# FRONTEND DEVELOPMENT

## WINTERSEMESTER 2022

# PERSISTING DATA IN THE BROWSER

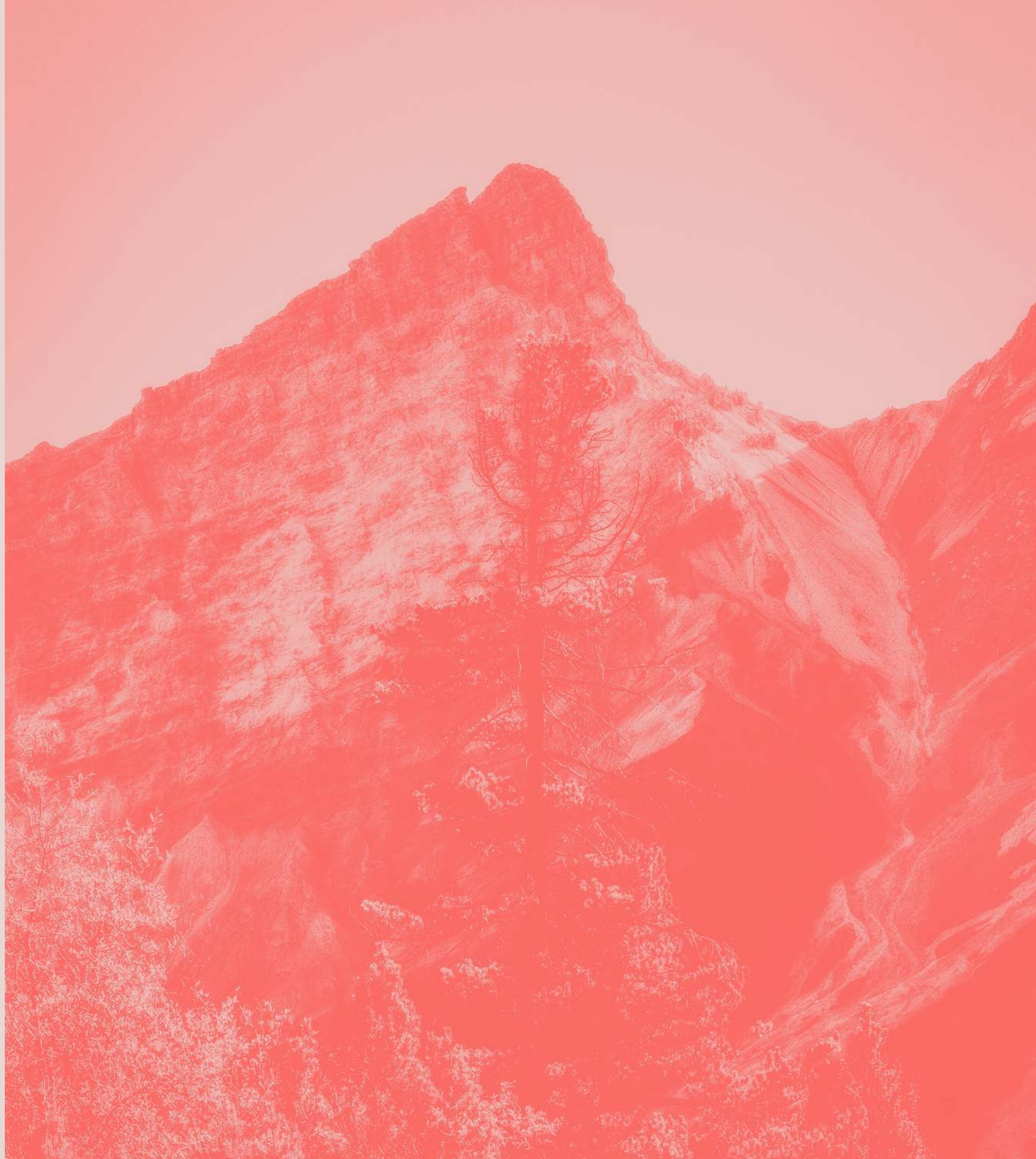
# AGENDA

» History State

» Session Storage

» Local Storage

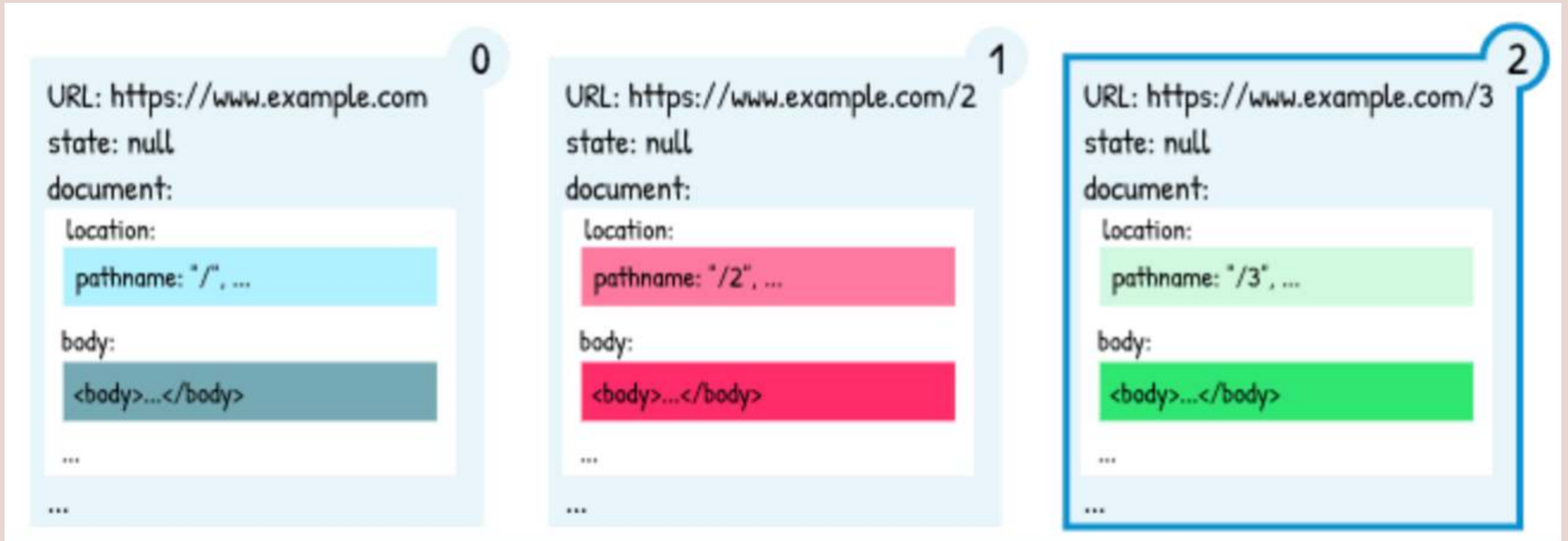
» Indexed DB



# HISTORY STATE

# HISTORY STATE
## RECAP

"Each browser tab has a "browsing context". The browsing context maintains a "session history", which is essentially an array of location entries. [^2]"



# HISTORY STATE
## RECAP PUSHSTATE

```
// history.pushState(<state>, <title>, <url>)

history.pushState(null, 'page 1', '/page1')
history.pushState({ page: 2 }, 'page 2', '/page2')
// 1)                ^^^^^^^^^^^^
// 1) add some state to the session history

history.state // => { page: 2 }
history.back(-1)
history.state // => null
```

