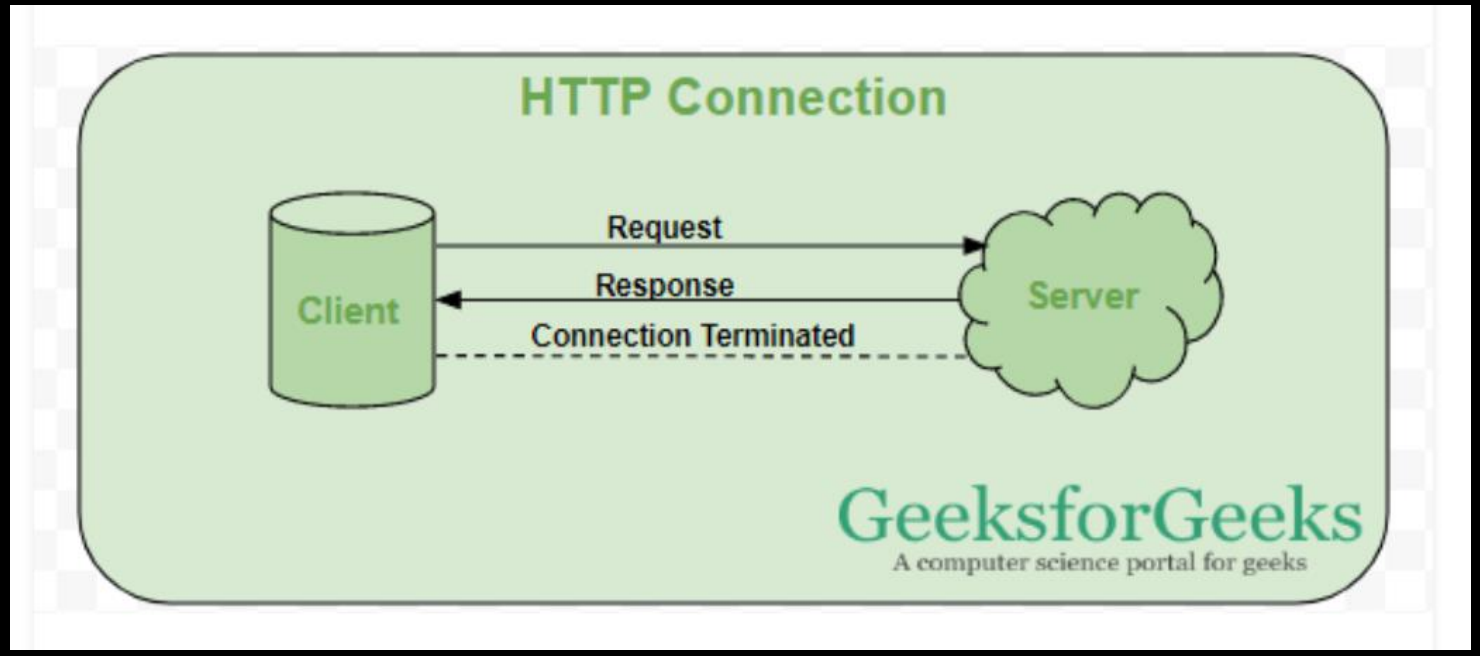
Version1-WebSockets

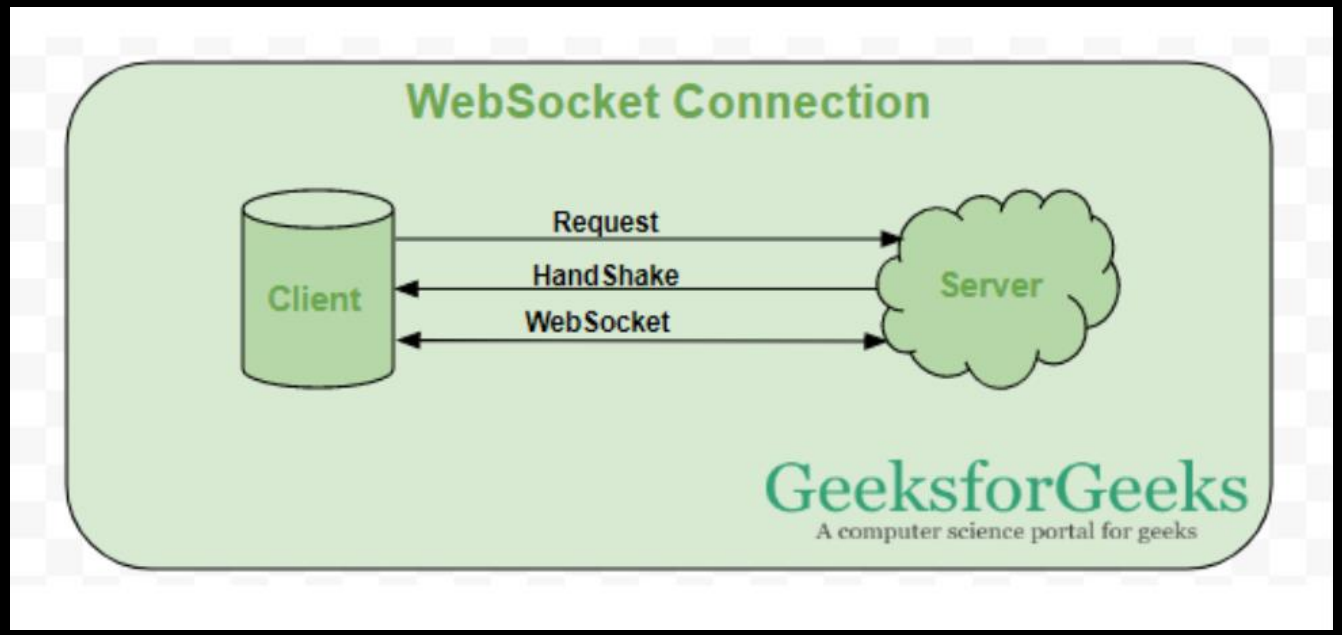
Http

- פרוטוקול חד כיווני בשכבת האפליקציה בו הלקוח שולח בקשה לשרת ומקבל תגובה.
- כל תגובה של השרת מזוהה עם <u>בקשה מסוימת</u> מצד הלקוח, (stateless)
- רץ על גבי פרוטוקול TCP



WebSockets

- פרוטוקול תקשורת דו כיווני בין שרת ללקוח
- ברגע שלקוח מתחבר לשרת, נעשה handshkaing וחיבור חדש. החיבור יתנתק רק על ידי הלקוח\השרת.
- ההודעות המועברות בין השרת ללקוח מועברות על ידי החיבור שנוצר.

