

# 4b. Makra Visual Basic pro Microsoft Excel

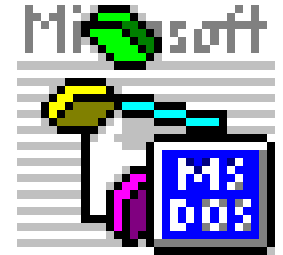


## Makra – funkce a metody

# Z historie



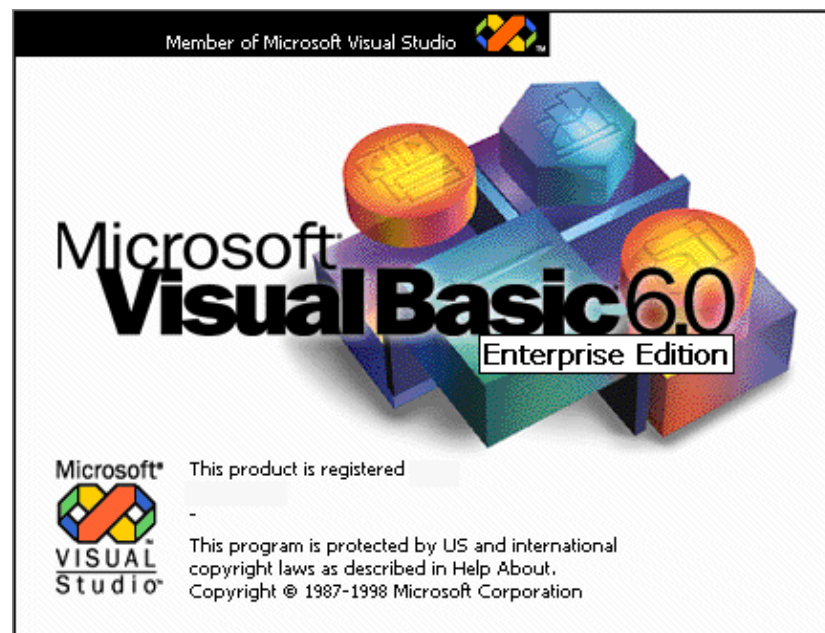
- Možnost napsat vlastní funkci/makro je v Excelu od první verze v roce 1985.
- Do roku 1993 (verze 5) byla makra zaznamenávána ve vlastním jazyce Excelu a ukládána jako soubory .xlm.
- Starší verze maker jsou zpětně kompatibilní, ale není doporučné jejich použití z hlediska bezpečnosti.
- Od verze 5 je možné makra zaznamenávat v jazyce Visual Basic.
- Visual Basic byl vyvinut v roce 1991 kombinací staršího jazyka Basic (1964) a prostředí Ruby společnosti Tripod.



# Visual Basic makro



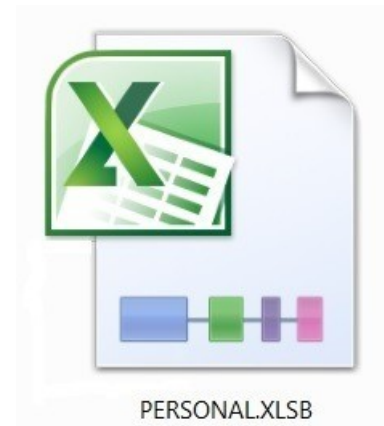
- Účelem maker v Excelu je buď usnadnění opakujících se činností nebo zpřístupnění složitějších funkcí, kterých není možné dosáhnout při rozumné složitosti ručně, případně kombinace obého.
- Pomocí maker lze rovněž vkládat do listů Excelu interaktivní prvky.
- „Všechno, co jde udělat ručně, lze udělat také pomocí makra.“
- Existují dva režimy zadávání maker – záznam přímo v prostředí Excelu a ruční zápis makra v jazyce Visual Basic.



