ממשק משתמש גרפי

כתבנו תוכנה, ואנחנו רוצים לאפשר למשתמשים שלנו לגשת אליה דרך ממשק-משתמש גרפי (ממ"ג, GUI) – עם טפסים, כפתורים, וכו'... איך עושים את זה?

בג'אבה יש הרבה חבילות שמאפשרות לבנות ממ"ג – חבילות בשם awt, swing, processing, והחדשה ביותר נכון לעכשיו – javafx. כל החבילות הללו מאפשרות לבנות ממ"ג שהוא חלק מהתוכנה ויושב על אותו מחשב שהתוכנה יושבת עליו.

אנחנו לא נלמד על אף אחת מהחבילות הללו; במקום זה, נלמד גישה אחרת לבניית ממ"ג – נלמד איך לבנות ממ"ג בנפרד מהתוכנה, כאפליקציית לקוח-שרת (client-server application).
לאפליקציה כזאת יש שני חלקים נפרדים:

- השָׁרָת (server) הוא התוכנה שמבצעת את כל החישובים מאחרי הקלעים (ולכן נקרא גם back-end). במקרה שלנו השרת יהיה כתוב בשפת ג'אבה.
- הלָקוֹחַ (client) הוא הממ"ג – הטפסים, הכפתורים והתפריטים הגלויים למשתמש (ולכן נקרא גם front-end). במקרה שלנו הלקוח יהיה כתוב בשפות html+javascript.

מדוע אנחנו מעדיפים להפריד את השרת מהלקוח? מכמה סיבות:

1. שפת ג'אבה טובה מאד לביצוע חישובים מהירים, אבל לא כל כך טובה לבניית ממ"ג. שפות html+javascript הרבה יותר מתאימות ונוחות לבניית ממ"ג. אנחנו מעדיפים להשתמש לכל מטרה בשפה המתאימה ביותר עבורה (זה כישרון חשוב למתכנתים באופן כללי – לשלוט בכמה שפות ולבחור את השפה המתאימה לכל משימה).
2. תיכנות של שָׁרָת ותיכנות של ממ"ג דורשים כישורים שונים: תיכנות של שרת דורש ידע באלגוריתמים, מבני-נתונים, מאגרי-נתונים ועוד; תיכנות של ממ"ג דורש ידע בעיצוב, גרפיקה, אסתטיקה וכד'. בחברות תוכנה מקובל שאנשים שונים עובדים בכל אחד מהתחומים, למשל: אנשים שמתמחים באלגוריתמים עובדים בצד השרת (back-end) ואנשים שמתמחים בעיצוב עובדים בצד הלקוח (front-end).
3. הפרדת השרת מהלקוח מאפשרת להריץ את הלקוח על מחשב מרוחק מהשרת. אנחנו נוכל להריץ את תוכנת השרת על מחשב באוניברסיטה (או במקום אחר), ואנשים יוכלו להתחבר אליה מהבית דרך הרשת.
4. שפות html ו javascript רצות על כל דפדפן; היום כמעט בכל מחשב יש דפדפן, כך שכל אחד יכול להריץ ממ"ג שנכתב בשפות אלו, בלי שום צורך בהתקנת תוכנות חדשות.

איך הלקוח יוצר קשר עם השרת?

הבנו שהלקוח והשרת הם תוכנות נפרדות שיכולות לשבת גם על מחשבים נפרדים. אבל איך הם יוצרים קשר ביניהם? – דרך הדפדפן. כל דפדפן יכול ליצור קשר עם שרתים שנמצאים ברחבי הרשת. כשאנחנו כותבים כתובת של אתר-אינטרנט בשורת-הכתובת של הדפדפן, אנחנו למעשה מנחים אותו ליצור קשר עם שרת מרוחק. כתובת מהצורה:

```
http://<computer-address>/<path>?<query>
```

מנחה את הדפדפן שלנו ליצור קשר עם מחשב שנמצא בכתובת <computer-address>, ולשלוח לו את הבקשה <path>?<query>. החלוקה של הבקשה לשני חלקים - <path> ו-<query> - היא חלוקה לוגית לשם הנוחות והסדר בלבד.

לדוגמה, הכתובת http://tora.us.fm/tnk/find.php?q=abc מנחה את הדפדפן לגשת למחשב ששמו tora.us.fm, ולשלוח לו את הבקשה המורכבת ממסלול tnk/find.php ומשאילתא q=abc. המחשב שנמצא בכתובת tora.us.fm יודע לבנות ולהחזיר את הדף המתאים לבקשה.

