

kurs فوتو حرف تدريبات

HTML



عبد اللطيف ايمش

نحو فهم أعمق لتقنيات

HTML5

ترجمة

عبد اللطيف ايمنش

عبد اللطيف ايمش



عبد اللطيف ايمش، مهندس مدنی، مهتم بالتقنيات خصوصاً تطوير الويب وإدارة الأنظمة والخواديم؛ مترجم كتاب «[سطر أوامر لينكس](#)» وكتاب «[دليل إدارة خواديم أوبنتو](#)» وكتاب «[تعلم JavaScript](#)» وكتاب «[تعلم jQuery](#)»، وشاركت في تأليف كتاب «[تعلم البرمجة بلغة PHP](#)». أنشأت عدة دورات تدريبية في أكاديمية حسوب (عن شهادة ICND1، ومقادمة إلى لغة PHP، ومدخل إلى كتابة سكريبتات Bash ...); يمكنك التواصل معي عبر بريدي الإلكتروني: abdallatif.ey@gmail.com.

هذا الكتاب

أنتج هذا الكتاب برعاية شركة حسوب وأكاديمية حسوب.



أكاديمية حسوب

تهدف أكاديمية حسوب إلى توفير مقالات ودورات عاليّة الجودة حول مجالاتٍ مختلفةٍ وبلغةٍ عربيةٍ فصيحةٍ. باب المُساهمة على الأكاديمية مفتوحٌ لكل من يرى في نفسه القدرة على توفير مقالاتٍ عالية الجودة.

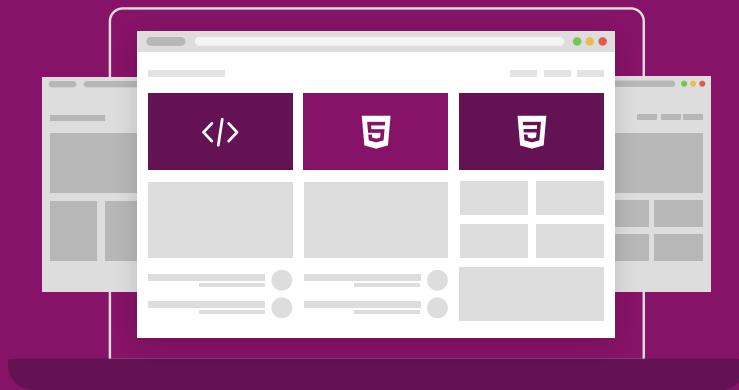
Academy.hsoub.com

شركة حسوب

تهدف حسوب لتطوير الويب العربي وخدمات الإنترنـت عن طريق توفير حلول عملية وسهلة الاستخدام لتحديات مختلفة تواجه المستخدمين في العالم العربي. يعمل في حسوب فريق شاب وشغوف من مختلف الدول العربية ومتـلك الشركة عدة خدمات يمكن معرفتها بزيارة موقع الشركة

Hsoub.com

دورة تطوير واجهات المستخدم



مميزات الدورة

- ✓ شهادة معتمدة من أكاديمية حسوب
- ✓ إرشادات من المدربين على مدار الساعة
- ✓ من الصفر دون الحاجة لخبرة مسبقة
- ✓ بناء معرض أعمال قوي بمشاريع حقيقة
- ✓ وصول مدى الحياة لمحتويات الدورة
- ✓ تدريبات مستمرة على الدورة مجاناً

اشترك الآن



جدول المحتويات

12.....	تقديم
14.....	تمهيد
15.....	1. HTML5 ليست شيئاً واحداً كبيراً
16.....	2. ليس عليك التخلص من كل شيء
17.....	3. يسهل البدء باستعمالها
18.....	4. إنها تعمل بالفعل!
19.....	5. HTML5 ستبقى وستتطور
21.....	كيف وصلنا إلى HTML5
22.....	1. أنواع MIME
23.....	2. شرخ مُسهب للكيفية إنشاء المعايير
35.....	3. سلالة مستمرة
38.....	4. التسلسل الزمني لتطوير HTML من 1997 إلى 2004
39.....	5. كل شيء تعرفه عن XHTML خطأ
42.....	6. رؤية تنافسية
45.....	7. مجموعة عمل WHAT
47.....	8. العودة إلى W3C
48.....	9. حاشية
49.....	10. مراجع إضافية

اكتشاف دعم ميزات HTML5

51.....	1. تقنيات الاكتشاف.....
52.....	2. مكتبة اكتشاف دعم ميزات HTML5 Modernizr
53.....	3. الكشف عن دعم الرسم عبر عنصر canvas
55.....	4. الكشف عن دعم النصوص في عنصر Canvas
57.....	5. الكشف عن دعم الفيديو
58.....	6. صيغ الفيديو
61.....	7. التخزين المحلي
63.....	8. Web Workers
64.....	9. تطبيقات الويب دون اتصال
66.....	10. تحديد الموقع الجغرافي
68.....	11. أنواع الإدخال في النماذج
70.....	12. النص البديل
71.....	13. التركيز التلقائي على النماذج
73.....	14. البيانات الوصفية
74.....	15. التاريخ
75.....	16. مراجع إضافية

البنية الهيكيلية لمستندات HTML5

78.....	1. نوع المستند - Doctype
81.....	2. العنصر الجذر
82.....	3. العنصر head
83.....	4. ترميز المحارف
85.....	5. الروابط العلاقية
87.....	ا. العلاقة rel = stylesheet

88.....	ب. العلاقة rel = alternate
89.....	ج. الروابط العلائقية الأخرى في HTML5
92.....	6. العناصر البنوية الجديدة في HTML5
95.....	7. تفصيل كيف تتعامل المتصفحات مع العناصر غير المعروفة
101.....	8. الترويسات - Headers
105.....	9. المقالات
109.....	10. الوقت والتاريخ
111.....	11. التنقل
114.....	12. التذبيبات
117.....	13. مراجع إضافية

الرسم عبر عنصر canvas

121.....	1. الأشكال البسيطة
124.....	2. الإحداثيات في عنصر canvas
126.....	3. المسارات
131.....	4. النص
136.....	5. التدرجات اللونية
141.....	6. الصور
146.....	7. ماذا عن متصفح IE؟
149.....	8. مثالٌ متكامل
157.....	9. مصادر إضافية

الفيديو

160.....	1. حاويات الفيديو
162.....	2. مرمازات الفيديو

164.....	H.264 .ا
165.....	Theora .ب
166.....	VP9 و VP8 .ج
167.....	3. مرمازات الصوت
169.....	.ا. MPEG-1 Audio Layer 3
170.....	.ب. Advanced Audio Coding
171.....	.ج. Vorbis
172.....	.د. Opus
173.....	4. الصيغ التي تعمل في الويب
175.....	5. مشاكل الترخيص مع فيديو H.264
178....	6. ترميز الفيديو باستخدام Miro Video Converter
182.....	7. ترميز فيديو H.264 باستخدام HandBrake
190.....	8. ترميز عدّة مقاطع إلى H.264 عبر HandBrake
192.....	9. ترميز فيديو WebM باستخدام FFMPEG
194.....	10. وأخيراً، الشيفرات
199.....	.ا. أنواع MIME تكشف عن وجهها القبيح
200.....	.11. ماذا عن متصفح IE؟
201.....	.12. مثال متكامل
203.....	.13. مصادر إضافية

تحديد الموقع الجغرافي

204.....	
205.....	1. واجهة تحديد الموقع الجغرافي البرمجية
206.....	2. أريني الشيفرة
210.....	3. التعامل مع الأخطاء
212.....	4. الخيارات المتاحة أمامك
215.....	5. ماذا عن متصفح IE؟

216.....	6. مكتبة geo.js
219.....	7. مثال متكمال
220.....	8. مصادر إضافية

التخزين المحلي

221.....	1. لمحه تاريخية عن التخزين المحلي قبل HTML5
223.....HTML5	2. تمهيد إلى التخزين المحلي في HTML5
225.....HTML5	3. استخدام التخزين المحلي في HTML5
227.....HTML5	ا. تتبع التغييرات في مساحة التخزين المحلي
229.....	ب. المحدوديات في المتصفحات الحالية
231.....	4. مثال عملي عن استخدام التخزين المحلي
232.....	5. مستقبل التخزين المحلي في تطبيقات الويب
235.....	6. مصادر إضافية

تطبيقات الويب التي تعمل دون اتصال..

240.....	1. ملف Manifest
242.....	ا. قسم NETWORK
244.....	ب. قسم FALLBACK
245.....	2. تسلسل الأحداث
248.....	3. تنقية الأخطاء
251.....	4. لتنشئ واحداً!
255.....	5. كلمةأخيرة
257.....	6. مصادر إضافية

النماذج

259.....	1. النص البديل
260.....	2. التركيز التلقائي على الحقول
261.....	

1. ضبط التركيز التلقائي في أقرب فرصة ممكنة.....	264
3. عناوين البريد الإلكتروني.....	267
4. عناوين الويب.....	270
5. إدخال الأرقام.....	272
6. تحديد الأرقام عبر المزلاج.....	276
7. مُنتقي التاريخ.....	277
8. حقول البحث.....	280
9. مُنتقي الألوان.....	282
10. التحقق من صحة مدخلات المستخدم.....	283
11. الحقول المطلوبة.....	285
12. مصادر إضافية.....	286

البيانات الوصفية.....288

1. ما هي البيانات الوصفية؟.....	290
2. النموذج الهيكلي للبيانات الوصفية.....	292
3. توصيف الأشخاص.....	297
1. التعرف على المقتطفات المنسقة.....	307
4. توصيف المنظمات.....	311
5. توصيف الأحداث.....	319
1. المقتطفات المنسقة من جديد!.....	326
6. توصيف المراجعات.....	329
7. مصادر إضافية.....	335

التعامل مع التاريخ.....336

1. السبب وراء تعديل التاريخ.....	337
2. طريقة تعديل التاريخ.....	340

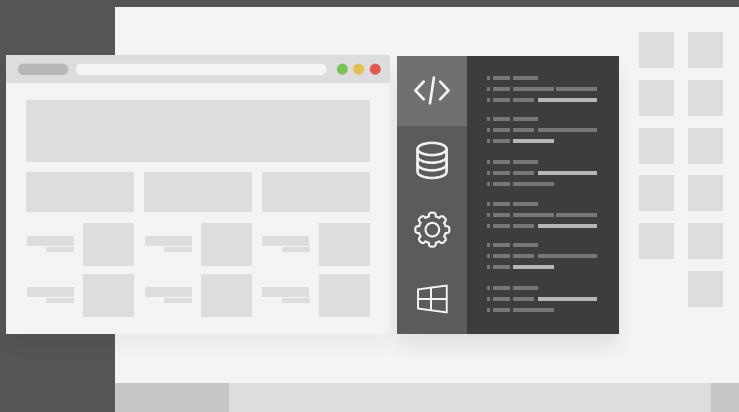
347 3. مصادر إضافية

الملحق الأول: دليل اكتشاف دعم المتصفح

348 HTML5 ميزات

358 1. مصادر إضافية

دورة علوم الحاسوب



دورة تدريبية متكاملة تضعك على بوابة الاحتراف
في تعلم أساسيات البرمجة وعلوم الحاسوب

التحق بالدورة الآن



تقديم

لا يخفى على أحدٍ سطوع نجم لغة HTML5 وانتشار تطبيقاتها انتشاراً كبيراً، إذ داع صيتها وأصبحت حديث الكثيرين لما تحتويه من تقنياتٍ مهمةٍ لتطبيقات الويب؛ فهي تتضمن كل ما يتعلق بتشغيل مقاطع الفيديو على صفحات الويب، وتوليد الرسوميات ديناميكياً، وتحديد الموقع الجغرافي للمستخدم، وإتاحة استعمال تطبيقات الويب دون اتصال، إضافةً إلى تنظيمها لبنية المستند الهيكلية تنظيمًا دقيقًا يُسهل تفسيرها من المتصفحات والبرمجيات الأخرى، والمزيد...

لذا جاء هذا الكتاب محاولاً تقديم مفاهيم HTML5 وتقنياتها وطرائق استخدامها إلى القارئ العربي، مدعماً بأمثلةٍ عمليةٍ تُسهل توضيح الأفكار؛ وحاولت فيه توفير أحدث المعلومات عن دعم تلك التقنيات قدر المستطاع، وأعدت النظر في بعض الفصول لتناسب التغيرات التي طرأت حديثاً.

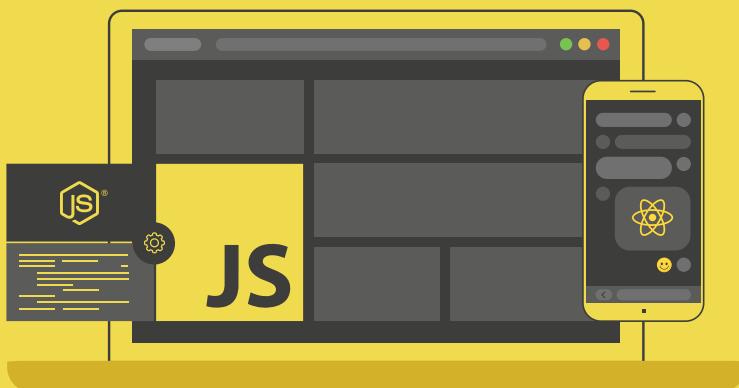
هذا الكتاب مترجمٌ عن كتاب «Dive Into HTML5» للمؤلف Mark Pilgrim المرخص برخصة المشاع الإبداعي CC BY 3.0، والذي نَشرَته O'Reilly لاحقاً باسم «HTML5: Up and Running». هذا الكتاب مرخص بموجب رخصة المشاع الإبداعي «نَسْب المُصَنَّف - الترخيص بالمثل 4.0» (CC BY 4.0). شعار HTML5 والشعارات البقية مرخصة برخصة المشاع الإبداعي CC BY-SA 4.0. وفي النهاية، أَحْمَد اللَّهُ عَلَى تَوْفِيقِه لِي بِإِتَامِ الْعَمَلِ عَلَى الْكِتَابِ، وَأَرْجُو أَنْ يَكُونَ إِضَافَةً مفيدةً لِلْمَكْتَبَةِ الْعَرَبِيَّةِ، وَاللَّهُ وَلِي التَّوْفِيقِ.

عبد اللطيف محمد أديب ايمش

2017/8/13

حلب، سوريا

دورة تطوير التطبيقات باستخدام لغة JavaScript



مميزات الدورة

- ✓ شهادة معتمدة من أكاديمية حسوب
- ✓ إرشادات من المدربين على مدار الساعة
- ✓ من الصفر دون الحاجة لخبرة مسبقة
- ✓ بناء معرض أعمال قوي بمشاريع حقيقة
- ✓ وصول مدى الحياة لمحتويات الدورة
- ✓ تحديثات مستمرة على الدورة مجاناً

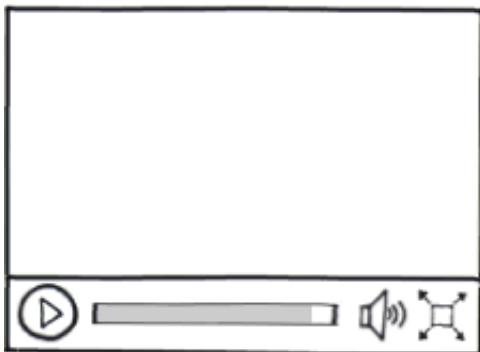
اشترك الآن



تہمید

1. HTML5 ليست شيئاً واحداً كبيراً

ربما تتساءل: «كيف يمكنني البدء باستعمال HTML5 إن لم تكون تدعمها المتصفحات القديمة؟» لكن السؤال نفسه سيعطل، فليست HTML5 شيئاً واحداً كبيراً، وإنما مجموعة من الميزات المنفصلة عن بعضها، أي أنك لن تحاول اكتشاف «دعم HTML5» في المتصفح، لأن ذلك غير منطقي؛ وإنما يمكنك استكشاف الدعم للمزايا المختلفة مثل التخزين المحلي، أو عرض الفيديو، أو تحديد الموقع الجغرافي.



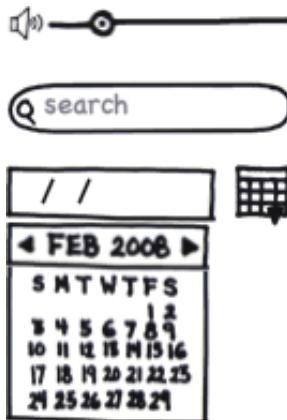
ربما تظن أن HTML هي مجموعة من الوسوم وتلك الأقواس التي تشبه الزاوية... إن هذا جزء مهم منها، لكنه لا يمثلها كلها. إذ تُعرف مواصفات HTML5 كيف تتفاعل تلك الوسوم مع لغة JavaScript وذلك عبر ما يُعرف بالمصطلح «DOM» (اختصار للعبارة Document Object Model). فلا تُعرف HTML وسماً باسم <video> فحسب، وإنما هنالك واجهة برمجية للتعامل مع كائنات الفيديو عبر DOM. يمكنك استعمال تلك الواجهة البرمجية (أي API) لكي تكتشف الدعم لمختلف صيغ الفيديو، ولكي تبدأ المقطع أو توقفه مؤقتاً، أو أن تكتم صوته، أو أن تعرف ما هو المقدار الذي تُرِّزَّل (downloaded) من الفيديو، وكل شيء آخر يلزمك لبناء تجربة مستخدم رائعة عند استعمال وسم <video> لعرض المقطع.

الفصل الثاني والملحق الأول

سيعلمك كيف تكتشف الدعم لمختلف ميزات HTML5 الجديدة.

2. ليس عليك التخلص من كل شيء

شئت أم أبيت، لا تستطيع أن تنكر أن HTML هي أنجح لغة تصويف (markup) على الإطلاق. بُنيت HTML على هذا النجاح، وليس عليك أن تتخلص عن الشيفرات التي كتبتها، وليس عليك إعادة تعلم أشياء تعرفها من قبل، فإن كان تطبيقك يعمل بالاربعة باستخدام HTML 4، فسيبقى يعمل اليوم في عصر HTML5.



لكن إن جئت لتحسين تطبيق الويب الخاص بك، فقد أتيت إلى المكان الصحيح. هذا مثال واقعي: تدعم HTML5 كل عناصر النماذج (forms) في HTML 4، لكنها تتضمن عناصر جديدة أيضًا. كثا ننتظر إضافة بعض تلك العناصر بفارق الصبر، مثل المزلاج (slider) ومنتقي التاريخ (date picker); وببعضها الآخر له ميزات خفية، فحقل email مثله ك莫斯يل حقل الإدخال النصي العادي، إلا أن متصفحات الهاتف الذكي ستحتفظ لوحة المفاتيح الظاهرة على الشاشة لتسهيل كتابة عناوين البريد الإلكتروني. بعض المتصفحات القديمة لا تدعم حقل email وستعامله على

أنّه حقل نصي عادي، وسيبقى النموذج يعمل دون تعديلات في الشيفرة أو استخدام أساليب ملتوية عبر JavaScript. هذا يعني أنك تستطيع تحسين النماذج في صفحاتكاليوم، حتى لو كان زوارك يستعملون متصفح IE 6.

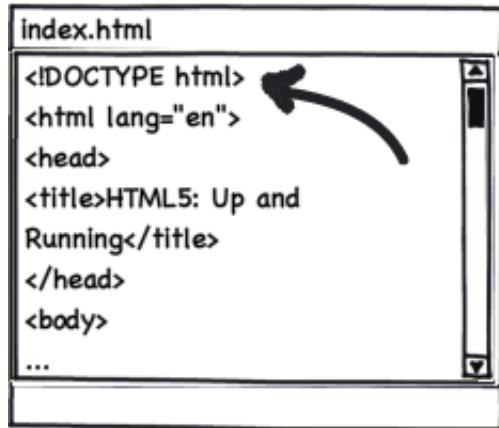
اقرأ عن التفاصيل الشيقة لنماذج HTML5 في [الفصل التاسع](#).

3. يسهل البدء باستعمالها

يمكن أن يكون «التحديث» إلى HTML5 بسيطًا لدرجة أن كل ما عليك فعله هو تعديل نوع المستند (doctype)، الذي يجب أن يكون أول سطر من كل صفحة HTML. تُعرف الإصدارات السابقة من HTML الكثير من أنواع المستند، وكان من الصعب اختيار النوع المناسب؛ لكن هنالك نوعdoctype وحيد في HTML5:

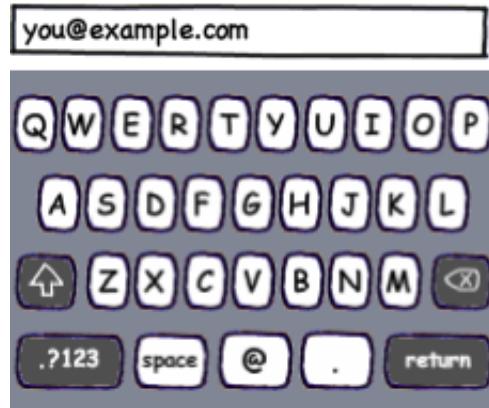
```
<!DOCTYPE html>
```

لن يضر التحديث إلى نمطdoctype في HTML5 شيفراتك المكتوبة، لأن جميع الوسوم المعرفة في HTML 4 ما تزال مدعومةً في HTML5، لكنها ستسمح لك باستعمال <section> و <article> و <header> و <footer>، يمكنك التعلم عن هذه العناصر الجديدة في [الفصل الثالث](#).



4. إنها تعمل بالفعل!

سواءً كنت تريده الرسم عبر canvas، أو تشغيل مقطع فيديو، أو تصميم نماذج أفضل، أو بناء تطبيقات ويب تعمل دون اتصال؛ فستجد أنَّ دعم HTML قد بلغ مبلغاً جيداً، إذ تدعم متصفحات عديدة خاصية canvas مثل Firefox و Safari و Chrome و Opera و متصفحات الهواتف الذكية ([الفصل الرابع](#)) و تشغيل الفيديو ([الفصل الخامس](#)) و تحديد الموقع الجغرافي ([الفصل السادس](#)) و التخزين المحلي ([الفصل السابع](#)) والمزيد. تدعم Google (في متصفحها) البيانات الوصفية الخاصة ([microdata](#)) ([الفصل العاشر](#))، وحتى مايكروسوفت -المشهورة بتأخرها عن اللحاق بركب دعم المعايير القياسية- تدعم أغلبية ميزات HTML5 في متصفح .«Internet Explorer 9»



يتضمن كل فصل من هذا الكتاب جداولً لتوفيقية المتصفحات الشهيرة مع الميزة المنشورة، ولكن الأهم من ذلك أنَّ كل فصلٍ يتضمن نقاشًا عن الخيارات المتاحة أمامك إن كنت تحتاج إلى دعم المتصفحات القديمة. تم توفير ميزات في HTML5 مثل تحديد الموقع الجغرافي ([الفصل السادس](#)) وتشغيل الفيديو ([الفصل الخامس](#)) في السابق عبر إضافات للمتصفح مثل Gears أو Flash. الميزات الأخرى، مثل canvas ([الفصل الرابع](#))، تستطيع محاكاتها محاكاةً تامةً باستعمال JavaScript.

سيعلمك هذا الكتاب كيف تستهدف المتصفحات التي تدعم تلك الميزات، دون أن تترك خلفك المتصفحات القديمة دون دعم.

5. HTML5 ستبقى وستتطور

ابتكر «Tim Berners-Lee» الشبكة العنكبوتية في بدايات التسعينيات من القرن الماضي، ثم أنشأ جمعية W3C لكي تكون المرجع في معايير الويب، وهذا ما فعلته تلك الجمعية لأكثر من 20 عاماً. هذا ما قالته W3C عن مستقبل معايير الويب في تموز/يوليو عام 2009:

يعلن المدير أنّه عندما ينتهي عقد مجموعة العمل على 2 XHTML كما هو مقرر في نهاية 2009، فلن يُمدد العقد. وبهذا -وبزيادة الموارد في مجموعة عمل HTML- فإن W3C تأمل بتسريع وتيرة العمل على HTML5، وتوضيح موقف W3C تجاه مستقبل HTML.

ستبقى HTML5 في المستقبل، لذا لنبدأ بتعلمها.

دورة تطوير التطبيقات باستخدام لغة بايثون



احترف البرمجة وتطوير التطبيقات مع أكاديمية حسوب
والتحق بسوق العمل فور انتهاءك من الدورة

التحق بالدورة الآن



كيف وصلنا إلى HTML5

وقد أخيراً بالصدفة على اقتباس من مطور في مشروع Mozilla عن التوتر الذي يحدث

عند العمل على صياغة المعايير القياسية:

تجب الموازنة بين المعايير وتطبيقاتها بدقة، فلا نريد أن يحدث تطبيق للمعايير قبل أن تنتهي صياغة المعيار، لأن الآخرين سيبدؤون بالاعتماد على تفاصيل التطبيق وهذا سيقيّد عملية تطوير المعايير. لكننا لا نريد انتهاء صوغ المعيار قبل أن يكون هناك تطبيق له، وقبل أن يجرب مطور ذاك المعيار في هذه التطبيقات، لأننا نحتاج إلى تغذية راجعة، وهذا أمر حساس جدًا ودقيق علينا التعامل معه بحذر.

ابق هذه المقوله في عقلك، ولنشرح كيف أصبحت HTML5 على ما هي عليه.

1. أنواع MIME

هذا الكتاب حول HTML5، وليس عن إحدى إصدارات HTML السابقة أو أي إصدارٍ من XHTML، لكن لفهم تاريخ HTML5 والدوافع خلف إنشائها تماماً، فعليك أن تستوعب بعض التفاصيل التقنية أولاً، تحديداً أنواع MIME.

في كل مرة يطلب فيها متصفح الويب صفحةً، فسيرسل الخادوم «ترويسات» (headers) قبل أن يرسل شيفرة الصفحة نفسها، وهذه الترويسات غير ظاهرة عموماً على الرغم من توفر أدوات تطويرية تمكّنك من رؤيتها إن كنت مهتماً بها. لكن الترويسات مهمة لأنها تُخبر المتصفح كيف يفسّر الشيفرات الموجودة في الصفحة، أحد أهم تلك الترويسات هو Content-Type وهو يبدو كما يلي:

Content-Type: text/html

«Content Type» أو نوع المحتوى أو نوع MIME، هذه الترويسة هي الشيء الوحيد الذي يُحدّد طبيعة المورد المُحدّد وكيف يجب عرضه. فالصور لها أنواع خاصة بها (image/jpeg، image/png، image/jpg)، وملفات CSS، JavaScript، JavaScript خاص بها، وكذلك الأمر لصفحات الأنماط (CSS)، فلكل شيء في الويب له نوع MIME خاص به.

وبالطبع الواقع معقدًّا أكثر من هذا، فالجيل الأول من خواديم الويب (وهنا أتكلم عن خواديم الويب في 1993) لم تكن ترسل ترويسة Content-Type لأنها لم تكن موجودةً في ذاك الوقت (إذ لم تستعمل إلا في 1994)، وكانت بعض المتصفحات الشهيرة تتتجاهل ترويسة Content-Type في بعض الظروف لأسبابٍ لها علاقة بالتوافقية (وهذا يسمى «content sniffing»)، لكن كقاعدة عامة: كل شيء يمكنك الحصول عليه على الويب (صفحات HTML، والصور، والسكريبتات، والفيديو، وملفات PDF، وكل شيء له رابط URL) يجب أن يُخدمًّا بإضافة نوع MIME المناسب في ترويسة Content-Type.

خُزن المعلومات السابقة في عقلك، وسنعود إليها لاحقًا.

2. شرح مُسَهِّبٌ لـ كيفية إنشاء المعايير

لماذا لدينا عنصرٌ باسم ؟ لن تسمع مثل هذا السؤال كل يوم. بكل تأكيد أنشأه أحدهم، إذ لم تظهر تلك الأشياء من العدم، فكل عنصر وكل خاصية (attribute) وكل ميزة من ميزات HTML التي استخدمتها من قبل قد أنشأها أحدهم وقرر كيف يجب أن تعمل وكتَّب ذلك كله؛ هؤلاء الأشخاص الذين كتبوا المعايير ليسوا خارقين وإنما مجرد بشر ولكنهم أذكياء.

أحد الأشياء الرائعة عن المعايير هي أنها طورت على الملا، أي أنك تستطيع العودة في الزمن إلى الوراء والإجابة عن النوع السابق من الأسئلة؛ إذ أن النقاشات كانت تجرى في القوائم البريدية، التي أرشفت وأصبحت متاحةً للعموم كي يبحثوا فيها؛ لذلك قررت أن «أنقب» في رسائل البريد الإلكتروني محاولاً الإجابة عن السؤال السابق: «لماذا لدينا عنصر ``؟»، كان علي البدء في حقبة قبل إنشاء منظمة باسم «رابط الشبكة العالمية» (World Wide Web) Consortium اختصاراً (W3C)، ذهبت إلى الأيام الأولى للويب، حيث كنت تستطيع عدّ صفحات الويب على أصابع اليدين.

في 25 شباط/فبراير من عام 1993، كتب Marc Andreessen

أود أن أقترح وسم HTML جديد اختياري هو:

IMG

يتطلب وسيطاً (argument) هو "url".SRC="

الذي يحدد صورة نقطية لكي يحاول المتصفح تنزيلها عبر الشبكة وتفسيرها على أنها صورة يجب تضمينها في النص مكان ظهور الوسم.

مثالٌ عنها:

``

(لا يوجد وسم إغلاق لها).

