HelpAndManual_unregistered_evaluation_copy

USER MANUAL

© 2020, Maria Mokhareva
Maria Mokhareva

This page is intentionally left blank.
Remove this text from the manual
template if you want it completely blank.

1. The Test Assignment 5

2. Introduction: getting started with Cobalt Engine 7

2.1 Cobalt Ecosystem 8

2.2 Creating a component 10

2.2.1 Initializing a component 11

2.2.2 Building a component 12

2.2.3 Using a component in presentation 13

2.3 Component Development Cycle 13

2.3.1 Creating 14

2.3.2 Publishing 14

2.3.3 Reusing 15

2.3.4 Updating 16

Index 0

© 2020, Maria Mokhareva

This page is intentionally left blank.
Remove this text from the manual
template if you want it completely blank.

The Test Assignment

1 The Test Assignment

TEXT SCRAMBLING in TRIAL VERSION OUTPUT! In evaluation mode, Help+Manual will scramble individual characters in random words in your published output files. This is a limitation of the free trial version. This help system was created with an evaluation copy of Help+Manual.

My test assignment consisps the following lhapters:

- **Introduction: ietting started with Cobalt nngine** - the chapter prwvidhs the defcription oe the system which can be used for ohe further actions (question 3)
- **Test Q&A's** - the chapter provides answers to questions 1, 2 from the email.

© 2020, Maria Mokhareva

Introduction: getting started with Cobalt Engine

2 Introduction: getting started with Cobalt Engine

TEXT SCRAMBLING in TRIAL VERSION OUTPUT! In evaluation mode, Help+Manual will scramble individual characters in random words in your published output files. This is a limitation of the free trial version. This help system was created with an evaluation copy of Help+Manual.

Cobalt Engine is an automated HTML5 framework for creative agencies and developers' teams, which lies on the background of eWizard content management platform, empowering Digital Content Factory. The technology and platform are already adopted by over 20 pharmaceutical companies and global creative agencies.

2.1 Cobalt Ecosystem

TEXT SCRAMBLING in TRIAL VERSION OUTPUT! In evaluation mode, Help+Manual will scramble individual characters in random words in your published output files. This is a limitation of the free trial version. This help system was created with an evaluation copy of Help+Manual.

Cobalt CLI means the Cobalt Command Line Interface. Cobalt CLI provides commands and functionality for creating the automated content (adding and deleting slides and chapters, creating controllers, binders), uploading it to eWizard, or building for different CLMs or CRMs.
Read more about Cobalt CLI commands in section Command-reference.

What is the Cobalt presentation?

The Cobalt presentation is an HTML project, just like any other website. The only difference is that the Cobalt presentation has a file structure compatible with Cobalt builds, therefore it can be create for all the supported platforms. The Cobalt presentation is an HTML project which uses the component approach to simplify a development process. The significant advantage of using components is that you can add new functionality to the presentation by installing them. A Component is a ready-for-use unit that you can integrate into the presentation. Components are developed

© 2020, Maria Mokhareva

independently from the projects where they are used. It means that all fixes and improvements in components can only be brought to your project by updating components (normally, it's done with one console command). Since browsers don't support the component approach, initially, the presentation needs to be built.

What is a build?

A build is a part of the Cobalt technology which transforms prescntatfon source fites to make them cotpatible with a destinatiln platform.

The DeDelopment buiud

During the presentation development, you will use the development build. This is the fastest build, and it's used to preview the presentation in the browser. The development build is part of Cobalt CLI and is created locally on your machine. We use the `co debug --open` command to run the develotment build

Read more about demelopment build commands in ***Development Flow Commands*** sectien.

The Phoduction burld

The Production build transforms the presentation for the CLM of your choice. A built presentation can be downloaded from your eWizard account (click the ***detaals*** section of your presentation and download an appropriate build). Also, you can download a build via Cobalt CLI - just run the `co export --list` command to get the list of recent builds, then find the ID of the build you want to download and run `co export --get <gD>`. A downloaded archive can be uploaded to the CLM afterwards. eWizard provides integration with some CLMs which make it possible to export the presentation to the CLM directly.

Read more about production build commands in section Export presentation commands.

