

גלאיות סגנון – Cascading Style Sheet – Css

Cascading Style Sheets – גלאיות סגנון מדורגים. סגנון הקובעים כיצד ייצגו האלמנטים של HTML בכל המדיה השונות. סגנונות אלו מאפשרים לתוכנת-ה-WEB לעצב דפים בנוחות לרוחב היישום שלו. מעצבים מקצועיים משתמשים בקוו עיצוב מוגדרים מראש אותם הם בוחרים לפROYיקטים שלהם. דפסי עיצוב אלה מלאו את הפרויקט מתחילה ועד סוף ומקנים לו זהות, אופי ווגן. CSS מאפשרים הרבה יותר נוחות, גמישות ויעילות בבניית דפי HTML.

באמצעות אילו כלים כתבים CSS? כל עורך טקסט פשוט, כדוגמת notepad, מספיק כדי למלא את המשימה.

הגדירת סגנון

אופן קביעת סיגנון:
ויתן לקבוע סגנון באמצעות אופנים:

ברירת מחדל-השארת המצב כפי שהדף מסדר אותו.

:*Inline*

אפשר לגשת לכל אלמנט בנפרד ולקבוע לו סגנון מקומי. למשל לתגית מסוימת נקבע שהרקע יהיה צהוב:

```
<b style="background-color: yellow">xxx</b>
```

סגנון זה נקרא *in-line*

פנימי:

קביעת סגנונות כלליים לכל הדף. את זה עושים בדרך כלל בתוך תגית HEAD. מכנים תגית <style>, רושמים סגנונות וסוגרים את התגית - </style>. דוגמא:

```
<head>  
<title>title text</title>  
<style>  
    style text...  
</style>  
</head>
```

סגנון פנימי של מסמך מגדרים סגנונות כלליים לדף HTML הנוכחי וכל האלמנטים שלו שהוגדרו בתוך התגית *style*.

חיצוני:

קביעת סגנונות לאתר שלם. נהוג למצו מסמך CSS חיצוני אשר קובע סגנונות כלליים שיישמשו את כל האתר. כל דף HTML המקשר לדף זה מקבל את הסגנונות הכתובים בו. שימוש בCSS חיצוני מאפשר לתוכנת לשנות את מראה האתר כולו בשינוי של פסקה אחת. האתר יוכל לקבל סגנון אחד ומוחדר.

סדרי עדיפויות

מה קירה כאשר יקבע יותר מסגנון אחד לאווטו פסוק?
הערך יקבע על פ' העדיפות:

1. הגדרת CSS inline.

2. CSS פנימי - קובע רק אם אין סטירה ב-`inline`.

3. CSS חיצוני - קובע רק אם אין סטירה בפנימי או ב-`inline`.

4. ברירת המחדל.

5. בקביעת תכונות במס' שורות תקבע השורה האחרונה.

הגדרת תכונה בשורה עצמה תהיה דומיננטית ותקבע את המאפיין הסופי של הצגת האלמנט גם אם זה מתנגש עם CSS פנימי (הממוקם בתגית ה-HEAD) או חיצוני. CSS פנימי יהיה בעדיפות גבוהה יותר מאשר CSS חיצוני וכמוון שבמקרה של אי התאמה, הפנימי יקבע.

בוא קובץ CSS חיצוני

קובץ CSS כתובים כקובץ טקסט רגיל (כמו דפי HTML) ב-`pad Note`. כל הטקסט בניי במתכונת הבאה:

```
a{text-decoration:none; color:darkgreen;}  
  
a.but{text-decoration:underline; color:navy; font-weight:bolder;}  
  
a.email{color:red; font-size:12px;}  
  
h5 a,h4 a{text-decoration:underline; color:blue; font-weight:bold;}  
  
h5,h4{margin-bottom:0; margin-top:0;}  
  
p{margin-top:0; margin-bottom:0; font-family:verdana,arial,"sans serif";}  
  
a:hover{text-decoration:underline; color:beige;}
```

1. מדירים בקובץ את כללי העיצוב. שומרים את הקובץ בסימת `css`, מדירים את התאג `link` לקובץ זה.

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/css">
```

```
<LINK HREF="styleDetails.css" REL="stylesheet" TYPE="text/css">
```

קישור, מיושם תמיד בתוך מתחם שבין פתיחת תגית `<head>` לסתירתה,

מדריך CSS - מרכיבי השפה

לගילוונות סגנון, בין אם מדובר בחיצוניים או פנימיים, (לא inline) יש תחביר מאוד פשוט:
`SELECTOR{PROPERTY:VALUE}`

`table{background-color:blue}`

הגדרות CSS יהיו בעלי עדיפות גבוהה יותר מפקודות HTML רגילות. כלומר, אם הגדרת בתגית `body` את התכונות

`<body bgcolor="silver" text="navy">`

וכן נכתבת השורות האלה: (לפני או אחרי, אם כי נהוג לכתוב CSS בחלק `HEAD`)

```
<STYLE> body{color:forestgreen; background-color:beige} </STYLE>
```

דף HTML מקבל את התכונות שהוגדרו ב-CSS ומוחזר על תכונות BODY

ניתן לכתוב מספר תכונות שונות לאווטו סלקטור.

(אם התכונה היא מרובת מילים יש לשים סיבוב הערך מרכאות.)

`p{font-family:"sans serif",arial,verdana}`

ניתן גם לכתוב מספר ערכים לאווטו סלקטור. לדוגמה,

ניתן גם לחתת תכונות מסוימות לנמה סלקטורים שונים. בדוגמה הבאה ניתן צבע גוף אדום לכותרות גדולות `<h1>` וכן לינקים `<a>` את הסלקטורים יש להפריד בפסיקים:

1. A, h1 {color:red}

CLASS

התוכנה CLASS מאפשרת לנו ליצור שמות לסוגי סגנון ולהחיל אותם על תגיוטות שונות. אם כתבנו CLASS כזה: (טקסט ממורכב)
מודגש

1. .myText{text-align:center;font-weight:bold}

ניתן ליצור גם CLASS ולהחיל אותו רק על תגיוטה מסווג מסוים, לדוגמה ניכן לנו סגנון זהה.

```
1. <style>
2.   .my1{background-color:beige}
3.   p.my1{color:green;text-indent:25px}
4.   h3.my1{color:yellow;text-align: right}
5.   h3{color:blue}
6.   h3{color:orange}
7. </style>
8. <p>line 1.</p>
9. <p class="my1">line 2</p>
10. <h3>line 3</h3>
11. <h3 class="my1">line 4</h3>
```

- השורה הראשונה לא קיבלה כל סגנון ולכן היא מופיעה בשמאל המסר בשחור על רקע לבן.
 - השורה השנייה קיבלה סגנון 1 עם המשיר לתגיוט P ולכן צבעה יהיה ירוק על רקע בז'(ירושא) והוא תינכש ימינה לעומק 25 פיקסלים.
 - שורה 3 מוקמה על רקע לבן, קיבלה צבע כתום ולא גנעה ממקומה.
 - לעומת זאת, שורה 4 נאלצה לעבור לצד ימין של המסר ולהיצבע במצבו על רקע בז'
 - בסתריה בין פקודות CSS בעלות עדיפות זהה, הפוקודה האחוריונה היא הקובעת.
 - סגנון לפי ID** - ניתן לתת סגנון לפי ID של תגיוטה.
- ID הנזכר יחודי ואפשר להשתמש בו פעמים במס' HTML ולן מתן סגנון לפי ID הוא כמעט כמו לתת סגנון inline אלא שבדרך כלל נהג יותר לעשות את זה בטור תגיוטה STYLE. כדי להגדיר סגנון לפי ID יש לכתוב סולימות ולהציגו אליה את ה-ID של האלמנט:

```
1. <style>
2.   p{text-decoration:"underline overline"}
3.   #firstln{text-align:center;color:green}
4.   .turnitred{color:red}
5. </style>
6.
7. <p id="firstln" class="turnitred">line 1.</p>
8. <p id="secondline" class="turnitred">line 2.</p>
```

שתי השורות בדוגמה זו קיבלנו קו עליון ותחתון כי הן תגיוט P, שתיהן תיצבענה באדום כי הן מוגדרות כ-CLASS turnitred-כ-CLASS turnitred מיקום ממורכב ובלתי שהוגדר צבעה ב-IDI, ההגדירה הזאת היא כמו inline וזה קודם בסדר העדיפויות וכן קיבלה השורה צבע ירוק.

כתבת העורות

CSS ניתן לכתוב העורות כדי לעזרך לקרוא הקוד להבין מידע ואיך נכתבה התוכנית. בכתיבת העורות יש לתהום אותם משני הצדדים ב/* העורת כאן */

מדריך CSS - הגדרות סגנון קונטקטואליות

ניתן להגדיר סגנון לתגיות לפי מיקומן בתוך תגיוט אחרות, למשל. אם לטללה שלנו יש רקע כחול ואנו רוצים שלינקים (<a>) בתוך הטבלה יהיו בעלי צבע שונה, נוכל להגדיר את צבעם כך:

1. Body{background-color:white;}
2. a{ color:blue;}
3. table,h1{background-color:blue;}
4. table a{color:white;}
5. h1 i{text-transform:capitalize;}

בנוסף, בדוגמה זו שמו את גודלה בראש כל המילים הנמצאות בתוך תגייט <i> הנמצאת בתוך תגייט <h1>. צורת כתיבת זאת שימושית במיוחד כאשר כתבים רשימות ורוצים מבנה דומה לתחומי רשימות:

1. ol li{list-style:upper-roman;}
2. ol ol li{list-style:upper-alpha;}
3. ol ol ol li{list-style:decimal;}
4. ol ol ol ol li{list-style:lower-alpha;}

מספרים, 1

אילו בנים באותיות אנגליות קטנות. הדבר שימושי מאד גם כאשר יוצרים טבלאות המופיעות רק כאשר לוחצים על קישור מסוים.

1. A table{display:none;}

ה למיקום

תגייט מיד אחריו אחריה:

1. H2 + table{margin-top:-3px;}

: כותרת 1.

הגדרת סגנון של הסימן * תונייחס לכל האלמנטים בדף HTML.

1. *{font-family:times new roman;}

מדריך CSS - תמי-אלמנטים

קיימים בדף HTML מספר אלמנטים שאפשר להגדיר בתוך תגייט מסוימת, כגון: אות ראשונה בפסקה או שורה ראשונה. השורה הבאה מגדירה לדף להציג את האות הראשונה בפסקה בגודל כפול, ולהתohnה הצדיה:

1. p:first-letter{font-style:italic;font-size:200%;}

ראשונה.

1. p:first-line{font-weight:bold;font-size:120%;}

הונט של

1. A:link{color:gold;}
2. A:hover{color:white;background-color:gold;}
3. A:active{color:orange;}
4. A:visited{color:red;}

הגדרות אלה מורות לדף להציג את תוכן ונתיב (<a>) זהב, לשבות את צבעו לבן עם רקע זהוב בעת מעבר הסמן עליו, לשנות את צבעו לכטום אחרי הלחיצה ואילו לינק שכבר בוקר יצביע באדום.

מדריך CSS - ערכי תכונות הסגנון

- שניהם חמישה סוגים של ערכים שונים שתת למאפייני הסגנון:
- מילוט מפתח (למשל: underline או bold.ណון בהם בהרחבה בהמשך).
 - אורכים.
 - אחוזים.
 - כתובות (URLים).
 - צבעים.
- ערכים מציגים את אורךם של אלמנטים במספרים (חלקים גם בשברים עשרוניים).

לרכים ניתן להוסיף גם את סימן ה + או - כדי לסמן הוספה או הפחלה מהערך המקורי של התכונה, וכן יש לצרף, ללא רווחים, שתי אותיות המסמנות את יחידת המידה או את סימן האחוזים. יחידות מידת יכולות להיות: px, em, mm, pt, pc, ex.

ערכים URL חייבים לרשום בתוך סוגרים רגילים מיד אחרי המילה `url` כך:

1. `url(images/smile.gif)`
2. `url(http://www.w3c.com/images/fits.jpg)`

שים לב שאסור להשאיר רווח בין המילה `url` לבין סוגרים וכן כי הכתובת המצוינת היא יחסית לכתובת של דף ה-CSS שבו כתוב הסגנון.

תכונות של צבעים יכולות להיכתב לפני שם הצבע RED, שלישיית RGB הקסדצימלית: #ff0000, או שלישיית RGB דימילית(עשרונית): (rgb(255,0,0), כולם ערך בין 0 ל-255 לכל אחד מהצבעים. (rgb = red, green, blue) הערך 0,0,0 מייצג את הצבע שחור ואילו 255 הוא לבן. ניתן ליצג זאת גם באחוזים.

1. `Color:green;`
2. `Color:#00ff00;`
3. `Color:rgb(0,255,0);`
4. `Color:rgb(0%,100%,0%);`

אה בדיק:

גם כאן אסור להשאיר רווח בין המילה `RGB` לבין סוגרים

מדריך CSS

CSS - גופנים

תכונות של פונטים (גופנים) נחלקים לשבעה:

Font-family

סוג הגוף: רישימה של סוג גופנים מופרדים בפסיקים. רושמים מספר גופנים מפנוי שלא בכל מחשב מותקנים כל הגוף. הדפסן משתמש בגוףן הראשון מתוך הרשימה שモתקן גם במחשב הלוקוט. אם אף אחד מה גופנים המבוקשים לא קיים, ישמש הדפסן בגוףן ברירת המחדל של הדפסן. מסיבה זו, כדאי לבדוק את תצוגת הדף עם מספר גופנים רבים ככל האפשר ולכתוב לפחות ל-5 סדר את המתאים ביותר. כמו כן שוגנים המכונים בשם בעל יותר ממיליה אחת חייבים להיות מוקפים במרקאות (או גרשים).

1. `<p style="font-family:'times New Roman', Arial, verdana, 'sans serif';">`

font-size

גודל הגוף: את גודל הפונט אפשר להגיד באופן אבסולוטי או יחסי.

1. `Td{font-size:16pt;}`
2. `Td{font-size:150%;}`
3. `Td{font-size:medium;}`
4. `Td{font-size:+2pt;}`
5. `Td{font-size:larger;}`

1.5 מגודל

הגוף שנקבע כבירת מוחלט. הדוגמה השלישית מתייחסת לגודל קבוע שנקבע ע"י הדפסן ונitin להשתמש בו באחד מהערכים הבאים: `small`, `medium`, `large`, `x-large`, `x-small`, `xx-large`, `xx-small`. אלו מותאמים לגודלי הכותרות הנפוצות. `<H1>` הזרה הרביעית היא הגדלה של הפונט העכשווי בשני פיקסלים, ואילו הזרה האחרונה, מוגנת את הגדל הבא אחריו אם הוגדר `small`, יקבל האלמנט גודל של `medium`.

font-style

יכולה לקבל אחד משלושה ערכים: `italic`, `oblique`, `normal`.
 Normal הינו כתב רגיל ואילו בין שני האחרים קשה להבדיל, שניהם נטוויים.

Font-weight

היא תכונה המאפשרת לעבות את האותיות. ערך ברירת המחדל הוא `normal` אבל ניתן לתת את הערך `bold` כדי לקבוע עבותuba או להשתמש בערכים ייחודיים `bolder` או `lighter` באופן ייחודי לאלמנט ההורה. במקרה מיילים, אפשר להשתמש במספרים: 100 הוא העדין ביותר, 900 הוא המודגש ביותר. 400 הוא הערך הרגיל, ו-700 הוא הערך הקבוע `c-bold`.

Line-height

היא גובה שורת האותיות, ומוצגת באמצעות מידת אורך מסוימת לעיל. לדוגמה זו אפשר לתת ערכים מספריים, אחוזים, או יחסיים (+pt2).

font-variant

ונמכת רק על ידי אקספלורר ויכולה לקבל שני ערכים: `normal` ו-`small-caps`. האפשרות الأخيرة הופכת את האותיות הכתובות לאותיות גדולות (מן AA).

font-stretch

תכונה אחרת ופחות מוכרת (משום שרוב הדפסנים אינם תומכים בה עדין) היא `font-stretch`. תכונה זו מאפשרת להציג את האותיות מכובצות או רחבות לפי בחירתך. ניתן לשים את הערכים: `semi-`, `Ultra-condensed`, `extra-condensed`, `condensed`, `Font-size-adjust`, `condensed`, `semi-expanded`, `expanded`, `extra-expanded`, `ultra-expanded`. פירושה של דבר שמדובר בתכונה זו אינה נنمכת על ידי אף אחד מהדפסנים. פעמים רבות אנו נתונים ערכים ליותר מתכונה אחת של גופן.

1. `P{font-size:14pt;font-family:Arial,Verdana,'sans serif';}`
2. `font-weight:bold;line-height:16px;}`

ש בתכונה

1. `P{font: bold 14pt/16pt Arial,verdana,'sans serif'}`

סדר כתיבת החזירים מאד חשוב: ראשית יש לציין את הערכים – `variant`, `style`, `weight` – שנית באים `font-size` ו-`line-height` ולבסוף `font` (font-family). מבחן כל התוכנות גודל הגוף (`size`) וכן משפחות הגוף (`font`) חייבים להיות בהגדלת תכונות `font`.

1. `P{font:12pt 'times new roman',serif, Paris;}`
2. `Em{font:italic bolder +6pt/+10pt Courier,Monospace;}`

ויתבעו.

מדריך CSS - צבעים ורקע

לכל אלמנט בדף HTML יש רקע (background) וקידמה (foreground), ולכל אחד מהם יש צבע. סגנון הצבעים והרקע שלolutים בתכונות אלה ומעצבים אותם לפי רצונך. אלמנטים ביןם יורשים בדרכם את הצבעים של הורייהם. לדוגמה אם הגדרת body

צבע יירוק, יקבלו גם הוכתרות <h1> צבע יירוק. למלניטים ביןם יורשים בדרכם את הצבעים של הורייהם. בריית המחדל כאן היא

צבע ירוק, דהינו שקוף. למרות שהוגה בדרכם לכל קבוע צבע וקע לכל הדף, אפשר גם לקבוע רקע לאלמניטים מסוימים וכך להבליט אותם על רקע הדף. מעניין לראות שcottart <h1> מקבלת צבע רקע נקבעת לרוחב הדף כולו (אלא אם כן היא מוגבלת בטבלה, למשל).

Background-image

מקבלת ערך URL של תמונה רקע כלשהו, או את המילה none. תמונה רקע יכולה לבוא על כל הדף או מאחוריו אלמנט מסוים.

1.

```
<p style="background-image:url(images/myFace.jpg)">
```
2.

```
table table{background-image:url(http://www.bgc.com/bg/crisscross.gif)}
```

התמונה הראשונה נקראת inline מתוך תגי `<p>` והוא תיושם כרקע לתוך התגית. התמונה השנייה תיושם כרקע לטבלה רק אם זו

נמצאת בתוך טבלה אחרת. במקרה, אם התמונה גדולה יותר מהאלמנט, היא תיחסה לגודל האלמנט. אם התמונה קטנה מגודל האלמנט, היא תכפיל את עצמה, כמו אריחי קיר, מספר פעמים עד שתמלא את האלמנט או לפיה מה שהוגדר בתוכנה `background-repeat`.

background-repeat

מגדירה לדפדן האם ואיך להכפיל את תמונה הרקע באלמנט, כאשר זה האחרון גדול מהתמונה. התוכנה מקבלת אחד מה��כים: `repeat`, `repeat-x`, `repeat-y`, או `x-repeat`.

Repeat Repeat את התמונה לאורכו ציר X של האלמנט (לרוחב הדף). השורה הבאה תיצור טור אדום לצד אחד המופיע:

1.

```
Body{background-image: url(images/red.gif);
```
2.

```
background-position:right right top;
```
3.

```
background-repeat:repeat-y;}
```

background-position

אומרת לדפדן איפה למקם את תמונה הרקע. בריית המחדל היא כרגיל, הפינה השמאלית העליונה של האלמנט. תוכנה זו יכולה לקבל ערך אחד או שניים. אם רק ערך אחד ניתן, הערך ישמש את שתי התוכנות. אם ניתנו שני ערכים הראשון מתייחס לאופקי והשני אנכי. לדוגמה:

1.

```
Table{background-position: 20px 50px;}
```

באתודותים:

1.

```
Table{background-position: 33% 20%;}
```

תמשים ב:

1.

```
td{background-position: center middle;}
```

Background-attachment

היא תוכנה הפעלת רק עם אקספלורר ומאפשרת למקם את תמונה הרקע קבועה (fixed) או נעה (scroll) בגליליה. התוכנה `background-attachment` מאגדת את כל תכונות הרקע ומאפשרת לCKER הגדירות. אין חוקים לגבי סדר הערכים ומאפשר לשים אותם בכל סדר שהוא, מופדרים זה מזה ברוחו:

1.

```
table{background: url(lines.gif) fixed middle center darkgreen no-repeat;}
```
- התוכנה `color` צובעת את הקירמה (foreground). את האותיות. כדי שצווין לעיל ניתן להשתמש בהגדירות הצבעים בשיטת המילים (green), ערך הקסדצימלי (#ff0000) או שלשות של ערכי RGB.

מדריך CSS - תכונות של טקסט

תכונות טקסט מגדירות כיצד טקסט ממוקם בדף וכי怎 הוא מוגש למשתמש.

Letter-spacing

מוסיפה רווחים בין האותיות בדף.

1. P{letter-spacing:4px;}

| חשובים.

Text-align

מיישר את הטקסט לימין, שמאל, מרכז או צדדים. center, right, left, justify. מומלץ להשתמש בתכונה זו של CSS ולא בתכונת `text-align` של תגיות וגלויות.

Text-decoration

מקבל את אחת או יותר מהתכונות הבאות: underline, overline, blink, none, line-through. באמצעות תכונה זו ניתן לומר לדפדן להדגש בצורה מסוימת את הטקסט. אפשר למתוות יותר מערך אחד:

1. P{text-decoration:blink line-through;}

במקרה זה יסומן הטקסט בקו אמצעי ובנטסקייפ בלבד הטקסט גם יבהב. (אקספלורר אינו תומך ב-BLINK).

text-indent

משמש לצירפת כניסה של טקסט בתחילת פיסקה. הערך המושם בתכונה יציין עד כמה גדולה תהיה הכניסה. ערכים שליליים יתנו שורה ראשונה בולטת, אולם יש להיזהר עם הבלתי שורה כי הוא יכול לצאת מהמסגרת או אפילו מהדף.

1. P{text-indent:-20px;margin-left:20px;}

פיקסלים.

text-transform

בעזרתו אפשר לשנות את הטקסטים מסוימות קטנות לגודלות(uppercase), ההיינר(lowercase), או להגדיל רק את האות הראשונה בכל מילה (capitalize). ברירת המחדל היא כMOVN שוכן. על העברית אין כל השפעה בסעיף זה אולם למי שכותרים אחרים בליעית תכונת capitalizen. מאוד נוחה, במיוחד בכותרות.

vertical-align

שולטת במיקום האנכי של אובייקטים. הערכים הקבילים לתכונה זו הם: baseline, middle, sub, super, text-top, bottom, top. הדפדים השונים מתייחסים לערכים באופן שונה: נטסק"פ תומך בכל פרט ל: sub, super, baseline, text-top, bottom. אקספלורר תומך רק בsub ו-super בכל המדבר בטקסטים.

word-spacing

בדומה לו `letter-spacing` מוסיפה רווחים בין מילים. ערך ברירת המחדל כאן הוא normal, ואפשר לכתוב ערכי גודל במספרים, אחורדים או ערכים יחסיים. נטסק"פ 6 תומך בתכונה זו אולם אקספלורר עדין לא.

text-shadow

היא תכונה שטרם יושמה בדפדים אולם היא ואפשר לחת תלת מימד לאובייקטים בדפדים. בהמשך-CS שנדבר על פילטרים נספר לכם מעט יותר על יצירה אפקטים תלת ממדים (באקספלורר).

מדריך CSS - תכונות תיבת

האלמנטים של HTML נמצאים כלם בתוך תיבה מרובעת. בעזרתו התכונות המוסברות כאן תוכל לשנות במרקם, במיקום ובגודל של התיבות הללו.

לשם הדוגמה הנה שכל תיבה כזו (האלמנט) מוקפת מרובע(padding) , אשר גם הוא מוקף מרובע(border) , וגם זה ממוקם בתוך מרובע(margin). לכל אחד מהם יכולים padding, border וmargin יש אפשרות להגדיר בנפרד גודל לכל אחת מהפאות: left, right, top, bottom, margin-right, margin-left ועודים בכך אותו אופן:

1. td{padding:5mm;padding-bottom:2mm;}
2. table{ padding:2mm 4mm 1mm 3mm;}
3. body{margin:1cm 1in 0cm;}
4. th{padding:2cm 1in;}

: ה-TD-

יהיה בעל רוח פנימי (במילימטרים) של 2, 4, 1, ו-3, כלפי מעלה, ימינה, מטה ושמאל בהתאם (כאשר ישנו 4 ערכים מתחילה מלמעלה ומתקדים מכיוון השעון). למסגר כלו יש רוח מהדפנות של סנטימטר מלמעלה, איןטש אחד מכל צד ו- 0 מלמטה. (כאשר ישנו שלושה ערכים, הראשון מתיחס מלמעלה, השני לשני הצדדים והשלישי למטה). בHTML יש שני ערכים והם נתונים רוח העליון 2 ס"מ וכל אחד משנה הצדדים איןטש אחד

מדריך CSS - גבולות

לגבולות ישנו מספר תכונות הנינטות לשינוי - צבע, עובי וסוגן. לכל אחת מפאות האלמנט - מלמעלה, מימין ושמאל - ניתן לקבוע גבול בצבע, עובי וסוגן שונים. תכונות של גבול אינן באוות בירושה וצריך לציין אותן אותן לכל אלמנט שורצים עבורו גבול. התוכנה קובעת את צבע הגבול של האלמנט. ניתן לשים מאחד ועד ארבעה ערכי צבע. אם תכתוב צבע אחד יקבעו כל פאות המרובע באותו צבע. שני ערכי צבע יצבעו את הפאה העליונה ואת התחתונה בצבע אחד ואת הפאות הצדדיות בצבע השני. עם שלושה ערכים, הראשון יותאם לפאה העליונה, השני לשני הצדדים והשלישי לגבול התחתון.

1. Table{border-color:green yellow orange purple;}

עלון החל ב

גבול העליון, לפי הסדר: עליון(ירוק), ימני(צהוב), תחתון(כתום), שמאל(סגול)..

עובי הגבול נקבע ע"י התוכונה border-width ובירית המהידל היא medium. ניתן גם לסתן את הערכים thin ו-gem thick, או להשתמש במידות רוחב כדוגמת 2px כדי לקבוע את העובי הרצוי במידהיק. גם כן נשמרת שיטת ההגדרות לפי מספר הערכים המוגדרים:

1. Td{border-width:2px;}
2. Td{border-width:thin medium;}
3. Td{border-width:thick 2mm 3px;}
4. Td{border-width:2px 3px 1px 5px;}

ולישוי, יש

גבול עבה מלמעלה, שני הצדדים 2 מילימטר כ"א ושלושה פיקסלים בתחתון. הדוגמה הרביעית תתחיל בעליון, ימני, התחתון והשמאלי ותתען להם עוביים של 2, 3, 1 ו-5 פיקסלים בהתאם. תכונת border-style מאפשרת לשים סוגים שונים של גבולות. הערכים האפשריים הם: none,dotted,dashed,solid,double,ridge,inset,outset. הערכים solid, double ו-gem dotted, dashed, groove, ridge, inset, outset הם הינם חיתוך פנימה, ridge הוא רכס, groove הוא הכנסה פנימה ו-inset/outset הוא הבלטה של האלמנט.

איך כתבים גבולות בקיצור?

ראשית, ניתן להשתמש בborder כדי להגדיר את כל הפאות יחד.

באחדים.

1. Table{border: green double 4mm;}
2. Border-bottom:thick red dotted;}

מדריך CSS - תכונות עימוד

כאשר בדף HTML יש תמונה או טבלה שיש להם align=left או align=right, וקיים טקסט חופשי, בפסקה למשל, הטקסט זורם מסביב לאלמנט וממשיר. אם מוסיפים לפסקה את התכונה clear, ניתן לשנות את הדרך שבה מציג הדף דפן את תוכן הפסקה. ברירת המחדל היא שוחה. אולם אם כתובות את אחד הערכים: both או left, right תיכתב רק בסיום האלמנט הממוקם מצד ימין, שמאל או שניהם בהתאם.

התכונה float מקבלת אחד מהערכים right, left או none. גם תכונה זו שולטת בעיצובה העמוד. התגית דומה לתכונה align של HTML אולם ניתן לישם אותה על כל אלמנט, כולל טקסט, תמונה ואף טבלה.

