**Критерии приема**

* Создан репозиторий goit-js-hw-06.
* При сдаче домашней работы есть две ссылки: на исходные файлы и рабочую страницу на GitHub Pages.
* Задания выполнены строго по ТЗ (нельзя изменять исходный HTML задания).
* При посещении живой страницы задания, в консоли нету ошибок и предупреждений.
* Имена переменных и функций понятные, описательные.
* Код отформатирован Prettier.

**Стартовые файлы**

[Скачай стартовые файлы](https://downgit.github.io/#/home?url=https://github.com/goitacademy/javascript-homework/tree/main/v2/06/src) с готовой разметкой и подключенными файлами скриптов для каждого задания. Скопируй их себе в проект.

**Задание 1**

В HTML есть список категорий ul#categories.

<ul id="categories">

<li class="item">

<h2>Animals</h2>

<ul>

<li>Cat</li>

<li>Hamster</li>

<li>Horse</li>

<li>Parrot</li>

</ul>

</li>

<li class="item">

<h2>Products</h2>

<ul>

<li>Bread</li>

<li>Prasley</li>

<li>Cheese</li>

</ul>

</li>

<li class="item">

<h2>Technologies</h2>

<ul>

<li>HTML</li>

<li>CSS</li>

<li>JavaScript</li>

<li>React</li>

<li>Node.js</li>

</ul>

</li>

</ul>

Напиши скрипт который:

1. Посчитает и выведет в консоль количество категорий в ul#categories, то есть элементов li.item.
2. Для каждого элемента li.item в списке ul#categories, найдет и выведет в консоль текст заголовка элемента (тега <h2>) и количество элементов в категории (всех вложенных в него <li>).

В результате, в консоли будут выведены такие сообщения.

Number of categories: 3

Category: Animals

Elements: 4

Category: Products

Elements: 3

Category: Technologies

Elements: 5

**Задание 2**

В HTML есть пустой список ul#ingredients.

<ul id="ingredients"></ul>

В JavaScript есть массив строк.

const ingredients = [

'Potatoes',

'Mushrooms',

'Garlic',

'Tomatos',

'Herbs',

'Condiments',

];

Напиши скрипт, который для каждого элемента массива ingredients:

1. Создаст отдельный элемент <li>. Обзательно используй метод document.createElement().
2. Добавит название ингредиента как его текстовое содержимое.
3. Добавит элементу класс item.
4. После чего вставит все <li> за одну операцию в список ul.ingredients.

**Задание 3**

Напиши скрипт для создания галереи изображений по массиву данных. В HTML есть список ul.gallery.

<ul class="gallery"></ul>

Используй массив объектов images для создания элементов <img> вложенных в <li>. Для создания разметки используй шаблонные строки и метод insertAdjacentHTML().

* Все элементы галереи должны добавляться в DOM за одну операцию вставки.
* Добавь минимальное оформление галереи флексбоксами или гридами через CSS классы.

const images = [

{

url: 'https://images.pexels.com/photos/140134/pexels-photo-140134.jpeg?dpr=2&h=750&w=1260',

alt: 'White and Black Long Fur Cat',

},

{

url: 'https://images.pexels.com/photos/213399/pexels-photo-213399.jpeg?dpr=2&h=750&w=1260',

alt: 'Orange and White Koi Fish Near Yellow Koi Fish',

},

{

url: 'https://images.pexels.com/photos/219943/pexels-photo-219943.jpeg?dpr=2&h=750&w=1260',

alt: 'Group of Horses Running',

},

];

**Задание 4**

Счетчик состоит из спана и кнопок, которые, при клике, должны увеличивать и уменьшать его значение на единицу.

<div id="counter">

<button type="button" data-action="decrement">-1</button>

<span id="value">0</span>

<button type="button" data-action="increment">+1</button>

</div>

* Создай переменную counterValue в которой будет храниться текущее значение счетчика и инициализируй её значением 0.
* Добавь слушатели кликов на кнопки, внутри которых увеличивай или уменьшай значение счтетчика.
* Обновляй интерфейс новым значением переменной counterValue.