# HISTORY STATE
## WHEN TO USE PUSH STATE [^4]

» data can be added to the current session history entry

» data does not need to be synced between tabs

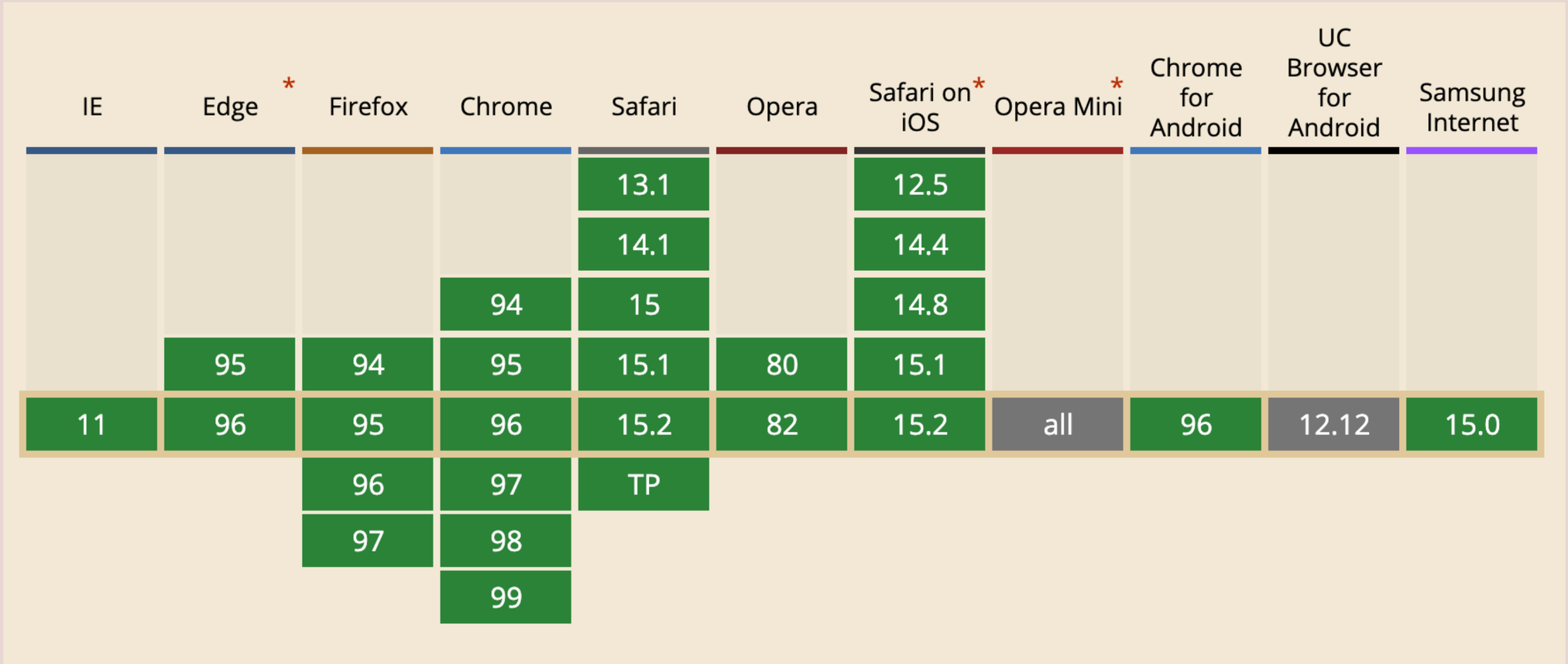
» data will get lost when page session ends

  » data will get lost when tab was closed [^3]

» size is limited to 2MiB

» data is not shared between tabs

# HISTORY STATE CAN I USE [^6]



| IE | Edge * | Firefox | Chrome | Safari | Opera | Safari on iOS * | Opera Mini * | Chrome for Android | UC Browser for Android | Samsung Internet |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 13.1 | | 12.5 | | | | |
| | | | | 14.1 | | 14.4 | | | | |
| | | | 94 | 15 | | 14.8 | | | | |
| | 95 | 94 | 95 | 15.1 | 80 | 15.1 | | | | |
| 11 | 96 | 95 | 96 | 15.2 | 82 | 15.2 | all | 96 | 12.12 | 15.0 |
| | | 96 | 97 | TP | | | | | | |
| | | 97 | 98 | | | | | | | |
| | | | 99 | | | | | | | |

# SESSION STORAGE

# SESSION STORAGE [^5]

- temporary storage which expires when the page session ends
- session storage is only available to the current origin and tab
  - 2 tabs don't share the same session storage
- when tab gets duplicated
  - session storage gets copied

