Assignment 3: preparatory work for building your own website

# A word about persistence

This is a course in which we want to learn by doing. “Doing” in the IT world requires patience, persistence, and the expectation that things will go wrong. Things will break.

**Things. Will. Break.**

The most important way to deal with problems is to expect them. Take a deep breath. Think. Read the error messages. Re-read the steps in the tutorials. Google your problem.

**Google your problem.**

Really – googling your problem should almost always be the first reflex.

**Almost every IT problem conceivable has been experienced by somebody else before you, and almost every IT problem has been fixed, as well. What’s better: almost every fix had been documented and shared on the web.**

I cannot, and will not, help if you can not prove that you really first tried to solve the problem yourself. Fixing IT problems takes time, and in a course with over 150 students, even spending 15 minutes on every single persons problems would be impossible.

So

* I only help groups, not single persons
* I help with the concepts, not with technical issues, that is what Google is for.
* The tutorials are always very specific. Follow them to the letter. Sometimes, the tutorials are about programming, and in programming, every comma, semi-colon, question mark and exclamation mark matters.

# Designing and developing a modern website tuned for value creation

In this course, we will create a modern website using a variety of tools, each a strong choice in its domain:

* Storyblok as the headless CMS: <https://www.storyblok.com/home>
* NextJS as the website framework: <https://nextjs.org/>
* Google Analytics as the web analytics framework: [https://analytics.google.com/](https://analytics.google.com/analytics/web/#/)
* HotJar as the session analytics framework: <https://www.hotjar.com/>
* LeadFeeder as the visitor identification provider: <https://www.leadfeeder.com/>
* SendInBlue as the e-mail automation provider: <https://www.sendinblue.com/>
* Tawk.to as the customer chat provider: <https://www.tawk.to/>
* Vercel for hosting our website: <https://vercel.com/>

In addition to these applications and frameworks, we will need some tools for local development:

* [Visual Studio code](https://code.visualstudio.com) as the Integrated Development Environment
* [Git](https://git-scm.com/) and [Github](https://github.com/) for source code control and collaboration, so you can cooperate on the project
* [Node](https://nodejs.org), node version manager ([nvm](https://github.com/nvm-sh/nvm) or [nvm for windows](https://github.com/coreybutler/nvm-windows)) for handling packages and running a local server
* [Windows PowerShell](https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.2) for various local development tasks such as installing packages using Node Package Manager

# Windows Powershell

We will be working in windows Powershell a lot. In order to start a new PowerShell, just hit the Windows button, and start typing Pow . An autocompleter should provide you with the option to start a Powershell now. Sometimes it will be handy to have admin privileges, so be sure to run the PowerShell as administrator (right-click, choose “run as administrator”).

# A primer on Git and GitHub

Git is a version control system (or VCS). Using Git, you can safely store code online, and work together on the same codebase, and even the same files, without interfering in each other's work. The place where the online code is kept is called a repository, or repo.

In essence, without a tool such as Git, it would be virtually impossible to work together on the project.

A standard online file-serving system such as OneDrive or DropBox has no support for coding on the same project at the same time. In development projects such as this, it is quite likely that people will change the same files at almost the same time. Using OneDrive or DropBox, this would lead to sync conflicts, and there would be no way of dealing with them in an efficient manner. A VCS such as Git has all the features needed to deal with these conflicts effectively, efficiently and safely.

GitHub is an online service that was created to host Git repositories. So, in essence, GitHub makes it possible to use Git without having to install Git on your own private server to host the Git repository.

If you want to learn more about Git and GitHub, and the difference between the two, this should be clarifying: <https://blog.devmountain.com/git-vs-github-whats-the-difference/>

Now, in order to get started with Git and GitHub, please follow the following tutorial: <https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners>

And finally, since GitHub has launched its own command line interface, which they refer to often, and which makes working with GitHub from the command line easier, we advise to install <https://cli.github.com/> . This will allow you to use the gh commands you can find throughout your GitHub account (e.g. to clone a repo).

If you want more information, you can check the full documentation here : <https://git-scm.com/book/en/v2> . However, for this course, you should be good with the introductory tutorial.

Now, to be clear, Git is just a fancy way of storing your files safely online in a way that multiple people can easily work together on the same project. Although it facilitates the process, it is not central to what we are trying to do in the project, so we won’t elaborate too much on it.

If you want to learn more about Git and GitHub, this is a great resource:

<https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/GitHub> . If you read the introduction and follow the Hello, World guide, you are well on your way to understanding Git and GitHub.

### **For the MC exam** you should know the things written in this chapter, no more and no less. So, not the things behind the links, just the content written on the page(s) of this chapter.

# A primer on Visual Studio Code

Visual Studio Code is a great IDE, or Integrated Develoment Environment (the app you use to develop software). It is Open Source, runs on many platform, is fast and light, and can be extended with a huge number of plugins to aid the development process. There really are only a few steps we must go through to get a feeling of what it is like to work with Visual Studio Code.

