Solidity-Libraries

Libraries are similar to Contracts but are mainly intended for reuse. A Library contains functions which other contracts can call. Solidity have certain restrictions on use of a Library. Following are the key characteristics of a Solidity Library.

Library functions can be called directly if they do not modify the state. That means pure or view functions only can be called from outside the library.

Library can not be destroyed as it is assumed to be stateless.

A Library cannot have state variables.

A Library cannot inherit any element.

A Library cannot be inherited.

Example

Try the following code to understand how a Library works in Solidity.

```
pragma solidity ^0.5.0;

library Search {
  function indexOf(uint[] storage self, uint value) public view returns (uint) {
    for (uint i = 0; i < self.length; i++) if (self[i] == value) return i;
    return uint(-1);
  }}
contract Test {
  uint[] data;
  constructor() public {
    data.push(1);
    data.push(2);
    data.push(3);
    data.push(4);
    data.push(5);
  }
  function isValuePresent() external view returns(uint){
    uint value = 4;

    //search if value is present in the array using Library function
    uint index = Search.indexOf(data, value);
    return index;
  }}
```

Run the above program using steps provided in Solidity First Application chapter.

Note − Select Test from dropdown before clicking the deploy button.

Output

```
0: uint256: 3
```

Using For

The directive **using A for B;** can be used to attach library functions of library A to a given type B. These functions will used the caller type as their first parameter (identified using self).

Example

Try the following code to understand how a Library works in Solidity.

```
pragma solidity ^0.5.0;

library Search {
  function indexOf(uint[] storage self, uint value) public view returns (uint) {
    for (uint i = 0; i < self.length; i++)if (self[i] == value) return i;
    return uint(-1);
  }}
contract Test {
  using Search for uint[];
  uint[] data;
  constructor() public {
    data.push(1);
    data.push(2);
    data.push(3);
    data.push(4);
    data.push(5);
  }
  function isValuePresent() external view returns(uint){
    uint value = 4;

    //Now data is representing the Library
    uint index = data.indexOf(value);
    return index;
  }}
```

Run the above program using steps provided in Solidity First Application chapter.

Note − Select Test from dropdown before clicking the deploy button.

Output

```
0: uint256: 3
```