1 Goal of this tutorial

Your local development infrastructure is up and ready.

You will download a prepared Ubuntu VM (Vmware player required).
To finish it depending on your requirements, you will install an IDE of your choice (here we use JetBrains IntelliJ Ultimate), install NodeJS/npm, configure your IDE and set up a first simple project using a NodeJS webserver and webpack with webpack-dev-server. Finally you will configure your project to use git with commandline and with IDE.

2 Download

Download an Ubuntu 18.04LTS Vmware instance:
https://www.mjs.ch/transfer/UbuntuDevbox_VMtools-CR.zip (2.3GB)
and unzip (12GB + 8GB RAM-Store) and start it.
Login: user (gibbiX12345) – if the screen lock is active, drag the clock upwards

Download IntelliJ: https://www.jetbrains.com/idea/download/#section=linux or
WebStorm: https://www.jetbrains.com/webstorm/download/#section=linux
(and drag it into the VM, if downloaded to your host).

3 Install IDE

- extract and install IDE:

```
sudo tar -xvf IntelliJIdea-2018.1.5.tar.gz -C /opt/
sudo mv /opt/idea-IU-181.5281.24/ /opt/IntelliJ
sh /opt/IntelliJ/bin/idea.sh
```

- create a new project of type “static web” with your IDE – you will find your project in `/home/user/IdeaProjects`
- choose featured plugins: live edit tool & nodejs
- settings -> languages -> javascript: ecmascript 6
- view -> tool windows -> terminal window
- assign more ram: help -> edit custom vm options -> `idea64.vmoptions`:

```
-Xmx2048m
-Xms1024m
```

4 Install NodeJS

```
sudo apt-get install curl
curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash -
sudo apt-get install -y nodejs
node -v
```

make some commands globally available:

```
sudo npm install -g webpack webpack-cli webpack-dev-server
sudo npm install -g http-server
```

5 Install GIT

```
sudo apt-get install git
git --version

#initial config for user
```

```
git config --global user.name "your name"
git config --global user.email "your@email"
git config --global color.ui auto
git config -list // show config
```

Github uses email-address to associate a push with a user/avatar, not the login-user. Therefore use the noreply email `<mygithubuser>@users.noreply.github.com`

To get the course content, clone the course repo to your disk:

```
git clone https://github.com/maenujem/javascript_course.git
```

It uses submodules in directory `/students` . These are pointing/containing the modified files (specific snapshots) from all student repos – this is to easily compare and discuss different solutions. You can get the initial content with the following command

```
git submodule update --init --remote --recursive
```

Update your local course content repo regularly using the following commands

```
git pull
git submodule update --remote --recursive
```

6 Create HelloWorld example

6.1 Simple (without using NodeJS and its webpack..modules)

- (if not done already) create your own github repo for your user at http://github.com/ using the browser
- clone the remote repo to your local disk - it will create a directory named with the repo name

```
git clone https://github.com/user-x/gibb_js.git
```

You can copy files from the course content repo and/or create your own ones here.

- In your project create a new folder eg. `/01a_helloweb`
- inside create the following folders:

```
/src
/src/styles
/src/images
/src/scripts
```

- also create a file `/src/index.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello WebDev</title>
    <link rel="stylesheet" href="./styles/main.css">
</head>
<body>

<h1>Hello WebDev's</h1>
<img src="./images/js.png">

<script>
// add simple JS script here for quick experimenting – place it in separate js later

window.alert('Hello webbie');
```

```
console.log('Hello console');

</script>

<!-- run via ide-webserver (./src/index.html) / http-server (/) without
nodejs/webpacking -->
<script src="./scripts/jquery.min.js"></script>
<script src="./index.js"></script>
</body>
</html>
```

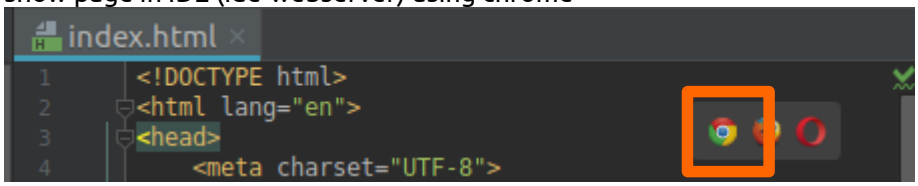
- also create a file `/src/index.js`

