

POLITECHNIKA ŚWIĘTOKRZYSKA W KIELCACH

Wydział Elektrotechniki, Automatyki i Informatyki

Projekt Technologii Obiektowych

Temat projektu:

41 – Porównanie rozwiązań związanych z testowaniem

Autorzy:

Daniel Szwajkowski, Kacper Jurek

Podział pracy:

Daniel Szwajkowski – testy jednostkowe (JUnit, JTest, TestNG),

Kacper Jurek – testy e2e (Cypress, Katalon Studio, Puppeteer).

1. Testy E2E

Głównym celem testowania End-To-End jest testowanie na podstawie doświadczenia użytkownika końcowego poprzez symulację rzeczywistego scenariusza użytkownika i walidację testowanego systemu wraz z komponentami pod kątem integracji i integralności danych. Ten rodzaj testów staje się coraz bardziej istotny, gdyż oprogramowanie jest coraz bardziej złożone i skomplikowane. Możemy dostrzec coraz większą liczbę integracji z innymi systemami, podsystemami, czy też usługami. W przypadku awarii jednego z takich elementów istnieje poważne ryzyko, że cały system może ulec awarii, czy też może nie spełniać swoich głównych założeń w sposób prawidłowy.

Testy E2E są zwykle przeprowadzane na gotowych systemach (w końcowej fazie developmentu – ponieważ muszą już w pełni działać wszystkie funkcjonalności, które będziemy testować).

| Testy funkcjonalne | Testy End to End |
|---|--|
| Testowanie jest ograniczone do jednego modułu, systemu, aplikacji | Testowanie obejmuje wiele aplikacji, usług i bardziej opiera się na pełnej integracji oprogramowania |
| Zapewnia, że testowane oprogramowanie spełnia kryteria akceptacji | Zapewnia, że procesy biznesowe działają prawidłowo, zgodnie z założeniami |
| Testuje sposób, w jaki pojedynczy użytkownik korzysta z aplikacji | Testuje sposób pracy wielu użytkowników w aplikacjach |
| Wykonuje się je na różnych etapach developmentu | Wykonuje się je, wtedy, gdy faza developmentu zbliża się do końca, lub gdy jest już zakończona. |

Jak wygląda cały proces testów E2E

Na standardowy proces E2E składają się następujące fazy:

Zbieranie wymagań testowych (w kontekście pełnych, biznesowych funkcjonalności)

Projektowanie scenariuszy testowych (do wcześniej zebranych wymagań procesów biznesowych)

Faza developmentu – tworzenie programowania

Testowanie End to End – wykonywanie wcześniej opracowanych scenariuszy testowych

Każda aplikacja komunikuje się z wieloma systemami i bazami danych poza własnym środowiskiem. To sprawia, że całościowy przepływ pracy aplikacji jest dość skomplikowany do przetestowania.

Testy End to End określają, czy różne zależności, integracje aplikacji działają poprawnie. Sprawdzają również, czy między wieloma komponentami systemu przesyłane i odbierane są poprawnie informacje.

Backend : testy End to End weryfikują warstwy bazy danych i integracji backendowych. Jest to konieczne, ponieważ podstawowa główna logika systemu dzieje się właśnie „pod spodem”

System wielowarstwowy : jeśli aplikacja ma złożoną architekturę, której przepływ pracy działa na wielu warstwach, testy E2E są niezbędne do weryfikacji ogólnych funkcji, a także interakcji między poszczególnymi warstwami w architekturze.

Środowisko rozproszone : jeśli aplikacja jest oparta na architekturze SOA (architektura zorientowana na usługi) lub środowiskach chmurowych, konieczne jest testowanie End to End. Jest również niezbędny w przypadku aplikacji składających się z wielu komponentów, które muszą działać w tandemie, aby zapewnić pomyślne działanie.

Spójne doświadczenie użytkownika : ponieważ testowanie E2E obejmuje frontend, zapewnia również, że aplikacja zapewnia wrażenia użytkownika działające na wielu urządzeniach, platformach i środowiskach. Na przykład testy zgodności przeglądarek jest ważną częścią testowania E2E w tym zakresie.

Cypress

Cypress to javascriptowy framework przede wszystkim do testów end to end. Twórcy tego frameworka zapewniają nas, że dzięki niemu: konfiguracja, tworzenie, uruchamianie oraz debugowanie testów będzie proste.

Idea cypressa jest zbliżona do koncepcji JEST – to znaczy zebranie wszystkich niezbędnych narzędzi takich jak framework, biblioteki asercji, wrappera czy dodatkowych bibliotek w jeden tool.

Co odróżnia Cypressa od innych narzędzi?

Nie opiera się na Selenium – zbudowany jest na nowej architekturze. Cypress działa w tej samej pętli uruchomieniowej, co twoja aplikacja.

Koncentruje się na sprawnym wykonywaniu testów – jego użycie jest intuicyjne i proste.

Jest uniwersalny – działa prawdopodobnie na wszystkim co może być uruchomione w przeglądarce. Dobrze powinien sobie poradzić nawet na starszych aplikacjach, nie tylko tych używających nowoczesnych frameworków.

Testy są pisane w javascriptcie – dla jednych wada dla innych zaleta ale cypress jest oparty na javascriptcie.

Wszystko czego potrzeba w 1 narzędziu – Nie trzeba instalować osobnych narzędzi aby uruchomić testy.

Narzędzie zarówno dla programistów jak i QA.

Cypress jest szybki – pozwala na równoczesny development i testowanie.

Instalacja

Instalacja Cypressa nie wydaje się niczym skomplikowanym, tym bardziej jeśli miało się już do czynienia z instalacją z użyciem npm.

```
cd /ścieżkaProjektu/
```

Zainstaluj cypressa poleceniem:

```
npm install cypress --save-dev
```

W ten sposób instalujemy najnowszą wersję cypressa lokalnie. I tak naprawdę, gdy instalacja dobiegnie do końca możemy zacząć działać. Możemy też ściągnąć paczkę instalacyjną ze strony cypress.io instalacja z konsoli jest dużo szybsza i wygodniejsza.

Po zainstalowaniu w folderze projektu pojawi się folder `node_modules` oraz plik `package-lock.json`.

By uruchomić Cypressa wystarczy użyć komendy:

node_modules\\.bin\\cypress open

Efektom wykonania tego polecenia powinno być uruchomienie Test Runnera. Po kliknięciu przycisku 'OK, got it' który się pojawił na ekranie runnera pojawi się stworzona struktura danych w naszym folderze projektowym. Niech to będzie nasza baza do nauki Cypressa.

Struktura plików

To jeszcze słów kilka o strukturze. Stworzone zostały 4 główne katalogi, które są godne uwagi:

Fixtures – pliki które zawierają statyczne dane, mogące być użyte podczas testów. Zwykle odnosimy się do nich z użyciem `cy.fixture()`. Używamy gdy chcemy przetestować funkcjonalność opartą na żądaniach sieciowych (xhr,/ ajax) , framework może sztucznie wstawić dane podczas testu czyli pełnić rolę Stub'a.

Test files – pliki testów domyślnie są umiejscowione w folderze integration. Cypress wygenerował dodatkowo folder examples z przykładami użycia.

Plugin files – umożliwiają korzystanie, modyfikowanie, rozszerzanie wewnętrznych zachowań frameworku. Domyślnie Cypress używa pluginów zawartych w katalogu plugins w pliku index.js. Można również pobrać bardzo dużo pluginów dostępnych na stronie cypress, często współtworzonych przez użytkowników.

Support files – tu można wrzucać reużywalne funkcje, niestandardowe polecenia itp które mają być dostępne dla wszystkich plików testowych.

Zalety Cypress:

Selenium obejmuje implementację sterowników przeglądarki dla skryptu do komunikacji z elementami sieciowymi na stronie. Ponieważ jednak przypadki testowe działają bezpośrednio w przeglądarce, instalacja Cypressa nie wymaga żadnych dodatkowych zależności ani pobierania.

Cypress jest używany zarówno przez programistów, jak i inżynierów QA, podczas gdy Selenium służy wyłącznie do automatyzacji testów. Cypress jest oparty na JavaScript, który jest popularnym językiem programowania typu front-end.

W Cypress nie ma dodatkowego obciążenia IDE. Po uruchomieniu Cypress prosi o wybranie IDE w celu zmodyfikowania skryptu testowego.

W porównaniu do Selenium, platforma Cypress zapewnia dokładniejsze wyniki. Dzieje się tak, ponieważ Cypress ma większą kontrolę nad całym procesem automatyzacji, co pozwala mu dobrze rozumieć, co dzieje się w przeglądarce i poza nią.

Instancje Cypress reagują w czasie rzeczywistym na zdarzenia i polecenia aplikacji. Ponowne wczytywanie w czasie rzeczywistym w Cypress automatycznie przeładowuje testy po wprowadzeniu zmian w aplikacji.

Wady Cypress:

Cypress jest obecnie obsługiwany tylko w przeglądarkach Chrome, Firefox, Edge, Brave i Electron. W rezultacie

Cypress jest mniej preferowaną opcją do testowania w różnych przeglądarkach.

Do budowy przypadków testowych obsługuje tylko framework JavaScript.

Cypress nie obsługuje zdalnego wykonywania.

