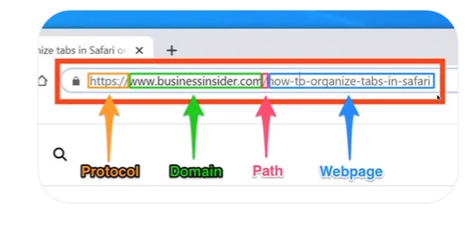
URL – uniform resource locator – link‘ai – pasauliniai adresai, kuriuose pasauliniame web‘e yra konkretūs dokumentai ir ištekliai

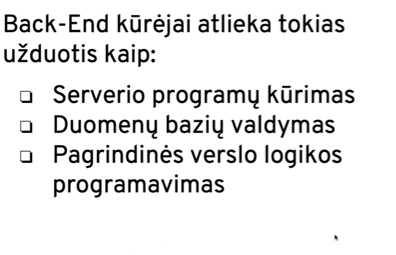
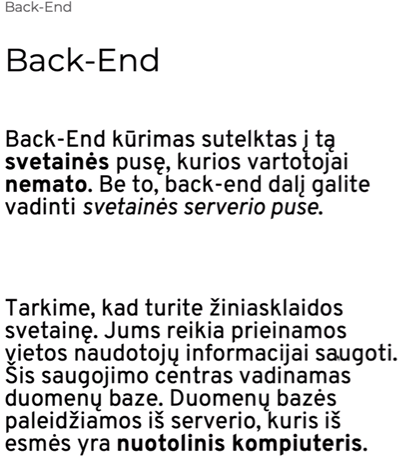
IP – interneto protokolo adresas – unikalus ID numeris kiekvienam įrenginiui



http – hypertext transfer protocol. Pagrindinis metodas informacijai pasauliniame tinkle pasiekti. +s reiškia, kad saugus pagal standartą. Jei nėra to, geriau nevest pvz savo banko kodų ten.

Front-end. Kuria vaizdą vartotojui – programavimo stilius, kuriame pagrindinis dėmesys skiriamas svetainės elementų ir f-jų, kurias vėliau matys vartotojas.

Back-end



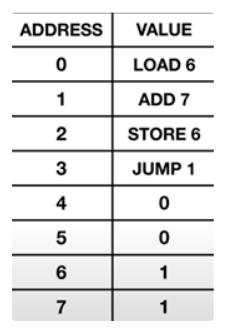
2psk

***Kompiuteriai ir kompiuterių mokslas***

Kalbant apie kompiuterius, mobiliuosius telefonus bei kitus elektroninius įrenginius dažnai girdime išsireiškimą „išmanusis įrenginys“, tačiau drįsčiau paprieštarauti. Visas „išmaniąsias“ funkcijas įrenginyje atlieka procesorius, kuris sugeba atlikti tik primityvius veiksmus: nuskaityti instrukcijas esančias atmintyje, jas iššifruoti ir įvykdyti. Iš šitų operacijų susidaro procesoriaus ciklas (angl. fetch-decode-execute). Kiekviena ciklo dalis yra valdoma procesoriaus laikmačio (angl. Timer). Baigus vykdyti vieną ciklą yra padidinama skaitiklio (angl. Counter) reikšmė ir ciklas prasideda iš naujo. Skaitiklis padeda programai „judėti“ pirmyn bei pasako kokią instrukciją vykdyti toliau. Taip pat, procesorius turi savo steką (angl. Stack) į kurį gali laikinai įrašyti reikšmes su kuriomis dirba.

## Kaip veikia kompiuterio procesorius?

Įsivaizduokite jog atmintyje turite tokį instrukcijų sąrašą, koks yra pavaizduotas paveikslėlyje. Siekiant parodyti veikimo principus pavaizduotos reikšmės yra parašytos „žmonėm suprantamesne“ kalba, realiame pasaulyje tokia programa būtų parašyta dvejetaine sistema - vienetukais ir nuliukais.

Panagrinėkite viršuje esančią iliustraciją.