# Uložení maker



- Makra lze ukládat jako součásti sešitů Excelu (v tom případě se mění přípona na *.xlsm*) nebo jako samostatné sešity maker.
- Každý uživatel má uložený na disku svého počítače nepřenositelný soubor *personal.xlsb*, do kterého může ukládat svá osobní makra – ta zůstávají k dispozici na daném počítači, ale ne jinde.
- Soubory *.xlsb* jsou zaznamenávány v jiném binárním kódu a umožňují rychlejší načítání (vhodné pro velké objemy dat). Lze do nich také ukládat makra.
- Makro uložené přímo v sešitě lze otevřít i na jiném počítači (obsahuje bezpečnostní riziko).



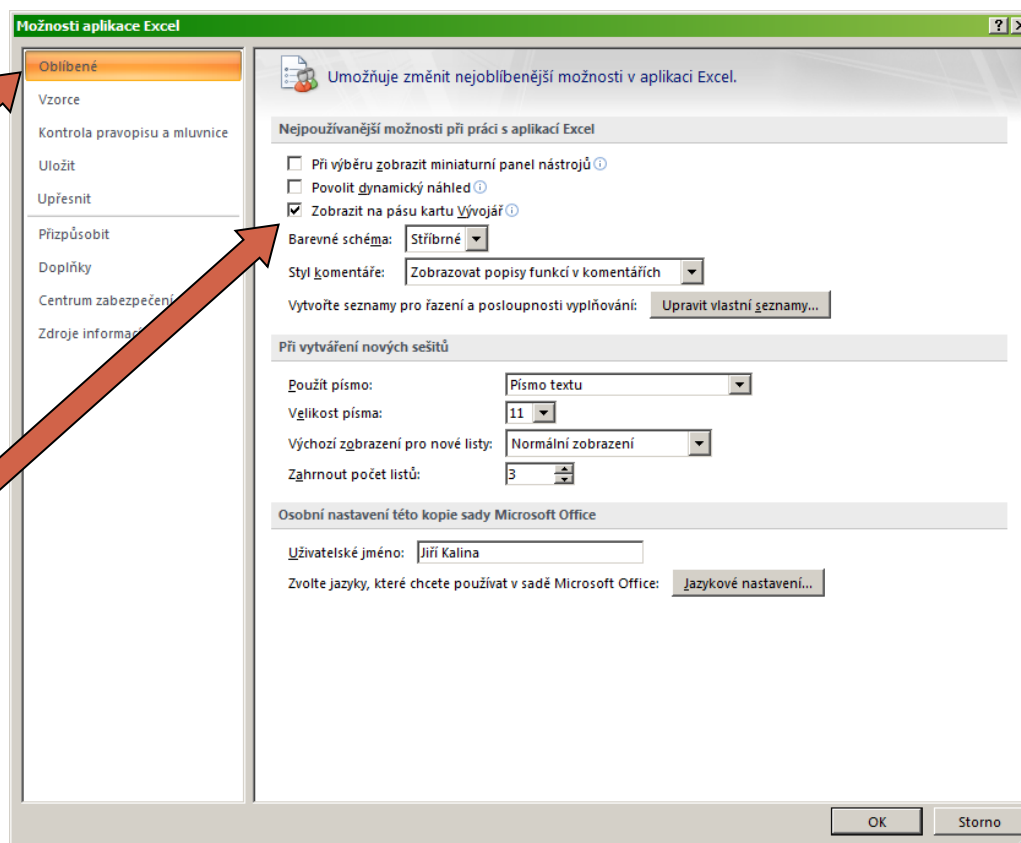
# Záznam makra



- Nejprve je nutné zpřístupnit v Excelu kartu Vývojář (od verze 2010):

Položka seznamu „Oblíbené“.

