Gherkin: Friend or Foe? Re-Evaluating Its Role in Modern Development

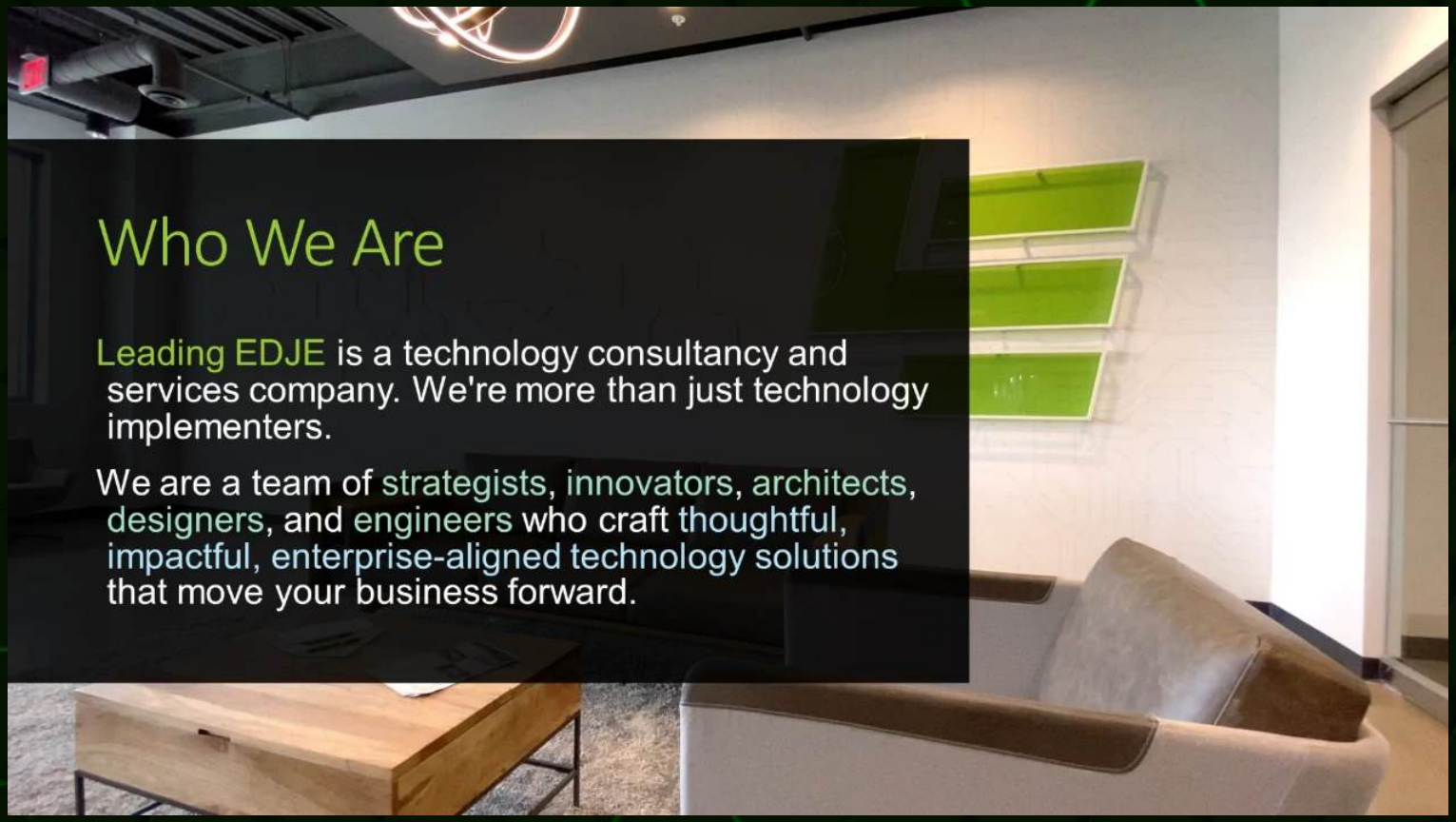
Bob Fornal

Entrepreneur
Code-squid provides solid, in-depth frontend training that is supported with real-world code projects. Blessed husband and proud father of two.

https://code-squid.com

Senior Solutions Developer
Leading EDJE, Inc.
Passionate about learning, testing, mentoring, speaking, and personal growth.

https://leadingedje.com


Who We Are
Leading EDJE is a technology consultancy and services company. We're more than just technology implementers.
We are a team of strategists, innovators, architects, designers, and engineers who craft thoughtful, impactful, enterprise-aligned technology solutions that move your business forward.