Taigi, **pradžioje procesoriaus skaitiklio reikšmė yra 0** (kompiuteriai pradeda skaičiuoti nuo 0, o ne nuo 1). Todėl nuskaitoma reikšmė esanti 0-oje atminties pozicijoje – kur ADDRESS yra lygus 0. Nuskaityta reikšmė „LOAD 6“ yra instrukcija, kurią procesorius **iššifruoja** **antrame ciklo žingsnyje**. **Trečiame žingsnyje programa vykdo instrukciją** – pasiima reikšmę „1“ esančią atminties 6-tame adrese ir patalpiną ją į savo steką. Taip užbaigiamas pirmas ciklas ir skaitiklio reikšmė pasikeičia iš „0“ į „1”.

Antro ciklo pradžioje skaitiklio reikšmė yra 1, todėl ir instrukcija bus paimama iš atminties adreso „1” – “ADD 7”. Ją iššifruos ir trečiame ciklo žingsnyje prie steke esančios reikšmės (1) pridės reikšmę esančią 7-intame adrese esančia reikšmę (1). Taip gaus reikšmę „2“, padidins skaitiklį +1 ir pradės naują ciklą.

Toliau programa pereis prie komandos „STORE 6”. Ciklo metu išsaugos steko viršuje esančią reikšmę į atminties bloką, kurio adresas yra „6”. Taip programa prieis prie komandos esančios adresu „3”. Šio ciklo metu instrukcija pasako jog procesoriui reikia savo skaitiklį pakeisti iš dabartinės reikšmės į reikšmę „1“. Ir viskas prasideda iš naujo. Vienintelis dalykas, kuris skirsis, tai, kad dabar reikšmė esanti steke bus 1, o nebe 0.

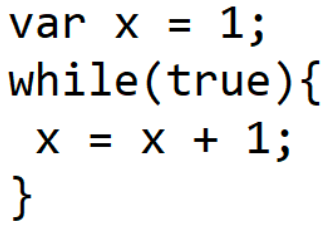
Ši programa yra ganėtinai paprasta: ji tiesiog skaičiuoja iki „begalybės“. Žodis „begalybės“ yra kabutėse tik dėl to, kad programa galiausiai sulūš. Tai įvyks, kai skaičius pasieks atminties bloko ribą ir jo padidinti nebeišeis. :(

Gali pasirodyti jog toks procesas yra ganėtinai lėtas, tačiau kompiuterio laikmatis skaičiuoja laiką išties greitai. Galimai girdėjote, kad procesoriaus greitis dažnai yra apibrėžiamas gigahercais (Ghz). Tai reiškia, kad kompiuteris geba atlikti milijardus instrukcijų per sekundę. Taip pat, modernūs procesoriai sugeba vykdyti kelias instrukcijas lygiagrečiai, kas dar labiau padidina jų spartumą.

Programavimas

Aišku, rašyti instrukcijas tiesiogiai procesoriui būtų sudėtinga, kodo rašymas užimtų daug laiko ir reikėtų pasukti galvą norint sukurti programą galinčią atlikti sudėtingus veiksmus.

Būtent todėl yra naudojamos aukštesnio lygio programavimo kalbos. Jų parašytą kodą daug lengviau suprasti žmogus, o kodą sukompiliavus gaunamos tos pačios instrukcijos. Pavyzdžiui ta pati programa parašyta aukštesnio lygio kalba atrodytų daug maž taip:



Daugiau apie kompiuterių mokslą

Šių kursų metu apie kompiuterių mokslą kalbėsime tik paviršutiniškai. Todėl jeigu norite sužinoti, galite internete ar YouTube paieškoti įvairių papildomų kursų. Vienas iš tokių video kursų yra *Crash Course Computer Science.*

https://www.youtube.com/s/desktop/ce262d3b/img/favicon_32x32.png

[YouTube](https://youtu.be/tpIctyqH29Q)

***Programavimas ir programavimo kalbos***

## Kas yra programavimo kalbos? Kam jos reikalingos?

Visi standartiniai kompiuteriai supranta tą pačią kalbą - dvejetainę (angl. “binary”), 0 ir 1. Tačiau, tiek suprasti, tiek programuoti šia kalba yra išties rimtas iššūkis. Tam, kad nereikėtų programuoti nuliukais ir vienetukas, buvo sukurtos pirmosios programavimo kalbos, kurios buvo skaitomesnės bei lengviau suprantamos. Iš esmės, jos išverčia žmogaus parašytas komandas į dvejetaines instrukcijas.