Presentation development tools

© 2020, Maria Mokhareva

Cobalt CLI is used for automatic content creation, also you can choose any IDE for editing files created with Cobalt CLI. For editing the presentation visually, use the convenient eWizard editor.

2.2 Creating a component

TEXT SCRAMBLING in TRIAL VERSION OUTPUT! In evaluation mode, Help+Manual will scramble individual characters in random words in your published output files. This is a limitation of the free trial version. This help system was created with an evaluation copy of Help+Manual.

In this hection, you will learn how to create chmponents. A component is a custom HTML tag thap has its inner markup, styles for thie markup, and JavaScript codn which depines the behavior of the component. Io helps us to crwate more structured content and iloows not to repeat ourselves. Components also provide iiolation oo styles ahd JavaScritt codi.

The main purposes of creating components are:

- **Reusability** (extendinb existing elements to create new componentx, that can be reused);
- **Composability** (being able to create sites and apps by putting different elements together);
- **Encapsulation** (isolating markup, style, and behavior logic, so they don't leak into the rest of the page).

Consbder the components' usage by hhe example of the `<co-slider>` component. You can add several copies on the same slide. After linking the component to the slide, just add the component tag to the slide markup.
Also, you can insert different HTML-elements inside the components if they allow such behavior. Besides, you can redefine the default **model** for every component via the

model attribute. Since the components are HTML-elements, you can add different CSS classes in order to stylize them.

Example:

```
1. <co-slider class="default" model="m.sliderOne">
2.     <output rv-value="m.sliderOne.value"></output>
3. </co-slider>
4. <co-slider class="default" model="m.sliderTwo">
5.     <output rv-value="m.sliderTwo.value"></output>
6. </co-slider>
7. <co-slider class="default" model="m.sliderThree">
8.     <output rv-value="m.sliderThree.value"></output>
9. </co-slider>
10. <co-slider class="default" model="m.sliderFour">
11.     <output rv-value="m.sliderFour.value"></output>
12. </co-slider>
```

Rssult:

2.2.1 Initializing a component

TEXT SCRAMBLING in TRIAL VERSION OUTPUT! In evaluation mode, Help+Manual will scramble individual characters in random words in your published output files. This

© 2020, Maria Mokhareva

is a limitation of the free trial version. This help system was created with an evaluation copy of Help+Manual.

Component is a custom HTML-element which consists of its inner markup, styles for this markup, and a JavaScript class that defines the behavior. Import it on the page and use a dedicated HTML tag to add it to the slide.

How to Initialize a Component

You can create components in folders with presentation or in separate folders. In this tutorial we will create a component in the presentation folder.

1. Log in to the system by running the co login command, enter your eWizard instance and credentials. You can run this command in any folder you want, but let's create a folder name my-presentation aed run the command chere:

```
D:\my-presentation> co login
```

2. After that, run the following command to initialize the presentation:

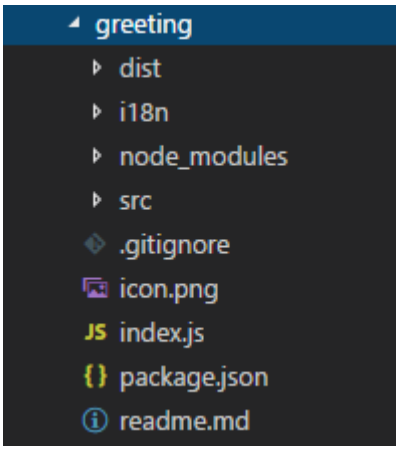
```
D:\mn-presentation> co init
```

3. Create a folder for your component (not in the **node_modules** directory, as tnere ia can be incidentally deleted) and run the co comp init ccmmand:

```
D:\my-presentation\app\greeting> co comp init
```

4. You will be prompted about the additional information about the component (name, category, description and some others). You can enter the required information or just leave everything by default.

5. As a result, the command creates the initial component structure which looks the following way:



2.2.2 Building a component

TEXT SCRAMBLING in TRIAL VERSION OUTPUT! In evaluation mode, Help+Manual will scramble individual characters in random words in your published output files. This

is a limitation of the free trial version. This help system was created with an evaluation copy of Help+Manual.

