

التحريك عبر CSS



محمد بغات

أكاديمية
حسوب 

التحريك عبر CSS

تَعَلَّم كيف تضيف حركات إلى موقعك الإلكتروني

تأليف

دونشان هاتشنسون

ترجمة

محمد بغات

تحرير

جميل بيلوني

محمد بغات



محمد بغات، حاصل على شهادة الدراسات العامة في الرياضيات، مهتم بالبرمجة والسيو، مترجم بموسوعة وأكاديمية حسوب، ومدون بمدونة حسوب. ترجمت عدة كتب في مجال البرمجة، وشاركت في ترجمة توثيقات Kotlin و Sass و Ruby و Cordova، ألقت كتاب «أساطير وحقائق عن السيو والتسويق الرقمي»، وألقت أحد عشر بحثًا ودليلاً تدريبيًا، خصوصًا في مجال الإدارة والتدريب والجمعيات المدنية.

يمكنك التواصل معي عبر حسابي البريدي:

meedawi.med@gmail.com

هذا الكتاب

أنتج هذا الكتاب برعاية شركة **حسوب** وأكاديمية **حسوب**.



أكاديمية حسوب

تهدف أكاديمية حسوب إلى توفير مقالات ودروس عالية الجودة حول مجالات مُختلفة وبلغة عربية فصيحة.

تقدم أكاديمية حسوب دورات شاملة بجودة عالية عن تعلم البرمجة بأحدث تقنياتها تعتمد على التطبيق العملي، مما يؤهل الطالب لدخول سوق العمل بثقة. تتكامل الأكاديمية مع موسوعة حسوب، التي توفر توثيقًا عربيًا شاملًا مدعمًا بالأمثلة للغات البرمجة.

باب المساهمة في الأكاديمية مفتوح لكل من يرى في نفسه القدرة على توفير مقالات أو كتب أو مسارات عالية الجودة.

Academy.hsoub.com



في مهمة لتطوير العالم العربي

شركة حسوب

تهدف حسوب لتطوير الويب العربي وخدمات الإنترنت عن طريق توفير حلول عملية وسهلة الاستخدام لتحديات مختلفة تواجه المستخدمين في العالم العربي. تشجع حسوب الشباب العربي للدخول إلى سوق العمل عن بعد بتوفيرها منصات عربية للعمل عن بعد، مستقل وخمسات؛ إضافةً إلى موقع بعيد، وكما أنها توفر خدمات للنقاشات الهادفة في حسوب I/O وخدمة رفع الصور عبر موقع صور.

يعمل في حسوب فريق شاب وشغوف من مختلف الدول العربية. ويمكن معرفة المزيد عن شركة حسوب والخدمات التي تقدمها بزيارة موقعها.

[Hsoub.com](https://hsoub.com)

جدول المحتويات

13.....تقديم

15.....بداية الرحلة

16.....1. عن المؤلف

17.....2. بنية الكتاب

18.....3. مساعدة ودعم

18.....4. هل تحتاج إلى إنعاش ذاكرتك بخصوص CSS؟

19.....5. الصور المتحركة والأمثلة الحية

19.....6. قسم التمارين

20.....ما هي استخدامات الحركات؟

21.....1. أبلغ من الكلمات

21.....2. ما المقصود بالتحريك في مجال تطوير المواقع؟

24.....3. مع القوة العظيمة، تأتي المسؤولية

25.....4. مصدر إلهام

26.....5. خلاصة الفصل

26.....6. تمرين

28.....تجهيز بيئة العمل لإنشاء الحركات

29.....1. التطوير في المتصفح

2. التطوير المحلي.....30
- ا. الخيار الأبسط: إنشاء ملفي HTML و CSS.....30
- ب. أدوات إضافية.....30
- ج. أداة البناء Gulp.....31
3. خلاصة الفصل.....32
4. تمرين.....32

مدخل إلى الانتقالات: التنقل بين الحالات. 33

1. الانتقالات.....34
2. خصائص الانتقال.....35
3. خلاصة الفصل.....37
4. تمرين.....37

مدخل إلى الحركات: مفهوم الحركة.....38

1. التحريك في المتصفح.....39
2. أمثلة.....40
3. الانتقال أم الحركة؟.....41
4. تمرين.....42

الانتقالات عمليًا.....43

1. الانتقالات.....44
2. مثال: تطبيق انتقال على زر.....45
3. البادئات وتوافق المتصفحات.....47
4. تمرين.....47

خصيات الانتقالات.....49

1. الصياغة المختصرة مقابل الصياغة المطوّلة.....50

- 51.....transition-property الخاصية ا.
- 51.....transition-duration الخاصية ب.
- 51.....transition-delay الخاصية ج.
- 52.....transition-timing-function الخاصية د.
- 52.....الأشياء التي لا يُطبَّق الانتقال عليها.
- 54.....تمرين

55.....دوال التوقيت

- 57.....التسارع الخطي (Linear)
- 58.....التسارع المتباطئ عند البداية (Ease-in)
- 58.....التسارع المتباطئ عند النهاية (Ease-out)
- 59.....التسارع المتباطئ عند البداية والنهاية
- 59.....التسارع المُخصَّص (منحني Cubic-bezier)
- 61.....الانتقال الخطوي (Steps)
- 63.....تمرين

64.....الانتقالات المتعددة

- 65.....المثال الأول: زر فني
- 67.....المثال الثاني: كشف الخلفية
- 69.....تطبيق انتقالات متعددة على عنصر واحد
- 70.....تمرين

71.....الانتقالات عبر JavaScript

- 72.....إضافة أو إزالة الأصناف
- 73.....التحكم في الانتقالات عبر JavaScript
- 74.....خلاصة الفصل
- 75.....تمرين

76.....الحركات عمليًا

- 77.....1. علاقة تكاملية
- 78.....2. الخاصية animation
- 78.....3. الإطارات المفتاحية
- 83.....4. البادئات
- 83.....5. تمرين

84.....خاصيات الحركة

- 85.....1. الخاصية animation-delay
- 85.....2. الخاصية animation-direction
- 86.....3. الخاصية animation-duration
- 86.....4. الخاصية animation-fill-mode
- 86.....5. الخاصية animation-iteration-count
- 87.....6. الخاصية animation-name
- 87.....7. الخاصية animation-play-state
- 87.....8. الخاصية animation-timing-function
- 88.....9. استخدام دوال التوقيت ضمن الإطارات المفتاحية
- 89.....10. تمرين

90.....الإطارات المفتاحية عمليًا

- 91.....1. أساسيات لابدّ منها
- 92.....2. مثال: تأثير اهتزاز الزر Save
- 96.....3. تمرين

97.....الحركات المتعددة المتزامنة

- 98.....1. إشارة المرور

2. مراجع أخرى.....103

3. تمرين.....103

104..... خلاصة ما تعلمناه عن الحركات

1. تمرين: إشارات المرور.....105

2. موجز: الحركات.....105

أ. الإطارات المفتاحية.....106

3. تجميع الحركة.....107

4. تمرين.....107

108..... رواية القصص عبر الحركات

1. الصور الرئيسية.....109

2. مثال: تمرير الخلفية.....110

أ. الجزء الأول: تحريك الخلفية.....111

ب. الجزء الثاني: إضافة انتقال الحومان.....113

ج. الجزء الثالث: إضافة رسالة.....114

3. خلاصة الفصل.....116

4. تمرين.....116

117..... حرب النجوم!

1. transform: ليست من خاصيات الحركات.....118

2. الخاصية transform ودوالها.....119

3. SVG و HTML و CSS.....119

4. تحريك الكلمتين Star و Wars.....120

5. لنجعلها ثلاثية الأبعاد.....122

6. تحريك الشعر The Force Awakens.....123

7. تمرين.....125

إظهار المحتوى أثناء التمرير.....126

1. المكتبة Wow.js.....127
- ا. استخدام Wow.js.....128
- ب. إضافة أصناف wow.....129
2. الإخفاء والعرض.....129
3. استخدام Animate.css.....130
4. استخدام Modernizr.....130
5. تمرين.....131

سهولة الوصول.....132

1. التأكد من سهولة الوصول إلى المحتوى.....133
2. إتاحة التحكم.....134
3. إتاحة مدخلات بديلة.....134
4. الارتباك.....134
5. لا تزعجني!.....135
6. تسهيل الوصول للجميع.....136
7. تمرين.....136

نهاية الرحلة.....137

1. ملخص التحريك في CSS.....138
2. أدوات مساعدة لإنشاء الحركات.....138
- ا. المكتبة Animate.css.....138
- ب. المكتبة Hover.css.....138
3. أدوات أخرى.....138
- ا. منصة GSAP.....139
- ب. المكتبة Snabbt.js.....139

- 139.....ج. المكتبة CSS Animate
- 139.....د. موقع Cubic-bezier.com
- 140.....4. ماذا بعد؟!.....

تقديم

لا يخفى على أي مطور ويب خصوصًا مطوري واجهة المستخدم الأمامية (front end developers) أهمية إضفاء بعض الحيوية على المواقع التي يصممونها عبر الحركات، إذ هنالك عدة فوائد يمكن تحصيلها من إضافة الحركات إلى صفحات الموقع أهمها الابتعاد عن السكون الممل، ولفت الانتباه، وتحسين التواصل مع الزائر أو المستخدم وغيرها؛ لذلك، أصبحت الحركات أمرًا جوهريًا في مواقع الويب لا يمكن الاستغناء عنها.

جاء هذا الكتاب لشرح مفهوم الحركة وكيفية تحريك العناصر باستخدام CSS فقط بدءًا من الحركات البسيطة وحتى الحركات المعقدة المتقدمة بالإضافة إلى التطرُّق إلى مناقشة مسألة متى يجب إضافة الحركات ومتى يجب الابتعاد عنها.

هذا الكتاب مترجم عن كتاب «[CSS Animation 101](#)» لصاحبه Donovan Hutchinson، ونأمل أن يكون إضافةً نافعَةً للمكتبة العربيَّة وأن يفيد القارئ العربي في الإلمام بموضوع التحريك عبر CSS.

هذا الكتاب مرخص بموجب رخصة المشاع الإبداعي Creative Commons «[نسب المُصنَّف](#)»

- غير تجاري - الترخيص بالمثل 4.0»

(Attribution-NonCommercial-ShareAlike 4.0 - CC BY-NC-SA 4.0).

أكاديمية حسوب

2019/9/13

بداية الرحلة

«أخبرني وسأنسى، علمني وسأذكّر، أشركني وسأتعلم.» — بنجامين فرانكلين

مرحبًا بك في كتاب «التحريك عبر CSS»، وشكرًا لاختيارك له.

أنا سعيد لأنك اخترت تعلم التحريك عبر CSS. هذا الكتاب هو مقدمة خفيفة وممتعة للموضوع، وآمل أن تستفيد منه أقصى استفادة. ستتعرف فيه على الانتقالات (transitions) والحركات (animations) في CSS، وستفهم مبادئ التحريك في CSS فهمًا جيدًا، بالإضافة إلى الأدوات اللازمة لإنشاء وتجربة الحركات في مشاريعك.

يجمع هذا الكتاب بين الأمثلة النظرية والعملية. ستتعلم فيه كيفية إعداد بيئة العمل خاصتك، وستطالع الكثير من الأمثلة عن الحركات في مشوارنا.

1. عن المؤلف

أنا دونفان، سأرافك في رحلتك في هذا الكتاب!

لقد أمضيت جزءًا كبيرًا من العقد الماضي في كتابة مقالات حول CSS ومواضيع أخرى. كما أنني عملت في مجال تصميم وتطوير المواقع منذ أواخر التسعينيات. مؤخرًا، صرت أكتب مقالات لأجل Smashing Magazine و Net Magazine و Tuts+ و Adobe Inspire وغيرها. كما أنشر مقالات عن مواضيع متنوعة على موقع Hop.ie، وفي هذا العام، ألّفت دروسًا على موقع CSSAnimation.rocks حول تقنيات التحريك المتقدمة في المتصفحات.

في النهار، أعمل مصمماً ومطوّر واجهات أماميّة، وأحب أن أدمج بين مبادئ تجربة المستخدم (UX) والحركات في التصميم. وفي المساء، أكتب التدوينات، وأحاول مواكبة ما يحدث في عالم تصميم المواقع.

هذا الكتاب هو مُقدّمة لموضوع التحريك في CSS، ولكننا سنغطي الكثير من المواد الأخرى في الطريق. الهدف هو فهم الخاصيتين transition و animation في CSS، وكيفية عملهما، وتجربتهما في أمثلة عمليّة.

بعد قراءة هذا الكتاب كاملاً، ستكون قادراً على إنشاء حركات مُتقدّمة وإضافتها إلى مشاريعك. أرجو لك رحلة ممتعة!

2. بنية الكتاب

سنغطي في هذا الكتاب الأمور التالية:

- أولاً ما هي الحركات؟ سنحاول فهم سبب اللجوء إلى إضافة الحركات. سنقدّم أيضاً الخاصيتين transition و animation، إضافةً إلى بعض مصادر الإلهام.
- ثم: سنُحدّث بتفصيلٍ عن الخاصية transition. ونتعلّم كيفية استخدامها لإنشاء انتقال بالإضافة إلى التعرّف على الخاصيّات التي تُمكننا من التحكّم في الحركة.
- بعد ذلك: سنُركّز على الخاصية animation، ونتعلّم كيفية إنشاء الإطارات المفتاحيّة التي تتيح لنا فعل أشياء تتجاوز إمكانيّات الانتقالات البسيطة.

- أخيرًا: سنجمع جميع الأجزاء التي تعلمناها في الفصول السابقة مع بعضها بعضًا. سندرس بعض الأمثلة المتقدّمة التي تستخدم كلا الخاصيّتين (animation و transition)، ونتعلّم كيفية جعل عملنا سهل الوصول (accessible)، ونشارك بعض مصادر CSS المفيدة التي يمكنك تطبيقها في مشاريعك، وأدوات JavaScript التي يمكن استخدامها لإنشاء تأثيرات أكثر تقدمًا.

3. مساعدة ودعم

إن كان لديك أيّ سؤال حول التحريك في CSS أو حول مجال آخر في برمجة الويب، فيمكنك أن تطرحه في قسم **الأسئلة البرمجية** في أكاديمية حسوب.

4. هل تحتاج إلى إنعاش ذاكرتك بخصوص CSS؟

إن لم تكن على دراية بلغة CSS، فقد يكون من المفيد أن تقتطع بعض الوقت للتعرف على مفاهيم هذه اللغة. ليس عليك أن تكون خبيرًا في CSS. إن كنت تعرف معنى «خاصية» (property)، فأنت جاهزٌ إذًا.

هذه بعض الأدوات والمراجع التي قد تجدها مفيدة:

• HTML

• CSS

• JavaScript

أنصحك بالاطلاع على **توثيق CSS** في موسوعة حسوب والرجوع إليه للاستزادة أو التذكُّر أو كشف أي غموض، إذ يشرح أغلب خاصيّات CSS شرحًا وافيًا مع أمثلة عمليّة عليها.

5. الصور المتحركة والأمثلة الحية

أحاول توفير مثال حي قابل للتشغيل مع كل مثال كامل، أرجو منك الضغط على الرابط للانتقال إلى صفحة التجريب لترى الشيفرة الكاملة للمثال أمامك وهي تُنفَّذ.

للأسف لا تدعم صيغة PDF الصور المتحركة دعمًا كاملًا، لذا قد تجد أن الصور المتحركة المذكور أنها تتحرك ساكنةً، فأرجو منك حينئذٍ أن تنقر على الصورة لكي تفتح لك في المتصفح وهي تتحرك كما ينبغي لها.

6. قسم التمارين

ستلاحظ أنّ كل الفصول تنتهي بتمارين. هذه التمارين اختيارية، ولكن أنصحك بالاطلاع عليها ومحاولة حلها، فستساعدك على ترسيخ ما تعلّمته. ستجد في كل أقسام التمارين اقتراحًا لمفهوم أو فكرة بحاجةٍ إلى تجريبها أو التفكير فيها. امنحها بعض الوقت، وستجد أنّ فهمك للتحريك قد تحسّن.

جاهز؟ فلنبداً رحلتنا لتعلم الحركات والتحريك في CSS!

ما هي استخدامات الحركات؟

"تتمحور الحركات حول خلق شعور بالحياة" — براد بيرد

قبل الدخول في الجانب الفني من التحريك عبر CSS، سنناقش سبب إنشاء الحركات في المقام الأول.

1. أبلغ من الكلمات

يمكن أن تُوصِل الحركة المعلومات بكفاءة، ويمكن استخدامها لجذب الانتباه، ولكن في جميع الحالات، فالهدف النهائي هو التواصل.

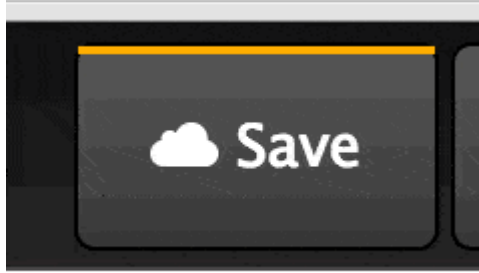
إدراج حركة في التصميمات يجعلها أكثر كفاءةً في إيصال المعلومة. الصور المتحركة أبلغ من الكلمات الملفوظة أو المكتوبة.

الحركة المُعبّرة والملائمة يمكن أن تضيف جاذبيّةً إلى تصاميمنا، و تعطي مصداقيّةً لعمَلنا، ذلك لأننا اعتدنا رؤية الحركة طوال الوقت في الواقع والطبيعة وكل مكان. لذلك، فإدخال الحركة إلى عملنا يجعله أكثر حيويّةً.

أصبحت الحركات مع التحسّن المستمر في دعم المتصفحات لها خيارًا قابلاً للتطبيق أكثر من أي وقت مضى. من نواح كثيرة، التحريك قد يكون بنفس أهمية الخطوات التي نستخدمها، والتخطيطات التي ننشئها.

2. ما المقصود بالتحريك في مجال تطوير المواقع؟

للحركات فائدتان رئيسيتان: توصيل المعلومات، وجذب الانتباه. يمكننا إيجاد عدّة طرائق لاستخدام هاتين الميزتين أثناء تصميم المواقع (المثال الحي).

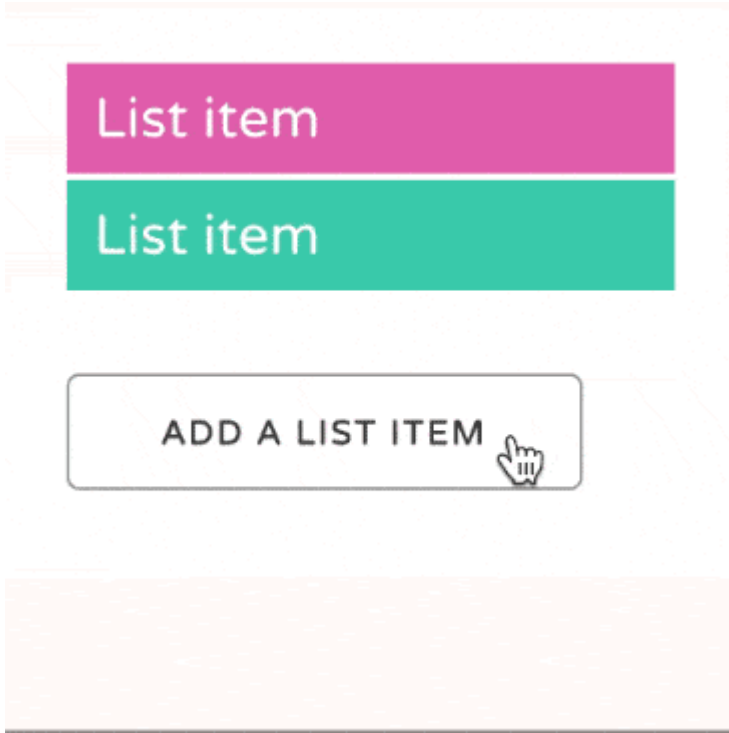


يمكن أن يكون التحريك معبّرًا، كما يحدث عندما يتململ زر الحفظ في CodePen ليذكّرنا
بضرورة حفظ عملنا:

أعيننا ماهرةً جدًّا في رصد الحركة؛ لذلك، فإنَّ إضافة حركات بسيطة إلى عناصر الصفحة يمكن
أن يضيفي عليها الحيوية ويلفت الانتباه لها.

