



# Construction d'un site e-commerce en JavaScript

Kanap  
10 quai de la charente  
75019 Paris 19

# La mission

- Développer le site e-commerce pour la boutique de vente de canapés Kanap.
- La partie affichage et liaison avec le serveur sont déjà développées.
- La mission consiste à
  - Intégrer les éléments dans les différentes pages WEB en Java Script
  - Mettre en place un plan de test.

# La mission en détails

- Dialoguer avec le serveur pour récupérer les informations
- Mettre à jour la page HTML d'accueil
- Mettre à jour la page produit avec le produit sélectionné par l'utilisateur.
- Mettre à jour la page panier avec les produits sélectionnés
  - Permettre la modification du panier:
    - Ajout , diminution ou suppression d'un élément du panier
  - Vérifier le formulaire avec les coordonnées de l'utilisateur
  - Envoyer la commande

# Le langage Java Script

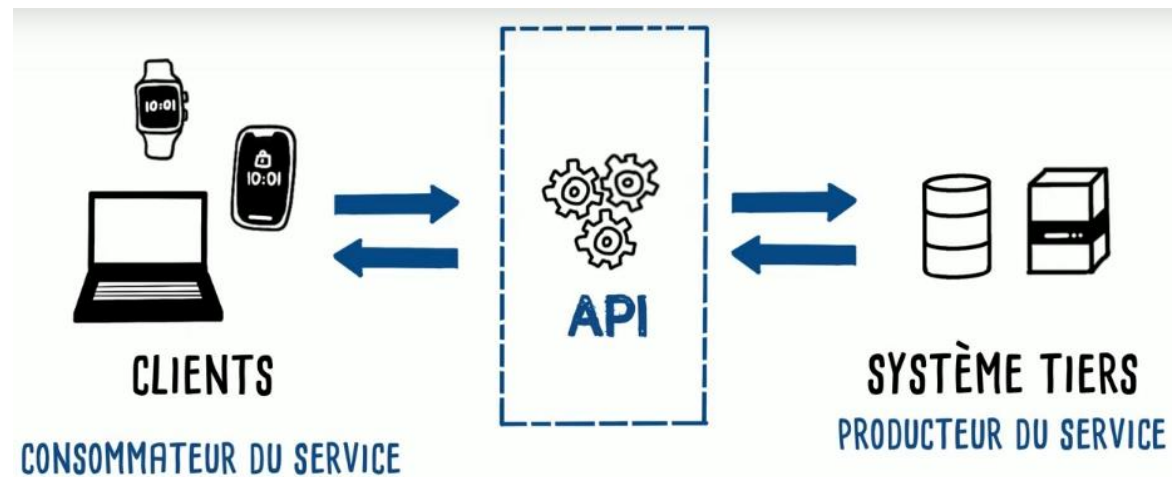
- Pour réaliser cette mission, il est demandé d'utiliser le langage Java Script.
- JavaScript est un langage de programmation qui permet d'implémenter des mécanismes complexes sur une page web.
- Java Script permet:
  - De modifier dynamiquement des pages HTML
  - D'écouter des évènements et de lancer un traitement approprié.
  - De dialoguer avec le serveur par l'intermédiaire d'une interface.
    - Envoi d'informations
    - Récupération d'informations

# Les 3 langages des sites WEB

- [HTML](#) est un langage de balises utilisé pour structurer et afficher un contenu web.
- [CSS](#) est un langage de règles de style utilisé pour mettre en forme le contenu HTML.
- [JavaScript](#) est un langage de programmation qui permet de créer du contenu mis à jour de façon dynamique.

# Définition: Une API qu'est-ce que c'est ?

- **API:** (Application Programming Interface) signifie en français : « Interface de Programmation d'Application».
- une interface de programmation permet d'accéder à des services fournis par un système tiers.



# L'API de Kanap

- Toutes les caractéristiques des produits disponibles sont stockés sur un serveur.
- L'API de KANAP permet de dialoguer avec ce serveur
- Il est possible grâce à des services adaptés de récupérer les informations de chaque produit.
- Lorsque la commande est terminée, la liste des produits commandés et les coordonnées de l'acheteur sont envoyées au serveur.