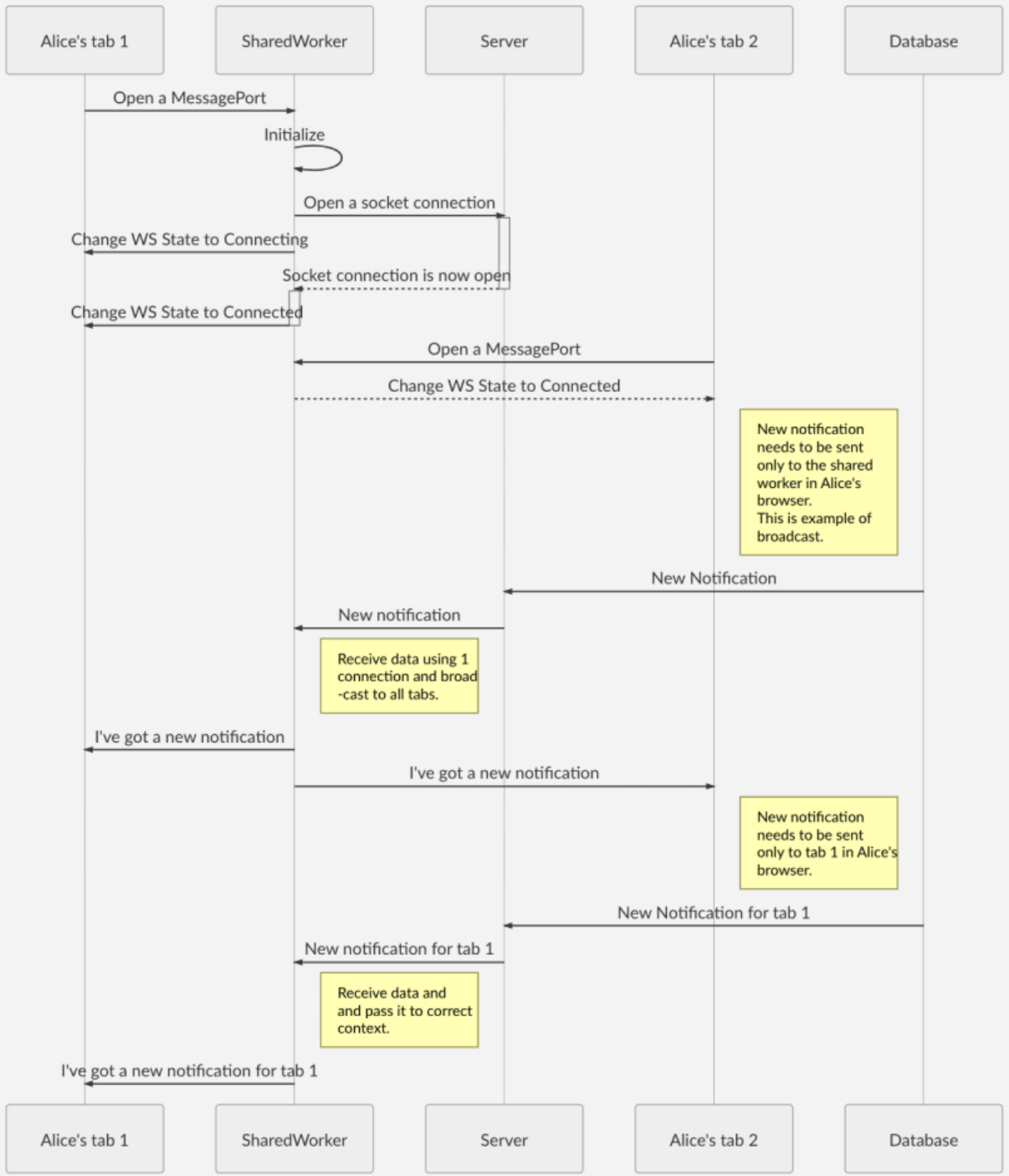


הבדלים בין websockets ל-Http

Http	Web sockets
פרוטוקול חד כיווני	תקשורת דו כיוונית בין שרת ללקוח
נשתמש כאשר לא נרצה לשמור על קשר בין שרת ללקוח זמן רב\ לכמות זמן שהיא לא ידועה	נשתמש כאשר נרצה לשמור על קשר קבוע בין השרת ללקוח
תקשורת איטית יותר	בדרך כלל תקשורת מהירה יותר מ-Http
Stateless-Stateless Protocol does not require the server to retain the server information or session details	StateFull-Stateful Protocol require server to save the status and session information

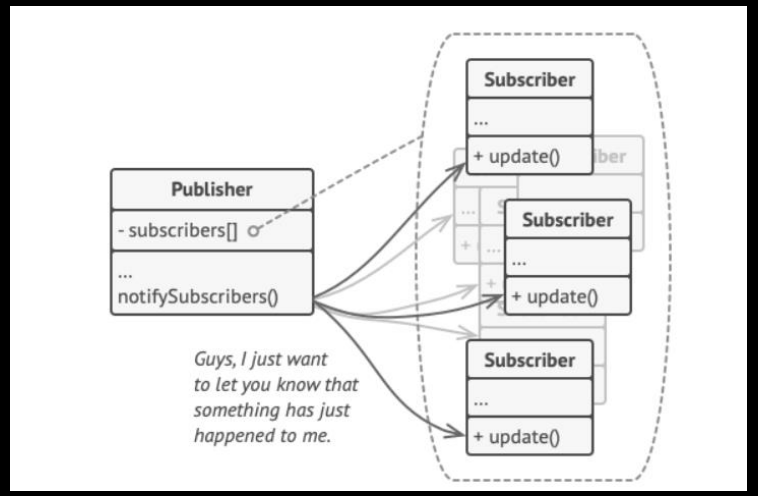
שאלה 2

- במערכת שלנו השתמשנו בבקשות HTTP ולא ב-websokcets.
- הבעייתיות במצב זה הוא שברגע שאנו מנווטים ל-url חדש החיבור של ה-socket נסגר. כלומר, לא ניתן לנווט לדף url חדש מבלי לסגור את החיבור של ה-socket .
- אחד מפתרונות האפשריים למצב זה הוא להשתמש ב-Webworker, אשר תוכנן על מנת לאפשר לכמה עמודים מאותו האתר לחלוק אותו.
- כאשר אנחנו משתמשים ב-HTML script הדף נהיה לא זמין עד שה-script מסתיים. Webworker זה אפליקציה שרצה ברקע באופן נפרד מסקריפטים אחרים, אשר מאפשרת לדף להמשיך לפעול.
- נוכל לפתוח חיבור ל-webworker, החיבור יפעל עד שכל הדפים ייסגרו, וחיבור יחיד יכול לשמש לכמה דפים עבור אותו לקוח.
- באופן זהף הלקוח יעבור בין כמה דפים מבלי לאבד את החיבור.