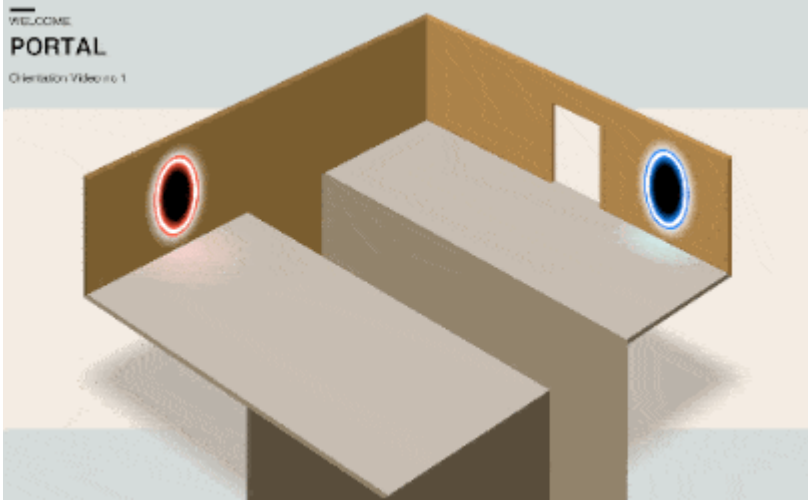
يمكننا أيضًا استخدام الحركات في عرض محتوى الصفحة (انظر هذه المقالة في أكاديمية

حسوب لتشرح لك المثال بالتفصيل):



يمكننا عن طريق تحريك المعلومات على الصفحة أن نعرض للقراء معلومة إضافية قد لا يمكن إيصالها بالمحتوى الساكن. يلفت التحريك الانتباه إلى المحتوى الجديد المضاف، ويرسم سياق تلك المعلومة الجديدة. فسيظهر المحتوى الجديد فجأة عند عدم تطبيق حركة عليه، وقد يظنُّ القارئ أنَّه كان موجودًا طوال الوقت.

يمكننا استخدام الحركات لرواية قصةٍ ما (تحريك Portal):



ما سبق مقتبسٌ من فيديو تعليمي للعبة Portal. كتابة القصة لا يحتاج دائمًا إلى أن يكون مرويًّا بالنصوص المكتوبة، إذ يمكننا إضافة حركةٍ معبّرةٍ، مثل إظهار تغييرات البيانات في المخطط بطريقةٍ يمكن للبيانات نفسها أن تروي قصة عبر الحركة.

3. مع القوة العظيمة، تأتي المسؤولية

يفتح التحريك أمامنا آفاقًا كثيرةً، إلا أنّ وجود الكثير من الأشياء المتحركة على الصفحة قد يشتت الانتباه. من الأفضل تجنّب الحركات عندما لا يكون إضافتها ضروريًّا، إذ حاول أن يكون تأثير كلِّ حركةٍ تضيفها تأثيرًا إيجابيًا.

قد يعني ذلك الاكتفاء بتحريك عنصر صغير فقط على صفحتك فالأفضل ما قلّ ودلّ كما قيل.

رغم كل شيء، إذا كنت ترغب في إنشاء تأثيرٍ بصريٍّ جذابٍ عبر إضافة الكثير من الحركات، فيمكنك ذلك، لكن احرص على الابتعاد عن المحتوى الذي تريد من المشاهدين أن يركّزوا عليه. يمكنك فعل ذلك عبر جعل الحركة تحدث مرةً واحدةً فقط، بدلاً من تكرارها باستمرار، أو بإيقاف الحركة عندما يبدأ الزائر أو المستخدم في تمرير الصفحة.

4. مصدر إلهام

التحريك له تاريخ طويل وغني. كتبت مؤخراً منشوراً حول مبادئ التحريك على الويب بعنوان **مبادئ التحريك في صفحات الويب باستخدام CSS**. المبادئ مستمدة من كتاب ديزني **Illusion of Life: Disney Animation (1981)**.

إذا كنت ترغب في التعمق في موضوع التحريك والحركات، فابحث عن مقاطع الفيديو التي تخص **Animator's Survival Kit**. موقع يوتيوب مليء بمصادر الإلهام والأفكار.

للحصول على أمثلة متميِّزة وإبداعية، خصّص بعض الوقت لتصفح موقع **Hover States**. يعرض هذا الموقع جميع أنواع الأمثلة المبدعة للتحريك على الشبكة. ستجد موقع **Dribbble.com** مفيداً أيضاً.

على سبيل المثال، إليك مثالاً رائعًا من Dribbble يعرض مبادئ التصميم المادي الخاصة بجوجل (Google's Material Design principles). البحث عن "animation" أو «حركة» أو «حركات» هو أحد الخيارات الجيدة للعثور على أفكار ملهمة من شتى بقاع الويب. أتتحقق كذلك بانتظام من مستجدات موقع CodePen، فهو مصدر رائع لاستلهام أفكار رائعة عن الحركات.

5. خلاصة الفصل

- التحريك تقنية نافعة ومفيدة للغاية.
- إذا استُخدم استخدامًا صحيحًا، يمكن أن يكون أداة مفيدة في التصميم.
- استخدمه لجذب الانتباه أو توصيل المعلومات.
- لا تبالغ فيه.
- إذا كنت تريد أن تتميز، فالتحريك قد يساعدك على ذلك.

6. تمرين

فكّر في عملك ومشاريعك، وكيف يمكن للحركات أن تساعدك.

يتحمس البعض ويبدؤون بتحريك كل شيء موجود في صفحة الموقع، لكن حاول البحث عن طرائق للاستفادة من الحركات بطريقة تساعد الزائر على فهم المحتوى فهمًا أفضل. هل تريد

أن توصل إلى زائري صفحتك شعورًا بالحركة والحيويّة؟ هل هناك تغيير مفاجئ في صفحتك يحدث بسرعة دون أن يُلحَظ وتري أنّ بالإمكان عرضه عرضًا أفضل عبر التحريك السلس؟

أخيرًا، ألقِ نظرةً على مواقع مثل [Hover States](#) و [Little Big Details](#) و [Dribbble](#).
فهذه المواقع ستساعدك كلما تحيّرت وأغلق عليك.

تجهيز بيئة العمل لإنشاء الحركات

2

«الناس لا يتعلمون المشي باتباع القواعد. بل يتعلمونه بالممارسة، والسقوط ثم النهوض» — ريتشارد برانسون

سنتعلم في هذه الفصل كيفية إنشاء حركات عبر CSS (CSS animations) ومشاهدتها في المتصفح. ولكن قبل الاسترسال في كتابة الشيفرة، من الأفضل أن نحدّد سير العمل أولاً. هناك طريقتان للتطوير: التطوير في المتصفح (online)، والتطوير المحلي (offline).

1. التطوير في المتصفح

إنّ أبسط طريقة للقيام بالتجارب الصغيرة هي التطوير عبر الإنترنت. الموقع الذي استخدمه غالبًا هو CodePen. موقع JS Fiddle يُعد خيارًا ممتازًا أيضًا.

في بقية هذا الفصل، سأستخدم CodePen لكتابة الأمثلة، لذا فمن المهم أن تكون على دراية بطريقة عمله.

CodePen هو منصة برمجية تسمح لك بإجراء تغييرات على شيفرة HTML و CSS و JavaScript، ورؤية النتائج مباشرةً. تنقسم الشاشة إلى أربعة أقسام: قسم المعاينة، وأقسام HTML و CSS و JavaScript. يوجد داخل كل قسم خيارات تسمح لك بإعداد اللغات (Sass بدلاً من CSS على سبيل المثال)، وغيرها من الأشياء المفيدة.

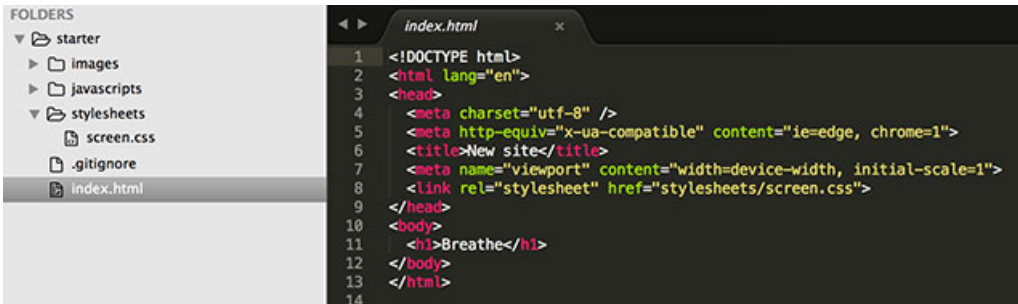
2. التطوير المحلي

أفضل التطوير محليًا للمشاريع الكبيرة. هناك عدّة طرائق للقيام بذلك، والتي هي أكثر فعالية وسرعة من العمل في المتصفح مباشرةً.

أ. الخيار الأبسط: إنشاء ملفي HTML و CSS

الخيار الأبسط هو إنشاء ملف HTML (filename.html) وربطه بملف CSS (filename.css). هذه طريقة مقبولة، لكن يمكن أن تكون بطيئة، إذ يتوجّب عليك التنقل كثيرًا بين المتصفح والمحرّر.

لقد أنشأت مجموعةً من ملفات HTML و CSS، ويمكنك نسخها واستخدامها لبدء العمل، وتستطيع تنزيلها من [هنا](#).



```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8" />
5 <meta http-equiv="x-ua-compatible" content="ie=edge, chrome=1">
6 <title>New site</title>
7 <meta name="viewport" content="width=device-width, initial-scale=1">
8 <link rel="stylesheet" href="stylesheets/screen.css">
9 </head>
10 <body>
11 <h1>Breathe</h1>
12 </body>
13 </html>
14

```

ب. أدوات إضافية

يمكنك بالطبع استخدام أي أداة تجدها مفيدةً ومريحةً لإنشاء صفحات الويب مثل Dreamweaver أو Macaw أو Muse أو Coda أو Sublime Code أو Visual Studio Code.

كل ما تحتاج له عملياً هو مُحَرَّر نصوص. تأتي بعض الأدوات الأخرى مع إمكانية التحرير البصري، واستخدامها إذا كنت تُفضِّل العمل بها.

أوصي شخصياً بالعمل مباشرةً على الشيفرة. سيساعدك فهم طريقة عمل CSS على إصلاح المشاكل، أو إنشاء تأثيرات لا يمكن أن توفرها الأدوات البصرية.

ج. أداة البناء Gulp

إذا كنت معتاداً على GitHub و Node ومعالجة الشيفرات، فقد ترغب في إعداد بيئة تطوير على جهازك.

أحب أداة Gulp، إذ تستمد سرعتها الكبيرة من اعتمادها على Node. يمكن تجميع الوحدات لتحويل Sass إلى CSS، والمزامنة مع المتصفحات بحيث لا تحتاج إلى تحديث المتصفح في كل مرة تحدّث فيها الشيفرة. (إن لم يكن لديك سابق معرفة بأداة البناء Gulp، فأنصحك بقراءة مقالة «دليلك الشامل إلى أداة البناء Gulp» قبل الانتقال إلى الفصل التالي).

إذا سبق واستخدمت Grunt، أو أدوات البناء الأخرى، فسيكون سير العمل مألوفاً.

لقد أنشأت مستودع GitHub لجعل التطوير المحلي أسرع. إذا كنت معتاداً على استخدام Git، فزُر صفحة المستودع، واطالع الملف readme الذي يحتوي إرشادات الإعداد.

يمكنك تحسينه ومشاركته إذا رغبت في ذلك.

3. خلاصة الفصل

أثناء تعلُّمك للتحريك في **CSS**، لا تتردد في التعديل على الشيفرة وتجربة أمور جديدة. قد ترغب في استضافة الشيفرة بنفسك، أو قد تُفضِّل استخدام **CodePen**، والأمر يعود لك في جميع الأحوال. تأكَّد من أنَّه يمكنك تحويل الأفكار إلى شيفرة بسلاسة.

4. تمرين

أنشئ حسابًا في **CodePen** ثمَّ أضف بعض شيفرات **HTML** و **CSS**، وانظر بنفسك كيف تتغير النتائج مباشرةً. من الجيد أيضًا مطالعة بعض أمثلة **CodePen** على الصفحة الرئيسية، وتعلُّم كيفية عملها.

إذا كنت ترغب في تجربة التطوير المحلي، نزل ملفات البدء المحلية (خطوة اختيارية):

- الخيار الأساسي: ملفات **HTML** و **CSS** لبدء المشروع.
- متقدم: ملفات البدء **Gulp & Sass**.

3

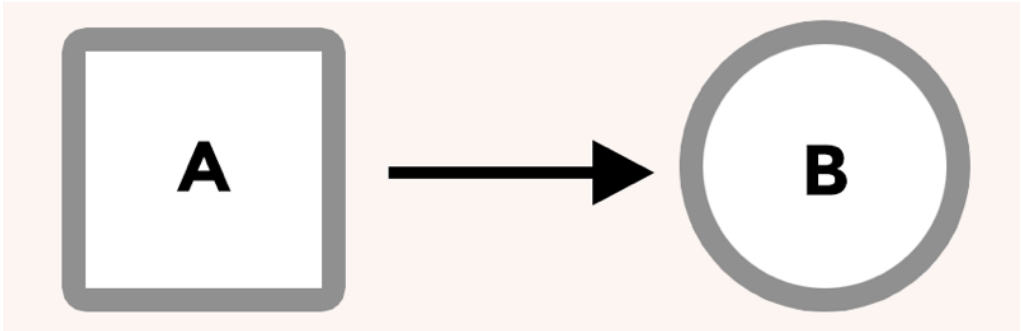
مدخل إلى الانتقالات: التنقل بين الحالات

سنلقي في هذا الفصل نظرةً عامةً على خاصية الانتقال **transition** في CSS. كانت المتصفحات فيما مضى بسيطةً للغاية. فقبل وقت غير بعيد، لم تكن قادرة حتى على عرض الصور، أو التعامل إلا مع حفنة من الخطوط. جاءت CSS بعد ذلك ومنحتنا القدرة على التحكم في شكل صفحات الويب ومظهرها.

التحريك (Animation) في المتصفحات ليس جديدًا. فقد وفّرت لنا بعض مكتبات JavaScript مثل Flash و Canvas وغيرها طرائقَ للتحريك، ولكن أُضيف خيار CSS في الآونة الأخيرة وأصبح متميزًا.

1. الانتقالات

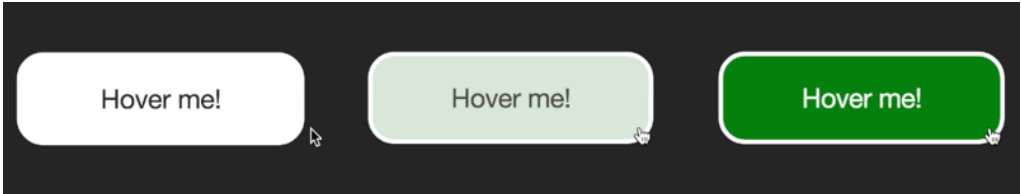
إحدى الطرائق التي تتيحها CSS للتحكم في الحركة في المتصفح هي باستخدام الخاصية **transition**. بلغة المتصفحات، فإنَّ الانتقال (transition) يعني الانتقال من حالة إلى أخرى.



نخبر المتصفح عندما نطبّق عملية الانتقال على عنصرٍ ما بأننا نريده أن ينقل العنصر من حالةٍ إلى أخرى، ويحسب تلقائيًا التغيُّر بين حالتين (مثال حي).



على سبيل المثال، يمكننا تغيير تنسيق عنصر ما عند تمرير مؤشر الفأرة عليه، فحين تطبيق عمليّة الانتقال سينقل المتصفح التنسيق الأولي للعنصر إلى التنسيق الجديد نقلًا سلسًا مما يوحي بتطبيق حركة وحيوية على العنصر (مثال حي).

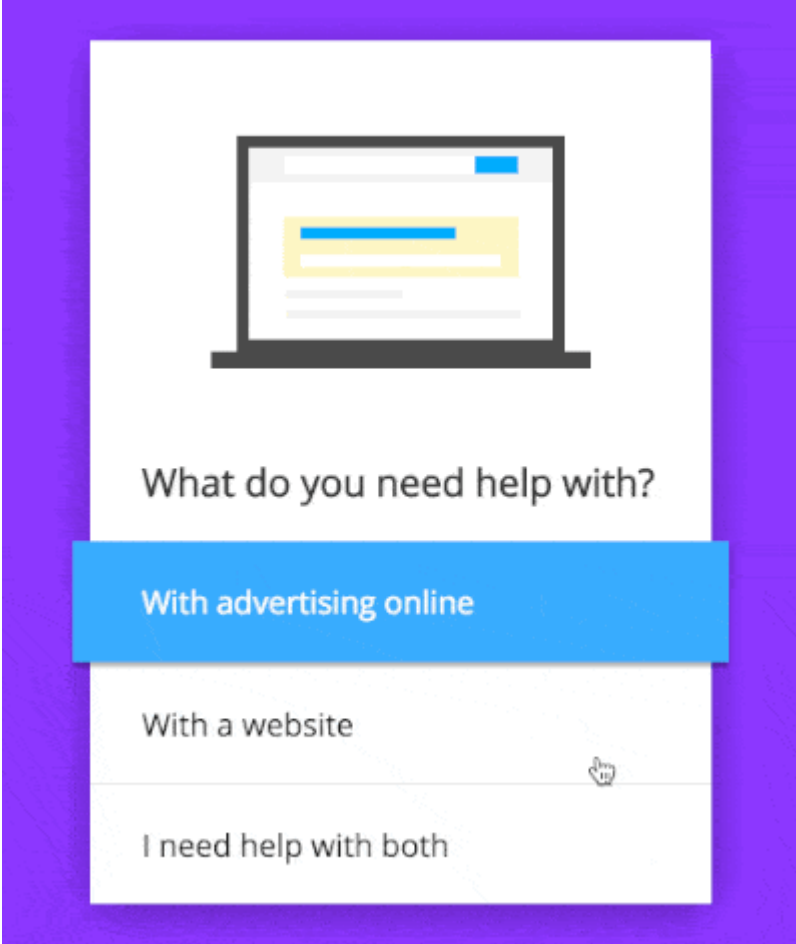


2. خصائص الانتقال

عندما نطبّق تحريك الانتقال على عنصر ما، فهناك خصائص كثيرة يمكن أن نتحكم في كيفية تطبيق الانتقال. يمكننا أن نسرّع عمليّة الانتقال أو نبطئها، أو نوخّرها، بل ويمكننا أن

نتحكم في وتيرة التغيير باستخدام دوال التوقيت (timing functions) أيضًا. سوف نتطرق إلى ما يعنيه هذا في الفصل التالي.

مثال حي آخر على الانتقالات المركبة:



سنناقش قريبًا كيفية استخدام الانتقالات لإنشاء مثل هذه الحركات.

3. خلاصة الفصل

الانتقال هو تغيير من حالة إلى أخرى. على سبيل المثال، عند تمرير مؤشر الفأرة فوق عنصر طُبِّق عليه انتقالٌ ما، فسيتميّز شكله عبر نقله من المظهر الذي كان عليه إلى مظهر وشكل آخر. سيبدو التغيير وكأنه حركة متصلة وسلسة بفضل خاصية الانتقال.

سنغيّر الواجهة في **الفصل التالي** للحديث عن الخاصية `animation` وكيف تختلف عن

الخاصية `transition`.

4. تمرين

كيف تبدو بيئة العمل خاصّتك؟ أنصحك بإلقاء نظرة على الشيفرة والبحث عن الخاصية

`transition` في شيفرة `CSS`. وهل يمكنك أنذاك تخمين ما تفعله؟

في المرة القادمة التي تتصفح فيها الويب، ابحث عن أمثلة جرى تطبيق الانتقالات عليها

واستكشف التغييرات التي أُجريت والتي أثارت اهتمامك، مثل ما يحدث عندما يُضَاف عنصر

جديد إلى الصفحة، أو عند تمرير مؤشر الفأرة فوق زر ما. ستجد أنّ الويب مليء بالحركات

المنشأة عبر الانتقالات بمجرد أن تبدأ بالبحث عنها.

