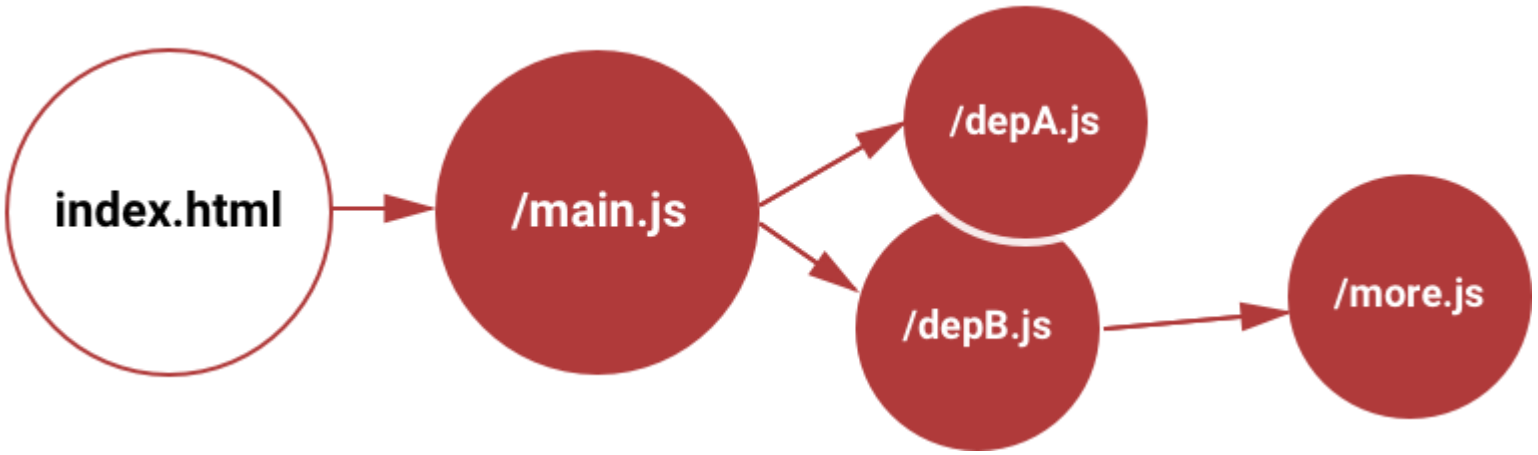
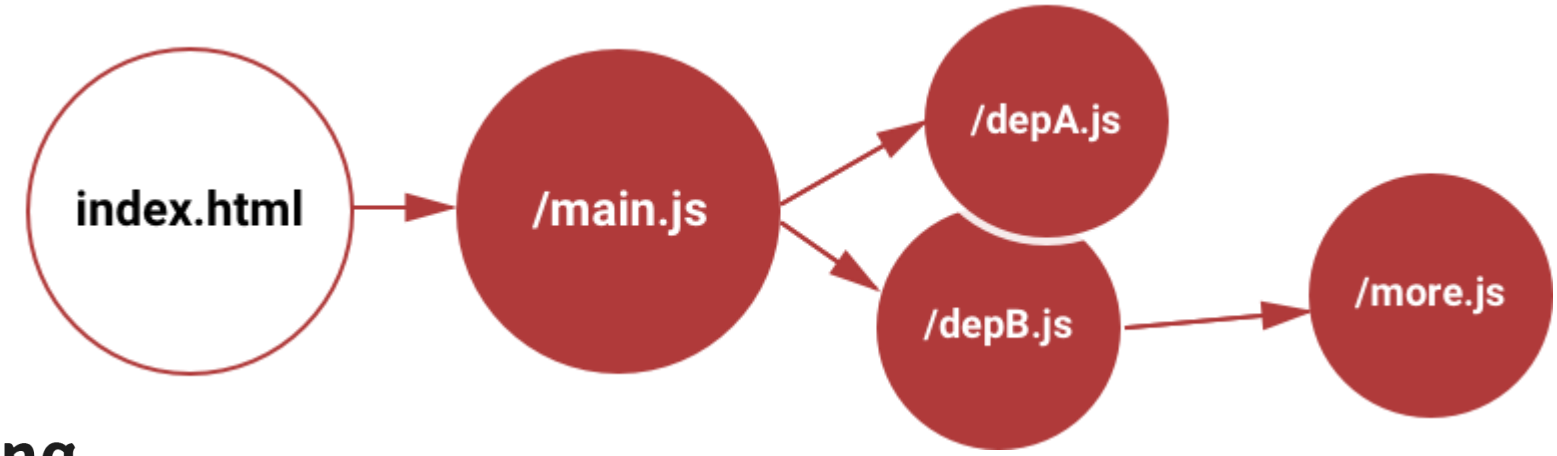
Switching to modules

