

Les tests

Plan

1. Vocabulaire
2. Pourquoi tester ?
3. Que teste t-on ?
4. La responsabilité du test
5. Comment écrire un test ?
6. Bonnes pratiques
7. Questions

Qui écrit des tests dans la salle ?

Vocabulaire

Résultat attendu (expected)

Résultat obtenu (actual)

Cas passant (happy path)

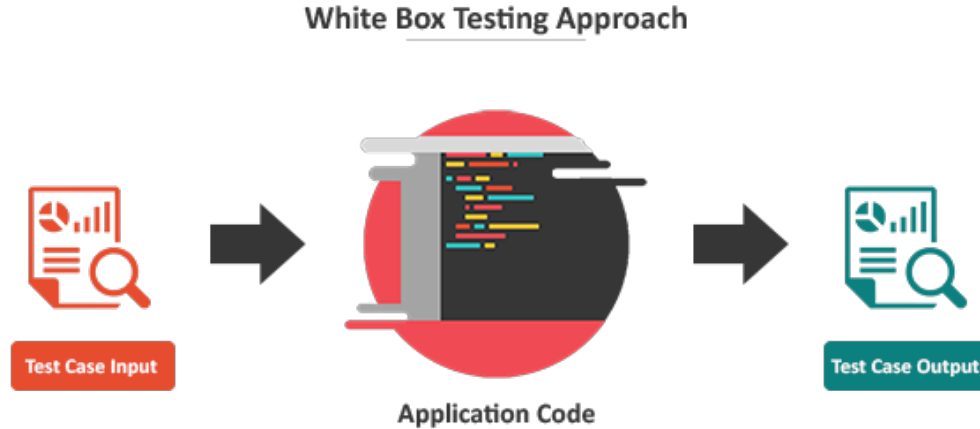
SUT (System Under Test) *le périmètre de la loupe*

Vocabulaire



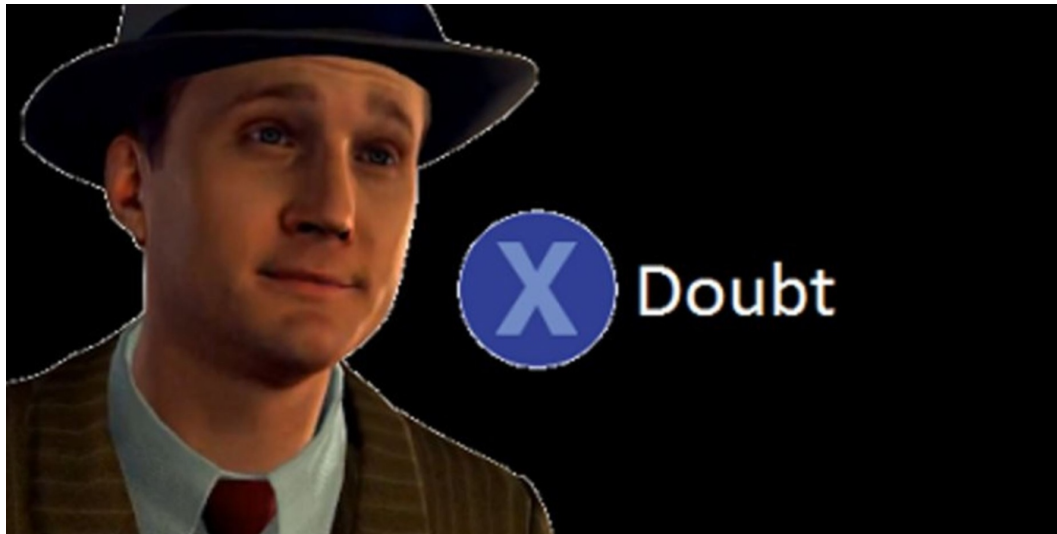
Test en boîte noire: on ne regarde que le résultat produit par notre boîte (un composant, un module, une application) pour une entrée donnée

Vocabulaire



Test en boîte blanche: on vérifie le comportement du code en fonction de son implémentation

Pourquoi tester ?