4

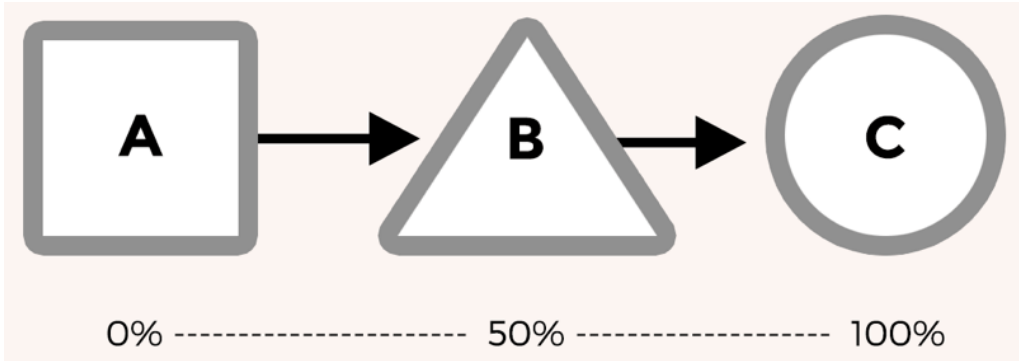
مدخل إلى الحركات: مفهوم الحركة

لقد ناقشنا في الفصول السابقة استخدامات الحركات وفوائدها، ووجدنا بعض مصادر الإلهام، وألقينا نظرة على الأدوات والمواقع التي قد تكون مفيدة في التطوير، وتعرّفنا على كيفية تطبيق حركة عبر نقل عنصر من حالة إلى أخرى. وسنتعرّف في هذا الفصل على الخاصية `animation`.

1. التحريك في المتصفح

الانتقال (`transition`) والحركة (`animation`) متشابهان، فكلاهما يأخذ شكل خاصية CSS، ويحدث خلال مدة زمنية معيّنة، إضافةً إلى خصائص أخرى تتحكم في كيفية إنشاء المتصفح للحركة وتنفيذها.

لما كانت الانتقالات تدور حول نقل العنصر من الحالة "A" إلى الحالة "B" نقلاً سلساً يوحي بحركة، فإنّ الحركات هي وسيلة لوصف مراحل متعدّدة من حالة العنصر أي تغيير حالة العنصر عدّة مرات خلال مدّة زمنيّة.



في المثال أعلاه، هناك 3 حالات (A و B و C) للعنصر. سيؤدي تطبيق الانتقال إلى نقل العنصر من الحالة A إلى C فحسب، في حين تسمح لنا عملية التحريك بتحديد مرحلة وسطية B

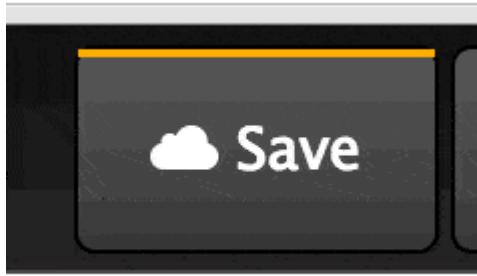
واحدة أو أكثر ينتقل إليها العنصر قبل الانتقال إلى الحالة النهائية.

تتصرّف الحركات أيضًا تصرّفًا مختلفًا، إذ يمكنها أن تبدأ تلقائيًا. بينما قد يتطلب الانتقال إضافة صنف، أو تغييرًا في الحالة، مثل تمرير الفأرة فوق العنصر (الحومان، hovering)، كما يمكن أن تبدأ الحركات عند تحميل الصفحة.

هذا يعني أنّه في حال رغبت في سرد قصة ما، أو لفت الانتباه إلى شيء ما على الصفحة، فتعدّ الحركات خيارًا جيدًا.

2. أمثلة

تُعدُّ حركة الزر "Save" التي نراها في Codepen مثالًا جيّدًا على الحركات.



تساعد هذه الحركة على لفت انتباه المستخدم لزر الحفظ لتذكيره بضرورة حفظ عمله. يتكون هذا التأثير (الحركة) من سلسلة من **الإطارات المفتاحية** (keyframes) التي تخبر المتصفح أن يَهْزِرَ الزر من اليسار إلى اليمين. سنتعمّق أكثر في شرح الإطارات المفتاحية في **الفصل الثاني عشر**.

يمكننا أن نفعل الكثير من الأشياء عبر تحريك ونقل الإطارات المفتاحية في CSS. هذا

مثال آخر ثلاثي الأبعاد.



المثال CSS Mac Plus متوفر على موقع CodePen، وهذا دليل مفصل لكيفية تصميمه.

3. الانتقال أم الحركة؟

تحدث الانتقالات عندما ينقل المتصفح عنصرًا من حالة إلى أخرى (من A إلى B) مما يوحي

بحدوث حركة. ويُطبَّق ذلك عادةً نتيجةً لحدث ما، مثل الحومان (hovering) فوق عنصر ما، أو

إضافة صنف أو إزالته باستخدام JavaScript.

الحركات أكثر ديناميّة، وتحدث تلقائيًا، وتتيح لك تغيير حالة عنصر بتسلسل معيّن عبر عدد من الإطارات المفتاحية التي تمثّل مراحل الحركة، ويمكن أن تُنفذ خلال حلقات تكرار (loops). على أي حال، سنعود للخاصيّة **animation** لاحقًا، إذ سنضع الانتقالات في **الفصل القادم** موضع التطبيق وسناقش بعض الأمثلة العمليّة.

4. تمرين

هل يمكنك التفكير في طرائق مبتكرة لاستخدام الحركات في صفحات الويب الخاصة بك؟ حاول أن تنتبه للحركات التي تحدث على المواقع التي تتصفحها وابحث عن الأشياء التي تتحرك بطريقة مُلفتة وافهم الفكرة القابعة خلف تطبيق الحركة وتعلّم منها. إذا نزلت ملفي HTML و CSS الابتدائيين، فألق نظرة على الخاصيّة **animation**. على عكس الانتقالات، تحتاج الحركات إلى جزء ثانٍ يسمى **الإطارات المفتاحية** (keyframes). حاول تغيير بعض القيم وشاهد ما يحدث.

5

الانتقالات عمليًا

الآن وبعد أن تعرّفنا على الخاصيّتين transition و animation، فقد حان الوقت للتعمّق أكثر في الانتقالات (transitions)، ومطالعة بعض الأمثلة.

1. الانتقالات

تحدث الانتقالات في المتصفح عندما يتحوّل عنصر من حالة إلى أخرى نتيجةً لحدثٍ ما. يرسم المتصفح **الإطارات المفتاحية** (keyframes) المُتولّدة بين الحالة الابتدائية والحالة النهائية مما يُولّد شعورًا بالحركة.

الخاصية transition هي إحدى خاصيّات CSS. أي مثلها مثل الخاصيّات height أو width أو border.

يمكننا كتابة الانتقالات في CSS على النحو التالي:

```
transition: background 0.5s linear;
```

في هذه الحالة، نقول للمتصفح أنّ أيّ انتقال لخاصيّة الخلفيّة (background) سيستغرق نصف ثانية، إضافةً إلى استخدام دالة التوقيت «الخطي» (linear).

قد تُنسب الخاصية المذكورة أعلاه في تغيير خلفيّة الزر عند تمرير مؤشر الفأرة فوقه:

```
button {
  background: white;
  transition: background 0.5s linear;
}

button:hover {
```

```
background: green;
}
```

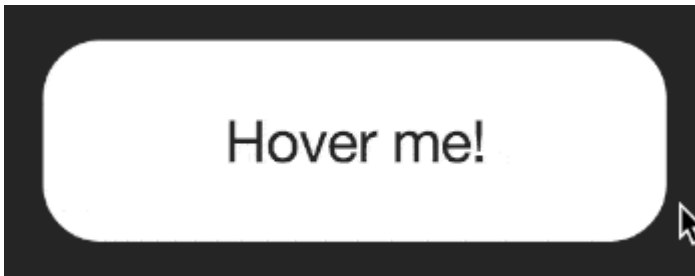
لاحظ الخاصية `transition` في المُحدِّد `button`، والتي تقول للمتصفح أن يُطبِّق الانتقال على أي تغيير في الحالة، مثل عند التغيير المؤقت الذي تُحدِّثه حالة الحومان (`hover`)، وكذلك عند العودة إلى الحالة الأصليَّة بعد زوال الحومان.

إذا طبَّقنا الخاصية `transition` على حالة الحومان (`hover`) وحسب، فلن يبدأ تأثير الانتقال إلا إلى حالة الحومان، ولكن ليس عند العودة للحالة الأولى.

إليك هذان المثالان التوضيحيان. قد تجد أنَّ هذه الأمثلة تحتوي على بعض الشيفرات غير المفهومة. لا تقلق، فسأفضِّل كل شيء في الفصول المقبلة، لكن لا تتردد بتجريب وتعديل القيم لمعرفة ما يحدث.

2. مثال: تطبيق انتقال على زر

إليك مثالاً في [CodePen](#) يوضح تأثير الحومان (`hover effect`). في [CodePen](#)، يمكنك إجراء تغييرات على HTML و CSS ورؤية النتائج مباشرة.



الشيء المهم الذي ينبغي التركيز عليه هو الخاصيات التي تبدأ بالبادئة -transition.

```
transition-property: all;
transition-duration: 0.4s;
transition-timing-function: ease-out;
```

تخبر هذه الشيفرة المتصفح بنوع الحركة التي ينبغي إنشاؤها عند الانتقال من حالة عدم الحومان (non-hover state)، إلى حالة الحومان (hover state). كما تُخبر المتصفح بالتحريك (النقل) التدريجي لكل الخصائص (الألوان والحجم والموضع... إلخ). خلال 0.4 ثانية.

حاول تغيير بعض هذه القيم مثل القيمة "0.4s" إلى قيمة أكبر، مثل "2s" (ثانيتان) ثم لاحظ كيف ستبدو الحركة؟ يمكنك كذلك تغيير الخاصية من all إلى background لحصر تطبيق تأثير الانتقال على الخلفية فقط.

لإنشاء تأثير ممتع، حاول تغيير قيمة transition-timing-function

من ease-out إلى:

```
transition-timing-function: cubic-bezier
(.59, -0.26, .33, 1.42)
```

تضفي دالة التوقيت cubic bezier على الحركة متعةً وجاذبيةً. سنعود لدوال التوقيت

بمزيد من التفاصيل في الفصول اللاحقة.

3. البادئات وتوافق المتصفحات

لم أضف بادئات المتصفحات (vendor prefixes) في شيفرات الأمثلة وذلك لتسهيل قراءة الشيفرة، ولكن في حال كنت تستخدم الشيفرة في مرحلة الإنتاج، فسيتوجب عليك إضافتها إذا أردت دعم المتصفحات القديمة لضمان تشغيل الحركات عند شريحة من زوار موقعك أصحاب المتصفحات القديمة (وياحبذا لو وضعت رسالةً تخبرهم فيها أن يحدثوا متصفحهم :-).

أحب استخدام الأداة Autoprefixer (وهو خيار متاح في Codepen، اضغط على أيقونة الإعدادات "cog" في قسم CSS لتفعيله)، ويمكن تفعيله باستخدام أدوات البناء، مثل Grunt أو Gulp (مشروح في [هذا المقال](#) كيفية استعمال Gulp مع Autoprefixer لإضافة البادئات تلقائيًا). بالمقابل، يمكنك كتابة البادئات يدويًا على النحو التالي:

```
-webkit-transition: ...;
-moz-transition: ...;
transition: ...;
```

4. تمرين

حرّر الزر في المثال، وأضف أفكارك الخاصة إليه. يمكنك تغيير الشكل (shape)، أو الإطار (border)، أو أي خاصية أخرى. جرّب واستمتع، فالهدف هو التأكد من أنك تفهم كيف يؤثّر الانتقال على حالة الحومان الخاصة بالعنصر.

لمزيد من الإلهام، تحقّق من تنسيق الحومان في [هذا الرابط](#). هناك الكثير من الأمثلة الرائعة

لاستلهام الأفكار.

إذا كنت تريد التعمق في الموضوع أكثر، فأُنشئ مشروعًا جديدًا في CodePen، مع إنشاء عنصر يتغير من حالة إلى أخرى عند الحومان عليه. حاول معرفة ما إذا كان يمكن أن يحتوي داخله عنصرًا يتحرك بمعدل مختلف. لا تقلق إذا لم تنجح في ذلك، فسنغطي خاصيّات التحريك بمزيد من التفاصيل في الفصول القادمة.

6

خاصيات الانتقالات

تعلمنا في الفصول الماضية كيفية عمل الخاصية transition، وسنلقي في هذا الفصل نظرةً على خاصيات *transition الفرعية.

1. الصياغة المختصرة مقابل الصياغة المطوّلة

عند كتابة شيفرة CSS، يمكننا في كثير من الأحيان اختصار عدة خاصيات في خاصية واحدة مختصرة. على سبيل المثال، الصياغة المختصرة للخاصية **padding** تُكتب على النحو التالي:

```
padding: 10px 20px 15px 25px;
```

وهذا يكافئ:

```
padding-top: 10px;
padding-right: 20px;
padding-bottom: 15px;
padding-left: 25px;
```

يمكننا بالطريقة نفسها كتابة الخاصية transition باختصار:

```
transition: all 0.5s 1s linear;
```

وهذا هو الشكل العام لها (لصيغة المفصلة الكاملة، راجع قسم **البنية الرسمية** في صفحة

الخاصية transition في **موسوعة حسوب**):

```
transition: [property] [duration] [delay]
           [timing-function];
```

يمكن كتابة كل واحدة من هذه الخاصيات بخاصية منفردة:

```
transition-property: all;
transition-duration: 0.5s;
transition-delay: 1s;
transition-timing-function: linear;
```

لنلق نظرة على هذه الخاصيات.

أ. الخاصية `transition-property`

عادةً ما ترد هذه الخاصية في بداية الصياغة المختزلة، وتحدّد خاصية CSS للعنصر التي سيطبّق المتصفح عليها الانتقال. لتغيير الخلفية على سبيل المثال، يمكن استخدام القيمة `background` معها. من الممكن أيضاً استخدام القيمة `all`، والتي تشمل جميع خاصيات CSS التي يطبّق عليها الانتقال.

ب. الخاصية `transition-duration`

تمثّل قيمة الخاصية `transition-duration` المدة التي يستغرقها الانتقال. فترة انتقال تساوي 3s (ثلاث ثوانٍ) ستكون أطول بثلاث مرات من فترة انتقال تساوي 1000ms (ألف ميلي ثانية).

ج. الخاصية `transition-delay`

تخبر الخاصية `transition-delay` المتصفح بالانتظار لمدة محدّدة قبل تطبيق الانتقال. أي تمثّل هذه الخاصية قيمةً زمنيّةً، ويمكن تحديدها بالثانية مثل 3s (تكافئ ثلاث ثوانٍ) أو الملي ثانية مثل 100ms (تكافئ مئة ميلي ثانية). بشكل مكافئ، يمكنك كتابة تلك القيمة على شكل 0.1s بدلاً من 100ms والأمر متروك لك.

د. الخاصية transition-timing-function

تستخدم كلُّ من الانتقالات (transitions) والحركات (animations) دوال توقيت (timing functions). تحتاج دوال التوقيت هذه إلى فصلٍ خاصٍّ بها، لذا سنؤجل الحديث عنها إلى الفصل التالي. لكن بإيجاز، تعطي دوال التوقيت للحركات حيويَّةً كبيرة.

2. الأشياء التي لا يُطبَّق الانتقال عليها

رغم أنَّه يمكن تطبيق الانتقالات على الخاصيَّات `color` و `border` و `background-position` وغيرها، إلا أنَّ هناك بعض الخاصيَّات التي لا يمكن تطبيق الانتقال عليها. فمثلاً، لا يمكن تطبيق الانتقال على عائلة الخطوط `font-family`، لأنَّ هذا يعني أنَّ على المتصفح إنشاء إطارات مفتاحية (keyframes) بين صورتين مختلفتين تمامًا من الخطوط.

صور الخلفية المُنشأة باستخدام CSS، مثل التدرجات (generated gradients)، لا يمكن تحريك خاصيَّاتها لأنَّ ذلك يعني أنَّه سيتوجب على المتصفح استبدال صورة الخلفيَّة عند كلِّ إطار مفتاحي للحركة، لذلك فهو غير مدعوم. لكي تعرف إذا كانت الخاصية تدعم الحركات، فانظر إلى قيمة «قابلة للتحريك» في بطاقة الخاصية في توثيق CSS في موسوعة حسوب. لكن يمكن تحريك خاصيَّات من قبيل `opacity` و `background-position`. وعن طريق تحريك صور الخلفيَّة أو إخفائها، ويمكنك بذلك إنشاء تأثيرات جذَّابة.

يمكنك مشاهدة مثال Baymax، وفيه تُحرَّك صورة الخلفيَّة لتوليد حركةٍ رائعة.



يُستخدَم تأثير مماثل لإحداث تأثير اللمعان على الأزرار، إذ يُحرَك تدرج الخلفيّة في

مقدّمة الزر.

A rectangular button with rounded corners, featuring a light orange-to-white gradient background. The text "SHINY EFFECT" is centered on the button in a white, bold, sans-serif font.

SHINY EFFECT

3. تمرين

لقد أنشأت مثالاً على Codepen لتجربة الانتقالات. يطبّق الانتقال فيه على عنصر شكله شكل مُعيّن لنقله في أثناء حومان مؤشر الفأرة فوقه إلى شكل الدائرة بتأثيرٍ جدّابٍ ورائعٍ. حاول تغيير بعض الخاصيّات لمعرفة ما سيحدث.

إذا كنت ترغب في التعمق أكثر في هذا الموضوع، فاضغط على الزر "Fork" لإنشاء نسختك الخاصّة، ويمكنك بعد ذلك حفظ عملك في حساب Codepen الخاصّ بك.

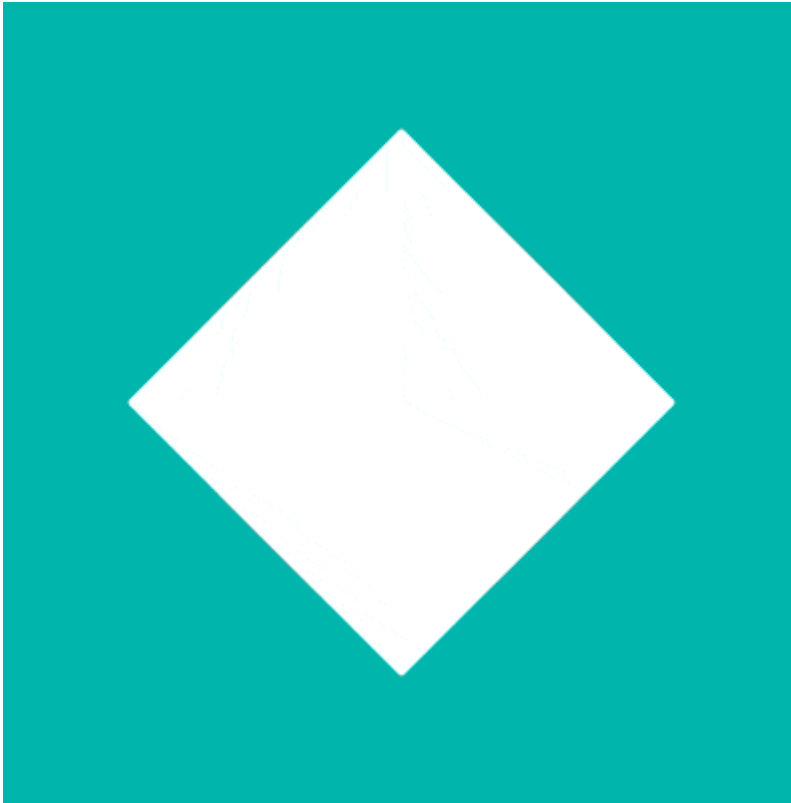
دوال التوقيت

7

دالة التوقيت (timing function) تتحكم في وتيرة وسرعة التغيرات التي تحدث إبّان الانتقال (transition)، ويمكن باستخدامها جعل الانتقالات أكثر حيويّة.