Katalon Studio

Katalon Studio to solidne narzędzie do automatyzacji wydane po raz pierwszy w styczniu 2015 roku z silnikiem opartym na Selenium. Przede wszystkim Katalon jest przeznaczony do tworzenia i ponownego wykorzystywania automatycznych skryptów testowych dla interfejsu użytkownika bez kodowania. Katalon Studio umożliwia przeprowadzanie automatycznych testów elementów interfejsu użytkownika, w tym wyskakujących okienek, ramek iFrames i czasu oczekiwania. Narzędzie można uruchomić w systemach Microsoft Windows, macOS i Linux.

Pierwotnie wydana jako bezpłatne rozwiązanie, w październiku 2019 r. wprowadzono bardziej bogatą w funkcje wersję Katalon Studio Enterprise, a także Katalon Runtime Engine, aby zapewnić elastyczne opcje dla różnych potrzeb. Jednak podstawowe Katalon Studio przeznaczone dla użytkowników indywidualnych nadal pozostaje bezpłatne.

Główną zaletą Katalon jest łatwość wdrożenia i szerszy zestaw integracji w porównaniu do Selenium, lidera rynku. Katalon ma podwójne interfejsy skryptowe dla użytkowników o różnych umiejętnościach programowania. Oznacza to, że testerzy z ograniczoną wiedzą techniczną mogą używać prostszego interfejsu użytkownika, który nie wymaga pisania kodu.

Tryb dla bardziej biegłych użytkowników ma dostęp do skryptów z podświetlaniem składni, sugestiami kodu i debugowaniem. Katalon Studio ma predefiniowaną strukturę artefaktów: wiele

szablonów przypadków testowych, zestawów testów, obiektów testowych i raportów. Katalon Studio obsługuje testowanie lokalne i zdalne, a także wykonywanie równoległych i sekwencyjnych. Działa na języku skryptowym Groovy (Java).

To rozwiązanie ma szybką konfigurację i wiele wstępnie zainstalowanych szablonów, które umożliwiają powtórzenie niektórych wzorców testowych. Katalon Studio to narzędzie obsługujące wiele przeglądarek, które obsługuje testowanie sieci Web, urządzeń mobilnych, pulpitu Windows i API. Te rozwiązania są wyposażone w moduły analityczne i rejestrujące. Przyjrzyjmy się bliżej każdemu produktowi Katalon.

Testowanie API

W przeciwieństwie do Selenium i Ranorex, w Katalon Studio możesz testować interfejsy API bez dodatkowych integracji. Ten wbudowany moduł pozwala testerom przeprowadzać kompleksowe testy API, automatyzować skrypty i utrzymywać testy. Główne funkcje tego modułu to autouzupełnianie kodu, inspekcja kodu, fragment kodu, referencje i debugger. Oprogramowanie obsługuje wszystkie typy żądań REST, SOAP/1.1 i SOAP/1.2 oraz wiele źródeł danych (XLS, XML, bazy danych z dynamicznym mapowaniem w celu maksymalizacji pokrycia testami).

Tryb testowania API umożliwia użytkownikom importowanie testów z takich narzędzi do testowania i edycji API, jak Swagger, Postman i WSDL. Testowanie Katalon API ma również wbudowaną przeglądarkę odpowiedzi z automatycznym formatowaniem i wyszukiwaniem dostępu do artefaktów.

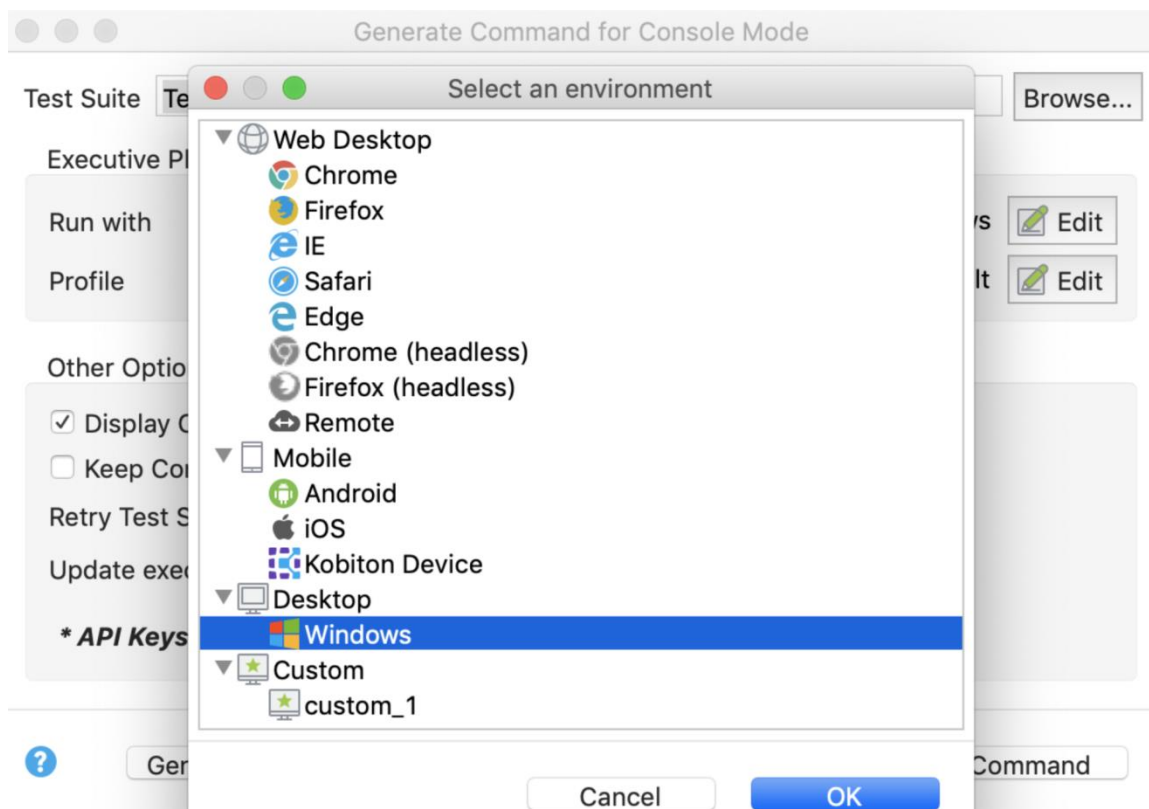
Testy internetowe

Katalon oferuje kompletne rozwiązanie do testowania stron internetowych z wbudowanymi integracjami Continuous Delivery/Continuous Integration i DevOps. Narzędzie jest zgodne z Selenium i ma narzędzia wspomagające kod, takie jak wbudowane szpiegowanie obiektów, refaktoryzacja kodu i odwołania w kontekście. Oferuje również inteligentny mechanizm lokalizacji XPath i szereg opcji raportowania. Poza tym natywne wtyczki łatwo integrują się z Jenkins, Git, JIRA i qTest.

Mechanizm wykonywania testów internetowych firmy Katalon wykorzystuje wiele konfiguracji i zestawów danych. Pozwala również na dostosowanie przepływu pracy i automatyczne uruchamianie go. Moduł umożliwia importowanie zewnętrznych bibliotek w celu usprawnienia funkcji automatyzacji.

Testy mobilne

Ten moduł umożliwia użytkownikom testowanie mobilnej sieci Web, natywnych aplikacji iOS i Android oraz aplikacji hybrydowych. Dzięki natywnej integracji za pomocą Appium, narzędzia do automatyzacji testów aplikacji mobilnych, moduł wspiera testowanie najnowszych platform i urządzeń mobilnych bez dodatkowych instalacji. Katalon przyspiesza testowanie mobilne, wykrywając i przechowując obiekty. Dzięki możliwościom wykonywania w różnych środowiskach testy mogą być wykonywane lokalnie i zdalnie za pomocą rzeczywistych urządzeń, symulatorów lub usług w chmurze.



Wybór środowiska testowego

Źródło: [Katalon Studio Docs](#)

Moduł posiada bogaty zestaw wbudowanych i niestandardowych słów kluczowych Windows wspierających proces testowania. Główne funkcje to Szpiegowanie obiektów systemu Windows (w celu lokalizowania, przechwytywania i analizowania wszystkich obiektów aplikacji) oraz Rejestrowanie akcji systemu Windows (w celu rejestrowania działań testowych, sprawdzania wszystkich uruchomionych obiektów aplikacji i prezentowania ich w widoku drzewa).

Silnik wykonawczy Katalon

Katalon Runtime Engine (KRE) to dodatek Katalon Studio, który umożliwia użytkownikom planowanie i wykonywanie testów automatyzacji w trybie konsoli i trybie CLI (interfejs wiersza poleceń). Istnieją różne scenariusze wdrażania KRE. Może służyć do planowania testów, integracji z systemem CI/CD lub łączenia testów w celu wykonania w wirtualnych kontenerach, takich jak Docker.

Katalon TestOps (dawniej Katalon Analytics)

TestOps to aplikacja internetowa zaprojektowana do organizowania testów i DevOps. Jego kluczowe moduły to zarządzanie testami, planowanie testów i wykonywanie testów. Jest bezproblemowo zintegrowany z innymi produktami Katalon, frameworkami testowymi i narzędziami CI/CD.

Katalon TestOps ma potężne funkcje raportowania i analizy. Narzędzie zapewnia wgląd w działania związane z testowaniem, tworząc raporty z testów w czasie rzeczywistym i umożliwiając użytkownikom monitorowanie jakości testu, pokrycia, łamliwości itp. Obszary testowe mogą być traktowane priorytetowo, podkreślając najważniejsze części. Wszystkie wzorce wykonania są rejestrowane na zrzutach ekranu i filmach, a wyniki testów są wyświetlane jako określone KPI na

pulpitach nawigacyjnych. Można również skonfigurować alerty, które można dostosować, aby zapewnić pełną kontrolę nad procesem testowania.

