**מהם השינויים העיקריים ב CSS3**

* CSS3 מציע הרבה יותר סלקטורים ופסאודו-אלמנטים.
* טיפול פשוט יותר בעמודות מרובות, בלי צורך בהצפת אלמנטים.
* גמישות רבה בשינוי גודל תיבה לפי הצורך.
* הוספת אלמנטים ויזואליים מתקדמים לאתר ללא צורך בפוטושופ: בכלל כך, הוספת צל לתיבות ולטקסט, פינות מעוגלות לתיבות, אפקטים של שקיפות ומעברי צבע.
* אפשרות לאנימציה ו-Roll-over ללא צורך ב-JavaScript.
* תכונות רקע חדשות ומתקדמות.
* ואפילו עיצוב אתר רספונסיבי עם תצוגה מותאמת לרוחב המסך.

**CSS3 סלקטורים**

**סלקטורים** מאפשרים לבחור לאיזה אלמנטים להוסיף איזה כללי CSS

ב-CSS3 נוספו מספר רב של סלקטורים, ולכן אנחנו יכולים להיות מאוד מדויקים בבחירת האלמנטים שעליהם נפעיל את כללי העיצוב. באמצעות הסלקטורים החדשים אנחנו יכולים להכיל את כללי העיצוב לפי התאמה למיקום בהיררכיה ולפי יכולות מתקדמות שנכיר בחלק זה.

## סלקטורים של attributes

מחלקות (class) ומזהים (id) שימשו לאבחנה בין אלמנטים שונים בדף אינטרנט כבר בגרסאות קודמות של CCS

אבל CSS3 מאפשר בחירה על סמך כל attribute של HTML לדוגמה, על סמך href

כדי לבחור ב-attribute בתור סלקטור, נוסיף אותו בתוך סוגריים מרובעים:

אם אנחנו רוצים לבחור בכל האלמנטים שה-attribute שלהם הוא href זה הקוד שבו נשתמש:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

a {

display: block;

}

[href] {

color: red;

}

</style>

</head>

<body>

<a href="#">click here</a>

<a href="https://www.facebook.com">facebook</a>

<a>click me</a>

</body>

</html> עכשיו, אם נרצה לבחור על סמך attribute וערך. נוסיף את הערך ל-attribute בתוך הסוגריים המרובעים **["attribute="value"]**

לדוגמה, נבחר את כל הלינקים שה- href שלהם מצביע ל#:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

a {

display: block;

}

[href] {

color: red;

}

[href="#"] {

background-color: yellow;

}

</style>

</head>

<body>

<a href="#">click here</a>

<a href="https://www.facebook.com">facebook</a>

<a>click me</a>

</body>

</html>

כדי להיות יותר ספציפיים בבחירה, אנחנו יכולים להוסיף את סוג הסלקטור לפני הסוגריים המרובעים, ללא רווח בין סוג הסלקטור ובין הסוגריים המרובעים. לדוגמה:

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

a {

display: block;

}

[href] {

color: red;

}

#a1[href="#"] {

background-color: yellow;

}

</style>

</head>

<body>

<a id="a1" href="#">click here 1</a>

<a id="a2" href="#">click here 2</a>

<a id="a3" href="#">click here 23</a>

<a href="https://www.facebook.com">facebook</a>

<a>click me</a>

</body>

</html>

## בחירה באלמנטים על בסיס regular expressions

כפי שראינו, אנחנו יכולים לבחור לפי ערכו של attribute

אבל אנחנו יכולים לבחור גם על בסיס תנאי של תוכן ה attribute.

אם אנחנו רוצים שכל מה שמתחיל בביטוי מסויים יושפע, נשתמש ב- =^.

לדוגמה, אם href מסתיים בביטוי מסוים או מתחיל בביטוי מסוים.

לדוגמה, נבחר את כל הלינקים שה- id שלהם מתחיל באות a:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

a {

display: block;

margin:10px;

}

[href] {

color: red;

}

#a1[href="#"] {

background-color: yellow;

}

a[id^="a"] {

border-style:dashed;

}

</style>

</head>

<body>

<h1 id="a0">css ex</h1>

<a id="a1" href="#">click here 1</a>

<a id="a2" href="#">click here 2</a>

<a id="a3" href="#">click here 23</a>

<a id="linka" href="https://www.facebook.com">facebook</a>

<a>click me</a>

</body>

</html>

אם אנחנו רוצים שכל מה שמופיע בו הביטוי a יושפע, ללא תלות במיקום, נשתמש ב- =\*.

לדוגמה, נבחר את כל הלינקים שה- id שלהם מכיל את האות a:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

a {

display: block;

margin:10px;

}

[href] {

color: red;

}

#a1[href="#"] {

background-color: yellow;

}

a[id\*="a"] {

border-style:dashed;

}

</style>

</head>

<body>

<h1 id="a0">css ex</h1>

<a id="a1" href="#">click here 1</a>

<a id="a2" href="#">click here 2</a>

<a id="a3" href="#">click here 23</a>

<a id="linka" href="https://www.facebook.com">facebook</a>

<a>click me</a>

</body>

</html>

אם אנחנו רוצים שרק מה שמסתיים בביטוי מסויים- יושפע, נשתמש ב- =$.

לדוגמה, נבחר את כל הלינקים שה- id שלהם מסתיים עם האות a:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

a {

display: block;

margin:10px;

}

[href] {

color: red;

}

#a1[href="#"] {

background-color: yellow;

}

a[id$="a"] {

border-style:dashed;

}

</style>

</head>

<body>

<h1 id="a0">css ex</h1>

<a id="a1" href="#">click here 1</a>

<a id="a2" href="#">click here 2</a>

<a id="a3" href="#">click here 23</a>

<a id="linka" href="https://www.facebook.com">facebook</a>

<a>click me</a>

</body>

</html>

## סלקטורים שמבוססים על מיקום בהיררכיה

אם אנחנו רוצים לבחור אלמנטים על בסיס מיקומם בהיררכיה, נוכל להשתמש בסלקטור **()nth-child**:

לדוגמה, אם אנחנו רוצים לבחור רק את השורה השנייה ברשימה, אז נשתמש בסלקטור באופן הבא:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

li:nth-child(2) {

color: red;

