BÁO CÁO: Mapty App: OOP, Geolocation, External Libraries, and More!

1. Project Overview:

* Là một website cho phép truy cập vị trí dựa theo kinh độ và vĩ độ trình duyệt của người dùng. Cho phép ghi lại các bài tập workout, có 2 loại là running và cycling.
* Sau khi click vào một điểm trên bản đồ, phần sidebar sẽ hiện ra một form bắt người dùng nhập vào Distance(km), Duration(min), Cadence(step/min)/Elev Gain(meters). Sau đó website trả về một list cái bài tập workout trên sidebar và hiển thị pop up(có trạng thái là running hoặc cycling) tại điểm đã click.
* Website lưu trữ dữ liệu trên local stogare nên khi tắt và mở lại, dữ liệu cũ vẫn còn ở đó.

1. How to plan a Project:

* User story(Description of the application’s functionsality from the user’s perspective. All user stories put toghether describe the entire application) → Feartures → Flowchart(What we will build) → Architecture(How we build it) [PLANNING STEP]
* Development Step(Implementation of our plan using code)

1. Using the Geolocation API:

* Code: navigator.geolocation.getCurrentPosition(function(), function())
* Sử dụng để lấy kinh độ và vĩ độ từ trình duyệt của người dùng

1. Displaying a Map Using Leaflet Library:

* Link: <https://leafletjs.com/>

Code: var map = L.map('map').setView([51.505(kđ), -0.09(vđ)], 13(tham số zoom));

L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {

attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'

}).addTo(map);

L.marker([51.5, -0.09]).addTo(map)

.bindPopup('A pretty CSS3 popup.<br> Easily customizable.')

.openPopup();

* Đây là một thư viện cho phép chúng ta hiển thị bản độ dựa theo kinh độ và vĩ độ từ trình duyệt của người dùng.
* Trong thư viện có phần docs, nơi chứa tất cả các phương thức(how to use).
* Nhúng đoạn mã ở phần download vào phần Head của HTML(Tài liệu CDN):

<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" integrity="sha256-p4NxAoJBhIIN+hmNHrzRCf9tD/miZyoHS5obTRR9BMY=" crossorigin="" />

<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js" integrity="sha256-20nQCchB9co0qIjJZRGuk2/Z9VM+kNiyxNV1lvTlZBo=" crossorigin=""></script>

* Chú ý trình tự tải, nhớ thêm defer vào thẻ script.

1. Display a Map Marker & Rendering Workout input Form:

let map, mapEvent;

//API Geolocation

console.log(navigator.geolocation);

if (navigator.geolocation)

navigator.geolocation.getCurrentPosition(

function (position) {

const { latitude } = position.coords;

const { longitude } = position.coords;

console.log(`https://www.google.com/maps/@${latitude},${longitude}`);

const coords = [latitude, longitude];

map = L.map('map').setView(coords, 13);

// console.log(map);

// https://tile.openstreetmap.org/{z}/{x}/{y}.png - oldlinkstyle

L.tileLayer('https://tile.openstreetmap.fr/hot/{z}/{x}/{y}.png', {

attribution:

'&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors',

}).addTo(map);

map.on('click', function (mapE) {

mapEvent = mapE;

form.classList.remove('hidden');

inputDistance.focus();

});

},

function () {

alert('Could not get your position');

}

);

form.addEventListener('submit', function (e) {

e.preventDefault();

//Clear input filed

inputDistance.value =

inputCadence.value =

inputDuration.value =

inputElevation.value =

'';

// Display a markup

console.log(mapEvent);

const { lat, lng } = mapEvent.latlng;

L.marker([lat, lng])

.addTo(map)

.bindPopup(

L.popup({

maxWidth: 250,

minWidth: 100,

autoClose: false,

closeOnClick: false,

className: 'running-popup',

})

)

.setPopupContent('Workout')

.openPopup();

});

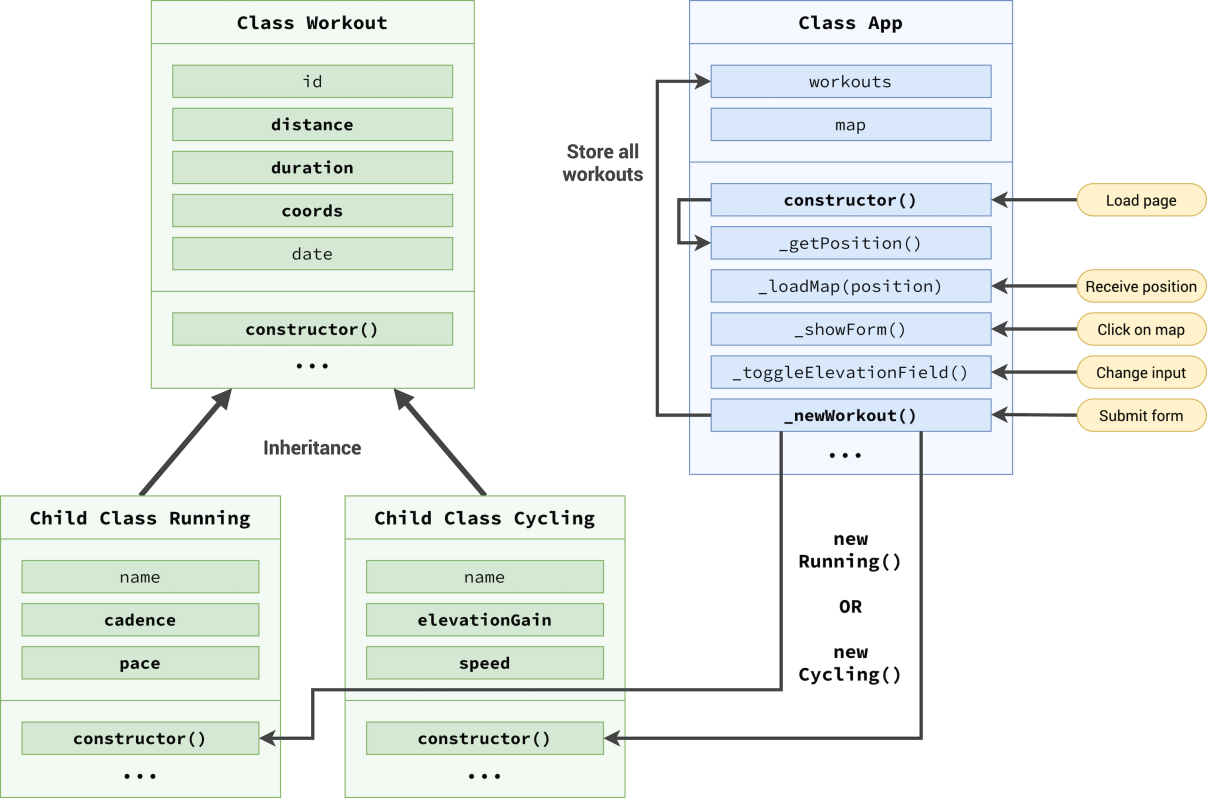
inputType.addEventListener('change', function () {

inputElevation.closest('.form\_\_row').classList.toggle('form\_\_row--hidden');

inputCadence.closest('.form\_\_row').classList.toggle('form\_\_row--hidden');

});

1. Refactoring for Project Architecture:



CODE:

class App {

  #map;

  #mapEvent;

  constructor() {

    this.\_getPosition();

    form.addEventListener('submit', this.\_newWorkout.bind(this));

    inputType.addEventListener('change', this.\_toggleElevationField.bind(this));

  }

  \_getPosition() {

    if (navigator.geolocation)

      navigator.geolocation.getCurrentPosition(

        this.\_loadMap.bind(this),

        function () {

          alert('Could not get your position');

        }

      );

  }

  \_loadMap(position) {

    const { latitude } = position.coords;

    const { longitude } = position.coords;

    console.log(`https://www.google.com/maps/@${latitude},${longitude}`);

    const coords = [latitude, longitude];

    this.#map = L.map('map').setView(coords, 13);

    //   console.log(map);

    // https://tile.openstreetmap.org/{z}/{x}/{y}.png - oldlinkstyle

    L.tileLayer('https://tile.openstreetmap.fr/hot/{z}/{x}/{y}.png', {

      attribution:

        '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors',

    }).addTo(this.#map);

    this.#map.on('click', this.\_showForm.bind(this));

  }

  \_showForm(mapE) {

    this.#mapEvent = mapE;

    form.classList.remove('hidden');

    inputDistance.focus();

  }

  \_toggleElevationField() {

    inputElevation.closest('.form\_\_row').classList.toggle('form\_\_row--hidden');

    inputCadence.closest('.form\_\_row').classList.toggle('form\_\_row--hidden');

  }

  \_newWorkout(e) {

    e.preventDefault();

    //Clear input filed

    inputDistance.value =

      inputCadence.value =

      inputDuration.value =

      inputElevation.value =

        '';

    // Display a markup

    const { lat, lng } = this.#mapEvent.latlng;

    L.marker([lat, lng])

      .addTo(this.#map)

      .bindPopup(

        L.popup({

          maxWidth: 250,

          minWidth: 100,

          autoClose: false,

          closeOnClick: false,

          className: 'running-popup',

        })

      )

      .setPopupContent('Workout')

      .openPopup();

  }

}

const app = new App();

1. Managing Workout Data: Creating Classes & Creating a New Workout

CODE:

class Workout {

  date = new Date();

  id = (Date.now() + '').slice(-10);

  constructor(coords, distance, duration) {

    this.coords = coords;

    this.distance = distance; // in km

    this.duration = duration; // in min

  }

}

class Running extends Workout {

  type = 'running';

  constructor(coords, distance, duration, cadence) {

    super(coords, distance, duration);

    this.cadence = cadence;

    this.calcPace();

    // this.type = 'running'

  }

  calcPace() {

    // min / km

    this.pace = this.duration / this.distance;

    return this.pace;

  }

}

class Cycling extends Workout {

  type = 'cycling';

  constructor(coords, distance, duration, elevationGain) {

    super(coords, distance, duration);

    this.elevationGain = elevationGain;

    this.calcSpeed();

    this.type = 'cycling';

  }

  calcSpeed() {

    // km / h

    this.speed = this.distance / (this.duration / 60);

  }

}

// Test Class

// const run1 = new Running([39, 12], 5.2, 24, 178);

// const cycling1 = new Cycling([39, 12], 5.2, 24, 578);

// console.log(run1, cycling1);

//////////////////////////

//APPLICATION ARCHITECTURE

// let map, mapEvent;

class App {

  #map;

  #mapEvent;

  #workout = [];

  constructor() {

    this.\_getPosition();

    form.addEventListener('submit', this.\_newWorkout.bind(this));

    inputType.addEventListener('change', this.\_toggleElevationField.bind(this));

  }

  \_getPosition() {

    if (navigator.geolocation)

      navigator.geolocation.getCurrentPosition(

        this.\_loadMap.bind(this),

        function () {

          alert('Could not get your position');

        }

      );

  }

  \_loadMap(position) {

    const { latitude } = position.coords;

    const { longitude } = position.coords;

    console.log(`https://www.google.com/maps/@${latitude},${longitude}`);

    const coords = [latitude, longitude];

    this.#map = L.map('map').setView(coords, 13);

    //   console.log(map);

    // https://tile.openstreetmap.org/{z}/{x}/{y}.png - oldlinkstyle

    L.tileLayer('https://tile.openstreetmap.fr/hot/{z}/{x}/{y}.png', {

      attribution:

        '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors',

    }).addTo(this.#map);

    this.#map.on('click', this.\_showForm.bind(this));

  }

  \_showForm(mapE) {

    this.#mapEvent = mapE;

    form.classList.remove('hidden');

    inputDistance.focus();

  }

  \_toggleElevationField() {

    inputElevation.closest('.form\_\_row').classList.toggle('form\_\_row--hidden');

    inputCadence.closest('.form\_\_row').classList.toggle('form\_\_row--hidden');

  }

  \_newWorkout(e) {

    const validInputs = (...inputs) =>

      inputs.every(inp => Number.isFinite(inp));

    const allPositive = (...inputs) => inputs.every(inp => inp > 0);

    e.preventDefault();

    // Get data from form

    const type = inputType.value;

    const distance = +inputDistance.value;

    const duration = +inputDuration.value;

    const { lat, lng } = this.#mapEvent.latlng;

    let workout;

    // If workout running, create running object

    if (type === 'running') {

      const cadence = +inputCadence.value;

      // Check if data is valid

      if (

        // !Number.isFinite(distance) ||

        // !Number.isFinite(duration) ||

        // !Number.isFinite(cadence)

        !validInputs(distance, duration, cadence) ||

        !allPositive(distance, duration, cadence)

      )

        return alert('Inpust have to be positive number!');

      workout = new Running([lat, lng], distance, duration, cadence);

    }

    // If workout cycling, create cycling object

    if (type === 'cycling') {

      const elevation = +inputElevation.value;

      if (

        !validInputs(distance, duration, elevation) ||

        !allPositive(distance, duration)

      )

        return alert('Inpust have to be positive number!');

      workout = new Cycling([lat, lng], distance, duration, elevation);

    }

    // Add new object to workout array

    this.#workout.push(workout);

    console.log(workout);

    // Render workout on map as marker

    this.renderWorkoutMarker(workout);

    // Render workout on list

    // Hide form + clear input fields

    inputDistance.value =

      inputCadence.value =

      inputDuration.value =

      inputElevation.value =

        '';

  }

  renderWorkoutMarker(workout) {

    console.log(workout.coords);

    L.marker(workout.coords)

      .addTo(this.#map)

      .bindPopup(

        L.popup({

          maxWidth: 250,

          minWidth: 100,

          autoClose: false,

          closeOnClick: false,

          className: `${workout.type}-popup`,

        })

      )

      .setPopupContent('workout')

      .openPopup();

  }

}

const app = new App();