Visual Studio Mastery

Be More Productive While Coding

SHORTCUTS AND PRODUCTIVITY TIPS

by Eng. Tod Vachev

1. Code Selection, Navigation and Files Navigation

ARROW KEYS (←↑↓→) – Moves your cursor through your code, left, right, up and down.

HOME – Moves your cursor to the beginning of your current row

END – Moves your cursor to the end of your current row

CTRL +] – If your cursor is currently placed on a bracket, it will be moved to its matching closing/opening bracket.

CTRL + ENTER – Inserts a new row above your current row and moves your cursor on that new row

SHIFT + ENTER – Inserts a new row bellow your current row and moves your cursor on that new row

CTRL + G – Moves your cursor to a specific line number

CTRL + UP KEY (↑) – Scrolls the file view up (just like the scroll wheel on your mouse)

CTRL + DOWN KEY (↓) – Scrolls the file view down (just like the scroll wheel on your mouse)

CTRL + LEFT RIGHT (← →) – Hold the **CONTROL** key modifier and press **LEFT** or **RIGHT** arrow keys to move the cursor faster throughout your code

CTRL + SHIFT + LEFT RIGHT (← →) – Hold the **CONTROL** and **SHIFT** key modifiers and press **LEFT** or **RIGHT** arrow keys to select whole words from your code

ALT + UP DOWN (↑↓) – Hold the **ALT** key modifier and press **UP** or **DOWN** arrow keys to move the selected row(s) of code up or down.

ALT + SHIFT + LEFT MOUSE BUTTON – Hold the **ALT** and **SHIFT** key modifiers and drag with your **LEFT MOUSE BUTTON** to select a specific part of your code spread on multiple rows

ALT + SHIFT + ARROW KEYS (←↑↓→) – Hold the **ALT** and **SHIFT** key modifiers and use the **ARROW KEYS** to move the cursor in your code to select a specific part of your code spread on multiple rows (more precise than using the mouse)

SHIFT + HOME – Selects everything on the left side of your cursor in your current row

SHIFT + END – Selects everything on the right side of your cursor in your current row

CTRL + W – Selects the current word in which your cursor is placed

CTRL + A – Selects everything in your current file

Finding code in files

F12 – Takes you to the actual implementation in the "home" file of the field, property, method or class on which your cursor is currently placed at

ALT + F12 – Opens a small temporary window with showing you the actual implementation of the field, property, method or class in which your cursor is currently placed at. Allows for quick modifications without switching files

CTRL + T, CTRL + , (CTRL and COMMA) – Opens a search bar, allowing you to quickly find a file, field, property, method, class etc. by writing its name (or using a partial name)

CTRL + ; – Moves your cursor in the search bar of the Solution Explorer window. allowing you to quickly find a file, field, property, method, class etc. by writing its name (or using a partial name)

Bookmarks

CTRL + K + K – Creates a new bookmark on the row on which your cursor is currently placed at

CTRL + K + W – Opens the bookmark window

CTRL + K + N – Moves you to the next bookmark

CTRL + K + P – Moves you to the previous bookmark

2. Code Manipulation

BACKSPACE – Deletes one character to the left of your cursor

DELETE – Deletes one character to the right of your cursor

BACKSPACE + CTRL – Deletes a whole word to the left of your cursor (will delete only a single character if there is a special symbol on the left side of the cursor, like a bracket for example)

DELETE + CTRL – Deletes a whole word to the right of your cursor (will delete only a single character if there is a special symbol on the left side of the cursor, like a bracket for example)

CTRL + C – Either select some piece of code and hold **CONTROL** key modifier and press **C** to copy the selected code, or just place your cursor on a given row of code without selecting anything and perform the same operation to copy the whole row

CTRL + V – Hold the **CONTROL** key modifier and press **V** to paste the copied text from the previous operation

CTRL + Z – Undo your last modification (can be used multiple consecutive times), "goes back in time"

CTRL + Y – Redo your last modification (can be used multiple consecutive times, after **CTRL+Z** was used multiple times), "goes forward in time". Keep in mind that after you do **CTRL + Z** and you perform a new modifaction, Redo will not be available since a new chain of commands is now created

Auto format:

CTRL + K + D – Hold the **CONTROL** key modifier and press **K** and then press **D** (do it fast) to auto format your code. This will correct the spacing within your code, but it will not add or remove empty rows where it is necessary!