Let the app grow



Javascript modules

- When coding modern javascript we use modules
- Lets review



type="module"

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>JS Samples</title>
  </head>
  <body>
    <h1>JS Modules</h1>
    <button class="btn-foo" onclick="onFoo()">Foo</button>
    <script type="module" src="js/main.js"></script>
  </body>
</html>
```

strict mode

When working with modules, we are automatically in strict mode

- bye bye `'use strict'`

```
bla=0 // Uncaught ReferenceError: bla is not defined
```

No globals

- When working with modules, variables are not global
- We can define global variable by accessing the window object

```
var gNotGlobal = 'Nope' // I'm known in the file only
window.gGlobal = 'Yep' // I am a global
```

No global event handlers

- When working with modules, functions are not global

```
<button class="btn-foo" onclick="onFoo()">Foo</button>
```

- Here is a simple workaround:

```
window.onFoo = onFoo

function onFoo() {
    console.log('Foo!')
}
```

Named imports and exports

Here is how

```
// in file: util.service.js

export const utilService = {
    saveToStorage,
    loadFromStorage,
    makeId
}
```

```
// some other file:
import { utilService } from './util.service.js'
```

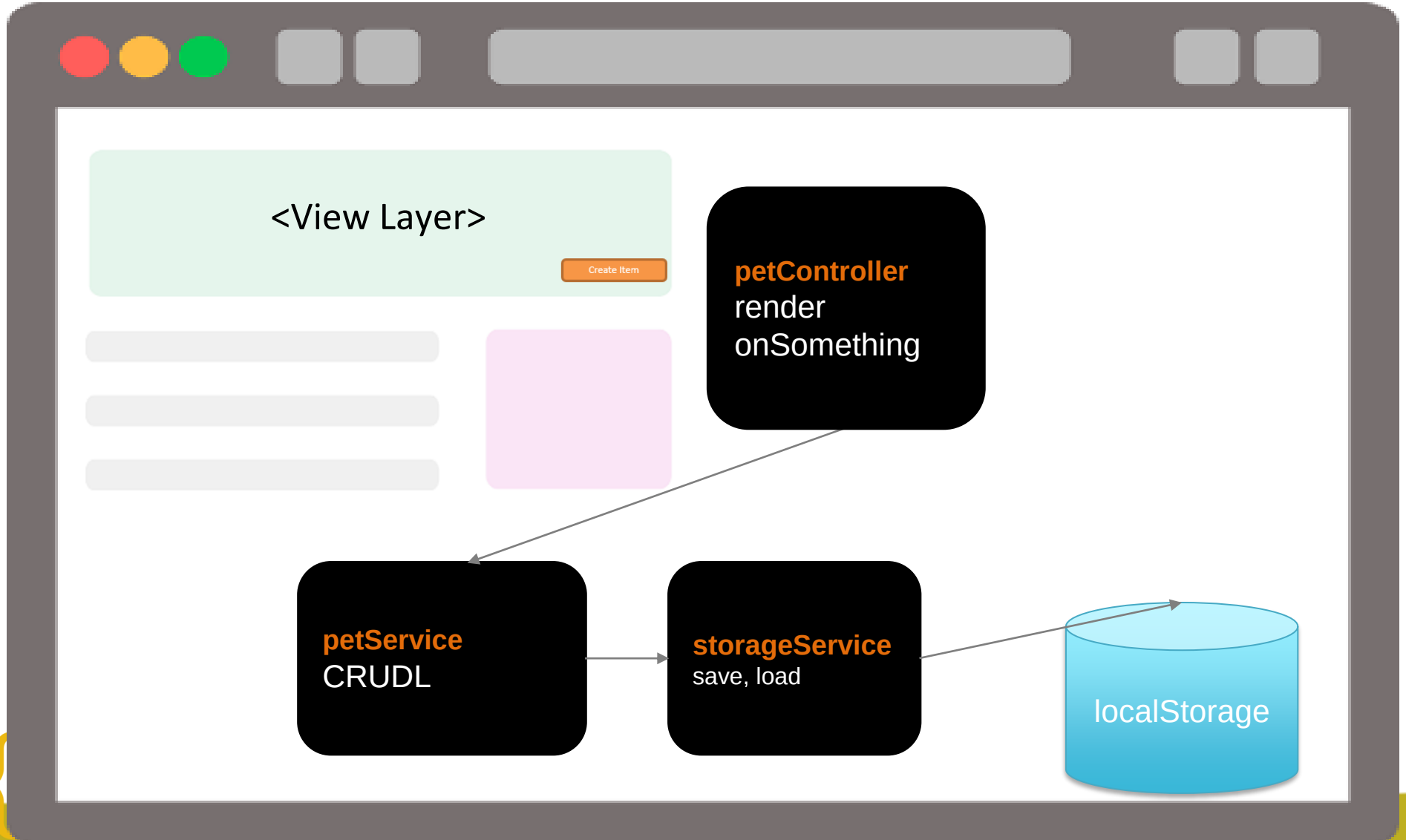
CRUDL

As we've seen before, when building apps, we usually have some entities that the application manages:

We usually need to:

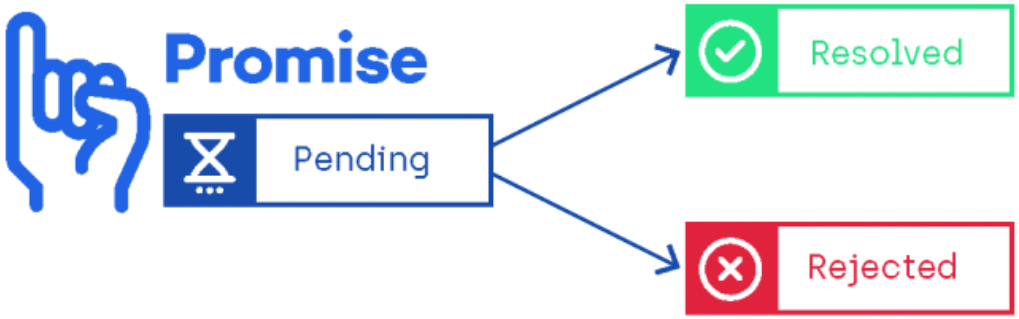
- Create – add a new entity
- Read – read the entire details of the entity
- Update – update the entity
- Delete – remove the entity
- List – Read a list of the entity preview
  - (filtered / ordered / paging)

Back to our MVC



Switching our CRUDL service to work with promises

- In our frontend, our the services will later use AJAX to call the server and perform the CRUDL on the database
- So the response will come as a promise



Meet the async-storage service

This service implements functions for our CRUDL with a Promise API

