Jest Test Documentation

SaltOS 4.0 r2031

May 2025

Contents

1 Jest Libraries

1.1 Setup file for the unit tests

```
lib/jest.setup.js
```

This file contains the code that initialize the unit tests

1.1.1 Detect what type of test is running, and load the needed setup

```
if ('fetch' in global) {
```

1.2 Setup file for the unit tests

```
lib/jsdom.setup.js
```

This file contains the code that initialize the unit tests

1.2.1 Needed by all user interface features

```
global.bootstrap = require('../../code/web/lib/bootstrap/bootstrap.bundle.
    min.js');
```

1.2.2 Needed by bootstrap, core and proxy modules

```
global.md5 = require('../../code/web/lib/md5/md5.min.js');
```

1.2.3 Needed by bootstrap module

```
global.window.matchMedia = function() {
```

1.2.4 This is the same that object.js for the global scope

```
global.saltos = {};
```

1.2.5 Load all files of the project

```
/*const files = `core,bootstrap,storage,hash,token,auth,window,
```

1.2.6 My Require

```
global.myrequire = (file, fns)
```

This function is intended to add the needed module.exports at the end of the file that you want to process, this is a temporary action used only for the require action, and at the end, before the return, the original code is saved as a restore action to maintain the original code file

1.3 Setup file for the unit tests

```
lib/node.setup.js
```

This file contains the code that initialize the unit tests

1.3.1 This is the same that object.js for the global scope

```
global.saltos = {};
```

1.3.2 Needed by core module

```
global.window = {
```

1.3.3 Needed by core module

```
global.document = {
```

1.3.4 Load all files of the project

```
require(`../../code/web/js/core.js`);
```

1.3.5 My Pause

```
global.mypause = (page, delay)
```

This function is intended to do a pause inside the browser, to do it, we use a string instead of real code because istanbul tries to inject code and fails in runtime, one solution can be to put "istanbul ignore next" in a comment before the next page.evaluate, but I prefer to use a string in the page.evaluate because it is more simple for me

2 Jest Code

2.1 Apps unit tests

```
test_apps.js
```

This file contains the apps unit tests

2.1.1 Puppeteer setup

```
const puppeteer = require('puppeteer');
```

This lines contain the needed setup for run puppeteer and take screenshots

2.1.2 Global variables

```
let browser;
```

This variables contains the browser and page links

2.1.3 Before All

```
beforeAll(async ()
```

This function contains all code executed before all tests, in this case the features provided by this function includes the launch of the browser, set the screen size and start the javascript coverage

2.1.4 After All

```
afterAll(async ()
```

This function contains all code executed after all tests, in this case the features provided by this function include the stop of the javsacript coverage recording, the save feature to the desired storage path and the browser close

2.1.5 Workflow variables

```
let testFailed = false;
```

This variables allow to control the workflow of the test, the main idea is to skip all tests when one test fails

2.1.6 Before Each

```
beforeEach(()
```

This function contains all code executed before each test, in this case the features provided by this function includes the control of the workflow

2.1.7 After Each

```
afterEach(()
```

This function contains all code executed after each test, in this case the features provided by this function includes the control of the workflow

2.1.8 App Login

```
describe('App Login', ()
```

This test is intended to validate the correctness of the login screen and their particularities, too tries to check the dashboard and the logout to close the loop of tests

2.1.9 Action Login

```
test('Action Login', async ()
```

This function tries to do a test to validate that the login screen appear correctly without issues

2.1.10 Action Login Ko Red

```
test('Action Login Ko Red', async ()
```

This function tries to validate the form when no data is found

2.1.11 Action Login Ko Green

```
test('Action Login Ko Green', async ()
```

This function tries to validate the form when invalid data is found

2.1.12 Action Dashboard

```
test('Action Dashboard', async ()
```

This function tries to execute the login with valid credentials, to validate the correctness, the dashboard will appear

2.1.13 Action Reload

```
test('Action Reload', async ()
```

This function tries to do a reload of the previous page, the validation of this test must accomplish when a valid dashbord appear

2.1.14 Action Logout

```
test('Action Logout', async ()
```

This test is intended to check the correctness of the logout feature, to to it the test tries to locate the logout button placed in the dropdown menu of the nabvar and click to the third element

2.2 Bootstrap unit tests

```
test_bootstrap.js
```

This file contains the bootstrap unit tests

2.2.1 Puppeteer setup

```
const puppeteer = require('puppeteer');
```

This lines contain the needed setup for run puppeteer and take screenshots

2.2.2 Bootstrap

```
describe('Bootstrap', ()
```

This test contains the code needed to create all widgets and validate the correctness of them

2.2.3 Global variables

```
let browser;
```

This variables contains the browser and page links

2.2.4 Before All

```
beforeAll(async ()
```

This function contains all code executed before all tests

2.2.5 After All

```
afterAll(async ()
```

This function contains all code executed after all tests

2.2.6 Prepare the test.each iterator

```
const fs = require('fs');
```

2.2.7 Real test

```
test.each(json)('$label', async field
```

This function executes the real test for each json item, its able to create a widget and validate the correctness of the widget comparing the new widget screenshot to the backup widget screenshot

2.3 Core unit tests

```
test_core.js
```

This file contains comprehensive unit tests for the core functionality of the SaltOS framework, including error handling, AJAX operations, utility functions, and service worker integration.

