

Day 4

How to be a Good Programmer!

Programmer Tips

Optional, but these can improve your productivity!

Keyboard Shortcuts and Text Editors

Keyboard Shortcuts

- **CTRL + F**
- **CTRL + LEFT/RIGHT**
 - Skips to the beginning of the word
 - Useful for deleting whole word or skipping through words quickly
- **CTRL + SHIFT + LEFT/RIGHT**
- **CTRL + SHIFT + UP/DOWN**
- **CTRL + BACKSPACE**
- **CTRL + A**
 - Select all words in the file
- **WINDOWS + E**
 - Open Windows file explorer

Sublime Keyboard Shortcuts

- **CTRL + B**
- **CTRL + X**
 - Cut line
- **CTRL + L**
 - Select entire line
 - Repeat to select multiple lines
- **CTRL + D**
 - Select word and display all other occurrences of word
- **CTRL + K**
 - Delete line
- **CTRL + ALT + UP/DOWN**
 - Move line up or down
- **ALT + SHIFT + 2**

Text Editors

- **Notepad++**
 - Difficulty: low
 - Extensibility: low



```
1 Main.sublime-menu < Package Control.py < Default.sublime-commands < example-packages.json <
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
```

- Atom
 - Difficulty: low
 - Extensibility: medium

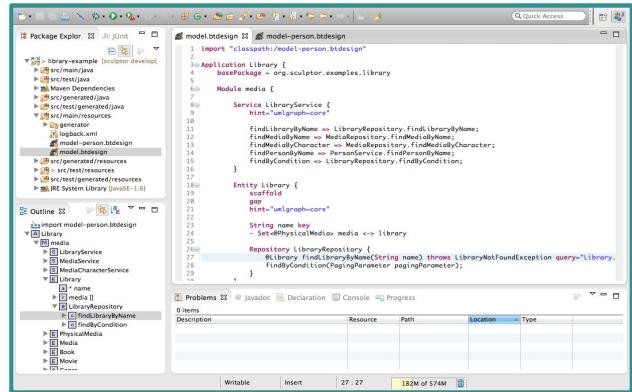
```
D:\source\notepad+->trunk\PowerEditor\bin\shortcuts.xml - Notepad+ -  
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?  
  
Notepad_plus_Windows.h Notepad_plus.h Notepad_plus.cpp CMakelists.txt Notepad_Plugin.cpp HelpBigSwitch.cpp  
Notepad_Plugin.h Notepad_Plugin.h Notepad_Plugin.cpp HelpNotification.cpp  
  
<NotepadPlus>  
  <InternalCommands />  
  <Macros>  
    <Macro name="TwinReadAndSave" Ctrl="no" Alt="yes" Shift="yes" Key="83">  
      <Action type="2" message="0" wParam="42024" lParam="0" sParam=""/>  
      <Action type="2" message="0" wParam="41006" lParam="0" sParam=""/>  
    </Macro>  
  </Macros>  
  <ScriptedCommands>  
    <Command name="Launch in Firefox" Ctrl="no" Alt="yes" Shift="yes" Key="88">firefox &quot;$ (FULL_CURRENT_PAT  
    <Command name="Launch in IE" Ctrl="no" Alt="yes" Shift="yes" Key="13">explorer &quot;$ (FULL_CURRENT_PAT  
    <Command name="Launch in Chrome" Ctrl="no" Alt="yes" Shift="yes" Key="82">chrome &quot;$ (FULL_CURRENT_PAT  
    <Command name="Launch in Safari" Ctrl="no" Alt="yes" Shift="yes" Key="70">safari &quot;$ (FULL_CURRENT_PAT  
    <Command name="Open php file" Ctrl="no" Alt="yes" Shift="no" Key="112">http://www.phpedit.net/index.php?file=$ (CURRENT_FILE)  
    <Command name="Open http://www.google.com/search?query=$ (CURRENT_QUERY)" Ctrl="no" Alt="yes" Shift="no" Key="114">http://www.google.com/search?query=$ (CURRENT_QUERY)  
    <Command name="Wikipedia Search" Ctrl="no" Alt="yes" Shift="no" Key="116">http://en.wikipedia.org/wikis  
    <Command name="Open file" Ctrl="no" Alt="yes" Shift="no" Key="118">$(NPP_DIRECTORY).notepad+.exe $ (CURRENT_FILE)  
    <Command name="Open in another instance" Ctrl="no" Alt="yes" Shift="no" Key="117">$(NPP_DIRECTORY).notepad+ $(CURRENT_FILE)  
    <Command name="Open current dir" Ctrl="no" Alt="no" Shift="no" Key="70">cmd /k cd $ (CURRENT_DIRECTORY)  
    <Command name="Send via Outlook" Ctrl="yes" Alt="yes" Shift="yes" Key="79">outlook &quot;$ (FULL_CURRENT_PAT  
  </ScriptedCommands>  
  <Plugins/>  
  <HelpBigSwitch />  
</NotepadPlus>
```

- **Sublime Text**
 - Difficulty: low
 - Extensibility: low
 - Beauty: high

Text Editors

- **Eclipse**
 - Specialty: Java
 - Extensibility: medium

- **Vim or Emacs**
 - Difficulty: high
 - Extensibility: extremely high
 - Flexibility: extremely high



- **Visual Studio**
 - Difficulty: medium
 - Extensibility: medium
 - Specialty: Large Projects

**Saving Backups
and
Comments
and
Refactoring
and
Pseudocode**

Saving backups / versions

The screenshot shows a list of GitHub commits from a repository. Each commit is represented by a card with a title, author, date, and a green checkmark indicating success. To the right of each commit are three icons: a clipboard, a copy button with the commit's SHA-1 hash, and a diff icon.

