Web3 Basics

The web3 library is an open source JavaScript library that can be used to connect to Ethereum nodes from the same or a remote computer. It allows IPC as well as RPC to connect to Ethereum nodes. web3 is a client-side library and can be used alongside a web page and query and can submit transactions to Ethereum nodes.

Before we start we need some tools and dependencies. Please install the following:

- Node.js and npm (comes with Node)
- Git
- Geth

Step 1: Create a Project Directory

Let's create a project folder in a new console window:

```
$ mkdir web3_basics

$ cd web3_basics
```

Next, run the npm init command to create a package.json file, which will store project dependencies:

```
$ npm init
```

Step 2: Create a simple solidity contract file SimpleStorage.sol

```
pragma solidity ^0.5.0;
contract SimpleStorage {

  uint public storedData;

 function set(uint x) public {
     storedData = x;
  }

  function get() public view returns (uint retVal) {
     return storedData;
  }
}
```

Step 3: Install Project Dependencies

```
$ npm install web3@1.0.0-beta.48 --save
$ npm install solc@0.5.0 --save
```

Step 4: Open node console

From Command Prompt, enter the node workspace by executing the node command

```
$ node
>
```

It is a quick and easy way to test simple Node.js/JavaScript code.

For Example

```
$ node
>for (i=0;i<=10;i++){ console.log(i) }
```

Step 5: Execute Following Commands one by one

Once in the node workspace, type the following commands to connect to an Ethereum node. The Ethereum node could be a Geth-based private network.

```
// Import libraries
>Web3 = require('web3');
>solc = require('solc');
>fs = require('fs');
// Setup RPC connection
>web3 = new Web3(new
Web3.providers.HttpProvider('http://localhost:8545'));
//get accounts
>web3.eth.getAccounts().then(e => console.log(e));
//first Account
>var firstAccount;
>web3.eth.getAccounts().then(e => {
 firstAccount = e[0];
 console.log("A: " + firstAccount);
})
//get Account Balance

>web3.eth.getBalance(firstAccount, (err, wei) => {
  balance = web3.utils.fromWei(wei, 'ether')
console.log(balance);
```

```
})

//Read content of Smart Contract
>CONTRACT_FILE = 'SimpleStorage.sol'
>content =fs.readFileSync(CONTRACT_FILE).toString()
//JSON formatted input
>input = {
  language: 'Solidity',
  sources: {
    [CONTRACT_FILE]: {
      content: content
    }
  },
  settings: {
    outputSelection: {
      '*': {
        '*': ['*']
      }
    }
  }
}

//Compile input using solc
>compiled = solc.compile(JSON.stringify(input))
//Json form of compiled output
>output = JSON.parse(compiled)
//Take abi from output
>abi = output.contracts[CONTRACT_FILE]['SimpleStorage'].abi
//Take bytecode from output
>bytecode =
output.contracts[CONTRACT_FILE]['SimpleStorage'].evm.bytecode.object
//Contract Instance
>SimpleStorage = new web3.eth.Contract(abi);
//Deployment of contract
//send
>SimpleStorageTx = SimpleStorage.deploy({data: '0x' +
bytecode}).send({from:firstAccount , gas: 1000000})
//ContractAddress
>contractAddress=SimpleStorage.address
//send
SimpleStorage.methods.set(12).send({from:firstAccount })

//call
SimpleStorage.methods.get().call({from: firstAccount}).then((result)
=> {
    console.log(result);
});
```