Rejestrator Katalon

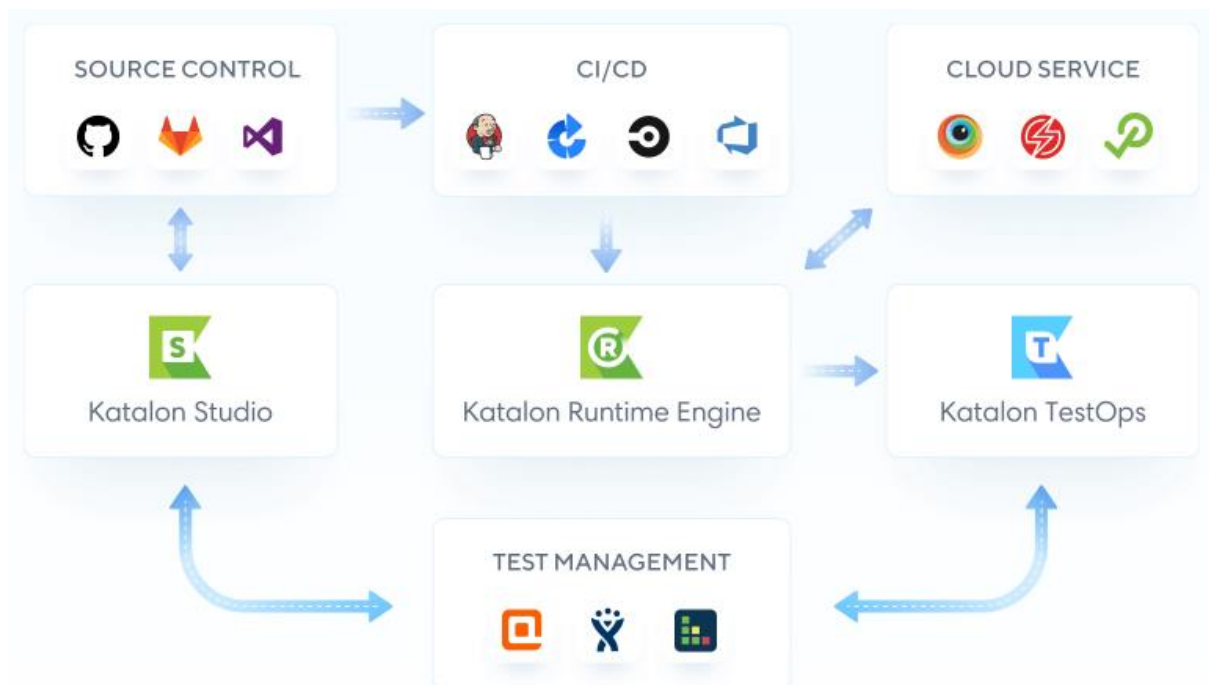
Katalon jest dostarczany z modułem nagrywania opartym na Selenium dla przeglądarek Chrome, Edge i Firefox. Rejestruje testy, pozwala testerom przeprowadzać debugowanie, automatyzować zarządzanie testami i eksportować skrypty testowe do C#, Java, Ruby, Python, Groovy i Robot Framework. Katalon Recorder posiada funkcję nagrywania i odtwarzania dla testerów bez umiejętności programowania i zapisuje metody stosowane wcześniej w testach.

Zalety korzystania z Katalon Studio

Mimo że Katalon Studio jest stosunkowo nowym narzędziem, zyskało już znaczną popularność (Katalon twierdzi, że jest używane przez ponad 65 000 firm w ponad 160 krajach) i zostało nawet nazwane 2020 Gartner Peer Insights Customers' Choice — i nie bez powodu. Zobaczmy, jakie są jego główne zalety.

+ Zintegrowany z przepływem pracy CI/DevOps i innymi narzędziami

Katalon nie wymaga dodatkowych rozszerzeń do przeprowadzania testów, ponieważ oferuje szeroki zakres przydatnych integracji. Na przykład połączenie z qTest, scentralizowanym narzędziem do zarządzania testami, które ułatwia komunikację zespołu QA, pozwala na kontrolowanie wymagań testowych, planowanie przypadków testowych i zarządzanie defektami. Użytkownicy mogą również łatwo zintegrować się z innymi platformami zarządzania SDLC (cyklem rozwoju oprogramowania), takimi jak JIRA, TestRail i TestLink.



Połączenie produktów Katalon ze zintegrowanymi narzędziami

Źródło: katalon.com

Katalon jest zintegrowany z wieloma narzędziami używanymi do obsługi CI/CD i DevOps, takimi jak Jenkins, Bamboo, TeamCity, CircleCI i Travis CI. Wprowadzone w maju 2021 r. Katalon Studio 8 obejmuje natywną integrację z usługami Azure DevOps Services, a mianowicie Azure Test Plans. Teraz użytkownicy mogą mapować przypadki testowe w Azure DevOps na zautomatyzowane przypadki testowe w Katalon Studio, a także automatycznie wysyłać dzienniki i raporty wykonywania testów z Katalon Studio do uruchomienia testowego w Azure DevOps.

Katalon Studio obsługuje również integrację z platformami współpracy (Git, Microsoft Teams i Slack) oraz platformami wykonawczymi (SauceLabs, BrowserStack, Selenium Grid i Kobiton).

Integracje Katalon Studio





+ Intuicyjny pulpit analityczny i raporty













Wbudowany moduł testowania opartego na danych pozwala użytkownikom analizować testy na podstawie zarejestrowanych scenariuszy. Katalon wyświetla wyniki analityczne w postaci wbudowanych raportów, które można wyeksportować w formacie PDF, HTML, Excel lub CSV. Raporty są wizualnie intuicyjne i proste w obsłudze.

Test Case's Log

Test Log | Information | Integration

☒ Info ☒ Passed ☒ Failed ☒ Error ☐ Incomplete ☐ Warning ☐ Not Run

Search here...    

| Item | Description | Elapsed | | |
|--|----------------------------------|---------|---|---|
| >  1. openBrowser | | 14.085s |  |  |
| >  2. click | Click on 'Book Appointment' butt | 0.395s | |  |
| ▼  3. setText | Input username | 3.471s | |  |
|  Finding Test Object with id 'Object Repo | | | | |
|  Finding Test Object with id 'Object Repo | | | | |
|  Checking object | | | | |
|  Unable to set text (Root cause: java.lang. | | |  | |

Information | Image

Name setText

Start 2017-02-20 11:20:07 **End** 2017-02-20 11:20:11 **Elapsed** 3.471s

Description Input username

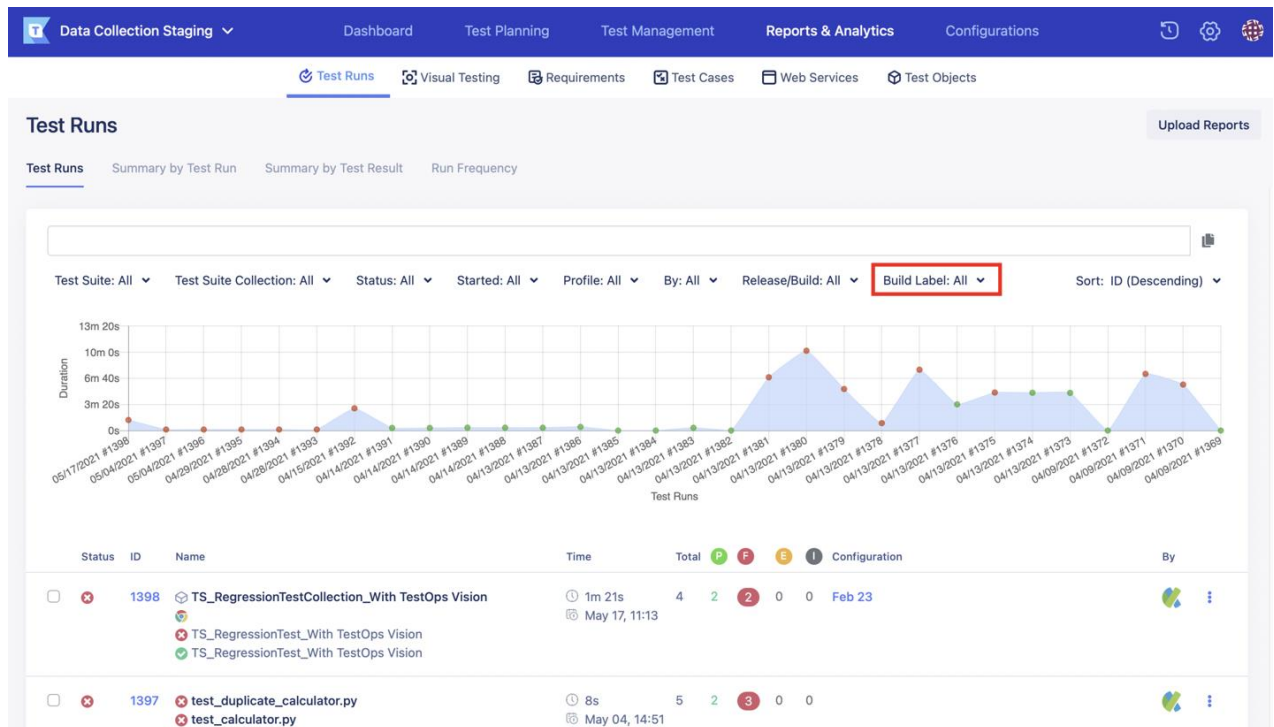
Message Unable to set text (Root cause: java.lang.IllegalArgumentException: Object is null)

Karta dziennika testów

Źródło: [Katalon Studio Docs](#)

Katalon Studio Enterprise oferuje więcej opcji raportowania, tj. raporty z zestawu testów i historię raportów, filmy dla przeglądarek bezgłowych itp.