# Les infos de Kanap dans le serveur

Le serveur contient tous les produits disponibles à la vente avec pour chaque produit les informations suivantes:

- Id
- Nom
- Description
- Tableau de couleurs possibles pour le modèle
- Prix
- Lien vers l'URL de l'image
- Texte alternatif pour l'image



# Dialogue avec l'API – Informations du serveur

La première étape du projet consiste à recueillir les informations du serveur pour qu'il nous renvoie les produits à afficher sur la page d'accueil.

Ce recueil de données se fait par des requêtes sur l'API .

# Dialogue avec l'API

- Pour communiquer avec l'API, il faut lui envoyer des requêtes respectant le protocole HTTP.
- Avec:
  - Une méthode:
    - **GET** : permet de **récupérer** des ressources
    - **POST** : permet de **créer** ou **modifier** une ressource
    - **PUT** : permet de **modifier** une ressource
    - **DELETE** : Permet de **supprimer** une ressource
  - Une URL
  - Des données

# Dialogue avec l'API de KANAP : fetch

Fetch : est un ensemble d'objets et de fonctions mis à disposition par le langage JavaScript, afin d'exécuter des requêtes HTTP de manière asynchrone.

Ex:

```
let response = await fetch(URL du serveur );  
if (response.ok) {  
    data = await response.json();
```

fetch demande au serveur de lui renvoyer les données

Les données sont renvoyées au format json

await permet d'attendre la réponse le temps d'interroger le serveur.

# Dialogue avec l'API de Kanap

- Pour le projet Kanap, les informations sur le serveur sont récupérées grâce à fetch pour:
  - Afficher la page d'accueil
  - Sur la page produit, afficher le produit sélectionné
  - Sur la page panier, rechercher les informations de chaque produit affiché comme le prix, l'image.
- Pour chaque appel de fetch, le mot clé await permet d'attendre l'exécution de la commande avant d'en utiliser le résultat.

# Format des données: JSON

- Tous les objets de Kanap stockés sont au format JSON: JavaScript Object Notation qui est le format le plus simple pour transmettre les données.
  - **Json.parse()** transforme le json en objet java script.
  - La méthode **JSON.stringify()** convertit une valeur JavaScript en chaîne JSON
- Exemple de format json du projet:
  - `{"id":"034707184e8e4eefb46400b5a3774b5f","nb":"1","couleur":"Red"}`
- Objet JS correspondant:
  - `{id: '034707184e8e4eefb46400b5a3774b5f', nb: '1', couleur: 'Red'}`
    - **couleur:** "Red"
    - **id:** "034707184e8e4eefb46400b5a3774b5f"
    - **nb:** "1"
    - **[[Prototype]]:** Object

# Le DOM

- Une fois les infos récupérées sur le serveur, il faut modifier la page HTML afin de les afficher à l'écran.
- Il faut pour cela utiliser le DOM



# Le DOM

- **DOM:Document Object Model:** est une interface de programmation normalisée par le W3C , qui permet à des scripts d'examiner et de modifier le contenu du navigateur WEB.
- Grâce au DOM, il est possible d'accéder à tous les éléments d'une page HTML, de les créer, les modifier ou les supprimer.
- Le document est représenté comme un arbre.
- À l'aide du DOM, un script peut modifier le document présent dans le navigateur en ajoutant ou en supprimant des nœuds de l'arbre

# Le DOM : fonctions js pour y accéder

- Le Java script accède aux éléments du DOM par des fonctions spécifiques:
- Il est possible d'accéder aux éléments par leur ids, aux class CSS par leur nom.
  - Id: les id du html sont accessibles par la fonction
    - `document.getElementById`: accède à un élément par son id
      - -> Importance de n'avoir qu'un seul nom d'id dans une page html.
    - `document.getElementsByClassName`
      - donne la liste de tous les éléments d'une classe d'un nom donné



