
BREAKING THE ICE
WITH
REGULAR EXPRESSIONS

Contents

Why Regular Expressions?

Imagine we run a website that asks people to enter their Orlando-area phone number to receive local restaurant offers.

407-555-1212

First 3 numbers have to be either 407 or 321 (Orlando area code)

Why Regular Expressions?

Imagine we run a website that asks people to enter their Orlando-area phone number to receive local restaurant offers.

Next, there could be a dash,
but it's optional

Why Regular Expressions?

Imagine we run a website that asks people to enter their Orlando-area phone number to receive local restaurant offers.

407-555-1212

Followed by 3 more numbers

Why Regular Expressions?

Imagine we run a website that asks people to enter their Orlando-area phone number to receive local restaurant offers.

407-555-1212

Another optional dash

Why Regular Expressions?

Imagine we run a website that asks people to enter their Orlando-area phone number to receive local restaurant offers.

407-555-1212

And finally, 4 more numbers

What would the code to validate this look like?

Validating Input Without Regular Expressions

```
if
  ((chars[0] == 4 && chars[1] == 0 && chars[2] == 7)
    ||
   (chars[0] == 3 && chars[1] == 2 && chars[2] == 1))
  &&
  chars[3] == "-" || chars[3] == "" && …
```

… and that's just the first few numbers!

The same validation using a regular expression

```
if (chars.match(/regular expression/))
```

FYI, the syntax can be different per language

Understanding a Regular Expression

```
if (chars.match(/regular expression/))
```

Subject String

The string where we're looking for a match is called the subject string

aka, regex

A set of characters that represent rules for searching or matching text in a string in a concise, predictable way

Our First Regular Expression

Does the phone number have a 407 area code?

```
if (chars.match(/407/))
```

Literally matches the characters 407

number *subject string*

407-555-1212

True!

Not Finding a Match

Does the phone number have a 407 area code?

```
if (chars.match(/407/))
```

number *subject string*

321-555-1212

False!

Does the phone number have a 407 or 321 area code?

Using the OR / Alternation Operator

Does the phone number have a 407 or 321 area code?

```
if (chars.match(/407|321/))
```

Takes left-most match first

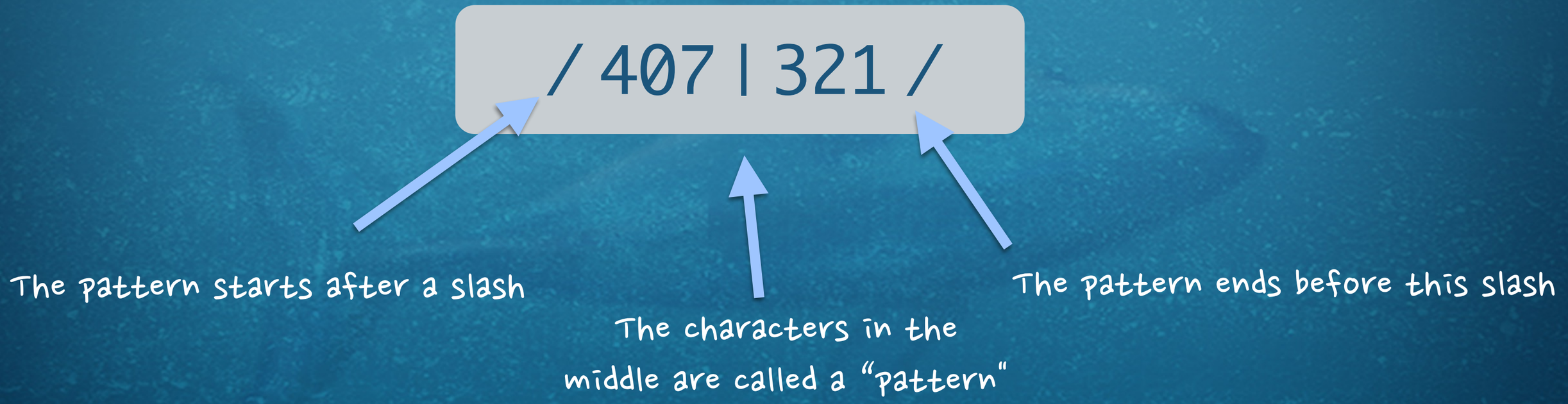
Matches the characters on the left or right

number *subject string*

321-555-1212

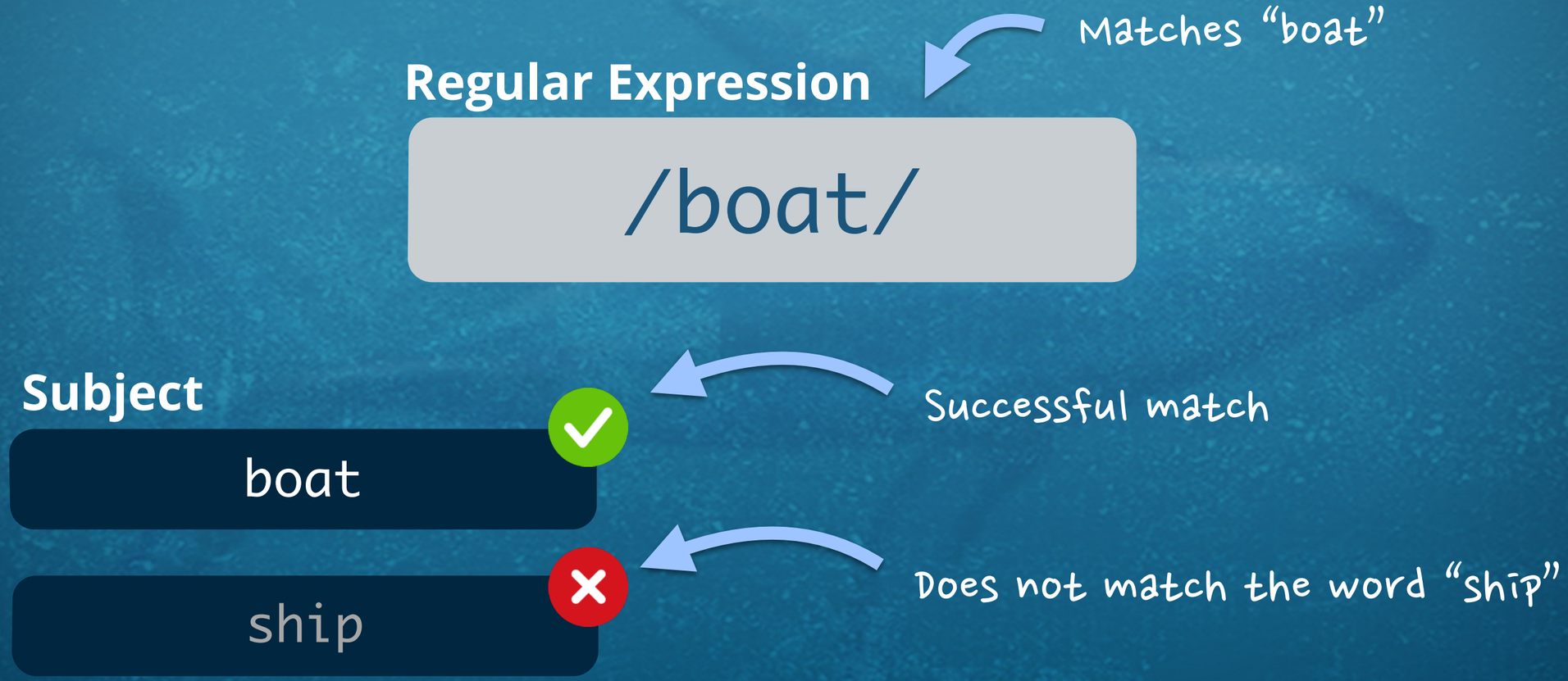
True!

The Parts of a Regular Expression

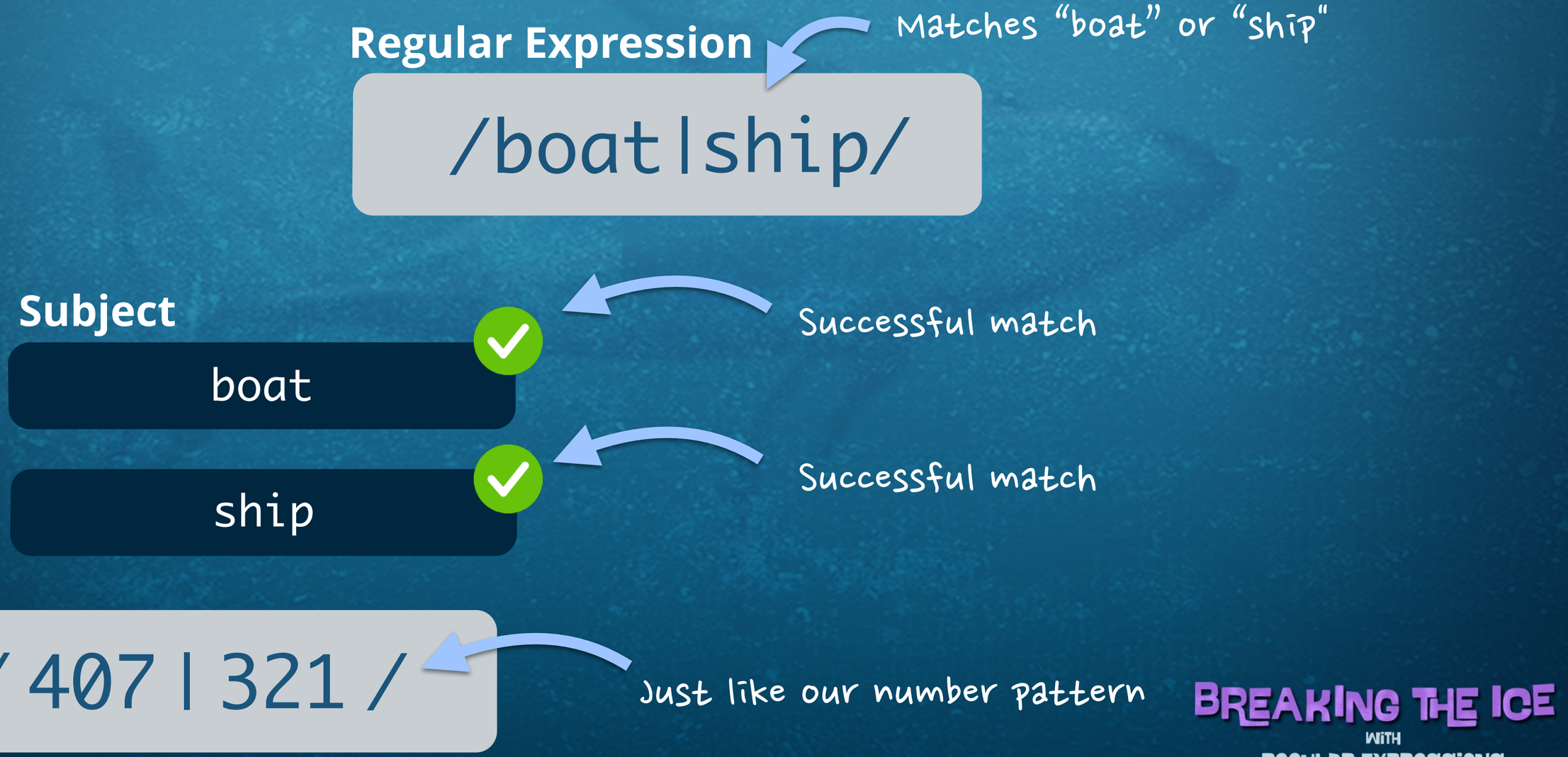


Writing regex is really like asking the question,
"Does a group of characters match a specific pattern?"

Not Just Numbers



Number OR letters



What Are Regular Expressions Used For?

Common uses include:

Validations

Phone numbers
Email
Passwords
Domain names

Searching

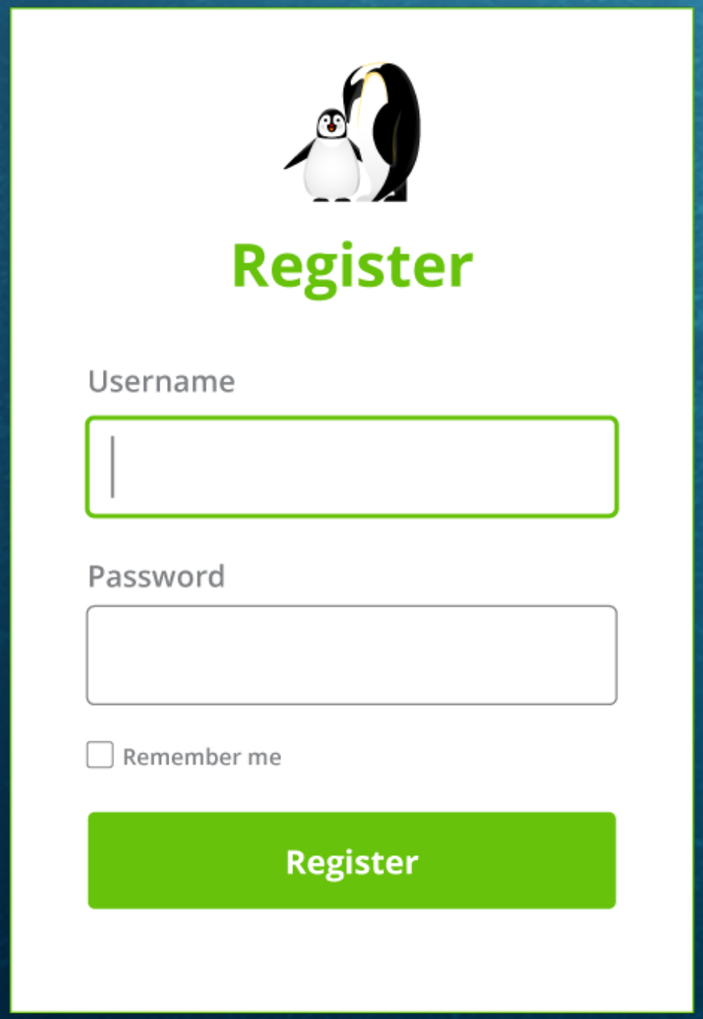
Words in a sentence
Unwanted characters
Extracting sections
Replacing/cleaning/formatting