2.3.1 Load all needed files of the project

```
const files = `core,gettext,token`.split(',');
```

2.3.2 Reset mocks before each test

```
beforeEach(()
```

Initializes mock implementations for console.log and fetch API before each test case runs

2.3.3 Restore mocks after each test

```
afterEach(()
```

Restores original implementations of mocked functions after each test case completes

2.3.4 Test suite for error and log reporting

```
describe('saltos.core.adderror/addlog', ()
```

Contains tests for core error handling and logging functionality

2.3.5 Setup before each test in this suite

```
beforeEach(()
```

Mocks core dependencies including AJAX, token, and gettext functions

2.3.6 Test error reporting functionality

```
test('saltos.core.adderror should send error details to the server', async
    ()
```

Verifies that error details including stack traces are properly processed and sent to the server

2.3.7 Test log reporting functionality

```
test('saltos.core.addlog should send log message to the server', ()
```

Verifies that log messages are properly formatted and sent to the server

2.3.8 Test suite for global error handling

```
describe('window.addEventListener for error and unhandledrejection', ()
```

Contains tests for window-level error and unhandled promise rejection handlers

2.3.9 Setup before each test in this suite

```
beforeEach(()
```

Mocks the core error reporting function

2.3.10 Test global error handling

```
test('should call saltos.core.adderror when an error occurs', ()
```

Verifies that window error events are properly captured and reported

2.3.11 Test unhandled promise rejection handling

```
test('should call saltos.core.adderror when an unhandledrejection occurs',
    ()
```

Verifies that unhandled promise rejections are properly captured and reported

2.3.12 Test parameter validation functionality

```
test('saltos.core.check_params', ()
```

Verifies that missing parameters are properly initialized with default values

2.3.13 Test unique ID generation

```
test('saltos.core.uniqid', ()
```

Verifies that generated IDs are unique and follow expected format

2.3.14 Test suite for visibility detection

```
describe('when_visible', ()
```

Contains tests for the when_visible utility function

2.3.15 Setup before each test in this suite

```
beforeEach(()
```

Configures fake timers and clears DOM

2.3.16 Cleanup after each test in this suite

```
afterEach(()
```

Clears timers and restores real timer implementation

2.3.17 Test basic visibility detection

```
test('executes the callback when the object becomes visible', ()
```

Verifies callback is executed when element becomes visible

2.3.18 Test visibility detection with ID

```
test('works setting the id as test-div', ()
```

Verifies function works with elements that have IDs

2.3.19 Test visibility detection with string ID

```
test('works with an string id instead of an object', ()
```

Verifies function works when passed an element ID string

2.3.20 Test invalid input handling

```
test('throws an error for unsupported obj type', ()
```

Verifies function throws error for unsupported input types

2.3.21 Test delayed element attachment

```
test('append the object after some iterations', ()
```

Verifies function works when element is added to DOM after initialization

2.3.22 Test element removal handling

```
test('throws an error if the object disappears before being visible', ()
```

Verifies function throws error if element is removed before becoming visible

2.3.23 Test key code extraction

```
test('saltos.core.get_keycode', ()
```

Verifies function correctly extracts key codes from different event properties

2.3.24 Test key name resolution

```
test('saltos.core.get_keyname', ()
```

Verifies function correctly maps key codes to common key names

2.3.25 Test suite for HTML element creation

```
describe('saltos.core.html', ()
```

Contains tests for the html utility function

2.3.26 Test single argument usage

```
test('creates a div with inner HTML when only one argument is passed', ()
```

Verifies function creates element from HTML string with optimization

2.3.27 Test two argument usage

```
test('creates the specified element with inner HTML when two arguments are
    passed', ()
```

