Překladače - Definice jazyka

NTI/PRK, LS 2024, LenkaKT

Úvodní vaření mozků

Jak je definován přirozený jazyk?

Co musíte znát, chcete-li správně mluvit česky?

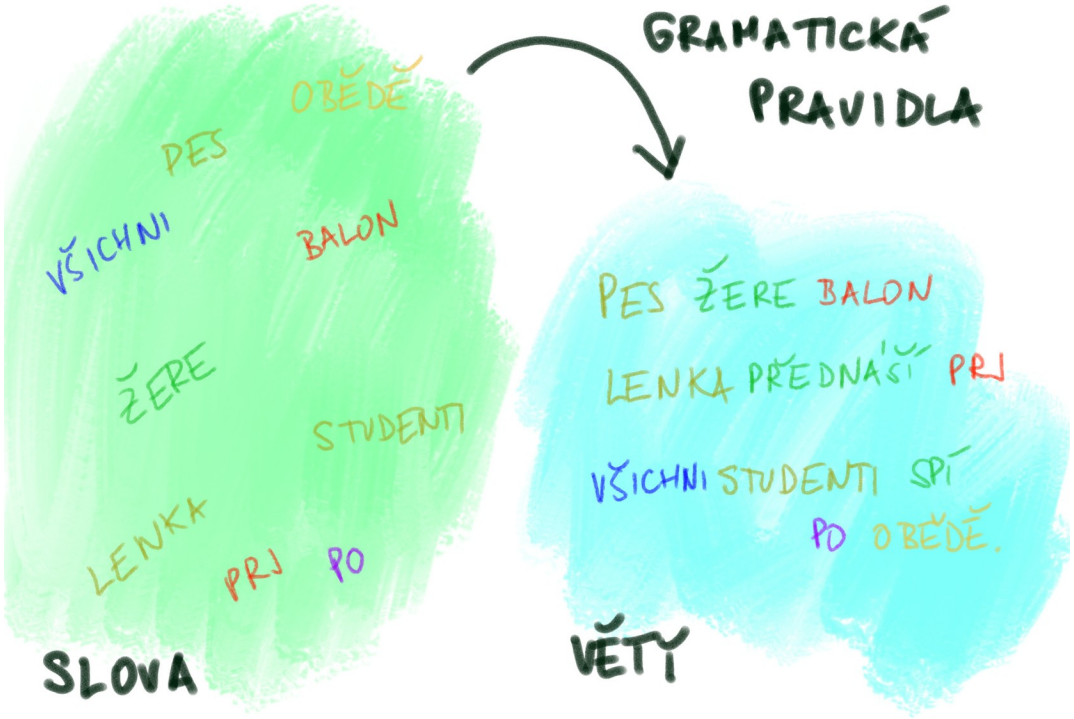
Jak se “to” jmenuje?

Z čeho se “to” skládá?

Co “to” definuje?

A jak “to” souvisí s programovacími jazyky?

Co definuje jazyk?



K zamyšlení: Množina sigma je konečná - co je sigma přirozeného jazyka? (Slov je přece nekonečně).