- add upload link
leoncheng57 committed 28 days ago ✓
f3c43bf
- updated circle
Charleen Wang committed 28 days ago ✓
3c4dbe3
- added oval and circle
Charleen Wang committed 28 days ago ✓
145a8a3
- css guide
Charleen Wang committed 28 days ago ✓
3d797c4

Commits on Jul 2, 2018

- move files
leoncheng57 committed on Jul 2 ✓
48e2ebb
- add recipe, advertisement example links
leoncheng57 committed on Jul 2 ✓
1df3799

- Saves your butt if your program crashes
- Try to use descriptive names

Writing Comments

```
1 class Sprite:
2     name = "default_name"
3     hometown = "default_hometown"
4
5     # Make sure to use 'self' as a parameter
6     # And use it when calling class variables
7     def get_description(self):
8         result = ""
9         result += "Name: " + self.name + "\n"
10        result += "Hometown: " + self.hometown + "\n"
11        return result
12
13    # Different from 'self' class vars and input parameter
14    def set_name(self, name):
15        self.name = name
16
17    def set_hometown(self, hometown):
18        if (hometown == "NYC"):
19            hometown = "NYC (da best)"
20        self.hometown = hometown
21
22    # Without 'self' parameter, it is a Static class function
23    def generic_greeting():
24        return "Hi, how are you?"
25
```

- Easier to understand for teammates
- Easier to understand for future you

Refactoring / Don't Repeat Code (DRY)

```
def drive_left():

    if (gas not in engine):
        return "Error"
    start_engine()
    if (people_in_front == True):
        return "Error"
    step_on_gas()

    rotate_wheel_to_left()
    continue_driving()
    return "Success"

def drive_right():

    if (gas not in engine):
        return "Error"
    start_engine()
    if (people_in_front == True):
        return "Error"
    step_on_gas()

    rotate_wheel_to_right()
    continue_driving()
    return "Success"
```

```
def ready_to_drive():
    if (gas not in engine):
        return "Error"
    start_engine()
    if (people_in_front == True):
        return "Error"
    step_on_gas()

def drive_left():
    ready_to_drive()

    rotate_wheel_to_left()
    continue_driving()
    return "Success"

def drive_right():
    ready_to_drive()

    rotate_wheel_to_right()
    continue_driving()
    return "Success"
```

- Shorter, more readable code
- Easier to write future code
- Fewer mistakes in the future if changes made

Pseudocode

algorithm/code but in words

Power of 2:

Read n.

Initialize power to 1.

Repeat n times:

 Double power.

return power



```
def power_of_two(n):  
    # Initialize power to 1.  
    power = 1  
    # Repeat n times:  
    for i in range(n):  
        # Double power  
        power *= 2  
    # return power  
    return power
```

**Naming
and
Documentation
and
Testing**

Naming

```
class JuniorRegister:

    student_list = ['Charleen', 'Leon', 'Chris']
    MAX_CLASS_SIZE = 200

    def __init__(self, x, y):
        self._x = x
        self._y = y

    def this_is_my_function(self):
```

- Variables, functions, methods, packages, modules
Lower_case_with_underscores
- Classes and Exceptions
CamelCase
- Private methods
__double_leading_underscore(self, ...)
- Constants
ALL_CAPS_WITH_UNDERSCORES

General Naming Guidelines

- Use intention-revealing names
 - functions, variables, classes and everything
 - For example:
 - number instead of nu, num, n
 - Product instead of pro, p etc

```
class Algebra:  
  
    number =  
    operation =  
  
    def my_product(numbers):  
        product =
```

General Naming Guidelines

- Use intention-revealing names
- Avoid one-letter variables (esp. l, O, I)
 - will be confused with 1 or o
 - makes your life difficult

General Naming Guidelines

- Use intention-revealing names
- Avoid one-letter variables (esp. l, O, I)
- Avoid redundant labeling

```
import audio
core = audio.Core()
controller = audio.Controller()
```

YES

```
import audio
core = audio.AudioCore()
controller = audio.AudioController()
```

NO

General Naming Guidelines

- Use intention-revealing names
- Avoid one-letter variables (esp. l, O, I)
- Avoid redundant labeling
- Prefer "reverse notation"

```
elements = ...
elements_active = ...
elements_defunct = ...
```

