

TD "Test HTML / JS avec Cypress"

Notion de test "end-to-end"

Un test unitaire est censé tester qu'un seul composant (ou bloc de code).

A l'inverse, un test "end-to-end" permet de tester le bon fonctionnement de l'ensemble d'une application en voyant celle-ci comme une "boîte noire".

Caractéristiques générales des tests "end-to-end" (appelés aussi "tests d'intégration")

- Prérequis : L'ensemble de l'application doit être préalablement démarrée (avec quelquefois back-end + base de données en arrière-plan)
- C'est très souvent via l'interface graphique (web / HTML ou pas) que les tests interagissent avec l'application
- Un test de ce type permet plus ou moins de simuler le comportement d'un utilisateur qui manipule l'application à travers les différent(e)s écrans / pages

Les principales technologies disponibles pour effectuer des tests "end-to-end" ou "d'interfaces graphiques web / HTML" sont :

- Selenium (la référence depuis plus de 15 ans)
- Cypress (technologie plus moderne)
- ...

Présentation de cypress

Cypress est une technologie JavaScript un peu plus moderne que Selenium permettant de déclencher des tests "end-to-end".

Les principales différences entre Cypress et Selenium sont que :

- Selenium fonctionne au-dehors d'un navigateur et les ordres sont donnés au navigateur via l'intermédiaire d'un WebDriver
- Cypress fonctionne directement en JavaScript dans un navigateur : il est donc beaucoup plus rapide et peut beaucoup plus facilement interagir avec l'arbre DOM et XHR / Ajax

Selenium a cependant été utilisé massivement par un très grand nombre de testeurs. Selenium reste une RÉFÉRENCE incontournable.

Le "Test Runner" Cypress est open-source et fourni sous licence MIT.

Préparation d'un projet npm pour test Cypress

Récupérez le projet exemple `basic_html_test_cypress` en tant que sous partie du répertoire `tp-js` du référentiel <https://github.com/didier-tp/isika-al-frontend-tp> ou bien créez le de toutes pièces selon les indications suivantes:

- Créer un nouveau répertoire de projet `tp-js/basic_html_test_cypress` pour y intégrer une copie des fichiers suivants:
- `tp-js/basic_html_test_cypress/calculatriceV3.html`

```
<html>
<head>
  <title>calculatrice V3</title>
  <script src="./js/calcul.js" ></script>
  <script src="./js/calculatriceV3.js" ></script>
</head>
<body id="main" >
  <h1>calculatrice V3</h1>
  a: <input id="a" type="text" /> <br/>
  b: <input id="b" type="text" /> <br/>
  <input type="button" value="a+b" id="btn_op_addition" />
  <input type="button" value="a*b" id="btn_op_multiplication" />
  <input type="button" value="a-b" id="btn_op_soustraction" />
  <input type="button" value="a/b" id="btn_op_division" /> <br/>
  res:<span id="spanRes"></span>
  <hr/>
  <input type="checkbox" id="cbHisto" checked /> voir l'historique <br/>
  <ul id="ulHistorique" >
    <!-- <li>2*3=6</li> -->
  </ul>
</body>
</html>
```

- `tp-js/basic_html_test_cypress/js/calcul.js`

```
function calculerOp(op,a,b){
  a = Number(a);
  b = Number(b);
  var res = 0;
  if(op == '+'){
    res = a+b;
  }else if(op == '*'){
    res=a*b;
  }else{
    res =eval ("a"+op+"b"); //eval("a-b") ou eval("a/b") ou ...
  }
  return res;
}
```

- `tp-js/basic_html_test_cypress/js/calculatriceV3.js`

```
window.addEventListener("load",function(){

  var zoneCheckBox = document.getElementById("cbHisto");
  zoneCheckBox.addEventListener("change",cacherOuMontrerHistorique);

  var tabOpName = ["addition","soustraction","multiplication",
"division"];
  var tabOp = ["+", "-", "*", "/"];
```

```

    for(let i in tabOp){
        document.getElementById("btn_op_" +
tabOpName[i]).addEventListener("click" ,
            function(){calculerOperation(tabOp[i]);}
        );
    }
});

function cacherOuMontrerHistorique(){
    var zoneHistorique = document.getElementById("ulHistorique");
    var zoneCheckBox = document.querySelector("#cbHisto");
    if(zoneCheckBox.checked){
        zoneHistorique.style.display="block";
    }else{
        zoneHistorique.style.display="none";
    }
}

function calculerOperation(op){
    var a = Number(document.getElementById("a").value);
    var b = Number(document.getElementById("b").value);
    var res =calculerOp(op,a,b);
    document.getElementById("spanRes").innerHTML="<b>"+res+"</b>";
    var zoneHistorique = document.getElementById("ulHistorique");
    var li = document.createElement("li");
    li.innerHTML=""+ a + op + b +"="+res ;
    li.style.fontStyle='italic';
    zoneHistorique.appendChild(li);
}

