ГКП НА ПХВ «ВЫСШИЙ КОЛЛЕДЖ»

«ASTANA POLYTECHNIC» АКИМАТА ГОРОДА АСТАНЫ

ОТЧЕТ

ПО УЧЕБНОЙ ПРАКТИКЕ

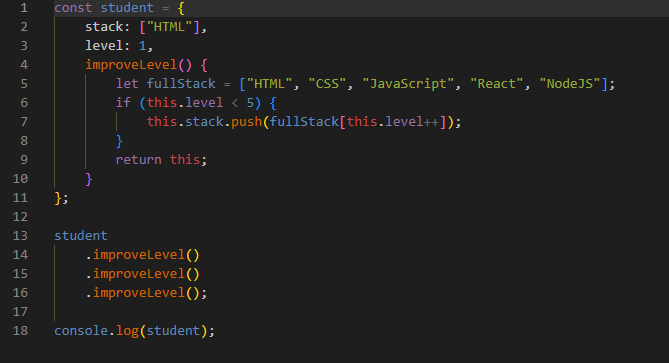
Практика по Веб программирование

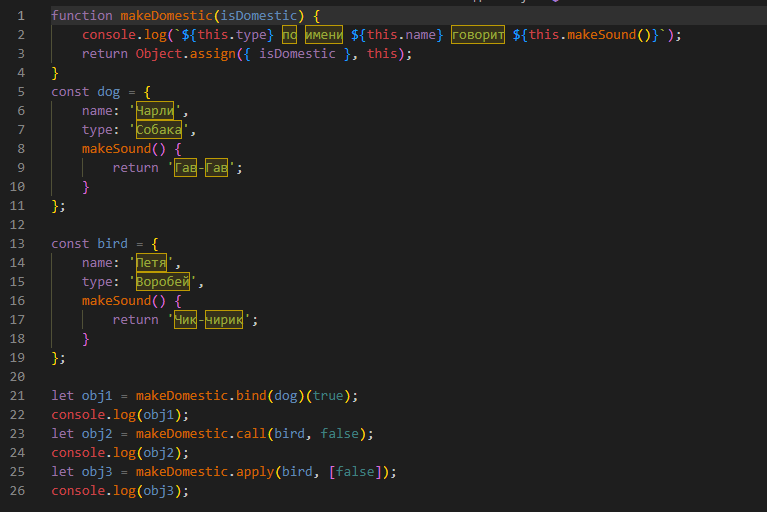
Выполнил(а): Калыбай Алихан Группа: Ис-21-15рб

Астана, 2022 г.

**День 1**

**Задание #1**  
Представьте, что вас как веб-разработчика попросили разработать веб-платформу, которая будет отслеживать прогресс студентов в обучении веб-разработке.  
Для начала вам необходимо создать объект student, в котором будут 3 свойства:  
1. stack. Массив из строк, где каждая строка - это технология. Изначально, массив stack должен быть равен [‘HTML’].  
2. level. Тип данных number. level отвечает за текущий уровень студента и изначально равен 1.  
3. improveLevel. Функция, которая сначала увеличивает значение свойства level у студента на единицу. Если level равен 2-м, то вам необходимо добавить в конец массива stack значение ‘CSS’, если 3-м - добавляете в конец stack значение ‘JavaScript’, 4-м - ‘React’, 5-ти - ‘NodeJS’. Если значение level стало больше 5-ти, то вам необходимо вывести в модальном окне через alert сообщение “Студент выучил все технологии!”. Также функция improveLevel должна возвращать в самом конце обновленный объект student.  
В итоге значение свойства stack после выполнения кода ниже должно быть равно ["HTML", "CSS", "JavaScript", "React", "NodeJS"].  
student  
.improveLevel()  
.improveLevel()  
.improveLevel()  
.improveLevel()  
.improveLevel()  
Примечание: цепочка из подряд идущих функций impoveLevel работает так, потому что improveLevel возвращает объект student, а у объекта student есть метод impoveLevel.  
Посмотрите на пример:  
const student = {  
improveLevel() {  
return this;  
}  
};  
student  
.improveLevel()  
.improveLevel()  
  
.improveLevel();