איך בונים שרת?

עכשיו נראה איך אנחנו יכולים לבנות שָׁרָת בעצמנו, בשפת ג'אבה. ישנן הרבה חבילות המאפשרות לבנות שרתים בג'אבה, כגון: tomcat, jetty, spark ועוד. אנחנו נראה כאן דרך שאינה דורשת שום התקנה של חבילה מיוחדת – נשתמש בחבילה com.sun.net.httpserver.HttpServer.

הקוד המלא נמצא בתיקיית דוגמאות הקוד, src/lesson5/ReversingServer.java. כאן נתאר רק את הפרטים העיקריים.

השלב הראשון הוא ליצור שרת חדש. לשם כך, אנחנו צריכים להגיד לו באיזו **שלוחה** (port) להאזין. על כל מחשב יכולים לרוץ הרבה שרתים בו-זמנית, בתנאי שכל שרת מאזין לשלוחה אחרת. נניח שאנו רוצים שהשרת שלנו יאזין בשלוחה 8001. אז נאתחל את השרת ע"י:

```
HttpServer server = HttpServer.create(new
InetSocketAddress(8001), 0);
```

עכשיו, כדי לגשת לשרת שלנו, יצטרכו לציין גם את כתובת המחשב וגם את השלוחה, באופן הבא:

```
http://<computer-address>:8001
```

הדבר דומה למשרד שיש לו מספר-טלפון ראשי אחד והרבה שלוחות.

השלב הבא הוא להגיד לשרת שלנו איך לטפל בבקשות. כאמור, כשלקוח פונה לשרת, הוא שולח לו בקשה – הטקסט שנמצא אחרי שם המחשב (ומספר השלוחה) הוא הבקשה. כדוגמה ראשונה, נבנה שרת שמקבל בקשה שהיא מחרוזת, ומחזיר את המחרוזת ההפוכה. הלקוח יפנה לשרת שלנו באופן הבא:

```
http://<computer-address>:8001/reverse?abc
```

הבקשה היא הטקסט:

```
/reverse?abc
```

הבקשה מחולקת לשני חלקים – מסלול /reverse ושאילתא abc.

אנחנו נטפל בבקשה כך:

```
server.createContext("/reverse", request -> {
    String input = request.getRequestURI().getQuery();
    String output = new StringBuilder(input).reverse().toString();
    request.getResponseHeaders().set
          ("Access-Control-Allow-Origin", "*");
```

```
    request.getResponseHeaders().set
          ("Content-Type", "text/plain");
    request.sendResponseHeaders(200, 0);
    try (OutputStream os = request.getResponseBody()) {
        os.write(output.getBytes());
    }
});
```

אנחנו אומרים לשרת ליצור הֶקְשֶׁר (context), כלומר, לקשר בין תחילית של מסלול (במקרה זה reverse) לבין פעולה כלשהי שתקרה כשמישהו יבקש מסלול שמתחיל כך.
הפעולה מוגדרת בעזרת ממשק עם מתודה אחת; כפי שלמדנו באחד השיעורים הקודמים, ניתן לממש ממשק כזה ע"י ביטוי עם חץ ("ביטוי למדא"): לפני החץ נמצא הארגומנט (במקרה זה request) ואחרי החץ נמצא המימוש.

הארגומנט request מכיל בתוכו את כל פרטי הבקשה שנשלחה אלינו. כרגע אנחנו מתעניינים רק בשאילתא – query – החלק שאחרי ה"?". זהו הקלט שלנו. הפלט הוא הקלט במהופך.
אחרי שחישבנו את הפלט, אנחנו צריכים לשלוח אותו כתגובה (response) לבקשה.
לפני-כן אנחנו שולחים "כותרות" (headers): הראשונה נועדה להרשות לכל אדם מכל מקום ליצור קשר עם השרת שלנו; השניה מודיעה ללקוח שהתגובה היא מסוג טקסט פשוט (text/plain).
השלישית מודיעה ללקוח שהפעולה הסתיימה בהצלחה – קוד 200 (זה הקוד שאומר "הכל בסדר"). לא נאריך בזה יותר מדי – זה שייך (כנראה) לקורס בתקשורת. אחרי ששלחנו את הכותרות אנחנו כותבים את התגובה עצמה.

