

Procédure de Test Orinoco

Class Product.js :

- ❖ Récupération de tous les produits :

- On appelle la méthode `all()` de la manière suivante : `Product.all()`
- Le résultat devrait être équivalent à l'image :

```
▼ (5) [{"_id": "5be1ed3f1c9d44000030b061", "name": "Zurss 50S", "price": 49900, "description": "Lorem ipsum..."}, {"_id": "5be1ef211c9d44000030b062", "name": "Hirsch 400DTS", "description": "Lorem ipsum dolor sit..."}, {"_id": "5be9bc241c9d440000a730e7", "name": "Franck JS 105", "price": 209900, "description": "Lorem ..."}, {"_id": "5be9c4471c9d440000a730e8", "name": "Kuros TTS", "description": "Lorem ipsum dolor sit ame..."}, {"_id": "5be9c4c71c9d440000a730e9", "name": "Katatone", "description": "Lorem ipsum dolor sit amet..."}]
```

- ❖ Récupération d'un produit par l'ID :

- On appelle la méthode `get()` en lui passant l'ID du produit, de la manière suivante : `Product.get("5be1ed3f1c9d44000030b061")`
- Le résultat devrait être équivalent à l'image :

```
▼ {lenses: Array(2), _id: "5be1ed3f1c9d44000030b061", name: "Zurss 50S", price: 49900, description: "Lorem ipsum dolo..."}
  description: "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labor..."
  imageUrl: "http://localhost:3000/images/vcam_1.jpg"
  lenses: (2) ["35mm 1.4", "50mm 1.6"]
  name: "Zurss 50S"
  price: 49900
  _id: "5be1ed3f1c9d44000030b061"
  __proto__: Object
```

- ❖ Génération de la Card du produit en HTML

- On appelle la méthode `show()` en lui passant en paramètre un "Product" de type Object, exemple :

```
Product.show({
  id: '5be1ed3f1c9d44000030b061',
  description: 'Lorem ipsum dolor sit amet',
  price: 49900,
  imageUrl: 'http://localhost:3000/images/vcam_1.jpg'
})
```

- `Show()` renvoie une string, comme dans l'image ci-dessous

```
<div class="col-md-4">
  <div class="card-product card border-primary mb-3" style="max-width: 20rem;" data-product_id="5be1ed3f1c9d44000030b061" data-product_price="49900">
    <div class="card-header"><a href="product.html?id=5be1ed3f1c9d44000030b061">Zurss 50S</a></div>
    <div class="card-body" style="background-image: linear-gradient(to top right, transparent 48%, #444 48%, #444 52%, #fff 52%); background-size: cover; background-position: center;>
      <p class="text-white">Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
      <h4 class="float-right text-white">49900 €</h4>
    </div>
    <button type="button" class="btn btn-primary">Ajouter au panier</button>
  </div>
</div>
```

Class Cart.js :

❖ Récupération de tous les éléments dans le panier

- On appelle la méthode `all()`, comme suit: `Cart.all()`
- Le résultat devrait ressembler à cela:

```
▶ {id: "5be1ed3f1c9d44000030b061", count: 3, price: "49900"}  
▶ {id: "5be1ef211c9d44000030b062", count: 2, price: "309900"}
```

❖ Ajout d'un produit dans le panier

- Utilisation de la méthode `add()`, comme suit:

```
Cart.add({  
  id: product_id,  
  count: 1,  
  price: product.price  
})
```

- Si le produit est inexistant dans le panier alors il s'ajoute dans la liste, sinon la valeur s'incrémente de 1
- Le résultat devrait être proche de ça :

```
{id: "5be1ef211c9d44000030b062", count: 2, price: "309900"}
```

❖ Compter le nombre de même produit ajouté dans le panier

- On appelle la méthode `count()`, en lui passant l'identifiant du produit, de la manière suivante : `Cart.count("5be1ed3f1c9d44000030b061")`
- Produit dans le panier :

```
▼ 0: {id: "5be1ed3f1c9d44000030b061", count: 3, price: "49900"}  
  count: 3  
  id: "5be1ed3f1c9d44000030b061"  
  price: "49900"
```

- Résultat retourné par la méthode :

```
3
```

❖ Incrémenter un produit d'un produit, si il a le même ID

- La méthode `increment Count()`, doit obligatoirement avoir l'ID du produit en paramètre, et peut facultativement avoir un nombre en 2ème paramètre
- Si on appelle la méthode de la manière suivante : `IncrementCount("5be1ed3f1c9d44000030b061")`, elle ajoutera automatiquement + 1 au produit.

Celle-ci fera passer le produit dans le panier à 4 éléments

```
▼ 0: {id: "5be1ed3f1c9d44000030b061", count: 4, price: "49900"}  
  count: 4  
  id: "5be1ed3f1c9d44000030b061"  
  price: "49900"
```

- Si on lui passe un nombre, celui-ci changera le résultat de "count" contenu dans l'objet du produit dans le panier.

```
▼ 0: {id: "5be1ed3f1c9d44000030b061", count: 34, price: "49900"}  
  count: 34  
  id: "5be1ed3f1c9d44000030b061"  
  price: "49900"
```

-

❖ Retirer des produits dans le panier

- Utilisation de la méthode `decrementCount("5be1ed3f1c9d44000030b061")`
- même fonctionnement que `incrementCount()` mais à l'inverse au lieu d'ajouter, cela retire.