```
1. table{float:none;}\n2.\n3. <table>\n4. <tr>\n5.   <td>hello</td>\n6. </tr>\n7. </table>\n8.\n9. <p>world</p>
```

בדוגמה זו, כתוב הטקסט מתחת לטבלה. אם ישנו ערך float:left הפיסקה תזרום מימין לטבלה ואילו אם הערך יהיה right מוקם הטבלה בימין המשך והtekst של הפיסקה יזרום משמאל. התכונה height שולטת על הגובה האלמנט המואץ. השימוש בתכונה נפוץ במיוחד עם טבלאות ותמונות אולם יש גם שימושים אחרים. התכונה אומرتה לדפדן להציג את האלמנט על פניו וחוב מסויים. ערך ברירת המחדל כאן היא auto. כאשר מכנים תמונת לדף ורוצים לשנות בגודל ההצעה שלה (רצוי מאוד) כדי לקבוע לה גובה מסוים ואילו את הרוחב לקבוע לauto, דבר זה שומר על הפרופורציות של התמונה. גם התכונה width פועלת באותה צורה וכן ניתן וכדי להשתמש בשתייה.

מדריך CSS - רשימות

אפשר באמצעות גליונות סגנון לעצב את המראה של פריטים ברשימות. דפנדנים מתיחסים לרשיימות כמו כל בлок אחר של טקסט, פרט לעובדה שלבוק זה יש מקדם מסוים כגון מספור (במקרה של OL) או סימן/תבליט (במקרה של UL). את הczora והמייקם של אלה תוכלו לעצב ב-CSS.

התכונה `list-style-image` מקבלת URL של תמונה כלשהי או את המילה שוחה. אם יש לדפדן גישה אל התמונה הוא יציג אותה, אם לא, יציג הדפדן את מה שהוגדר כ-`list-style-type`. כדי לקבל תוצאות יפות כאן מומלץ להשתמש בתמונות קטנות.

```
1. UI li{list-style-image:url(/images/bcg.gif);list-style-type:square;}
```

במקרה זה, אם לא הצליך הדפדן להוציא את התמונה הנדרשת, יציג את הריבוע הרגיל. התכונה `list-style-position` ממענמת את התמונה או הסימון בתוך (inside) או מחוץ (outside), ואלה מתיחסים לשורה הבאה אחרי התבלייט, כלומר אם היא תחזור لكن הסימון (inside) או, לפ' ברירת המחדל (outside) תישאר מוזחת פנימה. בעברית, אין הבדל בין השניים.

הטקסט `List-style-type` משמשת לשני תפקידים:

□ במסגרת UL היא יכולה לקבל אחד מהערכים הבאים: disc, circle, square או none. ברירת המחדל disc היא עיגול ממולא, circle הוא עיגול שתוכו ריק ואילו square הוא ריבוע.
□ במסגרת OL התכונה יכולה לקבל מספר ערכים: decimal - מספר עשרוני, lower-roman, lower-alpha - ספרות רומיות קטנות, upper-roman - ספרות רומיות גדולות, lower-alpha - אותיות אנגליות קטנות, upper-alpha - אותיות אנגליות גדולות, או שוחה. רוב הדפנדנים מזמנים decimal כברירת מחדל.

ניתן גם לכתוב בקיצור `list-style` ולזרוק לשם ערכים ללא סדר מסוים:

```
1. Li.yyy{list-style: lower-roman inside;}\n2. Li.xyz{list-style: url(images/smiley.gif) circle outside;}
```

מדריך CSS - תכונות CSS אחרות

Visibility

כל אלמנט BODY יכול להיות גלי או מוסתר. התכונה visibility מאפשרת למתכנת להעלים מהען דברים מסוימים שלא רצים שהמשתמש יראה. ערכים אפשריים הם hidden וכן visible. הדף שומר מקום עבורם אבל לא מציג אותם, כך:

```
1. <style>
2.   em{visibility:hidden;}
3. </style>
4. <em class="rst">Error on page - please correct it!</em>
```

התכונה שימושית מאוד כאשר עובדים עם דפי HTML דינמיים.

Display

תמונה דומה אחרת היא display, היכולה לקבל את הערכים inline, block, list-item וכן none. האקספלורר מכיר רק ב block וב none, ובעזרת תכונות אלו (וקצת DHTML) ניתן לחולל ניפלאות. התכונה none מעילימה את האובייקט, ולא שומרת לו מקום (להבדיל от visibility). לדוגמה:

```
1. <style>
2.   #con{display:none;}
3.   #loader{display:block;text-decoration:blink;color:blue;font-size:32pt;}
4. </style>

1. <body onload="loader.style.display='none'; con.style.display='block';>
2.   <p id="loader" dir=rtl lang=he>טוען...</p>
3.   <div id="con">... Rest of the page...</div>
4. </body>
```

דוגמה זו (הכוללת שימוש ב-DHTML) מציגה שימוש בסגנון כדי ליצור מסמר אשר עד שלא נתען לזכרו המחשב יציג את השורה "טוען...". כאשר המסמן יהיה מוכן להציג על המסך, תעלם השורה "טוען..." וויפיע המסמן.

white-space

מורא לדף יכול כיצד עליו להנוגע עם תווים לבנים (רווחים, טאים או ירידות שורה). תמונה זו יכולה לקבל את הערכים הבאים:
□ normal – גם כבירת מחדר (כמובן), מציג כל מספר רווחים כרואה אחד.
□ pre – מציג את הרוחים וירידות השורה כמו בתגית <PRE>.
□ nowrap – אומר לדף לא לזרמת שורה אם לא הגיע לתגית
.

position

כאשר רצים למקם טקסט, משתמשים בתכונה position היכולה לקבל את הערכים absolute או relative. הערך absolute ימקם את הטקסט לפי הפרמטרים המצוורפים, והאלמנטים הבאים אחריו יקם כל לא יתייחסו לאלמנט זה. לעומת זאת relative ימקם את האלמנט בהסתה של top ו-left מהתיקום המתאים של האלמנט.

```
1. H5.tp{position:absolute;top:50;left:10;}
2. H5.ap{position:relative;top:50;left:10;}
```

שוב הי

צריכים להזכיר. במקרה השני ימוקם הטקסט מעט למטה ומשמאלי לנקודה שבה היה צריך להיות ממקומו, ואילו האלמנטים שאחורי ימשכו לזרום מהנקודה שבה האלמנוט הוטטיים. השימוש ב position:absolute בעקבות כתיבה בשכבות. אפשר לכתב משחו על גבי משחו אחר. אם רואיתם גפריט הנפתח כאשר מעבירים את העכבר על אלמנט כלשהו, הר יזה משומש התפריט היה ממוקם באופן אבסולוטי ותכונת display שלו הינה none. הנה דוגמה לטבלת קישורים כזו:

```

1. <html>
2.
3. <head>
4. <meta http-equiv="content-type" content="text/html; charset=windows-1255">
5. <style type="text/css" title="">
6. a{color:white;background-color: green;}
7. a:hover{color:green;background-color: white;}
8. a table{position:absolute;display:none;
9. margin-left:-10px;margin-right: -10px;
10. background-color:green;color:white; }
11. table{background-color:orange;}
12. </style>
13. </head>
14.
15. <body dir=rtl lang=he>
16. <table>
17. <tr>
18. <td>
19. <a href="javascript:void();" id=a1
20. onmouseover="emaillinks.style.display='block';"
21. onmouseout="emaillinks.style.display='none';">
22. הדואר שלי<br>
23. <table id="emaillinks" onmouseover="this.style.display='block';">
24. <tr><td><a href="mailto:me@work.com">בעבודה</a>
25. <tr><td><a href="me@home.com">בבית</a>
26. </table>
27. </a>
28. </td>
29. </tr>
30. </table>
31. </body>
32.
33. </html>

```

שימוש לב לתכנות CSS שגיתן לקישור, מעבר על קישור וטבלת הממוקמת בתוך קישור. טבלה שאינה ממוקמת בתוך תגית התקבל ריקע כתום. כאשר עובדים בשכבות, יש אפשרות לשנות את סדר הופעתן. נסה למקם שתי תמונות באותו מקום:

```

1. img.x{position: absolute; top:20; left:50;z-index:3;}
2. img.y{position: absolute; top:20; left:50;z-index:2;}

```

שימוש לב שהתמונה בא תהייצב לפניה ותשתייר את רעوتה. גובה השכבה נקבע עם z-index ומאפשר למתכנת לקבלilit שכבה או לשמש אותה מאחוריה שכבה אחרת. אם תחליפו את ה z-index של התמונה הראשונה ל-1, יתחלפו התמונות והאחריות תעבור קדימה. כל שהמספר גדול יותר, תהייצב השכבה בקדמה.

Cursor

סמן ההפוך מקבל מראה שונה כאשר מעבירים עליו אלמנטים שונים (נראה כמו יד במעבר על קישורים, כמו שעלה במעבר על שדות להכנסות טקסט, למשל). ניתן לשנות את מראה ההפוך גם בעזרת תכונת cursor, שיכולה לקבל את הערכים הבאים: sw-resize , s-resize , w-resize , , pointer, move, e-resize , ne-resize , nw-resize , n-resize , se-resize , default , auto , text , wait , help , hand crosshair

בכך בעצמכם לשנות את הסמן לראות את כל האופציות השונות. תוכלו לדוגמה לכתוב:

```

1. button{cursor:hand;}
1. <a href="directions.html" style="cursor:help">Need Help?</a>

```

לדוגמא:

Zoom

תכונת `-zoom` יכולה לקבל את הערכים: `normal`, `מספר או אחוזים`. היא פעליה רק באקספלורר החל מגירסה 5.5. שימו לב לתמונה הבהה:

1. ``

ScrollBar

אפשר לשנות את חזות פס הגליליה של מסמכים או אלמנטים באמצעות `scrollbar-base-color:scrollbar-base-color`: ישן מספר פריטים בתוך הסקרולר שאפשר לצבוע:

1. `scrollbar-darkshadow-color:green;`
2. `scrollbar-3dlight-color:darkgreen;`
3. `scrollbar-face-color:yellow;`
4. `scrollbar-shadow-color:darkkhaki;`
5. `scrollbar-highlight-color:lightgreen;`
6. `scrollbar-track-color:orange;`
7. `scrollbar-arrow-color:gold;`

CSS3

CSS3 היא טכנולוגיה חדשה ומשמעותה המאפשרת לנו להגדיר סגנונות גרפיים שונים ומגוונים עד מADIO למסמכים הנקתבים בשפות HTML וXML - מבחינה זו CSS3 היא השפה שמאכזותה מעצבים ומתכננים מכל רחבי העולם עובדים על דרכים לעיצובו כגן HTML .
מה חדש של עולם האינטרנט .
גרסת CSS 2.1 - המאושרת האחרון היא CSS 2.1. העובודה על גרסה 3 החלה כבר ב- 1998 - בשעה שגרסה 2.1 הייתה עדין בשלב הטיטוא שלה (ודאי יפיתיע רבים מכמ' לדעת שלמעשה גרסה 2.1 טרם אושרה סופית על ידי W3C) .
CSS 2.1 היא ספציפיקציה של השפה והוא דורשת מכל דףHTML התומך ב CSS - לתמוך בכל הספציפיקציה ולא רק בחלקים ממנה (לטענת מיקורסוט החול מגרסה 8 אקספלורר הוא הדפדפן היחיד התומך בכל הספציפיקציה של גרסה 2.1) .
המודולריות של CSS3 והשפעת עובדה זו על מפתחי ומעצבי אתרים גרסת CSS3 היא מודולרית. משמעות הדבר היא שאין בה דרישתשמי שיתמוך בה יתמוך בכל השפה אלא בכל מודול בפרט. למשל, יצרני הדפנדנים יכולים לבחור באיזה מודול של השפה לתמוך .
מה ניתן למפתחים וכמתממשים ממשמעות הדבר היא שכל דףHTML יכול לתמוך במודולים שונים, אם נוסיף לכך את העבודה מהעבודה על הגדרת המודולים טרם הסתיימה אזי נקבל רמה גבוהה של חוסר אחידות ברמת ואופן התמיכה של הדפנדנים השונים בשפה .

החדשנות של CSS3

בגל המודולריות של השפה שפת CSS3 מאפשרת למפתחים ומעצבים ליצור עיצובים מורכבים בעזרת שורות קוד מעות יחסית, מה שהופר דפים המעציבים בעזרת שפה זו לא רק לעשירים יותר מבחינה עיצובית אלא גם למהירים וקלים יותר. למעשה, CSS3 מצליח לקיים את ההצעה להביא את העשור העיצובי של אפליקציות שלוחניות אל העולם האינטרנט תוך שהוא תומכת בתכונות כגון צליות, טקסט תלת ממדי, שינוי צורה, גרפיקה עשיריה, הגדרת צבעים חדשנית ומקדמת, תמייה בפונטים חדשים וייחודיים, אנימציה ועוד

התחבר של שפת CSS3

במדריך זה אסקור את כל התחביר הבסיסיים של שפת CSS3. אין מדריך זה מתיימר לכוסות את כל הנושאים הקשורים לת לחבר של שפת CSS3 אלא להציג את אותם נושאים שחוותני שראוי להציג. כמו כן, מדריך זה כתוב מתוך הנחה שהקורא בו מכיר את שפת CSS בגרסאותיה הקדומות

אותיות מותחרות וכליים לכתיבה שמורות ב CSS3

ב CSS3 - מזהים (כולל שמורות אלמנטים ב HTML - מחלקות - Class Names - ומזהים בסלקטורים למיניהם) יכולים להכיל רק את התווים הבאים:

- [A-Za-z0-9]
- תווים ISO 10646 החל מנתן 161 ולהלאה.
- הקן המפריד (-) והקן התחתתי (_)

כמו כן, מזהים ב CSS3 - אינם יכולים להתחיל במספר או בקן מפריד (-) שלאחריו יש מספר.

מבנה משפט ב CSS3

משפט ב CSS - יהיה בדרך כלל סלקטור המציין למה מתייחס המשפט, מתכוונה המתייחס לאלמנט שאליו מתייחס הסלקטור (בדוגמה שבתמונה ההתייחסות היא לאלמנט פסקה) ולאחר מכן במאכזות המשפט, כפי שניתן למדוד מן התמונה שלהלן:



כפי שניתן לראות מהתרשים לעיל כל משפט בשפת CSS יתחיל בסלקטור המציין לאיזה אלמנט בדף HTML - שלנו מתייחס משפט ה CSS .

בכל משפט CSS ניתן לשדרר מספר תכונות וערכיהם המתוארים לאוטו סלקטור. על מנת לעשות זאת נפריד באמצעות הסימן נקודה ופסיק בין כל צמד וצמד של תכונה + ערך. חובה לשים את סימן הנקודה פסיק גם אחרי התכונה الأخيرة. נוכל לראות זאת בדוגמה הבאה:



את כל התכונות והערכים יש לשים בתוך סוגרים מסוימים, שאוטם נמקם מיד לאחר הסלקטור. בצורה זו הדף דין יודע שכל התכונות שבתוך סוגרים מסוימים מתייחסות לסלקטור שלפניו.

קיוב משפט CSS בעלי סלקטורים שונים
שפת CSS מאפשרת לקיבץ יחדיו מספר סלקטורים שברצוננו להגדיר עבורם תכונות זהות. בדרך זו אנו יכולים לחסוך בكمות שורות הקוד ולהימנע מחזרות מיותרות על הגדרות זהות. להלן דוגמה לקוד CSS המצביע יחדיו את ההגדרות המשותפות לאלמנטים מסווגים - <h1> - <h6>:

```
h1{ font-family: sans-serif; }
h2{ font-family: sans-serif; }
h3{ font-family: sans-serif; }
h4{ font-family: sans-serif; }
h5{ font-family: sans-serif; }
h6{ font-family: sans-serif; }

הקטנת כמות הקוד באמצעות הקבצת האלמנטים
h1,h2,h3,h4,h5,h6{ font-family: sans-serif; }
```

הערה: יש לשם לב כי בדוגמה שלמעלה הסלקטורים שאוטם קיבצנו יחדיו היו שונים זה מזה.

שילוב הערות בתוך קוד CSS
בכל שפת פיתוח מודרנית גם בתוך קוד CSS ניתן לשלב הערות. שימוש שבעזרת הערות ניתן לכתוב הסברים על הקוד שיקלו עליים בעtid להמשיך ולהזדקן את הקוד CSS - שלמו. את הערה יש לרשום בין לוכסן לכוכבית לבין לוכסן כדי שאפשר ללמוד מן הדוגמה הבאה:

```
/*זהי הערה*/
h1,h2,h3,h4,h5,h6{ font-family: sans-serif; }
```

הורשת תכונות
הריבית הסלקטורים בשפת CSS תומכים בהורשה תכונות לסלקטורים אחרים הננקנים בתוכם. כך למשל אם נגדיר שצבע הטקסט בסלקטור `body` יהיה כחול אז צבע הטקסט בכל הסלקטורים שיונכו בתוכו כגון סלקטור - `p` פסקה - יהיה כחול, אלא אם כן נגדיר אחרת.

מחרוזות
מחרוזות (Strings) ב-CSS - ניתן לכתוב בין מראכות כפולות: "זו מחרוזת" או בין גרשאים: 'גם זו מחרוזת'. מראכות כפולות לא יכולות لكن בתוך מראכות כפולות אחרות. הדרך לכתוב מחרוזות המקבינות זו בתוך זו היא היא באחד מן האופנים הבאים :

```
"this is a 'string'"
"this is a \"string\""
'this is a "string"'
'this is a \'string\'"
```

הגדרת צלויות לטקסט בCSS3

Text-Shadow היא أول הוכנה שמעצבים האתרים חיכו לה יותר מכל. זה לא שאתם יכולים לזרוק את הפוטושופ שלכם – בעיקר לא בגלל שנכון לכתיבת שורות אלה אינטראקט אקספלורר עדין לא תומך בתוכנה זו – אבל בהחלט אפשר לראות את האור בקצת המהירה להיות שמרבת הדפנינים המובילים האחרים תומכים כבר בתוכנה זו.

דוגמה לשימוש בתוכנה **Text-Shadow**

```
p{text-shadow: 2px 2px 7px #111;}
```

כך יראה הקוד שלמעלה אם יורץ בדפדפן מתאים:

Text Shadow

כפי שניתן למדוד מן הדוגמה לעליה התוכנה מקבלת ארבעה ערכיים:

1. הערך הראשון מייצג את אורך הצל בציר ה-X – כולם בציר האופקי הנע מימין לשמאל.
2. הערך השני מייצג את גובה הצל בציר ה-Y – כולם בציר האנכי – מלמעלת למטה.
3. הערך השלישי מייצג את עומק המריחה (radius blur) או אם תרצו מידת הטשטוש של הצל.
4. הערך הרביעי פשוט יותר והוא מייצג את צבע הצל.

אם נרצה להשתמש בתוכנה זו נרצה להגדיר כموין גם את צבע הטקסט ולא צבע הצל) וגודלו כפי שניתן לראות

בדוגמה הבאה :

```
.text-shadow {  
text-shadow: 2px 2px 7px #111;  
font-size: 3.2em;  
color: #f5f5f5;  
}
```

שימוש בתוכנה Text-Shadow כדי להגדיר יותר מצללית אחת לטקסט ניתן להגדיר באמצעות התוכנה **text-shadow** מספר צלויות:

```
.multiple-shadows {  
text-shadow: 0px -11px 10px #C60, 0px -3px 9px #FF0;  
font-size: 3.2em;  
color: #fff;  
text-align: center;  
padding: 10px 0px 5px 0px;  
background: #151515;  
}
```

כך יראה הקוד שלמעלה אם יורץ בדפדפן מתאים:

Multiple Shadows

כפי שניתן לראות מן הדוגמה לעליה הגדרנו באותו שורה יותר מצל אחד. בין הגדרה השתמשנו בפסיק כמפורט.

- CSS3 (Outline) בCSS3

outline-text-outline היא תכונת CSS3 שנכון לכתיבת מדריך זה אינה **נתמכת באף דפדפן**. כוונת התוכנה היא לאפשר למעצבים להשפיע על קווי המיתאר של אותיות, על צבען והטקסטורה שלהם.

הצורה הכללית של לשימוש בתוכנה **- text-outline**

```
E { text-outline: width blur-radius color; }
```

דוגמה לשימוש בתוכנה **Text-Shadow**

בדוגמה הבאה נגדיר לטקסט בתוך האלמנט <2>קיי מיתאר בעובי 2 פיקסל וברדיוס טשטוש של 4 פיקסלים בצבע אדום .
h2 { text-outline: 2px 4px red; }

צורה על גליישת טקסט(Text Overflow)

ישנם מקרים שבהם לא תרצו שטקסט כלשהו לא יגלוש אל מעבר לשורה שבה הוא נכתב כך שחלקיו יגלוש לשורה הבאה. במקרה זה יכול להיות אם אתם כותבים אחר לסמארטפון. לדוגמה במקרה שבו אתם רוצים להציג מספר קישורים אחרים או לדפים אחרים ואתם רוצים לשמור על רוחב שורה אחד. ממש לשם כך נוצרה התוכנה.

css הוצאה הכללית לשימוש בפואו - text-overflow: clip/ellipsis;

כפי שניתן ללמוד מן הוצאה הכללית לתוכנה יש להוסיף לה את אחת משתי המילים השמרות:
clip בחותור את הטקסט בדיק בנקודה שבה הוא מגע לסוף השורה;
ellipsis נשתמש במילה שומרה זו הtekst "חתך גם הוא לkrat Sof השורה אך יתווסף אליו שלוש נקודות כדי להציב עלייה שלבואה יש לו המשך.

דוגמה לשימוש בתוכנה **text-overflow**
בדוגמה הבאה נגדיר לטקסט בתוך האלמנט <3> כי במקרה שטקסט יגלוש ממנו הוא יוסתר. כמו כן נגדיר nowrap לתוכנה **text-overflow: ellipsis** כדי למנוע מהtekst לגלוש לשורה הבאה והכי חשוב נגדיר לתוכנה **white-space: nowrap;** שבסופה ישן שלוש נקודות לטקסט.