Programavimo kalbos apjungia kelias kompiuteriui skirtas instrukcijas į lengvai suprantamus žodžius, funkcijas (pvz. kaip “while“ ciklas, kurį aptarėme prieš tai). Taip pat, aukštesnio lygio kalbos pasirūpina tokiais dalykais kaip: atminties valdymas, šiukšlių surinkimas bei dažniausiai yra nepriklausomos nuo naudojamos operacinės sistemos.

Kiekviena kalba skiriasi savo panaudojimo paskirtimi, turi savo stipriąsias ir silpnąsias vietas. Kai kurios padeda greičiau sukurti paprastas programas, tačiau nepalieka daug laisvės koreguoti konkrečias detales, kitos gali labai efektyviai spręsti uždavinius, bet tam reikalauja gilesnių žinių. Taip pat, viena nuo kitos jos skiriasi savo sintakse - kaip rašomas kodas ir kaip jis apdorojamas. Laimei pagrindinė logika išlieka ta pati tarp visų programavimo kalbų. Todėl išmokus vieną programavimo kalbą, dažnai greičiau galima įsisavinti naują.

[Programavimo kalbų yra labai daug](https://en.wikipedia.org/wiki/List_of_programming_languages) - vienos iš jų yra labai populiarios, kitos nelabai, kitos išvis jau yra nebenaudojamos. Kodėl taip yra? Programavimo kalbos sėkmės ar nesėkmės dažnai gali būti susijusios dalinai su išoriniais veiksnias, kurie nesusiję su pačia programavimo kalba. Pvz., [COBOL](https://en.wikipedia.org/wiki/COBOL) sėkmę JAV tikriausiai galima paaiškinti tuo, kad ji buvo rekomenduota naudoti valstybinėse institucijose programuoti tam tikriems uždaviniams . Analogiškai, [Fortran](https://en.wikipedia.org/wiki/Fortran) ir [PL/1](https://en.wikipedia.org/wiki/PL/I) sėkmę dalinai galima paaiškinti kelių kompiuterių gamintojų financine parama. Tačiau, nežiūrint į visas išorines priežastis, vis tik programuotojai apsprendžia, kurios iš kalbų naudojamos. Keletas priežasčių, kodėl programuotojai ir verslai vertina vienas kalbas labiau už kitas:

* Kalbos suderinamumas (t.y. kur galima paleisti šį kodą ir kaip jis apsijungia su kitom sistemom),
* iIšorinis palaikymas (t.y. ar yra bendruomenė ir gera dokumentacija),
* Sudėtingumas (t.y. ar lengva, ar sudėtinga išmokti - atsiminkime, kad programavimo lengvumas neatneša aukštos kodo kokybės, todėl kartais renkamės sudėtingesnes kalbas, tas, kurios gali užtikrinti aukštus saugumo standartus - pvz. kuriant banko sistemas),
* Efektyvumas (t.y. kaip išnaudoja serverio/kompiuterio resursus skirtingoms užduotims).

Kiekviena programa rašoma viena ar kita programavimo kalba, kuri vėliau verčiama mašininiu kodu, suprantamu kompiuteriui. Nors įmanoma tiesiogiai programuoti mašininiu kodu, aukšto lygio programavimo kalbos ženkliai supaprastinta kūrimo procesą.

Programavimo kalbų yra daug skirtingų tipų bei skirtingų sudėtingumo lygių, todėl kiekviena kalba labiau tinka tam tikros paskirties uždaviniams spręsti.

## Programų kūrimas

Programų kūrimas – sudėtingas procesas ir programavimas yra tik nedidelė šio proceso dalis. Pagrindinės ir dažniausiai pasitaikančios kūrimo dalys:

1. Reikalavimų surinkimas bei analizė.
2. Projektavimas ir įrankių (programavimo kalbos, platformos, duomenų bazių ir pan.), tinkamiausių problemai spręsti, parinkimas.
3. Programos rašymas pasirinkta programavimo kalba.
4. Testavimas. Jei testavimo metu paaiškėja, kad nepasiektas norimas kokybės lygis, taisomos problemos (grįžti į 3 žingsnį).
5. Dokumentavimas, jei reikia – vertimas į kitas kalbas.
6. Diegimas.