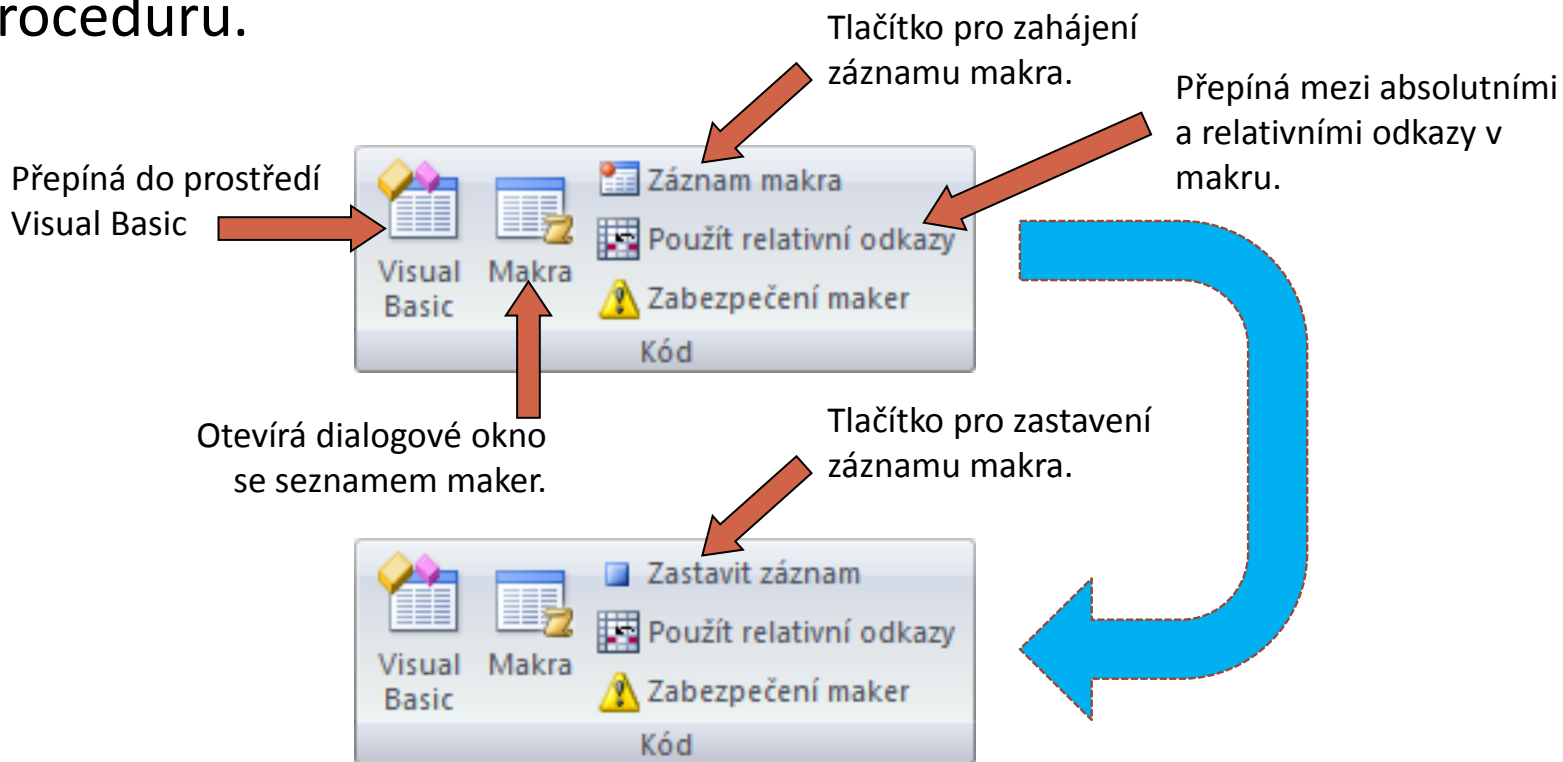
„Zobrazit na pásu kartu Vývojář“.