يمكن جعل الصورة تشير إلى رابط، وعندما سيمكن الضغط عليها مثل الروابط النصية العاديّة.

يجب على المتصفحات أن تكون مرنّة في دعمها لصيغ الصور، ومن المستحسن على سبيل المثال دعم الصيغتين Xbm و Xpm؛ وإن لم يتمكن المتصفح من تفسير صيغة معينة من الصور، فيمكن أن يضع ما يشاء بدلاً منها (سيوضع متصفح X Mosaic صورةً بديلةً افتراضيّةً).

هذه وظيفة مطلوبة في Mosaic X، وقد عملنا على تطبيقها وسنستخدمها محلياً على الأقل. أنا منفتح حالياً لأيّة اقتراحات حول كيفية التعامل مع ذلك في HTML، فإن كانت عندك فكرة أفضل مما قدمته، فرجاءً أعلموني بها، أنا أعلم أنّ هذه ليست صيغة مثالى للصور، لكنني لا أجد بديلاً عن قول «دع المتصفح يفعل ما يستطيع» وانتظار الحل المثالى لكي يأتي في المستقبل (ربما في يوم ما سنعتمد على أنواع MIME).

كانت صيغتا Xbm و Xpm صيغتى رسوميات شهيرتان في أنظمة يونكس. «Mosaic» كان من أوائل متصفحات الويب («X Mosaic» هو الإصدار الذي يعمل على أنظمة يونكس)، عندما كتب Marc Andreessen رسالة في بدايات 1993، لم يكن قد أسس الشركة التي جعلته مشهوراً (شركة Mosaic Communications Corporation)، ولم يكن بدأ العمل على المنتج الرائد في تلك الشركة: «Mosaic Netscape» (ربما تعرفها بأسمائها الحديثة «Netscape Navigator» و «Netscape Corporation»).

عبارته «ربما في يوم ما سنعتمد على أنواع MIME» كانت تشير إلى ميزة Content negotiation في HTTP، إذ يخبر العميل (أي متصفح الويب) الخادم (أي خادوم الويب) ما هي أنواع الوسائط التي يدعمها (مثل image/jpeg...) ومن ثم سيرسل الخادم الوسائط إلى العميل بالصيغة التي يفضلها. عُرِفت الصيغة الأصلية من HTTP في 1991 (وهي النسخة الوحيدة التي كانت موجودة في شباط/فبراير 1993) ولم تكن توفر طريقة للعملاء كي تخبر الخواديم عن صيغ الصور التي تدعهما، وهذه هي المعضلة التي واجهها Marc.

بعد عدّة ساعات، ردَّ Tony Johnson:

لدي شيءٌ شبيهٌ بما اقترحَت في 2.0 Midas (نستخدمه هنا في SLAC، وسيُنشر للجميع خلال أسبوعين)، عدا أنَّ جميع الأسماء مختلفة (وهنالك وسيطٌ إضافيٌ اسمه "NAME="name" وله الوظيفة نفسها تقريباً لوسُم IMG الذي اقترحْته):

```
<ICON name="NoEntry"
      href="http://note/foo/bar/NoEntry.xbm">
```

الفكرة في معامل name هي السماح للمتصفح بأن يكون لديه مجموعةً من الصور المضمنة داخلياً فيه، فإن طابق -الاسم صورةً مضمونةً داخلياً فسيستعمل تلك الصورة بدلاً من تنزيل الصورة المحددة، يمكن لخاصية name أن تمسِي «تميحةً» لمحفوظة الصورة للمتصفحات السطرية (أي تلك المتصفحات التي تعمل في سطر الأوامر) إذ ستضع رمزاً ما مكان الصورة.

لا أهتم كثيراً بأسماء الخصائص أو الوسوم، لكنني أظن أنه من المناسب أن نوحد

التسimpات. ولا أهتم أيضًا بالاختصارات (لم لا نستخدم `SOURCE= IMAGE=` على سبيل المثال)، وأنا أفضّل `ICON` لأنّه يعني أن الصورة يجب أن تكون صغيرة، لكن ألم نفرط في استعمالها؟

Midas هو متصفح آخر من فترة بدايات الويب، وكان منافسًا لمتصفح `X Mosaic`، وكان متعدد المنصات، إذ كان يعمل على أنظمة يونكس و `VMS`. أما «`SLAC`» فهي تشير إلى `SLAC National Accelerator Center` التي أصبح اسمها الآن «`Linear Accelerator Center`» التي استضافت أول خادوم ويب في الولايات المتحدة (وفي الواقع، **أول خادوم Laboratory** ويب **خارج أوروبا**). عندما كتب `Tony` هذه الرسالة، كانت `SLAC` من المحاربين القدامى في الشبكة العالمية، التي استضافت **خمس صفحات** على خادوم الويب الخاص بها لمدة 441 يومًا.

أكمل `Tony` قائلاً:

لما كتّا نتباحث في موضوع إضافة وسوم جديد، فلدي وسم آخر ذو وظيفة مشابهةٍ أود أن أضيف دعماً له في متصفح `2.0 Midas`، كما يلي:

<INCLUDE HREF="...">>

الغرض منه هو تضمين مستند آخر ضمن المستند الأول في مكان ورود الوسم، ويمكن أن يكون ذاك المستند من أي نوع، لكن الغرض الأساسي منه هو السماح بتضمين الصور (بحجمها الطبيعي) في المستندات.

أذّگر مرّة أخرى أن الغاية منه ستظهر بشكلٍ جلي عندما يسمح بروتوكول `HTTP2` بتحديد صيغة المستند المضمّن بشكلٍ منفصل.

قصد ببروتوكول «HTTP2» بروتوكول HTTP الأساسي المعرف في 1992، الذي لم يُطبق تطبيقاً كاملاً في بدايات 1993، وعُرِفت المسودة باسم HTTP2 ثم أصبحت معياراً قياسياً باسم HTTP 1.0» (على الرغم من أن ذلك قد جرى بعد ثلاثة أعوام)، تضمن بروتوكول HTTP 1.0 «**تروبيسات غرضها هو طلب تحديد صيغة المحتوى** request headers for content) .MIME أي أنواع negotiation :Tony أكمل

بديلٌ عما سبق هو:

See photo

لأفضل إضافة المزيد من الوظائف إلى وسم <A> لكن الفكرة هنا هي الحفاظ على التوافقية مع المتصفحات التي لا تراعي المعامل INCLUDE، فالغاية هي أن المتصفحات التي تفهم المعامل INCLUDE ستبدل النص (في هذه الحالة «See photo») وتضع بدلاً منه المستند (الصورة)، بينما المتصفحات الأقدم أو الأقل كفاءة ستتجاهل INCLUDE تماماً.

لم يُطبّق اقتراحه أبداً، إلا أن فكرة توفير نص إن لم تتوفر الصورة هي تقنية مهمة لتسهيل الوصول، التي كانت ناقصةً من اقتراح Marc الأولي لوسم . استعملت هذه الميزة في خاصية ، التي أساء متصفح Netscape معاملتها باعتبارها تلميحاً (tooltip). بعد عدّة ساعات من إرسال Tony لرسالته، ردّ Tim Berners-Lee عليه:

لقد تخيلت أنَّ طريقة تمثيل الصور هي:

```
<a name=fig1 href="fghjkdfghj" REL="EMBED, PRESENT">Figure</a>
```

حيث قيم الخاصية REL تعني:

- التضمين هنا عند العرض EMBED

- اعرضها كلما عُرِض المستند المصدر PRESENT

لاحظ أنك تستطيع استعمال تركيبات مختلفة من القيم السابقة، ولن يتعطل المتصفح إن لم يدعم إحداها.

أرى أنَّ استعمال الطريقة السابقة للأيقونات التي يمكن النقر عليها يعني تشغُّب وسوم <a>; حسناً... لكن لم أُرد وسماً خاصاً.

لم يُطبَّق هذا الاقتراح مطلقاً، لكن خاصية rel ما زالت موجودةً.

أضاف Jim Davis:

أظن أنَّ من الجيد وجود طريقة لتحديد نوع المحتوى، أي:

```
<IMG HREF="http://nsa.gov/pub/sounds/gorby.au" CONTENT-TYPE=audio/basic>
```

لكنني مستعدٌ للتعايش مع وجوب تحديد نوع المحتوى عبر امتداد الملف.

لم يُطبّق هذا الاقتراح قط، لكن أضاف متصفح Netscape لاحقاً دعماً لتضمين عناصر الوسائط المتعددة باستعمال عنصر `<embed>`.

سؤال : Jay C. Weber

مع أن الصور من أهم أنواع الوسائط التي أود أن تضمن في متصفحات الويب،
لكن لا أظن أن علينا إضافة طرق مختلفة لكل نوع من أنواع الوسائط على حدة؛
فما الذي حدث للحماس تجاه استعمال أنواع MIME؟

رد : Marc Andreessen

ما سبق ليس بديلاً عن استعمال MIME كآلية قياسية لتضمين المستندات، لكنه يُوفّر تطبيقاً بسيطاً وضرورياً لوظيفة منفصلة عن MIME.

استجابة : Jay C. Weber له قائلاً:

لننس مؤقتاً أنواع MIME إن كانت ستؤثر على وضوح المشكلة.

اعتراضي على نقاش «كيف يمكن أن ندعم تضمين الصور» وليس «كيف يمكن أن ندعم تضمين الكائنات لمختلف أنواع الوسائط المتعددة».

وإلا، فسوف يأتي أحدهم يقترح قائلاً «دعونا نضع وسماً جديداً هو `<AUD SRC="file://foobar.com/foo/bar/blargh.snd">` لتضمين الصوت».

لا أظن أن استعمال شيء عام وشامل سيكلفنا الكثير.

أدركنا بعد فوات الأوان أنَّ مخاوف Jay كان لها أساسٌ من الصحة لكن ذلك استغرق فترةً طويلةً، فقد أضافت HTML5 أخيرًا عنصري `<audio>` و `<video>`.

ردًا على رسالة Jay الأصلية، قال Dave Raggett:

هذا صحيح! أزيد دعماً لمختلف أنواع الصور، مع إمكانية تحديد النوع المفضل لدى المتصفح. وأرى أنَّ ملاحظة Tim عن دعم النقر على مناطق في الصورة مهمة.

لاحقاً في 1993، اقترح Dave Raggett معيار HTML+ الذي كان تطويراً لمعيار HTML. لكن لم يُطبّق اقتراحه مطلقاً، ثم حلَّ HTML 2.0 محله التي أعطت طابعاً رسمياً للميزات شائعة الاستعمال: «هذا المعيار يضفي توضيحاً وطابعاً رسمياً لمجموعة من ميزات HTML التي كانت شائعة الاستعمال قبل حزيران/يونيو عام 1994».

كتب Dave لاحقاً معيار HTML 3.0 المبني على مسودة HTML+ التي كتبها سابقاً، وذلك خارج إطار W3C للمعايير (المسمى Arena)؛ لكن لم تُطبّق HTML 3.0 أبداً، وحلَّ محلها HTML 3.2، التي رسّمت الميزات كالإصدارات السابقة HTML 2.0: «أضافت 3.2 HTML الميزات شائعة الاستخدام مثل الجداول، والبريمجات (applets) والتفاف النص حول الصور، بالإضافة إلى توافقيتها مع معيار HTML 2.0».

شارك Dave لاحقاً بوضع معيار HTML 4.0، وطور HTML Tidy، وشارك في المساعدة بتطوير XHTML، و MathML، و XForms، وغيرها من معايير W3C الحديثة.

بالعودة إلى 1993، رد Marc على Dave:

ربما علينا التفكير في لغةٍ لتوصيف الرسومات ذات غرضٍ عام يمكن فيها تضمين الروابط التشعبية المرتبطة بأيقوناتٍ أو صورٍ أو نصوصٍ أو أي شيء. هلرأي أحدكم شيئاً يشبه Intermedia بهذه الإمكانيات؟

Intermedia هو مشروعٌ من جامعة Brown، طُور من عام 1985 إلى 1991 وكان يعمل على نظام A/UX، الذي هو نظام شبيه بيونكس (Unix-like) لحواسيب ماكتنتوش في ذاك الوقت. راجت فكرة «لغة لتوصيف الرسومات ذات غرضٍ عام»، وذلك بدعم المتصفحات الحديثة لصيغة SVG (لغة توصيفية يمكن دمج السكريبتات معها)، و<canvas> (واجهة برمجية [API] [proprietary] [procedural] مبادرة، على الرغم من أن الأخيرة بدأت كإضافة مملوكة WHATGW extension) قبل أن يتم ترسيمها من رد Bill Janssen.

أنظمة أخرى علينا النظر إليها والتي فيها هذا المفهوم هي Andrew وSlate، إذا إن Andrew مبني على «إدراج الأشياء»، فكل نظامٍ منها فيه نوعٌ من أنواع الإدراجه، مثل النص، أو الصور النقطية، أو الرسومات، أو الرسوم المتحركة، أو الرسائل، أو جداول البيانات ... إلخ. وتسمح أيضًا بتكرار الإدراجه ضمن العناصر، أي أن أي نوع من الأشياء القابلة للإدراجه يمكن إدراجها في أي شيء يقبل أن تدرج فيه الأشياء، فعلى سبيل المثال، يمكن إدراجه كائن ضمن أي مكان في مربع النص، أو في منطقة في لوحة الرسم، أو في أي خلية في الجدول.

«Andrew» هو إشارة إلى Andrew User Interface System (إلا أنه كان يسمى في ذاك

الوقت Andrew Project).

في ذاك الحين، وجد Thomas Fine فكرةً مختلفةً:

أنا أرى أن أفضل طريقة لتضمين الصور في الويب هي استعمال أنواع MIME. أنا متأكد من دعم صيغة postscript كنوع من أنواع MIME، وهي تتعامل بشكلٍ رائع مع دمج النصوص والصور.

لكنها غير قابلة للنقر؟ ربما أنتم محقون، لكنني أظن أن هنالك حلّاً لذلك في Display Postscript، وذلك بتعريف أمرٍ ما يُحدّد عنوان URL ويستخدم المسار (path) الحالي كمنطقة مغلقة للزر القابل للضغط. ولأن postscript تتعامل تعاملًا جيدًا مع المسارات، فإن الفكرة السابقة تجعل من السهل إضافة الروابط.

«Display Postscript» هي تقنية لعرض Postscript على الشاشة ظهرت من NeXT و Adobe.

لم يُطبق هذا الاقتراح أبدًا، لكن الفكرة أن أفضل طريقة لحل مشاكل HTML هي استبدال شيءٍ ما آخر بها ما زالت تظهر من وقتٍ لآخر.

قال Tim Berners-Lee في الثاني من آذار/مارس عام 1993:

تسمح 2 HTML للمستند أن يتضمن أي نوع يستطيع المستخدم التعامل معه، وليس فقط أنواع MIME المُسجّلة.

نعم، أنا أظن أن هنالك حالات يمكن استعمال postscript فيها مع النصوص الفائقة (hypertext)، لكن لا أعلم إن كان عرض postscript كافيًّا، أنا أعلم أنَّ Adobe تحاول إرساء استخدام صيغة PDF المبنية على postscript التي يمكن أن تتواجد فيها الروابط؛ والقابلة للقراءة عبر مجموعةٍ من عارضات الملفات الاحتكارية.

أظن أن لغةً خاصةً بإضافة الروابط (ربما مبنية على Hytime؟) قد تسمح بتطوير معايير النصوص الفائقة والرسوميات/الفيديو كلاً على حدة، مما يفيدهما ممَّا.

دع وسم `IMG` يصبح `INCLUDE` واجعله يشير إلى نوع مستندات عادي، أو إن بدت `INCLUDE` شبيهةً بعبارة `include` في لغة `cpp`، التي يتوقع فيها الناس توفير شيفرة `SGML` لكي تُفسَّر مباشرةً - وهذا بالطبع ليس الغرض منها.

كان نظام مستندات قديم مبني على `SGML`، وقد أتَّرَّ كثيرةً في النقاشات الأولية **HyTime** للغة `HTML`، ولاحقًا لغة `XML`. لم يُطبَّق اقتراح Tim لوسم `<INCLUDE>` مطلقاً، لكنك تجده صداه واضحًا في عناصر `<iframe>` و `<embed>` و `<object>`.

في النهاية، قال **Marc Andreessen** في الثاني عشر من آذار/مارس عام 1993: بالعودة إلى موضوع تضمين الصور في المستندات، اقتربتُ من إصدار `Mosaic` 0.10، الذي سيدعم تضمين صور `GIF` و `XBM` كما ذكرتُ سابقًا...

لسنا مستعدين لدعم `INCLUDE/EMBED` في هذه المرحلة... لذا سنستعمل `` (وليس `ICON`، لأنه ليس من الضروري أن تكون كل الصور المُضمنة أيقونات). لن يحدد حالياً نوع الصورة (أي `content-type`)، لكننا نفتر في دعم ذلك (مع التكيف مع أنواع `MIME`)، وفي الحقيقة، آلية قراءة الصور التي نستعملها حالياً تستطيع تحديد صيغة الصورة آلياً، لذا لن يكون امتداد الملف أية قيمة.

3. سلالة مستمرة

- أنا منبهز كثيراً بجميع نواحي المحادثة الواقعية منذ 24 عاماً، التي أدت إلى إنشاء عنصر `HTML` الذي يستعمل في كل صفحة ويب تقريباً. ضع بالحسبان أنَّ:
 - `HTTP` ما زال موجوداً، ونجح بالتطور من الإصدار 0.9 إلى 1.0 ثم إلى 1.1.
 - وما يزال يتتطور.
- `HTML` ما زالت موجودةً، صيغة البيانات البدائية تلك التي لم تكن تدعم تضمين الصور! تطورت إلى الإصدار 2.0 ثم إلى 3.2 ثم إلى 4.0، أي أنَّ خط تطويرها لم ينقطع، لكنه بالتأكيد خطٌ متلوٍ ومترعرج، وفيه العديد من النهايات المغلقة. لكن ها نحن ذا في عام 2017، وما زالت **صفحات الويب من عام 1990** تُعرض بشكلٍ جيد في المتصفحات الحديثة. ولقد فتحت إحداها في متصفح هاتفي الحديث العامل بنظام أندرويد، ولم أحصل على رسالة تقول «الرجاء الانتظار إلى أن يتم استيراد الصيغة القديمة».
- ناتجت `HTML` من النقاشات بين صانعي المتصفحات والمطوريين وواضعين للمعايير والأشخاص الذين يحبون التحدث عنها. أغلبية الإصدارات الناجحة من `HTML` كانت

عبارة عن إضفاء صفةٍ رسميةٍ لما هو موجودٌ مُحاوِلَةً توجيهه نحو الطريق الصحيح. أي شخص يخبرك أنه يجب أن تكون HTML «نقية» (بتجاهل صانعي المتصفحات أو المطوريين أو كليهما) هو شخصٌ لديه معلوماتٌ خطأً. لم تكن HTML نقيةً أبداً، وأية محاولات لجعلها كذلك قد باءت بالفشل.

- لم يبقَ أيٌّ متصفحٌ منذ عام 1993 موجوداً كما هو. فتم التخلٰي عن متصفح Netscape لم يبقَ أيٌّ متصفحٌ من الصفر لإنشاء Mozilla Suite في 1998، ثم أعيدت كتابته من الصفر لإنشاء Navigator. ثم تم الاشتغال بإنشاء Firefox. وكانت لمتصفح Internet Explorer بداياتٌ متواضعةٌ في «Microsoft Plus!» لـWindows 95، إذ حُزِّمَ مع بعض سمات سطح المكتب ولعبة pinball (لكن بالطبع يمكن تتبع تاريخ المتصفح إلى أبعد من تلك النقطة).
- ما تزال بعض أنظمة التشغيل من عام 1993 موجودةً، لكن ليس لها صلة بالويب الحديث الذي نعرفه. فأغلبية من يستخدم الويب الآن يفعل ذلك باستخدام نظام ويندوز 2000 أو أحدث منه، أو على جهاز Mac يعمل بنظام OS X، أو على حاسوب مكتبي يعمل بتوزيعٍ من توزيعات ليثكس، أو على هاتف محمول مثل هواتف أندرويد أو iPhone. لكن كان يستعمل في عام 1993 إصدار ويندوز 1.3، وكان ليثكس يوّزع عبر ...Usenet.
- بقي بعض الأشخاص موجودين وما يزالون منهمكين في العمل على ما ندعوه «معايير الويب»، فقد بقي هؤلاء الأشخاص يعملون لأكثر من 20 عاماً، وعمل بعضهم على اللغات التي تسبق HTML التي يمكن تتبع تاريخها إلى ثمانينيات القرن الماضي.

- بالحديث عن اللغات التي سبقت HTML، أصبح من السهل نسيان الصيغ والأنظمة التي كانت موجودة عند إنشاء HTML بعد الانتشار الواسع والكبير للغة HTML والويب. ماذا عن Andrew؟ أو HyTime؟ لم تكن HyTime مشروعًا بحثيًّا أكاديميًّا لا شأن لها، وإنما كانت من معايير ISO، ولقد كانت تُستعمل لأغراض عسكرية، تستطيع القراءة عنها في [هذه الصفحة](#).

لم يُجب كل ما سبق عن تساؤلنا الأصلي: لماذا لدينا عنصر ``؟ لماذا ليس عنصر `<icon>`؟ أو عنصر `<include>`؟ لماذا ليس رابطًا مع خاصية `include`، أو مجموعة من قيم الخاصية `rel`؟ لماذا عنصر ``؟ الجواب بسيطٌ للغاية، لأن Marc Andreessen قام بتنفيذ (بناء) وتوفير العنصر `` في متصفحه والذي يقوم ببناء العنصر وتوفيره في متصفحه سيربح في نهاية المطاف.

لكن هذا لا يعني أنَّ من يُنفَّذ ويُوْفَر شيفرات للعناصر يريح دائمًا، فلا تننس أنَّ Adnrew و HyTime و Intermedia و وفرَت الشيفرات أيضًا، فبرمجة الميزة هي شرطٌ لازمٌ لكنه غير كافي للنجاح؛ ولا أقصد هنا أنَّ كتابة الشيفرات قبل المعيار هي الحل الأمثل. فوسم `` لم يكن يستعمل صيغة صور شائعة، ولم يُعرِّف كيف يلتقط النص حول الصورة، ولم يكن يدعم النص البديل أو محتوى احتياطي للمتصفحات القديمة، وبعد 24 عامًا، ما زلنا نعاني مع مشكلة [Content Sniffing](#)، وما تزال هذه المشكلة مصدرًا للثغرات الأمنية. ويمكنك تتبع كل هذا إلى 24 عامًا مضت، مروًّا بحرب المتصفحات، إلى 23 شباط/فبراير عام 1993، عندما قال Marc Andreessen بشكلٍ عابر «ربما في يومٍ ما سنعتمد على أنواع MIME» ومع ذلك نشر الشيفرة التي كتبها.

4. التسلسل الزمني لتطوير HTML من 1997 إلى 2004

في كانون الأول/ديسمبر من عام 1997، نشرت رابطة الشبكة العالمية (W3C) نسخة HTML 4.0 ثم أغلقت مجموعة عمل HTML Working Group) في الحال. وبعد أقل من شهرين، نشرت مجموعة عملٍ أخرى من W3C معيار XML 1.0، وبعد ذلك بثلاثة أشهر، أنشأ القائمون على W3C ورشة عمل (workshop) باسمها «[بلورة مستقبل HTML](#)» (Shaping the Future of HTML) للإجابة عن هذا السؤال: «هل تخلت W3C عن HTML؟» وكان جوابهم: كان من المتفق عليه في النقاشات التي أجريناها أن توسيع HTML 4.0 سيكون صعباً، وكذلك تحويل 4.0 إلى تطبيق XML. الطريقة المقترحة لتجاوز هذه العوائق هي بداية جديدة للجيل القادم من HTML بناءً على مجموعة من وسوم XML.

منحت W3C مجموعة عمل HTML هذا التكليف لإنشاء «مجموعة من وسوم XML»، كانت أول خطوة -في كانون الأول/ديسمبر 1998- هي كتابة مسودة لمعيار مؤقت (انتقالي) الذي كانت مهمته إعادة قولبة HTML في XML دون إضافة أيّة عناصر أو خاصيات جديدة، عُرِفَ هذا المعيار لاحقاً باسم «[XHTML 1.0](#)» وعرّف نوع MIME جديد لمستندات XHTML «application/xhtml+xml»، لكن لتسهيل انتقال صفحات HTML 4 الموجودة من قبل، فقد تضمن المعيار «الملحق C» الذي «يلخص إرشادات التصميم للمطورين الذين يريدون تشغيل مستندات XHTML في متصفحات HTML الموجودة من قبل»، إذ قال الملحق C أنه تستطيع كتابة ما يسمى «صفحات XHTML» لكن تستطيع تقديمها عبر نوع MIME القديم .text/html

الهدف الجديد هو النماذج (forms)، ففي آب/أغسطس 1999، نشرت مجموعة عمل HTML نفسها أول مسودةٍ لنماذج XHTML الموسعة وقد وضعوا التوقعات التي يرمون إلى تنفيذها في:

أول فقرة:

بعد دراساتٍ مطولةٍ، قررت مجموعة عمل HTML أن الأهداف للجيل الجديد من النماذج تتعارض مع الحفاظ على التوافقية مع المتصفحات المصممة على النسخ القديمة من HTML. فأحد أهدافنا توفير نمط نماذج جديد (نماذج XHTML الموسعة [XHTML Extended Forms]) بناءً على مجموعةٍ من المتطلبات المحددة بدقة، وهذه المتطلبات مشروحةٌ في هذا المستند وهي مبنيةٌ على الخبرات والتجارب مع طيفٍ واسعٍ من التطبيقات التي تستعمل النماذج.

وبعد عدة أشهر أُعيدت تسمية «XHTML Extended Forms» إلى «XForms» وانتقلت إلى مجموعةٍ عمل خاصةٍ بها، وعملت هذه المجموعة على التوازي مع مجموعة عمل HTML ونشرت في النهاية الإصدار الأول من XForms 1.0 في تشرين الأول/أكتوبر 2003. وفي تلك الأثناء - وبعد اكتمال التحويل إلى XML - حُطّت مجموعة عمل HTML أنظارها إلى إنشاء «الجيل الجديد من HTML»، ففي أيار/مايو 2001، نشروا الإصدار الأول من XHTML 1.1، الذي أضاف بعض الميزات الصغيرة على 1.0، لكنه أزال الملحق C، فيجب - بدءاً من الإصدار 1.1 - تخدام مستندات XHTML بنوع MIME الخاص بها application/xhtml+xml.

5. كل شيء تعرفه عن XHTML خطأ

لماذا أنواع MIME مهمةً جداً؟ لماذا ذكرها بين الحين والآخر؟ لثلاث كلمات: draconian (أي عدم التسامح في معالجة الأخطاء). وقد تميزت المتصفحات بأنها

متさまحة مع HTML، فلو أنشأت صفحة HTML ولكن نسيت وسم الإغلاق `</head>`، فستُظهرها المتصفحات على أية حال (لأنه هناك وسوماً معينة تشير ضمنياً إلى نهاية قسم `<head>` وبداية `<body>`)، إذ يجب عليك استعمال هيكلية معينة عند تداخل الوسوم (أي إغلاق الوسوم المفتوحة بنمط « الداخل آخرًا يخرج أولًا » `(last-in-first-out)` وأن آخر وسم مستعمل سيُغلق أولًا) لكن إن كتبت `<i></i>
` فستتعامل المتصفحات معها (بطريقة ما) وستكمل عرض الصفحة دون إظهار رسالة خطأ.

أدلت عدم كتابة شيفرات HTML بطريقة سليمة تماماً إلى جعل المطورين يكتبون شيفرات غير سليمة. بعض التقديرات تقول أن أكثر من 99% من صفحات HTML على الويب فيها خطأً ما على الأقل، ولما كانت هذه الأخطاء لا تسبب عرض رسالة خطأ مرئية، فلن يصلحها أحد.

رأى W3C هذه المشكلة الأصولية في الويب، ثم همت لتصحيحها. لغة XML، المنشورة في 1997، تخلصت من العملاء المتساهلين وقالت بأن جميع البرامج التي تستعمل XML يجب أن تعامل الأخطاء المرتبطة بطريقة « الكتابة السليمة » على أنها أخطاء فادحة (Fatal Errors). اشتهر مفهوم توقف المعالجة عند أول خطأ بالاسم « draconian error handling » وذلك نسبةً إلى القائد الإغريقي **Draco** الذي شرع عقوبة الموت لأية تجاوزات (ولو صغيرة نسبياً) لقوانينه. فعندما أعادت W3C صياغة HTML بمفردات XML، اعتبرت أن جميع المستندات التي تخدم بنوع `MIME` الجديد `application/xhtml+xml` ستخضع إلى سياسة `draconian error handling` أو فلو كان هناك خطأً ما في أسلوب صياغة صفحة XHTML-نسيان وسم الإغلاق `</head>` أو تداخل غير صحيح للوسوم على سبيل المثال - فلن يكون لمتصفح الويب خيار إلا إيقاف معالجة الصفحة وإظهار رسالة خطأ إلى المستخدم النهائي.

