JAVASCRIPT

JSON

JSON

- Le format JSON (pour JavaScript Object Notation) est le format le plus utilisé dans le Web.
- format d'échange pour les données dans des applications Web
- n'est pas utilisé qu'en JavaScript
- un ensemble de règles pour utiliser cette notation

JSON

- JSON est utilisé :
- En tant que format d'échange entre un client (typiquement un navigateur) et un serveur (backend),
- Non pas en tant que format de représentation de données
- Par google, yahoo (services Web)
- Dans le domaines des smartphones du fait de sa légéreté...

JSON

- Echange entre navigateur et serveur = texte
- Convertir objet javascript en JSON et envoyer au serveur
- Convertir JSON en javascript
  - Exemple1
- Recevoir format JSON et convertir en objet javascript
  - Exemple2
- JSON permet de sauvegarder des objets JavaScript en tant que texte.

Syntaxe

- Structure de base : paire clef-valeur (key-value)
  - "nom": "abrouk"
- Valeurs JSON
  - String
  - Number
  - object (JSON object)
  - Array
  - boolean
  - null

Syntaxe

- Un objet est un ensemble de paires clef-valeur, une clef apparait au plus une fois
  - { "prenom": "lylia", "nom": "abrouk"}
- Un objet peut être utilisé comme valeur (complexe) dans une paire clef-valeur.
  - "personne": { "prenom": "lylia", "nom": "abrouk"}
- Un tableau (array) est une liste de valeurs (dont le type n'est pas forcément le même).
- {
  "etudiants":["José", "Anna", "Patrick"]
  }

JSON / XML

- JSON plus léger et intuitif que XML,
- Facile à parser pour n'importe quel langage de programmation,
- JSON n'a pas de langage de spécification de schéma associé
- JSON n'a pas (encore) de langage de requête associé.
- Pour les applications AJAX, JSON est plus rapide et plus facile que XML:
- XML
  - Récupérer un document XML
  - Utilisez le DOM XML pour parcourir le document
  - Extraire des valeurs et stocker dans des variables
- Utiliser JSON
  - Récupèrer une chaîne JSON
  - JSON.Parse la *String* JSON
- Exemple

Parser

- Échanger des données vers/depuis un serveur Web.
- Réception de données : Type String
- Parser les données : JSON.parse (), les données deviennent un objet JavaScript.

JSON.stringify()

- Lors de l'envoi de données à un serveur Web, les données doivent être une String.
- Convertir un objet JavaScript en String avec JSON.stringify ().
- Exemple 4
- Exemple 5 : Date
- La fonction JSON.stringify () supprime toutes les fonctions d'un objet JavaScript, à la fois la clé et la valeur
  - Convertir la fonction en String avant l’appel à la fonction toString().

Les évènements

Introduction aux évènements

- Permettent d’avoir des pages réactives qui réagissent aux actions de l’utilisateur
- clic de la souris, touche pressée, passage de la souris, fin d'un chargement, lancement d'une vidéo, fin d'une piste audio, défilement, redimensionnement de la fenêtre, envoi d'un formulaire
- Tous les noeuds du DOM émettent des événements.

- héritent de l'interface Event
- Les événements sont émis mais encore faut-il les écouter pour y réagir
- Pour écouter un événement il faut ajouter un écouteur d'événement sur l'élément qui émet l'événement ciblé.
- passer à cet écouteur, une fonction de rappel appelée gestionnaire d'événement
- Cette fonction sera exécutée à chaque fois que l'événement écouté sera émis par le DOM

Utilisation des propriétés du DOM on*

- Les noeuds ont en effet des propriétés on<nom-événement> qui ont toute par défaut une valeur de null
- passer un gestionnaire d'événement, qui est pour rappel, une fonction de rappel
- Exemple1
- A chaque fois que nous cliquons sur le bouton, la fonction de rappel définie comme valeur de la propriété onclick est exécutée et va afficher un message dans la console.
- Valeur de this
  - pour les fonctions fléchées, le this est lié à l'environnement lexical (là où la fonction est déclarée). La valeur sera donc l'objet global : window.
  - pour les fonctions standard, this est liée au contexte d'exécution
  - Exemple2.html

La méthode addEventListener()

- Limite : il n'est possible de déclarer qu'un seul gestionnaire d'événement pour un type d'événement particulier.
- Méthode : ajouter autant de gestionnaires d'événement que nécessaire
  - La méthode addEventListener()
  - noeud.addEventListener(evenement, gestionnaire, options);
    - Nom de l'évènement
    - Gestionnaire d'évènement : fonction de rappel
    - Options : facultatives : once, capture, passive
  - possibilté de passer un objet comme gestionnaire d'événement
    - L' objet doit obligatoirement avoir la méthode handleEvent()

Supprimer un gestionnaire d'événement et déclencher un événement

- removeEventListener() : permet de supprimer un gestionnaire d'événement sur un noeud.
  - noeud.removeEventListener(evenement, gestionnaire, options);
  - Il faut que le gestionnaire d'événement ait un identifiant
  - Il faut déclarer une fonction afin de pouvoir accéder à l'identifiant
- Constructeur d’évènements spécifiques
  - par exemple un clic à des coordonnées précises

```
const evenment = new MouseEvent("click", {
  bubbles: true,
  clientX: 550,
  clientY: 550
});
```