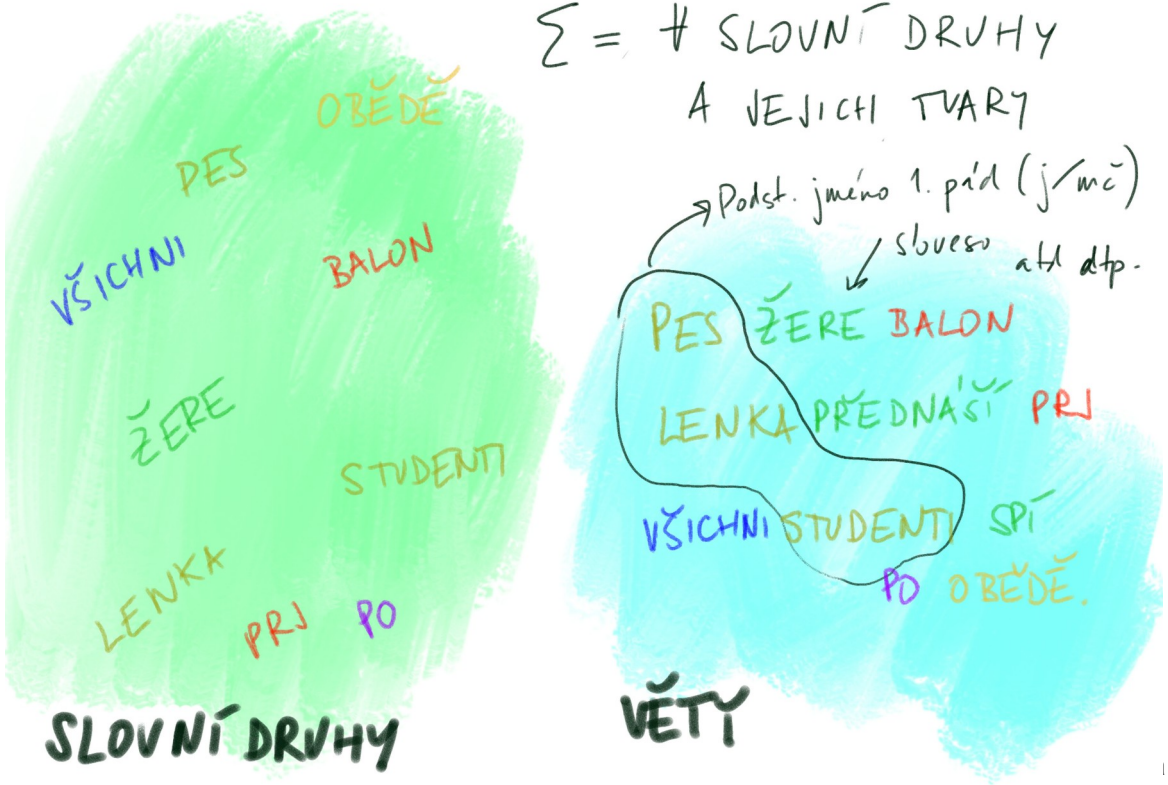
Formálně?

Přirozený jazyk tvoří **slova** spojená ve **větách** podle **gramatických pravidel**.

Problém: Slov je přece nekonečně!

Jak v tom uděláme pořádek?

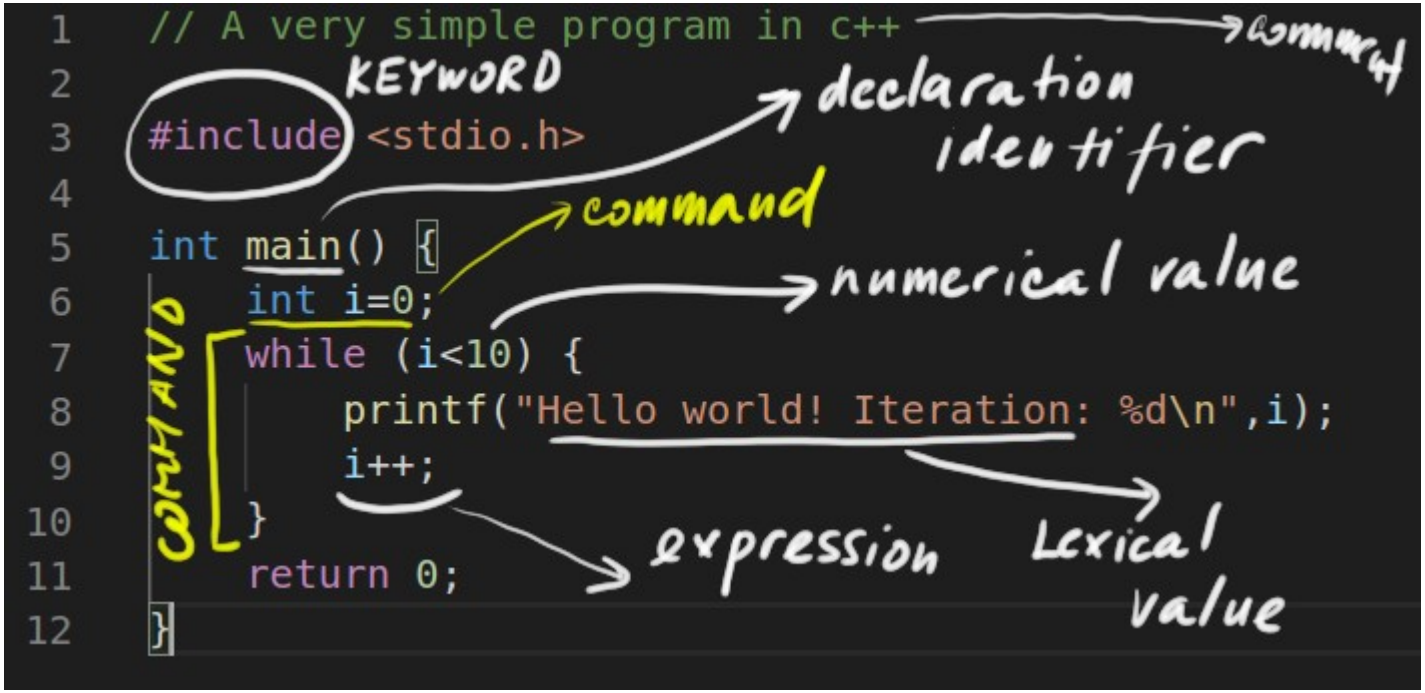
Slovní druhy!