***Web programavimas***

Web programavimas tai viena iš programavimo sričių, kur kaip galutinį produktą kuriame internetinį puslapį. Tai gana plati sritis, kurią sudaro daug įvairių technologijų veikiančių tarpusavyje, kad sukurtų veikiantį web puslapį.

## Kodėl verta pasirinkti web programaivmą?

**1. Didelė paklausa**

Yra didžiulė programuotojų paklausa, kadangi įmonės bei organizacijos nuolatos ieško būdų kaip tapti pranašesniais už savo konkurentus, sukuriant internetines svetaines, kurios yra turtingos vizualais ir suteikia stiprų, prasmingą vartotojo patirtį. Itin didelė paklausa yra labiau patyrusiems web programuotojams, kurių įmonės ieško ypatingai ir siūlo nemenkus atlyginimus.

**2. Lengva išmokti**

Web programavimas laikomas gan lengvu išmokti, kadangi internetas yra pilnas pamokų, pavyzdžių ir kitų resursų, suteikiančių galimybę gauti patirties ir reikiamų įgūdžių. Išsilavinimo reikalavimai full-stack web programuotojams varijuoja nuo mokyklinio diplomo iki bakalauro laipsnio, o tai reiškia, jog jūsų išsilavinimas nesutrukdys jums gauti savo svajonių darbą. Jeigu esate pakankamai motyvuoti, užsidegę ir turintys pakankamai techninių žinių, jūs jau pakeliui tapti profesionaliu front-end programuotoju.

**3. Lanksti karjera**

Web programavimo karjera suteikia itin daug lankstumo, lyginant su kitomis specialybėmis. Tai vienas tų darbų, kurį galite atlikti dirbdami iš namų. Darbo laikas taip pat ganėtinai lankstus priklausomai nuo to, su kokiu klientu dirbate.

Norite būti pats sau bosas? Jokių problemų. Pagal 2014 metų statistiką, 25% web programuotojų buvo savarankiškai įsidarbinę, ir šis skaičius tik augs ateityje. Be to, įgiję pakankamai patirties, galite įsidarbinti konsultantu, arba įkurti savo paties internetinių sprendimų įmonę.

**4. Darbo paklausos didėjimas**

Web programuotojų darbo paklausos augimas 2018-aisiais metais planuojamas daug didesnis, nei kitų sričių specialistų. Šiuo metu numatomas 24% front-end programuotojų darbo vietų augimas iki 2024 metų, o tai yra išties daug lyginant su kitomis darbų sritimis. Šią paklausą lems nuolatos didėjantis mobiliųjų prietaisų bei e-komercijos populiarumas.

**5. Kūrybiškumas**

Web programuotojų darbas yra visiškai susijęs su kūrybiškumu bei problemų sprendimu. Kaip web programuotojas, jūs turėsite įgauti įgūdžių, kad galėtumėte kurti personalizuotus puslapius ir sukurti puikią vartotojo patirtį. Visa tai stipriai priklausys nuo jūsų kūrybiškumo bei gebėjimo spręsti naujas problemas.

**6. Tai nuolatos besikeičianti sritis**

Web nuolatos kinta, vystosi ir tobulėja. Tai reiškia, jog jūs visada turėsite galimybę nuolatos tobulėti ir mokytis, naudoti naujausius įrankiais ir technologijas.

Ką turi mokėti Full-stack Web programuotojas?

O dabar pažiūrėkite, kokius dalykus reikės išmokti norint tapti Full-stack Web programuotoju.

https://www.youtube.com/s/desktop/ce262d3b/img/favicon_32x32.png

[YouTube](https://youtu.be/erEgovG9WBs)

