Response Validations

(Adding Tests)

- Status code
- Headers
- Cookies
- Response time
- Response body

Chai Assertion Library

```
pm.test("Test Name", function()
                    {
                        // assertion;
                    }
      );
```

```
pm.test("Test Name", () =>
                     {
                       // assertion;
                     }
      );
```

Testing status codes

Test for the response status code:

```
pm.test("Status code is 200", () => {
  pm.response.to.have.status(200);
});
```

If you want to test for the status code being one of a set, include them all in an array and use *one of*

```
pm.test("Successful POST request", () => {
pm.expect(pm.response.code).to.be.oneOf([201,202]);
});
```

Check the status code text:

```
pm.test("Status code name has string", () => {
  pm.response.to.have.status("Created");
});
```

Testing headers

Check that a response header is present:

```
pm.test("Content-Type header is present", () => {
  pm.response.to.have.header("Content-Type");
});
```

Test for a response header having a particular value:

```
pm.test("Content-Type header is application/json", () => {
  pm.expect(pm.response.headers.get('Content-
Type')).to.eql('application/json; charset=utf-8');
});
```

Testing cookies

Test if a cookie is present in the response:

```
pm.test("Cookie 'language' is present", () => {
  pm.expect(pm.cookies.has('language')).to.be.true;
});
```

Test for a particular cookie value:

```
pm.test("Cookie language has value 1", () => {
  pm.expect(pm.cookies.get('language')).to.eql('en-gb');
});
```

Testing response times

Test for the response time to be within a specified range:

```
pm.test("Response time is less than 200ms", () => {
  pm.expect(pm.response.responseTime).to.be.below(200);
});
```

Asserting a value type

Test the type of any part of the response:

```
{
    "id": 1,
    "name": "John",
    "location": "india",
    "phone": "1234567890",
    "courses": [
        "Java",
        "Selenium"
    ]
}
```

```
const jsonData = pm.response.json();
pm.test("Test data type of the response", () => {
  pm.expect(jsonData).to.be.an("object");
  pm.expect(jsonData.name).to.be.a("string");
  pm.expect(jsonData.id).to.be.a("number");
  pm.expect(jsonData.courses).to.be.an("array");
});
```

Asserting array properties

Check if an array is empty and if it contains particular items:

```
{
    "id": 1,
    "name": "John",
    "location": "india",
    "phone": "1234567890",
    "courses": [
        "Java",
        "Selenium"
    ]
}
```

```
pm.test("Test array properties", () => {
    //courses includes "Java"
  pm.expect(jsonData.courses).to.include("Java");
     //courses array must include all listed
  pm.expect(jsonData.courses)
     .to.have.members(["Java", "Selenium"]);
});
```

Validating JSON fields in Response

```
{
    "id": 1,
    "name": "John",
    "location": "india",
    "phone": "1234567890",
    "courses": [
        "Java",
        "Selenium"
    ]
}
```

```
pm.test("value of location field is India",()=> {
    var jsonData = pm.response.json();
    pm.expect(jsonData.id).to.eql(1);
    pm.expect(jsonData.name).to.eql("John");
    pm.expect(jsonData.location).to.eql("india");
    pm.expect(jsonData.phone).to.eql("1234567890");
    pm.expect(jsonData.courses[0]).to.eql("Java");
    pm.expect(jsonData.courses[1]).to.eql("Selenium");

});
```

Validating JSON Schema

Response

```
{
    "id": 1,
    "name": "John",
    "location": "india",
    "phone": "1234567890",
    "courses": [
        "Java",
        "Selenium"
    ]
}
```

JSON schema

```
var schema={
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "id": {
```

```
      "type": "integer"
    },
    "name": {
      "type": "string"
    },
    "location": {
      "type": "string"
    },
    "phone": {
      "type": "string"
    },
    "courses": {
      "type": "array",
      "items": [
        {
          "type": "string"
        },
        {
          "type": "string"
        }
      ]
    }
  },
  "required": [
    "id",
```

```
    "name",
    "location",
    "phone",
    "courses"
  ]
}
```

JSON schema Validation

```
pm.test('Schema is valid', function() {
  pm.expect(tv4.validate(jsonData, schema)).to.be.true;
});
```