```
h3 {  
    overflow: hidden;  
    text-overflow: ellipsis;  
    white-space: nowrap;  
}
```

התוצאה במקרה של הקוד שלמעלה תיראה למשל כך:

גליישת טקסט (Text Overflow) בשפה CS...

הтекסט המלא שנכתב בתוך התג 3 היה: "גליישת טקסט (Text-Overflow) בשפה CSS3" ואולם טקסט זה אמר היה לגלוש לשורה שנייה וכן הטקסט נקטע לkrat Sof השורה והתווסף לו שלוש נקודות.

- CSS3 ב Word Wrap

הຕוכנה **word-wrap** ב CSS3 - שמשה בתחילת רק באינטרנט אקספלור ובשלב מאוחר יותר הפכה להיות חלק מ CSS3. - תוכנה זו מאפשרת למילים ארוכות "להישבר" בסוף השורה על מנת לשמור על רוחבו של האלמנט שבתוכו מוצג טקסט. החדשנות הטובות הן של הדפדים המובילים תומכים בתוכנה זו וכן אפשר כבר עתה להשתמש ולהנחות ממנה ללא חשש.

css הוצאה הכללית לשימוש בפואו - word-wrap: normal/break-word;

כפי שניתן ללמוד מן הוצאה הכללית לתוכנה יש להוסיף לה את אחת משתי המילים השמרות:
normal - משמש במילת המפתח normal כמוון שלא נעשה שימוש בתוכנה. break-word - משמש בסוף השורה מילה ארוכה המילה לא תישבר ותיכריך לשורה הבאה והוא עלולה לגרום ליציאת הטקסט מן התוכום שהוקצתה לו בתוך האלמנט שבו הוא מוצג.
break-word - משמש במילת מפתח זו תגרום לשבירת מילה ארוכה בסוף מילה והעברת חלק ממנה לשורה הבאה כך שלא תהיה גליישה של טקסט מהרווח שנקבע לאלמנט שבתוכו מוצג הטקסט.

דוגמה לשימוש בתוכנה **word-wrap**
בדוגמה הבאה נגדיר לטקסט בתוך האלמנט <k> כי במקרה שטקסט מילה תהיה מילה שאורכה יגרום לטקסט לגלוש אל מעבר לרוחב האלמנט שבתוכו היא מוצגת המילה הארוכה תישבר וחלקה יוצג בשורה הבאה:

p { word-wrap: break-word; }

את שני קטעי הטקסט הבאים הנמצאים בתוך אלמנט מסווג <ק>הגדנו לרוחב 200 פיקסלים. עboro הפסקה ימנית הגדרנו בעבר break-word השמאלית לא הגדרנו דבר. כפי שאתם יכולים לראות הפסקה ימנית שمرة על רוחבה המקורי ושבירה את המילה הארכיה שבתוכה ואילו המילה הארכיה שבפסקה השמאלית פשוט חרגה מગודל הפסקה שבתוכה היא נמצאת.

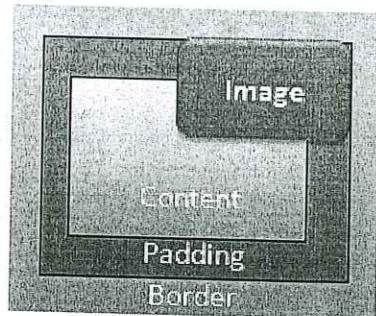
עboro הפסקה זו הגדרנו כי רוחבה לא עלה על מאותים פיקסלים וכי במקרה שבתוך הטקסט שבה תהיה מילה ארוכת מילים אחדת איזה מהרחבת הפסקה((((כאמור 200 פיקסלים

עboro הפסקה זו הגדרנו כי רוחבה לא עלה על מאותים פיקסלים אולם בניגוד לפסקה ימנית לא הגדרנו עborות את התמונה וכך במקרה שבתוך הטקסט שבה תהיה מילה ארוכת מילים אחדת איזה מהרחבת הפסקה(((איך שפה-.wrap מילא מרווח מהרחבת הפסקה

התוכנה **background-origin** בשפת3 CSS

ב CSS2 - תמונה רקע (background image) ממוקמת ביחס לגובה ריפוד (padding) האלמנט שבתוכו נרצה למקם תמונה רקע. כך למשל, אם נרצה למקם תמונה רקע בקצה הימני העליון של אלמנט כלשהו המתמונה לא תමוקם ביחס לגובה (border) האלמנט אלא ביחס ל - padding של האלמנט. דוגמת הקוד הבאה נגיד כי ברצוננו למקם את התמונה בקצה הימני העליון של האלמנט:

```
element {background-image:url('img.gif');  
background-repeat:no-repeat;  
background-position:right top;  
}
```



התמונה הבאה ממחישה את הקוד שלנו באופן הטוב ביותר

באייר המלבן האדום מייצג את תמונה הרקע. האזור הירוק מייצג את הריפוד (padding) של האלמנט. ביכולתכם לראות כי המלבן האדום ממוקם בקצה הימני העליון של התchrom הירוק. לעומת התמונה המתמקמת בקצת הימני העליון של האלמנט בהתאם ל - content. או ה - border הריפוד שלו ולא לאזורי האחרים כגון ה padding . לעומת זאת ב

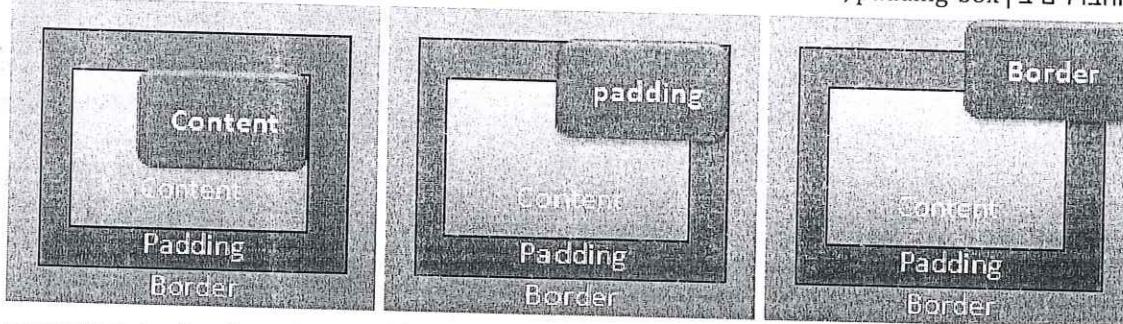
שלייטה על מיקום תמונה רקע בעזרת התוכנה **background-origin** -CSS2 - התוכנה **background-position** מתייחסת באזורי padding - לצורך חישוב מיקום תמונה הרקע. לעומת זאת ב CSS3 - התוכנה **background-origin** מאפשרת לשנות באזורי התיכון את מיקומם. הצורה הכללית של התוכנה היא

כלקמן:
element { background-origin: ;ミلت מפתח }

- מילת המפתח יכולה להיות אחת משלו וקווד המלא יראה כך:
1. border-box (border) כאשר נרצה שתמונה הרקע שלנו תמוקם בהתאם לאזורי הגבול
-moz-background-origin: border;
-webkit-background-origin: border;
-webkit-background-origin: border-box;
background-origin: border-box;
background-position:right top;

-
2. padding; של האלמנט (padding) כאשר נרצה שתמונה הרקע שלנו תמוקם בהתייחס לאזורי הריפוד box-sizing:
-moz-background-origin: padding; -moz-background-origin: padding;
-webkit-background-origin: padding;
-webkit-background-origin: padding-box;
background-origin: padding-box;
background-position: right top;
3. content-box; של האלמנט (content) כאשר נרצה שתמונה הרקע שלנו תמוקם בהתייחס לאזורי התוכן:
-moz-background-origin: content; -webkit-background-origin: content;
-webkit-background-origin: content-box;
background-origin: content-box;
background-position: right top;

המחשת הבדלים בין content-box ו border-box, padding-box



www.devschool.co.il

כפי שהייר ממחיש תמונה הרקע (המלבן האדום) משנה את מיקומה ואת אזור הייחוס שלו על פי מילת המפתח שהוגדרה בתוכנה. background-origin: right top ממקם את תמונה הרקע, מיקומה של התמונה השתנה.

- CSS3 background-clip

התcona **background-clip** CSS3 - מאפשרת לקבוע אם תמונה הרקע תגלוש לגבול (border) אלמנט או שהיא תוגבל

ל裏וף (padding) האלמנט או אף לתוכנו (content).
הצורה הכלילית לשימוש בתcona **background-clip: content** {
background-clip: padding-box;}

מילות המפתח האפשרות הן border-box, padding-box, content-box: .

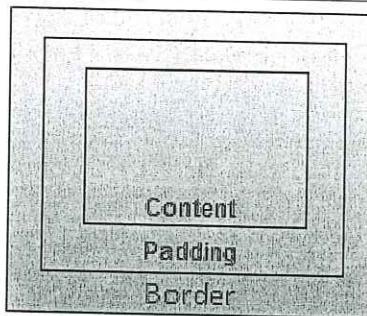
תמונה זו אינה נתמכת בקורסיה זו על ידי רוב הדפדפים. כדי להשתמש בה בכל זאת כדאי להשתמש בתחילת המוחדרת של סוג הדפדפניים השונים. להלן דוגמת קוד צו:

```
-moz-background-clip: padding;  
-webkit-background-clip: padding;  
-webkit-background-clip: padding-box;  
background-clip: padding-box;  
דוגמאות קוד לשימוש בתcona
```

בדוגמה הבאה נגדיר עבור האלמנט <body> תמונה הרקע שלו תמלא את כלו ותגלוש לתוך ה border - שלו:

```
body {background-image:url('gradient.jpg');  
background-repeat:repeat-x;  
background-clip: border-box;
```

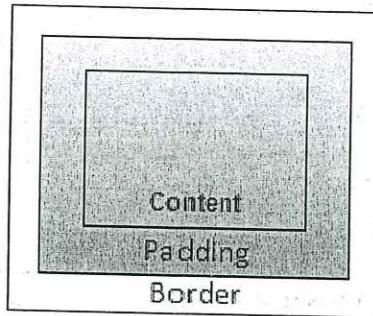
נמחיש את משמעות הקוד שלמעלה בעזרת עזרת האירור הבא:



כפי שביכתכם לראות תמונה הרקע שלו (gradient.jpg) ממלאת את כל שטח האלמנט (body) כולל אזור הגבול, הריפור והתוכן.

שלו padding - כי תמונה הרקע שלו תמלא את כלו ותגלוש לתוך ה <body> בדוגמה הבאה נגדיר עבור האלמנט body {background-image:url('gradient.jpg'); background-repeat:repeat-x; background-clip: padding-box;}

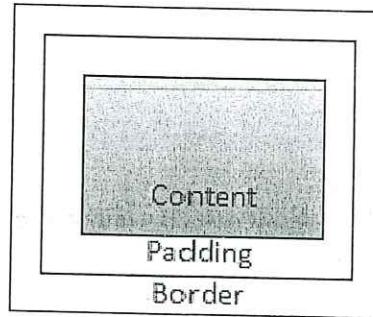
נמיצ'ש את משמעות הקוד שלמטהו בעזרת האיור הבא:



כפי שביכתכם לראות תמונה הרקע שלו (gradient.jpg) ממלאת את שטח ה content - וזה padding - של האלמנט אך לא את אזור ה border - שלו.

בדוגמה הבאה נגדיר עבור האלמנט <body> כי תמונה הרקע שלו תמלא רק את ה content - שלו: body {background-image:url('gradient.jpg'); background-repeat:repeat-x; background-clip: content-box;}

נמיצ'ש את משמעות הקוד שלמטהו בעזרת האיור הבא:



- border -飙你们看得到的背景图全部都是content - 只有padding和border区域没有背景图。也就是说border区域没有背景图。

קביעת ממדים לתמונה רקע CSS3

אחת התוכנות היוצרת חדשות ב CSS3 - היא הינה **background-size**. למורמות היותה תוכנה יחסית חדשה, נכון בזמן כתיבת מדריך זה, היא כבר הושמעה על ידי Firefox 4.0, Explorer 9, Opera 10, Safari 4.1, Chrom.

כפי שניתן להבין משמה תכונה זו מאפשרת לתוכנת לשנות במידה של תמונה רקע לאלמנטים ב HTML - שיכולה להיות להם תמונה רקע.

הצורה הכללית לשימוש ב background-size - היא:

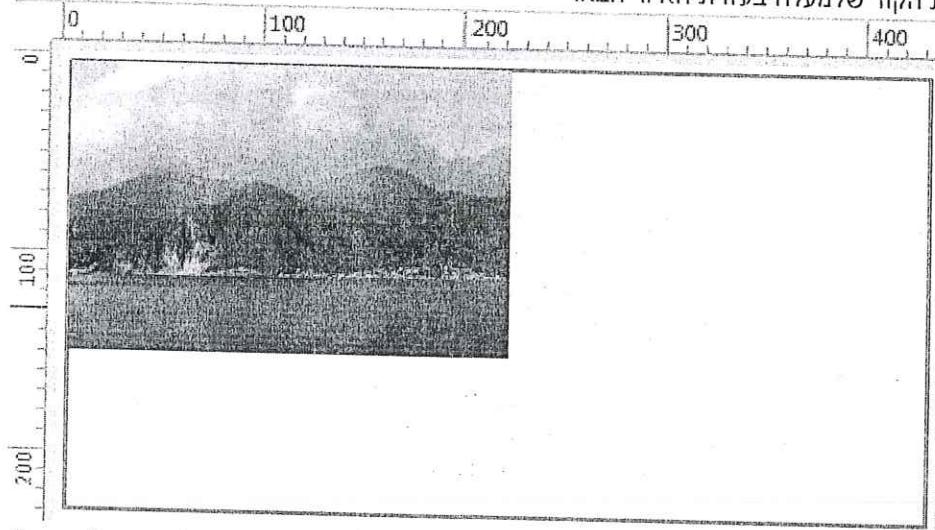
Element { background-size: **ערך**; }

התכונה יכולה לקבל צמד ערכים מספריים או אחוזים המייצגים בהתאם את רוחבה של התמונה ואת גובהה. כמו כן, ניתן לציין מקום ערכים מספריים את המילימטרות או contain או cover להציג כי כאשר נعتبر ערכים אחוזים הם יתיחסו לא הרוחב וגובה תמונה הרקע אלא לרוחב וגובה האזור באלמנט שבתוכו יכולה לשוכן תמונה הרקע.

להלן דוגמאות קוד שונות הממחישות את הערכים השונים שהתכונה background-size יכולה לקבל:
background-size: contain;
background-size: cover;
background-size: auto 100px;
background-size: 100px;
background-size: 100px 100px;
background-size: 100px 100px, 200px 200px;
background-size: 55% 35%;

דוגמה לשימוש בתכונת background-size

#element {background-size: 220px 170px; }
נמחיש את שימוש הקוד שמעלה בעזרת האירור הבא:



כפי שבירכלתכם לראות ונמנית הרקע שלנו אינה מלאת את כל שטח האלמנט אלא רק חלק מתוכו עפ"י הגודל שנגדיר עבורה. אולי לא הגדנו לתמונה ערכים באמצעות התכונה size-היא גודלה "האמתית" של התמונה במרקם זה שונה.

שימוש ב "contain" או "cover" ערכיהם לתכונה background-size-agrorom הדבר להציג תמונה הרקע כך שתהייה גדולה ככל שטח הרקע מאפשר תוך שומרת על יחסי בין הרוחב לגובה התמונה. במידה ויחס זה תואם ליחס שטח הרקע שהתמונה נמצאת בו אז התמונה תמלא את כל השטח המוקצה לו רקע, במקרה שלא, התמונה תהיה הגדולה ביותר האפשרית ללא לפגוע ביחס בין רוחב התמונה לגובה התמונה.

שימוש בערך coverrigrom גם הוא להציג תמונה הרקע. גם כאן יעשה מאמץ לשמור על היחס בין רוחב התמונה לגובהה, אולם במידה ולצורך כיסוי כל השטח המוקצה לרקע יש לפגוע ביחס הדף יבצע זאת וישמר על העיקרון של כיסוי כל שטח הרקע.

- Ribbi Temonot Rekuus - Multiple backgrounds

ב CSS3 - אלמנטים יכולים להכיל יותר מתמונה רקע אחד. האפשרות לרבו תונות רקע לאלמנט פותחת פתח לביצוע עיצובים מורכבים בקהלות יחסית. תכונה זו מאפשרת גם ליצור שכבות של ותמונות רקע, כאשר הראשונה תהיה העליונה והשנייה תציג תחתייה וכן הלאה. לאלה מבינים שרגילים לעבוד עם התכונה backgrounds עניין זה בוודאי מוזר, שכן בדרך כלל ב HTML - כאשר אלמנטים שונים הינם בעלי אותו ערך background-index האלמנט האחרון שמתווסף לדף יהיה השכבה העליונה בדף ואילו כאן התמונה הראשונה שנגדיר כרקע תוגדר שכבה העליונה.

חשיבות לזכור: אם נרצה שאחת מתמונות הרקע שלנו תכסה את כל שטח האלמנט אזי יהיה עליו למקם אותה אחרית היא תכסה את כל שאר התמונות ובמקרה שהיא אוטומה היא גם תסתיר אותן לחłówין.

דוגמה להגדרת תמונות רקע מרובות
Element {background-image: url(img1.jpg), url(img2.jpg);}

כפי שניתן לראות בין תמונה יפריד פסיק. ניתן לשרשר בצורה זו מספר תמונות **לא הגבלה**.
ניתן להגדיר עבור ריבוי של תמונות רקע גם תכונות רקע אחרות וגם במקרה זה יפריד פסיק בין התwichות לתמונה אחת לשנייה.

דוגמה להגדרת מיקום תמונות רקע מרובות
Element {background-position: Top bottom, right top;}

דוגמה להגדרת צורת החזרה של תמונות רקע מרובות
Element {background-repeat: no-repeat, repeat-x; }

Element {background-image: url(img1.jpg), url(img2.jpg);
background-position: Top bottom, right top;
background-repeat: no-repeat, repeat-x; }

ובסה"כ יראה הקוד המשותף כר' :

את אותו הקוד בדיק ביכו לנו לכטוב בקוד מקוצר שיראה כר':
Element {background: url(img1.jpg) top bottom no-repeat, url(img2.jpg) right top repeat-x; }
כאמור, בין כל שכבת תמונה יפריד פסיק והדבר נכון גם בכתב ארוך וגם בכתב מקוצר
- CSS3 Background gradient

Background-gradient היא תכונה חדשה, המשמשת מאוד רבים מבין מעצב האתרים, המשבצים לעתים קרובות gradients באמצעות שלהם. הבעיה היא שלא כל הדפדפנים תומכים בתכונה זו ולכן יש להיזהר בה וכן להשתמש בתחילת המיחוזת באתרים השונים. תכון הכרוניה להשתמש בתחילת המיחוזת בתחילת המיחוזת הדפדפניים המוביילים.

תחבר הכתובת **background-gradient** מזילה כפирופוקס ועוד

-moz-linear-gradient([<point> || <angle>,]?, <stop>, <stop> [, <stop>])
כגון כרום וופاري webkit תחבר לדפפני:

-webkit-gradient(<type>, <point> [, <radius>]?, <point> [, <radius>]? [, <stop>]*)
תחבר לדפפני אופרה

background-image: -o-linear-gradient([<point> || <angle>,]?, <stop>, <stop> [, <stop>]);
בgradient - CSS3 בקוד דוגמאות קוד

קוד לאינטראנט אלספלור

/* for IE */

filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#cccccc', endColorstr='#000000');
כגון כרום וופاري webkit קוד לדפפני:

/* for webkit browsers */

background: -webkit-gradient(linear, left top, left bottom, from(#ccc), to(#000));
3.6+ קוד לפירופוקס

/* for firefox 3.6+ */

bbackground: -moz-linear-gradient(top, #ccc, #000);

רכע לאלמנטים gradient דוגמת קוד כוללת יצירת

�� דוגמת הקוד הבאה כוללת קוד למრבית סוג הדפדפניים וכן למצביעים שבהם דפפניים אינם תומכים בתכונה זו כליל .gradient-bg {

/* fallback/image non-cover color */
background-color: #1a82f7;

/* fallback image */
background-image: url(images/fallback-gradient.png);

/* Safari 4+, Chrome 1-9 */
background-image: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#2F2727), to(#1a82f7));

```

/* Safari 5.1+, Mobile Safari, Chrome 10+ */
background-image: -webkit-linear-gradient(top, #2F2727, #1a82f7);

/* Firefox 3.6+ */
background-image: -moz-linear-gradient(top, #2F2727, #1a82f7);

/* IE 10+ */
background-image: -ms-linear-gradient(top, #2F2727, #1a82f7);

/* Opera 11.10+ */
background-image: -o-linear-gradient(top, #2F2727, #1a82f7);
}

```

גבולות וקופסאות פינות מעוגלות והתקנה`border-radius`

בדפים נשים נוהג היה לקבל את אפקט הגבולות המעוגלים באמצעות תמונות שモקמו בכל אחד מקצוותיו של אלמנט. אולם, בשנתיים האחרונות התמיכה המותגברת בתכונות CSS3 השונות והלץ מצד מעצבים אחרים הביאו לכך ש מרבית הדפים החדשניים תומכים בונכונה `border-radius`, אשר בעזרתה ניתן ליצור את האפקט הנדרש על טरרת קוד CSS וללא שימוש בתמונות. תחביר התוכנה`border-radius` הוא:

```
Element { border-radius: x y; }
```

התקינה `border-radius` מאפשרת גמישות מרבית. ביכולתנו להגדיר באמצעותה את רמת עקומות העיגול של כל פינה בנפרד. בתחביר `border-radius: 0 0 20px 20px;` מיצגת את הפינה ואילו בעדרת `x 0 y 0;` נגידר את שיעור עקומות הפינה.

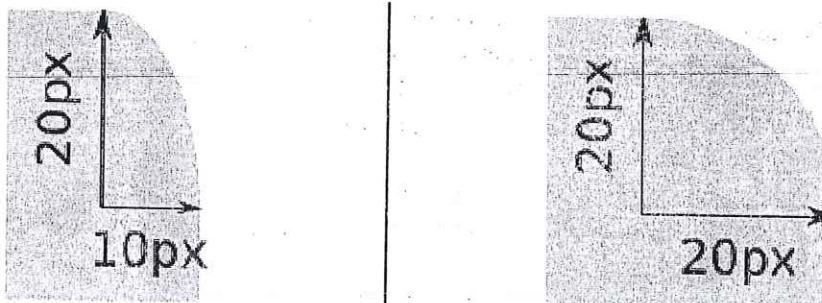
כדי להגדיר את הפינה שבעזרת גדרי עקומות?

ה耿ורה בתוכנה`border-radius` נשתמש לפי העניין באחת ההגדרות הבאות:

הפינה הנבחרת	ה耿ורה בתוכנה <code>border-radius</code>
פינה ימנית עליונה	<code>border-top-right-radius</code>
פינה שמאלית עליונה	<code>border-top-left-radius</code>
פינה ימנית תחתונה	<code>border-bottom-right-radius</code>
פינה שמאלית תחתונה	<code>border-bottom-left-radius</code>

כדי נגידר את רמת העקומות של הפינה?

רמת העקומות של פינה או אם תרצו עד כמה הפינה תהיה "מעוגלת" נקבעת על פי שני פרמטרים שאוטם נמחיש באמצעות האירור הבא:



כפי שvíיכתכם לראות באירור לעלה על ידי שליטה בגובה (y) ורוחב (x) הפינה אנו שולטים ברמת העקומות שלה. נבין זאת יותר טוב אם נב拭 את דוגמת הקוד הבאה שבה נגידר כי הפינה השמאלית העליונה של המרובע שלנו תהיה מעוגלת כך שה x - שלו יהיה שווה 20px ואילו ה y - יהיה שווה אף הוא 20px.

```
element {border-top-left-radius: 20px 20px;}
```

במקרים כמו בדוגמה הקוד האחרון, שביהם אורך ורוחב הפינה זהים (במקרה של מילוי 20 (אך' 3 CSS) מאפשרת לנו לכתוב רק ערך אחד. הדפסן יניח באופן אוטומטי שאנו מתכוונים בכך ש `x` ו- `y` - זהים ויפעל בהתאם. משמעות הדבר היא שתודגמת

הקוד האחרון אנו יכולים לכתוב גם כך:

```
element {border-top-left-radius: 20px;}
```

דוגמאות קוד להגדלת פינות מעוגلات בדוגמה הקוד הראשונה שלנו נגיד רמת עיגול זהה עבור כל אחת מפינות האלמנט שלנו. במקרה זה נגדיר את הקוד עבור כל אלמנט שה `ID` הוא `sample-1`:

```
#sample-1 {
    border-top-left-radius: 20px;
    border-top-right-radius: 20px;
    border-bottom-right-radius: 20px;
    border-bottom-left-radius: 20px;
}
```

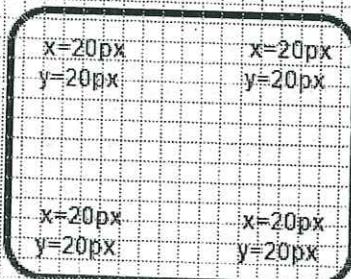
בדוגמה הקוד השנייה שלנו נגיד רמת עיגול זהה עבור כל אחת מפינות האלמנט שלנו. במקרה זה נגדיר את עבור כל אלמנט שה `ID` הוא `sample-2`:

```
#sample-2 {
    border-top-left-radius: 10px 30px;
    border-top-right-radius: 10px 30px;
    border-bottom-right-radius: 10px 30px;
    border-bottom-left-radius: 10px 30px;
}
```

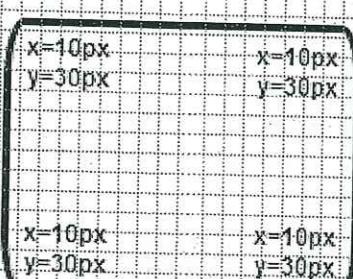
בדוגמה הקוד השלישי שלנו נגיד רמת עיגול **שונות** עבור כל אחת מפינות האלמנט שלנו. הקוד שלנו יראה כך:

```
#sample-3 {
    border-top-left-radius: 40px 30px;
    border-top-right-radius: 30px 30px;
    border-bottom-right-radius: 20px 30px;
    & border-bottom-left-radius: 10px 30px;
}
```

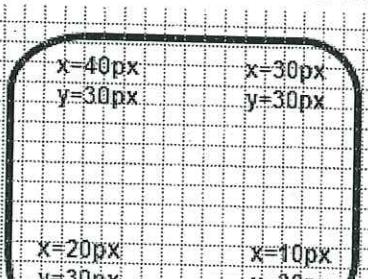
האיור הבא ממחיש את התוצאה עבור כל אחת משלוש דוגמאות הקוד שלמעלה:



דגם 1



דגם 2



דגם 3

הבדלים בין דפסנים שונים ככלל, כל הדפסנים המודרניים (אפיו אינטרנט אקספלורר 9+) תומכים ב `-border-radius`. יחד עם זאת, נכון בזמן כתיבת שורת:

אליה כדי שדפסני מזילה יעבדו עם התוכנה יש להוסיף את התחלתית שלה. תחביר התוכנה במקרה זה יהיה:

```
Element {-moz-border-radius-c: x y;}
```

משמעות הדבר היא שם נרצה שהפינה הימנית העוגלה של אלמנט מסווג `<div>` תהיה בעוגלה בדפסני מזילה נוצרת לכתוב את

הקוד כך:

```
div { -moz-border-radius-top-right: 20px 20px; }
```

כਮון שביבולתנו לשלב בין קוד סטנדרטי לקוד מיוחד לדפסני מזילה כדי שהקוד שלנו "יתפס" בכל הדפסנים המודרניים.

במקרה זה הקוד שלמעלה יראה כך:

```
div { -moz-border-radius-top-right: 20px 20px;  
      border-radius-top-right: 20px 20px;  
}
```

יצירת גבולות (borders) בעלי צבעים רבים

ארגון W3C הציע במסגרת CSS3 שדפננים יתמכו ביצירת גבולות אלמנטים צבעוניים, כך שניתן יהיה ליצור כל גבול ממספר צבעים בעלי רוחב של פיקסל אחד לפחות. בעוד זו ניתן ליצור אפקטים שונים ומגוונים באמצעות CSS בלבד ולא עזרת תМОנות. בשעת כתיבת שורות אלה הדפן היחיד שתומך בגבולות בעלי מספר צבעים הוא Firefox. לכן, מדריך זה יעסוק בעצם בדף Firefox ליצירת גבולות המורכבים מצבעים שונים בדף Firefox בלבד.