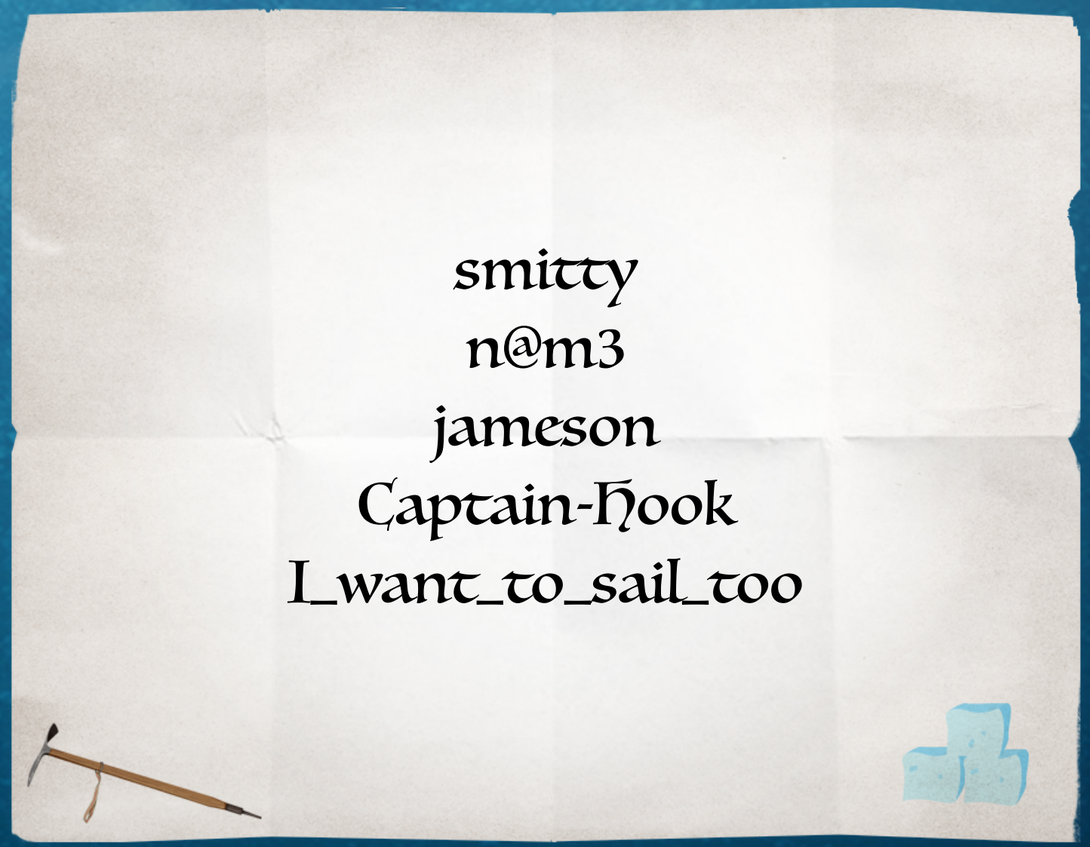
Section 2

The String Story

Our First Problem: Collecting Names

We are assembling a crew!





We Need to Check Multiple Usernames

How Can We Match on Multiple Versions of Names?

Match "smitty" - if no match, then match "james"

Regular Expression

```
/smitty|james/
```

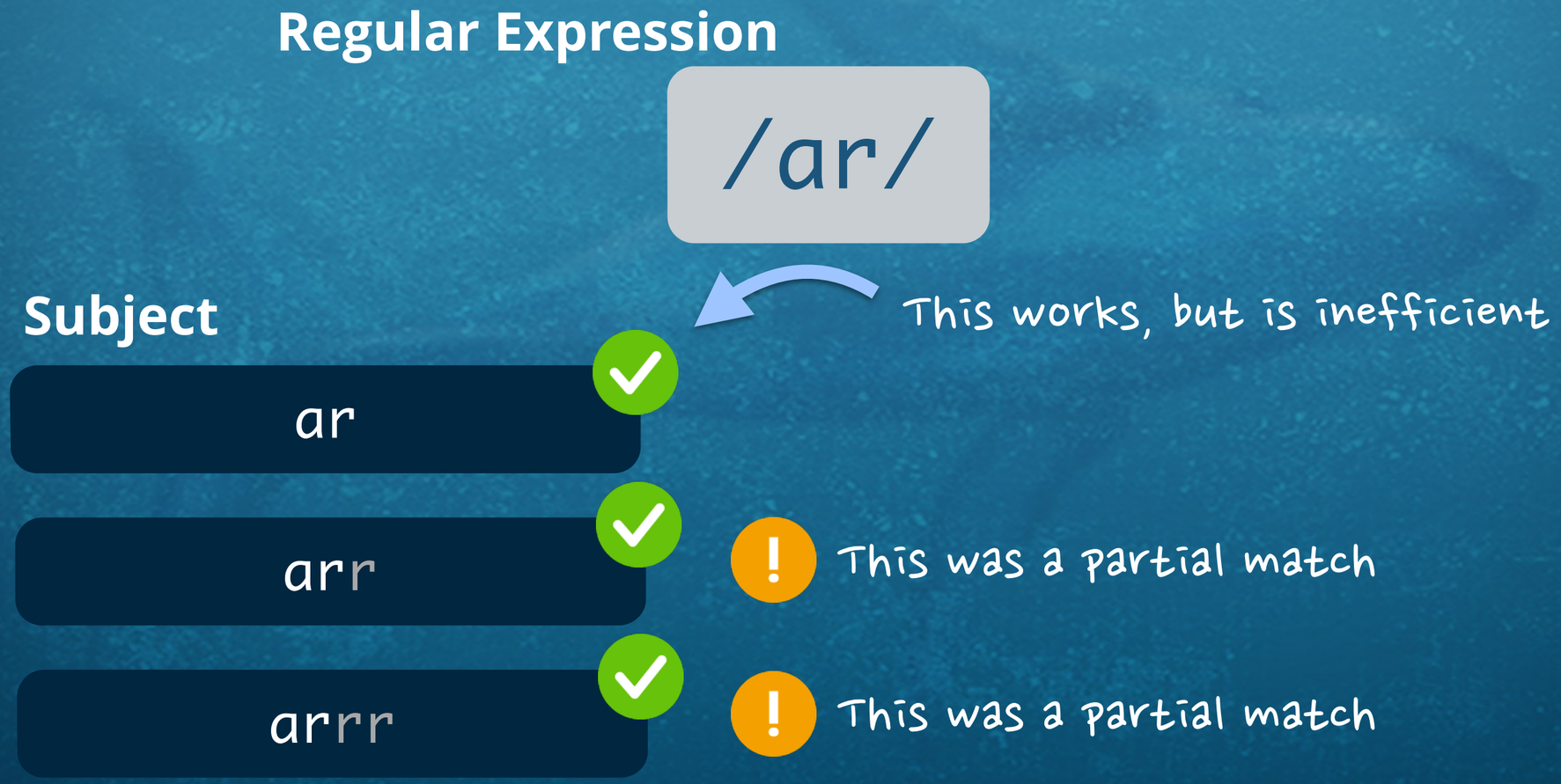
Subject

smitty

james

OR / alternation operator allows match on "smitty" or "james"

Problem: We Need to Repeat the “R”



Regular expressions look for matches anywhere in their subject

Problem: Long Expression Repeating

This is going to grow too large

Regular Expression

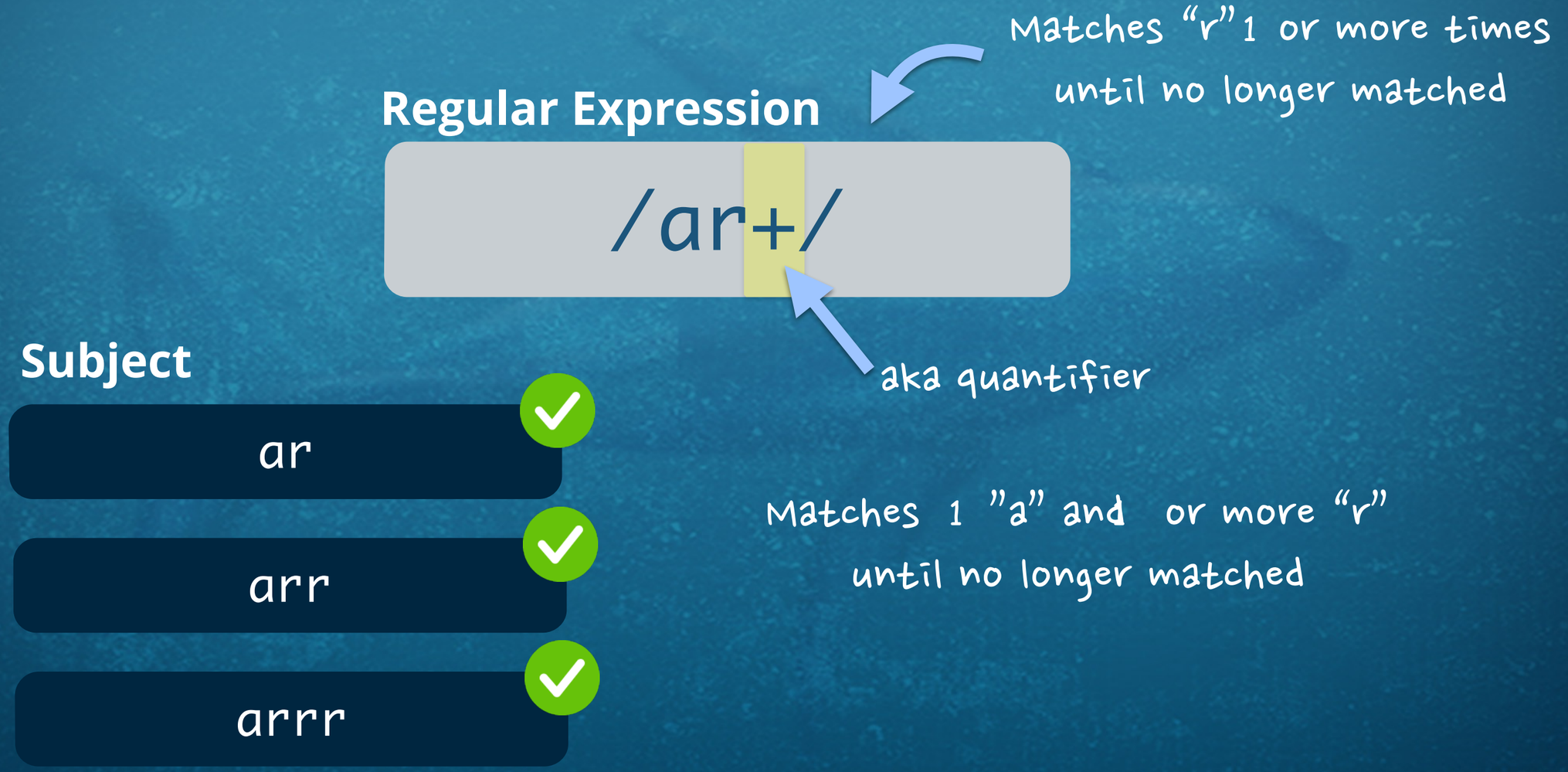
```
/ar|arr|arrr|arrrr/
```

There are just too many versions to account for

This works, but is inefficient

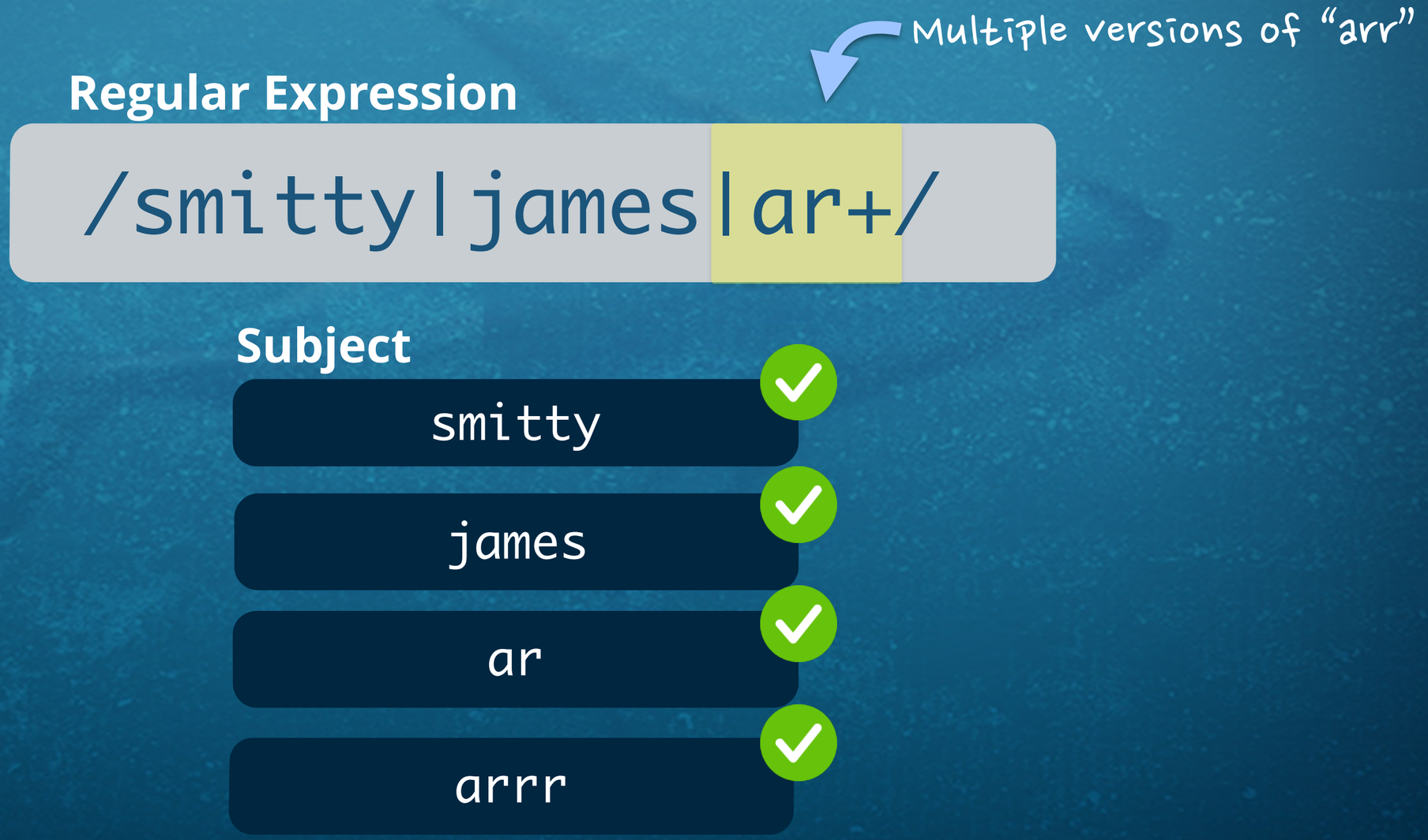
Subject

Solution: The Plus Operator



+ This means, "Look for the previous character 1 or more times."

