ספר זה הוא יצירה המוגנת בזכויות יוצרים. אתה קיבלת רשיון לא-בלעדי, לא-ייחודי, אישי, בלתי ניתן להעברה (למעט על פי דין), ובלתי ניתן להסבה לעשות שימוש אישי בספר זה לצרכים לימודיים בלבד.

אסור לך להעתיק את הספר, לשכפל אותו, לצור יצירות נגזרות ממנו או לפרסם אותו בכל צורה אחרת.

מותר לך לצטט קטעים קצרים מהספר במסגרת הגנת שימוש הוגן ,כלומר פסקה או שתיים, כאשר אתה מפנה למקור ומזכיר את רן בר-זיק כמחבר הספר.

הדוגמאות המובאות בספר זה הן בבעלות של רן בר-זיק, ואסור לך להשתמש בהן בתוך תוכנות שתפתח. אם אתה רוצה להכניס אותן לפרויקט שלך, שלח מייל ונדבר על זה.

ללמוד jQuery בעברית

מהדורה 1.0.0

רן בר-זיק

תוכן העניינים

על הספר 7

על המחבר 8

על jQuery 9

איך jQuery עובדת 10

ה-$ 15

מניעת התנגשויות עם ספריות אחרות באמצעות IIFE 15

DOM ו-jQuery 18

תרגיל 19

פתרון 19

HTML ו-CSS בסיסי 21

תחרות בין סלקטורים 35

בדיקה בכלי המפתחים 39

הוספת עיצוב לתגיות באמצעות תכונת style 40

תרגיל בסיס חשוב 41

סלקטורים 44

איך סלקטורים עובדים בג'אווהסקריפט 44

איך סלקטורים עובדים עם jQuery 46

סלקטורים עם תוצאות מרובות 48

סלקטורים מורכבים 50

דוקומנטציה 54

תרגיל 55

פתרון 55

תרגיל 58

פתרון 58

תרגיל 59

פתרון 59

תרגיל 60

פתרון 60

עיצוב CSS 61

שימוש בסיסי 61

שינוי כמה תכונות 64

קבלת מידע על תכונה 65

שינוי קלאסים 67

תרגיל 69

פתרון 69

תרגיל 70
פתרון..... 70
אירועים 71
אירוע קליק 71
רפרנס עצמי..... 80
כתיבה אלגנטית יותר של קוד 83
שימוש בשם אירוע 84
תרגיל 85
פתרון..... 85
תרגיל 86
פתרון..... 86
תרגיל 87
פתרון..... 87
מניפולציה ויצירת אלמנטים..... 88
קבלת ערכים מאלמנטים מסוג input..... 88
יצירת אלמנט 89
הוספת אירועים לאלמנט שהוספנו..... 92
מחיקת אלמנט 95
Traversing..... 96
עריכת תוכן של אלמנט 102
תרגיל 103
פתרון..... 103
תרגיל 104
פתרון..... 104
אפקטים..... 105
תרגיל 113
פתרון..... 113
תרגיל 114
פתרון..... 114
תרגיל 115
פתרון..... 115
AJAX..... 116
אובייקט jqXHR..... 121
תרגיל 126
פתרון..... 126
תרגיל 128
פתרון..... 128
שימוש בתוספים 130

תרגיל 134
פתרון..... 134
סיכום 136
ומה עכשיו? 136
נספח: וורדפרס ו-jQuery..... 137
בדיקת גרסת ה-jQuery בוורדפרס 137
מיקום קובץ ה-jQuery 139
הכנסת הקוד למערכת..... 139
הכנסת סקריפט לדף האדמין..... 142

על הספר

הספר הקצר "ללמוד jQuery בעברית" מלמד את ספרית jQuery מההתקנה ועד לשימוש בתוספים. כאשר הוא גם מלמד מעט על CSS ו-HTML – או לפחות את הבסיס שלהם. ספרית jQuery היא ספריה פופולרית מאוד, שנמצאת במיליוני אתרים וכן מגיעה כברירת מחדל עם כל אתר וורדפרס שהוא. כיום ממעטים לבנות מערכות קיימות באמצעותה, אך היא מאוד פופולרית בקרב בוני אתרים שרוצים פעולות ג'אווהסקריפט מצומצמות ולא לבנות מערכות שלמות עם ג'אווהסקריפט

הספר הקצר מיועד למתכנתים המכירים את ג'אווהסקריפט לעומק והוא מהווה המשך ישיר לספר "ללמוד ג'אווהסקריפט בעברית" ויוצא מנקודת הנחה שהנכם מכירים את העקרונות מאחורי שפת ג'אווהסקריפט, יודעים להפעיל סביבת פיתוח מסוג ויז'ואל סטודיו קוד וכן מכירים תכנות אסינכרוני. ובנוסף יש לכם ידע בסיסי ב-HTML וב-CSS.

בספר אנו נלמד על הספריה, כיצד עובדים איתה ומה היא מסוגלת לעשות. נבין איך אפשר להפעיל בה אירועים, ליצור בה טפסים, לשלוח ולקבל מידע וכמובן לעשות אפקטים מרהיבים. ל-jQuery יש תוספים רבים וכן ספריות עזר רבות כמו jQuery UI שמעשירות את יכולת הספריה הבסיסית. אנו נלמד להשתמש גם בהם.

על המחבר

רן בר-זיק הוא מפתח תוכנה משנת 1996 במגוון שפות ופלטפורמות ועובד כמפתח בכיר במרכזי פיתוח של חברות רב-לאומיות, מ-HPE ועד Verizon, שם הוא מפתח בטכניקות מתקדמות הן בצד הלקוח הן בצד השרת, ושם דגש על בניית תשתית פיתוח נכונה, על שימוש ב-CI\CD וכמובן על אבטחת מידע.

נוסף על עבודתו כמפתח במשרה מלאה, רן הוא עיתונאי ב"הארץ" במדור המחשבים, שם הוא מסקר נושאים הקשורים לטכנולוגיה ולאבטחת מידע וכותב על אינטרנט ורשתות.

משנת 2008 מפעיל רן את האתר "אינטרנט ישראל" (internet-israel.com), שהוא אתר טכני המכיל מדריכים, מאמרים והסברים על תכנות בעברית ומתעדכן לפחות פעם בשבוע.

רן נשוי ליעל ואב לארבעה ילדים: עומרי, כפיר, דניאל ומיכל. רץ למרחקים ארוכים וחובב טולקין מושבע.

