# Web Scraping, jeb tīmekļa vietņu *skrāpēšana*

Web scraping jeb tīmekļa vietņu skrāpēšana ir automatizēta mijiedarbošanās ar web lapām primāri datu iegūšanas, apstrādes un analīzes nolūkos. Web scraping metodes un instrumenti tiek lietoti situācijās, kur web lapām nav pieejama API ([Application Program Interface, jeb lietojumprogrammu saskarne](https://lv.wikipedia.org/wiki/Lietojumprogrammas_saskarne)), respektīvi lapām, kas nav būvētas atvērtai integrācijai ar citām sistēmām, bet ir paredzētas galvenokārt tikai cilvēku lietošanai. Skrāpētājs ir sava veida robots, kurš izdara garlaicīgo darbu, ejot cauri lērumam kādas vietnes apakš sadaļu ievācot tikai nepieciešamo info.

Tipiski piemēri interneta datu skrāpēšanas darbiem ir kontaktu iegūšana mārketinga vajadzībām (piemēram, konkurentu izplatītāju mājaslapas, e-pasta adreses), tiešsaistes tirgus lapu skrāpēšana, lai atrastu izdevīgākos darījumus (piemēram, Rīgas nekustamā īpašuma tirgus portālā ss.lv), datu ieguve akadēmiskajiem pētījumiem (es pats reiz savā maģistra darbā izmantoju skrāpēšanas tehnikas, lai pētītu Dark Web forumu komunikāciju tīklu struktūru), sporta bukmeikeru likmju analīzei (piemēram meklējot arbitrāžas iespējas starp dažādām bukmeikeru lapām) un daudzi citi. Pirmos divus no šiem piemēriem apskatīsim kursa laikā un pildīsim dažādus praktiskos uzdevumus saistībā ar tiem.

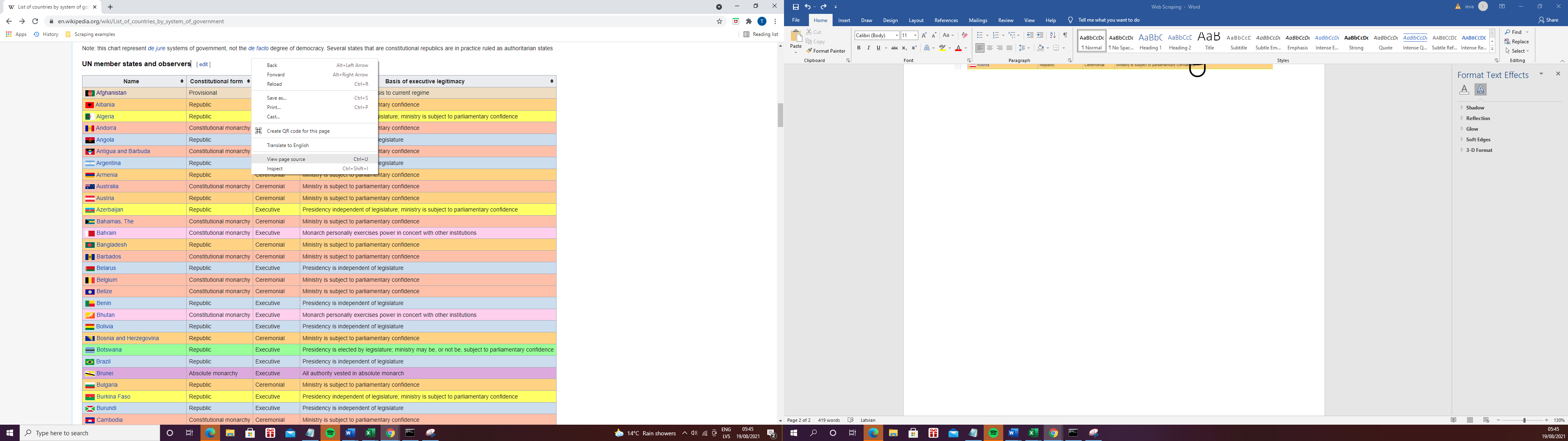
Skrāpētāju izveide var būt diezgan aizraujošs process, jo brīžiem var likt sajusties kā hakerim, kurš mijiedarbojās ar sistēmām, šajā gadījumā interneta vietnēm, neparedzētos veidos, lai iegūtu sev kādu priekšrocību pret tiem, kas šīs tehnikas nepārzin. Tomēr vienmēr ir jāatceras par skrāpēšanas ētiku un jābūt uzmanīgam, lai nepārkāptu kādus noteikumus. Neskatoties uz to, interneta vide attīstās tik ātri, un web lapu paliek tik daudz, ka lielākā daļa *skrāpētāju* paliek likumiski un ētiski pelēkās zonās, tādēļ bieži nākas uzticēties tikai savam morālajam kompasam, lai saprastu, vai konkrētais uzdevums ir piemērots, lai to risinātu, būvējot skrāpētājus. Kursa noslēgumā runāsim par to, kā veidot ētiskus skrāpētājus, kas pakļaujas datu aizsardzības regulām un politikai, lai izvairītos no potenciālām nepatikšanām.

