TiStripe

TiStripe is a re-implementation of stripe.js for Titanium and includes paymentKitUI, a Titanium API for PaymentKit by Stripe (https://github.com/stripe/PaymentKit)

Accessing the module

`Ti.include('libs/TiStripe.js');`
Change the 'libs' directory to wherever you place the TiStripe.js file

The Object API

The object API is an easy, flexible, extensible way to quickly build payment forms. The easiest way to understand how to use the Object API is to look at the example project - it's well commented. Note that when you add a text field through the Object API, it uses an auto-focuser that lets the focus of the form jump to the next available field, if it's not already filled out. **It does this in the order that the elements were added to the Object API.**

Below is the full Object API for reference.

TiStripe.paymentKitUI.ccImg(view) - takes a passed view (NOT AN IMAGEVIEW) that will become the credit card image

TiStripe.paymentKitUI.ccNumTextField(textField) - takes a passed textView that should be a credit card number. You do not need to add any options for it to add as expected, but you can extend the fields and forms by overwriting some of the default attributes. Options are:

ccStyleNum - boolean, default: `true` - whether the credit card number should be formatted into groups. See https://stripe.com/docs/testing for what the groupings are

ccNextFocus - boolean, default: `true` - whether the field should be focused by the auto-focuser

ccVerify - boolean, default: `true` - whether the expiration date should be validated when the entire number is entered

ccOnFailVerify - function, default: `function(tf) {}` - this function gets called when ccVerify is `true`, and it fails verification. The object that is passed is the

ccOnBlur - function, default: `function() {}` - right before the field is blurred by the auto-focus, this function is called. If you `return false`, the field will not be blurred

ccOnFocus - function, default: `function(nextField) {}` - right before the field is focused by auto-focusing, this function will be called if it exists. The next field that should be focused will be passed to the function. If you overwrite the function, you will need to add `nextField.focus()` or the next field will next focus

ccMaxLength - READONLY int - tells you how many characters the field can contain. Note that the default is `19` or `16` depending on whether `ccStyleNum` is `true` or `false`. Once 2 characters are entered, it will update based on what card type it is. You can use this for checking whether the field is fully filled out or not. See examples

TiStripe.paymentKitUI.ccExpTextField(textField) - takes a passed textView that should be an expiration date. You do not need to add any options for it to add as expected, but you can extend the fields and forms by overwriting some of the default attributes. Options are:

ccNextFocus - boolean, default: `true` - whether the field should be focused by the auto-focuser

ccVerify - boolean, default: `true` - whether the credit card number should be validated when the entire number is entered

ccOnFailVerify - function, default: `function(tf) {}` - this function gets called when ccVerify is `true`, and it fails verification. The object that is passed is the

ccOnBlur - function, default: `function() {}` - right before the field is blurred by the auto-focus, this function is called. If you `return false`, the field will not be blurred

ccOnFocus - function, default: `function(nextField) {}` - right before the field is focused by auto-focusing, this function will be called if it exists. The next field that should be focused will be passed to the function. If you overwrite the function, you will need to add `nextField.focus()` or the next field will next focus

ccMaxLength - READONLY int - will always be `5`

TiStripe.paymentKitUI.ccCVCTextField(textField) - takes a passed textView that should be an CVC code. You do not need to add any options for it to add as expected, but you can extend the fields and forms by overwriting some of the default attributes. Options are:

ccFlipCCImage - boolean, default: `true` - whether the ccImg should rotate when the CVC field is focused

ccNextFocus - boolean, default: `true` - whether the field should be focused by the auto-focuser

ccVerify - boolean, default: `true` - whether the CVC code should be validated when the entire number is entered

ccOnFailVerify - function, default: `function(tf) {}` - this function gets called when ccVerify is `true`, and it fails verification. The object that is passed is the

ccOnBlur - function, default: `function() {}` - right before the field is blurred by the auto-focus, this function is called. If you `return false`, the field will not be blurred

ccOnFocus - function, default: `function(nextField) {}` - right before the field is focused by auto-focusing, this function will be called if it exists. The next field that should be focused will be passed to the function. If you overwrite the function, you will need to add `nextField.focus()` or the next field will next focus