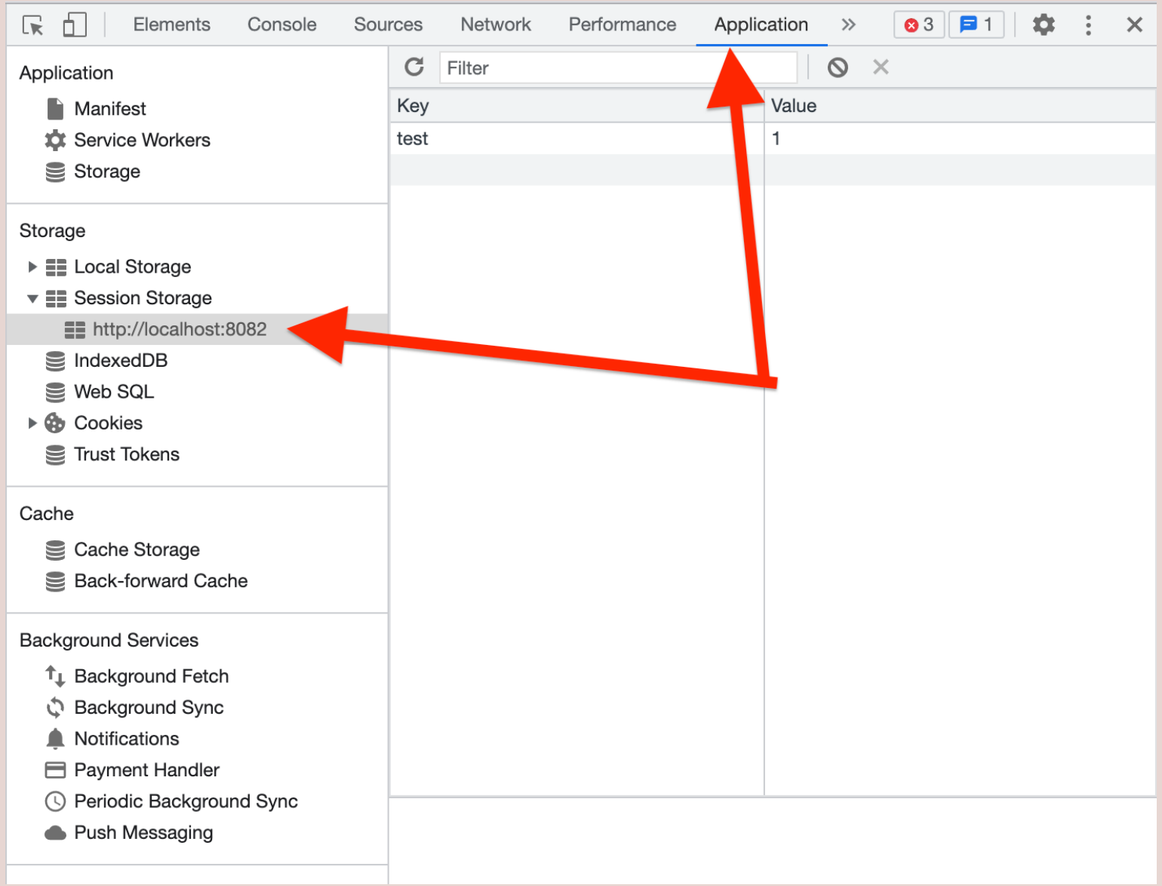
# SESSION STORAGE
## PERSISTING VALUES

```
window.sessionStorage.setItem('someKey1', 'some important value')
//                            1) ^^^^^^^^^^
//                                          2) ^^^^^^^^^^^^^^^^^^^^^^
// 1) the key where the value is stored
// 2) the value to be stored

window.sessionStorage.setItem('someKey2', JSON.stringify({ test: 1 }))
//                                          3) ^^^^^^^^^^^^^^^^
// 3) stringify the object
