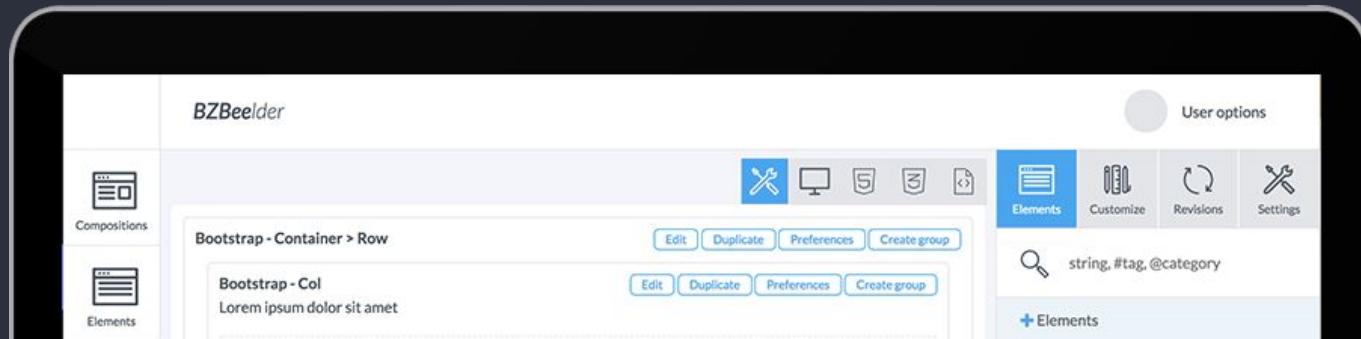


meet.js
kielce

CMS as a side project

Why it's good to have self side project



Who am I?

Blog: <http://fedojo.com>

GitHub: <https://github.com/fedojo>



BZBeelder

Trendy approach done with Node.js and Angular

The screenshot shows the BZBeelder application interface. At the top, there's a navigation bar with the title "BZBeelder". Below the title is a toolbar with several icons: a trash can, a monitor, a magnifying glass, a refresh symbol, and a file icon. To the right of the toolbar is a "User options" button. On the left side, there's a sidebar with two main categories: "Compositions" and "Elements". Under "Compositions", there's a single item labeled "Bootstrap - Container > Row". Under "Elements", there's a single item labeled "Bootstrap - Col" with the subtext "Lorem ipsum dolor sit amet". Both of these items have "Edit", "Duplicate", "Preferences", and "Create group" buttons next to them. In the center of the screen is a large workspace area with a dashed border, containing the text "Drop your elements here!". To the right of the workspace is a sidebar with four buttons: "Elements" (which is highlighted in blue), "Customize", "Revisions", and "Settings". Below these buttons is a search bar with the placeholder text "string, #tag, @category". At the bottom of the sidebar is a button labeled "+ Elements".

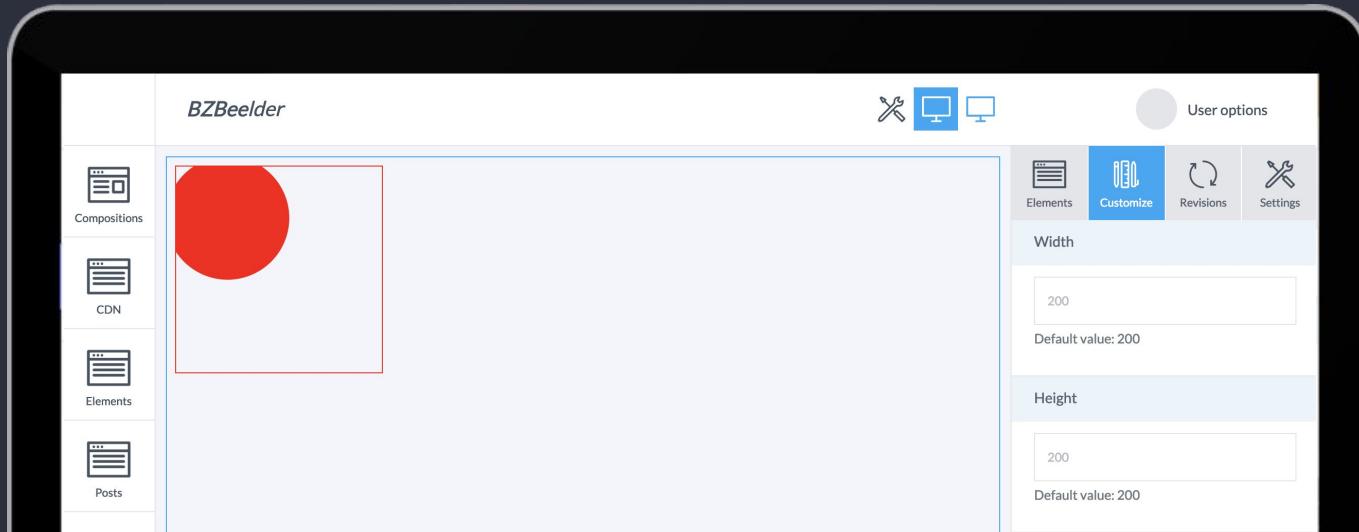
What is BZBeelder

- OpenSource CMS / framework (not yet fully published officially)
- HTML elements repository and website/template composer
- Blog engine
- Foundation for other purposes

The screenshot shows the BZBeelder web application interface. At the top, there's a header with the title "BZBeelder" and a "User options" button. On the left, a sidebar menu includes "Compositions" (selected), "CDN", and "Elements". The main content area is titled "Compositions list" and features a search bar and a "Create new" button. Below this, there's a section for "Display options". A table lists compositions with columns for "Name", "Create date", and "Actions". One row is visible, showing "SVGbased" created on "Dec 19, 2018, 10:41:09 AM" with "Compose" and "Remove" buttons.