Slovní druhy v programovacích jazycích?

```
1   // A very simple program in c++
2
3   #include <stdio.h>
4
5   int main() {
6       int i=0;
7       while (i<10) {
8           printf("Hello world! Iteration: %d\n",i);
9           i++;
10      }
11      return 0;
12  }
```

Slovní druhy v programovacích jazycích!



“Slovní druhy” - programování

Klíčová slova

Identifikátory (proměnné, funkce…)

Hodnoty (číselné, literální…)

Deklarace (funkce, makra, datové struktury, proměnné)

Příkazy (jednoduché, strukturované, složené…)

Výrazy (aritmetické, kombinace funkcí…)

Komentáře

Teorie: Abeceda jazyka

Definice: Abeceda ∑ je konečná množina symbolů.

Přirozené jazyky: Všechny slovní druhy ve všech tvarech a interpunkce.

Programovací jazyky: Klíčová slova, příkazy, operátory, interpunkce (mezery, středníky apod.)

Teorie: Slovo/Zdroják

Def: Slovo je libovolný řetězec symbolů z abecedy $\sum$. Symboly se mohou opakovat.

!!! Pozor - v teorii jazyků se používá “slovo” pro to, co je v praxi “věta”.

Pro programovací jazyky znamená “slovo” celý zdroják.

Def: Množina všech slov sestrojitelných nad abecedou $\sum$ se označuje $\sum^*$.

Poznámka: Množina $\sum^*$ je vždy nekonečná. Obsahuje prázdný symbol a prázdné slovo (řetězec prázdných symbolů).

Teorie: Jazyk

Def:

Jazyk je libovolná množina, konečná nebo nekonečná, sestavená ze slov sestrojených nad stejnou abecedou $\sum$.

Pozn.: Každý jazyk sestrojený nad abecedou $\sum$ je vždy podmnožinou $\sum^*$.

Teorie: Regulární jazyky

Jen pár poznámek:

- Regulární jazyky lze definovat regulárním výrazem.
- Regulární jazyky lze rozpoznávat konečným automatem.
- Regulární jazyky nemohou obsahovat vnořené závorky s neomezeným počtem úrovní.

Příklad: Dekadická čísla

Decimal number:

CORRECT
- ✓ −123.123
- ✓ +123
- ✓ −123.0
- ✓ −123.00
- ✓ 0.0
- ✓ 3.123

INCORRECT
- ✗ −.
- ✗ +.123
- ✗ .0
- ✗ .123
- ✗ +000
- ✗ −000

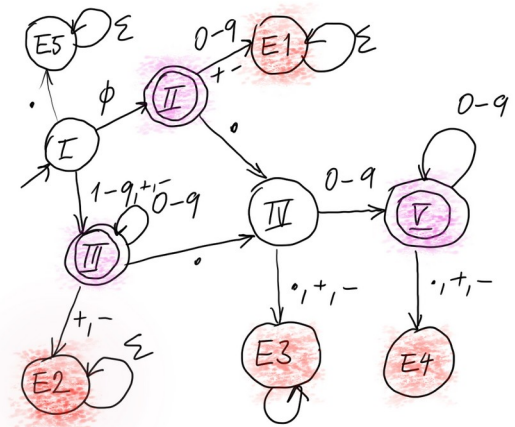
Regular expression:

$$(\underbrace{0}_{PI} \mid \underbrace{[+,-]?[1-9][0-9]^*}_{PII})(\underbrace{.[0-9][0-9]^*}_{PIII})?$$