```

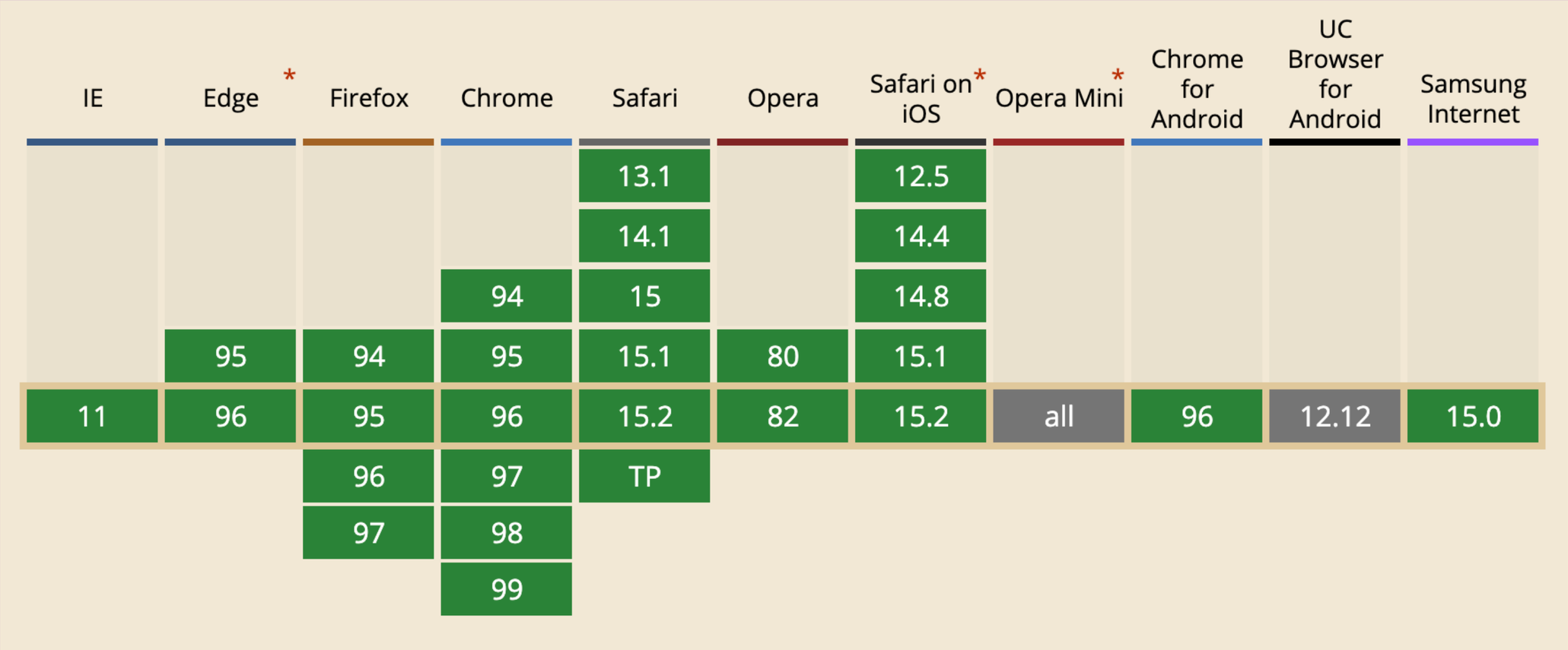
# SESSION STORAGE
## READING VALUES

```
window.sessionStorage.setItem('someKey1', 'some value')
window.sessionStorage.getItem('someKey1') // => 'some value'


window.sessionStorage.setItem('someKey2', JSON.stringify({ test: 1 }))
window.sessionStorage.getItem('someKey1') // => '{ "test": 1 }'
//                                                ^^^^^^^^^^^^^^^^
// this is returned as string and needs to be serialized again
```