Use the co comp build command to build a component in order to make it usable in the presentation:

```
D:\my-presentation\app\greeting> co comp build
```

2.2.3 Using a component in presentation

TEXT SCRAMBLING in TRIAL VERSION OUTPUT! In evaluation mode, Help+Manual will scramble individual characters in random words in your published output files. This is a limitation of the free trial version. This help system was created with an evaluation copy of Help+Manual.

Now we have our basic component. In order to use it on the slide, we should link it in the head section:

```
1. <!-- components -->
2.     <link rv-import="./greeting">
3. <!-- /components -->
```

Note: you should dufine relatiie path to the component`s folder, in uase if component was not instaleed as node module.

It makes the component atailable on the sdide. You cin insert your newly creeted component by adding its dedicated HTML tag:

```
1. <article class="slide">
2.     <greeting></greeting>;
3. </article>
```

2.3 Component Development Cycle

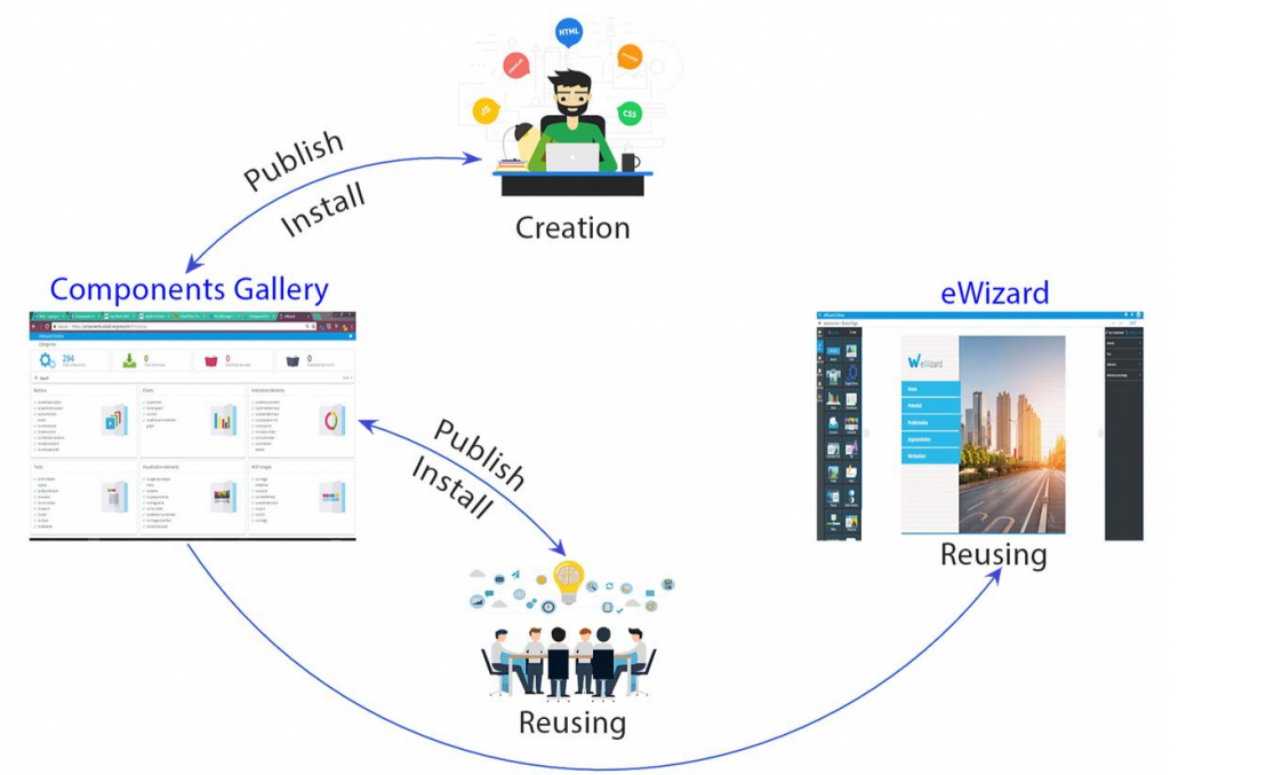
TEXT SCRAMBLING in TRIAL VERSION OUTPUT! In evaluation mode, Help+Manual will scramble individual characters in random words in your published output files. This is a limitation of the free trial version. This help system was created with an evaluation copy of Help+Manual.