הריצו את השרת שנמצא בתיקיית הקוד, אתם אמורים לראות הודעה האומרת שהשרת רץ, ומציעה לכם לפנות לכתובת http://127.0.0.1:8001/reverse?abc. המספר 127.0.0.1 הוא הכתובת של המחשב שלכם ברשת המקומית של המחשב שלכם (בכל מחשב יש רשת מקומית הכוללת את המחשב עצמו). במקום 127.0.0.1 אפשר גם לכתוב localhost.
פתחו דפדפן ולכו לכתובת זו; אתם אמורים לראות את התגובה – cba.

הערות:

- נניח שהרצתם את השרת, ואז שיניתם את הקוד והרצתם שוב. מה יקרה? - כנראה תראו הודעת שגיאה (חריגה) האומרת לכם שהכתובת בשימוש. כיוון שהשרת הישן עדיין רץ ומאזין בשלוחה 8001, אי אפשר להריץ שרת חדש שיאזין באותה שלוחה. צריך קודם-כל לסגור את השרת הישן (באקליפס, לחצו על הריבוע האדום).
- יש שלוחות שהן כבר תפוסות ע"י תהליכים אחרים הרצים במערכת. למשל, סקייפ בדרך-כלל מאזין בשלוחה 80. לכן, אם תנסו להריץ שרת שמאזין בשלוחה 80, תקבלו הודעת שגיאה דומה. בחרו שלוחה אחרת.

איך ניגשים לשרת מרחוק?

עקרונית, כל עוד השרת שלכם רץ, והמחשב שלכם מחובר לרשת, אפשר לגשת אליו ממחשבים אחרים המחוברים לאותה רשת. לשם כך אתם צריכים להגיד למפעילי המחשבים האחרים את הכתובת של המחשב שלכם. איך מבררים אותה?

- ביוניקס, הפקודה היא ifconfig
- בחלונות: הפקודה היא ipconfig (יש לכתוב אותה ב-cmd).

חפשו על המסך כתובת המורכבת מארבעה מספרים, עם הכיתוב IPv4 Address או כיתוב דומה.
ברשת של הכיתה באריאל, הכתובת צריכה להתחיל ב 10.0. ואחר-כך עוד שני מספרים.
למשל, כתובת אפשרית היא: http://10.0.3.45

ואז, סטודנטים מאותה כיתה (המחוברים לרשת הכיתתית) יוכלו לגשת לתוכנה שלכם דרך:
http://10.0.3.45:8001/reverse?abc

תרגיל כיתה: כתבו בעצמכם שרת המקבל כקלט מחרוזת כלשהי ומבצע עליה פעולה כלשהי לפי בחירתכם (אפשר להוריד את הקוד של ReversingServer מגיטהאב ולשנות אותו לפי רצונכם). פרסמו את כתובת המחשב שלכם ותנו לתלמידים אחרים בכיתה לגשת לשָרָת שבניתם.

ראינו איך ניגשים לשרת דרך שורת הכתובת של הדפדפן. אבל אנחנו רוצים לגשת לשרת בצורה "אנושית" יותר – דרך טפסים וכפתורים. לשם כך נשתמש בשפה חדשה.

איך בונים ממ"ג ב-html?

בסעיף זה נלמד איך בונים טופס פשוט בשפת html. לא נלמד את כל הסודות של שפה זו אלא רק את המינימום הנדרש לצורך השיעור. המעוניינים להרחיב יכולים למצוא מדריכים רבים באינטרנט.

שפת html היא שפה לתיאור דפים הנקראים ע"י דפדפן. כל דף מורכב מעצמים (הנקראים בשפה זו "אלמנטים"), הנמצאים זה בתוך זה במבנה היררכי. כל עצם מוגדר ע"י **תג פותח** ו**תג סוגר**. בתג הפותח מופיע סוג העצם בסוגריים משולשים ובתג הסוגר מופיע לוכסן ואחריו סוג העצם בסוגריים משולשים.

העצם הראשי במסמך הוא מסוג html; הוא נפתח בתג <html> ונסגר בתג </html>.

בתוך עצם זה ישנם שני עצמים:

- אחד מסוג head, נפתח בתג <head> ונסגר בתג </head>.
- אחד מסוג body, נפתח בתג <body> ונסגר בתג </body>.