How Can We Match on Multiple Versions of Names?



Matching on Multiple Versions of Names

Regular Expression

```
/smitty|james|ar+/
```

Subject

james

jameson

Last 2 letters are not matched

This was a partial match

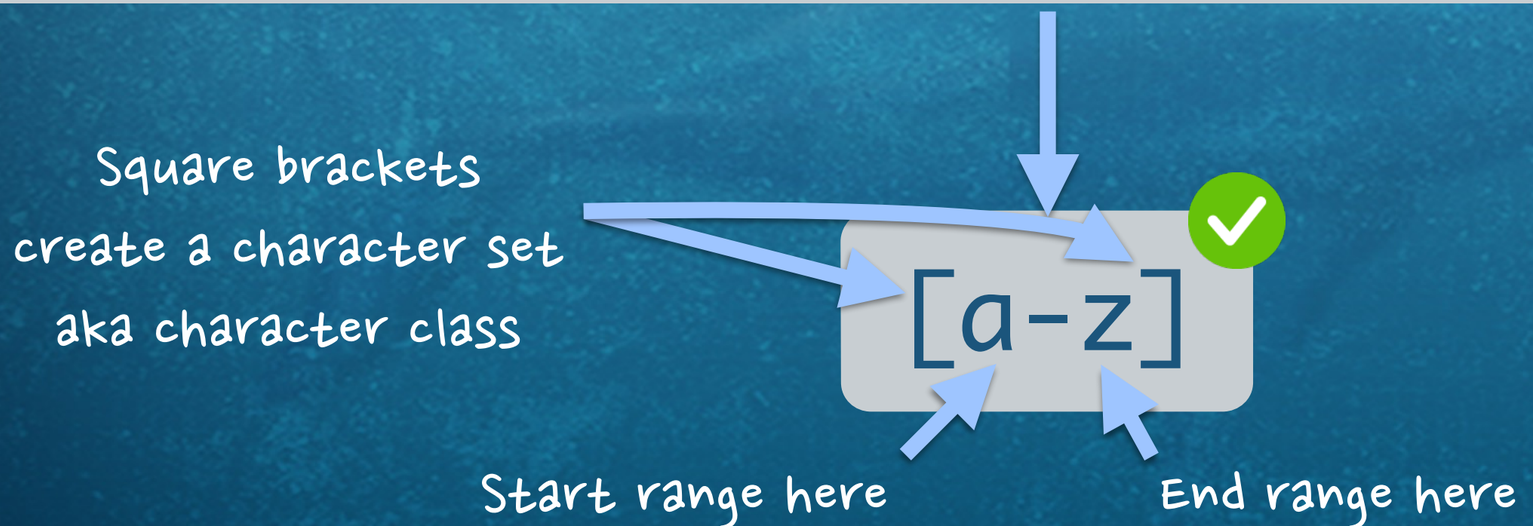
Regular expressions look for matches anywhere in their subject and take the left-most match first

Matching All Characters in the Alphabet Using Ranges

This works, but is inefficient

Regular Expression

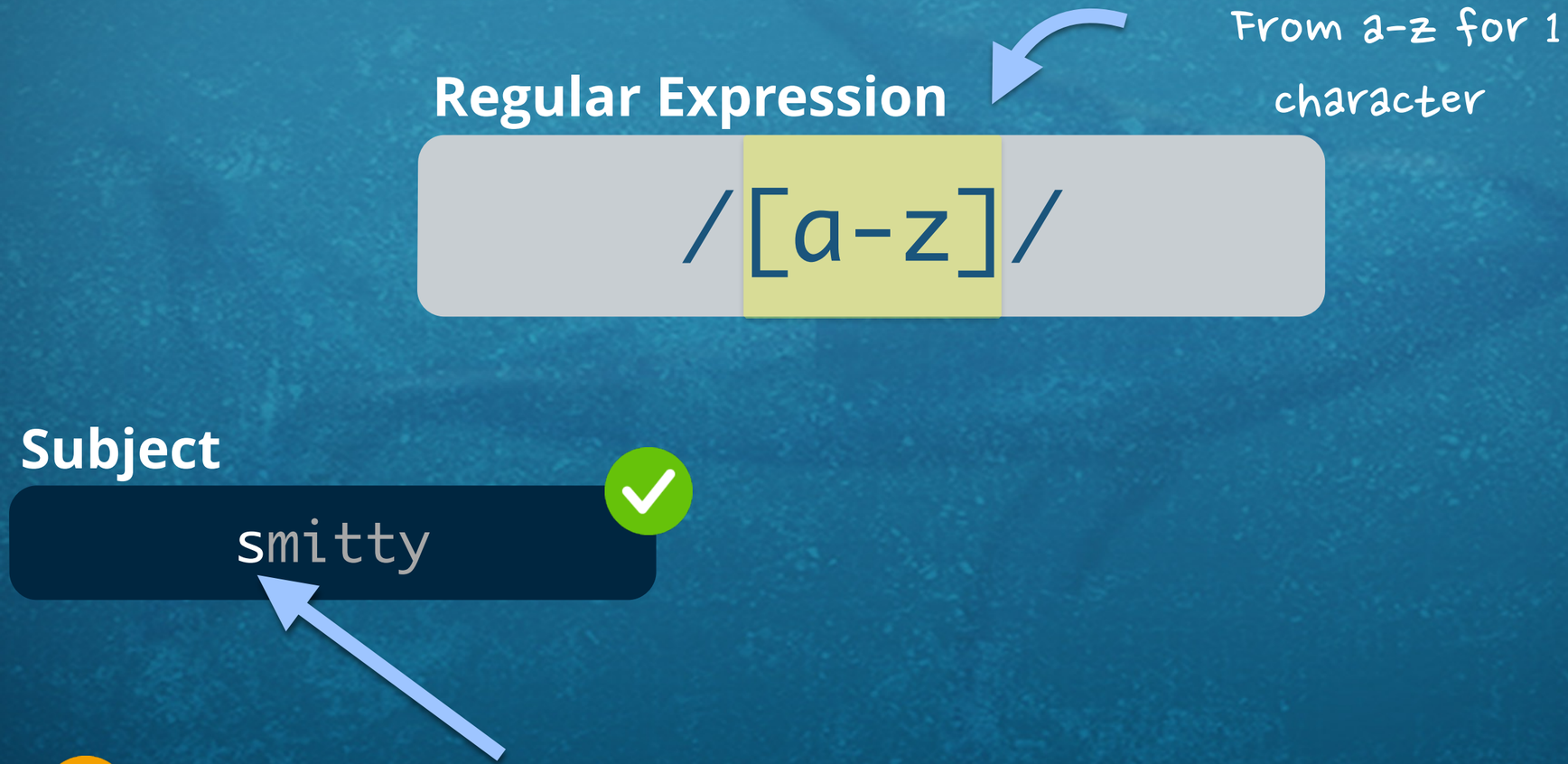
```
/a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z/
```



NOTES

- A range only works in a character set
- This character set represents 1 character

Using Our Character Set in a Pattern



Matching Multiple Characters With Multiple Character Sets

From a-z for 6 characters

Regular Expression

```
/[a-z][a-z][a-z][a-z][a-z][a-z]/
```

Subject

smitty

james

Does not match

How can we change this pattern
to accept any amount of letters?

Checking for 1 Character Set Multiple Times

Regular Expression

```
/[a-z]+/
```

Plus operator

+

This means, “Look for the previous character 1 or more times.”

Subject

- smitty ✓
- james ✓
- jameson ✓

Now they all match!

- ar ✓
- arr ✓
- arrr ✓

Problem: Capital Letters

Regular Expression

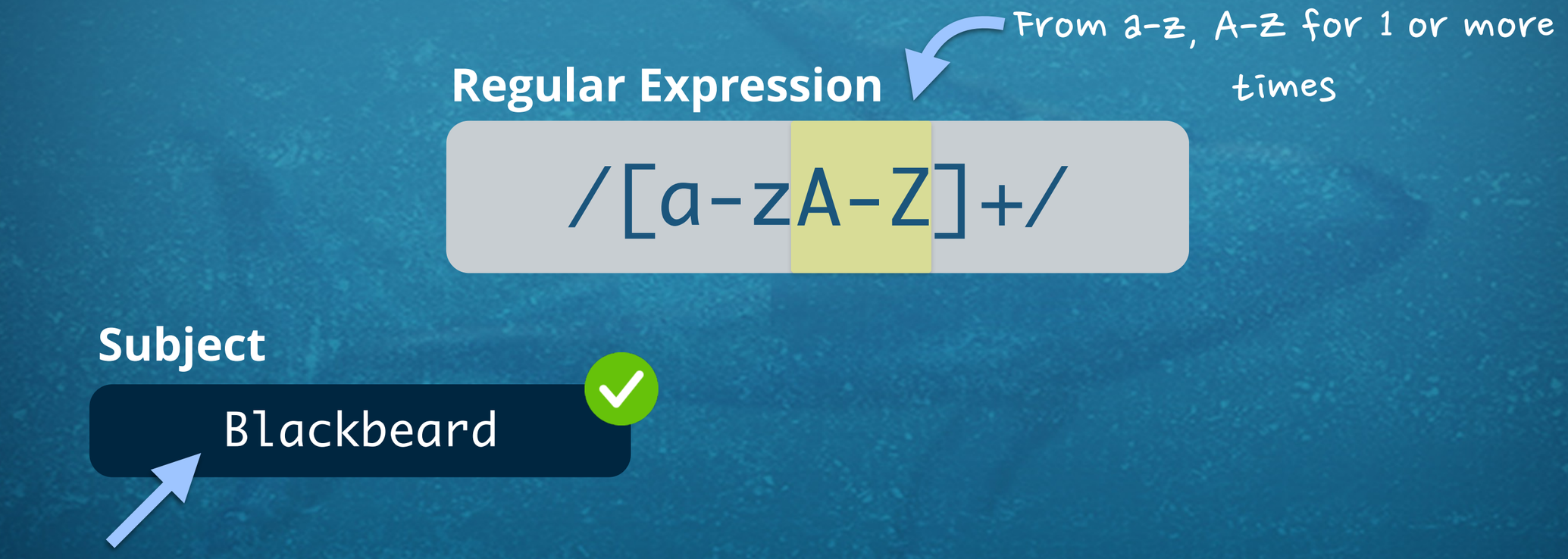
```
/[a-z]+/
```

Subject

Blackbeard

Problem: our pattern is ignoring capital letters

Matching Uppercase and Lowercase Characters



Simplifying Patterns With a Casing Modifier

Letters after final slash are called "modifiers"

Regular Expression

```
/[a-z]+/i
```

The `i` modifier means "case insensitive," which will match uppercase and lowercase characters.

Subject

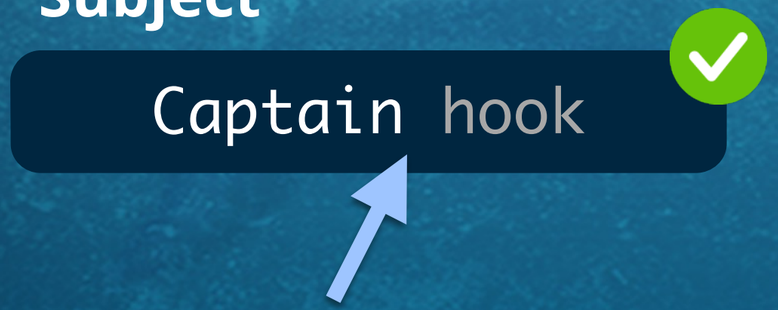
```
Blackbeard
```

Problem: Names With 2 Words Not Matching

Regular Expression

```
/[a-z]+/i
```

Subject



How Can We Match Whitespace?

using a literal space is hard to read - you might miss it!

\s

is the same as saying
"a whitespace character"

Whitespace can include:

- Spaces
- Tabs
- New lines

Matching Spaces in a Character Set

\s

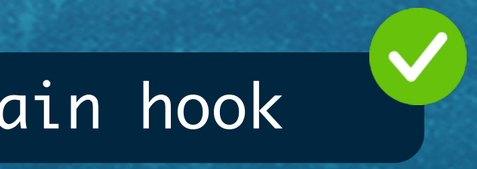
is the same as saying
"a whitespace character"

Regular Expression

```
/[a-z\s]+/i
```

Subject

```
Captain hook
```



NOTE: The order of characters doesn't matter here!

```
[\sa-z]+
```

is the same thing!

Problem: How Can We Match Numbers?