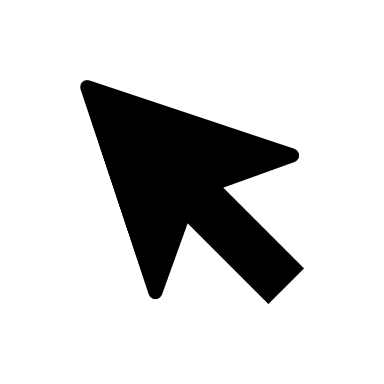
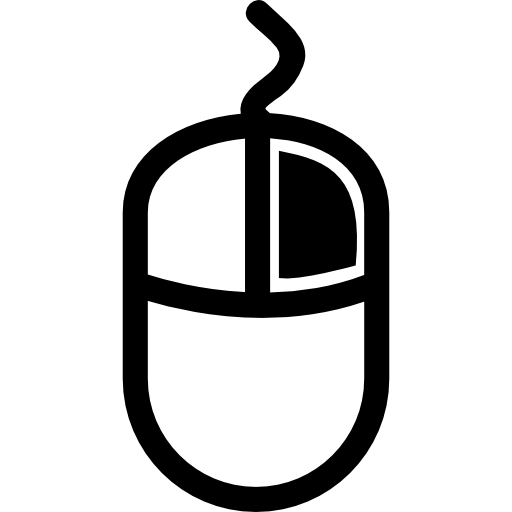
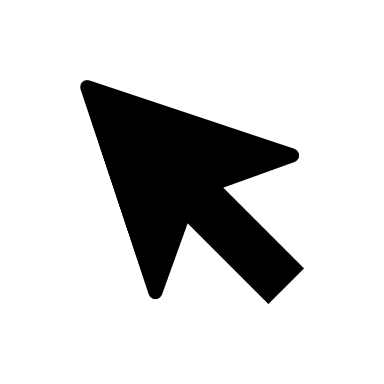
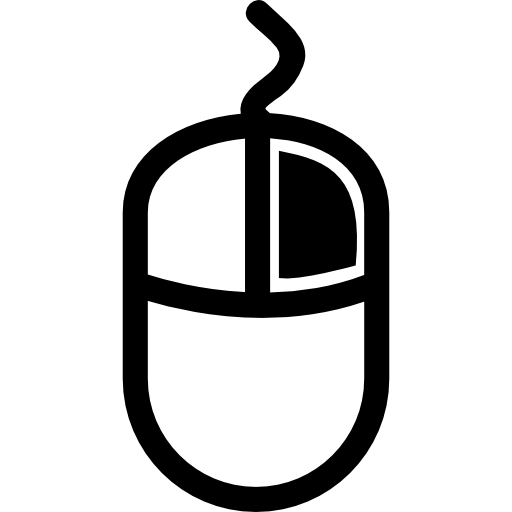
# Interneta vietņu pirmkods

Lai sāktu mācīties būvēt skrāpētājus, ir nedaudz jāsaprot, kā tiek būvētas mājaslapas. Nav obligāti jābūt web dizaina ekspertam un nav jāmāk veidot mājaslapas pašam, bet vajag prast “paskatīties zem motora pārsega” interneta pārlūkprogrammām, lai atrastu nepieciešamo.