* Download the program here: <https://code.visualstudio.com/>
* Go through the getting started videos you can find in the First Steps section to get acquainted: <https://code.visualstudio.com/docs>
* Visual Studio Code has the ability to install many useful additions called “Extensions”. You can find them using the blocks-like icon in the sidebar. For instance, let’s install an extension called “open in browser”, which will give you a convenient way to open a file in your browser:

Afbeelding met tekst, monitor, schermafbeelding, scherm

Automatisch gegenereerde beschrijving

In the example above, I have already installed the first extension in the list. You won’t have, so you will still see an install button. Click it to install the extension. Now you have the option to open any file in your browser quickly, which will come in handy later on.

Next, let’s create a very simple HTML webpage with Visual Studio Code. Start by clicking File>New File, then enter this code:

<!DOCTYPE html>

<html>

<head>

  <title>Example website HTML and CSS</title>

  <link href="HTML\_and\_css.css" rel="stylesheet" type="text/css" />

</head>

<body>

    <div id="allcontent">

        <div id="nav">

            <ul>

                <li>

                    <a href="overview.html">overview</a>

                </li>

                <li>

                    <a href="elementsandattributes.html">elements and attributes</a>

                </li>

            </ul>

        </div>

        <div id="content">

            <h1>HTML and CSS</h1>

            <p>Welcome to

                <span class="emphasis">this HTML page</span>. This website is used as an introduction to the principles of HTML and CSS.

        </div>

    </div>

</body>

</html>

Now save the file. If you do not see the file in the sidebar, you might not have the “Open editors” view open. Click on the three dots in the sidebar and enable “Open Editors”. Now you can right-click and select Save, to save the file somewhere.

A screenshot of a computer

Description automatically generated with medium confidence

You can now open the folder where you saved the file by clicking on “Open Folder” in the sidebar. The contents of the folder are visible now.

Next, let’s create a CSS file in the same folder, and name it “HTML\_and\_css.css”. You can use the “New File” icon in the sidebar to create the new file.

A screenshot of a computer

Description automatically generated with medium confidence

Copy this code in the file:

body {

    margin: 0px;

    padding: 0px;

    font-family: Arial, Helvetica, sans-serif;

    font-size: 14px;

}

.emphasis {

    font-weight: bold;

}

#nav{

    position: absolute;

    left: 0px;

    top: 0px;

    padding: 10px 0 0 0;

    margin: 20px 0 0 15px;

}

#nav ul {

    margin-top: 0px;

    margin-bottom: 10px;

}

#nav li {

    margin-bottom: 5px;

    font-weight: bold;

    font-size: 12px;

}

#nav a:link {

    color: #666666;

}

#nav a:hover {

    color: #FF6600;

}

#nav a:visited {

    color: rgb(22, 132, 175);

}

#content {

    position: absolute;

    left: 0px;

    top: 0px;

    margin: 20px 0 0 215px;

    padding: 0 15px;

}

#content h1 {

    font-size: 2em;

    color: #D45A12;

    padding: 20px 10px 0px 40px;

    border-width: 20px;

    border-color: black;

    border-style: solid;

    margin-bottom: 10px

}

Now, you can right-click the overview.html file, and choose “Open in default browser”, and your HTML file will open in the browser.

Later on, you will find that Visual Studio Code also comes in handy when developing full applications, with it’s excellent integration with tools such as Git, Node, Chrome, WebPack and so on…

This is a good time to open the folder example website folder (File>Open Folder…) in Visual Studio Code, and start learning about HTML and CSS. The content of that site is the HTML and CSS content that should be known for the exam.

### **For the MC exam** you should have followed the steps described in this chapter. So, not the things behind the links, just the content written on the page(s) of this chapter.

# HTML and CSS

You will need some experience with HTML and CSS. These are great places to start :

* In the previous chapter, we asked you to download and study the example website provided at Ufora. This example website contains everything you need to know for the multiple choice exam. However, in order to become more proficient in HTML and CSS, more knowledge is needed. That might come in handy when working on your project!
* <https://developer.mozilla.org/en-US/docs/Learn>
  + For learning HTML: <https://developer.mozilla.org/en-US/docs/Learn/HTML>
  + For learning CSS: <https://developer.mozilla.org/en-US/docs/Learn/CSS>
* If you want a guided tour to becoming a front-end web developer : check this path : <https://developer.mozilla.org/en-US/docs/Learn/Front-end_web_developer>
* An important addition to the CSS stack was the creation of CSS Grids. They can be extremely useful for positioning things on a webpage: <https://learncssgrid.com/>
* Flexbox is another useful part of the CSS toolkit: <https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox>
* In general, and as always, the Mozilla guides are great for HTML and CSS developers. Here is the overview of all CSS Layout tools: <https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout>

### **For the MC exam** you should know the content of the example website.

# Getting started with Node, Node Version Manager, and Node Package Manager

Node is a server-side javascript framework. In short, it allows you to use javascript to create server-side programs, such as command-line tools or server scripts.