Does not account for numbers

Regular Expression

```
/[a-z\s]+/i
```

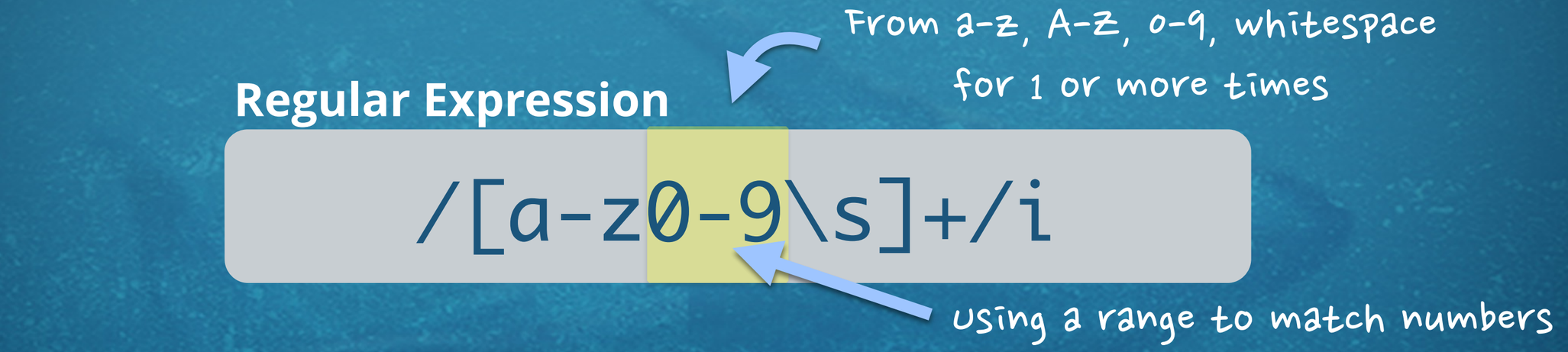
Subject

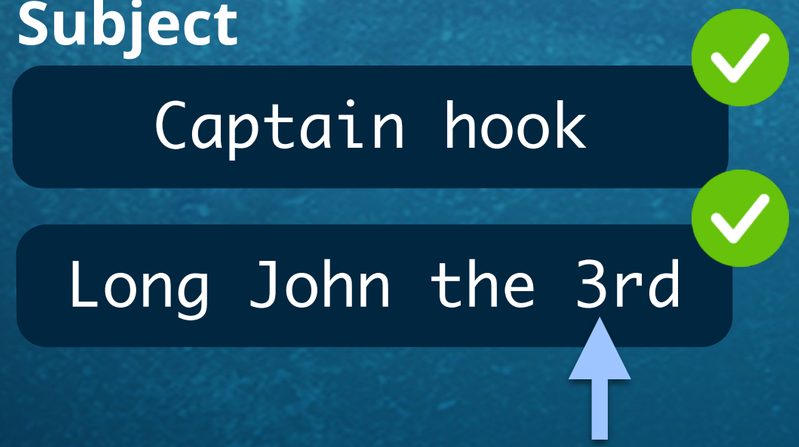
Captain hook

Long John the 3rd

Problem: we do not match any numbers

Matching Numbers in a Range



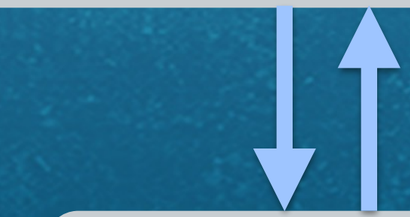


Refactoring With the Word Metacharacter

`\w` is the same as `[a-zA-Z0-9]`

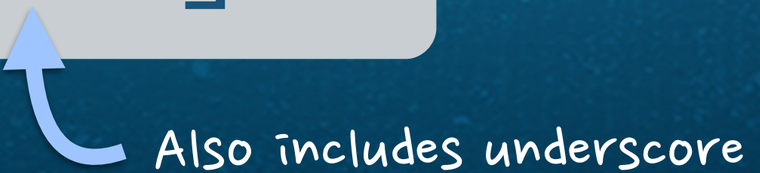
Regular Expression

```
/[a-z0-9\s]+/i
```



These have the same result,
but the shortcut is easier to read.

```
/[\w\s]+/
```



Also includes underscore

Analyze Our New Subject String

We now have our list of names for our voyage.

Let's work on verifying their emails!

Subject

`sara@example.com`

Starts with a word

Followed by an @ character

Followed by another word

BREAKING THE ICE WITH REGULAR EXPRESSIONS

Starting the Regular Expression

Regular Expression

```
/\w@\w/
```

Remember: `\w` only searches for 1 word-like character.

Subject

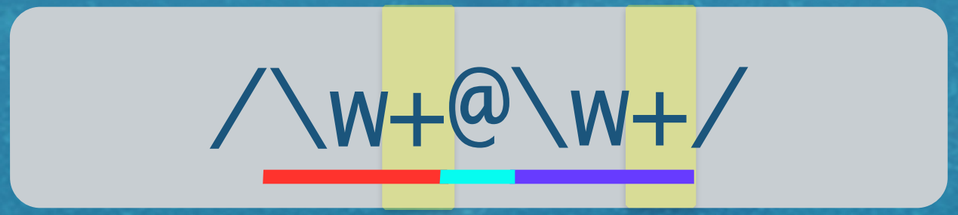
```
sara@example.com
```

Pattern Goals

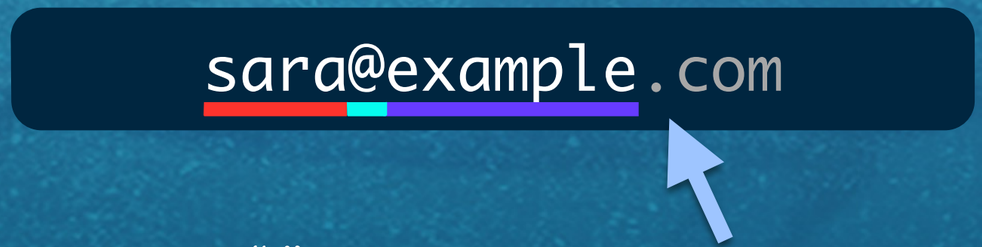
- 1 word character repeated 1 or more times
- ✅ @ symbol
- 1 word character repeated 1 or more times

Matching Full Words

Regular Expression



Subject



Because the "." is not a word, it is not matched

Pattern Goals

- ✅ 1 word character repeated 1 or more times
- ✅ @ symbol
- ✅ 1 word character repeated 1 or more times
- dot literal
- com

A Successful Match

Regular Expression

```
/\w+@\w+.\w+/
```

Subject

```
sara@example.com
```

We successfully matched!

Pattern Goals

- 1 word character repeated 1 or more times
- @ symbol
- 1 word character repeated 1 or more times
- dot literal
- com

BREAKING THE ICE WITH REGULAR EXPRESSIONS

Matching the .com

Regular Expression

```
/\w+@\w+.\w+/
```

Subject

sara@example.com

sara@example!com

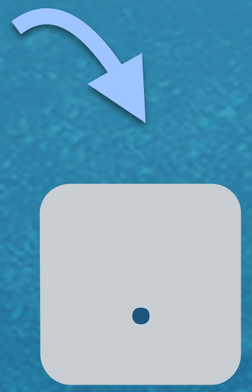
Hold up - an exclamation point shouldn't be matching here

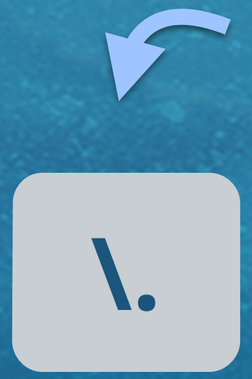
Pattern Goals

- 1 word character repeated 1 or more times
- @ symbol
- 1 word character repeated 1 or more times
- dot literal
- com

The . Wildcard

The “.” is a wildcard metacharacter

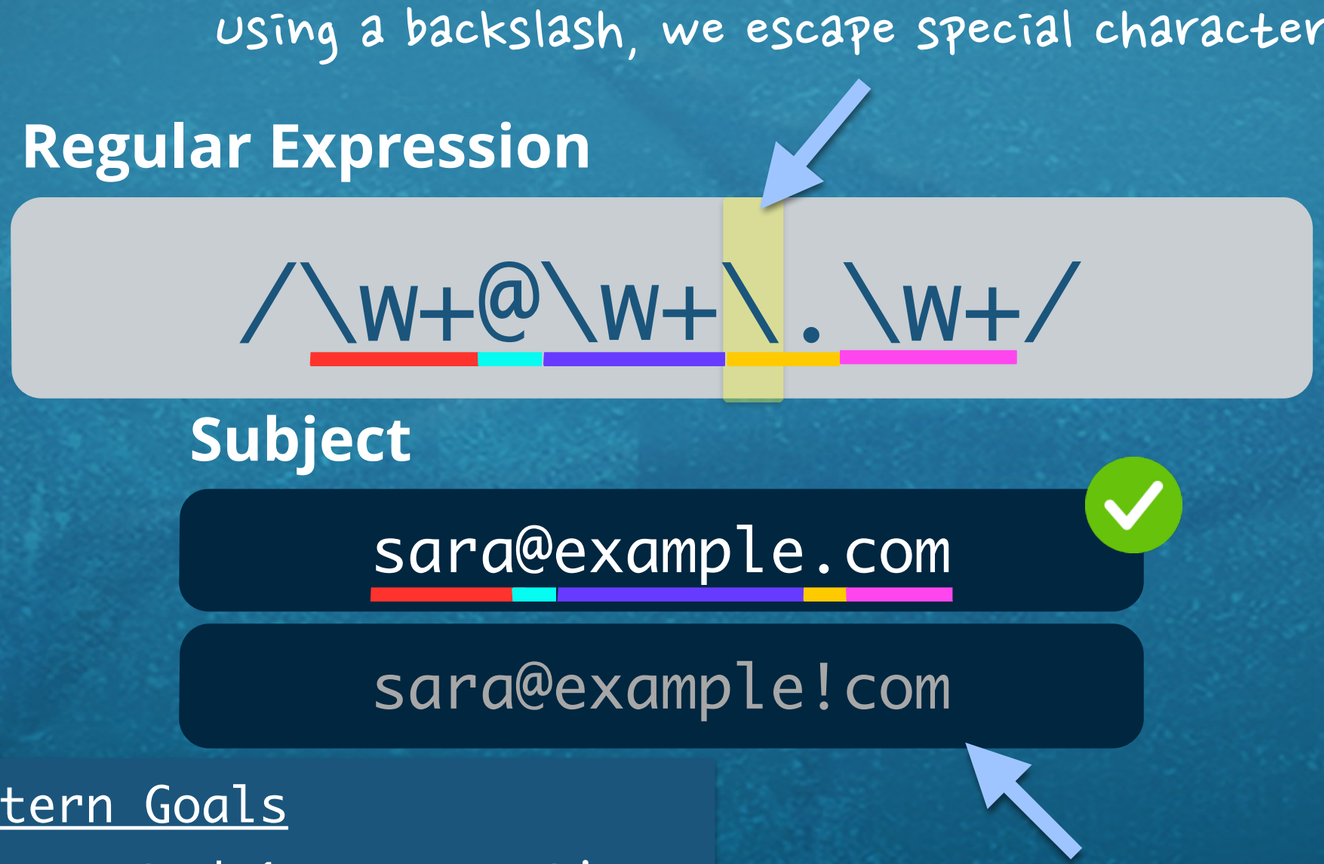




Escape the “.” with a backslash to match a literal period

The “.” wildcard matches any character except newline

Escaping a Character in a Pattern

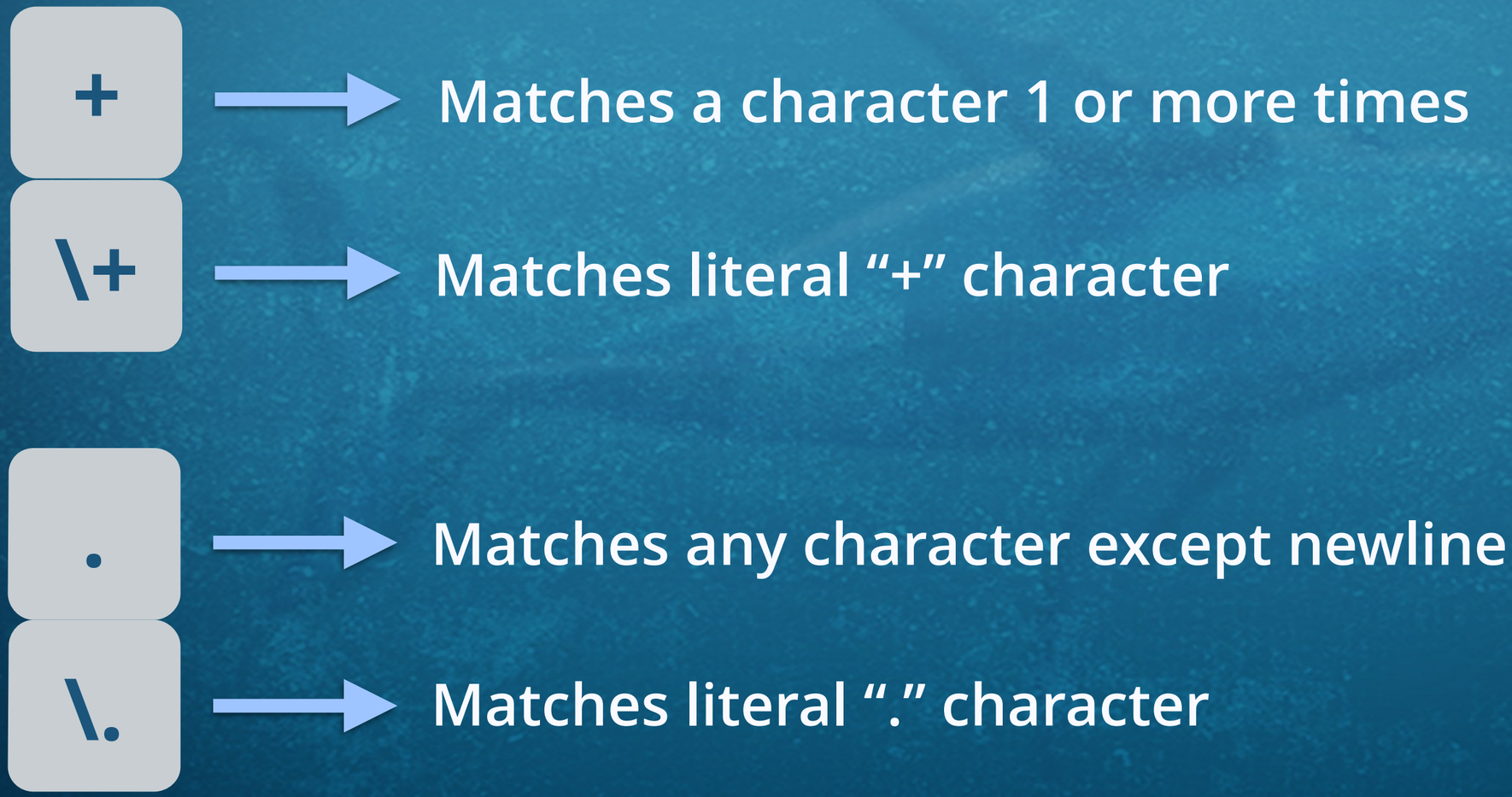


Pattern Goals

- 1 word character repeated 1 or more times
- @ symbol
- 1 word character repeated 1 or more times
- dot literal
- com

BREAKING THE ICE WITH REGULAR EXPRESSIONS

Escaping Characters



Being More Specific About the Top-level Domain (TLD)