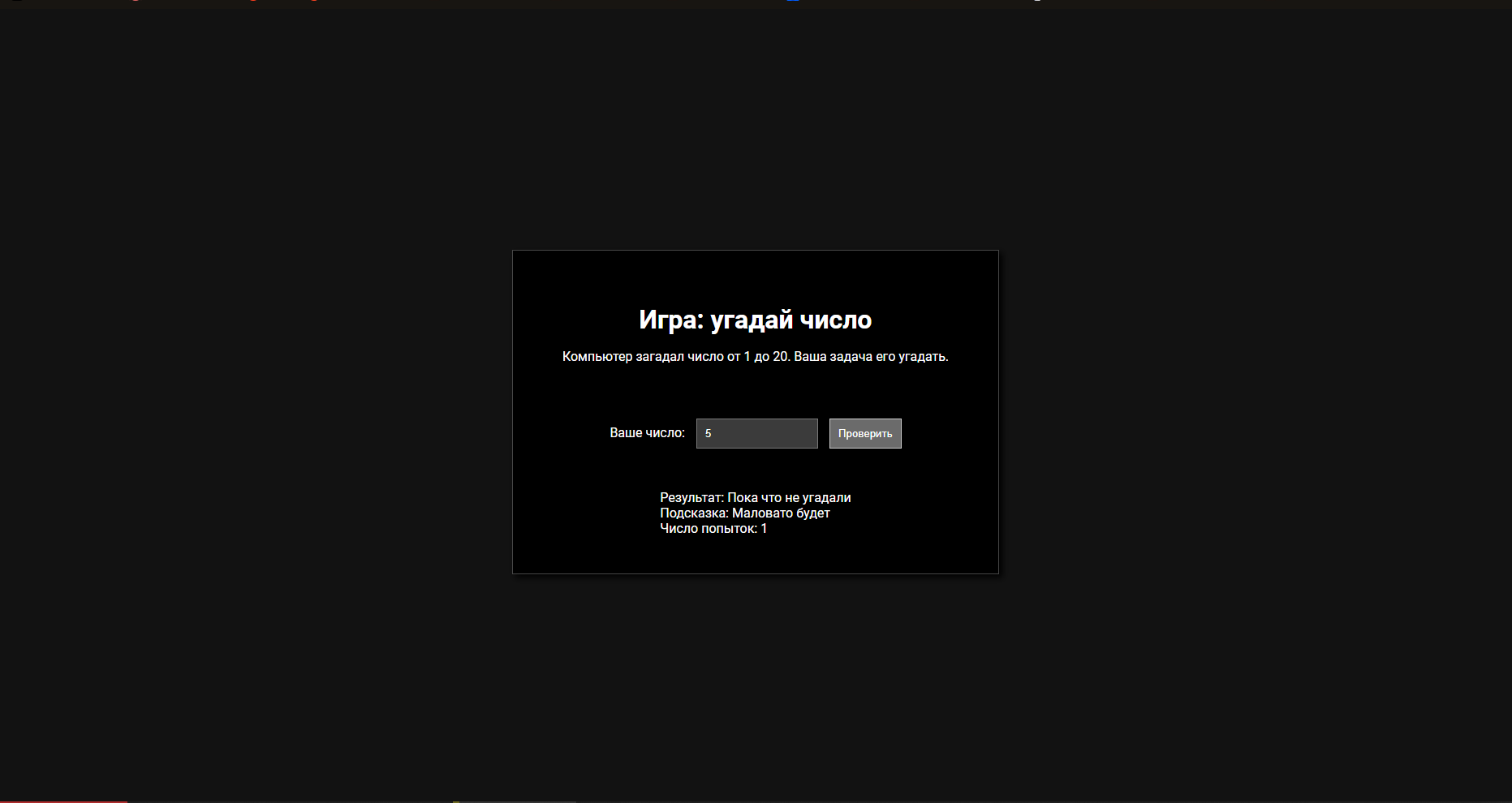
  
**Задание #2**  
У вас есть 2 объекта dog и bird, в которых содержится их описание.  
const dog = {  
name: 'Чарли',  
type: 'Собака',  
makeSound() {  
return 'Гав-Гав';  
}  
}  
const bird = {  
name: 'Петя',  
type: 'Воробей',  
makeSound() {  
return 'Чик-чирик';  
}  
}  
Вам необходимо создать функцию makeDomestic, которая будет работать с ключевым словом this (пример использования ниже). Она должна выводить в консоль всю информацию о животном с помощью сообщения: “type по имени name говорит sound”, где name - имя животного, type - его тип и sound - результат вызова функции makeSound.  
Также функция makeDomestic должна возвращать новый объект животного, в котором будут все прошлые параметры (name, type, makeSound), а также будет добавлен новый параметр isDomestic. Он является типом данных boolean и показывает, домашнее ли животное или нет. Значение для свойства isDomestic (true либо false) вам необходимо передать как параметр в функцию makeDomestic.  
Кроме этого для обращения к свойствам животного функция makeDomestic должна использовать только this. Т.е. вам разрешено обращаться к свойствам только через this.name либо this.makeSound, но никак не bird.name либо bird.makeSound.  
Вам необходимо решить эту задачу 3-мя способами: через bind, call и apply. Вызовите функцию makeDomestic 3 раза используя данные методы. При использовании bind, функция makeDomestic должна работать с объектом dog, при apply и call - с объектом  
bird. Не забывайте передавать параметр isDomestic в функцию при ее вызове.  
Примеры использования:  
function makeDomestic(isDomestic) {  
// Ваша реализация  
}  
/\*  
Сообщение в консоли: "Собака по имени Чарли говорит Гав-Гав"  
domesticDog = {  
name: 'Чарли',  
type: 'Собака',  
isDomestic: true,  
makeSound() {  
return 'Гав-Гав'  