Verifies function creates specified element type with content

2.3.28 Test HTML trimming

```
test('trims the inner HTML before setting it', ()
```

Verifies function trims whitespace from HTML content

2.3.29 Test multiple children handling

```
test('does not optimize if there are multiple children', ()
```

Verifies function preserves container element when multiple children exist

2.3.30 Test single child optimization

```
test('optimizes and returns the single child if present', ()
```

Verifies function optimizes by returning single child element directly

2.3.31 Test suite for AJAX functionality

```
describe('saltos.core.ajax', ()
```

Contains comprehensive tests for the core AJAX implementation

2.3.32 Test successful GET request

```
test('makes a successful GET request and calls success callback', async ()
```

Verifies proper request construction and success callback handling

2.3.33 Test error response handling

```
test('handles non-200 responses and calls error callback', async ()
```

Verifies proper error callback invocation for non-200 responses

2.3.34 Test POST request handling

```
test('makes a POST request with a body', async ()
```

Verifies proper construction of POST requests with body content

2.3.35 Test request abortion

```
test('calls abort callback when request is aborted', async ()
```

Verifies proper handling of aborted requests

2.3.36 Test network failure handling

```
test('calls error callback on network failure', async ()
```

Verifies proper error callback invocation for network failures

2.3.37 Test unexpected error handling

```
test('throws an error for unexpected failures', async ()
```

Verifies proper propagation of unexpected errors

2.3.38 Test unsupported method handling

```
test('throws an error for unsupported HTTP method', ()
```

Verifies proper error throwing for unsupported HTTP methods

2.3.39 Test XML response handling

```
test('handles XML response correctly', async ()
```

Verifies proper parsing of XML response content

2.3.40 Test plain text response handling

```
test('handles plain text response correctly', async ()
```

Verifies proper handling of plain text responses

2.3.41 Test key normalization

```
test('saltos.core.fix_key', ()
```

Verifies function properly removes suffixes from keys

2.3.42 Test object copying

```
test('saltos.core.copy_object', ()
```

Verifies function creates proper shallow copies of objects

2.3.43 Test suite for resource loading

```
describe('saltos.core.require', ()
```

Contains tests for the require functionality

2.3.44 Setup before each test in this suite

```
beforeEach(()
```

Configures fake timers for testing asynchronous operations

2.3.45 Cleanup after each test in this suite

```
afterEach(()
```

Restores real timer implementation

2.3.46 Test JavaScript file loading

```
test('loads a JavaScript file successfully', async ()
```

Verifies proper loading and injection of JavaScript files

2.3.47 Test CSS file loading

```
test('loads a CSS file successfully', async ()
```

Verifies proper loading and injection of CSS files

2.3.48 Test cached resource handling

```
test('does not reload already loaded files', async ()
```

Verifies that already loaded resources are not reloaded

2.3.49 Test pending resource handling

```
test('waits when file is already loading', async ()
```

Verifies proper waiting behavior for resources already being loaded

2.3.50 Test module file loading

```
test('loads a JavaScript module file successfully', async ()
```

Verifies proper handling of JavaScript module files

2.3.51 Test hashed resource loading

```
test('loads a CSS file with hash successfully', async ()
```

Verifies proper handling of resources with cache-busting hashes

2.3.52 Test boolean evaluation

```
test('saltos.core.eval_bool', ()
```

Verifies proper conversion of various values to boolean

2.3.53 Test string conversion

```
test('saltos.core.toString', ()
```

Verifies proper string conversion of various values

2.3.54 Test attribute value detection

```
test('saltos.core.is_attr_value', ()
```

Verifies proper identification of attribute-value objects

2.3.55 Test attribute-value joining

```
test('saltos.core.join_attr_value', ()
```

Verifies proper merging of attribute and value objects

2.3.56 saltos.core.encode_bad_chars

```
test('saltos.core.encode_bad_chars', ()
```

This function performs the test of the encode_bad_chars function

2.3.57 saltos.core.__get_code_from_file_and_line

```
test('saltos.core.__get_code_from_file_and_line', ()
```

This function performs the test of the __get_code_from_file_and_line function

2.3.58 saltos.core.timestamp

```
test('saltos.core.timestamp', ()
```

This function performs the test of the timestamp function

2.3.59 saltos.core.human_size

```
test('saltos.core.human_size', ()
```

This function performs the test of the human_size function

2.3.60 saltos.core.is_number

```
test('saltos.core.is_number', ()
```

This function performs the test of the is_number function

2.3.61 saltos.core.is_function

```
test('saltos.core.is_function', ()
```

This function performs the test of the is_function function

2.3.62 Core Module Tests

```
describe('Core Module Tests', ()
```

This test suite validates the core functionalities of the SaltOS framework, ensuring that essential features operate correctly under different scenarios.

2.3.63 Test service worker registration

```
test('Registers service worker if supported and on HTTPS', async ()
```

This test checks if the service worker is registered properly when the browser supports service workers and the application is running over HTTPS.

2.3.64 Test proxy function messaging

```
test('Proxy function sends message to service worker', ()
```

This test ensures that the proxy function sends the correct message to the service worker's controller using the 'postMessage' method.

2.3.65 Test proxy synchronization on online events

```
test('Triggers proxy sync on online event', ()
```

This test verifies that the proxy synchronization is triggered correctly when the browser's online event is fired.

2.3.66 Test network protocol detection

```
test('Check network detects protocols correctly', async ()
```

This test checks that the 'check_network' function accurately detects HTTP and HTTPS protocols by simulating network conditions.

2.4 Customers unit tests

```
test_customers.js
```

This file contains the customers unit tests

2.4.1 Puppeteer setup

```
const puppeteer = require('puppeteer');
```

This lines contain the needed setup for run puppeteer and take screenshots

2.4.2 Global variables

```
let browser;
```

This variables contains the browser and page links

2.4.3 Before All

```
beforeAll(async ()
```

This function contains all code executed before all tests, in this case the features provided by this function includes the launch of the browser, set the screen size and start the javascript coverage

2.4.4 After All

```
afterAll(async ()
```

This function contains all code executed after all tests, in this case the features provided by this function include the stop of the javsacript coverage recording, the save feature to the desired storage path and the browser close

2.4.5 Workflow variables

```
let testFailed = false;
```

This variables allow to control the workflow of the test, the main idea is to skip all tests when one test fails

2.4.6 Before Each

```
beforeEach(()
```

This function contains all code executed before each test, in this case the features provided by this function includes the control of the workflow

