Version Control

a.k.a. Revision Control , a.k.a Source Control

סביר להניח שאתה כבר שם

פרוייקט גמר.docx
פרוייקט גמר סופי.docx
פרוייקט גמר סופי ביותר!.docx
myPic.jpg
myPic_old.jpg
btsd_25_3_2017.zip

אתה גם עובד נכון

שיתוף קבצים לעבודת צוות ביצעת באמצעות dropbox
חזרת אחורה לגירסאות שנשלחו ב gmail או באיזה disk-on-key
וכפי שראינו, בסיום גירסא שמרת עותק קשיח
החזקת תיקיה עותק והעברת ממנה לתיקיה המרכזית ב dropbox

למה?

בשביל מסמכים אישיים, הניהול שעשית עד כה היה סבבה, אבל לפרוייקט של תכנות?
נראה לך שבאמת בשרתים של apple יושבת תיקיה iOS11_beta1 ששם נמצא הקוד
שעליו עובדים
ושכל מתכנת מחזיק אצלו תיקיה נפרדת של הקוד?
כנראה שלא...

ניהול קוד גדול, שיותר ממתכנת אחד עובד עליו
ועל אחת כמה וכמה צוות שמוציאים ממנו מספר גירסאות
צריך ניהול קוד בריא על מנת להמנע מ**כאוס**

עקרונות בניהול קוד יעיל

גיבוי ושחזור - שמירת קבצים לאחר עריכה, רוצה לראות מה היה כתוב בקוד לפני שנתיים שעבד טוב? אין בעיה, רק תיגש לשם.

סנכרון - מתכנת יכול להעלות את השינויים שלו ולהשאר מעודכן בשינויים שעשה הצוות

אפשר להתחרט קצת- עבדת על איזה פונקציה והרסת את הפרוייקט? לא קרה שום דבר, רק תחזור לנקודה האחרונה

אפשר להתחרט הרבה - וואלה, גירסא 1.0.8 הייתה יותר יציבה? בא נראה מה היה שם

בלי קסמים - מעקב אחרי תוספות ושינויים, והסבר למהות השינוי

מי הסתום שכתב את זה? אופס, זה אני :) - לכל קוד שמסתנכרן יש תיעוד מי המשתמש שביצע

שטח אש - רוצה לצאת להרפתקאה עם סוף לא ידוע בלי לשאת בתוצאות? ניתן להקים עותק נפרד של הקוד עם כל הפינוקים של ניהול קוד, היה מוצלח? אפשר למזג.. לא יצא משהו? תשמיד ראיות...

נראה לי שזה עדיף על ניהול תיקיה משותפת...

סלנג בסיסי

מסד הנתונים שמחזיק את כל הקבצים - **Repository** (**repo**)