**Задание 5**

Напиши скрипт который, при наборе текста в инпуте input#name-input (событие input), подставляет его текущее значение в span#name-output. Если инпут пустой, в спане должна отображаться строка "Anonymous".

<input type="text" id="name-input" placeholder="Please enter your name" />

<h1>Hello, <span id="name-output">Anonymous</span>!</h1>

**Задание 6**

Напиши скрипт, который при потере фокуса на инпуте (событие blur), проверяет его содержимое на правильное количество введённых символов.

<input

type="text"

id="validation-input"

data-length="6"

placeholder="Please enter 6 symbols"

/>

* Сколько символов должно быть в инпуте, указывается в его атрибуте data-length.
* Если введено подходящее количество символов, то border инпута становится зелёным, если неправильное - красным.

Для добавления стилей, используй CSS-классы valid и invalid, которые мы уже добавили в исходные файлы задания.

#validation-input {

border: 3px solid #bdbdbd;

}

#validation-input.valid {

border-color: #4caf50;

}

#validation-input.invalid {

border-color: #f44336;

}

**Задание 7**

Напиши скрипт, который реагирует на изменение значения input#font-size-control (событие input) и изменяет инлайн-стиль span#text обновляя свойство font-size. В результате при перетаскивании ползунка будет меняться размер текста.

<input id="font-size-control" type="range" min="16" max="96" />

<br />

<span id="text">Abracadabra!</span>

**Задание 8**

Напиши скрипт управления формой логина.

<form class="login-form">

<label>

Email

<input type="email" name="email" />

</label>

<label>

Password

<input type="password" name="password" />

</label>

<button type="submit">Login</button>

</form>

1. Обработка отправки формы form.login-form должна быть по событию submit.
2. При отправке формы страница не должна перезагружаться.
3. Если в форме есть незаполненные поля, выводи alert с предупреждением о том, что все поля должны быть заполнены.
4. Если пользователь заполнил все поля и отправил форму, собери значения полей в обьект, где имя поля будет именем свойства, а значение поля - значением свойства. Для доступа к элементам формы используй свойство elements.
5. Выведи обьект с введенными данными в консоль и очисти значения полей формы методом reset.

**Задание 9**

Напиши скрипт, который изменяет цвета фона элемента <body> через инлайн стиль при клике на button.change-color и выводит значение цвета в span.color.

<div class="widget">

<p>Background color: <span class="color">-</span></p>

<button type="button" class="change-color">Change color</button>

</div>

Для генерации случайного цвета используй функцию getRandomHexColor.

function getRandomHexColor() {

return `#${Math.floor(Math.random() \* 16777215).toString(16)}`;

}

**Задание 10 (выполнять не обязательно)**

Напиши скрипт создания и очистки коллекции элементов. Пользователь вводит количество элементов в input и нажимает кнопку Создать, после чего рендерится коллекция. При нажатии на кнопку Очистить, коллекция элементов очищается.

<div id="controls">

<input type="number" min="1" max="100" step="1" />

<button type="button" data-create>Create</button>

<button type="button" data-destroy>Destroy</button>

</div>

<div id="boxes"></div>

Создай функцию createBoxes(amount), которая принимает один параметр - число. Функция создает столько <div>, сколько указано в amount и добавляет их в div#boxes.

1. Размеры самого первого <div> - 30px на 30px.
2. Каждый элемент после первого, должен быть шире и выше предыдущего на 10px.
3. Все элементы должены иметь случайный цвет фона в формате HEX. Используй готовую функцию getRandomHexColor для получения цвета.

function getRandomHexColor() {

return `#${Math.floor(Math.random() \* 16777215).toString(16)}`;

}

Создай функцию destroyBoxes(), которая очищает содержимое div#boxes, тем самым удаляя все созданные элементы.