# Záznam makra



- Jednoduchý způsob vytvoření makra. K dispozici jsou pouze standardně přístupné funkce, ale lze je pomocí makra opakovat jako proceduru.



# Záznam makra



- Před spuštěním záznamu makra:

Uživatelský název makra.

Klávesová zkratka neodporující standardním zkratkám. Musí jít o písmeno nebo příbuzný znak. V případě kolize navrhuje Excel varianty Ctrl nebo Ctrl+Shift.

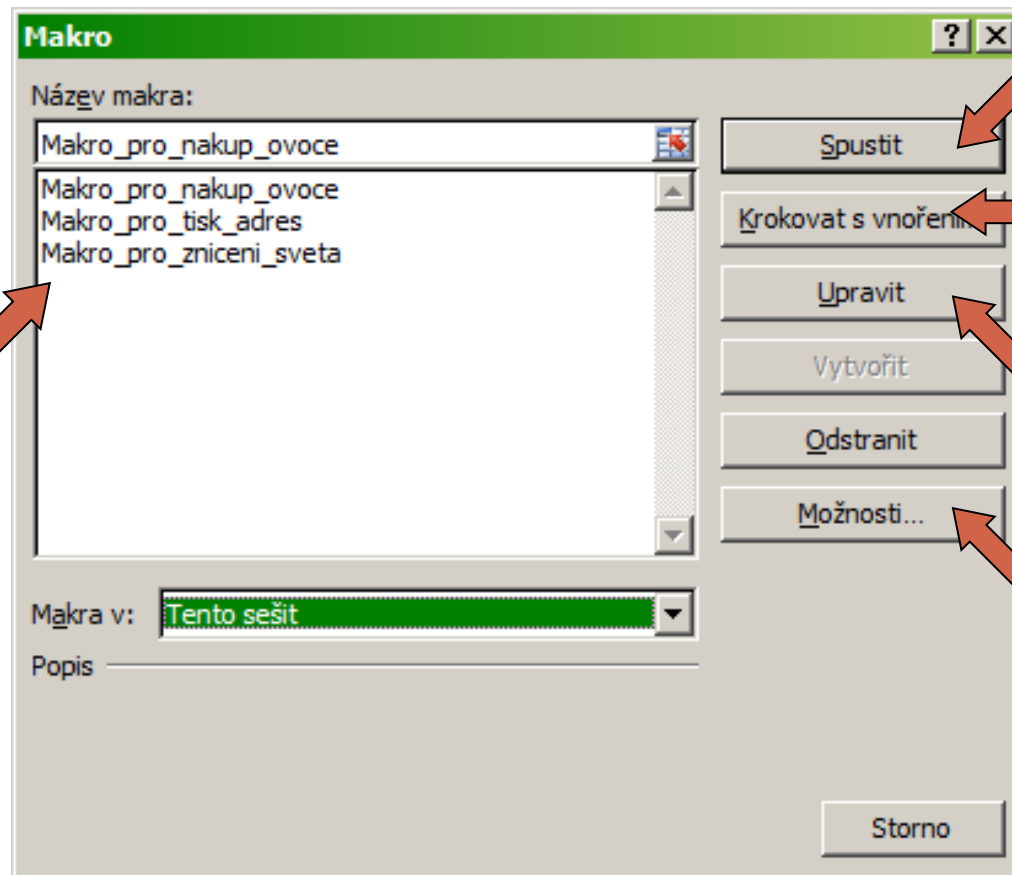
Místo pro uložení makra.

Volitelný popis makra.

# Záznam makra



- Okno pro spuštění maker:



Seznam vytvořených maker.

Spuštění vybraného makra.

Krokování makra v prostředí VB.

Úprav makra v prostředí VB.

Změna popisu a klávesové zkratky.



