



SAUCECON 19

WELCOME TO THE WORKSHOP

# Best Practices for Automated Testing

# AGENDA

After completing this workshop, you will be able to:



Code top quality tests



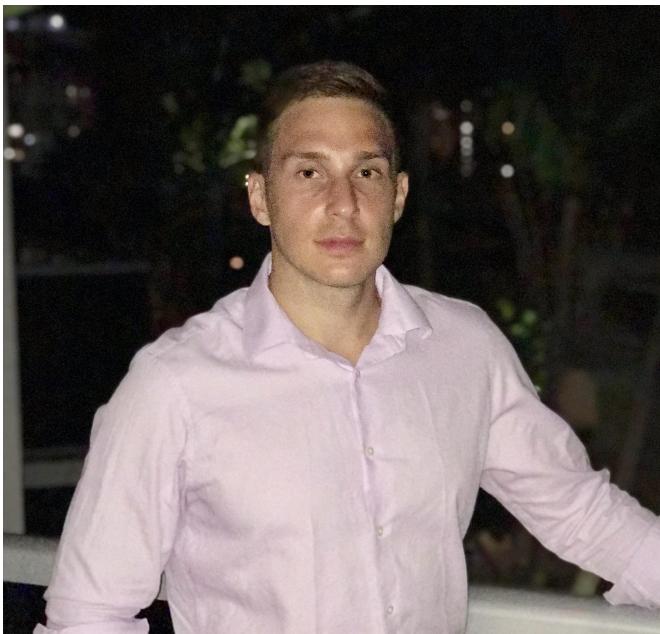
Run tests in parallel



Create proper page objects



Master synchronization



@Nikolay\_A00  
[www.ultimateqa.com](http://www.ultimateqa.com)

# Your Trainer

Nikolay Advolodkin

- Solutions Architect at Sauce Labs
- Contributing author to [Continuous Testing for DevOps Professionals](#)
- Test automation instructor and owner at [www.ultimateqa.com](http://www.ultimateqa.com)
- Voted one of top 33 automation engineer three years in a row by [TechBeacon.com](#)
- Automation instructor at TestAutomationU
- Animal lover

