CS47: Cross-Platform Mobile Development

Lecture 4B: AsyncStorage, SecureStore and PouchDB

James Landay
Abdallah AbuHashem
Tiffany Manuel
Cisco Vlahakis
Vy Mai

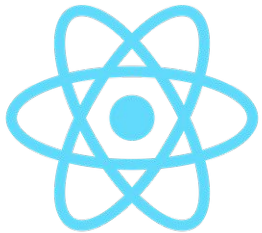
https://cs47.stanford.edu

cs47-fall19.slack.com

Fall 2019

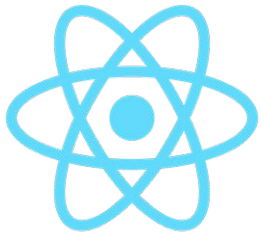
Overview for today

- AsyncStorage
- SecureStore
- PouchDB
- Build a to-do list



Overview for today

- AsyncStorage
- SecureStore
- PouchDB
- Build a to-do list

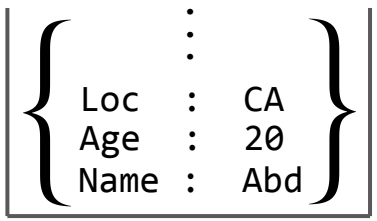


AsyncStorage

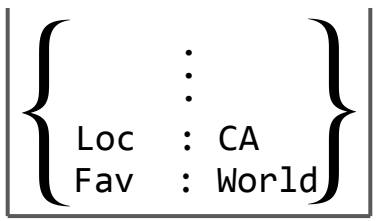
A simple, unencrypted, asynchronous, persistent, key-value storage system that is global to the app.

AsyncStorage

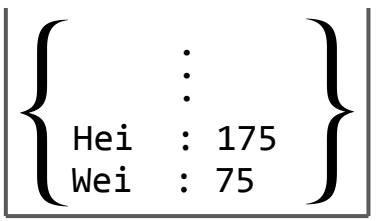
A simple, unencrypted, asynchronous, persistent, key-value storage system that is global to the app.



Tinder



NYT

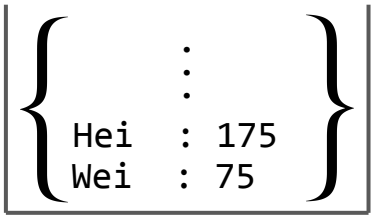


Jedi ID

Using AsyncStorage

Pushing Data

```
try {
  await AsyncStorage.setItem('year', '97');
} catch (error) {
  // Error saving data
}
```



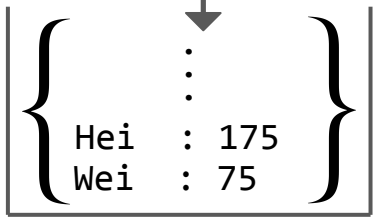
Jedi ID

Using AsyncStorage

Pushing Data

```
try {
  await AsyncStorage.setItem('year', '97');
} catch (error) {
  // Error saving data
}
```

year : 97

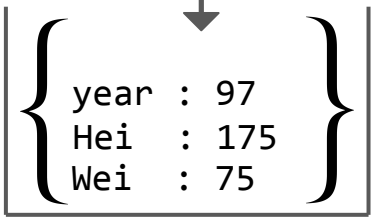


Jedi ID

Using AsyncStorage

Pushing Data

```
try {
  await AsyncStorage.setItem('year', '97');
} catch (error) {
  // Error saving data
}
```



Jedi ID

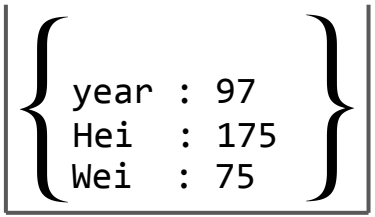
Using AsyncStorage

Pushing Data

```
try {
  await AsyncStorage.setItem('year', '97');
} catch (error) {
  // Error saving data
}
```

Fetching Data

```
try {
  const value = await AsyncStorage.getItem('Hei');
  if (value !== null){
    // We have data!!
    console.log(value);
  }
} catch (error) {
  // Error retrieving data
}
```