Jeszcze więcej możliwości raportowania i analizy jest dostępnych dzięki integracji narzędzia TestOps.



Testy w Katalon TestOps

Źródło: [Katalon Studio Docs](#)

+ Obsługuje kilka rodzajów testów

W Katalon Studio można uruchomić następujące typy testów:

Oparte na słowach kluczowych. To podejście działa w przypadku testerów, którzy nie są biegli w pisaniu skryptów. Tester może przeprowadzić automatyczny test na podstawie słów kluczowych reprezentujących działania użytkowników w AUT (Aplikacje w trakcie testowania). Słowa kluczowe można dostosować.

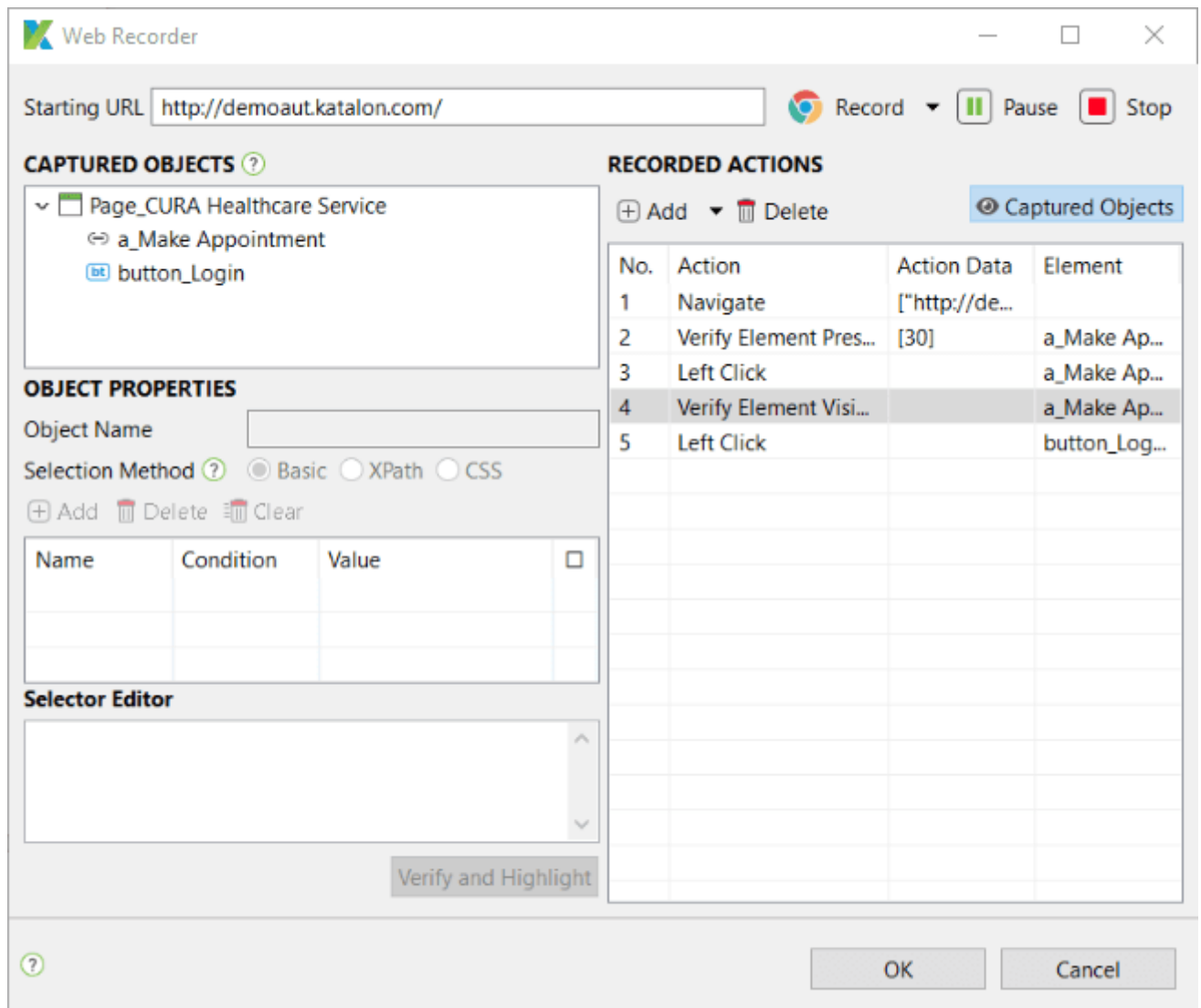
Oparte na danych. Strategia wykonywania testów, która umożliwia odczytywanie skryptów testowych z różnych źródeł danych. Dane są pobierane z pul danych, plików Excel, obiektów ADO, plików CSV i źródeł ODBC.

Testowanie API. Obsługuje źródła danych XLS, XML i bazy danych z dynamicznym mapowaniem dla lepszego pokrycia testów. Ponadto testy są zgodne z Cucumber, frameworkiem opartym na BDD (programowanie sterowane zachowaniem) do testów akceptacyjnych aplikacji internetowych.

Różne przeglądarki. Katalon obsługuje wszystkie popularne przeglądarki i umożliwia przeprowadzanie automatycznych testów w środowiskach Chrome, Firefox, Internet Explorer, Safari, Edge, Remote, Headless i niestandardowych.

+ Przyjazny dla użytkownika graficzny interfejs użytkownika

Interfejs użytkownika Katalon oferuje bogatą grafikę z widokami drzewa, tabelami i menu, które umożliwiają użytkownikowi łatwe zarządzanie artefaktami testów. Testerzy uważają, że interfejs narzędzia jest wygodny, przyjazny dla użytkownika, przejrzysty i łatwy do zrozumienia nawet dla użytkowników nietechnicznych.



Interfejs Katalon Studio

Źródło: [Dokumentacja Katalon Studio](#)

+ Łagodna krzywa uczenia się + materiały do nauki

Katalon Studio słynie z prostoty zarówno instalacji, jak i użytkowania. Jest łatwy w konfiguracji dzięki wielu zintegrowanym pakietom. Jest dobry zarówno dla początkujących, jak i zaawansowanych użytkowników, którzy mogą korzystać z całego IDE i pisać zaawansowane skrypty. Deweloperzy mają dostęp do wielu filmów szkoleniowych i dokumentacji dla użytkowników: Katalon ma repozytoria GitHub, listę kursów online na Udemy i kursy wideo na YouTube.

Katalon posiada w pełni funkcjonalną darmową wersję, co przez większość użytkowników jest uważane za zaletę. Jednak wersja Enterprise oferuje więcej funkcji i dostęp do wszystkich wtyczek Katalon oraz licencji offline, więc jest zalecana dla zespołów i skalowalnych projektów, podczas gdy bezpłatna wersja podstawowa jest oferowana do użytku indywidualnego.

Wady korzystania z Katalon Studio

Każdy produkt ma swoje dobre i złe strony. Przyjrzyjmy się wadom Katalon Studio.

– Brak języków skryptowych

W przeciwieństwie do Selenium i TestComplete, jedynym językiem skryptowym obsługiwany przez Katalon jest Groovy. Ten język skryptowy należy do rodziny Java, więc każdy, kto zna Javę, może z niego korzystać. Ale użytkownicy chcieliby widzieć więcej obsługiwanych języków.

– Mała społeczność

Ponieważ Katalon został opracowany w 2015 roku, społeczność jest mniejsza niż bardziej dojrzałych konkurentów i możesz doświadczyć braku wsparcia ze strony testerów rówieśniczych. Tak więc, dla porównania, na dzień tej aktualizacji jest 706 pytań oznaczonych Katalon Studio na Stack Overflow, podczas gdy dla Selenium jest 87 555 pytań. Jednak według Capterra, Gartner i Stack Overflow Katalon jest znacznie bardziej popularny niż Ranorex i inne narzędzia do automatyzacji testów.

Istnieje dedykowane forum Katalon i rosnąca społeczność na GitHub, więc wraz z dojrzewaniem produktu i zdobywaniem kolejnych wielbicieli, łatwiej jest znaleźć odpowiedzi na pojawiające się pytania.

– Nie jest open source

Narzędzie posiada zamknięty kod źródłowy, co skutkuje mniejszą liczbą programistów w społeczności. Selenium, jeden z głównych konkurentów Katalon, to narzędzie typu open source, które umożliwia inżynierom dostosowywanie go lub korzystanie z pakietów stworzonych przez społeczność.

Jednak Katalon Studio ma kilka otwartych komponentów, platformę Katalium o otwartym kodzie źródłowym oraz sklep Katalon o otwartym kodzie źródłowym z wtyczkami, które opisaliśmy powyżej.

– Problemy z wydajnością

Na przykład czasami narzędzie zawiesza się lub może zacząć się opóźniać, problematyczna jest weryfikacja tekstu i obiektów w ramach iframe. Testowanie mobilne zajmuje więcej czasu ze względu na konieczność przechwytywania i pisania kodu.