“tester c’est douter”

Pourquoi tester ?



“ça coûte”

Pourquoi tester ?



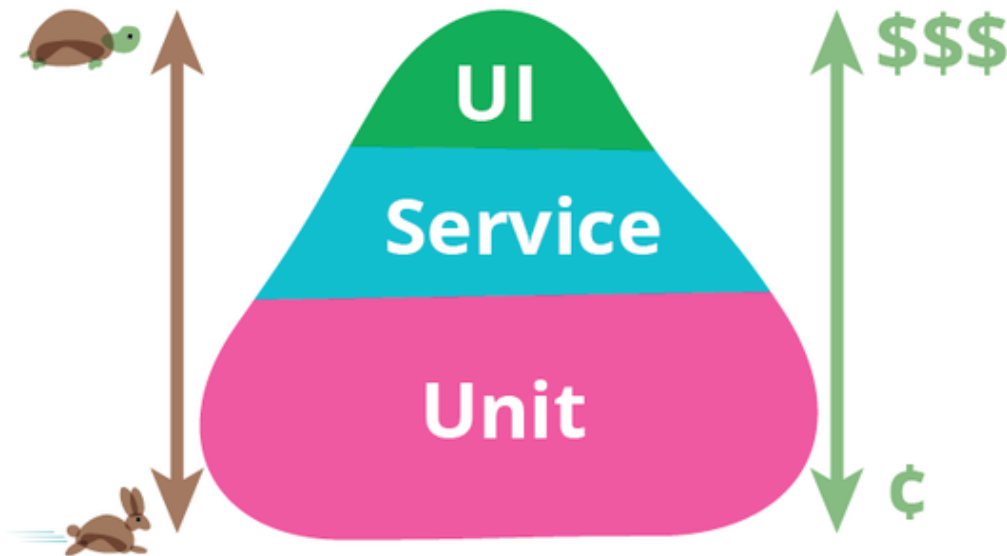
“du code supplémentaire à maintenir”

Pourquoi tester ?

- Qualité des livrables stable
 - Vérification fréquente de non régression
 - Détection plus tôt des bugs (moins de bugs en prod)
 - Meilleur contrôle des anomalies
 - Encourage quelques bonnes pratiques (SOLID, DRY)
- Amélioration de la productivité
 - Confiance dans le code
 - Refactor décomplexé
 - Réduit la phase de tests manuels
- Auto documentation technique

Qu'est-ce qu'il faut tester ?

- Pas tout !
- Ce qui est testable
- Le code métier
- Les fonctionnalités sensibles



Qui teste ?

Les développeurs (évidemment)

La qualité

Le client ou le PO ? (les specs au moins)

Quelques types de tests automatisables

- Test unitaire: test d'un composant isolé du reste du système
- Test d'intégration: test d'un composant au sein d'autres composants
- Test système: test métier, "de bout en bout"
- Tests exploratoires
 - monkey tests
 - amplification de tests



Comment écrire un test ?

Fixture: les fonctions ou méthodes qui permettent de mettre en place le contexte de test (@Setup, @Teardown, la matrice de test)

Mock: Créature factice qui se fait passer pour une fonction, méthode ou objet accessible par votre SUT. Le mock peut aussi agir comme un espion car en plus d'avoir un comportement qu'on lui aura donné, il tient compte de tous les appels qu'il a reçu ! On parle alors de Spy.

Mailsender

<https://github.com/paraita/les-tests-avec-java>

Astuces et bonnes pratiques

Un test ne vérifie qu'une chose à la fois

Un test est facile à lire

Un test utilise des *literals*

On ne teste pas ce qui ne nous appartient pas

On évite de parler au réseau et au file system

On teste tôt

Quelques liens

<https://github.com/paraita/les-tests-avec-java>

<http://www.cftl.fr/>

<https://www.istqb.org/>

<https://www.stamp-project.eu/view/main/>

Questions ?