# SESSION STORAGE
# IN CHROME DEV TOOLS



# SESSION STORAGE CAN I USE [^6]



| IE | Edge * | Firefox | Chrome | Safari | Opera | Safari on iOS * | Opera Mini * | Chrome for Android | UC Browser for Android | Samsung Internet |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 13.1 | | 12.5 | | | | |
| | | | | 14.1 | | 14.4 | | | | |
| | | | 94 | 15 | | 14.8 | | | | |
| | 95 | 94 | 95 | 15.1 | 80 | 15.1 | | | | |
| 11 | 96 | 95 | 96 | 15.2 | 82 | 15.2 | all | 96 | 12.12 | 15.0 |
| | | 96 | 97 | TP | | | | | | |
| | | 97 | 98 | | | | | | | |
| | | | 99 | | | | | | | |

# SESSIONSTORAGE
## EXERCISE TIME (15 MINUTES)

- Open your quiz application on a new branch
  - try to store the selected answer in the sessionStorage

# LOCAL STORAGE

# LOCAL STORAGE

- persistent storage which does not expire
- key/values can be stored
  - only supports strings as data types
- max size roughly 5MB (depending on the browser)
- blocks the event loop (for read and writes)
  - should be avoided for large datasets

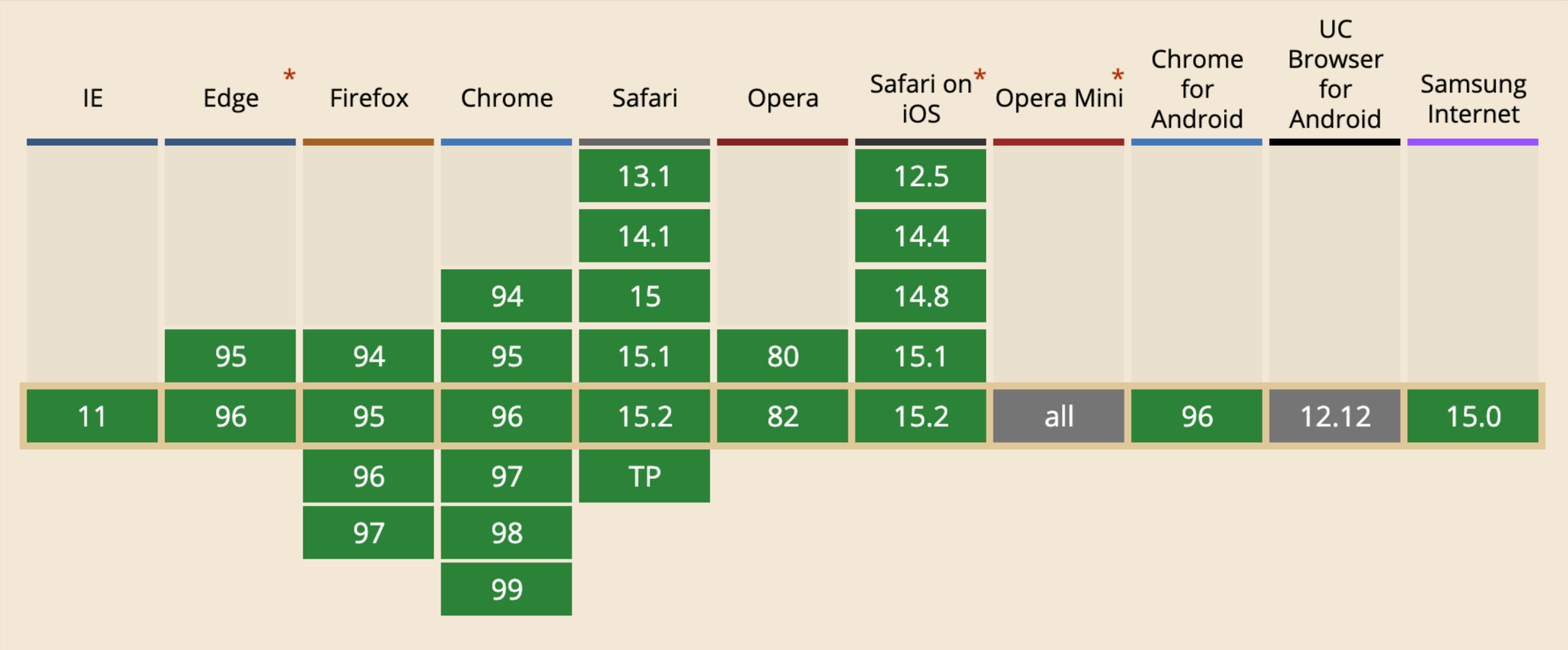
# LOCAL STORAGE
## PERSISTING VALUES

```
window.localStorage.setItem('someKey1', 'some important value')
//                          1) ^^^^^^^^^
//                                      2) ^^^^^^^^^^^^^^^^^^^^^
// 1) the key where the value is stored
// 2) the value to be stored

window.localStorage.setItem('someKey2', JSON.stringify({ test: 1 }))
//                                      3) ^^^^^^^^^^^^^^^
// 3) stringify the object
```

