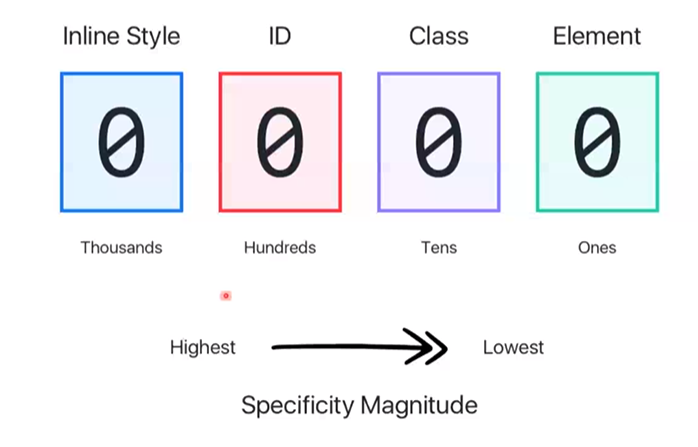
0116



1000 🡪100 🡪10🡪1

Pagal indekso svorį

Inline yra kai pvz <p style=“color:red“>

Cheatas yra jei mazenes svarbos eilutej, pvz po spalvos dedi !imprtant. tada jis issikelia. Bet cia extra atveju daryt

<http://web.simmons.edu/~grovesd/comm244/notes/week4/css-reset>

atsidare reseta arba

iterpiam ji i savo faila. Sukuriam nauja #reset.css ir nukopijuojam is psl viska i ta faila. Ir sulinkinam html faile.

Tdl reset linka reikia det pries stiliu linka (link css). Tada pirma nuskaito narsykle reseta, o tada mano stiliu

<http://web.simmons.edu/~grovesd/comm244/notes/week4/reset.css>

teorija

CSS inheritance arba lietuviškai paveldėjimu.

**ne visi** CSS stiliai yra paveldimi iš tėvų vaikams. CSS nepaveldimos pločio savybės, padding, margin ir border

Paveldėjimas buvo apie stilius, kurie **nėra uždėti tiesiogiai HTML tagui**, bet paveldimi iš tėvinių elementų. Tuo tarpu cascading, paaiškina kaip naršyklė nusprendžia kokius CSS stilius taikyti, kai jie yra uždėti **tiesiogiai** konkrečiam tagui.

Pvz. kaip ir kodėl naršyklė nusprendžia, kad <h1> teksto spalva yra seagreen, o ne blue.

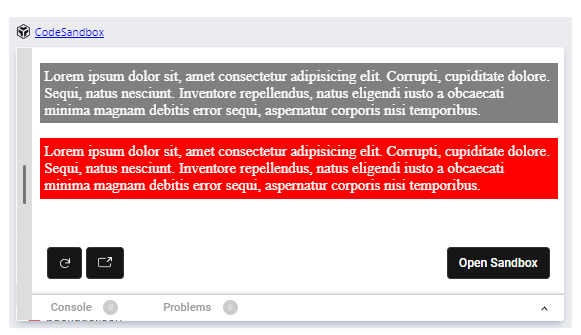
[CodeSandbox](https://codesandbox.io/embed/quirky-hill-l5e9h8?fontsize=14&hidenavigation=1&theme=light)

Trumpai naršyklė atsižvelgia į tris veiksnius, išvardintus didėjančia svarbos tvarka (vėlesni perrašo ankstesnius):

1. Kodo eilės tvarka (angl. source order)
2. CSS taisyklės specifiškumas (angl. specificity)
3. Svarba (angl. importance)
4. **CSS kodo vietos įtaka**
5. Be šių 3 veiksnių svarbi ir CSS kodo vieta (angl. origin). Jeigu keliose vietose yra parašytos tokios pačios taisyklės, tada naršyklė sprendžia kurį stilių taikyti atsižvelgus į tai kur šitas CSS kodas parašytas. Žemiau lentelėje yra pavaizduota eilės tvarka, kuria naršyklė pasirinks taikyti stilius.