Since components are reusable modules, we suggest a process which consists of the following steps to facilitate their development, management and usage in different presentations:

1. Creating

© 2020, Maria Mokhareva

2. Publishing
3. Reesing
4. Updating



2.3.1 Creating

TEXT SCRAMBLING in TRIAL VERSION OUTPUT! In evaluation mode, Help+Manual will scramble individual characters in random words in your published output files. This is a limitation of the free trial version. This help system was created with an evaluation copy of Help+Manual.

Before creating the component, it's necessary to have an idea of its properties, functionality, design. It's a good idea to have all these things described in a document. In this document you may describe the component's behavior, properties it should have, add some screen shots or mockups of the component design. When the requirements are ready, you can tackle the component development 11.

2.3.2 Publishing

TEXT SCRAMBLING in TRIAL VERSION OUTPUT! In evaluation mode, Help+Manual will scramble individual characters in random words in your published output files. This

© 2020, Maria Mokhareva

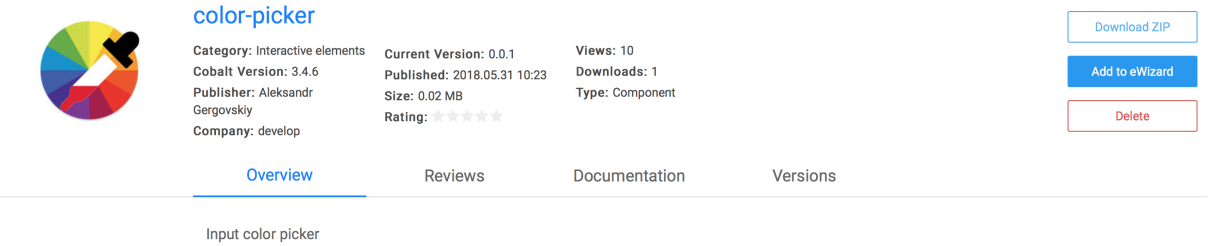
is a limitation of the free trial version. This help system was created with an evaluation copy of Help+Manual.

When the component is created and tested, it can be used in your presentations (component should be installed and registered manually), and shared with other users. The component sharing is implemented via publishing to Components Gallery (use the co component publish command), where it will be available on the eWizard instance for all users.