The Node Package Manager (NPM) is a node-based package manager. A package manager can be used to install software packages on a computer. More specifically, we will use it to install packages (additional software) that can enhance our own website.

The Node Version Manager (NVM), is a program that allows you to install, and use, versions of node and NPM on your computer. It is helpful in installing node, but becomes even more useful if you need to use more than one version of node, since it will allow you to easily switch between versions of node.

Although Node, NPM, and NVM are interesting by themselves,

* Use one of these handy guides to install nvm-windows (if you are on windows):
  + <https://github.com/coreybutler/nvm-windows>
  + <https://docs.microsoft.com/en-us/windows/dev-environment/javascript/nodejs-on-windows>
* If you are on OS X, you have two options.
  + If you also want to use the node version manager, so that you can switch between node versions if required, you can follow this guide : <https://tecadmin.net/install-nvm-macos-with-homebrew/>
  + If you just want Node and NPM, use this guide : <https://nodesource.com/blog/installing-nodejs-tutorial-mac-os-x/>
  + For this course, both options are fine. However, if you should ever decide to do this professionally, Always go for NVM.
* Now, having installed nvm, it is easy to install the latest “long term support” version of node using a Windows Powershell as admin, and executing the code (does not matter in which folder
  + nvm install lts (on Windows) or nvm install --lts (on Linux of MacOS)
* Now you can check which version of node is installed by executing:
  + node –version
* If you execute nvm list (or nvm ls on Linux or Mac), you can a list the installed node versions

Text

Description automatically generated

### **For the MC exam** you should know the things written in this chapter, no more and no less. So, not the things behind the links, just the content written on the page(s) of this chapter.

# On to the development part

Ok, so after having acquired some experience with:

* Windows Powershell
* Git and GitHub
* Visual Studio Code (and some HTML and CSS)
* Node, NPM and NVM

It is time to start the development of our site. Since the development of our site will be in NextJS, which is a web framework that used React, which in itself is a web framework that uses JavaScript, we will start with some familiarization with JavaScript, React and NextJS.

# A good introduction to JavaScript

Javascript is \_the\_ programming language of the web. Where HTML and CSS are declarative languages (meaning that you declare what the result of your code should be), Javascript is a real procedural language, meaning that you can define functions that perform certain tasks. Any tasks that you might think of.

Let’s first start with some more accessible tutorials.

* A good starting point for learning the basics of JavaScript can be found here: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript>
  + Follow this tutorial up to and including the “Functions” chapter.
  + For trying out the code on the page, create a simple HTML file, and link a js file to it using the script tag. In the example below, we have created a script.js file, linked it to the HTML file, and opened the HTML file in the browser. If we then open the Chrome developer tools (using F12 on Windows), we can see the output of our javascript file.

Graphical user interface

Description automatically generated

* Another nice introduction to Javascript can be found here :
  + <https://developer.mozilla.org/en-US/docs/Learn/JavaScript>

### **For the MC exam** you should have knowledge of the following sections from the <https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript> introduction:

* Variables
* Operators
* Control Structures
* Objects
* Arrays

# A good introduction to React

React is a javascript library for building user interfaces. It is the basis of NextJS, which is what we will be using in our website project.

## Getting started

Here you can find the getting started guide for React: <https://reactjs.org/docs/getting-started.html>

The first page contains a lot of useful links, but do not forget to follow the table of contents up to and including the “Main Concepts” section.

For most examples in the Main Concepts section, you can get away with just adding React as a <script> tag, as described at <https://reactjs.org/docs/add-react-to-a-website.html> , even though this page also explains how you can create a more advanced “React App” using the create-react-app approach. We will discuss this later.

If you want to sync your work online, use the following steps:

1. Create a folder at C:\Source\UGent\ScriptBasedReact (you can freely choose this folder, but using this location will make it easier to follow the steps)
2. Now create a private repo at your GitHub (you should already have an account from a previous chapter) using the name ScriptBasedReact (again, can be freely chosen, but using this name will make it easier to follow along)

Graphical user interface, text, application, email

Description automatically generated

1. Now, open a PowerShell and change the directory to the one we just created using the cd command (change directory command): cd c:\Source\UGent\ScriptBasedReact
2. Now, follow the steps described here (and as described in the GitHub page you see right after creating the repo):

Afbeelding met tekst

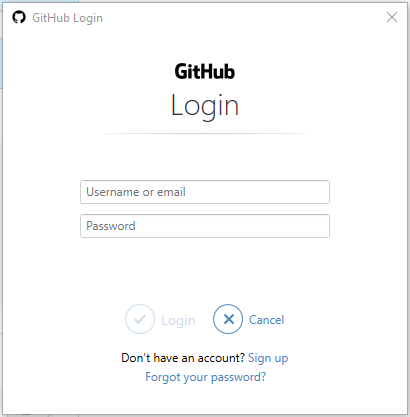
Automatisch gegenereerde beschrijving

Copy (Ctrl+C on the webpage) paste (just right-click in the PowerShell after having copied the line from the webpage) each line into your PowerShell and watch the output. This should provide you with an output similar to this:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

It is likely that you will be asked to log in to your GitHub account:



If this is the case, the password is not the password you use to log in to github.com, instead, it is a personal access token. You can find out how to create one here: <https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>

You might get an error message when logging in using the window shown above, but you will be able to log in again in your PowerShell. That one should work…

Now, download the files you find at <https://reactjs.org/docs/add-react-to-a-website.html> (There is a line of text called “Download the full example”. Find it on the page. Unzip the downloaded file, and copy the files in C:\Source\UGent\ScriptBasedReact .