}

</style>

</head>

<body>

<ul>

<li id="l1">class 303</li>

<li id="l2">floor 3</li>

<li id="l3">Hamasger</li>

<li id="l4">Tel - Aviv</li>

<li id="l5">Israel</li>

</ul>

</body>

</html>

באופן דומה, אם רוצים לבחור אלמנטים מסוף הרשימה, אז משתמשים ב

-**()nth-last-child:**, לדוגמה האלמנט השני מהסוף ברשימה יכול להראות כך:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

li:nth-child(2) {

color: red;

}

li:nth-last-child(2) {

color: blue;

}

</style>

</head>

<body>

<ul>

<li id="l1">class 303</li>

<li id="l2">floor 3</li>

<li id="l3">Hamasger</li>

<li id="l4">Tel - Aviv</li>

<li id="l5">Israel</li>

</ul>

</body>

</html>

סלקטור מאוד שימושי ברשימות ארוכות או טבלאות, הוא סלקטור שייתן לנו דגם פסים. לדוגמה, רקע כתום לשורות האי-זוגיות (odd), ורקע ירוק לשורות הזוגיות (even).

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

li:nth-child(2) {

color: red;

}

li:nth-last-child(2) {

color: blue;

}

li:nth-child(even) {

background-color: orange;

}

li:nth-child(odd) {

background-color: green;

}

</style>

</head>

<body>

<ul>

<li id="l1">class 303</li>

<li id="l2">floor 3</li>

<li id="l3">Hamasger</li>

<li id="l4">Tel - Aviv</li>

<li id="l5">Israel</li>

</ul>

</body>

</html>

עבור כלל יותר מורכב, לדוגמה שכל אלמנט שלישי יקבל עיצוב שונה. הקוד הבא יכול לעזור לנו בכך:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

li:nth-child(2) {

color: red;

}

li:nth-last-child(2) {

color: blue;

}

li:nth-child(even) {

background-color: orange;

}

li:nth-child(odd) {

background-color: green;

}

li:nth-child(3n) {

border:dashed;

}

</style>

</head>

<body>

<ul>

<li id="l1">class 303</li>

<li id="l2">floor 3</li>

<li id="l3">Hamasger</li>

<li id="l4">Tel - Aviv</li>

<li id="l5">Israel</li>

</ul>

</body>

</html>

ואפשר גם להתחיל באלמנט הראשון במקום בשלישי על ידי שימוש בדפוס הבא:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

li:nth-child(2) {

color: red;

}

li:nth-last-child(2) {

color: blue;

}

li:nth-child(even) {

background-color: orange;

}

li:nth-child(odd) {

background-color: green;

}

li:nth-child(3n) {

border: dashed;

}

li:nth-child(3n+1) {

border: 5px solid yellow;

}

</style>

</head>

<body>

<ul>

<li id="l1">class 303</li>

<li id="l2">floor 3</li>

<li id="l3">Hamasger</li>

<li id="l4">Tel - Aviv</li>

<li id="l5">Israel</li>

</ul>

</body>

</html>

והקוד הבא יתחיל את הספירה מהאלמנט השני:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

li:nth-child(2) {

color: red;

}

li:nth-last-child(2) {

color: blue;

}

li:nth-child(even) {

background-color: orange;

}

li:nth-child(odd) {

background-color: green;

}

li:nth-child(3n) {

border: dashed;

}

li:nth-child(3n+1) {

border: 5px solid yellow;

}

li:nth-child(3n+2) {

border: 5px solid magenta;

}

</style>

</head>

<body>

<ul>

<li id="l1">class 303</li>

<li id="l2">floor 3</li>

<li id="l3">Hamasger</li>

<li id="l4">Tel - Aviv</li>

<li id="l5">Israel</li>

</ul>

</body>

</html>

## סלקטורים חדשים למחלקות (class) ולמזהים (id)

מחלקות ומזהים שימשו רבות לבחירת אלמנטים בגרסאות קודמות של CSS

אולם CSS3 , מציע לנו בחירה הרבה יותר מדויקת של האלמנטים הרצויים לנו.

הסלקטור הכי שימושי הוא סלקטור not שמאפשר לנו להימנע מבחירת מחלקות מסוימות. המבנה הוא  **not(.class):**.

לדוגמה, בקוד ה-html המצורף, אנחנו מעוניינים לצבוע את כל ה-li, מלבד li ששייך ל-class מסוג c1 שאנחנו לא רוצים לצבוע.

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

li:not(.c1) {

color: red;

}

</style>

</head>

<body>

<ul>

<li id="l1">class 303</li>

<li id="l2" class="c1">floor 3</li>

<li id="l3">Hamasger</li>

<li id="l4">Tel - Aviv</li>

<li id="l5">Israel</li>

</ul>

</body>

</html>

אם מעוניינים לבחור יותר מסלקטור אחד, משרשרים את הסלקטורים. לדוגמה:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

li:not(.c1):not(.c2) {

color: red;

}

</style>

</head>

<body>

<ul>

<li id="l1">class 303</li>

<li id="l2" class="c1">floor 3</li>

<li id="l3">Hamasger</li>

<li id="l4">Tel - Aviv</li>

<li id="l5" class="c2">Israel</li>

</ul>

</body>

## </html>

## סלקטורים מועילים לטפסים

במקרה שתרצו לעצב באופן שונה כפתור או שדה במצב enable או disable, תוכלו לעשות זאת באמצעות הסלקטורים הבאים :

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

input:enabled {

border-color: red;

}

input:disabled {

border-color: green;

}

</style>

</head>

<body>

<input type="text" disabled />

<input type="text" />

</body>

</html>

# CSS3מודל התיבה הגמישה - flex box model

**מודל התיבה הגמישה** שמשמש בתקן CSS3 נועד לענות על שני צרכים עיקריים בעיצוב דפי אינטרנט:

• הצורך למקם אלמנטים זה לצד זה בקלות,

• הצורך להתאים את גודלם של אלמנטים על המסך לממשקים שונים (ממסכים בעלי מסך רחב ועד לסמארטפונים)