```
console.log('Hello main');


function sayHello(name) {
    console.log(`Hello function: ${name}`);
}

sayHello('me');


function helloWidget() {
    var element = document.createElement('div');
    element.innerHTML = 'Hello Widget - click on me';
    element.setAttribute('id', 'hellowidget');
    element.classList.add('hellostyle');
    return element;
}

document.body.appendChild(helloWidget());

$('#hellowidget').click(function(){
    $(this).after($('<b></b>').text('Hello jQuery'));
});
```

- also create a file `/src/styles/main.css`

```
body {background-color: turquoise}
.hellostyle {color: darkviolet}
```

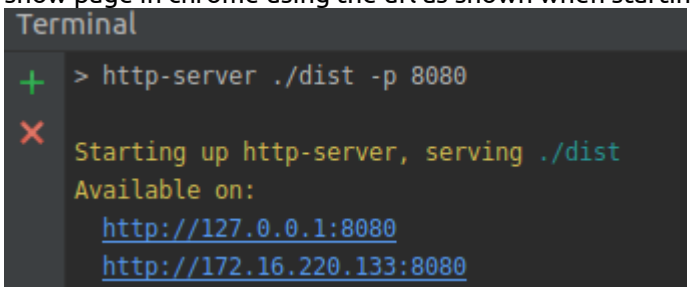
- also create a file `/src/scripts/jquery.min.js` (https://code.jquery.com/jquery-3.3.1.min.js)
- show page in IDE (ide-webserver) using chrome



- start (global) NodeJS http-server in `01a_helloweb`
  (`http-server <pathtowebroot_or_cwd> <myoptions>`)

```
http-server  ./src -p 8080
http-server -help
```

- show page in chrome using the url as shown when starting the http-server



- now create a file `./.gitignore` (rules for files not to version with git):

```
node_modules/
dist/
*.iml
.idea/
```

- add the current state to your github repo using command line in your project directory

```
#add to index/staging
git add .

#commit snapshot to local repo/HEAD
git commit -m "Initial commit"

#push local branch to remote (origin) repo's (master) branch
git push -u origin master
```

6.2 Less simple (using NodeJS and its webpack..modules)

- In your project create a new folder eg. `/01b_helloweb` and copy the contents from `/01a_helloweb` into it
- edit `index.html` to look like this (remove some stuff that will be provided by NodeJS-modules and webpack):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello WebDev</title>
</head>
<body>
<h1>Hello WebDev's</h1>
<img src="./images/js.png">

<!-- run via ide-webserver (./dist/index.html) / http-server (./dist) or webpack-dev-
server after webpacking -- <script src="./scripts/bundle.js"></script> will be
inserted by HtmlWebpackPlugin -->
</body>
</html>
```

- edit `index.js` to look like this (move some JS code to `helloWidget.js` and add imports):

```
import './styles/main.css'; // Load application wide styles
import $ from 'jquery'; // Import jquery library
import { sayHello, helloWidget } from './scripts/helloWidget'; // Import the exported
function

console.log('Hello main');
```

```
sayHello('me');

document.body.appendChild(helloWidget());
$('#hellowidget').click(function(){
    $(this).after($('<b></b>').text('Hello jQuery'));
});
```

- also create a file `/src/scripts/helloWidget.js`

```
console.log('Hello widget');

export function sayHello(name) {
    console.log(`Hello function: ${name}`);
}

export function helloWidget() {
    var element = document.createElement('div');
    element.innerHTML = 'Hello Widget - click on me';
    element.setAttribute('id', 'hellowidget');
    element.classList.add('hellostyle');
    return element;
}
```

