Kod szkolenia: **JS/TDD**

Tytuł szkolenia: **Test-Driven Development w JavaScript**

Dni: 2

Opis:

Adresaci szkolenia:

Szkolenie adresowane jest do programistów JavaScript. Kierowane jest do osób chcących uzyskać większe bezpieczeństwo procesu tworzenia oprogramowania (mniej regresji i nieprzewidzianych defektów) oraz lepszą jakość i czytelność tworzonego kodu.

Cel szkolenia:

Uczestnicy czują się pewnie stosując TDD w codziennej pracy swojego zespołu.

Jako że TDD jest trudne, szkolenie nie sprawi, że wszyscy staną się ekspertami. Jednak przeszkolenie całego zespołu daje dużą szansę, że zespół przyjmie tę praktykę.

W ramach szkolenia uczestnicy poznają i przyswoją cykl pracy TDD, nauczą się projektować oprogramowanie pod względem testowalności i tworzyć czytelny kod. Poznają biblioteki ułatwiające stosowanie TDD oraz umożliwiające testowanie na różnych poziomach.

Mocne strony szkolenia:

Uczestnicy przyswajają wiedzę w najskuteczniejszy możliwy sposób – praktykując TDD podczas serii ćwiczeń. Każdy moduł zawiera warsztaty. Większość ćwiczeń polega na test-drive'owaniu przy użyciu prezentowanych technik. Szkolenie uwzględnia specyfikę języka JavaScript, duży nacisk położony jest na testowanie kodu asynchronicznego.

Wymagania:

Znajomość JavaScript.

Parametry szkolenia:

2*8 godzin (2*7 godzin netto) wykładów i warsztatów (z wyraźną przewagą warsztatów).

Wielkość grupy: maks. 8-10 osób.

Program szkolenia:

sages

Podstawy TDD

- dlaczego TDD może Ci pomóc
- stosowanie cyklu TDD Red-Green-Refactor
- definicja TDD

Testowanie jednostkowe

- rodzaje i poziomy testów: jednostkowe, integracyjne, akceptacyjne…
- co to jest test jednostkowy?
- specyfika testowania aplikacji JS
- środowisko do testowania w JavaScript: Karma, Grunt, PhantomJS, Jasmine*
- możliwości Jasmine
- pisanie czytelnych asercji przy użyciu biblioteki Chai*

Tworzenie utrzymywalnych testów

- nazewnictwo testów
- struktura testu Arrange/Act/Assert
- dane w testach
- testy sparametryzowane

Testy współpracujących obiektów

- wykorzystanie dublerów testowych (Test Doubles) w celu izolacji testowanego kodu
- różnice między mockami a stubami
- mockowanie w Jasmine
- testy zależne od czasu i daty
- testowanie requestów HTTP - biblioteka Sinon.JS

Sages sp. z o.o., ul. Nowogrodzka 62c, 02-002 Warszawa
tel. +48 22 203 56 00 fax: +48 22 203 56 01
e-mail: biuro@sages.com.pl web: www.sages.com.pl

sages

Projektowanie testowalnej aplikacji

- modularyzacja aplikacji
- enkapsulacja a testowalność
- Dependency Injection (wstrzykiwanie zależności)
- zasada pojedynczej odpowiedzialności (Single Responsibility Principle)

Refaktoryzacja

- jak rozpoznać i naprawić brzydkie zapachy w kodzie (Code Smells)
- pisanie czystego kodu w JavaScript - przydatne wzorce
- najczęściej stosowane refaktoryzacje
- refaktoryzowanie małymi krokami przy przechodzących testach

Krótkie spojrzenie na BDD/Specification by Example

- używanie przykładów do komunikacji z biznesem
- automatyzowanie przykładów z wykorzystaniem Cucumber-JS

Testy symulujące działania użytkownika z Protractor

- kiedy potrzebujemy testów end-to-end
- zalety i wady testów z poziomu przeglądarki
- wprowadzenie do Protractora

* możliwe użycie innych bibliotek i narzędzi na życzenie klienta

Sages sp. z o.o., ul. Nowogrodzka 62c, 02-002 Warszawa
tel. +48 22 203 56 00 fax: +48 22 203 56 01
e-mail: biuro@sages.com.pl web: www.sages.com.pl