# LOCAL STORAGE
## READING VALUES

```
window.localStorage.setItem('someKey1', 'some value')
window.localStorage.getItem('someKey1') // => 'some value'


window.localStorage.setItem('someKey2', JSON.stringify({ test: 1 }))
window.localStorage.getItem('someKey1') // => '{ "test": 1 }'
//                                            ^^^^^^^^^^^^^^^^
// this is returned as string and needs to be serialized again
```

# SESSION STORAGE CAN I USE [^6]



| IE | Edge * | Firefox | Chrome | Safari | Opera | Safari on iOS * | Opera Mini * | Chrome for Android | UC Browser for Android | Samsung Internet |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 13.1 | | 12.5 | | | | |
| | | | | 14.1 | | 14.4 | | | | |
| | | | 94 | 15 | | 14.8 | | | | |
| | 95 | 94 | 95 | 15.1 | 80 | 15.1 | | | | |
| 11 | 96 | 95 | 96 | 15.2 | 82 | 15.2 | all | 96 | 12.12 | 15.0 |
| | | 96 | 97 | TP | | | | | | |
| | | 97 | 98 | | | | | | | |
| | | | 99 | | | | | | | |

# INDEXEDDB

# INDEXEDDB

» Adds possibility to store data

» non-relational database

  » does not use SQL

# - SUPPORTED IN MANY BROWSERS

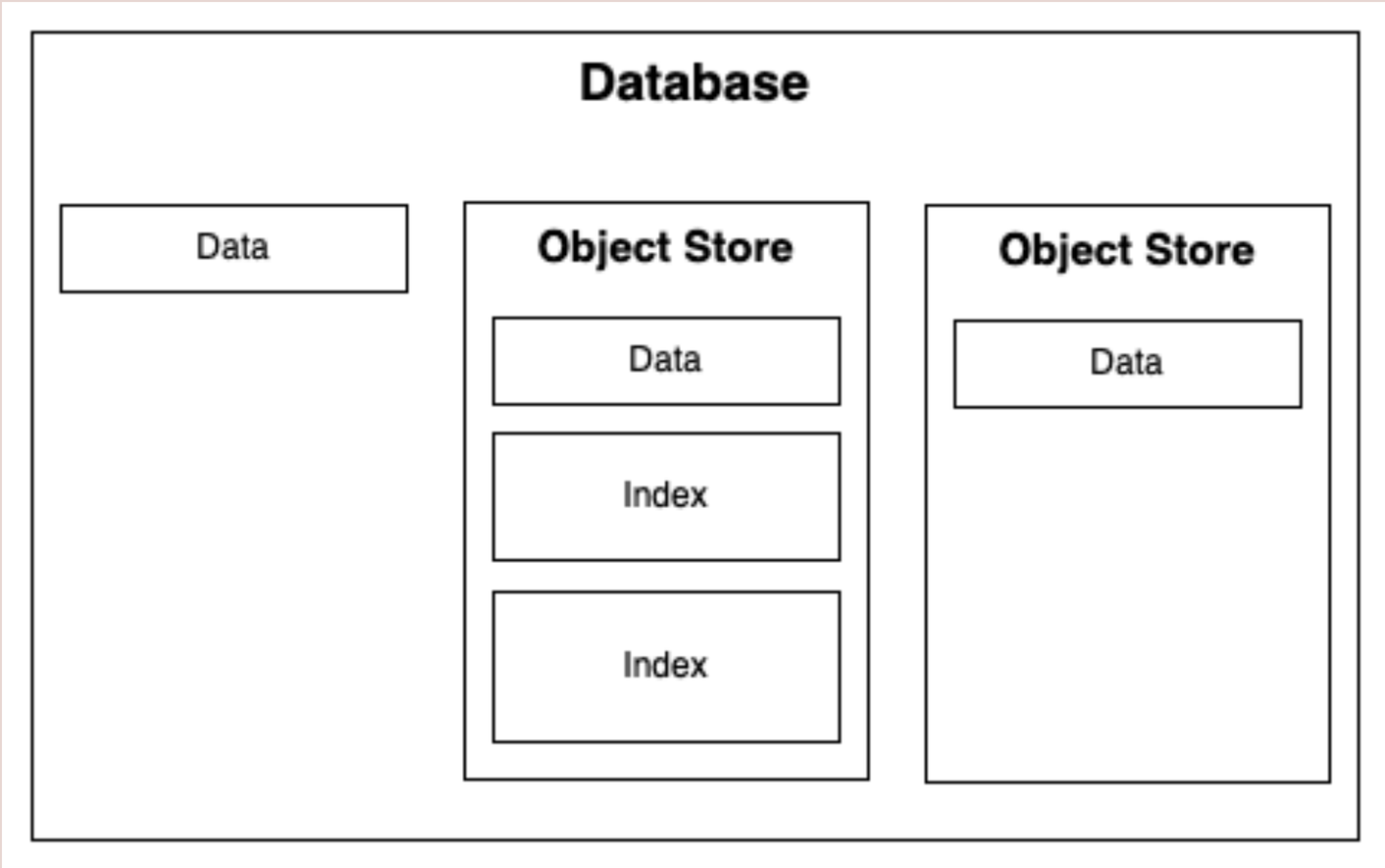
# WHAT IS INDEXEDDB

IndexedDB provides an object store in the browser

» non-relational database

# INDEXEDDB STRUCTURE

## CAN I USE [^6]



# INDEXEDDB STRUCTURE

- Database (a group of related data)
  - each database can have multiple object stores
- Object Store
  - similar to a table in relational DBs
  - contains data
  - eg. user profile object store
  - contains index

# USAGE WITH IDB
## INSTALLATION

» IDB is a small wrapper around indexedDB

  » original implementation was not promise based

» helps with data migration

» installation via NPM
  npm i idb

» via CDN
  import idb from 'https://cdn.jsdelivr.net/npm/

# USAGE WITH IDB
## CREATING A DATABASE

```