CTRL + R + W – Toggles little dots in your code on, that will show you all the free spaces in your code, allowing you to easily spot any missing or additional spaces

TAB – Moves everything on the **RIGHT** side of the cursor one **TAB** (4 spaces) to the right. If multiple rows are selected, it will move all code on these rows one **TAB** to the right.

SHIFT + TAB – Moves everything on the **LEFT** side of the cursor one **TAB** (4 spaces) to the right. If multiple rows are selected, it will move all code on these rows one **TAB** to the **LEFT**.

Find and Replace

CTRL + R + R – Place your cursor within the variable that you want to rename, hold the **CONTROL** key modifier and press **R** and then press **R** again (do it fast), after that rename the variable and press **ENTER** to apply the changes

Camel Humps

Camel Humps let you quickly find what you are looking for by using only the first (capitalized) characters of its name. For example if you want to find the method **W**rite**L**ine you can only write Console.**WL** and the Camel Humps feature will find any members of the Console class that consist of two words, the first starting with **W** the second with **L**.

Minimize and Maximize

CTRL + M + M – Will minimize or maximize the current body of code in which your cursor is currently placed at

3. Commenting

CTRL + K + C – Hold the **CONTROL** key modifier and press and hold **K** then press **C** (do it fast) to comment the selected rows of code. If no rows are selected, only the row that has your cursor will be commented

CTRL + K + U - Hold the **CONTROL** key modifier and press and hold **K** then press **U** (do it fast) to uncomment the selected rows of code. If no rows are selected, only the row that has your cursor will be uncommented

Comment switch:

Alternating – Delete the first slash **/** from the first comment and then add it again, to quickly switch between the two versions of the code.

//*

... Some Code ...

... Some Code ...

... Some Code ...

/*/

... Some Code ...

... Some Code ...

... Some Code ...

//*/

Turn on Both – Delete the star in the middle between the two slashes to turn on both codes

//*

... Some Code ...

... Some Code ...

... Some Code ...

//

... Some Code ...

... Some Code ...

... Some Code ...

//*/

Single piece of code – use the starting and end parts of the trick to apply only on one piece of code to turn it on or off. Delete the slash on the first row or add it to turn it off or on.

//*

... Some Code ...

... Some Code ...

... Some Code ...

//*/

4. Visual Studio User Interface Management

SHIFT + ALT + ENTER – Enables distraction free mode, hides most of the windows, buttons and menus of Visual Studio and sets your code editor in Full Screen mode. Use the shortcut again to turn off distraction free mode

CTRL + ALT + PAGE UP/DOWN – Switch between opened files, **PAGE UP** go to the right file of your current file, **PAGE DOWN** goes to the left file. Works in Distraction Free Mode

CTRL + TAB – Opens a window that lets you select a specific file or window to switch your focus to. Move between the options with the arrow keys or your mouse

CTRL + F4 – Closes the currently viewed file

Hold and drag the currently opened file and release it on a given spot (like I show you in the course) to view two or more files at the same time (vertical/horizontal mode)
Note that **CTRL + ALT + PAGE UP/DOWN** will not switch between the split windows, it will cycle only through the files in your current split group. You can use **CTRL TAB** to switch to a file from the other group.

5. Building and Debugging Projects

Don't use **Console.ReadLine();** and **F5** to hold the console when you run a project! This is **very bad practice** promoted by developers that are not aware of the fact that there is an actual shortcut that lets you run your project in normal mode and automatically hold the console at the end.

Run your project with **CTRL + F5** instead, even the font and its size will be different from the F5 running!

CTRL + F5 – runs your project in "normal mode" without debugging

F5 – runs your project in "Debug Mode"

F9 – toggles a breakpoint on or off

CTRL + ALT + B – opens the breakpoint window

F10 – Step Over, available in DEBUGGING MODE, moves one line further in the code execution. If the line goes into a member of another class, the code in the other class will be executed and you will simply move to the next line, it will "Step Over" the underlying code.

F11 – Step Into, available in DEBUGGING MODE, will strictly follow your code execution line by line and will "Step Into" the underlying code if the next line has such code.

Right click on a breakpoint to add a condition

Condititions – Conditional breakpoints are activated only when a certain condition is met. For example if a given variable reaches a certain value, or if the breakpoint was hit a number of times.

Right click on a breakpoint to add an action

Actions – Actions Breakpoints can be used to log information in the Output window. Use curly brackets to get access to variables from the code.

Actions and Conditions **can be mixed,** to log information only when a certain condition is met.

6. Snippets

Snippets are a way to quickly write a predefined block of code that is used frequently.