}  
}  
  
\*/

  
**Задание #3**  
Дан объект footballer, в котором есть некоторая информация о футболисте и действия, которые он может выполнять.  
const footballer = {  
fullName: 'Cristiano Ronaldo',  
attack: () => {  
console.log(`${this.fullName} сейчас с мячом и начинает атаку!`);  
},  
scoreGoal(sound) {  
console.log(`${this.fullName} забил гол! Вот это да!`);  
this.celebrate(sound);  
},  
celebrate(sound) {  
console.log(sound);  
},  
goToSubstitution: function(newPlayer) {  
console.log(`${this.fullName} уходит на замену. На поле выходит ${newPlayer}`);  
}  
};  
const attack = footballer.attack;  
const score = footballer.scoreGoal;  
const substitute = footballer.goToSubstitution;  
attack();  
score('Сиииии');  
substitute('Paulo Dibala');  
К сожалению, данный код работает неверно. В некоторых случаях он выводит undefined, а в других - вообще ошибку.  
Вам необходимо исправить данный код используя bind, call и apply.  
● Функцию attack необходимо исправить при помощи bind + нужно вспомнить особенность стрелочных функций (нет своего this)  
● Функцию score - при помощи call  
● Функцию substitute - при помощи apply

****

**День 2**

Напишите логику игры Угадай число!  
Вам предоставляется готовая верстка интерфейса игры.  
Ваша задача разобраться с разметкой и стилями страницы игры и разработать функционал.  
  
Требования  
Число генерируется случайно в диапазоне от 1 до 20.  
На старте количество очков составляет 20 и при каждом неверном ответе уменьшается на единицу.  
Лучший результат сохраняется до конца игры (Пока не обновят страницу).  
Когда игрок угадает число выводится соответствующее сообщение и игра начинается заново.  
Если игрок не угадал число то выводится подсказка: "Слишком много", если число больше сгенерированного или "Слишком мало", если число меньше сгенерированного.