What is Gherkin?

Gherkin is a structured, human-readable language used in software development, particularly within the context of Behavior-Driven Development (BDD).

- It is designed to express test scenarios and user stories in a way that's clear, concise, and understandable by both technical and non-technical stakeholders.
- It uses keywords like Given, When, Then, And, and But to describe the context, actions, and expected results of a scenario.

In essence, Gherkin ...

- Bridges the gap between business requirements and technical implementation.
- It provides a common language for expressing and documenting software behavior.

```
Feature: Effective Use of Gherkin in Automated
  Testing

  Background:
    Given Gherkin is a domain-specific language used
      for behavior-driven development (BDD)
    And it is intended to bridge communication
      between technical and non-technical stakeholders
```

What is Gherkin?

Definition

- Structured Language
- Human Readable
- Used in BDD
- Linked to Automation

Benefits

- Improved Communication
- Clear documentation
- Increased efficiency
- Easier to understand

```
Feature: Effective Use of Gherkin in Automated
  Testing (continued)

  Scenario: Understanding the dual nature of Gherkin
    in development
    Given Gherkin uses structured keywords such as
      "Feature", "Scenario", "Given", "When", "Then",
      "And", and "But"
    When it is used well, it enhances communication
      and collaboration
    Then it improves the readability and
      understanding of software specifications
```

When is Gherkin Used

Card Type	Gherkin Use	Impact
Epics	NO	
Stories	**Behavior Driven Development**	Too General
Tasks	(often)	Specific
Sub-Tasks	(often)	Too Specific (or not used)
Bugs	(somtimes)	**Good Fit**
Custom	?	?

Gherkin: Pros

Benefits of using Gherkin:

- Improved communication: Gherkin enables better collaboration between different stakeholders.
- Clear documentation: Gherkin specifications serve as documentation for the software's behavior.
- Increased efficiency: Automated tests based on Gherkin specifications can be run quickly and consistently.
- Easier to understand: The plain English syntax makes it easier for non-technical people to understand the software's functionality.

```
Feature: Effective Use of Gherkin in Automated
  Testing (continued)

  Scenario: Identifying potential pitfalls of Gherkin
    Given some implementations of Gherkin can
      sabotage the development process
    When there are unclear feature descriptions
    And scenarios become overly complex
    And there is misalignment between tests and
      actual business requirements
    Then Gherkin can hinder automated testing and
      development progress
```

Gherkin: Cons

Pitfalls of Gherkin

1. Extra Layer of Complexity: Need to create feature files that then need to be linked to step definitions.
2. Two Truths: The scenarios often don't lend themselves well to automation. Some steps need more thought and detail to become effective automation tests.
3. Harder to Read Tests: As feature files grow, the scenarios become more generic. Tests become a tangled mess of steps and lose clarity.
4. Slower Test Execution: An extra execution layer simply makes tests run slower.
5. Over-Specification: In an attempt to cover every angle, teams end up adding way too much detail to the scenarios.

```
Feature: Effective Use of Gherkin in Automated
  Testing (continued)

  Scenario: Using Gherkin properly
    Given there are common mistakes in implementing
      Gherkin
    When strategies are put in place to mitigate
      issues
    Then Gherkin can be used effectively to support
      and enhance testing processes
    And development efforts can become more efficient
```

Consider: Code First Automation

A code-first approach to test automation ...