```

- Dans le répertoire `tp-js/basic_html_test_cypress` , ajouter le fichier `index.html` comportant le code HTML suivant :

```

<html>
<body>
    <a href="calculatriceV3.html">calculatriceV3</a>
</body>
</html>

```

- Installer une version "LTS" de Node.js si ce n'est pas déjà fait
- Se placer dans le répertoire du projet `tp-js/basic_html_test_cypress` et lancer la commande `npm init` dans une fenêtre CMD et appuyer plein de fois sur `Enter` pour valider les choix par défaut

Remarque : Dans le cadre d'un test "end-to-end" de type "boîte noire" , **peu importe** si le code est en **pur javascript/api_DOM** ou bien si l'application est codée via **jquery** ou via **angular** ou **react**. Il est par contre fortement conseillé que **chaque zone importante** des pages HTML à tester comporte un **id précis**

Installation de Cypress

- Se placer dans le répertoire du projet `tp-js/basic_html_test_cypress` et lancer la commande suivante :
- `npm install cypress --save-dev` ou bien tout simplement `npm install` si est `cypress` déjà présent dans le fichier `package.json`

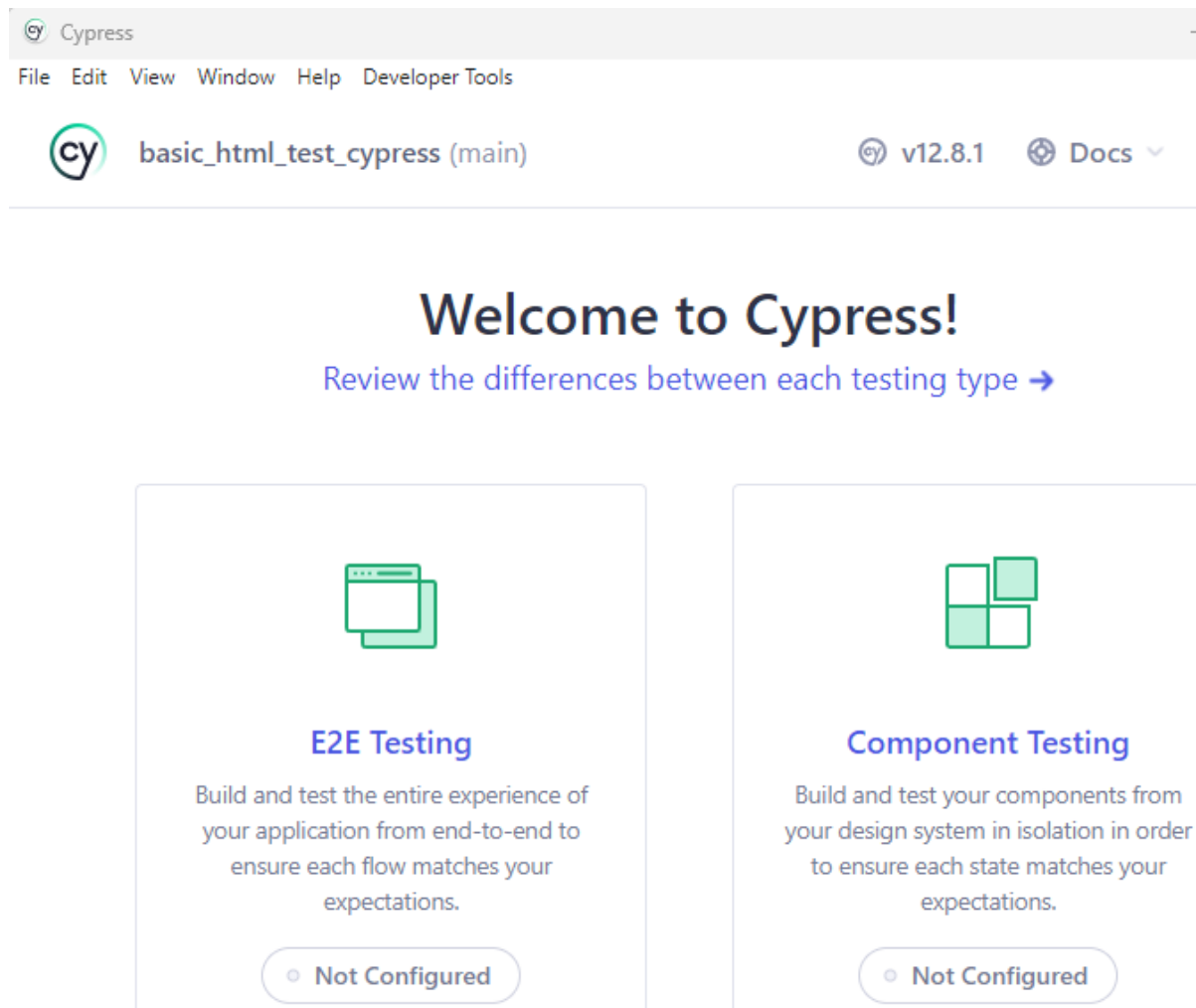
Attention : Prévoir de longues minutes de téléchargement ...