ספרין הקודמים של בר-זיק: "ללמוד ג'אווהסקריפט בעברית", "ללמוד Node.js בעברית", "ללמוד ריאקט בעברית" ו"ללמוד MySQL בעברית".

על jQuery

ספרית jQuery נוצרה לראשונה בשנת 2006 על ידי המתכנת ג'ון רסיג, כספרית קוד פתוח אשר מטפלת ב-DOM: ג'אווהסקריפט בצד לקוח שמפעיל ומעשיר את היכולות של ג'אווהסקריפט בניהול ותפעול דפי HTML. למרות קיומן של ספריות אחרות, בזכות המבנה הפשוט שלה והדוקומנטציה המצוינת, היא הפכה לפופולרית מאוד כמעט מההתחלה. בשעת כתיבת שורות אלו, jQuery נמצאת ב-75% ממיליון האתרים הפופולריים ביותר בעולם ובמיליוני אתרים אחרים.

ספרית jQuery קלה להתקנה ושימוש ואיפשרה למתכנתים שעבדו בסביבת ווב כאוטית ובעייתית, עם דפדפנים שונים שהציגו ג'אווהסקריפט ו-CSS בדרכים שונות, לעבוד באופן מאוד פשוט והיתה הדרך המובילה לבנות סביבות עשירות ומחוכמות. היא היתה פורצת הדרך הכל מה שקשור להפיכת צד הלקוח לחלק חשוב בתכנות אינטרנט וסללה את הדרך לספריות אחרות כמו אנגולר, ריאקט ו-vue.

עם השנים הועם זהרה של jQuery, למרות שהיא נמצאת במיליוני אתרים וגם כברירת מחדל במערכת הפופולרית וורדפרס, מעט מתכנתים משתמשים בה לבניית מערכות מאפס ונוטים לגשת לספריות אחרות כמו ריאקט למשל – עדיין יש מיליוני שורות קוד שעובדות ב-jQuery והיא עדיין פופולרית בסקריפטים ותוספים עם פעילות מוגבלת. אם אתם צריכים וידג'ט קטן שמציג מחירי מניות, שולח או מקבל מידע, מציג גלריה או פעולות קטנות נוספות הקשורות להנפשה, יש סיכוי ש-jQuery יתאים לכם. אם את בוני אתרים בוורדפרס, הידע ב-jQuery יאפשר לכם לתקן תקלות שונות הקשורות לתוספים בפלטפורמה ולבצע המרה לעברית או לעבודה באתרים שכתובים מימין לשמאל. פרילנסרים שצריכים לבצע פרויקטים קטנים, בוודאי יוכלו להפיק תועלת מהספר ומההכרות עם jQuery. גם הלימוד שלה קל מאוד.

איך jQuery עובדת

בבסיס, jQuery נטענת לסקופ הגלובלי – כלומר היא זמינה בכל נקודה בסקריפט שלכם מרגע מהטעינה בסקופ הגלובלי – כלומר window או globalThis. מדוב בסקופ הראשי של הדפדפן שאפשר לקרוא לו בכל רגע נתון ובכל נקודה בסקריפט. מה שהספריה עושה ברגע שהיא נטענת זה להעמיס לסקופ הזה את כל המידע שיש בספריה ומרגע הטעינה אפשר פשוט להשתמש בקוד מבוסס jQuery. זה נשמע פשוט וזה אכן פשוט.

את הטעינה של jQuery ניתן לבצע בכמה דרכים. אך הדרך הפשוטה והמקובלת היא באמצעות קריאה לספריה של jQuery כפי שעושים לכל סקריפט – באמצעות תגית ה-script שיש בדף. כאשר קובץ הספריה יכול להיות בשרת שלכם או בשרת מרוחק. מקובל, ברוב המקומות, לקרוא לקובץ jQuery משרת מרוחק בשם CDN. ראשי תיבות של Content Delivery Network. רשת של שרתים שמכילה קבצים של ספריות פופולריות.

ה-CDN של jQuery נמצא בכתובת הזו: https://code.jquery.com ואם תכנסו אליה, תוכלו לראות שיש לנו כמה גרסאות – 1,2 ו-3. אנו רוצים להשתמש בחדשה יותר, אז אנו נשתמש ב-3. גם פה יש לנו כמה גרסאות: גרסת פיתוח לא מכווצות וגרסאות מכווצות יותר. כדאי לנו לבחור בגרסה המכווצת: minified.

שימו לב: יש גם גרסת slim שלא מכילה אנימציות. אנו משתמשים בגרסה המלאה. minified בלבד.