Regular Expression

4 common TLDs

```
/\w+@\w+\.com|net|org|edu/i
```

Subject

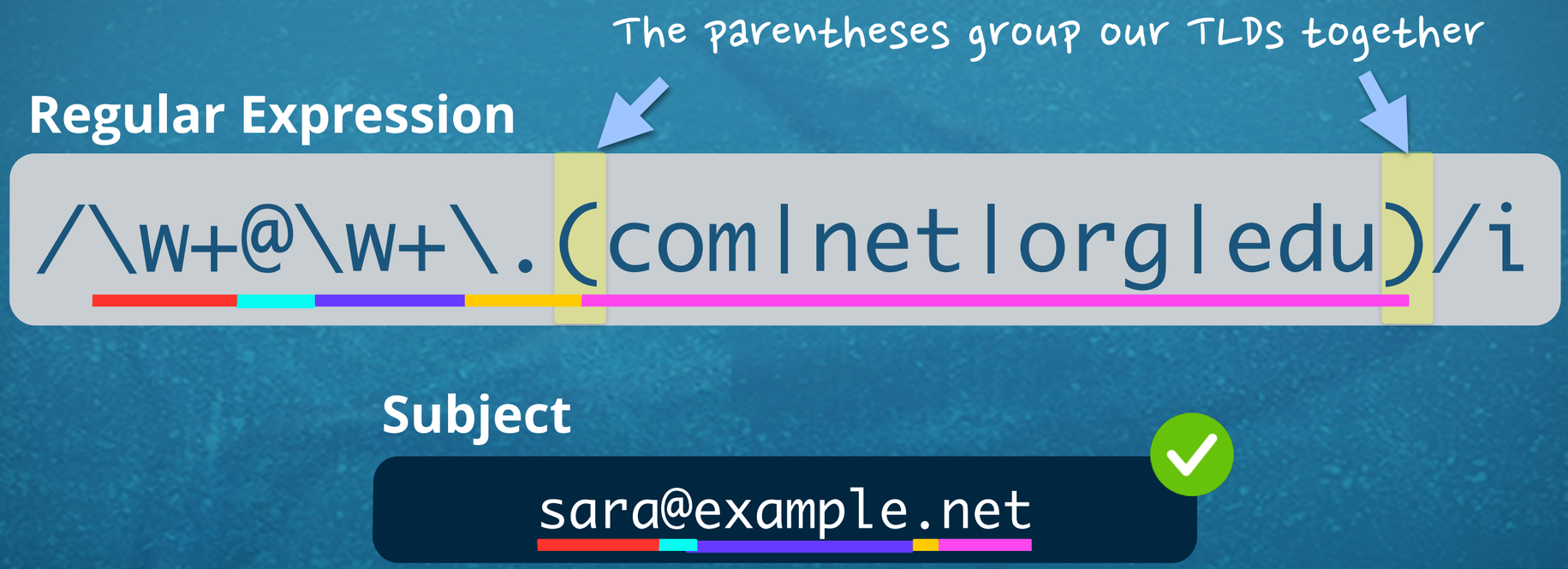
```
sara@example.net
```

OR / alternation operators not evaluating properly

Pattern Goals

- ✗ 1 word character repeated 1 or more times
- ✗ @ symbol
- ✗ 1 word character repeated 1 or more times
- ✗ dot literal
- ✓ com or net or org or edu

Grouping Together TLDs



Pattern Goals

- 1 word character repeated 1 or more times
- @ symbol
- 1 word character repeated 1 or more times
- dot literal
- com or net or org or edu

Problem: Need to Ignore Characters Before and After a Subject

Regular Expression

```
/\w+@\w+\.(com|net|org|edu)/i
```

Subject

sara@example.comlksh

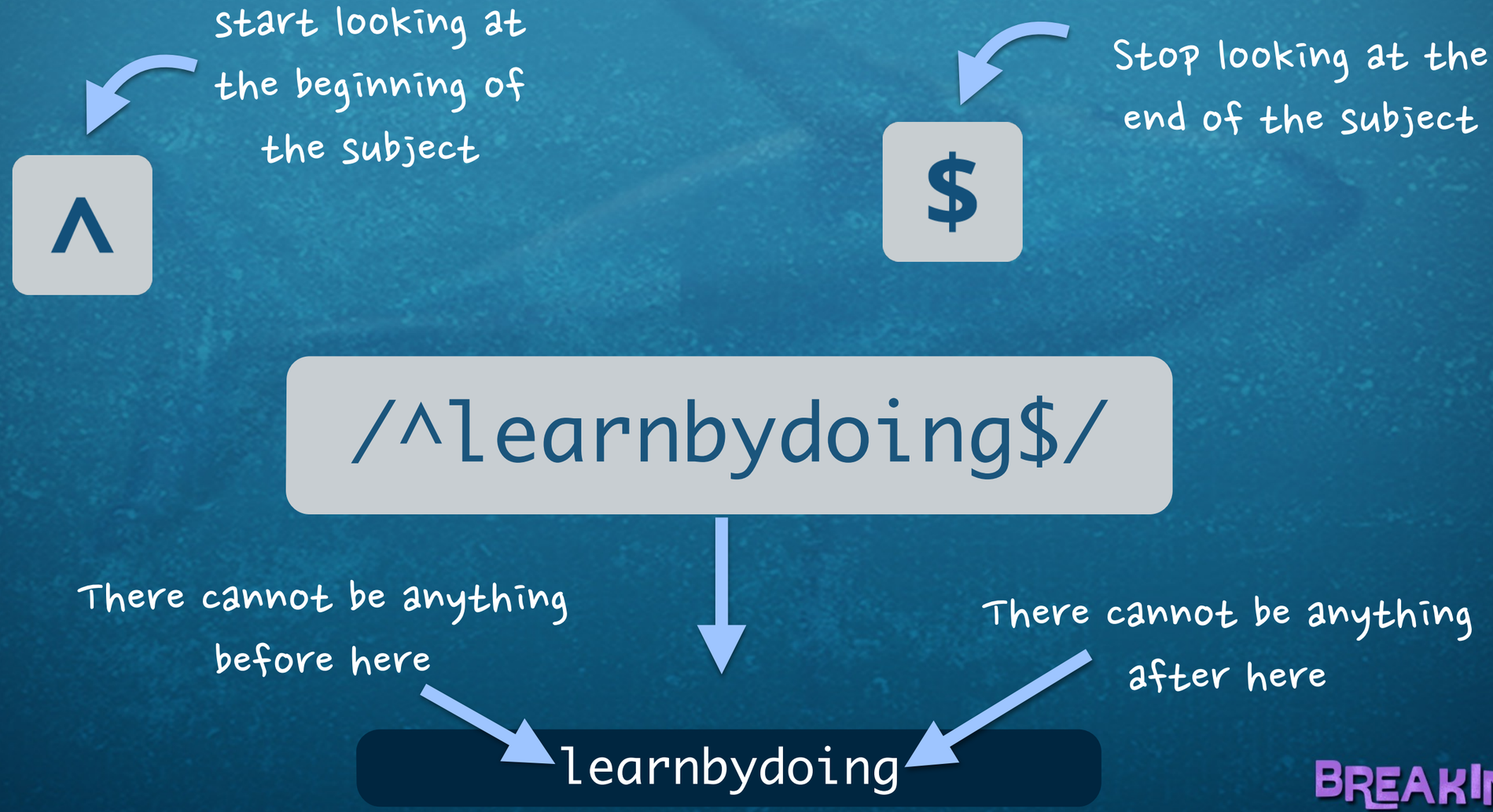
We get a partial match but shouldn't!
These characters should cause the match to fail.

Pattern Goals

- 1 word character repeated 1 or more times
- @ symbol
- 1 word character repeated 1 or more times
- dot literal
- matches com or net or org or edu
- no text before or after email

BREAKING THE ICE
WITH
REGULAR EXPRESSIONS

Introducing Anchors



Adding Anchors to Our Pattern

Regular Expression

```
/^\w+@\w+\.(com|net|org|edu)$/i
```

start matching at the beginning of the line

stop matching at the end of the line

Pattern Goals

- 1 word character repeated 1 or more times
- @ symbol
- 1 word character repeated 1 or more times
- dot literal
- matches com or net or org or edu
- no text before or after email

Subject

sara@example.com

sara@example.comlksh

Great - our invalid email is no longer matched!

Contents

Joining the Voyage

Find each shipmate's answer as to whether they are willing to voyage!

Subject Strings	Accepted Answer Keywords
ok, i will do it	ok
okie dokie	Okay
Ahooy, Okay!!!	sure
why sure, i can go	yes
arrr, yes matey	y
My answer me mate, is yes	
y	

Starting the Expression by Matching the First Confirmation

using ok literal to directly match "ok"

Regular Expression

```
/ok/
```

Possible Subjects

ok, i will do it

okie dokie

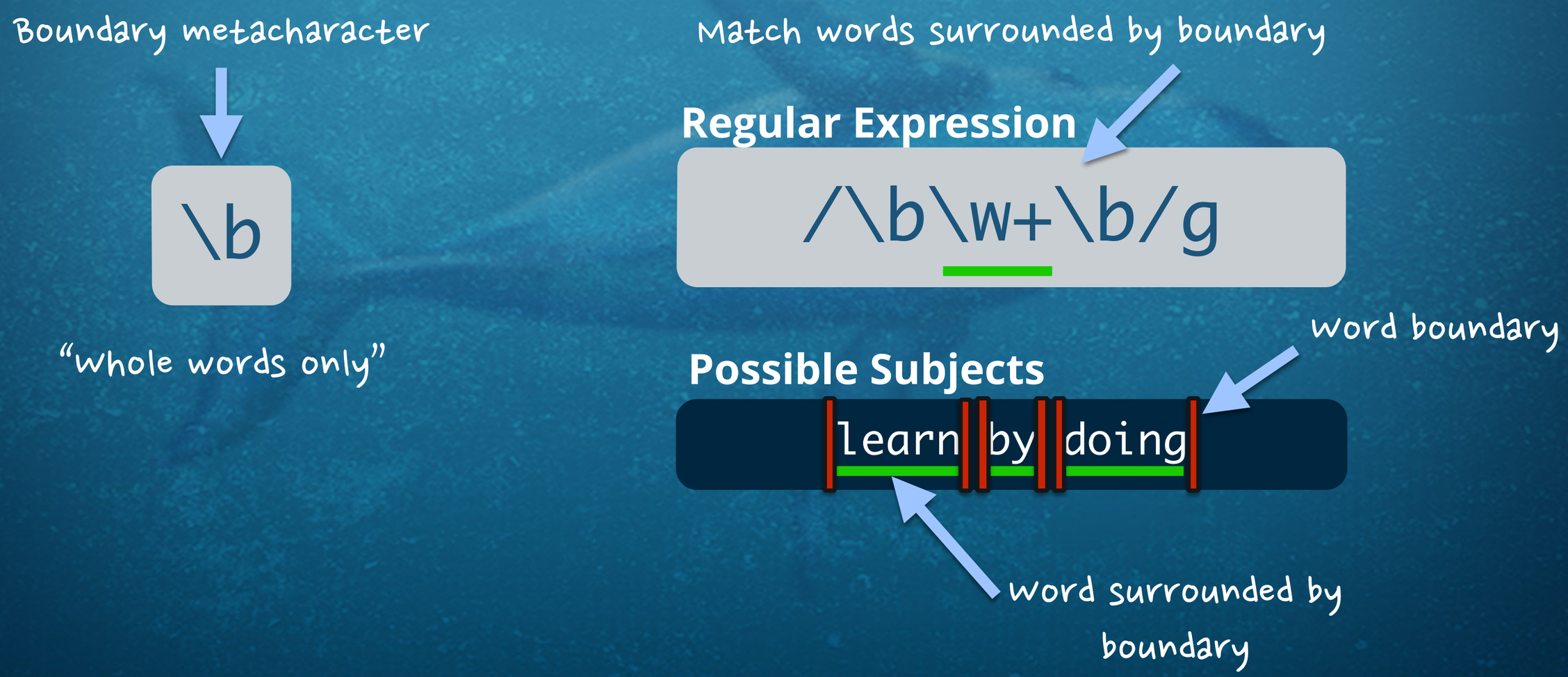
Pattern Goals

- ok
- Okay
- sure
- yes
- y
- answer by itself
- ignore other text

we want the answer to be by itself and not part of another word

BREAKING THE ICE WITH REGULAR EXPRESSIONS

Detecting Word Boundaries With the Boundary Metacharacter



Using Boundaries in a Pattern

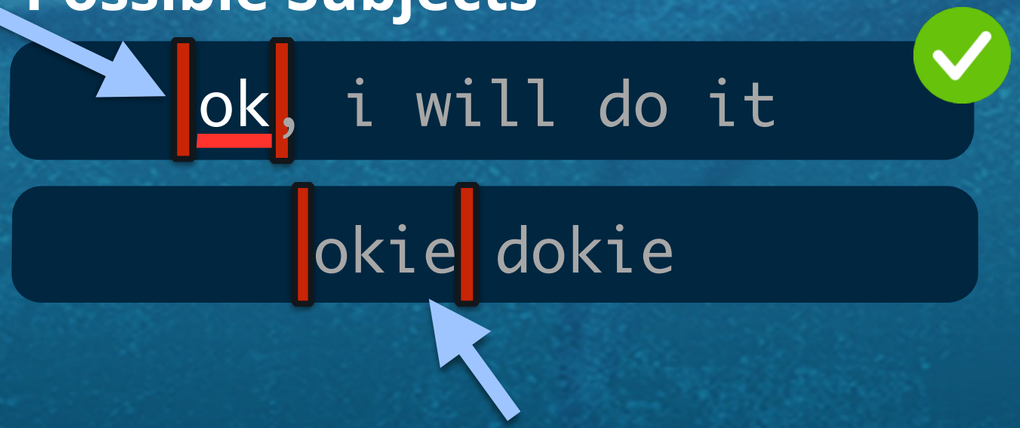
Regular Expression

```
/\bok\b/
```

Ensures our match is a single word

`\b` = boundary metacharacter

Possible Subjects



Great - we don't get a match because "ok" does not have a boundary on the right, it only has letters

Pattern Goals

- ✅ ok
  - Okay
  - sure
  - yes
  - y
- ✅ answer by itself
- ✅ ignore other text

Problem: We Need to Match Multiple Versions of a Word

Regular Expression

```
/\bok\b/
```

Possible Subjects

```
Ahooy, Okay!!!
```

wait - we want "okay" to match too

Pattern Goals

- ✅ ok
  - Okay
  - sure
  - yes
  - y
- ✅ answer by itself
- ✅ ignore other text

Inefficient Matching With Multiple OR Statements

Regular Expression

```
/\bok\b|\bokay\b/
```

"ok" OR "okay"

But what if there were 50 different options?