فيما يلي مثال للانتقال كره باستخدام دالة توقيت خطيّة، إذ تتحرك جيئةً وذهابًا بوتيرة ثابتةٍ وازنه بهذا المثال الذي يستخدم دوال التوقيت المسماة **cubic-bezier**. ستلاحظ فرقًا كبيرًا في وتيرة الحركة.

في هذا المثال، سنستخدم دالة **cubic-bezier** مُخصّصة:



تُرجع دالة cubic-bezier في هذا المثال الحركة إلى الوراء قليلاً قبل التحول بسرعة إلى الحالة الثانية، إذ تتجاوزها فعلياً ثمّ تعود إليها.

أمثلة CSS التي ذكرناها في البداية، وحالة الحومان (hover state) التي أشرنا إليها في كل الأمثلة تعتمد على دالة توقيت.

سنلقي نظرة على كل تلك الدوال، ونتعرّف على كيفية تأثيرها على الطريقة التي تتحرك بها العناصر.

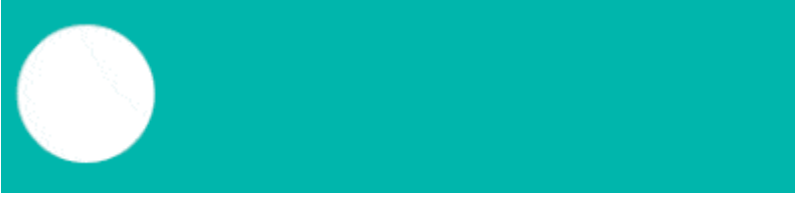
إذا كنت ترغب بالتجريب على هذه الأمثلة، فقد أعددت مشروعاً على CodePen يمكنك رؤيته والتعديل عليه.

1. التسارع الخطي (Linear)



يتغير التسارع الخطي (linear transition) بمعدّل ثابت من البداية إلى النهاية. ونظراً لعدم وجود منحنى في حركة الانتقال، فهو لا يتسارع أو يتباطأ. يمكن أن يكون هذا مفيداً إن كنت تريد إنشاء حركة ذات تسارع ثابت، مثل مشهد خلفية نافذة قطار متحرك، أو الدوران الثابت.

2. التسارع المتباطئ عند البداية (Ease-in)



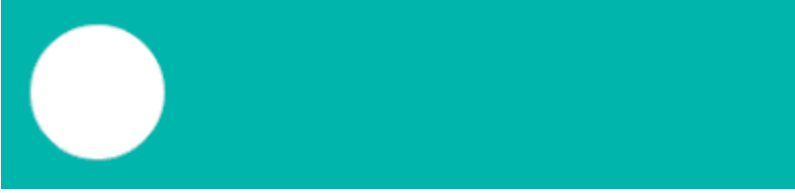
تبدأ دالة التسارع المتباطئ عند البداية (ease-in timing function) ببطء ثمّ تتسارع في نهاية الانتقال. هذا يشبه كرةً تتدحرج من أعلى التل، إذ تبدأ ببطء، وتنتهي بسرعة كبيرة في الأسفل.

3. التسارع المتباطئ عند النهاية (Ease-out)



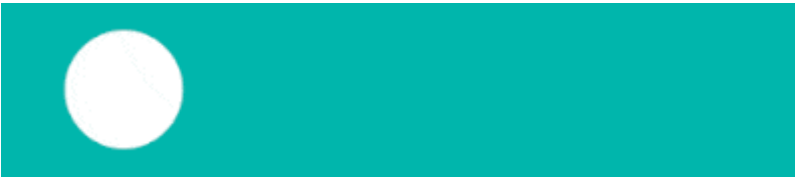
تنصّرف دالة التسارع المتباطئ عند النهاية (ease-out timing function) تصرّفًا معاكسًا لدالة التسارع المتباطئ عند الانطلاق، إذ تبدأ بسرعة ثمّ تتباطأ في النهاية. هذا مفيد في الحالات التي تحتاج فيها إلى إعطاء الزائر شعورًا بأنّ شيئًا ما يندفع من خارج الشاشة بسرعة ثمّ تنخفض سرعته تدريجيًا حتى يتوقف.

4. التسارع المتباطئ عند البداية والنهاية (Ease-in-out)



دالة التسارع المتباطئ عند البداية والنهاية (ease-in-out timing function)، هي مزيج من دالتي التسارع المتباطئ عند البداية و التسارع المتباطئ عند النهاية، إذ تبدأ ببطء ثمّ تتسارع خلال الجزء الأوسط من الحركة ثمّ تتباطأ عند النهاية. يمكن أن تمثّل سيارة تبدأ من حالة توقف تام، ثمّ تتسارع، ثمّ تنخفض سرعتها حتى التوقف. كما قد تكون مفيدة لإنشاء حركة تمثّل عملية التحميل.

5. التسارع المُخصّص (منحني Cubic-bezier)



جميع دوال التوقيت التي رأيناها حتى الآن هي أمثلة لمنحني cubic bezier. وهو منحني يصف شكل دالة التوقيت. وهكذا، فإنّ تحديد دالة توقيت عبر منحني cubic bezier يكافئ إنشاء دالة توقيت خاصة بنا.

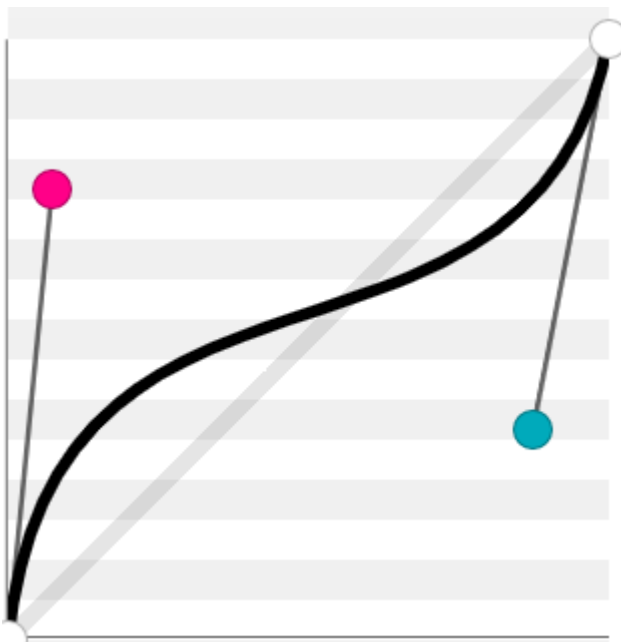
تتألف دوال توقيت cubic bezier من أربع قيم، تمثل إحداثيتين (coordinates).

صيغة cubic-bezier قد تبدو على الشكل التالي:

transition-timing-function:

cubic-bezier(0.075, 0.75, 0.875, 0.36);

الإحداثيتان هنا هما (0.075, 0.75) و (0.875, 0.36). وتبدو على الرسم البياني كما يلي:



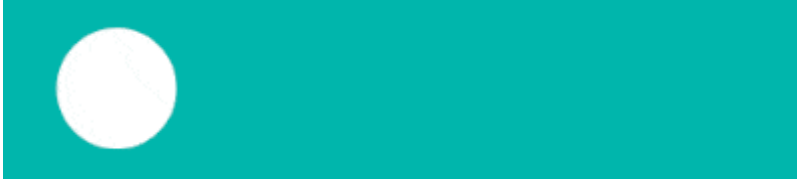
بدلاً من إنشاء المنحنيات يدويًا، فأنا أستخدم موقع cubic-bezier.com. يُمكنك هذا

الموقع من إنشاء بعض التأثيرات الرائعة.

تصبح منحنيات cubic-bezier ممتعةً عند استخدام قيم أكبر من 1، إذ من الممكن إنشاء

انتقالات تتخطى الحدود وترتد.

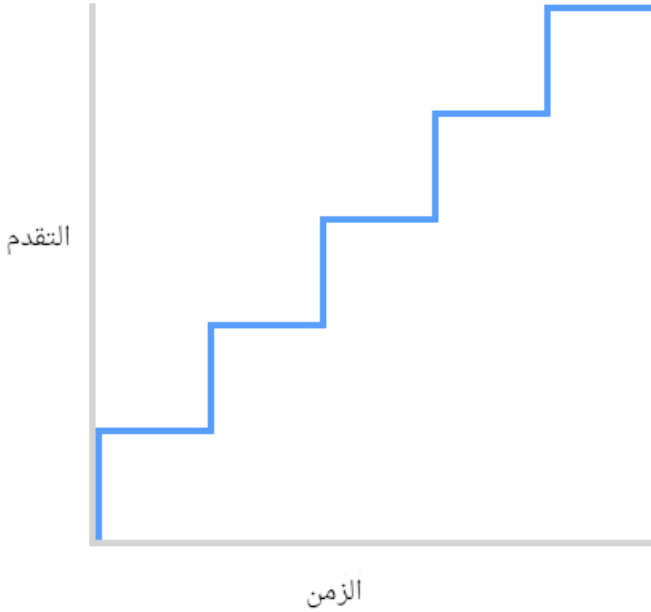
6. الانتقال الخطوي (Steps)



لما كانت معظم دوال التوقيت تعتمد على المنحنيات، فإنَّ الدالة steps تُقسِّم عملية

الانتقال إلى مجموعة من المراحل، وتقفز من مرحلة إلى التي تليها. على سبيل المثال، إذا حدَّدت

القيمة (4) steps، فإنَّ مدَّة الانتقال سَتُقسَّم إلى أربع خطوات منفصلة (انظر الصورة أعلاه).



هذا مفيد للحركات المفاجئة. على سبيل المثال، «دائرة التحميل» (loading spinner)، أو شخصية متحركة للعبة فيديو. من خلال تعيين موضع الخلفية في بداية سلسلة الإطارات (frames)، يمكن استخدام دالة التوقيت steps للقفز من إطار لآخر، وإحداث شعور بالحركة. للاطلاع على مثال عما نتحدث عنه، تفحص مقال [حركة زر تفضيل التفريدة الخاص بتويتر](#).

يمكنك أيضًا تحديد ما إذا كان الانتقال سيحتفظ بالإطار الأول لجزء من مدة الانتقال، أم أنه سيحتفظ بالإطار النهائي. الوضع الافتراضي هو الإطار النهائي، على أساس أن الإطار الأول قد ظهر بالفعل قبل أن تبدأ الحركة.

يمكننا تحديد الخيار المناسب عند تحديد الخطوات:

```
transition: all 2s steps(10, start);  
transition: all 2s steps(10, end);
```

7. تمرين

تتمهً للتمرين السابق: حاول تغيير قيمة transition-timing-function، وشاهد كيف

يغيّر ذلك الطريقة التي يظهر بها الانتقال.

يمكنك أيضًا محاولة تغيير القيم في هذا المثال التوضيحي. من الناحية الفنية، فهي حركة

(animation) وليست انتقالاً (transition)، لكنّ دالة التوقيت تُطبّق بالطريقة نفسها.

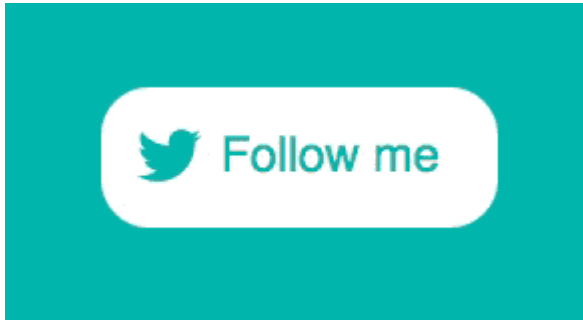
8

الانتقالات المتعددة

لقد ناقشنا في المقالات السابقة كيف يُنشئ الانتقال (transition) الحركة عبر نقل العنصر من حالة إلى أخرى. سنرى في هذا الفصل ما يحدث عندما تُطبَّق عملية انتقال واحدة على عنصر واحد تحدث له عدَّة تغييرات، وكيفية استخدام عدة انتقالات معًا لتحسين الحركات.

1. المثال الأول: زر فني

رأينا في المقالات السابقة تأثيرًا بسيطًا للحومان على الأزرار (button hover)، ويمكننا دمج عدَّة انتقالات على الزر نفسه للحصول على تأثير أكثر تعقيدًا (تجربة حية).



في هذا المثال، يجمع تأثير الحومان بين عدَّة تغييرات للحالة، ولكنها مُحدَّدة جميعًا عبر

انتقال واحد:

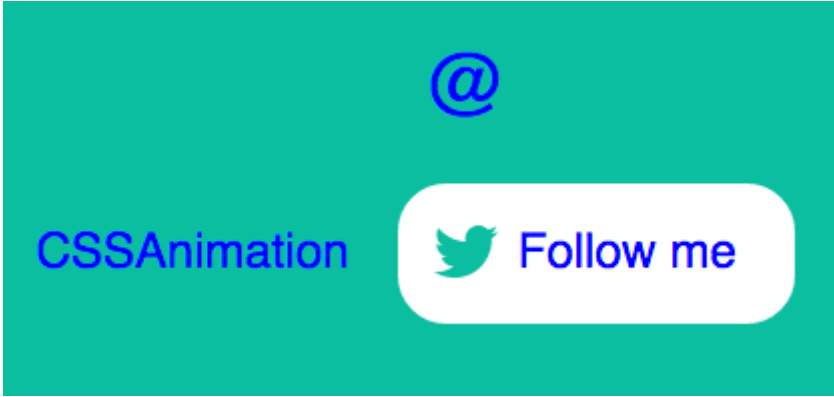
```
transition: transform 0.4s
            cubic-bezier(.72, -0.61, .25, 1.51);
```

إليك كيف يعمل هذا المثال. يتألف الزر من أيقونتين ونصين. في الحالة الأولى الساكنة

(حالة عدم الحومان non-hover state)، يوضع النص "Follow me" (تابعني) وأيقونة

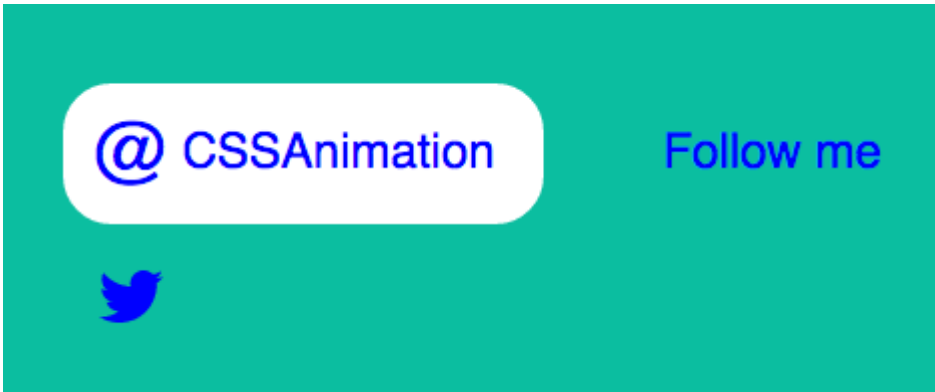
Twitter داخل الزر. نضع الرمز @ والنص "CSSAnimation" خارج الزر على النحو التالي

(الحالة الأولية الساكنة للزر):



ثمّ نضيف حالة ثانية تُطبَّق عند التحوي على الزر، إذ نضع العناصر الموجودة خارج الزر

داخله، كما يلي:



يمكننا فعل هذا باستخدام انتقالات **CSS**. على سبيل المثال، يُحدّد موضع أيقونة Twitter

باستخدام التموضع المطلق (absolute positioning). يمكن عند إعداد الأيقونة تحديد

موضعها في المكان الذي نريد باستخدام القيم left و top:

```
.icon {
  position: absolute;
  left: 0.75em;
  top: 0.75em;
}
```

ثمّ نضيف حالة حومان للزر، ونضع أيقونة Twitter خارج الزر عبر الخاصية `transform`:

```
a:hover .icon {
  transform: translateY(3em);
}
```

إضافة `overflow: hidden` إلى الحاوية (container) يعني أنّ العناصر الموجودة خارج

الزر لن تظهر.

ستختفي الأيقونة فجأةً في غياب الانتقال نظرًا لأنّ كل عنصر من العناصر الموجودة داخل

الزر من النوع `span`، فيمكننا تطبيق الانتقال عليها جميعًا دفعةً واحدة:

```
span {
  transition: transform 0.4s
    cubic-bezier(.72, -0.61, .25, 1.51);
}
```

هذا يعني أنّه سيُطبَّق انتقالٌ على كل عناصر `span` إذا تغيّرت حالتها، مثل الحومان عليها.

تُطبَّق الحيلة نفسها على الأجزاء الأخرى من الزر. يمكنك الاطلاع على [هذا المثال بالكامل](#)

على [CodePen](#).

2. المثال الثاني: كشف الخلفية

في [هذا المثال](#)، أعددت بطاقةً تحوي نصًا، مع عرض نص عند الحومان عليها أيضًا.



Hover cat

الحالة الأوليّة (حالة عدم الحومان) للبطاقة تُظهر عنوانًا، ولكنّ عتامة (**opacity**) نص الفقرة تساوي الصفر (أي النص غير مرئي). عند الحومان عليها، نغيّر قيمة العتامة إلى القيمة 1 لإظهار النص، مع تغيير ارتفاع حاوية النص.

في غياب الانتقال، **سيبدو هكذا**. عندما نمزّر مؤشر الفأرة فوق البطاقة، فسيكون التغيير مفاجئًا. سينتغيّر الانطباع تمامًا مع إضافة انتقالين، **وها هي النتيجة**.

يبدو الانتقال الأول (المكتوب بالصياغة المختزلة هذه المرة) كما يلي:

```
transition: all 0.5s cubic-bezier(.48, -0.28, .41, 1.4);
```

يُخبر هذا المتصفح بأنَّ عليه تحريك جميع الخاصِّيات على مدار 0.5 ثانية، مع استخدام

انتقال cubic-bezier لجعله يرتد. وفي هذه الحالة، فهو يؤثر على ارتفاع حاوية النص.

يجعل الانتقال الثاني النص يتحرك. سنستخدم هنا دالة التوقيت ease-out:

```
transition: all 0.4s ease-out;
```

يمكننا تحقيق الكثير من التأثيرات من خلال تغيير الحالات في وضع الحومان. في هذا

المثال، يُعطى لارتفاع العنصر **div** ذو الصنف **info** وللفقرة الموجودة داخله عندما يكونان في

الحالة `..card: hover`.

استخدمنا في هذا المثال انتقالين، ويتحرك كل جزء فيه بطريقة مختلفة. وجود عناصر

تتحرك بسرعات مختلفة يمكن أن يضيف جاذبيَّةً لعملية الانتقال. يمكنك [مشاهدة هذا المثال](#)

كاملاً على [CodePen](#).

3. تطبيق انتقالات متعددة على عنصر واحد

بالإضافة إلى استخدام عدَّة انتقالات على عدَّة عناصر، يمكننا أيضًا استخدام عدَّة عمليات

انتقال على عنصر واحد.

يمكن استخدام ذلك عندما تحتاج إلى تغيير خلفيَّة عنصرٍ ما بشكل منفصل عن حدوده. إذ

قد يكون تطبيق انتقال واحد على جميع الخاصِّيات غير مناسب.

يمكننا تحقيق ذلك من خلال الجمع بين عدة انتقالات في نفس التعليمة، إذ تُفصل

الانتقالات بفواصل بينها.

إليك مثالاً السطر التالي:

```
transition: background 1s ease-out,
            border 0.5s linear;
```

يُطبَّق الانتقال الأول هنا على الخلفية فقط، أمَّا الثاني (بعد الفاصلة) فينطبق فقط على

الحدود. هذا يعني أنَّ حالة الحومان التي تُعَيَّر الخلفية ستستغرق ثانية واحدة، فيما سيستغرق

انتقال الحدود 0.5 ثانية.

4. تمرين

درسنا في هذا الفصل كيف يمكن إضافة عدَّة تأثيرات عبر انتقال واحد، وكيف يمكن

استخدام مجموعة من الانتقالات معًا. يجدر بك إلقاء نظرةٍ فاحصةٍ على الأمثلة التالية في

:CodePen

- المثال الأول: زر فني.

- المثال الثاني: بطاقة Cat Hover.

هل يمكنك استخدام هذه الأنواع من الانتقالات في مشروع تعمل عليه حاليًا؟

لقد غطينا الكثير حتى الآن. سوف نتحدث في الفصل التالي عن كيفية تطبيق هذه

الانتقالات باستخدام JavaScript.