# Rules of Engagement



 KEEP  
CALM  
&  
FOLLOW  
THE RULES

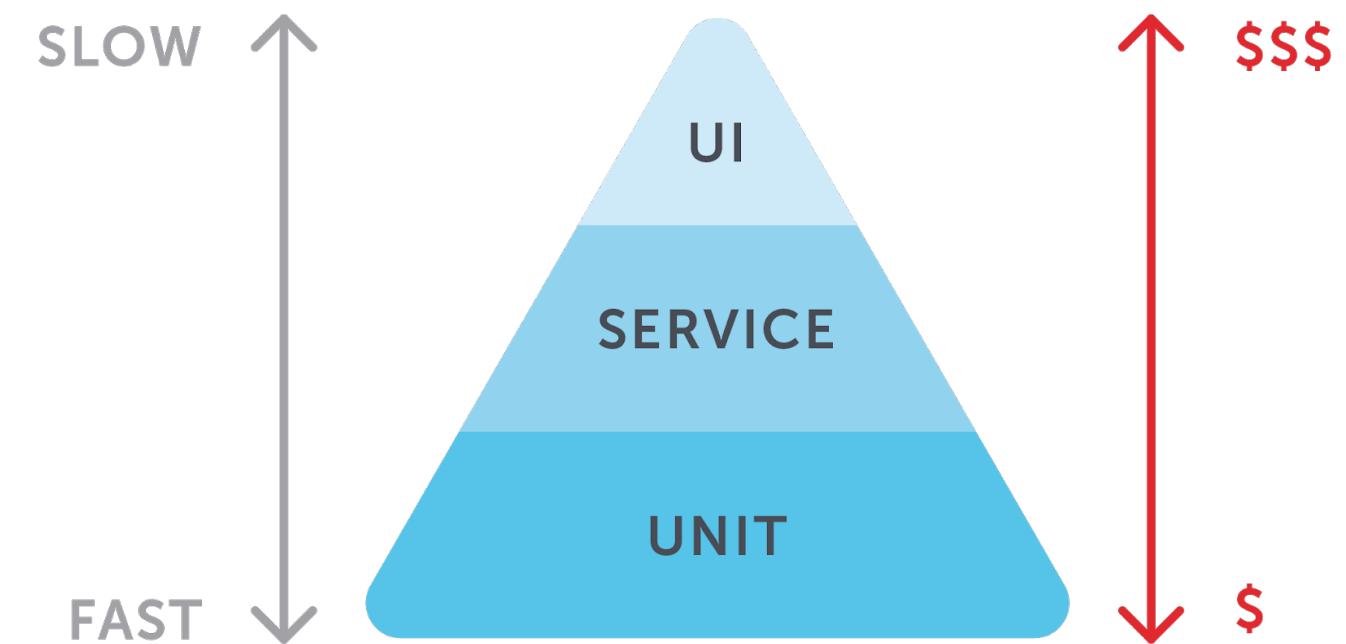
Don't worry about  
mistakes



Participation is  
Encouraged

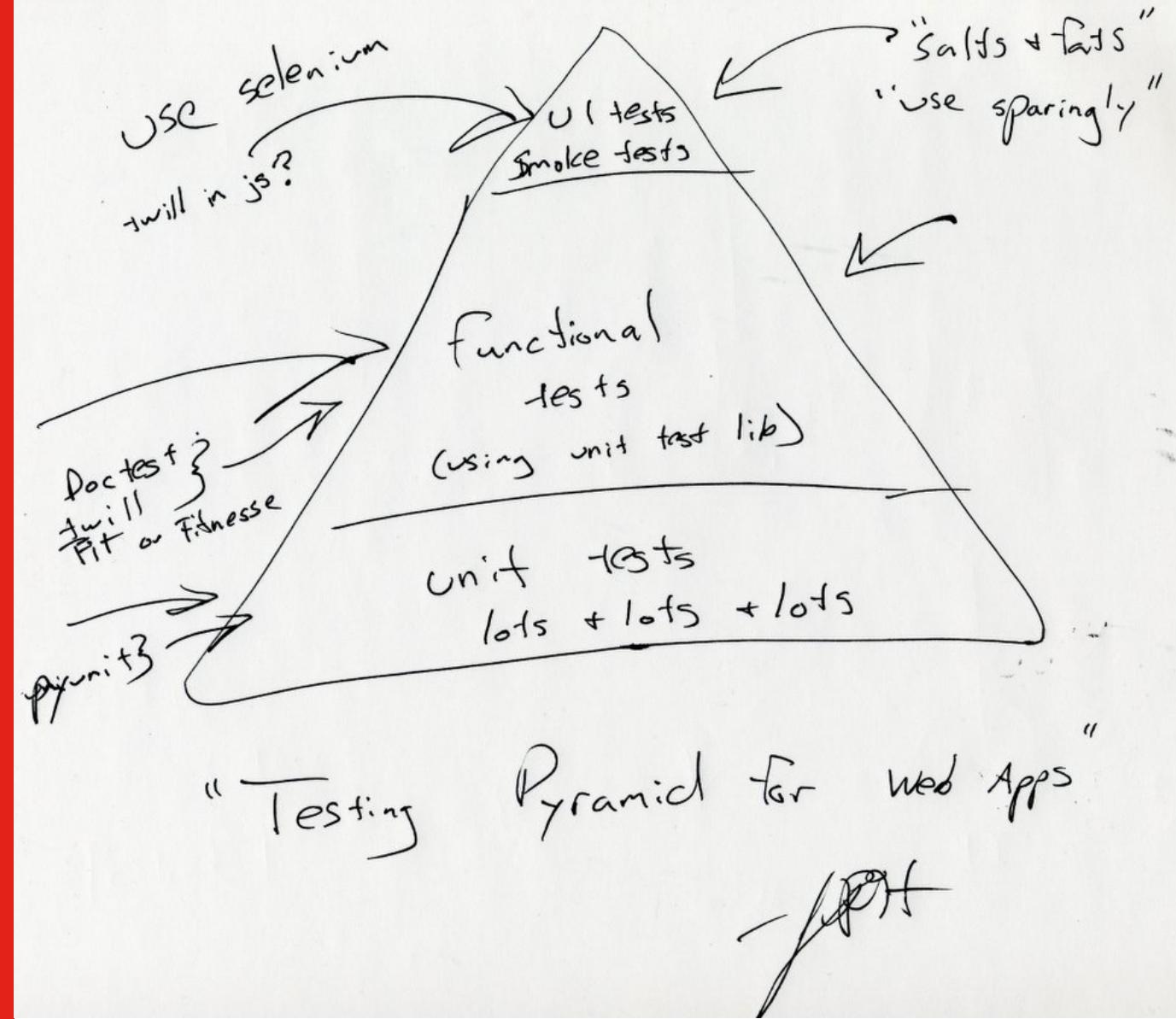


# Automation Pyramid

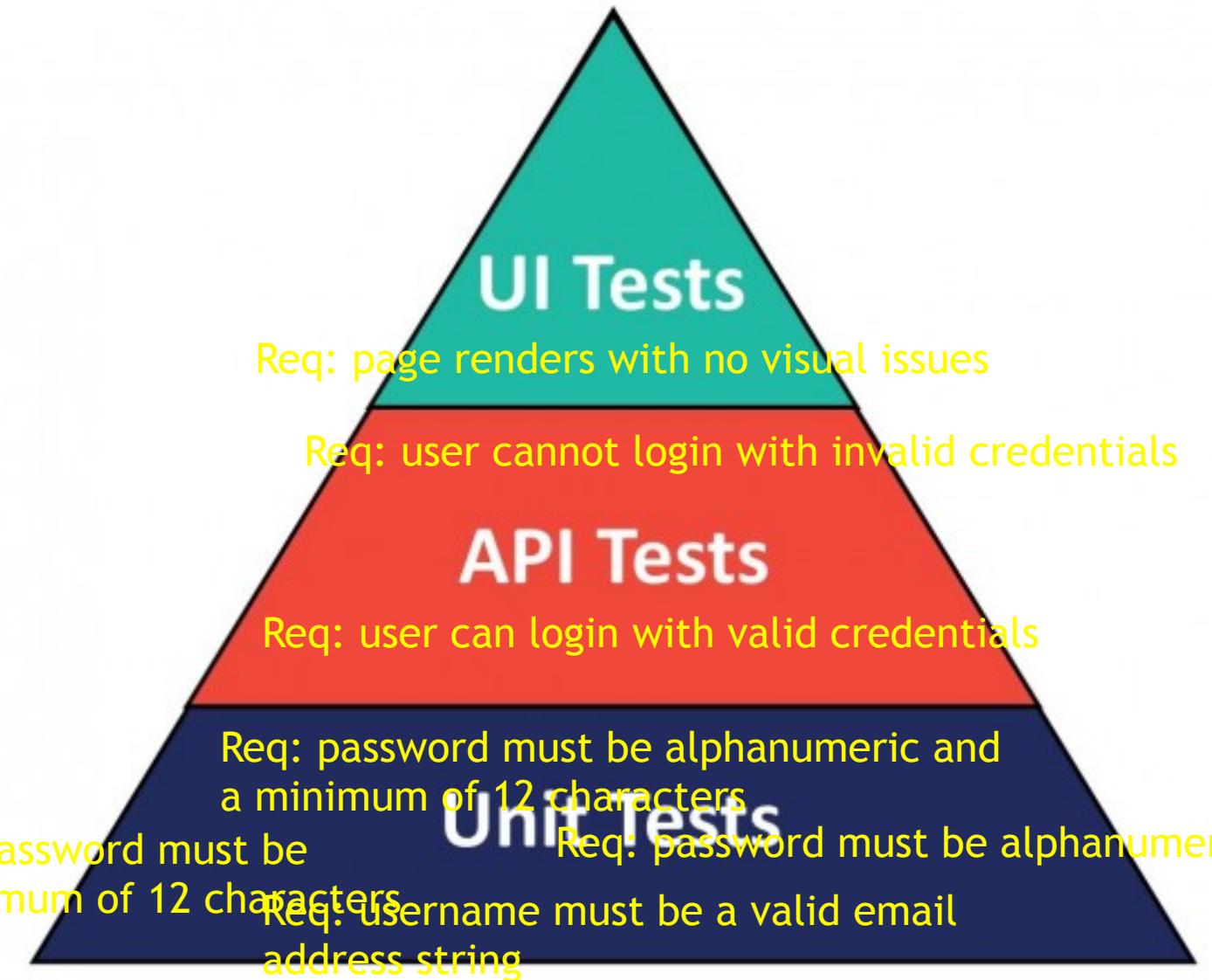


# Automation Pyramid

## Jason Huggins



# Login Feature



# Best Practices for Test Case Design



# Have an Object Oriented Approach

---

1. Identify code that fits into at least 3 different sections
  - a. Set-up
  - b. Test(s)
  - c. Teardown
2. Separate the different parts into functions with the appropriate annotations
3. Analyze script for any **repetitive identifications or actions** and store values/actions in variables



“Duplication is  
the primary  
enemy of a well-  
designed system.”  
- Bob Martin



# Page Objects



# Page Object



The diagram illustrates the implementation of a Page Object Model (POM) for a login page. On the left, in the Visual Studio IDE, the `SauceDemoLoginPage` class is shown. It includes a constructor that takes an `IWebDriver` parameter, a private field for the login button locator, and methods for loading the page, navigating to the URL, and getting the username, password fields, and login button. On the right, a screenshot of a web browser shows the `https://www.saucedemo.com` address bar. Below it, there are input fields for 'Username' and 'Password', and a blue 'LOGIN' button. A pink arrow points from the browser's address bar to the constructor in the code, and an orange arrow points from the 'LOGIN' button to the 'LoginButton' property in the code.

```
public class SauceDemoLoginPage : BasePage
{
    public SauceDemoLoginPage(IWebDriver driver) : base(driver)
    {
    }

    private readonly By _loginButtonLocator = By.ClassName("login-button");
    public bool IsLoaded => new Wait(_driver, _loginButtonLocator).IsVisible();
    public IWebElement UsernameField => _driver.FindElement(By.ClassName("logi"));
    public IWebElement PasswordField => _driver.FindElement(By.CssSelector("[t"));
    public IWebElement LoginButton => _driver.FindElement(_loginButtonLocator);

    public SauceDemoLoginPage Open()
    {
        _driver.Navigate().GoToUrl(BaseUrl);
        return this;
    }
}
```

https://www.saucedemo.com

Username

Password

LOGIN

## Use Page Object Pattern

---

- Improves Maintainability
  - Update in one place
  - Know where to update
  - Reduces duplication
- Improves Readability
- Improves Organization