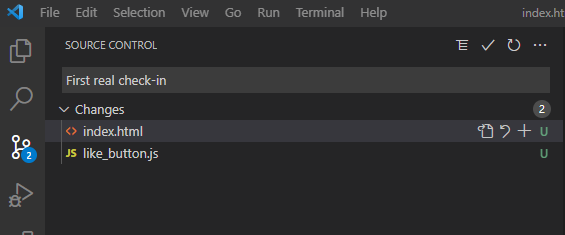
Now, open this folder in Visual Studio Code.

This will give you a great starting point for playing around with the examples. Check out the index.html page in your browser (right-click and choose “Show in browser”):

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Now, if you want to have these files synced with your online repo, you can head over to the Git tab, provide a nice comment, and click the check-mark:



Now, you will see a sync button. Click it. After syncing, these files will be available in your online repo, and you can access them from anywhere.

Graphical user interface, text, application

Description automatically generated

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Now, if you follow along the Main Concepts guides, and you get to the JSX part, do not forget that you first need to do the “Add JSX to a Project” and “Run JSX Preprocessor” steps from the <https://reactjs.org/docs/add-react-to-a-website.html> page before they will work… They are way down at the bottom of the page.

Next, a good structure for trying these JSX expressions is the following:

Afbeelding met tekst, monitor, schermafbeelding

Automatisch gegenereerde beschrijving

With this being the content of the HTML file:

<!DOCTYPE html>

<html>

  <head>

    <meta charset="UTF-8" />

    <title>Add React in One Minute</title>

  </head>

  <body>

    <!-- We will put our React component inside this div. -->

    <div id="root"></div>

    <!-- Load React. -->

    <!-- Note: when deploying, replace "development.js" with "production.min.js". -->

    <script src="https://unpkg.com/react@16/umd/react.development.js" crossorigin></script>

    <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js" crossorigin></script>

    <!-- Load our React component. -->

    <script src="JSX-expressions-1.js"></script>

  </body>

</html>

And this being the content of the JSX-expressions-1.js file:

function formatName(user) {

  return user.firstName + ' ' + user.lastName;

}

const user = {

  firstName: 'Harper',

  lastName: 'Perez'

};

const element = (

  <h1>

    Hello, {formatName(user)}!

  </h1>

);

ReactDOM.render(

  element,

  document.getElementById('root')

);

Now, you can just copy paste the examples from the tutorials into the js file, and the webpage will update accordingly (ONLY if you have executed the command from the “Run JSX Preprocessor” section of the webpage at <https://reactjs.org/docs/add-react-to-a-website.html> .

In chapter 5, ignore the parts starting from “Using state correctly”, they are beyond scope.

Learn upto and including chapter 6. Then you are good to advance to the next section on Create React App.

### **For the MC exam** you should know how to sync (push, pull, sync etc.) with GitHub. There won’t be any questions from the React part of this section. You should really follow the simple steps above though, if you want to contribute to the group project, which you have to…

## Create React App

There is a suggestion of using create-react-app for creating a basic react app to play around with. This is a good step up from the previous way of working (adding React to a webpage using the script tag). To be clear: the complete explanation of the create-react-app app can be found here: <https://create-react-app.dev/> , and the git page is here <https://github.com/facebook/create-react-app> (Hey, React was created by Facebook!)

* + When you run npm start while doing the create-react-app tutorial, you might be starting a Node service that wants to communicate across the web for the first time. If this is the case, you will be asked if node can communicate across the firewall. If so, allow access to Private Networks:

Afbeelding met tekst

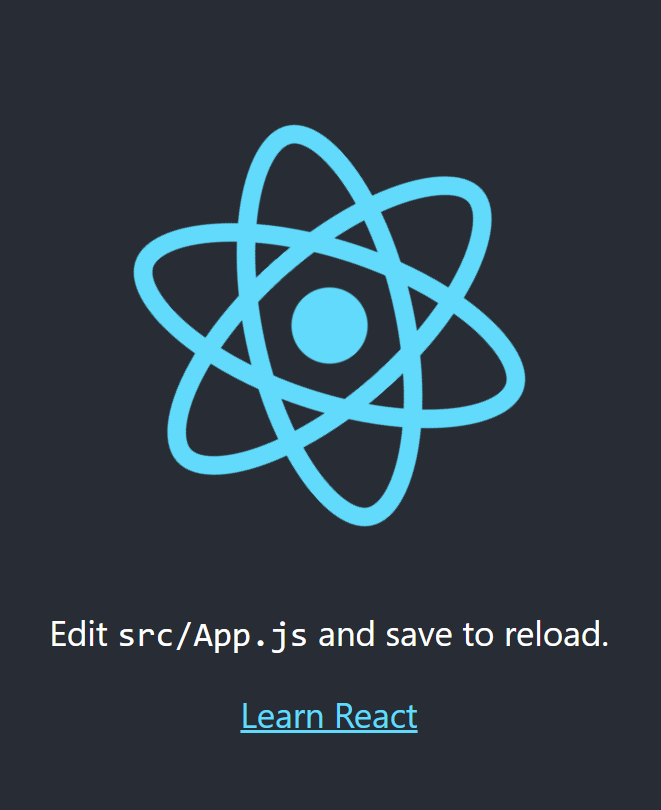
Automatisch gegenereerde beschrijving

Something along these lines should be the output of changing to the my-app directory and running npm start:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

* + After running npm start, you will be able to see your first React app at <http://localhost:3000/>

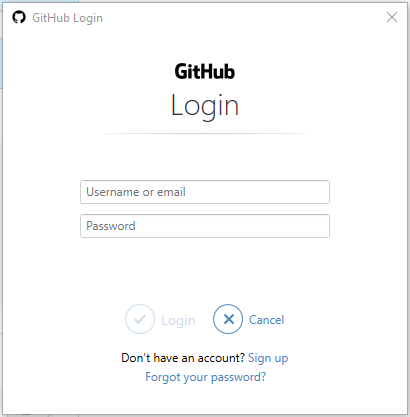


* + Now that you have created your first React App, follow this guide to connect it to your own personal GitHub account:
    1. First, create a new online repo, called CreateReactApp (the name can actually be chosen freely, but this one seems appropriate here). Do not create the git ignore, readme, etc. files (leave all boxes unchecked).
    2. After the creation, Github is so nice to tell you the steps to follow to connect your local repo to the online repo:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

* In our case, we need to push an existing repository from the command line, so:
  + git remote add origin <https://github.com/JanMoonsUGent/CreateReactApp.git> (obviously replace the URL with your own)
  + git branch -M main
  + git push -u origin main
* After doing the git push, you will likely be asked to log in to github:



* The password, however, is not the password you use to log in to github.com, instead it is a personal access token. You can find out how to create one here: <https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>

This should be the result:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Now, we are all good to start tracking our changes between the local and the online version of our repo!

If you need more information on connecting your local repo to your online repo, you can follow these guides:

* <https://gist.github.com/mindplace/b4b094157d7a3be6afd2c96370d39fad>
* Starting from “Adding a project to GitHub without the GitHub CLI”: <https://docs.github.com/en/github/importing-your-projects-to-github/importing-source-code-to-github/adding-an-existing-project-to-github-using-the-command-line> .

Doing the actual updating will be a lot easier from Visual Studio code though… To check if everything is working fine, I have just added a text file called PAT.txt to the folder my-app (so to the folder of my repo) . (BTW, you are NEVER supposed to store passwords or tokes in your repo in a real environment, and just create a new PAT if you ever need it). Your Visual Studio Code GIT page should show the change:

Afbeelding met tekst, schermafbeelding, monitor

Automatisch gegenereerde beschrijving

The U at the end of the PAT.txt line indicated that this file is untracked (a new file). Enter a message such as “Added Personal Access Token” in the message field, and click on the checkmark at the top to “Commit” this change. Committing a change is akin to confirming that you actually want to have these changes in your repo. It is a first step toward syncing these changes online. Afterwards, you will see a button that will allow you to sync these changes online: Afbeelding met tekst, schermafbeelding, monitor

Automatisch gegenereerde beschrijving

Go ahead and sync these changes… You will get a confirmation dialog. Choose “OK, don’t show again”. If all went well, you will now have this additional PAT.txt file in your repo. Everything is set up perfectly.

Afbeelding met tekst

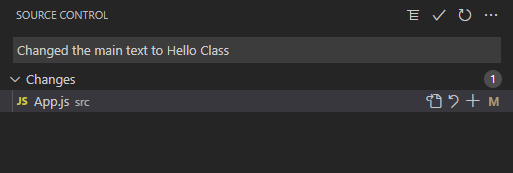
Automatisch gegenereerde beschrijving

Now, let us go back to VS Code, and just change the “Learn React” text in the src/App.js file to something else, such as “Hello Class!”, het Ctrl+S to save the file, and see if everything is working and updating properly: Afbeelding met tekst

Automatisch gegenereerde beschrijving



Now, finally, check in this change again:



Now, sync, and see if the change is also applied online:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Seems like our development environment is working great. You are all set up now to follow some more react tutorials!