עד עכשיו ב- CSS היינו צריכים להשתמש בהצפת אלמנטים (float) כדי לסדר אלמנטים ועמודות האחד לצד השני, בתקן CSS3 נוכל להשתמש בתכונה חדישה ומתקדמת

**box-orient** שהיא ידידותית בהרבה למטרה זו. בנוסף, נוכל להשתמש בתכונת

**box-flex** כדי להתאים ביתר קלות את גודלם של אלמנטים בדף לגודלו של ההתקן שבו אנחנו משתמשים.

כדי להתחיל לעבוד עם ערכת התיבה (webkit-box-) שמציע לנו Chrome אנחנו חייבים לעשות את שני הדברים הבאים:

**ברמת ה-HTML**, נגדיר אלמנט אב כלשהו ומספר אלמנטים בנים שישבו בתוכו. בדוגמת הקוד הבאה, אנחנו מייצרים שלוש פסקאות שיושבות בתוך אלמנט אב:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

</style>

</head>

<body>

<div id="wrapper">

<p class="column">עמודה ראשונה</p>

<p class="column">עמודה שניה</p>

<p class="column">עמודה שלישית</p>

</div>

</body>

</html>

**ברמת ה-CSS**, נגדיר בתוך האלמנט האב :

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

#wrapper {

width: 100%;

display: -webkit-box;

}

</style>

</head>

<body>

<div id="wrapper">

<p class="column">עמודה ראשונה</p>

<p class="column">עמודה שניה</p>

<p class="column">עמודה שלישית</p>

</div>

</body>

## </html>

## סידור אלמנטים זה לצד זה

כעת נוסיף מעט הגדרות בחלק העיצוב:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

#wrapper {

border: 1px solid black;

width: 100%;

display: -webkit-box;

-webkit-box-orient: horizontal;

}

.column {

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

}

</style>

</head>

<body>

<div id="wrapper">

<p class="column">עמודה ראשונה</p>

<p class="column">עמודה שניה</p>

<p class="column">עמודה שלישית</p>

</div>

</body>

</html>

* נשים לב, שהוספנו webkit לאלמנט האב wrapper, ולא לילדים. (column) התוצאה שהעמודות מסודרות אחת לצד השנייה.
* הערכים ש **webkit-box-orient-** יכול לקבל הם: **horizontal** לסידור האלמנטים זה לצד זה או **vertical** לסידור האלמנטים האחד מעל לשני.

## מרכוז האלמנטים בדף

אלמנטים של CSS ניתן למרכז לרוחב הדף (באופן הוריזוטלי) ולגובה הדף (באופן ורטיקלי). כדי למרכז צריך להכיר שתי תכונות:

• **webkit-box-pack**שמיישר את האלמנט לפי הציר שקבענו באמצעות webkit-box-orient-.

• **webkit-box-align**שמיישר את האלמנט בציר השני. אם קבענו שה-box-orient יהיה horizontal, אז קובע את היישור בציר הורטיקלי, ולהפך.

**יישור למרכז לרוחב הדף**

את התכונה הזו מוסיפים לאלמנט שעוטף את העמודות, wrapper. הקוד ייראה כך:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

#wrapper {

width: 100%;

border: 1px solid black;

display: -webkit-box;

-webkit-box-orient: horizontal;

-webkit-box-pack: center;

}

.column {

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

}

</style>

</head>

<body>

<div id="wrapper">

<p class="column">עמודה ראשונה</p>

<p class="column">עמודה שניה</p>

<p class="column">עמודה שלישית</p>

</div>

</body>

</html>

**יישור למרכז אורך הדף**

**webkit-box-pack**  יכול למרכז את האלמנטים אם הוא מקבל את הערך **center**

בנוסף, אם יקבל את הערך **end** ידביק את האלמנטים לצד השמאלי של האלמנט האב, ואם יקבל את הערך **start** ידביק את האלמנטים לצד הימני.

כמובן, שאם היינו ממקמים את האלמנטים אחד מעל לשני ולא אחד לצד השני, start היה מדביק את האלמנטים במעלה הדף, ו-end היה מדביק את האלמנטים בתחתית הדף.

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

#wrapper {

border: 1px solid black;

height: 250px;

display: -webkit-box;

-webkit-box-orient: horizontal;

-webkit-box-pack: center;

-webkit-box-align: center;

}

.column {

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

height: 150px;

}

</style>

</head>

<body>

<div id="wrapper">

<p class="column">עמודה ראשונה</p>

<p class="column">עמודה שניה</p>

<p class="column">עמודה שלישית</p>

</div>

</body>

</html>

יכולות-**webkit-box-align** :

* יכול למקם אלמנטים במרכז הדף כשמשתמשים בערך **center**
* הוא גם יכול, למקם אלמנטים במעלה הדף כשמשתמשים בערך **start**
* בתחתית הדף כשמשתמשים בערך **end**
* אפשר גם להשתמש בערך **stretch** שיימתח את האלמנט לכל גובהו של הדף

באותה מידה, אם האלמנטים היו ממוקמים אחד מעל לשני אז center היה ממקם את האלמנטים במרכז לרוחב הדף, start היה ממקם את האלמנטים בימין, ו-end בשמאל.

## מתיחת האלמנטים בלי חישובים באמצעות CSS3

נניח שאנחנו רוצים שהאלמנט המרכזי ייקח את כל השטח הפנוי לכל אורך הדף, וישאיר לעמודה הראשונה והשלישית את הצדדים. לשם כך, לא צריך חישובים מסובכים כי ב-CSS3 עומדת לרשותנו התכונה **webkit-box-flex**.

לדוגמה, אם ברצוננו למתוח רק את העמודה השנייה, נשנה את קוד ה-HTML וניתן לעמודה classייחודי.

בכדי למתוח את העמודה השנייה, נוסיף לקוד ה-CSS את השורות עבור column\_stretch נשים לב במיוחד לשורה

**webkit-box-flex:1**

שמורה לדפדפן למתוח את האלמנט לכל האורך, כך שלא יישאר "שטח לבן" מלבד שוליים.

נסתכל על התוצאה. אפשר לראות שהאלמנט המרכזי עכשיו מתוח וממלא את כל החלל שבין שני האלמנטים שמצדדיו.