|  |  |  |
| --- | --- | --- |
| Eilės tvarka (didesnis skaičius “ima viršų”) | CSS kodo vieta | Svarba |
| 1 | user-agent (browser) | normal |
| 2 | user | normal |
| 3 | author (developer) | normal |
| 4 | CSS @keyframe animations |  |
| 5 | author (developer) | !important |
| 6 | user | !important |
| 7 | user-agent (browser) | !important |
| 8 | CSS transitions |  |

Pažiūrėkime, kaip tai veikia praktiškai. Pavyzdyje žemiau turime HTML su dviem <p> elementais. Vienas iš jų turi nustatytą klasę, o kitas - klasę bei id. Nors bendrai visiems <p> elementams yra taikoma mėlyna fono spalva, <p class="class"> atvaizduojamas su pilku fonu, o <p id="id" class="class"> su raudonu. Būtent todėl, nes #id selektoriaus specifiškumo skaičius yra didesnis už .class, o .class didesnis už p.



## **Svarba (angl. importance)**

CSS turi specialus žodelį !important, kuris leidžia nepaisyti visų specifiškumo skaičiavimo taisykles ir kodo tvarką. Panaudojus šį specialų žodelį, naršyklė ignoruoja cascading taisykles ir tiesiog pritaiko Jūsų konkrečią savybę neatsižvelgiant į jas.

!important reikėtų naudoti labai atsakingai ir su saiku. Daugiau apie argumentus kodėl to nerekomentuojama daryti galite paskaityti [Specificity - CSS: Cascading Style Sheets | MDN](https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity#the_!important_exception) puslapyje.

#### Naršyklės pagal nutylėjimą taikomi stiliai

Paskautinį kartą atnaujinta: Jun 28, 2022 | Gedas Gardauskas

Naršyklės irgi pagal nutylėjimą taiko tam tikrus stilius, jie dar kitaip vadinami User-agent stylesheets. Deja, tarp naršyklių jie nėra visiškai identiški. [Browser Default Styles](https://browserdefaultstyles.com/) puslapyje galite pasitikrinti, kokius stilius taiko skirtingos naršyklės tam tikram HTML elementui.

Pavyzdžiui, net pats paprasčiausias h1 elementas turi skirtingus stilius tarp naršyklių. Kitų elementų skirtumai gali būti ir dar didesni.



Programuotojai stengiasi šiuos skirtumus kuo labiau suvienodinti, kad atvaizdavimas tarp naršyklių būtų labiau nuspėjamas. Nors sunku viską numatyti ir neišvengiami įvairūs “bug’ai”, ypač Apple mėgsta pritaikyti “Think different” 😃.

Šiai problemai spręsti dažniausiai yra naudojami du pagrindiniai būdai: reset arba normalize.

|  |  |
| --- | --- |
| **Reset** | **Normalize** |
| Resets all the has styles that come with the browser’s user agent. | Provides cross-browser consistency in the default styling of HTML elements which are provided by the browser’s User Agent. |
| Resetting have a big chain of selectors, and they make a lot of unnecessary overrides in the styling. | Not many big chains of CSS Selectors to be seen while normalizing as it utilizes User Agent’s styles. |
| It is hard to debug as it is nearly impossible to identify bugs. | Debugging is easy while normalizing. |
| Resetting is Non-Modular (no section breaking in styles). | Normalizing is Modular (styling is divided into sections for ease). |

Daugiau truputį detaliau apie juos ir kaip juos pritaikyti iš praktinės pusės sužinosite vaizdo įraše.

<https://youtu.be/eCrQviuQyEo>

[**SEKANTI TEMA**](https://codeacademy.online/dashboard/modules/html2/6IrlgQaW7MwoZBPDEwkc/7274576)