1) Demonstrate how we can interact with Smart Contract usingWeb3.js.

Ans. Web3.js is a library collection of data which is used for users to interact with Smart Contracts using local or remote ethereum nodes. Interacting with a smart contract using Web3.js involves a few steps, including setting up a Web3 instance, getting the contract's ABI (Application Binary Interface), creating a contract instance, and then calling the contract's method. Below is a step-by-step guide on how to do this.

1. **Install Web3**

   First, you need to have Node.js installed on your machine. Then, you can install Web3.js using npm:

   ```
   npm install web3
   ```

2. **Set up Web**

   Next you need to set up a web3 instance. You need a provider which can be a local ethereum node or remote, or a service like Infura.

   ```
   const Web3 = require('web3');

   // Connect to a local Ethereum node or Infura

   const web3 = new Web3('https://mainnet.infura.io/v3/YOUR_INFURA_PROJECT_ID');
   ```

3. **Get the Contact's ABI Address**

   You need the ABI and the contract address to interact with the smart contract. The ABI defines the contract's interface.

   ```
   const contractABI = [ /* ABI goes here */ ];

   const contractAddress = '0xYourContractAddress';
   ```

4. **Create a contract instance**

   Now create a contract instance using ABI and contract address

   ```
   const myContract = new web3.eth.Contract(contractABI, contractAddress);
   ```

5. **Call Contract Methods**

   You can call methods of the contract. For example, to call a 'view' function

```
myContract.methods.someViewFunction().call()
  .then(result => {
    console.log(result);
  })
  .catch(error => {
    console.error(error);
  });
```

6. **Listening to Events**

   You can also listen for events emitted by the Smart Contact:

```
myContract.events.MyEvent({
  filter: {}, // Filter options (optional)
  fromBlock: 'latest' // Start block
}, (error, event) => {
  if (error) {
    console.error(error);
  } else {
    console.log(event);
  }
});
```

Summary

1. Set up Web3 with a provider.
2. Obtain the ABI and address of the contract.
3. Create a contract instance using Web3.
4. Call or send methods on the contract using ‘.call()’ or ‘.send()’.
5. Listen to events if needed.

This basic setup allows you to interact with any Ethereum Smart Contract using Web3.js.

Summary of Outputs

- No direct output during the setup of Web3 and contact instance creation.
- Contract Call returns the result of the function
- Sending a transaction return a transaction receipt
- Event listener logs emitted events.

To get actual outputs, we need to run the code with a real Ethereum node or service like Infura, using an actual contract ABI, address and valid account credentials.