# GOOGLE FIREBASE

## - BASIC -

Nicolas Fazio
@FazioNico

# PROGRAMME

Présentation de Firebase
Le model NoSQL
Installation
Connection au serveur
Real-Time-Database
Organisation des donnée
User Authentification
Gestion des droits et sécurité

# PRÉSENTATION DE FIREBASE

# FIREBASE EN QUELQUES MOTS

Firebase est un outil en ligne qui permet de stocker des données ou des fichiers et d'authentifier les utilisateurs d'une application.

L'idée est de proposer une zone Back-End en Javascript pour le développement d'application et le stockage des informations.

Site officiel: https://firebase.google.com/

# LES BBDD NOSQL

NoSQL ne veut pas dire que ce n'est un model de stockage de donnée SQL, mais que ce n'est pas QUE un model SQL !
**NoSQL = NotOnlySQL**

Ce model de gestion des données est apparu pour palier au désaventage du model de gestion SQL.

Voici en illustration simplifiée ce qui change:

# SQL VS NOSQL

| Item | Price | Quantity | Status |
|---|---|---|---|
| Apples | $1 | 7 | null |
| Banane | $2 | 18 | null |
| Audi | $80'000 | 2 | neuf |
| Fial | $12'000 | 2 | occasion |

# SQL VS NOSQL

```
{
  Item: 'pomme',
  price: '1.-',
  Quantity: '7'
}
```

```
{
  Item: 'banane',
  price: '1.-',
  Quantity: '18'
}
```

```
{
  Item: 'audi',
  price: '80'000.-',
  Quantity: '2',
  status: 'neuf'
}
```

```
{
  Item: 'fiat',
  price: '12'000.-',
  Quantity: '2',
  status: 'occasion'
}
```

# UTILISER FIREBASE AVEC UN PROJET JAVASCRIPT

# INSTALLATION DE FIREBASE ET CONNECTION AU SERVEUR

Aller sur le site officiel:
https://firebase.google.com/
Cliquer sur **connexion** et **se connecter** ou **créer un compte**, puis sur **accéder à la console** et enfin sur **créer un projet**.

Choisir ensuite **ajouter Firebase à votre app web**. Firebase génère tout seul le code d'intégration! **Copier et coller** le code javascript dans votre fichier **index.html**

C'est tout ;-)

# UTILISATION DE FIREBASE

**firebase.database()**
Stockez et synchronisez des données en temps réel sur tous les clients connectés.

**firebase.auth()**
Authentifiez et gérez les utilisateurs à partir de plusieurs fournisseurs sans code côté serveur.

Doc officiel pour WebApp:
https://firebase.google.com/docs/web/setup

# FIREBASE REALTIME DATABASE

# CONNECTION À LA BASE DE DONNÉE

Firebase RealTime Database est une base de données hébergée sur un cloud. Les données sont stockées au format JSON et synchronisé en temps réel à chaque client connecté.

**Pour accéder à votre base de donnée:**
- initialliser Firebase puis:

```
let database = firebase.database();
```

# CONNECTION À LA BASE DE DONNÉE

Voici ce que cela donne:

```
// Set the configuration for your app
// TODO: Replace with your project's config object
let config = {
  apiKey: "apiKey",
  authDomain: "projectId.firebaseapp.com",
  databaseURL: "https://databaseName.firebaseio.com",
  storageBucket: "bucket.appspot.com"
};
firebase.initializeApp(config);

// Get a reference to the database service
let database = firebase.database();
```

Maintenant la base de donnée Firebase est prête à être utilisée.

# TP

- Créer un projet 'firebase-training' sur firebase.google.com
- Créer un fichier index.html et y intégrer le projet Firebase 'firebase-training'
- Se connecter à votre RealTime Database

# ORGANISER ET STRUCTURER LA BASE DE DONNÉE

# ORGANISER ET STRUCTURER LA BASE DE DONNÉE

Concevoir une bonne organisation structurelle de votre base de donnée demande de la reflexion et de savoir dans que contexte les informations vont être utilisées (use case).

Sans quoi vous allez continuellement réorganiser votre structure de data et donc perdre du temps sur le développement de votre application.

# ORGANISATION "FLAT"

Lorsque vous récupérez des données à un emplacement dans votre base de données, **vous récupérez également tous ses nœuds enfants**.

Et quand vous accordez à quelqu'un l'accès pour lire ou écrire un noeud dans votre base de données, **vous leur accordez également l'accès à toutes les données sous ce nœud**.