# Záznam makra



- Pokud je makro uloženo v osobní šabloně PERSONAL.XLSB nebo jiném souboru, je (v případě funkce) nutné jej v Excelu volat s názvem souboru před názvem vzorce:
- Např. =PERSONAL.XLSB!PRVOCISLO(A1)

# Spouštění makra



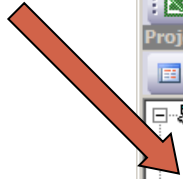
- Zobrazení karty Vývojář:  
***Soubor > Možnosti > Přizpůsobit pás karet***

# Visual Basic

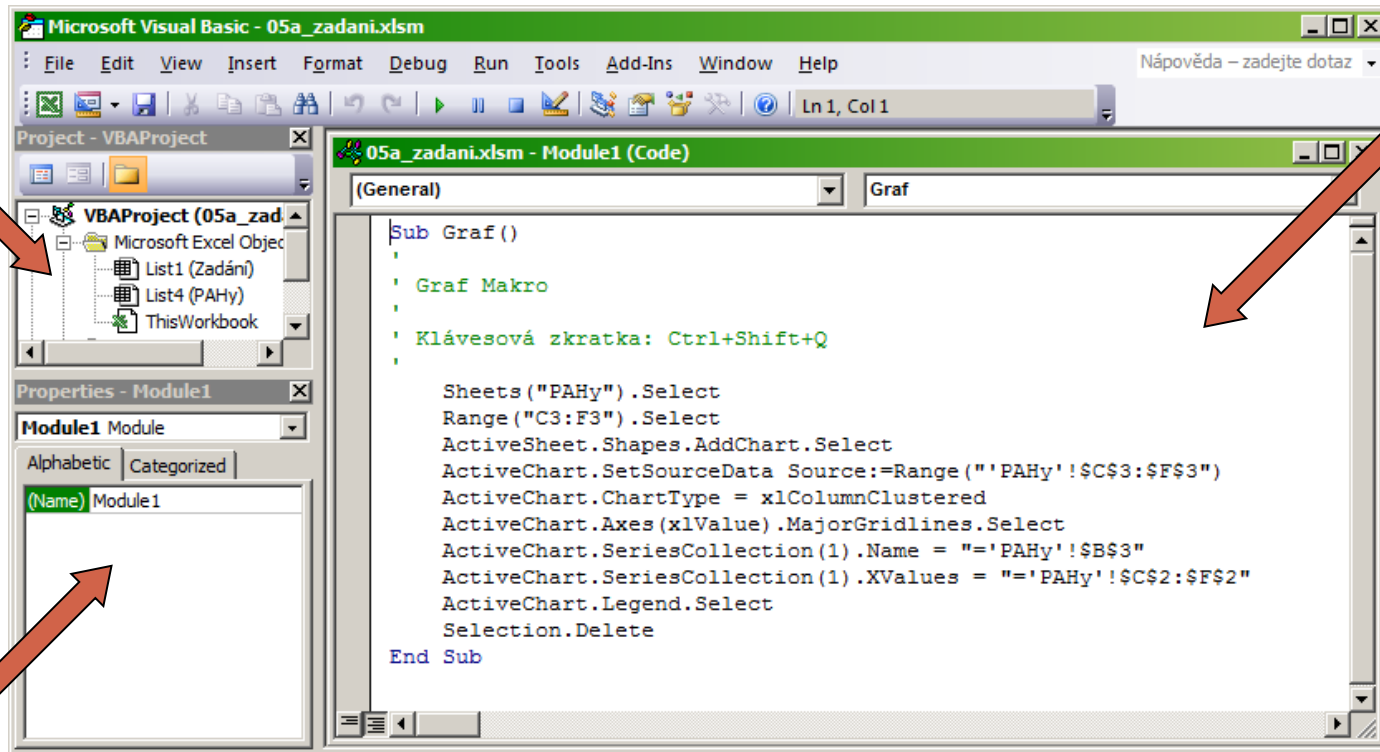
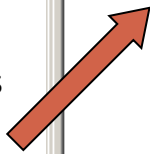