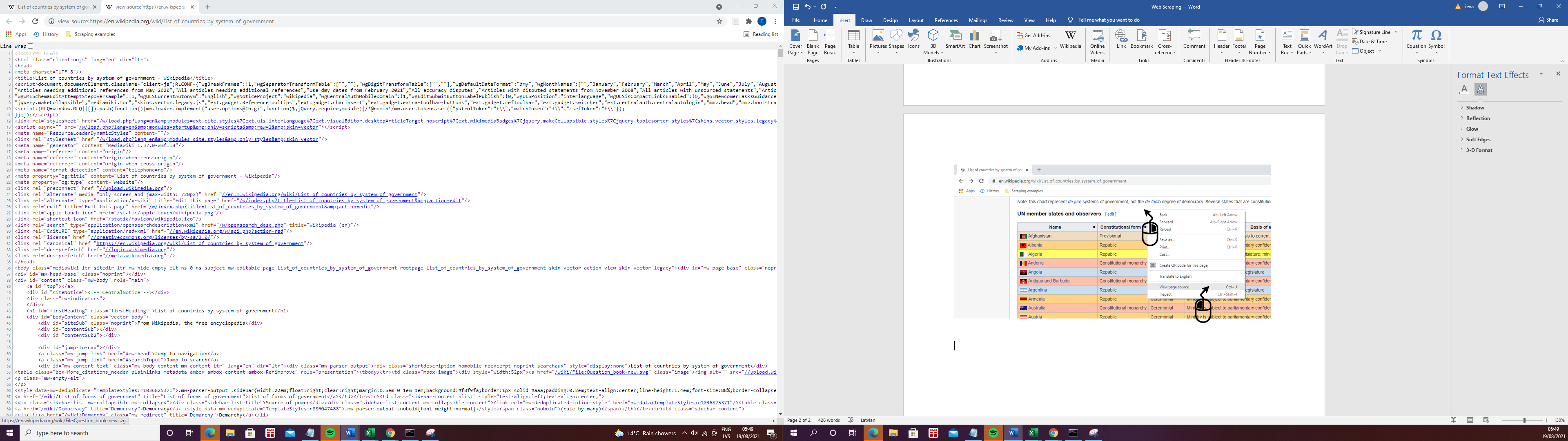
Sāksim ar vienkāršu piemēru, izmantojot pasaulē lielāko enciklopēdiju – interneta vietni Wikipedia un populārāko pārlūkprogrammu [Google Chrome](https://www.google.com/chrome/). [Šī vikipēdijas lapa](https://en.wikipedia.org/wiki/List_of_countries_by_system_of_government) apkopo visu Apvienoto Nāciju valstu valdības sistēmas vienā tabulā. Pieņemsim, ka kādam pētījumam mums nepieciešams iegūt procentuālo sadalījumu starp to, cik daudz pasaulē ir republiku, konstitucionālo vai absolūto monarhiju, vai cita veida pārvaldes formu. Mēs varētu vienkārši ņemt un manuāli saskaitīt katru, bet tas būtu laikietilpīgi, ar lielu iespēju kļūdīties un galu beigās mēs taču gribam iemācīties web scraping.

Atverot lapu pārlūkā mēs redzam dažādi sakārtotu informāciju – tabulas, linkus, izvēlnes, bildes, sarakstus – viss ir sakārtots tā, lai būtu pēc iespējas vieglāk uztverams cilvēkam. Šo vizuālo sakārtojumu veic konkrētā pārlūkprogramma, mūsu gadījumā Google Chrome. Pārlūkprogrammas var uzskatīt kā tulku starp mājaslapas pirmkodu un cilvēkiem saprotamu attēlojumu. Ja mēs gribētu apskatīties kā pārlūkprogrammas pašas “redz” mājaslapu, to varam izdarīt, spiežot labo peles klikšķi kādā tukšākā lapas apgabalā → View page source, vai vienkārši izmantojot Ctrl+U īsceļu.

