Security

Validation and sanitisation

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

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-zA-Z0-9]{3,}$/)
```

[ ]	specifies alternative options
+	indicates one or more

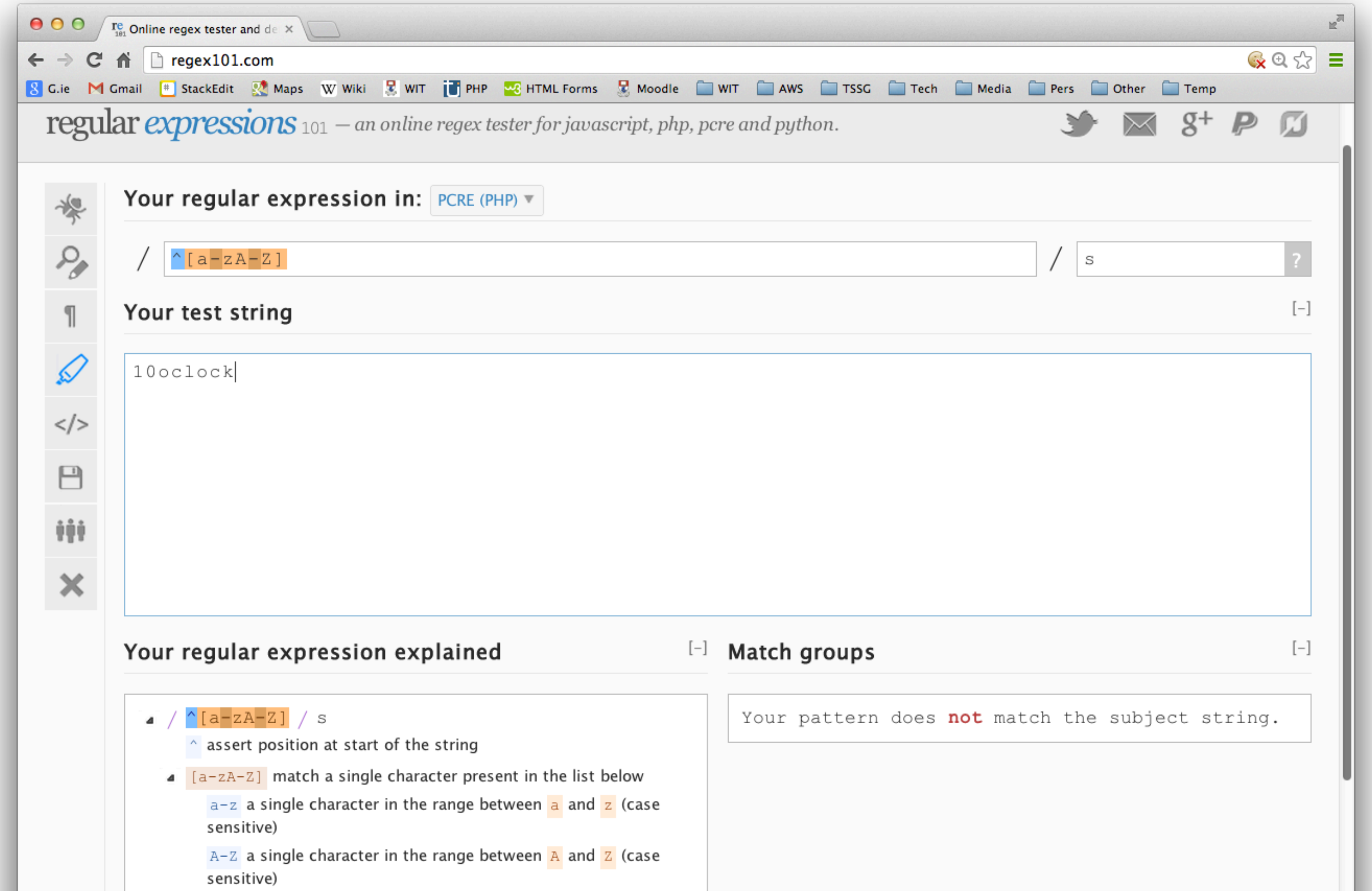
This pattern checks if the input string has a minimum of 3 characters and only contains alphanumeric characters.

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: http://regex101.com/ (others exists as well)


Online regex tester and de
regex101.com
G.ie
Gmail
StackEdit
Maps
Wiki
WIT
PHP
HTML Forms
Moodle
WIT
AWS
TSSG
Tech
Media
Pers
Other
Temp
regular expressions 101 — an online regex tester for javascript, php, pcre and python.

Your regular expression in: PCRE (PHP)

/ `^[a-zA-Z]` / s

Your test string