لم تكن هذه الفكرة شائعةً على نطاقٍ واسع، لأن النسبة المقدّرة للأخطاء في صفحات الويب هي 99%， هذا يعني أن رسائل الخطأ سُتعرّض على المستخدم طوال الوقت، ولقلة الميزات التي توفرها XHTML 1.0 و 1.1، فقرر مطورو الويب تجاهل application/xhtml+xml، لكن هذا لا يعني أنهم تجاهلوا XHTML بالكلية، لأن الملحق C من معيار XHTML 1.0 أعطى مطوري الويب ثغرةً يمكنهم استعمالها: «اكتب شيئاً يشبه بنية XHTML، لكن خدمته بنوع MIME القديم XHTML «text/html»، وهذا ما فعله الآلاف من مطوري الويب، إذ «حدّثوا» صفحاتهم إلى بنية XHTML «text/html» لكن بقيت تلك الصفحات مخدّمة بنوع MIME القديم .text/html».

ولليوم، ما تزال تعلن ملايين الصفحات أنها XHTML، فيبدؤون بنوع المستند doctype الخاص بلغة XHTML في أول سطر، ويستعملون أحرفًا صغيرةً لأسماء الوسوم، ويستعملون علامات الاقتباس حول قيم الخصيّات، ويضيفون خطًا مائلاً بعد العناصر الفارغة مثل
 و <hr/> لكن قلّة قليلةً من تلك الصفحات تخدم بنوع MIME الجديد application/xhtml+xml، الذي سيُفْعَل عدم التسهيل في الأخطاء الخاص بلغة XML. فأيّة صفحة تخدم بنوع MIME القديم text/html -بغض النظر عن نوع المستند (doctype) أو بنية الوسوم- سُتُفسّر باستعمال مفسّر HTML المتساهل، وسيتم تجاهل الأخطاء دون إظهار أيّة رسالة، ودون تحذير المستخدم (أو أي شخص آخر) حتى لو كانت الصفحة فيها أخطاء تقنية.

فتحت 1.0 XHTML هذه النافذة، ولكن 1.1 XHTML أغلقتها، والنسخة التي لن تُصدر 2.0 XHTML استمرت في منهج عدم التسهيل في الأخطاء، وهذا هو السبب وراء ادعاء ملايين الصفحات على أنها 1.0 XHTML وأنّ حفنة قليلة منهم هي 1.1 (أو 2.0 XHTML)، لكن هل أنت تستعمل XHTML حقًا؟ تحقق من نوع MIME (إن لم تكن تعرف ما هو نوع MIME الذي

تستعمله، فأنا أضمن لك أنك ما زلت تستعمل `text/html`، فما لم تُخَدِّم صفحاتك بنوع MIME الجديد `application/xhtml+xml`، فإن «XHTML» هي XML بالاسم فقط.

6. رؤية تنافسية

في حزيران/يونيو 2004، أقامت W3C ورشة عمل حول تطبيقات الويب والمستندات **المجَمَعة**، حضر هذا الاجتماع ممثلوون عن مصنعي ثلاثة متصفحات، وشركات تطوير الويب، وأعضاء آخرون من W3C، ومجموعة من الجهات المهمة بما في ذلك منظمة Mozilla وشركة Opera Software، وأعطوا رؤيتهم لمستقبل الويب: **تطوير معيار HTML 4 لتضمين ميزات جديدة تساعد مطوري تطبيقات الويب الحديثة**.

تمثّل المبادئ السبعة الآتية ما نعتقد أنها أهم المتطلبات الالزمة لهذا العمل:

- التوافقية مع الإصدارات القديمة، ومسار واضح للهجرة

يجب أن تكون تقنيات تطبيقات الويب مبنيةٌ على تقنياتٍ مألوفةٍ للمطوريين، بما في ذلك HTML و CSS و JavaScript و DOM.

يجب أن تطبّق الميزات الأساسية لتطبيقات الويب باستعمال السكريبتات وصفحات الأنماط في IE6، لذا يكون هنالك مسارٌ واضح للهجرة نصب عيني المطوريين. أيٌ حلٌ لا يمكن أن يستعمل مع المتصفحات ذات الشعوبية الكبيرة حالياً دون حاجةٍ إلى استعمال إضافاتٍ لن يكون ناجحاً.

- نظام معالجة للأخطاء مُعرَّف بدقة

يجب تفصيل كيفية معالجة الأخطاء إلى درجة لا تحتاج المتصفحات فيها إلى إنشاء نظام معالجة أخطاء مخصص أو بناء واحدٍ مشتقٍ من المتصفحات الأخرى.

- لا يجب عرض الأخطاء على المستخدمين

يجب أن تحدّد المعايير السلوك اللازم اتباعه عند حدوث أي نوع من الأخطاء، ويجب أن يكون نظام معالجة الأخطاء متتساهاً (كما في CSS) بدلاً من أن يكون ظاهراً للمستخدم وكارثياً (كما في XML).

- الاستعمال العملي

يجب أن ثبّر إضافة كل ميزة إلى معايير تطبيقات الويب بحالات الاستعمال العملي لها، لكن العكس ليس صحيحاً بالضرورة: ليست كل حالة استخدام تتطلب إنشاء ميزة جديدة. ويفضّل أن تكون حالات الاستخدام ذات أساسٍ في موقع حقيقةٍ استخدمَ فيها المطوروْن حلاً ليس مثالياً للالتفاف على القصور في التقنية.

- سيبقى استخدام السكريبتات قائماً

لكن يجب الابتعاد عنه إذا توفرت وسوم مناسبة. ويجب أن لا تتدخل السكريبتات في طريقة العرض، وأن تكون مستقلةً عن الجهاز المشغل لها.

- يجب تجنب التفريق بين الأجهزة

يجب أن يتمكن المطورو من استخدام نفس الميزات الموجودة في نسخة سطح المكتب ونسخة الهواتف المحمولة لنفس المتصفح.

- عملية مفتوحة

استفاد الويب من كونه مطوّراً في بيئة مفتوحة. وستصبح تطبيقات الويب أساسية في المستقبل، ويجب أن يكون تطويرها في بيئة مفتوحةً أيضًا، ويجب إتاحة القوائم البريدية والأرشيفات ومسودات المعايير للجميع.

سئل المشاركون في ورشة العمل في استطلاعِ للرأي: «هل يجب على W3C تطوير إضافات لوظائف جاهزة في HTML و CSS، وإضافات لأساس DOM لكي تتحقق متطلبات تطوير تطبيقات الويب متوسطة التعقيد. أم عليها تطوير واجهة برمجية (API) كاملة على مستوى النظام (OS-level)؟ (هذا الاقتراح من Ian Hickson، من Opera Software)؟ كان التصويت 11 إلى 8 ضد هذا الاقتراح، وكتبت W3C في **ملخص الورشة**: «في الوقت الراهن، لا تبني W3C أن تستثمر أية موارد في العمل على موضوع استطلاع الرأي: "تطوير إضافات إلى HTML و CSS لتطبيقات الويب" في مجموعات عمل W3C الحالية».

وبعد هذا القرار، كان لدى الأشخاص الذين اقترحوا تطوير HTML ونماذج HTML خيارين فقط: الاستسلام، أو إكمال عملهم خارج إطار W3C، وقرروا المضي قدماً في الخيار الثاني، وسجلوا النطاق whatwg.org، وفي حزيران/يونيو 2004، **ولدت مجموعة عمل WHAT**.

7. مجموعة عمل WHAT

ما هي مجموعة عمل WHAT؟ سأقتبس من كلامهم:

مجموعة عمل تقنيات تطبيقات الويب (Web Hypertext Applications) هي مشاركة مفتوحة وغير رسمية لصانعي Technology Working Group متصفحات الويب والأطراف المهتمة بتطوير تطبيقات الويب. تهدف المجموعة إلى تطوير تطبيقات الويب، مع العزم على إرسال النتائج إلى المنظمات الواضعة للمعايير القياسية، مما سيشكل أساساً للعمل على توسيع HTML المعيارية.

يأتي تشكيل هذا المنتدى بعد عدّة أشهر من العمل عبر رسائل البريد الإلكتروني الخاصة حول المعايير المقترحة لمثل هذه التقنيات، الأمر الذي نرّج عليه الآن هو توسيعة نماذج HTML 4 لدعم الميزات التي يطلبها المطورون دون التسبب بمشاكل تتعلق بالتوافقية مع المحتوى الحالي. أنشئت هذه المجموعة للتأكيد على أن مستقبل تطوير هذه المعايير سيكون في قائمة بريدية مفتوحةٍ ومُؤرشفةٍ يمكن للجميع الوصول إليها.

الجملة الأساسية في الفقرة السابقة هي «دون التسبب بمشاكل تتعلق بالتوافقية»، ليست XHTML (دون الثغرة التي وفرها الملحق C) متوافقةً مع HTML، وتتطلب نوع MIME خاص بها، وXForms ليست متوافقةً مع نماذج HTML لعدم القدرة على استعمالها إلا مع الصفحات المخدّمة بنوع MIME الخاص بصفحات XHTML، هذا يعني أن XForms تتطلب عدم التساهل في التعامل مع الأخطاء.

فبدلاً من أن نرمي ما يقارب عقداً من الزمن من العمل على HTML ونجعل 99% من صفحات الويب الموجودة حالياً غير قابلة للاستخدام، قررت مجموعة عمل WHAT أن تنتهي منهجاً آخر: توثيق خوارزميات التعامل مع الأخطاء «المتسامحة» التي تستعملها المتصفحات.

كانت وما زالت المتصفحات متساهلةً مع الأخطاء HTML، لكن لم يتعد أحد نفسه ويوثق آلية التسهال بالضبط. لدى متصفح NCSA Mosaic خوارزميات خاصةً به للتعامل مع الأخطاء، وحاول Opera أن يقلّده، ثم حاول Internet Explorer تقليد Netscape، ثم حاول Firefox تقليد Safari، ثم حاول Internet Explorer تقليد Firefox، وهلم جراً حتى يومنا هذا.

إذ ضيع المطوروون آلاف الساعات محاولين جعل منتجاتهم متوافقة مع منتجات منافسيهم. قد تظن أنَّ هذا سيأخذ وقتاً طويلاً، وأنْت محق. فقد أخذ من مجموعة عمل WHAT خمسة أعوامٍ من العمل للنجاح في توثيق **كيفية تفسير HTML** (ما عدا بعض الحالات الخاصة جداً والغامضة) بطريقة متوافقةٍ مع جميع المحتوى الموجود على الويب. فلا يوجد في الخوارزمية النهائية أية حالة يتوقف فيها مفسر HTML عن إكمال تفسير صفحة HTML ويعرض رسالة خطأ للمستخدم النهائي.

وفي الفترة التي كانت تجري فيها الهندسة العكسية، كانت تعمل مجموعة WHAT بصمتٍ على أشياء أخرى أيضاً، منها معيارٌ سمي في بادئ الأمر «[Web Forms 2.0](#)» الذي أضاف بعض عناصر التحكم إلى نماذج HTML (ستتعلم المزيد عن نماذج HTML في [فصل النماذج](#)). ومسودة [Web Applications 1.0](#)» التي حوت ميزاتٍ أخرى رئيسيةٍ مثل canvas (انظر: [فصل الرسم عبر عنصر canvas](#)) والدعم المضمن لتشغيل الصوت والفيديو دون إضافات (انظر [فصل الفيديو](#)).

8. العودة إلى W3C

لمدة تقارب السنين ونصف، تجاهلت W3C ومجموعة عمل WHAT بعضها بعضاً، على الرغم من أن مجموعة عمل WHAT كانت ترتكز على نماذج الويب وميزات HTML الجديدة، ومجموعة عمل W3C مشغولة بإصدار 2.0 من XHTML، لكن بحلول تشرين الأول/أكتوبر، كان جلياً أن مجموعة عمل WHAT قد أحدثت زخماً كبيراً، بينما كانت XHTML 2.0 تقع على شكل مسودة لم يتم تطبيقها من أي متصفح رئيسي. في تشرين الأول/أكتوبر، أعلن Tim Berners-Lee مؤسس W3C أن **W3C ستعمل مع مجموعة عمل WHAT لتطوير HTML**.

أصبحت بعض الأمور واضحةً بعد عدّة سنوات، من الضروري تطوير HTML تطويرياً تدريجياً، فمحاولة جعل العالم ينتقل إلى XML - بما في ذلك وضع علامات اقتباس حول قيم الخصائص وشروط مائلة في نهاية الوسوم الفارغة، ومجالات الأسماء - لم تجد نفعاً، إذ لم يتخذ المطورون أية أفعالٍ لسببٍ رئيسي هو عدم شكوك المتصفحات. تحركت بعض المجتمعات الكبيرة وتمتعت بشمار الأنظمة المحدّدة بدقة، لكن ليس كلها.

الخطة الآن هي إنشاء مجموعة HTML جديدة. وعلى النقيض من المجموعة السابقة، مهمة هذه المجموعة هي عمل تطويرات وتحسينات تدريجية على HTML، والعمل أيضاً بالتوازي على XHTML، وسيكون لها رئيس منفصل وطاقم عمل آخر. وستعمل على HTML و XHTML معاً. لدينا دعم قوي لمثل هذه المجموعة من الكثيرين الذين تحدثنا معهم، بما في ذلك صانعي المتصفحات.

سيكون هنالك أيضًا عمل على النماذج، وهو موضوع شائك، لأن HTML و XForms هما لغتي نماذج، ونماذج HTML شائعة للغاية، وهنالك الكثير من التطبيقات والاسخدامات لنماذج XForms، وفي هذه الأثناء، وجدنا أن اقتراح HTML بالإضافة إلى نماذج HTML منطقى ومقبول فالخطة هي توسيعة نماذج HTML بالحسبان.

واحد من أول الأشياء التي قررت مجموعة عمل W3C الجديدة فعلها هي إعادة تسمية «Web Applications 1.0» إلى «HTML5»، ومن هنا بدأت رحلة هذا الكتاب.

9. حاشية

في تشرين الأول/أكتوبر 2009، أغلقت W3C مجموعة عمل XHTML 2 وأصدرت هذه الإفادة لشرح قرارها:

عندما أعلنت W3C مجموعات عمل HTML و XHTML في آذار/مارس 2007، أشرنا أننا سنكمل مراقبة استخدام XHTML 2. أدركت W3C أهمية إرسال إشارة واضحة إلى المجتمع حول مستقبل HTML.

على الرغم من أننا نقدر قيمة مساهمات مجموعة عمل XHTML على مر السنين، وبعد التشاور مع المشاركين، قررت إدارة W3C السماح بانتهاء مدة عمل المجموعة في نهاية 2009 وعدم تجديدها.

10. مراجع إضافية

Ian Hickson، مسودة قديمة كتبها [The History of the Web](#) •

Henri Sivonen، Michael Smith وآخرون، كتابتها [HTML/History](#) •

Scott Reynen كتابتها [A Brief History of HTML](#) •



لبيع وشراء الخدمات المصغرة

أكبر سوق عربي لبيع وشراء الخدمات المصغرة
اعرض خدماتك أو احصل على ما تريده بأسعار تبدأ من \$5 فقط

تصفح الخدمات

اكتشاف دعم ميزات HTML5



ربما تتساءل: «كيف أستطيع البدء باستخدام HTML5 إن لم تكن تدعمها المتصفحات القديمة؟» لكن السؤال نفسه مُضلل، فليست HTML5 شيئاً واحداً كبيراً، وإنما هي مجموعة من الميزات المتفرقة، فلا يمكنك الكشف عن «دعم HTML5» لأن هذا غير منطقي، وإنما يمكنك الكشف عن دعم الميزات المتفرقة مثل canvas أو تشغيل الفيديو أو تحديد الموقع الجغرافي.

1. تقنيات الاكتشاف

عندما يحمل متصفحك صفحة ويب، فإنه ينشئ ما نسميه «Document Object Model» (اختصاراً DOM)، الذي هو مجموعة من الكائنات التي تمثل عناصر HTML الموجودة في الصفحة، وكل عنصر -أي كل وسم <p> أو <div> أو ...- يمثل في DOM بـكائنٍ مستقل (هناك كائنات عامةٌ مثل window و document، التي لا ترتبط بعناصر محددة).

تشارك جميع كائنات DOM بمجموعةٍ من الخصائص المشتركة، لكن بعض الكائنات خصائص أكثر من بعضاها الآخر. ويكون لدى بعض الكائنات خصائصٌ فريدةٌ في المتصفحات التي تدعم ميزات HTML5؛ لذلك يكون من الكافي عادةً إلقاء نظرة على خصائص شجرة DOM لتعرف ما هي الميزات المدعومة.

هناك أربع تقنياتٍ أساسية لاكتشاف دعم المتصفح لميزة معينة، وهي بالترتيب من الأبسط إلى الأكثر تعقيداً:

1. التتحقق من وجود خاصية معينة في كائن عام (مثلاً navigator أو window)، انظر إلى

قسم **تحديد الموقع الجغرافي**.

2. إنشاء عنصر، ثم التتحقق من وجود خاصية معينة في ذاك العنصر، انظر إلى قسم **Canvas** الكشف عن دعم

. 3. إنشاء عنصر، ثم التحقق من وجود دالة (method) معينة في ذاك العنصر، ثم استدعاء

تلك الدالة والتحقق من القيمة التي تعيدها. انظر إلى قسم **معرفة صيغ**

الفيديو المدعومة.

. 4. إنشاء عنصٍ، وضبط خاصيٌّ فيه إلى قيمةٍ معينة، ثم التتحقق من احتفاظ تلك الخاصية

بقيمتها. انظر إلى قسم **معرفة أنواع حقول <input>** المدعومة.

HTML5: مكتبة اكتشاف دعم ميزات Modernizr .2

هي مكتبة JavaScript مفتوحة المصدر مُرخصة MIT مهمتها اكتشاف

الدعم للعديد من ميزات HTML5 و CSS3، وأنصحك باستعمال آخر إصدار منها دومًا.

عليك تضمينها عبر عنصر <script> في ترويسة (header) صفحتك كما يلي

لكي تستعملها:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Dive Into HTML5</title>
  <script src="modernizr.min.js"></script>
</head>
<body>
  ...
</body>
</html>
```

ستعمل مكتبة Modernizr تلقائيًّا، فلا توجد هنالك دالة () لكي

تستدعيها. فعندما يشغَل السكريبت، فإنه يُنشئ كائناً عامًّا اسمه Modernizr يحتوي على

مجموعةٍ من الخصائص المنطقية (أي True أو False) لكل ميزةٍ يمكنه الكشف عن دعمها. فعلى

سبيل المثال، إذا كان متصفحك يدعم الرسم عبر عنصر canvas، فإن قيمة الخاصية ستكون true، وإن لم يكن متصفحك يدعمها، فستكون قيمة الخاصية Modernizr.canvas .false مساويةً إلى Modernizr.canvas

```
if (Modernizr.canvas) {
    // لرسم بعض الأشكال!
} else {
    // لا يوجد دعم لخاصية canvas
}
```

3. الكشف عن دعم الرسم عبر عنصر canvas

تُعرّف HTML العنصر <canvas> على أنه «لوحة نقطية ذات أبعاد معينة يمكن استخدامها لعرض المخططات ورسومات الألعاب وغيرها من الصور المرئية برمجياً». ويمثل مستطيلاً في صفحتك حيث تُستخدم JavaScript لرسم أي شيء تريده فيه، وتُعرّف HTML5 مجموعةً من الدوال (تدعى «canvas API») لرسم الأشكال (shapes) وتعريف المسارات (paths) وإنشاء التدرجات اللونية وتطبيقات التحويلات (transformations) على العناصر.

سنستخدم التقنية الثانية للكشف عن الدعم للتحقق من دعم المتصفح للعنصر canvas، فإذا دعم متصفحك واجهة canvas البرمجية (API) فهذا يعني أن الكائن في DOM الذي يمثل عنصر canvas سيمتلك الدالة () getContext، وإن لم يكن يدعم متصفحك واجهة canvas البرمجية، فسيملك الكائن المنشأ لعنصر canvas الخاصيات العامة لكل العناصر، وليس من بينها أي شيء متعلق بتقنية canvas.

```
function supports_canvas() {
    return !!document.createElement('canvas').getContext;
}
```

تبدأ هذه الدالة عملها بإنشاء عنصر <canvas> لا حول له ولا قوة، لكن ذاك العنصر لن يرتبط مطلقاً بصفحتك، ولن يراه أحد، وإنما سيبقى عائماً بالذاكرة دون أن يفعل شيئاً.

```
return !!document.createElement('canvas').getContext;
```

وبعد إنشاء عنصر <canvas>، فسنختبر وجود الدالة getContext()، إذ ستكون هذه الدالة موجودةً إذا كان المتصفح يدعم واجهة canvas البرمجية.

```
return !!document.createElement('canvas').getContext;
```

وفي النهاية، استعملنا علامة النفي المزدوجة (!!) كي تكون النتيجة قيمةً منطقيةً (أي .false أو true

```
return !!document.createElement('canvas').getContext;
```

تكتشف هذه الدالة الدعم لأغلبية مكونات واجهة canvas البرمجية، بما في ذلك **الأشكال**، **والمسارات**، **والدرجات اللونية والأنماط**. لكنها لا تكتشف وجود المكتبة الخارجية explorerCanvas التي تسمح باستخدام واجهة canvas البرمجية في الإصدارات القديمة من Internet Explorer متصفح .

وبدلاً من كتابة الدالة السابقة بنفسك، تستطيع استعمال Modernizr للكشف عن دعم واجهة canvas البرمجية.

```

if (Modernizr.canvas) {
    // لرسم بعض الأشكال !
} else {
    // ): canvas لا يوجد دعم لخاصية
}

```

هناك اختبار منفصل للتحقق من دعم واجهة canvas text البرمجية.

4. الكشف عن دعم النصوص في عنصر Canvas

حتى لو كان متصفحك يدعم واجهة canvas البرمجية، فقد لا يدعم واجهة canvas text البرمجية، فتُثقلت واجهة canvas البرمجية على مر الزمن، وأضيّقت دوال النصوص في فترة لاحقة، لهذا هناك بعض المتصفحات التي تدعم canvas لكن قبل أن يكتمل العمل على دوال النصوص.

سنستخدم **التقنية الثانية** للكشف عن الدعم للتحقق من دعم المتصفح للعنصر canvas، فإن دعم متصفحك واجهة canvas البرمجية (API) فهذا يعني أن الكائن في DOM الذي يمثل عنصر canvas **سيملك الدالة** (`getContext()`، وإن لم يكن يدعم متصفحك واجهة `<canvas>` البرمجية، فسيملك الكائن المنشأ لعنصر `<canvas>` **الخصائص العامة** لكل العناصر، وليس من بينها أي شيء متعلق بتقنية `.canvas`

```

function supports_canvas_text() {
    if (!supports_canvas()) { return false; }
    var dummy_canvas = document.createElement('canvas');
    var context = dummy_canvas.getContext('2d');
    return typeof context.fillText == 'function';
}

```

تبدأ الدالة السابقة **بالتحقق من دعم العنصر canvas** باستخدام الدالة

`supports_canvas()` التي رأيتها في القسم السابق، فإن لم يكن يدعم متصفحك واجهة `canvas` البرمجية، فهو بالتأكيد لن يدعم إضافة النصوص إلى عناصر `canvas`.

```
if (!supports_canvas()) { return false; }
```

ثم سنشئ عنصر `<canvas>` جديد ثم نحاول الوصول إلى «لوحة الرسم» (`drawing`)

قد `supports_canvas()`، ومن المؤكد أن ما سبق سيعمل دون مشاكل لأن الدالة (`context`) تتحقق من وجود الدالة (`getContext()` في جميع عناصر `canvas`).

```
var dummy_canvas = document.createElement('canvas');
var context = dummy_canvas.getContext('2d');
```

وفي النهاية، ستحقق إذا كان لدى لوحة الرسم الدالة (`fillText()`، فإذا كانت تملكها `canvas` فهناك دعم للنصوص في

```
return typeof context.fillText == 'function';
```

وبدلاً من كتابة الدالة السابقة بنفسك، تستطيع استعمال **Modernizr** للكشف عن دعم واجهة `canvas text` البرمجية.

```
if (Modernizr.canvasText) {
    // لنسع بعض النصوص!
} else {
    // (): canvas لا يوجد دعم لكتابة النصوص في عناصر
}
```

5. الكشف عن دعم الفيديو

أضافت HTML5 عنصراً جديداً هو <video> لتضمين مقاطع الفيديو في صفحات الويب، كان تضمين الفيديو في السابق من المستحيلات دون استخدام إضافات خارجية مثل Adobe Flash® أو Apple QuickTime®. صمم عنصر <video> ليستعمل دون الحاجة إلى أية سكريبتات للكشف عن الدعم، فيمكنك تحديد عدّة مسارات لمقاطع الفيديو، وستختار المتصفحات التي تدعم HTML5 video أحدها بناءً على الصيغة التي تدعمها.

المتصفحات التي لا تدعم HTML5 video ستتجاهل وسم <video> تماماً. لكنك تستطيع استخدام ذلك لصالحك بإخبارها أن **تشغل المقطع** باستخدام إضافة خارجية. برمج Kroc Camen حلاً اسمه **Video for Everybody!** الذي يستخدم دعم الفيديو في HTML5 عند توفره، لكنه سيعود إلى استخدام Flash أو QuickTime أو JavaScript مطلقاً، ويعمل نظرياً على أي متصفح، بما في ذلك متصفحات الهواتف المحمولة. إذا أردت القيام بأكثر من مجرد وضع الفيديو في صفحتك وتشغبله، فستحتاج إلى استخدام JavaScript، نستخدم **التقنية الثانية للتحقق من دعم الفيديو**. فإذا كان متصفحك يدعم video HTML5 الذي سينشره المتصفح لتمثيل عنصر <video> سيملك الخاصية `canPlayType()`. وإن لم يكن يدعم متصفحك video HTML5، فإن كائن DOM المنشأ لعنصر <video> سيملك **الخصائص العامة لأي عنصر**. يمكنك التحقق من دعم الفيديو عبر هذه الدالة:

```
function supports_video() {
    return !!document.createElement('video').canPlayType;
}
```

وبدلاً من كتابة الدالة السابقة بنفسك، تستطيع استعمال Modernizr للكشف عن دعم .HTML5 video

```
if (Modernizr.video) {
    // لتشغيل مقاطع الفيديو!
} else {
    // لا يوجد دعم للفيديو:
    // ربما علينا استخدام QuickTime أو Flash بدلاً منه
}
```

سأشرح -في **الفصل الخاص بالفيديو**- حلا آخر يستعمل تقنيات الكشف السابقة لتحويل عناصر <video> إلى مشغلات فيديو مبنية على تقنية Flash، وذلك لتشغيل الفيديو على المتصفحات التي لا تدعم .HTML5 video

هناك اختبار منفصل للتحقق من صيغ الفيديو التي يمكن للمتصفح تشغيلها، مشروح في الفقرة الآتية.

6. صيغ الفيديو

مثلاً صيغ الفيديو كمثيل اللغات المكتوبة، فقد تحتوي صحيفة إنكليزية على المعلومات نفسها التي تحتويها صحيفة عربية، لكن إن كنت تجيد قراءة اللغة العربية فقط، فستكون إحدى تلك الصحفتين مفيدةً لك. ولتشغيل مقطع فيديو، يجب أن يفهم المتصفح «اللغة» التي كتب فيها هذا المقطع.

تسمى «اللغة» التي تكتب فيها مقاطع الفيديو «بالمرماز» (codec) الذي هو الخوارزمية المستخدمة لترميز (encode) مقطع الفيديو إلى سلسلة من البتات، وهناك عدّة مرمازات، لكن أيها تستعمل؟ في الواقع، من المؤسف أن المتصفحات لم تتوافق على مرماز معين، لكنهم قللوا الخيارات إلى خيارين فقط. أحدهما احتكاري وكان يكلف مالاً (بسبب براعة الاختراع)، لكنه يعمل في متصفح Safari وفي iPhone (وهو يعمل أيضاً في مشغلات Flash إن كنت تستعمل حلاً مثل Video for Everybody!). أما المرماز الآخر فهو مجاني وي العمل على المتصفحات مفتوحة المصدر مثل Firefox و Chromium.

نستخدم التقنية الثالثة لمعرفة صيغ الفيديو المدعومة. وإذا كان متصفحك يدعم HTML5 video، فإن كائن DOM الذي سيئشه المتصفح لتمثيل عنصر <video> سيملك الخاصية canPlayType(). ستخبرك الطريقة الآتية إذا كان يدعم متصفحك صيغة فيديو معينة. تتحقق هذه الدالة من دعم الصيغة المحمية بحقوق براعة الاختراع والمدعومة من أجهزة iPhone و Mac.

```
function supports_h264_baseline_video() {
    if (!supports_video()) { return false; }
    var v = document.createElement("video");
    return v.canPlayType('video/mp4; codecs="avc1.42E01E,
    mp4a.40.2"');
}
```

تبداً هذه الدالة بالتحقق من دعم تشغيل الفيديو عبر HTML5 في المتصفح، وذلك باستخدام الدالة supports_video() التي رأيتها في القسم السابق. فإن لم يكن يدعم متصفحك video، فهو بالتأكيد لن يدعم أية صيغة من صيغ الفيديو.

```
if (!supports_video()) { return false; }
```

ثم سنشئ الدالة عنصر <video> (لكن دون أن تضيفه إلى الصفحة، أي أنه لن يكون مرئياً) ونستدعي الدالة supports_video()، من المؤكد وجود هذه الدالة لأن الدالة canPlayType() تحقق منها في الخطوة السابقة.