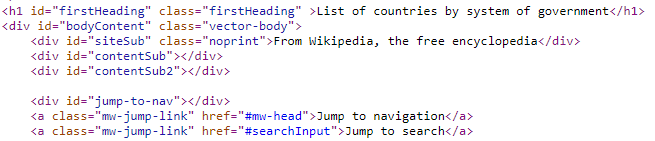




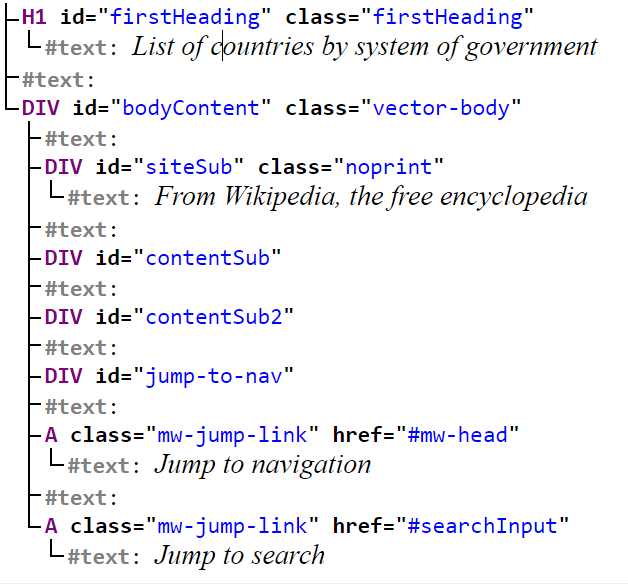
To izdarot jaunā cilnē atveras šīs konkrētās lapas pirmkods. Šaušalīgs, cilvēkam neuztverams skats ar vairāk kā 2000 koda rindām. Toties pārlūkam šis ir tieši tas, kas nepieciešams, lai tas zinātu, kas jāattēlo, kurā vietā uz ekrāna.



Un arī mums būs jāmācās saprast šī pirmkoda struktūra, jāsaprot kā tajā orientēties, lai būvētu programmas, kas tāpat kā pārlūki saprot šo interneta “zem-pārsega” valodu. Pārejot pirmā brīža izmisumam, kas varētu rasties, ieraugot šīs garās koda rindas, un parullējot cauri visam pirmkodam, lēnām sāk rasties sajūta, ka šim visam apakšā ir vienojoša struktūra. Šo struktūru saīsina ar abreviatūru DOM (Document Object Model). Vizuāli to iespējams iztēloties kā loģisko koku, apskatīsim tuvāk konkrētā koda vienu apgabalu