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

#wrapper {

border: 1px solid black;

height: 250px;

display: -webkit-box;

-webkit-box-orient: horizontal;

-webkit-box-pack: center;

-webkit-box-align: stretch;

}

.column {

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

}

.column\_stretch {

-webkit-box-flex: 1;

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

}

</style>

</head>

<body>

<div id="wrapper">

<p class="column">עמודה ראשונה</p>

<p class="column\_stretch">עמודה שנייה</p>

<p class="column">עמודה שלישית</p>

</div>

</body>

</html>

עכשיו נאמר, שאנחנו רוצים שהאלמנטים יהיו בעלי רוחב שווה. את המטרה הזו ניתן להשיג ללא קושי באמצעות ערך **box-flex** של 1 לכל האלמנטים.

הדפדפן יידע לתת לכל האלמנטים בדיוק את אותו רוחב, ללא תלות ברוחב המסך. כך ייראה הקוד:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

#wrapper {

border: 1px solid black;

height: 250px;

display: -webkit-box;

-webkit-box-orient: horizontal;

-webkit-box-pack: center;

-webkit-box-align: stretch;

}

.column {

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

-webkit-box-flex: 1;

}

.column\_stretch {

-webkit-box-flex: 1;

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

}

</style>

</head>

<body>

<div id="wrapper">

<p class="column">עמודה ראשונה</p>

<p class="column\_stretch">עמודה שנייה</p>

<p class="column">עמודה שלישית</p>

</div>

</body>

</html>

אם נרצה שהעמודה האמצעית תהיה בדיוק רחבה פי-2.8 מכל עמודה אחרת. גם למקרה זה ניתן להשתמש ב**box-flex**. פשוט נקפיד שהעמודה האמצעית תקבל ערך **box-flex** של 2.8, והעמודות הצדדיות יקבלו ערך **box-flex** של 1.

זה בדיוק מה שעושה הקוד הבא:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

#wrapper {

border: 1px solid black;

height: 250px;

display: -webkit-box;

-webkit-box-orient: horizontal;

-webkit-box-pack: center;

-webkit-box-align: stretch;

}

.column {

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

-webkit-box-flex: 1;

}

.column\_stretch {

-webkit-box-flex: 2.8;

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

}

</style>

</head>

<body>

<div id="wrapper">

<p class="column">עמודה ראשונה</p>

<p class="column\_stretch">עמודה שנייה</p>

<p class="column">עמודה שלישית</p>

</div>

</body>

</html>

כך זה נראה כשנריץ על הדפדפן. אפשר לראות שרוחבו של האלמנט האמצעי תמיד יהיה פי-2.8 מהאלמנטים הצדדיים ללא תלות ברוחב המסך.

נשים לב ש **box-flex** יכול לקבל כל ערך שלם או בשברים עשרוניים. ערך מיוחד ש**box-flex**  יכול לקבל הוא ערך של אפס, ובמקרה כזה אנחנו צריכים לתת לאלמנט רוחב קבוע באמצעות התכונה width. הקוד ייראה כך:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

#wrapper {

border: 1px solid black;

height: 250px;

display: -webkit-box;

-webkit-box-orient: horizontal;

-webkit-box-pack: center;

-webkit-box-align: stretch;

}

.column {

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

-webkit-box-flex: 0;

width: 200px;

}

</style>

</head>

<body>

<div id="wrapper">

<p class="column">עמודה ראשונה</p>

<p class="column">עמודה שנייה</p>

<p class="column">עמודה שלישית</p>

</div>

</body>

</html>

## שינוי סדר האלמנטים בעזרת CSS3

הודות ל -CSS3 נוכל לשנות את סדר העמודות:

• קודם ניתן לכל עמודה שם מחלקה נפרד בקובץ הhtml

• אח"כ נשתמש בתכונה **ordinal-group** שתגדיר את המיקום של כל אלמנט משמאל לימין.

אלמנט שיקבל ערך **ordinal-group** של 1 יהיה הכי שמאלי, זה שיקבל ערך 2 יהיה ימני יותר, וכיו"ב.

הקוד ייראה כך:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

#wrapper {

border: 1px solid black;

height: 250px;

display: -webkit-box;

-webkit-box-orient: horizontal;

-webkit-box-pack: center;

-webkit-box-align: stretch;

}

.column\_right {

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

-webkit-box-flex: 1;

width: 200px;

-webkit-box-ordinal-group: 2;

}

.column\_left {

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

-webkit-box-flex: 1;

width: 200px;

-webkit-box-ordinal-group: 1;

}

.column\_stretch {

-webkit-box-flex: 2.8;

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

-webkit-box-ordinal-group: 3;

}

</style>

</head>

<body>

<div id="wrapper">

<p class="column\_right">עמודה ראשונה</p>

<p class="column\_stretch">עמודה שנייה</p>

<p class="column\_left">עמודה שלישית</p>

</div>

</body>

</html>

תכונה מאוד שימושית לאתרים דו-לשוניים שבהם עוברים בין השפה העברית שכתובה מימין לשמאל לשפה שכתובה משמאל לימין דוגמת אנגלית. במצב כזה, אנחנו יכולים להדרש להפוך את סדר העמודות. לשם כך, נוכל להשתמש בתכונה**webkit-box-direction**  עם ערך של **reverse** או עם הערך ברירת המחדל**normal**

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

#wrapper {

border: 1px solid black;

height: 250px;

display: -webkit-box;

-webkit-box-orient: horizontal;

-webkit-box-pack: center;

-webkit-box-align: stretch;

-webkit-box-direction: reverse;

}

.column\_right {

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

-webkit-box-flex: 1;

width: 200px;

-webkit-box-ordinal-group: 2;

}

.column\_left {

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

-webkit-box-flex: 1;

width: 200px;

-webkit-box-ordinal-group: 1;

}

.column\_stretch {

-webkit-box-flex: 2.8;

border: 1px solid black;

background-color: yellow;

padding: 10px;

margin: 7px;

-webkit-box-ordinal-group: 3;