Alice's tab 1
SharedWorker
Server
Alice's tab 2
Database
Open a MessagePort
Initialize
Open a socket connection
Change WS State to Connecting
Socket connection is now open
Change WS State to Connected
Open a MessagePort
Change WS State to Connected
New notification needs to be sent only to the shared worker in Alice's browser.
This is example of broadcast.
New Notification
New notification
Receive data using 1 connection and broad -cast to all tabs.
I've got a new notification
I've got a new notification
New notification needs to be sent only to tab 1 in Alice's browser.
New Notification for tab 1
New notification for tab 1
Receive data and and pass it to correct context.
I've got a new notification for tab 1
Alice's tab 1
SharedWorker
Server
Alice's tab 2
Database

שאלה 3

- מערכת המסחר נועדה לתת שירות ללקוח.
- הלקוח יכול לבצע פעולות שונות במערכת המסחר ולבקש ממנה בקשות ומידע, מערכת המסחר היא השרת במודל שלנו.
- לעומת זאת, מערכת המסחר לא יכולה וצריכה לתקשר עם הלקוח. התפקיד היחידי של מערכת המסחר היא למלא אחר פעולות הלקוח.
- מודל הארכיטקטורה שלנו פועל לפי שכבות, כאשר התקשורת פועלת כלפי מטה (כלומר כל שכבה יכולה להעביר הודעה לשכבה מתחתיה).
- במידה והתקשורת תהיה דו כיוונית- יעברו הודעות גם בין השכבות התחתונות לשכבות העליונות, בניגוד למודל הארכיטקטורה שלנו.



שאלה 4

- נשתמש בתבנית ה-observer על ידי שימוש בתבנית זו, כל לקוח חדש אשר מעוניין לקבל התרעות יירשם למערכת בתור observer
- כאשר המערכת תוכל לתקשר עם כל הלקוחות באמצעות ()notifyobserver
- כרגע התקשורת במערכת שלנו היא חד כיוונית, הלקוח שולח הודעות והשרת לא יכול לשלוח לו הודעות חזרה (אלא רק כתגובות על ההודעות שנשלחו).
- כלומר במידה והשרת ישלח הודעה ללקוח, הלקוח לא יהיה שם על מנת לקבל את ההודעה (לא יהיה מי שימתין להודעה זו).
- על ידי שימוש בתבנית ה-observer נוכל לגרום ללקוח להמתין להודעות מהשרת ובכך לממש תקשורת דו כיוונית.

שאלה 4 סעיף א

- מימוש התקשורת מצריך הגדרה של רכיב ארכיטקטורה חדש, ה-handleobserver
- תפקידו של רכיב זה יהיה לנהל את ההודעות מהשרת ולרשום משתמשים חדשים אשר מעוניינים לקבל הודעות מהשרת.
- רכיב זה יהיה בין ה-domain layer ל-service layer ובכך גם יהיה ניתן לשלוח הודעות ללקוחות הרשומים, וגם לרשום לקוחות חדשים על ידי שימוש ב-domain layer.

התרעות בזמן אמת

- נתמוך בקבלת התראות בזמן אמת על ידי כך שבכל תרחיש נגדיר את הקונפיגורציה של המערכת (מוצרים, משתמשים, חנויות וכדומה) ואת המצב הסופי אליו נרצה להגיע בתרחיש.
- בבדיקות הקבלה נשתמש בטיימר, אשר בכל נקודת זמן יבקשו בקשה מהשרת.
- לאחר קבלת התשובה מהשרת נשווה את מצב המערכת למצב הסופי אליו רצינו להגיע, בדרך זו נוכל להשוות בין המצבים.
- על ידי שימוש בטיימר נוכל לקבל בזמן אמת מידע על השימוש במערכת ועל דברים שהשתנו בה.