- create an initial `package.json` (NodeJS npm) for your project (in `01b_helloweb`) – accept the defaults

```
npm init
```

- edit `package.json` to look like this:

```
{
  "name": "01b_helloweb",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "build": "webpack",
    "start": "http-server ./dist -p 8080"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "css-loader": "^0.28.11",
    "file-loader": "^1.1.11",
    "html-loader": "^0.5.5",
    "html-webpack-plugin": "^3.2.0",
    "style-loader": "^0.21.0",
    "webpack": "^4.15.0",
    "webpack-cli": "^3.0.8",
    "webpack-dev-server": "^3.1.4"
  },
  "dependencies": {
    "http-server": "^0.11.1",
    "jquery": "^3.3.1"
  }
}
```

- (the dependencies above have been added incrementally as required using the following command – fetching the latest stable version from the web)

```
npm install --save http-server
npm install --save jquery
npm install --save-dev webpack webpack-cli webpack-dev-server
npm install --save-dev css-loader style-loader file-loader html-loader html-webpack-plugin
```

- also create a file `/webpack.config.js` (it already contains some useful settings)

```
const Path = require("path"); // for resolving to absolute paths
const HtmlWebpackPlugin = require('html-webpack-plugin'); // for processing html (eg. caching hash) -- npm install --save-dev html-webpack-plugin
module.exports = {
    mode: 'development',
    devtool: 'source-map',
    context: __dirname + '/src', // set context for entry, default is cwd, __dirname is absolute path to /
    entry: './index.js', // context + path here, default: ./src/index.js
    output: {
        filename: './scripts/bundle.js' // default: ./dist/main.js
    },
    devServer: {
        contentBase: Path.resolve(__dirname, 'src') // static content
        // default path is output.filename (without /dist or /src)
    },
    module: { // loader for processing/transforming matching modules/things found in require/import
         rules: [
    // css -> put in to bundle.js
            {
                test: /\.css$/,
                use:['style-loader','css-loader'] // 2. style-loader <-- 1. css-loader
                // 2. put string into style-tag; npm install --save-dev style-loader
                // 1. collect referenced css into a string; npm install --save-dev css-loader
            },
    // images
             {
                test: /\.(png|jpg|gif)$/,
                use: [
                    {
                        loader: 'file-loader', // npm install --save-dev file-loader
                        options: {
                            name: './images/[name].[ext]'
                        }
                    }
                ]
            },
    // parse html -> images:file-loader
    // using together with HtmlWebpackPlugin to copy image: deactivate stylesheet in html and import in index.js (else: Error: return window && document && document.all && !window.atob; ReferenceError: window is not defined)
             {
                 test: /\.(html)$/,
                 use: {
                     loader: 'html-loader',
                     options: {
                         attrs: ['img:src', 'link:href']
                     }
```

```
                    }
                }
            ]
    },
    // instancing plugin loaded with require
    plugins: [
        // create html from template and adding bundle.js - hash where to see?
        new HtmlWebpackPlugin({
            template: 'index.html',  // why not ./src/index.html necessary ???
            filename: 'index.html'
        })
    ]
}
```

- also create a file `/README.md`

```
install with: npm install
dev with: webpack-dev-server -> open with http://localhost:8080/webpack-dev-
server
build with: npm run build
start with: npm start -> open with: http://127.0.0.1:8080
```

- apply the changes made directly in package.json and get the required node_modules from web

```
npm install
```

- start global webpack-dev-server in `01b_helloweb` for live updating scripts in browser (the generated bundle.js is in memory - run webpack to generate it to `./dist` – and *.html auto-reloads only after changes in *.js)

```
webpack-dev-server
```

- open in browser using webpack-dev-server: http://localhost:8080/webpack-dev-server (opening in browser using ide-webserver might not work because the browser might not yet support imports in JS and needs a processed file – which is what webpack does here)