בתוך head יש עצמים המתארים את המסמך באופן כללי, כגון title – כותרת המסמך (שרואים בשורת הכותרת של הדפדפן) ו meta charset – הקידוד של הטקסט במסמך (בדרך כלל utf-8).

בתוך body נמצא גוף המסמך – בטקסט פשוט או בתוך עצמים נוספים. העצמים העיקריים הם:

- div – בלוק של טקסט (בשורות נפרדות);
- span – רצף של טקסט (בשורה אחת);
- form – טופס. בתוך טופס אפשר לשים:
  - input – שדה טקסט;
  - button – כפתור;
  - select – תפריט לבחירה (כל אפשרות לבחירה היא עצם מסוג option);
  - ועוד הרבה.

לכל עצם אפשר להוסיף **סגנון**. לשם כך יש לשים, בתוך התג הפותח של העצם, הגדרה של style. למשל:

```
<div style='background:yellow'>  abc </div>
```

מגדיר עצם מסוג div עם צבע-רקע צהוב, שבתוכו כתוב abc.

```
<form style='border:blue solid 1pt'><button>send</button></form>
```

מגדיר טופס מוקף במלבן כחול שבתוכו כפתור אחד עם כיתוב send.

הטקסט שבתוך ה-style הוא בשפת CSS; ניתן למצוא ברשת מדריכים רבים על השפה הזאת והפקודות שלה.

דוגמה בסיסית לדף html עם טופס נמצאת בקובץ client/reverse0.html. הדף הזה הוא סטטי בלבד – אפשר ללחוץ על הכפתור אבל לא יקרה שום דבר. כדי שמשהו יקרה על הדף, צריך להשתמש בשפה נוספת.

איך מוסיפים התנהגות לממ"ג ב-javascript?

בסעיף זה נלמד פקודות בסיסיות בשפת javascript. שוב, נלמד רק את המינימום הדרוש לצורך השיעור; להרחבה חפשו מדריכים באינטרנט.

אנחנו נלמד javascript יחד עם תוסף שנקרא jQuery, שהוא כל-כך נפוץ עד שאפשר כמעט להחשיב אותו כחלק מהשפה. התוסף בא בקובץ אחד עם סיומת js. יש להוריד את הקובץ ולשמור אותו על המחשב ליד הקובץ html שלכם. לאחר מכן יש להכניס אותו לקובץ html. לשם כך משתמשים בפקודת script. למשל, אם הקובץ של התוסף נקרא jquery-3.2.1.min.js, אז צריך להוסיף, בתוך ה-head של ה-html שלכם, את השורה הבאה:

```
<script src='jquery-3.2.1.min.js'></script>
```

השורה הזאת היא כמו import של ג'אבה – היא אומרת לדפדפן להכליל במסמך את פונקציות javascript שנמצאות בקובץ.

כדי להפעיל פקודות ב-javascript. צריך להוסיף עוד עצם script , הפעם בסוף המסמך – לפני התג הסוגר </body>. הנה קטע javascript מינימלי:

```
<script>
     alert("Hello world")
</script>
```

אם תוסיפו את הקטע הזה לקובץ html שלכם ותרעננו את הדף בדפדפן, תראו חלון קופץ ובו "Hello world". זה נחמד, אבל אנחנו לא רוצים שחלונות יקפצו בכל פעם שמישהו טוען את הדף – אנחנו רוצים שיקרו דברים רק כשמישהו לוחץ על הכפתור. לשם כך, במקום ה-alert הנ"ל, נכתוב את הקטע הבא בתוך ה-script (נמצא בקובץ client/reverse1.html):

```
$("button").click(
     function() {
          var input = $("input").val();
          alert("The input is " + input);
          return false
     }
)
```

השורה הראשונה ניגשת לכפתור button ומוסיפה פונקציה שאמורה לקרות כאשר מקליקים על הכפתור.

בתוך הפונקציה הזאת, השורה הראשונה ניגשת לשדה הטקסט input וקוראת את הערך שלו, והשורה השניה מקפיצה חלון עם הערך. השורה השלישית מחזירה ערך false האומר למערכת שלא צריך לטעון את הדף מחדש אחרי שלוחצים על הכפתור.

עכשיו כבר יש לנו דף-לקוח פעיל ודינאמי, אבל הוא עדיין לא יוצר קשר עם השרת. לשם כך נלמד פקודה חדשה.