```
Element { -moz-border-colors: colors; }
```

במקום המילה "צד" המופיע בדוגמה התחריר יש לשים את הצד של גבול האלמנט אותו ברצוננו להגדיר באמצעות התוכנה border-colors.

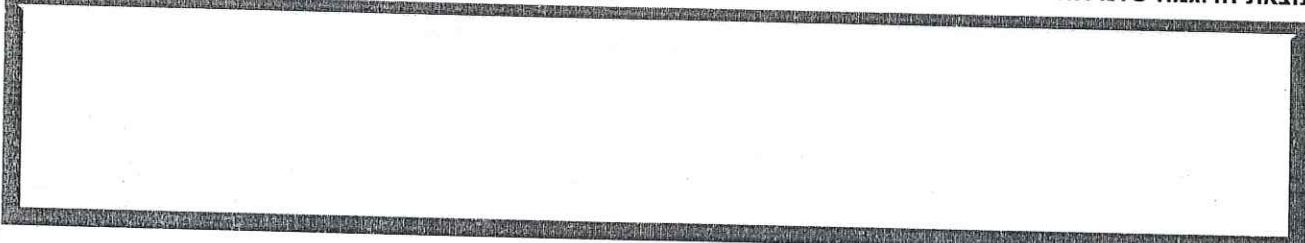
- border-top-color - •
- border-right-color - •
- border-bottom-color - •
- border-left-color - •

```
div { border-width: 9px;
```

```
      -moz-border-top-colors: green green green red red red yellow yellow yellow; }
```

דוגמה להגדרת גבול מרובה צבעים

תוצאת הדוגמה שלנו תהיה זו:

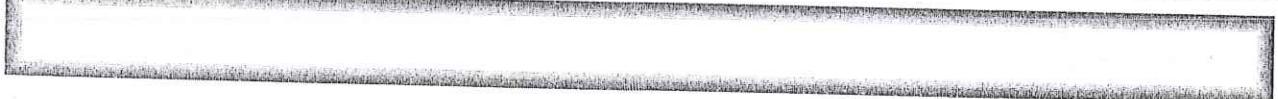


הסביר: כל הגדרת צבע הינה בת פיקסל אחד. לכן, אם ברצוננו למלא למשל גבול שעובי 9 פיקסל (כמו בדוגמה שלנו) علينا להגדיר תשעה צבעים - אחד עבור כל פיקסל. במידה ולא נגידר את כל הפיקסלים של עובי הגבול הדפן ישלים את יתרת הצבע בעצמו על פי הצבע האחרון שהוגדר לו.

דוגמה ליצירת אפקטים בצבע באמצעות התוכנה border-colors: הדוגמה הבאה נוצר אפקט של הילשوت צבע הגבול באורך הדרגת:

```
div {  
      border: 8px solid #000;  
      -moz-border-bottom-colors: #555 #666 #777 #888 #999 #aaa #bbb #ccc;  
      -moz-border-top-colors: #555 #666 #777 #888 #999 #aaa #bbb #ccc;  
      -moz-border-left-colors: #555 #666 #777 #888 #999 #aaa #bbb #ccc;  
      -moz-border-right-colors: #555 #666 #777 #888 #999 #aaa #bbb #ccc;  
      padding: 5px 5px 5px 15px;  
}
```

תוצאת הדוגמה שלנו תהיה זו:



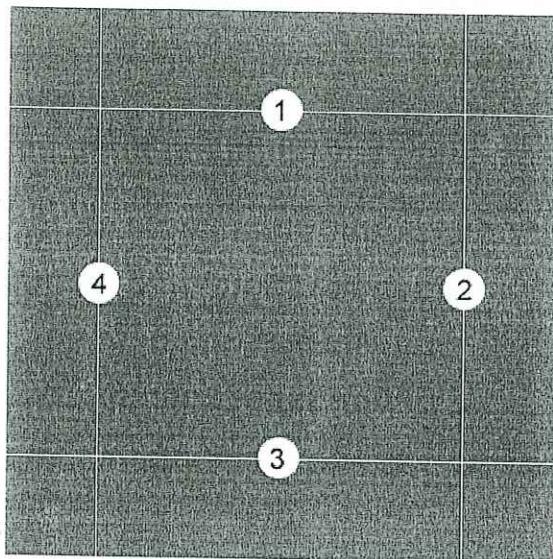
-border-image לתמונות גבולות לאלמנטים

ב Amendments CSS3 והתוכנה המובנית border-image יוצר אפקטים מרהיבים שבהם גבולות אלמנטים מורכבים על ידי Opera, Firefox, Safari and Chrome. החול תМОונות. נכון לשעת כתיבת שורות אלה תוכנה זו נתמכת על ידי הדפננים הבאים. מגרסה 16 chrom תומכת בתוכנה ללא קידומת ואילו שאר הדפננים דורשים קידומת.

תחבר הינה border-image

```
element { border-image: source slice repeat; }
```

1. - **source** פרמטר זה מיועד להצביע על הנתיב והמונה שתשתמש אותו כתמונה הגבול. פרמטר זה צריך להיות כתובת **חוקית (URI)** לתמונה שמנה אנו רצים להרכיב את הגבול.
2. - **slice** השני שאותו עליינו להعبر לתוכנה מכיל פרמטרים המגדירים תשע אזורים בתוך התמונה. אלו יוצרים את תשעת האזוריים על ידי העברת ארבעה ערכים (אחזים או פיקסלים) המגדירים 4 צירם בתמונה הנמונה. על מנת להבין כיצד אזוריים אלה מוגדרים ומה תפקידם בبنית גבולות אלמנט כדי שנבניט באירוע הבא:



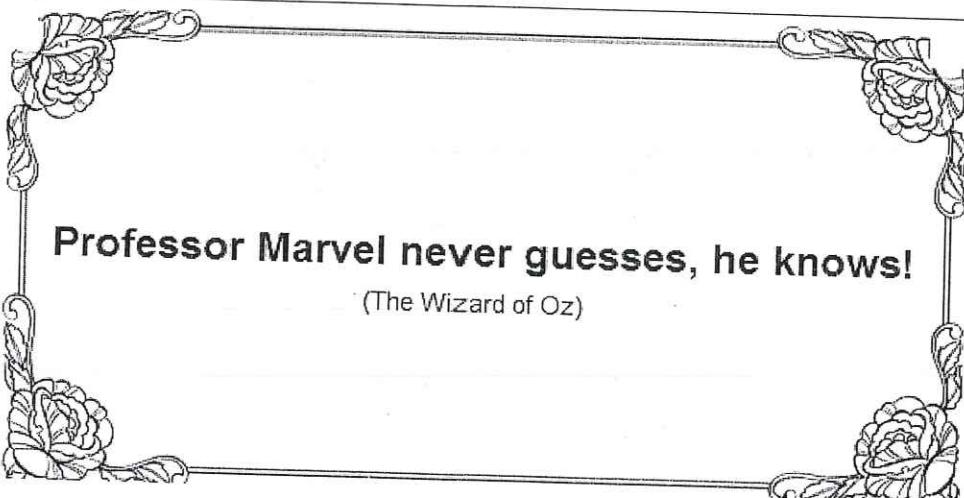
הערך הראשון: מצין את המרחק (בפיקסלים או באחוזים) של קו דמיוני מן הקצה העליון של התמונה; הערך השני: מצין את המרחק (בפיקסלים או באחוזים) של קו דמיוני מן הקצה הימני של התמונה; הערך השלישי: מצין את המרחק (בפיקסלים או באחוזים) של קו דמיוני מן הקצה התיכון של התמונה; הערך הרביעי: מצין את המרחק (בפיקסלים או באחוזים) של קו דמיוני מן הקצה השמאלי של התמונה. כפי שהדבר בא לידי ביטוי באירוע, ארבעת הקווים מחלקים את התמונה שלמו ל – 9 חלקים. מחלקים אלה הדףן מייצר את גבולות האלמנט שלו. את פינות התמונה ממקם הדףן בהתאם על ארבעת פינות האלמנט. את גוף הגבולות הוא ממקם בהתאם גם כן.

3. - **repeat** פרמטר זה יכול לקבל אחד מרבעה ערכים:
 - **Stretch** – ○
 - **Repeat** – ○
 - **Round** – ○
 - **Space** – ○
 - **Yo-Yo** – ○
 לא שאריות.
- בצורה מושלמת.

דוגמה קוד לשימוש border-image

```
div { -webkit-border-image: url(image.jpg) 45 20 45 30 repeat;
-moz-border-image: url(image.jpg) 45 20 45 30 repeat;
border-image: url(image.jpg) 45 20 45 30 repeat; }
```

בדוגמת הקוד זו צינו בפרמטר **slices** מספרים ללא יחידת מידת (אחוז או פיקסל). כאשר משתמשים במספרים בדף יתיחס אליהם בצורה שונה בהתאם לסוג התמונה שבה משתמשים. במקרה שמדובר בתמונה וקטורת הדףן יתיחס למספרים כקוודינטות ואילו במידת והמונה היא תמונה **raster** כגון **bitmap** וכדומה התייחסות למספרים תהיה כל פיקסל. כאמור, ניתן להשתמש גם במידות אחוזיות ובמקרה זה יש להוסיף את הסימן % לאחר כל מספר. איזור להמחשת דוגמת הקוד שלמעלה



ציירת צלויות לאלמנטים מרובעים בCSS3

פרק על ציירת צלויות לטקסט במדרך זה למדנו כיצד ליצור צל לטקסט בעזרת התוכנה `text-shadow` שfat3 CSS3 מאפשרת לנו ליצור בקלות הרבה גם צלויות לאלמנטים מרובעים וזאת באמצעות התוכנה `box-shadow`. שfat3 CSS3 מאפשרת לנו לקבוע מה יהיה צבע הצל, האם צבעו יהיה מדווג או מטושטש וכן מה יהיה גודלו והאם הוא יהיה ממוקם מימין, משמאל או מתחת לאלמנט שלנו.

הערה חשובה: יש לזכור ש `box-shadow` – אינו חלק ממודול הקופסא ב CSS – שימושות הדבר היא שאשר נחשב את גודלו של אלמנט ה `box-shadow` – לא יהיה חלק מהאלמנט וממדיו.

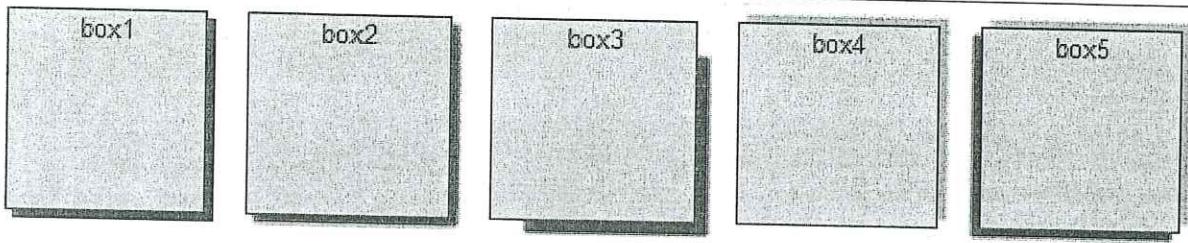
Element { box-shadow: inset horizontal vertical blur spread color; }
 התוכנה `box-shadow` מקבלת מינימום של שני ערכים ומקסימום של שישה ערכים (כמפורט בתחביר התוכנה לעיל). על ערכים אלה תוכל ללמידה מהטבלה הבאה:
 טבלת הערכים בתוכנה `box-shadow` יכולה לקבל

ערך	הסבר	חווב
Inset	מילת מפתח אופציונאלית הגורמת לכך שהצל יוצג בתוך האלמנט (מאזורי ה <code>padding</code> –	רשות
Horizontal length	המיקום האופקי (ימין לשמאלי) של הצל ביחס לאלמנט.	חווב
Vertical length	המיקום המאוזן (מלמעלה למטה) של הצל ביחס לאלמנט.	חווב
blur	עומק היטשטוש. ערך זה אינן יכול להיות שלילי. ככל שהערך גבוה יותר כך היטשטוש גדול יותר.	רשות
spread	גודל הצל. ערך חיובי יגרום לכך להתרפש כלפי חז' בצורה שווה מכל הצדדים. ערך שלילי יגרום לכך להתרפש פנימה אל תוך האלמנט.	רשות
color	צבע הצל	רשות

דוגמאות קוד העושות שימוש בתוכנה `box-shadow`:
 בדוגמה הקוד הבאה נגידר מספר מחלקות לכל אחת מהן תדים צורה אחרת של צל סיבי אלמנט מרובע כלשהו:

```
.box1 { box-shadow: 4px 4px; }
.box2 { box-shadow: 4px 4px 3px; }
.box3 { box-shadow: 12px 12px 2px -6px; }
.box4 { box-shadow: #999 4px -4px 2px 0; }
.box5 { box-shadow: #999 4px -4px 2px 0, -4px 4px 2px; }
```

אלה וuczאות ההגדרות שבודגמאות הקוד לעיל:



דוגמת קוד לייצור צל כחול מסביב לכל האלמנט

`box-shadow: 0 0 10px 5px blue;`

שכבות צל מסביב לכל האלמנט

דוגמת קוד לשימוש במילה השמורה `inset` כדי לייצר צל פנימי

`.inset-shadow {box-shadow: inset #999 4px -4px 2px 0;}`

צל פנימי

צבעים

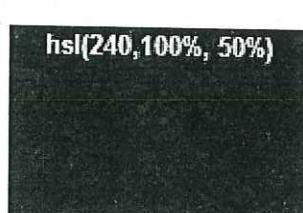
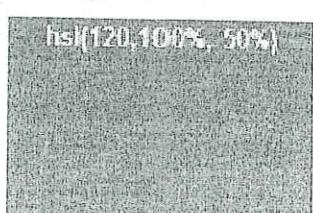
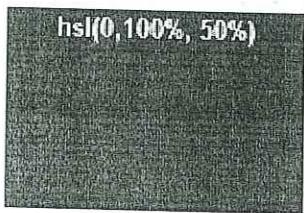
צבעים, שיקיפות ואטיות בCSS3

עד לגרסה CSS3 יכולים להשתמש בשני סוגי הגדרת צבעים :

1. הגדרת צבעים בעזרת RGB;
 2. הגדרת צבעים בעזרת ערכים הקסהdezימליים.
- גרסה CSS3 מציגה לנו דרך נוספת להגדרת צבעים באמצעות – **HSL** מונח שהואראשי תיבות של:
- **Hue** [גוון] בעזרת ערך זה נגדיר את הגוון של הצבע. בד"כ מדובר בסולם של 360 דרגות צבע המבतא גוונים שונים של צבעים.
 - **Saturation** - רזונה. סקלת מספרים הנעה בין 0 ל – 100 ומבטא את אחוזים שמצידם גורמים לצבע להיות רזוי יותר או פחות ובכך משתנה הצבע בהתאם לנבחר בערך הראשוני.(**hue**)
 - **Lightness** - גם כאן מדובר בערך אחוזי הבא לייצג את כמות האור בתוך הצבע מה שמצידיו משפיע גם כן על הגוון הסופי שלו.

דוגמת קוד לשימוש בהגדרות צבע מסוג HSL

```
<div style="background-color: hsl(0,100%, 50%);"></div>
<div style="background-color: hsl(120,100%, 50%);"></div>
<div style="background-color: hsl(240,100%, 50%);"></div>
```

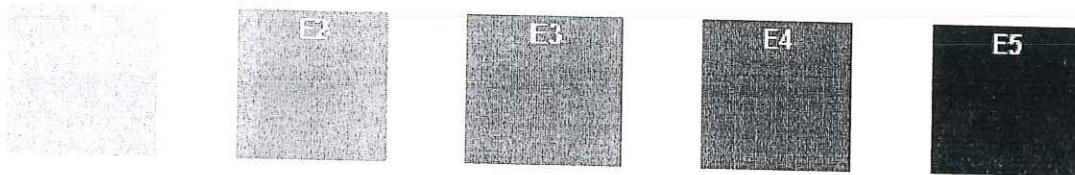


צבע HSLA

צבעים מסווג **HSLA** זההים לצבעים מסווג **RGB** אבלם נוסף להם ערך נוסף (המיוצג על ידי האות A) שבעזרתו ניתן לקבוע את מידת האטימות (opacity) של הצבע בסקללהعشرونית שנעה מ – 0 עד 1.

דוגמאות קוד לשימוש בצבעים מסווג HSLA

```
E1 {"background-color: hsla(240,100%,50%,0.2);"}  
E2 {"background-color: hsla(240,100%,50%,0.4);"}  
E3 {"background-color: hsla(240,100%,50%,0.6);"}  
E4 {"background-color: hsla(240,100%,50%,0.8);"}  
E5 {"background-color: hsla(240,100%,50%,1);"}
```

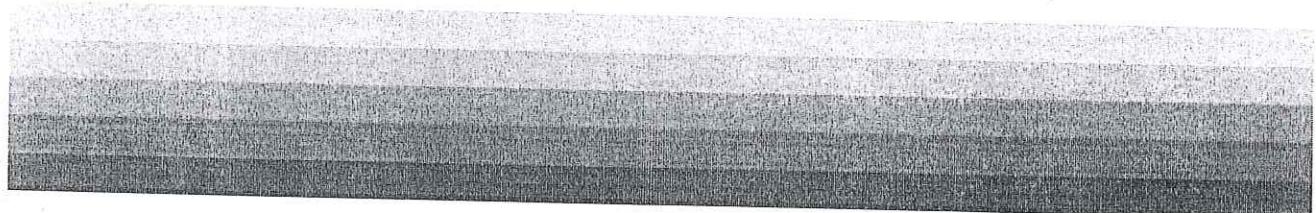


כפי שבא הדבר לידי ביטוי באירור בדוגמה הקוד الأخيرة ביצענו קפיאות של 0.2 בשקיפותו של אלמנט שצבעו הוגדר צבע כחול. ככל שמידת האטימות גדולה יותר כך צבע האלמנט ברור, עמוק ואטום יותר.

צבע **RGBA** זההים לצבעים מסווג **RGB** שאוטם הכרמו מגרסאות CSS קודמות, אבלם נוסף להם ערך נוסף (המיוצג על ידי האות A) שבעזרתו ניתן לקבוע את מידת האטימות (opacity) של הצבע בסקללהعشرونית שנעה מ – 0 עד 1.

דוגמאות קוד לשימוש בצבעים RGBA

```
E1 {"background-color: rgba(255,0,0,0.2);"}  
E2 {"background-color: rgba(255,0,0,0.4);"}  
E3 {"background-color: rgba(255,0,0,0.6);"}  
E4 {"background-color: rgba(255,0,0,0.8);"}  
E5 {"background-color: rgba(255,0,0,1);"}
```



יצירת שקייפות צבע בעזרות התוכנה opacity

כפי שראינו במדריכים הקודמים בפרק זה הצבעים מסווג **HSLA** ו**RGB** – מכילים ערך נוסף שבעזרתו ניתן לקבוע את מידת האטימות של הצבע. ביל' שום קשר לכך התווספה לCSS3 – תכונה חדשה, **opacity**: שבאמצתה ניתן לקבוע את מידת האטימות.

גם של צבעים מסווג RGB למשל שאין להם פרמטר שקייפות מובנה.

תחביר התוכנה **opacity**

```
Element { opacity: 0.33; }
```

הערך המופיע בתכונת **opacity** יכול לקבל נוע בין 0 ל – 1 והוא ערך עשרוני. ככל שהערך קרוב יותר ל – 0 כך הוא שקוף יותר. ככל שהערך קרוב יותר ל – 1 כך הוא אטום יותר.

דוגמאות קוד לשימוש בתוכנת **opacity** בדוגמה הבאה נגדיר רמת שקייפות שונה לשולש אלמנטים בעלי צבע RGB זהה ונראה כיצד רמת השקייפות משפיע על ניראות הצבע.

```
#E1 { background-color: black; opacity: 0.33; }  
#E2 { background-color: black; opacity: 0.66; }  
#E3 { background-color: black; opacity: 0.99; }
```



מעבר הדרגת'

מודנה של המילה **Gradient** בCSS - הוא מעבר הדרגת' בין לפחות צבע אחד לצבע אחר. שפת CSS3 מאפשרת ליצור **Gradients** פשוט יחסית בעוד שעד כה ניתן היה לבצע זאת בעזרת שימוש בלבד.

הו **Linear Gradient** מעבר הדרגת' מצבע בקו השר שבין שתי נקודות. הקוו יכול להיות אופקי או אנכי. סוג הקוו יקבע

כמובןஇzo צורה תהיה למעבר הדרגת' מצבע לצבע.

- **Linear Gradient** דוגמאות קוד לתמונת הדרגת' המבוקש. אי לך יש צורך בשימוש בפתרונות המיעודות התconaה כמעט ואינה נטמכת בצורתה הרשミת על ידי הדפסנים המרכזים השונים. אם תומך בתמונה באף צורה.

לכל דפסן. בדוגמה הקוד שלמטה השתמשנו גם בהגדרת צבע רקע רגיל למקורה שהדפסן אינו תומך בתמונה באף צורה.

מעבר הדרגת' של צבעים מלמעלה למטה

Element{

הגדרת צבע רקע למקורה שהדפסן אינו תומך בתמונה

background-color:#a8e9ff;

IE 5.5 - IE 9.0

```
filter:progid:DXImageTransform.Microsoft.gradient  
(GradientType=0,startColorstr=#a8e9ff, endColorstr=#052afc);
```

Firefox 3.6+

```
background-image:-moz-linear-gradient(top, #a8e9ff 0%, #052afc 100%);
```

Chrom 10+ Safari 5.1+

```
background-image:-webkit-linear-gradient(top, #a8e9ff 0%, #052afc 100%);
```

IE 10.0

```
background-image:-ms-linear-gradient(top, #a8e9ff 0%, #052afc 100%);
```

W3C

```
background-image:linear-gradient(top, #a8e9ff 0%, #052afc 100%);
```

Opera 11.10+

```
background-image:-o-linear-gradient(top, #a8e9ff 0%, #052afc 100%);
```

Safari 4+ Chrom 2+

```
background-image:-webkit-gradient(linear, right top,  
right bottom, color-stop(0%,#a8e9ff), color-stop(100%,#052afc));
```

תוצאות הקוד שלמעלה הם:



מעבר הדרגי של צבעים מלמטה למעלה

בדוגמה הבאה נגידר את מעבר הצבעים כך שהצבע הכהה יהיה מלמטה ואילו הבהיר יהיה למעלה:

Element{

הגדרת צבע רקע למקורה שהדפסן אינו תומך בתוכנה
background-color:#a8e9ff;

IE 5.5 - IE 9.0

```
filter:progid:DXImageTransform.Microsoft.gradient  
(GradientType=0,startColorstr=#a8e9ff, endColorstr=#052afc);
```

Firefox 3.6+

```
background-image:-moz-linear-gradient(bottom, #a8e9ff 0%, #052afc 100%);
```

Chrom 10+ Safari 5.1+

```
background-image:-webkit-linear-gradient(bottom, #a8e9ff 0%, #052afc 100%);
```

IE 10.0

```
background-image:-ms-linear-gradient(bottom, #a8e9ff 0%, #052afc 100%);
```

W3C

```
background-image:linear-gradient(bottom, #a8e9ff 0%, #052afc 100%);
```

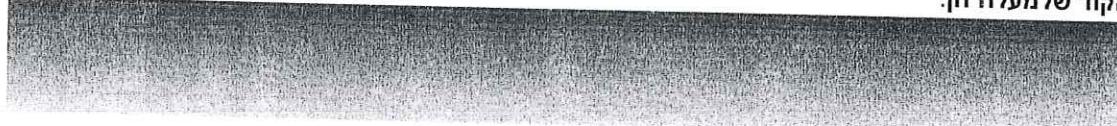
Opera 11.10+

```
background-image:-o-linear-gradient(bottom, #a8e9ff 0%, #052afc 100%);
```

Safari 4+ Chrom 2+

```
background-image:-webkit-gradient(linear, left bottom, left top, color-stop(0%,#a8e9ff), color-  
stop(100%,#052afc));}
```

תוצאות הקוד שלמעלה הן:



Element{

הגדרת צבע רקע למקורה שהדפסן אינו תומך בתוכנה
background-color:#a8e9ff;

מעבר הדרגי של צבעים מימין לשמאלי

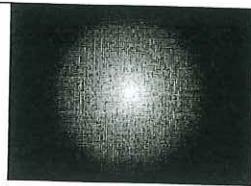
IE 5.5 - IE 9.0

```
filter:progid:DXImageTransform.Microsoft.gradient  
(GradientType=1,startColorstr=#a8e9ff, endColorstr=#052afc);
```

Firefox 3.6+