YES

```
elements = ...
active_elements = ...
defunct_elements = ...
```

NO

Documentation

```
def my_product(numbers):
    """ Compute the product of a sequence of numbers.

    Parameters
    -----
    numbers : sequence
        list of numbers to multiply

    Returns
    -----
    product : number
        the final product

    Raises
    -----
    TypeError
        if argument is not a sequence or sequence contains
        types that can't be multiplied
    """

```

Documentation

- Not only for others, but also for yourself !
- Minimum requirement: at least a docstring
- Document arguments and return objects, including types
- For complex algorithms, document every line
- For big programs/projects provide an how-to or quick-start your website

Testing & Debugging

- Testing increases the confidence that your code works correctly, not only for yourself but also for your reviewers
- Tests are the only way to trust your code!
- It might take you a while to get used to writing them
- Unit case: The smallest testable piece of code

Unit Tests

- **Unit Test:** The smallest testable piece of code
- Why?
 - i. Easier to test the whole program, if the units work
 - ii. Can modify parts, and be sure the rest still works
 - iii. Provides examples of how to use code

Unit Tests Example

```
import unittest

def my_product ( numbers ):
    """ Compute the product of a sequence of numbers . """
    total = 1
    for item in numbers :
        total *= item
    return total

class TestUM(unittest.TestCase):

    def test_numbers_3_4(self):
        self.assertEqual( my_product([3,4]), 12)

    def test_strings_a_3(self):
        self.assertEqual( my_product([2,3,1]), 6)

if __name__ == '__main__':
    unittest.main()
```

Unit Tests Example

```
76
77     import unittest
78
79     def my_product ( numbers ):
80         """ Compute the product of a sequence of numbers . """
81         total = 1
82         for item in numbers :
83             total *= item
84         return total
85
86     class TestUM(unittest.TestCase):
87
88         def test_numbers_3_4(self):
89             self.assertEqual( my_product([3,4]), 12)
90
91         def test_strings_a_3(self):
92             self.assertEqual( my_product(['2','3','1']), 6)
93
94         ..
95
96         -----
97
98     Ran 2 tests in 0.000s
99
100    OK
101    [Finished in 0.2s]
```

Important...

- Testing all functions and adherence to specifications
- Finding all edge cases
- Testing false arguments e.g. passing in a string when should only integers allowed

```
def test_numbers_3_4(self):  
    self.assertEqual( my_product([3,"aa"]), "Input type error!")
```

Resource for Python Conventions

PEP 8 -- Style Guide for Python Code

PEP:	8
Title:	Style Guide for Python Code
Author:	Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>
Status:	Active
Type:	Process
Created:	05-Jul-2001
Post-History:	05-Jul-2001, 01-Aug-2013

Programmer Tips

- Use **terminal** for running and managing files
- Use **GitHub** for teamwork and version control (saving backups)
- **Keyboard shortcuts** for easier coding
- Customize your **text editor** for more comfortable coding
- Write out **pseudocode** to extract your ideas before starting to code
- Write **comments** to make your teammates and future you understand
- **Refactoring** or **DRY** so that code is reused
- **Name** variables, **document** methods, and **test** your code

Effective Presentation Tips and Tricks

Why do we
care about
presentation
skills?

Story Time :)

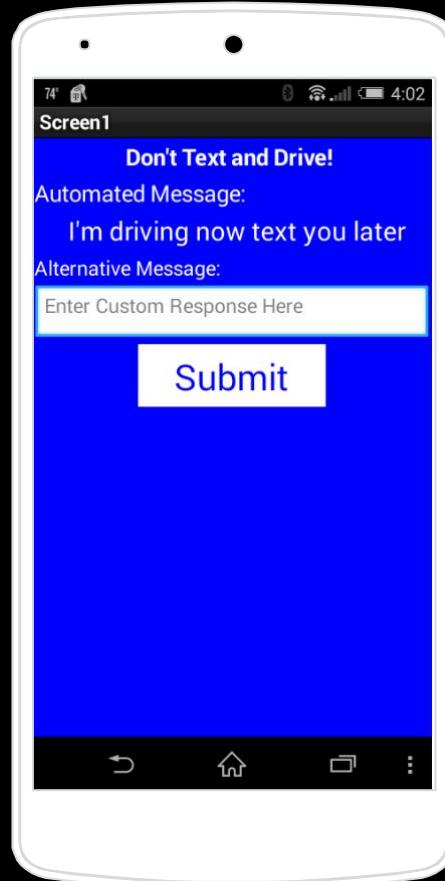
TextBuddy

1. read out loud the received text
2. send an automated message back
3. customize automated message

Ugly

Minimal function

Has purpose



relevance

49% of drivers under the age of 35 text and drive



60% of all drivers use their phones while they drive

every year, 1.6 million car accidents are cell phone related



These accidents cause about a half million injuries and take 6,000 lives

key stats

people who text and drive are driving outside of their lanes 10% of the time



they are also 3X more likely to crash

Images and diagrams

Persuasion



Customer Loyalty

A bad mission statement:

To sell the best computers and make lots of money!