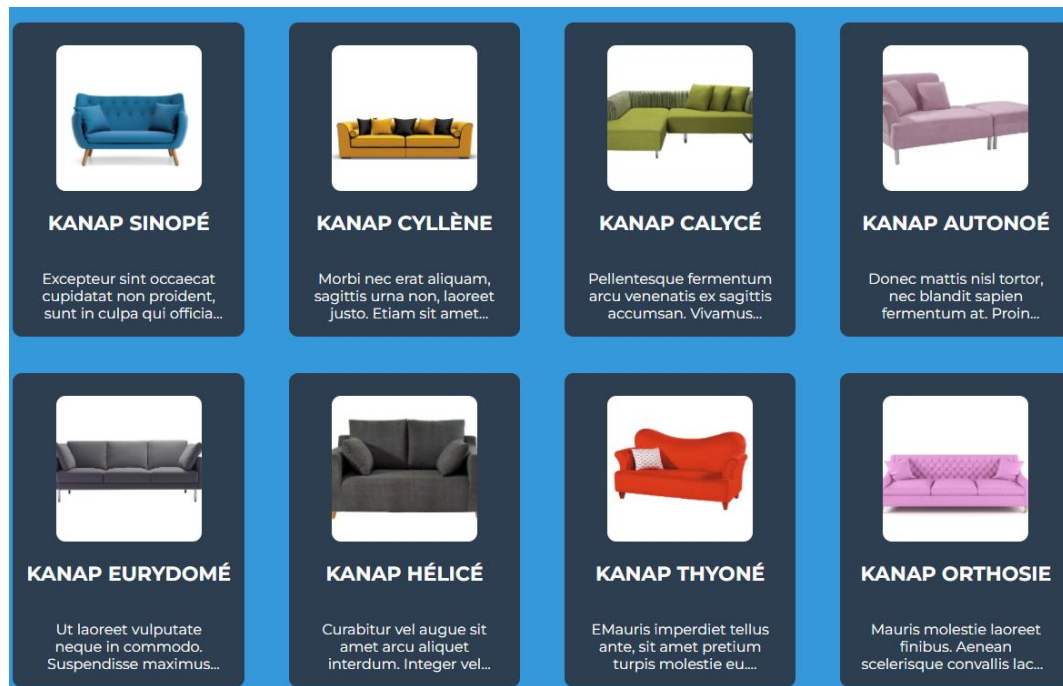
# Le DOM

- **Le Java script peut modifier les éléments du DOM:**
- Par exemple, il est possible de créer de nouveaux éléments:
  - CreateElement crée un nouvel élément
  - Et appendchild le relie à son parent.
- Exemple: création d'une nouvelle article enfant de la section id="cart\_\_items"
  - `let eltSection = document.getElementById(cart__items);`
  - `let newArticle = document.createElement("article");`
  - `eltSection.appendChild(newArticle);`

Résultat HTML: `<section id="cart__items">`  
`<article ..... /article>`

# DOM: Affichage page accueil de Kanap

- A partir des informations du serveur, il est possible de compléter la page accueil en construisant chaque élément à afficher



# DOM: Page accueil de Kanap

- Dans la page index, chaque article est défini comme suit:

```
<a href="./product.html?id=42">
```

```
<article>
```

```

```

```
<h3 class="productName">Kanap name1</h3>
```

```
<p class="productDescription">Description</p>
```

```
</article>
```