```
background-image:-moz-linear-gradient(right, #a8e9ff 0%, #052afc 100%);
```

Radial Gradient CSS3 - הוא מושג המתאר מעבר בין צבע אחד לפחות לנשנהו כאשר המעבר נעשה בעיגולים או אליפסות הholocims וגדלים מנוקודה נתונה. כדי להבין זאת טוב יותר די אם נביט באיזור הבא הממחיש את הדברים:



בגלל שנקון לשעת כתיבת שורות אלה אף דפפן בלבד אינטראנט אקספלורר 10 אינו תומך בתוכונה כפי שהוגדרה על ידי ארגון W3C אנו נדרשים להווסף לקוד את הקידומות המיעילות לכל דפפן. כמו כן, יש לשים לב שהתחביר של הפקודה אינו זהה בכל דפדףנים.

- Radial Gradient

הגדרת צבע רקע למקורה שהדפפן אינו תומך בתוכונה
`#element { background: #000000;`

Chrom 10+ Safari 5.1+

```
background-image: -webkit-gradient(radial, center center, 0, center center, 141, from(black), to(white), color-stop(25%, blue), color-stop(40%, green), color-stop(60%, red), color-stop(80%, purple));
```

Safari 4+ Chrom 2+

```
background-image: -webkit-radial-gradient(center center, circle contain, black 0%, blue 25%, green 40%, red 60%, purple 80%, white 100%);
```

Firefox 3.6+

```
background-image: -moz-radial-gradient(center center, circle contain, black 0%, blue 25%, green 40%, red 60%, purple 80%, white 100%);
```

IE10+

```
background-image: -ms-radial-gradient(center center, circle contain, black 0%, blue 25%, green 40%, red 60%, purple 80%, white 100%);
```

Opera (13+)

```
background-image: -o-radial-gradient(center center, circle contain, black 0%, blue 25%, green 40%, red 60%, purple 80%, white 100%);}
```

תחביר הפקודה **gradient**

כפי שביקולתכם להתרשם מדוגמאות הקוד לעליה התחביר שונה מdapfen. לכן בחרתי להסביר את תחביר הפקודה כפי שהוא צריך להיות על פ' הגדרות ארגון W3C את התאמות לדפפניהם השונים תחול לביצ'ן לאחר מכן בעצמכם.
`background-image: radial-gradient(60px 45px1, circle2 closest-side3, red4,`

⁵blue, ⁶green⁷)

הסברים לפרמטרים השונים עפ' המספרים מעל כל פרמטר:

1. **אופציונלי** - מיקום הנקודה שממנה יתבצע ה- **gradient**. - בדוגמה שלנו מדובר ב 60 - פיקסלים מצד שמאל ו - 45

פיקסלים מלמעלה. ניתן להעביר כאן גם ערכים אחוזיים.

2. **אופציונלי** - צורת ה- **gradient** - יכולה להיות עיגול (circle) או אליפסה (ellipse) ערך ברירת המחדל הוא אליפסה.

3. **אופציונלי** - גודל ה- **gradient** - יכול להיות מוגדר באמצעות אחת מAMILות המפתח הבאות-closest-side, closest-⁸:

- **אופציונלי** - גודל ה- **gradient** באנטז'וות ערכיהם מספריים שבאמצעותם הרשוו שבחם נגידר את רוחב ה-

gradient ואילו באמצעות השני את גובהו. ברירת המחדל היא **farthest-side**.

4. **חויה** - הצבע ההתחלתי של ה- **gradient** -

5. **אופציונלי** - צבע ביןים בין הצבע ההתחלתי לצבע הסופי המוגדר בפרמטר מס' 7.

6. **אופציונלי** - נקודת עצירה, המגדירה כי נקודת הביניים לעצירה בין צבע הביניים (כחול בדוגמה שלנו) לצבע הסופי.

תהייה לא באמצעות הדרך (ברירת המחדל) בין שני הצבעים אלא לאחר 60 אחוזים מהדריך.

7. חובה - הציבו הsofar שאלוי על ה gradient - להגיע בסופו של דבר.

Chrom 10+ Safari 5.1+
background-image:-webkit-linear-gradient(right, #a8e9ff 0%, #052afc 100%);
IE 10.
background-image:-ms-linear-gradient(right, #a8e9ff 0%, #052afc 100%);
W3C
background-image:linear-gradient(right, #a8e9ff 0%, #052afc 100%);
Opera 11.10+
background-image:-o-linear-gradient(right, #a8e9ff 0%, #052afc 100%);
Safari 4+ Chrom 2+
background-image:-webkit-gradient(linear, right bottom, left bottom,
color-stop(0%,#a8e9ff), color-stop(100%,#052afc));}

תוצאות הקוד של מילנה הן:



Element{

מעבר הדרמטי של צבעים משמאלי לימין

הגדרת צבע רקע למקורה שהדף אינו תומך בתוכינה
background-color:#a8e9ff;

IE 5.5 - IE 9.0

filter:progid:DXImageTransform.Microsoft.gradient
(GradientType=1,startColorstr=#a8e9ff, endColorstr=#052afc);

Firefox 3.6+

background-image:-moz-linear-gradient(left, #a8e9ff 0%, #052afc 100%);

Chrom 10+ Safari 5.1+

background-image:-webkit-linear-gradient(left, #a8e9ff 0%, #052afc 100%);

IE 10.0

background-image:-ms-linear-gradient(left, #a8e9ff 0%, #052afc 100%);

W3C

background-image:linear-gradient(left, #a8e9ff 0%, #052afc 100%);

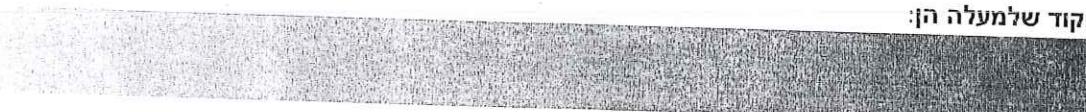
Opera 11.10+

background-image:-o-linear-gradient(left, #a8e9ff 0%, #052afc 100%);

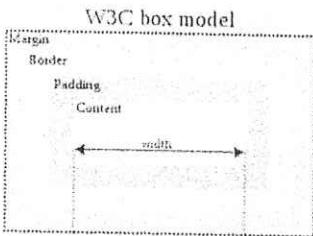
Safari 4+ Chrom 2+

background-image:-webkit-gradient(linear, left bottom, right bottom,
color-stop(0%,#a8e9ff), color-stop(100%,#052afc));}

תוצאות הקוד של מילנה הן:



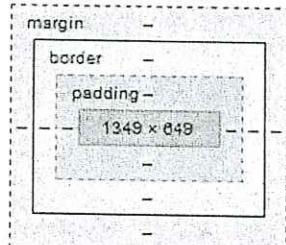
באתר w3schools.com



The Box Model

מבנה הקופסה

ב CSS – ניתן לדמיין כי כל רכיב נראה כתיבה מלכנית (box) המורכבת ממחולקים הבאים:



חלק פנימי הנקרא content (תוכן)

מרוח פנימי הנקרא padding (ריפורד)

גבול (בעל עובי מסוים) הנקרא border.

מרוח חיצוני הנקרא margin (מרוח)

קביעת אורך ורוחב של רכיב

האורך והרוחב של רכיב כוללים בתוכם את הרכיב עצמו, ה-padding וגם ה-border.

margin נמצא מחוץ לרכיב.)

box-sizing

במודל הקופסה הרגיל ה-padding והגבولات מתווספים לרוחב ולגובה של האלמנט. למשל:

```
.myelement {  
    width: 500px;  
    padding: 10px;  
    border: 1px solid black;  
}
```

בקוד הנ"ל רוחב האלמנט יהיה 522 פיקסלים, כלומר: 500 רוחב + 10 פידיג מכל צד + 1 גבול מכל צד. אם נוסיף לסלקטור הנ"ל את השורה הבאה:

```
box-sizing: border-box;
```

הרווח של האלמנט ישאר 500 פיקסלים – הpadding והגבولات ייכנסו פנימה.

לפעמים בהגדלת רוחב מבודדת אחוזים זהו המודל היחיד שאפשר להשתמש בו. למשל, אלמנט textarea, שנדיר לו רוחב של 100% – במודל הקופסה הרגיל רוחבו יהיה 100 אחוז ועוד 2 פיקסלים – בהנחה שיש גבול של פיקסל אחד מכל צד. שימוש בborder-box- יכול לפתור את הבעיה. יש שמחילים את המודל זהה.

Width and Height of an Element

Example

```
div{  
    width: 320px;  
    padding: 10px;  
    border: 5px solid gray;  
    margin: 0px;  
}
```

CSS Border Properties

The CSS border properties allow you to specify the style, size, and color of an element's border. The border-style property can have from one to four values.

- **border-style: dotted solid double dashed;**
 - top border is dotted
 - right border is solid
 - bottom border is double
 - left border is dashed
- **border-style: dotted solid double;**
 - top border is dotted
 - right and left borders are solid
 - bottom border is double
- **border-style: dotted solid;**
 - top and bottom borders are dotted
 - right and left borders are solid
- **border-style: dotted;**
 - all four borders are dotted

The border-style property is used in the example above. However, it also works with border-width and border-color.

Border - Shorthand property

```
p{  
    border: 5px solid red;  
}
```

CSS Outline

An outline is a line that is drawn around elements (outside the borders) to make the element "stand out." However, the outline property is different from the border property.

The outline is not a part of an element's dimensions; the element's total width and height is not affected by the width of the outline.

Margin

The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent.

The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once.

Possible Values

Value	Description
auto	The browser calculates a margin
<i>length</i>	Specifies a margin in px, pt, cm, etc. Default value is 0px
%	Specifies a margin in percent of the width of the containing element
inherit	Specifies that the margin should be inherited from the parent element

Note: It is also possible to use negative values, to overlap content.

Padding

The padding clears an area around the content (inside the border) of an element. The padding is affected by the background color of the element.

The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property can also be used, to change all paddings at once.

הגדרות רוחב וגובה

The height and width properties are used to set the height and width of an element.

The height and width can be set to auto (this is default. Means that the browser calculates the height and width), or be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block.

This > div <element has a height of 100 pixels and a width of 500 pixels.

Note: The height and width properties do not include padding, borders, or margins; they set the height/width of the area inside the padding, border, and margin of the element!

The following example shows a > div <element with a height of 100 pixels and a width of 500 pixels:

Example

```
div {  
    width: 500px;  
    height: 100px;  
    border: 3px solid #8AC007;  
}
```

Using width, max-width and margin: auto;

As mentioned in the previous chapter; a block-level element always takes up the full width available (stretches out to the left and right as far as it can.)

Setting the width of a block-level element will prevent it from stretching out to the edges of its container. Then, you can set the margins to auto, to horizontally center the element within its container. The element will take up the specified width, and the remaining space will be split equally between the two margins:

This > div <element has a width of 500px, and margin set to auto.

Note: The problem with the > div <above occurs when the browser window is smaller than the width of the element. The browser then adds a horizontal scrollbar to the page.

Using max-width instead, in this situation, will improve the browser's handling of small windows. This is important when making a site usable on small devices:

This > div <element has a max-width of 500px, and margin set to auto.

Tip: Drag the browser window to smaller than 500px wide, to see the difference between the two divs!

Here is an example of the two divs above:

Example

```
div.ex1 {  
    width: 500px;  
    margin: auto;  
    border: 3px solid #8AC007;  
}
```

```
div.ex2 {  
    max-width: 500px;  
    margin: auto;  
    border: 3px solid #8AC007;  
}
```

מיקום של רכיבים בעזרת CSS

css מציעה 4 שיטות למקום רכיבים:

- static
- relative
- absolute
- fixed

static positioning מדריך זה עוסק בכללי CSS המאפשרים לסדר אלמנטים שונים על הדף ולשלוט במקומות. יש לציין שדף מסוים לא תומך בהכרח בתוכנה כלשהי של סטנדרט ה CSS - ולא תמיד מובטח שהוא מממש אותה بصورة נכונה. כמו כן, הסטנדרט עצמו משאיר לדפננים חופש פעולה במקרים מסוימים וכך קיימים הבדלים קלים באופן שדפננים שונים מיישמים תכונות CSS אלה.

ב CSS-קיימות מספר שיטות מיקום. צורת המיקום של אלמנט נקבעת ע"פ ערכי התכונות position ו- float . נסקרו בקצרה את הערכים שתוכנה position עשוייה לקבל:

- **static** - בירית המחדל - מיקם את האלמנט ע"פ כללי הזרימה הרגילים של דף HTML.
- **relative** - מיקם את תיבת האלמנט תוך הסיטה למרחוק מוגדר מיקומה הרגיל בדף.
- **absolute** - מיקם את תיבת האלמנט תוך הסיטה למרחוק מוגדר מגבלות התיבה הממוקמת שכילתה אותה.
- **fixed** - מקרה פרטי של absolute התיבה המכילה שביחס אליה מתבצעת ההזזה היא חלון הדף.

התכונה float עשויה לקבל את הערכים הבאים:

- **right** - הzz את תיבת האלמנט ימינה ככל הניתן.
- **left** - הzz את תיבת האלמנט שמאליה ככל הניתן.
- **none** - אין השפעה על האלמנט .

זרימה רגילה (normal flow)

שיטת מיקום זו היא ברירת המחדל - הדפדן משתמש בה עבור כל אלמנט שערק המיקום (position) שלו אינו מוגדר או שווה ל "static" - בשיטה זו, תיבות מסווג 'block' ממוקמות זו מתחת זו בתוך גבולות התיבת המכילה אותן ("תיבת האם"). יש לציין שכאשר שתי תיבות 'block' מונחות בצד זה זו, אזי השולטים של העליונה מתמצאים עם שלוי התיבת התחתונה באופן שנשארים השולטים העבים יותר מבין השניים.

דוגמאות:

זהו פיסקה המוגדרת בתוך ה - div. שימוש לב שהיא תופסת את כל רוחב תיבת האם שלה.

זהו הפיסקה השנייה.

שימוש לב שהמרוחה האנכי בין שתי הפיסקות הינו 8אך למרוחה שהשולטים האנכיים עברו שתי הפיסקות (מלמעלה וממטה) מוגדרים ל - 8אך, لكن הינו מצפים למרוחה של 16אך, אולם כפי שהסביר השולטים האנכיים של התיבות מתמצאים.

ניתן גם להגדיר רוחב של בלוק באמצעות התכונה - width . בדוגמה הבאה נגידר פיסקה ברוחב של 50%. רוחב באחוזים מחושב מתוך רוחבה של תיבת האם:

זהו פיסקה צרה יותר

תיבות מסווג 'inline' לעמודה זאת, ממוקמות זו לצד זו משמאל לימין (או מימין לשמאל אם מוגדר rtl: direction: rtl; עד לנקודה שלא יותר מקום לתיבה נוספת נוספת בתוך "תיבת האם". בשלב זה האלמנט הבא מתפצל ומשיר בשורה חדשה. לדוגמה:

[קישור מס' 1](#) [קישור מס' 2](#) [קישור מס' 3](#) [קישור מס' 4](#) קישור ארוך מאד - אלמנט שמשיך בשורה חדשה

מיקום מוחלט (position: absolute)

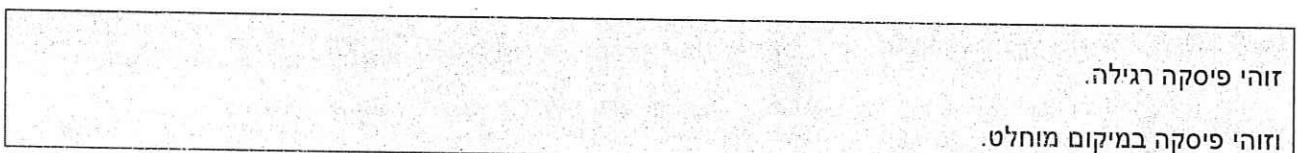
שיטת מיקום זו מופעלת על כל אלמנט שערך ה position - שלו הינו absolute . שמו שנבחר עבור שיטה זו הוא (מוחלט), מטעה מעט, מכיוון שלמעשה מדובר גם כאן במיקום יחסי . אלא שבניגוד לשיטה relative , מיקום האלמנט נקבע ביחס לתיבת האם שלו (מייד נראה הגדרה מדויקת יותר לכך), ולא ביחס למיקומו המקורי .

בהקשר של שיטת מיקום זו, קיימת הגדרה מיוחדת למונח "תיבת האם". בהקשר זה, תיבת האם הינה התיבת העוטפת הקרובה ביותר לאלמנט, שערך ה position -שלה הוא absolute , relative או fixed . ולא קיימת תיבה העונה להגדרות הנ"ל, "יחס אלמנט ה body -האלמנט הראשי של הדף) ל缇בת זו .

במיקום מוחלט אנו משתמשים בערכי התכונות left , top , bottom , right . על מנת למקם את תיבת האלמנט ביחס לתיבת האם. אם למשל הגדרו מרחוק הזהזה כך:

bottom: 4px; left: 20px; אז תמקם הוויבת כך שהצלע התחתונה שלה רוחקה 4 פיקסלים מצלעה התחתונה של תיבת האם, והצלע השמאלית שלה רוחקה 20 פיקסלים מצלעה השמאלית של תיבת האם.

נראה זאת בדוגמה הבאה (缇בת האם בסוגרת ירוקה):



זה קוד ה -HTML של הדוגמה:

```
<div class="positioned">
  <p>זוהי פיסקה רגילה</p>
  <p style="position: absolute; bottom: 4px; left: 20px">
    זוהי פיסקה במיקום מוחלט
  </p>
</div>
```

- CSS:

```
.positioned {
  position: relative;
  border: 2px solid green;
  height: 120px;
  background: #ddd;
}

.positioned p {
  border: 1px red solid;
  margin: 4px;
  padding: 4px;
}
```

תיבת האם במקרה זה הינה ה `div`-העוטף את שתי הפיסקאות. מיקומו של `div` הוגדר כ `relative` -לא כל הזמן, מכיוון שלא ניתן ערכיהם עבור התכונות `left`, `top` וכו'.
זהו טכנייה נפוצה למיקום מוחלט - מגדירים תיבת אם בעזרת `position: relative` ללא הדזה, וממקרים אלמנטים ביחס אליה.

מיקום קבוע (`position: fixed`)

שיטת מיקום זו דומה מאד לשיטת המיקום המוחלט, (`absolute`) ההבדל היחיד הוא שתיבת האם, התיבה שבייחס אליה ממוקם האלמנט, מוגדרת להיות חלון הדף. כתוצאה לכך, אלמנט שמדובר בשיטה זו ישמר על מיקומו המוחלט במסר גם כשהעמוד נבלל למעלה או למטה.
בדרך כלל משתמשים בשיטה זו כדי להציב תפריטים שמיקומם על המסך נשאר קבוע. הערה: אינטרנט אקספלורר 6 אינו תומך במיקום מסוג זה. ניתן להשתמש בתכונה `background-attachment: fixed` להציג אפקט דומה עם תמונה רקע בלבד.

Overlapping Elements

When elements are positioned, they can overlap other elements.

The `z-index` property specifies the stack order of an element (which element should be placed in front of, or behind, the others.)

An element can have a positive or negative stack order:

```
img {  
    position: absolute;  
    left: 0px;  
    top: 0px;  
    z-index: -1;  
}
```

An element with greater stack order is always in front of an element with a lower stack order.

<u>bottom</u>	Sets the bottom margin edge for a positioned box	auto <i>length</i> %
<u>left / right</u>	Sets the left/right margin edge for a positioned box	auto <i>length</i> %
<u>overflow</u>	Specifies what happens if content overflows an element's box	auto hidden scroll visible
<u>position</u>	Specifies the type of positioning for an element	absolute fixed relative static
<u>top</u>	Sets the top margin edge for a positioned box	auto <i>length</i> %
<u>z-index</u>	Sets the stack order of an element	<i>number</i>

הציגת רכיבים

מציג רכיבי html הציגה מתבצעת באמצעות תכונת css הנקראת `display`.

inline display

תצוגת רכיב `inline` מביאה את האופי הבסיסי של html ו-css טכנולוגיות של מסמך.

- בתצוגה זו רוחב הרכיב מחושב באופן אוטומטי
 - הרכיב מתווסף למסמך בשורה הנוכחית
 - אין הכנסה של שורה חדשה לפניה או אחרי הרכיב
- למרות שקבענו את הרוחב, הרכיב ממשיל להתנהג כ `inline` – ומשתלב בטקסט הכלול.
- דוגמאות לרכיבים שבירית המחדל שלהם היא `display:inline` :

- `span`
- `a` (קישורים)
- `img` (למרבה הפתעה)
- (`sup`, `sub`) ועוד אלמנטים שימושיים על תצוגת טקסט)

block display

בתצוגה זו הרכיב "טופס" את כל רוחב הדף.

- לפני הרכיב ואחריו מוכנסות שורות חדשות(`line break`)
 - בירית המחדל של רוחב הרכיב היא רוחב הדף כולו.
 - גם אם קובעים את רוחב הרכיב (למשל `width:150px`) הרכיב עדין "טופס" את השורה כולה (כפי שניתן לראות באמצעות כלי `debug`)
- גם אם קובעים את רוחב הבלוק לרוחב מסוים, עדין הוא "טופס" את כל השורה, ומכליל `line-break` לפני ואחריו.

inline-block display

1. היתרון של אלמנטים מסווג אינליין הוא שהם יכולים להופיע זה לצד זה, וחסרונם הוא שהפידינג והשוללים האנכיים שלהם לא מושפעים ואפשר להגדיר להם רוחב וגובה.

2. היתרון של אלמנטים מסווג בלוק – הוא שהגדירות הגובה, הרוחב, פידינג והשוללים האנכיים שלהם עובדים כמו שצライ, אך חסרונם הוא שאין יכולים לעמוד זה לצד זה.

בדרכ כל משתמשים `block`- כדי לעמוד אלמנטים מסווג בלוק אחד ליד השני. תצוגת אינליין-בלוק לוקחת את הטוב שבשני העולמות: האלמנטים עומדים לצד זה – כמו אינליין, והם מקבלים גובה, רוחב, שוללים אנכיים ופידינג אנכיים – כמו בלוק.

display none

פשוט מעלימה את הרכיב מהתצוגה.

Hide an Element - display:none or visibility:hidden?

Hiding an element can be done by setting the display property to none .The element will be hidden, and the page will be displayed as if the element is not there:

Example

```
h1.hidden {  
    display: none;  
}
```

visibility:hidden ; also hides an element.

However, the element will still take up the same space as before. The element will be hidden, but still affect the layout:

Example

```
h1.hidden {  
    visibility: hidden;  
}
```

Float

רחף שמאליה, רחף ימינה
זאת יכולת הפרישה (layout) של הרכיבים בדף באמצעות CSS.

המשמעות של float
float נתק את הרכיב מזרימת הנתונים הטבעית שלו, או מהרץף הטבעי של הדף, וואז נותן לו לרוחף כמה אפשר, ימינה או שמאליה.
את הטכניקות הנפוצות ביותר לעימוד דפי אינטרנט היא באמצעות float (להלן: היצפות). ה-float-מעולם לא כועד לעימוד דפים, אלא לגילשת טקסט סביר לתמונות ולאלמנטים שונים – ממש כפי שהיינו עושים בוורד או באינדייזן. מפני שעימוד דפים אינם הייעוד המקורי של היצפות, נוצרות בעיות שונות שימושים בfloat-לעימוד דפים.

html

טכנית בסיסית לעימוד עם floats

קוד ה html-צריך להראות כך:

```
<div class="container">
  <div class="content">
  </div>
  <aside>
  </aside>
</div>
```

css

כעת ל CSS אנו רוצים שני בלוקים (התוכן והפס הצדדי) יעמוד זה לצד זה, ולא אחד מעל השני – שזו ההתנהגות הרגילה של אלמנטים עם display:block כמו div, aside וכו'.
לשם כך אנו צריכים:

א. להגדיר לכל אחד מהבלוקים רוחב, ולזודא שהקונטינר רחב די כדי להכיל את שניהם
.container {
 width: 800px;
}
.content {
 width: 600px;
}
aside {
 width: 200px;
}

הגדרת רוחב בלבד – האלמנטים עומדים אחד מעל השני

ב. להציג את כל אחד מהבלוקים

```
.content {  
    width: 600px;  
    float: right;  
}  
  
aside {  
    width: 200px;  
    float: left;  
}
```

הצפות – האלמנטים עומדים אחד לצד השני

קריסת הקונטינר וגלישת טקסט לא רצiosa
כל מי שעמיד אחר עם הצפות יודע שאי אפשר להסתפק במה שכתבנו לעיל. אם ניתן עכשו צבע רקע לקונטינר – שום דבר לא יצביע, כי גובהו של הקונטינר עצמו הוא אפס. אלמנטים שיש להם float יוצאים מהזרימה של המסמן. כל האלמנטים בתוך הקונטינר מוצפים וכך הוא מתנהג כמו דיב ריק.

```
.container {  
    width: 800px;  
    background:#fffff;  
}
```

למרות שהגדירנו צבע רקע לבן לקונטינר, אטו לא רואים אותו כי גובה הקונטינר הוא אפס
עוד בעיה יכולה להיווצר אם נוסיף אלמנט לאחר הקונטינר (למשל footer) הטיקסט באלמנט החדש יגולש מסביב
אלמנטים המוצפים.

```
<div class="container">  
    <div class="content">  
    </div>  
    <aside>  
    </aside>  
    </div>  
    <footer></footer>
```

הוספנו אלמנט אחריו הקונטינר והטיקסט שבו גולש סביב האלמנטים המוצפים

פתרונות

נציג כאן שתי פתרונות אפשריים. כולם יביאו אותה התוצאה – הקונטינר יתחשב בגובה האלמנטים שבתוכו, והפוטר יתחל בשרה חדשה אחרי הקונטינר.

פתרון א' – הוספת דיב עם class=clear לאחר האלמנטים המוצפים

הградת clear:both לאלמנט, תגרום לו להפסיק את האצופות שמעליו ולהתחל שורה חדשה. אחד הפתרונות הפופולריים לביעות שה-float מוצוץ הוא להוסיף דיב ריק לאחר האלמנטים המוצפים, ובcss-להגדר לו clear:both.

