МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

\_\_\_\_\_\_\_\_ Лабораторная работа №2\_\_\_\_\_\_\_\_\_\_\_

(наименование темы проекта или работы)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

(курсовому проекту, ОТЧЕТ по лабораторной работе)

по дисциплине

Информатика и компьютерные технологии \_

(наименование дисциплины)

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_Савкин А.Е.\_\_ \_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Давыдов М. А.\_\_\_\_\_

(подпись) (фамилия, и.,о.)

\_\_\_24-ИСз\_\_\_\_\_\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Задание к выполнению лабораторной работы № 2**

**Реализация вебсайта**

**Задание на лабораторную работу:**

Сделать вебсайт используя HTML, CSS, JS

**Цель работы:**

Научиться использовать современные технологии веб разработки.

**Ход работы:**

Написал карточную игру, цель которой найти пары карточек с одинаковой лицевой стороной. Также сделал главное меню для игры.

Сначала хочу остановиться на реализации навигации между игровым полем и главным меню. Каждый раз, когда мы перемещаемся, создается новый контейнер с контентом, а старый удаляется. Анимация перемещения достигается сменой классов CSS.

Сначала мы удаляем тот контейнер, который у нас уже есть. Но не сразу, а после того, как закончится анимация перемещения:

function destroySlidingWindow(direction){

  // later, we should probably make sure the direction starts with an upper case

  const slidingWindow = document.querySelector(".slidingWindow");

  slidingWindow.setAttribute("class",

    `slidingWindow slidingWindow\_destroyToThe${direction}`);

  setTimeout(()=>{

    slidingWindow.remove();

  }, computeAnimationDuration(slidingWindow));

}

Далее мы создаем новый контейнер, который будет выезжать сбоку. Направление задается переменной direction.

*function createSlidingWindow(direction){*

// later, we should probably make sure the direction starts with an upper case

  let slidingWindow = document.createElement("div");

  slidingWindow.setAttribute("class",

    `slidingWindow slidingWindow\_comingToThe${direction}`);

  let body = document.getElementById("body");

  let dialogWindow = document.querySelector(".dialogWindowContainer");

  body.insertBefore(slidingWindow, dialogWindow);

  return slidingWindow;

}

А вот так, например, выглядит класс контейнера, который приходит справа налево:

.slidingWindow\_comingToTheLeft {

    animation-name: comingToTheLeft;

}

@keyframes comingToTheLeft {

    from {

        translate: 100vw 0;

      }

      to {

        translate: 0 0;

      }

}

Сейчас покажу, как создаются карты. Сначала мы создаем HTML элементы с их CSS классами:

function makeADeck(numberOfCards) {

  let fileNames = makeFileNamesList(numberOfCards);

  let gamingTable = document.createElement("div");

  gamingTable.setAttribute("class", "gamingTable");

  const slidingWindow = createSlidingWindow("Left");

  slidingWindow.appendChild(gamingTable);

  for (let i = 0; i < numberOfCards; i++) {

    let flipcard = document.createElement("div");

    let flipcard\_\_frontSide = document.createElement("div");

    let flipcard\_\_backSide = document.createElement("div");

    let flipcard\_\_picture = document.createElement("img");

    flipcard.className = "flipcard";

    flipcard\_\_frontSide.className = "flipcard\_\_frontSide\_obscure";

    flipcard\_\_backSide.className = "flipcard\_\_backSide\_obscure";

    flipcard\_\_picture.className = "flipcard\_\_picture";

    flipcard\_\_picture.setAttribute("src", fileNames[i]);

    flipcard\_\_picture.setAttribute('draggable', false);

    flipcard\_\_frontSide. appendChild(flipcard\_\_picture);

    flipcard.appendChild(flipcard\_\_frontSide);

    flipcard.appendChild(flipcard\_\_backSide);

    gamingTable.appendChild(flipcard);

  }

А потом добавляем eventListener на каждую карточку:

function makeFlipcardsAlive(){

  window.prevOpenedCard = null; // the variable is sort of global at this point

  const flipcards = document.querySelectorAll(".flipcard");

  flipcards.forEach(function(flipcard){

    flipcard.addEventListener("click", cardLogic)

  })

}

Callback функция для eventListener-а содержит логику игры. Тут, например, определяются условия победы:

function cardLogic (){

  flipACard(this);

  let prevPicture = "";

  try {

    prevPicture = prevOpenedCard.firstElementChild.

      firstElementChild.getAttribute("src");

  } catch {}

  let currPicture = this.firstElementChild.

    firstElementChild.getAttribute("src");

  if (prevOpenedCard === null){

    prevOpenedCard = this;

  } else if (prevOpenedCard === this){

    console.log("same card");

    prevOpenedCard = null;

  } else if (prevPicture === currPicture){

    // Later we should probably add an aura thing there

    console.log("same picture!");

    // Make it impossible to flip already discovered cards

    prevOpenedCard.removeEventListener("click", cardLogic);

    this.removeEventListener("click", cardLogic);

    window.numberOfCards = window.numberOfCards - 2;

    if (window.numberOfCards === 0) {

      console.log("Pat yourself on a back!");

      fadeTheGamingTable();

      // createADialogWindow();

      showTheDialogWindow();

    }

    prevOpenedCard = null;

  } else {

    // Here we copy the last two cards

    // to let the player to continue to flip cards

    let copy1 = this;

    let copy2 = prevOpenedCard;

    setTimeout(() => {

      flipACard(copy1);

      flipACard(copy2);

    }, 900);

    prevOpenedCard = null;

  }

}

Еще хотел рассказать про алгоритм, в результате работы которого, карточки в игре всегда будут лежать в случайном порядке:

// Returns an array of file names for us to put unique images on cards

function makeFileNamesList (numberOfCards) {

  let numberOfPictures = (numberOfCards - (numberOfCards % 2)) / 2

  // Fisher-Yates shuffle

  // We need to shuffle the cards, don't we?

  function shuffle(array) {

    for (let i = array.length - 1; i > 0; i--) {

      let j = Math.floor(Math.random() \* (i + 1));

      [array[i], array[j]] = [array[j], array[i]];

    }

    return array;

  }

Сначала мы создаем упорядоченный набор имен файлов с картинками:

  // Creating the array itself

  let fileNames = new Array(numberOfCards);

  for (let i = 0; i < numberOfCards; i++){

    if (i < numberOfPictures) {

      fileNames[i] = `img/${i.toString().padStart(2, "0")}.png`;

    } else {

      fileNames[i] =

        `img/${(i - numberOfPictures).toString().padStart(2, "0")}.png`;

    }

  }

Затем перемешиваем этот массив по алгоритму Фишера-Ейтса: Меняем местами все элементы со случайным элементом из массива, который идет раньше, начиная с последнего элемента:

  // Fisher-Yates shuffle

  // We need to shuffle the cards, don't we?

  function shuffle(array) {

    for (let i = array.length - 1; i > 0; i--) {

      let j = Math.floor(Math.random() \* (i + 1));

      [array[i], array[j]] = [array[j], array[i]];

    }

    return array;

  }

По функциональной части, все, что хотел рассказать — рассказал. Сейчас хочу показать, чему научился в CSS. Казалось, добавить шрифт на страницу очень просто, но у меня возникала ошибка, которая съела мой ланч. Шрифт не работал, потому, что я не указал свойство font-family в корне. Этот код нужно добавить перед всеми остальными определениями:

@font-face {

    font-family: "YuseiMagic";

    src: url("../fonts/YuseiMagic-Regular.ttf") format('truetype');

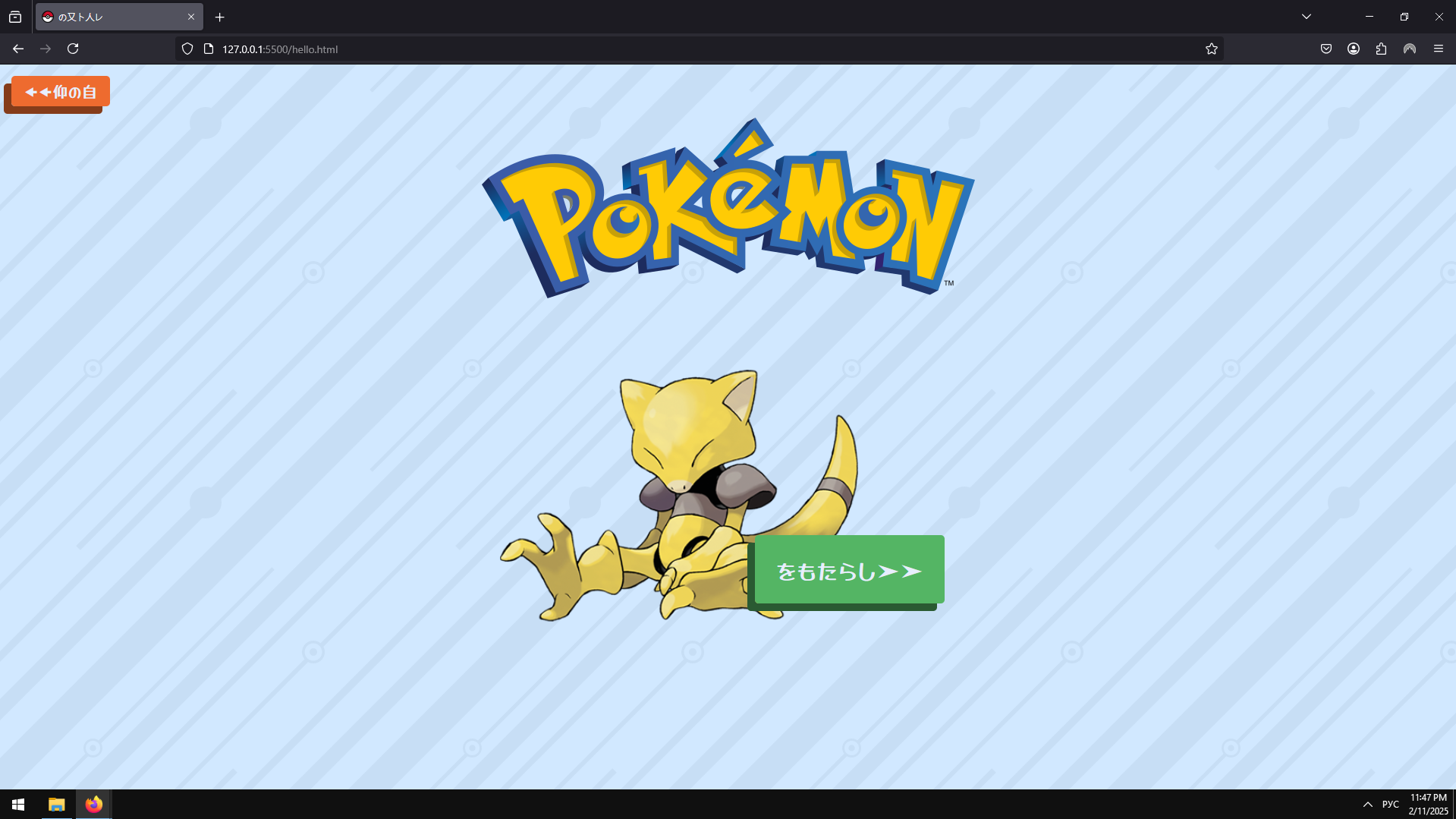
   }

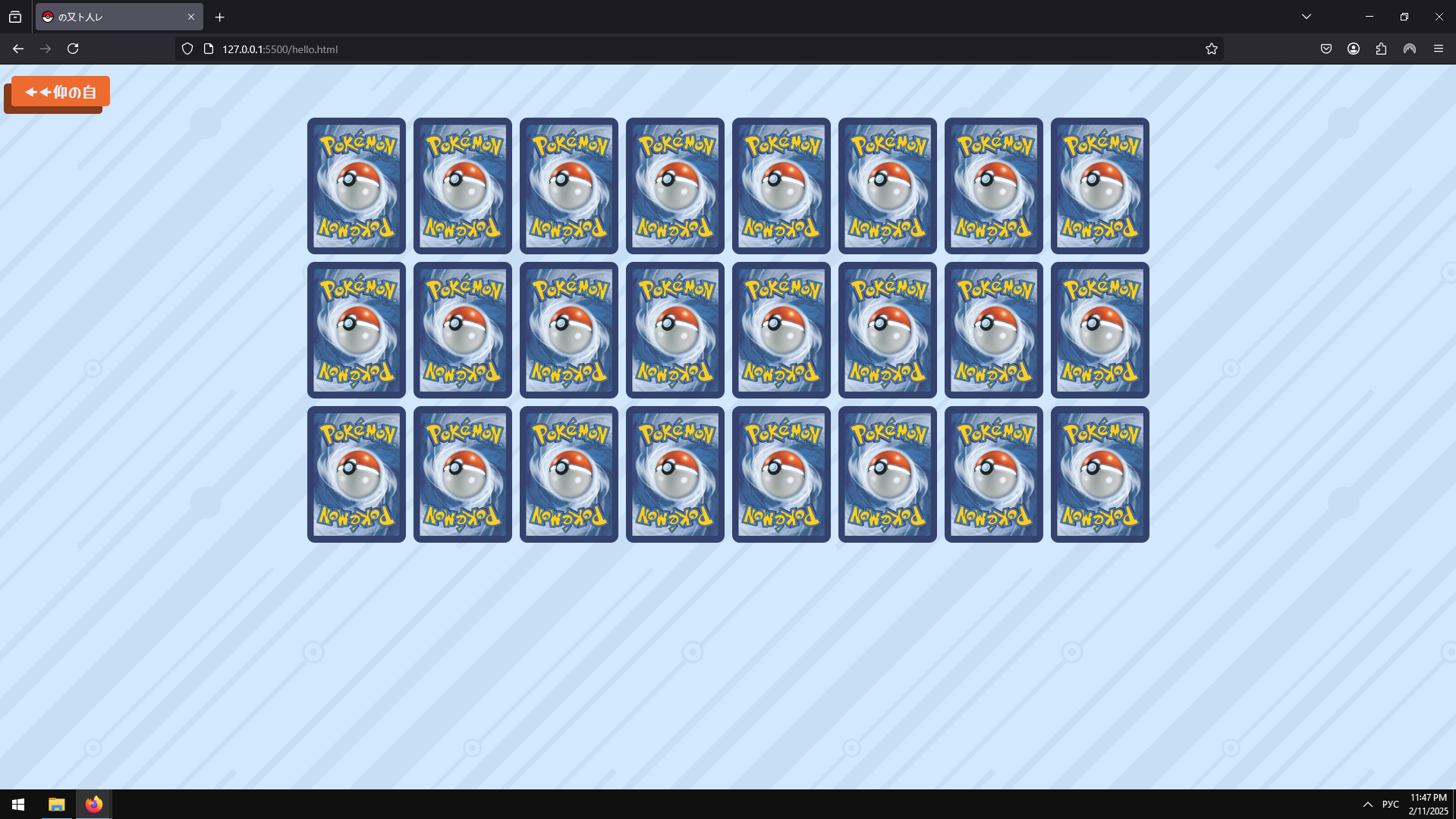
\*, ::after, ::before{

    font-family: "YuseiMagic";

}

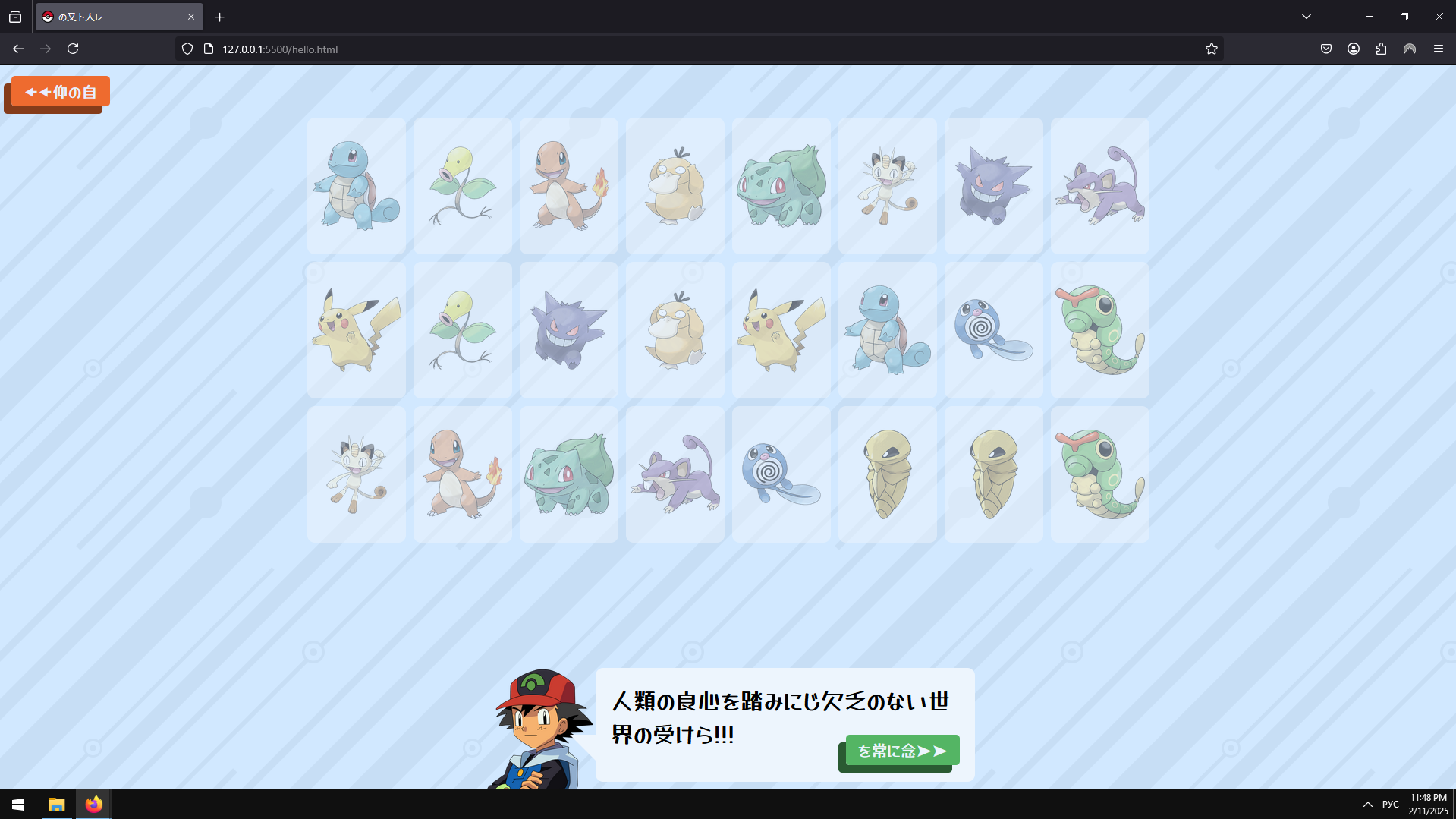
Скриншоты из игры:











**Вывод:**

Сделали красивый вебсайт, который можно показать при трудоустройстве.