Les tests, en vrai, ça donne quoi ?

Meetup Tests

<https://github.com/ch4mpy>

Bao-Loc

<https://github.com/ch4mpy/bao-loc>

Trouver les solutions au puzzle ci contre avec les entiers distincts de 1 à 9 pour un résultat de 66

Grille de "Bao-Loc"

		-		
+	*		-	=
13	12		11	10
*	+		+	-
/		+	*	/

Micro-service REST

- Java / Spring (boot, web, data-jpa, hateoas, test, restdocs)
- Structure en couches: entité / service / web

Front-end web

- Angular

Teste quoi ?

Acceptance (e2e)

- Application en “boîte noire”

UT (Composant de mon code)

- Code métier (services, classes métiers, ...)
- @Controllers (droits cinématique, interface publique)
- Composants de l'interface graphique

Pour quoi
faire ?

Fiabilité du code produit

- Chasser les bugs avant la MEP

Affiner la conception

- Faible couplage
- Fine granularité des fonctionnalités

Documenter

- Cas d'utilisation
- Interface publique (restdocs)

Reproduire les bugs rapportés

- Prévention des regressions récurantes

Avec Quoi ?

Frameworks !

- Jupiter / Junit / Mockito
- Karma / Jasmine / Protractor / Cypress

Code maison

- En dernier recours

Outils

- CLI
- Plugins IDE
- Plugins maven
- CI

Quand ?

Dès le début du projet

- Acceptance ?

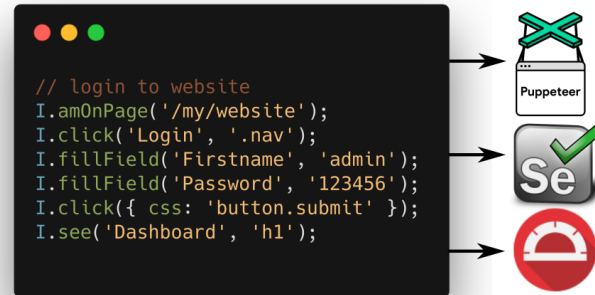
Tout le temps

- Avant chaque commit
- CI doit jouer les tests
- CI doit vérifier des critères de qualité

Tests fonctionnels de SPA

[codeceptjs](#)

jfbaillette@g-echo.fr / ltran@g-echo.fr



Tests fonctionnels

- Se comporter comme l'utilisateur,
- Dérouler des scénarios de validation,
- Tester des non régressions.

Répétable

codeceptjs

Web et mobile,

React, vue, angular

Dev in browser

CI/CD



Scenario Driven

Write acceptance tests from user's perspective. Make tests readable and easy to follow.



Driver Agnostic

Run your tests via **WebDriver**, **Puppeteer**, **TestCafe**, **Protractor**, **Appium**. The code is the same.

[Learn More](#)



Interactive Debug

Control tests as they run. Pause tests at any point and execute commands to try locators.

[Learn More](#)



Rich Locators

Use semantic locators, CSS, XPath to find elements on page

[Learn More](#)



PageObjects

PageObjects are essential to write stable and reusable code!

[Learn More](#)



Web & Mobile Testing

Test native mobile apps using **Appium** or **Detox**.

[Learn More](#)



Cucumber-like BDD

Automate business scenarios as you do in CucumberJS

[Learn More](#)



Beautiful Reports

Integrated with Allure reporter

[Learn More](#)



Data Management

Create fake data and clean it up via REST API

[Learn More](#)



Parallel Testing

Tests are split into chunks and executed in multiple processes.