Puppeteer

Puppeteer to biblioteka Node.js, która zapewnia wysokopoziomowy interfejs API do kontrolowania Chrome lub Chromium przez protokół DevTools. Można go skonfigurować tak, aby działał bez głowy. Puppeteer jest najlepszym narzędziem do automatyzacji, jeśli szybkość i wydajność testów automatycznych są ważniejsze niż przenośność i kompatybilność. Ma przewagę nad Selenium dzięki lepszej kontroli nad Chrome.

Zasadniczo Puppeteer jest narzędziem automatyzacji, a nie narzędziem testowym. Oznacza to, że jest niezwykle popularny w przypadkach użycia, takich jak scraping, generowanie plików PDF itp.

Puppeteer używa tego samego protokołu debuggera, którego Selenium (no, ChromeDriver) używa do wykonywania kliknięć, a w praktyce Puppeteer (Playwright, o którym będziemy rozmawiać później) i

Selenium używają tego samego kodu do wykonywania kliknięć. W praktyce architektura Puppeteer wygląda mniej więcej tak:

Niektóre z najistotniejszych cech Puppeteer to:

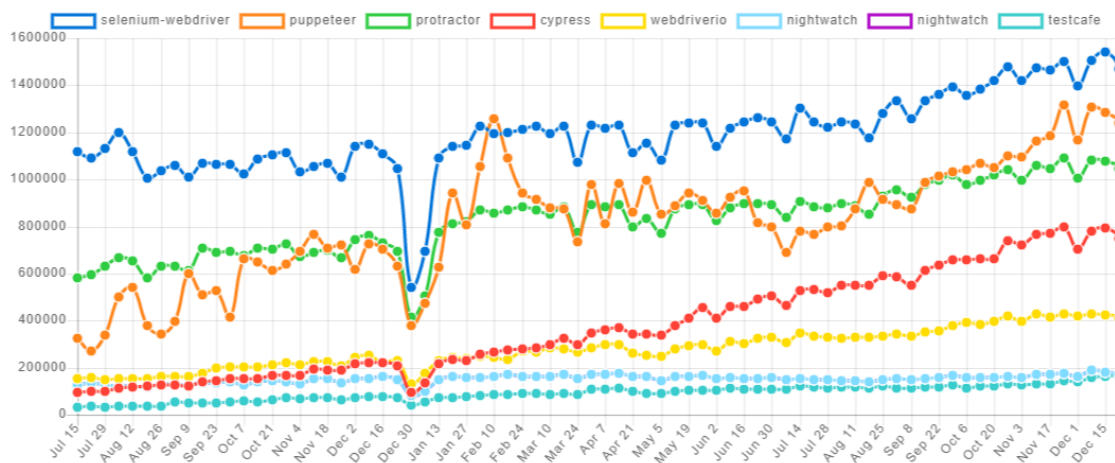
Puppeteer zapewnia większą kontrolę nad Chrome

Domyślny tryb Puppeteera jest niezwykle szybki



Wykonuje zrzuty ekranu i pliki PDF stron do testowania interfejsu użytkownika

Mierzy czas renderowania i ładowania za pomocą narzędzia analizy wydajności Chrome

Katalon Studio kontra Puppeteer w przemyśle



Jak widać na powyższym obrazku, Puppeteer wyprzedził od sierpnia i cały czas dobrze rywalizuje z Katalon. Te dane pokazują trendy pobierania za pomocą npm (Node Package Manager) w latach 2018-2019 i sugerują, że wyścig o automatyzację sieci toczy się między Katalon a Puppeteer.

| | | |
|------------------------------|--|--|
| Cechy | Puppeteer  | Katalon  |
| Obsługa języka programowania | Tylko JavaScript | Java, Python, Node.js i C# |
| Obsługa przeglądarki | Tylko Chrome | Chrome, Mozilla, Safari, IE, Opera |
| Spółeczność | Grupy dyskusyjne Google, GitHub i przepełnienie stosu | Szerokie wsparcie społeczności na wielu forach |
| Szybkość wykonania | Szybciej, ale tylko z Chrome | Stosunkowo wolniej |
| Instalacja i konfiguracja | Łatwa konfiguracja za pomocą jednego polecenia | Stosunkowo skomplikowane dla nowego użytkownika |
| Obsługa wielu platform | Nie | Tak |
| Nagranie | Nie | Tak |
| Zrzuty ekranu | Obsługa zarówno obrazów, jak i plików PDF | Tylko obsługa obrazów |
| Obsługa platformy testowej | Sieć | Web i Mobile z Appium |

Z powyższego porównania jasno wynika, że Puppeteer będzie najlepszym wyborem, gdy musimy wykonać testy na poziomie jednostki dla dowolnej aplikacji internetowej i wymagane jest szybkie i elastyczne rozwiązanie. Katalon będzie lepszym wyborem w przypadkach, w których mamy aplikację internetową i mobilną, a obsługa wielu platform jest naszym wymogiem numer jeden.



Puppeteer zapewnia również bezpośredni dostęp do CDP, jeśli tego potrzebujesz. Co może być czasami bardzo przydatne i ogólnie wydaje się, że jest mniej ruchomych części.

Plusy :

Prosty w konfiguracji

Dobra dokumentacja

Automatycznie instaluje Chrome w działającej wersji

Cienkie opakowanie

Dwukierunkowy (zdarzenia) – automatyzacja takich rzeczy jak logi konsoli jest łatwa

Utrzymywane przez Google.

Najpierw JavaScript, więc kod wydaje się bardzo naturalny

Minusy :

Ograniczona obsługa wielu przeglądarek — tylko Chrome i Firefox

Jest jak Framework dla automatyzacji, a nie framework testowy — często trzeba ponownie wdrażać narzędzia związane z testowaniem

Automatyczna konfiguracja przeglądarki pobiera Chromium, a nie Chrome i istnieją między nimi pewne różnice.

Wykonane testy

Na początku przedstawione zostaną wyniki dla Cypress.

Testy zostały wykonywane 1/10/100 razy.

Wszystkie testy wykonane raz zajmują od 59 do 65 sekund. Prawie połowę tego czasu obejmują testy responsywności. Najszybsze natomiast są testy związane z ciasteczkami strony.

Tests 21 1 65.66

cypress/integration/2-advanced-examples/moje-testy/hello-world.test.js

cy.location() - get window.location

TEST BODY

- 1 location
- 2 - assert expected '' to be empty
- 3 - assert expected https://moodle.tu.kielce.pl/ to equal https://moodle.tu.kielce.pl/
- 4 - assert expected moodle.tu.kielce.pl to equal moodle.tu.kielce.pl
- 5 - assert expected moodle.tu.kielce.pl to equal moodle.tu.kielce.pl
- 6 - assert expected https://moodle.tu.kielce.pl to equal moodle.tu.kielce.pl

AssertionError

Timed out retrying after 4000ms: expected 'https://moodle.tu.kielce.pl' to equal 'moodle.tu.kielce.pl'

cypress/integration/2-advanced-examples/moje-testy/hello-world.test.js:205:3

```

203 |     expect(location.host).to.eq('moodle.tu.kielce.pl')
204 |     expect(location.hostname).to.eq('moodle.tu.kielce.pl')
> 205 |     expect(location.origin).to.eq('moodle.tu.kielce.pl')
      |                                     ^
206 |     expect(location.pathname).to.eq('/mod/forum/')
207 |     expect(location.port).to.eq('')
208 |     expect(location.protocol).to.eq('https:')

```

View stack trace Print to console

- ✓ cy.url() - get the current URL
- ▼ Cypress.version
 - ✓ Get current version of Cypress being run
- ▼ Cypress.spec
 - ✓ Get current spec information
- ▼ Viewports/responsivness test
 - ✓ Test responsywnosci

Tests 21 1 65.66

cypress/integration/2-advanced-examples/moje-testy/hello-world.test.js

- ▼ Page
 - ✓ Website visit
- ▼ Login
 - ✓ Login page through
 - ✓ Sign in
- ▼ Windows
 - ✓ Windows properties
 - ✓ global windows views
 - ✓ document object
 - ✓ check the header
- ▼ Cookies set/check/clear
 - ✓ cy.getCookie() - get a browser cookie
 - ✓ cy.getCookies() - get browser cookies
 - ✓ .preserveOnce() - preserve cookies by key
 - ✓ .defaults() - set defaults for all cookies
 - ✓ cy.setCookie() - set a browser cookie
 - ✓ cy.clearCookie() - clear a browser cookie
 - ✓ cy.clearCookies() - clear browser cookies
- ▼ Cypress.config()
 - ✓ Get and set configuration options
- ▼ Cypress.arch
 - ✓ Get CPU architecture name of underlying OS
- ▼ Location
 - ✓ cy.hash() - get the current URL hash
 - ✗ cy.location() - get window.location

Tests 84 1 64.26

cypress/integration/2-advanced-examples/moje-testy/hello-world.test.js

cy.location() - get window.location

TEST BODY

- 1 location
- 2 - assert expected '' to be empty
- 3 - assert expected https://moodle.tu.kielce.pl/ to equal https://moodle.tu.kielce.pl/
- 4 - assert expected moodle.tu.kielce.pl to equal moodle.tu.kielce.pl
- 5 - assert expected moodle.tu.kielce.pl to equal moodle.tu.kielce.pl
- 6 - assert expected https://moodle.tu.kielce.pl to equal moodle.tu.kielce.pl

AssertionError

Timed out retrying after 4000ms: expected 'https://moodle.tu.kielce.pl' to equal 'moodle.tu.kielce.pl'

cypress/integration/2-advanced-examples/moje-testy/hello-world.test.js:206:3

```
4
204 |     expect(location.host).to.eq('moodle.tu.kielce.pl')
205 |     expect(location.hostname).to.eq('moodle.tu.kielce.pl')
> 206 |     expect(location.origin).to.eq('moodle.tu.kielce.pl')
      |                                     ^
207 |     expect(location.pathname).to.eq('/mod/forum/')
208 |     expect(location.port).to.eq('')
209 |     expect(location.protocol).to.eq('https:')
```

View stack trace Print to console

- ✓ cy.url() - get the current URL
- ▼ Cypress.version
 - ✓ Get current version of Cypress being run
- ▼ Cypress.spec
 - ✓ Get current spec information
- ▼ Viewports/responsivness test
 - ✓ Test responsywnosci

Wykonanie wszystkich testów 10 krotnie przyniosło rezultat zawierający się w przeciętnym wyniku dla pojedynczego ich wykonania. W tym przypadku można nawet zauważyć krótszy czas o 1 sekundę.