2.4.7 After Each

```
afterEach(()
```

This function contains all code executed after each test, in this case the features provided by this function includes the control of the workflow

2.4.8 App Customers

```
describe('App Customers', ()
```

This test is intended to validate the correctness of the customers application by execute the list, more, reset, create, cancel, view, close, edit, back, insert, update and delete features and validate with the expected screenshot

2.4.9 Action List

```
test('Action List', async ()
```

This part of the test tries to load the list screen

2.4.10 Action List

```
test('Action More', async ()
```

This part of the test tries to validate the correctness of the more feature

2.4.11 Action Reset

```
test('Action Reset', async ()
```

This part of the test tries to validate the correctness of the reset feature

2.4.12 Action Create

```
test('Action Create', async ()
```

This part of the test tries to load the create screen

2.4.13 Action Cancel

```
test('Action Cancel', async ()
```

This part of the test tries to validate the correctness of the cancel feature

2.4.14 Action View

```
test('Action View', async ()
```

This part of the test tries to load the view screen

2.4.15 Action Close

```
test('Action Close', async ()
```

This part of the test tries to validate the correctness of the close feature

2.4.16 Action Edit

```
test('Action Edit', async ()
```

This part of the test tries to load the edit screen

2.4.17 Action Go Back

```
test('Action Go Back', async ()
```

This part of the test tries to validate the correctness of the go back feature

2.4.18 Action Insert

```
test('Action Insert', async ()
```

This part of the test tries to validate the correctness of the insert feature

2.4.19 Action Update

```
test('Action Update', async ()
```

This part of the test tries to validate the correctness of the update feature

2.4.20 Action Delete

```
test('Action Delete', async ()
```

This part of the test tries to validate the correctness of the delete feature

2.5 Emails unit tests

```
test_emails.js
```

This file contains the emails unit tests

2.5.1 Puppeteer setup

```
const puppeteer = require('puppeteer');
```

This lines contain the needed setup for run puppeteer and take screenshots

2.5.2 Global variables

```
let browser;
```

This variables contains the browser and page links

2.5.3 Before All

```
beforeAll(async ()
```

This function contains all code executed before all tests, in this case the features provided by this function includes the launch of the browser, set the screen size and start the javascript coverage

2.5.4 After All

```
afterAll(async ()
```

This function contains all code executed after all tests, in this case the features provided by this function include the stop of the javsacript coverage recording, the save feature to the desired storage path and the browser close

2.5.5 Workflow variables

```
let testFailed = false;
```

This variables allow to control the workflow of the test, the main idea is to skip all tests when one test fails

2.5.6 Before Each

```
beforeEach(()
```

This function contains all code executed before each test, in this case the features provided by this function includes the control of the workflow

2.5.7 After Each

```
afterEach(()
```

This function contains all code executed after each test, in this case the features provided by this function includes the control of the workflow

2.5.8 App Emails

```
describe('App Emails', ()
```

This test is intended to validate the correctness of the emails application by execute the list, control+f, profile, help, filter, create and view features and validate with the expected screenshot

2.5.9 Action List

```
test('Action List', async ()
```

This action tries to sets the emails as new and not new of all emails that appear in the screen

2.5.10 Action Control F

```
test('Action Control F', async ()
```

This part of the test tries to validate the control+f focus

2.5.11 Action Profile

```
test('Action Profile', async ()
```

This part of the test tries to load the profile screen

2.5.12 Action Help

```
test('Action Help', async ()
```

This part of the test tries to load the help screen

2.5.13 Action Filter

```
test('Action Filter', async ()
```

This part of the test tries to load the filter screen

2.5.14 Action Create

```
test('Action Create', async ()
```

This part of the test tries to load the create screen

2.5.15 Action View

```
test('Action View', async ()
```

This part of the test tries to load the view screen

2.6 Filter unit tests

```
test_filter.js
```

This file contains unit tests for the filter management system including initialization, loading, saving, and UI interactions

2.6.1 Load all needed files of the project

```
const files = `app,backup,core,driver,filter,form,gettext,hash,storage,
    token`.split(',');
```

2.6.2 Reset mocks before each test

```
beforeEach(()
```

Ensures all Jest mocks are reset before each test case runs

2.6.3 Restore mocks after each test

```
afterEach(()
```

Ensures all Jest mocks are restored to their original implementations after each test case completes

2.6.4 Test suite for filter initialization

```
describe('saltos.filter.init()', ()
```

Contains tests for initializing the filter cache from server data

2.6.5 Setup before each test in this suite

```
beforeEach(()
```

Mocks window location and sets up AJAX response for filter data

2.6.6 Test filter cache initialization

```
test('should initialize filter cache', async ()
```

Verifies that the init function makes an AJAX call to load filters and properly populates the cache

2.6.7 Test cache reuse

```
test('should not make ajax call if cache is already populated', async ()
```

Verifies that subsequent init calls don't make AJAX requests when cache is already populated