const db = await openDB('test-db', 1, {
//            1) ^^^^^^
//                   2) ^^^^^^^^
//                              3) ^
// 1) open/connect to the database
// 2) name of the database
// 3) version number

    upgrade(db, currentVersion) {
//  ^^^^^^^^^^^^^^ define database migrations

        if (currentVersion === 0) {
            db.createObjectStore('users', { keyPath: 'email' })
//       4) ^^^^^^^^^^^^^^^^^^^^
//                            5) ^^^^^^
//                                       6) ^^^^^^
// 4) create an objectStore
// 5) the name of the ObjectStore
// 6) define the path to the key (value should be unique)
        }
    },
})
```

# USAGE WITH IDB
## INSERTING/UPDATING DATA

```
await db.put('users', { name: 'Mike', email: 'mike@test.com', age: 14 })
//          1) ^^^^^^^
//          2)           ^^^^^^^
// 1) the name of the ObjectStore
// 2) the value to persist
```

# USAGE WITH IDB
## DELETING DATA

```
await db.delete('users', 'mike@test.com')
//          1)    ^^^^^^^
//          2)                  ^^^^^^^^^^^^^^^
// 1) the name of the ObjectStore
// 2) the key to delete from the ObjectStore
```

# USAGE WITH IDB
## REMOVE ALL ENTRIES

```
await db.clear('users')
//          1)    ^^^^^^^
// 1) the name of the ObjectStore to be cleared
```

# USAGE WITH IDB
## CREATING AN INDEX

```
const db = await openDB('test-db', 2, {
//                              1) ^
    upgrade(db, currentVersion) {
        if (currentVersion === 0) {
            // ... don't touch existing migrations
        }

        if (currentVersion === 1) {
          store.createIndex('nameIdx', 'name');
          //              2) ^^^^^^
          //              3)             ^^^^^
        }
    },
})

// 1) increase the version number
// 2) define a name for the index
// 3) define the property to be indexed
```