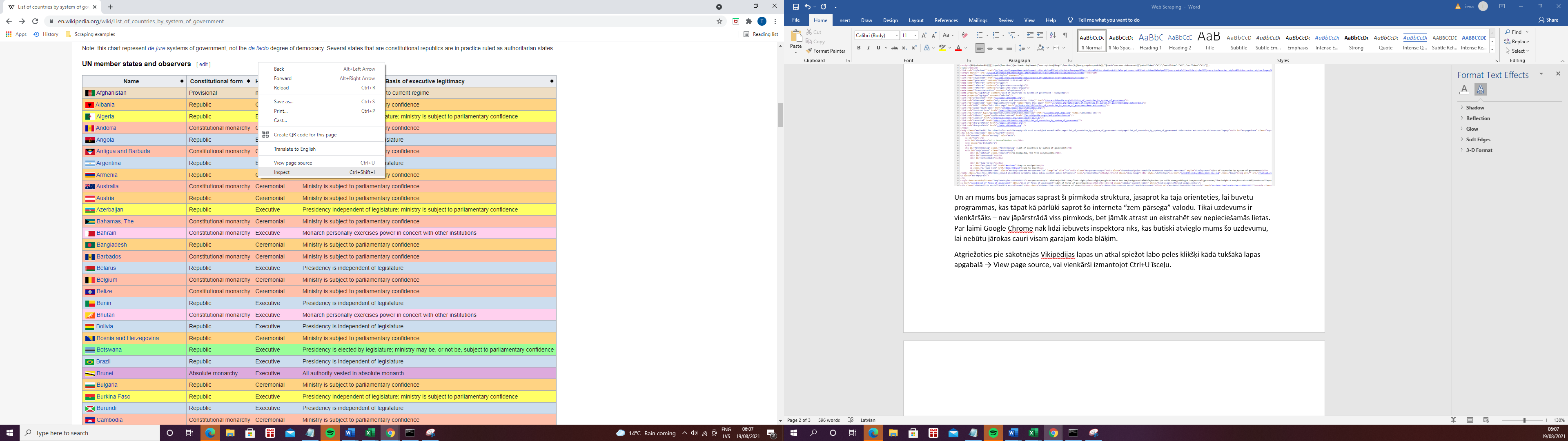


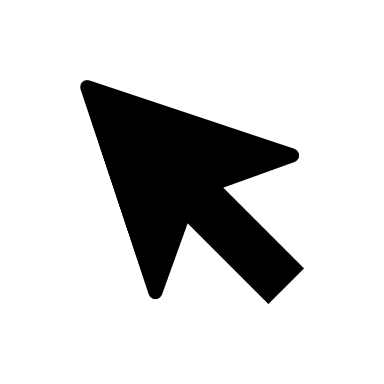
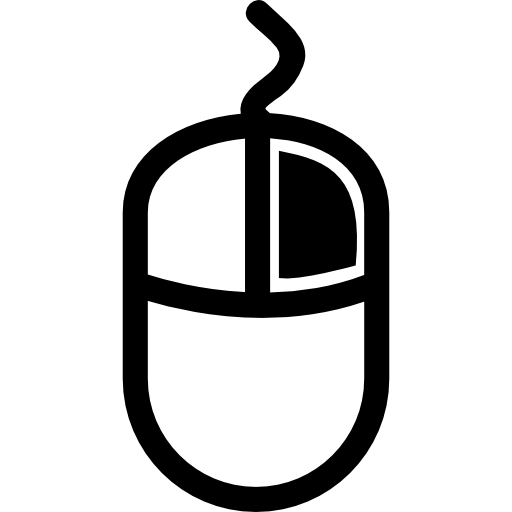
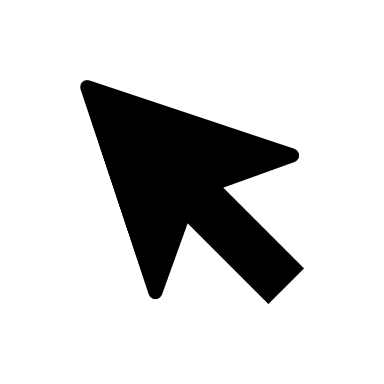
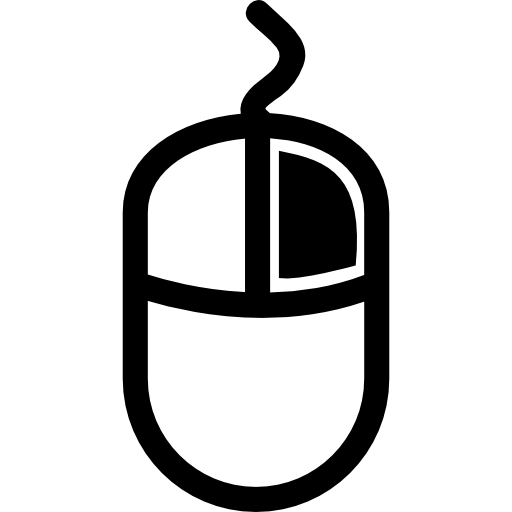
Uzreiz pamanām, ka atsevišķas koda sadaļas ir ievertas starp “<” un “>” simboliem. Šos sauc par tagiem. Loģiskajā kokā tos vizuāli reprezentētu ar mezglu punktiem. Katrs tags sākas ar nosaukumu – pirmo simbolu virkni starp “<” un pirmo atstarpi. Šajā gadījumā ir trīs veidu tagu nosaukumi <h1>, <div> un <a>. “h1” tipiski ir virsraksta tags, “div” ir nodalījuma tags, kas atdala atsevišķas lapas nodalījumus un “a” ir linku tags. Pamanam arī, ka tagi iezīmē kaut kāda satura sākumu un beigas. Beigu tagam pēc “<” simbola seko “/” simbols iezīmējot šī taga satura beigas, kā piemēram pirmās rindiņas noslēgumā ir </h1> (šajā piemērā nav redzams, kur noslēdzas otrās rindiņas “div” tags, vietas taupīšanas nolūkos). Starp šiem sākuma un beigu tagiem ir saturs, piemēram pirmajā rindiņā tas ir virsraksts “*List of countries by system of government*”. Vēl tagiem ir atribūti, kas arī ir ietverti sākuma tagā starp “<” un “>” simboliem uzreiz aiz taga nosaukuma. Piemēram pirmajā rindiņā esošajam “h1” tagam ir divi atribūti “id” un “class”, kuru nosaukumi šajā gadījumā abiem ir vienādi – “firstHeading”. Par to, kādēļ katrs tags un tā atribūti ir nepieciešami šobrīd mums nav jāuztraucās, ir svarīgi saprast struktūru, kas vizuāli attēlots kā loģiskā koka grafiks zemāk