[Learn More](#)



Reduced Flakiness

Automatically retry failed steps

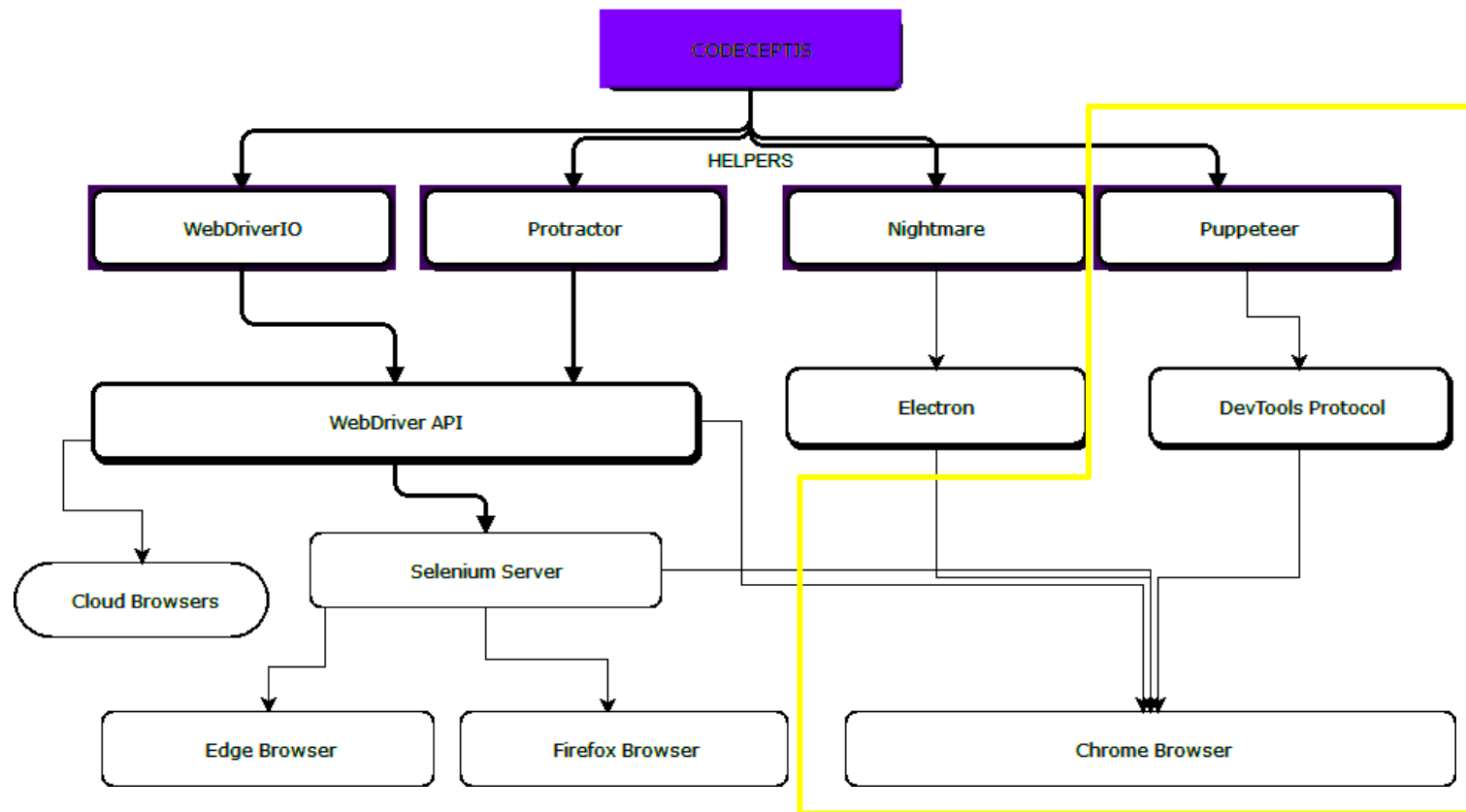
[Learn More](#)