Name	Create date	Actions
SVGbased	Dec 19, 2018, 10:41:09 AM	Compose Remove

Tech stack

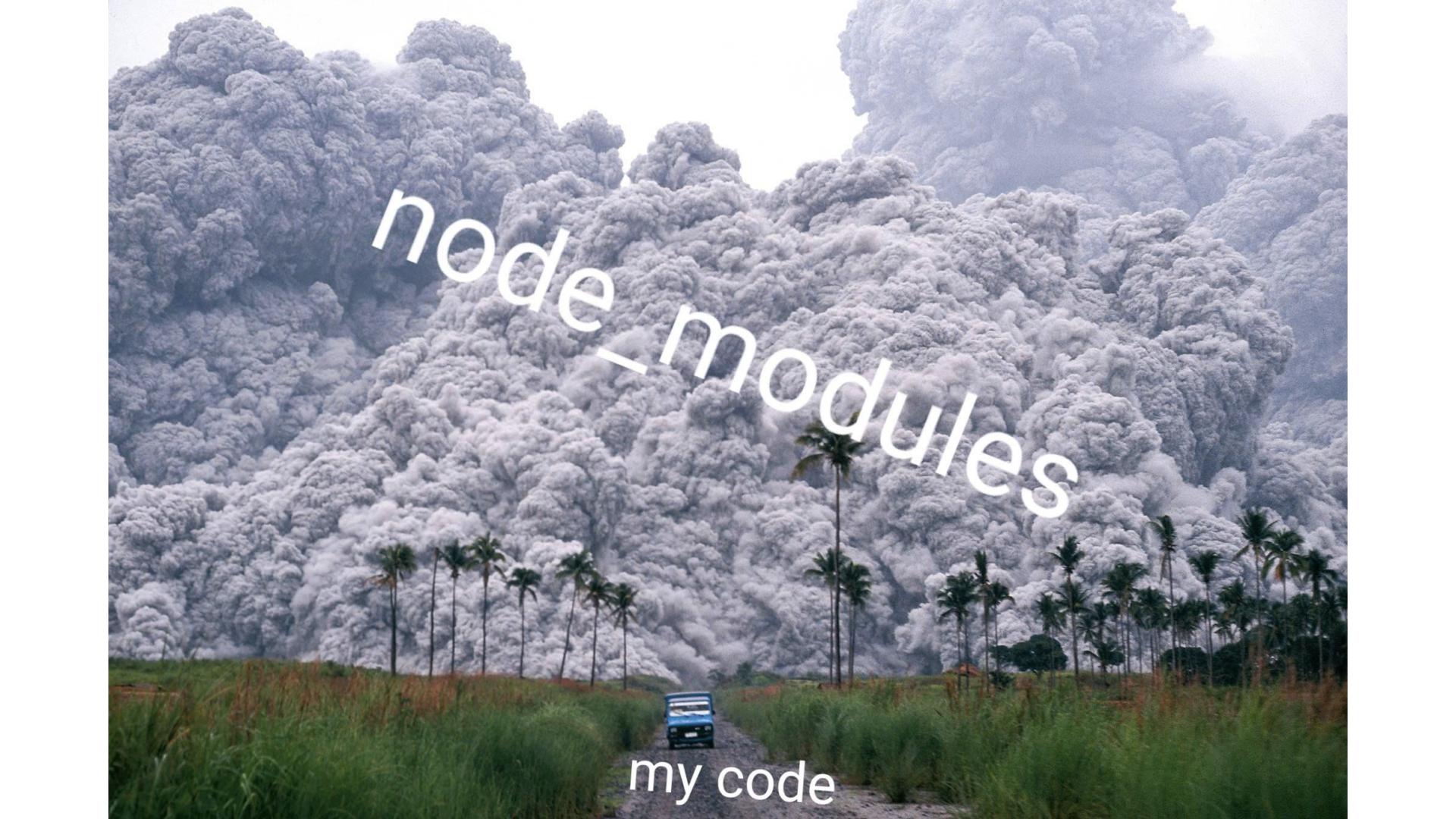


Tech stack

- NodeJS / ExpressJS / Mongoose
- MongoDB
- Angular 5



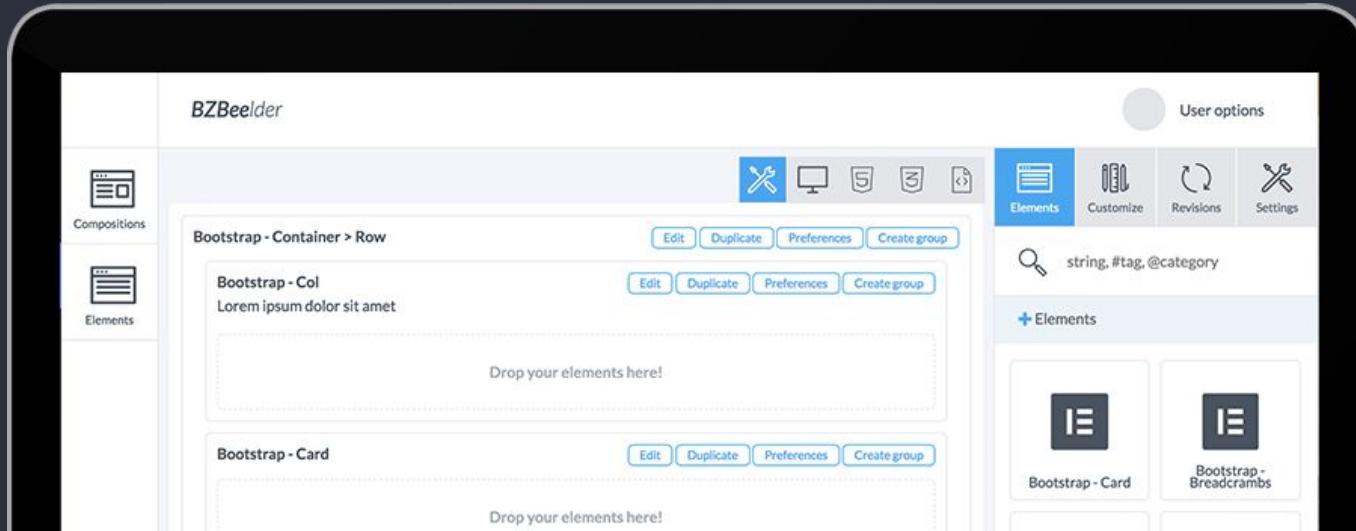
Another project based on MEAN stack...
BLEH!

A photograph of a volcanic eruption. A massive, dark grey plume of ash and smoke rises from the volcano, filling most of the frame. In the foreground, there is a field of tall green grass and several palm trees. A small blue vehicle is parked on a dirt road that leads towards the volcano. The sky is overcast and grey.

node_modules

my code

CMS?



What is CMS?

Who is using CMS?

Client?

Developer?

Story

The screenshot shows the BZBeelder interface, a content management system designed for creating stories. The top navigation bar includes 'User options' (User icon), 'Elements' (grid icon), 'Customize' (gear icon), 'Revisions' (refresh icon), and 'Settings' (cog icon). A search bar at the top right contains the placeholder text 'string, #tag, @category'. The main workspace displays two story components:

- Bootstrap - Container > Row**: This component contains a single 'Bootstrap - Col' element. The 'Col' element has the text 'Lorem ipsum dolor sit amet' and includes 'Edit', 'Duplicate', 'Preferences', and 'Create group' buttons.
- Bootstrap - Card**: This component contains a placeholder message 'Drop your elements here!' and includes 'Edit', 'Duplicate', 'Preferences', and 'Create group' buttons.

The left sidebar features two sections: 'Compositions' (document icon) and 'Elements' (grid icon). The 'Elements' section is currently active, showing a list of available components: 'Bootstrap - Card' and 'Bootstrap - Breadcrumbs'. Each component is represented by a small icon and a label.

Why next CMS?

- I've been using Wordpress / Typo3 / Drupal (a bit)
- I need something "own"
- I need something more elastic
- I need a tool which gives me opportunity to create projects based on previously created elements and repository of those elements



Why next CMS? ...

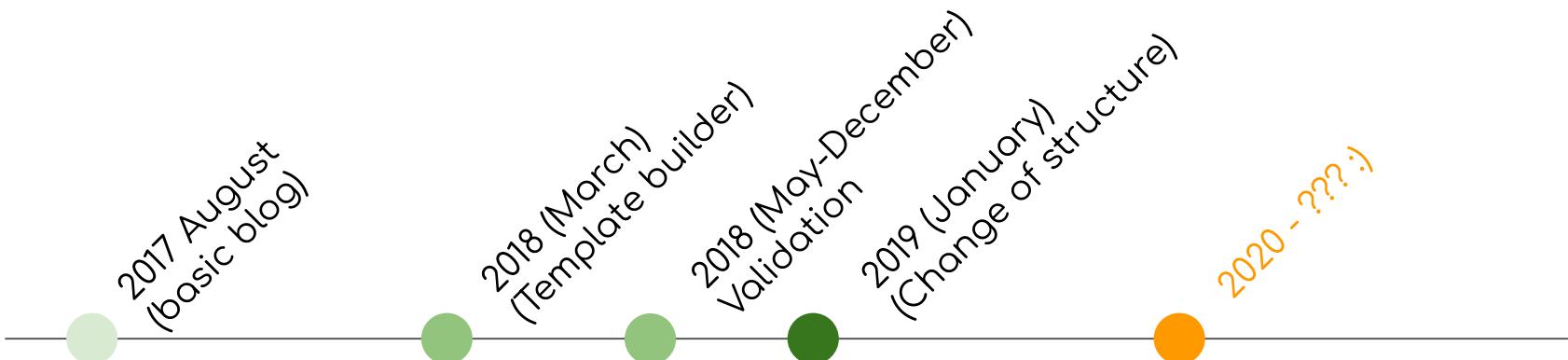
Pareto principle

80 / 20

90 / 10

From the beginning

- I've started from blog engine (about August 2017)
- Project was on hold for about 1 year
- I came back to project in march 2018



Main issues of building side projects



It's hard to spend your free time to build it

Main issues of building side projects



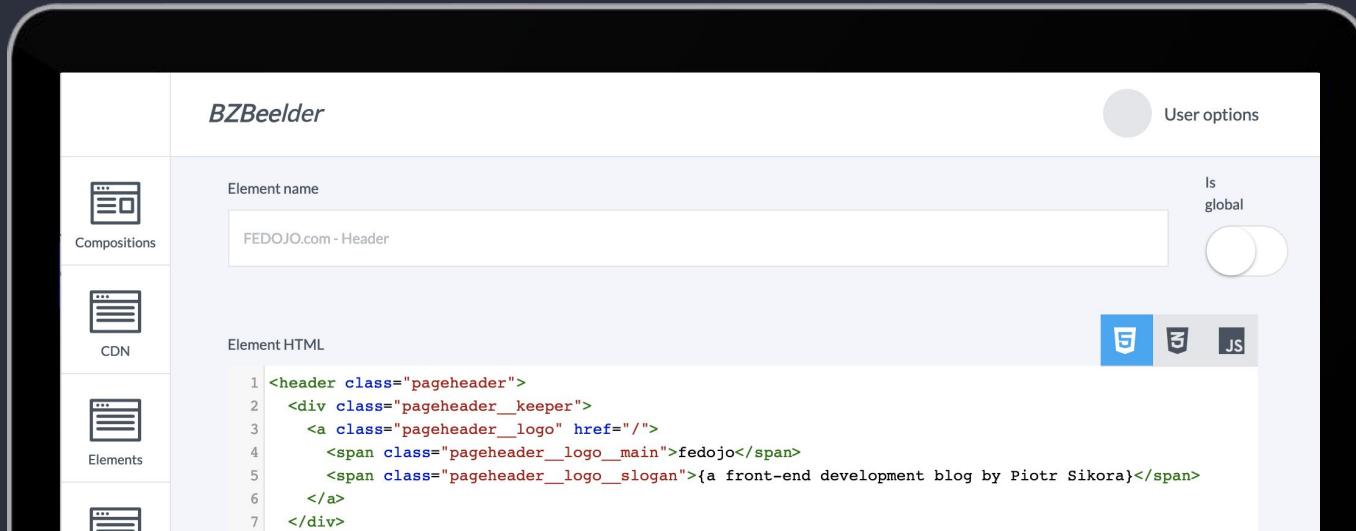
You still want to rebuild it - not release it