Premier lancement de cypress

Remarque : La commande `npx` sert simplement à lancer `npx cypress open` avec une syntaxe plus concise que l'équivalent `./node_modules/.bin/cypress open`.




- Lancer la commande `npx cypress open`

Remarque : Le premier lancement (avec un choix de E2E testing) va permettre la création de `cypress.config.js`, du sous-répertoire `cypress` et de quelques sous-répertoires. En cliquant sur **continue** on peut choisir un navigateur (ex: *Chrome*) puis dans la nouvelle fenêtre on pourra choisir **create new spec** avec comme nom `cypress/e2e/myTest.spec.cy.js`




 Cypress

File Edit View Window Help Developer Tools


 basic_html_test_cypress (main) > E2E Testing  v12.8.1  Docs ▾

Configuration files


We added the following files to your project:




cypress.config.js
The Cypress config file for E2E testing.



cypress\support\e2e.js
The support file that is bundled and loaded before each E2E spec.




cypress\support\commands.js
A support file that is useful for creating custom Cypress commands and overwriting existing ones.







cypress\fixtures\example.json
Added an example fixtures file/folder

Continue


 Cypress


File Edit View Window Help Developer Tools


 basic_html_test_cypress (main) > E2E Testing  v12.8.1  Docs ▾  Log in


Choose a browser

Choose your preferred browser for E2E testing.


Chrome
v111


Edge
v111


Electron
v106


Firefox
v111

Start E2E Testing in Chrome

Enter the path for your new spec

 cypress\e2e\myTest.spec.cy.js

Create spec

Back

Après un lancement de `cypress open`, il faut en théorie choisir un test à lancer au sein de la fenêtre de sélection. Nous devons cependant commencer par mieux écrire notre nouveau test.

- À arrêter (en fermant cette fenêtre) et à relancer ultérieurement.

Écriture d'un nouveau test Cypress

- Au sein du répertoire `tp-js/basic_html_test_cypress/cypress/e2e`, créer ou bien modifier le fichier `myTest.spec.cy.js` qui comportera à peu près le code suivant :

```
describe('My HTML/JS Tests', () => {
  it('good addition in calculatriceV3.html', () => {

    //partir de index.html
    cy.visit("http://localhost:3000/index.html")

    //cliquer sur le lien comportant 'calculatriceV3'
    cy.contains('calculatriceV3').click()
    cy.wait(50)
    // Should be on a new URL which includes '/calculatrice'
    cy.url().should('include', '/calculatrice')

    // Get an input, type data into it
    //and verify that the value has been updated
    cy.get('#a')
      .type('5')
      .should('have.value', '5')

    cy.get('#b')
      .type('6')
      .should('have.value', '6')

    //declencher click sur bouton addition
```

```

    cy.get('#btn_op_addition')
      .click()

    //vérifier que la zone d'id spanRes comporte le texte '11'
    cy.get('#spanRes')
      .should('have.text', '11')
  })

  it('good multiplication in calculatriceV3.html', () => {

    //visiter calculatriceV3.html
    cy.visit("http://localhost:3000/calculatriceV3.html")

    cy.get('#a').type('3').should('have.value', '3')

    cy.get('#b').type('4').should('have.value', '4')

    //declencher click sur bouton multiplication
    //cy.get('#btn_op_multiplication').click()
    cy.get('#btn_op_multiplication').trigger("click")

    //vérifier que la zone d'id spanRes comporte le texte '12'
    cy.get('#spanRes')
      .should('have.text', '12')
  })

  it('Historique cache ou bien affiche', () => {
    //visiter calculatriceV3.html
    cy.visit("http://localhost:3000/calculatriceV3.html")

    cy.get('#a').type('2')
    cy.get('#b').type('3')
    cy.get('#btn_op_addition').click()
    cy.get('#spanRes').should('have.text', '5')

    cy.get("#cbHisto").check()
    cy.get("#ulHistorique").should('be.visible')//ok meme si display:block

    cy.get("#cbHisto").unchecked()
    cy.get("#ulHistorique").should('be.hidden')//ok meme si display:none
  })
})