Multi-Session Testing

Run a test using several browser windows

Architecture



Exemple sur SPA React

Puppeteer et chromium

```
Feature('Login Test');

//process.env.REACT_APP_REMOTE_LOGIN
//process.env.REACT_APP_REMOTE_PASSWD

Scenario('test wrong and correct login', async (I) => {
  I.amOnPage('http://localhost:3000/');
  I.see('...', 'h1');
  I.see('Connexion', 'h3');
  I.see('Se connecter', '.btn-primary');
  I.seeElement('#logo');

  I.fillField('email', 'toto@mail.com');
  I.fillField('password', 'wrongpassword123');
  I.click('Se connecter');
  I.see('Connexion impossible (login, mot de passe ou pas de connexion à la base de données)', '.toast');
  I.dontSee('Login réussi', '.toast');

  I.refreshPage();
  // await I.clearField('input[name=email]');
  // await I.clearField('input[name=password]');
  I.fillField('email', 'ltran@g-echo.fr');
  I.fillField('password', ' ');
  I.click('Se connecter');
  await I.see(' - EBIOS RM 2018', 'h6');
  I.see('Login réussi', '.toast');
  I.seeElement('h6');

  await I.click('#dropdownMenuButton');
  I.wait(2);
  I.seeElement('.dropdown-item');
  await I.click('Déconnexion');

  // On retrouve bien les titres de la page login.
  I.see('...', 'h1');
  I.see('Connexion', 'h3');
  I.dontSeeElement('h6');
});
```

Tip

Lors de votre installation puppeteer / chromium, vous rencontrerez le problème suivant:

```
✘ "before each" hook: codeceptjs.before for "test wrong and correct login" in 1298ms
```

```
Error: Failed to launch the browser process!
```

```
[4635:4635:0514/162016.628184:FATAL:zygote_host_impl_linux.cc(116)] No usable sandbox! Update your kernel or see https://chromium.googlesource.com/chromium/src/+/master/docs/linux_suid_sandbox_development.md for more information on developing with the SUID sandbox. If you want to live dangerously and need an immediate workaround, you can try using --no-sandbox.
```

Solution sur <https://codecept.io/helpers/Puppeteer/#puppeteer>

Plugins

<https://codecept.io/plugins/#allure>

Configuration

Configuration can either be taken from a corresponding helper (deprecated) or a from plugin config (recommended).

```
plugins: {  
  puppeteerCoverage: {  
    enabled: true  
  }  
}
```

Possible config options:

- `coverageDir` : directory to dump coverage files
- `uniqueFileName` : generate a unique filename by adding uuid

The screenshot displays the Allure test results interface. On the left, a sidebar shows navigation options: Overview, Categories, Suites, Graphs, Timeline, Behaviors, and Packages. The main area shows a list of test suites. Under the 'Yahoo test' suite, the test case '#1 Nightmare basic test' is highlighted in yellow and marked as failed. The test body for this case includes the following steps:

- I am on page "http://yahoo.com" (5s 534ms)
- I fill field "p", "github nightmare" (566ms)
- I click "Search Web" (506ms)
- I wait for element "#main", 2 (2s 001ms)
- Last Screenshot (373.7 KB)

The error message for the failed test is: "element (#main) still not present on page after 2 sec waiting for selector "#main" failed: timeout 2000ms exceeded". The test body also shows a 'Testing Begins' step that failed.

<https://codecept.io/plugins/#puppeteercoverage>

Usages avancés

Piloter par les données

Data Driven Tests

Execute the same scenario on a different data set.

