Soft-Werke Learning Center. Frontend

# Overview

The learning plan consists of two main parts:

1. learning how to develop static web pages with HTML and CSS
2. writing client logic with JavaScript.

It does not teach the basics of HTML, CSS of JavaScript. The plan expects the you to already have that knowledge. Instead it focuses on how to write well-structured, maintainable code, using common development tools, techniques and practices.

The end goal is to prepare the learner for frontend development in real-world projects.

# Schedule

Effort values are estimated for a person with basic knowledge of writing HTML, CSS and JavaScript.

|  |  |
| --- | --- |
| **Lesson** | **Effort** |
| Part 1. Development Environment | 2 days |
| Part 2. HTML & CSS | 5 days |
| Part 3. JavaScript | 12 days |

# Part 1. Development Environment

### Theory

Look through the Tools and Coding Practices listed in the Addendum. Study each of them.

### Practice

1. Set up basic frontend development environment on your local machine.
2. Create a new frontend project.
3. Implement a simple web page. Functionality:
   1. Page shows text Hello World.
   2. Text is in burgundy.
   3. Text is a heading.
   4. Text is centered vertically and horizontally in the window.
4. Additional requirements (use Gulp):
   1. Page is accessible via localhost, port 3000.
   2. Page is updated with changes made in code, in real-time.
   3. CSS is minified.

Note: use the Tools listed in the Addendum.

# Part 2. HTML & CSS

### Task

Create a static landing website with several pages according to supplied designs.

**Note**: designs and other materials are provided separately.

Website requirements:

1. Responsive design
2. Cross-browser for last version of Chrome, Mozilla, Edge, IE11
3. No vertical window scrolling on desktop version
4. Use semantic HTML elements (verify with w3c validator)

Project requirements:

1. Website is accessible via localhost, port 3000
2. Pages are updated with changes made in code, in real-time.
3. Project can be built differently for development and production environments.
4. Build configuration for production only:
   1. Minify CSS
   2. Provide source maps for minified resources
   3. Optimize media resources (images etc.)
5. Integrate stylelint into your code editor

Development requirements:

1. Use the Coding Practices listed in the Addendum
2. Commit your code every day. Write sensible commit messages

See Addendum for tools and technologies used.

### Languages and libraries

* HTML
* CSS, LESS

# Part 3. JavaScript

### Task

Develop a simple single-page client-side Library application that allows visitors to work with Books.

Use Cases:

Anonymous user:

* can view information about public books
* can login to become an authorizes user

Authorized user:

* can view information about all books
* can add new books
* can delete books
* can logout to become an anonymous user

Primary entities:

1. Book – can be created and deleted.
   1. ID
   2. Position (sets order of books – 1, 2, 3, 4, …)
   3. Title
   4. Description
   5. Author (created by user)
   6. Created date
   7. Genre (from a pre-defined list; can have several)
   8. Public/private (one or the other)
2. User – a list of user accounts is constant for this app.
   1. ID
   2. Email
   3. Password
   4. Name (full name)

Functional overview:

1. New books
   1. User creates new books using a web form.
   2. Form is in the main content block of the page.
   3. Form is used for creating new books.
   4. Form is visible only to an authorized user.
   5. A new book can only be created by an authorized user.
   6. Form has validation applicable for its fields.
2. List of books
   1. List is in the main content block of the page.
   2. List has public and private books:
      1. Public books are viewable by any user.
      2. Private books are only viewable by authorized users.
   3. One row shows information on one book.
   4. Deletion of books
      1. Every row has a button to delete that book.
      2. Only authorized users see the button.
      3. An authorized user can delete any book, including those he did not create.
      4. When a book is deleted, it is removed from the list and the data storage.
   5. When a new book is added, it automatically appears in the list.
   6. Sorting of books
      1. Authorized user can change the position of a book by dragging its row to a new position
      2. Change of positions is persistent.
3. Counter
   1. Counter is in the header of the page.
   2. Counter shows how many books there are in total.
      1. Only public books are counted if the user is not authorized.
   3. The number in the counter is updated automatically when books are deleted or added.
4. Login
   1. Login form is in a modal window.
      1. To log in, user has to provide their email and password
      2. Authorization is processed on client-side
   2. Other controls are in the header.
   3. They are changed automatically according to the current state of the user.
   4. Logged out state
      1. Shows login button
      2. Login button opens the login modal
   5. Logged in state
      1. Shows logout button and a welcome text (Hello, <username>)
      2. Logout button logs the user out

Data storage:

1. Immutable (for this task) data, like user accounts and genres, are kept on the server.
2. Books are stored on client-side and can be shared between different users.
3. Session of an authorized user has a time limit.

