

Jour 2

JS



JS



Modules



Qu'est-ce qu'un module ?

Un module est un fichier qui contient du code JavaScript.

- 📌 Un fichier JavaScript peut contenir plusieurs modules.
- 📌 Un module peut contenir plusieurs classes.
- 📌 Un module peut contenir plusieurs fonctions.
- 📌 Les modules sont utilisés pour organiser le code.



Déclaration d'un module

Pour déclarer un module, on utilise le mot clé `export` devant le nom de la classe.

```
export class Personnage {  
  constructor(nom, sante, force) {  
    this.nom = nom;  
    this.sante = sante;  
    this.force = force;  
  }  
  
  decrire() {  
    return `${this.nom} a ${this.sante} points de vie et ${this.force} en force`;  
  }  
}
```



Import d'un module

Pour importer un module, on utilise le mot clé `import` suivi du nom de la classe.



Import simple:

```
import { Personnage } from "../personnage.js";  
  
const aventurier = new Personnage("Aurora", 150, 25);  
console.log(aventurier.decrire());
```



Import multiple:

```
import { Personnage, Aventurier } from "./personnage.js";  
  
const aventurier = new Aventurier("Aurora", 150, 25, 1000);  
console.log(aventurier.decrire());
```



Import avec alias:

```
import { Personnage as Perso } from "./personnage.js";  
  
const aventurier = new Perso("Aurora", 150, 25);  
console.log(aventurier.decrire());
```




Import de tout:

```
import * as Personnage from "./personnage.js";  
  
const aventurier = new Personnage.Aventurier("Aurora", 150, 25, 1000);  
console.log(aventurier.decrire());
```



TP

Modulariser le code du TP précédent en utilisant les modules



Surcharge et mot-clé (arguments)



Surcharge

- 📌 La surcharge est une fonctionnalité qui permet de définir plusieurs fonctions avec le même nom mais avec des paramètres différents
- 📌 La surcharge n'est pas disponible en JavaScript
- 📌 En JavaScript, on peut simuler la surcharge en utilisant les arguments



Arguments

- 📌 Le mot clé `arguments` permet de récupérer les paramètres d'une fonction
- 📌 `arguments` est un objet, semblable à un tableau
- 📌 `arguments` possède une propriété `length` et ses propriétés sont indexées à partir de 0 mais il ne possède aucune des méthodes natives de Array.



Example

```
function addition() {  
  let result = 0;  
  for (let i = 0; i < arguments.length; i++) {  
    result += arguments[i];  
  }  
  return result;  
}  
  
console.log(addition(1, 2, 3, 4, 5)); // 15
```



Rest parameters

- 📌 Les rest parameters permettent de récupérer les paramètres d'une fonction sous forme de tableau
- 📌 Les rest parameters sont définis en utilisant les 3 points `...`



Example

```
function addition(...numbers) {  
  let result = 0;  
  for (let i = 0; i < numbers.length; i++) {  
    result += numbers[i];  
  }  
  return result;  
}  
  
console.log(addition(1, 2, 3, 4, 5)); // 15
```


Exemple de surcharge



```
function addition(...Args) {  
    switch (Args.length) {  
        // if no argument  
        case 0:  
            console.log('No argument is passed.');
```


 // if only one argument
 case 1:
 console.log('You have to pass at least two arguments to perform addition.');
 // multiple arguments
 default:
 let sum = 0;
 let length = Args.length;

 for (var i = 0; i < length; i++) {
 sum = sum + Args[i];
 }
 return sum;
 }
}

console.log(addition(1, 2, 3, 4, 5)); // 15



Closures

- 📌 Une fonction de closure est une fonction qui est définie à l'intérieur d'une autre fonction.
- 📌 Une fonction de closure a accès aux variables de la fonction parente.