Let's say you want to test login for different user accounts. In this case, you need to create a datatable and fill it in with credentials. Then use `Data().Scenario` to include this data and generate multiple scenarios:

```
// Define data table inside a test or load from another module
let accounts = new DataTable(['login', 'password']); //
accounts.add(['dave', '123456']); // adding records to a table
accounts.add(['admin', '123456']);

// You can skip some data. But add them to report as skipped (just like with usual scenar
accounts.xadd(['admin', '23456'])

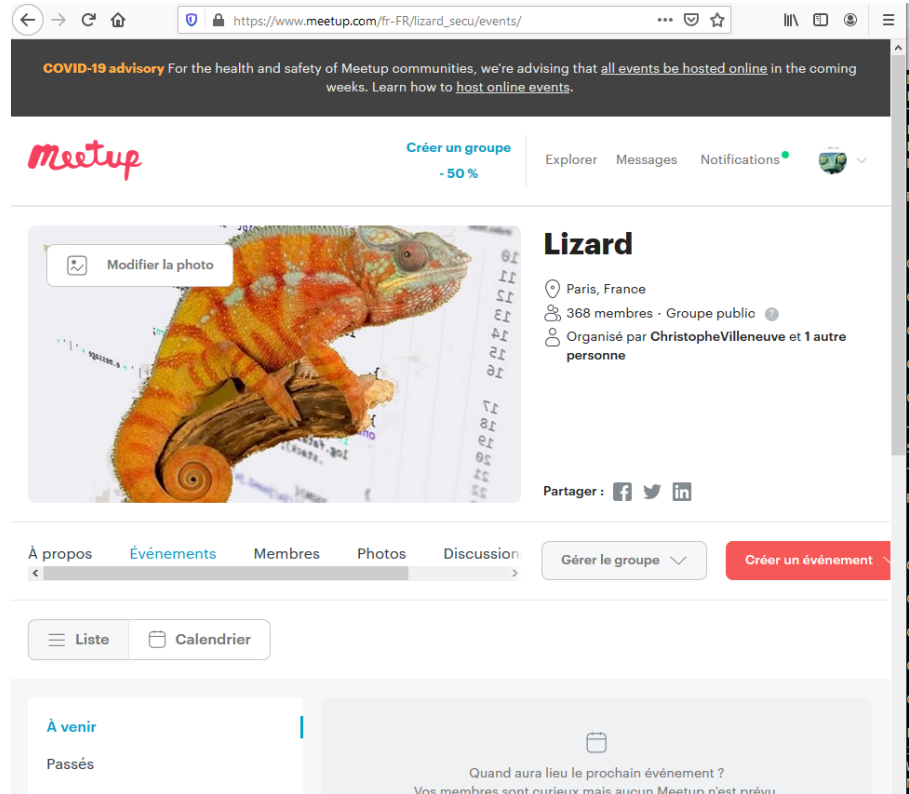
// Pass dataTable to Data()
// Use special param 'current' to get current data set
Data(accounts).Scenario('Test Login', (I, current) => {
  I.fillField('Username', current.login); // current is reserved!
  I.fillField('Password', current.password);
  I.click('Sign In');
  I.see('Welcome '+ current.login);
});
```


Encore plus loin ?

<https://www.g-echo.fr>

[hack-tualité](#)

Meetups [lizard](#)



The screenshot shows a web browser displaying a Meetup group page for 'Lizard'. At the top, there is a COVID-19 advisory banner. Below it, the Meetup logo is visible along with a 'Créer un groupe -50%' button. The group name 'Lizard' is prominently displayed, followed by its location 'Paris, France', '368 membres', and the organizer 'ChristopheVilleneuve et 1 autre personne'. A large image of a colorful lizard is featured with a 'Modifier la photo' button. Navigation tabs include 'À propos', 'Événements', 'Membres', 'Photos', and 'Discussion'. A 'Gérer le groupe' dropdown and a 'Créer un événement' button are also present. At the bottom, there are 'Liste' and 'Calendrier' buttons, and a section for 'À venir' (Upcoming) events, which currently shows no events with the message 'Quand aura lieu le prochain événement ? Vos membres sont curieux mais aucun Meetup n'est prévu.'

Tests d'accessibilité par la pratique



Christophe Villeneuve



@hellosct1

@hellosct1@mamot.fr



Aujourd'hui

- Accessibilité : Rappel
- Tests de validation manuelle
- Tests de validation automatique
dans un environnement CI / CD



Accessibilité : Rappel

Accessibilité numérique ?