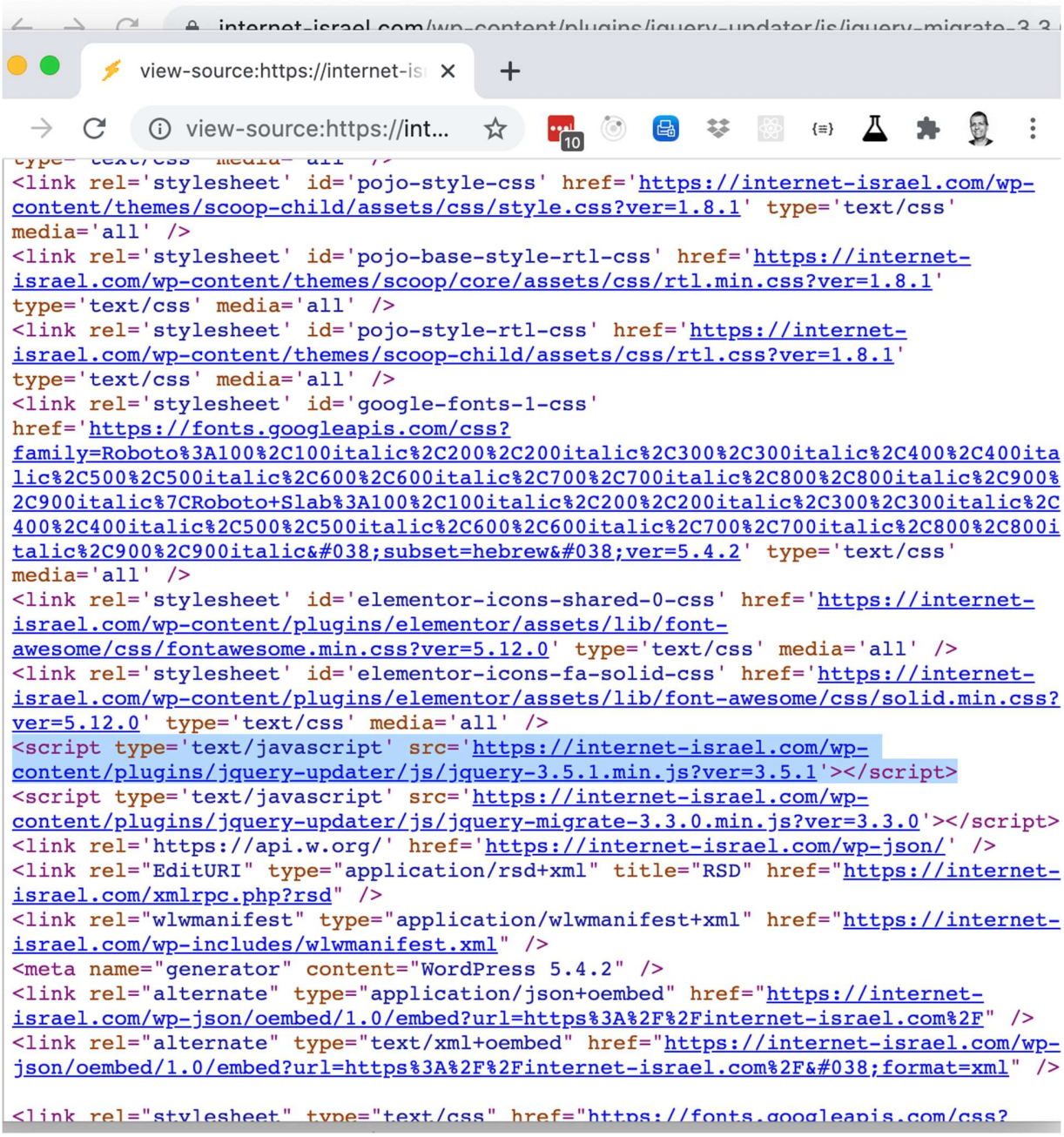
את כתובת האינטרנט אנו נעתיק ונציב מעל תגית הסקריפט שבה יש את הקוד שלנו או התגית שבה אנו קוראים לקוד שלנו:

```
<!doctype html>
<html>

<head>
  <meta charset="utf-8">
</head>

<body>
  <script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
  <script>
    // Our code.
  </script>
</body>

</html>
```

ניתן לראות שהגרסה שבה אני משתמש היא 3.5.1 והיא גרסת minified. כלומר גרסה שהסירו ממנה את כל הרווחים וצמצמו אותה כך שתהיה קטנה ככל האפשר.

מייד אחרי שאנו טוענים את ה-jQuery, אנו יכולים לעבוד.

אפשר לראות טעינה של jQuery גם בעולם האמיתי בכל אתר וורדפרס. למשל, בבלוג האישי שלי, אפשר לראות שקובץ ה-jQuery נטען ממש בתחילת הדף. זה אומר שכל קובץ שבא אחרי הטעינה הזו כבר יכול להשתמש ב-jQuery.



ברגע שספרית jQuery נטענת, יש לנו את אובייקט jQuery זמין. אם תשמרו את הקוד שלעיל כקובץ HTML ותכנסו לכלי המפתחים, שאיתו למדנו לעבוד בספר "ללמוד ג'אווהסקריפט בעברית", תכנסו לקונסולה ותקלידו jQuery. תוכלו לראות שאכן קיימת כזו פונקציה. על גבי הסקופ הגלובלי.

מדובר בעצם בפונקציה, שממנה מתחיל הכל. הפונקציה הזו מחזירה אובייקט שחושף המון מתודות שמעשירות אותו. זה נשמע מורכב אך בקרוב זה יראה פשוט. בסופו של דבר כל ספרית jQuery מתנקזת לאובייקט הזה. אובייקט שמכיל המון כלים שימושיים אבל שכולם נקראים ומופעלים ממנו. מצד אחד זה הופך את ההפעלה של jQuery לקלה במיוחד – הפעל את הפונקציה לתוך אובייקט או אובייקט DOM והשתמש במתודות. מצד שני, זה עלול להעמיס על הזכרון ולגרום לבעיות ביצועים. אבל אנחנו בתהליך הלמידה ורוצים להתחיל לעבוד. אז כדאי שנבדוק שבאמת הגרסה שלנו תקינה.

העתיקו והדביקו את הקוד הזה ופיתחו אותו עם הדפדפן שלכם. פיתחו את כלי המפתחים ותוכלו לראות את הגרסה של jQuery.

```
<!doctype html>
<html>

<head>
  <meta charset="utf-8">
</head>

<body>
  <script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
  <script>
    const jQueryInstance = jQuery();
    console.log(jQueryInstance.jquery); // 3.5.1 ...
  </script>
</body>

</html>
```

מה שעשינו פה הוא דבר פשוט. לקחנו אובייקט ואיכלסנו אותו בתוצאה של אובייקט jQuery.

באובייקט הזה יש את כל המתודות והתכונות של הספריה. עד כדי כך. אחת מהתכונות היא jquery והיא מציגה את גרסת jQuery. במקרה הזה 3.5.1 וכן את ספריות ברירת המחדל שנטענות יחד עם הספריה. אם הכל תקין, הקישור לספרית jQuery עובד והכל בסדר, זה מה שנראה.

ה-$

לפונקצית jQuery יש קיצור שהפך להיות מאוד מזוהה איתה – ה-$. במקום להקליד jQuery כפונקציה, אנו יכולים להקליד $. כלומר הקוד הזה יעבוד היטב:

```
    const jQueryInstance = $();
    console.log(jQueryInstance.jquery); // 3.5.1 ...
```

הקיצור הזה מאוד מקובל בסקריפטים של jQuery ואנו נשתמש בו. צריך רק לזכור ש-$ הוא jQuery. כמעט בכל קוד של jQuery יש שימוש בקיצור הזה.

מניעת התנגשויות עם ספריות אחרות באמצעות IIFE

כיוון ש-jQuery עובדת בסקופ גלובלי, וגם הסקריפטים שלה כתובים גם הם בסקופ גלובלי, שימוש בה עלולה לזהם את הסקופ הגלובלי ולגרום לבעיות לא צפויות. זו הסיבה שמאוד כדאי כבר כעת להתרגל לעטוף את הסקריפטים שלנו ב-IIFE שהן ראשי תיבות של Immediately Invoked Function Expression. מדובר בדרך פשוטה ויעילה ליצור סקופ פרטי.