9

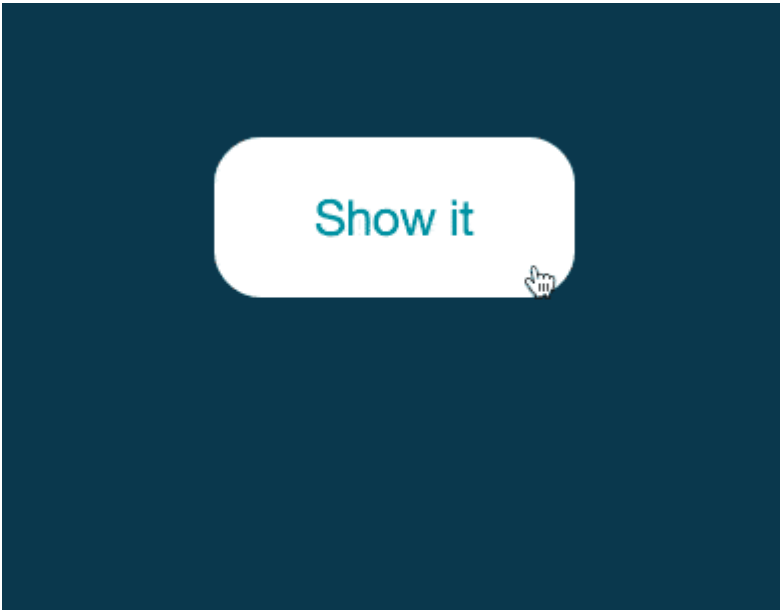
الانتقالات عبر JavaScript

استخدمنا في الفصول السابقة الخاصية **transition** للتحريك بين حالتين: حالة عدم الحومان (non-hover state) وحالة الحومان (hover state).

تحدّث تلك الانتقالات عند الحومان (تمرير مؤشر الفأرة) فوق العنصر. بيد أنّ هذه ليست الطريقة الوحيدة لبدء الحركة، سنشرح في هذا الفصل طريقتين لفعل ذلك عبر استخدام **JavaScript**.

1. إضافة أو إزالة الأصناف

أهم ما يميّز الانتقالات هي قدرتها على تحريك العنصر بين حالتين، لذا يمكننا إنشاء تلك الحالات على هيئة أصناف (classes) منفصلة ثمّ نضيف أو نزيل تلك الأصناف باستخدام **JavaScript** (مثال حي).



يتألف هذا المثال من زر ومحتوى مُتضمَّن في عنصر `div`. سيكون في البداية لحاوية

المحتوى صنف `hide` في CSS، الصنف `hide` له الخاصية `opacity: 0`.

لدينا أيضًا صنفًا ثانٍ في CSS يُسمَّى `show`. هذا الصنف لديه عتامة (`opacity`) مساوية

للقيمة 1.

عند النقر فوق الزر، يتراوح صنف العنصر `div` بين `hide` و `show`. ولأجل تحريكه، نطبِّق

عملية انتقال على العنصر `div` أيضًا. يمكنك مطالعة هذا المثال على [CodePen](#).

إذا كنت ترغب في التعمُّق أكثر في الموضوع، فأنصحك بقراءة المقال [Adding Appeal](#)

[to Your Animations on the Web](#).

ستتعلم في نهاية هذا الفصل كيفية تطبيق الانتقالات والحركات أثناء التمرير (scrolling).

2. التحكم في الانتقالات عبر JavaScript

يمكننا إجراء أكثر من عملية إضافة أو إزالة الأصناف. فيمكننا باستخدام JavaScript ضبط

خاصيات CSS مباشرةً على النحو التالي:

```
element.style.transition = `opacity 1s ease-out`;
```

في هذه الحالة، يمثِّل `element` عنصرًا حدَّدناه. على سبيل المثال، إذا كان لعنصر ما

المُعرَّف `js-show`، فيمكنك تطبيق الانتقال عليه باستخدام `getElementById`:

```
document.getElementById(`js-show`)
```

```
.style.transition = `opacity 1s ease-out`;
```

عندما تفعل ذلك، يجب ألا تنسَ تضمين بادئات المتصفحات (vendor prefixes) أيضًا.

سيُكتب ما سبق على النحو التالي:

```
document.getElementById(`js-show`)
  .style.webkitTransition = `opacity 1s ease-out`;
document.getElementById(`js-show`)
  .style.transition = `opacity 1s ease-out`;
```

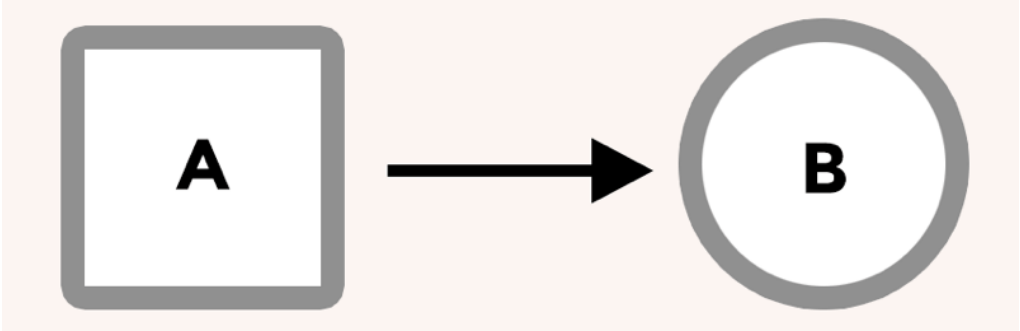
في هذا المثال، سُنطَبَّق `webkitTransition` على كل المتصفحات التي تقبل البادئة

-webkit- في CSS.

3. خلاصة الفصل

درسنا في الفصول السابقة الخاصية `transition`، وتعلّمنا كيفية استخدامها لجعل

المتصفح يُحرِّك عنصرًا من حالة إلى أخرى.



وقد تعلمنا في مشوارنا حتى الآن عدّة خاصيّات، مثل: `duration` (المدة) و `delay`

(التأخير) و `timing functions` (دوال التوقيت). من خلال الجمع بينها، يمكننا تصميم حركات

جذابة، بل وتطبيق عدّة انتقالات على العنصر نفسه لإنشاء حركات معقّدة.

وأخيرًا، تزودنا من هذا الفصل بتعلّم كيفية تطبيق هذه الانتقالات باستخدام JavaScript. الانتقالات ليست سوى جزء بسيط من موضوع التحريك في CSS. سنغطي في الفصول اللاحقة الخاصية `animation` بالتفصيل.

4. تمرين

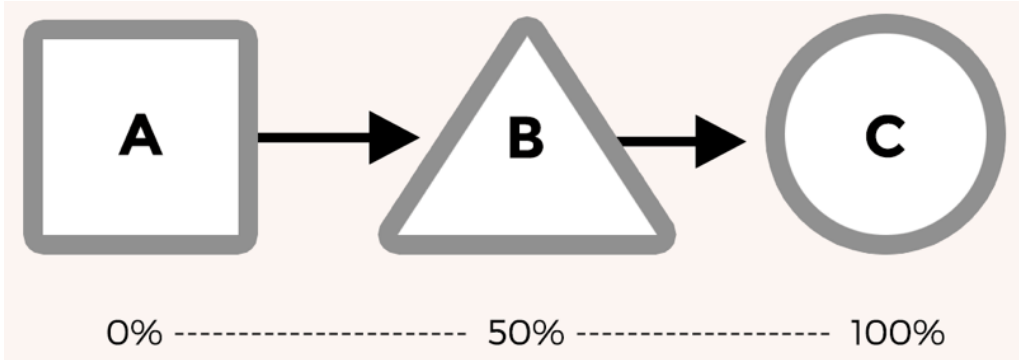
قبل أن ننتقل إلى الخاصية `animation`، خذ بعض الوقت في التفكير في كيفية استخدام الانتقالات.

هل يمكنك التفكير في طرائق يمكن أن تساعد بها الانتقالات على تسهيل التفاعلات، أو تغييرات الحالة على صفحاتك؟ كيف يمكنها أن تضيف جاذبيةً إلى موقعك؟ حاول أن تسأل نفسك مثل هذه الأسئلة وأن تبحث عن إجابات لها.

الحركات عملياً

10

انتهينا في الفصول الماضية من التعرّف بالتفصيل على الخاصية `transition`، وسننتقل في هذا الفصل والفصول اللاحقة للتعرّف على الخاصية `animation`. الصورة الآتية تمثل تحريك عنصر بنقله عبر عدة حالات متتالية تمثّل مراحل أو محطات الحركة:



1. علاقة تكاملية

تُطبّق الخاصية `animation` على العناصر تطبيقًا مشابهًا للخاصية `transition`. كما أنّها تحتاج إلى جزء ثانٍ، يسمى الإطارات المفتاحية (`keyframes`) التي تمثّل محطات الحركة.

```
.element {
  animation: ...
}
```

```
@keyframes animation-name {
  /* تكتب الإطارات المفتاحية هنا */
}
```

تتمثّل إحدى فوائد تعريف الإطارات المفتاحية `keyframes` في جزء منفصل من الشيفرة

في أنّها تتيح لنا إنشاء حركات قابلة لإعادة الاستخدام.

2. الخاصية animation

تُطبق الإطارات المفتاحية على العناصر باستخدام الخاصية animation وهي تشبه transition إلى حد بعيد، ولكنها تتميز عنها ببعض الخاصيات الإضافية. يمكن كتابة الخاصية animation بالشكل المختصر التالي (لصيغة المفصلة الكاملة، راجع قسم **البنية الرسمية** في صفحة الخاصية animation في **موسوعة حسوب**):

```
animation: change-background 4s linear infinite;
```

ستبدو الصياغة في حال كتابتها كخاصيات منفردة (أي animation-*) كما يلي:

```
animation-name: change-background;
```

```
animation-duration: 4s;
```

```
animation-timing-function: linear;
```

```
animation-repeat: infinite;
```

القيم المعطاة للخاصية transition هي عبارة عن خاصيات (properties)، مثل

background أو all، أما الخاصية animation فتُعطى اسم مجموعة الإطارات المفتاحية التي تُشكّل تسلسل الحركة.

تتميز الحركات ببعض الخاصيات التي ليست لدى الانتقالات. على سبيل المثال، يمكننا

برمجة الحركة لكي تُكرّر ذهابًا وإيابًا، بدلًا من إعادة تكرار الحركة من البداية كلّ مرّة.

3. الإطارات المفتاحية

مجموعة الإطارات المفتاحية keyframes في CSS هي عبارة عن سلسلة من النقاط

المرجعية على طول مسار الحركة تمثّل المحطات التي يجب على الحركة المرور عليها. ويكتب

كل إطار مفتاحي بنسبة مئوية من مدّة الحركة (طول الحركة).

أفضل طريقة لوصف هذا المفهوم هي باستخدام مثال توضيحي. لنبدأ بعنصر `div` على صفحة، والذي تتغير خلفيته بمرور الوقت. فهو يبدأ بخلفية زرقاء، ثم تتغير إلى خلفية برتقالية، ثم خلفية خضراء في النهاية.



إذا حاولنا أن نوضح لشخص ما كيف يحدث هذا التغيير في ألوان الخلفية بمرور الوقت، فقد نصف الأمر هكذا:

«البدء بخلفية زرقاء، ثم خلفية برتقالية في منتصف الطريق، ثم الانتهاء بخلفية خضراء.»

أو إذا أردنا أن نكون أكثر دقةً، فيمكننا استخدام النسب المئوية لشرح توقيت التغييرات:

«عند قطع 0% من الطريق (أي بداية الطريق)، فإنَّ الخلفية ستكون زرقاء اللون، ثمَّ عند

الوصول إلى 50% من الوقت، فستصير الخلفية برتقاليةً، وعند الوصول إلى نسبة 100% (أي

نهاية الطريق)، ستصير الخلفية خضراء.»

يمكننا تلخيص ذلك على النحو التالي:

0% Blue

50% Green

100% Orange

من خلال تحديد هذه النسب المئوية، فقد أنشأنا سلسلة من «النقاط المرجعية» التي تمثل

محطات ينبغي أن تمرَّ بها الحركة. كل ما نحتاج إليه الآن هو إخبار المتصفح بأنَّ هذه النسب هي

في الواقع إطارات مفتاحية keyframes ذات اسم محدد. والنتيجة هي:

```
@keyframes change-background {
  0% {
    background: blue;
  }
  50% {
    background: orange;
  }
  100% {
    background: green;
  }
}
```

لقد سمَّينا هذه الإطارات المفتاحية بالاسم change-background. سنستخدمها لاحقًا في

حركة ما عندما نريد تطبيق الإطارات المفتاحية على عنصر.

بقراءة الشيفرة من الأعلى إلى الأسفل، تصف النسب المئوية موضع الإطارات الرئيسية في

مسار الحركة. يمكننا أن نرى ذلك بشكل عملي في الصورة أسفله (مثال حي):


```
@keyframes change-background {
```

```
  0% {
```

```
    background: blue;
```

```
  }
```

```
  50% {
```

```
    background: orange;
```

```
  }
```

```
  100% {
```

```
    background: green;
```

```
  }
```

```
}
```



أثناء حدوث الحركة، يُنشئ المتصفح الإطارات المفتاحية الوسطية اللازمة للانتقال من كل لون من ألوان الخلفية إلى اللون التالي. بإخبار المتصفح أننا أردنا أن يبدأ العنصر **div** بلون معين، وأن عليه أن يأخذ لوناً آخر في منتصف الطريق، ثم ينتهي بلون ثالث، فإنه يتولى إنشاء الحركة اللازمة بين كل نقطة من تلك النقاط. يمكنك مراجعة هذا المثال على [CodePen](#).

لقد ذكرت في وقت سابق إمكانية استخدام الخاصية **animation-direction** لتحديد جهة تكرار الحركة أي هل سيكرر تنفيذ الحركة من البداية أم من النهاية.

رأينا في الصورة السابقة كيف يبدو إعادة تكرار الحركة من البداية وإليك كيف ستبدو الحركة عند إعادة تكرارها من النهاية (حالة الارتداد):

```
@keyframes change-background {
```

```
  0% {
```

```
    background: blue;
```

```
  }
```

```
  50% {
```

```
    background: orange;
```

```
  }
```

```
  100% {
```

```
    background: green;
```

```
  }
```

```
}
```



في هذه الحالة، غيّرت قيمة الخاصية `animation-direction` إلى `alternate`. يمكنك

الاطلاع على ذلك في [موقع CodePen](#).

4. البادئات

فيما سبق، كان من الضروري استخدام البادئة -webkit- في الخاصية `animation`. لن أضيفها إلى الأمثلة، ولكنّها ضرورية لعمل الحركات في بعض المتصفحات القديمة. أما حاليًا فلا حاجة لها لدعم المتصفحات الحديثة للحركات دون بادئات.

يمكنك استخدام خيار "Autoprefixer" في CodePen الموجود ضمن إعدادات CSS. بالنسبة للتطوير المحلي، فأنا أستخدم أداة Autoprefixer الخاصة بأداة البناء Gulp. (يشرح القسم «تحسين دعم المتصفحات لخاصيّات CSS وشيفرة JavaScript» في المقال [دليلك الشامل إلى أداة البناء Gulp](#) كيفية أتمتة هذه العملية تلقائيًا). أداة `Prefix Free` هي بديل جيد أيضًا.

5. تمرين

افتح [مثال إطارات المفاتيح هذا](#) وحاول تغيير الشيفرة. جرّب بهذا المثال وحاول تعطيله ثم إصلاحه. وحاول أيضًا ابتكار حركات مثيرة وخاصّة بك.

خاصيات الحركة

11

قبل شرح المزيد من الأمثلة عن الحركات، دعنا نلقي نظرةً على خاصيّات `animation` التفصيلية في CSS أولاً.

على غرار الخاصية `transition`، يمكن كتابة الخاصية `animation` باستخدام الصياغة المختزلة، أو يمكن تحديد أي من الخاصيّات *`animation-` فرادى.

1. الخاصية `animation-delay`

وكما في الخاصية `transition-delay`، يمكننا استخدام هذه الخاصية لتأخير الحركة مدّةً محدّدةً من الزمن قبل البدء. قد يكون ذلك مفيداً في المواقف التي تُشغّل فيها عدّة حركات. إذا كانت الحركات تتكرر دورياً، فلن يُطبّق التأخير إلا على المرة الأولى. كما لن يُطبّق التأخير سوى عند تطبيق الحركة على العنصر.

من الممكن إعطاء هذه الخاصية قيمةً سالبةً، مثل `-1s`؛ سيؤدي ذلك إلى بدء الحركة كما لو أنّها قد مرّت ثانية من زمنها سلفاً.

2. الخاصية `animation-direction`

الحركات عادةً ما تبدأ من 0% وتنتهي عند 100%، إذ تمثّل النسبة 0% نقطة بداية الحركة، وتمثّل النسبة 100% نقطة النهاية. تأخذ الخاصية `animation-direction` القيم `normal` و `reverse` و `alternate` و `alternate-reverse`، وتُستخدم للتحكم في اتجاه الحركة. تؤدي القيمة `reverse` إلى تشغيل الحركة تشغيلاً معكوساً من النهاية إلى البداية، أي من 100% إلى 0%، بينما يؤدي استعمال القيمة `alternate` إلى ارتداد الحركة أي تبدأ من البداية 0% إلى النهاية 100%، ثمّ تعود مرة أخرى إلى البداية 0%.

3. الخاصية animation-duration

تمثّل هذه الخاصية مدة الحركة الزمنية. على غرار `transition-duration`، تأخذ هذه الخاصية قيمةً مثل 1s، التي تمثّل ثانيةً واحدةً، أو 200ms، التي تمثّل مئتي ميلي ثانية.

4. الخاصية animation-fill-mode

افتراضياً، سيعود العنصر إلى حالته الطبيعية بعد انتهاء الحركة. يمكننا باستخدام الخاصية `animation-fill-mode` جعل العنصر يحافظ على القيم التي يضبطها أول أو آخر إطار مفتاحي.

يؤدي استخدام القيمة `forwards` إلى توقيف الحركة عند الإطار المفتاحي الأخير. فيما تعود القيمة `backwards` إلى الإطار المفتاحي الأول عندما ينتهي تنفيذ الحركة. يمكنك مطالعة هذا المثال على Hop.ie. تُشغّل الحركة مرةً واحدةً، وتنتهي عند الإطار الأخير وذلك بسبب استخدام القيمة `forwards`.

5. الخاصية animation-iteration-count

تمثّل هذه الخاصية عدد مرات تكرار تنفيذ الحركة. افتراضياً، سننقذ الحركة مرةً واحدةً. يمكنك تحديد عدد معيّن مع هذه الخاصية لتكرار الحركة بقدره، أو استخدام القيمة `infinite` لتكرار تنفيذ الحركة إلى الأبد.

6. الخاصية animation-name

تشير الخاصية `animation-name` إلى الإطارات المفتاحية المرتبطة بالحركة. على سبيل المثال، إذا ضبطت قيمة الخاصية `animation-name` إلى `foo`، فسُتستخدم مجموعة الإطارات المفتاحية ذات الاسم `foo` المُعرّفة على النحو التالي:

```
@keyframes foo {
  ...
}
```

7. الخاصية animation-play-state

إذا احتجت إلى إيقاف الحركة مؤقتاً أو استئنافها، فستتيح لك هذه الخاصية القيام بذلك. تأخذ هذه الخاصية إحدى القيمتين `running` (القيمة الافتراضية) أو `paused`. يمكن تعيين قيمة هذه الخاصية باستخدام `JavaScript`.

8. الخاصية animation-timing-function

تأخذ هذه الخاصية القيمة نفسها التي تأخذها الخاصية `transition-timing-function`، ولكن مع اختلاف طفيف. فبينما تُطبّق دوال التوقيت، مثل `ease-out`، على عملية الانتقال (`transition`) بأكملها، تُطبّق دالة التوقيت هنا أثناء الانتقال بين الإطارات المفتاحية للحركة.

