Security

Validation and sanitization

Input Validation

- In a web app, validation should be carried out on every form element to guarantee that the input is correct.
- Processing incorrect input values can make your application give unpredictable results.
- Risks include
  - SQL Injection
  - Cross-site scripting
  - Buffer overflows
  - Leakage of site internal design through error messages
- Validation for security should always be carried out on the **server** side
  - HTML form attributes and JavaScript validation can be an aid to users but are useless for security

Whitelists vs Blacklists

- **Blacklist validation** is testing an input against a set of unacceptable values
  - Default policy is "accept"
- **Whitelist validation** is testing an input against a set of possible correct values
  - Default policy is "reject"
- Whitelist validation is generally best for security, but tends to be more restrictive and may conflict with user-friendliness
- e.g. EC2 security group *whitelists* specific protocols and ports; all others are blocked

Validation in Hapi

- Modern frameworks have extensive features to support data validation and sanitisation.
- e.g.
  - **joi** for input validation
  - **disinfect** for sanitisation

Regular Expressions

- `Joi.string().regex()` checks the value provided is a string matching a particular **regular expression**
- Example

```
Joi.string().regex(/^[A-Z][a-zA-Z'-]{3,}$/)
```

[ ] specifies alternative options
{*a*,*b*} at least *a*; at most *b*

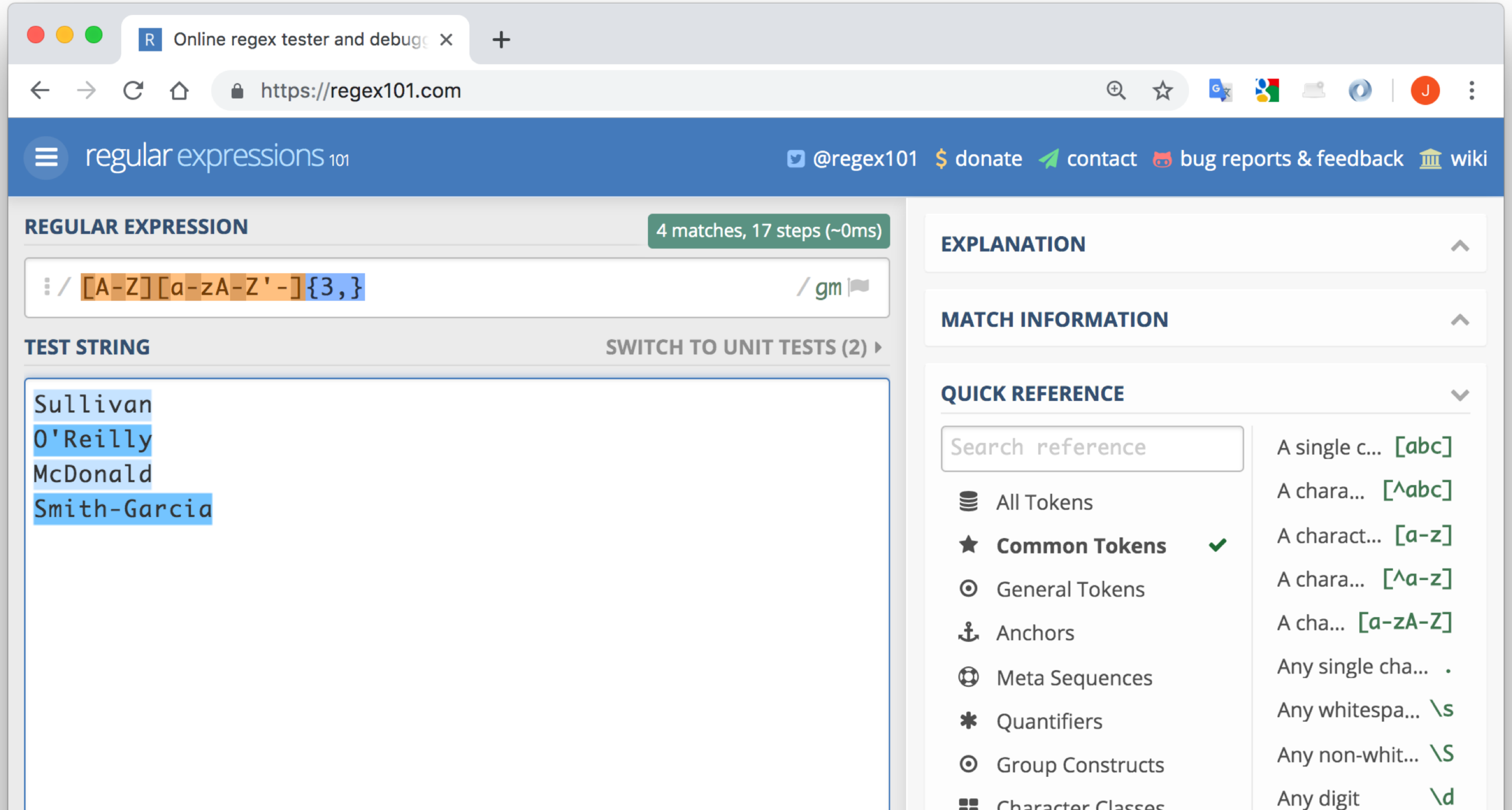
This pattern checks if the input string starts with an upper case letter and has a minimum of 3 additional characters in total, each of which must be alphabetic, an apostrophe or a hyphen