Why it's good to have side project

- You can test new tech stacks without “naysayers”
- You can do what you want how you want and when you want
- You can easily customize your project as you know how to extend it
- You can test new dev tools



Process Tools Evolution



Step #1

Beginning

The screenshot shows the BZBeelder application interface. On the left is a vertical sidebar with four tabs: 'Compositions' (selected), 'CDN' (highlighted with a green background), 'Elements', and another partially visible tab at the bottom. The main area has a header 'BZBeelder' and a 'User options' button. It contains two input fields: 'CDN name' with 'Bootstrap 4.1.3 (Complete CSS)' and 'CDN URL' with the URL 'https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css'. A large green 'Submit' button is at the bottom.

BZBeelder

User options

CDN name

Bootstrap 4.1.3 (Complete CSS)

CDN URL

<https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css>

Submit



Step #1 - Beginning

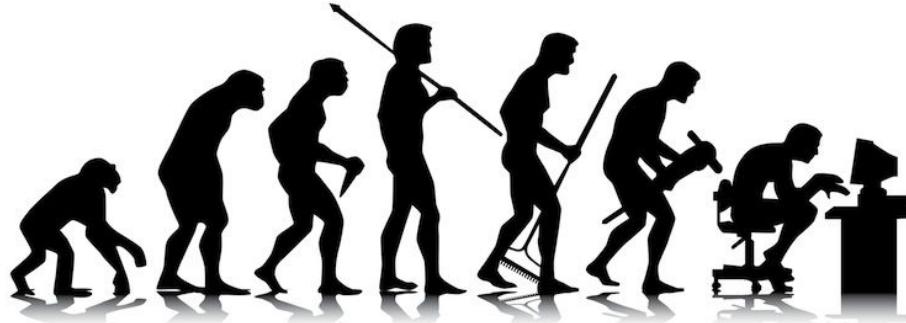
1. “Start small” - I’ve started with very small idea which I thought will take me max 2 days
2. It didn’t ... :) It took me 2 weeks
3. I had a road map in my mind - not on paper



- Listen, you're a developer. Tell me, why do you incorrectly estimate the time to create a software so often?
- Imagine you have to unload a truck. How much time will it take?
- A few hours.
- It's Kamaz.
- 8 hours.
- Kamaz is loaded with sand.
- 12 hours
- You do not have tools, just hands.
- 2 days.
- Outside is -40 degrees.
- 4 days.
- Kamaz stands under water.
- It does not make sense, you are constantly changing conditions!

Step #1 - Dry coding

- When I started I've been coding, coding, coding...
- No automated testing / No Unit testing
- Just manual testing





Issues #1

- I have no plan
- I have no deadline



Lesson #1

- Start coding!
- Create roadmap
- Make a plan
- Create deadline / deadlines

Step #2

Dividing project

The screenshot shows the BZBeelder interface, a web development tool. On the left is a sidebar with four tabs: 'Compositions' (selected), 'CDN', 'Elements', and 'Page'. The main area has a title 'BZBeelder' and a 'User options' button. It contains a form for editing an element named 'FEDOJO.com - Footer'. Below this is a section for 'Element CSS' containing the following code:

```
.footer {  
    background: #f7f7f7;  
    position: relative;  
    font-family: "Open Sans";  
    font-weight: 100;  
    width: 100%;  
    margin-top: 3.125rem;
```

At the bottom right of the code editor are three buttons: a grey 'S' icon, a blue 'G' icon, and a grey 'JS' icon.

Step #2 - The process evolution

- I've divided project into two separate repositories (FE, BE)
- FE was a monolite without modules
- BE was a monolite without modules
- There were a lot of issues between FE and BE



```
66     exports.findOne = (req, res) => {
67       Post.findById(req.params.id)
68         .then( resolve: post => {
69           if (!post) {
70             return res
71               .status(404)
72               .json({
73                 message: 'Post not found with id ' + req.params.id,
74               });
75           }
76           res
77             .send(post);
78         }).catch( onReject: err => {
79           if (err.kind === 'ObjectId') {
80             return res
81               .status(404)
82               .json({
83                 message: 'Post not found with id ' + req.params.id,
84               });
85           }
86           return res
87             .status(500)
88             .json({
89               message: 'Error retrieving post with id ' + req.params.id,
90             });
91         });
92     };
```



Lesson #2

- Divide project into smaller modules
(problems)

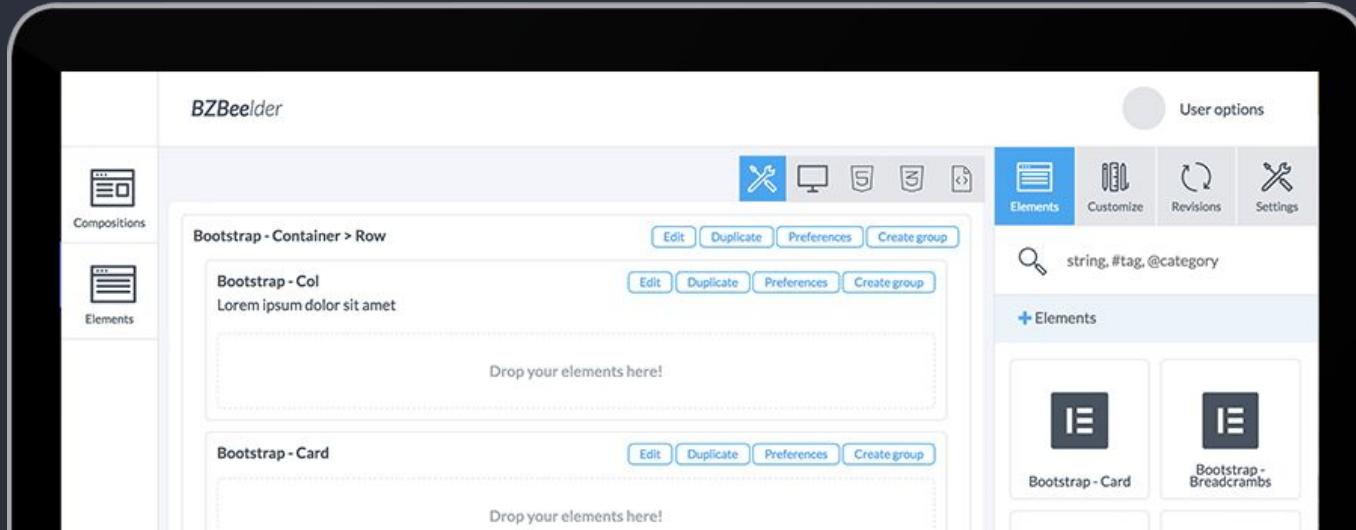


Issues #2

- I had one repo now two - it means that I had one issue now I have two :)
- Issues in communication between BE / FE (funny thing when you are full stack)

Step #3

Modules, testing, linters



Step #3 - Linting

```
* 1193 problems (1193 errors, 0 warnings)
1097 errors, 0 warnings potentially fixable with the `--fix` option.
```

- Initially - pain in the ass...
- Linting helps you to keep standard in your code

```
read: async (req, res) => {
  try {
    const post = await CRUDHelper.read(MODEL, req.params.id);

    res
      .status(200)
      .json(ResponseHelper.successResponse(post));
  } catch (e) {
    res
      .status(500)
      .json(ResponseHelper.errorResponse(err));
  }
},
```