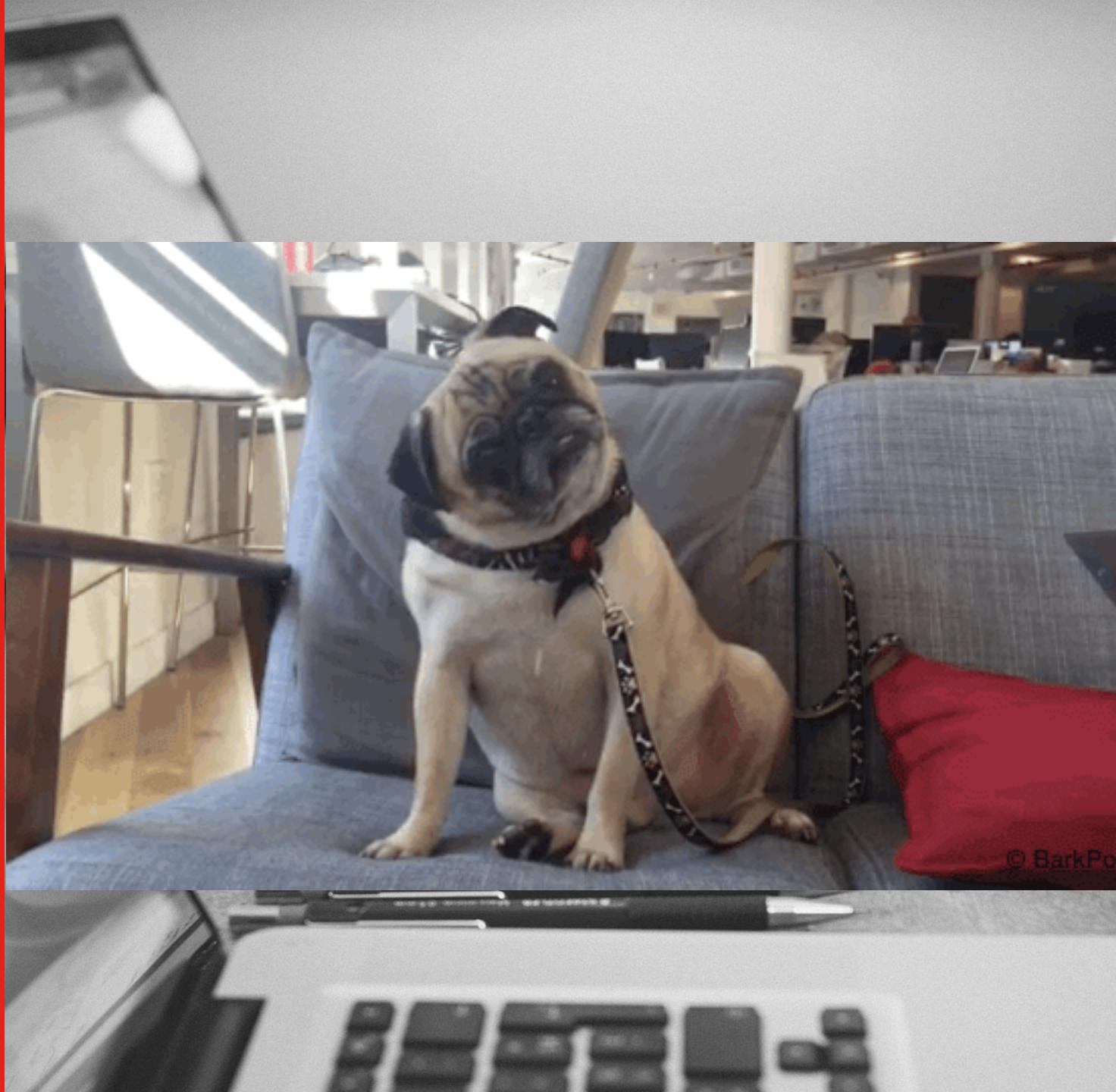


# Page Object Anti-Patterns

---

- Overpopulating BasePage
  - A place for common components of all pages
  - 200 lines of code max
- Over-populating a Page Object
- Public Methods/properties not representing user actions
  - OpenExcelFile()
  - ConnectToSql()
  - ConvertStringToArray()
  - ClickButton()
  - WaitForButtonToBeDisplayed()

# Questions?



# Coding Time



# What's Wrong With This Test?



```
18 @Test
19 public void fullCustomerJourney(Method method) throws MalformedURLException {
20
21     // Input your SauceLabs Credentials
22     String sauceUsername = System.getenv("SAUCE_USERNAME");
23     String sauceAccessKey = System.getenv("SAUCE_ACCESS_KEY");
24
25     MutableCapabilities capabilities = new MutableCapabilities();
26
27     //sets browser to Safari
28     capabilities.setCapability("browserName", "Safari");
29
30     //sets operating system to macOS version 10.13
31     capabilities.setCapability("platform", "macOS 10.13");
32
33     //sets the browser version to 11.1
34     capabilities.setCapability("version", "11.1");
35
36     //sets your test case name so that it shows up in Sauce Labs
37     capabilities.setCapability("name", method.getName());
38
39     capabilities.setCapability("username", sauceUsername);
40     capabilities.setCapability("accessKey", sauceAccessKey);
41
42     //instantiates a remote WebDriver object with your desired capabilities
43     driver = new RemoteWebDriver(new URL("https://ondemand.saucelabs.com/wd/hub"), capabilities);
44
45     //navigate to the url of the Sauce Labs Sample app
46     driver.navigate().to("https://www.saucedemo.com");
47     // navigate to desired page
48     driver.get("https://www.saucedemo.com");
```



SAUCECON 19

# Atomic Tests

# Test One Thing

Achieving atomicity might  
be the most powerful  
way to cure all test automation  
problems



# Non-Atomic Test

```
20 public void EndToEndTest()
21 {
22     SauceReporter.SetBuildName("AntiPatternTests3");
23     var loginPage = new SauceDemoLoginPage(Driver);
24     //test loading of login page
25     loginPage.Open().IsLoaded.Should().BeTrue("the login page should load successfully.");
26     loginPage.UsernameField.Displayed.Should().BeTrue("the page is loaded, so the username field should exist");
27     loginPage.PasswordField.Displayed.Should().BeTrue("the page is loaded, so the password field should exist");
28
29     //test login with valid user
30     var productsPage = loginPage.Login("standard_user", "secret_sauce");
31     productsPage.IsLoaded.Should().BeTrue("we successfully logged in and the home page should load.");
32     productsPage.Logout();
33     loginPage.IsLoaded.Should().BeTrue("we successfully logged out, so the login page should be visible");
```

# Atomic Test

```
20 public void EndToEndTest()
21 {
22     SauceReporter.SetBuildName("AntiPatternTests3");
23     var loginPage = new SauceDemoLoginPage(Driver);
24     //test loading of login page
25     loginPage.Open().IsLoaded.Should().BeTrue("the login page should load successfully.");
```



[Test]  
0 references | nikolay-advolodkin, 25 days ago | 1 author, 3 changes | 0 exceptions

```
public void LoginPageShouldLoad()
{
    _loginPage.Open().IsLoaded.Should().BeTrue("the login page should load successfully.");
}
```