- Integrated development environment (IDE):

Project explorer



Properties window



Okno pro psaní kódu

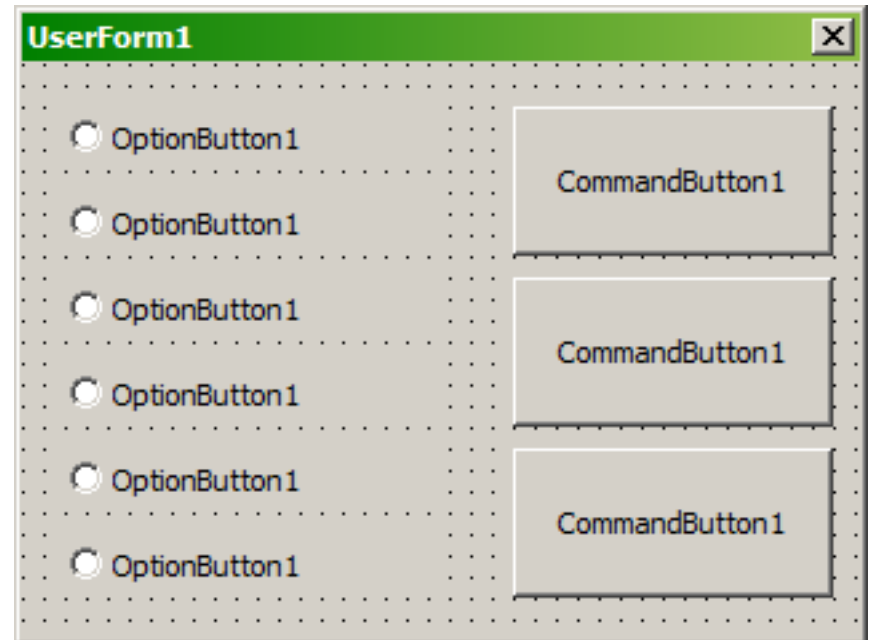


# Visual Basic



## Několik úvodních poznámek k jazyku Visual Basic

- jazyk není case sensitive (nerozlišuje malá a velká písmena),
- do kódu lze vepisovat komentáře uvozené apostrofem ' ,
- mezery a odsazení nemají vliv na interpretaci kódu,
- důležité je rozdělení řádků – jedna funkce na jeden řádek,
- více funkcí na řádku je možné spojit pomocí dvojtečky ; ,
- dlouhé řádky lze rozdělit pomocí kombinace ,\_ .



# Visual Basic



- Dvě základní entity, které lze vytvářet v prostředí Visual Basic jsou metody a funkce (+ objekty).
- Vytvořené funkce se automaticky přenáší do prostředí Excelu (konkrétního sešitu typu .xlsm, ke kterému je makro připojeno).
- Funkce se od metody liší tím, že má definovanou nějakou návratovou hodnotu.
- Makra nahraná pomocí záznamu maker v Excelu jsou automaticky považována za metody.
- Funkce i metody se zadávají jako zdrojový kód psaný uživatelem nebo generovaný programem do okna kódu a uvozují se speciálními výrazy.

# Visual Basic - funkce



- Každá funkce je uvozena a uzavřena specifickými příkazy:

**Function** **nazev\_funkce**(arg1, arg2,...) **As**  
**typ**

**tělo funkce**

**End Function**

- Tělo funkce se skládá z operací, v nichž jsou pro výpočet využity proměnné specifikované na vstupu do funkce (argumenty z 1. řádku funkce) a funkce jazyka Visual Basic.
- Návrátová hodnota funkce je určena přiřazením hodnoty do názvu funkce.

**nazev\_funkce = arg1 + arg2**

# Visual Basic - metody



- Každá metoda je uvozena a uzavřena specifickými příkazy:

**Sub** nazev\_metody(arg1, arg2,...)

**tělo metody**

**End Sub**

- Tělo metody se skládá z operací, v nichž jsou pro výpočet využity proměnné specifikované na vstupu do metody a funkce jazyka Visual Basic.

