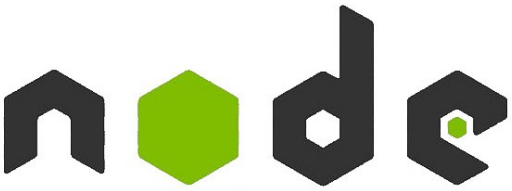
Rapport Explicatif

Lien du projet sur GitHub

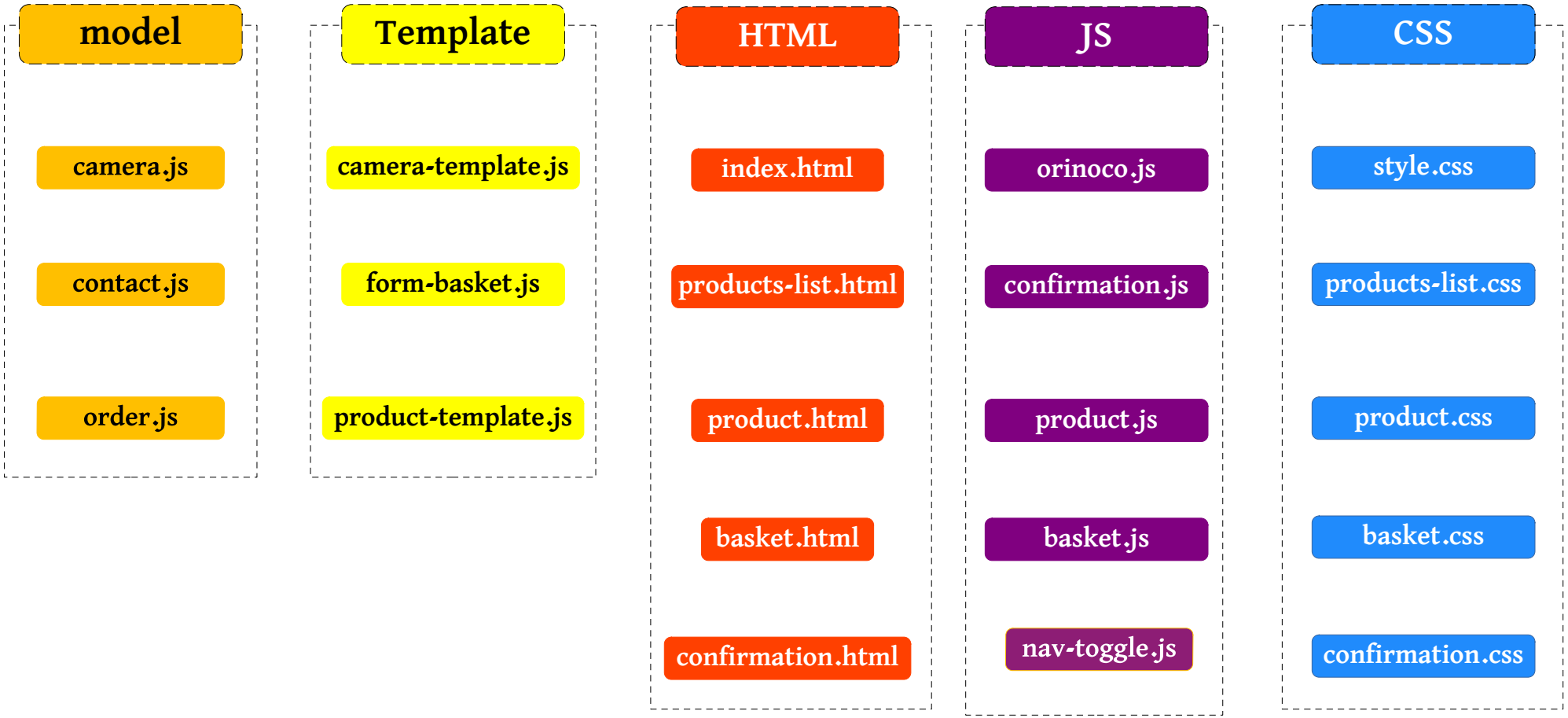
Appuyez ici

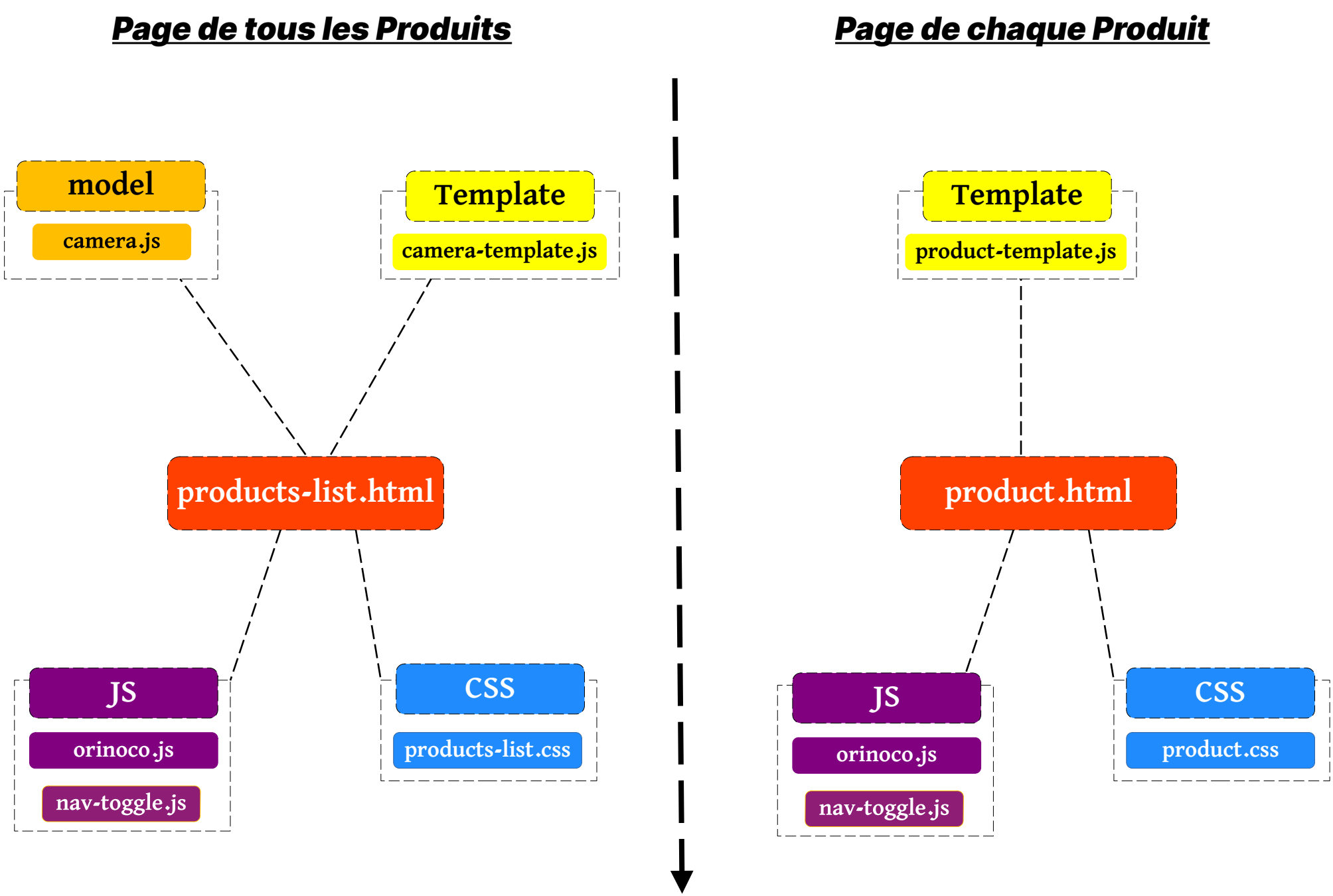


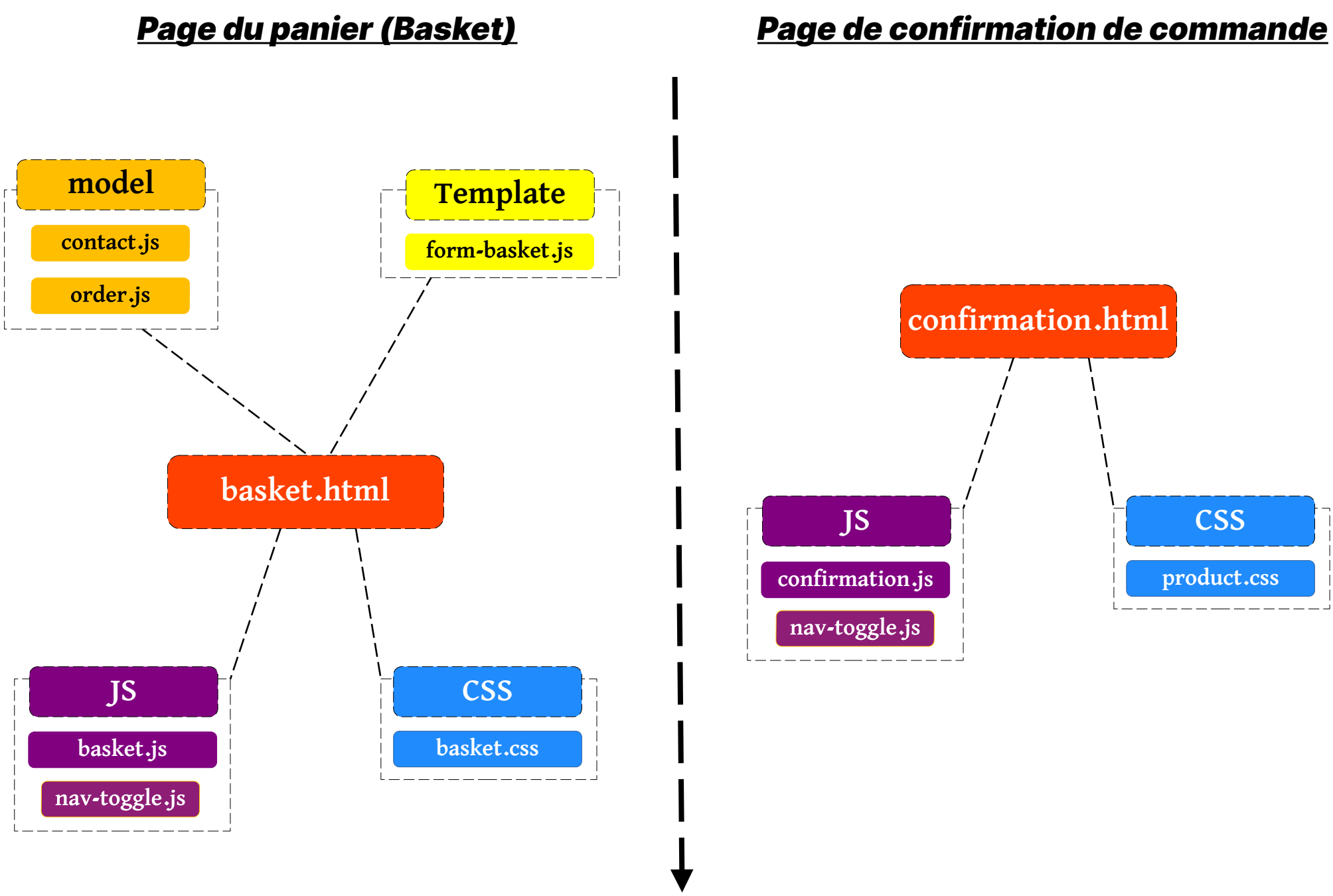
Orinoco

Préparé par
M. EL-KHABOU. E

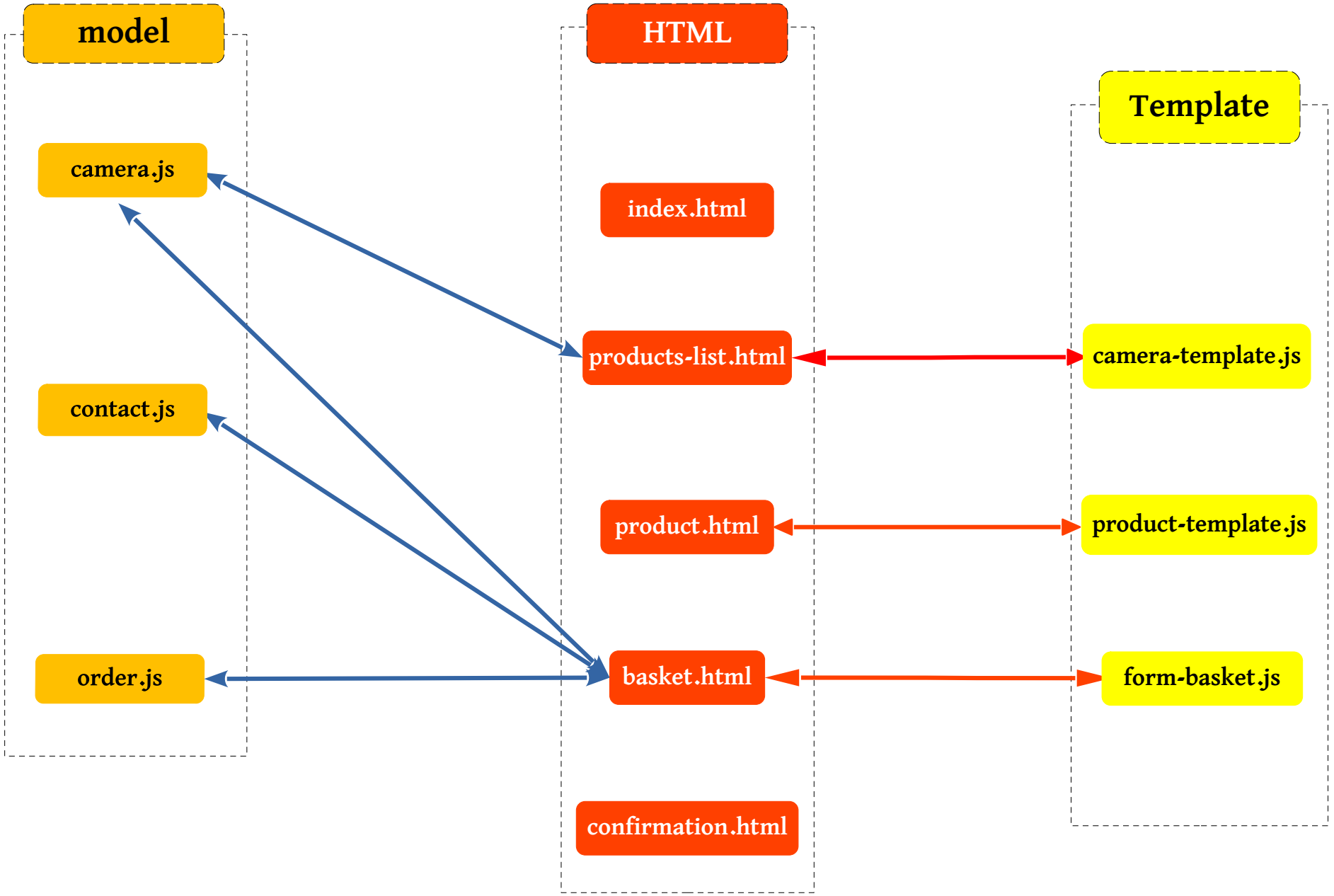
Comprendre la répartition de mes fichiers dans le Front-End




Page de tous les Produits
Page de chaque Produit
model
camera.js
Template
camera-template.js
products-list.html
JS
orinoco.js
nav-toggle.js
CSS
products-list.css
Template
product-template.js
product.html
JS
orinoco.js
nav-toggle.js
CSS
product.css


Page du panier (Basket)
Page de confirmation de commande
model
contact.js
order.js
Template
form-basket.js
basket.html
JS
basket.js
nav-toggle.js
CSS
basket.css
confirmation.html
JS
confirmation.js
nav-toggle.js
CSS
product.css

Les relations entre HTML, Les fichier model et les Templates sans le JS



Model : camera.js

Nous avons créé une classe que nous appelons la classe « Camera » et qui permet de créer un objet (camera):

Quand nous récupérons la liste des Cameras de l'API (service), nous utiliserons cette classe ou cet objet créé pour qu'il s'affiche sur la page y effarante : par exemple, nous afficherons cet objet sur la page **product-list.html**.