Example

```
function makeAdder(x) {  
  return function(y) {  
    return x + y;  
  };  
}  
  
var add5 = makeAdder(5);  
var add10 = makeAdder(10);  
  
console.log(add5(2)); // 7  
console.log(add10(2)); // 12
```

Exemple

JS



```
<p id="aide">Des aides seront affichées ici</p>
<p>E-mail : <input type="text" id="email" name="email"></p>
<p>Nom : <input type="text" id="nom" name="nom"></p>
<p>Âge : <input type="text" id="âge" name="âge"></p>

<script>
function afficheAide(aide) {
    document.getElementById('aide').innerHTML = aide;
}

function creerCallbackAide(aide) {
    return function() {
        afficheAide(aide);
    };
}

function prepareAide() {
    var texteAide = [
        {'id': 'email', 'aide': 'Votre adresse e-mail'},
        {'id': 'nom', 'aide': 'Votre prénom et nom'},
        {'id': 'âge', 'aide': 'Your age (you must be over 16)'}
    ];

    for (var i = 0; i < texteAide.length; i++) {
        var item = texteAide[i];
        document.getElementById(item.id).onfocus = creerCallbackAide(item.aide);
    }
}

prepareAide();
</script>
```



Impacts des "closures" sur la lisibilité

- 📌 Les closures permettent de rendre le code plus lisible
- 📌 Les closures permettent de limiter la portée des variables
- 📌 Les closures permettent de ne pas repeter du code
- 📌 Les closures sont un investissement pour le code



- 📌 Créer des fonctions qui permettent d'effectuer plusieurs attaques placer en parametre (nombre de parametre variable) afin de pouvoir appliquer un nombre d'attaque variable avec une seule fonction
- 📌 Créer un comportement different si un seul parametre est passé et qu'il s'agit d'un array afin de pouvoir executer une queue d'attaque (ex: attaquer 3 fois avec une attaque de 10 points de degats, 5 fois avec une attaque de 5 points de degats, 2 fois avec une attaque de 20 points de degats)
- 📌 Integrer un systeme de dommage sur le temps (ex: empoisonnement, brulure, sort de guerison sur le temps, etc...)



Structuration et qualité du code



Séparation en multiple fichiers

- 📌 Le code JavaScript peut être structuré en plusieurs fichiers
- 📌 Il est important de séparer le code en plusieurs fichiers afin de faciliter la maintenance et la compréhension du code



Asynchronous Module Definition (AMD)

- 📌 AMD est un format de module qui permet de charger les modules de manière asynchrone
- 📌 AMD est utilisé par RequireJS



AMD avec Require.js

- 📌 **RequireJS est une bibliothèque JavaScript qui permet de charger les modules de manière asynchrone**
- 📌 **RequireJS est basé sur le format de module AMD**



- 📌 Il permet de charger les modules de manière asynchrone via le mot clé `require`
- 📌 Il permet de définir des modules via le mot clé `define`
- 📌 Pour exporter quelque chose d'un module, il faut le retourner via le mot clé `return`



- 📌 Pour importer quelque chose d'un module, il faut utiliser le mot clé `require`
- 📌 `require` prend en paramètre un tableau de dépendances et une fonction de callback
- 📌 `define` prend en paramètre un tableau de dépendances et une fonction de callback qui prend en paramètre les dépendances

Qualité avec JSHint et JSLint





JSLint

- 📌 **JSLint est un outil qui permet de vérifier la qualité du code JavaScript**
- 📌 **JSLint est basé sur les règles de Douglas Crockford**
- 📌 **JSLint est plus strict que JSHint**




JSHint

- 📌 JSHint est un outil qui permet de vérifier la qualité du code JavaScript
- 📌 JSHint est basé sur les règles de JSLint et a été créé suite à des divergences de vision entre les développeurs de JSLint et ceux de JSHint
- 📌 JSHint est plus flexible que JSLint



TP

 **Corriger les erreurs de syntaxe et de qualité du code de votre jeu grace a JSHint**



L'héritage jQuery





Bases de jQuery

- 📌 jQuery est une bibliothèque JavaScript qui permet de simplifier l'écriture du code JavaScript
- 📌 jQuery est basé sur le sélecteur CSS
- 📌 jQuery est basé sur le modèle événementiel



Comment utiliser jQuery

 `$(document).ready(callback)` pour attendre que le DOM soit chargé

 `$(selector).click(callback)` pour sélectionner un élément et effectuer une action lors du click



Exploitation des sélecteurs en jQuery

jQuery permet d'utiliser les sélecteurs CSS:

 `$('p')` sélectionne tous les éléments `<p>`

 `$('.intro')` sélectionne tous les éléments avec la classe `intro`

 `$('#firstname')` sélectionne l'élément avec l'identifiant `firstname`

 `$('p.intro')` sélectionne tous les éléments `<p>` avec la classe `intro`




Manipulation du DOM avec jQuery

📌 `$(selector).html()` retourne le contenu HTML de l'élément sélectionné


📌 `$(selector).html(content)` définit le contenu HTML de l'élément sélectionné




 `$(selector).text()` **retourne le contenu textuel de l'élément sélectionné**

 `$(selector).text(content)` **définit le contenu textuel de l'élément sélectionné**