}

</style>

</head>

<body>

<div id="wrapper">

<p class="column\_right">עמודה ראשונה</p>

<p class="column\_stretch">עמודה שנייה</p>

<p class="column\_left">עמודה שלישית</p>

</div>

</body>

</html>

## סיכום תכונות מודל התיבה הגמישה

|  |  |  |
| --- | --- | --- |
| **משמעות** | **ערכים** | **Chrome** |
| מאתחל | אין | display: -webkit-box |
| מסדר אלמנטים לגובה או לרוחב | horizontal | vertical | webkit-box-orient- |
| 0 רוחב קבוע,  1 רוחב גמיש,  גדול מ-1 - רחב פי כמה מאלמנט שמוגדר כ-1 | 0, 1 או כל ערך גבוה מ-1 כולל עשרוני | webkit-box-flex- |
| ממקם את האלמנט ביחס לציר שנקבע על ידי -box-orient | start | end | center | webkit-box-pack- |
| ממקם את האלמנט בציר ההפוך | start | end | center | stretch | webkit-box-align- |
| סידור האלמנטים הילדים מימין לשמאל או מלמעלה | normal | reverse | webkit-box-direction- |
| מיקום האלמנט | מספר שלם | webkit-box-ordinal-group- |

# פינות מעוגלות בעזרת CSS3

ניתן להוסיף לתיבות באתר האינטרנט פינות מעוגלות ממש בקלות עם CSS3 באמצעות התכונה  .**border-radius** כשהערך בפיקסלים או באחוזים מציין את מידת העיגול,

ערך גבוה יותר פירושו פינות מעוגלות יותר, וערך של 0 פירושו פינות מרובעות.

ניתן להשתמש ביחידות של פיקסלים, אבל גם באחוזים. לדוגמה:

לדוגמה, הקוד הבא שמשתמש בפיקסלים כדי לעצב פינות תיבה ברדיוס 25 פיקסלים:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

width: 300px;

height: 120px;

background: yellow;

border: 1px solid black;

text-align: center;

border-radius: 25px;

}

</style>

</head>

<body>

<div class="rcbox"></div>

</body>

</html>

בגלל שהשתמשנו בערכים מוחלטים בפיקסלים אפשר לראות שהפינות מעוגלות במידה שווה בציר האנכי והאופקי. לעומת זאת, שימוש באחוזים ייתן מידה שונה של עיגול בציר האנכי והאופקי כתלות באורך וברוחב של התיבה. כדי לראות את ההבדל, נחליף את ערך **border-radius** מ-25px ל-25% לדוגמה:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

width: 300px;

height: 120px;

background: yellow;

border: 1px solid black;

text-align: center;

border-radius: 25%;

}

</style>

</head>

<body>

<div class="rcbox"></div>

</body>

</html>

אפשר לראות שהציר האופקי ארוך יותר מהציר האנכי בגלל שאחוזים נותנים תוצאה יחסית.

## ערכים שונים לפינות שונות

בעוד הדוגמה שלמעלה נותנת לנו ארבע פינות מעוגלות באותה מידה, ייתכן שנרצה לגוון כך שרק חלק מהפינות יהיו מעוגלות או שחלק מהפינות יהיו מעוגלות יותר מאחרות. לצורך כך, נוכל לפרק את תכונת **border-radius** ל-4 תכונות נפרדות:

לדוגמה:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

width: 300px;

height: 120px;

background: yellow;

border: 1px solid black;

text-align: center;

border-bottom-left-radius: 35px;

border-bottom-right-radius: 27.5px;

border-top-left-radius: 20px;

border-top-right-radius: 0px;

}

</style>

</head>

<body>

<div class="rcbox"></div>

</body>

</html>

אפשר לראות שפינות שונות קיבלו מידה שונה של עיגול.

עוד אפשרות מעניינת שמציע CSS3 היא לעגל באופן סימטרי את שתי זוגות הפינות. לשם כך משתמשים ב**radius-border-**  ואחריו שני ערכים. הערך הראשון מעגל את הפינה השמאלית העליונה והימנית התחתונה, והערך השני מעגל את שתי הפינות הנותרות. לדוגמה:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

width: 300px;

height: 120px;

background: yellow;

border: 1px solid black;

text-align: center;

border-radius: 50px 10px;

}

</style>

</head>

<body>

<div class="rcbox"></div>

</body>

</html>

## ערכים שונים בציר אופקי ואנכי

אנחנו יכולים לתת לכל תכונה של **border-radius** שני ערכים במקום אחד כדי לשלוט טוב יותר במידת העיגול בציר האופקי (הוריזונטלי) והאנכי (ורטיקלי). הערך הראשון מתייחס לציר האופקי והערך השני מתייחס לציר האנכי. לדוגמה:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

width: 300px;

height: 120px;

background: yellow;

border: 1px solid black;

text-align: center;

border-top-left-radius: 100px 20px;

}

</style>

</head>

<body>

<div class="rcbox"></div>

</body>

</html>

אם נותנים ערך אחד אופקי וערך אחד אנכי, התוצאה תהיה 4 פינות מעוגלות באותו אופן. נסתכל על הדוגמה הבאה:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

width: 300px;

height: 120px;

background: yellow;

border: 1px solid black;

text-align: center;

border-radius: 100px 20px;

}

</style>

</head>

<body>

<div class="rcbox"></div>

</body>

</html>

ניתן לתת ערכים שונים ל-4 פינות התיבה באמצעות **border-radius** ואחריו 4 ערכים. סדר הערכים הוא:

border-radius: top-left top-right bottom-right bottom-left;

הערכים מופרדים ברווח בודד ללא פסיקים. לדוגמה:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

width: 300px;

height: 120px;

background: yellow;

border: 1px solid black;

text-align: center;

border-radius: 80px 30px 10px 0px;

}

</style>

</head>

<body>

<div class="rcbox"></div>

</body>

</html>

## שימוש ב border-radius

שימוש יצירתי ב **border-radius**יכול להניב תוצאות יצירתיות. לדוגמה, אם ניתן לכל הפינות border-radius של מחצית מהרוחב והאורך, ונקפיד על אורך ורוחב שווים, תתקבל צורת עיגול.