איך יוצרים קשר עם השרת בעזרת jquery?

נשתמש בפקודה ajax, באופן הבא (ראו בקובץ reverse2.html):

```
$.ajax(
     {
          url: "http://127.0.0.1:8001/reverse?" + $("input").val()
     }
).then(function(output)
     {
          alert("The output I got is " + output)
     }
);
```

הפקודה מקבלת פרמטר אחד שהוא עצם עם שדות. בשפת javascript אפשר ליצור עצמים עם שדות בלי להגדיר את המחלקה שלהם לפני-כן; פשוט שמים את השדות בתוך סוגריים מסולסלים. במקרה שלנו, אנחנו יוצרים עצם עם שדה אחד בשם url, שערכו הוא הקישור לשרת שלנו – אותו קישור שראינו קודם. השאילתא היא המחרוזת שנכתבה בתוך ה-input.

הקריאה ל-ajax היא אסינכרונית. כשהקריאה תסתיים, נגיע לפונקציה שנמצאת בתוך ה-then, ואז נציג חלון קופץ עם התשובה.

איך מאפשרים ללקוחות מרוחקים לקרוא את ה-html שלנו?

ראינו שלקוחות מרוחקים יכולים לגשת לשרת שלנו, אבל הלקוח (כלומר קובץ ה html) עדיין נמצא על המחשב שלנו. כדי לתת ללקוחות מרוחקים לקרוא גם את קבצי הלקוח, פשוט נוסיף לשרת עוד הֶקְשֶׁר. בכל פעם שמישהו ישלח לשרת שלנו בקשה שהמסלול שלה הוא מהצורה:

```
/file/<filename>
```

השרת יקרא את הקובץ המתאים מתיקיית ה-client ויחזיר אותו.

הקוד המלא נמצא ב-ReversingServer.java. כאן נראה קוד מקוצר (שאינו כולל בדיקה אם הקובץ כבר קיים):

```
server.createContext("/file", request -> {
     String fileName =
request.getRequestURI().getPath().replaceAll("/file/", "");
     Path path = Paths.get("client", fileName);
     String output = new String(Files.readAllBytes(path),
StandardCharsets.UTF_8);
```

```
     request.getResponseHeaders().set("Access-Control-Allow-
Origin", "*");
     request.getResponseHeaders().set("Content-Type",
"text/html");
     request.sendResponseHeaders(200, 0);
     try (OutputStream os = request.getResponseBody()) {
        os.write(output.getBytes());
     }
});
```

עכשיו, נשנה את הקוד בדף ה-html כך שלא יפנה לשרת 127.0.0.1 (שהוא המחשב המקומי), אלא יפנה לכתובת יחסית. בקריאה לפקודה ajax, במקום:

```
url: "http://127.0.0.1:8001/reverse?" + input
```

נכתוב:

```
url: "/reverse?" + input
```

הדף המשופר נמצא ב- client/reverse3.html.

עכשיו, במקום לפתוח את הדף ישירות מהמחשב המקומי שלנו, נפתח אותו כך:

```
http://127.0.0.1:8001/file/reverse3.html
```

הדפדפן יקרא לשרת שלנו וישלח את הבקשה file/reverse3.html והשרת יחזיר לו את הקובץ המתאים.
מי שנמצא על מחשב מרוחק, יוכל לכתוב במקום 127.0.0.1 את הכתובת של המחשב שלנו (למשל 10.0.34.45) ויראה את אותו קובץ.

לסיום נעשה עוד שינוי קטן בקוד ה-javascript שלנו. במקום להקפיץ חלון בכל פעם שמישהו לוחץ על הכפתור, אנחנו נוסיף את התשובה של השרת בתוך דף ה-html. לשם כך נוסיף לדף בלוק טקסט (div) עם מזהה (id):

```
<div id='output'></div>
```

אם רוצים, אפשר גם להוסיף לו style.
בפעולה שבאה אחרי ה-ajax (בתוך ה-then), במקום להקפיץ חלון עם התשובה, נכתוב:

```
$("div#output").append(
     "<div>The reverse of '"+input+"' is '"+output+"'</div>"
)
```

הפקודה הזאת ניגשת ל-div עם המזהה output, ומוסיפה לו קוד html. הקוד שמוסיפים הוא בעצמו div, וכתובה בו התשובה לשאילתא.
לחצו על הכפתור כמה פעמים עם קלטים שונים; תראו איך התשובה נוספת לתוך ה-div.