- 0 … zero
- [+,-]? … optional sign
- [1-9] … first number must not be zero (to avoid 000, 000.0)
- [0-9]* … other numbers
- .[0-9][0-9]* … at least one number after a dot

$$(PI \mid PII)(PIII)?$$

(PIII)? → decimal part optional



E1: 0+1
E2: 1+
E3: 1../1.+
E4: 1.0/1.0+
E5: .1

II: 0 (PI)
III: 1-9, 10, 11 (PII)
IV: 0.123, 10.23 (PIII)

Proč nejsou vnořené závorky regulární problém

$(a + (a + (a + a)))$ + $(a + a)$ ... 1 left, 1 right

3 open from left, 3 from right

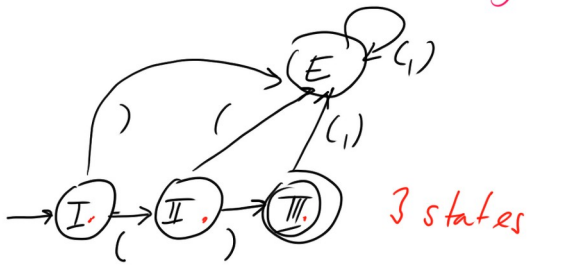
NON REGULAR

UNLIMITED Nr of brackets

→ LANGUAGE ONLY WITH BRACKETS:

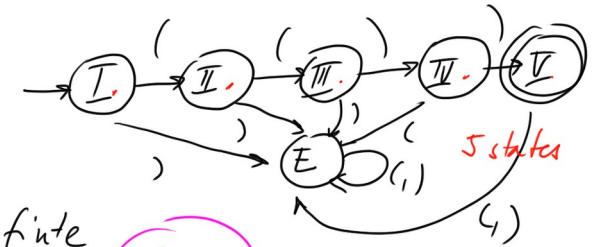
$(^n)^n \quad n \in (0, \infty)$

$n = 1$



3 states

$n = 2$



$n = k \Rightarrow$ states $k+2$ + error state

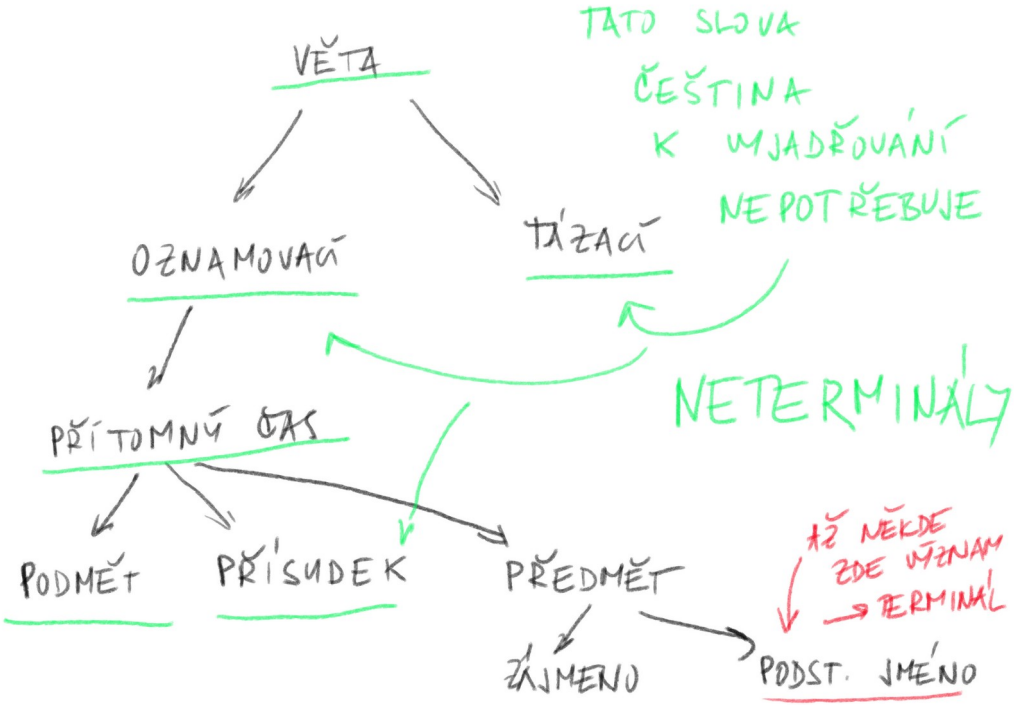
$n \to \infty \Rightarrow$ The automata with infinite