If we had a way to make "ay" optional, we could match with a single pattern.

```
/\bok(ay)\b/
```

How can we make this optional?

Pattern Goals

- ✅ ok
  Okay
  sure
  yes
  y
- ✅ answer by itself
- ✅ ignore other text

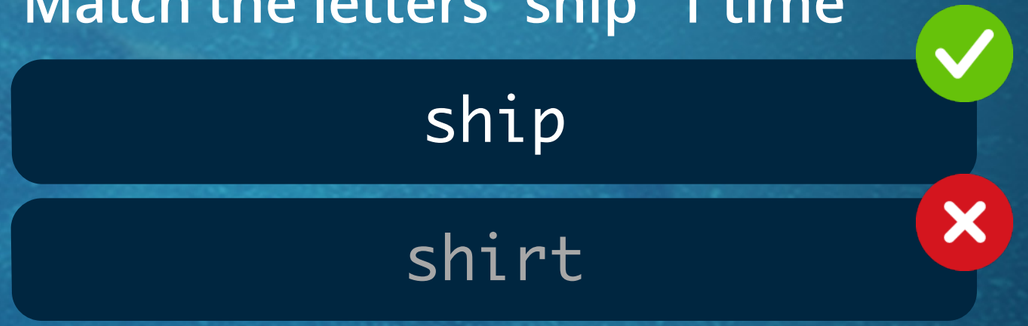
Checking for a Pattern 0 or More Times

Regular Expression

```
/ship/
```

All of these characters are required

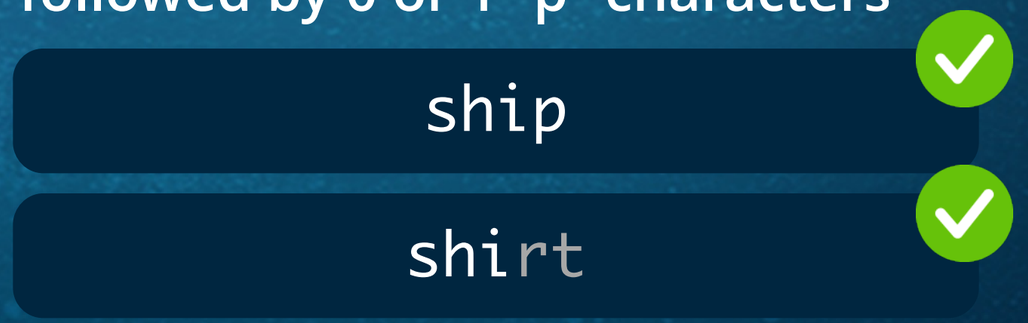
Match the letters “ship” 1 time




```
/ship?/
```

characters in a pattern that are followed by a question mark are optional

Match the letters “shi” 1 time followed by 0 or 1 “p” characters



Marking a Group of Characters as Optional

Regular Expression

```
/pirate\s(ship)?/
```

Group characters in parentheses followed by "?" to make them all optional

```
pirate ship
```

```
pirate boat
```

Matching Different Versions of the Same Word

Regular Expression

Match 0 or 1 times

"i" modifier for upper- and lowercase

```
/\bok(ay)?\b/i
```

Possible Subjects

Ahooy, Okay!!!

ok, i will do it

Pattern Goals

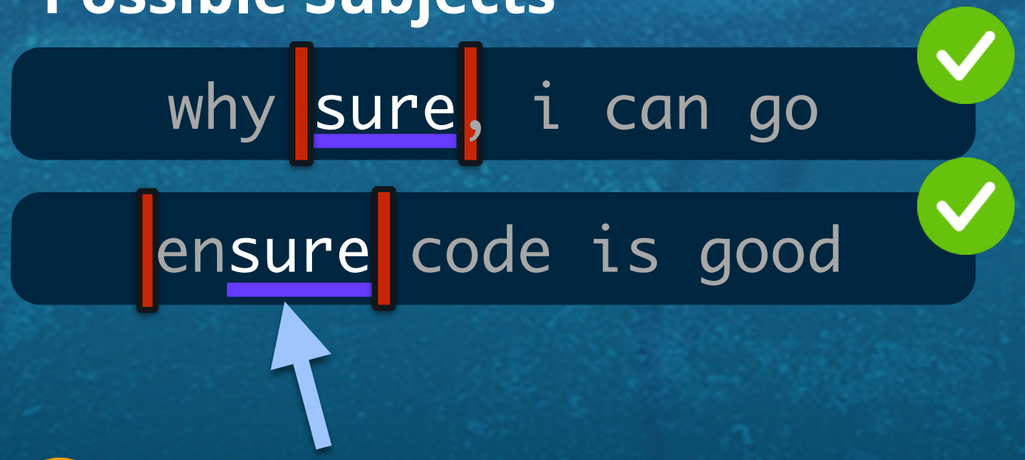
- ok
- Okay
- sure
- yes
- y
- answer by itself
- ignore other text

Problem: The OR Operator Is Splitting the Boundary

Regular Expression

```
/\bok(ay)?|sure\b/i
```

Possible Subjects



Pattern Goals

- ✓ ok
- ✓ Okay
- ✓ sure
- yes
- y
- ✗ answer by itself
- ✓ ignore other text

! we do not want a partial match of the answer

Solution: Group the Pattern Within the Boundary

Group ensures boundary is applied to all answers

Regular Expression

```
/\b(ok(ay)?|sure)\b/i
```

Pattern Goals

- ok
- Okay
- sure
- yes
- y
- answer by itself
- ignore other text

Possible Subjects

why sure, i can go

ensure code is good

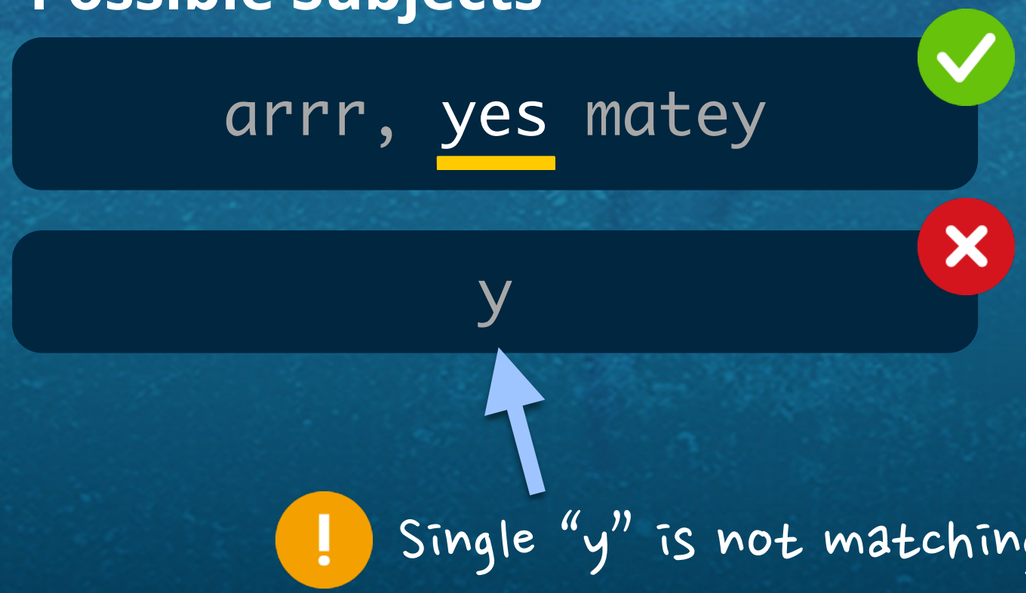
Great - "ensure" is no longer matched

Continuing to Match More Valid Subjects

Regular Expression

```
/\b(ok(ay)?|sure|yes)\b/i
```

Possible Subjects



Pattern Goals

- ok
- Okay
- sure
- yes

  y
- answer by itself
- ignore other text

Another Optional Part of the Answer

Regular Expression

```
/\b(ok(ay)?|surely(es)?)\b/i
```

Possible Subjects

arrr, yes matey

y

Pattern Goals

- ok
- Okay
- sure
- yes
- y
- answer by itself
- ignore other text

Our Finished Pattern

Regular Expression

```
/\b(ok(ay)?|sure|y(es)?)\b/i
```

Possible Subjects

- ok, i will do it
- Ahooy, Okay!!!
- why sure, i can go
- arr, yes matey
- y

Pattern Goals

- ok
- Okay
- sure
- yes
- y
- answer by itself
- ignore other text

Section 2

Confirmative

Sailor Taglines

All sailors have entered their taglines!

Work like a captain, play like a pirate.

Keep calm and say Arr.

Shiver me timbers matey.

Why are pirates pirates? cuz they arr.

Pattern Goals

does not contain numbers

40 characters or less

20 characters or more

Start With a Base Pattern for Matching the Tagline

Regular Expression

```
/[a-z]+/i
```

Subject

```
Work like a captain, play
like a pirate.
```

Not matching whitespace

Pattern Goals

- does not contain numbers
- 40 characters or less
- 20 characters or more

Expanding the Pattern to Include Whitespace

Regular Expression

```
/[a-z\s]+/i
```

Subject

```
Work like a captain, play
like a pirate.
```

we also need to match the comma

Pattern Goals

- does not contain numbers
- 40 characters or less
- 20 characters or more

Including A Comma

Regular Expression

```
/[a-z\s,]+/i
```

Subject

```
Work like a captain, play like a pirate.
```

Pattern Goals

- does not contain numbers
- 40 characters or less
- 20 characters or more

Writing a Shorter Pattern With the NOT Operator

Everything matched with this pattern…

```
/[a-z\s,]+/i
```

…is also matched by this one.

```
/[^\d]+/i
```

The "^" means "not" when placed within a character set

This means "any number"

Subject

Work like a captain, play like a pirate.

so, this pattern means "anything that's not a number"

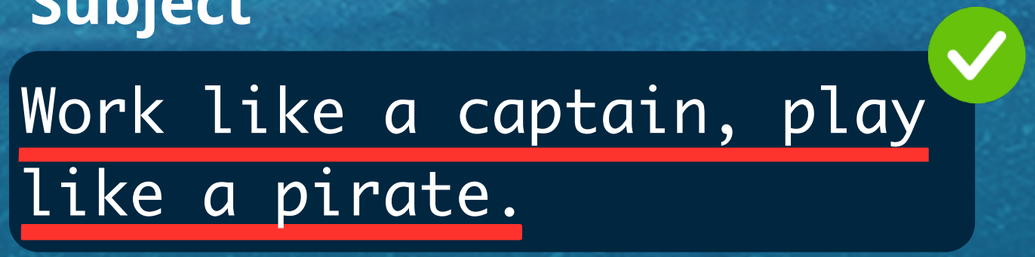
1 Caret Symbol — 2 Different Meanings

^ used to anchor beginning of subject

^ used to negate proceeding pattern

Regular Expression

Subject



Pattern Goals

- ✅ does not contain numbers
- 40 characters or less
- 20 characters or more

`\d` = digit character (number)

`^` = not the following character(s)

Even Shorter With Negated Shorthand Characters

`[^\d]` is the same as `\D`

match every character except numbers

`[^\s]` is the same as `\S`

match every character except whitespace

`[^\w]` is the same as `\W`

match every character except words

NOTE: You don't need a character set when using negated shorthand characters

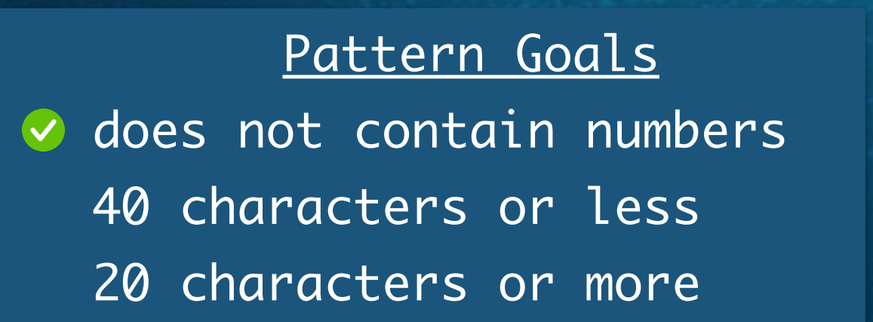
Using the Negated Shorthand Syntax in a Pattern

Regular Expression

```
/^\D+$/
```

Subject

```
Work like a captain, play like a pirate.
```



Problem: We Want to Set Min and Max Characters

Regular Expression

```
/^\D+$/
```

we want a minimum of 20 characters...

Subject

Work like a

Subject

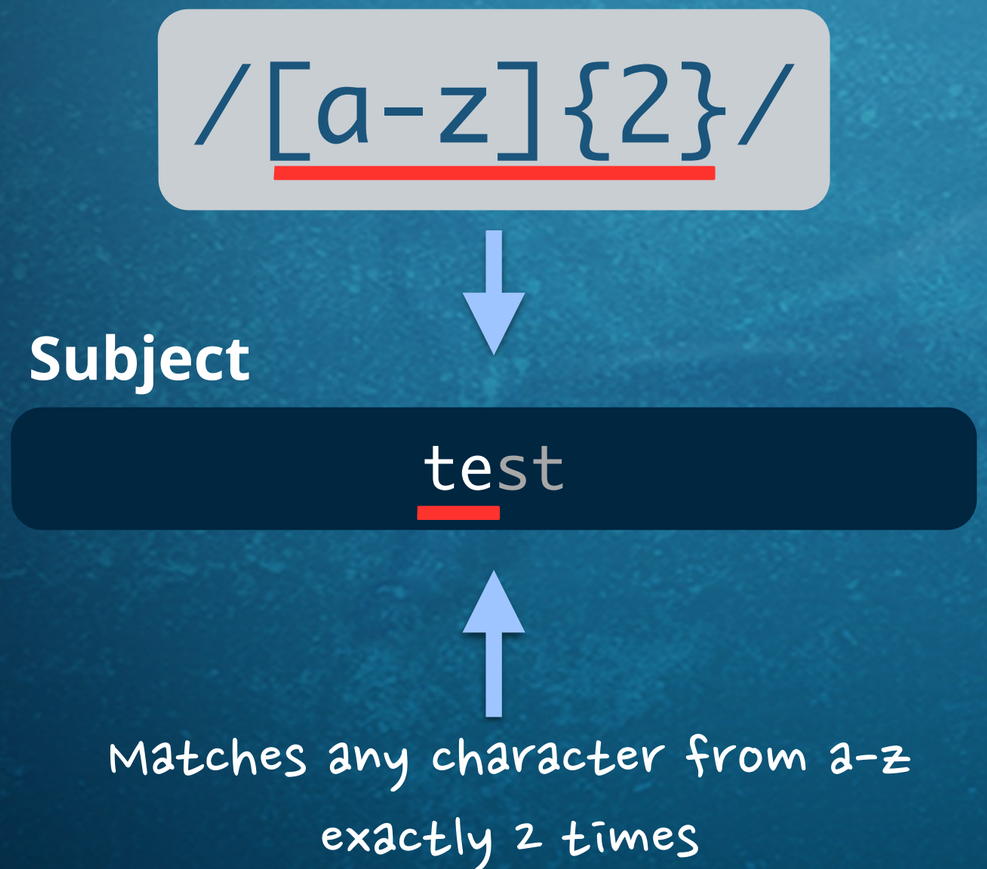
Work like a captain, play like a pirate. Work like a captain, play like a pirate. Work like a captain, play like a pirate.