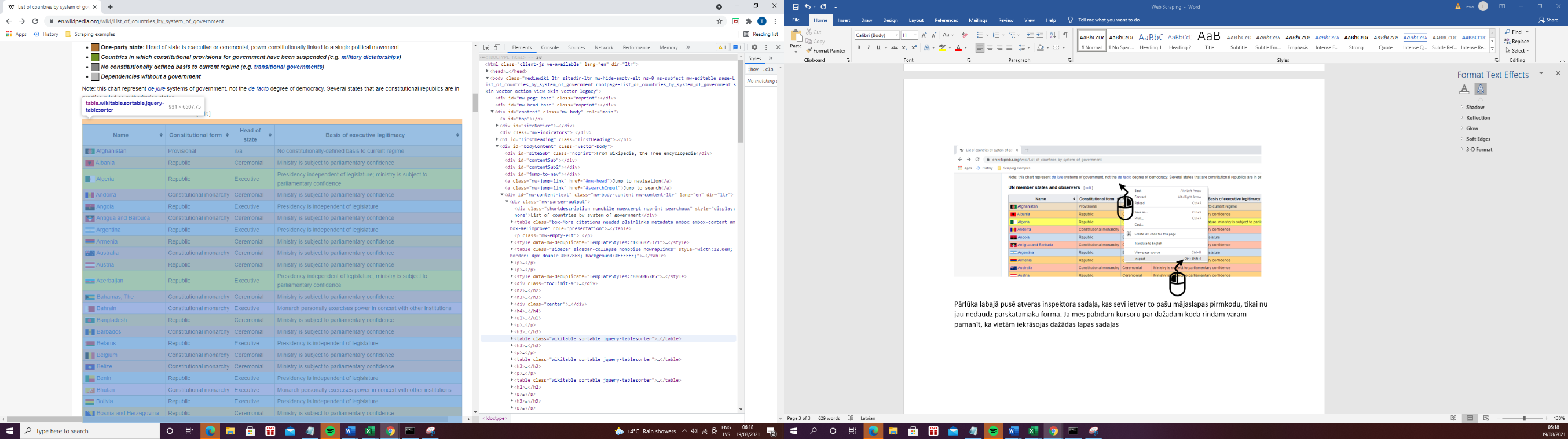


Tā kā šī ir tikai maza sadaļa no visas lapas koda, var secināt, ka visai lapai šis koks paliks ļoti komplicēts. Lai atvieglotu mums darbu un lai nebūtu jāpārstrādā viss pirmkods, bet varētu atrast un ekstrahēt sev nepieciešamās lietas, Google Chrome nāk līdzi iebūvēts inspektora rīks, kas būtiski atvieglo mums šo uzdevumu, lai nebūtu jārokas cauri visam garajam koda blāķim.