### **For the MC exam** there won’t be any questions from this specific section of the course. You should really follow the simple steps above though, if you want to contribute to the group project, which you have to…

## More React Tutorials

Another nice React tutorial can be found here: It is nice, because it lets you complete a simple project step-by-step:

<https://reactjs.org/tutorial/tutorial.html>

Be advised, though: the tutorial goes a bit beyond what is required for our CVVW course…

Finally, if you feel like you need another primer, you can follow this nice tutorial as well:

<https://www.taniarascia.com/getting-started-with-react/>

### **For the MC exam** there won’t be any questions from this specific section of the course. You should really follow the simple steps above though, if you want to contribute to the group project, which you have to…

# A good introduction to NextJS

OK, so now we are familiar with some very important developer tools, such as Git, GitHub, Node, Chrome Developer Tools and Visual Studio Code, and we at least have some notions of HTML, CSS, JS, and React.

Now we are ready to start the real work on our website. We will be using the NextJS framework available at nextjs.org. Before we get started, it is always a good idea to familiarize ourselves with the framework. Go through the tutorial here :

<https://nextjs.org/learn/basics/create-nextjs-app>

As described in the tutorial, you will be working toward this result : <https://next-learn-starter.vercel.app> , of which the code is available here : <https://github.com/vercel/next-learn/tree/master/basics/demo>

One additional comment: in order to start the project locally, let’s use the same c:\Source\UGent location.

So, start a PowerShell as administrator, do a cd c:\Source\UGent, and then:

C:\Source\UGent> npx create-next-app nextjs-blog --use-npm --example <https://github.com/vercel/next-learn/tree/master/basics/learn-starter>

This will create the necessary structure for our application. Now, we will link the project to our online repo in a different way. The project that was just created already contains information about an online repo, so we won’t create it online first again. In this case, open VS Code, and go to the Git tab:

Afbeelding met tekst, schermafbeelding, monitor, zwart

Automatisch gegenereerde beschrijving

Click on the Publish Changes button. You will be asked to log in to your github account. You will be taken to the GitHub website where you will be able to authorize VS Code. You will then be asked if you want to open the next link with VSCode, please do so. Next, in VS Code, you will be asked to open a URI. Choose “Open”.

Afbeelding met tekst, schermafbeelding, monitor, scherm

Automatisch gegenereerde beschrijving

Next, you will be asked which online repo you want to sync with, a public or a private one. Choose the private one.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

There you go. Synced. Now, if you cd into the next-js blog directory in your PowerShell, and then enter npm run dev, your new blog will be generated, and you will be able to visit it (after some time) at localhost:3000 in your browser.

Now, just follow along the online tutorial! The section “API Routes” is not relevant to our group work project and can be freely skipped.

### **For the MC exam** there won’t be any questions from this specific section of the course. You should really follow the simple steps above though, if you want to contribute to the group project, which you have to…

## Working on the same project from a different PC

Now, suppose you want to be able to work on the same project from a different PC. With Git and GitHub, that becomes really easy to do. Log in to GitHub, go to the relevant repo, and find the Code button. There, check out the Github CLI tab. This tab will give you the exact command you need to clone the repo onto a different PC.

Graphical user interface, text, application, email

Description automatically generated

For instance, after doing the initial work on my home PC, I went home and cloned the nextjs-blog repo to my home PC (in the same directory, although that is not required, it just makes it easy to talk about directories…):

Text

Description automatically generated

As you can see in the screenshot above, there is one more step. Make sure that you run “npm install” in the nextjs-blog folder, so that all the node packages are installed. In the nextjs-blog project, there is a gitignore file that tells git to ignore the node\_modules folder. So, any packages that were installed through node will not be synced with git (there is a good reason for this, but that is beyond the scope of this tutorial). So, in order to install these packages on your new PC, always run npm install after cloning a repo.

Now you can open the nextjs-blog directory in visual studio code, and you are good to go. The first time you open a new repo that you cloned from a remote location (such as GitHub), you will probably get this dialog:

Text

Description automatically generated

Obviously, in this case, you can choose to trust the authors.

Now, “npm run dev” again in the nextjs-blog folder, and your site is running on your new PC.

## Working as a team

Now, for the actual website group project, you will have to work together as a team. Although I would like everyone to go through these tutorials separately, now is a good time to try the team features a first time. Follow Step 2 as described here:

<https://medium.com/@jonathanmines/the-ultimate-github-collaboration-guide-df816e98fb67>

For our purposes, ignore steps 3 and 4. Although creating branches and merging them is a big part of collaborating in teams on code, it is outside our project's scope. We will always be working in the main brach (the one that was automatically created).

Choose one leader in the team. Have him/her set up the project (if not already done so), and let him/her add collaborators. Now everyone should be able to clone the repo locally using the gh command, make a small change in visual studio code, and sync those changes back to the main repo, and verify they are received.