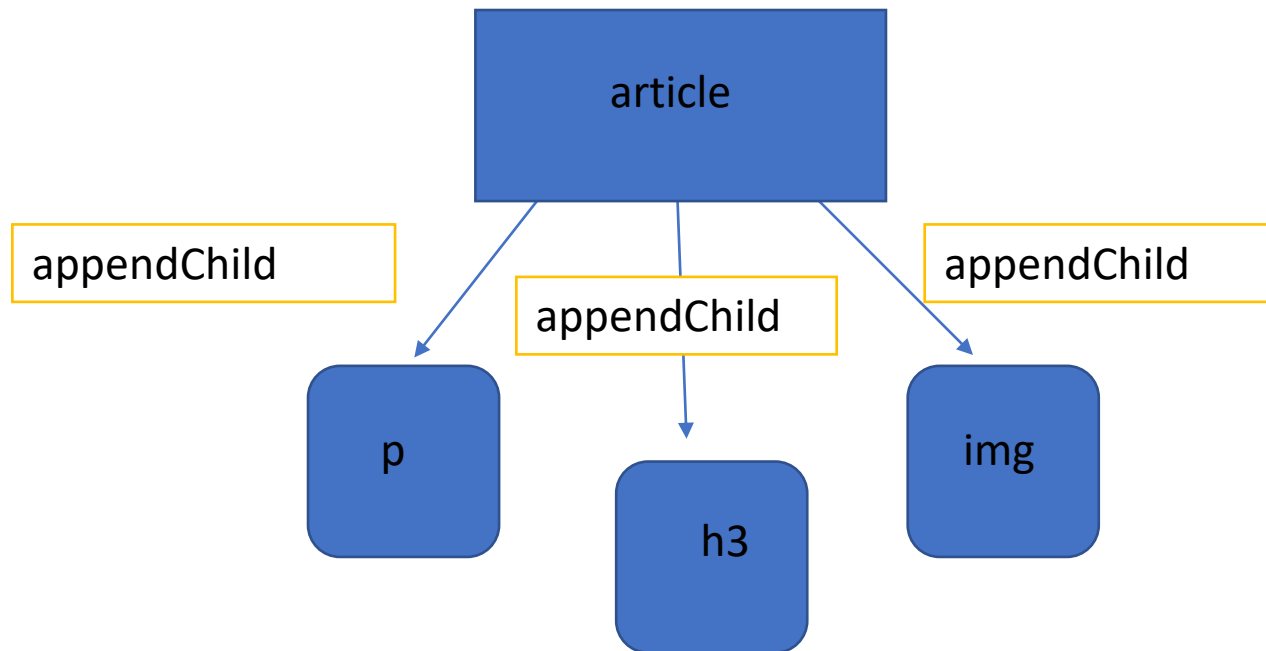
```
</a> -->
```

# DOM:Page accueil de Kanap traduction en js

- Le js va créer et remplir chaque élément HTML:
- `document.getElementById("items")`: récupère l'élément avec id "items"
- Création de tous les éléments enfants avec `document.createElement`
  - Le lien "a"
  - L'image: "img"
  - La balise "h3"
  - La balise "p"

# DOM:Page accueil de Kanap traduction en js

- A partir d'un élément Article, On ajoute les enfants avec appendChild



# DOM: Page accueil de Kanap traduction en js

On ajoute les classes des éléments avec

- `newh3.classList.add("productName");`
- Mise à jour du html avec `innerHTML`

Js:

- `newh3.classList.add("productName");`
- `newh3.innerHTML = Kanap name1;`
- Html correspondant mis à jour:  
`<h3 class="productName">Kanap name1</h3>`

# Transmission des informations par l'URL

- Dans le projet Kanap, il faut transmettre le produit choisi de la page d'accueil vers la page produit.
- Il faut ensuite transmettre les informations de couleur et de nombre de produits vers la page panier.
- Il faut enfin transmettre le numéro de la commande vers la page confirmation.

# Transmission des informations par l'URL

- Il est possible de transmettre des informations d'une page à une autre par l'URL
- Les paramètres sont écrits à la suite de l'URL destinatrice avec un "?" Avant le premier paramètre puis un "&" entre chaque paramètre.
  - Exemple: <URL>?id=value&param1=value&param2=value
- La méthode `URLSearchParams` permet de récupérer les paramètres de l'URL.



# Transmission des informations par l'URL

- Transmission entre page produit et panier
- `<Url>/cart.html?id=034707184e8e4eefb46400b5a3774b5f&color=Red&nb=2`
- Les paramètres transmis sont: id, color et nb
  - `id=034707184e8e4eefb46400b5a3774b5f`
  - `color=Red`
  - `nb=2`

# Transmission des informations par l'URL

- Exemple de code pour récupérer les informations de l'URL par
- URLSearchParams:

```
let searchParams = new URLSearchParams(url.search);  
if (searchParams.has("id")) {  
    param = searchParams.get("id");  
  
}
```

- has: vérifie l'existence du paramètre de nom "id"
- get: récupère la valeur du paramètre "id".

# Le local storage

Le stockage web local ou stockage DOM (Document Object Model), ou encore localStorage, est une **technique d'enregistrement de données dans un navigateur web.**

Les données sont mémorisées à vie dans un même navigateur.



# Le local storage

Chaque élément du local storage est représenté par:  
(Une clé,valeur)

Des fonctions spécifiques permettent d'accéder aux éléments du local storage.

# Le local storage de Kanap

- Seules les données spécifiques qui ne sont pas dans le serveur sont stockées dans le local storage:
- Pour chaque produit on aura:
  - La clé = id\_couleur
  - Les données:
    - Id
    - Couleur
    - Nombre de produits

# Le local storage de Kanap

- La clé du local storage sera aussi le nom de l'id de l'article HTML correspondant dans la page panier
- Grace à l'id, on pourra joindre l'article dans le HTML et le produit dans le local storage.
- Les autres informations du produit comme son prix seront à récupérer dans le serveur par fetch.