הבה ונזכר בעקרונות בסיסיים של ג'אווהסקריפט שמתכנתי ג'אווהסקריפט מכירים. נניח ואנחנו כותבים בסקופ הגלובלי, כשאנו כותבים בסקופ הזה, התנגשות של משתנים עלולה להחריב את הסקריפט שלי או לגרום שגיאות. למשל:

```
<script>
  const a = 'This is a!'
</script>
<script>
  const a = 'This is my A!';
  console.log(a);
</script>
<script>
```

```
  console.log(a);
</script>
```

התוצאה של הקוד הזה תהיה שגיאה כמובן. כיוון ש-a כבר הוגדר קודם בסקריפט הראשון בסקופ הגלובלי. ניסיון להגדירו שוב יביא לשגיאה. הפתרון הוא ליצור בסקריפט האמצעי סקופ פרטי באמצעות IIFE.

```
<script>
  const a = 'This is a!'
</script>
<script>
  (() => {
    const a = 'This is my A!';
    console.log(a);
  })();
</script>
<script>
  console.log(a);
</script>
```

המבנה הזה:

```
(() => {
  // IIFE
})();
```

הוא סקופ פרטי. כל מה שרץ בתוכו נמצא בסקופ משל עצמו ולא מזהם את הסקופ הגלובלי.

זה שימושי במיוחד ב-jQuery, גם לא לזהם את הסקופ הגלובלי וגם כדי לוודא שאנו יכולים להשתמש בסימן $. יש ספריות אחרות שמשתמשות ב-$ ומדובר בסימן מאוד פופולרי לספריות. זו הסיבה שהקוד שלנו לא יכול לסמוך על כך שהסימן $ שייך ל-jQuery. אם נשתמש ב-IIFE אנחנו יכולים להשתמש ב-$ כיוון שהוא בסקופ הפרטי שלנו.

```
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
  const $ = 'some library that uses $';
</script>
<script>
  (($) => {
    const jQueryInstance = $();
    console.log(jQueryInstance.jquery); // 3.5.1 ...
  })(jQuery);
</script>
<script>
  console.log($); // some library that uses $
</script>
```

ניתן לראות בדוגמה לעיל, שלמרות שאני דורס את סימן ה-$, אני עדיין יכול להשתמש בו בתוך ה-IIFE כי אני מגדיר אותו מחדש. בתוך הסקופ הפרטי שלי, שיצרתי עם ה-IIFE, הסימן $ קשור ל-jQuery ותמיד יהיה קשור. מחוץ לסקופ? יהום הסער, ה-$ מכיל 'some library that uses $'. אבל בתוך הסקופ שלנו, אני יכול להשתמש ב-$ כיוון שהוא מקושר ל-jQuery.

כל הקוד שאני אראה מהנקודה הזו, יהיה עטוף תמיד ב-IIFE. זו פרקטיקה מאוד מקובלת ומאוד מומלצת שמונעת, מניסיון, המון בעיות. לא תאמינו כמה ספריות משתלטות על ה-$ בלי צורך. כך אנו יכולים למנוע מצב שהקוד שלנו לא יעבוד. גם אנו לא צריכים לדאוג שהמשתנים שלנו ידרסו משתנים אחרים בסקופ הגלובלי.

jQuery-ו DOM

jQuery עובדת היטב עם אלמנטי DOM – כלומר אלמנטים של HTML שמועברים לג'אווהסקריפט ובפרק הבא נרחיב על כך. ברגע שאנו עובדים עם אלמנטים כאלו בלבד, בניגוד לג'אווהסקריפט רגיל, שבו אנו יוצרים משתנים, אובייקטים וכל הקוד שלנו רץ בשפה טהורה – אנו צריכים לוודא שהאלמנטים האלו נטענים וקיימים ב-HTML לפני הקוד שלנו.

יש כמה דרכים לעשות זאת. הדרך האלגנטית ביותר היא לשים את הסקריפט שלנו בתחתית ה-HTML. כך, כשהוא ירוץ, כל ה-HTML ייטען לזכרון. דרך נוספת היא להכניס תנאי מפורש עם document ready באופן הזה:

```
((($) => {
   $(document).ready(function(){
     console.log('runs when DOM is ready!');
   });
})(jQuery);
```

פונקצית ה-document ready גרמה לקוד לרוץ רק לאחר שכל האלמנטים בדף נטענו. היא בעצם נצמדת לאירוע ready של document שנורה רק לאחר שכל ה-HTML נטען. הקוד הזה מקובל במיוחד בקוד שלא תמיד יש לנו שליטה איפה הוא קיים. אם אתם כותבים תוסף או סניפט – קטע קוד קצר שבונה אתר יכול להציב באתר איפה שהוא רוצה – מומלץ בחום רב להציב את הקוד הזה.

בדוגמאות אני משתמש בקוד הזה.

תרגיל

צרו דף HTML שבו יש קוד jQuery מה-CDN וכן קוד שלכם שמשתמש ב-IIFE וב- document .ready. בתוך הקוד ירוץ console.log שבו כתוב Ready to write some jQuery code.

פתרון

```
<!doctype html>
<html>

<head>
  <meta charset="utf-8">
</head>
<body>

<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
(($) => {
   $(document).ready(function(){
     console.log('Ready to write some jQuery code');
   });
})(jQuery);
</script>
</body>

</html>
```

הפתרון פה הוא פשוט. יש לנו דף HTML פשוט, עם קוד HTML. אנו קוראים לסקריפט של jQuery באמצעות תגית SCRIPT עם SRC שמצביע על מיקומו של הסקריפט. במקרה שלנו ב-CDN.

מייד אחריה, יש תגית SCRIPT שמכילה את הקוד שלנו. הקוד שלנו נמצא בתוך IIFE:

```
((($) => {

})(jQuery);
```

שמופעלת מייד בתחילת הסקריפט ויוצרת סקופ פנימי משלה שה-$ שמור ל-jQuery.

בתוך ה-IIFE אנו שמים את document ready שמופעל ברגע שמסמך ה-HTML נטען ואנו יכולים להתחיל לעבוד:

```
  $(document).ready(function(){
    console.log('Ready to write some jQuery code');
  });
```

HTML ו-CSS בסיסי

jQuery עובדת עם אלמנטים של DOM. כלומר אלמנטים של HTML כפי שהם מתורגמים לג'אווהסקריפט. על מנת לדעת איך לעבוד איתם, אנו נעשה חזרה על בסיס ה-HTML, הלא היא התחביר הבסיסי של דף האינטרנט וה-CSS, שהוא השרירים.