# Visual Basic



## Primitivní datové typy jazyka Visual Basic

| Jméno          | Popis                                | Velikost               | Rozsah  |
|----------------|--------------------------------------|------------------------|---|
| <b>Integer</b> | Celé číslo                           | 32 bitů                | $-2^{31}$ až $2^{31}$                                 |
| <b>Long</b>    | Celé číslo, ale větší rozsah         | 64 bitů                | $-2^{63}$ až $2^{63}$                                 |
| <b>Boolean</b> | Logická hodnota (pravda, nepravda)   | 8 bitů                 | <b>True</b> nebo <b>False</b>                         |
| <b>String</b>  | Textová hodnota                      | 16 bitů pro každý znak | ---   |
| <b>Char</b>    | Znak                                 | 16 bitů                | 0 až $2^{-16}$  |
| <b>Double</b>  | Desetinné číslo s dvojitou přesností | 64 bitů                | $\pm 5 \times 10^{-324}$ až $\pm 1,7 \times 10^{308}$ |



# Visual Basic



## Některé užitečné funkce jazyka Visual Basic

- ❖ **If** podmínka **Then** příkaz (blok příkazů) **End If** (v případě bloku),
- ❖ **While** podmínka příkaz (blok příkazů) **Wend**
- ❖ **For i = a To b** příkaz **Next** – for cyklus pro předem daný počet kroků,
- ❖ **Sheets("název listu").Select** – výběr označeného listu,
- ❖ **Range("buňka1:buňka2").Select** – výběr oblasti buněk,
- ❖ **Range(buňka1, buňka2).Select** – totéž zadáno číselně,
- ❖ **ActiveCell.Offset(radky,sloupce)** – přesun do zadané buňky
- ❖ **a Mod b** – zbytek po celočíselném dělení čísla a číslem b,
- ❖ **Sqr(a)** – druhá odmocnina z čísla a,
- ❖ **Ceiling(a)** – zaokrouhlení čísla nahoru.

# Visual Basic – objekty a vlastnosti



- Objektově orientované programování pracuje s objekty, které mají určité specifikované vlastnosti.
- Visual Basic považuje v Excelu za objekt celý soubor, list, buňku, graf, ovládací prvek (tlačítko, zatržítko, formulář aj.).
- V editoru IDE lze měnit vlastnosti objektů v okně Properties window; některé lze měnit také přímo v Excelu (např. pojmenování listu, vybarvení buňky) a také samotnými makry.
- Vlastnost objektu lze odkazovat přes tečku ..
- Např. nastavení barvy buňky A1 na červenou se provede následujícím příkazem:

```
Range("A1").Interior.Color = Red
```

# Visual Basic – události



- Kromě vlastností se k objektu pojí také konkrétní události, které mohou být impulzem pro aktivaci funkce nebo metody.
- Každý objekt má svoji specifickou sadu událostí, kterých jsou desítky.
- Důležité události mohou být např.:
  - ❖ **Activate** – aktivace sešitu (otevření uloženého souboru),
  - ❖ **SheetActivate** – aktivace požadovaného listu,
  - ❖ **Click** – kliknutí na ovládací prvek,
  - ❖ **Change** – změna hodnoty prvku,
  - ❖ **Show** – zviditelnění prvku,
  - ❖ **Hide** – zneviditelnění prvku.

# Visual Basic – další zdroje



- Visual Basic je plnohodnotný programovací jazyk, k jeho obsažení by nestačil ani celý předmět E7541,
- existuje celá řada elektronických i klasických učebnic ve všech jazycích,
- příjemnou učebnici lze nalézt např. zde:  
<https://www.gvp.cz/ucebnice/VisBas>,
- řada věcí je intuitivních a lze na ně přijít i bez odborného základu.