L'accessibilité numérique signifie
que les sites web, technos et outils
sont conçus et développés
pour que les personnes handicapées
puissent les utiliser

C'est à dire



Percevoir



Comprendre



Naviguer



Interagir



Contribuer

Les types d'handicap



RGAA

- Référentiel Général d'Accessibilité pour les Administrations
- V 4.0 – publié 2019




Thématiques

- Images
- Cadres
- Couleurs
- Multimédia
- Tableaux
- Liens
- Scripts
- Éléments obligatoires
- Structure de l'information
- Présentation de l'information
- Formulaires
- Navigation
- Consultation

Exemple

1. Images

Critère 1.1. Chaque image **porteuse d'information** a-t-elle une **alternative textuelle** ? 

Test 1.1.1 : Chaque image (balise `` ou balise possédant l'attribut WAI-ARIA `role="img"`) **porteuse d'information** a-t-elle une **alternative textuelle** ?

Test 1.1.2 : Chaque **zone** d'une **image réactive** (balise `<area>`) **porteuse d'information** a-t-elle une **alternative textuelle** ?

Test 1.1.3 : Chaque bouton de type image (balise `<input>` avec l'attribut `type="image"`) a-t-il une **alternative textuelle** ?

Test 1.1.4 : Chaque zone cliquable d'une image réactive côté serveur est-t-elle doublée d'un lien dans la page ?

Test 1.1.5 : Chaque image vectorielle (balise `<svg>`) **porteuse d'information**, vérifie-t-elle ces conditions ?

- La balise `<svg>` possède un attribut WAI-ARIA `role="img"`.
- La balise `<svg>` a une **alternative textuelle**.



Tests de validation manuelle

Hints



- Vérifie les bonnes pratiques et les erreurs courantes :
 - Accessibilité
 - Performance
 - Sécurité
 - Compatibilité


- Addon Firefox
 - <https://addons.mozilla.org/fr/firefox/addon/webhint/>


- En ligne
 - <https://webhint.io/>


DEMO

Scan again


Configure a new scan

 Accessibility 0

 Compatibility 2 ●

 Performance 2 ●

 Pitfalls 0

 PWA 0

 Security 4 ●

Accessibility

- ▶ ARIA: **No issues**
- ▶ Color: **No issues**
- ▶ Forms: **No issues**
- ▶ Keyboard: **No issues**
- ▶ Language: **No issues**
- ▶ Name role value: **No issues**
- ▶ Other: **No issues**



Tests automatisés
environnement CI / CD

Tanaguru



- Outil open source (licence AGPL)
- Objectif de tester l'accessibilité du site
- Disponible
 - CI / CD
 - En ligne
 - Github



Démo Tanaguru

Exporter : [ODS](#) / [CSV](#) / [PDF](#)

Imprimer les résultats

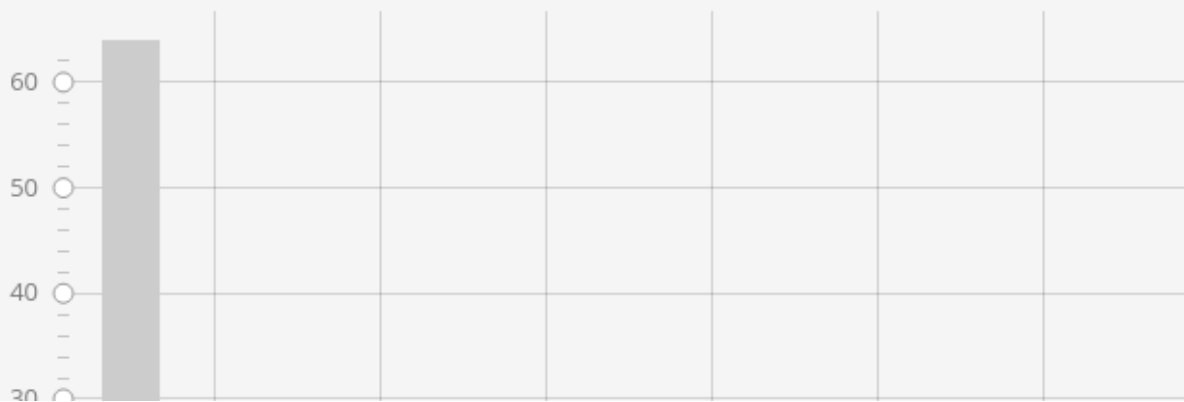
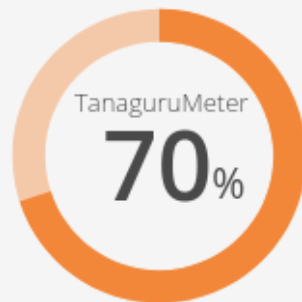
Relancer l'audit