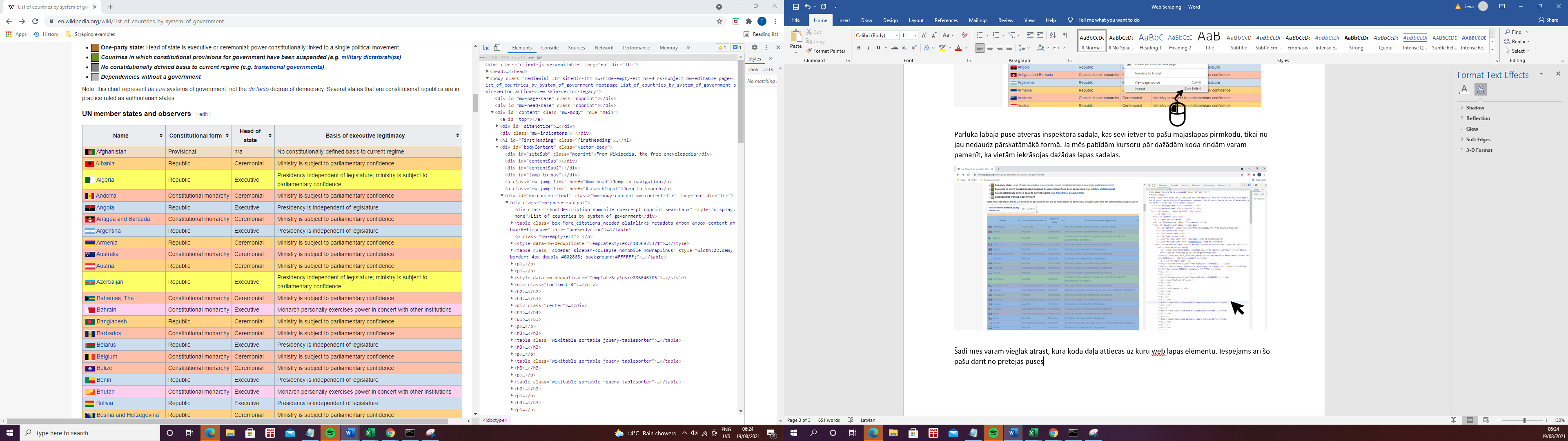
Atgriežoties pie sākotnējās Vikipēdijas lapas un atkal spiežot labo peles klikšķi kādā tukšākā lapas apgabalā → Inspect, vai vienkārši izmantojot Ctrl+Shift+I īsceļu.

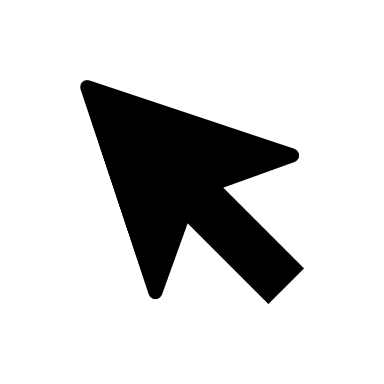
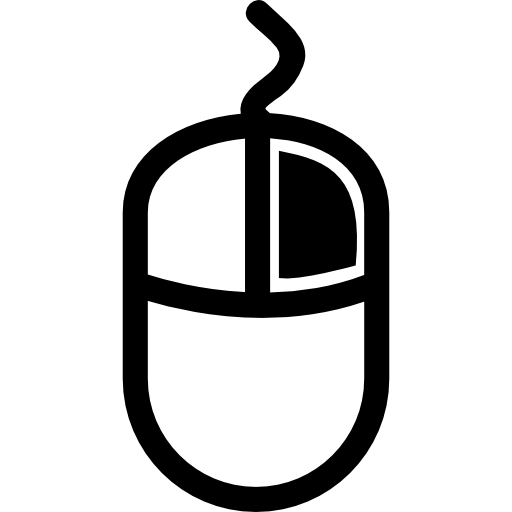