```
var v = document.createElement("video");
```

في الواقع، «صيغة الفيديو» هي مجموعة من عدّة أشياء، فبكلام تقني، أنت تسأل المتصفح إن كان يستطيع تشغيل فيديو H.264 بنمط Baseline مع صوت AAC LC في حاوية MPEG-4 (سنشرح ما يعنيه ما سبق بالتفصيل في [فصل الفيديو](#)).

```
return v.canPlayType('video/mp4; codecs="avc1.42E01E, mp4a.40.2"');
```

لن ثعيد الدالة canPlayType() القيمة true أو false، وإنما ستعيد سلسلة نصيةً آخذةً بالحسبان الطبيعة المعقّدة لصيغ الفيديو:

- «probably» إذا كان المتصفح واثقاً أنه يستطيع تشغيل هذه الصيغة
 - «maybe» إن ظن المتصفح أنه يستطيع تشغيل هذه الصيغة
 - «» (سلسلة نصية فارغة) إن كان المتصفح متأكداً أنه لن يستطيع تشغيل هذه الصيغة
- في النهاية، WebM هي صيغة فيديو جديدة ومفتوحة المصدر (وغير محمية ببراءة اختراع) دعمت في الإصدارات الجديدة من المتصفحات الحديثة، بما في ذلك Chrome و Firefox، و Opera. ويمكنك استخدام التقنية السابقة نفسها لاكتشاف دعم صيغة WebM.

```
function supports_webm_video() {
  if (!supports_video()) { return false; }
  var v = document.createElement("video");
  return v.canPlayType('video/webm; codecs="vp9, vorbis"');
}
```

بدلاً من كتابة هذه الدالة بنفسك، يمكنك استعمال Modernizr (الإصدار 1.5 أو ما بعده)

لاكتشاف الدعم لمختلف صيغ الفيديو في HTML5.

```
if (Modernizr.video) {
  // لتشغيل مقاطع الفيديو! لكن ما صيغتها؟
  if (Modernizr.video.webm) {
    // لتجرب WebM
  } else if (Modernizr.video.h264) {
    // لتجرب H.264 video + AAC audio
  }
}
```

7. التخزين المحلي

يوفر **التخزين المحلي** لموقع الويب طريقةً لتخزين المعلومات على حاسوبك ثم استعادتها

لاحقاً؛ مفهوم التخزين المحلي مشابه لمفهوم «الكوكيز» (cookies)، لكنه مصمم لكمية معلوماتٍ أكبر، فالكوكيز محدودة المساحة، وينرسلُها المتصفح إلى خادوم الويب في كل مرة يطلب فيها صفحة جديدة (مما يأخذ وقتاً أطول ويستهلك بيانات التراسل). أما تخزين HTML5 فهو يبقى في حاسوبك، وتستطيع موقع الويب الوصول إليه عبر JavaScript بعد أن يتم تحميل الصفحة.

س: هل التخزين المحلي هو جزءٌ من HTML5؟ ولماذا وضعيه في معيار منفصل؟

ج: الجواب المختصر هو «نعم»، التخزين المحلي هو جزءٌ من HTML5. أما الجواب الأطول فهو أن التخزين المحلي كان جزءاً من معيار HTML5 الرئيسي، لكنه أصبح معياراً منفصلاً بسبب شكوك بعض الأشخاص في مجموعة عمل HTML5 أن HTML5 أصبحت كبيرة جدًا. حتى لو كان ذلك يشبه تقسيم شطيرة إلى قطع صغيرة لتقليل كمية الحريرات التي تتناولها :-| أهلاً بك في العالم العجيب للمعايير القياسية.

سنستخدم التقنية الأولى لاكتشاف الدعم للتحقق من دعم المتصفح للتخزين المحلي، فإن دعم متصفحك للتخزين المحلي، فستكون هناك خاصية `window.localStorage` في كائن `window`. وإن لم يكن يدعم متصفحك التخزين المحلي، فلن تكون الخاصية `localStorage` معرفةً. وبسبب علة في الإصدارات القديمة من Firefox، سيسبب هذا الخيار حدوث استثناء (`exception`) إن كانت الكعكات (`cookies`) مُعطَّلةً، لذلك وضعنا الاختبار في عبارة `.try..catch`

```
function supports_local_storage() {
  try {
    return 'localStorage' in window && window['localStorage'] !!
== null;
  } catch(e){
    return false;
  }
}
```

بدلًا من كتابة هذه الدالة بنفسك، يمكنك استعمال Modernizr (الإصدار 1.1 أو ما بعده) لاكتشاف الدعم للتخزين المحلي.

```

if (Modernizr.localstorage) {
    // window.localStorage متوفرة !
} else {
    // لا يوجد دعم للتخزين المحلي :(
    // ربما تجرب Gears أو مكتبة أخرى
}

```

لاحظ أنَّ JavaScript حساسة لحالة الأحرف، إذ تدعى خاصية Modernizr «localStorage» (جميعها بأحرفٍ صغيرة)، أما خاصية DOM فهي «localStorage» (حرف S كبير).

س: ما مدى أمان قاعدة بيانات التخزين المحلي في HTML5؟ هل يستطيع أحد قراءتها؟
ج: أي شخص لديه وصولٌ فيزيائيٌّ لحاسوبك قد يستطيع عرض (أو حتى تعديل) البيانات الموجودة في قاعدة بيانات التخزين المحلي في HTML5. ويستطيع أي موقع ويب قراءة وتخزين القيم الخاصة به، لكن لا يستطيع الوصول إلى القيم التي خرَّتها المواقع الأخرى، وهذا يسمى [.same-origin restriction](#).

Web Workers .8

توفر ميزة Web Workers طريقةً قياسيةً لتشغيل JavaScript في الخلفية، إذ تستطيع تشغيل عدَّة «خيوط» (threads) التي تعمل في الوقت نفسه تقريباً (تذكر طريقة تشغيل الحاسوب لعدَّة تطبيقات معًا في آنٍ واحد)، تستطيع تلك الخيوط التي تعمل في الخلفية أن تجري عمليات حسابية معقدة، أو أن تجري طلبيات شبكة، أو أن تصل إلى [ال تخزين المحلي](#) في أثناء استجابة صفحة الويب إلى تفاعل المستخدم معها.

نستعمل طريقة الاكتشاف الأولى لمعرفة إن كان المتصفح يدعم واجهة Web Worker البرمجية، وذلك إن وجدت الخاصية Worker في الكائن العام window، وإن لم يكن يدعم متصفحك واجهة Web Worker البرمجية، فستكون خاصية Worker غير معرفة.

```
function supports_web_workers() {
    return !!window.Worker;
}
```

بدلاً من كتابة هذه الدالة بنفسك، يمكنك استعمال Modernizr (الإصدار 1.1 أو ما بعده) لاكتشاف الدعم لواجهة Web Workers البرمجية.

```
if (Modernizr.webworkers) {
    // window.Worker متوفرة !
} else {
    // (): Web Workers لا يوجد دعم لواجهة
    // ربما تجرب Gears أو مكتبة أخرى
}
```

لاحظ أن JavaScript حساسة لحالة الأحرف، إذ تدعى خاصية Modernizr «webworkers» (جميعها بأحرفٍ صغيرة)، أما خاصية DOM window.Worker فهي (حرف W كبير في Worker).

9. تطبيقات الويب دون اتصال

يمكن ببساطة قراءة صفحات الويب الثابتة دون اتصال: اتصل إلى الإنترنت، حمل صفحة الويب، اقطع اتصالك بالإنترنت، ثم سافر إلى بلد آخر، واقرأ الصفحة في وقت فراغك (يمكنك تخطي خطوة السفر إلى بلد آخر لتوفير الوقت). لكن ماذا عن تطبيقات الويب مثل Gmail أو

Google Docs تطبيقات ويب تعمل دون اتصال. HTML5، التي تمكّن الجميع (وليس فقط Google) من بناء

تبعد تطبيقات الويب التي تعمل دون اتصال كتطبيقات تعمل بوجود اتصال بالإنترنت، فهي أول مرة تزور فيها تطبيق ويب يدعم العمل دون اتصال، فسيخبر الخادم متصل بك ما هي الملفات التي يحتاج لها كي يعمل دون اتصال. قد تكون هذه الملفات من أي نوع: صفحات HTML، أو JavaScript، أو الصور أو حتى مقاطع الفيديو. وبعد أن ينزل متصل بك كل الملفات الضرورية تستطيع إعادة زيارة موقع الويب حتى لو لم تكن متصلة بالإنترنت، وسيلاحظ متصل بك أنك غير متصل وسيستعمل الملفات التي نزلها من قبل. وعندما تتصل مجدداً بالإنترنت، فيمكن رفع أيّة تعديلات أجريتها على خادوم الويب البعيد.

نستعمل طريقة الاكتشاف الأولى لمعرفة إن كان المتصفح يدعم تشغيل تطبيقات الويب دون اتصال، وذلك إن وجدت الخاصية applicationCache في الكائن العام window، وإن لم يكن يدعم متصل بك تشغيل تطبيقات الويب دون اتصال، فستكون خاصية applicationCache غير معرفة؛ يمكنك التحقق من دعم تشغيل تطبيقات الويب دون اتصال مع هذه الدالة:

```
function supports_offline() {
    return !!window.applicationCache;
}
```

بدلاً من كتابة هذه الدالة بنفسك، يمكنك استعمال Modernizr (الإصدار 1.1 أو ما بعده) لاكتشاف الدعم لتشغيل تطبيقات الويب دون اتصال.

```

if (Modernizr.applicationcache) {
    // window.applicationCache متوفرة !
} else {
    // لا يوجد دعم للتطبيقات دون اتصال :(
    // ربما تجرب Gears أو مكتبة أخرى //
}

```

لاحظ أنَّ JavaScript حساسة لحالة الأحرف، إذ تدعى خاصية Modernizr «applicationcache» (جميعها بـأحرفٍ صفيرية)، أما خاصية DOM فهي .(Cache) كبير في window.applicationCache.

10. تحديد الموقع الجغرافي

يفيد تحديد الموقع الجغرافي في معرفة أين أنت في هذا الكوكب و (اختيارياً) مشاركة تلك المعلومات مع الأشخاص الذين تثق بهم، هنالك أكثر من طريقة لمعرفة أين أنت: عبر عنوان IP، أو عبر اتصال شبكتك اللاسلكية، أو أي برج تغطية خلوية تتصل منه، أو عبر عتاد GPS الذي يحسب إحداثيات موقعك الحالي عبر المعلومات التي ترسلها الأقمار الاصطناعية في السماء.

س: هل تحديد الموقع الجغرافي جزءٌ من HTML5؟ ولماذا تتحدث عنه إذًا؟

ج: لقد أضيف دعم تحديد الموقع الجغرافي من المتصفحات مع ميزات HTML5 الجديدة. لكن إذا ابتعينا الدقة، يُوفِّر معيار تحديد الموقع الجغرافي من **مجموعة عمل Geolocation**، التي هي مجموعة عمل منفصلة عن مجموعة عمل HTML5، لكننا سنتحدث عن تحديد الموقع الجغرافي في هذا الكتاب على أية حال، لأنَّه جزءٌ من التطوير الذي يحدث في الويب في الوقت الراهن.

نستعمل طريقة الاكتشاف الأولى لمعرفة إن كان المتصفح يدعم واجهة تحديد الموقع الجغرافي البرمجية، وذلك إن وُجدت الخاصية `geolocation` في الكائن العام `navigator`، وإن لم يكن يدعم متصفحك تحديد الموقع الجغرافي، فستكون خاصية `geolocation` غير معروفة. يمكنك التتحقق من دعم تحديد الموقع الجغرافي مع هذه الدالة:

```
function supports_geolocation() {
    return !!navigator.geolocation;
}
```

بدلاً من كتابة هذه الدالة بنفسك، يمكنك استعمال Modernizr لاكتشاف الدعم لتحديد الموقع الجغرافي.

```
if (Modernizr.geolocation) {
    // لنكتشف أين أنت الآن!
} else {
    // لا يوجد دعم لتحديد الموقع الجغرافي: (
    // ربما تجرب Gears أو مكتبة أخرى //
}
```

إن لم يدعم متصفحك واجهة تحديد الموقع الجغرافي البرمجية داخلياً، فلا تيأس. فهناك التي هي إضافة مفتوحة المصدر للمتصفحات من Google التي تعمل على ويندوز و Mac OS X ولينكس والهواتف العاملة بنظامي ويندوز وأندرويد (أعلنت Google أخيراً إيقاف تطوير هذه المكتبة). إذ توفر ميزاتٍ للمتصفحات القديمة التي لا تدعم الأشياء الجديدة التي تحدثنا عنها في هذا الفصل. إحدى الميزات التي توفرها Gears هي تحديد الموقع الجغرافي، لكنها ليست مطابقة لواجهة `navigator.geolocation` البرمجية، لكنها تخدم نفس الغرض.

سيشرح **الفصل الخاص بتحديد الموقع الجغرافي** بالتفصيل كيفية استخدام مختلف الواجهات البرمجية السابقة.

11. أنواع الإدخال في النماذج

أنت تعرف الكثير عن نماذج الويب، صحيح؟ أنشئ عنصر `<form>` ثم أضف بعض عناصر `<input type="password">` إليه وربما عنصر `<input type="text">`، ثم أنه النموذج بزر `.<input type="submit">`.

حسناً، ذلك جزءٌ يسيراً من النماذج، إذ أضافت HTML5 حوالي ثلاثة عشر نوعاً من أنواع المدخلات التي يمكنك استعمالها في نماذجك.

- | | |
|----|---|
| 1 | <code><input type="search"></code> لحقول البحث |
| 2 | <code><input type="number"></code> لإدخال الأرقام |
| 3 | <code><input type="range"></code> للمزلاج (slider) لتحديد مجال من الأعداد |
| 4 | <code><input type="color"></code> لاختيار الألوان |
| 5 | <code><input type="tel"></code> لأرقام الهواتف |
| 6 | <code><input type="url"></code> لعناوين الويب |
| 7 | <code><input type="email"></code> لعناوين البريد الإلكتروني |
| 8 | <code><input type="date"></code> التقويم لاختيار التاريخ |
| 9 | <code><input type="month"></code> للأشهر |
| 10 | <code><input type="week"></code> للأسابيع |

.11 `<input type="time">` للوقت والتاريخ بدقة

.12 `<input type="datetime">` لتحديد الوقت والتاريخ بدقة

.13 `<input type="datetime-local">` للوقت والتاريخ المحليين

سنستخدم **التقنية الرابعة** لاكتشاف أنواع الحقول المدعومة في النماذج. في البداية

سننشئ عنصر `<input>` في الذاكرة، وسيكون نوع الحقل الافتراضي لجميع عناصر `<input>` هو "text"، وسيوضح لك لماذا هذا مهمً جدًا.

```
var i = document.createElement("input");
```

ثم سنضبط خاصية `type` في عنصر `<input>` إلى نوع حقل الإدخال الذي تريد معرفة إن كان مدعومًا أم لا.

```
i.setAttribute("type", "color");
```

إن كان يدعم متصفحك نوع حقل الإدخال المعين، فستحتفظ خاصية `type` بالقيمة التي ضبطتها، أما إن لم يكن يدعم متصفحك نوع الحقل المعين، فسيتجاهل القيمة التي ضبطتها وستبقى قيمة الخاصية `type` مساويةً إلى "text".

```
return i.type !== "text"
```

بدلاً من كتابة 13 دالة منفصلة يدوياً، تستطيع استخدام Modernizr لاكتشاف الدعم لجميع أنواع حقول الإدخال المعرفة في HTML5.

تعيد مكتبة Modernizr استخدام عنصر `<input>` وحيد لكي تكتشف ما هي أنواع حقول الإدخال المدعومة. ثم تبني جدولًا من نوع `hash` باسم `Modernizr.inputtypes` يحتوي على

13 مفتاح (خواص `type` في HTML5) و 13 قيمة منطقية (أي `true` إذا كان الحقل مدعوماً، أو `false` إن لم يكن كذلك).

```
if (!Modernizr.inputtypes.date) {
    // ): <input type="date">
    // ربما تستعمل مكتبة Dojo أو jQueryUI
}
```

12. النص البديل

بالإضافة إلى **أنواع حقول الإدخال الجديدة**، تضمنت HTML5 بعض الإضافات الصغيرة للنماذج. أحدها هو إمكانية وضع نص بديل (`placeholder`) في حقل الإدخال.

يُعرض النص البديل في حقل الإدخال طالما كان الحقل فارغاً، وبمجرد أن تكتب شيئاً في الحقل فسيختفي ذاك النص البديل. هناك لقطات للشاشة في **فصل النماذج** يمكنك النظر إليها إن واجهت صعوبةً في تحيله.

سنستخدم **التقنية الثانية للكشف عن الدعم** للتحقق من دعم المتصفح للنص البديل في حقول الإدخال، فإن دعم متصفحك النص البديل فهذا يعني أن الكائن في DOM الذي يمثل عنصر `<input>` سيملك الخاصية `placeholder` (حتى لو لم تضع خاصية `placeholder` في شيفرة HTML)، وإن لم يكن يدعم متصفحك النص البديل، فلن يملّك الكائن المنشأ لعنصر `.placeholder`.

```
function supports_input_placeholder() {
    var i = document.createElement('input');
    return 'placeholder' in i;
}
```

بدلاً من كتابة هذه الدالة بنفسك، يمكنك استعمال Modernizr (الإصدار 1.1 أو ما بعده) لاكتشاف الدعم للنص البديل في حقول الإدخال.

```
if (Modernizr.input.placeholder) {
    // يمكنك استعمال النص البديل في حقول الإدخال!
} else {
    // لا يوجد دعم للنص البديل :(
    // استعمل سكريبت لفعل ذلك //
}
```

13. التركيز التلقائي على النماذج

تستعمل مواقع الويب JavaScript للتركيز (focus) على حقل من حقول الإدخال في نماذج الويب. على سبيل المثال، الصفحة الرئيسية لمحرك البحث Google سترُكِّز تلقائياً (auto-focus) على حقل البحث كي تستطيع كتابة ما الذي تريده البحث عنه مباشرةً دون الحاجة إلى النقر على حقل الإدخال، ربما هذا ملائم للكثيرين، لكنه مزعج للمستخدمين المتقدمين أو لأولى الاحتياجات الخاصة، فإن ضغطت على زر المسافة (space) متوقعاً أن تُمرر إلى الأسفل، فلن يتم ذلك، لوجود المؤشر في حقل إدخال (وستكتب مسافة فارغة في حقل الإدخال بدلاً من التمرير)، وإن ركزت على حقل إدخال مختلف في أثناء تحميل الصفحة، فسيحرك سكريبت التركيز التلقائي التركيز إلى الحقل المحدد بعد إكمال تحميل الصفحة، مما يؤدي إلى كتابتك في المكان الخطأ. ولأن التركيز التلقائي كان يُعَذَّب عبر JavaScript، فمن الصعب التعامل مع جميع الحالات، وليس هنالك ملجاً من التركيز التلقائي على الحقول لمن لا يريد ذلك!

ولحل هذه المشكلة، قدّمت HTML5 **خاصية autofocus** في جميع عناصر نماذج الويب. ووظيفة هذه الخاصية واضحةً من اسمها: نقل التركيز إلى حقل إدخال معين. ولأنها شيفرة HTML بدلًا من كونها سكريبت، فإن سلوكها سيكون متناغمًا ومتماثلًا في كل موقع الويب، ويمكن لصانعي المتصفحات (أو مطوري الإضافات) توفير طريقة لكي يُعطل المستخدمون إمكانية التركيز التلقائي.

سنستخدم **التقنية الثانية للكشف عن الدعم** للتحقق من دعم المتصفح للتركيز التلقائي في حقول الإدخال، فإن دعمً متصرفُك التركيز التلقائي فهذا يعني أن الكائن في DOM الذي يُمثل عنصر `<input>` سيملك الخاصية `autofocus` (حتى لو لم تضع خاصية `autofocus` في شيفرة `<input>`)، وإن لم يكن يدعم متصرفُك التركيز التلقائي، فلن يملك الكائن المنشَأ لعنصر `autofocus`. يمكنك اكتشاف دعم التركيز التلقائي عبر هذه الدالة:

```
function supports_input_autofocus() {
  var i = document.createElement('input');
  return 'autofocus' in i;
}
```

بدلًا من كتابة هذه الدالة بنفسك، يمكنك استعمال Modernizr (الإصدار 1.1 أو ما بعده) لاكتشاف الدعم للتركيز التلقائي في حقول الإدخال.

```
if (Modernizr.input.autofocus) {
  التركيز التلقائي يعمل! //
} else {
  التركيز التلقائي غير مدعوم: (
  // استعمل سكريبت لفعل ذلك //
}
```

14. البيانات الوصفية

البيانات الوصفية (Microdata) هي الطريقة القياسية لتوفير هيكلية معنوية لصفحات الويب. على سبيل المثال، يمكنك استعمال البيانات الوصفية لتصريح أن صورةً ما مرخصةً بإحدى رخص المشاع الإبداعي. وكما سترى في [الفصل العاشر](#)، يمكنك استعمال البيانات الوصفية لتوصيف صفحة «معلومات عنِّي»، فيمكن للمتصفحات -أو إضافات المتصفحات- أو محركات البحث تحويل تلك البيانات الوصفية إلى [vCard](#)، التي هي صيغة معيارية لمشاركة معلومات الاتصال؛ يمكنك أيضًا تعريف أنواع خاصة بك من البيانات الوصفية.

معيار البيانات الوصفية في HTML5 يتضمن شيفرات HTML (تستعملها محركات البحث بشكلٍ أساسي) ومجموعة من دوال DOM (تستعملها المتصفحات بشكلٍ أساسي). لا حرج في تضمين البيانات الوصفية في صفحات الويب، فهي مجرد خاصيات ذات معنى خاص، وستتجاهلها محركات البحث التي لا تستطيع تفسير البيانات الوصفية. لكن إن كنت تريد الوصول إلى أو تعديل البيانات الوصفية عبر DOM، فعليك أن تتحقق أن متصفحك يدعم واجهة البيانات الوصفية البرمجية (API).

نستعمل طريقة [اكتشاف الأولى](#) لمعرفة إن كان المتصفح يدعم واجهة البيانات الوصفية البرمجية، وذلك إن وجدت الدالة `(document.getItems)` في الكائن العام `getItems`، وإن لم يكن يدعم متصفحك البيانات الوصفية، فستكون الدالة `(getItems)` غير معرفة.

```
function supports_microdata_api() {
    return !!document.getItems;
}
```

بدلًا من كتابة هذه الدالة، يمكنك استعمال Modernizr لاكتشاف الدعم للبيانات الوصفية.

```

if (Modernizr.microdata) {
    // هنالك دعم للبيانات الوصفية!
} else {
    // البيانات الوصفية غير مدعومة:
}

```

15. التأريخ

واجهة التأريخ البرمجية في HTML5 هي طريقة معيارية لتعديل تاريخ (history)

المتصفح عبر السكريبتات، جزء من هذه الواجهة -التنقل في التأريخ- كان متوفراً في الإصدارات السابقة من HTML. أما الجزء الجديد في HTML5 هو طريقة إضافة مدخلات جديدة إلى تاريخ المتصفح، والاستجابة عندما تُحذف تلك المدخلات عبر ضغط المستخدم لزر الرجوع، وهذا يعني أن معرف URL سيبيّق يعمل عمله كمُعَرّف فريد للمورد الحالي، حتى في التطبيقات التي تعتمد اعتماداً كبيراً على السكريبتات التي لا تجري عملية تحديث لكاملا الصفحة.

نستعمل طريقة **اكتشاف الأولى** لمعرفة إن كان المتصفح يدعم واجهة التأريخ البرمجية،

وذلك إن وجدت الدالة `pushState()` في الكائن العام `history`، وإن لم يكن يدعم متصفحك واجهة التأريخ البرمجية، فستكون الدالة `pushState()` غير معرفة.

```

function supports_history_api() {
    return !(window.history && history.pushState);
}

```

بدلاً من كتابة هذه الدالة بنفسك، يمكنك استعمال Modernizr (الإصدار 1.6 أو ما بعده)

لاكتشاف الدعم لواجهة التأريخ البرمجية.

```

if (Modernizr.history) {
    يمكنك تعديل تاريخ المتصفح! //
} else {
    لا يوجد دعم لتعديل التاريخ: (
    // استعمل مكتبة مثل History.js
}

```

16. مراجع إضافية

المعايير:

- **<canvas>**
- **<video>**
- **<input>** أنواع حقول الإدخال
- **<input placeholder>** الخاصية
- **<input autofocus>** الخاصية
- **التخزين المحلي في HTML5**
- **Web Workers**
- **تطبيقات الويب التي تعمل دون اتصال**
- **واجهة تحديد الموقع الجغرافي البرمجية**
- **التاريخ**

مكتبات JavaScript:

- **Modernizr**, مكتبة اكتشاف الدعم لميزات HTML5
- **geo.js**, مكتبة لإضافة الدعم لواجهة تحديد الموضع

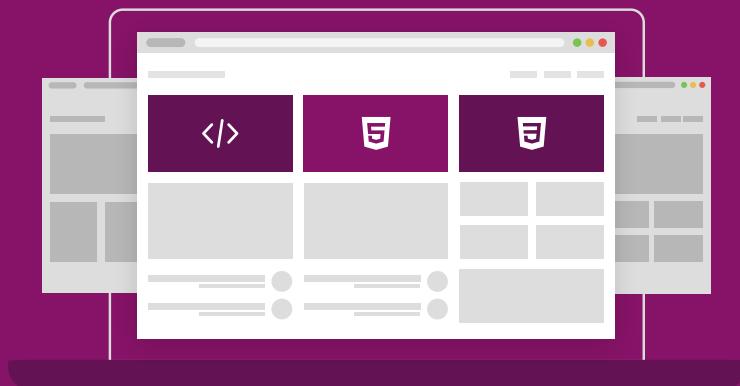
Video for Everybody! •

مقالات أخرى مفيدة:

معاملات أنواع الفيديو •

دليل Internet Explorer 9 للمطورين •

دورة تطوير واجهات المستخدم



ابداً مسارك المهني كمطور واجهات المواقع والمتاجر الإلكترونية
فور انتهاءك من الدورة

التحق بالدورة الآن



٣

البنية الهيكلية لمستندات HTML5

سأأخذ -في هذا الفصل- صفحة HTML موجودة مسبقاً خالية من الأخطاء ثم سُسْخّنها؛ وستصبح أجزاء منها أقصر، وبعضاً أطول، لكنها ستصبح أكثر تنظيماً من قبل. هذه هي **الصفحة المقصودة**، تأملها ملياً، ولا تكمل القراءة إلا بعد أن تنظر إلى مصدرها (عبر «عرض المصدر») مرةً واحدةً على الأقل.

1. نوع المستند – Doctype

من بداية المستند:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

هذا ما ندعوه «نوع المستند» (أو doctype). هناك تاريخٌ طويلاً يقف خلف نوع المستند. فوجد مطورو مايكروسوفت أثناء عملهم على متصفح Internet Explorer 5 لنظام ماك أنفسهم أمام مشكلة مفاجئة: لقد حسّنوا النسخة الجديدة من المتصفح في دعمها للمعايير (standards) كثيراً إلى حدِ جعل الصفحات القديمة لا تظهر بشكلٍ سليم، إذ طوّرت تلك الصفحات بناءً على التجاوزات (quirks) التي سمحَت بها المتصفحات الرئيسية في ذاك الوقت، خصيصاً Internet Explorer 4 و Netscape 4. لقد كان IE5/Mac متطوراً للغاية إلى درجة أنه لم يستطع تشغيل الصفحات القديمة!

أتت مايكروسوفت بحلٍ مبتكر، فقبل عرض الصفحة سينظر متصفح IE5/Mac إلى «نوع المستند» (doctype) الذي يكون عادةً في أول سطر من مستند HTML (وحتى قبل عنصر <html>)، فالصفحات القديمة (التي كانت تعتمد على التجاوزات التي تسمح بها المتصفحات القديمة) لم تكن تملك سطر doctype على الإطلاق، وسيعرضها متصفح IE5/Mac كما كانت

تفعل المتصفحات القديمة. ولكن يتم «تفعيل» دعم المعايير الجديدة، كان على مطوري الصفحات أن يضيفوا نوع المستند الصحيح قبل وسم <html>.

انتشرت هذه الفكرة كالنار في الهشيم، وأمسى لجميع المتصفحات الرئيسية نمط التجاوزات (quirks mode) و «نمط المعايير» (standards mode). ولأننا نعلم أن الأمور تخرج عن السيطرة بسهولة في الويب: حاولت Mozilla أن تطلق الإصدار 1.1 من متصفحها، لكن المطوريين اكتشفوا أن هنالك صفحات تُعرض بنمط المعايير (standards mode) إلا أنها تعتمد على تجاوز (quirk) وحيد، الذي أزالته Mozilla من محرك العرض الخاص بها، وبهذا تعطلت آلاف الصفحات في آنٍ واحد، ولهذا أشّؤوا ما يسمى «نمط المعايير التقريري» (almost standards mode).

في مقالته الرائعة «Activating Browser Modes with Doctype»، لـ Henri Sivonen، مختل الأنظمة:

نط التجاوزات - Quirks Mode

تنتهك المتصفحات معايير الويب لكي تتجنب عرض الصفحات - التي طُوّرت اعتقاداً على الممارسات الشائعة في نهاية التسعينيات من القرن الماضي - بشكل خطأ.

نط المعايير - Standards Mode

تحاول المتصفحات أن تعامل المستندات تبعاً للمعايير القياسية، تدعوه HTML5 هذا النط «بالنمط الحالي من التجاوزات» (no quirks mode).

نمط المعايير التقريري - Almost Standards Mode

يوجد في Firefox و Safari و Opera (منذ الإصدار 7.5) و IE8 نمط اسمه «نمط المعايير التقريري» التي لا يَتَّبِعُ فيه القياس الرأسي (vertical) لخلايا الجداول معيار CSS2، تدعوه HTML5 هذا النمط «بنمط التجاوزات المحدودة» (limited quirks mode).

ملاحظة: عليك أن تقرأ بقية المقالة السابقة، لأنني بسطتها كثيراً هنا. يجدر بالذكر وجود عدّة أنواع مستندات (doctypes) في IE5/Mac لم تكن تصنّف على أنها متوافقة مع المعايير، فقد ازدادت قائمة التجاوزات مع مرور الزمن، وكذلك ازداد عدد أنواع المستندات التي تسبب استعمال «نمط التجاوزات»، ففي آخر مرة حاولت فيها إحصاءها، كان هنالك 5 أنواع مستندات تسبب استعمال «نمط المعايير التقريري» و 73 تسبب استعمال «نمط التجاوزات» لكنني أظن أنني نسيت إحصاء بعضها. ولست هنا في صدد الحديث عن ما يفعله Internet Explorer 8 للتبديل بين أنماط العرض الأربع! هذا **رسمٌ توضيحيٌ له**. أين كنا؟ تذكريت، نوع المستند:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

صدق وأن كان نوع المستند السابق هو أحد الأنواع الخمسة عشر التي تُفْقَلُ «نمط المعايير» في جميع المتصفحات الحديثة. ليس هنالك أي خطأ فيه، فإن أحببت نوع المستند السابق، فاتركه كما هو، أو بدله إلى نوع مستند HTML5، الذي هو أقصر وأجمل ويفْعَلُ «نمط المعايير» أيضاً في جميع المتصفحات الحديثة. هذا هو:

```
<!DOCTYPE html>
```

إنه بسيط جدًا، و تستطيع كتابته يدوياً بسهولة دون أن تخطئ فيه.

2. العنصر الجذر

صفحة HTML هي سلسلة من العناصر المترادفة، إذ تشبه بنية الصفحة الشجرة. فبعض العناصر هي «أخوة» (siblings) كعنصري شجرة يتفرعان من نفس الفرع، ويمكن لبعضها أن يكون «ابنًا» (child) لبعضها الآخر، كفرع صغير يمتد من فرع كبير (والعكس صحيح؛ العنصر الذي يحتوي بقية العناصر اسمه «الأب» [parent] للعناصر التي تكون «أبناء» مباشرين له، أو أن يكون «الجد» [ancestor] لأحفاده). العناصر التي ليس لها أبناء تسمى «الأوراق» (leaf)، أما العنصر الأساسي الذي هو الجد لكل العناصر الباقية في الصفحة فيسمى «العنصر الجذر» (root) الذي هو الوسم `<html>` دوماً.

يبدو العنصر الجذر في [صفحتنا السابقة](#) كما يلي:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      lang="en"
      xml:lang="en">
```

لا يوجد شيء خطأ فيه، و تستطيع الإبقاء عليه، وهو صحيح تماماً في HTML5، لكنه بعض أجزائه لم تعد ضرورية في HTML5، و تستطيع حذفها.