2.6.8 Test suite for filter loading

```
describe('saltos.filter.load()', ()
```

Contains tests for loading filter data into forms and triggering searches

2.6.9 Setup before each test in this suite

```
beforeEach(()
```

Mocks form data and driver search functions

2.6.10 Test loading existing filter

```
test('should load filter data and trigger search', ()
```

Verifies that loading a filter populates form data and triggers a search

2.6.11 Test loading non-existent filter

```
test('should only trigger search if filter not found', ()
```

Verifies that loading an unknown filter only triggers search without modifying form data

2.6.12 Test suite for filter updates

```
describe('saltos.filter.update()', ()
```

Contains tests for updating the local filter cache

2.6.13 Test updating filter cache

```
test('should update filter cache', ()
```

Verifies that update modifies the filter cache correctly

2.6.14 Test suite for filter saving

```
describe('saltos.filter.save()', ()
```

Contains tests for saving filters to both cache and server

2.6.15 Test saving new filter

```
test('should save new filter to cache and server', ()
```

Verifies that new filters are added to cache and saved to server

2.6.16 Test updating existing filter

```
test('should update existing filter if data changed', ()
```

Verifies that modified filters are updated in cache and server

2.6.17 Test unchanged filter

```
test('should not make server call if data unchanged', ()
```

Verifies that unchanged filters don't trigger server updates

2.6.18 Test filter deletion

```
test('should delete filter when data is null', ()
```

Verifies that filters are removed from cache and server when set to null

2.6.19 Test deleting non-existent filter

```
test('should not delete non-existent filter', ()
```

Verifies that attempting to delete unknown filters doesn't trigger server calls

2.6.20 Test suite for filter UI buttons

```
describe('saltos.filter.button()', ()
```

Contains tests for all filter-related UI actions

2.6.21 Setup before each test in this suite

```
beforeEach(()
```

Sets up DOM elements and mocks filter functions

2.6.22 Test load action

```
test('should handle load action', ()
```

Verifies that load button loads selected filter and resets selection

2.6.23 Test update action

```
test('should handle update action', ()
```

Verifies that update button saves current form data to selected filter

2.6.24 Test delete action

```
test('should handle delete action', ()
```

Verifies that delete button removes selected filter and refreshes UI

2.6.25 Test create action

```
test('should handle create action', ()
```

Verifies that create button saves current form data as new filter

2.6.26 Test rename action

```
test('should handle rename action', ()
```

Verifies that rename button moves filter data to new name and removes old filter entry

2.6.27 Test invalid action

```
test('should do nothing for invalid actions', ()
```

Verifies that unknown actions don't modify filter cache

2.6.28 Test suite for filter selection UI

```
describe('saltos.filter.select()', ()
```

Contains tests for updating filter dropdown and tree view

2.6.29 Setup before each test in this suite

```
beforeEach(()
```

Populates filter cache with test data

2.6.30 Test UI update

```
test('should update select options and jstree', ()
```

Verifies that select updates dropdown options and tree view

2.6.31 Test missing tree element

```
test('should handle missing form elements gracefully', ()
```

Verifies graceful handling when tree element is missing

2.6.32 Test missing select element

```
test('should handle missing select elements gracefully', ()
```

Verifies graceful handling when select element is missing

2.6.33 Test missing form elements

```
test('should handle missing form elements gracefully', ()
```

Verifies graceful handling when all form elements are missing

2.7 Core unit tests

```
test_gettext.js
```

This file contains the core unit tests

2.7.1 Load all needed files of the project

```
const files = 'bootstrap,core,gettext,storage'.split(',');
```

2.7.2 berofeEach used in this test

```
beforeEach(()
```

2.7.3 afterEach used in this test

```
afterEach(()
```

2.7.4 saltos.gettext.bootstrap.set/get/get_short/unset

```
test('saltos.gettext.set/get/get_short/unset', ()
```

This function performs the tests of the set/get/get_short and unset functions

2.7.5 saltos.gettext.bootstrap.field

```
test('saltos.gettext.bootstrap.field', ()
```

This function performs the tests of the field function

2.7.6 saltos.gettext.bootstrap.modal

```
test('saltos.gettext.bootstrap.modal', ()
```

This function performs the tests of the modal function

2.7.7 saltos.gettext.bootstrap.toast

```
test('saltos.gettext.bootstrap.toast', ()
```

This function performs the tests of the toast function

2.7.8 saltos.gettext.bootstrap.menu

```
test('saltos.gettext.bootstrap.menu', ()
```

This function performs the tests of the menu function

2.7.9 saltos.gettext.bootstrap.offcanvas

```
test('saltos.gettext.bootstrap.offcanvas', ()
```

This function performs the tests of the offcanvas function

2.8 Hash unit tests

```
test_hash.js
```

This file contains unit tests for hash management functionality including hash parsing, manipulation, and event triggering

2.8.1 Load all needed files of the project

```
const files = `hash`.split(',');
```

2.8.2 Reset mocks before each test

```
beforeEach(()
```

Initializes mock implementations for history API functions and resets all mocks between test cases

2.8.3 Restore mocks after each test

```
afterEach(()
```

Restores original implementations of mocked functions after each test case completes

2.8.4 Test suite for hash helper function

```
describe('saltos.hash.__helper', ()
```

Contains tests for the internal hash normalization function that cleans hash strings by removing special characters

