* Mnozstvo pristupov k vyvoju frontendu, casto skoncia ako obrosky monolit so vsetkymi uskaliami
* Mikrofronend inspirovany microservismi – modelovane podla business domeny a skryvanie implementacie, decentralizacia rozhodnuti a nasadeni novych verzii. To prinasa uskalia: komunikacni overhead, problem s vykonom a problemi roznych user interface prvkov ak nie je zvladnuty design
* Pristupy k tvorbe frontendu web aplikacie:
  + JAMstack architecture – staticke stranky s dynamickymi vlastnostami pomocou JS a API volaniami nasadene na CDN – ziadny server, kompletna separacia frontendu a backendu
  + Aplikacia vykreslena klientom – HTML a JS subory, pouziva sa manipulacia s DOM na zmenu pohladu bez znovunacitania stranky
    - SPA – single-page applikacia – jedno staticke HTML a simulacia navigacie napriec strankami – pouzivatel ostava stale na tej istej stranke. Vyuziva sa JS a HTTP pre ziskanie dat, rooting point, life-cycle methods
  + Isomorfne aplikacie – kombinacia vykreslovania serverom a klientom. Uvodna stranka vygenerova serverom na zaklade vzoru a dat z databazi a API volani – rychlejsie nacitanie ako SPA avsak pomalsie umoznenie interakcie – caka sa na javascript
  + Staticke HTML – jednoduche stranky s obmedzenimi moznostami

**Mikrofrontend** – rozsirenie myslienky mikrosluzieb na frontend – viacero mensich aplikacii v jednej – bezia, su vyvijane a nasadane samostatne. Web aplikacia ako kombinacia funkcii a biznis podoblasti.

* Horizotalne rozdelene – viacero aplikacii na jednej stranke, je potrebna koordinacia timov, je potrebne zabezpecit komunikaciu medzi aplikaciami, napr. Pomocou eventov(udalosti?)
* Vertikalne rozdelene – kazda stranka moze pouzivat inu aplikaciu

Oboje: zdielanie informacii napr. Pomocou query stringu, url, session storage, local storage

**Kompozicie mikrofrontendu**:

* Client-side – jednotlive aplikacie sa spajaju na strane klienta – klient nacita skelet do ktoreho su “vlozene” jednotlive aplikacie
* Edge-side – na strane CDN, na XML zalozene Edge Side Include (ESI), nie kazdy implementuje rovnako
* Server-side – origin server spaja rozne microfrontendi do vyslednej stranky, ktoru vracia, pre velmi dynamicke stranky problem skalovatelnosti

**Mikrosluzby a mikrofrontend**

Mikrosluzby – pomerne rozsirene, architektura orientovana na sluzby – viacero volne previazanych sluzieb ktore spolu komunikuju cez API, kazda sluzba ma jednu ulohu , zrele implementacie backendu

Na frontende skor monolity, microfrontend sa snazi prevzat principy z mikrosluzieb:

* Modelovane okolo biznis domen
* Doraz na automatizaciu
* Skryvanie implementacnych detailov
* Decentralizacia rozhodnuti
* Nezavysle nasadovanie
* Izolacia zlyhani

Napriek nedostatocnemu vyskumu, prve prace ukazuju potencial avsak stale nie je dostatocne zrela architektura

Systematic Multivocal Literature Review (MLR) - male mnozstvo peer-reviewed clankov, podrobne popisany sposob vyberu literatury, skusit sa pozriet na sposoby vyhodnotenie “sedej literatury”

Najcastejsie dovody adopcie pristupu: prilisna komplexita monolitu (mnozstvo kodu, komplikovane previazanie, problem s organizaciou) a potreba skalovania (timu, nezavysle nasadzovanie, rychlost nasadzovania)

Dalsie: zmena coding styles, pomale zaskolenie, komplikovane inovacie, prilisna abstrakcia

Benefity:

* Moznost pouzit viacero technologii
* Samostatne timy zamerana na funkcionalitu
* Samostatny vyvoj, nasadzovanie, sprava micro frontendovych aplikacii
* Lepsia testovatelnost
* Izolacia chyb a odolnost
* Skalovanie
* Rychlejsie zaskolenie
* Rychlejsie uvodne nacitanie stranky
* Zlepsena vykonnost
* Moznost inovacii

Problemy:

* Zvacsenie mnozstva dat
* Opakovanie kodu
* Zdielane zavislosti
* Konzistencia pouzivatelskeho rozhrania
* Monitorovanie
* Zvysena komplexita
* Sprava (governance)
* „ostrovy znalosti“
* Rozdielnost prostredi
* Zvysenie rizika pri vydavani aktualizacii
* Pristupnost

Mikrofrontend nie je odpoved na vsetko, na jednej strane umoznuje vacsie timy avsak prinasa zvacsenu komplexitu, neustale zlepsovanie automatizacie, zvacsenie mnozstva kodu, podrobny navrh architektury na zaciatku projektu

Otvorene otazky (v case pisania clanku, niektore mozu byt uz vyriesenie):

* Porovnanie s inymi pristupmi
* Vyhodnost duplikacie kodu
* Vzory a anti-vzory(anti-paterns)
* Kedy je mikrofrontend na pritaz
* Kombinacie s inymi pristupmi
* Migracia existujucich aplikacii na mikrofrontend
* Technicky dlh