representations and the potential for data loss. However, certain conversions, such as from `int` to `uint`, are allowed.

return uint(price) \* 1e10;

We can finalize our `view` function as follows:

Function getLatestPrice() public view returns (uint256) {

(,int answer,,,) = priceFeed.latestRoundData();

Return uint(answer) \* 1e10;

}

**Conclusion:**

This complete `getLatestPrice` function retrieves the latest price, adjusts the decimal places, and converts the value to an unsigned integer, making it compatible for its use inside other functions.

**Updates:**

Changes in Chainlink Data Feeds:

Last updated on August 23, 2024

Recently Chainlink Data Feeds changed the code to get the data feeds to:

Function getChainlinkDataFeedLatestAnswer() public view returns (int) {

// prettier-ignore

(

/\* uint80 roundID \*/,

Int answer,

/\*uint startedAt\*/,

/\*uint timeStamp\*/,

/\*uint80 answeredInRound\*/

) = dataFeed.latestRoundData();

Return answer;

}

As you may notice this is quite different from what is shown on the video, so please be sure to check the Chainlink Docs to be sure you are using the latest version: <https://docs.chain.link/data-feeds/using-data-feeds>