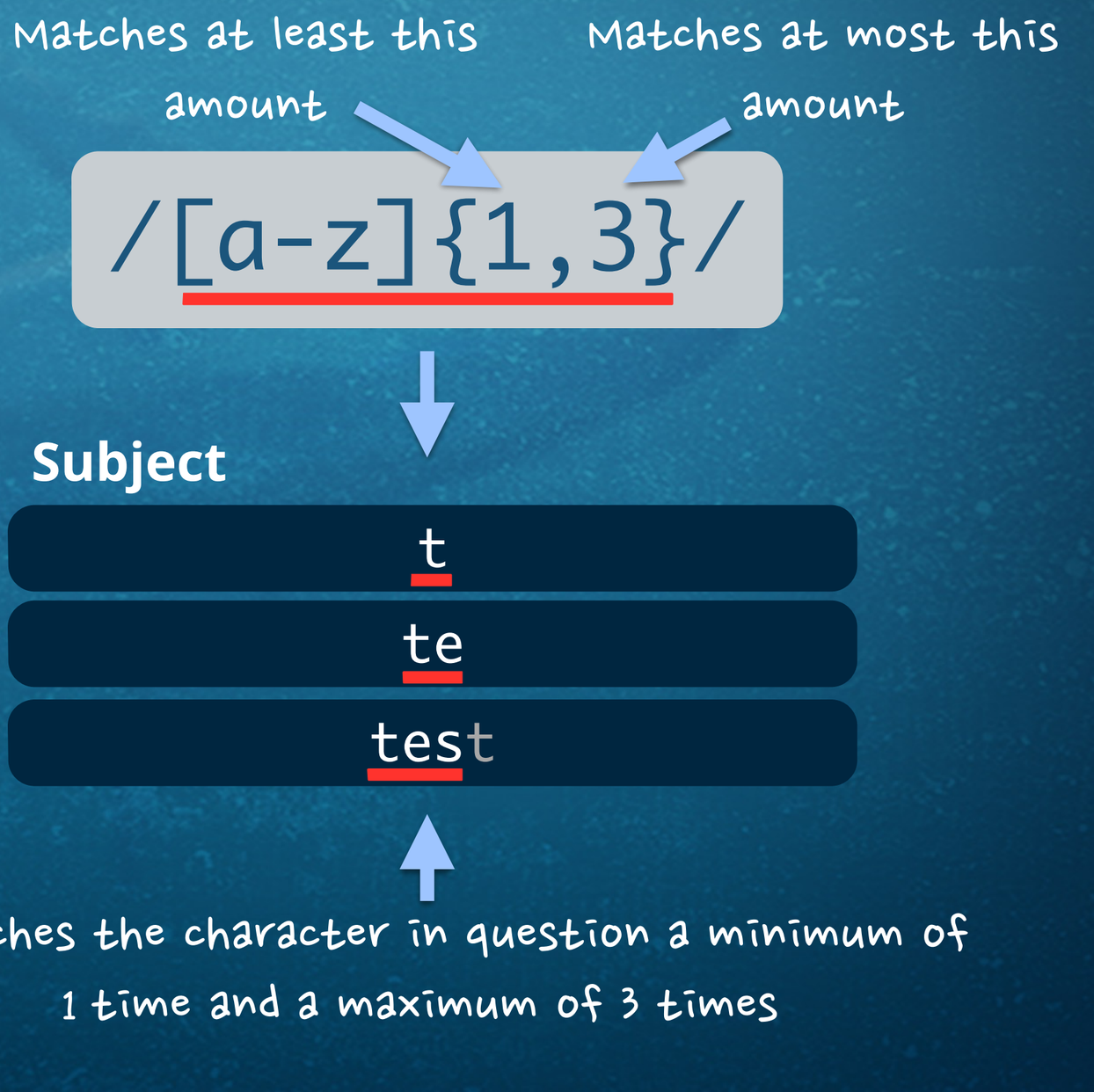
...and a maximum of 40 characters

Pattern Goals

- does not contain numbers
- 40 characters or less
- 20 characters or more

Matching a Specific Number of Times With Interval Expressions





First, Set the Minimum Amount of Characters

Ensure length is greater than or equal to 20

Regular Expression

```
/\D{20,}/i
```

Subject with at least 20 characters is matched

Subject

```
Work like a captain, play like a pirate.
```

Subject

```
Work like a captain.
```

Great! Subject with length less than 20 characters is not matched.

Pattern Goals

- does not contain numbers
- 40 characters or less
- 20 characters or more

Next, Set a Maximum Number of Characters

Regular Expression

```
/^\D{20,40}$/
```

Subject between 20 and 40 characters matched

Subject

Work like a captain, play like a pirate.

Great! Long subject no longer matched.

Work like a captain, play like a pirate. Work like a captain, play like a pirate. Work like a captain, play like a pirate.

Pattern Goals

- does not contain numbers
- 40 characters or less
- 20 characters or more

Contents

Finding More Than Just 1 Match

Single text file

King penguin
Emperor penguin
Wandering albatross
Arctic Tern
Rockhopper Penguin
Weddell seal
Narwhal

we want to find all the birds

Looking at the List of Birds as a Single, Long String

King penguin\nEmperor penguin\nWandering albatross\nArctic Tern\nNarwhal\nRockhopper Penguin\nWeddell seal

\n

Multiline strings are delimited by the newline character

WARNING! Different operating systems might use other newline characters.

BREAKING THE ICE WITH REGULAR EXPRESSIONS

Analyze the Subject String

Subject

```
King penguin
Emperor penguin
Wandering albatross
Arctic Tern
Narwhal
Rockhopper Penguin
Weddell seal
```

Has multiple lines separated by newline characters

Each animal name is 1-2 words

All animal names have mixed casing

Matching the First Animal Name

Regular Expression

```
/penguin/i
```

Match the first occurrence of "penguin"

But we want to match all penguins, not just the first

Subject

```
King penguin
Emperor penguin
Wandering albatross
Arctic Tern
Narwhal
Rockhopper Penguin
Weddell seal
```

Pattern Goals

- ✅ penguin
- albatross
- tern
- full name
- nothing before or after
- ❌ all occurrences

BREAKING THE ICE WITH REGULAR EXPRESSIONS

Global Modifier

Regular Expression

```
/penguin/ig
```

Match "penguin" as many times as possible with global modifier

Subject

```
King penguin
Emperor penguin
Wandering albatross
Arctic Tern
Narwhal
Rockhopper Penguin
Weddell seal
```

Pattern Goals

- ✅ penguin
  - albatross
  - tern
  - full name
  - nothing before or after
- ✅ all occurrences

Matching the Full Name of Just Penguin Animals

Regular Expression

```
/\w+\spenguin/ig
```

Subject

```
King penguin
Emperor penguin
Wandering albatross
Arctic Tern
Narwhal
Rockhopper Penguin
Weddell seal
```

Pattern Goals

- ✅ penguin
  - albatross
  - tern
- ✅ full name
  - nothing before or after
- ✅ all occurrences

BREAKING THE ICE WITH REGULAR EXPRESSIONS

Our Anchors Aren't Working!

Regular Expression

```
/^\w+\spenguin$/ig
```

Subject

```
King penguin
Emperor penguin
Wandering albatross
Arctic Tern
Narwhal
Rockhopper Penguin
Weddell seal
```

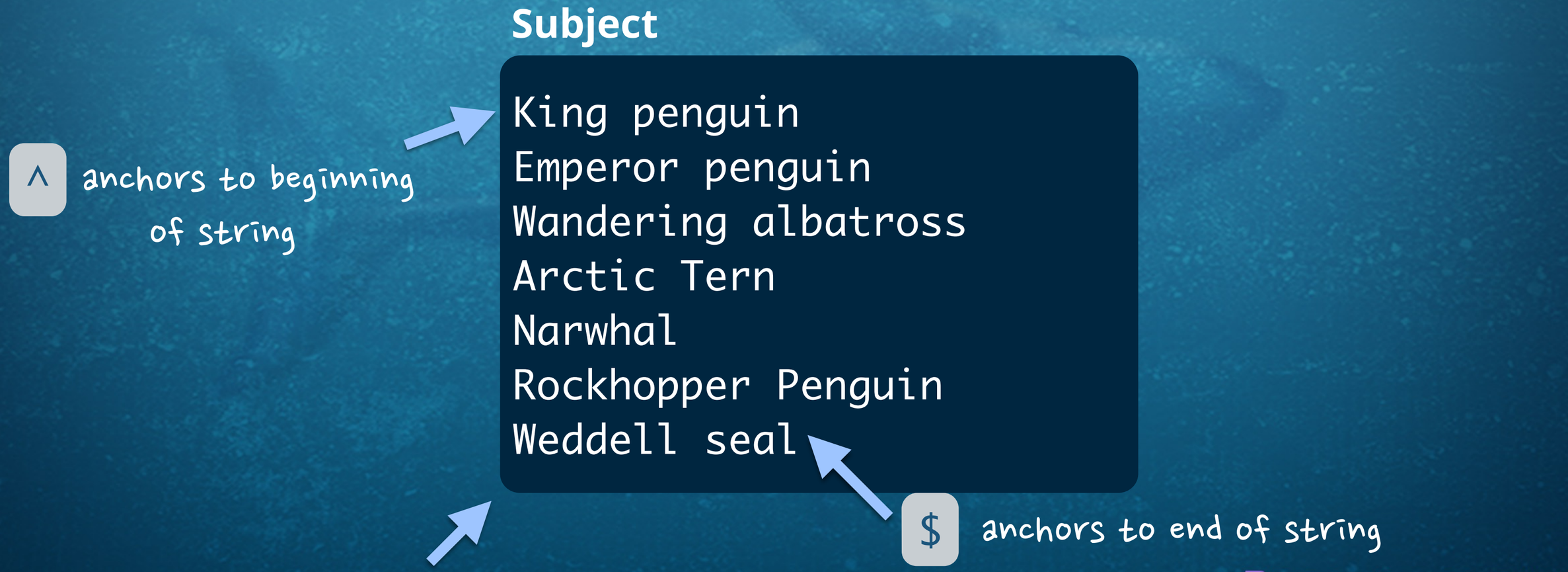
Now nothing is matching!

Pattern Goals

- ✗ penguin
- ✗ albatross
- ✗ tern
- ✗ full name
- ✗ nothing before or after
- ✗ all occurrences

BREAKING THE ICE WITH REGULAR EXPRESSIONS

Anchors in a Normal Subject String



Anchors in a Multiline String

Regular Expression

```
/^\w+\spenguin$/mig
```

Subject

```
King penguin
Emperor penguin
Wandering albatross
Arctic Tern
Narwhal
Rockhopper Penguin
Weddell seal
```

^ anchors to beginning of line and not entire subject

$ anchors to end of line and not entire subject

m = multiline modifier

Pattern Goals

- ✓ penguin
  albatross
  tern
- ✓ full name
- ✓ nothing before or after
- ✓ all occurrences

BREAKING THE ICE WITH REGULAR EXPRESSIONS

Capturing the Full Animal Name

Adds "albatross" to search pattern

Regular Expression

```
/^\w+\s(penguin|albatross)$/mig
```

Subject

King penguin

Emperor penguin

Wandering albatross

Arctic Tern

Narwhal

Rockhopper Penguin

Weddell seal

Pattern Goals

- ✓ penguin
- ✓ albatross
- tern
- ✓ full name
- ✓ nothing before or after
- ✓ all occurrences

BREAKING THE ICE WITH REGULAR EXPRESSIONS

Capturing the Full Animal Name

Adds "tern" to search pattern

Regular Expression

```
/^\w+\s(penguin|albatross|tern)$/mig
```

Subject

King penguin
Emperor penguin
Wandering albatross
Arctic Tern
Narwhal
Rockhopper Penguin
Weddell seal

Pattern Goals

- penguin
- albatross
- tern
- full name
- nothing before or after
- all occurrences

BREAKING THE ICE WITH REGULAR EXPRESSIONS

Contents

Every Crew Has a Home

1 Reindeer Lane, North Pole, AK 99705

120 East 4th Street, Juneau, AK 99705

Building the Number and Street Name

Regular Expression

```
\d+\s[\w\s]+\w{4,6},\s
```

Some numbers and 2 words followed by a comma and a space

Possible Subjects

```
1 Reindeer Lane,
```

Number and street name fully matched

Building the City Pattern

Regular Expression

```
[\w\s]+,\s
```

city name of any number of characters followed by a comma and a space

Possible Subjects

city is completely matched

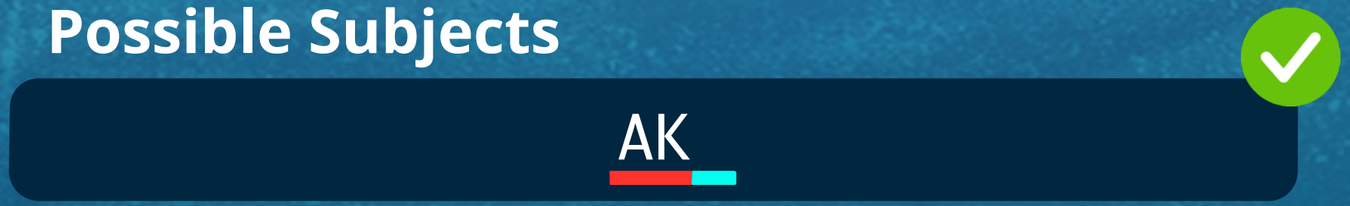
Building the State Pattern

Regular Expression

```
\w{2}\s
```

2-letter state followed by a space

Possible Subjects



State is completely matched

Building the Zip Pattern

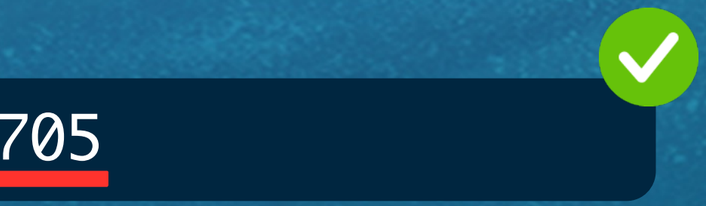
Regular Expression

```
\d{5}
```

5-digit zip

Possible Subjects

99705



Zip code is completely matched

Our Fully Matching Pattern

Regular Expression

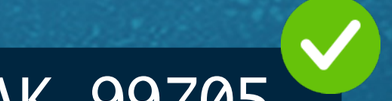
```
/^\d+\s[\w\s]+\w{4,6},\s[\w\s]+,\s\w{2}\s\d{5}$/ig
```