C'est pourquoi , il est préférable de garder votre structure de données **le plus plat possible**.

# ORGANISATION "FLAT"

```
{
  "chats": {
    "one": {
      "title": "Historical Tech Pioneers",
      "messages": {
        "m1": {
            "sender": "ghopper",
            "message": "Relay malfunction found. Cause: moth."
        },
        "m2": { ... },
        // a very long list of messages
      }
    },
    "two": { ... }
  }
}
```

suite ...

# ORGANISATION "FLAT"

Par exemple , la liste des titres des conversations de chat nécessite l'arbre des chats entiers , y compris tous les membres et messages. Tous devra être téléchargés sur le client .

Il vaut donc mieux 'aplatir' votre structure de donnée pour permettre une selection plus fine, et donc obtenir une plus grande réactivité de l'interface du client.

Voici une solution possible qui aurait permis une meilleures itération des données stockées:

# ORGANISATION "FLAT"

```
{
  "chats": {
    "one": {
      "title": "Historical Tech Pioneers",
      "lastMessage": "ghopper: Relay malfunction found. Cause: moth.",
      "timestamp": 1459361875666
    },
    "two": { ... },
    "three": { ... }
  },

  "members": {
    "one": {
      "ghopper": true,
      "alovelace": true,
      "eclarke": true
    },
    "two": { ... },
    "three": { ... }
  },

  "messages": {
    "one": {
```

```
      "m1": {
        "name": "eclarke",
        "message": "The relay seems to be malfunctioning.",
        "timestamp": 1459361875337
      },
      "m2": { ... },
      "m3": { ... }
    },
    "two": { ... },
    "three": { ... }
  }
}
```

# ORGANISATION "FLAT"

Pour plus de détails sur l'organisation des données dans Firebase, se référer à la documentation officiel qui est très complète pour autant que l'on prenne le temps de bien lire chaque informations.

https://firebase.google.com/docs/database/web/structure-data

# FIREBASE SAVE, UPDATE, OR DELETE DATA

# SAUVEGARDER DES DATAS

Il existe 3 principal méthodes pour écrire dans la base de donnée Firebase.

- **set()**: écrit ou remplace les infos d'un noeud spécifique.
- **push()**ajoute un noeud à une collection ou sous-noeud avec une ID générée par Firebase.
- **update()**mettre à jour la valeur d'une clé d'un noeud spécifique sans changer le reste des datas

# EXEMPLES BASIC:

```
function writeUserData(userId, name, email, imageUrl) {
  firebase.database().ref('users/' + userId).set({
    username: name,
    email: email,
    profile_picture : imageUrl
  });
}

function writeNewPost(uid, username, picture, title, body) {
  // A post entry.
  var postData = {
    author: username,
    uid: uid,
    body: body,
    title: title,
    starCount: 0,
    authorPic: picture
  };
  // Get a key for a new Post.
  var newPostKey = firebase.database().ref().child('posts').push().key;
  // Write the new post's data simultaneously in the posts list and the user's post list.
  var updates = {};
  updates['/posts/' + newPostKey] = postData;
```

```
  updates['/user-posts/' + uid + '/' + newPostKey] = postData;
  return firebase.database().ref().update(updates);
}
```

# SUPPRIMER DES DATAS

Pour supprimer des noeuds de la base de donnée Firebase, il existe une méthode remove() mais un simple update() avec la valeur NULL supprimera automatiquement le noeud.

Pour plus de détails sur les méthodes Firebase, ce référer à la documentation:
https://firebase.google.com/docs/database/web/save-data

# FIREBASE RETRIEVE DATA

# RETROUVER LES DATAS DANS FIREBASE

Pour retrouver les datas stockées dans Firebase, on utilise les fonction asynchrone fournis par l'outil. Les Listeners !

Et voici la liste des events que l'on peut associer:

- **value**: lire et écoute les changement d'un noeud.
- **child_added**: retrouver une liste d'item ajouter.
- **child_changed**: écouter les changement d'un item dans une liste
- **child_removed**: écouter lors d'une suppression d'item.
- **child_moved**: écouter lors du déplacement d'un item.

# RETROUVER LES DATAS DANS FIREBASE

L'exemple suivant montre une application de blogging sociale récupérer le nombre d' étoiles d'un poste dans la base de données.

```
// récupérer la référence starCount
// d'un post spécifique (posteId) et y stocker dans starCountRef
var starCountRef = firebase.database().ref('posts/' + postId + '/starCount');

// Créer un listener avec l'event 'value'
// pour récupérer la valeur contenu dans la référence
starCountRef.on('value', function(snapshot) {

    // les valeur sont contenu dans snapshot.val()
    // et passée à une fonction pour être traitée plus loin...
    updateStarCount(postElement, snapshot.val());

});
```

# RETROUVER LES DATAS DANS FIREBASE

Tous les events Firebase fonctionne sur le même principe.
Il est donc très facile de manipuler ses données pour autant que la structure soit bien pensée (use case).