1. Efficiency & Speed: One of the biggest advantages of a code-first approach is efficiency.

```
Feature: Effective Use of Gherkin in Automated
  Testing (continued)

  Scenario: Gaining practical insights into Gherkin
    usage
    Given attendees are seeking practical tips for
      using Gherkin
    When they are provided with recommended practices
      for optimizing Gherkin
    Then they can ensure Gherkin is a powerful tool
      to aid in the development process
```

Simple Login Testing

login.feature

login.js

code-first.js

```
Feature: User Login

  Background:
    Given the user is on the login page

  Scenario: Successful login with valid credentials
    Given the user enters a valid username and password
    When the user clicks the loging button
    Then the user should be redirected to the dashboard
    And a welcome message should be displayed
```

Consider: Code First Automation

A code-first approach to test automation ...

1. Efficiency & Speed: One of the biggest advantages of a code-first approach is efficiency.
2. More Suitable for API Testing: Cleaner, more direct, and faster.

```
Feature: Effective Use of Gherkin in Automated
  Testing (continued)

  Scenario: Gaining practical insights into Gherkin
    usage
    Given attendees are seeking practical tips for
      using Gherkin
    When they are provided with recommended practices
      for optimizing Gherkin
    Then they can ensure Gherkin is a powerful tool
      to aid in the development process
```

Simple API Testing

api.feature

api.js

code-first.js

```
Feature: Manage Fruit via API

  Background:
    Given the API base URL is "https://example.com/api/v1"

  Scenario: Retrieve a list of Fruit
    When the system sends a GET request to "/fruits"
    Then the response status code should be 200
    And the response should contain a list of fruit
    And the list should include "Apple"

  Scenario: Add a new Fruit
    Given the request payload is:
      """
      {
        "name": "Banana",
        "id": 101
      }
      """
    When the system sends a POST request to "/fruits" with the payload
    Then the response status code should be 201
    And the response should contain:
```

Consider: Code First Automation

A code-first approach to test automation ...

1. Efficiency & Speed: One of the biggest advantages of a code-first approach is efficiency.
2. More Suitable for API Testing: Cleaner, more direct, and faster.
3. Easier Debugging & Failure Analysis: Identifying the failing step in the Feature File is the first action to debug where a failure might be. Only then can you begin start to isolate and identify what the root cause might be.

```
Feature: Effective Use of Gherkin in Automated
  Testing (continued)

  Scenario: Gaining practical insights into Gherkin
    usage
    Given attendees are seeking practical tips for
      using Gherkin
    When they are provided with recommended practices
      for optimizing Gherkin
    Then they can ensure Gherkin is a powerful tool
      to aid in the development process
```

Lightweight Test Documentation

Current ...

- Clunky user stories that get built by business
- Eventually they are passed to the team for refinement

Possible ...

- Effective test notes should be added to the story
- As opposed to Gherkin Scenarios

Valid Login

- Verify that a registered user can login with valid credentials
- Expand test coverage to include different user types (i.e. admin, user, read-only)
- Use existing credentials from test.ts for consistency
- Test against multiple devices, browsers, and screens

Invalid Password Handling

- Verify that an appropriate error message displays when the user enters an incorrect password
- (consider) Mock the authentication endpoint to always return an error response

API-Level Testing

- Isolate the auth endpoint and create API tests to validate login behavior under various conditions

When is Gherkin Used

Card Type	Gherkin First	Impact	Alternative
Epics	NO		
Stories	**Behavior Driven Development**	Too General	Effective Test Notes
Tasks	(often)	Specific	Effective Code First and Test Notes
Sub-Tasks	(often)	Too Specific (or not used)	Effective Code First and Test Notes
Bugs	(somtimes)	**Good Fit**	**Gherkin**
Custom	?	?	?

How is Gherkin Used

Organization uses ...

- Behavior-Driven Development (BDD)
- Story Test-Driven Development (Feature, SDD)
- Test-Driven Development (TDD)
- Acceptance Test-Driven Development (ATDD)

Gherkin implementations ...

- Not used or used minimally.
- Used correctly (unicorn).
- Tripping hazard (everywhere).

Developer Notes and Test Documentation ...

- Often shoved into Comments.
- Occasionally added by 3-Amigos


TIME FOR QUESTIONS