↳ $k+2+1 \to \infty$ → nr. of states!

Definice jazyka

- Pro programovací jazyky nemůžeme použít regulární výrazy.
- Potřebujeme gramatiku, která jazyk definuje. Ideálně bezkontextovou.

Co potřebujeme, abychom sestrojili gramatiku?



Co potřebujeme pro gramatiku?

- Pomocná slova, pomáhající popsat vazby v pravidlech - NETERMINÁLY (nikdy nevisí na konci větve ve stromu odvození)
- Skutečné symboly (slovní druhy a jejich tvary) - TERMINÁLY
- Popis vazeb mezi neterminály a cesty k řetězci terminálů - přepisovací pravidla
- Určení začátku - první neterminál s nejobecnějším výrazem

Teorie: Gramatika

Def: Gramatika je každá čtveřice G=(N, ∑, P, S), kde:

N je konečná množina **neterminálů** (*kategorie použité k popisu pravidel).*

∑ konečná množina **terminálů** (*abeceda, resp. "slovní druhy").*

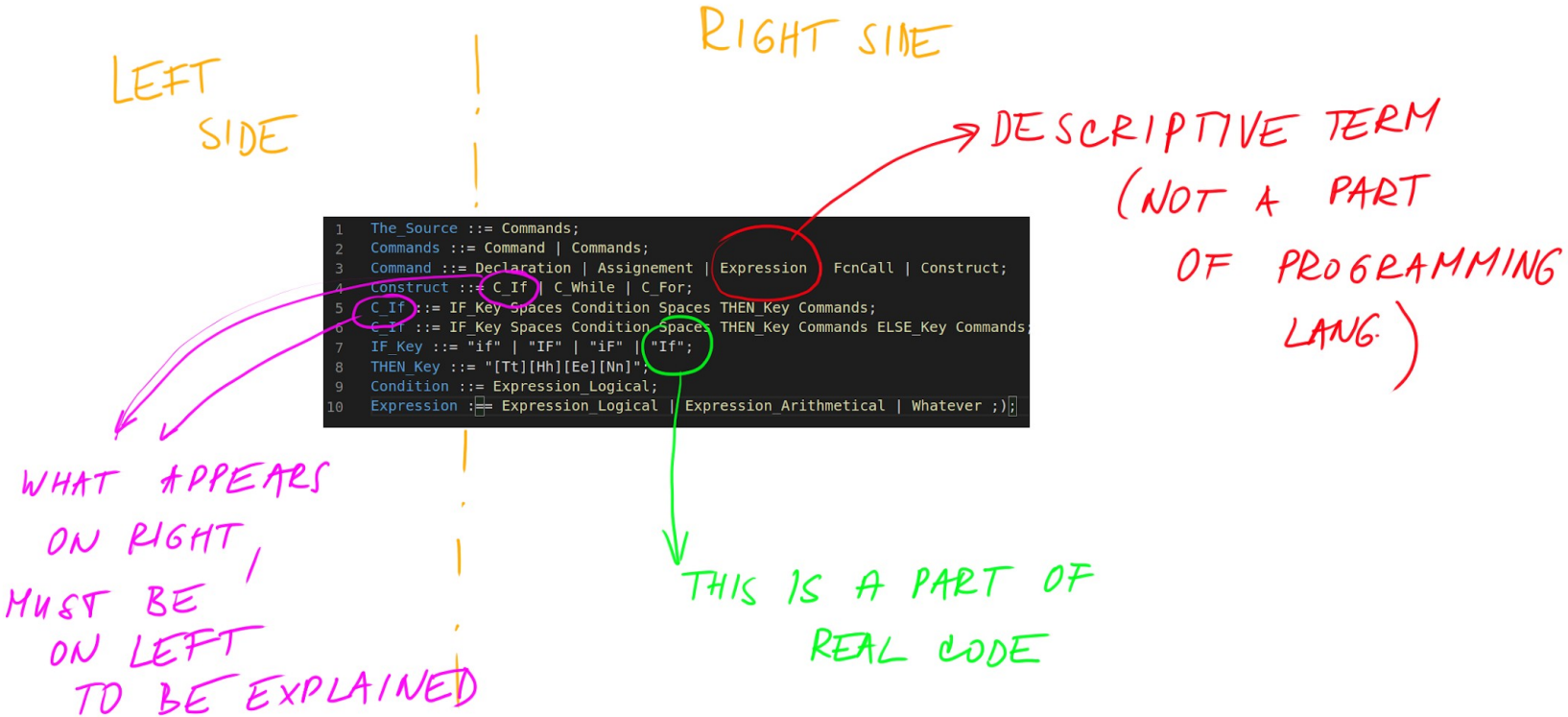
P je přepisovací systém *(pravidla popisující odvození řetězců terminálních symbolů z počátku).*