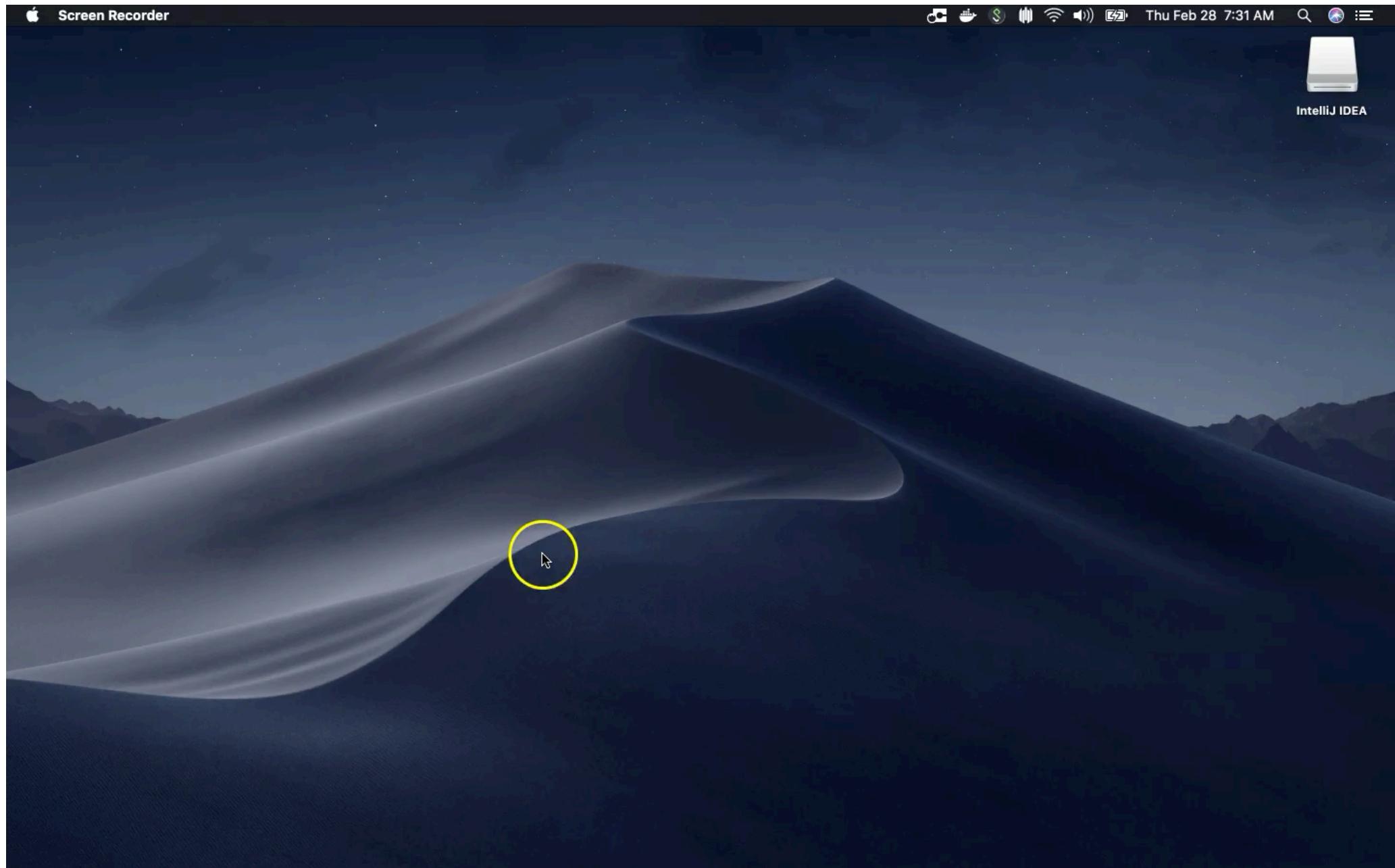
# End-to-End Tests?



A close-up photograph of a blue denim jacket's seams and stitching against a grey background. The jacket features dark brown thread stitching along the seams. A large rectangular area of the jacket is highlighted with a semi-transparent blue overlay, which contains the text "Seams and testability".

Seams and testability

# Using a Web API



# Login with a Browser Cookie

The screenshot shows the Chrome DevTools interface with the Application tab selected. On the left, the Storage section is expanded, showing Local Storage, Session Storage, IndexedDB, Web SQL, and Cookies. The Cookies section lists several entries, each with a circular icon and a URL. Two specific entries are highlighted with large green arrows pointing to them: 'https://www.amazon.com' and 'https://s.amazon-adsystem.com'. To the right of the cookies, there is a sidebar with a 'Filter' button and a list of names: at-main, csm-hit, i18n-prefs, lc-main, session-id, session-token, skin, sst-main, and ubid-main.

Name
at-main
csm-hit
i18n-prefs
lc-main
session-id
session-token
skin
sst-main
ubid-main

# DB Data Manipulation



# It Won't Be Easy



Questions?



# Let's Create An Atomic Test...



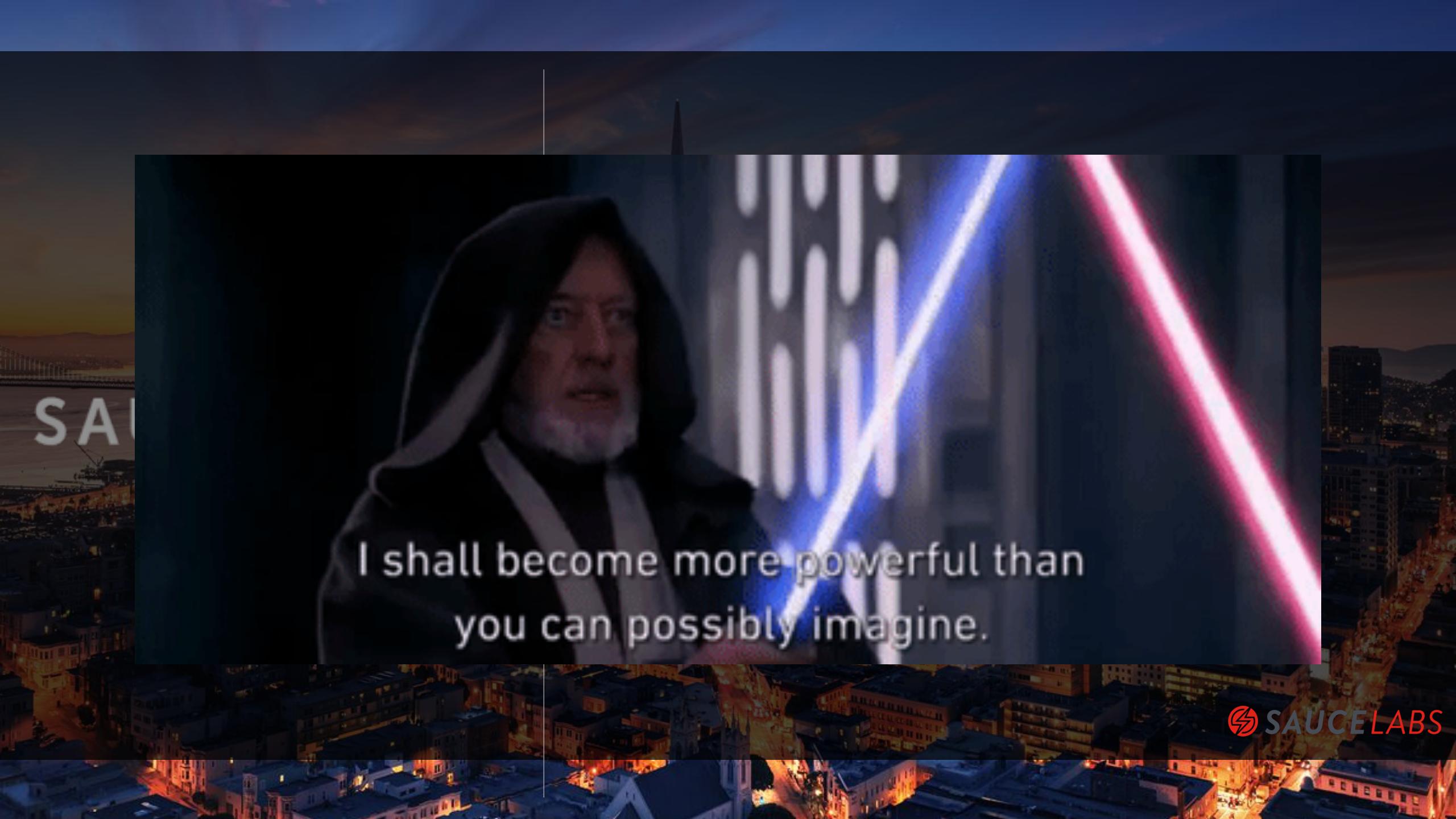
 SAUCE LABS



SAUCECON 19

How to achieve  
parallel test  
execution?