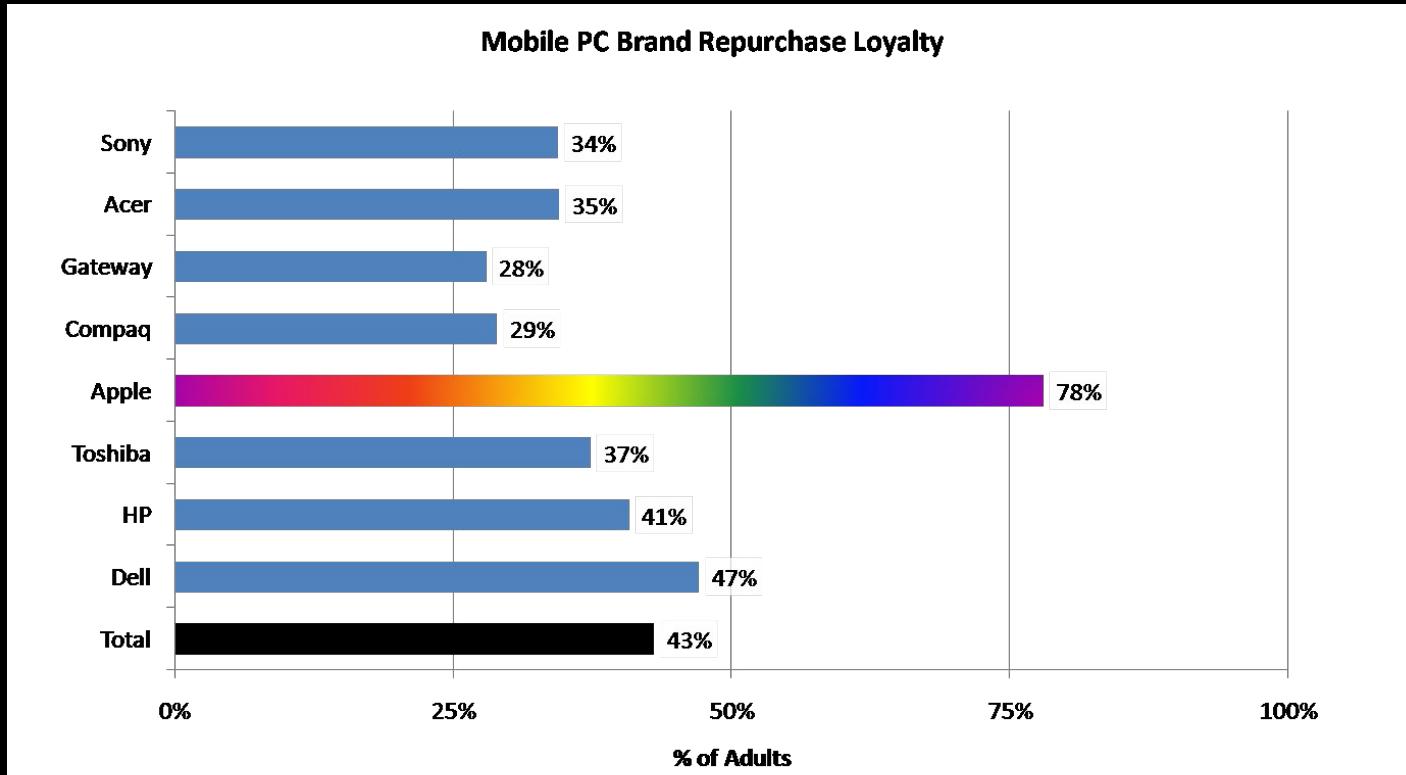
Steve Jobs:

"To make a contribution to the world by making tools for the mind that advance humankind."

Dell:

none...

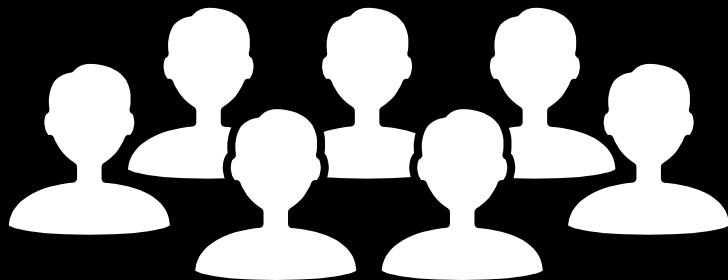
Apple Brand Retention



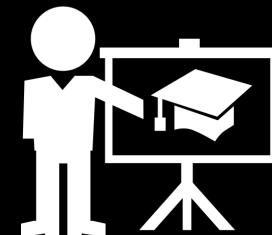


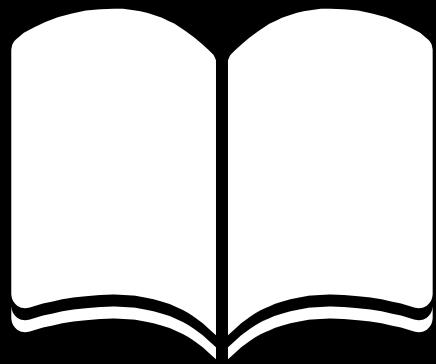
PURPOSE

**WHY are you presenting?
WHY am I listening to you?
WHY is this important?**

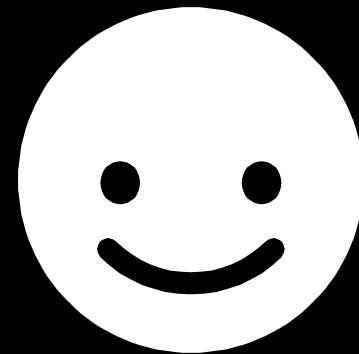


AUDIENCE





education



entertainment

Tell a story

Once upon a time...