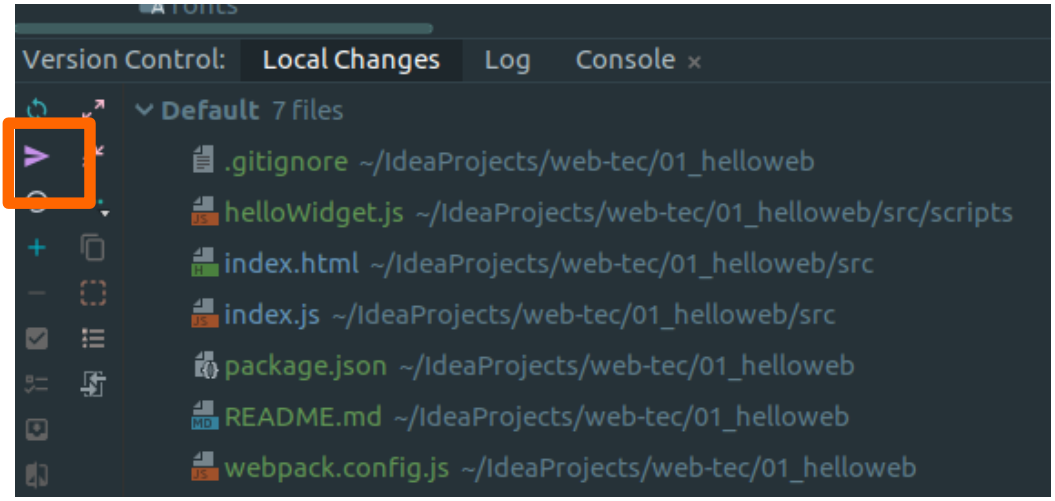
- generate the files for deployment using webpack (NodeJS module using webpack.config.js) run in `01b_helloweb`. This will generate `./dist/scripts/bundle.js` containing all javascript and styles. It will also add necessary links in index.html and create all other files necessary.
  `./dist` is the folder you would distribute to your webserver.