## The result

If you work your way through the tutorial, you’ll end up with a working, deployed website:

Graphical user interface, application, Teams

Description automatically generated

### **For the MC exam** there won’t be any questions from this specific section of the course. You should really follow the simple steps above though, if you want to contribute to the group project, which you have to…

# NextJS and StoryBlok

Now it is time to bring our two main components together – NextJS (and thus HTML, CSS, JS, React) and StoryBlok (our data layer as a headless CMS). You can find a page dedicated to NextJS and Storyblok working together here:

<https://www.storyblok.com/tc/nextjs>

### The first tutorial

A first tutorial (currently broken!) to follow is this one:

<https://www.storyblok.com/tp/add-a-headless-cms-to-next-js-in-5-minutes>

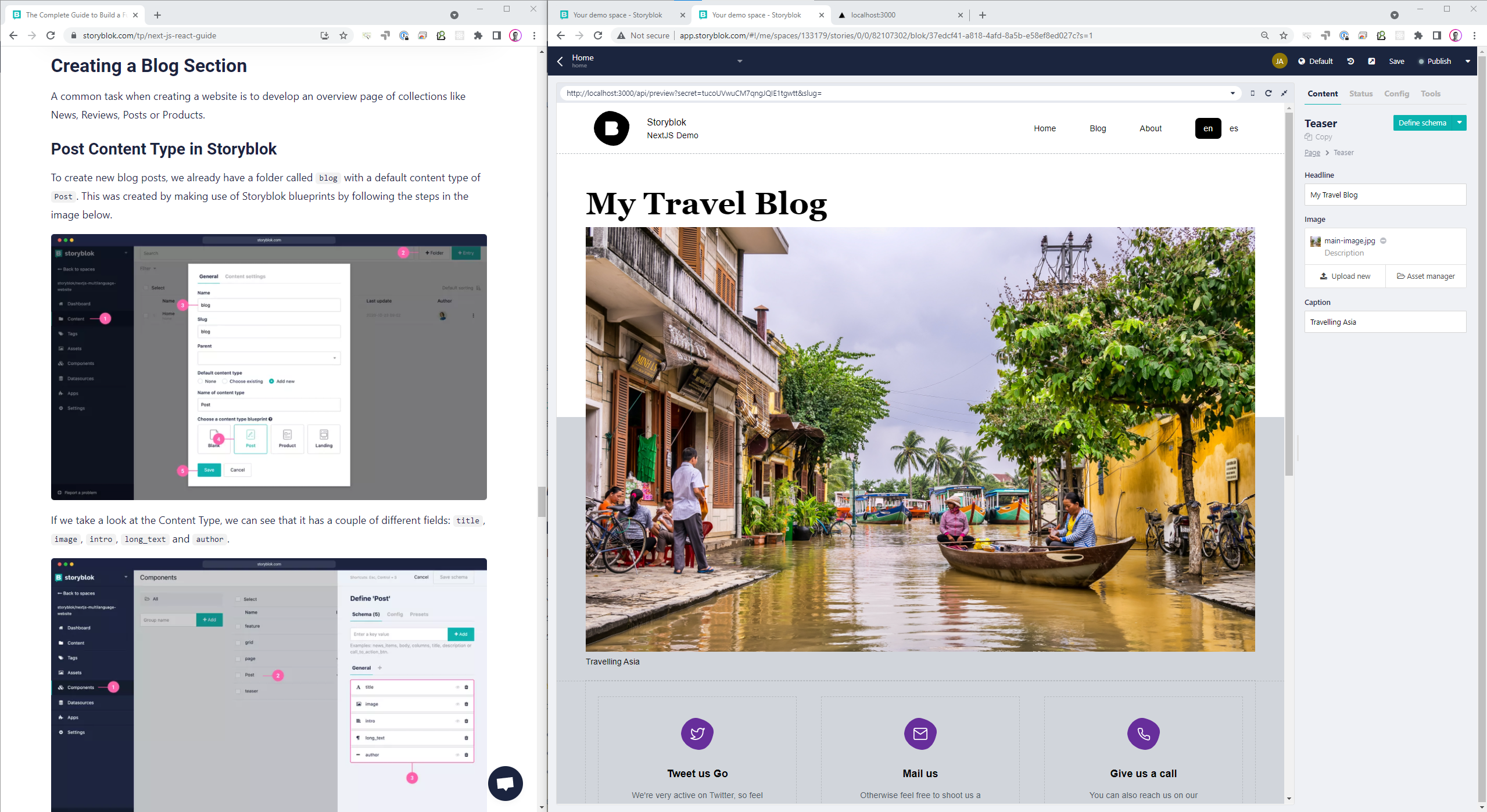
it teaches you how you can connect a very simple NextJS site to StoryBlok. Unfortunately, currently the tutorial appears broken. If still so, please continue with the next tutorial.