מחשב (מקומי או מרוחק) המתחזק את הנ"ל - **Server**

אתה, כלומר המחשב המחובר ל Repo - **Client**

התיקיה אצלך במחשב שבה נמצא הפרוייקט - **Working Copy / Working Set**

הדרך שהביאנו עד הלום - **Trunk / Main**

סלנג פעולות בסיסיות

הוספת קובץ חדש ל repo, או להתחיל מעקב על קובץ קיים - **Add**

באיזו גירסא נמצא הקובץ - **Revision, Commit**

ה Revision האחרון (למשל, תיקון באג 45) ב Repo - **Head**

הורדת הגירסא האחרונה - **Checkout / Pull**

העלאת קבצים ל Repo - **Check In / Push**

הערה המתלווה להעלאה - **Check in message**

רשימת השינויים בהעלאה - **Chanage Log / Histroy**

סנכרון אל מול הגירסא האחרונה - **Update / Sync / Pull**

ביטול השינויים וחזרה ל commit האחרון - **Revert / Discard**

פעולות מתקדמות

יצירת העתק נפרד של העבודה הקיימת, וכן ההעתק הראשי - **Branch**

השוואת שני קבצים למציאת ההבדלים בין גירסאות שונות - **Diff / Change / Delta**

מיזוג העבודה שעשה מישהו אחר אל ההעתק המקומי - **Merge**

אי התאמה המונעת merge - **Conflict**

תיקון אי ההתאמה - **Resolve**

נעילת קובץ כך שאף אחד לא יוכל לערוך אותו - **Locking***

שבירת הנעילה (הנועל יצא לויפסנה והוא לא זמין) - **Breaking the lock**

תרחישי עבודה - הקמת Reposetory

1. המקים מבצע clone ל repo
2. הוספת קובץ readme וקבצים התחלתיים אחרים
3. דחיפת השינוי לשרת

```
$ git clone …
$ git add readme.txt
$ git add .
$ git commit -m “initial commit”
$ git remote add origin git@github.com:username/new_repo
$ git push -u origin master
```

תרחישי עבודה - כתיבת קוד

1. משיכת הקוד הקיים
2. הקמת branch מקומי חדש
3. עבודה על ה branch המקומי
4. סיום העבודה על ה branch המקומי

הערה: יש לבצע commit בסוף כל יום, ניתן לאחסן את ה branch בשרת ובסיום התהליך למחוק אותו

5. משיכת ה branch הראשי
6. מיזוג ה branch המקומי אל זה הראשי
7. פתירת אי התאמות אם יש
8. דחיפת השינויים לשרת

תרחיש עבודה - כתיבת קוד

תהליך כתיבת המשימה

$ git checkout -b task40
Switched to a new branch “task40”

//doing some changes
//vim index.html

$ git add .
$ git commit -a -m ‘DP counter is 16 bits [task 40]’

$ git checkout master
Switched to branch 'master'

תרחיש עבודה - כתיבת קוד

פתאום, קרתה תקלה, מסתבר שיש באג שמאוד דחוף לפתור כרגע

$ git checkout -b hotfix
Switched to a new branch 'hotfix'

//vim index.html

$ git commit -a -m 'AC timeout critical bug fixed'
fixed the AC timeout critical bug [hotfix 1fb7853]
(+)file changed, 2 insertions 1

$ git checkout master
$ git merge hotfoot
Updating f42c576..3a0874c

$ git branch -d hotfix
Deleted branch hotfix (3a0874c)

$ git push origin master

תרחיש עבודה - סיום כתיבת הקוד

ברוך השם, השמש תזרח גם מחר, נעלה עכשיו את הפיצ׳ר שכתבנו

$ git checkout task40
Switched to branch "task40"

// $ vim index.html
// continue working

$ git commit -a -m 'finished DP counter [task 40]', **#40**

$ git checkout master
Switched to branch 'master'

$ git merge task40

$ git branch -d task40

$ git push origin master

תרחיש עבודה - תוך כדי עבודה

תוך כדי העבודה על branch, אפשר לגבות את הדברים שנעשו, כדאי לבדוק שה git
רואה את השינויים שאמורים להיות, ולבצע עליהם commit
ניתן גם להעלות את השינויים לשרת

$ git status

On branch task41

$ git add .

$ git commit -m “phase 1 on new feature”

$ git push

פתירת Conflicts

בעיקרון, יש לבחור, הקוד שלי או הקוד שלו

בכלים שמנהלים את ה git עם UI זה יותר נחמד לדעתי

ב terminal ניתן להשתמש בכלים של פתירת בעיות כמו opendiff, שגם זה תכלס כלי של UI

כלים UI

Source Tree

GitHub client

דרכי התממשקות מאובטחת

https

ssh

משהו נוסף - git ignore

אם עובדים עם קודים צד שלישי, המנוהלים באמצעות כלי חכם דוגמת
cocoa-pods, composer
ודומיהם

לפעמים הספריות צד שלישי שוקלות לא מעט, ועדיף לשקול לא לכלול אותן בגיט

gitignore. לכן מגדירים קובץ

ושם ניתן לסמן קבצים ו\או תיקיות להתעלמות

```
$ touch .gitignore
```

```
file.txt
folder/
```

דוגמאות שימושיות נוספות תמצאו בקישור הבא

https://github.com/github/gitignore

סיימנו

שאלות?