```
10oclock
```

Your regular expression explained

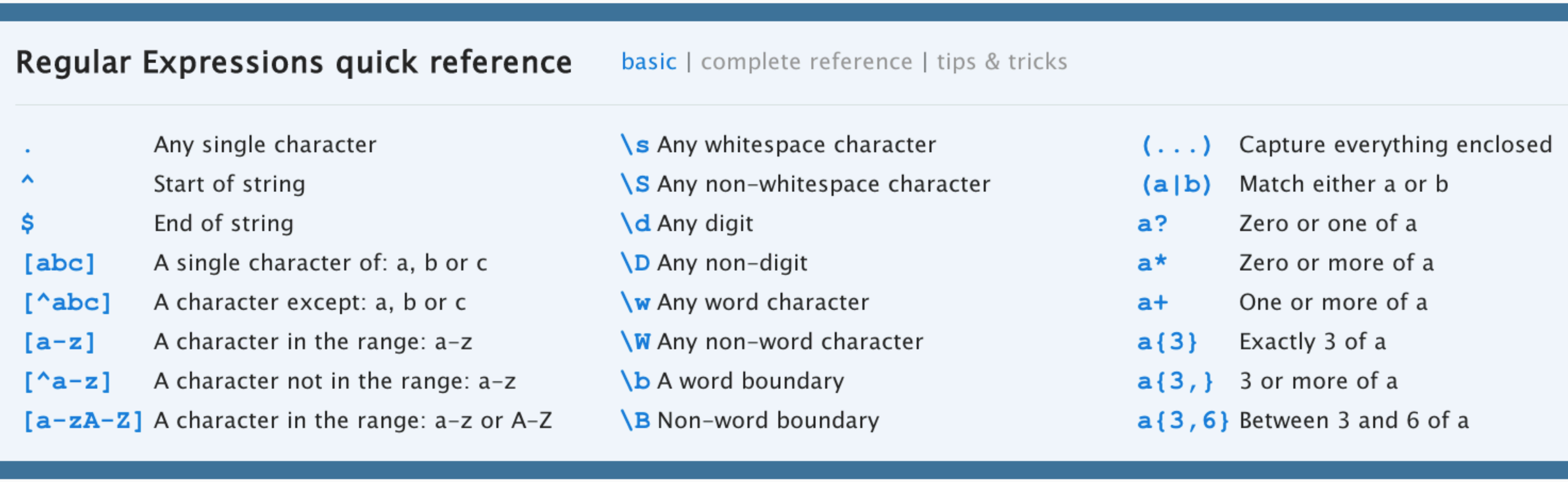
```
/ ^[a-zA-Z] / s
```

- `^` assert position at start of the string
- `[a-zA-Z]` match a single character present in the list below
  - `a-z` a single character in the range between `a` and `z` (case sensitive)
  - `A-Z` a single character in the range between `A` and `Z` (case sensitive)

Match groups

```
Your pattern does not match the subject string.
```

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

disinfect

- Can be based on route query, payload, and params
- Options:
  - **deleteEmpty** - remove empty query or payload keys.
  - **deleteWhitespace** - remove whitespace query, payload, or params keys.
  - **disinfectQuery** - sanitize query strings.
  - **disinfectParams** - sanitize url params.
  - **disinfectPayload** - sanitize payload.
  - **genericSanitizer** - custom synchronous function to do the sanitization of query, payload, and params.
  - **querySanitizer** - custom synchronous function to do the sanitization of query strings.
  - **paramsSanitizer** - custom synchronous function to do the sanitization of url params.
  - **payloadSanitizer** - custom synchronous function to do the sanitization of payload.