```

- Ajuster éventuellement le fichier `cypress.config.js` de la façon suivante :

```

const { defineConfig } = require("cypress");

module.exports = defineConfig({
  e2e: {
    setupNodeEvents(on, config) {
      // implement node event listeners here
    },
  },
  video : false
});

```

Démarrage de l'application web HTML / JS à tester via lite-server

- Lancer, si besoin, l'installation du mini serveur lite-server via la commande suivante :

```
npm install -g lite-server
```

- Démarrer ensuite le serveur via la commande suivante (à lancer dans le répertoire contenant `index.html`) :

```
lite-server
```

Remarque : <http://localhost:3000> et Futur arrêt via `Ctrl-C`

Lancement d'un test Cypress

- Lancer la commande suivante sur une seule ligne (depuis le répertoire du projet `basic_html_test_cypress`) :

```
npx cypress run --spec "cypress/e2e/myTest.spec.cy.js" --browser chrome  
>test_report.txt
```

- Remarques :
 - Option `headless` par défaut pour ne pas montrer le navigateur sous contrôle
 - En absence de l'option `--browser chrome` alors le navigateur "electron" utilisé par défaut
 - Ajouter dans `e2e:{ ...}` (à la fin), `"video" : false` dans `cypress.config.js` permet de désactiver l'enregistrement vidéo
- Rapport `test_report.txt` généré :

...

```
Cypress:    6.8.0  
Browser:    Chrome 89  
Specs:      1 found (myTest.spec.js)  
Searched:   cypress\integration\myTest.spec.js
```

```
Running:  myTest.spec.js  
(1 of 1)
```

```
My HTML/JS Tests  
  ✓ good addition in calculatriceV3.html (1519ms)  
  ✓ good multiplication in calculatriceV3.html (1424ms)  
  ✓ Historique cache ou bien affiche (2633ms)
```

```
3 passing (9s)
```


...

Remarque : On peut également se contenter de lancer `npx cypress open` puis choisir le test à déclencher dans la fenêtre `cypress ...`

- Introduire volontairement 2 petites erreurs dans le fichier `js/calcul.js` comme, par exemple :

```
if(op == '+'){
    res = a+b+1;
}else if(op == '*'){
    res=a*b-1;
}
```

- Détails obtenus :

```
10 get      #spanRes
11 - assert  expected <span#spanRes> to have text '12', but
           the text was '11'
```

❶ AssertionError

Timed out retrying after 4000ms: expected '<span#spanRes>' to have text '12', but the text was '11'

```
cypress/integration/myTest.spec.js:51:8

  49 |     //vérifier que la zone d'id spanRes comporte 1
  50 |     cy.get('#spanRes')
> 51 |     .should('have.text', '12')
    |         ^
  52 | })
  53 |
```

- Rétablir le bon code dans le fichier `js/calcul.js` pour obtenir normalement le résultat suivant :

← → ↺

localhost:3000/_/#/tests/integration\myTest.spec.js

Chrome est contrôlé par un logiciel de test automatisé.

Tests

✓ 3 ✗ -- ⌛ --

03.69

🔍 ⌂ ↺

http://localhost:3000/calculatriceV3.html

cypress\integration\myTest.spec.js

My HTML/JS Tests

✓ good addition in calculatriceV3.html

✓ good multiplication in calculatriceV3.html

TEST BODY

1 visit http://localhost:3000/calculatriceV3.html

(xhr) GET 200 /browser-sync/socket.io/?EIO=3&transpo...

2 get #a

3 - type 3

(xhr) POST 200 /browser-sync/socket.io/?EIO=3&transpo...

(xhr) GET 200 /browser-sync/socket.io/?EIO=3&transpo...

(xhr) POST 200 /browser-sync/socket.io/?EIO=3&transpo...

(xhr) GET 200 /browser-sync/socket.io/?EIO=3&transpo...

(xhr) POST 200 /browser-sync/socket.io/?EIO=3&transpo...

4 - assert expected <input#a> to have value '3'

5 get #b

6 - type 4

(xhr) POST 200 /browser-sync/socket.io/?EIO=3&transpo...

(xhr) POST 200 /browser-sync/socket.io/?EIO=3&transpo...

7 - assert expected <input#b> to have value '4'

8 get #btn_op_multiplication

9 - trigger click

10 get #spanRes

11 - assert expected <span#spanRes> to have text '12'

✓ Historique cache ou bien affiche

calculatrice V3

a:

b:

a+b

a*b

a-b

a/b

res:5

☐ voir l'historique

file:///C:/Conf/entreprise/d2f/jupyter_wksp/d2f/Td_cypress/td_cypress.html

10/10