### The second tutorial

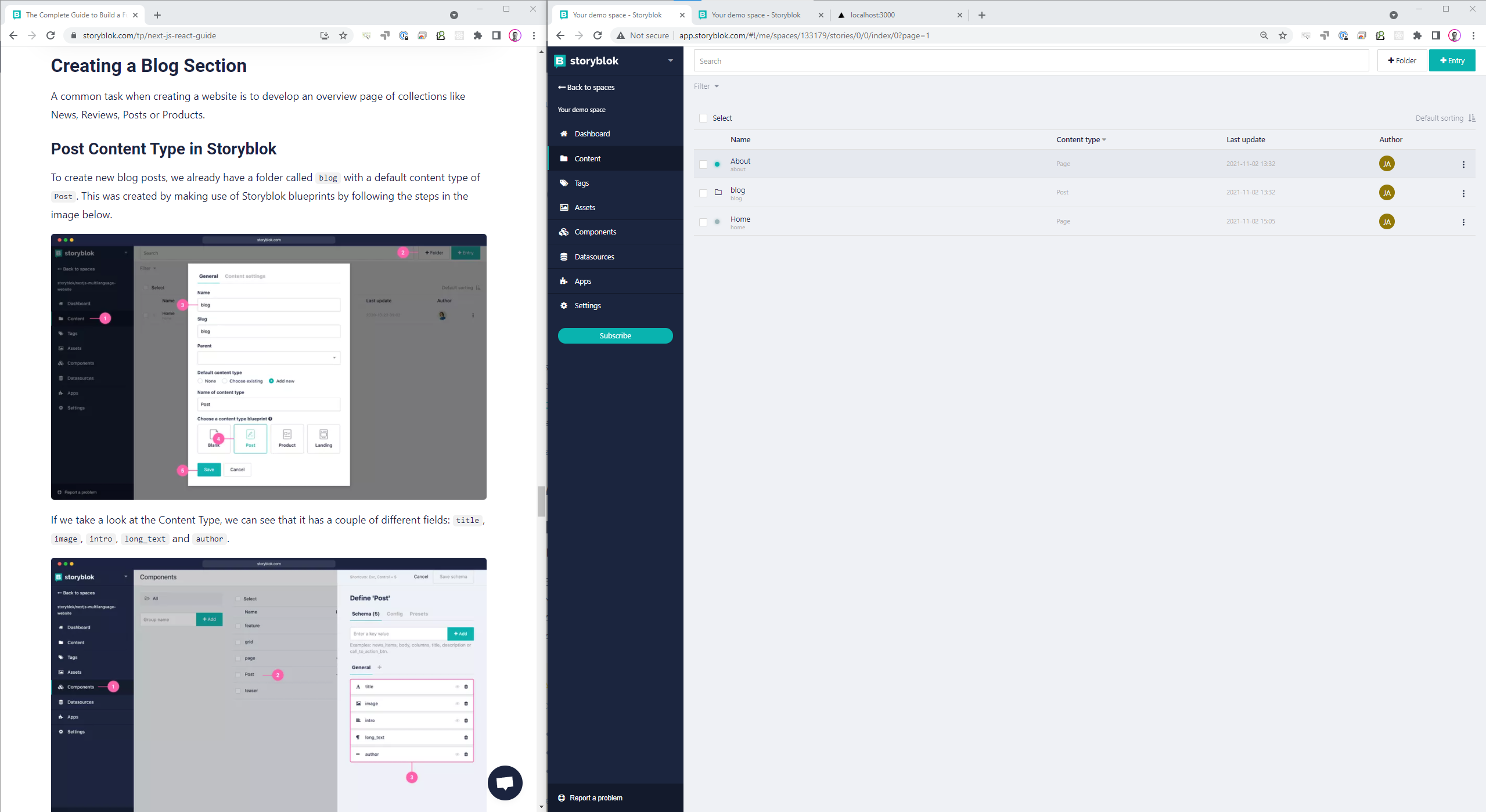
After this initial tutorial, I would advise you to follow the following tutorial. Not every part of the tutorial is relevant, but it shows you a more complex website being developed in NextJS and StoryBlok – one close to the end result we are hoping to create. In fact, our own IMDB project is based on this tutorial as well.

<https://www.storyblok.com/tp/next-js-react-guide>

While working your way through the tutorial, I would advise you to use your screen real estate well. Pin the tutorial to the left of your screen, and pin a browser containing StoryBlok to the right (I have given this window a bit more room):



The StoryBlok window has a few tabs. One containing the preview (as you see above), one containing the admin screen to make edits to structure and content, and one containing the localhost:3000 local site (in order to debug certain things). Good window management makes for a much better developer experience! Add additional tabs as needed, but keep the tutorial window visible at all times, so you do not loose track of the goal.



When you first try to check in your changes so that you can deploy to Vercel, it will tell you that you do not have the permissions to push to the original Git repo (logically so), and the system will ask you if you want to create a new repo for this project. Choose yes.

Our very own website project

Now, it is time to deploy to Vercel. Just log in to your Vercel account, and in the Overview page, choose New. This will lead to a screen where you can choose the project from your GitHub account to deploy (I had named the project imdbplus, since the nextjs-multilanguage-website is the basis for our imdbplus project – you should just choose the nextjs-multilanguage-website at this stage):

Graphical user interface, application

Description automatically generated

Next, you do not have to create a team, and just accept all default settings and start the deploy:

Graphical user interface, application

Description automatically generated

### **For the MC exam** you should know how to deploy to Vercel. For the rest there won’t be any questions from this specific section of the course. You should really follow the simple steps above though, if you want to contribute to the group project, which you have to…