أول شيء علينا مناقشه هو خاصية `xmlns`، التي هي من بقايا [XHTML 1.0](#)، وتقول أن العناصر في هذه الصفحة تكون في مجال أسماء (namespace) XHTML، لكن العناصر في [HTML5 تكون دوماً في مجال الأسماء السابق](#)، لذلك لست بحاجة إلى التصريح عن ذلك.

فستعمل صفحة HTML5 في جميع المتصفحات بنفس الطريقة سواءً كانت هذه الخاصية موجودةً أم لا.

سنحصل على عنصر الجذر الآتي بعد حذف خاصية `:xml:lang`:

```
<html lang="en" xml:lang="en">
```

الخاصيتان `lang` و `xml:lang` تُعرّفان لغة صفحة HTML (إذ ترمز `en` إلى الإنكليزية و `ar` للعربية؛ وإن كانت لغة صفتتك مختلفةً، فراجع [الصفحة الآتية](#))؛ لكن لماذا لدينا خاصيتين بنفس الاسم؟ هذا أيضًا من بقايا XHTML، فخاصية `lang` هي التي تملك التأثير في HTML5، ويمكنك إبقاء خاصية `xml:lang` إن شئت، لكن إن فعلت ذلك عليك الحرص على [أن تماشى](#) قيمتها قيمة الخاصية `lang`.

هل ت يريد التخلص منها؟ حسناً، هذا يترك لنا العنصر الجذر الآتي:

```
<html lang="en">
```

3. العنصر `head`

يكون عادةً أول «أولاد» العنصر الجذر هو العنصر `<head>`، الذي يحتوي على بياناتٍ وصفية (metadata)؛ أي تلك المعلومات حول الصفحة، بدلاً من «جسم» الصفحة نفسه (إذ يكون جسم الصفحة محتوىً في عنصر `<body>`). العنصر `<head>` نفسه بسيطٌ ولم يتغير في HTML5، لكن الذي تغير هو الأشياء التي تحتوي ضمنه، ولهذا سنلقي نظرةً إلى [الصفحة السابقة](#):

```
<head>
  <meta http-equiv="Content-Type" content="text/html;
  charset=utf-8" />
  <title>My Weblog</title>
```

```

<link rel="stylesheet" type="text/css" href="style-
original.css" />
<link rel="alternate" type="application/atom+xml"
      title="My Weblog feed"
      href="/feed/" />
<link rel="search"
      type="application/opensearchdescription+xml"
      title="My Weblog search"
      href="opensearch.xml" />
<link rel="shortcut icon" href="/favicon.ico" />
</head>

```

.<meta بعنصر

4. ترميز المحارف

عندما تحاول تخيل النصوص فربما تفكّر في «مجموعة من المحارف والرموز التي تراها على شاشة الكمبيوتر» لكن الحواسيب لا تتعامل مع المحارف والرموز وإنما مع البتات والبايتات، فكل قطعة من النص شاهدتها على شاشة حاسوبك تكون مخزنةً بترميز محارف character (encoding) معين. وهناك **المئات من مختلف ترميزات المحارف**، بعضها مخصص للغات معينة مثل العربية أو الصينية أو الإنكليزية، ويمكن أن يستعمل بعضها الآخر في عددٍ لغات؛ وبشكل عام، يوفر ترميز المحارف ربطاً بين الأشياء التي تراها على شاشتك والأشياء التي يخزنها الكمبيوتر في الذاكرة أو على القرص.

في الواقع، الأمر معقدٌ أكثر من ذلك بكثير، فقد يظهر نفس المحرف في أكثر من ترميز، لكن من المحتمل أن يستعمل كل ترميز سلسلة مختلفة من البايتات لتخزين المحرف في الذاكرة أو على القرص؛ أي يمكنك افتراض أنَّ ترميز المحارف هو مفتاح لك تشفير النصوص. فإذا ما أعطاك أحدهم سلسلةً من البايتات واصفاً إياها «بالنص»، فعليك أن تعرف ما هو ترميز المحارف

الذي عليك استعماله لكي تفك الترميز وتعرف أي محرف عليك عرضه (أو إجراء العمليات عليه، أو أيًا كان غرضك).

إذاً كيف يعرف متصفحك ما هو ترميز المحارف للبيانات التي يُرسلها خادوم الويب؟ أنا مسروء لأنك سألت، فإن كنت تعرف ترويسات HTTP (أي HTTP headers)، فستشاهد ترويسةً كالآتية:

```
Content-Type: text/html; charset="utf-8"
```

باختصار، تقول الترويسة السابقة أنَّ خادوم الويب يظن أنه يُرسل مستند HTML، ويظن أنَّ المستند يستعمل ترميز UTF-8. وللأسف. لا يملك إلا قلةً من مطوري الويب تحكمًا بخادوم HTTP. خذ Blogger مثلاً: يوفر أشخاص عديدون المحتوى، لكن الخواديم مداراة من Google؛ ولهذا السبب وفرت HTML طريقةً لتحديد ترميز المحارف في مستند HTML نفسه، لذلك ستشاهد شيئاً شبيهاً بما يلي:

```
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
```

باختصار، يقول السطر السابق أنَّ مطور الصفحة «يظن» أنه كتب صفحة HTML باستخدام ترميز UTF-8.

ما تزال كلا الطريقتين تعملان في HTML5، لكن يفضل استعمال ترويسة HTTP، وستلغي تأثير وسم `<meta>` إن وجدت؛ لكن لا يستطيع كل شخص أن يضبط ترويسات HTTP، لذلك بقي وسم `<meta>` موجوداً، لكنه أصبح أسهل في HTML5، فيمكنك كتابته كالتالي:

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

يعمل السطر السابق في جميع المتصفحات، لكن كيف أتي الشكل المختصر؟ هذا هو **أفضل**

توضيح استطعت إيجاده:

السبب المنطقي لإضافة خاصية charset في `<meta charset="">` هي أنَّ المتصفحات تستعملها داخليًّا، لأنَّ المطورين عادةً ينسون إضافة علامات اقتباس كما في:

```
<META HTTP-EQUIV=Content-Type CONTENT=text/html;
charset=ISO-8859-1>
```

هناك عدّة اختبارات ل洲 `<meta charset>` إن أردت الاطلاع عليها.

س: أنا لا أستعمل المحارف الغريبة، هل يجدر بي تحديد ترميز المحارف في صفحاتي؟

ج: نعم، عليك دومًا أن تحدد ترميز المحارف في كل صفحة HTML تخدمها، لأن عدم تحديد الترميز قد يؤدي إلى ثغراتٍ أمنية.

الخلاصة: ترميز المحارف هو موضوعٌ معقد، ولم يُبسط خالٍ عقوِّي تطوير البرمجيات، وعليك دومًا تحديد ترميز المحارف في كل صفحة HTML، أو ستحصل لك أشياءً مريعة. يمكنك تحديد ترميز المحارف عبر ترويسة Content-Type في HTTP، أو عبر `<meta http-equiv>`، أو الصيغة المختصرة `<meta charset>`، لكن احذر أن تنساها لمصلحة الجميع.

5. الروابط العلاقية

الروابط العاديَّة (`<a href>`) تُشير ببساطة إلى صفحة أخرى، أما الروابط العلاقية (Link relations) فهي طريقة لتفسير لماذا تشير إلى صفحةٍ أخرى، وهي تنهي الجملة «أنا أشير إلى

الصفحة الآتية لأنها...»

- لأنها صفحة أنماط (stylesheet) تحتوي على قواعد CSS يجب على متصفحك تطبيقها على هذا المستند.
- لأنها تغذية (feed) تحتوي على محتوى الصفحة الحالية نفسه، لكن بصيغة قياسية قابلة للاشتراك (subscribable).
- لأنها ترجمة لهذه الصفحة بلغة أخرى.
- لأنها تحوي محتوى الصفحة الحالية نفسه، لكن بصيغة PDF.
- لأنها تمثل الفصل القادم من كتاب إلكتروني، الذي تكون الصفحة الحالية جزءاً منه.
- وهكذا. تقسم HTML5 الروابط العلاقة إلى **تصنيفين**:

يمكن إنشاء روابط من كلا التصنيفين عبر عنصر link. **الروابط إلى الموارد الخارجية** (external resources) هي روابط إلى موارد يمكن أن تستعمل لإضافة أشياء إلى المستند الحالي، أما **الروابط التشعبية** (hyperlink links) فهي روابط إلى مستندات أخرى. ...

يعتمد السلوك الذي سينفذ للروابط إلى الموارد الخارجية على العلاقة relationship التي تربط المورد الخارجي بالمستند الحالي.

أول الأمثلة التي ذكرتها أعلاه هو رابط إلى مورد خارجي (`rel="stylesheet"`)؛ أما البقية فهي روابط تشعبية (hyperlinks) إلى مستندات أخرى، إذ تستطيع تتبع تلك الروابط إذا أردت ذلك، لكنها غير مطلوبة لعرض الصفحة الحالية.

غالبًا ما نرى الروابط العلاقية في عناصر `<link>` في قسم `<head>` من الصفحة. ويمكن أن تستعمل الروابط العلاقية في عناصر `<a>` لكن هذا أمر غير شائع على الرغم من أنه مسموح. تسمح HTML5 أيضًا بوجود بعض العلاقات في عناصر `<area>` لكن هذا أقل شيوعًا أيضًا (لم تسمح HTML 4 بخاصية `rel` في عناصر `<area>`؛ انظر إلى [جدول لكامل الروابط العلاقية](#) لترى إن كان بإمكانك استعمال قيمة معينة لخاصية `rel`.

س: هل يمكنني أن أنشئ الروابط العلاقية الخاصة بي؟

ج: يبدو أن هناك أفكارًا لا تنتهي عن روابط علاقية جديدة. وفي محاولة للحد من عمل [أشياء عبئية](#)، قررت مجموعة عمل WHAT أن تنشئ سجلًا بقيم `rel` المفترضة وتعزّف آليًّا الموافقة عليها.

١. العلاقة `rel=stylesheet`

لنلق نظرةً إلى أول رابط علاقي في [صفحتنا](#):

```
<link rel="stylesheet" href="style-original.css"
      type="text/css" />
```

هذا أكثر رابط علاقي مستعمل في العالم (حرفياً). يشير عنصر `link` `rel="stylesheet"` إلى قواعد CSS المخزنة في ملف منفصل. يمكنك تحسين السطر السابق قليلاً في HTML5 بحذف خاصية `type` لأن هناك لغة وحيدة لأنماط التنسيق في الويب ألا وهي CSS، ولهذا تكون هي القيمة الافتراضية للخاصية `type` وهذا هو السلوك الافتراضي في جميع المتصفحات (ربما يأتي أحدهم ويختار لغة جديدة لأنماط التنسيق في يومٍ ما، عندها استعمل الخاصية `type` مجددًا).

```
<link rel="stylesheet" href="style-original.css" />
```

ب. العلاقة **rel = alternate**

نأمل عملنا مع **صفحتنا**:

```
<link rel="alternate"
      type="application/atom+xml"
      title="My Weblog feed"
      href="/feed/" />
```

هذا الرابط العلقي شائع أيضاً. يُفَعِّل عنصر `<link rel="alternate">` ذو نوع وسائط `feed` أو `RSS` في خاصية `type`-`discover` شيئاً يدعى «الاكتشاف التلقائي للتغذية» (`auto`) التي تسمح للتطبيقات التي تُعرض التغذية (مثل `feedly`) أن تكتشف أنّ موقعاً ما لديه تغذية (`feed`) لآخر المقالات التي تُنشر فيه. تدعم بعض المتصفحات الاكتشاف التلقائي للتغذية عبر عرض أيقونة خاصة بجوار URL (وعلى عكس `rel="stylesheet"`، خاصية `type` لها تأثيرٌ هنا، لذلك لا تنس وضعها).

للرابط العلقي `rel="alternate"` حالات استخدامٍ غريبة بعض الشيء، حتى في `HTML 4`. ثم تم توضيح التعريف في `HTML5` وتوسيعه ليصف بدقة محتوى الويب الموجود. وكما رأيت سابقاً، استعمال `rel="alternate"` مع `type=application/atom+xml` يشير إلى تغذية Atom للصفحة الحالية، لكن يمكنك أيضاً استخدام `rel="alternate"` مع قيم أخرى للخاصية `type` للإشارة إلى نفس المحتوى لكن بصيغة أخرى مثل `.PDF`.

وضعت `HTML5` حداً للجدل القائم حول كيفية إنشاء روابط إلى ترجمات أخرى للمستند. تقول `HTML 4` باستخدام خاصية `lang` بالإضافة إلى `rel="alternate"` لتحديد لغة المستند

المُشار إليه بالوصلة؛ لكن هذا غير صحيح. فملحق تصحيح أخطاء 4 HTML (HTML 4 Errata) يسرد أربعة أخطاء صريحة في معيار 4 HTML. أحد تلك الأخطاء هو طريقة تحديد لغة المستند الموصول إليه عبر "rel="alternate"، فالطريقة الصحيحة المذكورة في ملحق تصحيح الأخطاء ومن بعده في HTML5 هي استخدام الخاصية hreflang؛ لكن للأسف لم تدمج تلك التصحيحات في معيار 4 HTML الرسمي، لأنه لم يعد أحدُ في مجموعة عمل W3C HTML يعمل على HTML في ذاك الوقت.

ج. الروابط العلائقية الأخرى في HTML5

تستعمل العلاقة "rel="author" لعمل رابط إلى معلوماتٍ حول مؤلف الصفحة، يمكن أن يكون هذا عنوان بريد إلكتروني mailto: لكن ذلك ليس ضروريًا؛ إذ يمكن أن يكون ببساطة رابط إلى نموذج اتصال أو إلى صفحة «عن المؤلف».

لكن المستند الحالي يُشَغِّل جزءاً منه -أي الموقع-، أظن أن هذه العلاقة انتشرت في البداية عبر WordPress التي تستعملها في الروابط (links) التي يتركها المعلقون.

عرَّفت 4 HTML العلاقات "rel="prev" و "rel="next" و "rel="start" و "rel="end" للإشارة إلى العلاقات بين الصفحات التي تكون جزءاً من سلسلة (مثل فصول كتابٍ ما، أو سلسلة مقالات في مدونة). العلاقة الوحيدة التي أُستعملت استعمالاً صحيحاً هي "rel="next" و "rel="begin" و "rel="previous" و "rel="last" و "rel="first"؛ بدلاً من "rel="prev" و "rel="next" واستعملوا "rel="begin" و "rel="end" بدلاً من "rel="start" و "rel="end"؛ حتى أنهم اخترعوا علاقةً جديدةً "rel="up" للإشارة إلى الصفحة «الأب».

ضَمَّنَت HTML5 العلاقَة "rel="first" التي كانت أشهر طريقة لقول «أول صفحة في السلسلة» ("rel="start") هي علاقَة ليست موجودة في الموصفة، لكن تم توفيرها للتوفيقية). وضَمَّنَت أيضًا rel="previous" و rel="next" HTML كما في 4 rel="prev" (و دعمت أيضًا rel="last" أي آخر مقال في السلسلة) و rel="up" (أي آخر مقال في السلسلة) و rel="up" هي النظر إلى قائمة التنقل التفصيلية (breadcrumb) أفضل طريقة لتخيل rel="up" هي النظر إلى قائمة التنقل التفصيلية (navigation) أو على الأقل تخيلها؛ ربما تكون الصفحة الرئيسية هي أول صفحة في قائمة التنقل، والصفحة الحالية في نهاية التفرعات. تُشير العلاقة "rel="up" إلى أقرب صفحة تعلو الصفحة الحالية في قائمة التنقل.

علاقَة rel="icon" هي ثانية أكثر علاقَة انتشاراً بعد rel="stylesheet"، وهي تأتي عادةً مع علاقَة shortcut كما في المثال الآتي:

```
<link rel="shortcut icon" href="/favicon.ico">
```

جميع المتصفحات الرئيسية تدعم طريقة الاستخدام السابقة لعرض أيقونة صغيرة خاصة بالصفحة، وتَظَهُر عادةً في شريط عنوان المتصفحات بجوار URL، أو في لسان (tab) الصفحة، أو كلاهما.

ومن الجديد في HTML5: قابلية استعمال الخاصية sizes مع الرابط العلقي icon لتحديد أبعاد الأيقونة المشار إليها.

ابتكِر مجتمع microformats علاقَة "rel="license" التي تعني أنَّ المستند المشار إليه يحتوي على شروط رخصة حقوق النشر التي يخضع لها هذا المستند.

العلاقة "rel="nofollow" «تُشير إلى أنَّ ناشر الصفحة ليس مسؤولاً عن محتويات الرابط، أو أنَّ الرابط موضوع في الصفحة لوجود علاقة تجارية بين الصفتين»، وتم ابتكارها من Google، ثم أصبحت من المعايير ضمن مجتمع microformats. أضافت ووردبريس العلاقة rel="nofollow" إلى الروابط المقضاة من المعلقين. كان التفكير السائد هو إن كانت روابط rel="nofollow" لا تؤثر على ترتيب الصفحة (PageRank)، فسيستسلم المخربون ولن يكملوا حماولاتهم بإرسال تعليقات عشوائية (spam) في المدونات؛ لكن هذا لم يحصل، ولكن بقيت العلاقة "rel="nofollow" موجودةً.

العلاقة "rel="noreferrer" «تشير إلى عدم السماح بإعطاء معلومات عن الصفحة المرجعية (referrer) عند اتباع الرابط».

العلاقة "rel="pingback" تحدُّد عنوان خادوم «pingback»، كما هو مشرح في مواصفة Pingback: «نظام pingback هو طريقة لمدونات الويب لكي تحصل على إشعار تلقائياً عندما pingback تُشير إليها موقع الويب الأخرى». طبَّقت برمجيات التدوين (لا سيما ووردبريس) آلية pingback لإشعار الكتاب عندما تضيف رابط لهم عند إنشاء تدوينة جديدة.

العلاقة "rel="prefetch" «تُشير إلى أنَّ الجلب المسبق وتخزين المورد (resource) المحدَّد سيكون مفيداً في غالب الأمر، لأنَّه من المحتمل أنَّ المستخدم سيستفيد من ذاك المورد»، `<link rel="prefetch" href="URL of top search result">` تُضيف محركات البحث أحياً نتائج البحث إن شعرت أنَّ أول نتيجة شائعة أكثر من غيرها.

العلاقة "rel="search" «تعني أنَّ المستند المشار إليه يوفرواجهةً مخصصة للبحث في المستند الحالي والمصادر المتعلقة به». إذا أردت فعلاً أن تستفيد من العلاقة "rel="search" ،

فيجب أن تشير إلى مستند **OpenSearch** الذي يصف كيف يجب أن تكون تركيبة وصلات URL للبحث في الموقع الحالي عن كلمة معينة. مستندات OpenSearch (وروابط `rel="search"`) مدعومة في Internet Explorer (OpenSearch) منذ الإصدار السادس، وفي Mozilla Firefox منذ الإصدار الثاني.

العلاقة `rel="tag"` تعني أنَّ الوسم (tag) الذي يمثله المستند المُشار إليه يُطبق على المستند الحالي». تم ابتكار إضافة الوسوم (أي الكلمات المفتاحية للمقالات) عبر خاصية `rel` من **Technorati** لمساعدتهم في تصنيف مقالات المدونة. كانت تشير إليها التدوينات والدورس القديمة بالمصطلح «Technorati tags» (هذا صحيح! إذ أقنعت شركة تجارية العالم بأسره بإضافة حقل بيانات وصفية الذي سهل العمل على الشركة). ثم **أصبح جزءاً من المعايير في مجتمع microformats**, إذ أطلقوا عليه `rel="tag"`. أغلبية أنظمة التدوين التي تسمح بارتباط التصنيفات (categories) والوسوم (tags) بمقالات مخصصة ستتعلّمها بعلاقة `rel="tag"`, لكن المتصفحات لا تفعل أي شيء خاص عند وجود هذه العلاقة؛ لكن صُممَت لمحركات البحث لكي تستعملها كإشارة أو علامة على محتوى الصفحة.

6. العناصر البنوية الجديدة في HTML5

لا يقتصر عمل HTML5 على تقصير طول الشيفرات (على الرغم من أنها تفعل ذلك في مواطن عديدة)، لكنها عرَّفَت أيضًا عناصر بنوية جديدة.

`<section>`

يمثّل العنصر `section` مستندًا عموميًّا (`generic`), والقسم (`section`) في هذا السياق هو `(sections)` تجميعٌ موضوعيٌّ للمحتوى، يأتي عادةً مع ترويسة. ربما نستطيع تشبيه الأقسام

بالفصول (chapters)، أو بالصفحات في مربع فيه أكثر من لسان (tab)، أو بمجموعة من الأقسام في أطروحة (أو رسالة). يمكن تقسيم الصفحة الرئيسية لموقع الويب إلى عدّة أقسام للتمهيد وللأخبار ولمعلومات التواصل.

<nav>

يُمثل العنصر `nav` قسماً من الصفحة الذي يحتوي روابط إلى صفحات أخرى أو إلى أجزاء من الصفحة الحالية. ليس من الضروري أن تكون جميع مجموعات الروابط في الصفحة موجودة في عنصر `nav`، فالعنصر `nav` يلائم الأقسام التي تحتوي على عناصر التنقل الرئيسية فقط. خصوصاً أنه من الشائع وضع قائمة مختصرة من وصلات الصفحات الشائعة في الموضع في التذييل (footer)، مثل شروط الخدمة، والصفحة الرئيسية، وصفحة حقوق النشر. ويكون العنصر `footer` في هذه الحالات كافياً دون الحاجة إلى العنصر `nav`.

<article>

يُمثل العنصر `article` مكوّناً من مكونات الصفحة التي تحتوي على مجموعة من العناصر في مستند أو صفحة أو تطبيق أو موقع، والتي يمكن توزيعها بشكلٍ مستقل عن بقية الصفحة أو إعادة استخدامها في أماكن أخرى. أمثلة عن هذا العنصر تتضمن: مواضيع المنتديات، أو مقالات في مجلة أو صحيفة، أو منشور في مدونة، أو تعليق نشره مستخدّم ما، أو أيّة عناصر مستقلة من المحتوى.

<aside>

يُمثل العنصر `aside` قسماً من صفحةٍ ما يتضمن محتوى مرتبط بشكلٍ ما بالمحظوظ الذي يحيط بعنصر `aside`، الذي يمكن اعتباره منفصلاً عن ذاك المحتوى. تمثّل مثل هذه الأقسام عادةً بشريط أو بمربع جانبي (sidebar) في الكتب المطبوعة.

يمكن استخدام هذا العنصر لخدمة نفس غرض الشريط الجانبي في الكتب المطبوعة في عرض الاقتباسات، أو للإعلانات، أو لمجموعة من عناصر `nav`، أو لأي محتوى آخر يمكن اعتباره منفصلاً عن محتوى الصفحة الرئيسية.

<hgroup>

يُمثل العنصر `hgroup` ترويسة قسمٍ ما. يُستعمل هذا العنصر لتجميع عناصر `h1-h6` عندما تحتوي الترويسة على أكثر من مستوى، مثل العناوين الفرعية، أو العناوين البديلة، أو الشعار (tagline).

<header>

يُمثل العنصر `header` مجموعةً من العناصر التمهيدية أو التي تُساعد في التنقل، ويحتوي عادةً العنصر `header` على ترويسة القسم (عنصر `h1-h6` أو عنصر `hgroup`)، لكن هذا ليس ضرورياً. يمكن أن يُستخدم العنصر `header` أيضاً لاحتواء جدول المحتويات، أو نموذج البحث، أو الشعارات (logos) بأنواعها.

<footer>

يُمثل العنصر `footer` تزييلاً لأقرب عنصر أب فيه محتوى، أو للعنصر الجذر (`root`). يحتوي التزييل عادةً على معلومات حول القسم المرتبط به مثل من الذي كتبه، وروابط إلى المستندات ذات الصلة، أو معلومات حقوق النشر، وأشياء أخرى من هذا القبيل. ليس من الضروري أن تظهر التزييلات في نهاية كل قسم من المحتوى. وعندما يحتوي عنصر `footer` على أقسام كاملة، فسيُمثل حينها ملحقاً أو فهرساً أو كلمة ختامية للكاتب أو الشروط المفصلة للرخصة، وما شابه ذلك.

<time>

يُمثّل العنصر `time` الوقت إما بنظام 24 ساعة، أو التاريخ الدقيق في التقويم الغريغوري المبكر (proleptic Gregorian calendar)، ويمكن أن يحتوي اختيارياً على إزاحة للمنطقة الزمنية (time-zone offset).

<mark>

يُمثّل العنصر `mark` نصاً مُعلماً (marked) لإشارة إليه.

أنا متأكد أنك متخصص ومتلهف للبدء في استخدام العناصر الجديدة السابقة، وإن فلم تكن لتقرأ هذا الفصل، لكن قبل ذلك عليناأخذ رحلة صغيرة.

7. تفصيل كيف تتعامل المتصفحات مع العناصر غير المعروفة

لدى كل متصفح قائمة بعناصر HTML التي يدعمها، فقائمة متصفح Firefox -على سبيل المثال- مخزنة في ملف `nsElementTable.cpp`، تُعامل العناصر غير الموجودة في هذه القائمة كعناصر غير معروفة. هناك مشكلتان أساسيتان مع العناصر غير المعروفة:

1. **كيف يجب أن تُنسق؟** افتراضياً، لدى العنصر `<p>` مسافة تفصله عن العناصر في الأعلى وفي الأسفل، أما `<blockquote>` فله محاذاة مع هامش (margin) أيسر، ويُعرض العنصر `<h1>` بخطٍ أكبر، لكن ما هي الأنماط (styles) الافتراضية التي يجب تطبيقها على العناصر غير المعروفة؟

2. **كيف سيبدو كائن DOM للعنصر؟** يتضمن ملف `nsElementTable.cpp` معلوماتٍ عن العناصر التي تستطيع احتواء عناصر أخرى، فلو كتبت شيفرة مثل `<p><p>`، فسيغلق وسم الفقرة (`p`) الثاني الوسم الأول، وبهذا سيصبح العنصران «أخوة» ولن يكونا «أب» و«ابن». لكن إن كتبت `<p>`، فلن يُغلق الوسم `span` الفقرة، لأنَّ المتصفح

في مثاناً) يعلم أنّ <p> هو عنصر «كتلي» (block) ويمكن أن يحتوي على عنصر «سطري» (inline) مثل , لذلك سينتهي المطاف بعنصر «ابنًا» للعنصر <p> في شجرة DOM.