דפי אינטרנט מורכבים מ-HTML שזה ראשי תיבות של Hyper Text Markup Language. זו לא שפה אלא תחביר – דרך מקובלת לכתוב ישויות כדי שדפדפן ידע לעבוד איתם. זה נשמע מורכב אבל בפועל מאוד פשוט. אם נלחץ באמצעות מקש העכבר הימני על כל דף שהדפדפן מציג ונבחר ב"הראה מקור" או "view source". אנו נראה המון טקסט שנראה כך:

```
<תגית>
   <תגית נוספת>
      <תגית שנסגרת בעצמה ולא מכילה מידע />
    <תגית נוספת>מידע טקסטואלי</תגית נוספת>
 <תגית נוספת />
</תגית>
```

תגית HTML היא כל מה שיש סביבו <>. ישנן כמה סוגי תגיות כאשר לכל תגית תפקיד משלה ועיצוב משלה בדפדפן. את התגיות כותבים באותיות קטנות. אם, בדף שבו כתבנו את הסקריפט של jQuery נכניס תגיות, נוכל לראות את השינוי בדפדפן. ממש כמו בג'אווהסקריפט.

למשל תגית h1 שמסמנת כותרת ראשית. תקן WC3, הקונסורציום העולמי, שאחראי על התקינה, קובע שיש רק תגית כזו במסמך HTML אחד.

```
<body>
  <h1>כותרת</h1>
</body>
```

ה-HTML נמצא בד"כ בין תגיות ה-body במסמך HTML תקני. אנו שומרים את המסמך ופותחים אותו עם כל דפדפן. בדיוק כפי שאנו עושים עם ג'אווהסקריפט.

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
</head>

<body>
  <!-- HTML Goes here-->
  <script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
  <script>
    (($) => {
      $(document).ready(() => {
        console.log('Ready to write some jQuery code');
      });
    })(jQuery);
  </script>
</body>

</html>
```

חלק מההתגיות יכולות להכיל תגיות אחרות. למשל תגית p – פסקה (paragraph באנגלית) יכולה להכיל תגית strong – דגש.

```
<body>
  <h1>כותרת</h1>
  <p>
    <strong>טקסט מודגש</strong>
  </p>
</body>
```

תגיות יכולות להכיל כמה תגיות אחרות. למשל תגית ul – ראשי תיבות באנגלית של unordered list – רשימה לא מסודרת, יכולה להכיל תגיות li – רשימה.

```
<body>
  <h1>כותרת</h1>
  <p>
    <strong>טקסט מודגש</strong>
  </p>
  <p>
    <ul>
      <li>פריט אחד</li>
      <li>פריט שני</li>
      <li>פריט שלישי</li>
    </ul>
  </p>
</body>
```

ניתן לכתוב טקסט באופן חופשי בתוך תגיות מסוימות, לעתים לצד תגיות אחרות. למשל, אם אני רוצה **להדגיש** חלק ממשפט מסוים שנמצא בפסקה מסוימת, אכתוב את הטקסט ורק החלק המודגש יהיה בתוך תגית strong. למשל:

```
<body>
  <p>
    פסקה עם טקסט<strong>מודגש</strong>. שאר הטקסט רגיל.
  </p>
</body>
```

ישנן תגיות שלא יכולות להכיל טקסט או תגיות אחרות. תגיות כאלו, כמו למשל שבירת שורה: br. לא יכילו דבר ויסגרו את עצמן עם קו נטוי. כך למשל כאן יש לי שתי תגיות – br המוקדשת לשבירת שורה ו-hr המוקדשת לקו אופקי שלא יכולות להכיל שום תגית אחרת או טקסט, הן נסגרות עם קו נטוי וחץ:

```
<body>
  <br />
  <p>
    פסקה כלשהי.
  </p>
  <hr />
</body>
```

מבנה התגיות הוא פשוט והוא אמור להיות פשוט כל עוד מקפידים על העקרונות – כל תגית תמיד נסגרת ואלו שלא? נסגרות בעצמן. אין כזה דבר תגית פתוחה.

בנוסף, לכל תגית יכולות להיות תכונות. התכונות כתובות ליד שם התגית באופן הזה:

<תגית תכונה1="ערך התכונה" תכונה2="ערך התכונה" תכונה4="ערך התכונה"></תגית>

התכונה הזו, באנגלית attribute, יכולה להשפיע על התגית או לסמן אותה. למשל, תכונת href בתגית a, קישור, מצביעה לאן הקישור יוליך את המשתמש אם הוא ילחץ עליו. כאן למשל יש תגיות של קישורים לאתרי חדשות שונים:

```
<body>
  <h1>אתרי החדשות שלי</h1>
  <ul>
    <li>
      <a href="https://haaretz.co.il">
        הארץ
      </a>
    </li>
    <li>
      <a href="https://calcalist.co.il">
        כלכליסט
```

```
      </a>
    </li>
    <li>
      <a href="https://themarker.co.il">
        דהמרקר
      </a>
    </li>
  </ul>
</body>
```

תכונת href מכילה את כתובת הקישור – כלומר המקום שנגיע אליו אם נלחץ על הקישור.

דוגמה נוספת ומאוד פופולרית היא תכונת src בתגית img. תגית תמונה. התכונה הזו קובעת את מיקום הקובץ. כאשר המיקום יכול להיות יחסי, כלומר יחסי לקובץ ה-HTML או כל כתובת שהיא ברשת. בדוגמה הבאה, התמונה מגיעה משרת מרוחק:

```
<body>
  <h1>תמונה יפה</h1>
  <p>
    <img src="https://picsum.photos/536/354" />
  </p>
</body>
```

יש תגיות שאין להן תכונות טבעיות שמשפיעות על התנהגותן ויש כאלו שכן. אך לכל התגיות, ללא יוצא מהכלל, יכולות להיות שתי תכונות חשובות מאוד. תכונת id ותכונת class. לתכונות אלו אין השפעה על התנהגות התגית, אך הן מסייעות לנו לאתר את התגית ולעצב אותה באמצעות CSS או באמצעות ג'אווהסקריפט ובמקרה שלנו: jQuery.

הערך של תכונת id מתחילה באות אנגלית ויכולה להכיל כל תו שהוא ללא רווחים. למשל:

```
<body>
  <h1 id="MyID-1">תמונה יפה</h1>
  <p>
    <img src="https://picsum.photos/536/354" />
  </p>
</body>
```

התקינה של ה-WC3 קובעת שלכל אלמנט יש id ייחודי לו. ממש כמו תעודת זהות. המספר שלכם הוא המספר שלכם. לא של אחד אחר.

תגית ה-class יכולה להכיל כמה ערכים שמופרדים באמצעות רווח. כמו id, כל class חייב להתחיל באות באנגלית ואחר כך כל תו (חוץ מרווח). כאשר יכולים להיות כמה קלאסים לאלמנט אחד. בנוסף, שם קלאס יכול להיות משותף לכמה תגיות. לשם הקבלה, זה כמו שם משפחה. אם שם המשפחה שלי הוא בר-זיק, זה לא מונע ממישהו אחר גם להיות עם שם משפחה זהה.

בדוגמה הזו למשל, לתגית h1 יש כמה קלאסים ולתגית img יש קלאס אחד.

```
<body>
  <h1 class="MyClass-1 MyClass-2 MyClass-3">תמונה יפה</h1>
  <p>
    <img class="MyClass-1 src="https://picsum.photos/536/354" />
  </p>
</body>
```

אנו משתמשים בקלאס וב-id לצביעת אלמנטים ב-CSS ובחירה שלהם או בחירה שלהם בסקריפט jQuery.

הדרך המרכזית לעיצוב אלמנטים היא באמצעות CSS. ראשי תיבות של cascading style sheets. מדובר בטקסט שהדפדפן מפרש כדי לעצב את אלמנטי ה-HTML השונים.

לכל תגית יש עיצוב ברירת מחדל. למשל h1, כותרת, היא בגודל יותר גדול מהפונט הממוצע. תגית strong היא מודגשת. יש תגיות שאין להן עיצוב ברירת מחדל כלל כמו תגית div או span. עם CSS אנו יכולים לשנות עיצוב של כל תגית שהיא: גודל, צורה, צבע. אפשר לקבוע מימדים של תגית, צבע רקע, צורה, מיקום. כל דבר שהוא. CSS, כמו HTML הוא תחביר, לא שפה.

על מנת שהדפדפן יקרא את ה-CSS, הוא צריך להיות זמין לו. כאשר CSS יכול להיות בתוך דף ה-HTML או מבחוץ לו (כמו קובץ ג'אווהסקריפט). אני אראה כאן רק CSS שנכתב במסגרת ה-HTML. כאשר אנו רוצים CSS כזה, אנו פותחים תגית style באופן הבא:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <style>
    /* CSS is here*/
  </style>
</head>

<body>
  <h1 class="MyClass-1 MyClass-2 MyClass-3">תמונה יפה</h1>
  <p>
    <span>התמונה שלי</span>
    <img class="MyClass-1 src="https://picsum.photos/536/354" />
  </p>
  <script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
  <script>
    (($) => {
      $(document).ready(() => {
        console.log('Ready to write some jQuery code');
      });
    })(jQuery);
  </script>
</body>

</html>
```

התחביר של CSS הוא פשוט מאוד. סלקטור, שמסמן תגית אחת או יותר ותכונות העיצוב שלו.

סלקטור יכול להיות למשל תגית. אחרי כן יש סוגריים מסולסלות ורשימת תכונות העיצוב וערכיהן.

הדוגמה הבאה מראה כמה זה פשוט. הסלקטור הוא תגית h1. הערך העיצובי הוא הצבע ואנו קובעים שהוא יהיה אדום.

```
h1 {
```

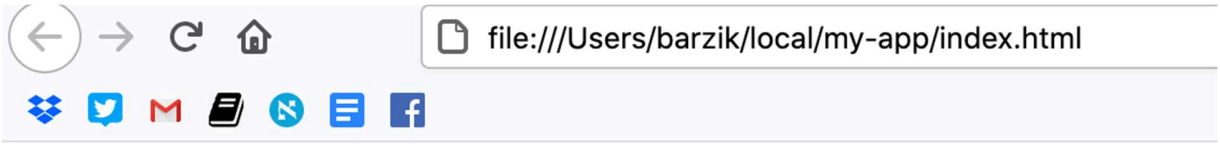
```
  color: red;
}
```

אם נציב את ה-CSS הזה בדף ה-HTML שיש בו תגית h1. למשל כך:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <style>
    h1 {
      color: red;
      text-align: center;
    }
  </style>
</head>

<body>
  <h1 class="MyClass-1 MyClass-2 MyClass-3">תמונה יפה</h1>
  <p>
    <span>התמונה שלי</span>
    <img class="MyClass-1" src="https://picsum.photos/536/354"
/>
  </p>
  <script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
  <script>
    (($) => {
      $(document).ready(() => {
        console.log('Ready to write some jQuery code');
      });
    })(jQuery);
  </script>
</body>
```

```
</html>
```

ונפתח את העמוד באמצעות דפדפן, נוכל לראות שצבע הטקסט של הכותרת אדום.



תמונה יפה

אם אנו רוצים להכניס ערך נוסף, אנו יכולים באמצעות הכנסת ערך, חשוב להקפיד על התו ; בסיום כל תכונה.

הוספת התכונה הבאה תגרום לכותרת להיות במרכז עמוד.

```
<style>
  h1 {
    color: red;
    text-align: center;
  }
</style>
```

בהפרדה של פסיק, אנו יכולים לבחור כמה סלקטורים. למשל:

```
<style>
  h1, img {
```

```
    color: red;
    text-align: center;
  }
</style>
```

התכונות בתוך הסוגריים יהיו רלוונטיות גם לתגיות h1 וגם לתגיות img.

הסלקטור יכול להיות יותר משמעותי משם התגית. וכאן הקלאס של תגית ה-HTML או ה-id יכולים להיות שימושיים. במקום להכניס כמה שמות של אלמנטים, אני יכול להשתמש בשם קלאס כסלקטור CSS. סלקטור מבוסס קלאס הוא נקודה ואז שם הקלאס. אם למשל הקלאס שלי הוא My-Class-1, אז הסלקטור יהיה נקודה ושם הקלאס. ממש כך:

```
<style>
  .My-Class-1 {
    text-align: center;
  }
</style>
```

בדרך כלל נרצה ששמות הקלאסים יהיו יותר משמעותיים, כך למשל אני יוצר קלאס בשם center-align. כל תג שיקבל אותו ימורכז.

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
<style>
  .center-aligned {
    text-align: center;
  }
</style>
</head>

<body>
  <h1 class="center-aligned">תמונה יפה</h1>
  <p class="center-aligned">
    <span>התמונה שלי</span>
    <img class="MyClass-1" src="https://picsum.photos/536/354"
/>
  </p>
  <script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
  <script>
    (($) => {
      $(document).ready(() => {
        console.log('Ready to write some jQuery code');
      });
    })(jQuery);
  </script>
</body>

</html>
```

מאוד מקובל להשתמש בסלקטורים מבוססי קלאסים.

