**Project tasks**

**1. "Розумний будинок"**

1) Розробіть архітектуру програмного забезпечення, що реалізує моделювання системи управління компонентами "розумного будинку".

2) Функціональні вимоги до системи моделювання компонентів "розумного будинку":

- в системі має бути змодельоване не менше 3 типів компонент "розумного будинку", наприклад: світильник, обігрів, жалюзі, сигналізація та ін.;

- інтерфейс програми: Web-інтерфейс на основі роботи з DOM;

- управління програмою: за допомогою графічного Web-інтерфейсу, який повинен дозволяти:

а) конфігурувати "розумний будинок" (додавати, видаляти компоненти);

б) керувати станом компонент "розумного будинку" (вмикати/вимикати, закривати/відкривати компоненти та інші команди);

в) вітаються додаткові команди на розсуд розробника;

- в процесі виконання, програма видає інформацію про поточний стан всіх компонент "розумного будинку".

3) Розробіть програму, яка моделює роботу системи управління компонентами "розумного будинку".

4) Не функціональні вимоги до програми:

- при розробці програми повинен **використовуватися об'єктно-орієнтований підхід з підтримкою всіх можливостей ООП**.

5) Система моделювання (управління) розумним будинком в складі мінімум 3 типів компонентів (як зазначено в функціональних вимогах) повинна включати 1 обов'язковий компонент з розширеними можливостями управління. Наприклад, телевізор, програмна модель якого включає всі можливі стани сутності "телевізор" і можливості для керування ним, наприклад: включення / вимикання, отримання списку каналів, перемикання каналів, пошук каналів, налаштування телевізора і т.д.

2. **Project “Game collection”**

Implement classes on diagrams in JavaScript using ES6 syntax.

BaseName should have such fields:

* Title – contains the name of game;
* minMaxPlayers – contains the minimal and maximal count of players;
* age – contains the minimal age of players;
* rating[] – this array contains all marks, set to the game by users;
* genre[] – this array should contain the list of categories which the game belongs to.

Also this class should constructor that accepts one parameter of String type(it sets title).

BaseGame should have next methods:

* getFullInfo() – shows the whole information about object;
* setMyRaiting(Number) – lets to set your mark for the game;
* fillDate() – lets to input the whole information about the game;
* addGenre(String) – lets to attach the new category to the list of categories;
* countRaiting() – lets count the average rating.

Class Expansions should be derived from BaseGame. So it has all base class characteristics. Except them it should have his own field content which contains the difference this extension from the base game. Also it has method getBaseGameName() which return the base game’s name.

Collection should have such fields:

* gameList – should contains the list of all games and their extensions (instances of previous classes);
* collectionName – contains the name of collection.

Also this class should constructor that accepts one parameter of String type(it sets name).

Collection should have next methods:

* addItem() – lets to add new item (instance of previous class) to collection;
* isInCollection() – checks if some item is in collection;
* countList() – lets to calculate the count of games in collection;
* countBaseGames() – lets to calculate the count of Base Games in collection;
* countExpansions() – lets to calculate the count of Base Games in collection.

|  |
| --- |
| **BaseGame** |
| + title |
| + minMaxPlayers |
| + playingTime |
| + age |
| - raiting [] |
| + genre[] |
|  |
| + constructor(String) |
| + getFullInfo() |
| + setMyRaiting(Number) |
| + fillData() |
| + addGenre(String) |
| + countRaiting() |

|  |
| --- |
| **Expansions(BaseGame)** |
| + content |
|  |
| + getBaseGameName() |

|  |
| --- |
| **Collection** |
| + gameList[] |
| + collectionName |
|  |
| + constructor(String) |
| + addItem() |
| + isInCollection() |
| + countList() |
| + countBaseGames() |
| + countExpansions() |

**3. Project “LocalStorage”**

Розробити Web-додаток форми реєстрації користувача для пошуку роботи – сторінка містить поля для введення користувацьких даних (прізвище, ім’я, вік, освіта, бажана посада). Додати можливість зберігання даних користувача на стороні клієнта (використовуючи LocalStorage), а також їх вичитування з локальної бази**.**