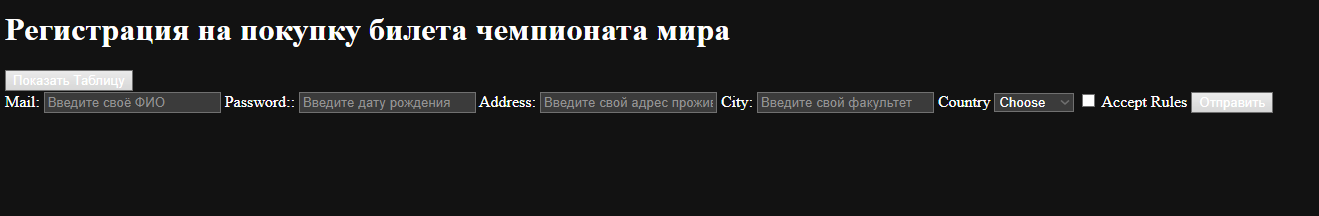
****

**День 3**

Реализуйте компонент <MyForm> отображающий форму из шести элементов:

* email - инпут типа email
* password - инпут типа password
* address - textarea
* city - текстовый инпут
* country - select со следующими значениями: argentina, russia, china.
* Accept Rules - checkbox

После сабмита формы, появляется таблица в которой показываются значения всех полей. Из этой формы можно вернуться в редактирование по кнопке . При этом все данные должны оказаться на своих местах.  
  
Строки сортируются в алфавитном порядке по именам в первом столбце. В вашем случае результирующая таблица может отличаться, все зависит от того какие данные выбраны.  
  
Пример формы:  
<form name="myForm">  
<div class="form-row">  
<div class="form-group col-md-6">  
<label for="email" class="col-form-label">Email</label>  
<input type="email" name="email" class="form-control" id="email" placeholder="Email">  
</div>  
<div class="form-group col-md-6">  
<label for="password" class="col-form-label">Password</label>  
<input type="password" name="password" class="form-control" id="password" placeholder="Password">  
</div>  
</div>  
 <div class="form-group">  
    <label for="address" class="col-form-label">Address</label>  
    <textarea type="text" class="form-control" name="address" id="address" placeholder="1234 Main St"></textarea>  
  </div>  
  <div class="form-row">  
    <div class="form-group col-md-6">  
      <label for="city" class="col-form-label">City</label>  
      <input type="text" class="form-control" name="city" id="city">  
    </div>  
    <div class="form-group col-md-6">  
      <label for="country" class="col-form-label">Country</label>  
      <select id="country" name="country" class="form-control">  
        <option>Choose</option>  
        <option value="argentina">Argentina</option>  
        <option value="russia">Russia</option>  
        <option value="china">China</option>  
      </select>  
    </div>  
  </div>  
  <div class="form-group">  
    <div class="form-check">  
      <label class="form-check-label" for="rules">  
        <input id="rules" type="checkbox" name="acceptRules" class="form-check-input">  
        Accept Rules  
      </label>  
    </div>  
  </div>  
  <button type="submit" class="btn btn-primary">Sign in</button>  
</form>  
  
**После отправки формы:**  
<div>  
  <button type="button" class="btn btn-primary">Back</button>  
  <table class="table">  
    <tbody>  
      <tr>  
        <td>acceptRules</td>  
        <td>true</td>  
      </tr>  
      <tr>  
        <td>address</td>  
        <td>lenina street</td>  
      </tr>  
      <tr>  
        <td>city</td>  
        <td>moscow</td>  
      </tr>  
      <tr>  
        <td>country</td>  
        <td>russia</td>  
      </tr>  
      <tr>  
        <td>email</td>  
        <td>[my@email.com](https://mail.google.com/mail/?view=cm&fs=1&to=my%40email.com&authuser=2)</td>  
      </tr>  
      <tr>  
        <td>password</td>  
        <td>qwerty</td>  
      </tr>  
    </tbody>  
  </table>  
</div>

