Postman Scripting Functions & Methods

1. pm.variables.get('varName') / pm.environment.get('varName') / pm.globals.get('varName')

Purpose: Retrieve a stored variable from the appropriate scope (variable, environment, or global).

```
let token = pm.environment.get('authToken');
```

Great for chaining requests or controlling flow based on environment values.

---

2. pm.variables.set('varName', value) / pm.environment.set('varName', value)

Purpose: Store or update a variable (local, environment, or global).

```
pm.environment.set('userId', pm.response.json().id);
```

Essential for dynamic request chaining and reusable workflows.

---

3. pm.sendRequest(requestObject, callback)

Purpose: Execute an additional HTTP request inside a script (useful for token refresh or external calls).

```
pm.sendRequest({
  url: pm.environment.get('baseUrl') + '/auth/refresh',
  method: 'POST',
  header: {'Content-Type': 'application/json'},
  body: { mode: 'raw', raw: JSON.stringify({ refreshToken }) }
}, (err, res) => {
  pm.environment.set('authToken', res.json().token);
});
```

---

4. pm.test(name, function)

Purpose: Define and group test assertions.

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

---

5. pm.expect(expression)

Purpose: Create assertions using Chai.js-style expectations.

```
pm.test("Price is positive", () => {
  let price = pm.response.json().price;
  pm.expect(price).to.be.above(0);
});
```

---

6. pm.response.to.have.status(code)

Purpose: Assert the HTTP response status code.

```
pm.test("Created successfully", () => {
  pm.response.to.have.status(201);
});
```

---

7. pm.response.to.be.json / .xml / .html

Purpose: Validate the format of the response body.

```
pm.test("Response is JSON", () => {
  pm.response.to.be.json;
});
```

---

8. pm.response.json() / pm.response.text()

Purpose: Extract and parse response data.

```
let data = pm.response.json();
let user = data.user;
```

---

9. pm.expect(string).to.match(regex)

Purpose: Perform regex-based assertion checks.

```
pm.test("Email format is valid", () => {
  let email = pm.response.json().email;
  pm.expect(email).to.match(/^[^\s@]+@[^\s@]+\.[^\s@]+$/);
});
```

---

10. postman.setNextRequest('requestName')

Purpose: Control the execution order of requests in a collection run.

```
if (pm.response.code !== 200) {
  postman.setNextRequest('Retry Login');
}
```

---

11. console.log(...)

Purpose: Debug and inspect values in the Postman Console.

```
console.log("User data:", pm.response.json());
```