Tests

21

10

--

93.65

cypress/integration/2-advanced-examples/moje-testy/hello-world.test.js

Open S

cy.location() - get window.location

TEST BODY

1 location

2 - assert expected '' to be empty

3 - assert expected https://moodle.tu.kielce.pl/ to equal https://moodle.tu.kielce.pl/

4 - assert expected moodle.tu.kielce.pl to equal moodle.tu.kielce.pl

5 - assert expected moodle.tu.kielce.pl to equal moodle.tu.kielce.pl

6 - assert expected https://moodle.tu.kielce.pl to equal moodle.tu.kielce.pl

AssertionError

Timed out retrying after 4000ms: expected 'https://moodle.tu.kielce.pl' to equal 'moodle.tu.kielce.pl'

cypress/integration/2-advanced-examples/moje-testy/hello-world.test.js:203:3

4

201 | expect(location.host).to.eq('moodle.tu.kielce.pl')

202 | expect(location.hostname).to.eq('moodle.tu.kielce.pl')

> 203 | expect(location.origin).to.eq('moodle.tu.kielce.pl')

| ^

204 | expect(location.pathname).to.eq('/mod/forum/')

205 | expect(location.port).to.eq('')

206 | expect(location.protocol).to.eq('https:')

View stack trace

Print to console

cy.url() - get the current URL

Cypress.version

Get current version of Cypress being run

Cypress.spec

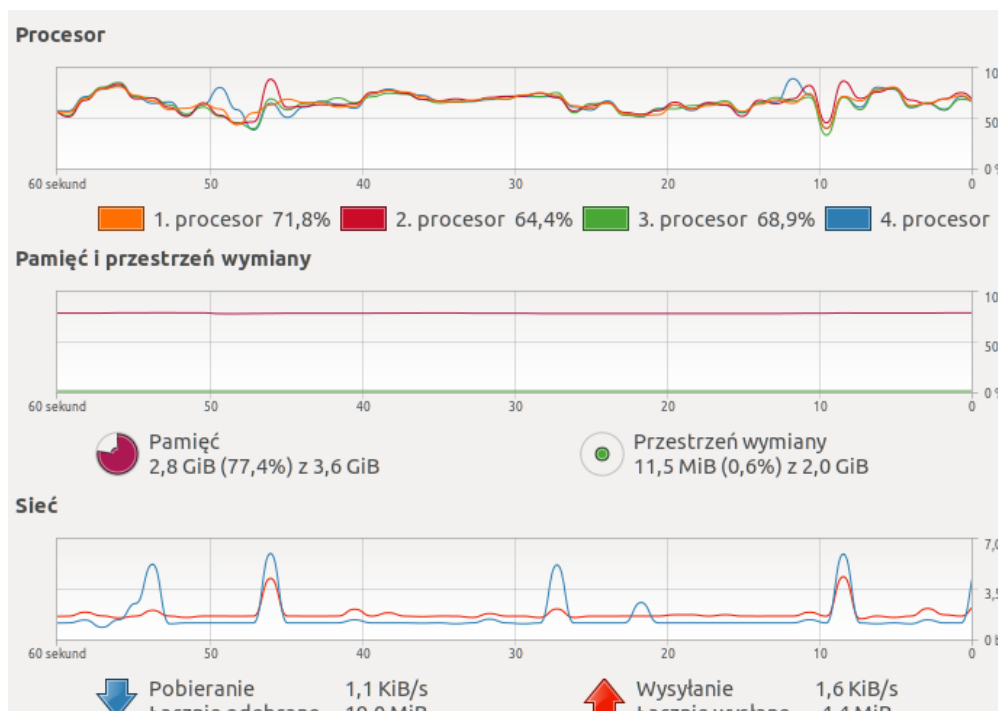
Get current spec information

Viewports/responsivness test

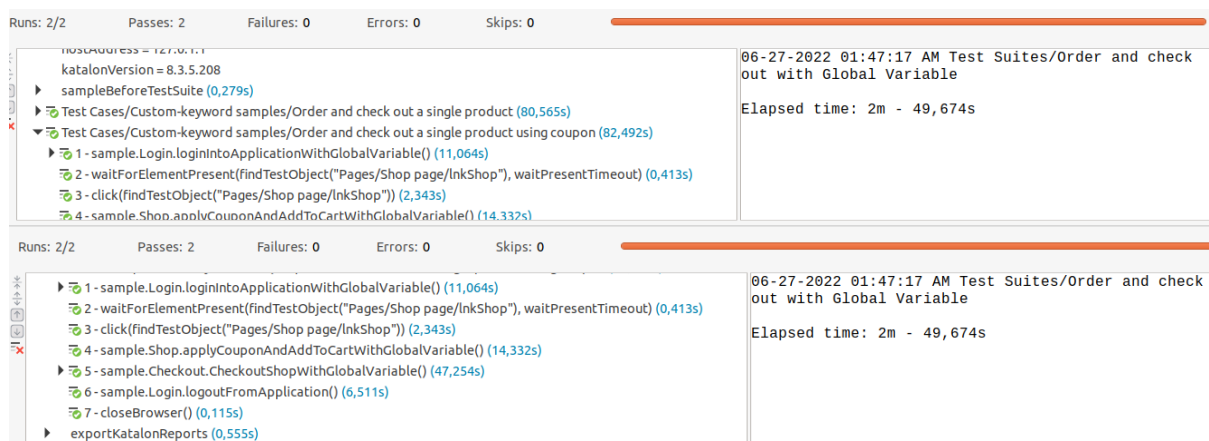
Test responsywnosci

Wykonanie testów 100 – nie licząc testów responsywności – one dalej uruchamiane są 1 krotnie zwiększa czas ich wykonanie o kolejne 30 sekund. Cały proces jest wykonywany od 90 do 97 sekund. Wydłużeniu uległ proces logowania, obciąża również w znacznym stopniu zasoby komputera w związku z wymaganą ilością odpalonych kart przeglądarki.

Wykonanie wszystkich testów łącznie z responsywnością wiąże się z bardzo długim czasem wykonywania, aż 280 do 300 sekund. W tym wypadku sporą rolę odgrywa obciążenie procesora i pamięci podczas uruchamiania kart przeglądarki Chrome.



Katalon Studio



Problems Event Log Console Log Viewer Self-healing insights

Runs: 1/1 Passes: 1 Failures: 0 Errors: 0 Skips: 0

6.1 - sample.Shop.addToCart(productName.toString(), urlProduct) (7,257s)
 6.1 - sample.Shop.addToCart(productName.toString(), urlProduct) (7,072s)
 7 - sample.Checkout.CheckoutShopWithGlobalVariable() (39,307s)
 The list of options is: [SELECT A COUNTRY..., ÅLAND ISLANDS, AFGHANISTAN, ALBANIA, ALGERIA, AM
 The selected options list is: [Vietnam]
 Unable to find the element located by 'By.xpath: //*[@class = 'blockUI blockOverlay']'. Please recheck
 Unable to find the element located by 'By.xpath: //*[@class = 'blockUI blockOverlay']'. Please recheck
 8 - closeBrowser() (0,213s)

06-27-2022 02:04:53 AM
 sample.Shop.addToCart(productName.toString(), urlProduct)
 Elapsed time: 7,257s
 sample.Shop.addToCart is PASSED

Runs: 1/1 Passes: 1 Failures: 0 Errors: 0 Skips: 0

Test Cases/Data-driven samples/Order and check out multiple products (90,487s)
 (Default) productName = Flying Ninja
 1 - sample.Login.loginIntoApplicationWithGlobalVariable() (14,625s)
 2 - waitForElementPresent(findTestObject("Pages/Shop page/lnkShop"), waitPresentTimeout) (0,400s)
 3 - click(findTestObject("Pages/Shop page/lnkShop")) (2,385s)
 4 - product = findTestData(dataFile) (1,508s)
 5 - productList = }.collect(Collectors.toList()) (0,131s)
 6 - for (def productName : productList) (30,454s)

06-27-2022 02:04:17 AM Test Cases/Data-driven samples/
 Order and check out multiple products
 Elapsed time: 1m - 30,487s
 Test Cases/Data-driven samples/Order and check out
 multiple products

Test Cases Table

☒ Passed
 ☒ Failed
 ☒ Error
 ☒ Incomplete
 ☒ Skipped

Export report [Katalon TestOps](#) [Show Test Case Details](#)

Search here...