***Front-end vs Back-end***

## Front-end

Front-end – programavimo kodas, kuris perskaitomas ir vykdomas pas vartotoją (pvz. naršyklėje ar telefono programėlėje). Vartotojas atsisiunčia kodą, visus algoritmus ir logiką, ir kompiuteris paverčia kodą realiu rezultatu. Kodo paleidimas kompiuteryje atneša naudos - nereikia nuolatos siųstis viso rezultato, užtenka atsisiųsti tik reikiamą informaciją. Kartais, tai greičiau veikia ir atrodo dinamiškiau (puslapio perkrauti nereikia, o rezultatai keičiasi). Geras pavyzdys - Gmail ar Facebook, kur spaudinėjat atrodo natūraliai išoka kiti elementai (pvz. žinučių langas ar panašiai). Tačiau, kodą siųsti vartotojui gali būti bėda - ypatingai, jei kode yra slaptos informacijos (pvz. prisijungimų prie duomenų bazės, slaptų algoritmų - tokių kaip banko kredito apskaičiavimas). Trys pagrindinės kalbos su kuriomis rašome Front-end'ą: HTML, CSS ir JavaScript. HTML naudojamas turiniui ir struktūrai, CSS – stiliui, o JavaScript – logikai (truputį detaliau apie tai vėliau). HTML ir CSS kalbos naudojamos tik Front-End'e, nes jos skirtos sugeneruoti puslapio vaizdui (tai gali padaryti tik naršyklė, todėl būna kodą siųsti varotojui), bet čia jaudintis nereikia - šios programavimo/kodavimo kalbos neturi logikos, tad ir jokios slaptos informacijos neperduos - tik tekstą ir stilių. JavaScript yra bendrinė programavimo kalba - tinka tiek aprašyti Front-end logiką, tiek back-end'e servisus.

## Back-end

Back-end yra kodas yra paleidžiamas ne vartotojo įrenginyje, o serveryje. Vartotojas serverio užklausia, serveryje esantis Back-end kodas pagal aprašytą logiką sugeneruoja atsakymą ir jį grąžina. Nors logiką galime aprašyti ir Front-end'e, kartais tai yra efektyviau su galingais serverio resursais, o gal tiesiog nenorime viešinti kodo logikos. Back-end'ą galėtume rašyti su viena iš šių kalbų: JavaScript, PHP, Python, JAVA, C# ir t.t.

## Realus pavyzdys: Front-end vs Back-end?

https://www.youtube.com/s/desktop/ce262d3b/img/favicon_32x32.png

[YouTube](https://youtu.be/n4kTk2SAH1E)

## Kaip atskirti Front-end nuo Back-end?

Pirmą kartą susidūrus su šiais terminais yra ganėtinai sunku suvokti bei atskirti, kur baigiasi Front-end ir kur prasideda Back-end. Tačiau yra vienas paprastas triukas, kuris gali jums padėti. Taigi, norint atskirti kur yra kas, reikėtų pagalvoti, kurioje vietoje yra vykdomas kodas. Jei skaičiavimai atliekami vartotojo įrenginyje - tai bus Front-end, o jeigu užklausa yra siunčiama į kažkokį nutolusį serverį ir skaičiavimai atliekami ten - tai bus Back-end.

***Pažvelkime giliau į Front-end kalbas***

Ką tik susipažinome su programavimo kalbomis bei išmokome atskirti Front-end'ą nuo Back-end'o. Kursus pradedame nuo front-end'o - kodo, kuris perskaitomas ir paleidžiamas pas vartotoją (o ne serveryje). Pažiūrėkime iš ko susideda front-end'as!

https://www.youtube.com/s/desktop/ce262d3b/img/favicon_32x32.png

[YouTube](https://youtu.be/GuxAZH4W62w)

## Kaip veikia Front-end? (supaprastintai)

Mes, vartotojai, į naršyklę įrašome puslapio pavadinimą - ir su interneto magijos pagalba - mūsų naršyklė sužino į kokį serverį kreiptis (t.y. jo IP adresą). Iš karto užmezga ryšį HTTP protokolu ir atsiunčia atsakymą (angl. response) - HTML failą. Naršyklė perskaito šį failą ir atvaizduoja rezultatą ekrane.

Kaip žinome - HTML tėra puslapio struktūrą ir turinys. O tai kaip atsiranda spalvos? Kadangi mes atsisiunčiame tik HTML failą - būtent jame turi būti aprašyta, iš kur atsisiųsti stilių (CSS) ar logiką/dinamiškumą (JS). Tai vadinam failų importavimu - į HTML.

### HTML

HyperText Markup Language - dažnai vadinama kodavimo kalba (ji nėra laikoma programavimo kalba, nes neturi logikos). HTML kalba yra naudojama aprašyti puslapio turinį ir suteikti jam struktūrą.