 SAUCE LABS



I shall become more powerful than  
you can possibly imagine.

# 400% Reduction in Suite execution speed

SAUCECON

Passed

18 tests ran in **20m 7s**

5 Failed ▾

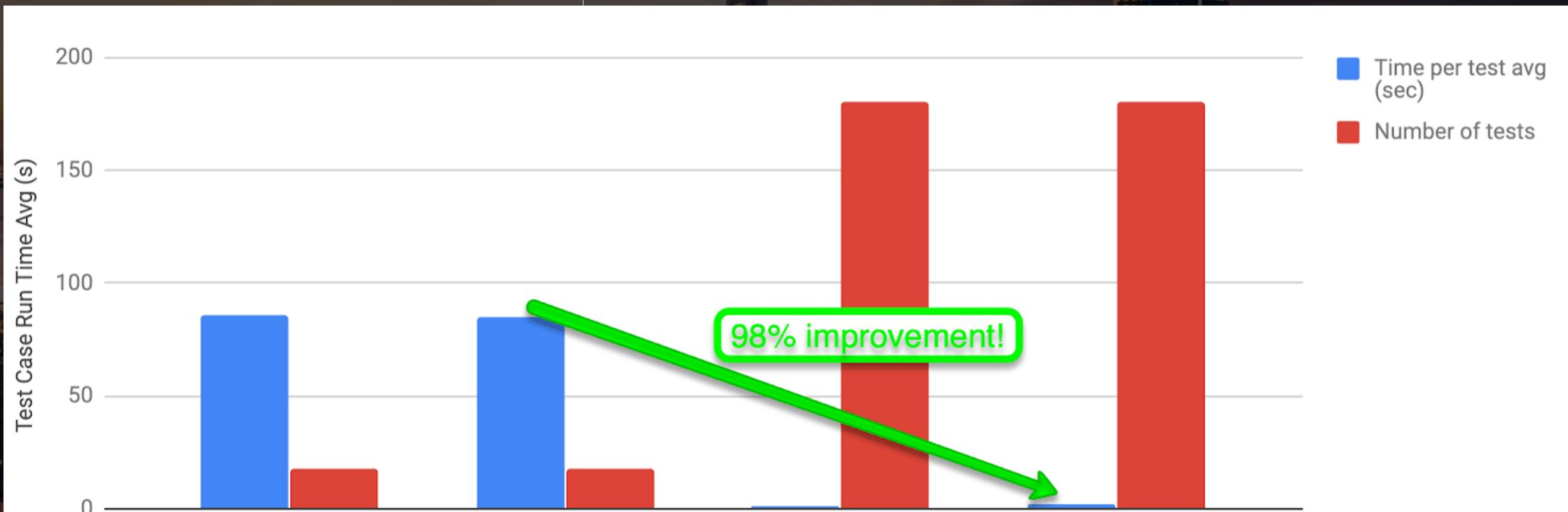
35 tests ran in **19h 43m 45s**

Passed

180 tests ran in **3m 55s**

 SAUCE LABS

# 98% improvement in Test run time

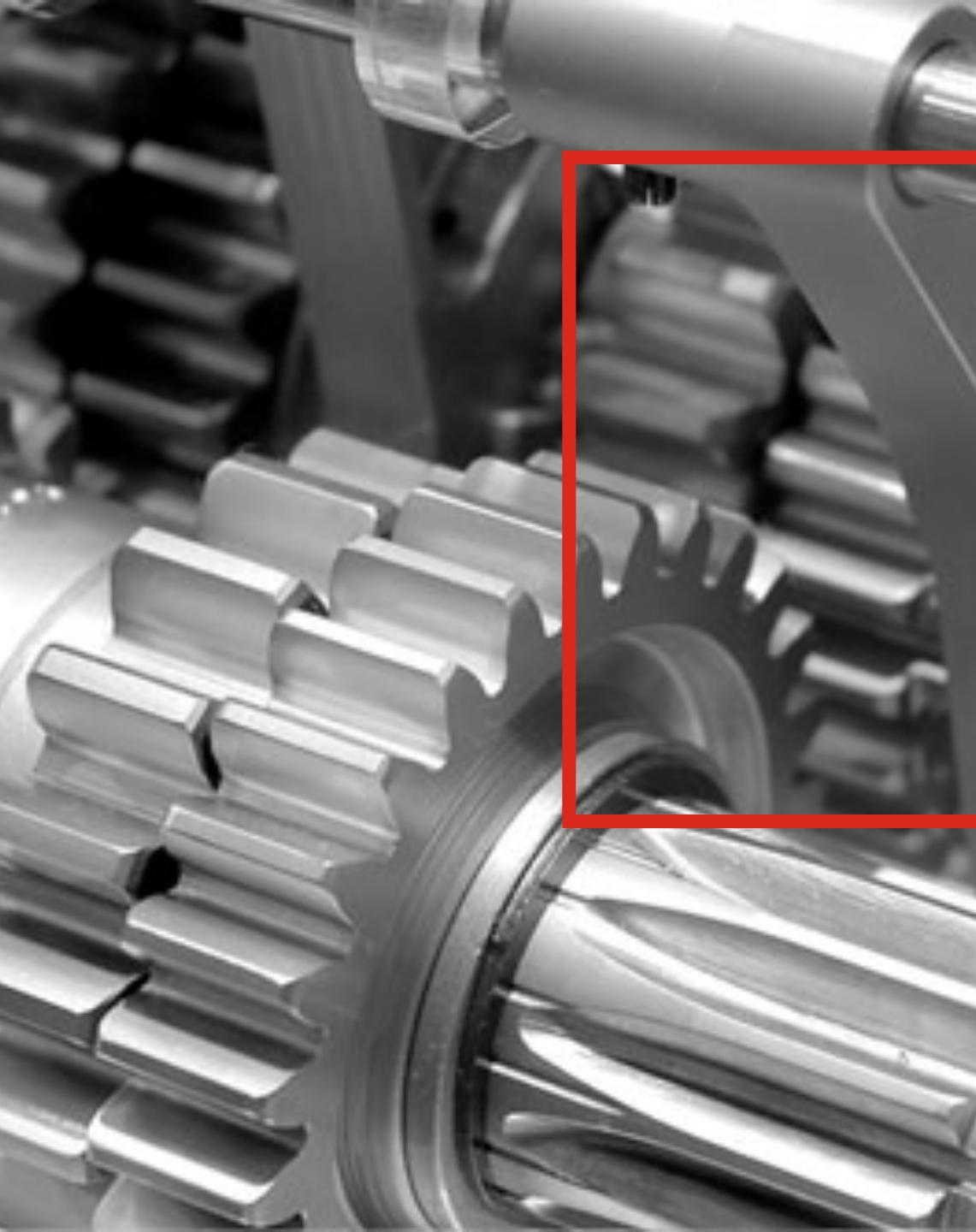




**7,000 tests  
in 7 min**

A blurred night photograph of a multi-lane highway. The image shows streaks of light from moving vehicles, creating a sense of speed and motion. The streaks are primarily yellow and white on the left side, transitioning to red and orange on the right side. The background is dark, suggesting a rural or highway setting at night.