# Le local storage de Kanap

- Le local storage est mis à jour à chaque changement sur un item:
  - Ajout d'un produit: ajout de la clé dans le local storage
  - Suppression d'un produit: suppression de la clé dans le local storage
    - `localStorage.removeItem("cle");`
- Changement du nombre de produits: mise à jour du nombre de produits dans le local storage:
  - `getItem`: pour lire un elt
  - `setItem` pour mettre à jour un élément.
- Envoi de la commande: réinitialisation du local storage par la commande `"clear": localStorage.clear();`

# Les évènements

les événements sont des actions ou des occurrences qui se produisent dans le système.

le système déclenche un signal lorsqu'un événement se produit, et fournit un mécanisme pour exécuter automatiquement une action



Un événement peut être :

- Un click
- Le survol
- Le changement d'une donnée
- Une action sera déclenchée sur chaque événement par du code adapté.



# Ecouter les évènements

Le Java Script permet d'écouter les évènements grâce à la fonction `AddEventListener`

Exemple :

```
elts.addEventListener("click", async function () {Function})
```

- Attendra le click sur un élément
- et déclenchera une action par la fonction appelée.



# Ecouter les évènements: Kanap

- **Pour le projet, on écouterà les évènements:**
  - Sur les boutons "supprimer" de la page panier
  - Sur les entrées des champs "nb" de la page panier
  - Sur les champs du formulaire
  - Sur le bouton "Commander"

# Ecouter les évènements: Kanap

- **Bouton "supprimer" de la page panier:**
  - **Ecoute du click sur le bouton**
    - Suppression du noeud HTML correspondant.
    - Suppression de la clé dans le local storage.
    - Mise à jour de la quantité totale
    - Mise à jour du prix total
- **Input de la page panier**
  - **Ecoute du "change" de la valeur**
    - Correction du nombre d'éléments
    - Mise à jour de la quantité totale
    - Mise à jour du prix total

# Ecouter les évènements: Kanap

- **Champs du formulaire de la page panier:**
  - Ecoute du click sur les champs du formulaire
    - Vérification du champ rentré
- **Bouton "commander" de la page panier**
  - Ecoute du "click" sur le bouton
    - Si tous les champs du formulaire sont corrects, la commande est envoyée à l'API

# Les formulaires et les regex

- Le projet Kanap contient un formulaire que l'utilisateur devra remplir avant d'envoyer sa commande.
- Avant d'envoyer un formulaire, il est indispensable d'en vérifier les éléments.
- La vérification des chaînes de caractères se fait à l'aide de modèles appelés regex.
- Par exemple:
  - [a-zA-Z] autorisera des lettres minuscules ou majuscules.
  - [0-9] des chiffres.

# Les formulaires et les regex

- Pour le formulaire de Kanap, on vérifiera:
  - Que tous les champs sont remplis
  - Que les noms et prénoms ne contiennent pas de chiffre
  - Que l'adresse mail contient un @
- Tant que le formulaire ne sera pas rempli correctement, un message d'erreur avertira l'utilisateur et la commande ne pourra pas partir.

# Envoi des données à l'API

- La dernière partie du développement du projet consiste à envoyer les informations de la commande.
- Ceci se fait grâce à une requête POST composée de:
  - Les coordonnées de l'utilisateur
  - Un tableau avec les ids des produits sélectionnés.
- L'API répond à la requête en renvoyant un numéro de commande qui sera affichée dans la page confirmation.

# Tests d'acceptation

- Chaque développement doit être accompagné d'un plan de test.
- On distingue:
  - Les tests nominaux: fonctionnement normal
  - Les tests d'erreur: fonctionnement quand une erreur est commise ou une action non demandée est effectuée
  - Les tests aux limites quand on utilise le produit au maximum de sa capacité.
- Pour réaliser le plan de test de Kanap:
  - Pour chaque page, test du fonctionnement normal
  - Test des fonctionnements en erreur