### CSS

Cascading Style Sheets - stilizavimo kalba. Kaip ir HTML, ji nėra laikoma programavimo kalba. Jos pagrindinis tikslas yra apipavidalinti ir stilizuoti HTML turinį. Skirtingai nei HTML, CSS veikti viena negali. Ją būtina naudoti kartu su HTML.

### JS

JavaScript - programavimo kalba, kuri vadovaujasi [ECMAScript](https://en.wikipedia.org/wiki/ECMAScript) standartu. Kol kas turėkime galvoje, kad tiek ECMAScript (ES), tiek JavaScript (JS) yra tas pats dalykas, o kiek vėliau įsigilinsime į šios programavimo kalbos painią istoriją ir evoliuciją.

Būtent JS jau yra tikra programavimo kalba - su ja galima rašyti logiką, kurti puslapio dinamiškumą. Nors ji kilusi iš Front-end'o - dabar yra plačiai naudojama visose industrijos dalyse - tiek Front-end'e, tiek Back-end'e; nuo puslapių ir mobiliųjų programėlių kūrimo iki procesų automatizacijos ir robotų valdymo.

Daugelis didžiausių technologinių įmonių plačiai taiko JS ir vertina jos atnešamą naudą bei greitį, palyginus su alternatyvomis. Iš tiesų, viena stipriausių šios kalbos savybių - galimybė kodą rašyti ta pačia sintakse tiek Front-end'e, tiek Back-end'e (ko negalima daryti su nei viena kita programavimo kalba) - o tai sutaupo ir laiko, ir resursų.

Dažnai ši kalba painiojama su Java - bet tai dvi, visiškai skirtingos programavimo kalbos. Kodėl pavadinimas panašus? Na, vėl - paini istorija nulėmė, kad Java populiarumu pasinaudojo ir JavaScript.

***Kaip veikia internetas?***

Mes išsiaiškinome kaip su trim muškietininkais (HTML, CSS ir JS 😃) mes galime susikurti internetinį puslapį ir kaip šį puslapį atvaizduoja naršyklė. O kaip mus pasiekia tas puslapis? Koks tas kelias nuo kompiuterio iki serverio ir atgal? Ir kas tas magiškas internetas? Daug klausimų, mažai atsakymų…

Pažiūrėkite vaizdo įrašą ir susipažinkite su pagrindiniais interneto veikimo principais, kurie padės jums atsakyti į šiuos klausimus.

https://www.youtube.com/s/desktop/ce262d3b/img/favicon_32x32.png

[YouTube](https://youtu.be/AEaKrq3SpW8)

Kartais atrodo, kad internetas “kabo debesyse”, deja, kol kas dar ne - didžioji dalis interneto yra tiekiama laidais. Tad ryšys tarp Amerikos ir Europos sujungtas povandeniniu laidu. Ar žinojote, kad viena didesnių problemų su šiais laidais - rykliai? Jie retkarčiais atsikanda laidų ir nutraukia ryšį didžiausiems puslapiams!

Dėl privatumo bei greičio, kai kurios įmonės ir institucijos pačios tiesiasi savo laidus - pvz. IBM turi tiesioginį laidą per visą metropolitaninį miestą iki duomenų centro - kad saugiai ir nenutrūkstamai gautume didelius duomenų kiekius.

Šiais laikais didžiausios tech kompanijos turi daugiausiai povandeninių kabelių. [Submarine Cable Map](https://www.submarinecablemap.com/) puslapyje galite pamatyti kiek iš viso yra povandeninių interneto kabelių.

Taip pat, rekomenduoju susipažinti su įdomia interneto statistika ir faktais [Internet Health Vitals | Facts and Figures — The Internet Health Report 2020](https://2020.internethealthreport.org/slideshow-internet-health/#1).

[**UŽBAIGTI PAMOKĄ**](https://codeacademy.online/dashboard/modules/intro2/)

[20:26] Lukas Birgiolas

<https://www.w3schools.com/html/html_basic.asp>

HTML Basic

W3Schools offers free online tutorials, references and exercises in all the major languages of the web. Covering popular subjects like HTML, CSS, JavaScript, Python, SQL, Java, and many, many more.

<https://code.visualstudio.com/>