2.8.5 Test hash with leading

```
test('should remove leading #', ()
```

Verifies the helper removes the # character from the beginning

2.8.6 Test hash with leading /

```
test('should remove leading /', ()
```

Verifies the helper removes the / character from the beginning

2.8.7 Test hash with leading #/

```
test('should remove both leading # and /', ()
```

Verifies the helper removes both # and / characters from the beginning

2.8.8 Test empty hash

```
test('should return empty string for empty input', ()
```

Verifies the helper returns empty string for empty input

2.8.9 Test clean hash

```
test('should not modify clean hash', ()
```

Verifies the helper returns unchanged input when no special characters

2.8.10 Test suite for hash getter function

```
describe('saltos.hash.get', ()
```

Contains tests for retrieving the current window hash with proper formatting

2.8.11 Test getting current hash

```
test('should return current hash without #', ()
```

Verifies the function returns hash without # prefix

2.8.12 Test getting empty hash

```
test('should return empty string when no hash', ()
```

Verifies the function returns empty string when no hash exists

2.8.13 Test getting hash with leading /

```
test('should remove leading /', ()
```

Verifies the function removes leading / from the hash

2.8.14 Test suite for hash setter function

```
describe('saltos.hash.set', ()
```

Contains tests for replacing the current hash using history.replaceState

2.8.15 Test setting new hash

```
test('should set new hash with proper format', ()
```

Verifies the function updates hash with proper format and uses replaceState

2.8.16 Test setting same hash

```
test('should not set same hash', ()
```

Verifies the function doesn't update history when hash doesn't change

2.8.17 Test setting hash with

```
test('should handle hash with #', ()
```

Verifies the function properly handles input containing #

2.8.18 Test setting hash with /

```
test('should handle hash with /', ()
```

Verifies the function properly handles input containing /

2.8.19 Test setting empty hash

```
test('should handle empty hash', ()
```

Verifies the function properly handles empty hash input

2.8.20 Test suite for hash adder function

```
describe('saltos.hash.add', ()
```

Contains tests for adding new hash entries using history.pushState

2.8.21 Test adding new hash

```
test('should add new hash with proper format', ()
```

Verifies the function updates hash with proper format and uses pushState

2.8.22 Test adding same hash

```
test('should not add same hash', ()
```

Verifies the function doesn't update history when hash doesn't change

2.8.23 Test adding hash with

```
test('should handle hash with #', ()
```

Verifies the function properly handles input containing #

2.8.24 Test adding hash with /

```
test('should handle hash with /', ()
```

Verifies the function properly handles input containing /

2.8.25 Test adding empty hash

```
test('should handle empty hash', ()
```

Verifies the function properly handles empty hash input

2.8.26 Test suite for URL hash extraction

```
describe('saltos.hash.url2hash', ()
```

Contains tests for extracting hash fragments from URLs

2.8.27 Test extracting hash from URL

```
test('should extract hash from URL', ()
```

Verifies the function extracts hash fragment correctly

2.8.28 Test extracting hash with /

```
test('should handle URL with / after #', ()
```

Verifies the function handles hashes containing /

2.8.29 Test URL without hash

```
test('should return empty string when no hash', ()
```

Verifies the function returns empty string when no hash exists

2.8.30 Test suite for hash change events

```
describe('saltos.hash.trigger', ()
```

Contains tests for triggering hashchange events

2.8.31 Test triggering hash change

```
test('should dispatch hashchange event', ()
```

Verifies the function dispatches a hashchange event

2.9 Invoices unit tests

```
test_invoices.js
```

This file contains the invoices unit tests

2.9.1 Puppeteer setup

```
const puppeteer = require('puppeteer');
```

This lines contain the needed setup for run puppeteer and take screenshots

2.9.2 Global variables

```
let browser;
```

This variables contains the browser and page links

2.9.3 Before All

```
beforeAll(async ()
```

This function contains all code executed before all tests, in this case the features provided by this function includes the launch of the browser, set the screen size and start the javascript coverage

2.9.4 After All

```
afterAll(async ()
```

This function contains all code executed after all tests, in this case the features provided by this function include the stop of the javsacript coverage recording, the save feature to the desired storage path and the browser close

2.9.5 Workflow variables

```
let testFailed = false;
```

This variables allow to control the workflow of the test, the main idea is to skip all tests when one test fails

2.9.6 Before Each

```
beforeEach(()
```

This function contains all code executed before each test, in this case the features provided by this function includes the control of the workflow

2.9.7 After Each

```
afterEach(()
```

This function contains all code executed after each test, in this case the features provided by this function includes the control of the workflow

2.9.8 App Invoices

```
describe('App Invoices', ()
```

This test is intended to validate the correctness of the invoices application by execute the list, checkbox, create, view and edit features and validate with the expected screenshot

2.9.9 Action List

```
test('Action List', async ()
```

This part of the test tries to load the list screen

2.9.10 Action List

```
test('Action Checkbox', async ()
```

This part of the test tries to validate the correctness of the checkbox feature

2.9.11 Action Create

```
test('Action Create', async ()
```

This part of the test tries to load the create screen

2.9.12 Action View

```
test('Action View', async ()
```

This part of the test tries to load the view screen

2.9.13 Action Edit

```
test('Action Edit', async ()
```

This part of the test tries to load the edit screen

2.10 Proxy unit tests

```
test_proxy.js
```

This file contains the proxy unit tests

2.10.1 berofeEach used in this test

```
beforeEach(()
```

2.10.2 afterEach used in this test

```
afterEach(()
```

2.10.3 Load the needed environment of the proxy part

```
Object.assign(global, myrequire(
```

2.10.4 md5

```
test('md5', ()
```

This function performs the test of the md5 function

2.10.5 human_size

```
test('human_size', ()
```

This function performs the test of the human_size function

2.11 Push unit tests

```
test_push.js
```

This file contains unit tests for the push notification functionality and favicon status updates in the SaltOS framework.