```
export const storageService = {
    post,   // Create
    get,    // Read
    put,    // Update
    remove, // Delete
    query   // List
}
```

Meet the async-storage service

The query function is used by the other functions:

```
function query(entityType, delay = 500) {
    var entities = JSON.parse(localStorage.getItem(entityType)) || []
    return new Promise(resolve => setTimeout(() => resolve(entities), delay))
}

function get(entityType, entityId) {
    return query(entityType).then(entities => {
        const entity = entities.find(entity => entity.id === entityId)
        return entity
    })
}

function remove(entityType, entityId) {
    return query(entityType).then(entities => {
        const idx = entities.findIndex(entity => entity.id === entityId)
        entities.splice(idx, 1)
        _save(entityType, entities)
    })
}
```

Pet service

This service uses the async-storage-service to provide CRUDL on cars:

```
function get(petId) {
    return storageService.get(PET_KEY, petId)
}

function remove(petId) {
    return storageService.remove(PET_KEY, petId)
}

function save(pet) {
    if (pet.id) {
        return storageService.put(PET_KEY, pet)
    } else {
        return storageService.post(PET_KEY, pet)
    }
}
```

Pet service filtering support

Our service supports filtering:

```
var gFilterBy = {txt: '', minScore : 0}
function query() {
    return storageService.query(PET_KEY)
        .then(pets => {
            if (gFilterBy.txt) {
                const regex = new RegExp(gFilterBy.txt, 'i')
                pets = pets.filter(pet => regex.test(pet.name))
            }
            if (gFilterBy.minScore) {
                pets = pets.filter(pet => pet.score >= gFilterBy.minScore)
            }
            return pets
        })
}
```

Pet service demo data

Our service demonstrates creating demo data

```
function _createDemoPets() {
    const petNames = ['Bobi', 'Charli', 'Pinchi']
    const petDescs = ['Bobi is an amazing dog',
           'Charli is a curious cat', 'Just one look at Pinchi']

    const pets = petNames.map((petName, i) => {
        const pet = _createPet(petName)
        pet.desc = petDescs[i]
        return pet
    })

    utilService.saveToStorage(PET_KEY, pets)
}

function _createPet(name) {
    const pet = getEmptyPet()
    pet.id = utilService.makeId()
    pet.type = utilService.randomPetType()
    pet.name = name || utilService.randomPetName(pet.type)
    pet.birth = utilService.randomPastTime()
    return pet
}
```

Switching to modules

Done

We are Ready