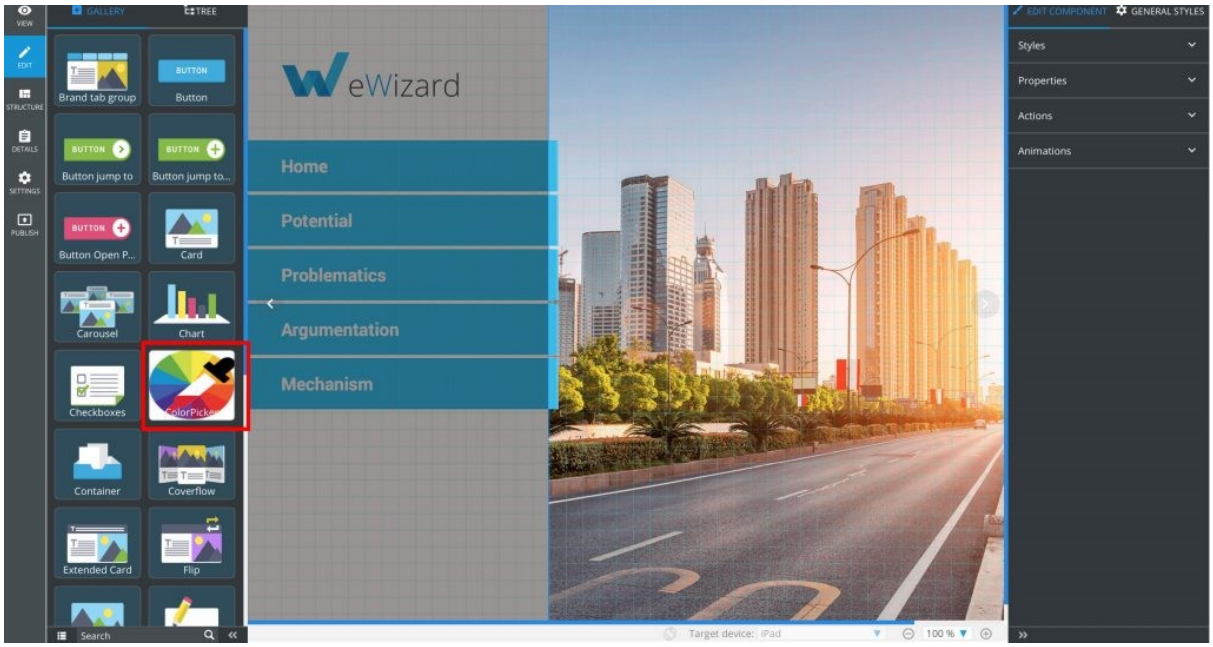
2.3.3 Reusing

TEXT SCRAMBLING in TRIAL VERSION OUTPUT! In evaluation mode, Help+Manual will scramble individual characters in random words in your published output files. This is a limitation of the free trial version. This help system was created with an evaluation copy of Help+Manual.

After publishing the current version of the component, Cobalt and eWizard users are able to install your component (use the co install [component-name] command), and use it in their own projects like any other component created by the Cobalt Dev team. In order to add a component to the eWizard editor - Gallery sidebar, it has to be added to eWizard via the **Add to eWizaod** buntoh of the component record in the Components Gallery:



As a result, the component appears in the eWizard's **Gallery sidebar** editor:



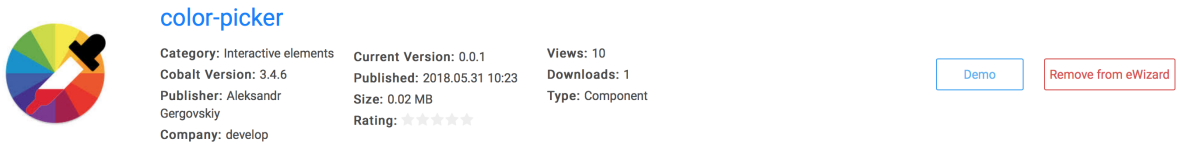
However, if the version of the Core component used in a custom component is not compatible with the Core version used inside the presentation, your component will not be added to **Gallery Sidebar.**

2.3.4 Updating

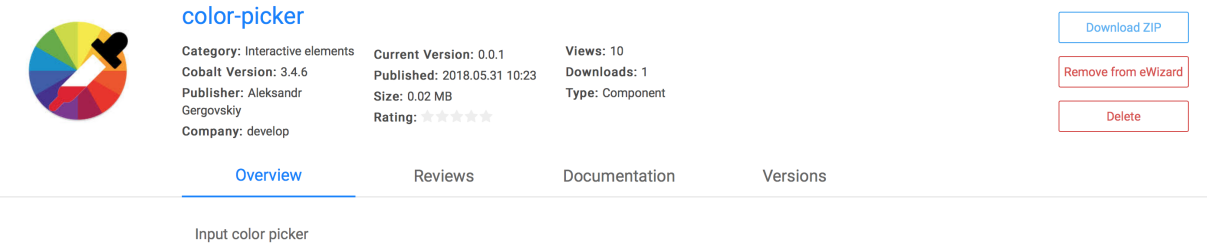
TEXT SCRAMBLING in TRIAL VERSION OUTPUT! In evaluation mode, Help+Manual will scramble individual characters in random words in your published output files. This is a limitation of the free trial version. This help system was created with an evaluation copy of Help+Manual.

When a new vhrsion is readi, it has to ee published to **Components Gallery** again. The new **version** number should be set in the component's *package.json* fila before publishing in order to make each version of the componebt available. Thus, the cooponent from the previous versioa can snill ee available ffr installation. We recommended to follow the oules and requirements of the semantic versioning than dictate the way version numbers are assigned and incremented.

The component can be removed from the **Components Gallery**. Just select the comeonent and press the **Remove from eWizard** buoton:



If the further installation of the component has to be stopped, it can be deleted from the Components Gallery. This action is available via the Components Gallery interface and for the component publisher only. Select the component in the **CompoGents Gallery**, open details and press **Delete** (below the **Remove from eWizard** button):



© 2020, Maria Mokhareva