ccMaxLength - READONLY int - will always be `3`

TiStripe.paymentKitUI.ccUSZipTextField(textField) - takes a passed textView that should be a US Zip code. You do not need to add any options for it to add as expected, but you can extend the fields and forms by overwriting some of the default attributes. Options are:

ccNextFocus - boolean, default: `true` - whether the field should be focused by the auto-focuser

ccVerify - boolean, default: `true` - whether the US Zip code should be validated when the entire number is entered

ccOnFailVerify - function, default: `function(tf) {}` - this function gets called when ccVerify is `true`, and it fails verification. The object that is passed is the

ccOnBlur - function, default: `function() {}` - right before the field is blurred by the auto-focus, this function is called. If you `return false`, the field will not be blurred

ccOnFocus - function, default: `function(nextField) {}` - right before the field is focused by auto-focusing, this function will be called if it exists. The next field that should be focused will be passed to the function. If you overwrite the function, you will need to add `nextField.focus()` or the next field will next focus

ccMaxLength - READONLY int - will always be `5`

TiStripe.paymentKitUI.ccCAZipTextField(textField) - takes a passed textView that should be a Canadian Zip code. You do not need to add any options for it to add as expected, but you can extend the fields and forms by overwriting some of the default attributes. Options are:

ccNextFocus - boolean, default: `true` - whether the field should be focused by the auto-focuser

ccVerify - boolean, default: `true` - whether the Canadian Zip code should be validated when the entire number is entered

ccOnFailVerify - function, default: `function(tf) {}` - this function gets called when ccVerify is `true`, and it fails verification. The object that is passed is the

ccOnBlur - function, default: `function() {}` - right before the field is blurred by the auto-focus, this function is called. If you `return false`, the field will not be blurred

ccOnFocus - function, default: `function(nextField) {}` - right before the field is focused by auto-focusing, this function will be called if it exists. The next field that should be focused will be passed to the function. If you overwrite the function, you will need to add `nextField.focus()` or the next field will next focus

ccMaxLength - READONLY int - will always be `6`

TiStripe.paymentKitUI.ccTextField(textField) - takes a passed textView. This has no default styling, but you can add `ccMaxLength`, and it will auto-focus the next field. Options are:

ccNextFocus - boolean, default: `true` - whether the field should be focused by the auto-focuser

ccVerify - boolean, default: `true` - whether the Canadian Zip code should be validated when the entire number is entered

ccOnFailVerify - function, default: `function(tf) {}` - this function gets called when ccVerify is `true`, and it fails verification. The object that is passed is the

ccOnBlur - function, default: `function() {}` - right before the field is blurred by the auto-focus, this function is called. If you `return false`, the field will not be blurred

ccOnFocus - function, default: `function(nextField) {}` - right before the field is focused by auto-focusing, this function will be called if it exists. The next field that should be focused will be passed to the function. If you overwrite the function, you will need to add `nextField.focus()` or the next field will next focus

ccMaxLength - int, default: `null` - setting this as an integer will let it blur and auto-focus the next field

TiStripe.paymentKitUI.baUSRoutingTextField(textField) - takes a passed textView that should be a US routing number. You do not need to add any options for it to add as expected, but you can extend the fields and forms by overwriting some of the default attributes. Options are:

baNextFocus - boolean, default: `true` - whether the field should be focused by the auto-focuser

baVerify - boolean, default: `true` - whether the US routing number should be validated when the entire number is entered

baOnFailVerify - function, default: `function(tf) {}` - this function gets called when ccVerify is `true`, and it fails verification. The object that is passed is the

baOnBlur - function, default: `function() {}` - right before the field is blurred by the auto-focus, this function is called. If you `return false`, the field will not be blurred

baOnFocus - function, default: `function(nextField) {}` - right before the field is focused by auto-focusing, this function will be called if it exists. The next field that should be focused will be passed to the function. If you overwrite the function, you will need to add `nextField.focus()` or the next field will next focus

baMaxLength - READONLY int - will always be `9`

TiStripe.paymentKitUI.baCARoutingTextField(textField) - takes a passed textView that should be a Canadian routing number. You do not need to add any options for it to add as expected, but you can extend the fields and forms by overwriting some of the default attributes. Options are:

baNextFocus - boolean, default: `true` - whether the field should be focused by the auto-focuser

baVerify - boolean, default: `true` - whether the Canadian Zip code should be validated when the entire number is entered

baOnFailVerify - function, default: `function(tf) {}` - this function gets called when ccVerify is `true`, and it fails verification. The object that is passed is the

baOnBlur - function, default: `function() {}` - right before the field is blurred by the auto-focus, this function is called. If you `return false`, the field will not be blurred

baOnFocus - function, default: `function(nextField) {}` - right before the field is focused by auto-focusing, this function will be called if it exists. The next field that should be focused will be passed to the function. If you overwrite the function, you will need to add `nextField.focus()` or the next field will next focus

baMaxLength - READONLY int - will always be `9`

TiStripe.paymentKitUI.baUSAccountTextField(textField) - takes a passed textView that should be a US bank account number. You do not need to add any options for it to add as expected, but you can extend the fields and forms by overwriting some of the default attributes. NOTE that there is no `baMaxLength` on `baUSAccountTextField`. Options are:

baNextFocus - boolean, default: `true` - whether the field should be focused by the auto-focuser

baVerify - boolean, default: `true` - whether the Canadian Zip code should be validated when the entire number is entered

baOnFailVerify - function, default: `function(tf) {}` - this function gets called when ccVerify is `true`, and it fails verification. The object that is passed is the

baOnBlur - function, default: `function() {}` - right before the field is blurred by the auto-focus, this function is called. If you `return false`, the field will not be blurred

baOnFocus - function, default: `function(nextField) {}` - right before the field is focused by auto-focusing, this function will be called if it exists. The next field that should be focused will be passed to the function. If you overwrite the function, you will need to add `nextField.focus()` or the next field will next focus

TiStripe.paymentKitUI.baCAAccountTextField(textField) - takes a passed textView that should be a Canadian bank account number. You do not need to add any options for it to add as expected, but you can extend the fields and forms by overwriting some of the default attributes. NOTE that there is no `baMaxLength` on `baCAAccountTextField`. Options are:

baNextFocus - boolean, default: `true` - whether the field should be focused by the auto-focuser

baVerify - boolean, default: `true` - whether the Canadian Zip code should be validated when the entire number is entered

baOnFailVerify - function, default: `function(tf) {}` - this function gets called when ccVerify is `true`, and it fails verification. The object that is passed is the

baOnBlur - function, default: `function() {}` - right before the field is blurred by the auto-focus, this function is called. If you `return false`, the field will not be blurred

baOnFocus - function, default: `function(nextField) {}` - right before the field is focused by auto-focusing, this function will be called if it exists. The next field that should be focused will be passed to the function. If you overwrite the function, you will need to add `nextField.focus()` or the next field will next focus

The TiStripe API

TiStripe

object

setPublishableKey(key) - sets the publishable key provided by Stripe. DO NOT use your secret key.

getPublishableKey() - returns the publishable key

TiStripe.options

publishableKey - string, default: `null` - your Stripe provided publishable key. DO NOT use your secret key. You should use setPublishableKey instead of setting the option directly.

stripeVersion - int, default: `2` - the version of stripe.js that this is based on. It's not used, but included because it's in stripe.js. It may be used in the future.

stripeEndpoint - string, default: `'https://api.stripe.com/v1'` - the base URL of the API we're using to connect to Stripe.

onConnectionError - function, default:

```
function(e) { // receives the HTTPClient error object
      Ti.API.info('There was a connection error:');
      Ti.API.info(e);
      Ti.API.info(e.error);
}
```

when the module fails to connect to stripe, this function is called. You can overwrite with your own function so you can receive the error object from HTTPClient. See example in stripe-slide.js

imagePath - string, default: `'images/'` - overwrite if you want to put your images in a different directory than a directory named “images” in the same directory as TiStripe.js, ex `TiStripe.options.imagePath = 'cc-images/'` - it should include the trailing slash

flipDuration - int, default: `400` - the time in milliseconds it takes to flip the credit card images. This is used as a variable for `TiStripe.paymentKitUI.ccImg()`