סלקטור נוסף הוא מבוסס id, הוא כמובן ייחודי לתגית HTML שמשתמשת בו. הוא כולל את שם ה-id והסימן # לפניו. למשל:

```
<style>
  #bigImage {
    border: 3px solid black;
  }
</style>
```

הסלקטור הוא לכל אלמנט שיש לו את ה-id שנקרא bigImage והוא מקנה לו גבול שחור. לפי התקינה של ה-WC3, לא יכולים להיות שני אלמנטים עם אותו id.

אם אנו נבחר לתת את ה-id של ה-bigImage לתמונה, נוכל לראות שיש גבול שחור ויציב מסביבה, ברוחב של 3 פיקסלים. פיקסל הוא גודל נקודה על המסך. 3 פיקסלים הן 3 נקודות ואנו רואים את יחידת הפיקסל בלא מעט מקומות ב-CSS.

אם יש שני סלקטורים שמכילים תכונות מנוגדות, למשל, ב-HTML הזה:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
<style>
h1 {
  color: red;
}
.header-class {
  color: purple;
}
#header {
  color: green;
}
</style>
</head>

<body>
  <h1 id="header" class="header-class">תמונה יפה</h1>
  <script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
  <script>
    (($) => {
      $(document).ready(() => {
        console.log('Ready to write some jQuery code');
      });
    })(jQuery);
  </script>
</body>

</html>
```

מה יהיה הצבע של תגית ה-h1? יש שלושה סלקטורים המצביעים על h1. אך יש סלקטורים חזקים יותר ופחות.

סלקטור ה-id הוא חזק יותר מסלקטור הקלאס וסלקטור הקלאס חזק יותר מסלקטור התגית. במקרה הזה, צבע הכותרת תהיה ירוקה.

במידה והסלקטורים שקולים. למשל:

```
h1 {
  color: red;
}
h1 {
  color: purple;
}
```

ינצח זה שהכי למטה. כלומר מרונדר אחרון.

חלק חשוב ומשמעותי ב-CSS וגם ב-jQuery הוא הקינון. יש לנו תגיות שנמצאות בתוך תגיות שנמצאות בתוך תגיות. אם תציצו באתרי אינטרנט קיימים, תוכלו לראות שיש באתרים לעתים עשרים רמות קינון ואף יותר. זה חלק מהכוח של HTML. אנו יכולים ליצור היררכית קינון. כלומר לקבוע שתכונות עיצוב מסוימות יינתנו רק לתגית מסוימת שנמצאת בתוך תגית אחרת שנמצאת בתוך קלאס שנמצא בתוך id. ההיררכיה הזו מאפשרת לנו דיוק גם ללא שימוש ב-id. היררכיה היא באמצעות רווח בין הסלקטורים למשל:

p span strong

מכוון לכל תגית strong שנמצאת בתוך תגית span שנמצאת בתוך תגית p. הסלקטור המקונן הזה לא יעבוד על תגיות strong.

כמובן שהסלקטורים יכולים להיות עם קלאס או id, לא רק שמות תגיות. למשל סלקטור מקונן של:

#myId .myClass p

יעבוד רק על תגיות p שנמצאות בקלאס בשם myClass שנמצא ב-id מסוג myId.

הנה דוגמה לסלקטור מורכב שמסתמך על תגיות בלבד.

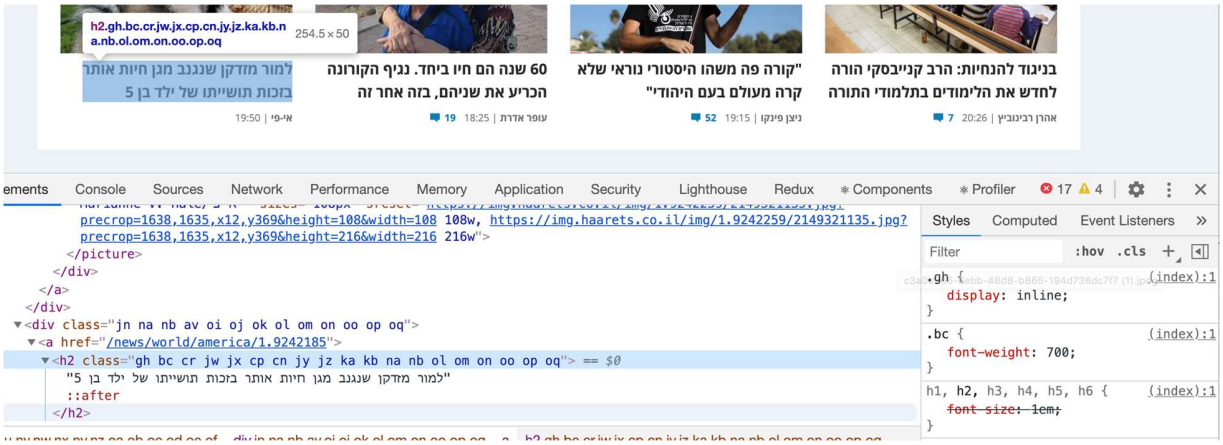
```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <style>
  div p ul li {
    color: black;
  }
  </style>
</head>

<body>
  <h1>אתרי החדשות שלי</h1>
<div>
  <p class="the-list-class">
    <ul>
      <li>
        <a href="https://haaretz.co.il">
          הארץ
        </a>
      </li>
      <li>
        <a href="https://calcalist.co.il">
          כלכליסט
        </a>
      </li>
      <li>
```

```
        <a href="https://themarker.co.il">
          דהמרקר
        </a>
      </li>
    </ul>
  </p>
</div>

  <script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
  <script>
    (($) => {
      $(document).ready(() => {
        console.log('Ready to write some jQuery code');
      });
    })(jQuery);
  </script>
</body>