S je počáteční symbol, S ∈ N.

Gramatika programovacího jazyka (ukázka)

```
1   The_Source ::= Commands;
2   Commands ::= Command | Commands;
3   Command ::= Declaration | Assignement | Expression | FcnCall | Construct;
4   Construct ::= C_If | C_While | C_For;
5   C_If ::= IF_Key Spaces Condition Spaces THEN_Key Commands;
6   C_If ::= IF_Key Spaces Condition Spaces THEN_Key Commands ELSE_Key Commands;
7   IF_Key ::= "if" | "IF" | "iF" | "If";
8   THEN_Key ::= "[Tt][Hh][Ee][Nn]";
9   Condition ::= Expression_Logical;
10  Expression ::= Expression_Logical | Expression_Arithmetical | Whatever ;);
```

Bližší pohled



Zpět k teorii

∑ = {**if**, **IF, ;, while, end, {, }, ….**}

- Klíčová slova, operátory, interpunkce, zástupci hodnot a identifikátorů…

N = {Command; Expression; FcnCall...}

- Výrazy užité v popisu pravidel.

Levá strana pravidla

- Gramatiky programovacích jazyků se snažíme vytvářet jako **bezkontextové.**
  - Na levé straně pravidla **je vždy právě jeden neterminál.**
  - **Terminály** se mohou vyskytnout jen na pravé straně.
  - Tyto typy pravidel se dají konvertovat do **zásobníkových automatů** (syntaktické analyzátory).

Proč se učit o gramatikách

- Na počátku všech existujících programovacích jazyků byla gramatika.
- Když porozumíme gramatice, porozumíme jazyku.
- Už nikdy nebudete googlit, jak napsat tu z..ou binární konstantu v jazyce Java!
- Nebudete se ptát na StackOverfow, budete psát odpovědi.

Jak se gramatiky zapisují?

- Co potřebuji:
  - Oddělení levé a pravé strany (::=, ->, atd.)
  - Odlišení terminálů a neterminálů (tučné vs. normální písmo, uvozovky vs. bez uvozovek)
  - “Operátory” pro pravou stranu:
    - Označení pro nebo - umožní sloučit více pravidel v jedno.
    - Případně “regulární” syntax pro zjednodušení zápisu pravidla:
      - X smí být někde “právě jednou”, X smí být jednou nebo víc apod.

BNF a EBNF

- Klasické formy pro zápis gramatiky
- E=extended, regulární výrazy na pravé straně
- Standardizováno
- https://en.wikipedia.org/wiki/Extended_Backus%E2%80%93Naur_form

Pravidla lze zakreslit i diagramem (čeká nás ve cvičení)

- https://athena.ecs.csus.edu/~gordonvs/135/resources/05ebnfSyntaxDiagrams.pdf

slido

Jak se mi to dnes líbilo?

Click **Present with Slido** or install our Chrome extension to activate this poll while presenting.

slido

Kolik bylo nového?

Click **Present with Slido** or install our Chrome extension to activate this poll while presenting.

slido

Jak to bylo těžké?

Click **Present with Slido** or install our Chrome extension to activate this poll while presenting.