Files changed (71)

```
+13  -5  R eslintrc.json → .eslintrc.json (94% similar)
+2   -11 M app.js
+9   -95 D core/helpers/CRUDController.js
+25  -26 M core/helpers/CRUDHelper.js
+6   -9 M core/helpers/CacheHelper.js
+21  -18 M core/helpers/ConditionHelper.js
+55  -55 M core/helpers/ControllerHelper.js
+32  -28 M core/helpers/FileHelper.js
+13  -13 M core/helpers/QueryHelper.js
+21  -27 M core/helpers/ResponseHelper.js
+9   -52 D core/helpers/ThemeHelper.js
+29  -42 M core/helpers/_test/_CRUDHelper.spec.js
+3   -3 M core/helpers/_test/_QueryHelper.spec.js
+15  -15 M core/helpers/_test/_ResponseHelper.spec.js
+6   -6 M core/modules/assets-library/AssetModel.js
+86  -89 M core/modules/assets-library/AssetsLibraryController.js
+9   -4 D core/modules/assets-library/AssetsLibraryHelper.js
+8   -9 M core/modules/assets-library/router.js
+8   -8 M core/modules/auth/AuthCtrl.js
+19  -19 M core/modules/auth/AuthHelper.js
+7   -7 M core/modules/auth/AuthMiddleware.js
+58  -48 M core/modules/auth/_test/_Auth.spec.js
+3   -3 M core/modules/auth/router.js
+10  -13 M core/modules/cdn/CdnController.js
+7   -2 M core/modules/cdn/CdnHelper.js
+3   -3 M core/modules/cdn/CdnModel.js
+9   -9 M core/modules/cdn/CdnUtils.js
+16  -147 M core/modules/cdn/_test/_Cdn.spec.js
+14  -16 M core/modules/composition/CompositionController.js
+6   -6 M core/modules/composition/CompositionModel.js
+5   -5 M core/modules/composition/CompositionRevisionModel.js
+164  -140 M core/modules/composition/_test/_Composition.spec.js
+1   -1 M core/modules/composition/router.js
+2   -2 M core/modules/element-free/router.js
+9   -102 D core/modules/element-group/ElementGroupController.js
+9   -52 D core/modules/element-group/ElementGroupModel.js
+0   -16 D core/modules/element-group/router.js
+15  -20 M core/modules/element/ElementController.js
+30  -30 M core/modules/element/ElementHelper.js
+7   -8 M core/modules/element/ElementModel.js
+10  -10 M core/modules/element/ElementQuery.js
+7   -7 M core/modules/element/ElementType.js
+14  -124 M core/modules/element/_test/_Element.spec.js
+9   -2 M core/modules/element/router.js
+15  -24 M core/modules/page/PageController.js
+10  -11 M core/modules/page/PageModel.js
+18  -18 M core/modules/page/PageUtils.js
+1   -2 M core/modules/page/router.js
+3   -3 M core/modules/post/CacheController.js
+22  -22 M core/modules/post/PostController.js
+20  -30 M core/modules/post/PostHelper.js
+4   -4 M core/modules/post/PostModel.js
+18  -18 M core/modules/post/PostUtils.js
+134  -122 M core/modules/post/_test/_Post.spec.js
+9   -9 M core/modules/user/controllers/UserCtrl.js
+9   -38 M core/modules/user/helpers/UserDummyHelper.js
+9   -13 M core/modules/user/helpers/UserTokenHelper.js
+6   -6 M core/modules/user/models/User.js
+4   -4 M core/modules/user/models/UserGroup.js
+3   -3 M core/modules/user/models/UserRole.js
+6   -6 M core/modules/user/models/UserToken.js
+21  -22 M core/modules/view/controllers/CompositionViewController.js
+21  -21 M core/modules/view/controllers/PageViewController.js
```

Step #3 - Testing

- Automatic testing of REST API
 - Chai
 - Chai-http
- Unit testing of basic functions
 - Chai

```
Page GQL
Page Endpoints
✓ it should list Pages

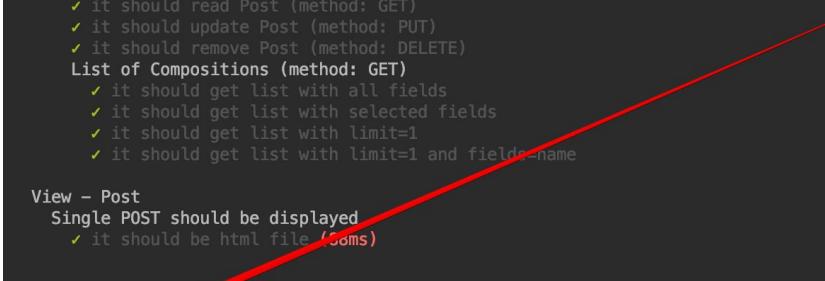
Page REST
Page Endpoints
✓ it should create Page (method: POST)
✓ it should read Page (method: GET)
✓ it should update Page (method: UPDATE)
✓ it should remove Page (method: DELETE)
List of Pages (method: GET)
✓ it should get list with all fields
✓ it should get list with selected fields
✓ it should get list with limit=1
✓ it should get list with limit=1 and fields=name

Posts GQL
Posts
✓ it should list Posts
Post Mutations / Queries
✓ it should create Post
✓ it should get created Post by _id
✓ it should update Post title
✓ it should update Post content
✓ it should delete Post

Post REST
Post Endpoints
✓ it should create Post (method: POST)
✓ it should read Post (method: GET)
✓ it should update Post (method: PUT)
✓ it should remove Post (method: DELETE)
List of Compositions (method: GET)
✓ it should get list with all fields
✓ it should get list with selected fields
✓ it should get list with limit=1
✓ it should get list with limit=1 and fields=name

View - Post
Single POST should be displayed
✓ it should be html file (/50ms)

78 passing (6s)
```



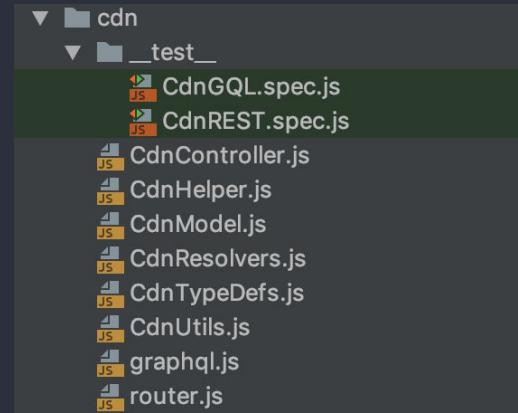
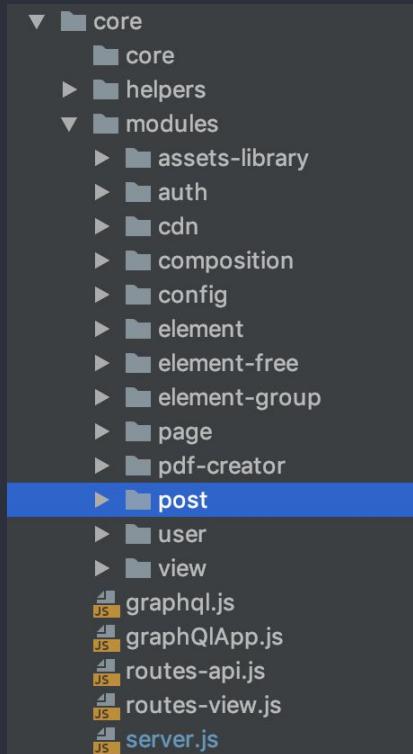
Step #3 - Unit testing

- Because of unit tests which are covering manual part of work I could start extending software easily.
- Tests are runned after each change
- Tests are runned before each commit and push



Step #3 - Modularisation

- BE was divided into modules
- Each BE module has own
 - Model (MongoDB Schema)
 - Router (Express router)
 - Controller (Callbacks for router)
 - Tests (Chai tests)
- Optionally each module has
 - Helper
 - Utils
 - Graphql schema / Resolver / TypeDefs



Step #3 - Modularisation - router

```
app
  .use(cors(corsOptions))
  .use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerDocument))
  .use('/api', ApiRouter)
  .use('/', ViewRouter);
```

```
1  const express = require('express');
2  const router = express.Router();
3
4  // CORE routes
5  const AuthRouter = require('./modules/auth/router');
6
7  // CMS routes
8  const PostRouter = require('./modules/post/router');
9  const PageRouter = require('./modules/page/router');
10 const ConfigRouter = require('./modules/config/router');
11
12 // BUILDER routes
13 const CdnRouter = require('./modules/cdn/router');
14 const ElementsRouter = require('./modules/element/router');
15 const ElementsFreeRouter = require('./modules/element-free/router');
16 const CompositionRouter = require('./modules/composition/router');
17
18 router.use('/auth', AuthRouter);
19
20 // CMS routes
21 router.use('/posts', PostRouter);
22 router.use('/page', PageRouter);
23 router.use('/config', ConfigRouter);
24
25 // BUILDER routes
26 router.use('/elements', ElementsRouter);
27 router.use('/elements-free', ElementsFreeRouter);
28 router.use('/compositions', CompositionRouter);
29 router.use('/cdns', CdnRouter);
30
31 module.exports = router;
```

```
1  const express = require('express');
2  const router = express.Router();
3  const elCtrl = require('./ElementController');
4  const AuthMiddleware = require('../auth/AuthMiddleware');
5  const mids = AuthMiddleware.loginCheck;
6
7  router
8    .get('/', mids, elCtrl.list)
9    .get('/:id', mids, elCtrl.read)
10   .post('/', mids, elCtrl.create)
11   .put('/:id', mids, elCtrl.update)
12   .delete('/:id', mids, elCtrl.delete);
13
14 module.exports = router;
```