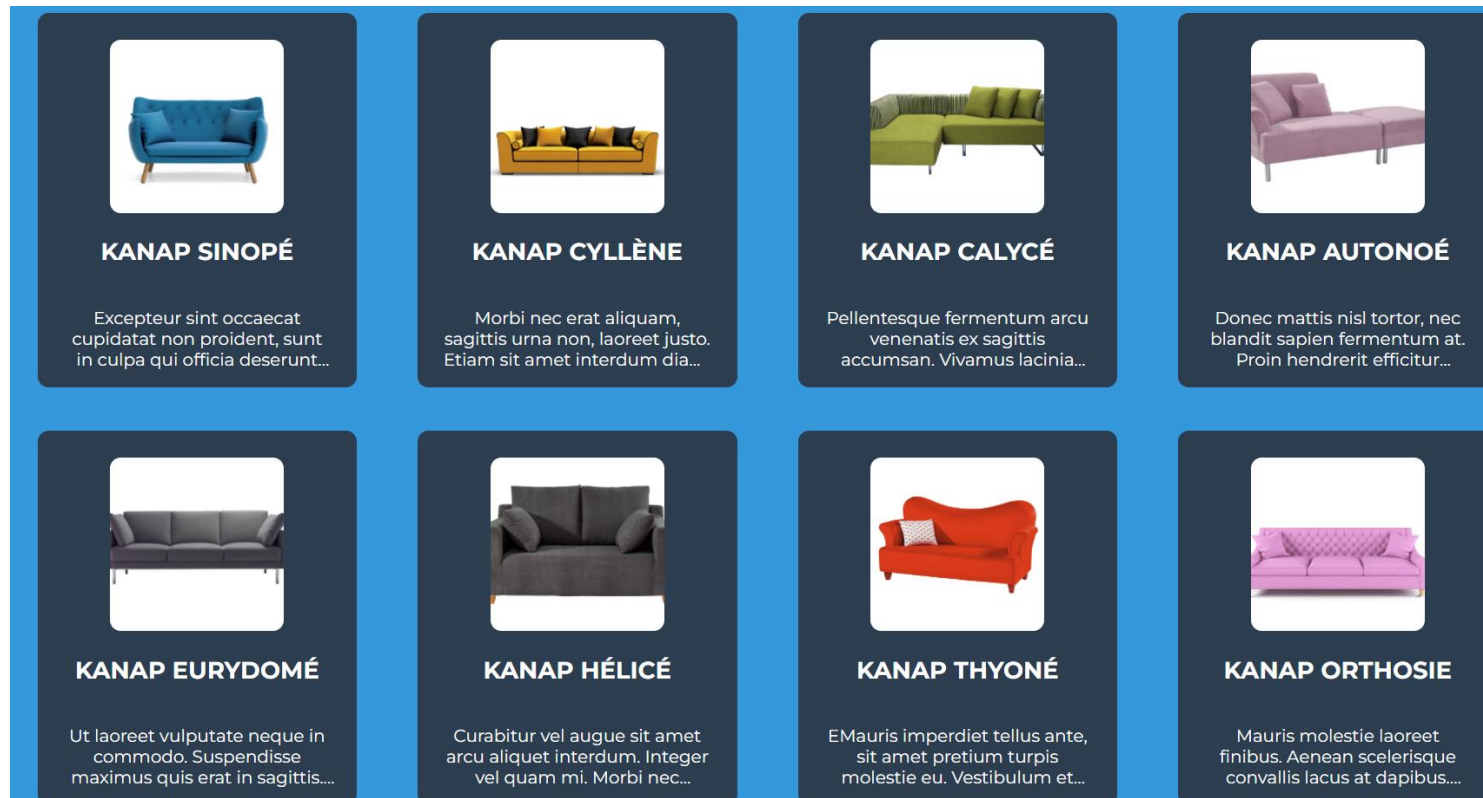


# Tests d'acceptation: les familles de test

Famille de test	Types de tests
Page accueil	Tests nominaux Tests aux limites
Page produit	Tests nominaux Tests d'erreurs Tests aux limites
Page panier - éléments du panier	Tests nominaux Tests d'erreur Tests aux limites
Page panier - formulaire	Tests nominaux Tests d'erreur
Page confirmation	Test nominal

# Test de la page d'accueil

- Vérification de l'affichage de la page avec tous les produits:



# Test nominal page produit

- Test de l'affichage du produit
  - avec les bonnes informations: prix, photo
- Possibilité de choisir une couleur
- Possibilité de choisir un nombre de produits
- Bascule sur la page panier



## Kanap Thyoné

Prix : 1999€

### Description :

EMauris imperdiet tellus ante, sit amet pretium turpis molestie eu. Vestibulum et egestas eros. Vestibulum non lacus orci.

