WHIT ES
PA
CE

Hi, there!

I’m Luciano Battagliero

@battaglr

Why you should care about whitespace

Well, to be fair…

Most of the time **you shouldn't need** to worry about those slippy invisible things

Unless you're programming in whitespace, then it's kind of a big deal… and maybe you're with too much time in your hands

Our **tools** should take care of them **without** too much **intervention** from our side.

After all, we have far more interesting things to do

Why are they relevant then?

They enhance the **authoring experience**

Making our code **easier** to **read** and to **understand**

And our **diffs** more **meaningful**
and easier to **review**

Whitespace in programming

Two main types

Syntactically **significant** whitespace

Syntactically **insignificant** whitespace

We are going to focus on the latter

They **can be omitted** without changing the meaning of the code

Most are **equivalent** and **interchangeable**

Usually a **sequence** of them are **equivalent** to a **single one**

Whitespace characters

Space

Tab

Newline or line ending

Get X-ray vision!

To see the whitespace characters, you can enable the “**show invisibles**” option in your editor

Pick a side

Tabs vs. **spaces**

The never ending war

Indenting with **spaces**

Fixed width

```
var foobar = function(obj, attrs) {
··var i = 0,
······len = obj.length;

··for (i; i < len; i++) { … }
}
```

Okay!

Indenting with **tabs**

Variable width (usually configurable)

```
var foobar = function(obj, attrs) {
  var i = 0,
      len = obj.length;

  for (i; i < len; i++) { … }
}
```

Okay!

Indenting with **spaces** and **tabs**

```
var foobar = function(obj, attrs) {
—var i = 0,
—····len = obj.length;

—for (i; i < len; i++) { … }
}
```

Okay? Okay!

Indenting with **whatever**

```
var foobar = function(obj, attrs) {
——var i = 0,
··————len = obj.length;

··for (i; i < len; i++) { … }
}
```

Tab stops that don’t correlate with the amount of space characters could result in code that is really hard to read

```
var foobar = function(obj, attrs) {
───var i = 0,
··──────len = obj.length;

··for (i; i < len; i++) { … }
}
```

Complete mayhem!

How to **avoid** this **invisible chaos**?

Choose one

Stick with it

Use it consistently

Choose. One.
Stick. With. It.
Use. It. Consistently.

Trim and exterminate

```
var foobar = function(obj, attrs) {·
··var i = 0,···
······len = obj.length;··
··········
··for (i; i < len; i++) { … }
}····
```

OH. MY. GOD!

Do you want to go to the **end of the line** with a keystroke? Good luck!

Were you expecting to land on the **first column**? Bummer!

```
var foobar = function(obj, attrs) {
··var i = 0,
······len = obj.length;

··for (i; i < len; i++) { … }
}
```

That’s better!

Mo' lines, mo' problems

Each OS **handle newlines** characters **differently**

Were you expecting an universally accepted standard? You, fool!

LF, **CR** or **CRLF**

```
var foobar = function(obj, attrs) {¬
··var i = 0,¬
······len = obj.length;¬
¬
··for (i; i < len; i++) { … }¬
}¬
```

LF: Unix, Linux and OS X

```
var foobar = function(obj, attrs) {\n
··var i = 0,\n
······len = obj.length;\n
\n
··for (i; i < len; i++) { … }\n
}\n
```

CR: Mac OS 9-

```
var foobar = function(obj, attrs) {\r
··var i = 0,\r
······len = obj.length;\r
\r
··for (i; i < len; i++) { … }\r
}\r
```

CRLF: Windows and MS-DOS

```
var foobar = function(obj, attrs) {\r\n
··var i = 0,\r\n
······len = obj.length;\r\n
\r\n
··for (i; i < len; i++) { … }\r\n
}\r\n
```

The thing is: you can **mix all three** in a **single file**!

An apparently harmless copy and paste could be the beginning of something very, very messy

```
var foobar = function(obj, attrs) {\r
··var i = 0,\r\n
······len = obj.length;\n
\n
··for (i; i < len; i++) { … }\r\n
}\r
```

Holy invisible cow!

Psst! Git will consider that the whole line was modified even if the only change was the newline character

Oh! Good luck with a merge conflict in a file that uses different newline characters

Normalizing newlines in **Git**

`.gitattributes`

```
* text=auto
```

```
*.css  text eol=lf
*.html text eol=lf
*.js   text eol=lf
```

No **newline** at **end of file**

You sure have seen the following message

```
\ No newline at end of file
```

That’s due to how **Unix** and Unix-like systems **handle text files**

Some files **without a newline** character at the end of it may **not** be recognised as a proper **text file**

Certain utilities could behave oddly if they aren’t able to **recognize** the **end of a file** and the **beginning of another**

By adding just **one character**
we could avoid possible
odd behaviours

Diff without noise

Oh, look I have another merge conf...

throws laptop out of window

A **clean codebase** means **clean diffs**

Clean diffs make our **work faster**

Our **reviews easier**

And our **merge conflicts**
a bit **less irritating**

When things are **already messy**

What if you're working on a **dirty codebase**?

Avoid modifying whitespace on a file **unless** you have a **good** an **objective reason**

Avoid to commit changes in lines where **only whitespace** has been modified **together with** changes in the **code**

A file to rule them all

`.editorconfig`

```
root = true

[*]
charset = utf-8
end_of_line = lf
indent_style = space
indent_size = 4
insert_final_newline = true
trim_trailing_whitespace = true
```

```
[*.md]
trim_trailing_whitespace = false
```

```
[package.json]
indent_size = 2
```

For more info go to
editorconfig.org

Why don’t we just **configure** our **tools**?

Within a team **not everybody** uses **the same** one

You depend on **every** single one
of the **team members** doing it
to make it work

It’s **tedious** and **time consuming**

And who wants to do that anyway?

Questions?

Thanks!

Countless whitespace characters were harmed
in the making of this presentation