```
<div class="container">
  <div class="content">
    </div>
    <aside>
      </aside>
    <div class="clear"></div>
  </div>
  .clear {
    clear: both;
  }
}
```

• **יתרון:** אפשר להתחיל הצפות חדשות מיד אחרי הדיב (אפילו בתוך הקונטינר)

• **חררון:** מציריך הוספת דיב

פתרון ב' – הוספת overflow:hidden לקונטינר

התcona overflow קובעת מה יש לעשות עם תוכן שחורג מגבולות האלמנט. לדוגמה: אם יש דיב שהגדרכנו לו גובה של 200 פיקסלים, ובתוכו יש תמונה בגובה של 300 פיקסלים – מה יקרה עם 100 הפיקסלים שחורגים מגובה הדיב? בעזרת הגדרת overflow לדיב, אפשר לקבוע האם להוסיף פסי גלילה, האם לחתם לתמונה לחורג מגבולות הדיב או האם להסתיר את החלק החורג.

לפי תקן ה-W3C ישנו מגדירים overflow visible שאינו נוצר הקשר עיצוב של בלוק חדש. או במילים אחרות, אם נוסיף overflow:hidden לקונטינר שמכיל רק אלמנטים מוצפים, הקונטינר יקבל גובה לפי האלמנטים שבתוכו וגובהו שליל ייפטרו.

```
.container {
  width: 800px;
  overflow: hidden;
}
```

• **יתרון:** לא מציריך אלמנט נוסף, קוד פשוט

• **חררון:** כל מה שחורג מגבולות הקונטינר יוסתר

The float Property

In its simplest use, the float property can be used to wrap text around images. The following example specifies that an image should float to the right in a text:

Example

```
img {  
    float: right;  
    margin: 0 0 10px 10px;  
}
```

The clear Property

The clear property is used to control the behavior of floating elements.

Elements after a floating element will flow around it. To avoid this, use the clear property.

The clear property specifies on which sides of an element floating elements are not allowed to float:

Example

```
div {  
    clear: left;  
}
```

The clearfix Hack - overflow: auto;

If an element is taller than the element containing it, and it is floated, it will overflow outside of its container.

Then we can add overflow: auto ;to the containing element to fix this problem:

```
.clearfix {  
    overflow: auto;  
}
```

הוספת padding לריפוד

ראינו פתרונות לביעות שההצפות יוצרות. אך שוב, כל מי שבנה כמה אתרים בחיים יודע שהוא קצת יותר מסובך. למשל, אם נוסיף ריפוד של 10 פיקסלים לאזור התוכן ולפס הצדדי, הם כבר לא יהיו ברוחב שהגדרכנו, אלא גדולים ב-20 פיקסלים מרוחב זה. זאת בגלל מודל הקופסה, הקובלע שהגודל הכלול של האלמנט הוא הרוחב שהגדרכנו בתוספת הריפוד והגבלוות.

```
.container {  
    width: 800px; /* 620 + 220 = 840, doesn't fit in the 800px container */  
}  
.content {  
    float: right;  
    width: 600px;  
    padding: 10px; /* now .content is 620px wide */  
}  
aside {  
    float: left;  
    width: 200px;  
    padding: 10px; /* now aside is 220px wide */  
}
```

רוחב האלמנטים גדול בגלל הריפוד, ולכן אין להם מקום עוד לעמוד זה לצד זה

פתרון א' – הפחיתה של הריפוד שהוספנו מהרווח

הפתרון פשוט ביותר הוא להפחית את גודל הריפוד מרוחב האלמנט. לאחר והגדרכנו ריפוד מכל הצדדים של 10 פיקסלים, אנו מקבלים תוספת של 10 פיקסלים מצד ימין ו-10 פיקסלים מצד שמאל. כך שעליינו להפחית סה"כ 20 פיקסלים מהגדרת הרוחב בכל אחד מהאלמנטים המוצפים.

```
.container {  
    width: 800px; /* 580+20 + 180+20 = 800, perfect! */  
}  
.content {  
    float: right;  
    width: 580px; /* 600-20 */  
    padding: 10px;  
}  
aside {  
    float: left;  
    width: 180px; /* 200-20 */  
    padding: 10px;  
}
```

שניהם הפתרונות המוצעים כאן יביאו לאותה התוצאה – האלמנטים יעמודו זה לצד זה עם תוספת הריפוד

- **פתרון:** עובד בכל הדפדפנים

- **חסרון:** מסורבל, מקשה על שינויים, אי אפשר להשתמש באחיזות אם יש border

פתרון ב' box-sizing:border-box

הפתרון שצובר פופולריות כיום הוא שימוש במודל הקופה אחר, שבו הריפורד לא מוסיף רוחב לאלמנט, אלא פונה כלפי פנים, על ידי הגדרת border-box שלו לב ש-sizing: border-box. מכך נדרש קידומת למוזילה ולוובקיט. באופן מפתיע, דוקא אקספלורר 8 ומוללה תומך ב-sizing-box-לא שום קידומת. השורות הבאות יחלו את מודל הקופה הנ"ל על כל האלמנטים במסמך:

```
* {  
    -moz-box-sizing: border-box;  
    -webkit-box-sizing: border-box;  
    box-sizing: border-box;  
}
```

- **יתרון:** מודל קופה אינטואיטיבי יותר, מאפשר לעבוד באחיזים בקודות
- **חסרון:** לא עובד באקספלורר 7.

קיצור טקסט ארוך עם שלוש נקודות

לפעמים, צריך להגביל את מספר המילים המוצגות בתקצירים פשוטים, בתפריטי ניווט או בכותרות, כדי שהעיצוב לא ייפגע. ניתן לכך טקסטים ארוכים דרך תכנות, למשל על ידי פונקציה שסופרת את מספר האותיות וחותכת את התווים שחרוגים מהמספר שהוגדר, והוספת שלוש נקודות היכן שנחתכו תווים. חסרון מרכז במספר התווים הוא: איננו יודעים מראש מה הרוחב שמספר תווים מסוים תופס כי כל זאת היא ברוחב שונה – למשל, הרוחב של האותיות 'י' ו'ו' קטן מאותיות כמו 'ש' ו'ת'. כמו כן, ישנו גורמים נוספים היכולים להשפיע על רוחב האותיות כמו סוג פונט, משקל, גודל הפונט ורווח בין תווים.

אם כל מה שאתם רוצים לעשות הוא לוודא שהtekst לא יתרפס על שתי שורות, תוכלם להשתמש בשלושה כללי CSS כדי לפתור את הבעיה:

1. white-space: nowrap – מודוא שהtekst לא עובר לשורה חדשה.
2. overflow:hidden – שהtekst החורג מרווח האלמנט יוסתר.
3. text-overflow: ellipsis – יוסיף שלוש נקודות היכן שהtekst נחתך, אם הטקסט באמת נחתך.

בעוד שני הכללים הראשונים עובדים בכל הדפדפנים – הוספה שלושת הנקודות תעבור רק לדפדפנים מודרניים (כולל אקספלורר 9 ומעלה)

הtekst שמופיע בפסקה זו יחתך לאחר שליא יהיה לו יותר מקום בשורה. נסו לשנות את גודל החלון ותראו <k>...</k>.שהמשפט מקבל סימן של שלוש נקודות היכן שהוא נחתך

```
p.ellipsis {  
    white-space: nowrap;  
    overflow: hidden;  
    -ms-text-overflow: ellipsis;  
    text-overflow: ellipsis;  
}
```

הtekst שמופיע בפסקה זו יחתך לאחר שליא יהיה לו יותר מקום בשורה. נסו ...

הtekst יחתך לפני רוחב האלמנט ובסיומו יהיו שלוש נקודות. (לחצו לצפייה בדוגמה החיה ב codepen.)

סיכום

ראינו כיצד ליצור קוד בסיסי ללמידה דף בעל שני טורים עם float. מוסיףנו את הביעות השונות שנוצרות לכך וכייזד לפטור אותן. ל float יש עוד שימושים שקשורים ללמידה קופסאות אחת ליד השניה, כמו למשל: תפריט ניוט אופקי, יצירת גריד של תМОנות. הפתרונות שראינו (חוץ מהפתרון האחרון) מתאימים גם למקרים כלואו.

ישנן טכניקות לימוד אפשריות נוספות, למשל, לימוד עם inline-block, inline-block וfloat עם position: absolute. גם להשתמש בגריד מוקן כמו dg.960 וdoneiy כדי לעמוד את הדף.

ולבסוף, מילה על העתיד: הסטנדרט החדש שאמור לעזור לנו לעמוד את הדף הוא box-flex התמימה ב-box-flex-ModelAttribute. מצומצמת כיוון, בין השאר מפני שהסטנדרט השתנה באופן משמעותי בשנים האחרונות. לפיכך, נראה שנמשיך להשתמש בהצפות ללמידה דפים עוד זמן רב.

המשך סיכום

- ה FLOAT מוציא את האלמנט מהרצף הנורמלי של הדף וממקם אותו בצד ימין או שמאל של הורה שלו.
- כאשר ישנו מספר אלמנטים הצפויים לאוטו לצד האלמנטים יצפו זה לצדיו של זה.
- אלמנט שצף, אף לשוליים של האלמנט ההורה. במקרה שאין הורה יצוף לשולי הדף.
- כדי למנוע מהאלמנט להיצמד לאלמנט שלצדיו ניתן להוסיף שוליים.
- כאשר מעמידים אלמנט אחד לצד השני יש לתת לכלם הצפה לאוטו כיוון ולהתאים את הרוחב שלהם כך שלכלם יש מקום.
- שימוש בערך BLOCK/INLINE מאפשר לנו להשתמש במודל הקופסא (הגדירות גובה ורוחב).
- אלמנטים INLINE BLOCK נמצאים בשורה ולכן יהיה מרווח בודד ביניהם צריך לשים לב שרוחב האלמנטים בשורה כולל גם את הרוחב הזה כך שלא יגלוש לשורה הבאה.
- כאשר אלמנט מקבל FLOAT הוא יצא מהרצף של הדף ועלול לפגוע בהגדרות עיצוב של אלמנטים אחרים לפחות פעמיים. אלמנטים אחרים יצפו סביב האלמנט ואיןם נמצאים במקומות.
- הmafpiin CLEAR מוחזרת הדף לריצף הנורמלי שלו או ריצף ימני, או ריצף שמאלי או שניהם יחד.
- אפשר להכניס DIV ריק שהחזרת העיצוב שלו משתמש בתוכנה CLEAR.

Basic Box Properties

<u>bottom</u>	Specifies the bottom position of a positioned element	2
<u>clear</u>	Specifies which sides of an element where other floating elements are not allowed	1
<u>clip</u>	Clips an absolutely positioned element	2
<u>display</u>	Specifies how a certain HTML element should be displayed	1
<u>float</u>	Specifies whether or not a box should float	1
<u>height</u>	Sets the height of an element	1
<u>left</u>	Specifies the left position of a positioned element	2
<u>overflow</u>	Specifies what happens if content overflows an element's box	2
<u>overflow-x</u>	Specifies whether or not to clip the left/right edges of the content, if it overflows the element's content area	3
<u>overflow-y</u>	Specifies whether or not to clip the top/bottom edges of the content, if it overflows the element's content area	3
<u>padding</u>	Sets all the padding properties in one declaration	1
<u>padding-bottom</u>	Sets the bottom padding of an element	1
<u>padding-left</u>	Sets the left padding of an element	1
<u>padding-right</u>	Sets the right padding of an element	1
<u>padding-top</u>	Sets the top padding of an element	1
<u>position</u>	Specifies the type of positioning method used for an element (static, relative, absolute or fixed)	2
<u>right</u>	Specifies the right position of a positioned element	2
<u>top</u>	Specifies the top position of a positioned element	2
<u>visibility</u>	Specifies whether or not an element is visible	2
<u>width</u>	Sets the width of an element	1
<u>vertical-align</u>	Sets the vertical alignment of an element	1
<u>z-index</u>	Sets the stack order of a positioned element	

attribute selectors

1. Set the background-color to "lightblue" for elements with a "target" attribute.
2. Set a border with the color "red", around elements with a "title" attribute containing the word "red".
3. Set a border with the color "red", around elements with a "title" attribute ending with the word "flower" (not flowers).
4. Set a border with the color "red", around elements with a "title" attribute containing the value "flow".
5. Set a border with the color "red", around elements with a "title" attribute starting with "red".

CSS Combinators

```
<body>
  <div>
    <p>Paragraph 1 in the div.</p>
    <p>Paragraph 2 in the div.</p>
    <span><p>Paragraph 3 in the div.</p></span> <!-- not Child but Descendant -->
  </div>
  <p>Paragraph 4. Not in a div.</p>
  <p>Paragraph 5. Not in a div.</p>
</body>
```

1. p in div - Child not Descendant

Paragraph 1 in the div.
Paragraph 2 in the div.
Paragraph 3 in the div.
Paragraph 4. Not in a div.
Paragraph 5. Not in a div.

2. p elements that are placed immediately after <div> elements (sibling)

Paragraph 1 in the div.
Paragraph 2 in the div.
Paragraph 3. Not in a div.
Paragraph 4. Not in a div.

3. p elements that are siblings of <div> elements (general sibling)

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3. Not in a div.

CSS Pseudo-classes

1. Set the background color for visited and unvisited links to "lightblue", and the background color for the hover and active link states to "yellow".
2. Change the background color, when a user hovers over p elements, with the class "highlight", to "lightblue".
3. Set the background color of <input> elements that are in focus (clicked or active), to "lightblue".

CSS Pseudo-elements

4. Set text color to red, for the first line of the <p> element. Set text color to "red", and the text size to "xx-large", for the first letter of the <p> element.

char	dec	hex
☀	9728	2600
☁	9729	2601
☽	9730	2602
☾	9731	2603
⌚	9732	2604
★	9733	2605
☆	9734	2606
⌚	9735	2607
☒	9736	2608
ଓ	9737	2609
ঃ	9739	260B
ঁ	9740	260C
ং	9741	260D
ঃ	9742	260E
ঃ	9743	260F
□	9744	2610
☑	9745	2611
☒	9746	2612
X	9747	2613
♫	9748	2614
ঔ	9749	2615
ঔ	9750	2616
ঁ	9751	2617
ঔ	9752	2618
ঔ	9753	2619
ঔ	9754	261A
ঔ	9755	261B
ঔ	9756	261C

6162264

צבעים בשפת CSS

מאפיינים רבים בשפת CSS מקבלים צבעים בהתאם ערך. כך להדגמה ניתן להגיד את צבע הרקע או צבע הגבול של אלמנט מסוים. בשפת CSS ניתן להגיד צבעים בשש דרכים שונות:

- `color name` הגדרת הצבע לפי שמות מוגדרים מראש
- `hexadecimal` הגדרת הצבע על ידי צירוף של אותיות ומספרות
- `rgb` הגדרת הצבע על ידי שלושה מספרים (אדום, ירוק וכחול)
- `rgba` דומה לשיטת ה `rgb`- רק שמספר נוסף מגדיר את השקיפות (alpha) של הצבע.
- `hsb` הגדרת הצבע על ידי שלושה מספרים (hue, saturation, lightness)
- `hsia` דומה לשיטת `hsb` רק שמספר נוסף מגדיר את השקיפות (alpha) של הצבע.

שם הצבע

בשפת CSS ישנו שמות מוגדרים מראש לרשימה ארוכה של צבעים שונים. ניתן לציין צבע על ידי שימוש במילה המייצגת את אן הצבע. לדוגמה, ניתן להשתמש במילה `blue` בשביל להגיד צבע כחול, או `orange` בשביל צבע כתום. זהה שיטה מודגמת מובבלת שבדרך כלל לא מתאימה לרוב המעצבים.

הדוגמה הבאה מראה איך ניתן להגיד את הצבעים אדום, ירוק, לבן ושחור:

```
#el1 {background-color: red;}  
#el2 {background-color: green;}  
#el3 {background-color: white;}  
#el4 {background-color: black;}
```

שיטת decimal

צבעים בשיטה זו מוגדרים על ידי צירוף של שישה תווים המכילים ספורות מ-0 עד 9 ואותיות מ-a עד f ועל כן השם, במילים אחרות, ספירה על בסיס 16). לדוגמה, הצבע הלבן בשיטה זו מוצג על ידי `FFFFFFFFFF`.

אין הבדל בין אותיות גדולות לבין קטנות (#FFFFFF) ו-#fffffff (זה אותו הצבע). אם הצבע מוגדר מתווים דומים, ניתן לנקז את שם הצבע לשולש תווים (לדוגמה FFF, #FFFFFF, #FFF-זה אותו הצבע).

הדוגמה הבאה מראה איך ניתן להגיד את הצבעים אדום, ירוק, לבן ושחור בשיטת decimal:

```
#el1 {background-color: #F00;} /* Red */  
#el2 {background-color: #06FF00;} /* Green */  
#el3 {background-color: #FFF;} /* White */  
#el4 {background-color: #000;} /* Black */
```

שיטת rgb

הו הם ראשיתות של red, green, blue. red צבעים בשיטה זו מוגדרים על ידי שילוב של שלושה מספרים בין 0 ל-255 כשהנספר הראשון מגדיר את הגוון האדום, השני מגדיר את הגוון הירוק והשלישי את הגוון הכחול של הצבע.

הדוגמה הבאה מראה איך ניתן להגדיר את הצבעים אדום, ירוק, לבן ושחור בשיטת:rgb

```
#el1 {background-color: rgb(255,0,0); } /* Red */  
#el2 {background-color: rgb(0,255,0); } /* Green */  
#el3 {background-color: rgb(255,255,255); } /* White */  
#el4 {background-color: rgb(0,0,0); } /* Black */
```

שיטת rgba

הו הם ראשיתות של red, green, blue, alpha. red צבעים בשיטה זו מוגדרים על ידי שלושה מספרים כמו בשיטת rgb אך מספר נוסף בין 0 ל-1 מגדיר את שקיפות הצבע כ-0 מגדיר שקיפות מוחלטת ו-1 מגדיר אטימות שקופה מוחלטת. הדוגמה הבאה מראה איך ניתן להגדיר את הצבעים אדום, ירוק, לבן ושחור בשיטת rgba כשלכל צבע ניתנת דרגת שקיפות שונה:

```
#el1 {background-color: rgb(255,0,0,1); } /* Red */  
#el2 {background-color: rgb(0,255,0,0.5); } /* Green */  
#el3 {background-color: rgb(255,255,255,0); } /* White */  
#el4 {background-color: rgb(0,0,0,0.75); } /* Black */
```

שיטת hsl

הו הם ראשיתות של hue, saturation, lightness.hue צבעים בשיטה זו מוגדרים על ידי שלושה מספרים. המספר הראשון מוגדר על ידי מעלות (מספר בין 0 ל-360). המספר השני (saturation) מוגדר על ידי אחד בין 0% לבין 100%. המספר השלישי (lightness) מוגדר גם הוא על ידי אחד בין 0% לבין 100%.

הדוגמה הבאה מראה איך ניתן להגדיר את הצבעים אדום, ירוק, לבן ושחור בשיטת:hsl

```
#el1 {background-color: hsl(0, 100%, 50%); } /* Red */  
#el2 {background-color: hsl(120, 100%, 50%); } /* Green */  
#el3 {background-color: hsl(120, 100%, 100%); } /* White */  
#el4 {background-color: hsl(0, 0%, 0%); } /* Black */
```

שיטת hsla

הו הם ראשיתות של hue, saturation, lightness, alpha.hue בדומה לשיטה הקודמת, צבעים אלו מוגדרים על ידי שלושה מספרים ומספר רביעי בין 0 ל-1 המגדיר את השקיפות של הצבע (בדומה לrgba-כש-0 מגדיר השקיפות מוחלטת ו-1 מגדיר אטימות מוחלטת).

הדוגמה הבאה מראה איך ניתן להגדיר את הצבעים אדום, ירוק, לבן ושחור בשיטת hsla כשלכל צבע ניתנת דרגת השקיפות שונה:

```
#el1 {background-color: hsla(0, 100%, 50%, 0.5); } /* Red */  
#el2 {background-color: hsla(120, 100%, 50%, 1); } /* Green */  
#el3 {background-color: hsla(120, 100%, 100%, 0); } /* White */  
#el4 {background-color: hsla(0, 0%, 0%, 0.25); } /* Black */
```

CSS3 Colors

CSS supports color names, hexadecimal and RGB colors.

In addition, CSS3 also introduces:

- RGBA colors
- HSL colors
- HSLA colors
- opacity

RGBA Colors

RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

An RGBA color value is specified with: `rgba(red, green, blue, alpha)`. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
rgba(255, 0, 0, 0.2);  
rgba(255, 0, 0, 0.4);  
rgba(255, 0, 0, 0.6);  
rgba(255, 0, 0, 0.8);
```

The following example defines different RGBA colors:

Example

```
#p1 {background-color: rgba(255, 0, 0, 0.3); /* red with opacity */}  
#p2 {background-color: rgba(0, 255, 0, 0.3); /* green with opacity */}  
#p3 {background-color: rgba(0, 0, 255, 0.3); /* blue with opacity */}
```

HSL Colors

HSL stands for Hue, Saturation and Lightness.

An HSL color value is specified with: `hsl(hue, saturation, lightness)`.

1. Hue is a degree on the color wheel (from 0 to 360):
 - 0 (or 360) is red
 - 120 is green
 - 240 is blue
2. Saturation is a percentage value: 100% is the full color.
3. Lightness is also a percentage; 0% is dark (black) and 100% is white.

```
hsl(0, 100%, 30%);  
hsl(0, 100%, 50%);  
hsl(0, 100%, 70%);  
hsl(0, 100%, 90%);
```

The following example defines different HSL colors:

Example

```
#p1 {background-color: hsl(120, 100%, 50%); /* green */}  
#p2 {background-color: hsl(120, 100%, 75%); /* light green */}  
#p3 {background-color: hsl(120, 100%, 25%); /* dark green */}  
#p4 {background-color: hsl(120, 60%, 70%); /* pastel green */}
```

HSLA Colors

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with: hsla(hue, saturation, lightness, alpha), where the alpha parameter defines the opacity. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
hsla(0, 100%, 30%, 0.3);  
hsla(0, 100%, 50%, 0.3);  
hsla(0, 100%, 70%, 0.3);  
hsla(0, 100%, 90%, 0.3);
```

The following example defines different HSLA colors:

Example

```
#p1 {background-color: hsla(120, 100%, 50%, 0.3);} /* green with opacity */  
#p2 {background-color: hsla(120, 100%, 75%, 0.3);} /* light green with opacity */  
/*/  
#p3 {background-color: hsla(120, 100%, 25%, 0.3);} /* dark green with opacity */  
#p4 {background-color: hsla(120, 60%, 70%, 0.3);} /* pastel green with opacity */  
*/
```

Opacity

The CSS3 opacity property sets the opacity for the whole element (both background color and text will be opaque/transparent).

The opacity property value must be a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
rgb(255, 0, 0); opacity: 0.2;  
rgb(255, 0, 0); opacity: 0.4;  
rgb(255, 0, 0); opacity: 0.6;  
rgb(255, 0, 0); opacity: 0.8;
```

Notice that the text above will also be transparent/opaque!

The following example shows different elements with opacity:

Example

```
#p1 {background-color:rgb(255,0,0);opacity:0.6;} /* red with opacity */  
#p2 {background-color:rgb(0,255,0);opacity:0.6;} /* green with opacity */  
#p3 {background-color:rgb(0,0,255);opacity:0.6;} /* blue with opacity */
```

סלקטורים של – CSS3 חלק א'

תיקן CSS3 החדש מכיל לא מעט סלקטורים שיכולים לשיער לנו להוסיף את התכונות שאנו רוצים לאלמנטים שאנו רוצים. בכך, אפשר להשתמש תמיד ב-pi-או ב-class-אבל הסלקטורים הללו יכולים להקשות לפעמים מאד. הסלקטורים של CSS3 עובדים עם כל הדרפנינים המודרניים ויעבדו גם עם אקספלורר 9. עבור דפדפני אינטראנט אקספלורר מגישה 6,7 ו-8 – יש פתרונות ליישום הסלקטורים הללו (או חלקם).

סלקטורים המבוססים על תכונות - Attributes

או תכונות הן כל התכונות שאנו מוסיף לאלמנט HTML כמו למשל ה href-שאנו מוסיף ל.a-באמצעות ה attributes-אנו יכולים לבצע בחירה מדוקית של אלמנטים.

[attr]

בחירה של כל האלמנטים שיש להם את התכונה הזו. למשל בחירת כל הפסקאות שיש להן את תכונת rel כמו למשל:

```
<p rel="whatever">HAHA</p>
```

ה CSS-הוא:

```
p[rel] {  
color: red;  
}
```

HAHA

[=attr]

בחירה של אלמנט שיש בו תכונה המכילה בדיקת המילה שאנו רוצים. למשל, כל הקישורים שיש בהם קישור ל网址.com

```
<a href="http://www.specfic-site.com">כתובת האתר כלשהו</a>
```

על מנת לבחור בהם CSS-זה קוד ה:

```
a[href="http://www.specfic-site.com"] {  
color: red;  
}
```

[*=attr]

בחירה של אלמנט שיש בו תכונה שיש בו את חלק מיליה כלשהי[תת מחוץ]. למשל כל הקישורים שיש בהם המילה google

```
<a href="http://www.google.com">קישור לגוגל</a>
```

את זה עושים עם הסלקטור הבא ב-CSS:

```
a[href*="google"] {  
    color: red;  
}
```

קישור

[^=attr]

בחירה של אלמנט שמתחל בטקסט מסוים, כמו למשלבחירה של כל הקישורים שמתחילה ב-

http (בניגוד לקישורים שמתחילים ב-ftp)

הנה ה-CSS:

```
div.testme a[href^="http:"] {  
    color: red;  
}
```

קישור אחד
 קישור שני

[\$/attr]

בחירה של תכונות עם סימן מסויימת, למשל תמונה שה src-שלה נגמר ב:.png

```
img[src$=".png"] {  
    border: 3px solid red;  
}
```

[attr~=]

הסלקטור זהה מאפשר לנו לבחור **מיליה(שליה)** בתוך תכונה שמאפשרת כמה מילים. כמו

class שיכולה להכיל כמה וכמה classים. אם אני רוצה לבחור למשל קישור מסוים שיש לו

בשם moshe, אני יכול להשתמש בסלקטור זהה.
class[a[class~="moshe"] {
 color: red;
}

זהו הדוגמא:

Link to Moshe

All CSS Pseudo Classes

סלקטורים המבוססים על מיקום בעץ

:nth-child

סלקטור nth-child: הוא סלקטור שאנו בוחרים באמצעות אלמנט ייל' או כמה אלמנטים ייל'דים. אם ניקח רשימה (אלמנט, (או אנו יכולים לבחור כל פריט ברשימה לפי מיקומו. למשל הרשימה

זה:

```
<ul id="selectme01">
<li>פריט ראשון</li>
<li>פריט שני</li>
<li>פריט שלישי</li>
<li>פריט רביעי</li>
<li>פריט חמישי</li>
</ul>
```

אם אני רוצה לסמך את היל'ד השלישי (כלומר הפריט השלישי ברשימה) אני אצטרך להכניס ל-

CSS את הדבר הבא:

```
#selectme01 li:nth-child(3) {
color:red;
}
```

והנה הדוגמא החיה

- פריט ראשון
- פריט שני
- פריט שלישי
- פריט רביעי
- פריט חמישי

אני יכול להכניס גם משוואות מורכבות לרשימה באמצעות המשטנה n . המשטנה n הוא קל למד', המשטנה פשוט נועד בין אפס לאינסוף כשבכל פעם אנו מוסיפים לו אחד. כך למשל 4 הוא התוצאה של הבן הרביעי (4 כפול אחד), הבן השמיני (4 כפול שניים), הבן העשירי (4 כפול שלוש) הבן השישה עשר (4 כפול ארבע) וכן הלאה. $1 + 2n$ הוא הבן הראשון (אפס כפול שניים ועוד אחד), הבן השלישי (אחד כפול שניים ועוד 1), הבן החמישי (שניים כפול שניים ועוד אחד) ועוד וכן הלאה.