Choisir une couleur :


Red

Nombre d'article(s) (1-100) :

3

Ajouter au panier

# Tests page panier: affichage produits



Kanap Thyoné

Red

1999€

Qté

Supprimer

Total (3 articles) : 5997 €

# Tests de la page panier

- Sur la page panier, il faut vérifier:
  - L'ajout ou la suppression de produits avec la mise à jour du nombre de produits et du prix mis à jour

# Tests page panier : formulaire valide

Total (3 articles) : 5997 €

Prénom:  
Brigitte

Nom:  
Doe

Adresse:  
5 rue des cigognes

Ville:  
Turlututu

Email:  
Brigitte@hotmail.com

Commander !

# Test: Envoi de la commande



# Tests d'erreur : page produit

- Page produit:
  - Aucune sélection de couleur
  - Aucune sélection de nombre
  - Nombre de produits supérieure à 100
  - Nombre de produit en valeur décimale
    - Dans tous les cas, le test doit vérifier l'affichage d'un message d'erreur
    - Le nombre de produits est remis à 0



# Tests d'erreur page panier - éléments du panier

- Pour tester les erreurs de la page panier, on peut vérifier:
- -un nombre invalide: >100
  - Le nombre est forcé à 100
- Un nombre décimal
  - Le nombre est forcé à 1
- Une entrée égale à 0 : la suppression se fait par le bouton supprimer
  - Le nombre est forcé à 1
- Le test doit vérifier que chaque entrée invalide entraine l'affichage d'un message d'erreur.

# Test du formulaire

Test avec :

- champs vide
- Champ invalide : exemple adresse mail invalide
- Le test doit vérifier le message d'erreur inscrit sous le champ invalide.
- Le fait que la commande ne part pas tant que le formulaire n'est pas valide.



The image shows a blue rectangular box representing a form. Inside the box, the word "Email:" is written in white. Below it is a white rounded rectangular input field containing the text "toto". Underneath the input field, the text "Entrée invalide" is written in a light red color.

# Tests d'erreur sur le formulaire

Test d'un prénom invalide:

Par exemple un prénom avec des chiffres



Prénom:

Brigitte444

Entrée invalide

A screenshot of a web form with a blue background. The label 'Prénom:' is in white. The input field contains the text 'Brigitte444'. Below the input field, the text 'Entrée invalide' is displayed in red.

Test avec champ vide



Nom:

Valeur obligatoire requise

A screenshot of a web form with a blue background. The label 'Nom:' is in white. The input field is empty. Below the input field, the text 'Valeur obligatoire requise' is displayed in red.

# Tests d'acceptation

- Les tests d'acceptation sont utiles:
  - Pour tester le développement
  - Pour faire des tests de non régression en cas de modifications dans le code.
- Pour le plan de test, j'ai ajouté de façon précise les tests à exécuter et les résultats attendus :
  - Par exemple tel canapé, tel nombre, tel couleur
  - Ces informations facilitent les tests de non régression.

# Conclusion

La partie dynamique du projet Kanap a été implémentée.

Les jeux de tests

- permettent de tester le code implémenté
- serviront de tests de non régression en cas de modification du code.

# Liens utiles

- [Définition API en 4 mn](#)
- Cours grafikart : <https://grafikart.fr/formations/debuter-javascript>
- tuto local storage: <https://tutowebdesign.com/localstorage-javascript.php>
- UrlSearchParams: <https://www.journaldunet.fr/web-tech/developpement/1202495-comment-obtenir-les-parametres-d-url-en-javascript/>

# Liens utiles

- [https://developer.mozilla.org/fr/docs/Learn/JavaScript/Building\\_blocks/Events](https://developer.mozilla.org/fr/docs/Learn/JavaScript/Building_blocks/Events)
- jsonplaceholder : bibliotheque de json:  
<https://jsonplaceholder.typicode.com/users>

# Glossaire

- API: Application Programming Interface
- DOM: Document Object Model
- JSON: JavaScript Object Notation.
- [HTTP](#): HyperText Transfert Protocol