| No. | Name | Resolution | Video |
|-----|--|------------|-------|
| 1 | Order and check out a single product (1m - 20,565s) | | |
| 2 | Order and check out a single product using coupon (1m - 22,492s) | | |

| Summary | | Execution Settings | | Execution Environment | |
|-----------------|--|--------------------|-----------------------|-----------------------|--|
| Test Suite ID | Test Suites/Order and check out with Global Variable | | | | |
| Host name | linux - Lenovo-ideapad-320-15ISK | Local OS | Linux 64bit | | |
| Katalon version | 8.3.5.208 | Platform | Chrome 102.0.5005.115 | | |
| Start | 2022-06-27 01:47:22 | End | 2022-06-27 01:50:07 | | |
| Elapsed | 2m - 44,707s | | | | |

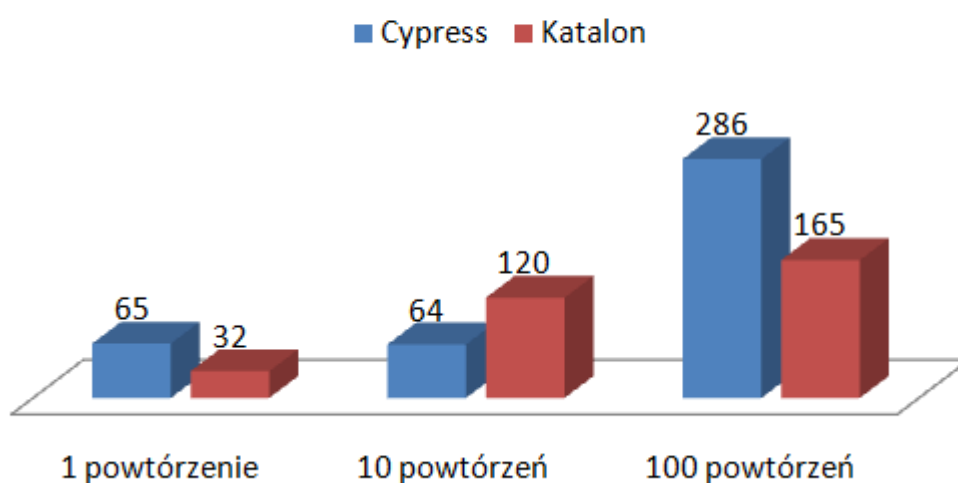
Narzędzie Katalon Studio wraz z wbudowanymi raportami w bardzo przejrzysty sposób informuje jaka część testów objęła największy wymiar czasowy. Test nie przekraczał czasu wykonania 2 minut i 50 sekund.

Katalon najszybciej współpracuje z dedykowanym Chromium driverem, występują delikatne zmiany względem Cypressa oraz normaliej przeglądarki Chrome. Zapotrzebowanie na zasoby komputera znacznie się zmniejsza a sam proces testowania jest znacznie szybszy. Niestety Katalon Studio, może tylko w moim przypadku często się zawieszał po wykonaniu danego scenariusza testowego. Nie miało to jednak większego wpływu na sam czas wykonania testów, a na przerwę między wykonaniem następnego.

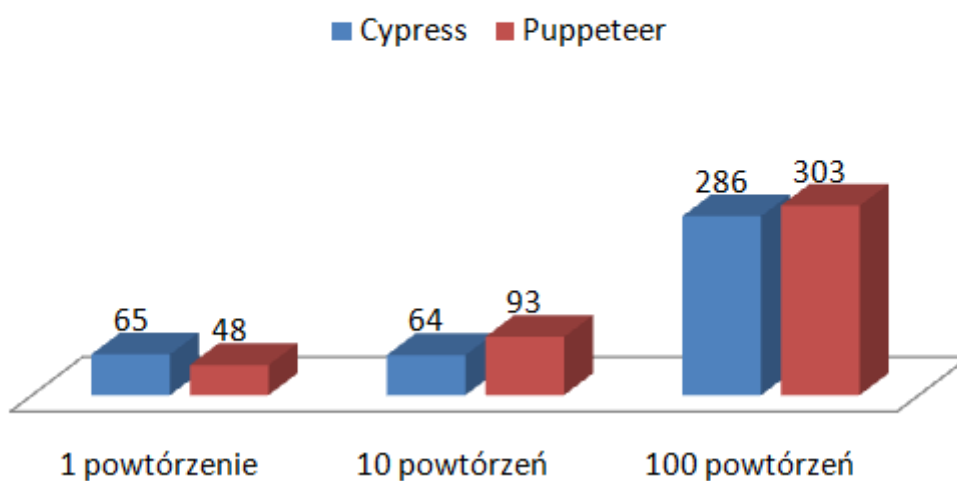
Katalon nie był testowany pod innymi przeglądarkami ze względu na problematyczną instalację odpowiedniej wersji sterownika.

Katalon pozwalał na ręczne pisanie kodu testu oraz nagrywanie i ponowne wykonanie zarejestrowanych czynności. We wstępnej fazie testów badane były RESTY na stronach demo, jednak ostatecznie nie zawarto ich w sprawozdaniu.

Czasy wykonania testów Cypress vs Katalon Studio



Czasy wykonania testów Cypress vs Puppeteer



Puppeteer

Puppeteer okazał się najbardziej ubogim frameworkiem jeśli chodzi o możliwość śledzenia czasów oraz zasobów komputera. Całość testów wykonywana została wewnątrz środowiska Visual Studio Code bez dedykowanego softu jak to było w przypadku Cypress bądź Katalon.

Wszystkie testy oraz ich wyniki przeprowadzane były za pośrednictwem terminala Visual Studio Code. Jako ciekawostka w porównaniu do innych testowanych frameworków, wyniki niektórych testów zostały zapisywane w postaci zrzutów ekranu po ich wykonaniu. Przykładem może być wykonanie testów API Map Google bądź ustawianie ciasteczek na platformie PayPal.

Inne testy po zakończeniu zwracały stosowny komunikat w terminalu.

Średni czas wykonania każdego testu mierzony przy pomocy stopera w telefonie wynosił 2-5 sekund.

Wykonane testy obejmowały:

Utworzenie okna alertu i zamknięcie gp ponownie.

Pobieranie/Przesył pliku

Znajdowanie obrazu według selektora klasy, pobieranie obraz, zapisanie go na dysku i ponownie odczytanie. Wykorzystane razem z metodą `.fileUpload()`.

Emulowanie urządzenia

Użycie wbudowanych deskryptorów urządzeń do emulacji iPhone'a 6. W rzeczywistości są to skróty do wywołania `page.setUserAgent()` i `page.setViewport()`.

Uzyskiwanie wartości wspólnych elementów formularza

Pobiera wartość często używanych elementów formularza HTML za pomocą `page.$eval()`

Pobierz listę linków

Pobiera linki na stronie głównej i zwraca je.

Pobierz wartość tekstową elementu

Pobiera wartość tekstową elementu przy użyciu metody `page.$eval`

Pobranie tytuł

Pobierz tytuł strony i wydrukuj go w konsoli.

Hover

Funkcja `hover` jest kombinacją przewijania i umieszczania myszy w stanie najechania na żądany element. Ten przykład unosi pierwszy utwór, który znajdujemy na stronie głównej soundcloud.com, który powinien uruchomić odtwarzanie i jak przyciski, aby były widoczne.

Czytanie klawiatury

Wpisuje do edytora tekstu.

Location_Faker

Sfałszowanie lokalizacji dla API geolokalizacji używanego przez przeglądarkę.

Aktywność myszy

Ładowanie strony, która odtwarza akcje myszy używane na niej.