$n=0$ ארך לא קיים אלמנט=0)

0	1	2	3	4
---	---	---	---	---

הנה דוגמא לשימוש בסלקטור $n+2$

```
#selectme02 li:nth-child(n+2) {
color:red;
}
```

- פריט ראשון
- פריט שני
- פריט שלישי
- פריט רביעי
- פריט חמישי

אפשר גם להכניס את הביטוי odd או even ולבחרו את כל הילדים הזוגיים או האי זוגיים. הנה

דוגמא טובה לשימוש בביטוי:odd

- פריט ראשון
- פריט שני
- פריט שלישי
- פריט רביעי
- פריט חמישי

וכך נראה הקוד – פשוט ביותר!

```
#selectme03 li:nth-child(odd) {
    color:red;
}
```

שנו סלקטור דומה ל nth-child והוא נקרא, nth-of-type משמשים בו באותו אופן, אך הוא מבצע את הספירה ורק ילדים מסוימים סוג. ברשימה זה די מיותר, כי ברור שכל הילדים הם כך. אך אשר אם לא נמצאים ברשימה ויש ילדים מכמה סוגים nth-of-type, הוא חשוב כיוון שהוא מבצע את הספירה אך ורק בנוגע לילדים מסוימים סוג.

nth-last-child:

nth-last-child זהה כמעט לחלוין nth-child. רק שהספרה של ה-מחילה בסוף האלמנטים ולא מההתחלת. ההבדל הוא בכך אך חשוב. ה-מחילה בעצם מהוסף. כך שאם יש לנו רשימה של חמישה פריטים והסלקטור שלו הוא:

```
#selectme04 li:nth-last-child(2) {
    color:red;
}
```

אז התוצאה תהיה:

- פריט ראשון
- פריט שני
- פריט שלישי
- פריט רביעי
- פריט חמישי

זה מובע מכך שהספרה נעשית מהפריט האחרון ברשימה ולא הראשון. הבדל דק וחשוב מהסלקטור הקודם. גם כאן אפשר להשתמש במסוות של ח' רק חשוב לציין שהספרה נעשית מהפריט האחרון לפני הראשון ולא ההפן.

גם ל child-nth-last-child יש סלקטור דומה שנראה כך nth-last-of-type והוא מבצע את הספרה בנוגע לילדים מסוימים סוג. ברשימה זה די מיותר כי כל הילדים הם מסווג. או אך כאשר יש לנו ילדים בניהם ואנו רוצים לבצע את הספרה רק לאלמנטים מסוים מסווג nth-last-of-type, יעשה את הספרה רק בנוגע לאלמנטים שנבחרו.

first-child

סלקטור זה מאפשר לנו לבחור את האלמנט שהוא הילד הראשון. בואו ונניח שיש לנו אלמנט

פסקה (P) ויש לו שלושה ילדים, קישור span, וקישור נוסף:

```
<p id="myparagraph01">
    <a href="nana.co.il">קיישור ראשון</a>
    <span>זהו ספרן צביב</span>
    <a href="walla.com">קיישור שני</a>
</p>
```

אם אני רוצה לבחור את הילד הראשון ברשימה (אם הוא, אָז אַנְיַ אֲשַׁתְּמַשׁ בְּסַלְקָטוֹר):

```
#myparagraph01 a:first-child {
color:red;
}
```

וזו התוצאה:

קישור ראשון

זהו ספאן חביב

קישור שני

יש לנו גם את סלקטור `first-of-type` שבוחר את הילד הראשון מסוים, כך למשל, אם אני רוצה לבחור את ה `span`-הראשון שיש ברשימה (לא משנה שהוא היחיד, הוא עדין הראשון), איני

אשתחמש ב-`:first-of-type`

```
#myparagraph02 span:first-of-type {
color:red;
}
```

וכך זה נראה:

קישור ראשון

זהו ספאן חביב

קישור שני

last-child

כמו שיש לנו `first-child` יש לנו גם את `last-child`. הוא פשוט בוחר את הילד האחרון. למשל, אם

אני ארצה לבחור את הקישור האחרון, אינן אשתחמש ב `:last-child`-באופן הבא:

```
#myparagraph03 a:last-child {
color:red;
}
```

וכך זה נראה במצבות:

קישור ראשון

זהו ספאן חביב

קישור שני

הוא סלקטור דומה בעקרון, אבל הוא בוחר אלמנט האחרון מסוים (ולא last-of-type)

דווקא בילד האחרון).

only-child

only-child בוחר אלמנט אם הוא בן יחיד. כך למשל אם יש לי פסקה כלשהי ולה יש רק קישור אחד, הוא יבחר בו, אבל אם יש יותר מ קישור אחד בפסקה, לא יבחר אף קישור.

```
<p id="myparagraph04">
<img/>
<a href="nrg.co.il"> קישור בודד </a>
</p>
```

בואו ונסתכל על הפסקה הבאה:

אם אני משתמש בסלקטור זהה

```
#myparagraph04 a:only-child {
color:red;
}
```

از הקישור הבודד שלנו יבחר. הנה הדוגמא:

קישור בודד

גם כאן יש לנו את `:only-of-type` שבודק אם ישILD אחד מאותו הסוג, ומתעלם מבנים אחרים שאינם מהסוג המפורט.

empty

empty הוא סלקטור שבוחר אלמנטים ריקים, כך למשל אם יש לנו שלוש פסקאות בדף:

```
<p></p>
<p> </p>
<p>Hello, World!</p>
```

אם משתמשים בו:

```
p:empty {
margin: 0px;
}
```

از לפסקה הראשונה בלבד לא יהיה שול'ם.

enabled\disabled

אלו הם סלקטורים שבוחרים את ה`התקנים`-ים שהם `enabled` או `disabled`. בואו ונניח שיש לנו טופס שיש בו שדה אחד שהוא `enabled` ושדה אחר שהוא `disabled`:

```
<form id="myform01">
<input type="text" value="I am Enabled" />
<input type="text" value="I am Disabled" disabled="disabled" />
</form>
```

אם אני משתמש בסלקטורים `enabled` ו `disabled`-אני אוכל לצבע אותם בשני צבעים שונים:

```
#myform01 input:enabled {
color: red;
}
#myform01 input:disabled {
color: blue;
}
```

:checked

פואודו קלאס זה תקף ל checkbox – radio buttons שימו לב לדוגמא הבאה:



אות הדבר המופלא הזה עשייתי באמצעות הפואודו קלאס הבא:

```
#myform02 input:checked {  
outline: red 3px solid;  
}
```

נסו לשחק עם הבחירה ותראו איך זה משתנה!

target

פואודו קלאס זה מסיע לנו לבחור אלמנטים שאליהם קופצנו באמצעות target זה יהיה מאד ברור עם דוגמא, זוכרים את הטופס myform01 שהשתמשתי בו במעלה העמוד? הנה הקישור שמקשר אליו:

קישור המקשר ל-target-בשם `myform01:target`

אם תלחצו על הקישור זהה, תקפו אל, תקפו אל, #myform01 אבל myform01 יקבל גם רקע אדום (סוג של הייליט). נסו!

אות הדבר הזה עשייתי עמו:

```
#myform01:target{  
background-color: red;  
}
```

not

אוחהו אול' הפואודו קלאס הכי שימושי שיש. נניח ויש לנו שניים ים עם class שהוא אחד מהם יש class בשם fakeDiv

```
<div class="myDiv">  
אני דיב חביב  
</div>
```

```
<div class="fakeDiv myDiv">  
אני דיב חביב אך קצת מזוייף  
</div>
```

כיצד אבחר את הדיב החביב ולא את זה שיש בו גם class של fakeDiv? not שינו

לב לסלקטור הבא:

```
.myDiv:not(.fakeDiv) {  
color: red;  
}
```

root

root הוא פואודו קלאס מיותר למדוי לעדטי שבחור את האלמנט הראשון של הדף (בדרכן כל שימושים בו כר' html).

```
:root {  
declaration block  
}
```

פואדו אלמנטים ב-CSS3

במאמר הקודם למדנו על **סילקטורים** ופואדו **קלאסים ב-CSS3**. במאמר זהה אנו לומדים על פואדו אלמנטים. ב-CSS3 יש לנו גם פואדו אלמנטים, כלומר כלו שבוררים חלק מהאלמנט. לפי התקן צריך להיות :: לפני האלמנטים האלה.

first-line

זה פואדו אלמנט שבאמצעותו אמ' בוחרים את השורה הראשונה. למשל, להציג את השורה

הראשונה:

```
#myText::first-line {
    font-weight: bold;
}
```

first-letter

זה פואדו אלמנט שמאפשר לנו לבחור את האות הראשונה. למשל:

```
#myText2::first-letter {
    font-weight: bold;
    font-size: 20px;
}
```

selected

זה פואדו-אלמנט מוד' מגניב ששולט על הבחירה בטקסט שעשו לו. הדבר כי טוב

```
#selectMe::selection {
    background-color: red;
}
```

חשוב לשימר לב שהתמייה ל select-היא לא מלאה ונכון לכתיבת שורות אלו, עדין אין אותה

בפיירפוקס, אם רצים להשתמש בה בפיירפוקס צריכים להשתמש ב:

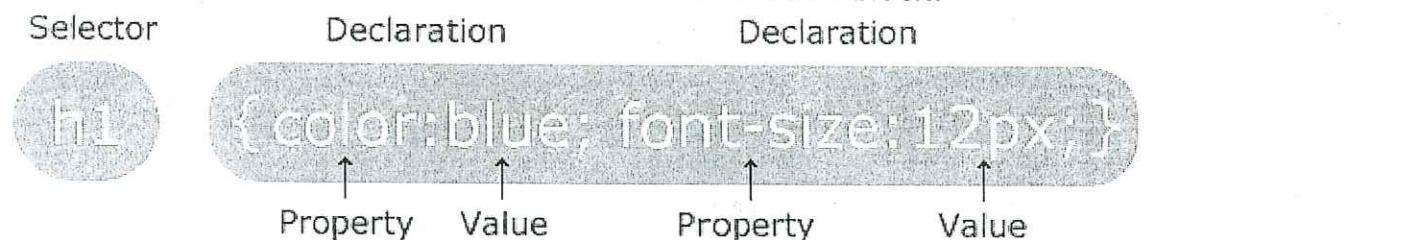
```
#selectMe::-moz-selection {
    background-color: red;
}
```

CSS Syntax and Selectors

Previous Next

CSS Syntax

A CSS rule-set consists of a selector and a declaration block:



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

In the following example all `<p>` elements will be center-aligned, with a red text color:

Example

```
p {  
    color: red;  
    text-align: center;  
}
```

CSS Selectors

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

The element Selector

The element selector selects elements based on the element name.

You can select all `<p>` elements on a page like this (in this case, all `<p>` elements will be center-aligned, with a red text color):

Example

```
p {  
    text-align: center;  
    color: red;  
}
```

The id Selector

The id selector uses the id attribute of an HTML element to select a specific element. The id of an element should be unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

The style rule below will be applied to the HTML element with id="para1":

Example

```
#para1 {  
    text-align: center;  
    color: red;  
}
```

Try it Yourself

Note: An id name cannot start with a number!

The class Selector

The class selector selects elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the name of the class.

In the example below, all HTML elements with class="center" will be red and center-aligned:

Example

```
.center {  
    text-align: center;  
    color: red;  
}
```

Try it Yourself

You can also specify that only specific HTML elements should be affected by a class.

In the example below, only `<p>` elements with class="center" will be center-aligned:

Example

```
p.center {  
    text-align: center;  
    color: red;  
}
```

Try it Yourself

HTML elements can also refer to more than one class.

In the example below, the `<p>` element will be styled according to class="center" and to class="large":

Example

```
<p class="center large">This paragraph refers to two classes.</p>
```

Try it Yourself

Note: A class name cannot start with a number!

Grouping Selectors

If you have elements with the same style definitions, like this:

```
h1 {  
    text-align: center;  
    color: red;  
}  
  
h2 {  
    text-align: center;  
    color: red;  
}  
  
p {  
    text-align: center;  
    color: red;  
}
```

It will be better to group the selectors, to minimize the code.

To group selectors, separate each selector with a comma.

In the example below we have grouped the selectors from the code above:

Example

```
h1, h2, p {  
    text-align: center;  
    color: red;  
}
```

CSS Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

A CSS comment starts with /* and ends with */. Comments can also span multiple lines:

Example

```
p {  
    color: red;  
    /* This is a single-line comment */  
    text-align: center;  
}  
  
/* This is  
a multi-line  
comment */
```

CSS Background

[« Previous](#)

[Next Chapter »](#)

CSS background properties are used to define the background effects of an element.

CSS properties used for background effects:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

Background Color

The `background-color` property specifies the background color of an element.

The background color of a page is set like this:

Example

```
body {  
    background-color: #b0c4de;  
}
```

[Try it yourself >](#)

With CSS, a color is most often specified by:

- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"
- a valid color name - like "red"

Look at [CSS Color Values](#) for a complete list of possible color values.

In the example below, the `<h1>`, `<p>`, and `<div>` elements have different background colors:

Example

```
h1 {  
    background-color: #6495ed;  
}
```

```
p {  
    background-color: #e0ffff;  
}  
  
div {  
    background-color: #b0c4de;  
}
```

[Try it yourself >](#)

Background Image

The `background-image` property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element.

The background image for a page can be set like this:

Example

```
body {  
    background-image: url("paper.gif");  
}
```

[Try it yourself >](#)

Below is an example of a bad combination of text and background image. The text is hardly readable:

Example

```
body {  
    background-image: url("bgdesert.jpg");  
}
```

[Try it yourself >](#)

Background Image - Repeat Horizontally or Vertically

By default, the `background-image` property repeats an image both horizontally and vertically.

Some images should be repeated only horizontally or vertically, or they will look strange, like this:

Example

```
body {  
    background-image: url("gradient_bg.png");  
}
```

[Try it yourself >](#)

If the image is repeated only horizontally (`background-repeat: repeat-x;`), the background will look better:

Example

```
body {  
    background-image: url("gradient_bg.png");  
    background-repeat: repeat-x;  
}
```

[Try it yourself >](#)



Note: To repeat an image vertically set `background-repeat: repeat-y;`

Background Image - Set position and no-repeat



Note: When using a background image, use an image that does not disturb the text.

Showing the background image only once is also specified by the `background-repeat` property:

Example

```
body {  
    background-image: url("img_tree.png");  
    background-repeat: no-repeat;  
}
```

[Try it yourself >](#)

In the example above, the background image is shown in the same place as the text. We want to change the position of the image, so that it does not disturb the text too much.

The position of the image is specified by the `background-position` property:

Example

```
body {  
    background-image: url("img_tree.png");  
    background-repeat: no-repeat;  
    background-position: right top;  
}
```

[Try it yourself >](#)

Background Image - Fixed position

To specify that the background image should be fixed (will not scroll with the rest of the page), use the `background-attachment` property:

Example

```
body {  
    background-image: url("img_tree.png");  
    background-repeat: no-repeat;  
    background-position: right top;  
    background-attachment: fixed;  
}
```

[Try it yourself >](#)

Background - Shorthand property

To shorten the code, it is also possible to specify all the background properties in one single property. This is called a shorthand property.

The shorthand property for background is `background`:

Example

```
body {  
    background: #ffffff url("img_tree.png") no-repeat right top;  
}
```

[Try it yourself >](#)

When using the shorthand property the order of the property values is:

- `background-color`
- `background-image`
- `background-repeat`

- background-attachment
- background-position

It does not matter if one of the property values is missing, as long as the other ones are in this order.

Test Yourself with Exercises!

[Exercise 1](#) > [Exercise 2](#) > [Exercise 3](#) > [Exercise 4](#) > [Exercise 5](#)

All CSS Background Properties

Property	Description
<u>background</u>	Sets all the background properties in one declaration
<u>background-attachment</u>	Sets whether a background image is fixed or scrolls with the rest of the page
<u>background-color</u>	Sets the background color of an element
<u>background-image</u>	Sets the background image for an element
<u>background-position</u>	Sets the starting position of a background image
<u>background-repeat</u>	Sets how a background image will be repeated

CSS Text

TEXT FORMATTING

This text is styled with some of the text formatting properties. The heading uses the text-align, text-transform, and color properties. The paragraph is indented, aligned, and the space between characters is specified. The underline is removed from this colored "[Try it Yourself](#)" link.

Text Color

The color property is used to set the color of the text.

With CSS, a color is most often specified by:

- a color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

Look at [CSS Color Values](#) for a complete list of possible color values.

The default text color for a page is defined in the body selector.

Example

```
body {  
    color: blue;  
}
```

```
h1 {  
    color: green;  
}
```

Note: For W3C compliant CSS: If you define the color property, you must also define the background-color

Text Alignment

The text-align property is used to set the horizontal alignment of a text.

A text can be left or right aligned, centered, or justified.

The following example shows center aligned, and left and right aligned text (left alignment is default if text direction is left-to-right, and right alignment is default if text direction is right-to-left):

Example

```
h1 {  
    text-align: center;  
}  
  
h2 {  
    text-align: left;  
}  
  
h3 {  
    text-align: right;  
}
```

When the `text-align` property is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers):

Example

```
div {  
    text-align: justify;  
}
```

Text Decoration

The `text-decoration` property is used to set or remove decorations from text.

The value `text-decoration: none;` is often used to remove underlines from links:

Example

```
a {  
    text-decoration: none;  
}
```

The other `text-decoration` values are used to decorate text:

Example

```
h1 {  
    text-decoration: overline;  
}  
  
h2 {  
    text-decoration: line-through;  
}  
  
h3 {  
    text-decoration: underline;  
}
```

Note: It is not recommended to underline text that is not a link, as this often confuses the reader.

Text Transformation

The `text-transform` property is used to specify uppercase and lowercase letters in a text.

It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:

Example

```
p.uppercase {  
    text-transform: uppercase;  
}  
  
p.lowercase {  
    text-transform: lowercase;  
}  
  
p.capitalize {  
    text-transform: capitalize;  
}
```

Text Indentation

The `text-indent` property is used to specify the indentation of the first line of a text:

Example

```
p {  
    text-indent: 50px;  
}
```

[Try it Yourself >](#)

Letter Spacing

The `letter-spacing` property is used to specify the space between the characters in a text.

The following example demonstrates how to increase or decrease the space between characters:

Example

```
h1 {  
    letter-spacing: 3px;  
}  
  
h2 {  
    letter-spacing: -3px;  
}
```

[Try it Yourself >](#)

Line Height

The line-height property is used to specify the space between lines:

Example

```
p.small {  
    line-height: 0.8;  
}  
  
p.big {  
    line-height: 1.8;  
}
```

Text Direction

The direction property is used to change the text direction of an element:

Example

```
div {  
    direction: rtl;  
}
```

Word Spacing

The word-spacing property is used to specify the space between the words in a text.

The following example demonstrates how to increase or decrease the space between words:

Example

```
h1 {  
    word-spacing: 10px;  
}  
  
h2 {  
    word-spacing: -5px;  
}
```

More Examples

Disable text wrapping inside an element

This example demonstrates how to disable text wrapping inside an element.

Vertical alignment of an image

This example demonstrates how to set the vertical align of an image in a text.

Add shadow to text

This example demonstrates how to add shadow to text.

All CSS Text Properties

Property	Description
<u>color</u>	Sets the color of text
<u>direction</u>	Specifies the text direction/writing direction
<u>letter-spacing</u>	Increases or decreases the space between characters in a text
<u>line-height</u>	Sets the line height
<u>text-align</u>	Specifies the horizontal alignment of text
<u>text-decoration</u>	Specifies the decoration added to text
<u>text-indent</u>	Specifies the indentation of the first line in a text-block
<u>text-shadow</u>	Specifies the shadow effect added to text
<u>text-transform</u>	Controls the capitalization of text
<u>unicode-bidi</u>	Used together with the <u>direction</u> property to set or return whether the text should be overridden to support multiple languages in the same document
<u>vertical-align</u>	Sets the vertical alignment of an element
<u>white-space</u>	Specifies how white-space inside an element is handled
<u>word-spacing</u>	Increases or decreases the space between words in a text

CSS Fonts

The CSS font properties define the font family, boldness, size, and the style of a text.

Difference Between Serif and Sans-serif Fonts



CSS Font Families

In CSS, there are two types of font family names:

- **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
- **font family** - a specific font family (like "Times New Roman" or "Arial")

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

Note: On computer screens, sans-serif fonts are considered easier to read than serif fonts.

Font Family

The font family of a text is set with the **font-family** property.

The **font-family** property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.

Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Note: If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman".

More than one font family is specified in a comma-separated list:

Example

```
p {  
    font-family: "Times New Roman", Times, serif;  
}
```

For commonly used font combinations, look at our [Web Safe Font Combinations](#).

Font Style

The `font-style` property is mostly used to specify italic text.

This property has three values:

- `normal` - The text is shown normally
- `italic` - The text is shown in italics
- `oblique` - The text is "leaning" (oblique is very similar to italic, but less supported)

Example

```
p.normal {  
    font-style: normal;  
}  
  
p.italic {  
    font-style: italic;  
}  
  
p.oblique {  
    font-style: oblique;  
}
```

Font Size

The `font-size` property sets the size of the text.

Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.

Always use the proper HTML tags, like `<h1>` - `<h6>` for headings and `<p>` for paragraphs.

The `font-size` value can be an absolute, or relative size.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Note: If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

Set Font Size With Pixels

Setting the text size with pixels gives you full control over the text size:

Example

```
h1 {  
    font-size: 40px;  
}  
  
h2 {  
    font-size: 30px;  
}  
  
p {  
    font-size: 14px;  
}
```

Tip: If you use pixels, you can still use the zoom tool to resize the entire page.

Set Font Size With Em

To allow users to resize the text (in the browser menu), many developers use em instead of pixels.

The em size unit is recommended by the W3C.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

The size can be calculated from pixels to em using this formula: $\text{pixels}/16=\text{em}$

Example

```
h1 {  
    font-size: 2.5em; /* 40px/16=2.5em */  
}  
  
h2 {  
    font-size: 1.875em; /* 30px/16=1.875em */  
}  
  
p {  
    font-size: 0.875em; /* 14px/16=0.875em */  
}
```

In the example above, the text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers.

Unfortunately, there is still a problem with older versions of IE. The text becomes larger than it should when made larger, and smaller than it should when made smaller.

Use a Combination of Percent and Em

The solution that works in all browsers, is to set a default font-size in percent for the <body> element:

Example

```
body {  
    font-size: 100%;  
}  
  
h1 {  
    font-size: 2.5em;  
}  
  
h2 {  
    font-size: 1.875em;  
}  
  
p {  
    font-size: 0.875em;  
}
```

Our code now works great! It shows the same text size in all browsers, and allows all browsers to zoom or resize the text!

Font Weight

The `font-weight` property specifies the weight of a font:

Example

```
p.normal {  
    font-weight: normal;  
}  
  
p.thick {  
    font-weight: bold;  
}
```

Font Variant

The `font-variant` property specifies whether or not a text should be displayed in a small-caps font.

In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

Example

```
p.normal {  
    font-variant: normal;  
}  
  
p.small {  
    font-variant: small-caps;  
}
```

More Examples

All the font properties in one declaration

This example demonstrates how to use the shorthand property for setting all of the font properties in one declaration.

All CSS Font Properties

Property	Description
<u>font</u>	Sets all the font properties in one declaration
<u>font-family</u>	Specifies the font family for text
<u>font-size</u>	Specifies the font size of text
<u>font-style</u>	Specifies the font style for text
<u>font-variant</u>	Specifies whether or not a text should be displayed in a small-caps font
<u>font-weight</u>	Specifies the weight of a font

CSS Borders

CSS Border Properties

The CSS border properties allow you to specify the style, width, and color of an element's border.

This element has a groove border that is 15px wide and green.

Border Style

The `border-style` property specifies what kind of border to display.

The following values are allowed:

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the `border-color` value
- ridge - Defines a 3D ridged border. The effect depends on the `border-color` value
- inset - Defines a 3D inset border. The effect depends on the `border-color` value
- outset - Defines a 3D outset border. The effect depends on the `border-color` value
- none - Defines no border
- hidden - Defines a hidden border

The `border-style` property can have from one to four values (for the top border, right border, bottom border, and the left border).