ذلك يعني أنّ الإطارات المفتاحية التالية ستشهد بداية سريعة للحركة ثمّ تتباطأ الحركة مع الاقتراب من الإطار 50%، ثمّ تُسرّع من جديد قبل أن تتباطأ مع الاقتراب من الإطار النهائي 100%.

```
@keyframes foo {
  0% {
    /* ease-out يجعله الدالة ثمّ بسرعة يبدأ
    */
    /* يتباطأ مع الاقتراب من 50% */
  }
  50% {
    /* مجدداً، يبدأ بسرعة ثمّ يتباطأ مع الاقتراب من 100% */
  }
  100% {
    /* النهاية */
  }
}
```

قد يكون ذلك مربكاً. أفضل عند إنشاء الحركات عبر الإطارات المفتاحية العمل بدالة التوقيت linear، ثمّ أتحكم بعدئذٍ في وتيرة الحركة باستخدام الإطارات المفتاحية. رغم ذلك، فإنّ **دوال التوقيت cubic-bezier** يمكن أن تُنشئ بعض التأثيرات الرائعة عند استخدامها مع الحركات، لذلك أنصحك بتجريبها.

9. استخدام دوال التوقيت ضمن الإطارات المفتاحية

تجدر الإشارة إلى أنّه عند تحديد دالة التوقيت الخاصّة بالحركة، فسُتطبّق دالة التوقيت على كل إطارات الحركة المفتاحية.

هذا يعني أنّه إذا أردت تحديد أربعة إطارات مفتاحية، فسُتطبّق دالة التوقيت على كلّ منها. فمثلاً، سوف تُبطئ دالة التوقيت ease-out الحركة عند الاقتراب من نهاية كل إطار مفتاحي.

لهذا السبب، عادةً ما يُفضّل استخدام دالة التوقيت linear، والتحكم في السرعة على

أساس كل إطار على حدة:


```
@keyframes my-animation {
  0% {
    ...
    animation-timing-function: linear;
  }
  50% {
    ...
    animation-timing-function: ease-out;
  }
}
```

في هذه الحالة، سيكون النصف الأول من الحركة خطياً، بينما يستخدم النصف الثاني دالة

التوقيت `ease-out`.

10. تمرين

لقد أنشأتُ مثالاً توضيحياً على [CodePen](#). الخاصيات مُدرجة في CSS، لذا وحاول تغيير

بعض هذه الخاصيات لمعرفة ما سيحدث.

الإطارات المفتاحية عملياً

12

اطَّلعنا في الفصل السابق على خاصيّات الحركات (الخاصيّات التفصيلية للخاصيّة animation)، ورأينا كيف أنّها تعتمد على **الإطارات المفتاحيّة** (keyframes). سنلقي في هذا الفصل نظرةً أعمق على الإطارات المفتاحية.

1. أساسيات لابدأ منها

قبل الخوض في مثال عملي، هناك بعض الأمور أود تغطيتها بخصوص الإطارات المفتاحيّة. الأمر الأول هو صياغة بديلة قد تراها في بعض الشيفرات، وتستخدم الكلمات المفتاحية from و to.

```
@keyframes name {
  from {
    ...
  }
  to {
    ...
  }
}
```

هذه الطريقة البسيطة يمكن أن تكون بديلاً عن كتابة 0% و 100%، وقد تكون أسهل للفهم، ومناسبة للحركات البسيطة.

قد تلاحظ أحياناً استخدام أكثر من نسبة مئوية واحدة على السطر نفسه. هذه الصياغة تجعل الحركة تتوقف لفترة من الزمن، أو تستقر على حالة معيّنة. إليك الشيفرة التالية مثالاً:

```
@keyframes name {
  0%, 20% {
    opacity: 0;
  }
```

```

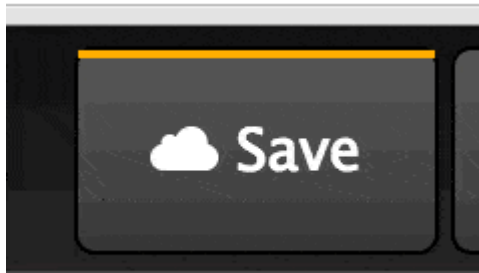
100% {
  opacity: 1;
}
}

```

في هذا المثال، سيبدأ العنصر بعتامةٍ (*opacity*) تساوي 0، وسيظل غير مرئي حتى انقضاء 20% من وقت الحركة، ثم ستبدأ قيمة العتامة بالتحرك تجاه القيمة 1. سنستخدم هذا في [الفصل اللاحق](#) لمزامنة الحركات.

2. مثال: تأثير اهتزاز الزر Save

هل تذكر مثال الزر "Save" من الفصل الأول؟ دعنا ننظر مجددًا في هذا المثال، وننظر في كيفية استخدام الإطارات المفتاحية *keyframes* إلى جانب الخاصية *animation* لإنشاء تأثير الاهتزاز (مثال حي).



قبل إضافة أي حركة، فقد نسقتُ الزر ليبدو مثل زر CodePen، إذ جعلت له إطارًا برتقاليًا في الأعلى، وتدرجًا داكنًا، وجعلت لون النص فيه أبيضًا. وقد استخدمت التموضع المطلق (*absolute positioning*) في المثال للتأكد من وضع الزر في منتصف الشاشة.

أول ما أفعله عمومًا هو تطبيق الخاصية `animation` على العنصر على النحو التالي:

```
button {
  animation: wiggle 2s linear infinite;
}
```

في هذه الحالة، سُنطَبِّق مجموعةً من الإطارات المفتاحية تحمل الاسم `wiggle`، وستمتد الحركة لمدةً ثانيتين باستخدام دالة التوقيت `linear`. كما سنستخدم أيضًا قيمةً جديدةً، وهي `infinite`.

القيمة `infinite` هنا تخص الخاصية `animation-iteration-count`. افتراضيًا، ستتكرّر الحركة مرةً واحدةً فقط، ولكن يمكننا أن نكررها عددًا محددًا من المرات أو إلى الأبد. في مثالنا هذا، ستتكرّر الحركة عددًا غير محدودٍ من المرات.

بعديًا، سنحدّد الإطارات المفتاحية الخاصة بحركة «الاهتزاز». وها هي النتيجة:

```
@keyframes wiggle {
  0%, 7% {
    transform: rotateZ(0);
  }
  15% {
    transform: rotateZ(-15deg);
  }
  20% {
    transform: rotateZ(10deg);
  }
  25% {
    transform: rotateZ(-10deg);
  }
  30% {
```

```
transform: rotateZ(6deg);
}
35% {
transform: rotateZ(-4deg);
}
40%, 100% {
transform: rotateZ(0);
}
}
```

لقد أعطينا المتصفح سلسلة من النقاط المرحليّة (way-points) ليُحرّك العنصر بينها. يُدوّر

الزر "Save" عند كل نقطة على المحور z. تبدأ الزوايا واسعةً، ثمّ تصغر خلال الحركة.

إليك كيف تميل الحركة الزر للخلف وللأمام مع مرور الوقت (مثال حي):

```
@keyframes wiggle {
  0%, 7% {
    transform: rotateZ(0);
  }
  15% {
    transform: rotateZ(-15deg);
  }
  20% {
    transform: rotateZ(10deg);
  }
  25% {
    transform: rotateZ(-10deg);
  }
  30% {
    transform: rotateZ(6deg);
  }
  35% {
    transform: rotateZ(-4deg);
  }
  40%, 100% {
    transform: rotateZ(0);
  }
}
```

A dark grey rectangular button with a thin orange border at the top, containing the word "Save" in white text.

يمكننا أن نرى أنَّ المتصفح ينشئ الخطوات الفاصلة بين الإطارات المفتاحية. ودون الاعتماد على دوال التوقيت المعقدة، نجحت هذه الحركة في إضافة الكثير من التميُّز إلى الزر. يمكنك مطالعة مثال زر "Save" كاملاً على [CodePen](#).

3. تمرين

لقد أنشأت مشروع CodePen جديد يتضمن عنصرًا متحركًا واحدًا يستخدم الخاصية `animation-timing-function` داخل الإطارات المفتاحية، مع سلسلة من الإطارات المفتاحية التي تنشئ حركةً معقدةً نسبيًا.

ما الذي سيحدث عند حذف بعض الإطارات؟ أو عند تغيير النسب المئوية؟ هل يمكنك جعل المكعب يتحرك تحركًا مختلفًا؟ اختر شكلًا بسيطًا، وحاول إضفاء بعض الحيوية عليه!

الحركات المتعددة المتزامنة

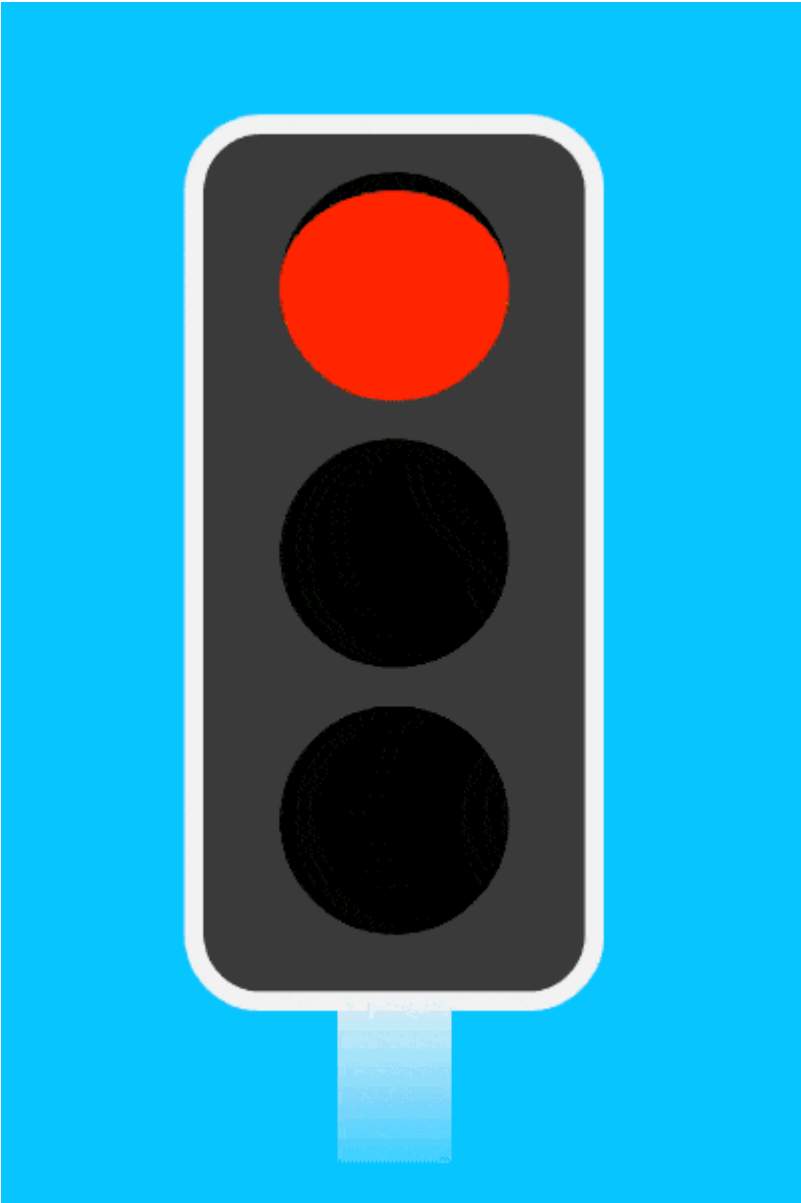
13

سنتحدّث في هذا الفصل عن كَيْفِيَّة استخدام عدَّة مجموعات من **الإطارات المفتاحية** (keyframes) تعمل بالتزامن لتوليد حركة مُعقَّدة مؤلَّفة من عدَّة حركات.

1. إشارة المرور

قد نرغب أحياناً في مزامنة عدَّة حركات بشكل يكون لكل حركة توقيتها الخاص. ومن الأمثلة الجيدة على هذا إشارة المرور.

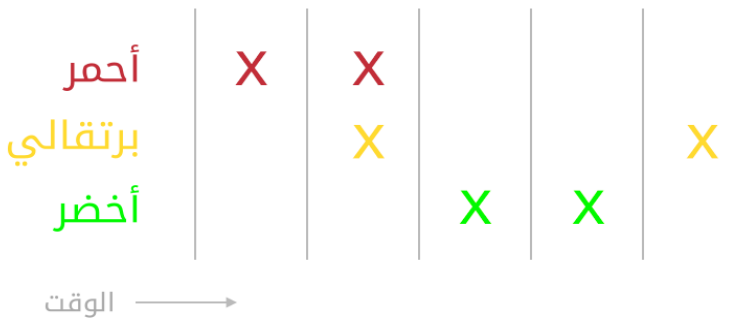
لدينا هنا نموذج بسيط لإشارة المرور في المملكة المتحدة (**مثال حي**):



لدينا ثلاثة مصابيح وكل منها يضيء وينطفئ بنمط خاص به. يمكننا إنشاء هذا عن طريق إعطاء كل مصباح حركة خاصة به.

```
.red {
  animation: red 10s linear infinite;
}
.amber {
  animation: amber 10s linear infinite;
}
.green {
  animation: green 10s linear infinite;
}
```

لدينا ثلاثة حركات، واحدة لكل مصباح. مدّة كل حركة متساوية للجميع، لذلك ستبقى الحركات متزامنة أثناء تكرارها الدوري. سنحتاج لأجل المتابعة إلى تحديد الإطارات المفتاحية. أثناء إنشاء هذا المثال، وجدت أنّه من المفيد النظر إلى المصابيح على أنّها شبكة (grid). تحدث الحركات من اليسار إلى اليمين، إذ أنّ كل مصباح سيكون إما في حالة انطفاء أو إضاءة في كل لحظة.



تنقسم الشبكة إلى خمسة أعمدة وهذا يعني أنه سنتعامل مع «خُمس المدة الزمنية» أي كل منها تستغرق 20% من مدّة الحركة، لذا سننشئ مجموعات من الإطارات المفتاحيّة اعتمادًا على تلك الأحماس.

بأخذ كل مصباح على حدة، يمكننا أن نبدأ بالمصباح الأحمر والذي سيكون مضيئًا في الخُمس الأول والثاني، ثم سينطفئ في بقية مراحل الحركة. الإطارات المفتاحيّة الناتجة هي كالتالي:

```
@keyframes red {
  0% {
    background: black;
  }
  2%, 40% {
    background-color: red;
  }
  42%, 100% {
    background: black;
  }
}
```

لقد أضفت فجوةً بقيمة 2% في البداية، وجعلت الجزء الثالث من الحركات يبدأ عند اللحظة 42%، لأنّ هذا سيضيف تأثير التلاشي على إشارة المرور. مثل هذه الأشياء الدقيقة هي التي تصنع الفرق (-).

مع إضاءة المصباح الأحمر، يُنتظر أن يضيء المصباح البرتقالي على الشبكة. سيكون المصباح البرتقالي مطفئًا في البداية، ثمّ سيضيء خلال الخُمس الثاني، ثمّ يطفأ خلال الخُمسين التاليين، ثمّ يضيء في الخمس الأخير. الإطارات المفتاحيّة لهذا المصباح هي:

```
@keyframes amber {
  0%, 20% {
    background: black;
  }
  22%, 40% {
    background: #FF7E00;
  }
  42%, 80% {
    background: black;
  }
  82%, 100% {
    background: #FF7E00;
  }
}
```

أما المصباح الأخضر، فسيكون مطفئًا خلال الخمسين الأوليين، ثمّ يضيء خلال الخمسين

التاليين، ثمّ يطفأ في الخمس الأخير.

```
@keyframes green {
  0%, 40% {
    background: black;
  }
  42%, 80% {
    background: green;
  }
  82%, 100% {
    background: black;
  }
}
```

يمكنك مطالعة المثال كاملاً من هنا.

2. مراجع أخرى

لمعرفة المزيد عن صياغة الإطارات المفتاحية، راجع المقال «صياغة الإطارات

المفتاحية للحركات».

3. تمرين

قد يبدو مثال إشارة المرور غريبًا بالنسبة لك لأنه يتبع نموذج المملكة المتحدة. هل يمكنك

تعديله ليتناسب مع نموذج إشارات المرور في بلدك؟ حسنًا، ليكن ذلك تمرين هذا القسم.

خلاصة ما تعلمناه عن الحركات

14

لقد غطينا الكثير من التفاصيل في الفصول السابقة وأرجو أن يكون كل شيء واضحًا الآن. عندما بدأت تعلم إنشاء الحركات في CSS، لم تكن مفاهيم التحريك والإطارات المفتاحية (keyframes) واضحة لي. إذا شعرت أن مفاهيم التحريك غير واضحة حتى الآن، فلا تقلق. استمر في الدراسة والتعلم والتجريب، وستتضح حيل ومفاهيم التحريك في HTML و CSS شيئًا فشيئًا.

سنلخص في هذا الفصل ما تعلمناه في الفصول السابقة، لكن سنعود أولاً إلى التمرين الذي رأيناه في الفصل الماضي.

1. تمرين: إشارات المرور

يُفترض أن يكون التمرين سهلًا طبعًا على افتراض أنك تعرف كيف تحله. لقد أنشأت نسخة مُحدّثة من مثال إشارات المرور وفق نموذج المملكة المتحدة، إذ غيّرت التسلسل هذه المرة لإزالة مرحلة «الأحمر + البرتقالي» ليوافق نظام إشارات المرور الأمريكية. يمكنك الاطلاع عليه من هنا. لقد جعلت نظام الألوان يطابق شكل إشارات المرور الأمريكية.

2. موجز: الحركات

سنلقي في هذا القسم نظرةً على الخاصية `animation`، وكيف تعمل مع الإطارات المفتاحية (keyframes).

رغم أن الخاصية `animation` تعمل بطريقة مشابهة للخاصية `transition`، إلا أنها تختلف عنها قليلًا. ففي حين أن الانتقال (transition) لن يحدث إلا عند تغيير العنصر، فإن الحركات (animations) يمكن أن تبدأ مباشرة.

باستخدام مختلف الخاصيّات، يمكن تكرار الحركات عددًا معيّنًا من المرات (أو إلى الأبد)، كما يمكن أن تبدأ بتأخير سالب، وهذا سيبدأ الحركة بعد أن تكون قد بدأت معالجتها تلك المدة الزمنيّة.

افتراضيًا، تبدأ الحركة من البداية إلى النهاية، ثمّ تقفز إلى حالتها الافتراضيّة. يمكننا تجميد الحركة في حالتها النهائيّة بإعطاء الخاصية `animation-direction` القيمة `forwards`. تستخدم الحركات **دوال التوقيت** مثل الانتقالات `ببدا` دوال التوقيت تنطبق على كل إطار رئيسي على حدة، وليس على كامل مجموعة الإطارات المفتاحيّة. يمكنك أيضًا تحديد قيمة `animation-timing-function` داخل الإطار المفتاحي لتتحكم تحكّمًا دقيقًا في الحركة.

أخيرًا، يمكن صياغة الحركات صياغةً مختصرةً كما هو الحال مع الانتقالات:

```
animation: keyframe-name 2s forwards linear;
```

١. الإطارات المفتاحيّة

من الضروري أن تشير كل حركة إلى مجموعة من الإطارات المفتاحيّة (keyframes). هذه الإطارات هي عبارة عن سلسلة من النسب المئوية التي تصف كل «مرحلة» من مراحل الحركة. سيبدأ المتصفح الفجوات تلقائيًا.

الإطارات المفتاحيّة لها اختصاراتها الخاصة (to و from)، والتي يمكنك استخدامها عندما تريد الاكتفاء بالانتقال من حالة إلى أخرى.

يمكن أن يؤدي سرد النسب المئويّة بجانب بعضها بعضًا إلى «توقف» الحركة في تلك المرحلة.

أخيرًا، من الممكن حذف الإطار المفتاحي 0%، وسيفترض المتصفح حينها أنّ تنسيق العنصر هو الحالة الابتدائية. على سبيل المثال، لجعل عنصرٍ ما يتلاشى (fade away)، لا يلزم بالضرورة أن تكون قيمة عتامته (opacity) مساويةً للقيمة 1 في البداية (على افتراض أنّ العنصر مرئي):

```
@keyframes name {
  100% {
    opacity: 0;
  }
}
```

3. تجميع الحركة

عندما نريد استخدام الحركة، سيكون لدينا دائمًا الجزءان التاليان:

```
.element {
  animation: keyframe-name ...
}
@keyframes keyframe-name {
  ...
}
```

4. تمرين

في هذه المرحلة، يجب أن تكون قادرًا على التفريق بين خاصيتي الحركة (animation) والانتقال (transition).