- hook your audience
- describe the problem
- introduce your idea
- argue why it's great
- concluding thoughts

The end!

Climax

The peak of the story.

Example:

- Joe gets fired from his job.

Rising Action

Series of events that raises the tension of the plot.

Example:

- Joe makes a mistake at work.
- Joe tries to cover it up.
- Joe is caught by his boss.
- Joe tries to lie.

Falling Action

The summarizing events after the peak.

Example:

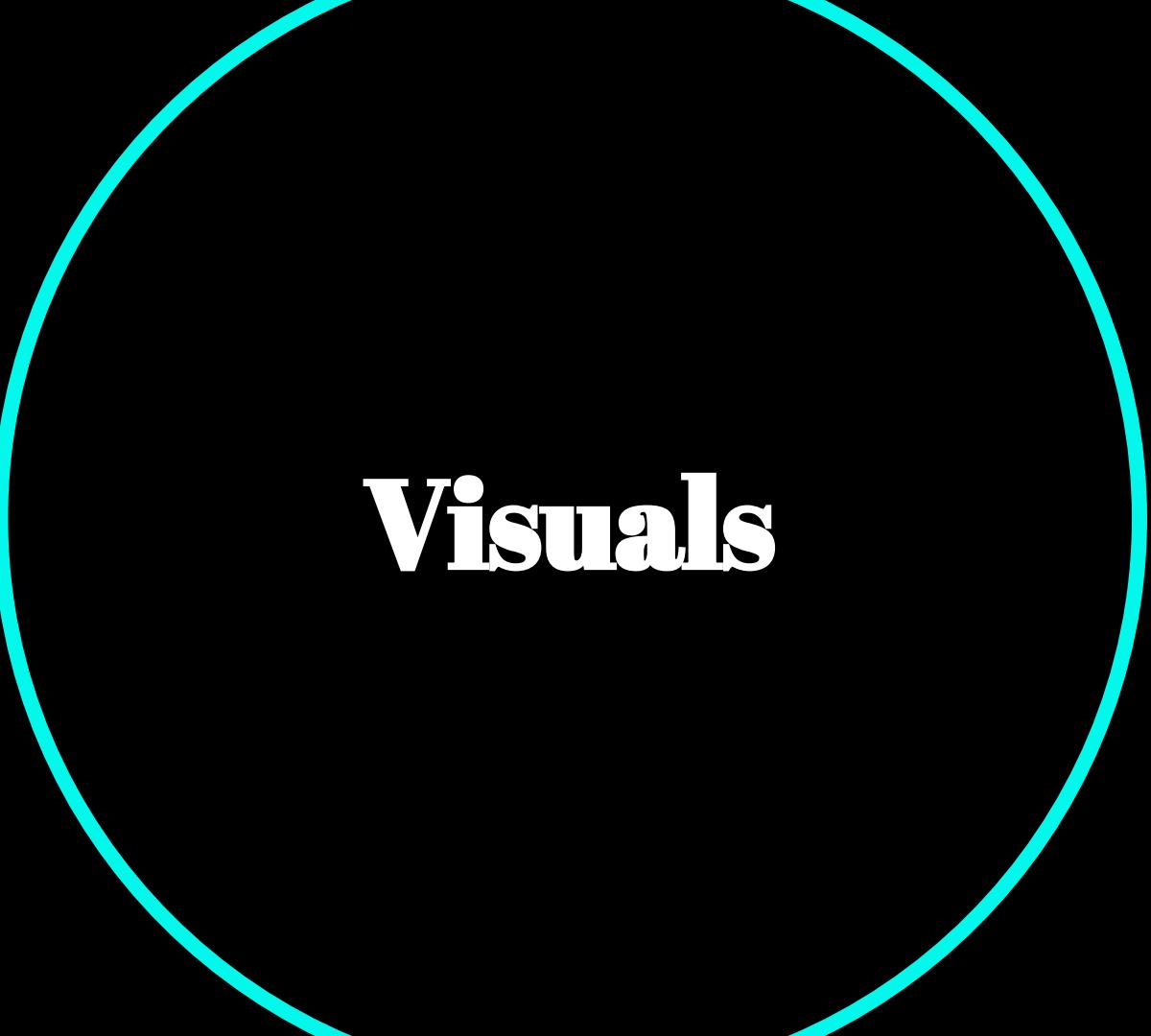
- Joe upset, drinks.
- Joe re-thinks his life.
- Joe decides to pursue his dreams.

Beginning

The Initial events that set the story in motion.

Example:

- Joe goes into work today, like any other day.