❖ Supprimer un article du panier

- On appelle la méthode `remove("5be1ed3f1c9d44000030b061")`
- L'ensemble de produit ayant cet ID sera supprimé du panier

❖ Récupérer un produit du panier

- On utilise la méthode `get("5be1ed3f1c9d44000030b061")`
- la méthode retourne :

```
▼ {id: "5be1ed3f1c9d44000030b061", count: 2, price: 49900}   
  count: 2  
  id: "5be1ed3f1c9d44000030b061"  
  price: 49900  
  ▶ __proto__: Object
```

❖ Récupérer le nombre total de produit dans le panier

- On appelle la méthode `numberProduct()`
- La méthode nous retournera alors le nombre de produits dans le panier en additionnant le `count` de chaque produit.

❖ Récupérer le montant total de l'ensemble des produits du paniers cumulé

- On appelle la méthode `getTotalPrice()`
- La méthode retourne le montant totale de du panier

❖ Savoir si un article est déjà dans le panier

- On appelle la méthode `has("5be1ed3f1c9d44000030b061")`
- Si ce produit est déjà dans le panier, alors la méthode retourne "true" sinon elle retournera "false" si aucun produit comportant l'id n'est trouvé

- ❖ Réinitialiser le panier
 - On appelle la méthode `reset()`
 - Renvoie un tableau vide dans le localStorage à la clé "panier" afin de supprimer l'ensemble des éléments du panier
- ❖ Afficher un produit dans le panier
 - On appelle la méthode `show("5be1ed3f1c9d44000030b061")` de manière asynchrone
 - Celle-ci nous renvoie le HTML suivant :

```

<div class="cart-item_infos">
  <p class="cart-item_price">49900 €</p>
  <a class="cart-item_name" href="product.html?id=5be1ed3f1c9d44000030b061">Zurss 50S</a>
  <label class="mt-2" data-children-count="1">
    Qté:
    <input type="number" min="1" value="2">
  </label>
</div>
<span class="btn-delete-item" data-remove-product="">
  <svg width="1rem" height="1rem" viewBox="0 0 16 16" class="bi bi-x-circle" fill="currentColor" xmlns="http://www.w3.org/2000/svg">
    <path fill-rule="evenodd" d="M8 15A7 7 0 1 0 8 1a7 7 0 0 0 14 0 1 0 0 0 0 16z"/>
    <path fill-rule="evenodd" d="M4.646 4.646a.5.5 0 0 1 .708 0L8 7.293L12.646 4.646a.5.5 0 0 1 .708 0L15.293 8.646a.5.5 0 0 1 0 1 0 0 0 0 16z"/>
  </svg>
</span>
```

Class ValidationForm.js :

- ❖ Vérifier si un email est valide
 - On appelle la méthode `emailsValid("alex@test.fr")`
 - Si l'email passé en paramètre correspond au critère d'un email la fonction renverra "true", sinon "false"
- ❖ Vérifier si le champ est vide ou null
 - On appelle la méthode `fieldsInvalid(NodeElement)`
 - Si le champs est vide ou null alors la méthode renverra "true" et dans le cas contraire "false"
- ❖ Rendre un champ invalide
 - On appelle la méthode `renderInvalidField(NodeElement, "Message d'erreur à afficher à l'utilisateur"(optionnel))`
 - La méthode changera l'input comme dans les exemples suivant :
 - Avant vérification

```
<div class="form-group" data-children-count="1">
  <label class="form-control-label" for="order_lastname">Nom :*</label>
  <input type="text" id="order_lastname" class="form-control">
</div>
```

- Après vérification

```
<div class="form-group" data-children-count="1">
  <label class="form-control-label" for="order_lastname">Nom :*</label>
  <input type="text" id="order_lastname" class="form-control is-invalid">
  <div class="invalid-feedback">Le champs est invalide</div>
</div>
```

- ❖ Rendre un champs valide

- On appelle la méthode `renderValidField(NodeElement)`
- la méthode supprimera la class `"is-invalid"` et la div avec la class `"invalid-feedback"` si l'élément était invalide avant et y ajoutera la class `"is-valid"` dans l'input

Test d'affichage:

- ❖ Affichage des 5 articles de la page "Boutique"

- Se rendre sur la page "Boutique"
- Vérifier si les articles suivants sont affiché :
 - Zurss 50S
 - Hirsch 400DTS
 - Franck JS 105
 - Kuros TTS
 - Katatone

- ❖ Ajout d'un article dans le panier

- Si je clique 1x sur le bouton "Ajouter au panier" de l'article Zurss 50S dans la page "Boutique"
- Je doit voir passer le compteur de produit dans le panier, situé à droite du menu au nombre de 1

- ❖ Voir la page produit de l'article "Zurss 50S"

- Lorsque je clique sur "Zurss 50S", je doit être redirigé à l'url suivant :
<http://localhost/P5/Orinoco/product.html?id=5be1ed3f1c9d44000030b061>

- Je dois voir sur la page les éléments suivants :



ZURSS 50S

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.


35mm 1.4 50mm 1.6

AJOUTER AU PANIER

❖ Voir le panier

- Cliquez sur le bouton “Mon panier” situé à droite du menu
- vous devez être sur une page avec l'url suivante :
<http://localhost/P5/Orinoco/cart.html>
- la page doit retourner un affichage équivalent

MON PANIER



49900 €

⊗

ZURSS 50S Qté:

TOTAL

SUB-TOTAL49900 €

COMMANDÉ