ألق نظرة على أمثلة [Principles of Animation for the Web](#). كل تلك الأمثلة مصنوعة حصريًا باستخدام HTML و CSS، وذلك باستخدام إطارات الحركة المفتاحية. حاول أن تنسخها، ثمّ تجرّبها وتعُدّل عليها.

رواية القصص عبر الحركات

15

تحدثنا في الفصول الماضية عن الخاصيتين **transition** و **animation**، لذا هيا بنا نطبّق

ما تعلمناه في هذا الفصل لإنشاء حركةٍ مع تأثير الحومان (hover effect).

1. الصور الرئيسية

تُفضّل العديد من المواقع عرض صورة كبيرة تجذب الانتباه في أعلى الصفحة الرئيسية.

تسمى أحياناً الصورة الرئيسية (hero image)، وعادةً ما تمتد على كامل عرض الصفحة.

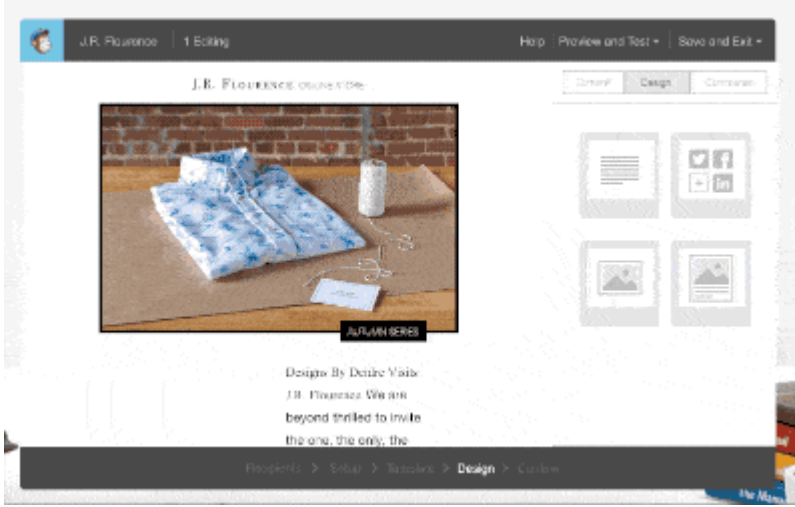
من الأمثلة الجيدة التي وجدتها في الآونة الأخيرة هي صفحة الهبوط الخاصة بموقع إطار

العمل Fabric، إذ توضّح عبر حركة بسيطة في CSS ووظيفة Fabric كإطار عمل

تركيب (modular) راجع [صفحة المثال](#) لتراه حيّاً.



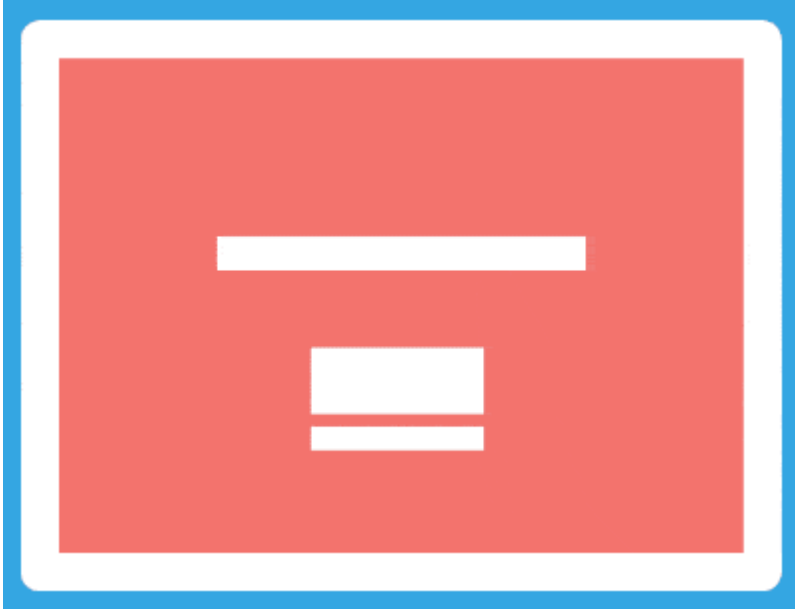
مثال آخر جيد نجده في الصفحة الرئيسيّة لموقع **Mailchimp**، إذ تحكي الصورة الرئيسيّة قصةً توضّح كيفية إنشاء رسائل البريد الإلكتروني.



نجد في هذين المثالين وغيرهما من الأمثلة أنّه جرى استخدام الحركات لشرح خدمات الموقع ببساطة.

2. مثال: تمرير الخلفية

سننشئ مثالاً خاصاً بنا مشابهاً لما قد رأيته آنفًا. في هذا المثال، صمّمت رسومات لصفحة تتحرك لأعلى وأسفل الشاشة.



تتوقف الحركة مؤقتًا لإضفاء طابع التفاعليّة وتظهر رسالة عندما يمر مؤشر الفأرة فوق الشاشة. وقد استخدمت كل من الحركات والانتقالات لتحقيق هذا التأثير. يمكنك الاطلاع على كامل هذا المثال من [هنا](#).

1. الجزء الأول: تحريك الخلفيّة

لإعداد هذا المثال، سنبدأ بعنصر HTML حاوي:

```
<div class="screen"></div>
```

يمكننا أن نجعل الصنف screen يبدو وكأنه شاشة أو iPad باستخدام التنسيقات التالية:

```
.screen {
  background: #e25865 url(//cssanimation.rocks/screen/
  images/screen_bg.png) no-repeat top center;
```

```
background-size: 100% auto;
border: 2em solid #fff;
border-radius: 1em;
width: 40em;
height: 30em;
}
```

لقد أعدنا بعض التنسيقات التي تُحدّد الحجم والحدود، وتعين صورة الخلفية.

يعتمد التأثير الذي نريد إنشاءه على تحريك صورة الخلفية الأطول من الشاشة والتي قيمة

الخاصية `background-size` الخاصة بها تساوي `100% auto`. هذا يعني أنّ الخلفية ستناسب عرض الحاوية، ولكنها ستكون أطول منها.

بتحديد صورة الخلفية المراد تحريكها، يمكننا الآن كتابة الإطارات المفتاحية التي ستجعلها

تبدو وكأنّ شخصًا ما يمرّ صفحة الويب للأسفل والأعلى:

```
@keyframes scroll {
  0%, 10% {
    background-position: 0 0;
  }
  20%, 30% {
    background-position: 0 -22em;
  }
  50%, 60% {
    background-position: 0 -44em;
  }
  90%, 100% {
    background-position: 0 0;
  }
}
```


الخاصية التي تُحرِّكها هي `background-position`. يمكننا عبر هذه الخاصية تحريك الخلفيّة للأعلى وللأسفل، إذ تبدأ من الموضع 0 0، ما يعني أنّ المسافة من اليسار، ومن الأعلى ستساويان الصفر.

جعلنا في الإطارات التالية الخلفيّة تتحرك إلى الأسفل (انطلاقاً من الأعلى) بمقدار 22em، ثمّ 44em، ثمّ تعود إلى أعلى الصفحة. سنستخدم الخاصية `animation` لتطبيق هذا على العنصر `.screen`.

```
.screen {
  animation: scroll 5s infinite
            cubic-bezier(.52, -0.39, .3, 1.43);
}
```

في المثال أعلاه، طبّقنا مجموعة من الإطارات المفتاحيّة التي تسمى `scroll`، وطلبنا منها أن تستمر 5 ثوانٍ، وتكرر باستمرار، وتستخدم **دالة توقيت** `cubic-bezier`. في هذه الحالة، تضيف الدالة `cubic-bezier` تأثير اهتزاز للحركة، وستبدو الحركة بدونها أقلّ واقعيّة.

يمكنك أن تجد دالة `cubic-bezier` المستخدمة في المثال في الموقع `cubic-`

bezier.com. إذا لم تزر هذا الموقع حتى الآن، فإني أوصي بزيارته الآن وبشدّة!

ب. الجزء الثاني: إضافة انتقال الحومان

يُستحسن إيقاف أو تجميد الحركة عند انتهائها، أو عندما تريد أن يركز القارئ على محتوى آخر. يمكن أن يتسبّب التحريك المستمر في تشتيت القارئ، لذلك دعنا نستفيد من الخاصية `animation-play-state` لإيقاف الحركة مؤقتاً عند الحومان على العنصر.

```
.screen:hover {
  animation-play-state: paused;
}
```

هذا يعني أنه عندما يُحوّمْ مؤشر الفأرة فوق موضع الحركة، تتوقّف الحركة مؤقتًا، وعندما

يبتعد المؤشر، تستأنف عملها وتعود الخاصية `animation-play-state` لقيمتها الافتراضية: `playing`.

يمكنك تحقيق ذلك باستخدام JavaScript أيضًا. يمكن أيضًا استخدام JavaScript

لتعطيل الحركة عندما يتفاعل المستخدم مع جزء آخر من الصفحة، أو عندما يُمرّر (scroll) الصفحة. سنلقي نظرة على كيفية تمكين الحركة عند التمرير لاحقًا.

ج. الجزء الثالث: إضافة رسالة

يمكننا إضافة المزيد مثل عرض رسالة خاصّة بالانتقال عندما يُحوّمْ المستخدم فوق

العنصر. سنحتاج لإنشاء ذلك إلى كتابة بعض شيفرات HTML:

```
<div class="screen">
  <div class="message">Hover message!</div>
</div>
```

في قسم [CSS من CodePen](#)، وضعنا هذه الرسالة في منتصف العنصر `screen`، وجعلناها

غير مرئية.

```
.message {
  /* ... تنسيق الرسالة ... */
  opacity: 0;
  transition: all 0.4s ease-out;
}
```

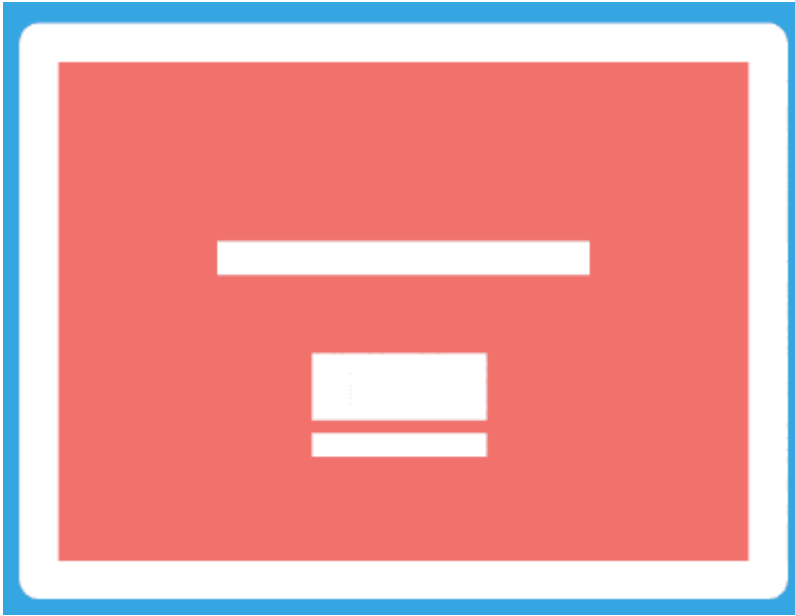
يمكننا بعد ذلك إظهارها أثناء الحومان باستخدام الانتقال:

```
.screen:hover .message {
  opacity: 1;
}
```

نظرًا لأننا أضفنا الخاصية **transition** إلى تنسيق العنصر `message`، فستحدث تأثيرات

الحركة عندما يحوم مؤشر الفأرة فوق العنصر، وكذلك عندما يغادرها. تبدو تأثيرات الحركة

والانتقال عند تجميدها كما يلي (مثال حي):



3. خلاصة الفصل

دمجنا في هذا الفصل الحركة والانتقال معًا لإنشاء تأثير يمكن أن يكون مفيدًا في تصميم صفحات الهبوط، أو سرد قصص المنتجات أو إخبار الزائر الغرض من موقعك أو كيفية استخدامه. استخدمنا الخاصية `animation-play-state` للتأكد من توقف الحركة عندما نريد.

4. تمرين

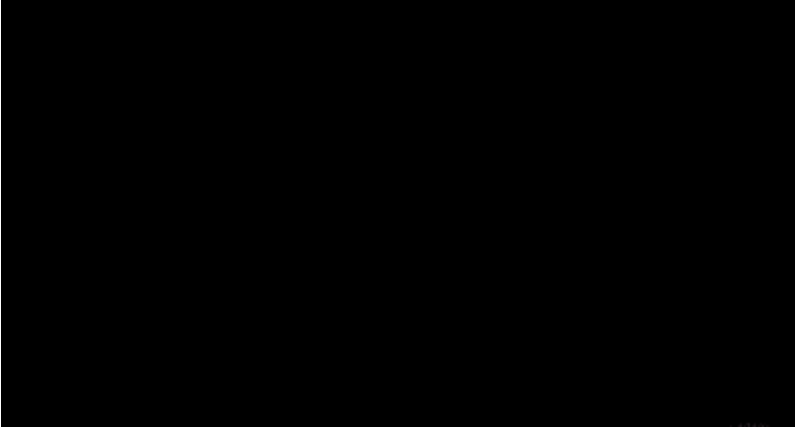
توقّف للحظة وفكّر في كل ما تعلمناه حتى الآن. لقد غطينا الكثير من المواضيع، أليس كذلك؟ يُعدُّ الجمع بين الحركة والانتقال وسيلةً ممتازةً لإضفاء الحيويّة على الصفحات. عند التفكير في كيفية تطبيق هذه التقنيات في مشاريعك، فكّر في كيفية التحكم فيها أيضًا. حاول أن تعرف متى تكون الحركة مفيدةً للمستخدمين، ومتى تكون عائقًا أو مصدر إزعاج لهم. فعليك أن تعرف متى عليك استعمال الحركات، ومتى عليك تجنبها.

حرب النجوم!

16

سنطبّق ما تعلّمناه في الفصول السابقة، وسنصنع تحريكًا متجهيًا (أي SVG) مرّحًا. سنبنّي

عنوان فيلم حرب النجوم من إعلان "The Force Awakens" (للمصدر).



يجمع هذا المثال بين التحريك وبين بعض خاصيّات CSS الأخرى، خصوصًا الخاصية

`transform` ودوال تحويلها `scale` و `translate` و `rotate`.

1. `transform`: ليست من خاصيّات الحركات

رغم أنّها تبدو وكأنّها من خاصيّات الحركات، إلا أنّ الخاصية `transform` تُستخدَم في

تحديد موضع، أو انحراف، أو حجم عنصر ما. يمكننا استخدام هذه الخاصية لإنشاء تأثيرات

رائعة، ولكن نحتاج للقيام بذلك إلى تحويل (`transform`) خاص لكل إطار مفتاحي

أو حالة تحركها.

2. الخاصية transform ودوالها: scale() و translate() و rotateY()

بإمكاننا تصغير العناصر أو تكبيرها باستخدام `scale`. ويمكننا باستخدام `translateZ` نقل العناصر على المحور Z، وهو المحور الذي يمثله رسم خط ينطلق من موضع نظر الزائر، وبيجه إلى الشاشة.

في هذا المثال، سنستخدم `scale` و `translateZ` معًا لاستحداث شعور بأنّ الكلمات تطير في الفضاء. بعد ذلك، سنستخدم `rotateY` لتدوير أحرف سطر الوصف. سيتطلب الدوران حول المحور Y قيام المتصفح بالرسم ثلاثي الأبعاد.

3. SVG و HTML و CSS

استعدادًا لهذا المثال، أنشأت ملفين من النوع `SVG` لأجل الجزئين `Star` و `Wars` من الشعار. لا تتردد في تنزيلهما واستخدامهما. إليك شيفرة `HTML` الخاصّة بالمثال:

```
<div class="starwars-demo">
  
  
  <h2 class="byline" id="byline">The Force
Awakens</h2>
</div>
```

لقد استخدمنا صورة ثابتة للنجوم في الخلفيّة. لم أتمكن من العثور على الخط الذي استُخدم في الإعلان الأصلي، لذلك استخدمت الخط "Lato". استخدمت التموضع المطلق (absolute positioning) لوضع المحتوى في منتصف الشاشة. سنحصل في البداية على الصورة التالية:



4. تحريك الكلمتين Star و Wars

نريد أن يظهر النص الكبير تدريجيًا، أي يبدأ بحجم كبير، ثمّ يصغر بمرور الوقت. وهذه فرصة جيدة لاستخدام الدالة `scale()`. سنستخدمها على الكلمة "Star" في الإطارات المفتاحيّة التالية:

```
@keyframes star {
  0% {
    opacity: 0;
    transform: scale(1.5) translateY(-0.75em);
```



```

}
20% {
  opacity: 1;
}
89% {
  opacity: 1;
  transform: scale(1);
}
100% {
  opacity: 0;
  transform: translateZ(-1000em);
}
}

```

هناك خاصيتان تتغيران خلال مسار هذه الحركة وهما **opacity** و **transform**. يبدو النص شفافاً في البداية بتغيير عتامته (**opacity**)، ثم يتلاشى في النهاية لكي نتمكن من إعادة تنفيذ الحركة.

يبدأ التحويل عن طريق تحديد الحجم عند القيمة 1.5. هذا يعني أن الحجم الأولي للنص سيكون أكبر بنسبة 150% من الحجم العادي. عند النقطة 89%، سنُعيّن الخاصية **transform** إلى القيمة **scale(1)**. هذا يعني أنه بين اللحظتين 0% و 89%، سينتقل الحجم من 150% إلى 100%.

يؤدي التحويل **transformZ** الأخير إلى تكبير الكلمتين بسرعة.

بشكل مماثل، يمكننا تطبيق الإطارات المفتاحية على الكلمة "Star":

```

.star {
  animation: star 10s ease-out infinite;
}

```

استخدمنا مجموعة مماثلة من الإطارات المفتاحية مع الكلمة "Wars".

5. لنجعلها ثلاثية الأبعاد

يتطلب استخدام التحويلات ثلاثية الأبعاد في CSS، سواءً كانت بالتحويل على المحور Z، أو الدوران حول المحورين Y و Z، أن نضع مرحلةً ثلاثية الأبعاد. وهذا يعني، في اصطلاح HTML، إنشاء حاوية (container)، وإخبار المتصفح بالحاجة إلى إنشاء بعض الحركات ثلاثية الأبعاد.

يمكننا فعل ذلك عن طريق إضافة الشيفرة التالية إلى العنصر `div` ذي

الصف `starwars-demo`:

```
.starwars-demo {
  perspective: 800px;
  transform-style: preserve3d;
}
```

تخبر هاتان الخاصيتان المتصفح بأنه ينبغي على أبناء الحاوية أن يتموضعوا وفق شكل

ثلاثي الأبعاد، بدلاً من وضعها في مستوي مسطح. يمكنك أن تجد المزيد من التفاصيل عن هذه الخاصية في توثيقها في موسوعة حاسوب.

ثانياً، تخبر الخاصية `perspective` المتصفح بمدى «عمق» المشهد. في هذا المثال، جعلناها

تساوي القيمة `800px`. تخلق القيم الأصغر تأثيرات «متطرفة» لأن المشهد سيكون أقصر.

سُنسّق الآن سطر الوصف.

6. تحريك الشعار The Force Awakens

تدور أحرف الشعار "The Force Awakens" في مكانها. يمكننا إنشاء هذا التأثير باستخدام التحويل `rotateY`. في هذا المثال، وضعنا كل حرف داخل عنصر من النوع `span` حتى نتمكن من تطبيق الحركة على كل حرف على حدة. إحدى المشكلات التي اكتشفناها سريعًا هي أنه لا توجد طريقة مباشرة لتحريك كل حرف في السطر. الحل الذي بدا لي هو وضع كل حرف يدويًا داخل وسم `span`. وقد نجح ذلك، ولكنه جعل شيفرة `HTML` فوضويّة قليلًا، لذلك استعضت عنها ببعض شيفرات `JavaScript` التي تضع كل حرف داخل عنصر `span` تلقائيًا.