Visuals

Color Usage

Colors: use the same colors consistently

Images: choose images that are similar in style

Fonts: use three or less fonts, use them the same way (Headings/body text)

Color Usage

Colors: use the same colors consistently

Images: choose images that are similar in style

Fonts: use three or less fonts, use them the same way (Headings/body text)

Color Usage

Colors: use the **same** colors consistently

Images: choose images that are **similar** in style

Fonts: use **three** or less fonts, use them the same way (Headings/body text)

Color Usage

Colors: use the same colors consistently

Images: choose images that are similar in style

Fonts: use three or less fonts, use them the same way (Headings/body text)

Consistency/Coherence

Colors: use the same colors consistently

Images: choose images that are similar in style

Fonts: use three or less fonts, use them the same way (Headings/body text)

PURPOSE

WHY are you presenting?
WHY am I listening to you?
WHY is this important?



AUDIENCE

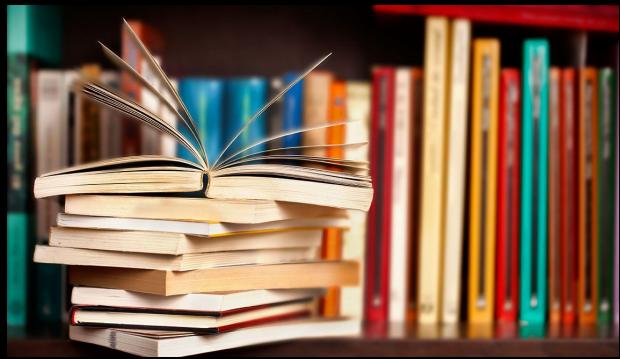


Consistency/Coherence

Colors: use the same colors consistently

Images: choose images that are similar in style

Fonts: use three or less fonts, use them the same way (Headings/body text)

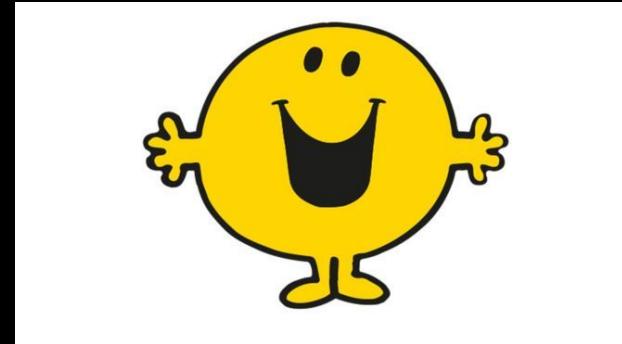
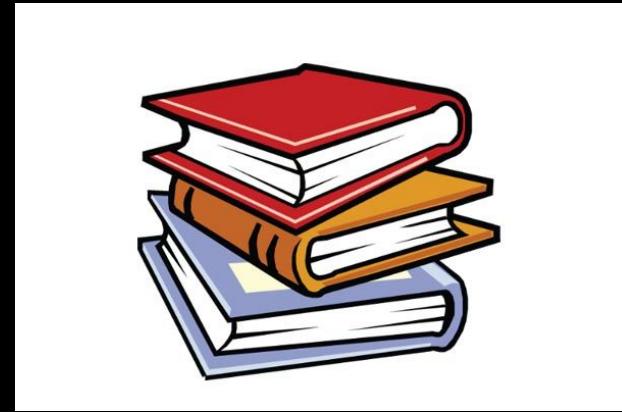


Consistency/Coherence

Colors: use the same colors consistently

Images: choose images that are similar in style

Fonts: use three or less fonts, use them the same way (Headings/body text)

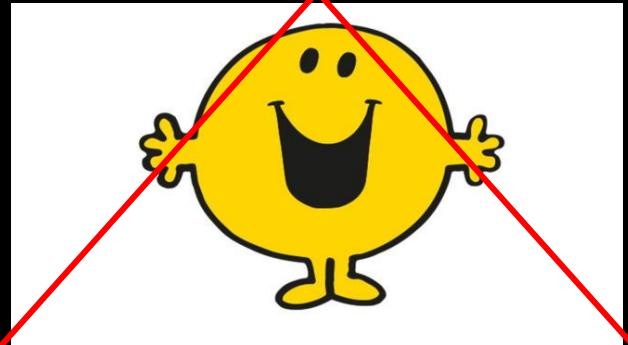
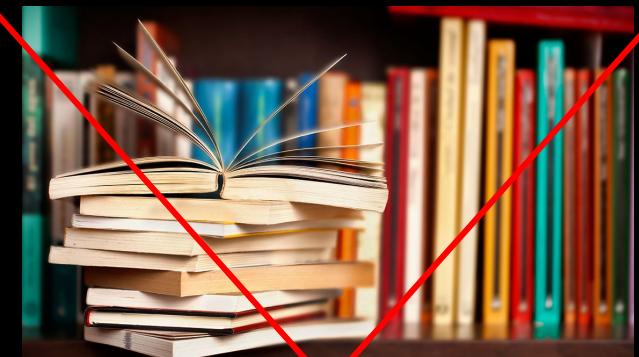


