Лекція: Якість коду, Об’єкти, Рядки, Масиви, Ітеративні об’єкти

План заняття

- 1. Якість коду: принципи, інструменти
- 2. Типи даних: короткий огляд
- 3. Рядки: методи та приклади
- 4. Об’єкти та деструктуризація
- 5. Масиви та методи масивів
- 6. Ітеративні об’єкти та протокол ітерації
- 7. Map і Set
- 8. WeakMap і WeakSet
- 9. Практичні патерни/антипатерни

Якість коду: що це?

- Читабельність і простота (KISS, DRY, YAGNI)
- Передбачуваність, тестованість, повторне використання
- Стандарт кодування: стилі, іменування, структура
- Автоматизація якості: лінтери, форматери, тести

Інструменти якості

- ESLint (правила якості) + Prettier (автоформат)
- Jest/Vitest для модульних тестів
- Husky + lint-staged для перевірок перед комітом
- DevTools: Performance/Memory/Recorder
- package.json: скрипти lint/format/test

```
// package.json (фрагмент)
{
  "scripts": {
    "lint": "eslint .",
    "format": "prettier -w .",
    "test": "vitest"
  }
}
```

Типи даних у JS (огляд)

- Примітиви: string, number, boolean, null, undefined, symbol, bigint
- Об’єкти: масиви, функції, звичайні об’єкти, Map/Set тощо
- typeof null === 'object' — історична особливість
- Перетворення типів (явні/неявні), truthy/falsy

Рядки (Strings)

- Створення: літерали, шаблонні рядки (template literals)
- Часті методи: length, includes, indexOf, slice, substring, replace, split, trim, padStart/End
- Юнікод і емодзі: Array.from('😳').length === 1

```
const name = 'Оксана';
console.log(`Привіт, ${name}!`.toUpperCase());
console.log('  hello '.trim().padStart(8, '-'));
```

Об’єкти (Objects)

- Літерал, доступ: obj.key / obj['key']
- Деструктуризація, shorthand, computed properties
- Перевірка: in, Object.hasOwn(), hasOwnProperty
- Копіювання: поверхневе ({...obj}), глибоке — окремими інструментами

```
const user = { id: 1, name: 'Ipa', role: 'student' };
const { name, role } = user; // деструктуризація
const copy = { ...user, role: 'admin' }; // поверхневе копіювання
```

Масиви (Arrays)

- Створення: [], Array.from, new Array(n)
- Базові методи: push/pop, shift/unshift, slice, splice, includes, indexOf
- Обхід: for, for...of, forEach

```
const arr = [3, 1, 4];
arr.push(1); // [3,1,4,1]
console.log(arr.slice(1)); // [1,4,1]
```

Методи масивів: «вісімка основи»

- map, filter, reduce, find, some, every, sort, forEach
- Ланцюжки трансформацій (pipeline)
- sort зі своїм компаратором

```
const users = [
  {name: 'Андрій', age: 19},
  {name: 'Оля', age: 22},
  {name: 'Іра', age: 17}
];
const adultsSorted = users
  .filter(u => u.age >= 18)
  .sort((a,b)=> a.age - b.age)
  .map(u => u.name);
console.log(adultsSorted); // [ 'Андрій', 'Оля' ]
```

reduce: агрегація та трюки

- Сума/середнє/групування
- Перетворення масиву в об’єкт або Map

```
const nums = [1,2,3,4];
const sum = nums.reduce((acc, n) => acc + n, 0);
const groupByLen = ['js','node','web','ui']
  .reduce((acc,w)=>{
    const k = w.length;
    (acc[k] ||= []).push(w);
    return acc;
  }, {});
```

Ітеративні об’єкти та протокол ітерації

- Iterable: має Symbol.iterator(), ітерується for...of
- Iterator: об’єкт з методом next()
- Перетворення: Array.from(iterable), оператор розпакування [...]

```
const iterable = {
  from: 1, to: 3,
  [Symbol.iterator]() {
    let i = this.from;
    return {
      next: () => ({ value: i, done: i++ > this.to })
    };
  }
};
console.log([...iterable]); // [1,2,3]
```

Об’єкт Map

- Пара ключ→значення, ключі будь-якого типу
- Методи: set, get, has, delete, clear, size
- Ітерація: map.keys(), map.values(), map.entries()

```
const m = new Map();
const objKey = { id: 1 };
m.set('role','student').set(objKey, {score:95});
console.log(m.get(objKey)); // {score:95}
```

Об’єкт Set

- Унікальна колекція значень
- Операції: перевірка належності, видалення, розмір
- Типові задачі: дедуплікація, множинні операції

```
const uniq = [...new Set([1,2,2,3,3,3])]; // [1,2,3]
// перетин
const A = new Set([1,2,3]);
const B = new Set([2,3,4]);
const inter = new Set([...A].filter(x => B.has(x))); // {2,3}
```

WeakMap та WeakSet

- WeakMap: ключі — тільки об’єкти, «слабкі» посилання (GC)
- WeakSet: множина об’єктів із «слабкими» посиланнями
- Не ітеруються, немає size
- Використання: кешування, приватні дані, мітки оброблених об’єктів

```
const cache = new WeakMap();
function heavy(obj){
  if(cache.has(obj)) return cache.get(obj);
  const res = obj; // уявна важка операція
  cache.set(obj, res);
  return res;
}
```

Об’єкти vs Map: коли що?

- Об’єкт: фіксована «форма», конфіг, JSON, прототипні методи
- Map: динамічні ключі будь-яких типів, часті вставки/видалення
- Порада: дані — Map/Set, модель сутності — Object/клас

Патерни та антипатерни масивів

- ❌ Мутація без потреби (splice, sort без копії) → ☑ копії: [...arr].sort()
- ❌ forEach для побудови нового масиву → ☑ map
- ❌ Вкладені цикли O(n^2) → ☑ Map/Set для пришвидшення

```
// погано: сортування мутує вихідний масив
arr.sort((a,b)=>a-b);
// добре: не змінюємо оригінал
const sorted = [...arr].sort((a,b)=>a-b);
```

Приклад: частотний словник слів (Map)

- Map зручний для підрахунків та аналітики

```
const text = 'js js map set map weakmap';
const freq = new Map();
for (const w of text.split(/\s+/)) {
  freq.set(w, (freq.get(w) || 0) + 1);
}
console.log([...freq.entries()]);
```

Приклад: унікальні дні активності (Set)

- Set — швидка дедуплікація

```
const iso = ['2025-09-21','2025-09-21','2025-09-22'];
const uniqueDays = new Set(iso);
console.log(uniqueDays.size); // 2
```

Приклад: кеш форматування дат (WeakMap)

- WeakMap допомагає уникнути утримання зайвих посилань (витоків пам’яті)

```
const fmt = new Intl.DateTimeFormat('uk-UA', { dateStyle:'medium' });
const cache = new WeakMap();
function fmtDate(d){
  if(!cache.has(d)) cache.set(d, fmt.format(d));
  return cache.get(d);
}
const d = new Date();
console.log(fmtDate(d));
```

Перевірки якості коду на практиці

- ESLint + Prettier у проєкті
- Правила: no-unused-vars, eqeqeq, no-implicit-coercion
- Іменування: camelCase, змістовні назви, маленькі функції

```
// .eslintrc.json (фрагмент)
{
  "extends": ["eslint:recommended"],
  "env": {"es2022": true, "browser": true},
  "rules": {"eqeqeq": "error", "no-unused-vars": "warn"}
}
```

Підсумки

- Принципи якості коду економлять час у майбутньому
- Масиви + їхні методи = основний інструмент щодня
- Ітеративний протокол → гнучкі колекції
- Map/Set/WeakMap/WeakSet — обираємо свідомо

Міні-практика

- 1) З масиву користувачів згенерувати Map id → user
- 2) Знайти перетин улюблених технологій двох користувачів через Set
- 3) Реалізувати groupBy(arr, key) на reduce

Посилання та ресурси

- MDN Web Docs: JavaScript, Array methods, Map/Set
- ESLint, Prettier, Vitest/Jest — офіційні сайти
- Chrome DevTools документація

Сквозний приклад: StudyTracker

- Завдання: трекер навчальних сесій (дата, тема, теги, хвилини)
- Мета: проаналізувати активність, улюблені теми, серії (streak)
- Використаємо: якість коду, об’єкти, рядки, масиви, ітерацію, Map/Set, WeakMap

Модель даних (об’єкти)

- Сесія: { date: Date, topic: string, tags: string[], minutes: number }
- Валідація та нормалізація даних

```
// session.js
export function makeSession({dateISO, topic, tags, minutes}){
  const d = new Date(dateISO);
  if (Number.isNaN(d)) throw new Error("Bad date");
  const t = String(topic).trim();
  const tagArr = Array.isArray(tags) ? tags :
String(tags||"").split(",").map(s=>s.trim()).filter(Boolean);
  const m = Number(minutes);
  if (!t || m<=0) throw new Error("Bad payload");
  return { date: d, topic: t, tags: tagArr, minutes: m };
}
```

Парсинг рядків у сесії (Strings)

- Вхід: '2025-09-21; JS масиви; js,arrays; 45'
- Розбиття, обрізання пробілів, перетворення типів

```
export function parseLine(line){
  const [dateISO, topic, tagsStr, minutesStr] = line.split(";").map(s=>s.trim());
  return makeSession({ dateISO, topic, tags: tagsStr.split(","), minutes:
Number(minutesStr) });
}
```

Методи масивів: аналітика

- Сума хвилин, топ-3 теми, серія активності
- map/filter/reduce + sort

```
export function totalMinutes(sessions){
  return sessions.reduce((acc,s)=>acc+s.minutes,0);
}
export function topTopics(sessions, k=3){
  const freq = new Map();
  sessions.forEach(s=> freq.set(s.topic, (freq.get(s.topic)||0)+s.minutes));
  return [...freq.entries()].sort((a,b)=>b[1]-a[1]).slice(0,k);
}
export function longestStreak(sessions){
  const days = [...new Set(sessions.map(s=> s.date.toISOString().slice(0,10)))].sort();
  let max=0, cur=1;
  for(let i=1;i<days.length;i++){
    const a=new Date(days[i-1]), b=new Date(days[i]);
    cur = ((b-a)===86400000) ? cur+1 : 1;
    max = Math.max(max, cur);
  }
  return max;
}
```

Ітеративні об’єкти: по тижнях

- Генератор групує сесії по ISO-тижнях
- Зручно для звітів і візуалізації

```
export function* groupByWeek(sessions){
  const byDay = new Map();
  sessions.forEach(s=>{
    const key = s.date.toISOString().slice(0,10);
    (byDay.get(key) || byDay.set(key,[]).get(key)).push(s);
  });
  const days = [...byDay.keys()].sort();
  const weekOf = d => Math.ceil(( (new Date(d) - new Date(`${d.slice(0,4)}-01-
01`))/86400000 + 1 )/7);
  let curWeek = null, bucket = [];
  for(const day of days){
    const w = weekOf(day);
    if(curWeek===null) curWeek=w;
    if(w!==curWeek){ yield { week: curWeek, sessions: bucket }; curWeek=w; bucket=[]; }
    bucket.push(...byDay.get(day));
  }
  if(bucket.length) yield { week: curWeek, sessions: bucket };
}
```

Map та Set у ділі

- Map: хвилини на тег/тему/тиждень
- Set: унікальні дні активності, унікальні теги

```
export function minutesByTag(sessions){
  const m = new Map();
  for(const s of sessions){
    for(const tag of s.tags){
      m.set(tag, (m.get(tag)||0) + s.minutes);
    }
  }
  return m; // [['js',120], ['arrays',90], ...]
}
```

WeakMap: кеш важких обчислень

- Кешування «складності сесії» без ризику витоків
- Ключ — об’єкт сесії

```
const scoreCache = new WeakMap();
export function sessionScore(session){
  if(scoreCache.has(session)) return scoreCache.get(session);
  // Уявне важке обчислення
  const score = session.minutes + (session.tags.includes('algorithms') ? 15 : 0);
  scoreCache.set(session, score);
  return score;
}
```

Якість коду в кейсі

- Модулі: session.js, analytics.js, iterables.js
- Тести на критичні функції (Vitest)
- ESLint/Prettier: формат, no-unused-vars, eqeqeq

```
// analytics.test.js (Vitest)
import { totalMinutes, longestStreak } from './analytics.js';
it('рахує суму хвилин', ()=>{
  expect(totalMinutes([{minutes: 10},{minutes: 5}])).toBe(15);
});
```