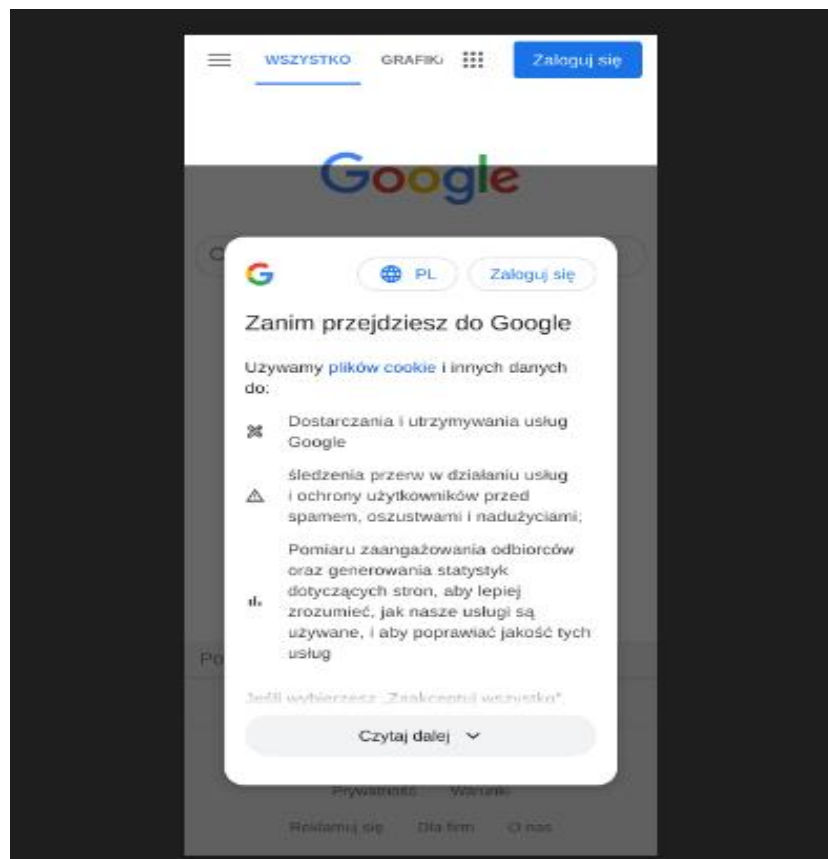
Render PDF

Renderuje plik PDF specyfikacji Puppeteer API. Jest to dość długa strona i wygeneruje ładny, wielostronicowy plik PDF w formacie A4.

```
PUPPETEER-EXAMPLES-MASTER
JS emulate_devices.js
JS forms.js
JS full.png
JS get_list_of_links.js
JS get_text_value.js
JS get_title.js
JS google_social.js
JS hover.js
JS hover.png
JS html.pdf
JS keyboard.js
JS keyboard.png
JS location_faker.js
JS location.png
JS mouse_click.png
JS mouse.js
JS nytimes.png
JS package-lock.json
JS package.json
JS paypal_login.png
JS pdf.js
JS README.md
JS request_interception.js
JS screenshots_parallel.js
JS screenshots_parallel_cologne_colleges.js
JS screenshots.js
JS set_cookie.js
JS trace.json
> OUTLINE
> TIMELINE

11 const browser = await puppeteer.launch({
12 const page = await browser.newPage()
13 await page.setViewport({ width: 1200, height: 800 })
14
15 await page.goto('https://moodle.tu.kielce.pl/')
16 const imageHref = await page.evaluate(sel => {
17   | return document.querySelector(sel).getAttribute('src').replace('/', '')
18   | }, '.hero-image')
19
20 const viewSource = await page.goto('https://moodle.tu.kielce.pl/' + imageHref)
21 const buffer = await viewSource.buffer()
22 await writeFileAsync(path.join(__dirname, 'psk.png'), buffer)
23 console.log('The file was saved!')
24
25 await readFileAsync(path.join(__dirname, 'psk.png'))
26 console.log('The file was read!')

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Linux@Lenovo-ideapad-320-15ISK:~/Pulpit/puppeteer-examples-master$ node alerts.js
This message is inside an alert box
Linux@Lenovo-ideapad-320-15ISK:~/Pulpit/puppeteer-examples-master$ node emulate_devices.js
Google
Linux@Lenovo-ideapad-320-15ISK:~/Pulpit/puppeteer-examples-master$ node forms.js
Checkbox checked status: false
Radio values: [ 'option1', 'option2', 'option3' ]
[ 'Open this select menu', '1', '2', '3' ]
Linux@Lenovo-ideapad-320-15ISK:~/Pulpit/puppeteer-examples-master$ node get_list_of_links.js
[]
Linux@Lenovo-ideapad-320-15ISK:~/Pulpit/puppeteer-examples-master$ node get_text_value.js
Hacker News
Linux@Lenovo-ideapad-320-15ISK:~/Pulpit/puppeteer-examples-master$ node get_title.js
Google
Linux@Lenovo-ideapad-320-15ISK:~/Pulpit/puppeteer-examples-master$ node hover.js
Linux@Lenovo-ideapad-320-15ISK:~/Pulpit/puppeteer-examples-master$ node keyboard.js
Linux@Lenovo-ideapad-320-15ISK:~/Pulpit/puppeteer-examples-master$ node location_faker.js
Linux@Lenovo-ideapad-320-15ISK:~/Pulpit/puppeteer-examples-master$ node mouse.js
Linux@Lenovo-ideapad-320-15ISK:~/Pulpit/puppeteer-examples-master$ node pdf.js
Linux@Lenovo-ideapad-320-15ISK:~/Pulpit/puppeteer-examples-master$ node request_interception.js
Linux@Lenovo-ideapad-320-15ISK:~/Pulpit/puppeteer-examples-master$ node set_cookie.js
Linux@Lenovo-ideapad-320-15ISK:~/Pulpit/puppeteer-examples-master$
```



Geolocation

Send feedback

Na tej stronie

Try Sample

Clone Sample

This example creates a map that displays the geographic location of a user or device on a Google map, through use of their browser's HTML5 Geolocation feature. The user must consent to location sharing or else an error is shown. [Read more about the W3C Geolocation standard.](#)

Na tej stronie używamy plików cookie Google, by móc świadczyć Ci usługi i analizować ruch.

Więcej szczegółów

OK

Mouse Event Test Page - Basic Version

[click here to test](#) [or here](#)
[click here to clear](#)

```
mousedown which=1 button=0 buttons=1
mouseup    which=1 button=0 buttons=0
click      which=1 button=0 buttons=0
```

This page contains a script for testing the events fired in different browsers when a mouse click occurs. Click on the "click here to test" link to see the events fired by the click.

This script will capture and log the following event types:

- mousedown - the mouse button goes down.
- mouseup - the mouse button goes up.
- click - a logical mouse click.
- dblclick - a logical mouse double click has occurred.
- contextmenu - a context menu has been triggered.

We do not consider mouseover, mouseoff or mousemove events here.

Different browsers generate different events in different orders. For each event, the script will give the values of the event.which and event.button variables, which indicate which mouse button was clicked.

Try left clicks, right clicks, and middle clicks. Try double clicks.

Every attempt has been made to disable the default actions of mouse clicks when clicking on the test link. However, in some browsers, the defaults cannot be

PayPal

set_by_cookie@domain.com

Password

Forgot password?

Log In

or

Sign Up

English | Français | Español | 中文

is growing. These documents reveal how.

Times investigation into China's surveillance state.

The Strange Tale of Tina Peters

The county clerk was consumed by election conspiracy theories and indicted in a bizarre scheme. Will Republicans nominate her to run elections in Colorado?

MAGA Voters Send a \$50 Million G.O.P. Plan Off the Rails in Illinois

Republican leaders think a moderate nominee could beat Gov. J.B. Pritzker. But the party's base prefers a far-right state senator — and he is getting help from Mr. Pritzker.

Proud Boys Did Not Follow Orders Given at Pre-Jan. 6 Meeting

The directives from the far-right group's leaders included obeying police lines and keeping away from ordinary protesters. But none of them were actually followed.

Abortion. Don't Believe It.

JENNI AVINS

Vacationing 101 for Those Who Have Forgotten How

MAUREEN DOWD

The Radical Reign of Clarence Thomas

LIVE EVENT | MON. JUNE 27

Inside the Jan. 6 Hearings With Jamie Raskin: A Times Virtual Event

11 a.m. E.D.T.

RSVP TO ATTEND

ROLE

TERMINAL

'option2', 'option3']

'1', '2', '3']

ISK:~/Pulpit/puppeteer-examples-master\$ node get_list_of_links.js



- Main page
- Contents
- Current events
- Random article
- About Wikipedia
- Contact us
- Donate

- Contribute
- Help
- Learn to edit
- Community portal
- Recent changes
- Upload file

- Tools
- What links here
- Related changes
- Special pages
- Permanent link
- Page information
- Cite this page
- Wikidata item

- Print/export
- Download as PDF
- Printable version

- In other projects
- Wikimedia Commons
- Wikiquote

0



From Wikipedia, the free encyclopedia

"Zero" and "Naught" redirect here. For other uses, see *0 (disambiguation)* and *Zero (disambiguation)*. For the Stolen Babies album, see *Naught (album)*.

This article is about the number and digit 0. Not to be confused with the letter O or the O mark used to represent affirmation.

0 (**zero**) is a number,^[1] and the numerical digit used to represent that number in numerals. It fulfills a central role in mathematics as the additive identity of the integers, real numbers, and many other algebraic structures. As a digit, 0 is used as a placeholder in place value systems. Names for the number 0 in English include **zero**, **nought** (UK), **naught** (US; */noʊt/*), **nil**, or—in contexts where at least one adjacent digit distinguishes it from the letter "O"—**oh** or **o** (*/oʊ/*). Informal or slang terms for zero include **zilch** and **zip**.^[2] *Ought* and *ought* (*/ɔːt/*),^[3] as well as *cipher*,^[4] have also been used historically.^{[5][6]}

Contents

- 1 Etymology
 - 1.1 Modern usage
- 2 History
 - 2.1 Ancient Near East
 - 2.2 Pre-Columbian Americas
 - 2.3 Classical antiquity
 - 2.4 China
 - 2.5 India
 - 2.5.1 Epigraphy
 - 2.6 Middle Ages
 - 2.6.1 Transmission to Islamic culture
 - 2.6.2 Transmission to Europe
- 3 Mathematics
 - 3.1 Elementary algebra
 - 3.2 Other branches of mathematics
 - 3.3 Related mathematical terms
- 4 Physics
- 5 Chemistry

| | |
|--|---|
| <div><div><div><div><div>←</div><div>−1</div></div><div><div>→</div><div>1</div></div></div><div><div><div>−1 0 1 2 3 4 5 6 7 8 9 ...</div><div>List of numbers — Integers</div><div>← 0 10 20 30 40 50 60 70 80 90 ...</div></div></div></div></div> | <div><div><div><div>0, zero, "oh" (<i>/oʊ/</i>), nought, naught, nil</div></div></div></div> |
| Cardinal | 0, zero, "oh" (<i>/oʊ/</i>), nought, naught, nil |
| Ordinal | Zeroth, noughth, 0th |
| Binary | 0 ₂ |
| Ternary | 0 ₃ |
| Octal | 0 ₈ |
| Duodecimal | 0 ₁₂ |
| Hexadecimal | 0 ₁₆ |
| Arabic, Kurdish, Persian, Sindhi, Urdu | ۰ |
| Hindu Numerals | ० |
| Chinese | 零, 〇 |
| Khmer | ០ |
| Thai | ๐ |