Consistency/Coherence

Colors: use the same colors consistently

Images: choose images that are similar in style

Fonts: use three or less fonts, use them the same way (Headings/body text)

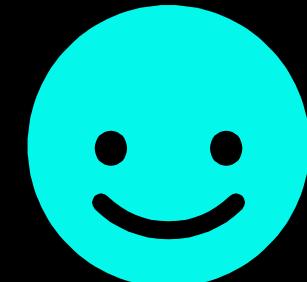
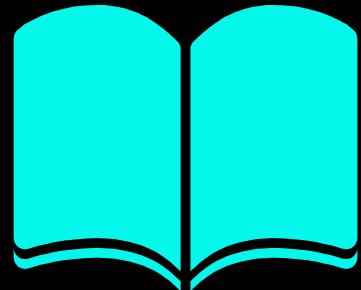


Consistency/Coherence

Colors: use the same colors consistently

Images: choose images that are similar in style

Fonts: use three or less fonts, use them the same way (Headings/body text)

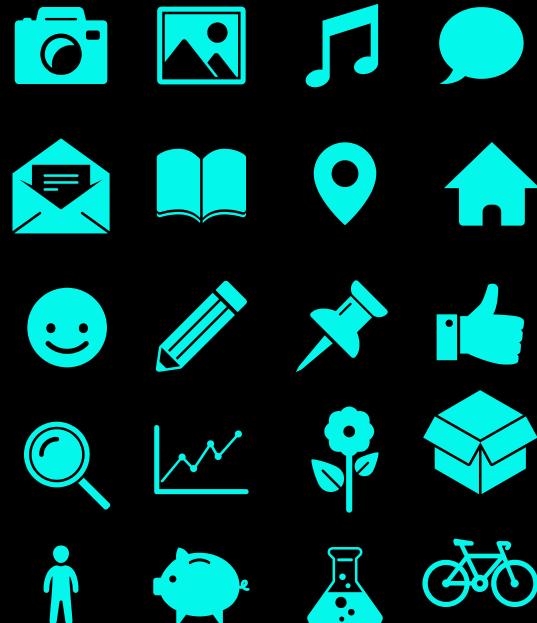


Consistency/Coherence

Colors: use the same colors consistently

Images: choose images that are similar in style

Fonts: use three or less fonts, use them the same way (Headings/body text)



Consistency/Coherence

Colors: use the same colors consistently

Images: choose images that are similar in style

Fonts: use three or less fonts, use them the same way (Headings/body text)



Abril Fatface



Raleway

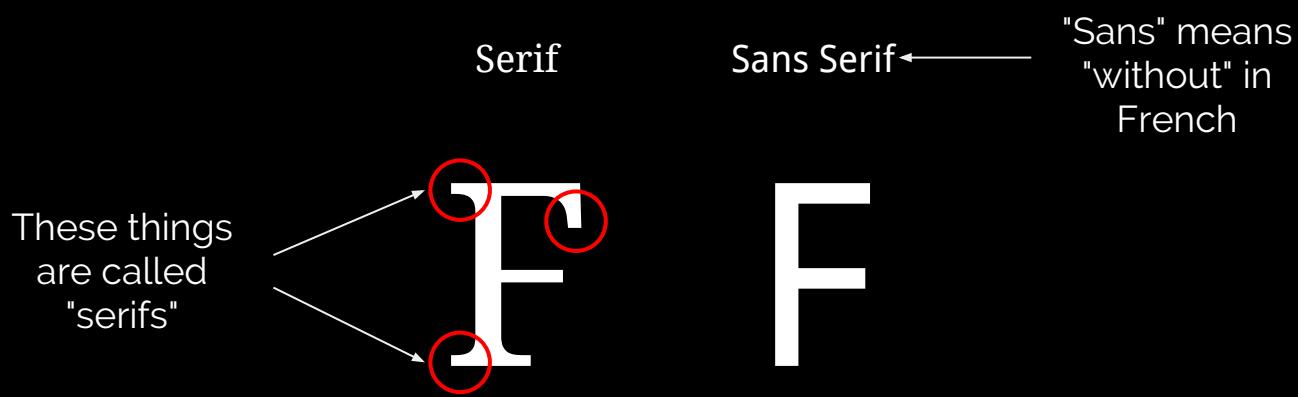
Fonts

Serif

Sans Serif

F F

Fonts



Fonts

Serif

F

Sans Serif

F

For larger blocks of
text

Easier to read

Formal

Modern, Simple, Clean,
Friendly

More eye catching

Informal

Limit Font Usage

1. Choose complementary or contrasting:
 - A. "PERSONALITY OF FONTS"
 - B. **CHUNKY AND BOLD VERSUS THIN AND LIGHT**
2. MIX serif and sans serif
3. Be aware of sizing
 - a. TRY ALL CAPS FOR SMALLER FONTS
4. Match content
5. Define a hierarchy:
 - a. Use the same font for the same type of thing
 - B. SEPARATE TO HEADINGS/TEXT



This is
too hard
to read!!