Url : <https://www.lesfruitsdelaparis.com>
Créé le : 4 juin 2020 00:09:10

Code HTML

Référentiel : [RGAA 3 2017](#)
Niveau : AA
[Détails](#)














Validé

14

Thématique 1 : Images


✓ 0 ✕ 1 ? 3 NA 60 ⌚ 1

- ▶ **1.1.1** Chaque image (balise `img`) a-t-elle un attribut `alt` ?   
- 1.1.2** Chaque zone (balise `area`) d'une image réactive a-t-elle un attribut `alt` ?   
- 1.1.3** Chaque bouton de type image (balise `input` avec l'attribut `type="image"`) a-t-il un attribut `alt` ?   
- 1.1.4** Chaque zone cliquable d'une image réactive coté serveur est-t-elle doublée d'un lien dans la page ?  

▼ **1.1.1** Chaque image (balise `img`) a-t-elle un attribut `alt` ?   

18 éléments testés

↳ **18 éléments invalides**

 L'attribut `alt` est absent pour les éléments suivants :

attribut <code>src</code>	Code HTML de l'element	Ligne
	<pre></pre>	Ligne 84

CI / CD

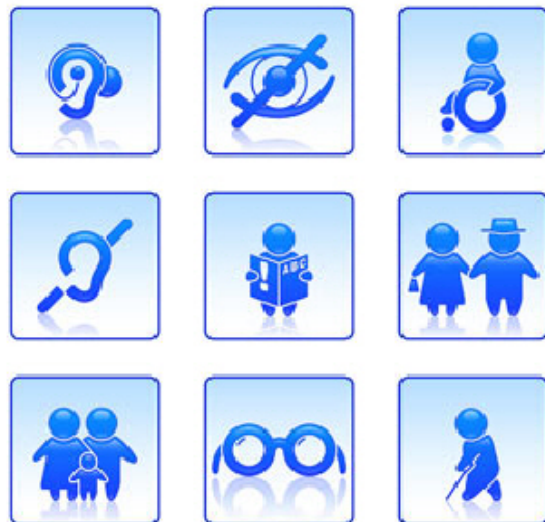
- Plugin Jenkins

The screenshot shows the Jenkins web interface for a project named 'Tanaguru-Runner'. The top navigation bar includes the Jenkins logo, a search bar, and the user 'mfaure' with a 'log out' link. Below the navigation bar, the breadcrumb 'Jenkins > Tanaguru-Runner' is visible, along with an 'ENABLE AUTO REFRESH' link. On the left side, there is a sidebar menu with several options: 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', 'Tanaguru', and 'Plots'. The 'Tanaguru' option is highlighted with a red box. The main content area is titled 'Project Tanaguru-Runner' and features a 'Disable Project' button. Below the title, there are four links: 'Tanaguru' (with a red box around its icon), 'Workspace', 'Recent Changes', and 'add description'.


Les liens

- W3C
 - <https://validator.w3.org/>
- RGAA
 - <https://www.numerique.gouv.fr/publications/rgaa-accessibilite/>
- WebHint
 - <https://webhint.io/>
- Tanaguru
 - <https://www.tanaguru.com/>
 - CI / CD
 - <https://tanaguru-jenkins-plugin.readthedocs.io>

Merci



Christophe Villeneuve

 @hellosct1



@hellosct1@mamot.fr