***Вимоги:***

1. Додати функцію, яка перевіряє наявність доступу до мережі.
2. Якщо функція повертає false - дані читаються та записуються в localStorage. Наприклад, нові дані користувача будуть спочатку записуватись в localStorage, а вже потім, при наявності мережі (інтернету) додаватись на сторінку. При першому відкриванні сторінки має перевірятись наявність даних в localStorage та виконуватися промальовування їх на сторінці.
3. Якщо функція повертає true, то дані мають відправлятись на сервер і читатись з сервера. В цьому випадку дозволяється додати функцію, яка **емулюватиме** відправку даних на сервер та читання з сервера.
4. При відкриванні сторінки має реєструватись Event Listener, що слухатиме зміну статуса мережі. Якщо користувач виходить онлайн, то на зміну статуса має спрацювати функція, що витягне дані з LocalStorage та опублікує на сторінці.
5. Після публікації даних на сторінці вони видаляються із LocalStorage.

**4. Project “IndexedDB”**

Розробити Web-додаток форми реєстрації користувача для пошуку роботи – сторінка містить поля для введення користувацьких даних (прізвище, ім’я, вік, освіта, бажана посада). Додати можливість зберігання даних користувача на стороні клієнта (використовуючи IndexedDB), а також їх вичитування з локальної бази**.**

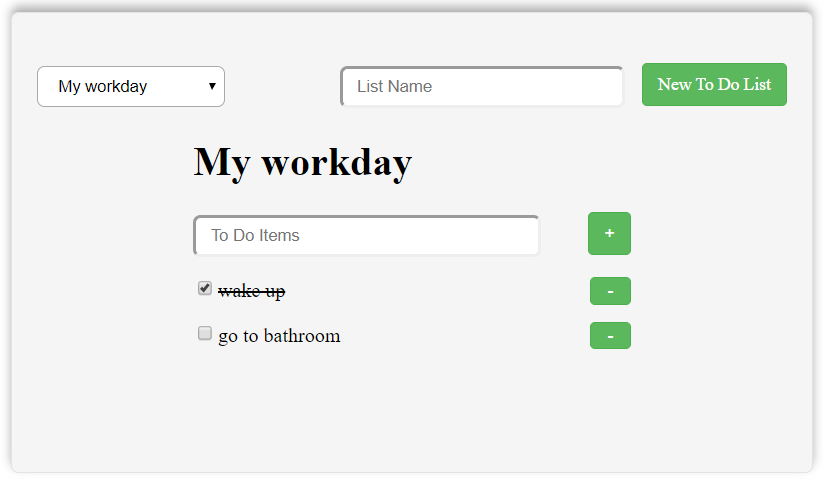
***Вимоги:***

1. Додати функцію, яка перевіряє наявність доступу до мережі.
2. Якщо функція повертає false - дані читаються та записуються в IndexedDB. Наприклад, нові дані користувача будуть спочатку записуватись в IndexedDB, а вже потім, при наявності мережі (інтернету) додаватись на сторінку. При першому відкриванні сторінки має перевірятись наявність даних в IndexedDB та виконуватися промальовування їх на сторінці.
3. Якщо функція повертає true, то дані мають відправлятись на сервер і читатись з сервера. В цьому випадку дозволяється додати функцію, яка **емулюватиме** відправку даних на сервер та читання з сервера.
4. При відкриванні сторінки має реєструватись Event Listener, що слухатиме зміну статуса мережі. Якщо користувач виходить онлайн, то на зміну статуса має спрацювати функція, що витягне дані з IndexedDB та опублікує на сторінці.

Після публікації даних на сторінці вони видаляються із IndexedDB.

**5. Project “To-Do-list”**

Створити web-аплікацію, яка дозволятиме формувати списки справ, які треба виконати на протязі дня. Необхідно передбачити можливість створення різної кількості списків справ. Для збереження даних використайте LocalStorage.

****

**6. App-calculator (using eval()).**

**7. Project “**[**Tic Tac Toe**](https://medium.com/front-end-weekly/tic-tac-toe-javascript-game-b0cd6e98edd9) **game”**

**. . . . .**