The most trivial example is the **Console.WriteLine();**
Snippets have shortcuts and you activate them by pressing the **TAB** key twice.

To use the **Console.WriteLine();** snippet you have to write its shortcut – **cw** – and press **TAB** twice, and it will unwrap the underlying snippet code.

Some other useful and frequently used built in snippet shortcuts, use any of them and press **TAB** twice:

do, while, for, foreach

if, else

class, ctor, prop, propfull, propg

enum, interface, namespace

Anatomy of a Snippet

CTRL + K + B – Opens the Snippets Manager, from here you can import your newly created snippets

Snippets exist in a "tag hierarchy" since we are creating them in **XML**.

<CodeSnippets> - all snippets exist under this tag

<CodeSnippet> - is a tag for an individual snippet and it is under **<CodeSnippets>**

A snippet has two main tags

<Header> - contains information about the snippet and its shortcut

- **<Title>** - Title of the Snippet
- **<Shortcut>** - Shortcut of the Snippet
- **<Author>** - The Author of the Snippet
- **<Description>** - Description of the Snippet
- **<HelpURL>** - A Helpful URL with a lengthy explanation about how to use the snippet(s)

<Snippet> - contains the actual code of the snippet as well as snippets or references that need to be added

- **<Imports><Import><Namespace>**Your.Namespace.Here**</Namespace></Import></Imports>** - Used to Import a Namespace along with the creation of the Snippet
- **<Declarations>** - Used to contain the Snippet Variables
    - **<Literal>** - Literals are the Variables of a Snippet
        - **<ID>** - The "name" of the Snippet that will be used to reference it
        - **<Tooltip>** - Short Information to be displayed about the literal (optional)
        - **<Default>** - Default Value for the Snippet Variable
        - To use a Literal within your Snippet Code, you must use its id surrounded with **$**, like this **$id$**. Refer to the Snippet Example bellow
- **<Code><!CDATA[***WRITE YOUR CODE HERE***]]>** This is where the actual code of the Snippet lives, between the square brackets of CDATA inside the Code tag.
- **Escaping the $** symbol is done by using two **$$** consecutively. That is needed if you want to actually have the **$** symbol in your code, for string interpolation for example. Refer to the snippet example bellow
- **end** - this is a reserved tag, it designates where your cursor will end up at after you are done with the snippet and press **ENTER**.

Snippet Example – You can download the full **.snippet** file with all examples (and practically useful and ready to be used snippets) from the **Source Lecture in the Snippets section of the course**:

```
<CodeSnippet Format="1.0.0">
  <Header>
    <Title>Console Write Line Snippet</Title>
    <Shortcut>cwl</Shortcut>
    <Description>Outputs Console.WriteLine(); in the code editor</Description>
  </Header>
  <Snippet>
    <Imports>
      <Import>
        <Namespace>System</Namespace>
      </Import>
    </Imports>
  <Declarations>
      <Literal>
        <ID>text</ID>
        <ToolTip>The text that will be output on the console</ToolTip>
        <Default>INPUT YOUR TEXT HERE</Default>
      </Literal>
    </Declarations>
    <Code Language="csharp">
      <![CDATA[Console.WriteLine($$"$text$");
      $end$]]>
    </Code>
  </Snippet>
</CodeSnippet>
```

Snippets Template – You can **download** this Template as a **.snippet file from the source lecture in the Snippets section of the course,** so that you don't have to write it out:

```
<?xml version="1.0" encoding="utf-8"?>
<CodeSnippets xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
  <CodeSnippet Format="1.0.0">
    <Header>
      <!-- Title of the Snippet -->
      <Title></Title>
      <!-- Shortcut to invoke the snippet -->
      <Shortcut></Shortcut>
      <!-- Description of the Snippet -->
      <Description></Description>
    </Header>
    <Snippet>
      <Imports>
        <Import>
          <!-- Import namespaces needed for the code snippet -->
          <Namespace></Namespace>
        </Import>
      </Imports>
      <Declarations>
      <!-- Literals are parts in your code bellow, that you would want to change after
invoking the snippet-->
        <Literal>
          <!-- The name of the literal in the code bellow between the $ $-->
          <ID></ID>
          <!-- Displayed info about that literal -->
          <ToolTip></ToolTip>
          <!-- Default value for the literal -->
          <Default></Default>
        </Literal>
      </Declarations>
      <Code Language="csharp">
        <![CDATA[]]> <!-- Write your code between the [] of CDATA-->
      </Code>
    </Snippet>
  </CodeSnippet>
</CodeSnippets>
```