تعجب المتصفحات على هذه الأسئلة بطرائق مختلفة (أعلم أنّ هذا أمر صادم). ومن بين جميع المتصفحات الرئيسية، يجرب Internet Explorer على كلا السؤالين بأكثر طريقة تسبب مشاكل! لكن جميع المتصفحات تحتاج بعض المساعدة في هذا المجال.

من السهل نسبيًا الإجابة على أول سؤال: لا تنسّق العناصر غير المعروفة بأي تنسيق مميز. اتركها «تراث» (inherit) خاصيات CSS، واترك الأمر لمطور الصفحة ليحدد التنسيق اللازم عبر CSS. وهذا يعمل في غالب الأوقات لكن هناك أمر صغير عليك أن تراعيه.

ملاحظة: تعرض جميع المتصفحات العناصر غير المعروفة كعناصر سطورية (inline)، أي لأن لديها قاعدة CSS الآتية:

هناك عناصر جديدة في HTML5 مُعرَّفة كعناصر كتيلية، أي أنها تستطيع احتواء العناصر الكتيلية الأخرى، وستنسقها المتصفحات التي تدعم HTML5 بالخاصية `display: block` افراضياً.

إذا أردت استخدام تلك العناصر في المتصفحات القديمة، فعليك إعادة تعريف خاصية `display` يدوياً:

```
article,aside,details,figcaption,figure,
footer,header,hgroup,menu,nav,section {
    display:block;
}
```

(أخذت الشيفرة السابقة من [HTML5 Reset Stylesheet](#) التي لها وظائف أخرى اعتبرها خارجةً عن نطاق هذا الفصل).

لكن انتظر قليلاً، لم يكن متصفح Internet Explorer قبل الإصدار التاسع يطبق أية تنسيدات إلى العناصر غير المعروفة، فمثلاً لو كانت عندك الشيفرة الآتية:

```
<style type="text/css">
    article { display: block; border: 1px solid red; }
</style>
...
<article>
<h1>Welcome to Initech</h1>
<p>This is your <span>first day</span>. </p>
</article>
```

فلن يعامل متصفح Internet Explorer (إلى الإصدار 8 IE) العنصر `<article>` على أنه عنصرٌ كتليٌ (block-level element)، ولن يُظهر أيضًا إطاراً أحمرًا حول عنصر `article`، فسيتم تجاهل جميع أنماط التنسيد. [حل الإصدار التاسع من Internet Explorer](#) هذه المشكلة.

المشكلة الثانية هي كائن DOM الذي سُئلَّ المت伤فات عندما تصادف عنصراً مجهولاً، وللمرة الثانية يكون أكثر متصفح فيه مشاكل بهذا الخصوص هو متصفح Internet Explorer ذو الإصدار 8 وما قبله ([حل الإصدار التاسع منه هذه المشكلة أيضًا](#)). إن لم يتعرف IE 8 على اسم العنصر، فسيضيف عنصراً إلى شجرة DOM كعقدة (node) فارغة دون أولاد (children). جميع العناصر التي تتوقع أن تكون «أبناء» للعنصر المجهول ستتمسّي «أخوة» (siblings) له. هذا تمثيل مبسط يوضح الفرق. هذه هي شجرة DOM التي تُنشئها HTML5:

```

article
|
+--h1 (child of article)
|   |
|   +--text node "Welcome to Initech"
|
+--p (child of article, sibling of h1)
|   |
|   +--text node "This is your "
|
+--span
|   |
|   +--text node "first day"
|
+--text node "."

```

أما هذه، فهي شجرة DOM التي كان يُنشئها Internet Explorer :

```

article (no children)
h1 (sibling of article)
|
+--text node "Welcome to Initech"
p (sibling of h1)
|
+--text node "This is your "
|
+--span
|   |
|   +--text node "first day"
|
+--text node "."

```

هناك حلٌّ التفافي عجيبٌ لهذه المشكلة؛ فلو أنشأت عنصر <article> فارغٌ عبر JavaScript قبل أن تستخدمه في صفحتك، فسيتعرف Internet Explorer بشكلٍ سحري على العنصر <article> وسيسمح لك بتنسيقه عبر CSS، وليس هناك حاجة إلى إدراج العنصر

الفارغ في شجرة DOM، فمجرد إنشاء عنصر وحيد (في كل صفحة) هو كافٍ ليعلم IE أنَّ عليه تنسيق العنصر الذي لم يتعرف عليه.

```
<html>
<head>
<style>
    article { display: block; border: 1px solid red; }
</style>
<script>document.createElement("article");</script>
</head>
<body>
<article>
    <h1>Welcome to Initech</h1>
    <p>This is your <span>first day</span>.</p>
</article>
</body>
</html>
```

يعمل ما سبق في جميع إصدارات Internet Explorer حتى الإصدار السادس! يمكننا توسيع هذه الطريقة لإنشاء نسخ فارغة من جميع عناصر HTML5 دفعًّا واحدة، لا تنس أنها لن تُدرج في شجرة DOM، لذا لن تراها أبدًا في الصفحة، ثم يمكنك البدء في استخدامها - أي العناصر - دون أن تقلق من عدم دعم المتصفحات القديمة لها.

هناك سكريبت باسم Shiv يوظف الطريقة السابقة، وهذه هي فكرته الأساسية:

```
<!--[if lt IE 9]>
<script>
    var e = ("abbr,article,aside,audio,canvas,datalist,details,"
+
        "figure,footer,header,hgroup,mark,menu,meter,nav,output," +
        "progress,section,time,video").split(',');
    for (var i = 0; i < e.length; i++) {
        document.createElement(e[i]);
    }
</script>
```

```
</script>
<![endif]-->
```

الأسطر `<!--[if lt IE 9]>` و `<!--[endif]-->` تسمى **التعليقات الشرطية**. إذ يعاملها Internet Explorer كأنها عبارة `if` الشرطية: «إن كان الإصدار الحالي من متصفح Internet Explorer أقل من الإصدار 9، فنفذ الشيفرة الآتية.»، وستعامل بقية المتصفحات تلك الشيفرة على أنها تعليقات HTML. فالخلاصة هي أنَّ متصفح Internet Explorer (الإصدار الثامن وما دونه) سيُنفذ السكريبت، لكن ستتجاهله بقية المتصفحات تماماً، وهذا يجعل صفحتك تُحمل بشكلٍ أسرع في المتصفحات التي لا تتطلب هذه العملية الالتفافية.

شيفرة JavaScript هي بسيطة نسبياً، المتغير `e` هو مصفوفة من السلالسل النصية مثل `"abbr"` و `"aside"` و `"article"` وهكذا، ثم ستأتي حلقة التكرار لتنشئ كل عنصر من العناصر المذكور اسمها وذلك باستدعاء الدالة `document.createElement()`، ولأننا سنعمل العنصر المُعاد من تلك الدالة، فلن ثُدرج تلك العناصر في شجرة DOM أبداً. لكن هذا كافٍ لجعل متصفح Internet Explorer يعامل تلك العناصر كما نرغب، وذلك عندما نستخدمها لاحقاً في الصفحة.

كلمة «لاحقاً» في الجملة السابقة مهمة، لأنَّه يجب أن يكون السكريبت في أعلى الصفحة، ويُحسن أن يكون ضمن عنصر `<head>`، وليس في أسفل الصفحة. وبهذه الطريقة سيُنفذ متصفح Internet Explorer السكريبت قبل معالجته للوسم والخصائص. وإن وضع السكريبت في أسفل الصفحة، فقد فات الأوان، وسيكون قد فسَّر متصفح Internet Explorer الوسم تفسيرًا خطأً وبنى شجرة DOM مغلوطة.

سكريبت HTML5 Shiv **مستضاف على GitHub** (إن كنت تتساءل، فهذا السكريبت مفتوح المصدر ومرخص برخصة MIT، لذلك تستطيع استخدامه في أي مشروع تعمل عليه). وإن أحببت، تستطيع استخدام الرابط المباشر إلى السكريبت بالإشارة إلى النسخة المستضافة كما يلي:

```
<head>
  <meta charset="utf-8" />
  <title>My Weblog</title>
  <!--[if lt IE 9]>
    <script
      src="https://raw.githubusercontent.com/aFarkas/html5shiv/master
      /src/html5shiv.js"></script>
    <![endif]-->
</head>
```

أصبحنا الآن جاهزين للبدء في استخدام العناصر الهيكلية الجديدة في HTML5.

8. الترويسات - Headers

لنعد الآن إلى **صفحتنا التي نعمل عليها**. ولننظر تحديداً إلى الترويسات (headers):

```
<div id="header">
  <h1>My Weblog</h1>
  <p class="tagline">A lot of effort went into making this
  effortless.</p>
</div>

...
<div class="entry">
  <h2>Travel day</h2>
</div>

...
```

```
<div class="entry">
  <h2>I'm going to Prague!</h2>
</div>
```

لا يوجد أي شيء خطأ في الشيفرات السابقة، ويمكنك أن تُبقي عليها إن شئت، فهي شيفرات HTML5 سليمة. لكن HTML5 توفر عناصر هيكلية إضافية للترويسات والأقسام. بدايةً، لتنخلص من العنصر `<div id="header">`. من الشائع إضافة مثل هذه العناصر لكنها لا تعني شيئاً، إذ لا يُعرف العنصر `div` أي نوع من الهيكلية، وكذلك الأمر للخاصية `id` (ليس من المسموح للمتصفحات إعطاء أي معنى لقيم خاصية `id`)، يمكنك تغيير قيمة الخاصية إلى `id="blabla"` وسيبقى لها نفس المعنى: لا شيء!

تُعرف HTML5 العنصر `<header>` لهذا الغرض، ولدي مواصفات HTML5 **أمثلة عملية لاستخدام العنصر `<header>`: وهذه هي طريقة استخدامه في **صفحتنا التي تُعدّلها**:**

```
<header>
  <h1>My Weblog</h1>
  <p class="tagline">A lot of effort went into making this
effortless.</p>
  ...
</header>
```

هذا أمرٌ حسن، فهذا العنصر يخبر الجميع أنه ترويسة. لكن ماذا عن الشعار البديل (`tagline`)؟ هذه مشكلة أخرى شائعة، التي ليس لها إلى الآن طريقة معيارية لكتابتها. فمثلاً الشعار البديل كمثال الترويسة الفرعية، لكنه «ملحق» بالترويسة الرئيسية. أي بكلام آخر، هو ترويسة فرعية ليس لها قسمٌ خاصٌ بها.

عناصر الترويسات مثل `<h1>` و `<h2>` تُعطي بنيةً لصفحتك، وعندما تأتي مجتمعةً ستتشكل تخطيطًا يمكنك استخدامه لتخيل (أو للتنقل في) صفحتك، يمكن لبرامج قراءة الشاشة أن تستخدم تخطيط الصفحة لمساعدة المستخدمين الذين يعانون من ضعف (screenreaders) في النظر على التنقل في صفحتك، وهناك أدوات وإضافات متصرّح تسمح لك برسم مخطط المستندات.

كانت وسوم `<h1>-<h6>` هي الطريقة الوحيدة لإنشاء تخطيط لصفحتك في HTML 4.

سيبدو مخطط صفحتنا التي نعمل عليها كما يلي:

```
My Weblog (h1)
  |
  +-Travel day (h2)
  |
  +-I'm going to Prague! (h2)
```

هذا أمرٌ حسنٌ، لكنه يعني أنه لا توجد طريقة لتوصيم سطر الشعار اللفظي (tagline): «A lot of effort went into making this effortless» فذلك سيضيف عقدةً وهميةً جديدةً إلى تخطيط المستند:

```
My Weblog (h1)
  |
  +-A lot of effort went into making this effortless. (h2)
  |
  +-Travel day (h2)
  |
  +-I'm going to Prague! (h2)
```

لكن هذه ليست بنية المستند الحقيقية، إذ لا يُمثل الشعار اللفظي قسماً (section); بل هو مجرد ترويسة فرعية.

ماذا لو وسمنا الشعار اللغظي بوسم `<h2>` ثم وسمنا كل ترويسة من ترويسات المقالة بوسم `<h3>`? لا تفكر في ذلك! لأنه أسوأ:

```
My Weblog (h1)
  |
  +--A lot of effort went into making this effortless. (h2)
    |
    +--Travel day (h3)
    |
    +--I'm going to Prague! (h3)
```

ما يزال لدينا عقدة (node) وهمية في مخطط المستند، لكنها «سرقت» الأولاد الذين ينتمون إلى العقدة الجذر (h1). وهنا ترقد المشكلة: لا توفر HTML طريقةً لتوسيم العنوان الفرعي دون إضافته إلى مخطط المستند؛ ومهما حاولت، ستظهر العبارة «`.into making this effortless`» كجزءٍ من المخطط. ولهذا السبب وضعناها في شيفرةٍ ليس لها معنى هيكلٍ مثل `.<p class="tagline">`

توفر HTML5 حلًّا لهذه المشكلة وهو العنصر `<hgroup>`، يعمل العنصر `<hgroup>` حاويةً لترويستين أو أكثر مرتبطة بعضها. ما الذي تعنيه كلمة «مرتبطة»؟ تعني أنه عندما تأتي مجتمعةً فهي تنشئ عقدة وحيدة في مخطط المستند.

فليكن لدينا الشيفرة الآتية:

```
<header>
  <hgroup>
    <h1>My Weblog</h1>
    <h2>A lot of effort went into making this effortless.</h2>
  </hgroup>
  ...
</header>
```

```
...
<div class="entry">
  <h2>Travel day</h2>
</div>

...
<div class="entry">
  <h2>I'm going to Prague!</h2>
</div>
```

وهذا هو مخطط الصفحة الذي سينشأ:

```
My Weblog (h1 of its hgroup)
|
+--Travel day (h2)
|
+--I'm going to Prague! (h2)
```

يمكنك اختبار صفحاتك بوساطة [HTML5 Outliner](#) للتتأكد من أنك تستعمل عناصر

الترويسات استعمالاً صحيحاً.

9. المقالات

لنكم مع [صفحتنا](#)، للنظر ماذا يمكننا أن نفعل مع هذه الشيفرة:

```
<div class="entry">
  <p class="post-date">October 22, 2009</p>
  <h2>
    <a href="#" rel="bookmark" title="link to this post">
      Travel day
    </a>
  </h2>
```

```

    </a>
  </h2>
</div>
```

أكرر مرةً أخرى أنَّ ما سبق صحيح تماماً في HTML5، لكن HTML5 توفر عنصراً مخصصاً لغرض توسيم المقالات في الصفحة، وأطلق عليه اسم `<article>`.

```

<article>
  <p class="post-date">October 22, 2009</p>
  <h2>
    <a href="#" rel="bookmark" title="link to this post">
      Travel day
    </a>
  </h2>
</article>
```

لكن ما سبق ليس بهذه البساطة، هناك تغييرٌ إضافيٌ عليه إجراؤه، سأريك ما هو أولاً، ثم أشرحه لك:

```

<article>
  <header>
    <p class="post-date">October 22, 2009</p>
    <h1>
      <a href="#" rel="bookmark" title="link to this post">
        Travel day
      </a>
    </h1>
  </header>
  ...
</article>
```

هل رأيت ما هو؟ بذَّلُث عنصر <h1> إلى <h2>, ووضعته في عنصر <header>, لقد شاهدت مثلاً عملياً عن العنصر <header>, فالغرض منه هو احتواء جميع العناصر التي تُشَكِّل ترويسة المقالة (وفي حالتنا هذه: عنوان المقالة وتاريخ نشرها). لكن ألا يجدر بنا استخدام عنصر <h1> وحيد في كل المستند؟ ألن يخرب ما فعلناه تخطيط المستند؟ لا، لكن إن أردنا أن نعرف لماذا، فعليينا العودة بالزمن إلى الوراء قليلاً...

الطريقة الوحيدة في HTML 4 لإنشاء تخطيط للمستند هي استخدام عناصر <h1>-<h6>, فإن أردت وجود عقدة (node) رئيسية وحيدة في مخططك فعليك استخدام <h1> مرةً واحدةً في المستند؛ لكن مواصفات HTML5 عرَّفت خوارزميةً لتوليد مخطط للمستند التي تأخذ بالحسبان العناصر البنوية الجديدة في HTML5. تقول تلك الخوارزمية أنَّ عنصر <article> يُنشئ قسماً (section) جديداً، أي أنَّه عقدة جديدةً في مخطط المستند، ولدي كلُّ قسمٍ في HTML5 عنصر <h1> خاصٌ به.

هذا تغييرٌ جرئٌ على طريقة تفسير مخطط الصفحات في HTML 4، وهذا هو السبب وراء كونه مفيداً: توَّلد العديد من صفحات الويب عبر القوالب؛ فجزءٌ من المحتوى مأخوذٌ من أحد المصادر ثم يضاف إلى أعلى الصفحة، وجزءٌ آخر مأخوذٌ من مصدرٍ آخر ومضاف إلى أسفل الصفحة. والعديد من الدروس التعليمية تتبع النهج الآتي: «هذه بعض شيفرات HTML، انسخها وألصقها في صفحتك»، ولا حرج في ذلك للقطع الصغيرة من الشيفرات، لكن ماذا لو كنت تنسخ قسماً (section) كاماًلاً؟ ستقرأ -في هذه الحالة- جملةً كهذه الجملة في الدرس:

«هذه بعض شيفرات HTML، انسخها وألصقها في محررك النصي، وصحح وسوم الترويسات لكي تطابق مخطط الصفحة التي ستصلك تلك الشيفرات فيها».

سأصوغ ما سبق بطريقةٍ أخرى: ليس هنالك في HTML 4 عنصر ترويسة عام (generic)، وفيها ستة عناصر ذات أرقام ثابتة للترويسات <h1>-<h6>، التي يجب أن تتشعب بنفس الترتيب. وهذا أمرٌ سيء، خصيصاً لو كانت صفحتك «جمّعة» تجميئاً بدلاً من كونها مطورةً من الصفر. حلّت HTML5 هذه المشكلة عبر عناصر التقسيم الجديدة والقواعد المحدّثة لعناصر الترويسات. فلو كنت تستخدم عناصر التقسيم الجديدة، فسأعطيك الشيفرة الآتية:

```
<article>
  <header>
    <h1>A syndicated post</h1>
  </header>
  <p>Lorem ipsum blah blah...</p>
</article>
```

وستنسخها وتلصقها في أي مكان في صفحتك دون أي تعديل. حتى لو كانت تحتوي العنصر <h1>، فلم يعد يُمثّل أية مشكلة، لأن كل شيء محتوى في عنصر <article> الذي بدوره يُعرّف عقدة في مخطط المستند، ويوفر العنصر <h1> عنواناً لتلك العقدة، وستبقى كل عناصر التقسيم الأخرى مكانها كما كانت من قبل.

الواقع معقد أكثر مما أخبرك به بسبب الحجم الهائل للويب. فعناصر التقسيم «الظاهرة» (explicit) (مثل <h1>) قد تتدخل بطريق غير متوقعة مع عناصر التقسيم «الضمنية» (implicit) (أي عناصر ضمن <article>) قد تتدخل بطريق غير متوقعة مع عناصر التقسيم «الضمنية» (implicit) (أي عناصر <h1>-<h6> نفسها). ستسهل الأمر على نفسك كثيراً إن استعملت إحدى الطريقتين، لكن لا تستخدماهما معاً. وإن كان لا بد، فتحقق من النتيجة في [HTML5 Outliner](#) لكي تتأكد أنَّ مخطط صفحتك منطقي.

10. الوقت والتاريخ

حسناً، هذه الميزة الجديدة مثيرة حقاً، لنكمل مع [صفحتنا التي نعمل عليها](#)، ولننظر إلى

السطر الآتي:

```
<div class="entry">
  <p class="post-date">October 22, 2009</p>
  <h2>Travel day</h2>
</div>
```

القصة القديمة نفسها، صحيح؟ نمط شائع لتحديد تاريخ النشر الذي لا توجد عناصر بنوية لتنفيذها، لهذا يلجأ المطورون إلى استخدام الوسوم العامة مع قيم `class` مخصصة. أكرر مرة أخرى، ما سبق صالح تماماً في HTML5، وليس مطلوباً منك تغييره، لكن HTML5 توفر حالاً مخصصاً لهذه الحالة: العنصر `<time>`.

هناك ثلاثة أجزاء للعنصر `<time>`:

1. بصمة وقت (`timestamp`) قابلة للقراءة من المتصفح

2. محتوى نصي ليظهر إلى المستخدم

3. خاصية `pubdate` الاختيارية

تحدد الخاصية `datetime` -في هذا المثال- التاريخ فقط، ولا تحديد الوقت. الصيغة المستعملة هي: السنة ممثلة بأربعة أرقام، والشهر برمدين، واليوم برمدين؛ والقيمة مفصلة بشرطات:

```
<time datetime="2009-10-22" pubdate>October 22, 2009</time>
```

أما إذا أردت تضمين الوقت أيضًا، فأضاف الحرف T بعد التاريخ، ثم الوقت بصيغة 24 ساعة، ثم فرق توقيت المنطقة الزمنية.

```
<time datetime="2009-10-22T13:59:47-04:00" pubdate>
  October 22, 2009 1:59pm EDT
</time>
```

(صيغة الوقت والتاريخ التي تستعملها HTML5 مرنة جدًا، وتحتوي مواصفات HTML5 على عدّة أمثلة عن السلاسل النصية الصالحة لاستخدامها للوقت والتاريخ).

لاحظ أنني عدّلت المحتوى النصي -الموجود بين وسم البداية <time> وإغلاقها </time>- ليوافق بصمة الوقت المحددة؛ لكن هذا ليس إجبارياً، ويمكن أن تضع ما تشاء طالما وفّرتك بصمة وقت قابلة للتفسير من المتصفح في خاصية `datetime`. أي أنّ ما يلي صالح تماماً في HTML5:

```
<time datetime="2009-10-22">last Thursday</time>
```

وهذا أيضًا:

```
<time datetime="2009-10-22"></time>
```

آخر قطعة من الأحجية هي خاصية `pubdate`، وهي قيمة منطقية (Boolean)، أي أنك تضيفها إن احتجت إليها، كما يلي:

```
<time datetime="2009-10-22" pubdate>October 22, 2009</time>
```

إن لم تحب استخدام الخاصيات المنطقية دون تحديد قيمة لها، فاستعمل الشكل المكافئ الآتي:

```
<time datetime="2009-10-22" pubdate="pubdate">October 22, 2009</time>
```

ما الغرض من الخاصية `pubdate`? هي تعني أحد الأمرين الآتيين: إما أن يكون عنصر `time` ضمن عنصر `article`, عندما سيعني أنَّ هذه هي بصمة الوقت لتاريخ نشر المقالة. أما لو كان العنصر `time` لا يقع ضمن عنصر `article` فهذا يعني أنَّ بصمة الوقت هي لتاريخ نشر المستند بأكمله.

هذه هي كامل المقالة (`article`) التي أعيدت صياغتها لتستفيد من ميزات HTML5 الحديثة:

```
<article>
  <header>
    <time datetime="2009-10-22" pubdate>
      October 22, 2009
    </time>
    <h1>
      <a href="#" rel="bookmark" title="link to this post">
        Travel day
      </a>
    </h1>
  </header>
  <p>Lorem ipsum dolor sit amet...</p>
</article>
```

11. التنقل

أحد أكثر الأقسام أهميةً في أي موقع ويب هو شريط التنقل. فهناك «ألسنة» (`tabs`) في موقع CNN.com في أعلى كل صفحة التي تعطي روابط لمختلف أقسام الأخبار: «Tech» و«Sports» و«Health» و«...إلخ»؛ وصفحة نتائج Google فيها شريط مشابه في أعلى الصفحة

ليمضحك إمكانية البحث في مختلف خدمات Google: «الصور» و «الفيديو» و «الخرائط» ... إلخ. **صفحتنا التي نعمل عليها** فيها شريط تنقل في الترويسة الذي يتضمن روابط إلى مختلف الأقسام الوهمية لموقعنا: «home» و «blog» و «about» و «gallery». هذه هي الطريقة التي كُتبَ بها شريط التنقل:

```
<div id="nav">
  <ul>
    <li><a href="#">home</a></li>
    <li><a href="#">blog</a></li>
    <li><a href="#">gallery</a></li>
    <li><a href="#">about</a></li>
  </ul>
</div>
```

أكرر مرةً أخرى، ما سبق صالح تماماً في HTML5؛ لكن على الرغم من أنه مكتوب على شكل قائمة ذات أربعة عناصر، إلا أنه لا يوجد شيء يدلّك على أنه جزءٌ من آلية التنقل في الموقع. ربما تستطيع أن تتوقع ذلك بصرياً، لكنه جزءاً من ترويسة الصفحة، ولأنك تستطيع قراءة نص الروابط، لكن لا يوجد شيء يميز هذه القائمة من الروابط عن غيرها هيكلياً.

من الذي يهتم بالبنية الهيكلية لقائمة التنقل في الموقع؟ الأشخاص أولو الاحتياجات الخاصة، ولماذا؟ تخيل هذا السيناريو: حركة أحد مستخدمي موقعك محدودة، ويصعب عليه استعمال الفأرة أو يستحيل ذلك. وللتعمويض عنها سيستعمل إضافةً في المتصفح تسمح له بالانتقال إلى (أو تحطّي) وصلات التنقل الرئيسية. أو تخيل هذا: بصره ضعيف، وربما يستعمل برنامجاً اسمه «screenreader» (أي قارئ الشاشة) الذي يستعمل تقنية تحويل النص إلى كلام لكي يلْحّص له صفحات الويب؛ وعادةً يأتي بعد قراءة عنوان الصفحة قراءةً القطع المهمة من المعلومات حول الصفحة التي هي وصلات التنقل الرئيسية. فإن أراد المستخدم أن يتنقل في

الموقع بسرعة، فسيخبر قارئ الشاشة أن ينتقل مباشرةً إلى قائمة التنقل ويفبدأ القراءة. أما لو أراد أن يتصفح الصفحة بسرعة، فسيخبر قارئ الشاشة أن يتخطى قائمة التنقل ويفبدأ بقراءة المحتوى الرئيسي للصفحة. وستجد في كلا الحالتين أنَّ قدرة تحديد قائمة التنقل لها أهمية برمجية.

لا يوجد أي خطأ في استعمال `<div id="nav">` لقائمة التنقل في موقعك، لكن ليس ذلك أمراً صائباً أيضاً! توفر HTML5 طريقةً هيكليةً لتوسيم أقسام التنقل عبر العنصر `<nav>`:

```
<nav>
  <ul>
    <li><a href="#">home</a></li>
    <li><a href="#">blog</a></li>
    <li><a href="#">gallery</a></li>
    <li><a href="#">about</a></li>
  </ul>
</nav>
```

س: هل «skip links» متوافقة مع العنصر `<nav>`? هل أحتج إلى إضافة `skip links` في HTML5؟
ج: skip links تسمح للقارئات بتخطي أقسام التنقل، وهذا أمرٌ مفید للمستخدمين أولى الاحتياجات الخاصة الذين يستعملون برمجيات خارجية لقراءة صفحات الويب وللتنقل فيها دون استخدام فأرة. بعد أن تُحدَّث قارئات الشاشة لكي تعرف على العنصر `<nav>`، فسيتمكن استخدام skip links لأن برمجيات قراءة الشاشة ستتصبح قادرة على تخطي قسم التنقل الموجود ضمن العنصر `<nav>`. لكن ستمضي فترةً قبل أن يُحدَّث كل مستخدمي الويب أولى الاحتياجات الخاصة برمجيات قراءة الشاشة، لذلك أرى أن عليك المتابعة في توفير skip links لتخطي أقسام `<nav>` إلى ذاك الحين.

12. التذييلات

أخيراً وليس آخرًا، وصلنا إلى نهاية **صفحتنا**، آخر شيء أريد التحدث عنه هو آخر شيء في

الصفحة: التذليل (footer). كتبنا التذليل في الصفحة الأصلية هكذا:

```
<div id="footer">
  <p>&#167;</p>
  <p>&#169; 2001&#8211;9 <a href="#">Mark Pilgrim</a></p>
</div>
```

أيضاً ما سبق صالح تماماً في HTML5، يمكنك الإبقاء عليه إن شئت، لكن HTML5 توفر لك عنصراً مخصصاً لهذا العرض ألا وهو العنصر `<footer>`.

```
<footer>
  <p>&#167;</p>
  <p>&#169; 2001&#8211;9 <a href="#">Mark Pilgrim</a></p>
</footer>
```

ما هو المحتوى الملائم وضعه في عنصر `<footer>`? ربما تضع فيه ما كنت تضع في `<div id="footer">`. حسناً، نحن الآن ندور في حلقة مفرغة، لكن صدقًا، هذا هو الجواب. يقول مواصفات HTML5: «يحتوي التذليل عادةً على معلوماتٍ حول القسم الذي ينتمي إليه كفن كتبه، وروابط إلى مستندات متعلقة، ومعلومات عن حقوق النشر، وأمورٍ مشابهة». وهذا ما لدينا في صفحتنا: عبارة مختصرة عن حقوق النشر ووصلة إلى صفحة «عن المؤلف». يمكنك أن تجد أمثلة عن التذليلات بالنظر في بعض المواقع المشهورة:

- في موقع CNN تذليلٌ يحتوي على عبارة حقوق النشر، وروابط إلى الترجمات، وروابط إلى «شروط الخدمة» (terms of services)، وسياسة الخصوصية، وصفحات «us» و «us» و «contact us» و «help».