100K tests  
Per day



## Mandatory Requirements for Parallelization

---

1. Tests must be *atomic*
2. Tests must be *autonomous*
3. Test data must be managed correctly
4. Avoid *static* keyword

A wide-angle photograph of a rural landscape. In the foreground, there's a small, dark, single-story house with a chimney, situated in a field. The land is a mix of green and brown, suggesting late autumn or early spring. A line of dark evergreen trees runs across the middle ground. In the background, there are more hills and mountains, all partially obscured by a thick, hazy atmosphere, giving the scene a misty and serene feel.

# Autonomous Tests

Self contained and isolated

### 3. Proper Test Data Management



# Danger



# Hard Coded Test Data

- Hard code data for use
- Concerns
  - Is Data in same state at end as at beginning?
  - What if test didn't pass?
  - Do you have to refresh data between uses?
  - Do you have to add new data for each new test?
  - Are you sure multiple tests with the same data is ok?



# Just-In-Time Test Data Management

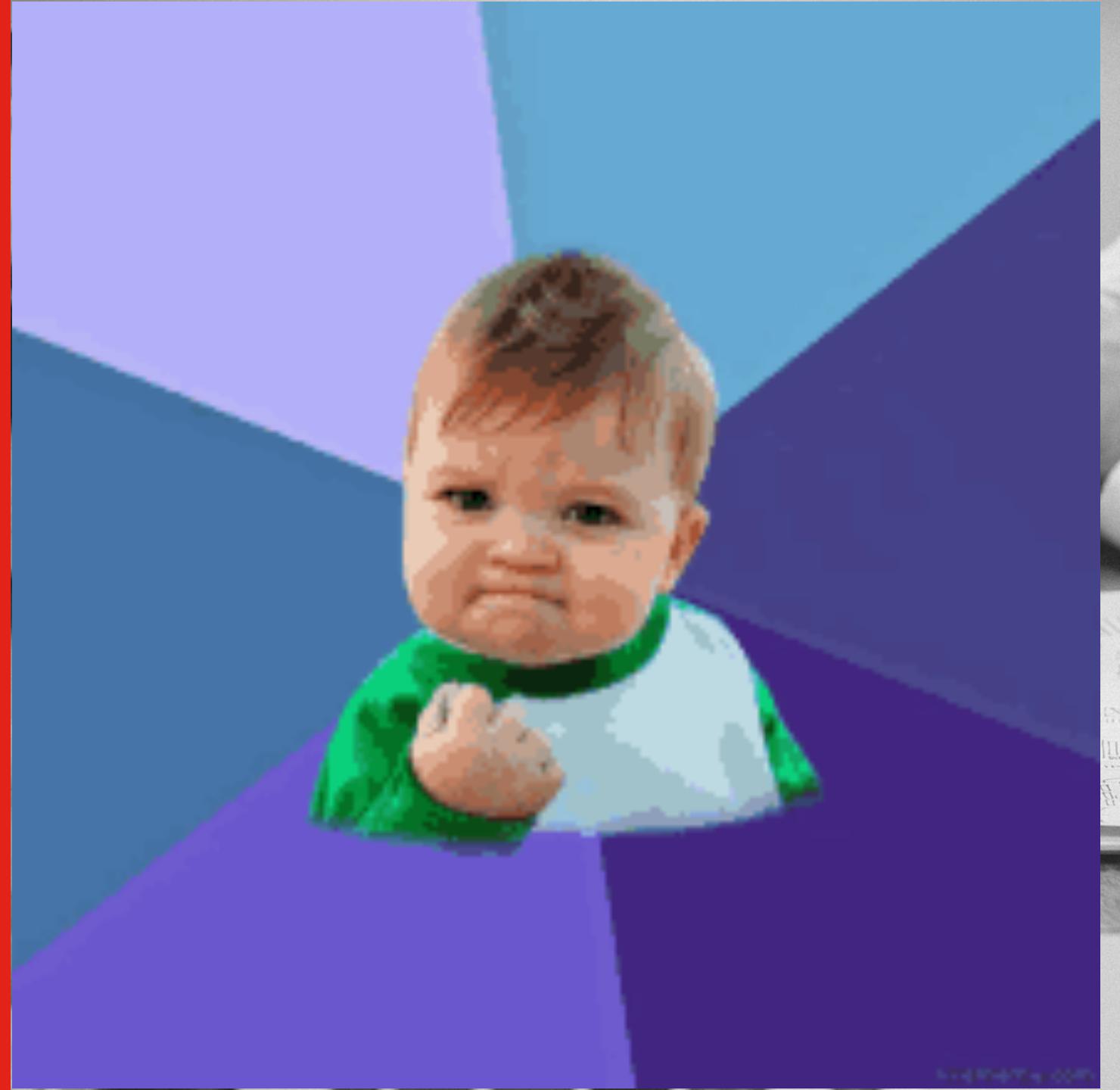
## 4. Don't be static about *static*



# Avoid *static* keyword

```
public class DriverInit { (using remote webdriver, browser name will be read from XML)
    public static WebDriver driver;
    public DriverInit() {
        switch (browser) {
            case "IE" : driver = ....;
            case "Firefox" : driver = ....;
        }
    }
}
```

# Bonus Tips!!



A LEGO Superman minifigure is shown flying through a blue and white gradient background. The minifigure has dark brown hair and a determined expression. He is wearing his iconic red and blue suit with the yellow 'S' emblem on the chest. His arms are outstretched, and his right hand is clenched into a fist, showing a yellow and red striped gauntlet. A red cape flows behind him.

Keep Tests  
Quick

A photograph of a sunset or sunrise sky. The colors transition from a deep blue at the top to a vibrant orange and red near the horizon. Silhouettes of many birds are scattered across the sky, particularly concentrated in two distinct groups: one on the left side and a larger, more horizontal group on the right side. The dark silhouette of a treeline is visible along the bottom edge of the frame.

