

# Smart Contract Interaction

Your frontend can interact with different blockchains using the built-in BOS API. Let's see how to create an application that reads and stores a greeting from a NEAR smart contract.

View of our Hello Near app when the user is logged-in

info Check the finished example at [near.social code page](#) .

## The Contract

We have deployed aHello World smart contract in the NEAR network at `hello.near-examples.near` . The contract exposes two methods:

- `set_greeting(greeting: string): void`  
• , which accepts a greeting and stores it in the contract state.
- `get_greeting(): string`  
• which returns the stored greeting.

## Retrieving the Greeting

Since we want to interact with the NEAR network, we will use the `Near` object from the BOS API.

```
const contract =  
"hello.near-examples.near" ; const greeting =  
Near . view ( contract ,  
"get_greeting" ,  
{ } ) ;  
return  
< div  
    { greeting }  
World < / div  
    ; Assuming the contract is storing "Hello" , this will render a simple:  
Hello World
```

## Changing the Greeting

To modify the greeting, we simply need to use `Near.call` to call the `set_greeting` method. This however, requires us to have a frontend in which the user can input the new greeting.

Lets create it in two steps:

1. Build the HTML that will be rendered
2. Add the logic to handle the function call

### 1. HTML Components

Use the following code to create a simple frontend, composed by a title, an input form to change the greeting, and a button to submit the change.

```
const contract =  
"hello.near-examples.near" ; const greeting =  
Near . view ( contract ,  
"get_greeting" ,  
{ } ) ;
```

```
// Define components const greetingForm =
( <
  < div className = "border border-black p-3"
    < label
      Update greeting < / label
    < input placeholder = "Howdy" onChange = { onInputChange }
  /
  < button className = "btn btn-primary mt-2" onClick = { onBtnClick }
    Save < / button
  < / div
  < /
);
const notLoggedInWarning =
< p
  Login to change the greeting < / p
  ;
// Render return
( <
  < div className = "container border border-info p-3"
    < h3 className = "text-center"
      The contract says : < span className = "text-decoration-underline"
        { greeting }
      < / span
    < / h3
  < p className = "text-center py-2"
    Look at that !
  A greeting stored on the NEAR blockchain . < / p
  { context . accountId
    ? greetingForm : notLoggedInWarning } < / div
  < /
  ) ; Relevant HTML There are two important things to notice in the code above:
```

1. onChange & onClick
2. : We have prepared our
3. and
4. to act when something happens. Particularly, we will build two methods: one when the input changes, and one when the button is pressed.
5. context.accountId
6. : We check if context.accountId
7. is set, which tells us if the user has logged in using their NEAR account, and thus can interact with NEAR contracts.

## 2. Handling User's Input

Having our component's view ready, we now need to define the logic for when the user inputs a new greeting and presses the `Submit` button. This is, we need to define the `onInputChange` and `onBtnClick` methods.

### **onInputChange**

When the user inputs a new greeting, we want to store it somewhere until the `Submit` button is pressed, for this, we can use the [application's State](#).

In BOS, the state is initialized through `State.init`, updated with `State.update`, and accessed through the `state` variable (notice the lowercase). Lets store the new greeting in the App's state:

```
State . init ( {  
  new_greeting :  
    ""  
} ) ;  
  
const  
onInputChange  
=  
( { target } )  
=>  
{ State . update ( {  
  new_greeting : target . value  
} ) ; } ;
```

### **onBtnClick**

The only thing left to do, is to handle when the user clicks the `Submit` button. What we want is to check if the user changed the greeting, and submit it to the contract.

```
const  
onBtnClick  
=  
( )  
=>  
{ if  
  ( ! state . new_greeting )  
  { return ; }  
  Near . call ( contract ,  
    "set_greeting" ,  
    { greeting : state . new_greeting , } ) ; } ;
```

## **Complete Example**

We have deployed a complete version of this example on the NEAR blockchain, so you can see its code and play with it.

tip \* Code \* : Check the code of this example at the [near.social code page](#) . \* Try It \* : Interact with the application at the [near.social page](#) . [Edit this page](#) Last updated on Feb 9, 2024 by gagdiez Was this page helpful? Yes No

[Previous BOS Loader](#) [Next Design Components](#)