Et, pour construire cet objet camera, on passe le produit comme paramètre au constructeur :

```
class Camera{
        constructor(product) {
                this.id = product._id;
                this.name = product.name;
                this.price =`${product.price / 100}.00 €`;
                this.description = product.description;
                this.imageUrl = product.imageUrl;
        }
}
```

Model : order.js (commande)

Nous créons aussi La **classe « Order »** permet de créer l'objet order (commande)

Cet objet order doit contenir les données de contact du client, ainsi que Les produits commandés existants dans le LocalStorage :

```
class Order{
        constructor(contact, products) {
                this.contact = contact;
                this.products = products;
        }
}
```

Model : contact.js

Nous créons aussi La classe « Contact » pour nous permettre de créer l'objet (contact) :

Cet objet contact contient les données du client qui passe la commande sur le site, et qui doit contenir les informations ou les éléments suivants :

- Le prénom = (firstName)
- Le nom = (lastName)
- L'adresse = (address)
- Le code postale = (postCode)
- La ville = (city)
- L'email = (email)
- Les commentaires = (commentary)

Pour créer cet objet contact, nous passons ces paramètres au constructeur :

```
class Contact{
        constructor(firstName, lastName, address, city, postCode, tel, email, commentary) {
                this.firstName = firstName;
                this.lastName = lastName;
                this.address = address;
                this.postCode = postCode;
                this.city = city;
                this.tel = tel;
                this.email = email;
                this.commentary = commentary;
        }
}
```

Template : camera-template.js

La fonction **buildCamera** permet d'afficher une camera dans un card bootstrap.

Quand nous récupérons les produits de l'API, on affiche les produits sur la page d'affichage de produits (products-list.html) Grace à cette fonction **buildCamera**.

Nous appelons la fonction buildCamera qui prend en charge l'objet camera pour afficher chaque produit sur la page products-list.html :

```
        function buildCamera(camera){
                return `
                <div class="p-2">
                <a href="../pages/product.html?id=${camera.id}">
                        <div class="card" style="width: 18rem;">
                                <img id="cardImg" src="${camera.imageUrl}" alt="Card image cap">
                                <div class="card-body">
                                        <h5 id="cardTitle">${camera.name}</h5>
                                                <p id="cardDescription">${camera.description}</p>
                                                <p id="cardPrice" class="price">${camera.price}</p>
                                </div>
                        </div>
                </a>
                </div>`;
        }
```

Template : product-template.js

Nous créons aussi la Structure du produit dans product.html :

```
function buildCameraProduct(camera){
        return `
        <div class="infoProduct">
                <div class="product">
                        <img id="cardImgProduct" src="${camera.imageUrl}" alt="Card image">
                        <div class="cardBodyProduct">
                                <h5 id="cardTitleProduct">${camera.name}</h5>
                                        <p id="cardDescriptionProduct">${camera.description}</p>
                                        <p id="cardPriceProduct">${camera.price / 100}.00 €</p>
                                                <div class="choiceLenses" role="search" aria-label="Choix du produit">
                                                        <label class="font-weight-bold" tabindex="0">Choisir un optique:</label>
                                                                <select id="cameraLenses" class="lenses-product"></select>
                                                </div>
                                                <div id="quantity">
                                                        <span id="plus" onclick="addQuantity(event)"><i class="fas fa-plus-circle"></i></span>
                                                                <input type="number" id="plusOrMinusClick" value="1"min="1"/>
                                                        <span id="minus" onclick="minusQuantity(event)"><i class="fas fa-minus-circle"></i></span>
                                                </div>
                        </div>
                </div>
        </div>`;
        }

//Fonction pour le tableau "lenses" dans une balise <select>
function buildCameraLensesSelectList (lenses) {
        const optionList = document.querySelector("#cameraLenses").options;

        lenses.forEach(option =>
        optionList.add(
                new Option(option, option)
        )
        );
}
```

```
//-----------------------------Quantité-----------------------------//
function addQuantity (event){
        event.preventDefault();
        const quantityClicks = document.getElementById("plusOrMinusClick");
        quantityClicks.valueAsNumber++;
}

function minusQuantity (event){
        event.preventDefault();
        const quantityClicks = document.getElementById("plusOrMinusClick");

        if (quantityClicks.valueAsNumber == 1){
                return;
        }
        document
                .getElementById("plusOrMinusClick")
                .value = quantityClicks.valueAsNumber -1;
}
```

Template : form-basket.js

Nous construisons ce template afin de mettre en place l'architecture du formulaire que le client doit remplir avec ses informations personnelles avant qu'il passe la commande.

