

Payable Methods

We can allow methods to accept token transfer together with the function call. This is done so that contracts can define a fee in tokens that needs to be paid when they are used. By default the methods are not payable and they will panic if someone will attempt to transfer tokens to them during the invocation. This is done for safety reason, in case someone accidentally transfers tokens during the function call.

To declare a method as payable, use the `#[payable]` annotation within the [near_bindgen macro](#) as follows:

[payable]

```
pub
fn
my_method ( & mut
self )
{ ... } This will allow themy_method function to be called and transfer balance to the contract.
```

Example:

[near_bindgen]

```
impl
Contract
{
```

[payable]

```
pub
fn
take_my_money ( & mut
self )
{ near_sdk :: env :: log_str ( "Thanks!" ) ; } pub
fn
do_not_take_my_money ( & mut
self )
{ near_sdk :: env :: log_str ( "Thanks!" ) ; } } is equivalent to:
```

[near_bindgen]

```
impl
Contract
{ pub
fn
take_my_money ( & mut
self )
{ near_sdk :: env :: log_str ( "Thanks!" ) ; } pub
```

```
fn
do_not_take_my_money ( & mut
self )
{ if
near_sdk :: env :: attached_deposit ( )
!=
0
{ near_sdk :: env :: panic_str ( "Method do_not_take_my_money doesn't accept deposit" ) ; } near_sdk :: env :: log_str (
"Thanks!" ) ; } }
```

[Previous Private Methods](#) [Next Serialization Protocols](#)