
Gemini



Gemini is a utility for regression testing the visual appearance of web pages.

Gemini allows you to:

- Work with different browsers:
 - Google Chrome (tested in latest version)
 - Mozilla Firefox (tested in latest version)
 - IE8+
 - Opera 12+
- Test separate sections of a web page
- Include the `box-shadow` and `outline` properties when calculating element position and size
- Ignore some special case differences between images (rendering artifacts, text caret, etc.)
- Gather CSS test coverage statistics

Gemini was created at Yandex and is especially useful to UI library developers.

Quick start

Installing

```
1 npm install -g gemini
2 npm install -g selenium-standalone
3 selenium-standalone install
```

Configuring

Put the `.gemini.js` file in the root of your project:

```
1 module.exports = {
2   rootUrl: 'http://yandex.ru',
3   gridUrl: 'http://127.0.0.1:4444/wd/hub',
4
5   browsers: {
6     chrome: {
7       desiredCapabilities: {
8         browserName: 'chrome'
```

```
9         }
10     }
11 }
12 };
```

Writing tests

Write a test and put it in the `gemini` folder in the root of your project:

```
1 gemini.suite('yandex-search', (suite) => {
2     suite.setUrl('/')
3     .setCaptureElements('.home-logo')
4     .capture('plain');
5 });
```

Saving reference images

You have written a new test and should save a reference image for it:

```
1 gemini update
```

Running tests

Start `selenium-standalone` in a separate tab before running the tests:

```
1 selenium-standalone start
```

Run gemini tests:

```
1 gemini test
```

Dependencies

Required software:

1. WebDriver server implementation. There are several options:
 - Selenium Server — for testing in different browsers. Launch with the `selenium-standalone start` command (if you will get error like “No Java runtime present, requesting install.” you should install Java Development Kit (JDK) for your OS.).

-
- ChromeDriver — for testing in Google Chrome. Launch with the `chromedriver --port=4444 --url-base=wd/hub` command.
 - PhantomJS — launch with the `phantomjs --webdriver=4444` command.
 - Cloud WebDriver services, such as SauceLabs or BrowserStack
2. Compiler with support for C++11 (GCC@4.6 or higher). This is a png-img requirement. Compiling on Windows machines requires the node-gyp prerequisites.

Installing

To install the utility, use the npm `install` command:

```
1 npm install -g gemini
```

Global installation is used for launching commands.

Configuring

Gemini is configured using a config file at the root of the project. Gemini can use one of the following files: `*.gemini.conf.js` `*.gemini.conf.json` `*.gemini.conf.yml` `*.gemini.js` `*.gemini.json` `*.gemini.yml`

Let's say we want to run our tests only in the locally installed **PhantomJS**.

In this case, the minimal configuration file will only need to have the root URL of your web app and the WebDriver capabilities of **PhantomJS**: For example,

```
1 rootUrl: http://yandex.com
2 browsers:
3   PhantomJS:
4     desiredCapabilities:
5       browserName: phantomjs
```

Also, you need to run **PhantomJS** manually in **WebDriver** mode:

```
1 phantomjs --webdriver=4444
```

If you are using a remote WebDriver server, you can specify its URL with the `gridUrl` option:

```
1 rootUrl: http://yandex.com
2 gridUrl: http://selenium.example.com:4444/wd/hub
3
4 browsers:
5   chrome:
```

```
6     desiredCapabilities:
7       browserName: chrome
8       version: "45.0"
9
10    firefox:
11      desiredCapabilities:
12        browserName: firefox
13        version: "39.0"
```

You can also set up each browser to have its own node:

```
1  rootUrl: http://yandex.com
2
3  browsers:
4    chrome:
5      gridUrl: http://chrome-node.example.com:4444/wd/hub
6      desiredCapabilities:
7        browserName: chrome
8        version: "45.0"
9
10   firefox:
11     gridUrl: http://firefox-node.example.com:4444/wd/hub
12     desiredCapabilities:
13       browserName: firefox
14       version: "39.0"
```

Other configuration options

See the details of the config file structure and available options.

Writing tests

Each of the blocks that are being tested may be in one of the determined states. States are tested with the help of chains of step-by-step actions declared in a block's test suites.

For example, let's write a test for a search block at yandex.com:

```
1  gemini.suite('yandex-search', function(suite) {
2    suite.setUrl('/')
3      .setCaptureElements('.search2__input')
4      .capture('plain')
5      .capture('with text', function(actions, find) {
6        actions.sendKeys(find('.search2__input .input__control'), '
7          hello gemini');
8      });
9  });
```

We are creating a new test suite `yandex-search`, assuming that we will capture the `.search2__input` element from the root URL `http://yandex.com`. We know that the block has two states:

- `plain` — right after the page is loaded
- `with text` — with the `hello gemini` text inserted into `.search2__input .input__control`

States are executed one after another in the order in which they are defined, without the browser reloading in between.

See the details of test creation methods.

Using CLI

To complete the test creation procedure, you need to take reference shots using the following command:

```
1 gemini update [paths to test suites]
```

To launch a test (to compare the current state of a block with a reference shot), use the command:

```
1 gemini test [paths to test suites]
```

See the details of interaction with CLI and available options.

GUI

You can use the `Gemini` graphical user interface instead of the command line. It is located in the `gemini-gui` package and must be installed additionally:

```
1 npm install -g gemini-gui
```

GUI advantages:

- Handy preview of reference shots
- Clear real-time demonstration of the differences between a reference shot and the current state of a block
- Easy to update reference shots

Plugins

Gemini can be extended with plugins. You can choose from the existing plugins or write your own. To use a plugin, install and enable it in your `.gemini.yml`:

```
1 system:
2   plugins:
3     some-awesome-plugin:
4       plugin-option: value
```

HTML report

To see the difference between the current state of a block and a reference picture more clearly, use the HTML reporter - plugin for gemini. This plugin produces HTML report, which displays reference image, current image and differences between them, for each state in each browser. When all tests are completed, you will see a link to HTML report.

Programmatic API

To use Gemini in your scripts or build tools, you can use the experimental programmatic API.

Events

To learn more about all events in Gemini, see the events documentation.