Regular Expressions

- A **regular expression (regex)** is a sequence of characters that specify a pattern to be matched.
- Very powerful concept as many computing applications involve pattern matching – for example:
  - Search engines
  - Natural (human) language processing
  - Intrusion detection
  - Computer forensics
  - Intelligence gathering (e.g. NSA…)
- A full treatment of regular expressions is beyond the scope of this module
  - Several textbooks just on regular expressions + many online resources

Regular Expressions

- A useful online regex tester: https://regex101.com/ (others exist as well)



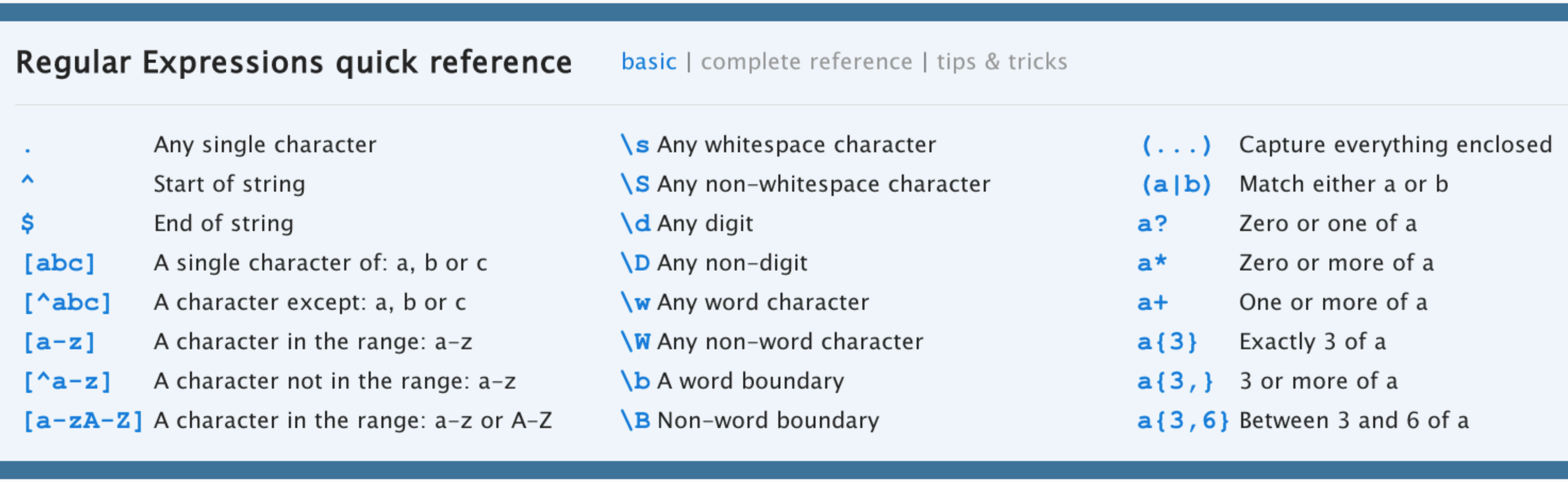
REGULAR EXPRESSION — 4 matches, 17 steps (~0ms)

```
/ [A-Z][a-zA-Z'-]{3,} / gm
```

TEST STRING

```
Sullivan
O'Reilly
McDonald
Smith-Garcia
```

RegEx Quick Reference



Regular Expressions quick reference basic | complete reference | tips & tricks

`.`	Any single character	`\s`	Any whitespace character	`(...)`	Capture everything enclosed
`^`	Start of string	`\S`	Any non-whitespace character	`(a|b)`	Match either a or b
`$`	End of string	`\d`	Any digit	`a?`	Zero or one of a
`[abc]`	A single character of: a, b or c	`\D`	Any non-digit	`a*`	Zero or more of a
`[^abc]`	A character except: a, b or c	`\w`	Any word character	`a+`	One or more of a
`[a-z]`	A character in the range: a-z	`\W`	Any non-word character	`a{3}`	Exactly 3 of a
`[^a-z]`	A character not in the range: a-z	`\b`	A word boundary	`a{3,}`	3 or more of a
`[a-zA-Z]`	A character in the range: a-z or A-Z	`\B`	Non-word boundary	`a{3,6}`	Between 3 and 6 of a

Sanitization vs Validation

- Sanitization tries to achieve similar objectives to validation, but in a somewhat different way
- Output of Validation is usually binary:
  - Valid: input is accepted
  - Invalid: input is not accepted
- Output of Sanitisation is a "cleaned" version of input:
  - Output is filtered input (e.g. certain characters removed or re-encoded)

- Several npm sanitization modules exist - see **disinfect** and **sanitize** for example

Sanitization examples

Sanitizer	Description	Example input	Example output
Email address	Remove all characters except letters, digits and !#$%&'*+-=?^_`{|}~@.[].	<jb@gmail.com>,	jb@gmail.com
Integer	Remove all characters except digits,+ and -	13a	13
HTML entities	Convert reserved characters in HTML to corresponding entities	< > &	&lt; &gr; &amp;
Restrict HTML	Whitelist certain tags: <a href="URL"> <em> <cite> <i> <strong> <b> <sub> <sup> <ul> <ol> <li>	<script>alert(1) </script>	alert(1)