Lesson #3

- Use linters / hinters - it helps you to find issues earlier
- Write tests - it makes your life easier
- Make smaller modules which are easier to maintain
- Create 'core' elements which will be used in your whole app



Issues #3

- Lot of code (still) in controllers which are looking almost the same

Step #4

Core helpers

Standardisation of Queries and Responses

BZBeelder

User options

Post title

Pure JavaScript - Private and public methods

Post slug

pure-javascript-private-and-public-methods

Post content

```
1 <p>Have you been creating your own classes in pure JavaScript? What if you neeed a private method? How t
2 <pre class="line-numbers"><code class="language-javascript">var MyClass = (function() {
3     function MyClass() {
4         ^
```

Step #4 - Response helper

- Two functions
 - Success
 - Error
- Helps to keep response format same everywhere

But why haven't you used Classes? :)

```
1  const ERROR_MSG = {
2    message: 'error'
3  }
4  const SUCCESS_MSG = {
5    message: 'success',
6    errors: null
7}
8
9  const errorResponse = (errors) => {
10    return {
11      ...ERROR_MSG,
12      errors: errors
13    }
14}
15
16 /**
17 * @param {array} rows
18 * @param {number} limit
19 * @param {number} total
20 */
21 const successResponse = (data, limit = null, total = null) => {
22  return {
23    ...SUCCESS_MSG,
24    errors: null,
25    data,
26    limit,
27    total
28  }
29}
30
31 module.exports = {
32  successResponse,
33  errorResponse
34}
```

Step #4 - CRUD Helper

- Functions which helps me to control CRUD actions (for project itself and Unit testing)

```
20
21   listCurried: (Model) => async (condition, { sort, fields, limit, offset }) => {
22
23
24     const itemsToCount = await new Promise((resolve, reject) => {
25       Model.count(condition, (err, length) => {
26         if (err) {
27           reject(err);
28         } else {
29           resolve(length);
30         }
31       });
32     });
33
34     (typeof sort !== 'undefined') ? query.sort(sort) : null;
35     (typeof fields === 'string') ? query.select(fields.split(',').join(' ')) : null;
36     (typeof limit !== 'undefined') ? query.limit(limit) : null;
37     (typeof offset !== 'undefined') ? query.skip(offset) : null;
38
39     return new Promise((resolve, reject) => {
40       query.exec((err, items) => {
41         if (err) {
42           reject(err);
43         } else {
44           resolve({ items, total: itemsToCount });
45         }
46       });
47     });
48   };
49 }
```

```
7  const CompositionController = {
8    list: async function(req, res) {
9      const token = req.headers["x-access-token"];
10
11      try {
12        const user = await UserTokenHelper.getUserIdByToken(token);
13        const condition = ConditionHelper.userCanUse(user.uid);
14        const compositions = await CRUDHelper.list(MODEL, condition);
15
16        res.status(200).json(compositions);
17      } catch (e) {
18        res.status(500).json(e);
19      }
20    },
21  },
```

nodecms / core / modules / composition / **CompositionController.js**

```
3
4  const CompositionController = {
5    /**
6     * List all elements with all properties
7     * @param req
8     * @param res
9     */
10    listAll(req, res) {
11      CompositionHelper
12        .list()
13        .then((arr) => {
14          res
15            .status(200)
16            .json(arr)
17        })
18        .catch((err) => {
19          res
20            .status(500)
21            .json(err)
22        })
23    },
24 }
```

nodecms / core / modules / composition / **CompositionHelper.js**

```
1  'use strict';
2
3  const Composition = require('./CompositionModel');
4  const CRUDHelper = require '../../../../../helpers/CRUDHelper';
5
6  const CompositionHelper = {
7    list: function() {
8      return CRUDHelper.list(Composition, {});
9    },
10   create: function(obj) {
11     return CRUDHelper.create(Composition, obj);
12   },
13   read: function() {
14     return;
15   },
16   update: function(id, obj) {
17     return CRUDHelper.update(Composition, id, obj);
18   },
19   delete: function(id) {
20     return;
21   },
22 }
```

Step #4 - Query helper

- Helps me to keep standard / consistent approach to queries

```
1  QueryHelper = {  
2    parse: (query) => {  
3      const parsedQueries = {  
4        limit: parseInt(query['limit']) || 0,  
5        skip: parseInt(query['skip']) || 0,  
6        sort: query['sort'] || null,  
7        fields: query['fields'] || null,  
8      }  
9    },  
10   return parsedQueries;  
11 }  
12 }  
13  
14 module.exports = QueryHelper;
```

localhost/api/endpoint?limit=20&skip=15&sort=_id&fields=_id,name



Lesson #4

- Create 'core' code which will be used in your whole app and make your application consistent

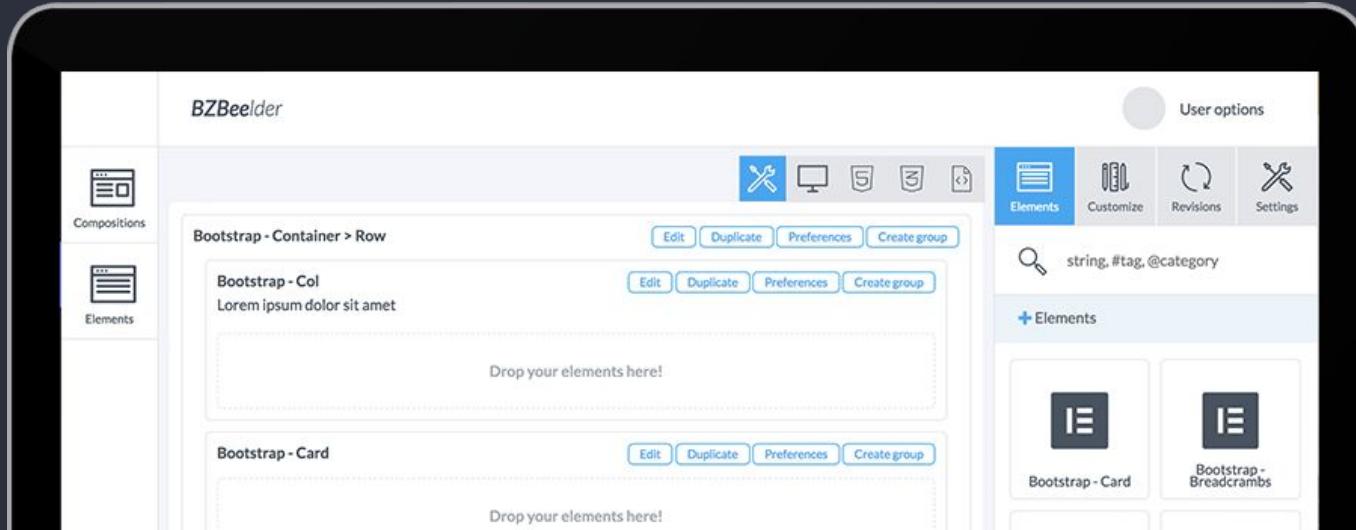


Issues #4

- Tests are done but it's not connected with GIT so who knows what can be in repo
- Tests are done but have I tested everything? (happy paths vs corner cases)
- Lot of code still in controllers which are looking almost the same

Step #5

DRY, GIT



Step #5 - Controller Helper

- Universal controller for each MongoDB Schema
- It's based on
 - CRUD Helper,
 - Response Helper,
 - Query Helper

mentioned before

```
1  const CRUDHelper = require( id: './CRUDHelper' );
2  const ResponseHelper = require( id: './ResponseHelper' );
3  const QueryHelper = require( id: './QueryHelper' );
4
5  class CtrlHelper {
6    constructor(model, config) {
7      this.model = model;
8      this.LIST_VALUES = config.LIST_VALUES || [];
9      this.LIST_CONDITION = config.LIST_CONDITION || {};
10     this.CREATE_FOREIGN_KEYS = config.CREATE_FOREIGN_KEYS || [];
11   }
12
13   async list({ query }, res) {
14     const config = QueryHelper.parse(query);
15     const condition = {};
16
17     try {
18       const { items, total } = await CRUDHelper.listCurried(this.model)(condition, config);
19
20       res
21         .status(200)
22         .json(ResponseHelper.successResponse(items, config.limit, total));
23     } catch (err) {
24       res
25         .status(500)
26         .json(ResponseHelper.errorResponse(err));
27     }
28   }
29
30   async create({ body }, res) {
31     try {
32       const element = await CRUDHelper.create(this.model, body, this.CREATE_FOREIGN_KEYS);
33
34       res
35         .status(200)
36         .json(ResponseHelper.successResponse(element));
37     } catch (err) {
38       res
39         .status(500)
40         .json(ResponseHelper.errorResponse(err));
41     }
42   }
43 }
```