Vous trouverez les détails d'utilisation de chaque events, ainsi que toutes les paramètres de filtrage des données possible dans la documentation Firebase.

https://firebase.google.com/docs/database/web/retrieve-data

# FONCTIONS DE TRIS ET FILTRES FIREBASE

Firebase propose des fonctions de tris et de filtrage des données.

Order by:

- **orderByChild()**: results by the value of a specified child key
- **orderByKey()**: results by child keys
- **orderByValue()**: results by child values

```
var myUserId = firebase.auth().currentUser.uid;
var userPostsRef = firebase.database().ref('user-posts/' + myUserId);
var topUserPostsRef = userPostsRef.orderByChild('starCount');
```

# FONCTIONS DE TRIS ET FILTRES FIREBAS

Filtering data:

- **limitToFirst()**: Sets the maximum number of items to return from the beginning of the ordered list of results.
- **limitToLast()**: Sets the maximum number of items to return from the end of the ordered list of results.
- **startAt()**: Return items greater than or equal to the specified key or value, depending on the order-by method chosen.
- **endAt()**: Return items less than or equal to the specified key or value, depending on the order-by method chosen.
- **equalTo()**: Return items equal to the specified key or value, depending on the order-by method chosen.

```
var recentPostsRef = firebase.database().ref('posts').limitToLast(100);
```

# FIREBASE
# USER AUTHENTICATION

# FIREBASE.AUTH()

Firebase met à disposition un excellent outils d'authentification des utilisateurs. Il permet une multitude de système d'authentification:

Google; Facebook; Twiter; Github; Email & password; user anonyme; custome auth système.

# AUTH. WITH GOOGLE

Comme pour la base de donnée, il faut instencier la méthode d'authentification de Firebase pour pouvoir l'utiliser.

```
let googleProvider = new firebase.auth.GoogleAuthProvider();
```

# AUTH. WITH GOOGLE

```
let googleProvider = new firebase.auth.GoogleAuthProvider();
firebase.auth().signInWithPopup(googleProvider)
.then(function(result) {
  // This gives you a Google Access Token.
  // You can use it to access the Google API.
  let token = result.credential.accessToken;
  // The signed-in user info.
  let user = result.user;
  // ...
}).catch(function(error) {
  // Handle Errors here.
  let errorCode = error.code;
  let errorMessage = error.message;
  // The email of the user's account used.
  let email = error.email;
  // The firebase.auth.AuthCredential type that was used.
  let credential = error.credential;
  // ...
});
```

POUR SE LOGOUT:

POUR SE LOGOUT:

```
firebase.auth().signOut().then(function() {
  // Sign-out successful.
}, function(error) {
  // An error happened.
});
```

# RÉCUPÉRER LES INFO DE L'USER

Pour récupérer les informations concernant l'utilisateur vous pouvez utiliser cette méthode:

```
firebase.auth().onAuthStateChanged(function(user) {
  if (user) {
    // User is signed in.
  } else {
    // No user is signed in.
  }
});
```

```
var user = firebase.auth().currentUser;

if (user) {
  // User is signed in.
} else {
  // No user is signed in.
}
```

# RÉCUPÉRER LES INFO DE L'USER

```
var user = firebase.auth().currentUser;
var name, email, photoUrl, uid;

if (user != null) {
  name = user.displayName;
  email = user.email;
  photoUrl = user.photoURL;
  uid = user.uid;
}
```

Il existe beaucoup de méthodes qui on chacune des champ d'application diverses. Je vous recommande de prendre le temps de consulter la doc afin de prendre connaissance des possibilité qu'offre les système d'authentification de Firebase.

Doc Firebase Auth: https://firebase.google.com/docs/auth/

# FIREBASE

## GESTION DES DROITS ET SÉCURITÉ

# DROIT EN ÉCRITURE ET LECTURE

Pour définir les droits d'écriture et de lecture des datas stockées sur Firebase, rendez-vous sous l'onglet "Règle" dans la section "Database"

Doc officiel:
https://firebase.google.com/docs/database/security/

# DES QUESTIONS?

# FIN ;-)