# USAGE WITH IDB
## MULTIPLE VERSION ISSUE

» indexedDBs might not be up to date with your code

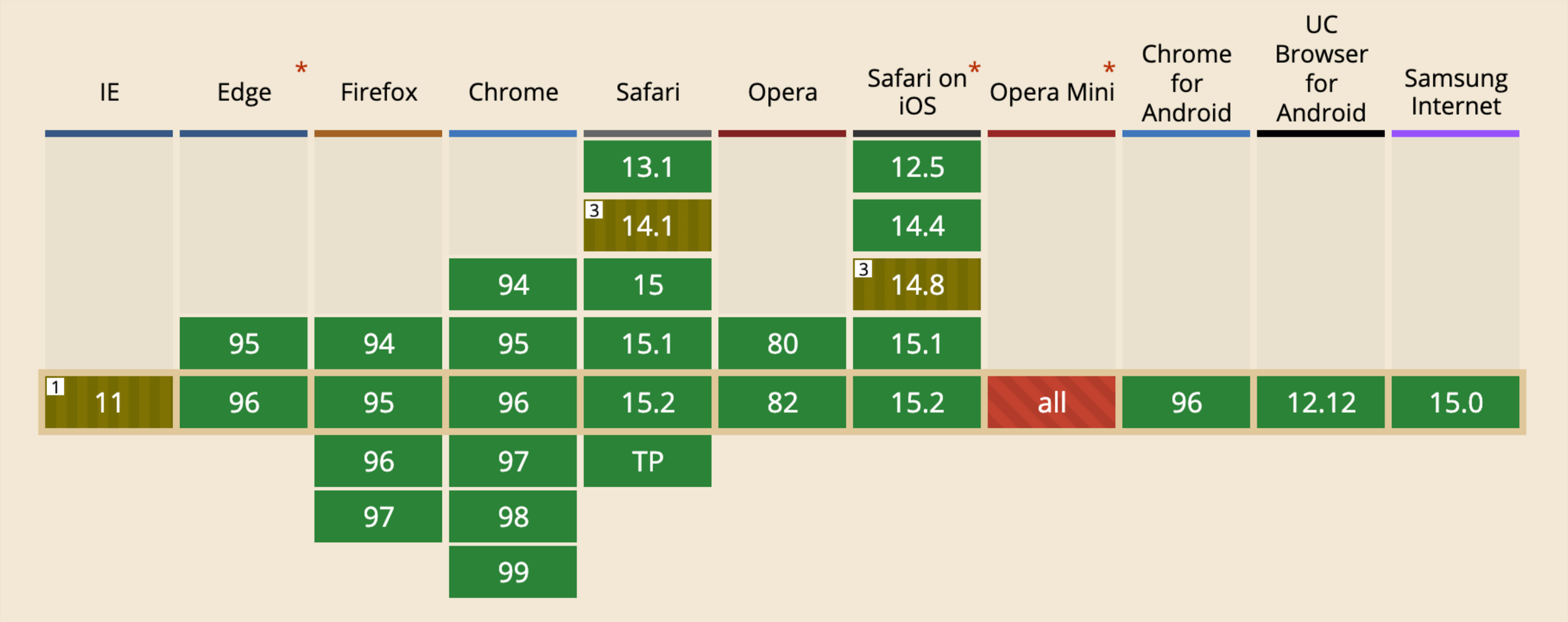
  » data migrations need to be written

» a user might be behind several db versions

# USAGE WITH IDB
## MIGRATE SEVERAL VERSIONS

```
const db = await openDB('test-db', 10, {
    upgrade(db, currentVersion, nextVersion) {
      for (let version = currentVersion; version <= nextVersion; version++) {
      //  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
      // iterate through all versions which hadn't been executed yet
        if (currentVersion === 0) {
          // first migration
        }
        if (currentVersion === 1) {
          // second migration
        }
        // ...
      }
    },
})
```

# INDEXEDDB [^6]


IE
Edge *
Firefox
Chrome
Safari
Opera
Safari on iOS *
Opera Mini *
Chrome for Android
UC Browser for Android
Samsung Internet
13.1
12.5
3 14.1
14.4
94
15
3 14.8
95
94
95
15.1
80
15.1
1 11
96
95
96
15.2
82
15.2
all
96
12.12
15.0
96
97
TP
97
98
99

# INDEXEDDB
# EXERCISE TIME

» Open your quiz application on a new branch

  » try to store your questions in the indexed db

» try to read the questions from there

# FEEDBACK

» Questions: tmayrhofer.lba@fh-salzburg.ac.at

» Feedback Link