Jedi ID

Using AsyncStorage

Pushing Data

```
try {
  await AsyncStorage.setItem('year', '97');
} catch (error) {
  // Error saving data
}
```

Fetching Data

```
try {
  const value = await AsyncStorage.getItem('Hei');
  if (value !== null){
    // We have data!!
    console.log(value);
  }
} catch (error) {
  // Error retrieving data
}
```

‘175’

```
{
  year : 97
  Hei  : 175
  Wei  : 75
}
```

Jedi ID

Using AsyncStorage

Pushing Data - `AsynchStorage.setItem(key, val)`

Fetching Data - `AsynchStorage.getItem(key)`

[Removing Data]

`removeItem()`

```
static removeItem(key: string, [callback]: ?(error: ?Error) => void)
```

Removes an item for a `key` and invokes a callback upon completion. Returns a `Promise` object.

Parameters:

NAME	TYPE	REQUIRED	DESCRIPTION
key	string	Yes	Key of the item to remove.
callback	? (error: ? Error) => void	No	Function that will be called with any error.

```
{
  year : 97
  Hei  : 175
  Wei  : 75
}
```

Jedi ID

Using AsyncStorage

Pushing Data - `AsyncStorage.setItem(key, val)`

Fetching Data - `AsyncStorage.getItem(key)`

Removing Data - `AsyncStorage.removeItem(key)`

Clearing - `AsyncStorage.clear()`

All keys - `AsyncStorage.getAllKeys()`

Using AsyncStorage

Multi pushing

```
try {
  var itemsList = [['Hei', '175'],['Wei', '75'],['year', '97']];
  await AsyncStorage.multiSet(itemsList);

} catch (error) {
  // Error saving data
}
```

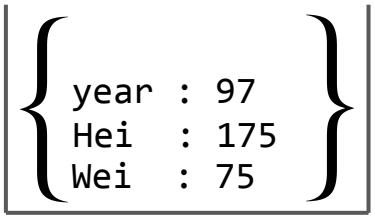
Jedi ID

Using AsyncStorage

Multi pushing

```
try {
  var itemsList = [['Hei', '175'],['Wei', '75'],['year', '97']];
  await AsyncStorage.multiSet(itemsList);

} catch (error) {
  // Error saving data
}
```



Jedi ID

Using AsyncStorage

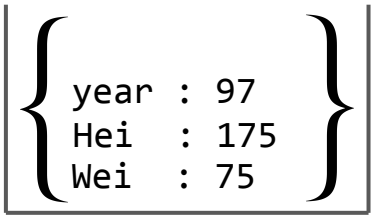
Multi pushing

```
try {
  var itemsList = [['Hei', '175'],['Wei', '75'],['year', '97']];
  await AsyncStorage.multiSet(itemsList);

} catch (error) {
  // Error saving data
}
```

Multi getting

```
try {
  var itemsList = ['year', 'Hei', 'Wei'];
  var values = await AsyncStorage.multiGet(itemsList);
} catch (error) {
  // Error getting data
}
```



Jedi ID

Using AsyncStorage

Multi pushing

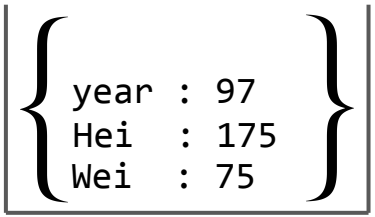
```
try {
  var itemsList = [['Hei', '175'],['Wei', '75'],['year', '97']];
  await AsyncStorage.multiSet(itemsList);

} catch (error) {
  // Error saving data
}
```

Multi getting

```
try {
  var itemsList = ['year', 'Hei', 'Wei'];
  var values = await AsyncStorage.multiGet(itemsList);
} catch (error) {
  // Error getting data
}
```

[[‘year’,‘97’], [‘Hei’,‘175’], [‘Wei’,‘75’]]