קוד ה-CSS הבא יוצר עיגול באמצעות אורך ורוחב של 100 פיקסלים ו- border-radius של 50 פיקסלים:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

width: 100px;

height: 100px;

background: yellow;

border: 1px solid black;

border-radius: 50px;

}

</style>

</head>

<body>

<div class="rcbox"></div>

</body>

</html>

אפשרות אחרת היא ליצור חצי עיגול. הקוד הבא מדגים יכולת זו:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

background: yellow;

border: 1px solid black;

width: 200px;

height: 100px;

border-radius: 0px 0px 200px 200px;

}

</style>

</head>

<body>

<div class="rcbox"></div>

</body>

</html>

בגלל שאנחנו רוצים חצי עיגול לרוחב המסך נקפיד על רוחב כפול מאורך, ואחר כך נמחק את החלקים שהם top באמצעות ערך 0 ולחלקים שמהווים את חלקו התחתון של המסך ניתן ערך של 200 פיקסלים כמו שנתנו לרוחב.

ולסיום, את צורת העלה שפונה ימינה ומסומן ב- (1) מייצר הקוד:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

background: yellow;

border: 1px solid black;

width: 200px;

height: 100px;

border-radius: 200px 0px;

}

</style>

</head>

<body>

<div class="rcbox"></div>

</body>

</html>

ואת צורת העלה שפונה לצד ההפוך (2) מייצר קוד זהה שבו הפכנו את ערכי הradius-border באופן הבא:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

background: yellow;

border: 1px solid black;

width: 200px;

height: 100px;

border-radius: 0px 200px;

}

</style>

</head>

<body>

<div class="rcbox"></div>

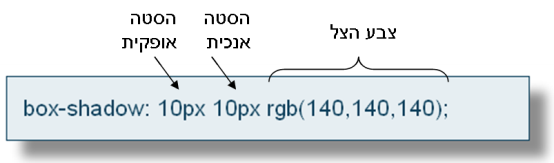
</body>

</html>

**מדריך CSS3 - הוספת צל לתיבות ולטקסט**

תכונת הbox-shadow היא אחת התכונות המעולות שהתווספו ל - CSS3, והיא נתמכת בכל הדפדפנים המובילים.

כדי להוסיף צל לאלמנטים, אנחנו חייבים לציין שלושה ערכים: הסטה בכיוון אופקי, הסטה בכיוון אנכי וצבע הצל. לדוגמה:



הערך הראשון מציין הסטה בכיוון אופקי, ואם יקבל ערך חיובי יסיט את הצל ימינה, ואם יקבל ערך שלילי יסיט את הצל שמאלה.

הערך השני מציין הסטה בכיוון אנכי, ואם יקבל ערך חיובי יסיט את הצל למטה, ואם ערך שלילי יסיט את הצל כלפי מעלה.

שתי הדוגמאות:

בדוגמה, 10px 10px הצל פונה למטה וימינה,

בדוגמה, -10px -10px הצל פונה למעלה ושמאלה

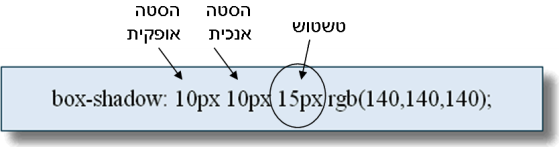
CSS3 כוון לצל התיבה box-shadow

הערך השלישי מציין את צבעו של הצל. כדי ליצור גוון צל אפור, אני חוזר שלוש פעמים על אותו מספר RGB) RGB מערבב בין הצבעים אדום, ירוק וכחול להשגת הגוון המבוקש).

ככל שהמספרים יתקרבו יותר ל-225 (הערך המקסימלי של (RGB כך יהיה הגוון כהה יותר, וכלל שהערכים ייטו יותר ל-0, כך הם יהיו כהים יותר.

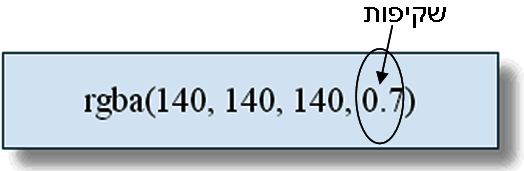
### טשטוש

ערך נוסף שאינו הכרחי, אבל תורם מאוד לתחושה של צל הוא ערך הטשטוש (blur) שמקבל ערכים בפיקסלים. ככל שערך זה גבוה יותר, כך יקבל הצל מידה גדולה יותר של טשטוש. לדוגמה:

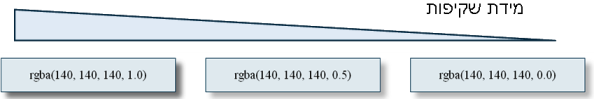


### שקיפות

דרך נוספת שעשויה לגרום לצל להראות טבעי יותר היא באמצעות הוספת שקיפות לצבע הצל. כדי להוסיף שקיפות משתמשים ב-rgba, שדומה ל-rgb ומוסיפה לו ערך רביעי של שקיפות (a משמעותו ערוץ אלפא ששולט בשקיפות). לדוגמה:



ערך השקיפות יכול לקבל ערכים שבין 0 (הכי שקוף) ועד 1 (הכי אטום). בדוגמה הבאה נתתי לצל מידה שונה של שקיפות, ואפשר לראות את ההבדלים בגוון הצל:



### יצירת צל פנימי

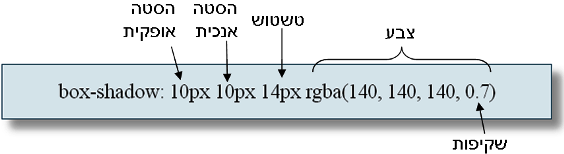
כדי לגרום לצל לפנות פנימה, נשתמש בערך inset. קוד זה ייראה כך:

box-shadow: **inset** 10px 10px 14px rgb(140,140,140);

והוא יגרום לתוצאה הבאה, שמדמה כפתור שנלחץ פנימה:



### נסכם את תכונות הצל של תיבה



כדי לייצר צל שפונה פנימה נוסיף את הערך inset.