Ce formulaire se chargera aussi de la validité des données entrées par le client :

```
const form = () => {
        const positionForm = document.getElementById("form");
        const structureFormulaire = `
        <form name="orderForm" class="form-order" onsubmit="handleSubmit(event)" role="search" aria-label="Formulaire">
                <h4 class="formTitle">Merci de complèter ce formulaire pour valider la commande :</h4>
                        <div class="input-form">
                                <div class="name">
                                        <input type="text" id="lastName" name="lastName" class="contact" pattern="[A-Z][A-Za-z' -]+" title="Le format requis : lettres uniquement" placeholder="Nom" aria-labelledby="Nom" required>
                                        <input type="text" id="firstName" name="firstName" class="contact" pattern="[A-Z][A-Za-z' -]+" title="Le format requis : lettres uniquement" placeholder="Prénom" aria-labelledby="Prénom" required>
                                </div>
                                <div class="address">
                                        <input type="text" id="adress" name="adress" class="contact-address" minlenght="5" maxlenght="40" title="Le format requis : lettres et chiffres uniquement" placeholder="Adresse" aria-labelledby="Adresse" required>
                                </div>
                                <div class="cp">
                                        <input type="text" id="postCode" name="postCode" class="contact" pattern="([A-Z]+[A-Z]?\-)?[0-9]{1,2} ?[0-9]{3}" minlenght="5" title="Le format requis : 5 chiffres minimum uniquement" placeholder="Code Postale" aria-
                                        labelledby="Code Postale" required>
                                        <input type="text" id="city" name="city" class="contact" pattern="[A-Z][A-Za-z' -]+" title="Le format requis : lettres uniquement" placeholder="Ville" aria-labelledby="Ville" required>
                                </div>
                                <div class="tel-mail">
                                        <input type="tel" id="tel" name="tel" class="contact-tel" pattern="0[1-9][0-9]{8}" max-lenght="10" title="Le format requis : 10 chiffres maximum uniquement" placeholder="Téléphone" aria-labelledby="Téléphone" required>
                                        <input type="email" id="email" name="email" class="contact-mail" pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,3}$" title="Le format requis : exemple@mail.com" placeholder="E-mail" required>
                                </div>
                                <div class="commentary">
                                        <textarea type="text" id="commentary" name="commentary" class="commentary" maxlength="200" title="Informations complémentaires" placeholder="Informations complémentaires à la livraison" aria-labelledby="Informations
                                        complementaires"></textarea>
                                </div>
                        </div>
                <!--Bouton validé la commande-->
                <div class="btn-basket" role="button" aria-label="Bouton" tabindex="0">
                        <button type="submit" id="validOrder" class="button">Valider la commande</button>
                </div>
        </form>
        `;

        positionForm.insertAdjacentHTML("afterend", structureFormulaire);
        };
        form();
```

nav-toggle.js

Une fonction presque existante sur toutes les pages du site, puisqu'elle contrôle le menu de navigation au clic de ce dernier:

une nav-bar du menu qui contient les liens de navigation et qui, lorsque l'écran a une largeur maximale de 667 pixels, les liens de navigation se transforment en menu à outils dans une icône (bootstrap) :

```
function menuToggle() {
	document.getElementById("myDropdown").classList.toggle("show");
}

window.onclick = function(event) {
	if (!event.target.matches('.dropbtn')) {

		var dropdowns = document.getElementsByClassName("dropdown-content");
		var i;
		for (i = 0; i < dropdowns.length; i++) {
			var openDropdown = dropdowns[i];
			if (openDropdown.classList.contains('show')) {
				openDropdown.classList.remove('show');
			}
		}
	}
}
```