Jedi ID

Using AsyncStorage

Pushing Data - `AsyncStorage.setItem(key, val)`

Fetching Data - `AsyncStorage.getItem(key)`

Removing Data - `AsyncStorage.removeItem(key)`

Clearing - `AsyncStorage.clear()`

All keys - `AsyncStorage.getAllKeys()`

Multi pushing - `AsyncStorage.multiSet(array<array<string>>)`

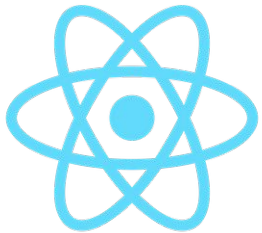
Multi getting - `AsyncStorage.multiGet(array<string>)`

NOTE

All functions of AsyncStorage are asynchronous (Did that surprise you? :o) Make sure you use async/await with it to get desired result

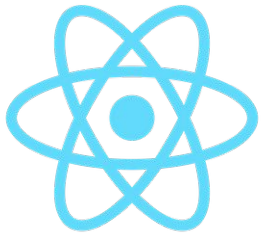
Overview for today

- AsyncStorage ✅
- SecureStore
- PouchDB
- Build a to-do list



Overview for today

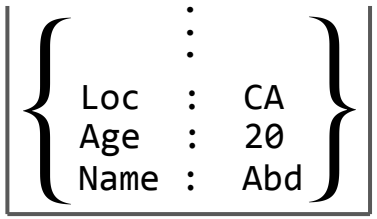
- AsyncStorage ✅
- ➔ SecureStore
- PouchDB
- Build a to-do list



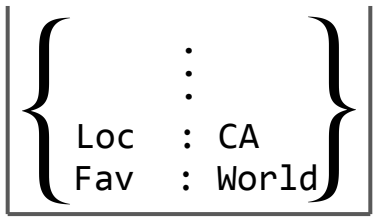
Secure Storage

SecureStore

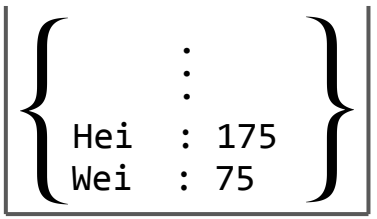
Provides a way to encrypt and securely store key–value pairs locally on the device. Each Expo project has a separate storage system and has no access to the storage of other Expo projects.



Tinder



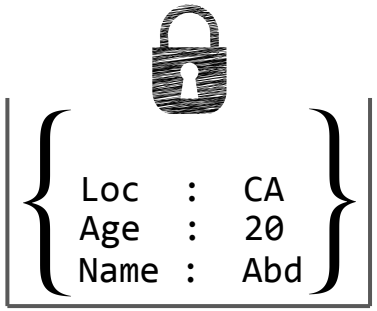
NYT



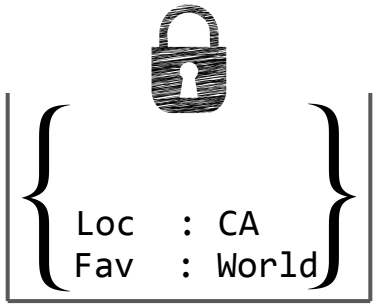
Jedi ID

SecureStore

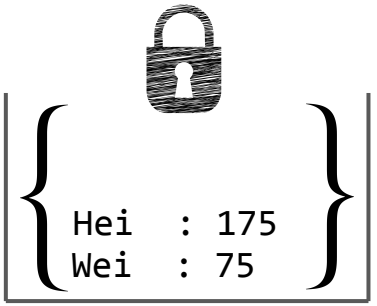
Provides a way to encrypt and securely store key–value pairs locally on the device. Each Expo project has a separate storage system and has no access to the storage of other Expo projects.



Tinder



NYT



Jedi ID

Using SecureStore

Pushing Data

```
try {
  await SecureStore.setItemAsync('year', '97');
} catch (error) {
  // Error saving data
}
```

Using SecureStore

Pushing Data

```
try {
  await SecureStore.setItemAsync('year', '97');
} catch (error) {
  // Error saving data
}
```

Getting Data

```
  try {

    const year = await SecureStore.getItemAsync('year');

  } catch (error) {
    // Error saving data
  }
```

Using SecureStore

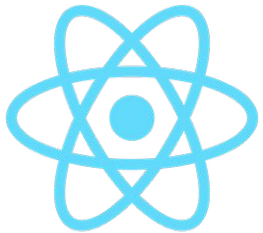
Pushing Data - `SecureStore.setItemAsync(key, value, options)`