Start Early

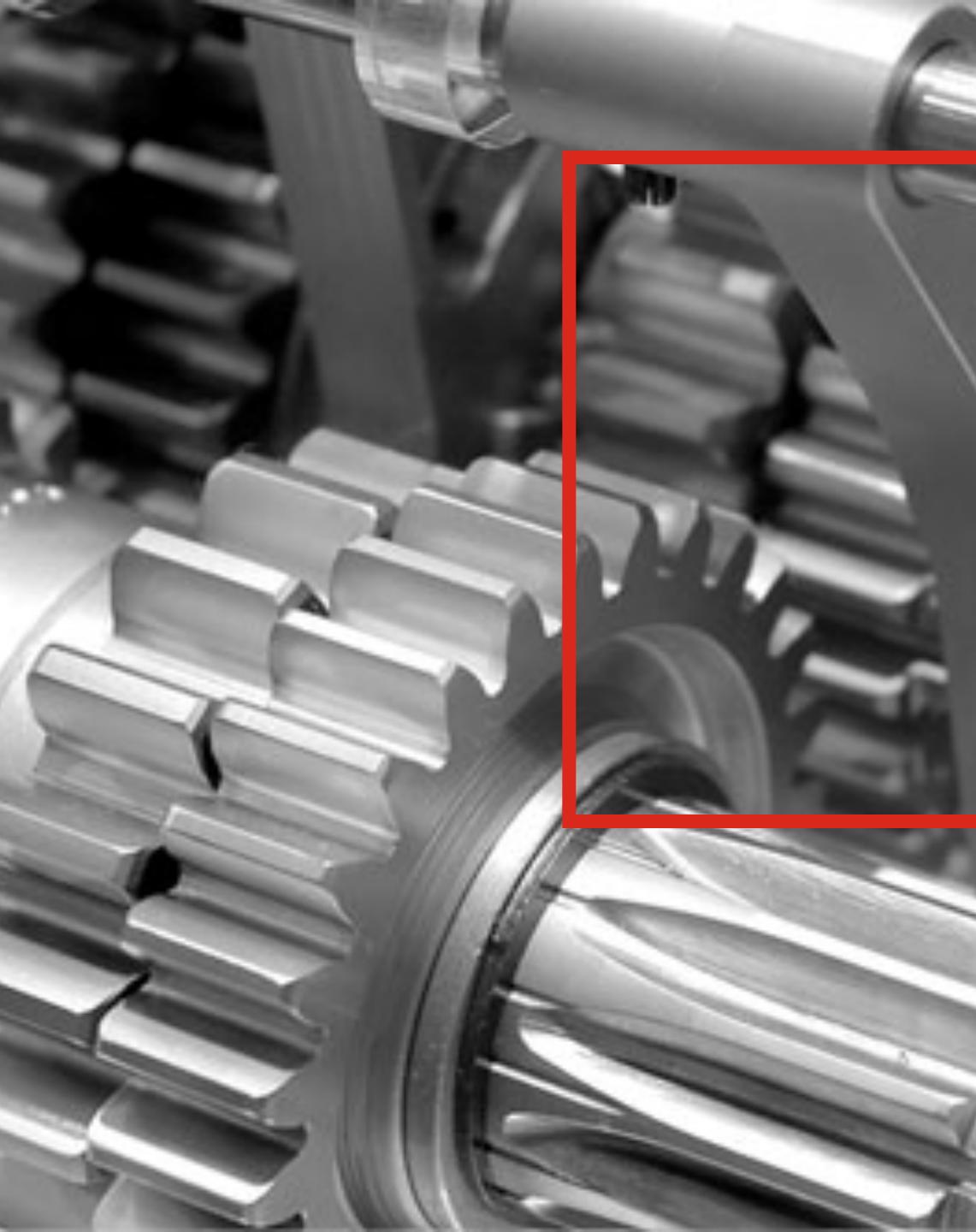
# Coding Time





SAUCECON 19

# Synchronization Strategies



## Use Waits to Account for Automation Speed

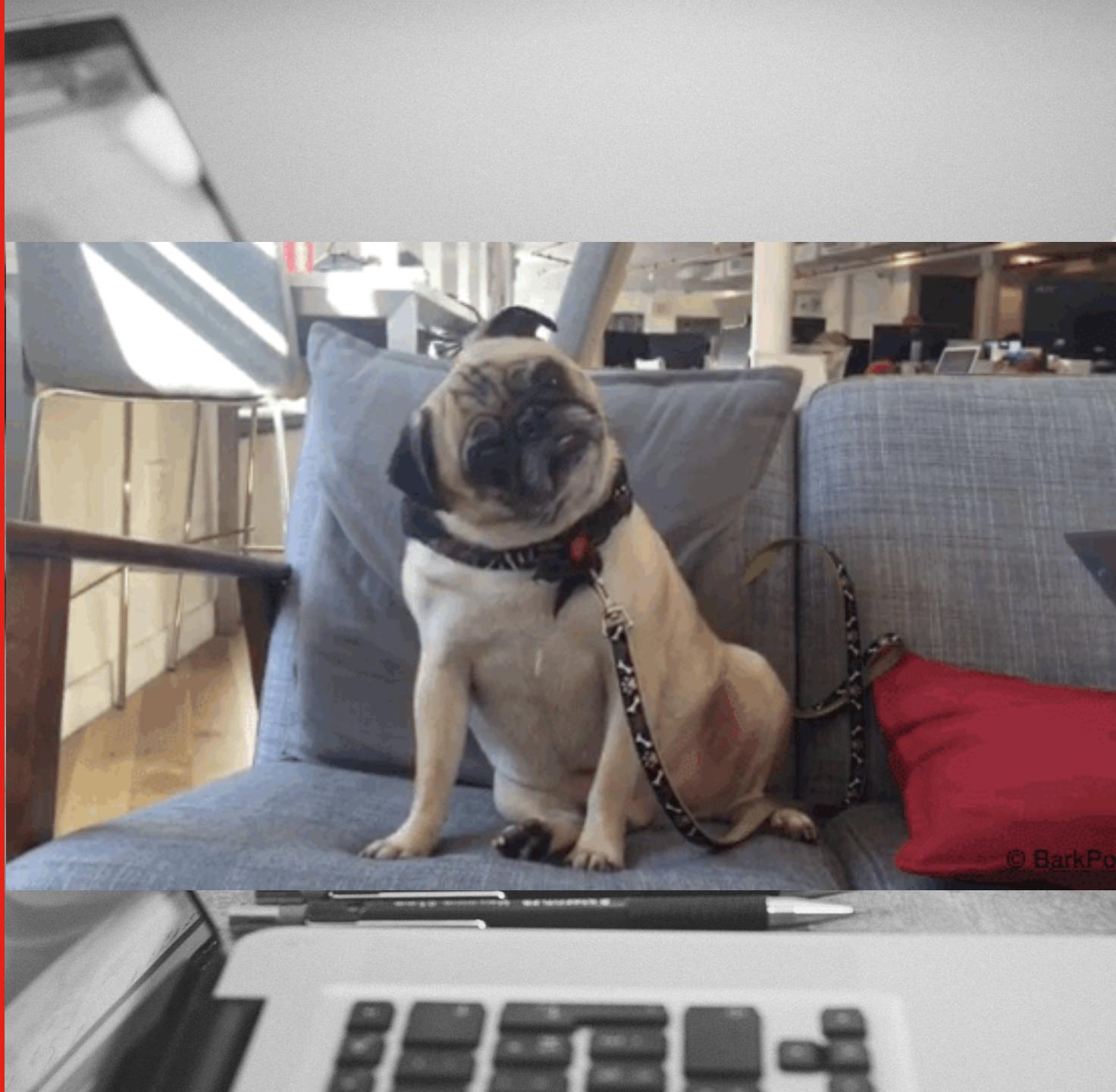
---

- Automation moves faster than human reactions and internet transmissions
- Use Waits to make sure elements are present before you try to test them

A scenic landscape featuring a deep blue lake in the foreground, surrounded by a dense forest of green coniferous trees. In the background, a range of majestic mountains is visible, their peaks bathed in a warm, golden-orange glow from the setting sun. The sky above is a clear, vibrant blue.

**~2,000% improvement in automation  
stability**

# What are the types of waits in Selenium?



# Implicit Wait

- Tell WebDriver to poll the DOM for a certain amount of time. This is literally using the `WebDriverWait()` method Selenium Webdriver offers
- Only needs to happen once
- Less flexible

## Implicit Wait

```
1 var _driver = new FirefoxDriver();
2 _driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(5));
```

ImplicitWait.cs hosted with

# Finds Elements in DOM

The screenshot shows a web page with a form and a developer tools interface.