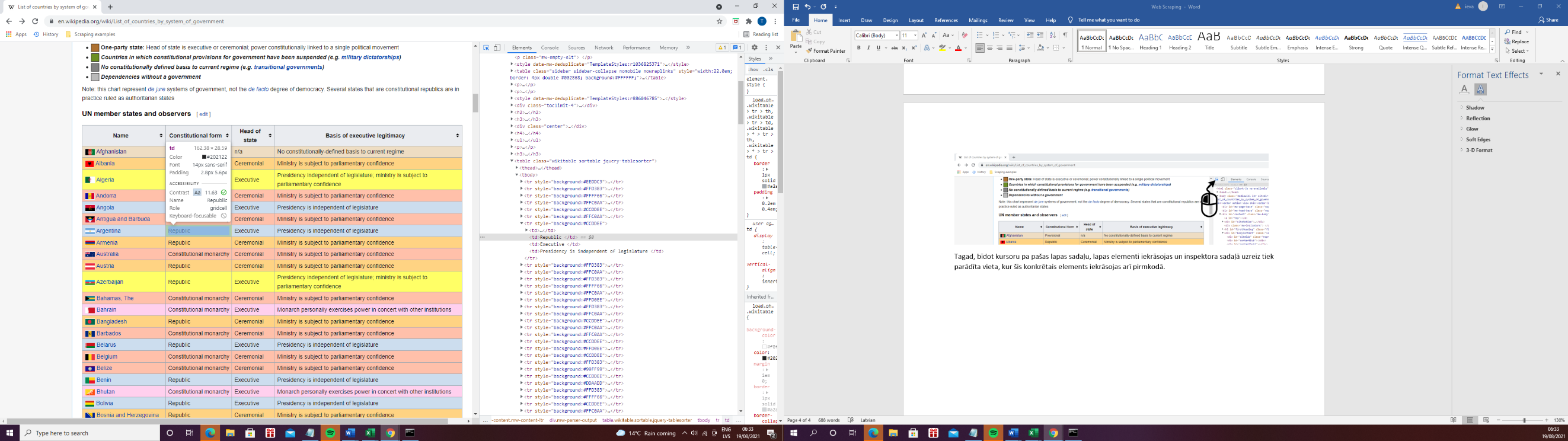


Pārlūka labajā pusē atveras inspektora sadaļa, kas sevī ietver to pašu mājaslapas pirmkodu, tikai nu jau nedaudz pārskatāmākā formā. Ja mēs pabīdām kursoru pār dažādām koda rindām varam pamanīt, ka vietām iekrāsojas dažādas lapas sadaļas.

Šādi mēs varam vieglāk atrast, kura koda daļa attiecas uz kuru web lapas elementu. Iespējams arī šo pašu darīt no pretējās puses. Inspektora sadaļas augšējā kreisajā stūrī atrodas ikona, ko nospiežot, tā iekrāsojas gaiši zila





Tagad, bīdot kursoru pa pašas lapas sadaļu, lapas elementi iekrāsojas un inspektora sadaļā uzreiz tiek parādīta vieta, kur šis konkrētais elements iekrāsojas arī pirmkodā.

Šeit mēs redzam, kur tieši kodā atrodas tā vieta, kas satur informāciju par to, ka Argentīna ir republika. Arī pats kods ir izkārtots koka formā, kur atsevišķas sadaļas iespējams paslēpt, vai tieši pretēji – paplašināt. Tālāk apskatīsim kā mēs šo varam izmantot automātiskai datu ekstrakcijai ar Python.

# Pirmais skrāpētājs

Lai varētu sākt automatizēt skrāpētāju mums vispirms vajadzēs lejuplādēt pašu lapu