2.11.1 Load all needed files of the project

```
const files = `app,bootstrap,core,gettext,push,storage,token`.split(',');
```

2.11.2 Reset mocks before each test

```
beforeEach(()
```

Ensures all Jest mocks are reset before each test case runs

2.11.3 Restore mocks after each test

```
afterEach(()
```

Ensures all Jest mocks are restored to their original implementations after each test case completes

2.11.4 Test suite for saltos.push.fn functionality

```
describe('saltos.push.fn', ()
```

Contains tests for the push notification system's main function, including various execution conditions and response handling

2.11.5 Setup before each test in this suite

```
beforeEach(()
```

Initializes mock implementations and resets push notification state

2.11.6 Test push function when already executing

```
test('should return early if executing is true', ()
```

Verifies that the push function exits early when a push operation is already in progress

2.11.7 Test push function without token

```
test('should return early if token is not available', ()
```

Verifies that the push function exits early when no authentication token is available

2.11.8 Test push function when offline

```
test('should return early if navigator is offline', ()
```

Verifies that the push function exits early when the browser is offline

2.11.9 Test push function with positive count

```
test('should return early if count is greater than or equal to 0', ()
```

Verifies that the push function exits early when the countdown counter hasn't reached zero

2.11.10 Test push function with successful response

```
test('should call ajax and handle success response', ()
```

Verifies that the push function makes an AJAX call and properly handles a successful response

2.11.11 Test push function with error response

```
test('should handle error response', ()
```

Verifies that the push function properly handles an error response from the server

2.11.12 Test suite for saltos.favicon.fn functionality

```
describe('saltos.favicon.fn', ()
```

Contains tests for the favicon status update functionality, including visibility state handling

2.11.13 Test favicon function activation

```
test('should start interval if bool is true and executing is false', ()
```

Verifies that the favicon update interval starts when the function is activated

2.11.14 Test favicon function deactivation

```
test('should clear interval if bool is false and executing is true', ()
```

Verifies that the favicon update interval stops when the function is deactivated

2.12 Screenshots unit tests

```
test_screenshots.js
```

This file contains the screenshots unit tests

2.12.1 Puppeteer setup

```
const puppeteer = require('puppeteer');
```

This lines contain the needed setup for run puppeteer and take screenshots

2.12.2 Global variables

```
let browser;
```

This variables contains the browser and page links

2.12.3 Before All

```
beforeAll(async ()
```

This function contains all code executed before all tests, in this case the features provided by this function includes the launch of the browser, set the screen size and start the javascript coverage

2.12.4 After All

```
afterAll(async ()
```

This function contains all code executed after all tests, in this case the features provided by this function include the stop of the javsacript coverage recording, the save feature to the desired storage path and the browser close

2.12.5 Workflow variables

```
let testFailed = false;
```

This variables allow to control the workflow of the test, the main idea is to skip all tests when one test fails

2.12.6 Before Each

```
beforeEach(()
```

This function contains all code executed before each test, in this case the features provided by this function includes the control of the workflow

2.12.7 After Each

```
afterEach(()
```

This function contains all code executed after each test, in this case the features provided by this function includes the control of the workflow

2.12.8 App Customers

```
describe('Screenshots', ()
```

This test is intended to validate the correctness of the customers application by execute the list, more, reset, create, cancel, view, close, edit, back, insert, update and delete features and validate with the expected screenshot

2.12.9 Action List

```
test('users login', async ()
```

This part of the test tries to load the list screen

2.13 Storage unit tests

```
test_storage.js
```

This file contains the storage unit tests

2.13.1 Load all needed files of the project

```
const files = 'core,storage'.split(',');
```

2.13.2 berofeEach used in this test

```
beforeEach(()
```

2.13.3 afterEach used in this test

```
afterEach(()
```

2.13.4 saltos.storage

```
test('saltos.storage', ()
```

This function performs the test of the storage functions

2.14 Tester unit tests

```
test_tester.js
```

This file contains the tester unit tests

2.14.1 Puppeteer setup

```
const puppeteer = require('puppeteer');
```

This lines contain the needed setup for run puppeteer and take screenshots

2.14.2 Global variables

```
let browser;
```

This variables contains the browser and page links

2.14.3 Before All

```
beforeAll(async ()
```

This function contains all code executed before all tests, in this case the features provided by this function includes the launch of the browser, set the screen size and start the javascript coverage

2.14.4 After All

```
afterAll(async ()
```

This function contains all code executed after all tests, in this case the features provided by this function include the stop of the javsacript coverage recording, the save feature to the desired storage path and the browser close