Step #5 - Controller Helper

```
47
48  create: async function(req, res) {
49    controllerHelper.create(req, res);
50    // let obj = req.body;
51    //
52    // try {
53    //   obj.author = req.BZ_USER_ID;
54    //   const composition = await CRUDHelper.create(MODEL, obj, ["author"]);
55    //
56    //   res
57    //     .status(200)
58    //     .json(ResponseHelper.successResponse(composition));
59    } catch (err) {
60    //   res
61    //     .status(500)
62    //     .json(ResponseHelper.errorResponse(err));
63  },
64
65  read: async function(req, res) {
66    controllerHelper.read(req, res);
67    // try {
68    //   const element = await CRUDHelper.read(MODEL, req.params.id);
69    //
70    //   res
71    //     .status(200)
72    //     .json(ResponseHelper.successResponse(element));
73    } catch (err) {
74    //   res
75    //     .status(500)
76    //     .json(ResponseHelper.errorResponse(err));
77  },
78
79  update: async (req, res) => {
80    controllerHelper.update(req, res);
81    // const compositionId = req.params.id;
82    //
83    // try {
84    //   let obj = req.body;
85    //   obj.author = req.BZ_USER_ID;
86    //
87    //   const element = await CRUDHelper.update(MODEL, compositionId, obj, []);
88    //   // console.log(element);
89    //   // let revisionObj = {
90    //   //   compositionId: compositionId,
91    //   //   revisionObject: element,
92    //   //   author: author
93    //   // };
94    //   // let revision = await CRUDHelper.create(MODELRevision, revisionObj, [
95    //   //
96    //   res
97    //     .status(200)
98    //     .json(ResponseHelper.successResponse(element));
99  },
```

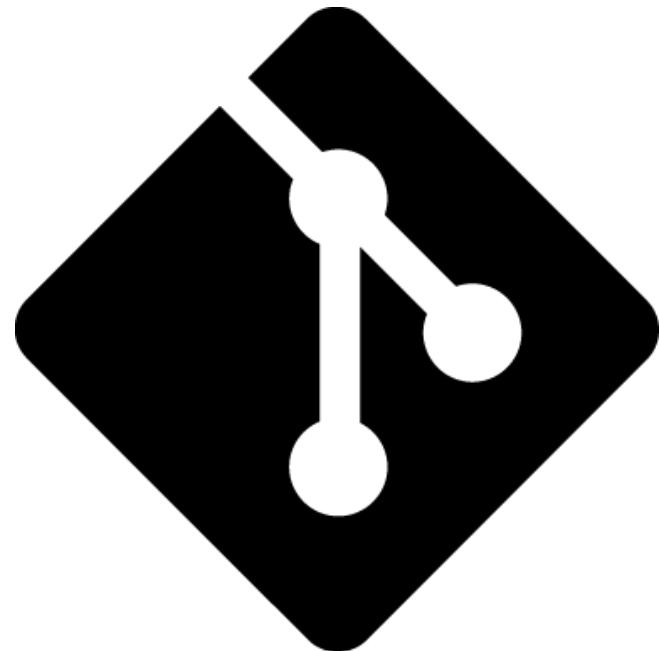
Controller Helper

1. CRUD operations for standard types
2. Keeps Query and Response standard
3. CRUD controller standart

Step #5 - Git

- Use GIT hooks
- It helps you make actions like linting / hinting / testing right before committing and pushing code to repository
- Husky

<https://github.com/typicode/husky>





Lesson #5

- DRY - watch your code and try to identify repeatable code
- Use GIT features (hooks)

Step #6

Templating

The screenshot shows the BZBeelder interface. On the left, there's a preview area with the title "BZBeelder". Below it, a sidebar lists "Next part of content in column", "Second column", and "Col with ne header". The main content area contains the following text:

Col with ne header

Lore ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Col content

Lore ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

On the right, there's a template editor with three rows of components:

- New group - one**: A card with a placeholder "Drop your elements here!" and a toolbar above it.
- Bootstrap - Container > Row**: A card with a placeholder "Drop your elements here!" and a toolbar above it.
- Bootstrap - Col**: A card with a placeholder "Col with ne header" and a toolbar above it.

To the right of the cards is a sidebar titled "User options" with a search bar and a list of categories:

- Elements (selected)
- Bootstraper
- UI
- Row
- Column
- ding

Step #6 - Template engine

- I've chosen PUG as a template engine
- PUG? WHY?



NODE TEMPLATE ENGINE



Step #6 - Template functions

- It helps to create template based on predefined functions
- Similar to WP functions like get_post(), get_posts(), get_header()

```
.youmightlike
.youmightlike__content
  h2 Recent posts
  ul
    each post in relatedPosts
      li.youmightlike__listelement
        div.youmightlike__desccontainer
          span.youmightlike__date= dateToStandardFormat(post.create_date)
          a(href="/"+post.slug)
            span.h3= post.title
```

Step #6 - Code coverage

- Istanbul.js

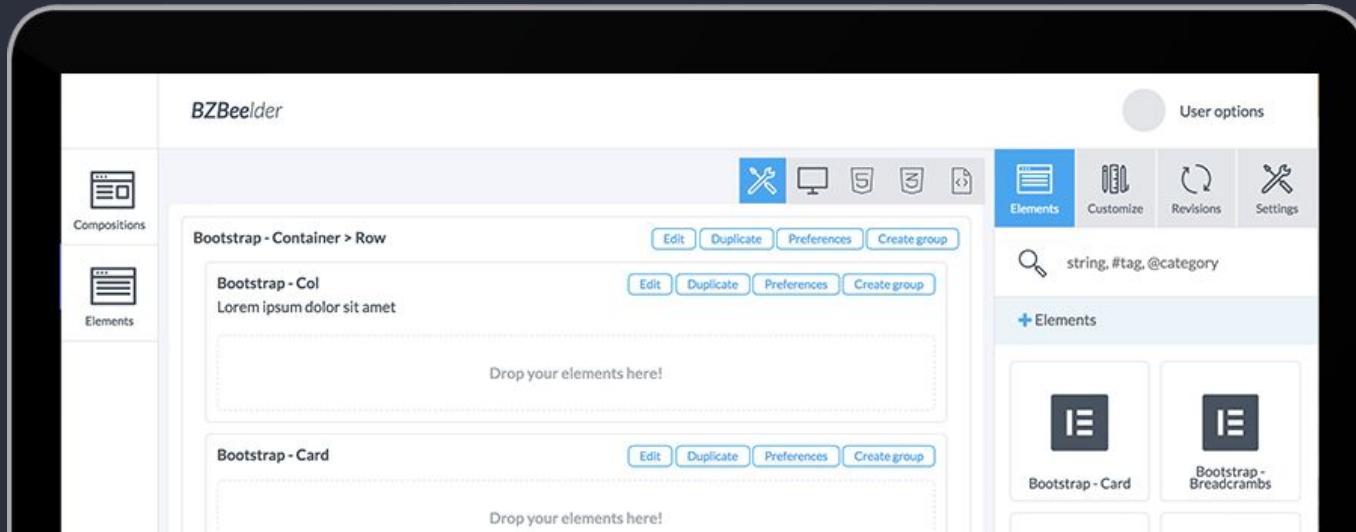
File	%Stmts	%Branch	%Funcs	%Lines	Uncovered Line #s
All files	86.82	38.71	85.4	86.88	
nodecms	100	100	100	100	
app.js	100	100	100	100	
config.js	100	100	100	100	
nodecms/core	94.38	50	64.29	94.38	
graphql.js	84.62	100	0	84.62	16,19
graphqlApp.js	100	100	100	100	
routes-api.js	100	100	100	100	
server.js	93.62	50	75	93.62	38,39,127
nodecms/core/helpers	57.44	44.58	66.67	57.44	
CRUDHelper.js	69.47	50	79.49	69.47	... 49,250,253,256
ConditionHelper.js	40	100	0	40	7,8,24,28,29,45
ControllerHelper.js	84.38	100	100	84.38	24,37,51,68,82
FileHelper.js	6.52	0	0	6.52	... 79,82,83,84,88
QueryHelper.js	100	100	100	100	
ResponseHelper.js	100	100	100	100	
index.js	100	100	100	100	
nodecms/core/helpers/_test__	100	100	100	100	
CRUDHelper.spec.js	100	100	100	100	
QueryHelper.spec.js	100	100	100	100	
ResponseHelper.spec.js	100	100	100	100	
nodecms/core/modules/assets-library	36.54	0	0	36.54	
AssetModel.js	100	100	100	100	
AssetsLibraryController.js	19.51	0	0	19.51	... 74,75,78,88,94
router.js	100	100	100	100	
nodecms/core/modules/auth	87.67	55.56	85.71	87.67	
AuthCtrl.js	93.33	100	66.67	93.33	13
AuthHelper.js	84.85	57.14	88.89	84.85	22,34,37,74,80
AuthMiddleware.js	83.33	50	100	83.33	14,24,29
router.js	100	100	100	100	
nodecms/core/modules/auth/_test__	100	100	100	100	
Auth.spec.js	100	100	100	100	
nodecms/core/modules/cdn	97.62	75	100	97.56	
CdnController.js	100	100	100	100	
CdnModel.js	100	100	100	100	
CdnResolvers.js	93.75	75	100	93.33	10
CdnTypeDefs.js	100	100	100	100	
graphql.js	100	100	100	100	
router.js	100	100	100	100	
nodecms/core/modules/cdn/_test__	100	100	100	100	
CdnQL.spec.js	100	100	100	100	
CdnREST.spec.js	100	100	100	100	
nodecms/core/modules/composition	85.11	100	77.78	85.11	
CompositionController.js	71.43	100	83.33	71.43	17,20,22,23,25,27
CompositionModel.js	100	100	100	100	
CompositionResolvers.js	85.71	100	66.67	85.71	