****

**День 4-5**

**Задание #1**  
Вам дан HTML код.  
Ваши коллеги разработчики реализовали систему, благодаря которой можно получать список пользователей по url: <https://jsonplaceholder.typicode.com/users>.  
Вам необходимо получить всех пользователей с помощью fetch и добавить данные о имени каждого пользователя внутрь html-элемента с id равным “data-container”.  
Для удобства необходимо добавить элемент span с текстом “Загрузка...” перед загрузкой пользователей, и спрятать этот элемент после загрузки данных о пользователях.  
Шаблон для HTML-элемента пользователя выглядит следующим образом:  
<li><a href="#">Имя пользователя</a></li>  
Примечание: обязательно не забывайте прописывать блоки catch. В них просто  
выводите ошибку в консоль при помощи console.error.  
 **Задание #2**  
Вы молодец! Если вы дошли до этого задания, то вы умеете получать данные и отображать их в HTML. Сейчас же задача будет посложнее.  
Вам необходимо создать функцию getUsersByIds, которая будет принимать массив с id пользователей. Вам нужно получить всех пользователей, у которых есть данные значения id. Используйте некоторый код из предыдущего задания и Promise.all для решения поставленной задачи.  
Добавьте данные о имени каждого пользователя внутрь html-элемента с id равным “data-container” . Также для удобства необходимо добавить элемент span с текстом “Загрузка...” перед загрузкой пользователей, и спрятать этот элемент после загрузки данных о пользователях.  
Шаблон для HTML-элемента пользователя выглядит следующим образом:  
<li><a href="#">Имя пользователя</a></li>  
Для тестирования функции getUsersByIds используйте данный код:  
getUsersByIds([5, 6, 2, 1])  
  
Примечание: обязательно не забывайте прописывать блоки catch. В них просто выводите ошибку в консоль при помощи console.error.  
  
**Задание #3**  
Представьте, что у вас появился новый проект, где есть следующая функциональность: отображение фотографии, которая быстрее всего загрузилась.  
Создайте функцию getFastestLoadedPhoto, которая принимает в себя 1 параметр ids, являющийся массивом параметров id у объекта photo. Чтобы получить информацию о фото, вам необходимо использовать следующий url <https://jsonplaceholder.typicode.com/photos/1> (1 - это id фотографии). С помощью массива ids получите данные о фотографии, которая быстрее всего загрузилась при fetch запросе. Для решения поставленной задачи используйте Promise.race .  
Для удобства необходимо добавить элемент span с текстом “Загрузка...” перед загрузкой фотографии, и спрятать этот элемент после загрузки данных.  
Для создания HTML-элемента фотографии используйте данный шаблон:  
<li class="photo-item">  
<img class="photo-item\_\_image" src="[https://via.placeholder.com/600/92c952"&gt](https://via.placeholder.com/600/92c952%22&gt);  
<h3 class="photo-item\_\_title">  
accusamus beatae ad facilis cum similique qui sunt  
</h3>  
</li>  
В src у img должно быть подставлено значение свойства url у фотографии, а в тег h3 - значение свойства title. Добавляйте конечную фотографию в элемент с id равным “data-container”.  
Для теста функции getFastestLoadedPhoto используйте данный код:  
getFastestLoadedPhoto([60, 12, 55])  
Примечание: обязательно не забывайте прописывать блоки catch. В них просто выводите ошибку в консоль при помощи console.error

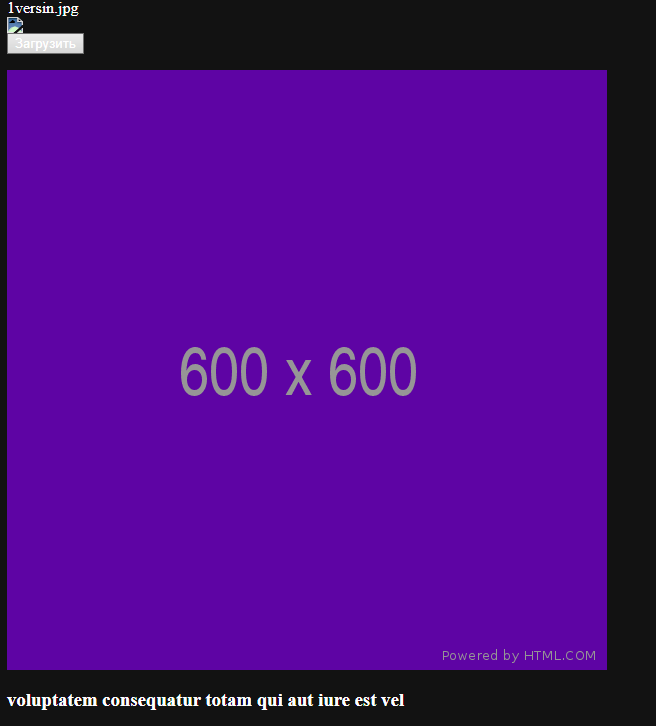
1.

****

**2.**

****

**3.**

****