</html>
```

אם יש כמה סלקטורים שמכוונים לאותה תגית, מי שקובע הוא התגית המקוננת העמוקה ביותר.

בדיקה בכלי המפתחים

כלי המפתחים של הדפדפן יכול לסייע לנו להבין איזה חלק ב-CSS משנה את העיצוב של התגית שאנו רוצים לראות. כלי המפתחים הזה נמצא בכל דפדפן ורק העיצוב שלו שונה מדפדפן לדפדפן. אנו מפעילים אותו באמצעות לחיצה על המקש הימני של העכבר ובבחירה ב-inspect או "בחן". ייפתח לנו חלון מפוצל. מצד אחד אנו יכולים לראות את מבנה התגיות ולבחור תגית ספציפית (שגם תואר על המסך) ומהצד השני את כללי ה- CSS השונים והסלקטורים שרלוונטיים לתגית הזו.



אפשר, באמצעות לחיצה על האלמנטים השונים, לשנות או להוסיף ערכים.

הוספת עיצוב לתגיות באמצעות תכונת style

דרך נוספת להוספה ישירה של תכונות CSS לתגית HTML היא באמצעות תכונת style שמכילה את הטקסט שיש בין הסוגריים המסולסלים. אם למשל אני רוצה שתגית h1 תהיה בצבע אדום, אני יכול להוסיף תכונת style כזו:

```
<h1 style="color:red;">אתרי החדשות שלי</h1>
```

אני יכול להוסיף כמה תכונות לתגית, כל תכונה צריכה להיות נפרדת מהשניה באמצעות התו ;.

```
<h1 style="color:red; text-align: center;">אתרי החדשות שלי</h1>
```

jQuery משתמשת בעיקר בדרך הזו על מנת לשנות או לקבוע עיצובים של תגית HTML. כל מה שיש בתגית style מנצח כל סלקטור שיש בתוך CSS אלא אם כן צמוד לסלקטור הזה המילה !important.

```
<style>
h1 {
  color: black !important;
}
</style>
```

וזה מה שצריך לזכור – כאשר jQuery משנה עיצוב מסוים, היא עושה את זה דרך תכונת ה-style ואנו יכולים לראות את השינוי הזה באמצעות כלי המפתחים. גם כאשר יש אנימציה. נראה אותה בתכונת ה-style.

תרגיל בסיס חשוב

שימו לב: התרגיל הזה חשוב מאוד כיוון שהוא מהווה בסיס לכל התרגולים הבאים.

צרו סביבת עבודה עם jQuery ועם HTML ו-CSS. זו סביבת העבודה שנתרגל עליה. שימו לב למגוון התגיות השונות. שמרו את הקובץ בעורך הקוד שלכם ופיתחו אותו עם הדפדפן.

```
<!doctype html>
<html>

<head>
  <meta charset="utf-8">
  <style>
    body {
      direction: rtl;
      text-align: right;
    }

    form input {
      display: block;
      margin: 10px;
    }

    form textarea {
      display: block;
      width: 400px;
      margin: 10px;
    }

    form button {
      display: block;
      width: 100px;
      margin: 10px;
    }
  </style>
```

```
</head>

<body>

  <div id="container">
    <h1>האתר שלי</h1>
    <div class="paragraph_container">
      <p> חזק חזק ונתחזק<strong>לורם איפסום לורם איפסום לורם איפסום</strong></p>
      <p class="middle"> חזק חזק<strong>לורם איפסום לורם איפסום לורם איפסום
ונתחזק</strong></p>
      <p class="last"> חזק חזק<em>לורם איפסום לורם איפסום לורם איפסום
ונתחזק</em></p>
      <div class="image_container">
        <img src="https://picsum.photos/id/1/200/300"
class="images" />
      </div>
      <p>
        <a href="https://internet-israel.com"
target="_blank">אינטרנט ישראל</a>
      </p>
      <p>
        <a href="https://hebdevbook.com">אתר הספרים</a>
      </p>
    </div>
    <div class="list_container">
      <ul>
        <li>1 פריט</li>
        <li>2 פריט</li>
        <li>3 פריט</li>
        <li>4 פריט</li>
      </ul>
    </div>
    <div>
      <form id="contact_form">
        <label for="contact-form-name">שם</label>
        <input type="text" id="contact-form-name" />
```

```
        <label for="contact-form-messag">הודעה</label>
        <textarea id="contact-form-message"></textarea>
        <button type="button">שגר ושכח</button>
      </form>
    </div>
  </div>

  <script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>

  <script>
(($) => {
  $(document).ready(function () {
    // jQuery script here
  });
})(jQuery);
  </script>
</body>

</html>
```

סלקטורים

תגיות HTML והעיצוב המוקדש לכל תגית נמצאות ב-HTML. ג'אווהסקריפט יודעת לגשת אל התגיות האלו באמצעות DOM, ראשי תיבות של Document Object Model. מודל שמאפשר לשפות תכנות שונות לגשת אל כל תגית HTML. כל תגית היא בעצם אובייקט ג'אווהסקריפט שיש לו תכונות ומתודות. בג'אווהסקריפט רגיל אנו ניגשים אל האובייקט הזה באמצעות מתודות קבועות כמו document.getElementById או document.getElementsByClassName שמחזיר מערך. כל המתודות האלו בעצם ממירות תגיות לאלמנטים של DOM.

עם jQuery אנו ניגשים אל ה-DOM באמצעות פונקצית jQuery, הלא היא ה-$. הפונקציה הזו מקבל כארגומנט סלקטור מבוסס CSS ומחזירה לנו מערך של אובייקטי DOM לפי הסלקטור. תמיד יחזור מערך, גם בסלקטורים שמחזירים אלמנט אחד.

איך סלקטורים עובדים בג'אווהסקריפט

כדי להבין את jQuery, נעשה חזרה קטנה על איך עובדים עם ג'אווהסקריפט בלבד מול ה-DOM.

למשל, אם יש לנו את התגית הזו ב-HTML:

```
<h1 id="header">הכותרת שלי</h1>
```

איך אני ממיר אותה לאלמנט ג'אווהסקריפט ב-DOM? באמצעות הפקודה getElementById :

```
const headerElement = document.getElementById('header');
```

מייד אחרי שיש לי את אלמנט ה-DOM. ועכשיו אני יכול להריץ את מתודות ה-DOM. למשל לשנות את הצבע של האלמנט לאדום:

```
headerElement.style.color = 'red';
```