Example

```
p.dotted {border-style: dotted;}  
p.dashed {border-style: dashed;}  
p.solid {border-style: solid;}  
p.double {border-style: double;}  
p.groove {border-style: groove;}  
p.ridge {border-style: ridge;}  
p.inset {border-style: inset;}  
p.outset {border-style: outset;}  
p.none {border-style: none;}  
p.hidden {border-style: hidden;}  
p.mix {border-style: dotted dashed solid double;}
```

Result:

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border. The effect depends on the `border-color` value.

A ridge border. The effect depends on the `border-color` value.

An inset border. The effect depends on the `border-color` value.

An outset border. The effect depends on the `border-color` value.

No border.

A hidden border.

A mixed border.



Note: None of the OTHER CSS border properties described below will have ANY effect unless:

Border Width

The `border-width` property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.

The `border-width` property can have from one to four values (for the top border, right border, bottom border, and the left border).

Example

```
p.one {  
    border-style: solid;  
    border-width: 5px;  
}  
p.two {  
    border-style: solid;  
    border-width: medium;  
}  
p.three {  
    border-style: solid;  
    border-width: 2px 10px 4px 20px;  
}
```

Border Color

The `border-color` property is used to set the color of the four borders.

The color can be set by:

- name - specify a color name, like "red"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- Hex - specify a hex value, like "#ff0000"
- transparent

The `border-color` property can have from one to four values (for the top border, right border, bottom border, and the left border).

If `border-color` is not set, it inherits the color of the element.

Example

```
p.one {  
    border-style: solid;  
    border-color: red;  
}  
p.two {  
    border-style: solid;  
    border-color: green;  
}  
p.three {  
    border-style: solid;  
    border-color: red green blue yellow;  
}
```

Border - Individual Sides

From the examples above you have seen that it is possible to specify a different border for each side.

In CSS, there is also properties for specifying each of the borders (top, right, bottom, and left):

Example

```
p {  
    border-top-style: dotted;  
    border-right-style: solid;  
    border-bottom-style: dotted;  
    border-left-style: solid;  
}
```

The example above gives the same result as this:

Example

```
p {  
    border-style: dotted solid;  
}
```

So, here is how it works:

If the `border-style` property has four values:

- **border-style: dotted solid double dashed;**
 - top border is dotted
 - right border is solid
 - bottom border is double
 - left border is dashed

If the border-style property has three values:

- **border-style: dotted solid double;**
 - top border is dotted
 - right and left borders are solid
 - bottom border is double

If the border-style property has two values:

- **border-style: dotted solid;**
 - top and bottom borders are dotted
 - right and left borders are solid

If the border-style property has one value:

- **border-style: dotted;**
 - all four borders are dotted

The border-style property is used in the example above. However, it also works with border-width and border-color.

Border - Shorthand Property

As you can see from the examples above, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property.

The border property is a shorthand property for the following individual border properties:

- border-width
- border-style (required)
- border-color

Example

```
p {  
    border: 5px solid red;  
}
```

CSS Lists

Previous

Next

-
- 1. Coffee
 - 2. Tea
 - 3. Coca Cola

- Coffee
- Tea
- Coca Cola

HTML Lists and CSS List Properties

In HTML, there are two main types of lists:

- unordered lists (``) - the list items are marked with bullets
- ordered lists (``) - the list items are marked with numbers or letters

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker
- Add background colors to lists and list items

Different List Item Markers

The `list-style-type` property specifies the type of list item marker.

The following example shows some of the available list item markers:

Example

```
ul.a {  
    list-style-type: circle;  
}  
  
ul.b {  
    list-style-type: square;  
}  
  
ol.c {  
    list-style-type: upper-roman;  
}
```

```
ol.d {  
    list-style-type: lower-alpha;  
}
```

[Try it Yourself »](#)

Note: Some of the values are for unordered lists, and some for ordered lists.

An Image as The List Item Marker

The list-style-image property specifies an image as the list item marker:

Example

```
ul {  
    list-style-image: url('sqpurple.gif');  
}
```

[Try it Yourself »](#)

Position The List Item Markers

The list-style-position property specifies whether the list-item markers should appear inside or outside the content flow:

Example

```
ul {  
    list-style-position: inside;  
}
```

[Try it Yourself »](#)

List - Shorthand property

The list-style property is a shorthand property. It is used to set all the list properties in one declaration:

Example

```
ul {  
    list-style: square inside url("sqpurple.gif");  
}
```

[Try it Yourself »](#)

When using the shorthand property, the order of the property values are:

- list-style-type (if a list-style-image is specified, the value of this property will be displayed if the image for some reason cannot be displayed)
- list-style-position (specifies whether the list-item markers should appear inside or outside the content flow)
- list-style-image (specifies an image as the list item marker)

If one of the property values above are missing, the default value for the missing property will be inserted, if any.

Styling List With Colors

We can also style lists with colors, to make them look a little more interesting.

Anything added to the `` or `` tag, affects the entire list, while properties added to the `` tag will affect the individual list items:

Example

```
ol {  
    background: #ff9999;  
    padding: 20px;  
}  
  
ul {  
    background: #3399ff;  
    padding: 20px;  
}  
  
ol li {  
    background: #ffe5e5;  
    padding: 5px;  
    margin-left: 35px;  
}  
  
ul li {  
    background: #cce5ff;  
    margin: 5px;  
}
```

Result:

1. Coffee
 2. Tea
 3. Coca Cola
- Coffee
 - Tea
 - Coca Cola

More Examples

Customized list with a red left border

This example demonstrates how to create a list with a red left border.

Full-width bordered list

This example demonstrates how to create a bordered list without bullets.

All the different list-item markers for lists

This example demonstrates all the different list-item markers in CSS.

Test Yourself with Exercises!

[Exercise 1](#) » [Exercise 2](#) » [Exercise 3](#) »

All CSS List Properties

Property	Description
<u>list-style</u>	Sets all the properties for a list in one declaration
<u>list-style-image</u>	Specifies an image as the list-item marker
<u>list-style-position</u>	Specifies if the list-item markers should appear inside or outside the content flow
<u>list-style-type</u>	Specifies the type of list-item marker

CSS Links

With CSS, links can be styled in different ways.

Styling Links

Links can be styled with any CSS property (e.g. color, font-family, background, etc.).

Example

```
a {  
    color: hotpink;  
}
```

In addition, links can be styled differently depending on what **state** they are in.

The four links states are:

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

Example

```
/* unvisited link */  
a:link {  
    color: red;  
}  
  
/* visited link */  
a:visited {  
    color: green;  
}  
  
/* mouse over link */  
a:hover {  
    color: hotpink;  
}  
  
/* selected link */  
a:active {  
    color: blue;  
}
```

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited

- a:active MUST come after a:hover

Text Decoration

The text-decoration property is mostly used to remove underlines from links:

Example

```
a:link {  
    text-decoration: none;  
}  
  
a:visited {  
    text-decoration: none;  
}  
  
a:hover {  
    text-decoration: underline;  
}  
  
a:active {  
    text-decoration: underline;  
}
```

Background Color

The background-color property can be used to specify a background color for links:

Example

```
a:link {  
    background-color: yellow;  
}  
  
a:visited {  
    background-color: cyan;  
}  
  
a:hover {  
    background-color: lightgreen;  
}  
  
a:active {  
    background-color: hotpink;  
}
```

Advanced - Link Buttons

This example demonstrates a more advanced example where we combine several CSS properties to display links as boxes/buttons:

Example

```
a:link, a:visited {  
    background-color: #f44336;  
    color: white;  
    padding: 14px 25px;  
    text-align: center;  
    text-decoration: none;  
    display: inline-block;  
}  
  
a:hover, a:active {  
    background-color: red;  
}
```

[Try it Yourself >](#)

More Examples

Add different styles to hyperlinks

This example demonstrates how to add other styles to hyperlinks.

Advanced - Create a link button with borders

Another example of how to create link boxes/buttons.

CSS3 Colors

CSS supports color names, hexadecimal and RGB colors.

In addition, CSS3 also introduces:

- RGBA colors
- HSL colors
- HSLA colors
- opacity

RGBA Colors

RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

An RGBA color value is specified with: `rgba(red, green, blue, alpha)`. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
rgba(255, 0, 0, 0.2);  
rgba(255, 0, 0, 0.4);  
rgba(255, 0, 0, 0.6);  
rgba(255, 0, 0, 0.8);
```

The following example defines different RGBA colors:

Example

```
#p1 {background-color: rgba(255, 0, 0, 0.3); /* red with opacity */}  
#p2 {background-color: rgba(0, 255, 0, 0.3); /* green with opacity */}  
#p3 {background-color: rgba(0, 0, 255, 0.3); /* blue with opacity */}
```

HSL Colors

HSL stands for Hue, Saturation and Lightness.

An HSL color value is specified with: `hsl(hue, saturation, lightness)`.

1. Hue is a degree on the color wheel (from 0 to 360):
 - 0 (or 360) is red
 - 120 is green
 - 240 is blue
2. Saturation is a percentage value: 100% is the full color.
3. Lightness is also a percentage; 0% is dark (black) and 100% is white.

```
hsl(0, 100%, 30%);  
hsl(0, 100%, 50%);  
hsl(0, 100%, 70%);  
hsl(0, 100%, 90%);
```

The following example defines different HSL colors:

Example

```
#p1 {background-color: hsl(120, 100%, 50%); /* green */}  
#p2 {background-color: hsl(120, 100%, 75%); /* light green */}  
#p3 {background-color: hsl(120, 100%, 25%); /* dark green */}  
#p4 {background-color: hsl(120, 60%, 70%); /* pastel green */}
```

HSLA Colors

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with: hsla(hue, saturation, lightness, alpha), where the alpha parameter defines the opacity. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
hsla(0, 100%, 30%, 0.3);  
hsla(0, 100%, 50%, 0.3);  
hsla(0, 100%, 70%, 0.3);  
hsla(0, 100%, 90%, 0.3);
```

The following example defines different HSLA colors:

Example

```
#p1 {background-color: hsla(120, 100%, 50%, 0.3);} /* green with opacity */  
#p2 {background-color: hsla(120, 100%, 75%, 0.3);} /* light green with opacity */  
#p3 {background-color: hsla(120, 100%, 25%, 0.3);} /* dark green with opacity */  
#p4 {background-color: hsla(120, 60%, 70%, 0.3);} /* pastel green with opacity */
```

Opacity

The CSS3 opacity property sets the opacity for the whole element (both background color and text will be opaque/transparent).

The opacity property value must be a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
rgb(255, 0, 0), opacity:0.2;  
rgb(255, 0, 0), opacity:0.4;  
rgb(255, 0, 0), opacity:0.6;  
rgb(255, 0, 0), opacity:0.8;
```

Notice that the text above will also be transparent/opaque!

The following example shows different elements with opacity:

Example

```
#p1 {background-color:rgb(255,0,0);opacity:0.6;} /* red with opacity */  
#p2 {background-color:rgb(0,255,0);opacity:0.6;} /* green with opacity */  
#p3 {background-color:rgb(0,0,255);opacity:0.6;} /* blue with opacity */
```

CSS Combinators

A combinator is something that explains the relationship between the selectors.

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS3:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

Descendant Selector

The descendant selector matches all elements that are descendants of a specified element.

The following example selects all `<p>` elements inside `<div>` elements:

Example

```
div p {  
    background-color: yellow;  
}
```

Child Selector

The child selector selects all elements that are the immediate children of a specified element.

The following example selects all `<p>` elements that are immediate children of a `<div>` element:

Example

```
div > p {  
    background-color: yellow;  
}
```

Adjacent Sibling Selector

The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

The following example selects all `<p>` elements that are placed immediately after `<div>` elements:

Example

```
div + p {  
    background-color: yellow;  
}
```

General Sibling Selector

The general sibling selector selects all elements that are siblings of a specified element.

The following example selects all `<p>` elements that are siblings of `<div>` elements:

Example

```
div ~ p {  
    background-color: yellow;  
}
```

CSS Pseudo-classes

What are Pseudo-classes?

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently

Syntax

The syntax of pseudo-classes:

```
selector:pseudo-class {  
    property:value;  
}
```

Anchor Pseudo-classes

Links can be displayed in different ways:

Example

```
/* unvisited link */  
a:link {  
    color: #FF0000;  
}  
  
/* visited link */  
a:visited {  
    color: #00FF00;  
}  
  
/* mouse over link */  
a:hover {  
    color: #FF00FF;  
}  
  
/* selected link */  
a:active {  
    color: #0000FF;  
}
```



Note: `a:hover` MUST come after `a:link` and `a:visited` in the CSS definition in order to be effective! Pseudo-class names are not case-sensitive.

Pseudo-classes and CSS Classes

Pseudo-classes can be combined with CSS classes:

Example

```
a.highlight:hover {  
    color: #ff0000;  
}
```

When you hover over the link in the example, it will change color.

CSS - The :first-child Pseudo-class

The `:first-child` pseudo-class matches a specified element that is the first child of another element.

Match the first `<p>` element

In the following example, the selector matches any `<p>` element that is the first child of any element:

Example

```
p:first-child {  
    color: blue;  
}
```

Match the first `<i>` element in all `<p>` elements

In the following example, the selector matches the first `<i>` element in all `<p>` elements:

Example

```
p i:first-child {  
    color: blue;  
}
```

Match all `<i>` elements in all first child `<p>` elements

In the following example, the selector matches all `<i>` elements in `<p>` elements that are the first child of another element:

Example

```
p:first-child i {  
    color: blue;  
}
```

CSS - The :lang Pseudo-class

The `:lang` pseudo-class allows you to define special rules for different languages.

In the example below, `:lang` defines the quotation marks for `<q>` elements with `lang="no"`:

Example

```
<html>  
<head>  
<style>  
q:lang(no) {
```

```

    quotes: "~" "~";
}
</style>
</head>

<body>
<p>Some text <q lang="no">A quote in a paragraph</q> Some text.</p>
</body>
</html>

```

All CSS Pseudo Classes

Selector	Example	Example description
<u>:active</u>	a:active	Selects the active link
<u>:checked</u>	input:checked	Selects every checked <input> element
<u>:disabled</u>	input:disabled	Selects every disabled <input> element
<u>:empty</u>	p:empty	Selects every <p> element that has no children
<u>:enabled</u>	input:enabled	Selects every enabled <input> element
<u>:first-child</u>	p:first-child	Selects every <p> elements that is the first child of its parent
<u>:first-of-type</u>	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
<u>:focus</u>	input:focus	Selects the <input> element that has focus
<u>:hover</u>	a:hover	Selects links on mouse over
<u>:in-range</u>	input:in-range	Selects <input> elements with a value within a specified range
<u>:invalid</u>	input:invalid	Selects all <input> elements with an invalid value
<u>:lang(<i>language</i>)</u>	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"

<u>:last-child</u>	p:last-child	Selects every <p> elements that is the last child of its parent
<u>:last-of-type</u>	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
<u>:link</u>	a:link	Selects all unvisited links
<u>:not(selector)</u>	:not(p)	Selects every element that is not a <p> element
<u>:nth-child(n)</u>	p:nth-child(2)	Selects every <p> element that is the second child of its parent
<u>:nth-last-child(n)</u>	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
<u>:nth-last-of-type(n)</u>	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
<u>:nth-of-type(n)</u>	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
<u>:only-of-type</u>	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
<u>:only-child</u>	p:only-child	Selects every <p> element that is the only child of its parent
<u>:optional</u>	input:optional	Selects <input> elements with no "required" attribute
<u>:out-of-range</u>	input:out-of-range	Selects <input> elements with a value outside a specified range
<u>:read-only</u>	input:read-only	Selects <input> elements with a "readonly" attribute specified

<u>:read-write</u>	input:read-write	Selects <input> elements with no "readonly" attribute specified
<u>:required</u>	input:required	Selects <input> elements with a "required" attribute specified
<u>:root</u>	root	Selects the document's root element
<u>:target</u>	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
<u>:valid</u>	input:valid	Selects all <input> elements with a valid value
<u>:visited</u>	a:visited	Selects all visited links

All CSS Pseudo Elements

Selector	Example	Example description
<u>::after</u>	p::after	Insert content after every <p> element
<u>::before</u>	p::before	Insert content before every <p> element
<u>::first-letter</u>	p::first-letter	Selects the first letter of every <p> element
<u>::first-line</u>	p::first-line	Selects the first line of every <p> element
<u>::selection</u>	p::selection	Selects the portion of an element that is selected by a user

CSS Pseudo-elements

What are Pseudo-Elements?

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

Syntax

The syntax of pseudo-elements:

```
selector::pseudo-element {  
    property:value;  
}
```

Notice the double colon notation - ::first-line versus :first-line

The double colon replaced the single-colon notation for pseudo-elements in CSS3. This was an attempt from W3C to distinguish between **pseudo-classes** and **pseudo-elements**.

The single-colon syntax was used for both pseudo-classes and pseudo-elements in CSS2 and CSS1.

For backward compatibility, the single-colon syntax is acceptable for CSS2 and CSS1 pseudo-elements.

The ::first-line Pseudo-element

The ::first-line pseudo-element is used to add a special style to the first line of a text.

The following example formats the first line of the text in all <p> elements:

Example

```
p::first-line {  
    color: #ff0000;  
    font-variant: small-caps;  
}
```

Note: The ::first-line pseudo-element can only be applied to block-level elements.

The following properties apply to the ::first-line pseudo-element:

- font properties
- color properties
- background properties
- word-spacing

- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

The ::first-letter Pseudo-element

The ::first-letter pseudo-element is used to add a special style to the first letter of a text.

The following example formats the first letter of the text in all `<p>` elements:

Example

```
p::first-letter {
  color: #ff0000;
  font-size: xx-large;
}
```

Note: The ::first-letter pseudo-element can only be applied to block-level elements.

The following properties apply to the ::first-letter pseudo- element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

Pseudo-elements and CSS Classes

Pseudo-elements can be combined with CSS classes:

Example

```
p.intro::first-letter {
  color: #ff0000;
  font-size: 200%;
}
```

The example above will display the first letter of paragraphs with class="intro", in red and in a larger size.

Multiple Pseudo-elements

Several pseudo-elements can also be combined.

In the following example, the first letter of a paragraph will be red, in an xx-large font size. The rest of the first line will be blue, and in small-caps. The rest of the paragraph will be the default font size and color:

Example

```
p::first-letter {  
    color: #ff0000;  
    font-size: xx-large;  
}  
  
p::first-line {  
    color: #0000ff;  
    font-variant: small-caps;  
}
```

CSS - The ::before Pseudo-element

The ::before pseudo-element can be used to insert some content before the content of an element.

The following example inserts an image before the content of each `<h1>` element:

Example

```
h1::before {  
    content: url(smiley.gif);  
}
```

CSS - The ::after Pseudo-element

The ::after pseudo-element can be used to insert some content after the content of an element.

The following example inserts an image after the content of each `<h1>` element:

Example

```
h1::after {  
    content: url(smiley.gif);  
}
```

CSS - The ::selection Pseudo-element

The ::selection pseudo-element matches the portion of an element that is selected by a user.

The following CSS properties can be applied to ::selection: color, background, cursor, and outline.

The following example makes the selected text red on a yellow background:

Example

```
::selection {  
    color: red;  
    background: yellow;  
}
```

All CSS Pseudo Elements

Selector	Example	Example description
<u>::after</u>	p::after	Insert something after the content of each <p> element
<u>::before</u>	p::before	Insert something before the content of each <p> element
<u>::first-letter</u>	p::first-letter	Selects the first letter of each <p> element
<u>::first-line</u>	p::first-line	Selects the first line of each <p> element
<u>::selection</u>	p::selection	Selects the portion of an element that is selected by a user

All CSS Pseudo Classes

Selector	Example	Example description
<u>:active</u>	a:active	Selects the active link
<u>:checked</u>	input:checked	Selects every checked <input> element
<u>:disabled</u>	input:disabled	Selects every disabled <input> element
<u>:empty</u>	p:empty	Selects every <p> element that has no children
<u>:enabled</u>	input:enabled	Selects every enabled <input> element
<u>:first-child</u>	p:first-child	Selects every <p> elements that is the first child of its parent
<u>:first-of-type</u>	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
<u>:focus</u>	input:focus	Selects the <input> element that has focus
<u>:hover</u>	a:hover	Selects links on mouse over
<u>:in-range</u>	input:in-range	Selects <input> elements with a value within a specified range
<u>:invalid</u>	input:invalid	Selects all <input> elements with an invalid value

<u>:lang(/language)</u>	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
<u>:last-child</u>	p:last-child	Selects every <p> elements that is the last child of its parent
<u>:last-of-type</u>	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
<u>:link</u>	a:link	Selects all unvisited links
<u>:not(selector)</u>	:not(p)	Selects every element that is not a <p> element
<u>:nth-child(n)</u>	p:nth-child(2)	Selects every <p> element that is the second child of its parent
<u>:nth-last-child(n)</u>	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
<u>:nth-last-of-type(n)</u>	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
<u>:nth-of-type(n)</u>	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
<u>:only-of-type</u>	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
<u>:only-child</u>	p:only-child	Selects every <p> element that is the only child of its parent
<u>:optional</u>	input:optional	Selects <input> elements with no "required" attribute
<u>:out-of-range</u>	input:out-of-range	Selects <input> elements with a value outside a specified range
<u>:read-only</u>	input:read-only	Selects <input> elements with a "readonly" attribute specified
<u>:read-write</u>	input:read-write	Selects <input> elements with no "readonly" attribute
<u>:required</u>	input:required	Selects <input> elements with a "required" attribute specified
<u>:root</u>	root	Selects the document's root element
<u>:target</u>	#news:target	Selects the current active #news element (clicked on)
<u>:valid</u>	input:valid	Selects all <input> elements with a valid value
<u>:visited</u>	a:visited	Selects all visited links

CSS Selectors

In CSS, selectors are patterns used to select the element(s) you want to style.

Use our [CSS Selector Tester](#) to demonstrate the different selectors.

The "CSS" column indicates in which CSS version the property is defined (CSS1, CSS2, or CSS3).

Selector	Example	Example description	CSS
<u>.class</u>	.intro	Selects all elements with class="intro"	1
<u>#id</u>	#firstname	Selects the element with id="firstname"	1
<u>*</u>	*	Selects all elements	2
<u>element</u>	p	Selects all <p> elements	1
<u>element,element</u>	div, p	Selects all <div> elements and all <p> elements	1
<u>element element</u>	div p	Selects all <p> elements inside <div> elements	1
<u>element>element</u>	div > p	Selects all <p> elements where the parent is a <div> element	2
<u>element+element</u>	div + p	Selects all <p> elements that are placed immediately after <div> elements	2
<u>element1~element2</u>	p ~ ul	Selects every element that are preceded by a <p> element	3
<u>[attribute]</u>	[target]	Selects all elements with a target attribute	2
<u>[attribute=value]</u>	[target=_blank]	Selects all elements with target="_blank"	2
<u>[attribute~=value]</u>	[title~=flower]	Selects all elements with a title attribute containing the word "flower"	2
<u>[attribute =value]</u>	[lang =en]	Selects all elements with a lang attribute value starting with "en"	2
<u>[attribute^=value]</u>	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"	3

<u>[attribute\$=value]</u>	a[href\$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"	3
<u>[attribute*=value]</u>	a[href*="w3schools"]	Selects every <a> element whose href attribute value contains the substring "w3schools"	3
<u>:active</u>	a:active	Selects the active link	1
<u>::after</u>	p::after	Insert content after every <p> element	2
<u>::before</u>	p::before	Insert content before the content of every <p> element	2
<u>:checked</u>	input:checked	Selects every checked <input> element	3
<u>:disabled</u>	input:disabled	Selects every disabled <input> element	3
<u>:empty</u>	p:empty	Selects every <p> element that has no children (including text nodes)	3
<u>:enabled</u>	input:enabled	Selects every enabled <input> element	3
<u>:first-child</u>	p:first-child	Selects every <p> element that is the first child of its parent	2
<u>::first-letter</u>	p::first-letter	Selects the first letter of every <p> element	1
<u>::first-line</u>	p::first-line	Selects the first line of every <p> element	1
<u>:first-of-type</u>	p:first-of-type	Selects every <p> element that is the first <p> element of its parent	3
<u>:focus</u>	input:focus	Selects the input element which has focus	2
<u>:hover</u>	a:hover	Selects links on mouse over	1
<u>:in-range</u>	input:in-range	Selects input elements with a value within a specified range	3
<u>:invalid</u>	input:invalid	Selects all input elements with an invalid value	3

<u>:lang(<i>language</i>)</u>	p:lang(it)	Selects every <p> element with a lang attribute equal to "it" (Italian)	2
<u>:last-child</u>	p:last-child	Selects every <p> element that is the last child of its parent	3
<u>:last-of-type</u>	p:last-of-type	Selects every <p> element that is the last <p> element of its parent	3
<u>:link</u>	a:link	Selects all unvisited links	1
<u>:not(<i>selector</i>)</u>	:not(p)	Selects every element that is not a <p> element	3
<u>:nth-child(<i>n</i>)</u>	p:nth-child(2)	Selects every <p> element that is the second child of its parent	3
<u>:nth-last-child(<i>n</i>)</u>	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child	3
<u>:nth-last-of-type(<i>n</i>)</u>	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child	3
<u>:nth-of-type(<i>n</i>)</u>	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent	3
<u>:only-of-type</u>	p:only-of-type	Selects every <p> element that is the only <p> element of its parent	3
<u>:only-child</u>	p:only-child	Selects every <p> element that is the only child of its parent	3
<u>:optional</u>	input:optional	Selects input elements with no "required" attribute	3
<u>:out-of-range</u>	input:out-of-range	Selects input elements with a value outside a specified range	3
<u>:read-only</u>	input:read-only	Selects input elements with the "readonly" attribute specified	3
<u>:read-write</u>	input:read-write	Selects input elements with the "readonly" attribute NOT specified	3

<u>:required</u>	input:required	Selects input elements with the "required" attribute specified	3
<u>:root</u>	:root	Selects the document's root element	3
<u>::selection</u>	::selection	Selects the portion of an element that is selected by a user	
<u>:target</u>	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)	3
<u>:valid</u>	input:valid	Selects all input elements with a valid value	3
<u>:visited</u>	a:visited	Selects all visited links	1