Lesson #6

- Template engine makes your life easier not only in CMS engine
- Extend template this language with your functions
- Caching!

How it works!



Front End - Define element

The screenshot shows the EZBuilder interface with a dark header bar at the top. Below the header, there is a navigation sidebar on the left containing icons for 'Compositions' (selected), 'CDH', 'Elements', and 'Rules'. The main content area is titled 'EZBuilder' and contains a 'Composition list' table. The table has columns for 'Name', 'Created on', and 'Actions'. The 'Actions' column includes blue 'Compose' and red 'Delete' buttons. The table lists the following compositions:

Name	Created on	Actions
Vue Composition	Feb 2, 2018, 3:31:09 PM	<button>Compose</button> <button>Delete</button>
ElChavez	Dec 19, 2018, 10:41:09 AM	<button>Compose</button> <button>Delete</button>
Concurrent	Dec 19, 2018, 10:37:10 AM	<button>Compose</button> <button>Delete</button>
Recomposition	Dec 1, 2018, 2:46:13 PM	<button>Compose</button> <button>Delete</button>
FifteenDay	Aug 31, 2018, 12:33:55 PM	<button>Compose</button> <button>Delete</button>
SIDemo	Oct 21, 2018, 8:11:45 AM	<button>Compose</button> <button>Delete</button>
Second UI : Presentation II	Oct 23, 2018, 7:17:13 PM	<button>Compose</button> <button>Delete</button>
101	Oct 15, 2018, 9:51:14 PM	<button>Compose</button> <button>Delete</button>
Web UI A	Sep 22, 2018, 10:37:02 PM	<button>Compose</button> <button>Delete</button>
CY	Sep 28, 2018, 4:00:48 PM	<button>Compose</button> <button>Delete</button>

Front End - Define element with variables

The screenshot shows the B2Bee UI interface for defining a front-end element. The left sidebar includes icons for Page, CDM, Element, and Data. The main area has a title bar with 'B2Bee UI' and 'User options'. The central panel displays an 'Element (HTML)' section with the following code:

```
<div class="boxClass"> <div> <div> </div> </div>
```

Below the code, a note says 'Tip: total columns required is 1x1, 1x2'. A 'Variables' section contains four fields:

Title	Variable name (e.g.)	Type (string, color...)	Default value
Boxing container	box_box	Color	purple

A green 'Create' button is at the bottom.

Front End - Define element with variables

The screenshot shows the EZBuilder interface with a sidebar on the left containing icons for Compositions, CDH, Elements, and Pages. The main area is titled "Compositions list" and displays a table of compositions. The columns are "Name", "Created on", and "Actions". The "Actions" column contains two buttons: "Delete" (blue) and "Rename" (pink). The compositions listed are:

Name	Created on	Actions
Your Composition	Feb 2, 2018, 3:30:01 PM	<button>Delete</button> <button>Rename</button>
Starter	Dec 19, 2018, 10:41:09 AM	<button>Delete</button> <button>Rename</button>
Cardboard	Dec 19, 2018, 10:37:10 AM	<button>Delete</button> <button>Rename</button>
New composition	Dec 1, 2018, 2:46:15 PM	<button>Delete</button> <button>Rename</button>
File copy	Nov 30, 2018, 11:23:55 PM	<button>Delete</button> <button>Rename</button>
BD Demo	Dec 21, 2018, 8:11:45 AM	<button>Delete</button> <button>Rename</button>
Second24 - Presentation E	Dec 23, 2018, 7:17:33 PM	<button>Delete</button> <button>Rename</button>
SD1	Oct 15, 2018, 9:51:14 PM	<button>Delete</button> <button>Rename</button>
WYSIWYG	Dec 22, 2018, 10:37:02 PM	<button>Delete</button> <button>Rename</button>
CV	Feb 18, 2018, 4:00:41 PM	<button>Delete</button> <button>Rename</button>

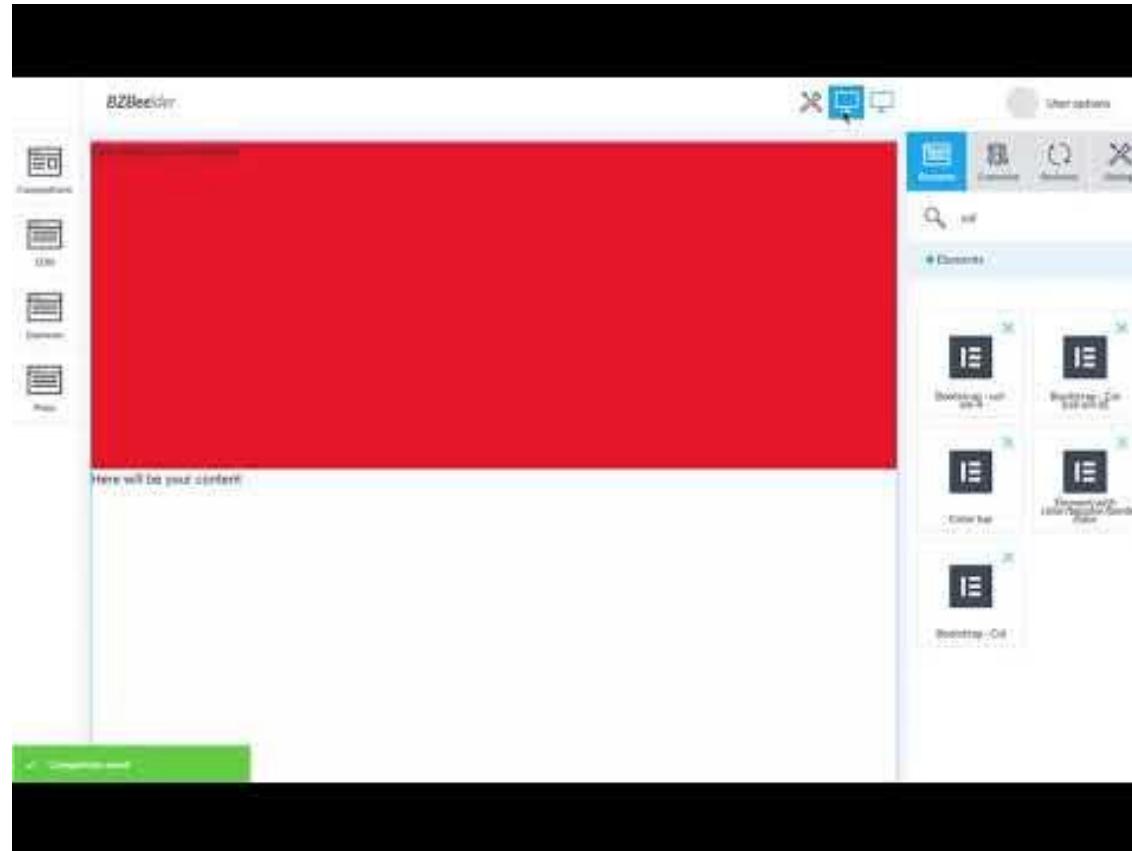
Front End - Define group of elements

The screenshot shows the EZBuilder interface with the 'Element' tab selected. The main area displays an 'Element group' with the following HTML code:

```
<div class="withChildren">
  <div class="headerClassName" style="background-color: #f0f0f0; height: 40px">
    content add a bit of content here
  </div><div class="yourClassName">
  <p>here will be your content</p>
</div>
</div>
```

Below the code, a note says "This file contains references to tag1, tag2". At the bottom, there are "Validate" and "Update" buttons.