 `$(selector).attr(attribute)` **retourne la valeur de l'attribut de l'élément sélectionné**

 `$(selector).attr(attribute, value)` **définit la valeur de l'attribut de l'élément sélectionné**




Quelques méthodes jQuery

-  `$(selector).hide()` **cache les éléments sélectionnés**
-  `$(selector).show()` **affiche les éléments sélectionnés**
-  `$(selector).filter(selector)` **sélectionne les éléments qui correspondent au sélecteur**
-  `$(selector).each(callback)` **exécute une fonction pour chaque élément sélectionné**



JQuery et les événements


-  `$(selector).click(callback)` exécute une fonction lors du **click sur l'élément sélectionné**
-  `$(selector).dblclick(callback)` exécute une fonction lors du **double click sur l'élément sélectionné**
-  `$(selector).hover(callback)` exécute une fonction lors du **survol de l'élément sélectionné**
-  `$(selector).focus(callback)` exécute une fonction lors de la **prise de focus sur l'élément sélectionné**





jQuery et les animations


- 📌 `$(selector).animate({params}, speed, callback)` **exécute une animation sur les éléments sélectionnés**
- 📌 `$(selector).slideDown(speed, callback)` **affiche les éléments sélectionnés avec une animation**
- 📌 `$(selector).slideUp(speed, callback)` **cache les éléments sélectionnés avec une animation**



 `$(selector).slideToggle(speed, callback)` **affiche ou cache les éléments sélectionnés avec une animation**

 `$(selector).fadeIn(speed, callback)` **affiche les éléments sélectionnés avec une animation**

 `$(selector).fadeOut(speed, callback)` **cache les éléments sélectionnés avec une animation**

 `$(selector).fadeToggle(speed, callback)` **affiche ou cache les éléments sélectionnés avec une animation**



- 📌 `$(selector).hide(speed, callback)` **cache les éléments sélectionnés**
- 📌 `$(selector).show(speed, callback)` **affiche les éléments sélectionnés**
- 📌 `$(selector).toggle(speed, callback)` **affiche ou cache les éléments sélectionnés**



JQuery et les effets

-  `$(selector).stop(stopAll, goToEnd)` **arrête les effets en cours sur les éléments sélectionnés**
-  `$(selector).delay(time)` **ajoute un délai sur les effets en cours sur les éléments sélectionnés**



Les composants graphiques de jQuery

jQuery UI est une bibliothèque qui permet d'ajouter des composants graphiques à une page web:

- 📌 `$(selector).accordion()` ajoute un accordéon sur les éléments sélectionnés
- 📌 `$(selector).autocomplete()` ajoute un champ de saisie avec auto-complétion sur les éléments sélectionnés
- 📌 `$(selector).button()` ajoute un bouton sur les éléments sélectionnés



-  `$(selector).datepicker()` ajoute un champ de saisie avec un calendrier sur les éléments sélectionnés
-  `$(selector).dialog()` ajoute une boîte de dialogue sur les éléments sélectionnés
-  `$(selector).menu()` ajoute un menu sur les éléments sélectionnés
-  `$(selector).progressbar()` ajoute une barre de progression sur les éléments sélectionnés
-  `$(selector).slider()` ajoute un curseur sur les éléments sélectionnés



Intérêts de jQuery par rapport JavaScript

- 📌 jQuery permet d'écrire moins de code
- 📌 jQuery permet de simplifier l'écriture du code JavaScript
- 📌 JQuery permet de simplifier la manipulation du DOM



- 📌 jQuery permet de simplifier la gestion des événements
- 📌 jQuery permet de simplifier la gestion des animations
- 📌 Jquery permet de simplifier la gestion des effets
- 📌 JQuery permet de ne pas réinventer la roue (ex: composants graphiques)



TP

- 📌 Ajouter des animation sur les éléments de la page web de notre jeu
- 📌 Ajouter des effets sur les personnages lors de l'attaque
- 📌 Ajouter des composants graphiques sur la page web de notre jeu (progress bar pour les points de vie, menu pour les actions, etc...)

Tout cela dans le but de rendre notre jeu plus agréable à jouer.