صفحة Google الرئيسية مشهورةً كثيراً، وفي أسفلها روابط إلى «Advertising» و «About Google» و «Business Solutions» و «Programs» و «W3C Updates»، وعبارة حقوق النشر، ورابط إلى سياسة الخصوصية في Google. وكل ذلك موجود في عنصر `<footer>`. شاعت في هذه الآونة ما ندعوه «Fat Footers» لنلق نظرةً على التذييل في موقع W3C، الذي يحتوي على ثلاثة أعمدة معروفة باسم `Navigation` و `Contact W3C` و `W3C Updates`. شيفرة التذييل تشبه ما يلي (تقريباً):

```
<div id="w3c_footer">
  <div class="w3c_footer-nav">
    <h3>Navigation</h3>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/standards/">Standards</a></li>
      <li><a href="/participate/">Participate</a></li>
      <li><a href="/Consortium/membership">Membership</a></li>
      <li><a href="/Consortium/">About W3C</a></li>
    </ul>
  </div>
  <div class="w3c_footer-nav">
    <h3>Contact W3C</h3>
    <ul>
      <li><a href="/Consortium/contact">Contact</a></li>
      <li><a href="/Help/">Help and FAQ</a></li>
      <li><a href="/Consortium/sup">Donate</a></li>
      <li><a href="/Consortium/siteindex">Site Map</a></li>
    </ul>
  </div>
  <div class="w3c_footer-nav">
    <h3>W3C Updates</h3>
    <ul>
      <li><a href="http://twitter.com/W3C">Twitter</a></li>
      <li><a href="http://identi.ca/w3c">Identiti.ca</a></li>
    </ul>
  </div>
</div>
```

```
</div>
<p class="copyright">Copyright © 2009 W3C</p>
</div>
```

سأجري التعديلات الآتية لتحويل ما سبق إلى شيفرات HTML5 هيكلية:

- تحويل الوسم `<footer>` إلى عنصر `<div id="w3c_footer">`.
- تحويل أول نسختين من `<nav>` إلى عنصر `<div class="w3c_footer-nav">`.
- والنسخة الثالثة إلى عنصر `<section>`.
- تحويل ترويسات `<h1>` إلى `<h3>`, لأنها داخل عنصر `<nav>` الذي يُنشئ قسماً في تحطيط المستند, مثل عنصر `<article>` تماماً.
- ستبدو الشيفرة النهائية كالتالي:

```
<footer>
  <nav>
    <h1>Navigation</h1>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/standards/">Standards</a></li>
      <li><a href="/participate/">Participate</a></li>
      <li><a href="/Consortium/membership">Membership</a></li>
      <li><a href="/Consortium/">About W3C</a></li>
    </ul>
  </nav>
  <nav>
    <h1>Contact W3C</h1>
    <ul>
      <li><a href="/Consortium/contact">Contact</a></li>
      <li><a href="/Help/">Help and FAQ</a></li>
      <li><a href="/Consortium/sup">Donate</a></li>
      <li><a href="/Consortium/siteindex">Site Map</a></li>
    </ul>
  </nav>
```

```

<section>
  <h1>W3C Updates</h1>
  <ul>
    <li><a href="http://twitter.com/W3C">Twitter</a></li>
    <li><a href="http://identi.ca/w3c">Identi.ca</a></li>
  </ul>
</section>
<p class="copyright">Copyright © 2009 W3C</p>
</footer>

```

13. مراجع إضافية

الصفحة التي كنا نعمل عليها في هذا الفصل:

- النسخة الأصلية (HTML 4)
- النسخة المعدلة (HTML5)

مراجع حول ترميز المحارف:

The Absolute Minimum Every Software Developer Absolutely, Positively •
مقالة

Joel Spolsky
لكتابها Must Know About Unicode and Character Sets (No Excuses!)

Spolsky
Characters vs. On Character Strings و On the Goodness of Unicode •
مقالات

Tim Bray Bytes
لكتابها

مراجع حول تفعيل HTML5 في متصفح Internet Explorer :

Sjoerd Visscher
بواسطة How to style unknown elements in IE •

John Resig
بواسطة HTML5 shiv •

Remy Sharp
بواسطة HTML5 enabling script •

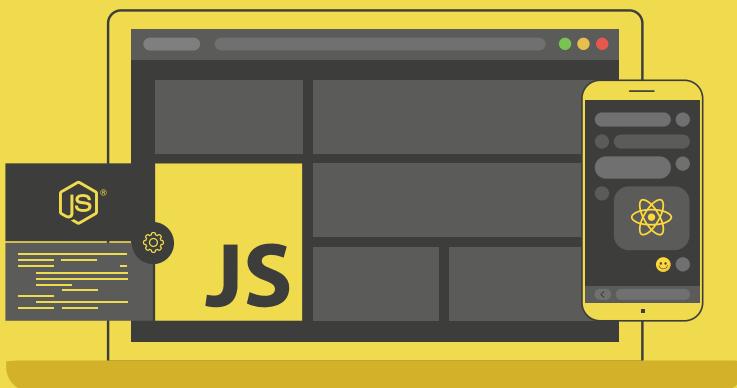
مراجع حول أنماط تفسير المتصفح للشيفرات، وأنواع doctype

- مقالة Henri Sivonen لصاحبها [Activating Browser Modes with Doctype](#). هذه هي المقالة الوحيدة التي عليك قراءتها في هذا الموضوع، لأنّ أيّة مقالة أخرى عن doctypes لا تستخدم المقالة السابقة كمرجع ستكون قدّيمةً أو غير كاملة أو فيها أخطاء.

مُدقّق HTML5 (validator) لشيفرات

html5.validator.nu

دورة تطوير التطبيقات باستخدام لغة JavaScript



احترف تطوير التطبيقات بلغة جافا سكريبت
انطلاقاً من أبسط المفاهيم وحتى بناء تطبيقات حقيقية

التحق بالدورة الآن



الرسم على قماش canvas

Σ

تعريف HTML **العنصر `canvas`** على أنه «لوحة نقطية ذات أبعاد معينة يمكن استخدامها لعرض المخططات ورسومات الألعاب وغيرها من الصور المرئية برمجياً». ويمثل مستطيلاً في صفحتك إذ تستطيع استعمال JavaScript لرسم أي شيء تريده فيه.

الدعم الأساسي للعنصر `canvas`:

Android	iPhone	Opera	Chrome	Safari	Firefox	IE
+1.0	+1.0	+10.0	+3.0	+3.0	+3.0	*+7.0

* يتطلب IE 7 و 8 مكتبة خارجية هي `explorercanvas`، أما IE 9 فهو يدعم العنصر `canvas` داخلياً.

كيف يبدو عنصر `canvas`? ليس له شكل، حقاً! ليس في عنصر `canvas` أي محتوى وليس له إطار حتى. يضاف عنصر `canvas` كالتالي:

```
<canvas width="300" height="225"></canvas>
```

لنضف إطاراً منقطاً لكي نستطيع أن نرى ما الذي نتعامل معه:



الشكل 1: عنصر `canvas` مع إطار

يمكن أن يكون لديك أكثر من عنصر `<canvas>` في نفس الصفحة، وسيظهر كل عنصر على حدة في شجرة DOM، ويحافظ كل عنصر `canvas` على خاصياته؛ فإن أعطيت كل عنصر `canvas` خاصية `id`، فيمكنك الوصول إليه كما تفعل مع أي عنصر آخر.

لنوضح الشيفرة السابقة لكي تتضمن خاصية `id`:

```
<canvas id="a" width="300" height="225"></canvas>
```

أصبح بإمكاننا بسهولة العثور على عنصر `<canvas>` السابق في شجرة DOM.

```
var a_canvas = document.getElementById("a");
```

1. الأشكال البسيطة

Android	iPhone	Opera	Chrome	Safari	Firefox	IE
+1.0	+1.0	+10.0	+3.0	+3.0	+3.0	*+7.0

* يتطلب IE 7 و 8 مكتبة خارجية هي `explorercanvas`، أما IE 9 فهو يدعم رسم الأشكال البسيطة في العنصر `<canvas>` داخلياً.

ببدأ كل عنصر `canvas` فارغاً، ثم علينا الرسم فيه. لنبدأ ببعض الأشكال.

يمكن استخدام الحدث `(action)` `onclick` لاستدعاء الدالة الآتية عندما يضغط المستخدم على زر ما:

```
function draw_b() {
    var b_canvas = document.getElementById("b");
    var b_context = b_canvas.getContext("2d");
    b_context.fillRect(50, 25, 150, 100);
}
```

لا يوجد شيء مميز في أول سطر من الدالة، إذ إن مهمته هي العثور على عنصر `DOM` الموجود في شجرة `<canvas>`

```
function draw_b() {
    var b_canvas = document.getElementById("b");
    var b_context = b_canvas.getContext("2d");
    b_context.fillRect(50, 25, 150, 100);
}
```

لدي كل عنصر `canvas` ما نسميه «إطار الرسم» (`drawing context`), الذي تحدث فيه كل الأمور المسليمة. وبعد أن تعثر على عنصر `<canvas>` في شجرة `DOM` (باستخدام `document.getElementById()` أو أيّة طريقة أخرى) ستستطيع أن تستدعي الدالة `.getContext()`، يجب عليك تمرير السلسلة النصية «`2d`» دوّماً إلى الدالة `(getContext())`.

س: هل يمكن رسم رسوميات 3D في `?canvas`

ج: نعم، عبر تقنية `WebGL` التي يمكنها رسم الأشكال ثلاثية الأبعاد في المتصفحات دون إضافات. تدعم أغلبية المتصفحات الحديثة هذه التقنية (`Firefox` الإصدار الرابع وما بعده، و `Chrome` الإصدار التاسع وما بعده، و `Opera` 12 وما بعده، و `Safari` 5.1 وما بعده، و `IE` 11): يتم العمل على تطوير هذه التقنية في مجموعة `WebGL عمل`.

إذاً، أصبح لديك عنصر `<canvas>` ولديك إطار الرسم (`drawing context`) الخاص به. إطار الرسم هو المكان الذي ستعرف فيه جميع دوال وخصائص الرسم. هنالك مجموعة كاملة من الخصائص والدوال المكررة لرسم المستطيلات:

- يمكن أن تكون الخاصية `fillStyle` لوناً من ألوان CSS، أو نقشاً (`pattern`)، أو تدرجاً (`gradient`) (سنذكر مزيداً من المعلومات عن التدرجات اللونية بعد قليل). القيمة

الافتراضية لهذه الخاصية هي اللون الأسود، لكنك تستطيع أن تضبطها لما تشاء. سيتذكر كل إطار رسم (drawing context) خاصياته طالما بقيت الصفحة مفتوحةً إلا إن فعلت شيئاً لإعادة ضبطه.

- الدالة `fillRect(x, y, width, height)` ترسم مستطيلاً مملوءاً باللون أو النقش `.fillStyle` الموجود في.
- الخاصية `strokeStyle` شبيهة بخاصية `fillStyle` لكن للحواف (stroke)، إذ يمكن أن تكون لون CSS أو نقشاً أو تدرجًا لونيًّا.
- الدالة `strokeRect(x, y, width, height)` ترسم مستطيلاً دون ملئه، إذ ترسم حواフェ وإطاره الخارجي فحسب، وتستعمل الخاصية `strokeStyle` لذلك.
- الدالة `clearRect(x, y, width, height)` تممسح كل البكسلات الموجودة في المستطيل المحدد.

س: هل يمكنني أن أعيد ضبط لوحة الرسم في عنصر `canvas`؟

ج: نعم، فسيؤدي تحديد عرض أو ارتفاع عنصر `<canvas>` إلى إعادة ضبط (reset) كل الخصائص في إطار الرسم إلى قيمها الافتراضية. لاحظ أنَّه ليس من الضروري تغيير العرض، إذ يمكنك أن تضبطه إلى قيمته الحالية كما يلي:

```
var b_canvas = document.getElementById("b");
b_canvas.width = b_canvas.width;
```

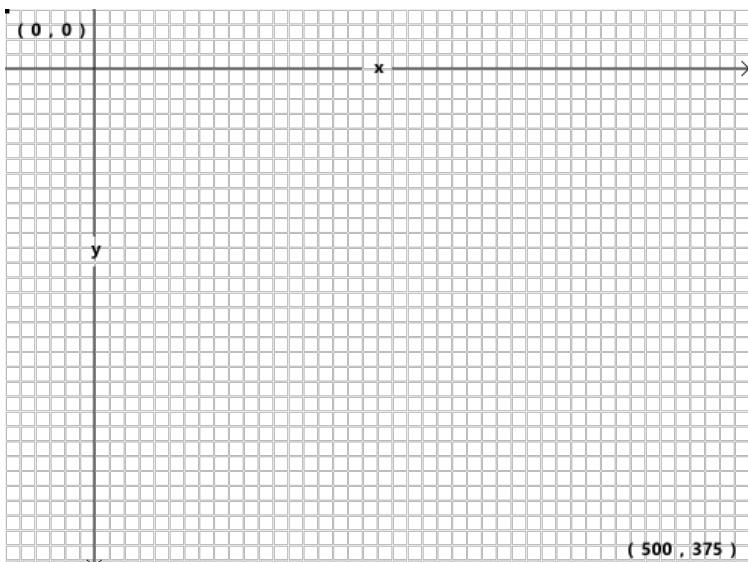
بالعودة إلى الشيفرة في المثال السابق...

```
var b_canvas = document.getElementById("b");
var b_context = b_canvas.getContext("2d");
b_context.fillRect(50, 25, 150, 100);
```

سترسم الدالة `fillRect()` عند استدعائها مستطيلاً وتملؤه بنمط الماء الحالي الذي هو اللون الأسود (إلا إذا غيرته). يُعرّف المستطيل عبر زاويته العليا اليسرى (25، 50)، وعرضه (150)، وارتفاعه (100). لكي نفهم ذلك جيداً، فلنلق نظرةً إلى نظام الإحداثيات في `canvas`.

2. الإحداثيات في عنصر `canvas`

يمكننا تخيل لوحة الرسم في عنصر `canvas` على أنها شبكة ثنائيةُ البعد، ويكون مبدأ الإحداثيات فيها (0، 0) في الزاوية العليا اليسرى من لوحة الرسم. تزداد القيم على المحور X عند الانتقال نحو الحافة اليمنى من لوحة الرسم، وتزداد القيم على المحور Y بالانتقال نحو الحافة السفلية من لوحة الرسم.



الشكل 2: توضيح لنظام الإحداثيات في عنصر `canvas`

رُسمَ الشكل السابق عبر عنصر `<canvas>` الذي يحتوي على:

- مجموعة من الخطوط الأفقية
 - مجموعة من الخطوط الرأسية (الشاقولية)
 - خطين أفقيين سوداويين
 - خطين صغيرين سوداويين مائلين يشكلان سهماً
 - خطين رأسيين سوداويين
 - خطين صغيرين سوداويين مائلين يشكلان السهم الآخر
 - الحرف «x»
 - الحرف «y»
 - النص «(0,0)» قرب الزاوية العليا اليسرى
 - النص «(500, 375)» قرب الزاوية السفلى اليمنى
 - نقطة في الزاوية العليا اليسرى، وأخرى في الزاوية السفلى اليمنى
- علينا أولاً تعريف العنصر `<canvas>` نفسه، إذ يحدد العنصر `<canvas>` خاصية العرض `width` والارتفاع `height`، والمعرف `id` كي نستطيع العثور عليه بسهولة لاحقاً.

```
<canvas id="c" width="500" height="375"></canvas>
```

ثم علينا كتابة سكريبت لكي نجد عنصر `<canvas>` في شجرة DOM ونحصل على إطار الرسم الخاص به.

```
var c_canvas = document.getElementById("c");
var context = c_canvas.getContext("2d");
```

نستطيع الآن البدء في رسم الخطوط.

3. المسارات

Android	iPhone	Opera	Chrome	Safari	Firefox	IE
+1.0	+1.0	+10.0	+3.0	+3.0	+3.0	*+7.0

* يتطلب IE 7 و 8 مكتبة خارجية هي `explorercanvas`، أما IE 9 فهو يدعم المسارات في العنصر `<canvas>` داخلياً.

تخيل أنك تريدين أن ترسم رسماً بالحبر، من المؤكد أنك لن تبدأ الرسم بأفلاام الحبر مباشرةً، لأنك قد تخطي؛ فعوضاً عن ذلك سترسم المستقيمات والمنحنيات بقلم الرصاص، ثم إن أعجبتك فيمكنك أن «تحبّرها».

هنا لك ما نسميه «المسارات» (`paths`) في عناصر `canvas`، وتعريف المسار يُشِّبه تماماً الرسم بقلم الرصاص؛ يمكنك رسم ما تشاء لكنه لن يكون جزءاً من اللوحة النهائية إلا إن أمسكت أقلام التخيير ومررتها فوق المسار الذي رسمته.

استعمل الدالتين الآتيتين لرسم المستقيمات «بقلم الرصاص»:

1. `moveTo(x, y)` تحريك قلم الرصاص إلى نقطة البداية المحددة.

2. `lineTo(x, y)` رسم خط إلى نقطة النهاية المحددة.

سيزيد حجم المسار كلما استدعيت الدالتين `moveTo()` و `lineTo()`. ترسم الدالتان السابقتان «بقلم الرصاص» (يمكنك أن تسمى هذه العملية ما تشاء)، أي أنك لن ترى شيئاً على لوحة الرسم إلى أن تستدعي إحدى دوال «التخيير».

لنبدأ برسم الشبكة:

```

رسم المستقيمات الرأسية //
for (var x = 0.5; x < 500; x += 10) {
  context.moveTo(x, 0);
  context.lineTo(x, 375);
}

رسم المستقيمات الأفقية //
for (var y = 0.5; y < 375; y += 10) {
  context.moveTo(0, y);
  context.lineTo(500, y);
}

```

تلك الدوال «رenerative»، أي لن يظهر شيءٌ على لوحة الرسم بعد؛ إذ سنحتاج إلى دالة «تحبير» لإظهار تلك الخطوط.

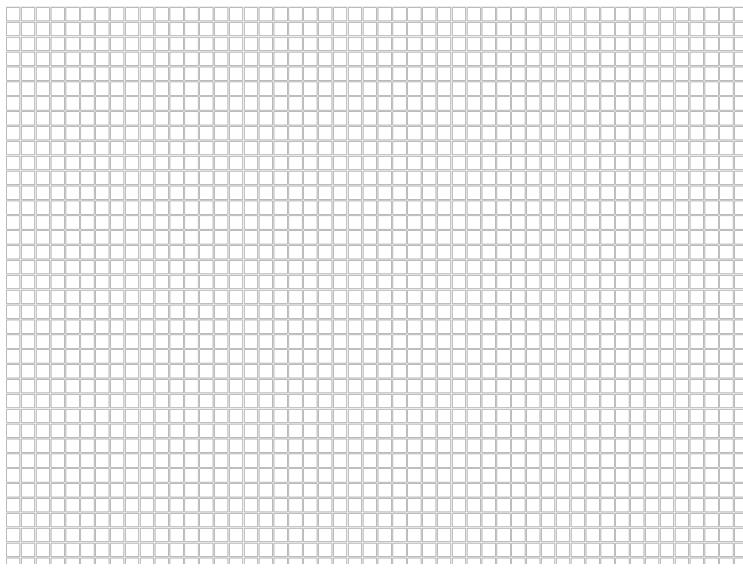
```

context.strokeStyle = "#eee";
context.stroke();

```

الدالة `stroke()` هي إحدى دوال «التحبير»، وهي تُحَبِّر المسار المعقد الذي عَرَفْتَه بدوال `moveTo()` و `lineTo()` السابقتين. خاصية `strokeStyle` تحكم بلون تلك الخطوط.

هذه هي النتيجة:

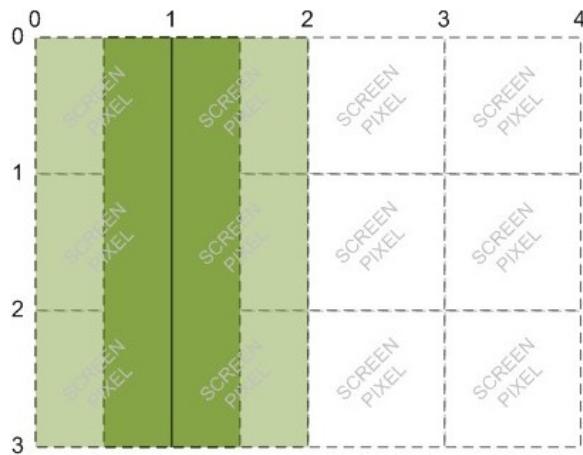


الشكل 3: شبكة مرسومة داخل عنصر canvas

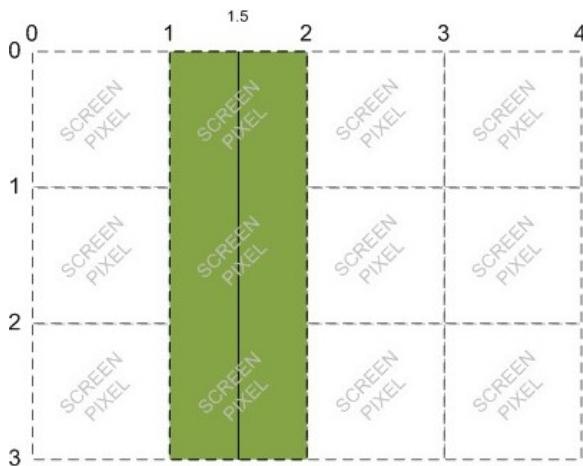
س: لماذا بدأت x و y من 0.5 وليس من 0؟

ج: تخيل أن كل بكسلي هو مربع كبير، وأن الإحداثيات ذات الرقم الكامل (0, 1, 2...) هي حواف تلك المربعات؛ فإذا أردت أن ترسم خطًا عرضه واحدة الأطوال (one-unit-wide) بين الإحداثيات ذات الرقم الصحيح، فسوف يمتد إلى أن يصل إلى طرفي المربع (انظر الشكل الآتي للإيضاح)، وسيكون الخط الناتج مرسومًا بعرض بكسلين. أما لرسم خط عرضه بكسلي واحد، فعليك أن تزيح الإحداثيات بمقدار 0.5 عمودياً على منحي (أو اتجاه) الخط.

على سبيل المثال، إذا كنت تحاول رسم خط من (0, 1) إلى (1, 3)، فسيرسم المتصفح خطًا يغطي 0.5 بكسلي على جانبي $x=1$ ، ولكن الشاشة غير قادرة على عرض نصف بكسلي، لذلك سيمتد الخط لكي يغطي بكسلين:



أما لو أردت أن ترسم خطًّا من (0, 0) إلى (1.5, 3)، فسيرسم المتصفح خطًّا يغطي 0.5 بكسل على طرفي $x=1.5$, الذي يؤدي إلى رسم خط بعرض 1 بكسل :



لرسم الآن السهم الأفقي. جميع الخطوط والمنحنيات الموجودة على نفس المسار سترسم بنفس اللون (أو التدرج اللوني، وستأتي لذكر ذلك لاحقاً). لكننا نريد أن نرسم السهم بلون مختلف (الأسود)، لذلك سنحتاج إلى البدء بمسارٍ (path) جديد.

```
مسار جديد //

context.beginPath();
context.moveTo(0, 40);
context.lineTo(240, 40);
context.moveTo(260, 40);
context.lineTo(500, 40);
context.moveTo(495, 35);
context.lineTo(500, 40);
context.lineTo(495, 45);
```

وبنفس الطريقة نرسم السهم الرأسى؛ ولقا كان السهم الرأسى بنفس لون السهم الأفقي، فلا حاجة إلى إنشاء مسار جديد، إذ سيشكل السهمان مساراً واحداً.

```
لا حاجة إلى إنشاء مسار جديد //

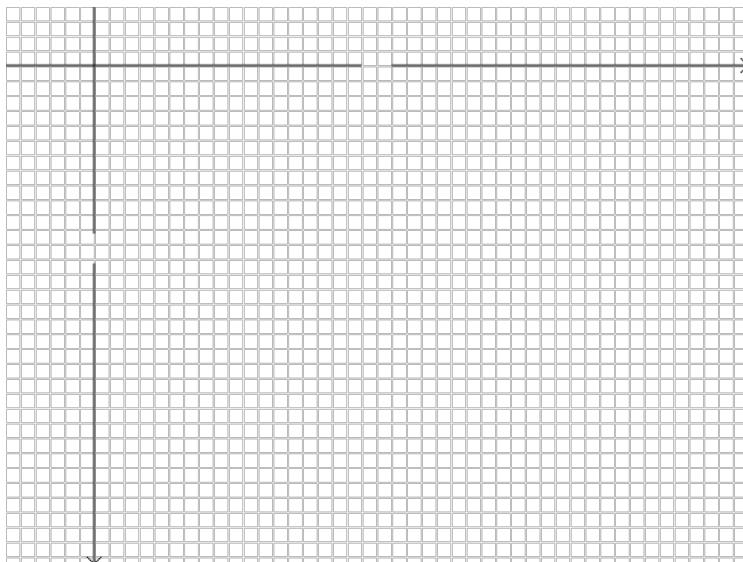
context.moveTo(60, 0);
context.lineTo(60, 153);
context.moveTo(60, 173);
context.lineTo(60, 375);
context.moveTo(65, 370);
context.lineTo(60, 375);
context.lineTo(55, 370);
```

لقد قلّت أنَّ هذين السهمين يجب أن يُرسما باللون الأسود، لكن لون `strokeStyle` ما يزال «الفضي الفاتح» (لا تتم عملية إعادة تعبيين قيم `strokeStyle` و `fillStyle` عندما تبدأ مساراً جديداً). لا بأس، لأننا رسمنا إلى الآن «بالرصاص» ولم «نُحِبّر» بعد، علينا الآن أن نضبط قيمة `strokeStyle` إلى اللون الأسود؛ وإلا سيرسم هذان السهمان بالفضي الفاتح وسيصعب

علينا رؤيتها. سُيُغَيِّر السطوان الآتيان اللون إلى الأسود ويرسمان الخطوط في لوحة الرسم:

```
context.strokeStyle = "#000";
context.stroke();
```

هذه هي النتيجة:



الشكل 4: المحوران مرسومان دون تسمية على لوحة الرسم

4. النص

Android	iPhone	Opera	Chrome	Safari	Firefox	IE
+1.0	+1.0	+10.0	+3.0	+3.0	**+3.0	*+7.0

* يتطلب IE 7 و 8 مكتبة خارجية هي `explorercanvas`, أما IE 9 فهو يدعم طباعة النصوص في العنصر `<canvas>` داخلياً.
** يتطلب Firefox 3.0 مكتبةً للتوافقية.

بالإضافة مقدرتنا على **رسم خطوط في لوحة الرسم**, يمكنك أيضًا «رسم» الجمل النصية. وعلى عكس النصوص في الصفحة المحيطة بلوحة الرسم, لا يوجد هنالك أيّة خاصيات CSS تتعلق بالتحطيط (layout) مثل floats و margins و padding و word wrapping. تستطيع ضبط بعض خاصيات الخط, ثم اختيار نقطة على لوحة الرسم و«رسم» نصك هناك.

تتوفر خاصيات الخطوط الآتية في لوحة الرسم:

- الخاصية `font` التي يمكن أن تُضبط إلى أي شيء تستطيع ضبطه في خاصية `font` في CSS. وهذا يتضمن نمط الخط، ونوع الخط، وسماكة الخط، وحجم الخط، وارتفاع السطر، وأسم «عائلة» الخط.
- الخاصية `textAlign` تتحكم بمحاذاة النص، وهي شبيهة (لكن ليس مماثلة تمامًا) بخاصية `text-align` في CSS. القيم المحتملة هي `right` و `left` و `start` و `end` و `.center` و `.center`.
- الخاصية `textBaseline` تتحكم في مكان «رسم» النص نسبيًّا إلى نقطة البداية. القيم `ideographic` هي `top` أو `middle` أو `bottom` أو `alphabetic` أو `hanging` أو `baseline`.

الخاصية `textBaseline` معقدة ودقيقة، لأن طريقة كتابة النصوص معقدة (لا أقصد هنا الإنكليزية، لكنك تستطيع رسم أي حرف يونيكود في عنصر `canvas`، وكتابة نصوص يونيكود معقدة)، **تشرح مواصفة HTML5 مختلف خطوط الأساس (baselines)** :

أعلى مربع `em` هو تقريبًا على مستوى أعلى المحارف في الخط (`font`), أما `middle` فهو مكان ارتكاز بعض المحارف مثل آء، أما خط `hanging baseline`»

فهو يقع في منتصف المسافة بين أعلى مربع em وأسفل مربع em. أما خط ideographic فهو مكان ارتكاز الأحرف مثل Á و ÿ و f و Ω، وخط alphabetic هو مكان ارتكاز المحارف مثل 私 و 達، وأسفل مربع em هو تقريباً أسفل المحارف الموجودة في خط (font) ما. قد يكون أعلى وأسفل الصندوق المحيط (bounding box) بعيداً عن خطوط الأساس السابقة نتيجةً لامتداد بعض المحارف خارج مربع em.