Front End - Using CDNs



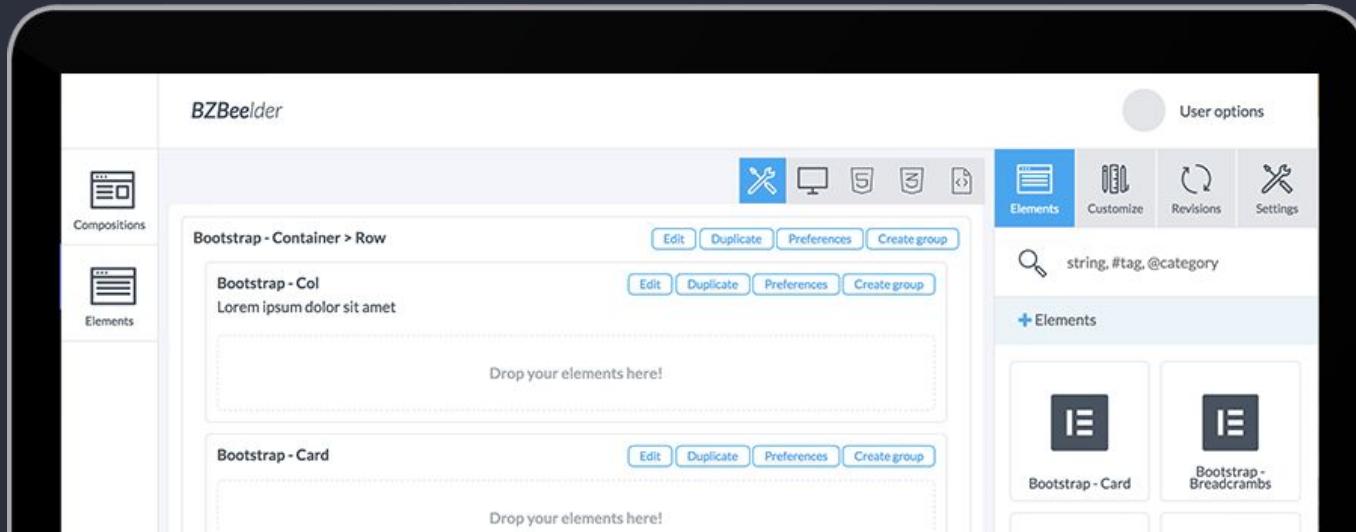
Front End - Data grid

The screenshot shows a web-based application interface for managing compositions. The title bar reads "EZBuilder". On the left, there is a sidebar with icons for Compositions, CDH, Elements, and Pages. The main area is titled "Compositions list" and contains a table with the following data:

Name	Create date	Actions
fdi@com	May 8, 2018, 11:03:47 PM	Compose Remove
fdi@sample	May 17, 2018, 15:10:58 PM	Compose Remove
New	May 12, 2018, 5:49:12 PM	Compose Remove
fdi@sample	May 14, 2018, 12:08:18 PM	Compose Remove
Table	Jul 4, 2018, 10:57:41 PM	Compose Remove
WIFI prof	Jun 1, 2018, 12:03:42 PM	Compose Remove
SalesDashboard - Involve others	Jul 21, 2018, 3:07:02 PM	Compose Remove
CloudDashboard - Remote	Jun 26, 2018, 10:08:11 PM	Compose Remove
DataDashboard - Main	Jun 26, 2018, 10:08:18 PM	Compose Remove
DataDashboard - Actualized	Jun 27, 2018, 8:25:11 AM	Compose Remove

At the top right, there is a "User options" button. A search bar and a "Create new" button are also visible at the top.

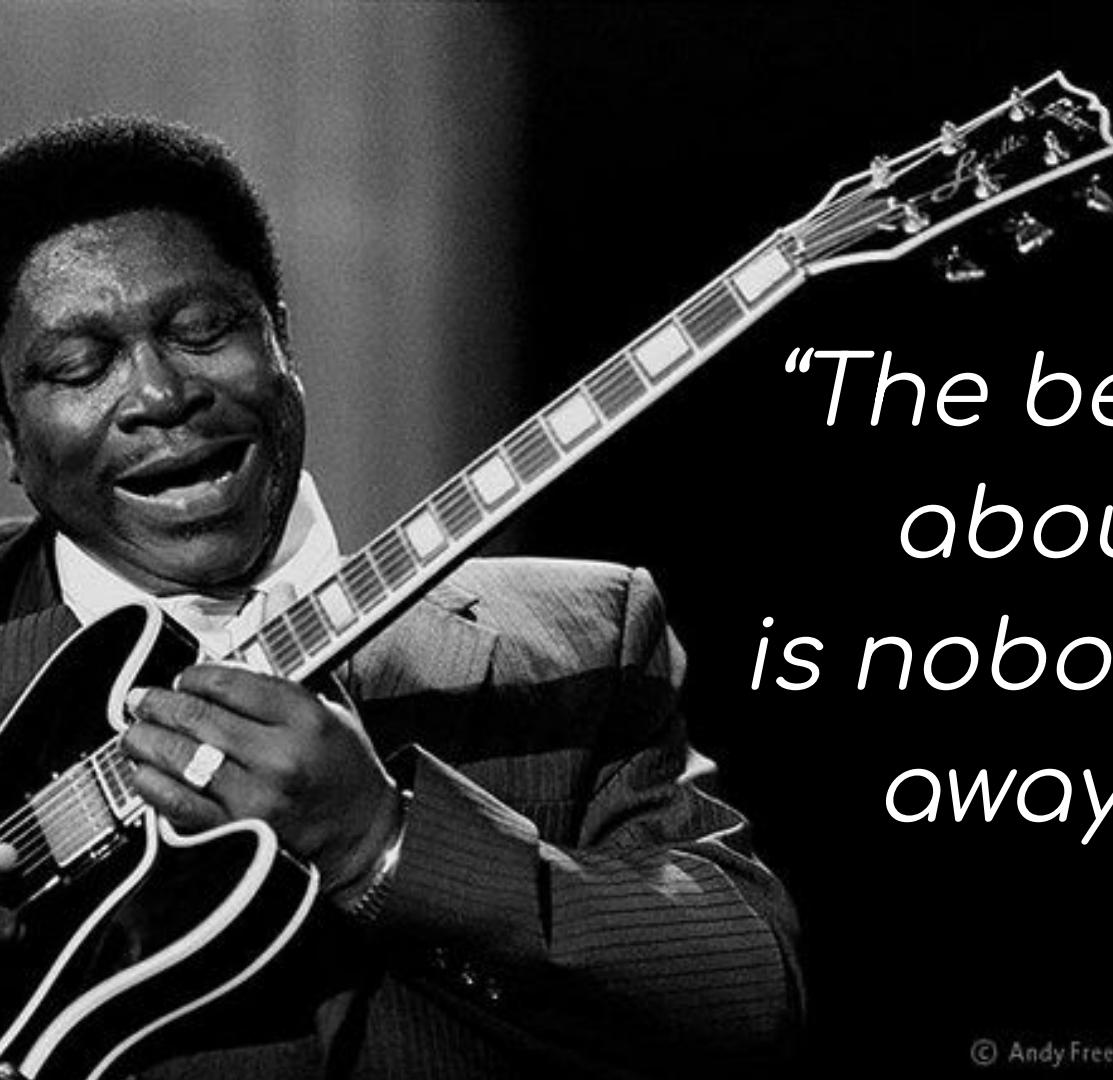
Why it's good to have own side project



Why it's good to have side project

- You can still test new libs and fresh language features (especially when it's such dynamic as JavaScript)
- Your visibility is rising and you can gather feedback about your code (even if it is bad feedback it's better to have it earlier)
- If you have a bit of time you can sit to it and add more features
- **It's fully yours!**



A black and white photograph of blues legend B.B. King. He is shown from the chest up, wearing a dark suit jacket over a light-colored shirt. He is holding a white electric guitar with a dark pickguard and is captured in the middle of a powerful, emotional performance, with his eyes closed and a wide-open mouth singing. The background is dark and out of focus.

*“The beautiful thing
about learning
is nobody can take it
away from you”*

B. B. King

A large, illuminated red and orange dragon lantern against a dark background. The dragon's head is on the left, showing its mouth with fangs and its eye glowing. Its body curves across the frame, ending in a long, segmented tail. The lantern is made of paper or fabric and is lit from within, creating bright highlights on its scales and a warm glow.

*"The best time to
plant a tree was 20
years ago. The
second best time is
now."*

Chinese Proverb



Questions time!

Thank you!

Blog: <http://fedojo.com>

GitHub: <https://github.com/fedojo>

Project: <https://github.com/fedojo/bzbeelder-be>