### תאימות box-shaddow לדפדפנים

תכונת box-shadow פועלת בכל הגרסאות האחרונות של הדפדפנים, אפילו ב-IE9. אבל כדי למנוע בעיה עם גרסאות מיושנות של Chrome, Safari ו-firefox נוסיף את הקידומות –moz ו-webkit . הקוד ייראה כך:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

background: yellow;

border: 1px solid black;

width: 200px;

height: 100px;

box-shadow: inset 10px 10px 14px rgb(140,140,140);

-moz-box-shadow: inset 10px 10px 14px rgb(140,140,140);

-webkit-box-shadow: inset 10px 10px 14px rgb(140,140,140);

}

</style>

</head>

<body>

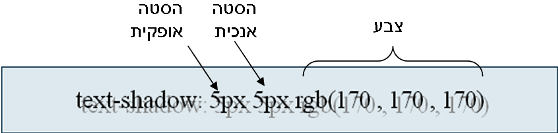
<div class="rcbox"></div>

</body>

</html>

## הצללה של טקסט

כפי שניתן להוסיף צל לתיבות באמצעות התכונה **box-shadow**, ניתן להוסיף צל לטקסט באמצעות התכונה **text-shadow**.



נראה דוגמה לקוד

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

background: yellow;

border: 1px solid black;

width: 200px;

height: 100px;

text-shadow: 5px 5px 5px rgb(170,170,170);

}

</style>

</head>

<body>

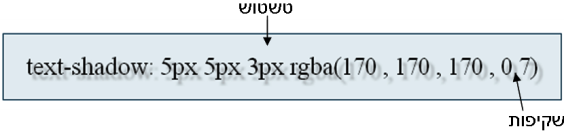
<div class="rcbox">hello</div>

</body>

</html>

:

ואם נרצה להוסיף אפקט של טשטוש ושל שקיפות,



נוכל להוסיף אותם כמו שהוספנו ל box-shadow נראה דוגמה לקוד:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

background: yellow;

border: 1px solid black;

width: 200px;

height: 100px;

text-shadow: 5px 5px 3px rgba(170,170,170,0.7);

}

</style>

</head>

<body>

<div class="rcbox">hello</div>

</body>

</html>

# צבע רקע מדורג באמצעות גרדיאנטים של CSS3

CSS3 מאפשר ליצור מעברים הדרגתיים מצבע לצבע (גרדיאנטים) ביתר קלות באמצעות התכונה  gradient, כאשר הגרדיאנט השימושי ביותר הוא גרדיאנט לינארי.

**גרדיאנט לינארי באמצעות CSS3**

גרדיאנט לינארי אפשר ליצור באמצעות התכונה linear-gradient , ותחביר שנראה כך:



הצבע משמאל יופיע למעלה, והצבע מימין יופיע בהדרגה מתחתיו. לדוגמה, בדפדפנים Firefox ו-Chrome הקוד ייראה כך. שימו לב לקידומת moz ו-webkit, שבגלל נושא התאימות בין דפדפנים שונים אנחנו צריכים להשתמש בקידומות שונות.

וכדי לתמוך בגרסאות שונות, ובדפדפנים מיושנים יותר כדאי להוסיף ברירת מחדל:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

border: 1px solid black;

width: 200px;

height: 100px;

background: green; /\*default\*/

background: -webkit-linear-gradient(top, #00FF00 0%, #FFFF00 100%); /\*Chrome 11\*/

background: -webkit-gradient(linear, left top, left bottom, color-stop(0, #00FF00), color-stop(1, #FFFF00)); /\*Chrome 10, Safari\*/

background: -moz-linear-gradient(top, #00FF00 0%, #FFFF00 100%); /\*Firefox\*/

background: -ms-linear-gradient(top, #00FF00 0%, #FFFF00 100%); /\*IE10\*/

background: linear-gradient(to bottom, #00FF00 0%, #FFFF00 100%); /\*W3C Markup\*/

}

</style>

</head>

<body>

<div class="rcbox">hello</div>

</body>

</html>

## כיוון הגרדיאנט

כדי לשלוט בכיוון הגרדיאנט נוסיף ערך שלישי לפני הצבעים. והוא הערך direction.

direction הוא הכיוון ממנו מתחיל הגרדיאנט. הכיוון יכול להיות במילים- "left", "right" עבור גרדיאנט אופקי, ו-"top", "bottom" עבור גרדיאנט אנכי. נראה את הדוגמאות הבאות:



כדי לראות את האפשרויות השונות נסו להחליף את left עם האפשרויות הבאות:

* **right** כדי לראות גרדיאנט שמתחיל מימין
* **top**  לגרדיאנט שמתחיל מלמעלה
* **bottom**  בשביל גרדיאנט שמתחיל מלמטה
* **top left**  לגרדיאנט שמתחיל בפינה שמאלית עליונה
* **top left** לגרדיאנט שמתחיל בפינה שמאלית עליונה
* **bottom right**  מתחיל בפינה ימנית תחתונה

## כיצד ליצור גרדיאנטים עם יותר משני גוונים?

כדי ליצור גרדיאנטים ובהם יותר משני גוונים, פשוט נוסיף מספר גוונים לקוד. לדוגמה:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

border: 1px solid black;

width: 200px;

height: 100px;

background: -webkit-linear-gradient(white, red, yellow); /\* Webkit (Chrome 11+) \*/

background: -moz-linear-gradient(white, red, yellow); /\* Firefox \*/

}

</style>

</head>

<body>

<div class="rcbox">hello</div>

</body>

</html>

באופן דומה, אפשר להציג את שבעת צבעי הקשת, באמצעות הקוד הפשוט הבא:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

border: 1px solid black;

width: 200px;

height: 100px;

background: -webkit-linear-gradient(left, orange, yellow, green, blue, indigo, violet); /\* Webkit (Chrome 11+) \*/

background: -moz-linear-gradient(left, orange, yellow, green, blue, indigo, violet); /\* Firefox \*/

}

</style>

</head>

<body>

<div class="rcbox">hello</div>

</body>

</html>

## חלוקה בלתי שווה של הגרדיאנט

בדוגמת הקשת, אנחנו יכולים לגרום לכך שכל צבע יתפוס שונה. לשם כך, נוסיף את המקום שעד אליו אנחנו רוצים לפרוס את הגוון באחוזים או בפיקסלים לצד הגוון.