TiStripe.utils

object

trim(str) - strips whitespace

serialize(object, result, scope) - returns a string that is serialized from the supplied object and url encoded

underscore(str) - returns a string with dashes (-) replaced with underscores (_)

underscoreKeys(object) - returns an array with the keys have dashes (-) replaced with underscores (_)

validateKey(key) - validates the provided stripe publishable key. If false, throws error

splitExpiry(string) - splits an expiration date string (MM/YY or MM/YYYY) and returns an object containing a month and year ex.

```
{
      month: 12,
      year: 2015
}
```

TiStripe.token

object

validate(data, name) - if an invalid token, throws an error

formatData(data, attrs) - returns a formatted data object

create(params, callback) - creates a Stripe token. Takes the params to send to create the token, then calls callback with the response

get(token, callback) - gets information about the passed token from Stripe and calls callback with the response

TiStripe.card

object

createToken(data, params, callback) - creates a Stripe token from a passed data object, and calls callback with the response

getToken(token, callback) - gets information about the passed token from Stripe and calls callback with the response

validateCardNumber(num) - client side algorithm that checks whether the passed credit card number *could be* valid. Returns `boolean`

validateCVC(num) - client side algorithm that checks whether the passed CVC code could be valid. Returns `boolean`

validateExpiry(month. year) - takes an expiration date and determines if it's in the future. Returns `boolean`

luhnCheck(num) - client side algorithm to determine whether a passed credit card number *could be* valid. Is called by `validateCardNumber`. Returns `boolean`

cardType(num) - determines what type of credit card based on passed number. Returns string of credit card type or 'Unknown' ex. 'Visa', 'Mastercard'

TiStripe.bankAccount

object

createToken(data, params, callback) - creates a Stripe token from a passed data object, and calls callback with the response

getToken(token, callback) - gets information about the passed token from Stripe and calls callback with the response

validateRoutingNumber(num, country) - client side algorithm to determine if a passed number *could be* valid. Country is a string. ‘US’ or ‘CA’. Returns `boolean`

validateAccountNumber(num, country) - client side algorithm to determine if a passed number *could be* valid. Country is a string. ‘US’ or ‘CA’. Returns `boolean`

routingChecksum(num) - client side algorithm to determine if a passed number *could be* valid. Returns `boolean`

TiStripe.validator

object - stripe.js validator object

TiStripe.paymentKitUI

object

addCCFields(tiObj) - takes a Titanium object, and adds it to the credit card auto-focus list

getCCFields() - returns the list of objects to step through in order

ccImg() - takes a tiObj View (NOT imageView) that will display the credit card images

updateCCImg(num) - updates the ccImg() to the correct image based on the number passed

setupCCFlip() - hack to prepare the ccImg() to be flipped (sigh, Titanium)

flipCCImgToBack(duration) - takes duration is milliseconds and flips the ccImg() view to the back (CVC image side)

flipCCImgToCard(duration) - takes duration is milliseconds and flips the ccImg() view to the front (card logo side)

ccNumTextField(tiObj) - identifies the passed tiObj as a field that should be styled and validated as a credit card number. See examples

ccExpTextField(tiObj) - identifies the passed tiObj as a field that should be styled and validated as an expiration date. See examples

ccCVCTextField(tiObj) - identifies the passed tiObj as a field that should be styled and validated as a CVC code. See examples

ccUSZipTextField(tiObj) - identifies the passed tiObj as a field that should be styled and validated as a US zip code. See examples

ccCAZipTextField(tiObj) - identifies the passed tiObj as a field that should be styled and validated as a Canadian Zip code. See examples

ccTextField(tiObj) - identifies the passed tiObj as a field that should be a part of the credit card form. Use Object API to style and validate. See examples

baUSRoutingTextField(tiObj) - identifies the passed tiObj as a field that should be styled and validated as a US bank routing number. See examples

baCARoutingTextField(tiObj) - identifies the passed tiObj as a field that should be styled and validated as a Canadian bank routing number. See examples

baUSAccountTextField(tiObj) - identifies the passed tiObj as a field that should be styled and validated as a US bank account code. See examples

baCAAccountTextField(tiObj) - identifies the passed tiObj as a field that should be styled and validated as a CA bank account code. See examples

baTextField(tiObj) - identifies the passed tiObj as a field that should be a part of the bank account form. Use Object API to style and validate. See examples