Mixing Fonts

1. Choose complementary or contrasting:
 - a. "personality of fonts"
 - b. Chunky and bold versus thin and light
2. Mix Serif and Sans Serif
3. Be aware of sizing
 - a. Try all caps for smaller fonts
4. Match content
5. Define a hierarchy:
 - a. Use the same font for the same type of thing
 - b. Separate to headings/text

Font Pairing

Similar Font Personalities

Chunky bold, thin and light

Serif and Sans Serif

Fancy Themed

Same Font Family

Special Elite
Consolas

Acme
Raleway

Sniglet
Garamond

Parisienne
Lora

Caveat Brush
Caveat

Comic Sans

Google Google



Microsoft



Microsoft



Love Letters

*I will always find
you*

**I WILL ALWAYS FIND
YOU**

Content

When you write your slides, less is more

With content that you put on it, it is important that you don't write too much stuff. Your audience can either listen to you or read the slide, but not both

Long sentences are difficult to follow and clutter the slide. Use shorter sentences to make what you are saying more clear.

Make sure the slide is not too cluttered so that your audience doesn't know where to look

Content

Less is more

Don't write too much

Audience can either listen or read

Use short sentences/phrases

Don't clutter your slides

Hello everyone!

Can you see me?

Do I have your
attention?

Post its and extra
shapes can help catch
attention

Yoo hooo~

Look at me!

Notice how this slide
is getting too
cluttered

<https://www.slidescarnival.com/>

<https://slidesgala.com/>

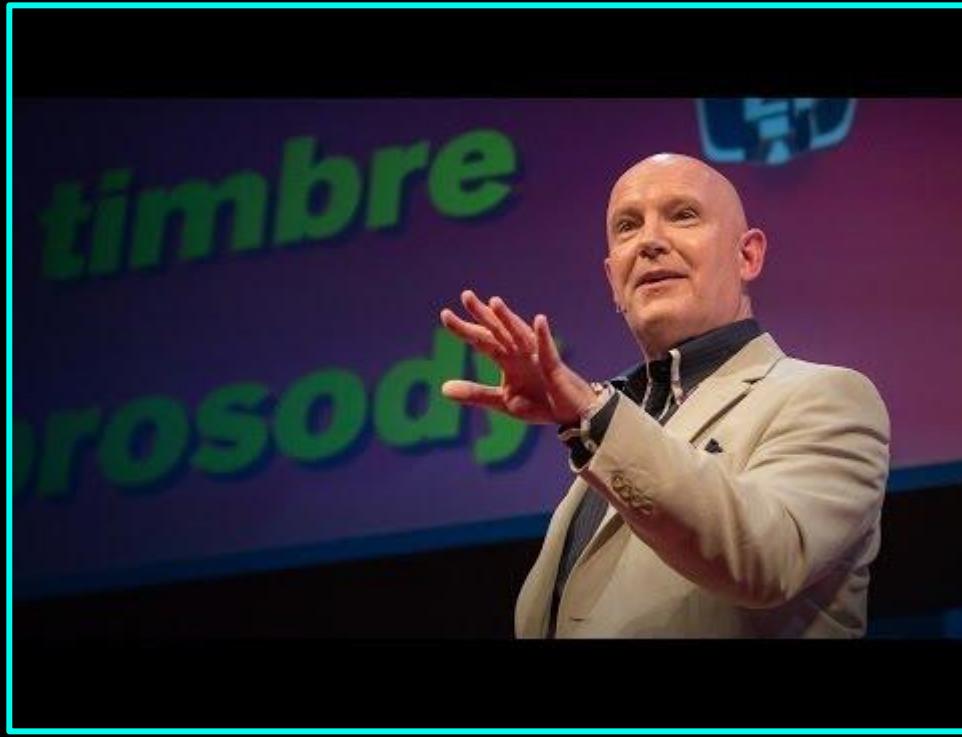
Make your own themes in Google Slides!

The screenshot shows the Google Slides interface with a dark theme. The top navigation bar includes File, Edit, View, Insert, Format, Slide, Arrange, Tools, Add-ons, and Help. Below the menu is a toolbar with icons for present, animations, and layout. The left sidebar displays a list of slides numbered 69, 70, and 71. A context menu is open over slide 71, with the 'Master' option highlighted and circled in red. Other options in the menu include Present (Ctrl+F5), Animations (Ctrl+Alt+Shift+B), Grid view (Ctrl+Alt+1), Zoom, Show ruler, Guides, Snap to, Show speaker notes (checked), and Full screen.

The main content area shows a slide titled "Simple Light" with the subtitle "Editing: Simple Light - Main point (Used by 3 slides)". The slide contains several text boxes with placeholder text: "Hello everyone!", "Can you see me?", "Do I have your attention?", and "https://www.slideshare.net". On the right side of the slide, there is a large text area with the instruction "Click to edit master title style" repeated multiple times.



How to sound smart in your TEDx Talk | Will Stephen | TEDxNewYork
<https://youtu.be/8SoFDjFBi8o>



How to speak so that people want to listen | Julian Treasure
<https://youtu.be/elho2SoZahl>