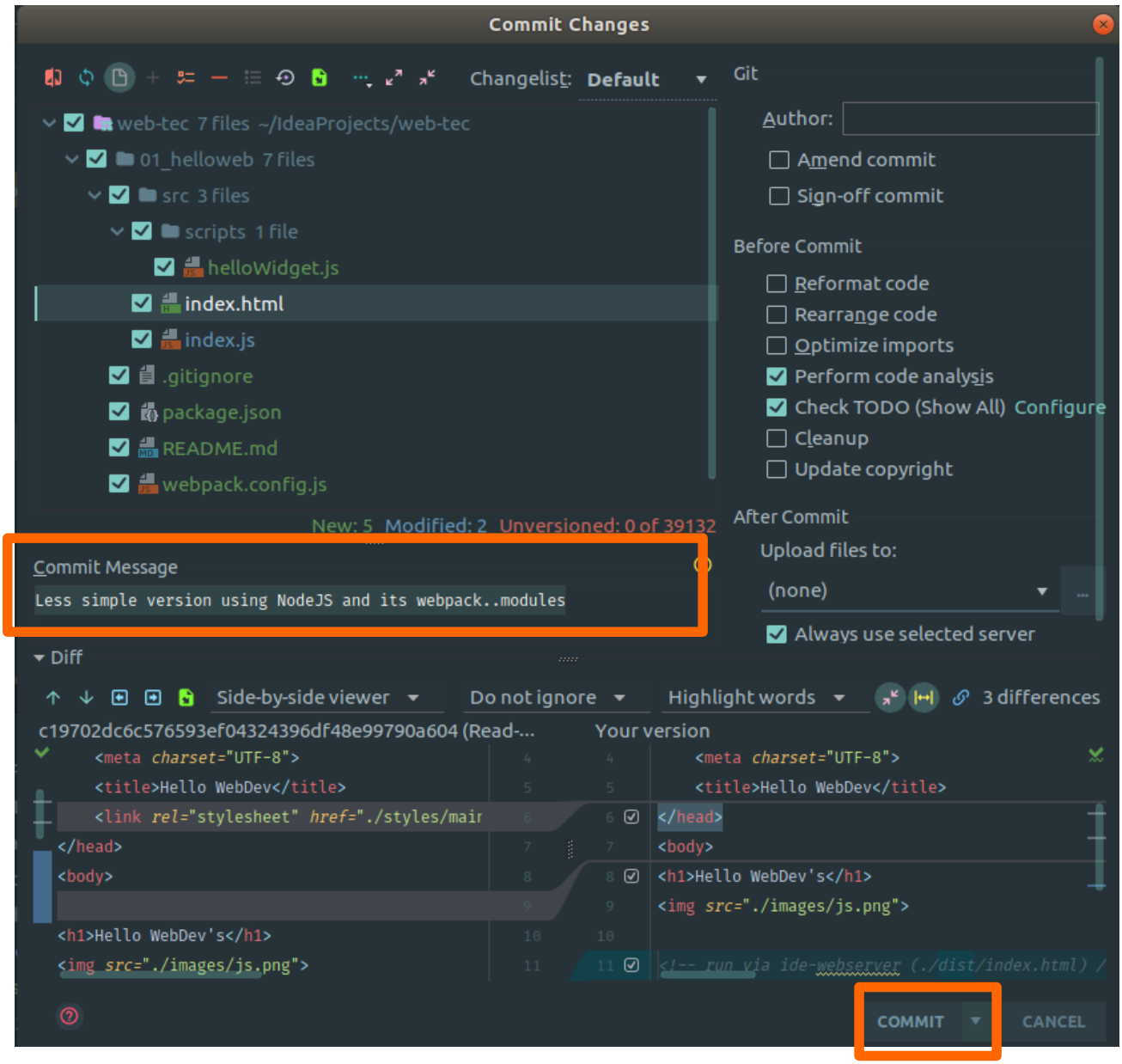
```
// using webpack.config.js with global webpack
webpack
// OR using package.json with npm/NodeJS (→ local/project's webpack module):
npm run build
```

- open in browser using http-server (NodeJS module - local project's http-server module - using package.json): start server (make sure webpack-dev-server is not runing on the same port at same time) and then use URL shown in browser