ניישם את התחביר בדוגמה שלהלן, שעובד ב-Chrome וב-Firefox.

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

border: 1px solid black;

width: 200px;

height: 100px;

background: -webkit-linear-gradient(left, orange 50%, yellow 75%, green 87%, blue 94%, indigo 98%, violet 100%); /\* Webkit (Chrome 11+) \*/

background: -moz-linear-gradient(left, orange 50%, yellow 75%, green 87%, blue 94%, indigo 98%, violet 100%); /\* Firefox \*/

}

</style>

</head>

<body>

<div class="rcbox">hello</div>

</body>

</html>

הכתום נפרש עד לאמצע האלמנט, אחריו הירוק נפרש מהאמצע ועד ל-75% מהאלמנט,

## שקיפות

כדי להוסיף שקיפות משתמשים ב-rgba, עם ארבעה ערכים. שלושה ערכים ראשונים מציינים לפי סדר אדום, ירוק וכחול, וערך רביעי שמציין שקיפות. שלושת הערכים הראשונים יכולים לקבל ערכים בין 0 ל-255, והשקיפות יכולה להיות מ-0 ועד 1. 1 מציין אטום, ו-0 שקוף.

rgba(xxx, xxx, xxx, 0.x)

לדוגמה, הגרדיאנט הבא שכולל שני גוונים ושקיפות של 0.6:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.rcbox {

border: 1px solid black;

width: 200px;

height: 100px;

background: linear-gradient(rgba(255, 0, 0, 0.6), rgba(0, 255, 0, 0.6));

}

</style>

</head>

<body>

<div class="rcbox">hello</div>

</body>

</html>

# אנימציות באמצעות CSS3

מה שמייצר אנימציות זו התכונה **transition** שקובעת את משך הזמן שבו מתבצע השינוי. התחביר הוא זה:

transition: <property> <time>;

הערך הראשון הוא התכונה שאותה רוצים לשנות, והערך השני הוא משך הזמן לשינוי התכונה.

במידה ונרצה לשנות יותר מתכונה אחת, נצטרך לציין כל אחת מהתכונות שאנחנו רוצים לשנות ואת משך הזמן לשינוי. לדוגמה, אם נרצה לסובב את האלמנט, לשנות את צבעו של הרקע, ולשנות את תכונות הטקסט, נשתמש בקוד הבא. שימו לב שבין התכונות שרוצים לשנות צריך להוסיף פסיק:

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

.text\_box {

background: rgb(230,238,242);

border: 2px solid blue;

opacity: 1;

transition: transform 2s, background 2s, opacity 2s;

}

.text\_box:hover {

transform: rotate(360deg);

background: violet;

opacity: 0.5;

}

</style>

</head>

<body>

<div class="text\_box">hello</div>

</body>

</html>

**תכונות רקע חדשות שמציע CSS3**

במדריך זה אסביר כיצד ניתן להשתמש בתכונות רקע חדשות שמציע CSS3, וגם כיצד להציג יותר מרקע אחד. במדריך השתמשתי ב-2 תמונות רקע, אבל אפשר בהחלט להשתמש ביותר מ-2 תמונות רקע.

קוד ה-css בשורה הראשונה **background-image** אפשר למצוא את המקור לשתי התמונות, כשביניהן פסיק.

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no">

<meta charset="utf-8">

<style>

body {

background-image: url(http://nebula.wsimg.com/ddaf4cba581f3acb21b541028382651f?AccessKeyId=7AB77D0C3703B086C02A&disposition=0&alloworigin=1), url(http://cdn.newsapi.com.au/image/v1/a0dcdb6534c2153758dd64818d260f6e);

background-position: top left, center;

background-repeat: no-repeat, no-repeat;

background-size: 220px 120px, 1200px 600px;

width: 1200px;

height: 600px;

}

</style>

</head>

<body>

</body>

</html>

אחר-כך, מוסיפים תכונות נוספות, ובכל אחת מהן הערך שמשמאל מתייחס לתמונה הראשונה, ואחריו מופרד בפסיק הערך שמיועד לתמונה הבאה בתור. התכונות שאותן מוסיפים (כולן הכרחיות)

**background-position** מאפשר למקם את התמונות, והוא יכול לקבל 2 ערכים לפי הפירוט הבא:

background-position: x y;

כש-x מזיז את האלמנט לרוחב המסך, ו-y לגובהו של המסך. בהתאם, הערך של x יכול להיות right|center|left, והערך של y יכול להיות top|center|bottom. לדוגמה, כדי למקם אלמנט בפינה הימנית התחתונה, נשתמש בקוד הבא:

background-position: right bottom;

כדי למקם אלמנטים באופן מדויק יותר, ניתן להשתמש באחוזים כאשר 0% 0% ימקם את האלמנט בפינה השמאלית העליונה, 100% 100% בפינה הימנית התחתונה, 50% 50% במרכז שני הצירים, וכן הלאה.

ניתן לקבל שליטה יותר מדויקת כשמשתמשים בארבעה ערכים במקום בשניים.

background-position: right 10px bottom 20px;

קוד זה יגרום למיקום הרקע 10 פיקסלים מימין ו-20 פיקסלים מתחתית האלמנט המכיל.

**background-repeat**  מאפשר לקבוע חזרה על התמונות בציר ה-X או בציר ה-Y, או למנוע חזרה על התמונות. הערכים המשמשים לכך הם: repeat-x, repeat-y ו no-repeat.

**background-size**  קובע את ממדי התמונות. כל תמונה מטופלת בנפרד. תכונה זו יכולה לקבל ערכים בפיקסלים ובאחוזים. הערך הימני מתייחס לציר ה-x והערך השמאלי מתייחס לציר ה-y. בדוגמה זו ראינו שימוש בפיקסלים אבל השימוש באחוזים יכול להתאים יותר להצגה במסכים בגדלים שונים . לדוגמה: אם נקבע את הערך על 100% התמונה תמלא את כל המסך בכל סוגי המסכים, ממכשירים ניידים ועד למסכים גדולים שמיועדים למחשבים שולחניים. הקוד נראה כך:

background-size: 100% 100% ;

**width** ו-**height** קובעים את ממדיה של התמונה הגדולה ביותר.