Anchor to beginning of subject

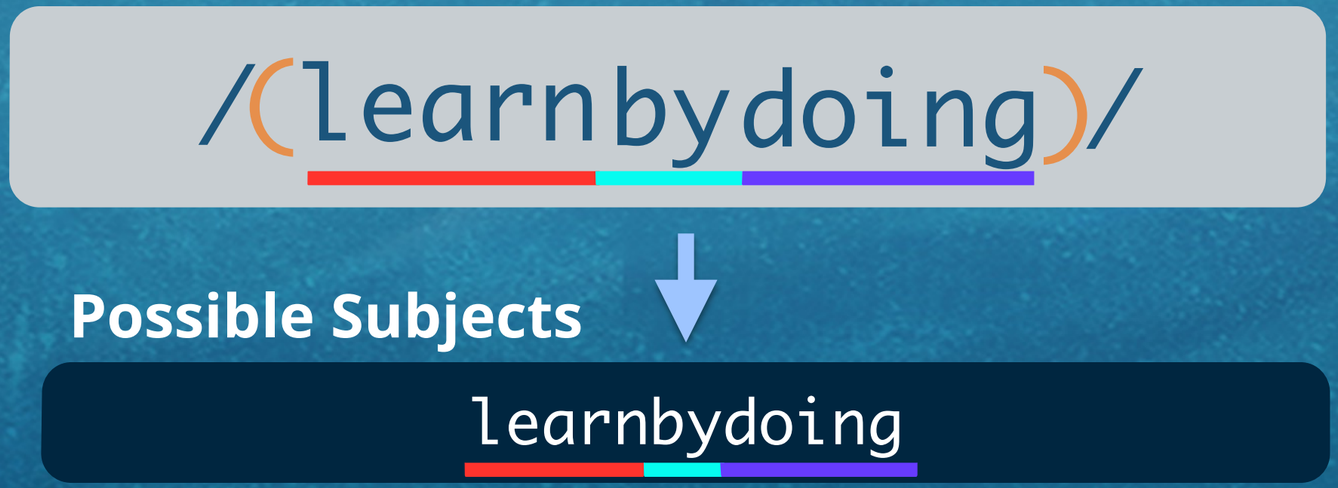
Anchor to end of subject

Possible Subjects

1 Reindeer Lane, North Pole, AK 99705



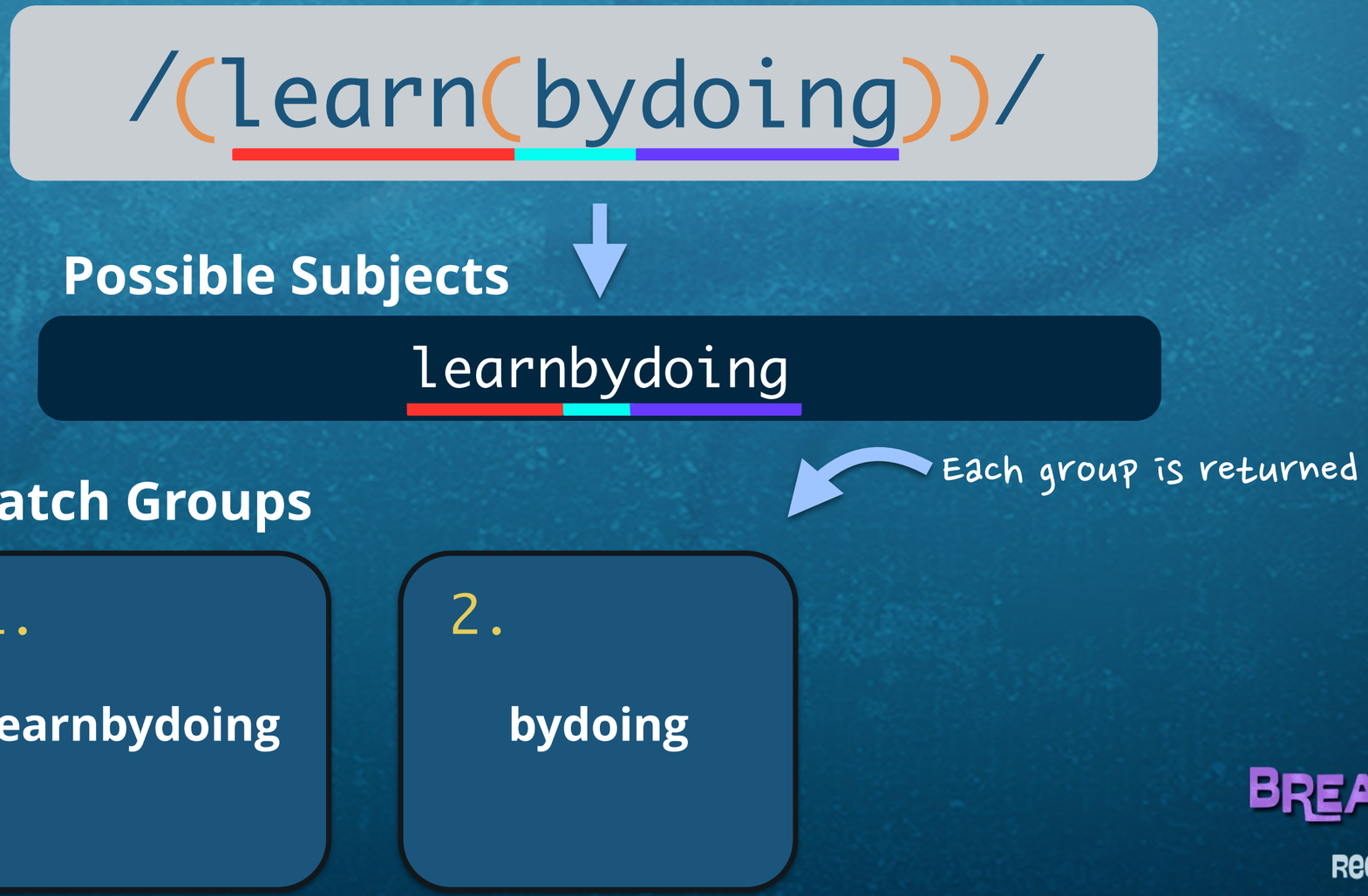
Matching Groups



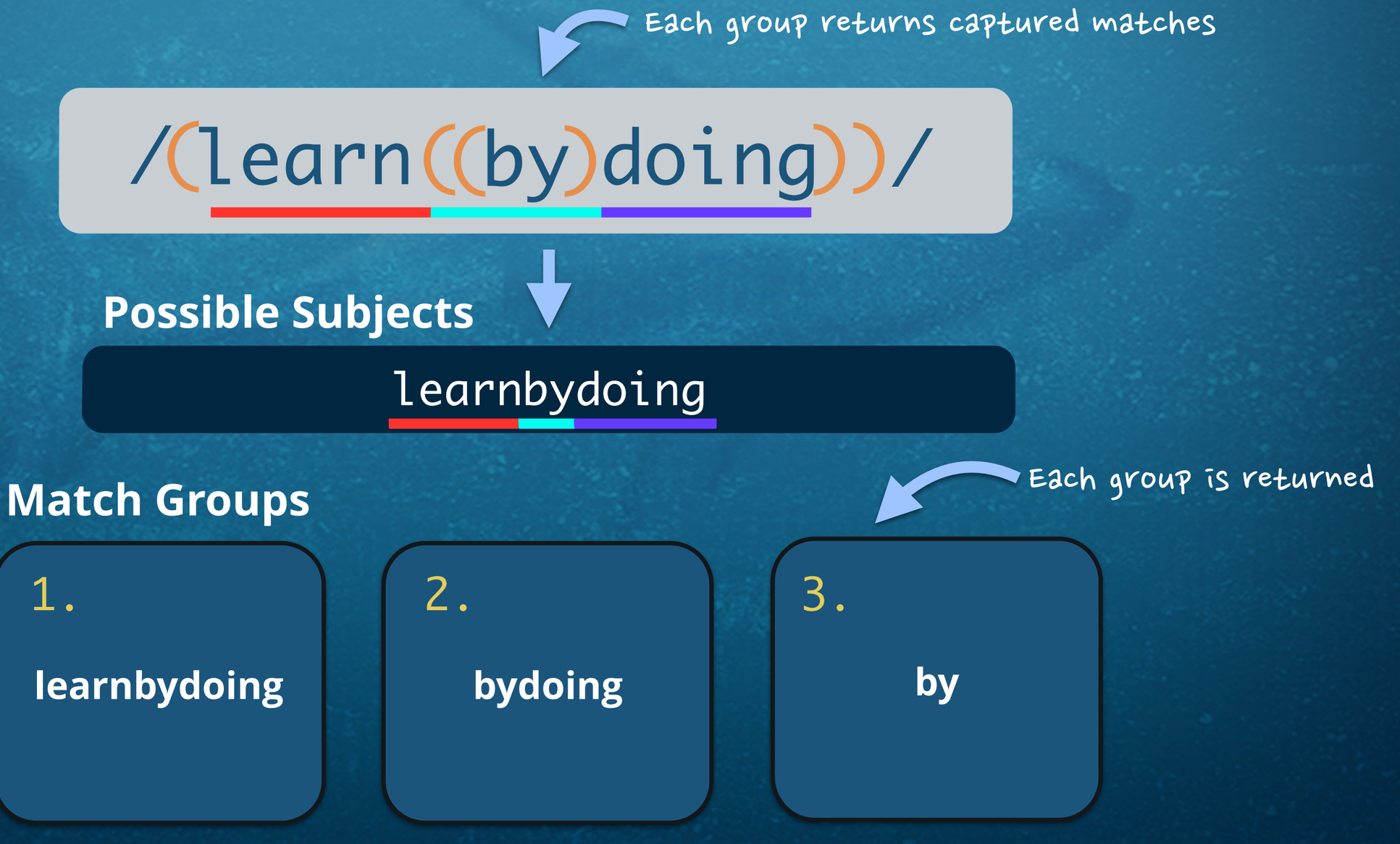
Match Groups

1. **learnbydoing**

Matching Groups



Matching Groups



Matching Groups — Street

Regular Expression

only return group of first section

```
/^(\d+\s[\w\s]+\w{4,6}),\s[\w\s]+,\s\w{2}\s\d{5}$/i
```

Possible Subjects

```
1 Reindeer Lane, North Pole, AK 99705
```

Match Groups

1. **1 Reindeer Lane**

Matching Groups — City

First match group for house number and street

Second match group for city

Regular Expression

```
/^(\d+\s[\w\s]+\w{4,6}),\s([\w\s]+),\s\w{2}\s\d{5}$/i
```

Possible Subjects

1 Reindeer Lane, North Pole, AK 99705

Match Groups

1. 1 Reindeer Lane
2. North Pole

BREAKING THE ICE WITH REGULAR EXPRESSIONS

Matching Groups — State

Third match group for 2-letter state

Regular Expression

```
/^(\d+\s[\w\s]+\w{4,6}),\s([\w\s]+),\s(\w{2})\s\d{5}$/i
```

Possible Subjects

1 Reindeer Lane, North Pole, AK 99705

Match Groups

1. 1 Reindeer Lane
2. North Pole
3. AK

BREAKING THE ICE WITH REGULAR EXPRESSIONS

Matching Groups — Zip

Final match group for zip code

Regular Expression

```
/^(\d+\s[\w\s]+\w{4,6}),\s([\w\s]+),\s(\w{2})\s(\d{5})$/i
```

Possible Subjects

1 Reindeer Lane, North Pole, AK 99705

Match Groups

1.	2.	3.	4.
1 Reindeer Lane	North Pole	AK	99705

Problem: Need to Restrict Potential Streets

Regular Expression

Should only match “street” or “lane”

```
/^(\d+\s[\w\s]+\w{4,6}),\s([\w\s]+),\s(\w{2})\s(\d{5})$/i
```

Possible Subjects

1 Reindeer Lane, North Pole, AK 99705

15 Discovery Road, Juneau, AK 99705

we only want to match “street” or “lane”

Solution: Restricting the Street

Regular Expression

Now we're only looking for "street" or "lane"

```
/^(\d+\s[\w\s]+(street|lane)),\s([\w\s]+),\s(\w{2})\s(\d{5})$/i
```

Possible Subjects

1 Reindeer Lane, North Pole, AK 99705

15 Discovery Road, Juneau, AK 99705

Now we do not match on "road"

Problem: Group Returns Unwanted Result

Group used for evaluation purposes

returns unwanted value

Regular Expression

```
/^(\d+\s[\w\s]+(street|lane)),\s([\w\s]+),\s(\w{2})\s(\d{5})$/i
```

Possible Subjects

```
1 Reindeer Lane, North Pole, AK 99705
```

Match Groups

1. **1 Reindeer Lane**
2. **Lane**
3. **North Pole**
4. **AK**
5. **99705**

we don't want to capture just the type of street!

Making the Street Type a Non-capturing Group

Non-capturing group

Regular Expression

```
/^(\d+\s[\w\s]+(?:street|lane)),\s([\w\s]+),\s(\w{2})\s(\d{5})$/i
```

Possible Subjects

1 Reindeer Lane, North Pole, AK 99705

Now we have the correct number of groups

Match Groups

1.	2.	3.	4.
1 Reindeer Lane	North Pole	AK	99705