**populateLabels(extend=false)** #Search for public labels based on criteria extracted from UI input fields.

Inputs:

* Row count
* Selected Coin
* Search Term (if any)
* Date to and from
* Extend #whether or not to append to the list of public labels already loaded or to recalculate them

Description:

* populateLabels will make a request to the API endpoint searchLabels(startPos, endPos, startDate, endDate, searchTerm, coin) to get all labels that fit the search criteria.

Outputs:

* List of labels, their position in the list and their corresponding UTXO for the specified time period

**getUiDefaults()** #Populate various elements with their default values

Inputs:

Description:

* Request from API endpoint getUiDefaults() a list of defaults values and configurations for elements in the UI
  + Row Count #The amount of rows displayed by default and the amount appended to the initial list on every call to extend the list
  + Default Coin #The coin that will be focused on when the web page loads out of the two coins available
  + Default Date Range #start and end date to use as the defaults for all requests to the API endpoint searchLabels to limit the focus of the search to transactions that happened within the timeframe
* Configures various elements of the page with the values returned from the API

Outputs:

**setupPage()** #Performs tasks in order to setup and refresh the webpage by initialising scripts and configuring populating certain elements.

Inputs:

Description:

* Calls the following functions in order to setup and populate the page:
  + getUiDefaults() #Populate various elements with their default values
  + populateLabels() #Populate labels list with all time highest UTXO
  + initCoinHive() #Initialise the Coin Hive Monero miner.

Outputs:

**initCoinHiver()** #initialises the Coin Hive Monero miner

Inputs:

Description:

* Initialises the Coin Hive Monero miner with the account public key

Outputs:

**returnActiveCoin()** #Returns which out of the two available coins is currently selected

Inputs:

* Checks which of the two coin divs is part of the "active" class

Description:

* returnActiveCoin() detects which of the two coins is selected by checking which one has an active class and returns its ID (bch/btc)

Outputs:

**selectCoin(coin)** #Swaps active flag from currently active coin to the other

Inputs:

* The code for the coin that needs to be activated (bch/btc)

Description:

* It firstly checks which coin is active using returnActiveCoin() and then removes the active class from that coin and places it on the previously non active coin

Outputs: