ChatGPT

Optimalna ESLint 9 konfiguracija za browser (ESM format)

Kako bismo omogućili da JavaScript fajl koji manipuliše DOM-om (npr. `accordion/script.js`) prođe ESLint proveru bez grešaka, potrebno je pravilno podesiti ESLint konfiguraciju i okruženje. U nastavku su preporučena podešavanja, uključujući ažurirani ESLint config u ESM formatu, izmene u **package.json** i primer koda za **accordion/script.js**. Ova rešenja su u skladu sa modernim praksama za početne web projekte.

ESLint podešavanje (eslint.config.mjs)

Optimalno rešenje koristi novi **flat config** format uveden u ESLint v9 (ESM modul umesto starog `.eslintrc`). U ovoj konfiguraciji uključićemo ESLint-ova preporučena pravila i definisati **browser globalne promenljive** kako bi linter prepoznao objekte poput `window` i `document`. Ključni koraci u konfiguraciji su:

1. **ESLint preporučena pravila** – Proširujemo osnovna ESLint pravila pomoću oficijalnog paketa `@eslint/js` (koji sadrži `eslint:recommended` set pravila). Ovo nam daje podrazumevane provere kvaliteta koda.
2. **Globalne promenljive za browser** – U flat config formatu ne postoji više jednostavno `"env": {"browser": true}` opcija; umesto toga, dodajemo **predefinisane globalne promenljive** za browser okruženje. Najlakši način je uvoz paketa `globals` i uključivanje `globals.browser` objekta, čime se automatski dodaju svi standardni globali za web pregledač [1] [2] (uključujući `window`, `document`, `console` itd.).
3. **Jezičke opcije** – Postavljamo ECMAScript verziju na najnoviju (`ecmaVersion: "latest"`) i označavamo da je kod pisan u modularnom formatu (`sourceType: "module"`). Ovo osigurava da ESLint ispravno razume moderne JavaScript sintakse i module.
4. **Prilagođene ESLint rule** – Zadržavamo ili podešavamo pravila specifična za projekat. U nastavku primer zadržava prethodna podešavanja: upozorava umesto greške na neiskorišćene promenljive (`no-unused-vars: "warn"`) i dozvoljava korišćenje `console` komandi (`no-console: "off"`), što je praktično tokom razvoja.

Sve navedeno objedinjeno je u novom fajlu `eslint.config.mjs` (ESM modul). *Napomena:* Pošto koristimo ESM format za ESLint konfiguraciju, pobrinuli smo se da je projekat podešen da to podrži. U ovom slučaju koristimo fajl sa ekstenzijom `.mjs` (što Node/ESLint automatski tretira kao ESM modul), a alternativno smo u **package.json** ažurirali `"type": "module"` za punu ESM podršku [3]. Time možemo koristiti `import` sintaksu u konfiguracionom fajlu.

```
// eslint.config.mjs
import { defineConfig } from "eslint/config";
import js from "@eslint/js";
import globals from "globals";

export default defineConfig([
  // 1. ESLint preporučena pravila (core rules)
```

```
  js.configs.recommended,

  // 2. Naša prilagođena podešavanja za sve .js fajlove u projektu
  {
    files: ["**/*.js"],
    languageOptions: {
      ecmaVersion: "latest",         // podržava najnoviji JS (ES202x)
      sourceType: "module",          // kod je u ES Module modu
      globals: {
        ...globals.browser           // dodaje globalne promenljive za
browser [2]
      }
    },
    rules: {
      // 3. Pravila prilagođena projektu:
      "no-unused-vars": "warn",      // neiskorišćene promenljive kao
upozorenje (a ne grešku)
      "no-console": "off"            // dozvoljeno koristiti console.log
(korisno tokom razvoja)
    }
  }
]);
```

Ovaj **eslint.config.mjs** uključiće ESLint-ova podrazumevana pravila i dodatno registrovati browser globalne promenljive. Na taj način, konstrukti poput `document.getElementById` ili `window.addEventListener` više neće izazivati `no-undef` greške, jer ESLint sada "zna" da te promenljive postoje u browser okruženju [1]. Konfiguracija je spremna za ESLint 9+ i poštuje nova pravila formata (koristi `defineConfig` sa nizom konfiguracionih objekata, umesto zastarelog `.eslintrc` formata).

Izmene u package.json

Da bi ESLint prepoznao novu ESM konfiguraciju i browser okruženje, potrebno je minimalno prilagoditi i **package.json** projekta. Konkretno:

- **ESM modul podrška:** Promenili smo `"type": "commonjs"` u `"type": "module"` kako bi Node i alati (poput ESLint-a) očekivali ESM module podrazumevano. (U suprotnom, mogli bismo ostaviti `"type": "commonjs"` ali onda *obavezno* koristimo `.mjs` ekstenziju za ESLint config što već činimo. Obe opcije rezultuju time da ESLint uspešno učita ESM config.)
- **Dev zavisnost `globals`:** Dodali smo paket **globals** u devDependencies, jer ga koristimo u ESLint config-u za definisanje globalnih promenljivih. `globals` je popularan paket održavan od strane zajednice (verzija **16.x** je najnovija) koji jednostavno izlistava standardne globalne promenljive za različita okruženja [4]. Time dobijamo praktično isto ponašanje kao ranije `"env: browser"` flag, ali u novom formatu.

U nastavku je ažurirani deo **package.json** sa ovim izmenama:

```
{
  "name": "html-css-js-portfolio",
```

```
  "version": "1.0.0",
  "description": "Collection of small frontend projects built with HTML, CSS
and JavaScript.",
  "main": "index.js",
  "scripts": {
    "lint:js": "eslint . --ext .js",
    "lint:css": "stylelint '**/*.css' --allow-empty-input",
    "lint:html": "htmlhint '**/*.html'",
    "lint": "npm run lint:js && npm run lint:css && npm run lint:html"
  },
  "type": "module",                /* Promenjeno sa "commonjs" na "module" */
  "devDependencies": {
    "@eslint/js": "^9.39.1",
    "eslint": "^9.39.1",
    "globals": "^16.5.0",          /* Dodat npm paket za browser globale */
    "htmlhint": "^1.7.1",
    "stylelint": "^16.25.0",
    "stylelint-config-standard": "^39.0.1"
  }
}
```

Nakon ove izmene, obavezno instalirajte nove zavisnosti (pokrenite `npm install`) da bi ESLint prepoznao dodatni paket. Sada je projekat podešen tako da ESLint 9 koristi **ESM konfiguraciju** i zna za browser globalne promenljive.

Primer JavaScript koda za accordion komponentu

Konačno, u nastavku je **primer validnog** fajla `accordion/script.js` koji implementira funkcionalnost FAQ akordeona. Ovaj skript dodaje JS ponašanje koje menja `aria-expanded` atribut i dodaje/uklanja klasu `is-open` na odgovarajućim elementima, u skladu sa pristupačnim dizajnom interfejsa. Kod je napisan tako da prolazi ESLint linting prema gore definisanim pravilima (koriste se moderna JS sintaksa i globalne promenljive `document` / `window` koje su sada priznate zahvaljujući konfiguraciji):

```
// accordion/script.js
const triggers = document.querySelectorAll(".accordion__trigger");

triggers.forEach((trigger) => {
  trigger.addEventListener("click", () => {
    const isExpanded = trigger.getAttribute("aria-expanded") === "true";
    const panelId = trigger.getAttribute("aria-controls");
    const panel = document.getElementById(panelId);

    // Prebacujemo aria-expanded sa true na false (ili obratno) za kliknuto
dugme
    trigger.setAttribute("aria-expanded", String(!isExpanded));

    // Togglujemo klasu .is-open na povezanom panelu da prikažemo ili
sakrijemo sadržaj
```

```
    panel.classList.toggle("is-open", !isExpanded);
  });
});
```

Objašnjenje koda: Selektujemo sva dugmad koja služe kao okidači akordeona (`.accordion__trigger`) i dodajemo svakom handler na događaj **click**. Kada se dugme klikne, proveravamo njegov trenutni `aria-expanded` status: - Ako je bio `"true"` (panel otvoren), treba da ga zatvorimo – postavimo `aria-expanded` na `"false"` i uklonimo klasu `is-open` sa panela. - Ako je bio `"false"` (panel zatvoren), otvaramo ga – postavimo `aria-expanded` na `"true"` i dodamo klasu `is-open` na odgovarajući panel element.

Vezu između dugmeta i njegovog panela ostvarujemo preko **ARIA atributa**: dugme ima atribut `aria-controls` sa vrednošću koja odgovara `id`-u panela. Tako dobijamo referencu na pravi `.accordion__panel` element pozivom `document.getElementById(panelId)`. Zatim koristimo `Element.classList` **API** za dodavanje/uklanjanje klase: metod `toggle` sa drugim argumentom (`panel.classList.toggle('is-open', !isExpanded)`) eksplicitno postavlja klasu samo ako je novi status "otvoren" (true), a skida je ako je "zatvoren" (false).

Ovakav kod poštuje pristupačnost: kada panel postane vidljiv, `aria-expanded` se postavi na `"true"`, a kada se sakrije na `"false"`, što pomaže korisnicima čitača ekrana da znaju stanje svakog pitanja. Istovremeno, vizuelno se kontrola ostvaruje CSS klasom `is-open` (npr. za prikaz/skrivanje sadržaja panela preko CSS-a).

Zaključak

Sa gore navedenim podešavanjima, vaš ESLint konfiguracioni fajl i prateći kod u potpunosti su usklađeni sa ESLint 9+ zahtevima. **ESLint** sada prepoznaje **browser globalne promenljive** (zahvaljujući uključenom `globals.browser` setu) i može da analizira ESM module. To znači da `accordion/script.js` (kao i ostali JS fajlovi u projektu) neće izbacivati lint greške za DOM manipulaciju ili slične konstrukte specifične za browser.

Sve date izmene predstavljaju produkcijski validan setup za HTML/CSS/JS projekat u VSCode + WSL okruženju. Korišćenjem modernog ESLint flat config formata i pravilnog ARIA upravljanja u kodu, obezbeđujete kvalitetan i održiv kodbase u skladu sa savremenim praksama za web razvoj. Sada možete kopirati prikazane fajlove u svoj repozitorijum i imati ispravnu ESLint postavku bez dodatnog podešavanja.

---

[1] [2] [4] Configure Language Options - ESLint - Pluggable JavaScript Linter
https://eslint.org/docs/latest/use/configure/language-options

[3] Configuration Files - ESLint - Pluggable JavaScript Linter
https://eslint.org/docs/latest/use/configure/configuration-files