2.14.5 Workflow variables

```
let testFailed = false;
```

This variables allow to control the workflow of the test, the main idea is to skip all tests when one test fails

2.14.6 Before Each

```
beforeEach(()
```

This function contains all code executed before each test, in this case the features provided by this function includes the control of the workflow

2.14.7 After Each

```
afterEach(()
```

This function contains all code executed after each test, in this case the features provided by this function includes the control of the workflow

2.14.8 App Tester

```
describe('App Tester', ()
```

This test is intended to validate the correctness of the tester application by execute the init, disabled, enabled, all bs_themes and all css_themes and validate with the expected screenshot

2.14.9 Action Init

```
test('Action Init', async ()
```

This part of the test tries to initialize the tester screen by provide a valid credentials and loads the tester application

2.14.10 Action Disabled

```
test('Action Disabled', async ()
```

This part of the test tries to disable all widgets

2.14.11 Action Enabled

```
test('Action Enabled', async ()
```

This part of the test tries to enable all widgets

2.14.12 List of bs_themes

```
const bs_themes = ['light', 'dark', 'auto'];
```

2.14.13 Action Bs Theme

```
test.each(bs_themes)('Action Bs Theme %s', async theme
```

This part of the test tries to set the differents bs_themes

2.14.14 List of css_themes

```
const css_themes = ['default', 'cerulean', 'cyborg', 'darkly', 'flatly', '
    journal', 'litera', 'lumen',
```

2.14.15 Action Css Theme

```
test.each(css_themes)('Action Css Theme %s', async theme
```

This part of the test tries to set the differents css_themes

2.15 Token unit tests

```
test_token.js
```

This file contains the token unit tests

2.15.1 Load all needed files of the project

```
const files = `core,storage,token`.split(',');
```

2.15.2 berofeEach used in this test

```
beforeEach(()
```

2.15.3 afterEach used in this test

```
afterEach(()
```

2.15.4 saltos.token

```
test('saltos.token', ()
```

This function performs the test of the token functions

2.16 Window unit tests

```
test_window.js
```

This file contains unit tests for window management functionality including window opening/closing and cross-tab communication

2.16.1 Load all needed files of the project

```
const files = `core,storage,window`.split(',');
```

2.16.2 Reset mocks before each test

```
beforeEach(()
```

Ensures all Jest mocks are reset before each test case runs

2.16.3 Restore mocks after each test

```
afterEach(()
```

Ensures all Jest mocks are restored to their original implementations after each test case completes

2.16.4 Test suite for window open/close functionality

```
describe('saltos.window.open/close', ()
```

Contains tests for opening new windows with different URL types and closing the current window

2.16.5 Setup before each test in this suite

```
beforeEach(()
```

Mocks window.open and window.close functions

2.16.6 Test opening app URLs

```
test('should call window.open with the app prefix', ()
```

Verifies that app URLs are properly prefixed with .#/ when opening new windows

2.16.7 Test opening HTTP URLs

```
test('should call window.open with the http prefix', ()
```

Verifies that HTTP URLs are passed through unchanged when opening new windows

2.16.8 Test opening HTTPS URLs

```
test('should call window.open with the https prefix', ()
```

Verifies that HTTPS URLs are passed through unchanged when opening new windows

2.16.9 Test unsupported URL protocols

```
test('should throw an error when call window.open with non supported
   protocol', ()
```

Verifies that attempting to open URLs with unsupported protocols throws an error

2.16.10 Test window closing

```
test('should call window.close', ()
```

Verifies that the close function properly calls window.close

2.16.11 Test suite for window event listeners

```
describe('saltos.window.listeners', ()
```

Contains tests for cross-tab communication functionality including setting listeners and sending events between tabs

2.16.12 Test setting event listeners

```
test('set_listener should add a listener for a specific event', ()
```

Verifies that listeners can be registered for specific events

2.16.13 Test removing event listeners

```
test('unset_listener should remove a listener for a specific event', ()
```

Verifies that listeners can be removed for specific events

2.16.14 Test sending events to current tab

```
test('send should trigger the listener in the same tab when scope is "me"',
    ()
```

Verifies that events with "me" scope only trigger callbacks in the current tab

2.16.15 Test sending events to other tabs

```
test('send should trigger the listener in other tabs when scope is "other
   "', ()
```

Verifies that events with "other" scope trigger callbacks in other tabs through localStorage events

2.16.16 Test sending events to all tabs

```
test('send should trigger the listener in all tabs when scope is "all"', ()
```

Verifies that events with "all" scope trigger callbacks in all tabs including the current one

2.16.17 Test sessionStorage event filtering

```
test('storage event listener should not trigger if the event is not from
   localStorage', ()
```

Verifies that events from sessionStorage don't trigger the cross-tab communication callbacks

2.16.18 Test event key filtering

```
test('storage event listener should not trigger if the key does not match',
    ()
```

Verifies that events with incorrect keys don't trigger the cross-tab communication callbacks

2.16.19 Test unregistered event filtering

```
test('storage event listener should not trigger if the event name is not in
    listeners', ()
```

Verifies that events for unregistered event names don't trigger any callbacks