سُطبّق آنذاك الحركة على كل حرف على حدة. سنبدأ أولًا بالإطارات المفتاحيّة:

```
@keyframes spin-letters {
  0%, 10% {
    opacity: 0;
    transform: rotateY(90deg);
  }
  30% {
    opacity: 1;
  }
  70%, 86% {
    transform: rotateY(0);
    opacity: 1;
  }
  95%, 100% {
    opacity: 0;
  }
}
```

في البداية، تكون الحروف مُدَوَّرة بزواوية 90 درجة، ثمَّ بزواوية 70% خلال الحركة، إذ تُحرَّك لمواجهة المُشاهد.

يمكننا تطبيق مجموعة الإطارات المفتاحية هذه على كل عناصر `span` على النحو التالي:

```
.byline span {
  animation: spin-letters 10s linear infinite;
}
```

والنتيجة هي أنَّ كل عناصر `span` التي تحتوي الحروف ستظهر تدرجيًا وتدور ببطء في

مكانها، قبل أن تتلاشى في نهاية الحركة.

بجمع كل ذلك معًا، سنحصل على المشهد التالي:



7. تمرين

إذا كنت تملك بعض الوقت، أشجعك على إلقاء نظرة على قسم CSS في

[نسخة CodePen](#).

قد تلاحظ وجود بعض استعلامات الوسائط `@media` في CSS، إذ نستخدم هذه

الاستعلامات لتحجيم المثال في الأجهزة الصغيرة. حاول تغيير بعض إطارات الحركة المفتاحية،

أو قيم الخاصية `transform` لمعرفة ما سيحدث.

إظهار المحتوى أثناء التمرير

17

أحد الاستخدامات الشائعة للحركات تحريك العناصر عند التمرير (scrolling) لأسفل الصفحة، لذلك ستتعلم في هذا الفصل كيفية القيام بذلك. إليك **المثال التوضيحي** الذي سنعمل عليه. نزلّه ثمّ حاول تمرير الصفحة للأسفل، وانظر كيف تظهر الاقتباسات وصور القطط في مكانها.

1. المكتبة Wow.js

تعرض العديد من المواقع حركات مُخصّصة عند تمرير الصفحة إلى نقاط معينة. يمكن أن تبدأ بتشغيل مقطع فيديو، أو تشغيل إطارات تحريك مفتاحيّة مُعقّدة، أو جعل بعض العناصر تظهر بالتدرّج للفت الانتباه إليها.

تُستخدَم في جميع تلك الحالات بعض شيفرات **JavaScript**، والتي تضيف صنفًا إلى عنصر محدّد عندما يصبح مرئيًا على الشاشة. يمكننا بعد ذلك إرفاق حركات مُخصّصة بذلك الصنف بشكل يؤدي إلى بدء المتصفح الحركة في الوقت المناسب الذي يظهر فيه العنصر أمام القارئ (الزائر).

هناك العديد من خيارات **JavaScript** التي يمكن أن تضيف أصنافًا، وأحد أفضل هذه الخيارات المكتبة **Wow.js**. سنستخدمها لإنشاء مثال بسيط يظهر فيه المحتوى تدريجيًا أثناء التمرير.



1. استخدام Wow.js

استخدام Wow.js يستلزم خطوتين. الخطوة الأولى هي **تنزيل إضافة JavaScript**. ضع الملف `wow.min.js` بعد تنزيله في مجلد JavaScript الخاصّ بالمشروع. الخطوة التالية هي الإشارة إلى هذا الملف من داخل شيفرة HTML:

```
<script src="javascripts/wow.min.js"></script>
```

(بافتراض أنّ المجلد يسمى `javascripts`؛ غيِّره إلى الاسم المناسب إن كان ذلك لازماً).

بعد ذلك، سنستدعي **JavaScript** باستخدام هذه التعليمات (الصفحة بعد الشيفرة السابقة):

```
<script>
  new WOW().init();
</script>
```


يمكننا الآن إضافة الصنف wow إلى المحتوى، وسوف تتكفل المكتبة Wow.js بتحديد ما إذا

أصبح المحتوى مرئيًا على النافذة أم لا لإظهاره بحركةٍ لافتةٍ.

ب. إضافة أصناف wow

إن أردت تحريك عنصر مُعيّن عند التمرير، فأضف إليه الصنف wow:

```
<p class="wow">...</p>
```

هذا يعني أنه عندما يظهر العنصر في نافذة المتصفح أثناء التمرير، تضيف Wow.js اسم

الصنف animated إلى قيمة الخاصية class على النحو التالي:

```
<p class="wow animated">...</p>
```

إن أضفنا حركة مُخصّصة على العناصر p. animated (أي العناصر p ذات الصنف

animated)، فلن تعمل الحركة إلا عند إضافة هذا الصنف.

2. الإخفاء والعرض

في مثالنا، سنخفي جميع العناصر ذات الصنف wow، وسنعرضها عندما يُضاف لها الصنف

animated. أولاً، سنخفيها عبر الشيفرة التالية:

```
.wow {
  opacity: 0;
  transition: all 0.5s 0.5s ease-out;
}
```

سُتطبّق أيضًا عملية انتقال (transition) هنا حتى يظهر العنصر تدريجيًا. لاحظ القيمة

0.5s، إذ أضفنا في هذا المثال تأخيرًا (delay) لمدة نصف ثانية، يضمن هذا أن يكون العنصر

موجودًا ضمن إطار العرض (viewport) قبل أن يبدأ بالظهور التدريجي.

تُحدّد الشيفرة التالية كيف سيبدو العنصر عندما يُضَاف إليه الصنف `:animated`:

```
.animated {
  opacity: 1;
}
```

صارت العناصر الآن تظهر ظهورًا تدريجيًا عند تمرير المستخدم الصفحة إلى الأسفل. **شاهد**

ذلك في التجربة الحية للمثال.

3. استخدام Animate.css

صُمّمت المكتبة Wow.js لتتكامل مع المكتبة `Animate.css`. لم أستخدمها في المثال

حتى الآن لأنني أفضل أن تفهم كيفية إنشاء الانتقالات بنفسك، لكن هناك بعض الانتقالات الممتازة

الجاهزة التي توفرها لنا المكتبة `Animate.css`.

في هذا المثال، استعملت `Animate.css`. لاحظ أنه لا توجد أيّة حركات أو انتقالات في

شيفرة `CSS`. بدلاً من ذلك، أضفت صنفًا إلى شيفرة `HTML` لإخبار `Animate.css` بالحركة

الواجب تطبيقها:

```
<section class="image wow bounceInUp">
```

يشير الصنف `bounceInUp` إلى أحد الحركات المُضمّنة في `Animate.css`. إن اخترت `cog`

في قسم `CSS` في المثال، فسترى أنني أشرت إلى المكتبة `Animate.css` ضمن

ملفات `CSS` الخارجية.

4. استخدام Modernizr

من الجيّد عمومًا الاحتياط من المواقف التي نخفي فيها المحتوى ثمّ نظهره باستخدام

`JavaScript`. فبعض المستخدمين لا يشقّلون `JavaScript` لسبب أو لآخر. يمكن استخدام

سكربت مثل **Modernizr** للتعامل مع هذه المشكلة، إذ سيضيف صنفًا `js` إلى جسم الصفحة (body)، ويُمكننا بعد ذلك استخدامه في تنسيقاتنا. لقد أضفت **Modernizr** في **هذا المثال** فاطلع عليه بتفحُّص.

5. تمرين

يُعدُّ إظهار المحتوى تدريجيًّا في مكانه مع تطبيق حركة ما بدايةً رائعةً، ولكن كيف يمكنك استخدام ذلك لتحسين تصميماتك ومواقعك؟ عندما تتصفح الإنترنت، ابحث عن المحتويات التي تظهر مع حركة أثناء التمرير لأسفل الصفحة وحلِّها وحاول تقليدها. الأهم من ذلك، اسأل نفسك السؤال التالي وحاول الإجابة عليه: متى تكون هذه التقنية نافعة؟ ومتى يُفضَّل تجنُّبها؟

سهولة الوصول

18

تعلّمنا في الفصول السابقة مفهوم التحريك وكيفية إنشاء الحركات. قبل ختام هذا الموضوع، سننرّيث قليلاً للتفكير فيما يمكننا القيام به للتأكد من أنّ الزائرين سيستفيدون من الحركات التي نضعها في مواقعنا.

هناك العديد من الطرائق التي يمكن أن تساعد بها الحركات في عرض المحتوى، لكنّها قد تكون مصدر إزعاج ومشاكل في بعض الأحيان.

1. التأكد من سهولة الوصول إلى المحتوى

حرّكنا في بعض الأمثلة المحتوى لإظهاره على الشاشة. إن كان المحتوى مخفياً في البداية، فمن الضروري الانتباه إلى أنّه لن يكون مخفياً أيضاً بالنسبة لبعض المستخدمين. أي يجب التأكد من وصول قارئ الشاشة إلى المحتوى المخفي بالنسبة لضعيفي البصر على سبيل المثال لا الحصر.

تدعم المتصفحات القديمة الحركات المُصمّمة عبر CSS بدرجات متفاوتة، وقد لا تعمل JavaScript دائماً (أي لا تكون مُفعّلة على جميع المتصفحات). يمكننا استخدام أدوات مثل Modernizr لتجاوز مثل هذه العقبات.

نستخدم الحركات لإضفاء معانٍ معيّنة على تصميماتنا. عند تصميم الحركات، ضع في الحسبان أنّ بعض الأشخاص لن يتمكنوا من رؤيتها؛ فربما يستخدمون قارئ الشاشة (screen reader)، أو أنّ إعدادات متصفحاتهم تمنع عمل الحركات بالشكل المطلوب. تأكد من توفر المعلومات المُهمّة حتى في حال لم تعمل الحركة.

كما أنّ تشغيل الفيديو تلقائياً قد يكون مصدر إزعاج لبعض المستخدمين، ويمكن أن تؤدي الحركات التي تعمل تلقائياً إلى تشتيت الانتباه عن بعض محتويات الصفحة. حاول حصر

استخدام الحركات في المواقع التي تريد من زوّاك أن يركّزوا على محتوى معيّن. قد يعني هذا أيضًا تحديد المدّة التي تستغرقها الحركة، أو التأكد من توقفها حتى يركّز الزائر على محتويات أخرى.

2. إتاحة التحكم

توصي منظمة W3C بأن يكون لأي محتوى يومض (blinking). خصوصًا إذا كان أحد زوارك مريضًا ويسبب الوميض نوبةً أو صداعًا له، أو يُمرّر، أو يُحدّث تلقائيًا لمدة تزيد عن خمس ثوانٍ وسيلةً لإيقافه مؤقتًا، أو إزالته. يُعدُّ استخدام الخاصية `animation-play-stat` إحدى طرائق إيقاف الحركة عند الحاجة، كما هو **موضح هنا عند التمرير**.

3. إتاحة مدخلات بديلة

يستخدم نسبة كبيرة من زوار موقعك الهواتف التي لا تحوي مؤشر (فأرة)، ولا توجد بها حالة حومان (hover state)، لذا من المهم أخذ هذا في الحسبان عند التصميم. أحد الخيارات التي أستخدمها غالبًا هو رصد اللمسة (tap) وإضافة الصنف الزائف `active`: إلى العناصر التي لمُست. أُضيف بعدئذٍ الانتقالات أو الحركات إلى الحالات `hover`: و `active`:

4. الارتباك

تبالغ في بعض الأحيان بعض المواقع بتحريك كل شيء فيها. هذا لن يقوِّض رسالة الموقع فحسب، بل يمكن أن يربك الزائر، وذلك ينطبق على واجهة المستخدم تحديدًا.

عند إضافة حركةٍ إلى عناصر الصفحة التي يتفاعل معها الزوار، تأكّد من أنّ هناك سببًا وجيهاً لاستخدام الحركات. لا يملك الزوار الوقت الكافي لتفكيك شيفرات ورموز التصميمات المُعقّدة المرئية أو الطريقة التي تتحرّك بها، لذلك تحرّى الوضوح والبساطة قدر الإمكان.

5. لا تزعجني!

المبالغة في تطبيق الحركات أو استخدام النوع الخطأ منها يمكن أن يزعج الآخرين. عندما قدّمت Apple نظام التشغيل iOS7، وهو نظام تشغيل مُحدّث لهواتفها، أدخلت فيه الكثير من الحركات وبعضها كان مزعجًا. وقد عانى آنذاك بعض الأشخاص، مثلي، معنأةً كبيرةً أثناء القراءة في السيارة أو الحافلة، لأنّ الحركة كانت مربكة، وآخرون عجزوا عن لعب ألعاب فيديو مُعيّنة لأكثر من بضع دقائق.

يرجع سبب حدوث ذلك إلى الطريقة التي يعمل بها **نظامنا الدهليزي**. لدينا ثلاثة أنابيب في آذاننا تساعدنا على تحديد مكان رأسنا في الفضاء ثلاثي الأبعاد. من الممكن أن نخدع عقولنا ونوهمها بأننا نتحرك، بيّد أنّ المشكلة تقع عندما لا يحسّ النظام الدهليزي بتلك الحركة، وسنشعر آنذاك بالدوار والارتباك.

التقيؤ عملية معقّدة، وهناك جزء من أدمغتنا متخصص في التعامل معه. يقع هذا الجزء من الدماغ بالقرب من الجزء الذي يتعامل مع التوازن، ولهذا نشعر بالغثيان.

يمكن أن يصبح هذا مشكلة أكبر مع تطور وانتشار التكنولوجيا القابلة للارتداء. تأكّد عند تصميم الحركات من اختبار مدى ارتياح وتقبُّل الناس لها. بعض الحلول أن تجرب الحركات التي تصممها مع أصدقائك وتأخذ رأيهم بها. يُعدّ موقع **Vestibular.org** نقطة انطلاق ممتازة للتعمّق أكثر في هذا الموضوع.

6. تسهيل الوصول للجميع

سهولة الوصول ليست نافعة للأشخاص الذين يستخدمون قارئات الشاشة، أو وسائل بديلة لتصفح المحتوى وحسب، فكثيرًا ما يتشتت انتباهنا، إذ ينقطع اتصال الشبكة، أو نتفحص هواتفنا أثناء الانتظار في طابور مكتب البريد، أو نتأكد من الاتجاهات على الخريطة أثناء القيادة.

خلاصة القول، يجب أن ننتبه إلى الطرق التي قد يستخدم بها الناس مواقعنا، ونحرص على ألا تكون الحركات التي نضعها فيها مزعجةً أو مربكةً لهم.

7. تمرين

إذا كانت وظيفتك تتضمن تصميم أو بناء واجهات لاستخدامها من الناس، فخصّص بعض الوقت لقراءة المقال [.NNGroup's Animation for Attention and Comprehension](#). فكر في الكيفية التي قد يستخدم بها الآخرون موقعك، وما المشاكل التي قد يواجهونها إذا لم يتمكنوا من رؤية الحركات.

نهاية الرحلة

19

هنيئًا لك! لقد أنهيت هذا الكتاب. أمل أن تكون قد استمتعت واستفدت منه. قبل أن تُنهي

هذا الموضوع، سنستعرض سويّةً بعض المصادر المفيدة للتعمق أكثر في عالم الحركات.

1. ملخص التحريك في CSS

لقد أعددت ملخصًا عن الانتقالات والتحريكات (بصيغة PDF) يلخص خاصيّاتها. وقد

صمّمته ليناسب صفحة واحدة بحجم A4.

2. أدوات مساعدة لإنشاء الحركات

من الجيّد أن تعرف كيفية إنشاء الحركات والانتقالات بنفسك، ولكن يكون في بعض

الأحيان من الأسهل إنشاؤها بالاعتماد على بعض المنصات الحاليّة. إليك بعض الأدوات الممتازة

التي يمكنك استخدامها لتوفير الوقت وإنشاء الحركات بسرعة.

أ. المكتبة `Animate.css`

يمكنك الإشارة إلى ملف CSS الخاصّ بالمكتبة `Animate.css`، وإضافة أيّ من أصناف

الحركات الجاهزة إلى العناصر المراد تحريكها. إليك مثالًا عن استخدام `Animate.css`.

ب. المكتبة `Hover.css`

تعدّ المكتبة `Hover.css` بديلًا جيّدًا عن `Animate.css`، إذ توفر مجموعةً كبيرةً من

الحركات الجاهزة لتطبيقها على الروابط أو الأزرار أو الشعارات أو أي عنصر من عناصر HTML.

3. أدوات أخرى

إنشاء الحركات عبر CSS وسيلةٌ قويةٌ جدًّا، ويمكنها تحقيق الكثير دون الحاجة إلى اللجوء

إلى تقنيات أخرى مثل `JavaScript`. ومع ذلك، فقد لا تكون هذه الوسيلة مناسبةً لجميع الحالات.

ستحتاج في بعض الأحيان إلى حركات أعقد، ويمكن آنذاك الاستعانة بلغة JavaScript. تعتمد أفضل الحزم على CSS للاستفادة من سرعتها، ودعم المتصفحات لها.

١. منصة GSAP

منصة GSAP قائمة على JavaScript لإنشاء حركات مُتقدِّمة، وتوفر تحكُّمًا دقيقًا، وأداءً رائعًا. هذه الأداة تتطلَّب بعض الوقت لتعلمها، لكنَّها مفيدةٌ ونافعةٌ.

ب. المكتبة Snabbt.js

المكتبة Snabbt قوية وفعَّالة لإنشاء حركات معقَّدة باستخدام دوال توقيت مخصَّصة، إذ تُنشئ مصفوفات تحويل (transform matrices) ليستخدمها المتصفح في تحريك الحركة، لهذا تتميِّز بأداء جيد جدًّا.

ج. المكتبة CSS Animate

المكتبة CSS Animate هي أداة ممتازة لإنشاء إطارات الحركة المفتاحية. إذ تساعد على تصميم الحركات عن طريق سحب العناصر، واستخدام دوال زمنية مفيدة.

د. موقع Cubic-bezier.com

عندما ترغب في إضفاء الحيويَّة على دوال التوقيت، فإنَّ موقع Cubic-bezier.com هو الأداة الأفضل لذلك، إذ يساعد على تخصيص دوال التوقيت. وقد تم تصميم أداة مشابهة في أدوات المطور في المتصفح Chrome.

4. ماذا بعد؟!

السؤال الذي قد يراودك الآن هو: «ماذا أفعل للانتقال إلى مستوى متقدّم في

مجال التحريك؟»

نصيحتي لك هي البحث عن التحديات. ابحث عن الإلهام في مواقع مثل [Dribbble.com](https://dribbble.com)

أو [CodePen.io](https://codepen.io). استوحي الأفكار من المنتجات الشهيرة (كثيرًا ما أحاكي تصاميم Apple) أو

الأفلام أو البرامج التلفزيونية. هل يمكن محاكاتها والاستفادة منها في تصميم المواقع؟ هل

ستكون لغة CSS هي الطريقة الأنسب للقيام بذلك؟ اسأل نفسك وابتح عن الإجابات بحثًا دقيقًا،

فالبحث هو عماد التعلّم.

توفر أيضًا **أكاديمية حسوب** الكثير من المحتوى حول مختلف تقنيات CSS، بما فيها تقنيات

التحريك. جمعت لك جميع الدروس التطبيقية المتعلقة بالحركات وإنشائها تجدها في [هذا المقال](#)

الشامل الذي يتحدّث عن التحريك في CSS ويزودك بجميع المقالات والدروس النظرية

والتطبيقية المتوفرة في الأكاديمية.

إن أردت ألا يضيع منك ما تعلّمته في هذا الكتاب، فضع كل ما تعلمته موضع الممارسة

والتطبيق. ابحث عن مواضع مناسبة لاستخدام الحركات واستخدمها، وابتح عن الحركات التي

لفتت انتباهك وأثارت اهتمامك وأعد تصميمها وعدل عليها. وحاول الاستمتاع أثناء رحلتك

التعليمية، فذلك سيجعلها أسهل وأمتع.

هنيئًا لك على الانتهاء من هذه المقدمة عن إنشاء الحركات عبر CSS! أمل أنّك استمتعت

بها واستفدت منها. ستجعلك إضافة الحركات مصممًا أفضل، وتجعل مشاريعك متميزة وجذّابة.