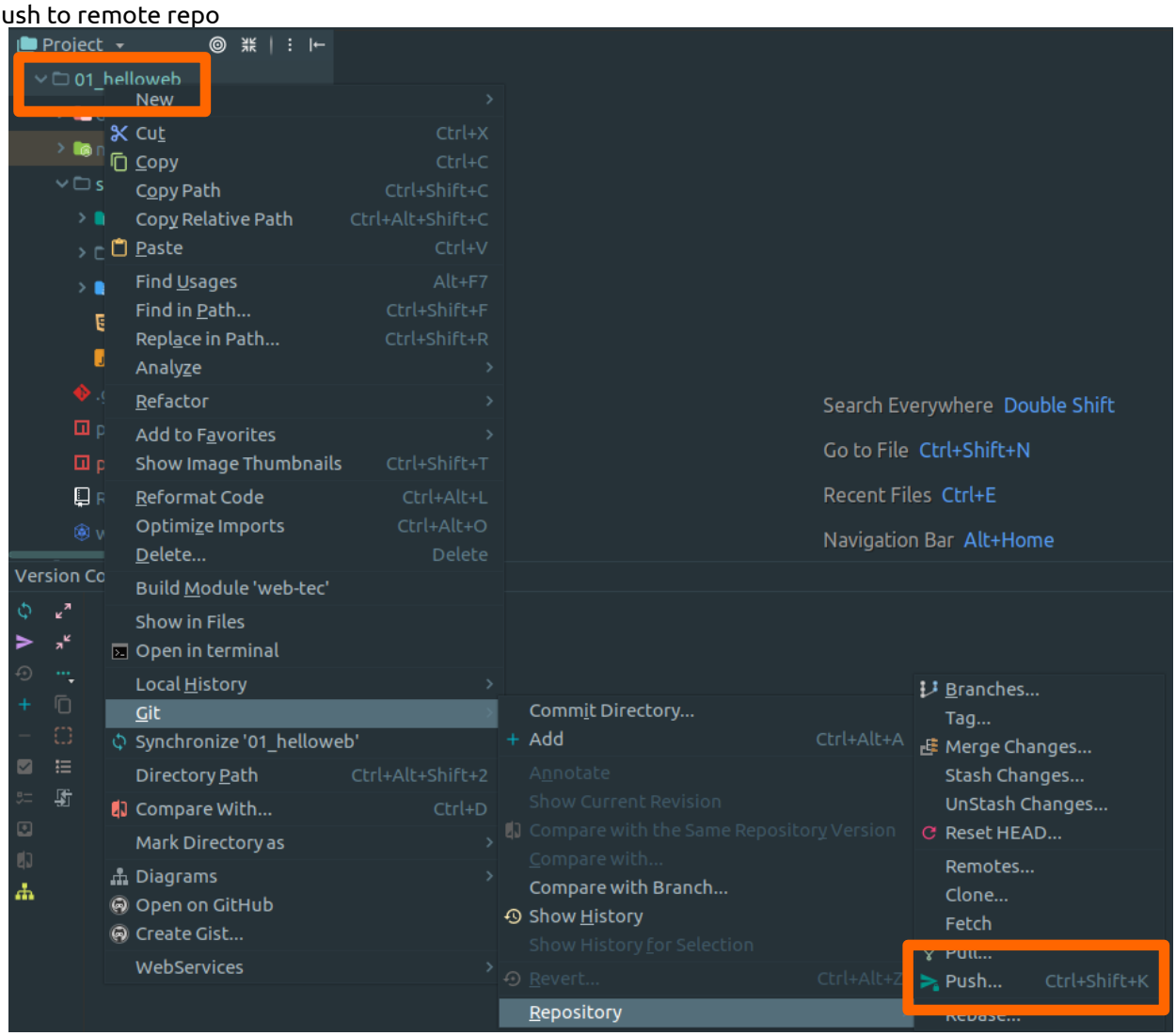
```
npm start
```

- open in IDE (ide-webserver) using chrome
- check the git settings (directory) in your IDE: settings -> version control
- add the current state to your github repo using IDE:
  - view -> tool windows -> version control -> commit to local repo


Version Control:
Local Changes
Log
Console
Default 7 files
.gitignore ~/IdeaProjects/web-tec/01_helloweb
helloWidget.js ~/IdeaProjects/web-tec/01_helloweb/src/scripts
index.html ~/IdeaProjects/web-tec/01_helloweb/src
index.js ~/IdeaProjects/web-tec/01_helloweb/src
package.json ~/IdeaProjects/web-tec/01_helloweb
README.md ~/IdeaProjects/web-tec/01_helloweb
webpack.config.js ~/IdeaProjects/web-tec/01_helloweb


Commit Changes
Changelist: Default
web-tec 7 files ~/IdeaProjects/web-tec
01_helloweb 7 files
src 3 files
scripts 1 file
helloWidget.js
index.html
index.js
.gitignore
package.json
README.md
webpack.config.js
New: 5 Modified: 2 Unversioned: 0 of 39132
Commit Message
Less simple version using NodeJS and its webpack..modules
Git
Author:
Amend commit
Sign-off commit
Before Commit
Reformat code
Rearrange code
Optimize imports
Perform code analysis
Check TODO (Show All) Configure
Cleanup
Update copyright
After Commit
Upload files to:
(none)
Always use selected server
Diff
Side-by-side viewer
Do not ignore
Highlight words
3 differences
c19702dc6c576593ef04324396df48e99790a604 (Read-...
Your version
COMMIT
CANCEL

push to remote repo



TODO: integrate eslint, babel, mocha, postman