Additional requirements:

* No page reloads.
* No CSS. For styling use Bootstrap classes.
* No ES6.
* Use MVC architectural pattern.
* No HTML in JavaScript code, use templates.
* For asynchronous calls use Promises.
* Focus on functionality rather than appearance.
* Use Design Patterns: Module and Mediator.
  + You can also use other Patterns, if you find them suitable for the task.

Start by designing the architecture.

Decompose the app into abstract modules.

What independent program components are there?

You can try drawing a component diagram to better visualize relations between components.

Note: due to lack of server-side logic the task has some simplifications and workarounds that are not present in real world applications:

* authentication of user on client side, including work with passwords
* storing content on client side
* etc.

### Languages and libraries

* HTML
* JavaScript (ECMAScript 5), jQuery 3
* Bootstrap 4

# Addendum

### Resources

An introductory guide for frontend developers is under revision and soon to be published. Wait for it.

Glossary for JavaScript:

<http://jargon.js.org/>

Library of refactoring techniques and design patterns:

<https://refactoring.guru/>

About JavaScript frontend architecture:

<https://www.innoarchitech.com/scalable-maintainable-javascript/>

### Full stack of technologies

* HTML
* CSS
* JavaScript (ECMAScript 5)
* LESS
* jQuery 3
* Bootstrap 4

### Coding Practices

#### Templating

Шаблонизация – нужно сразу приучаться писать HTML код на основе темплейтов (главная страница, внутренняя страница и т.п.) и шаблонов (header, footer и т.п.).

Рекомендую Nunjucks от Mozilla (для настройки потребуются Node.js и Gulp см. ниже):

<https://mozilla.github.io/nunjucks/>

#### CSS preprocessor

CSS препроцессоры – рекомендую использовать сразу в процессе, чтобы в дальнейшем быстро воспользоваться их приемуществами.

При этом сначала можно писать обычный CSS (чистый CSS понимает любой препроцессор) с использованием BEM-классов, а по мере вхождения можно добавлять использование переменных, mixins и т.п. В компании для большинства проектов используется LESS:

<http://lesscss.org/>

#### Code Structuring & Modularization

Структура HTML и CSS – необходимо выделять из дизайна UI компоненты и продумывать структурные имена классов.

Для CSS, лучше BEM-методологии трудно что-то предложить. Существуют разные вариации BEM, оптимальный вариант указан ниже:

<https://csswizardry.com/2013/01/mindbemding-getting-your-head-round-bem-syntax/>

Модульность в JavaScript помогает избавиться от спагетти-кода:

* <http://bradfrost.com/blog/post/atomic-web-design/>
* <http://www.adequatelygood.com/JavaScript-Module-Pattern-In-Depth.html> or <https://habr.com/post/117069/>
* <https://www.berndtgroup.net/thinking/blog/development/modularizing-your-front-end-code-for-long-term-maintainability-and-sanity>

#### Design Patterns

Шаблон или паттерн проектирования – повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

* <https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F>
* <https://addyosmani.com/resources/essentialjsdesignpatterns/book/>
* <https://addyosmani.com/largescalejavascript/>

### Tools

#### Code editor / IDE

Полноценный редактор для HTML, CSS, JavaScript кода (не блокнот, не Notepad++).

* <https://www.sublimetext.com/>
* <https://code.visualstudio.com/>

#### Version Control System

Система контроля версий кода Git. В рамках практики для хранения кода используется сервер GitLab, развернутый в сети Soft-Werke.

* <https://git-scm.com/>
* <https://vmgit.softwerke.local/>

Git How To:

* <https://guides.github.com/introduction/git-handbook/>
* <https://chris.beams.io/posts/git-commit/>

#### Build Tool

Автоматизация процесса разработки – все этапы разработки д.б. максимально автоматизированы (компиляция HTML из шаблонизатора, компиляция LESS,

автоматическая перезагрузка страниц, генерация SourceMap для CSS и т.п.). Для быстрой настройки автоматизации лучше всего подходит Gulp (требующий

некоторых базовых знаний JavaScript), кроме того практически для всех Frontend инструментов нужен Node.js:

* <https://nodejs.org/en/>
* <https://gulpjs.com/>

#### Code Linters

Статическая проверка кода на возможные ошибки и соответствие стандартам и практикам написания кода.

* <https://stylelint.io/>
* <https://eslint.org/>

#### \* Adobe Photoshop

Редактор изображений. В некоторых проектах первым шагом разработки является нарезка PSD файла на изображения, которые будут использованы в дизайне.

Не входит в план обучения как обязательная часть.