איך יוצרים גרפיקה ב-html?

יש שתי דרכים עיקריות להוסיף גרפיקה חופשית לדף html: אחת משתמשת בעצם canvas והשניה בעצם svg. אנחנו נשתמש ב-svg.

בתוך עצם מסוג svg אפשר להוסיף עצמים גרפיים מוקפים בתגים כמו בכל עצם של html. למשל, הקטע הבא:

```
<svg width='400' height='200'>
   <rect x="5" y="5" width="390" height="190" style="stroke:green;
stroke-width:3; fill:yellow" />
   <circle cx="50" y="50" r="50" style="fill:red" />
</svg>
```

מוסיף עצם svg חדש ברוחב 400 נקודות ובגובה 200 נקודות, שבתוכו מלבן צהוב ועיגול אדום.

אפשר להשתמש בפקודת append שלמדנו קודם כדי להוסיף צורות לתוך ה-svg באופן דינמי. למשל, הפונקציה הבאה מוסיפה עיגול כחול ברדיוס 5 בנקודה מסויימת (x,y):

```
function point(x,y) {
     $("svg").append(
          '<circle cx="' + x + '" cy="' + y + '" r="5" fill="blue"
stroke-width="0"/>'
     )
}
```

בדף client/plot0.html ישנו דף html+javascript (ללא צורך בשרת) המקבל כקלט מערך של נקודות ומציגה את הנקודות על svg. לא ניכנס לכל הפרטים של הדף; נתעכב כאן רק על הפורמט של הקלט. הפורמט הוא JSON – Javascript Object Notation. הוא מאפשר להגדיר אובייקטים כלשהם בקלות. במקרה זה, אנחנו מגדירים מערך (מוקף בסוגריים מרובעים), שבו כמה עצמים. כל עצם מוקף בסוגריים מסולסלים, ובתוכם יש רשימה של שדות וערכים. שם השדה מוקף בגרשיים, אחר-כך נקודתיים, ואחר-כך ערך השדה. למשל, הטקסט הבא מגדיר מערך ובו 4 נקודות:

```
[
     {"x":5,"y":5},
     {"x":395,"y":5},
     {"x":395,"y":195},
     {"x":5,"y":195}
]
```

ניתן לתרגם מחרוזת כזאת לעצם של ג'אבהסקריפט ע"י הפקודה: JSON.parse(text).
ניתן לתרגם עצם למחרוזת ע"י הפקודה: JSON.stringify(text).
שימו לב להבדל בין ג'אבה לג'אבהסקריפט: בג'אבה כדי להגדיר עצם חייבים לפני-כן להגדיר מחלקה שבה מגדירים מראש איזה שדות יהיו בכל עצם מהמחלקה. בג'אבהסקריפט אפשר להגדיר עצם בלי להגדיר מחלקה – פשוט מקיפים את השדות בסוגריים מסולסלים.

לחיצה על כפתור Submit מפעילה פונקציה אשר מבצעת parse על הטקסט שבתיבה, הופכת אותו לאובייקט, עוברת על כל הנקודות במערך ומציירת אותן.

השלב הבא הוא לבנות שָׁרָת בג'אבה, המקבל מחרוזת המייצגת פולינום, ומחזיר מחרוזת המייצגת אוסף של נקודות בפורמט הנ"ל. הקוד נמצא ב-PolynomServer.
הקוד שהופך מחרוזת לפולינום נמצא בבנאי של המחלקה Polynom בחבילה myMath.
אחרי שיוצרים פולינום, עוברים בלולאה על כל ערכי x בתחום, לכל ערך x מחשבים את ערך y המתאים (ע"י המתודה f של הפולינום), ויוצרים את המחרוזת המייצגת מערך של נקודות ב-JSON.

הערה: השרת כולל טיפול בסיסי בחריגות. חשוב מאד לטפל בחריגות בתוך השרת, כדי שהלקוח יקבל תשובה משמעותית במקרה שעשה שגיאה.

הריצו את השרת PolynomServer, נסו אותו קודם-כל דרך הדפדפן, אחר-כך דרך הטופס plot1.html, ובסוף בגישה מרחוק דרך כתובת ה-IP של המחשב שלכם. בהצלחה!!

סיכם: אראל סגל-הלוי.