Le bouillonnement et la capture

- plusieurs noeuds imbriqués , un écouteur d'événement sur chacun d’eux
  - quels écouteurs vont être déclenchés ?
  - dans quel ordre ?
  - Exemple4.html
- le gestionnaire d'événement sur le noeud le plus imbriqué qui est exécuté en premier

Empêcher le comportement par défaut

- L’objet Event :
  - La propriété type permet d'accéder au type de l'événement.
  - La propriété target est la référence à l'élément qui a émis l'événement le plus imbriqué
  - La propriété currentTarget est la référence à l'élément dont le gestionnaire d'événement est en train d'être exécuté. Elle équivaut à la valeur de this.
  - La propriété defaultPrevented permet de savoir si le comportement par défaut a été annulé avec la méthode preventDefault().
  - …
  - https://www.w3schools.com/jsref/obj_event.asp
- Stopper la propagation
  - preventDefault() : prévenir le comportement par défaut
  - evenement.preventDefault()

Gestion des erreurs

Gestion d’erreurs

- Les différentes erreurs et comment les gérer ?
  - une erreur non gérée va entraîner l'arrêt de l'exécution du script.
  - l'erreur sera affichée dans la console du navigateur en rouge

```
try {
  // Les instructions
} catch (erreur) {
  // Gestion de l'erreur
}
```

  - Utiliser les blocs try / catch
  - Si une erreur survient durant l'exécution des instructions dans le try, elle sera récupérée dans le catch.
  - Les instructions dans le bloc try ne sont pas exécutées après une erreur
  - le script ne sera pas interrompu et les instructions après les blocs s'exécuteront.
  - Il n'est pas obligatoire de récupérer l'erreur dans le bloc catch.

```
function isValidJSON(valeur){
  try {
    JSON.parse(valeur);
    return true;
  } catch {
    return false;
  }
}
```

Gestion d’erreurs

- Les blocs try / catch ne fonctionnent que pour les erreurs synchrones.
- Exemple -> Uncaught Error : une erreur qui n'est pas du tout gérée.
  - utiliser async / await
  - mettre les blocs try / catch dans la fonction asynchrone
- Gérer l’erreur dans le bloc catch
  - si une requête échoue, vous pouvez la réessayer
  - Si le serveur retourne une erreur, vous pouvez l'afficher à l'utilisateur

```
try {
  setTimeout(() => {throw "erreur"}, 1000);
} catch (e) {
  console.log( "pas catché " );
}
```

```
setTimeout(() => {
  try {
    throw "erreur"
  } catch (e) {
    console.log( "Catché" );
  }
}, 1000);
```

Gestion d’erreurs

- Bloc finally : permet d'exécuter des instructions de nettoyage ou de fin. Il est toujours exécuté, qu'il y ait une erreur ou non.

```
try {
    // instructions
} catch(e) {
  // gestion de l’erreur
} finally {
  // toujours exécuté, après le try ou le catch
}
```

Gestion d’erreurs
L’objet error natif

- Les types d'erreurs natifs sont :
  - Error : c'est l'objet d'une erreur générique.
- Les autres types d'erreurs héritent du constructeur Error :
  - EvalError : erreur d’évaluation en utilisant la fonction eval()
  - RangeError : erreur de valisation lorsqu’une valeur n’est pas dans les valeurs admises
  - ReferenceError : on utilise un identifiant (fonction ou variable qui n’existe pas)
  - SyntaxError : erreur de syntaxe lors de l’évaluation du code
  - TypeError : erreur de type lors de l’évaluation d’une variable ou d’un type
  - URIError : erreur lorsque des arguments invalides sont passées à encodeURI() ou decodeURI()
    - Le type d'erreur sera présent sur la propriété name d'un objet erreur.

SyntaxError

- Exemples :
  - Quotes
  - Parenthèses
  - Virgule dans un tableau
  - Nom malformé

ReferenceError

- Exemples :
  - Variables pas déclarées
  - Fonction non définie

TypeError

- Utilisation d’un objet ou une méthode qui n’existe pas
- Erreur casse (ex: Document. , document.Write)
- Elément DOM qui n’existe pas

RangeError

- Entier en dehors du range
- Exemple : déclaration de tableau
- toFixed
- toPrecision

Corriger les erreurs

- Degugger le script : Console
- Gérer les erreurs : try catch

Debugger

- Erreurs de syntaxe : empêche le code de s'exécuter
- Erreurs d’algorithme : ne pose aucun problème d'exécution
  - Empêchent le bon déroulement du programme

- Debuggage erreur syntaxique : La console
- Debuggage erreur d’algorithme : remonter jusqu’à l’erreur dans le script

Debugger

- Erreurs qu’on ne contrôle pas : Données d’un serveur qui ne répond pas

1. Message d’erreur
   - Le script qui cause le problème
   - Numéro de la ligne
   - Le type d’erreur
2. Inclure des message
   1. console.log(c);
   2. Breakpoint (debugger)

Debugger

- L'utilisation de console.log()
  - afficher la valeur de variables lors de l'exécution à l'endroit du code qui pose problème.
- Utilisation de DevTools
- Sur Chrome, pour ouvrir la console Devtools, il faut utiliser :
  - Sur Mac, faites cmd + alt + i.
  - Sur Linux ou Windows, faites ctrl + maj + i
  - Exemple
    - ouvrez la console
    - l'onglet Sources, puis sur Page ou Network

Debugger

- Observer des expressions
- Aller dans watch
- Utiliser la console dans un breakpoint
  - Pour obtenir le bon résultat
- Les breakpoints de modification du DOM