**Form Fields:**

- Full Name: First Last
- Credit Card: 4111111111111111
- Month: 01
- Year: 2050
- Purchase button

**Message:** Not visible...

**Developer Tools - Elements Tab:**

- Console
- Sources
- Network
- Performance
- Memory
- Application
- Security

**DOM Tree:**

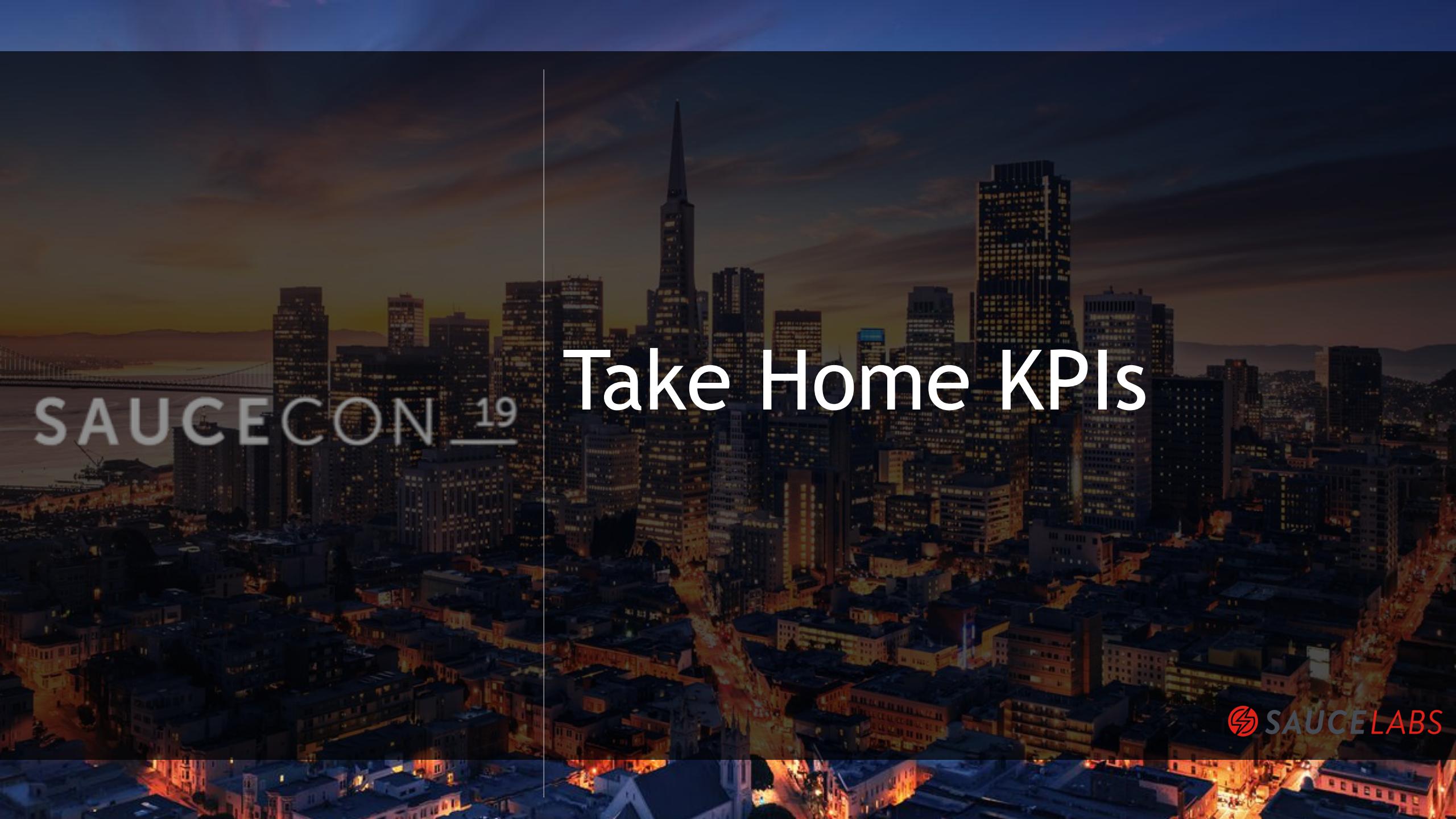
```
><tr>...</tr>
><tr>...</tr>
><tr>...</tr>
▼<tr>
  ▶ <td>...</td>
  ▶ <td>...</td>
  ▼<td>
    <div id="success" style="display: none;">Purchase complete!</div>
  </td>
```

# Explicit Wait

This is the code that you write to wait for some condition to occur before proceeding.

Recommended way to wait in your tests

```
1 //EXAMPLE 1 - This is the most convenient method provided to us by the Webdriver API
2 //ExpectedConditions class provides us many different options for locating an element
3 var wait = new WebDriverWait(_driver,TimeSpan.FromSeconds(10));
4 wait.Until(ExpectedConditions.ElementIsVisible(By.Id("elementId")));
5
6 //EXAMPLE 2 - This is a wait that dynamically checks for the presence of an element for a maximum amount of time, a bit burdensome
7 //because we had to write it
8 IWebDriver driver = new FirefoxDriver();
9 driver.Url = "http://somedomain/url_that_delays_loading";
10 WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
11 IWebElement myDynamicElement = wait.Until<IWebElement>((d) =>
12 {
13     return d.FindElement(By.Id("someDynamicElement"));
14 });
15
16 //EXAMPLE 2 - This is the worst kind of wait and should almost never be used
17 Thread.Sleep(10000)
```

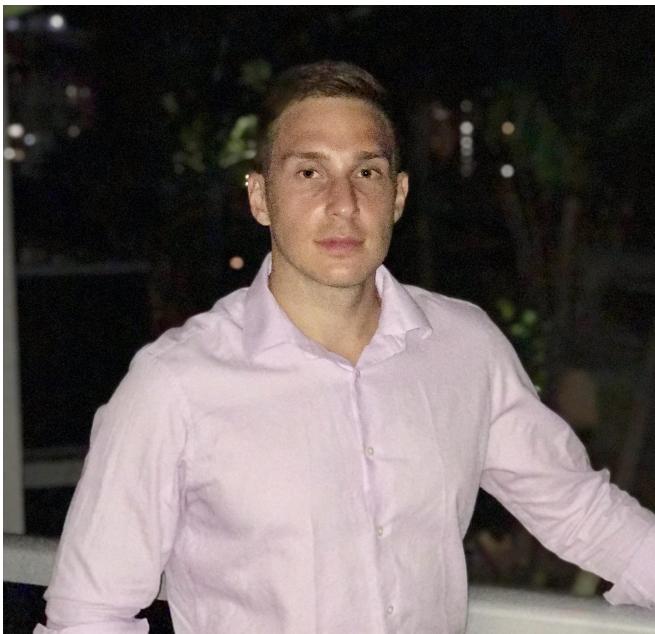


SAUCECON 19

# Take Home KPIs

# Key Performance Indicators of Automation

1. 100% automation reliability
2. 1 min/test max on your local resources
3. 10 min max for entire automation suite
4. Tests execute every pull request



@Nikolay\_A00  
[www.ultimateqa.com](http://www.ultimateqa.com)

# Thanks!

## Nikolay Advolodkin

- Solutions Architect at Sauce Labs
- Contributing author to [Continuous Testing for DevOps Professionals](#)
- Test automation instructor and owner at [www.ultimateqa.com](http://www.ultimateqa.com)
- Voted one of top 33 automation engineer three years in a row by [TechBeacon.com](http://TechBeacon.com)
- Automation instructor at TestAutomationU
- Animal lover



SAUCE LABS