Getting Data - `SecureStore.getItemAsync(key, value, options)`

Deleting Data - `SecureStore.deleteItemAsync(key, value, options)`

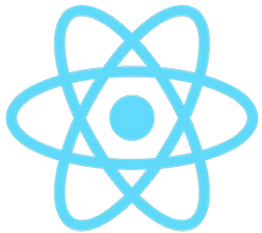
Overview for today

- AsyncStorage ✅
- SecureStore ✅
- PouchDB
- Build a to-do list



Overview for today

- AsyncStorage ✅
- SecureStore ✅
- ➔ PouchDB
- Build a to-do list



PouchDB

PouchDB is an **in-browser database** that allows applications to save data locally, so that users can enjoy all the features of an app even when they're offline. Plus, the data is synchronized between clients, so users can stay up-to-date wherever they go.

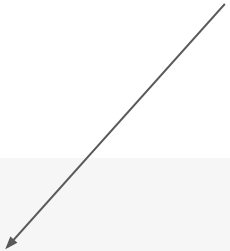
PouchDB

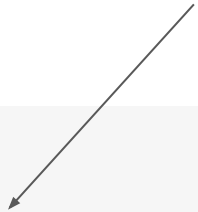
https://pouchdb.com/api.html

Initialize via name

Initialize via API URL

Initializing DB reference





Example Usage:

```
var db = new PouchDB('dbname');
// or
var db = new PouchDB('http://localhost:5984/dbname');
```

PouchDB

Putting data

Property-value pairs

Upon success, execute this code

Upon error, execute this code

```
db.put({
  _id: 'mydoc',
  title: 'Heroes'
}).then(function (response) {
  // handle response
}).catch(function (err) {
  console.log(err);
});
```

PouchDB

Getting data Return object with id “mydoc”

```
db.get('mydoc').then(function (doc) {
  // handle doc
}).catch(function (err) {
  console.log(err);
});
```

```
{
  "_id": "mydoc",
  "_rev": "1-A6157A5EA545C99B00FF904EEF05FD9F"
  "title": "Rock and Roll Heart",
}
```

PouchDB

Batch putting

Specify an array of objects

```
db.bulkDocs([
  {title : 'Lisa Says', _id: 'doc1'},
  {title : 'Space Oddity', _id: 'doc2'}
]).then(function (result) {
  // handle result
}).catch(function (err) {
  console.log(err);
});
```

PouchDB

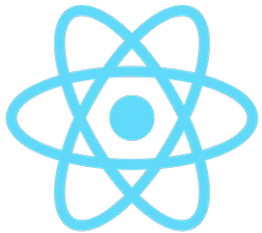
Batch deleting

```
db.bulkDocs([
  {
    title    : 'Lisa Says',
    _deleted : true,
    _id      : "doc1",
    _rev     : "1-84abc2a942007bee7cf55007cba56198"
  },
  {
    title    : 'Space Oddity',
    _deleted : true,
    _id      : "doc2",
    _rev     : "1-7b80fc50b6af7a905f368670429a757e"
  }
]).then(function (result) {
  // handle result
}).catch(function (err) {
  console.log(err);
});
```

To delete, set the _deleted flag to be true

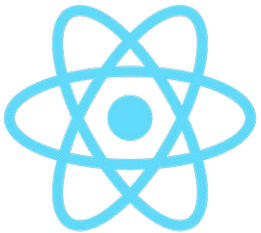
Overview for today

- AsyncStorage ✅
- SecureStore ✅
- PouchDB ✅
- Build a to-do list



Overview for today

- AsyncStorage ✅
- SecureStore ✅
- PouchDB ✅
- ➡ Build a to-do list



Exercise

STARTER CODE

CS47: Cross-Platform Mobile Development

Lecture 4B: AsyncStorage, SecureStore and PouchDB

James Landay
Abdallah AbuHashem
Tiffany Manuel
Cisco Vlahakis
Vy Mai

https://cs47.stanford.edu

cs47-fall19.slack.com

Fall 2019