الشكل 5: خطوط الأساس

تستطيع استخدام `textBaseline` لخاصية `top` أو `bottom` للأبجديات `middle` البسيطة مثل الإنكليزية دون أن تكرر للبقية.

لنرسم بعض النصوص! النص المرسوم داخل عنصر `canvas` يرث حجم الخط ونمطه من عنصر `<canvas>` نفسه، لكنك تستطيع إعادة تعريف تلك القيم بضبط خاصية `font` في إطار الرسم.

```
// تغيير نمط الخط
context.font = "bold 12px sans-serif";
context.fillText("x", 248, 43);
context.fillText("y", 58, 165);
```

ترسم الخاصية `fillText()` النص.

```
context.font = "bold 12px sans-serif";
// رسم النص
context.fillText("x", 248, 43);
context.fillText("y", 58, 165);
```

س: هل أستطيع استخدام مقاسات الخطوط النسبية لرسم النصوص في عنصر `canvas`؟

ج: نعم، مثل أي عنصر HTML آخر في صفحتك، يحسب العنصر `<canvas>` مقاس الخط بناءً على قواعد CSS في صفحتك. فإذا ضبطت خاصية `font` إلى مقاس خط نسبي مثل `1.5em` أو `150%`، فسيضرب متصفحك هذا الرقم بحجم الخط المحسوب لعنصر `<canvas>` نفسه.

أريد أن يكون أعلى النص الموجود في الزاوية العليا اليسرى على بعد `y=5` من الحافة العلوية، لكنني كسول ولا أريد قياس ارتفاع النص وحساب بُعد خط الأساس (`baseline`)، وإنما سأضبط `textBaseline` إلى `top` وسأمرر إحداثيات الزاوية العليا اليسرى من مربع النص.

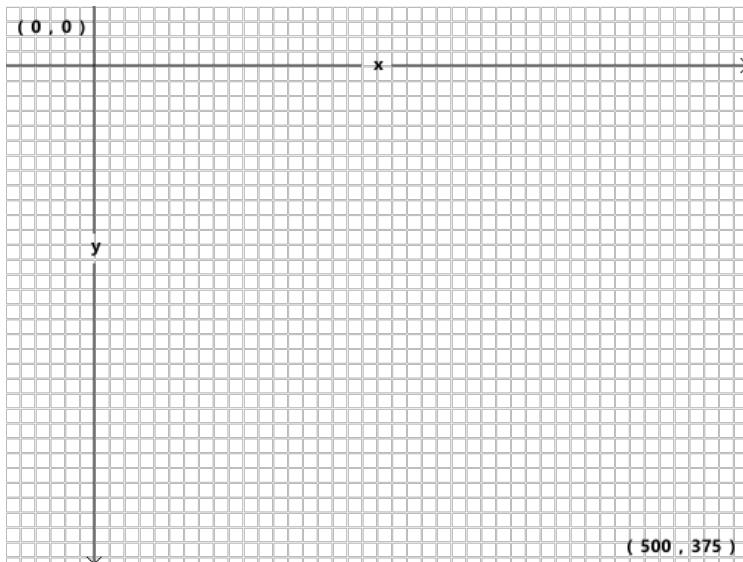
```
context.textBaseline = "top";
context.fillText("( 0 , 0 )", 8, 5);
```

أما النص في الزاوية السفلية اليمنى، فسأرسم الزاوية السفلية اليمنى للنص في الإحداثيات `(492, 370)`، أي بضعة بكسلات من الزاوية السفلية اليمنى من لوحة الرسم؛ لكنني لا أريد قياس `textBaseline` أو ارتفاع النص، لهذا أضبط الخاصية `textAlign` إلى `right` والخاصية

إلى bottom، ثم استدعى الدالة fillText() مُمّرّزاً إليها إحداثيات الزاوية السفلية اليمنى من مربع النص.

```
context.textAlign = "right";
context.textBaseline = "bottom";
context.fillText("( 500 , 375 )", 492, 370);
```

وهذه هي النتيجة:

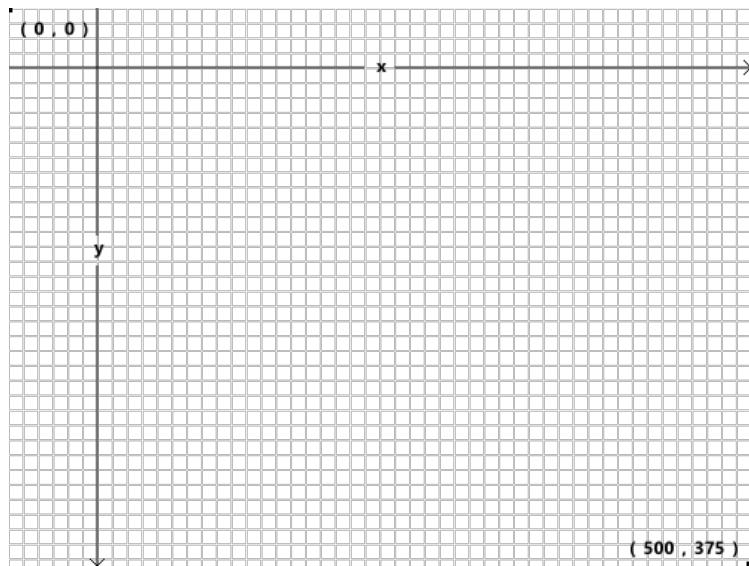


الشكل 6: تسمية المحاور

لقد نسيينا النقاط الموجودة على الزوايا! سنتعلم كيف نرسم الدوائر لاحقاً، أما الآن، فسأغش قليلاً وأرسمها كمستطيلات.

```
context.fillRect(0, 0, 3, 3);
context.fillRect(497, 372, 3, 3);
```

هذا كل ما في الأمر! هذه هي النتيجة النهائية:



الشكل 7: الرسم النهائي للإحداثيات عبر عنصر canvas

5. التدرجات اللونية

Android	iPhone	Opera	Chrome	Safari	Firefox	IE	
+1.0	+1.0	+10.0	+3.0	+3.0	+3.0	*+7.0	الدرجات اللونية الخطية
+1.0	+1.0	+10.0	+3.0	+3.0	+3.0	+9.0	الدرجات اللونية الشعاعية
* يتطلب IE 7 و 8 مكتبة خارجية هي <code>explorercanvas</code> ، أما IE 9 فهو يدعم التدرجات اللونية في العنصر <code><canvas></code> داخلياً.							

لقد تعلمنا سابقاً في هذا الفصل **كيف نرسم مستطيلًا مملوءًا بلونٍ ما**، ثم تعلمنا **كيف نرسم مستطيلًا ذا إطارٍ ملوئٍ بلونٍ معين**. لكن الأمر ليس محدوداً للألوان فقط، إذ يمكنك فعل ما تشاء بالدرجات اللونية، لنتظر إلى أحد الأمثلة.



الشكل 8: تدرج لوني من اليسار إلى اليمين

سيبدو وسم `canvas` كغيره من الوسوم:

```
<canvas id="d" width="300" height="225"></canvas>
```

علينا أولاً أن نعثر على عنصر `canvas` ثم نحصل على إطار الرسم:

```
var d_canvas = document.getElementById("d");
var context = d_canvas.getContext("2d");
```

يمكنا أن نبدأ بتعريف التدرج اللوني بعد حصولنا على إطار الرسم. التدرج اللوني هو

انتقالٌ ناعمٌ بين لونين أو أكثر. يمكن تعريف نوعين من التدرجات اللونية في إطار الرسم:

1. الدالة `createLinearGradient(x0, y0, x1, y1)` ترسم تدرجًا لونيًا عبر قطعة

مستقيمة من النقطة `(x0, y0)` إلى `(x1, y1)`.

2. الدالة `createRadialGradient(x0, y0, r0, x1, y1, r1)` ترسم تدرجًا لونيًا

عبر مخروط `(cone)` بين دائرتين. وتمثل أول ثلاثة معاملات `(parameters)` دائرة

البداية ذات المركز ($y_0, 0$) ونصف القطر r_0 . أما آخر ثلاثة معاملات فهي تمثل دائرة النهاية ذات المركز (y_1, x_1) ونصف القطر r_1 .
لنشئ تدرجًا لونيًّا خطًّيا (linear)، يمكن أن تكون التدرجات اللونية بأي قياس، لكننا سنجعل هذا التدرج بعرض 300 بكسل، مثل لوحة الرسم.

```
var my_gradient = context.createLinearGradient(0, 0, 300, 0);
```

لما كانت قيم y (الوسط الثاني والرابع) تساوي 0، فسيكون اتجاه التدرج اللوني من اليسار إلى اليمين.

يمكنك تعريف ألوان التدرج بعد أن تحصل على كائن التدرج اللوني. يمكن أن يكون للتدرج محطةً توقف لوني أو أكثر. التوقف اللوني (color stop) يمكن أن يكون في أي مكان في التدرج. ولإضافة توقف لوني، عليك تحديد مكانه ضمن التدرج اللوني، يمكن أن تكون تلك القيم بين 0 و 1.

لنعرف تدرجًا لونيًّا من الأسود إلى الأبيض.

```
my_gradient.addColorStop(0, "black");
my_gradient.addColorStop(1, "white");
```

لا يرسم أي شيء في لوحة الرسم عند تعريف التدرج اللوني، إذ سيُخزن ذاك الكائن في مكانٍ ما في الذاكرة. أما لرسم التدرج اللوني، فعليك أن تضبط خاصية `fillStyle` إلى ذاك التدرج ثم ترسم شكلًا ما مثل مستطيل أو مستقيم.

```
context.fillStyle = my_gradient;
context.fillRect(0, 0, 300, 225);
```

وهذه هي النتيجة:



الشكل 9: تدرج لوني من اليسار إلى اليمين

لنفترض أنك تريد تدرجًا لونيًّا من الأعلى إلى الأسفل؛ فسيكون عليك إنشاء كائن للتدرج اللوني تكون فيه قيمة x ثابتة (الوسطيتين الأول والثالث)، وتجعل قيم y (الوسطيين الثاني والرابع) تتراوح من 0 إلى ارتفاع لوحة الرسم.

```
قيم x تساوي 0، وقيم y متغيرة //
var my_gradient = context.createLinearGradient(0, 0, 0, 225);
my_gradient.addColorStop(0, "black");
my_gradient.addColorStop(1, "white");
context.fillStyle = my_gradient;
context.fillRect(0, 0, 300, 225);
```

وهذه هي النتيجة:

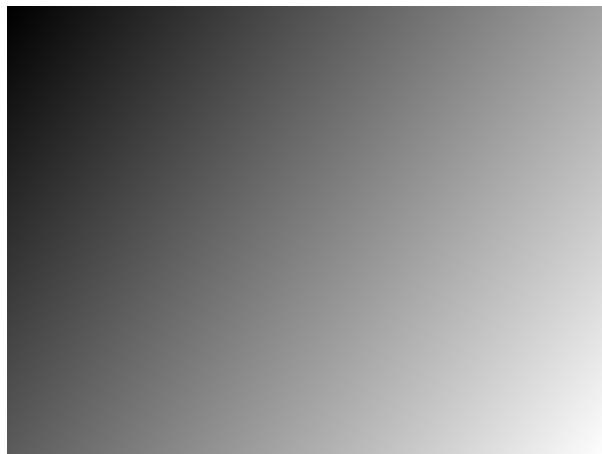


الشكل 10: تدرج لوني من الأعلى إلى الأسفل

و تستطيع أيضًا أن تجعل التدرجات اللونية قطبيةً.

```
// قيم x و y متغيرة //  
var my_gradient = context.createLinearGradient(0, 0, 300, 225);  
my_gradient.addColorStop(0, "black");  
my_gradient.addColorStop(1, "white");  
context.fillStyle = my_gradient;  
context.fillRect(0, 0, 300, 225);
```

وهذه هي النتيجة:



الشكل 11: تدرج لوني قطري

6. الصور

Android	iPhone	Opera	Chrome	Safari	Firefox	IE
+1.0	+1.0	+10.0	+3.0	+3.0	+3.0	*+7.0

* يتطلب IE 7 و 8 مكتبة خارجية هي `explorercanvas`, أما IE 9 فهو يدعم تضمين الصور في العنصر `<canvas>` داخلياً.

يُظهر الشكل الآتي قطةً معروضة عبر عنصر ``:

الشكل 12: قطة معروضة عبر عنصر `img`

أما هذه فهي نفس القطة لكن مرسومة في عنصر `canvas`:



الشكل 13: قطة معروضة ضمن عنصر `canvas`

هناك دالة اسمها `drawImage()` في إطار الرسم تُستعمل لتضمين صورة في عنصر `canvas`. تأخذ هذه الدالة ثلاثة، أو خمسة، أو تسعه وسائل:

- المعطية (`dx, dy`) هي إحداثيات الزاوية العليا اليسرى من الصورة. فلو مررنا الإحداثيات (`0, 0`) فسترسم الصورة في الزاوية العليا اليسرى من لوحة الرسم.

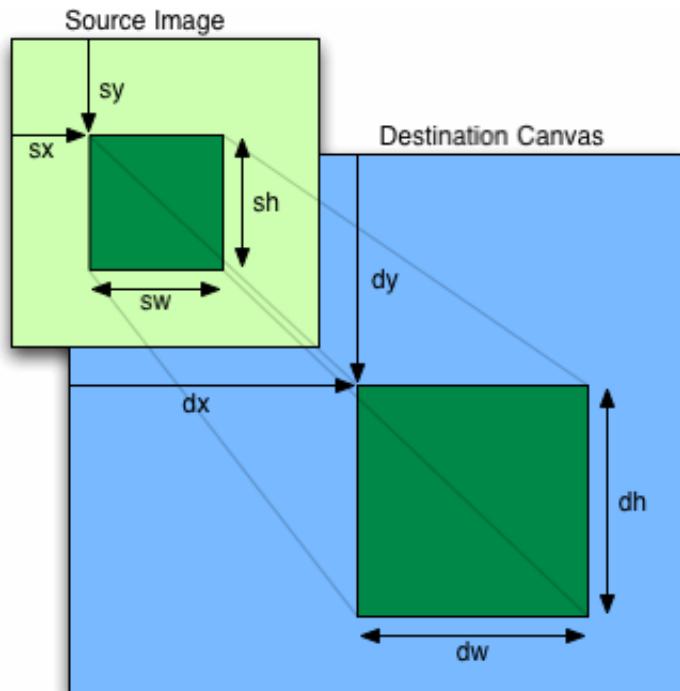
- وارتفاعها إلى `dh`, ثم ترسمها على لوحة الرسم في الإحداثيات (`dx, dy`).

- وتقطعها لتصبح مساويةً إلى المستطيل (`sx, sy, sw, sh`) ثم تغير أبعادها إلى (`dw, dh`) ثم ترسمها على لوحة الرسم في الإحداثيات (`dx, dy`).

توضّح مواصفات `HTML5 drawImage() parameters` الدالة

المستطيل المصدر هو المستطيل (ضمن حدود الصورة المصدرية) الذي تكون رؤوس أعلاه هي (sx, sy) ، و $(sx+sw, sy)$ ، و $(sx+sw, sy+sh)$ ، و $(sx, sy+sh)$.

أما المستطيل الوجهة، فهو المستطيل (ضمن حدود لوحة الرسم) الذي تكون رؤوس أعلاه هي النقاط (dx, dy) ، و $(dx+dw, dy)$ ، و $(dx+dw, dy+dh)$ ، و $(dx, dy+dh)$.



الشكل 14: تمثيل رسموي لمعاملات الدالة `drawImage`

لرسم صورة داخل لوحة الرسم، فستحتاج إلى صورة! يمكن أن تكون الصورة عنصر موجود مسبقاً، أو بإمكانك إنشاء كائن (Image) باستخدام JavaScript. وفي كلا الحالتين عليك أن تتأكد أنَّ الصورة محمَّلة تحميلاً كاملاً قبل أن ترسمها في لوحة الرسم.

إذا كنت تستخدم عنصر موجود مسبقاً، فيمكنك رسم الصورة في لوحة الرسم أثناء .window.onload الحدث

```
// استخدام عنصر
![sleeping cat](images/cat.png)

<script>
window.onload = function() {
    var canvas = document.getElementById("e");
    var context = canvas.getContext("2d");
    var cat = document.getElementById("cat");
    context.drawImage(cat, 0, 0);
};
</script>
```

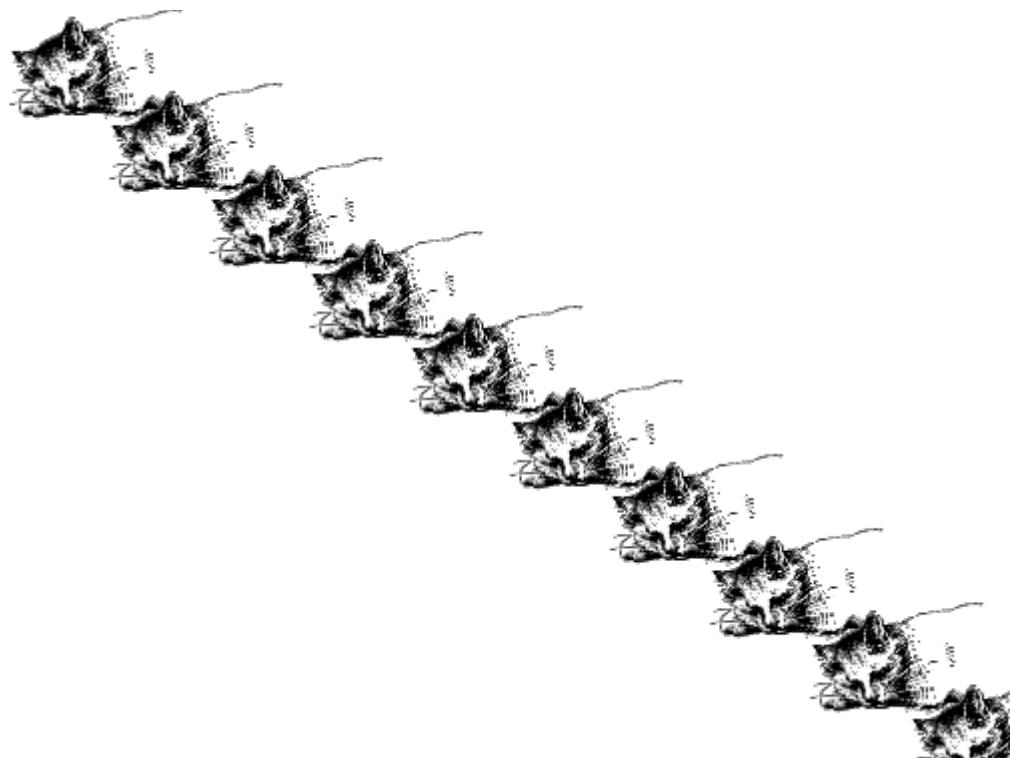
أما لو كنت تُنشئ كائناً الصورة عبر JavaScript، فيمكنك رسم الصورة داخل لوحة الرسم أثناء الحدث .Image.onload

```
// Image كائن()
<canvas id="e" width="177" height="113">
<script>
    var canvas = document.getElementById("e");
    var context = canvas.getContext("2d");
    var cat = new Image();
    cat.src = "images/cat.png";
    cat.onload = function() {
        context.drawImage(cat, 0, 0);
```

```
};  
</script>
```

المعاملان الاختياريان الثالث والرابع في دالة `drawImage()` يتحكمان في تغيير أبعاد الصورة.

هذه هي الصورة نفسها بعد تقليل أبعادها إلى النصف مرسومةً بشكل تكراري في إحداثياتٍ مختلفة ضمن لوحة الرسم نفسها.



الشكل 15: الكثير من القطط!

هذا هو السكريت الذي يؤدي إلى رسم الشكل السابق:

```

cat.onload = function() {
    for (var x = 0, y = 0;
        x < 500 && y < 375;
        x += 50, y += 37) {
        // تغيير أبعاد الصورة
        context.drawImage(cat, x, y, 88, 56);
    }
}

```

ربما تتساءل سؤالاً منطقياً: لماذا تريدين رسم صورة في لوحة الرسم أصلاً؟ بماذا ستستفيد من التعقيد الناتج عن رسم صورة داخل لوحة الرسم عوضاً عن استخدام عنصر `` وبعض خصائص CSS؟ حتى التأثير السابق (تكرار صورة القطة) يمكن أن يتم باستخدام 10 عناصر `` متداخلة.

الجواب البسيط هو: لنفس سبب حاجتك إلى **رسم النصوص في لوحة الرسم**. لاحظ كيف تضمن **مخطط إحداثيات canvas** نصاً ومستقيمات وأشكالاً فووضع النص ضمن لوحة الرسم هو جزءٌ من عملٍ أكبر. والمخططات الأكثر تعقيداً ستستفيد كثيراً من الدالة `drawImage()` لتضمين الأيقونات والرسوميات وغيرها.

7. ماذا عن متصفح IE؟

لم تكن الإصدارات الأقدم من الإصدار التاسع من متصفح Internet Explorer تدعم الواجهة البرمجية (API) لعنصر `canvas` (يدعم IE9 واجهة `canvas` البرمجية بالكامل). لكن الإصدارات القديمة من Internet Explorer تدعم تقنية مملوكة من مايكروسوفت اسمها VML، التي تفعل عدد من الأشياء التي يفعلها العنصر `<canvas>`، وبهذا ولدت المكتبة `excanvas.js`.

برخصة Apache التي تجعل من الممكن استعمال واجهة canvas البرمجية في متصفح Internet Explorer. عليك تضمين وسم `<script>` الآتي في أعلى صفتوك لكي تستخدمها.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Dive Into HTML5</title>
    <!--[if lt IE 9]>
        <script src="excanvas.js"></script>
    <![endif]-->
</head>
<body>
    ...
</body>
</html>
```

الأسطر `<!--[if lt IE 9]>` و `<![endif]-->` تسمى **التعليقات الشرطية**. إذ يعاملها Internet Explorer كأنها عبارة `if` الشرطية: «إن كان الإصدار الحالي من متصفح Internet Explorer أقل من الإصدار 9، فنفذ الشيفرة الآتية.»، وستعامل بقية المتصفحات تلك الشيفرة على أنها تعليقات HTML. فالخلاصة هي أن متصفح Internet Explorer (الإصدار الثامن والسابع) سيُنزل السكريبت ثم سيُنفذه، لكن ستتجاهله بقية المتصفحات تماماً، وهذا يجعل صفتوك تحمل بشكل أسرع في المتصفحات التي تدعم واجهة canvas البرمجية داخلياً (أي أنها لن تُنزل السكريبت، ولن تنفذه، ولن تفعّل أي شيء معه).

ليس عليك فعل أي شيء إضافي بعد تضمين `excanvas.js` في ترويسة صفتوك. ضمّن عناصر `<canvas>` في شيفرتوك أو أنشأها ديناميكياً باستخدام JavaScript، واتبع التعليمات

الواردة في هذا الفصل للحصول على لوحة رسم عنصر `<canvas>`، ثم تستطيع البدء برسم الأشكال والنصوص والنقوش.

ليس تماماً... هنالك بعض المحدوديات:

1. يمكن أن تكون **الدرجات اللونية خطية** فقط. التدرجات اللونية الشعاعية غير مدعومة.
2. يجب تكرار النقوش (patterns) في كلا الاتجاهين.
3. ميزة **Clipping regions** غير مدعومة.
4. **تغيير الأبعاد** غير المنتظم لا يستطيع تغيير أبعاد حدود الأشكال (strokes) بشكلٍ صحيح.
5. إنها بطيئة؛ ولا أظن أن ذلك صادم لأي شخص، فمن المعلوم أن مفسّر JavaScript في متصفح Internet Explorer أبطأ من غيره من المتصفحات؛ مما يالك برسم أشكال معقدة عبر مكتبة JavaScript ستتحول الدوال والأوامر إلى تقنية مختلفة تماماً؟ لن تلاحظ بطيئاً في الأداء في الأمثلة البسيطة مثل رسم بعض الخطوط وإجراء بعض العمليات على صورة، لكنك ستري ذلك بوضوح بعد أن تحاول إنشاء تأثيرات حركية باستخدام تقنية `.canvas`.

هنالك تحذير آخر حول استخدام `js.excanvas`، وهذه مشكلة واجهتني أثناء إنشاء أمثلة هذا الفصل. يُheiئ `ExplorerCanvas` واجهة البرمجية لعنصر `canvas` في مكان تضمينك لسكربت `js.excanvas` في صفحة `HTML`. لكن ذلك لا يعني أن `Internet Explorer` جاهز لاستخدامها مباشرةً، وفي بعض الحالات ستواجه حالة خاصة تكون فيها واجهة `canvas` جاهزة «تقريباً» (لكن ليس تماماً) للاستعمال. الأعراض الظاهرة لهذه المشكلة هي أن `Internet`

«object doesn't support this property or method» سيشتكي من أنَّ Explorer في كل مرة تحاول فيها فعل أي شيء مع عنصر `<canvas>` مثل الحصول على لوحة الرسم الخاصة به.

أسهل حل لهذه المشكلة هي تأجيل كل العمليات التي ستجريها على `canvas` إلى أن يحدث الحدث `onload`, ولكن هذا قد يستغرق بعضاً من الوقت، خصيصاً إذا كانت في صفحتك الكثير من الصور أو مقاطع الفيديو، التي ستؤخر الحدث `onload`, لكن ذلك سيعطي المكتبة `ExplorerCanvas` وقتاً لفعل اللازم.

8. مثالٌ متكمٌ

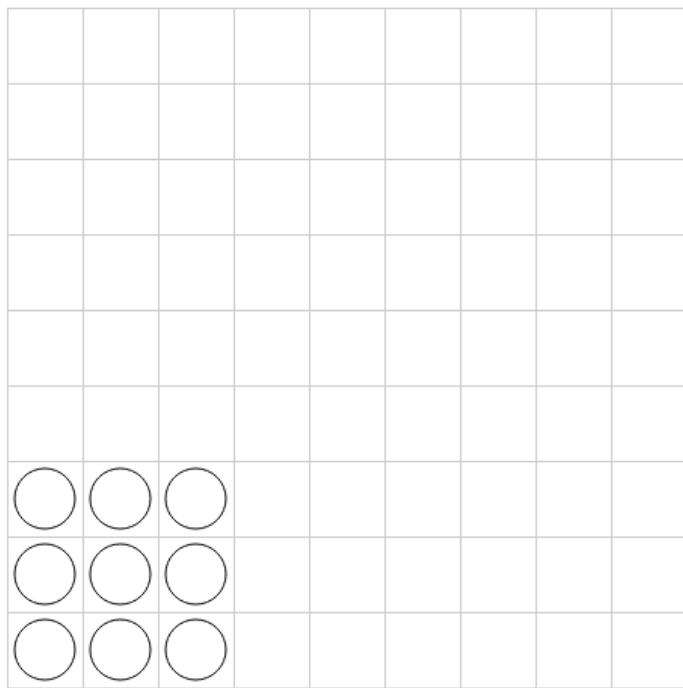
«الضامة» هي لعبة قديمةً جداً، وهناك عدّة نسخ منها؛ أنشأُت في هذا المثال نسخة السوليتر (أي التي يلعبها شخص واحد فقط) من الضامة ذات تسع قطعٍ في رقعةٍ مقاسها 9×9 . تتواجد القطع في بداية اللعبة في مربع 3×3 في الركن الأسفل الأيسر من الرقعة. هدف اللعبة هو نقل جميع القطع كي تُشكّل مربع 3×3 في الركن الأعلى الأيمن من الرقعة بأقلِّ عددٍ من الحركات.

هناك نوعان من الحركات المسموحة في الضامة:

- خذ قطعةً وحركها إلى أي مربع فارغ مجاور. المربع «الفارغ» يعني أنَّه لا يحتوي على قطعةٍ فيه حالياً. أما «مجاور» فهي تعني المربع الذي يقع مباشراً في شمال أو جنوب أو شرق أو غرب أو الشمال الغربي أو الشمال الشرقي أو الجنوب الغربي أو الجنوبي الشرقي من موقع القطعة الحالي (لا يمكن الالتفاف في الرقعة من طرفٍ آخر لآخر، فلو كانت قطعتك في العمود الأيسر من الرقعة، فلا يمكنك أن تحرکها نحو الغرب أو الشمال

الغربي أو الجنوب الغربي. وإذا كانت قطعتك في السطر الأدنى من الرقعة، فلا تستطيع تحريكها نحو الجنوب أو الجنوبي الشرقي أو الجنوبي الغربي).

- يمكنك أخذ قطعة وجعلها «تففر» فوق قطعة مجاورة، ويمكنك تكرار ذلك. هذا يعني أنّه إذا قفزت فوق قطعة مجاورة ثم قفزت فوق قطعة أخرى مجاورة لموقعك الجديد فكل هذا يُحسب على أنّه حركة واحدة. وفي الحقيقة، يتم احتساب أي عدد من القفزات على أنها خطوة واحدة (لأن الهدف من اللعبة هو تقليل العدد الكلي من الحركات؛ ويطلب اللعب بشكلٍ جيد في الصamaة بناءً واستخداماً لسلسل طويلة من القطع لكي تستطيع القطع الأخرى القفز قوفها في سلسل طويلة من الحركات).
- هذه لقطة شاشة للعبة، يمكنك زيارة [الصفحة الآتية](#) لتجربة اللعبة تجربة حيةً أو لـلقاء نظرة إلى مصدر الصفحة.



Moves: o

الشكل 16: لقطة شاشة للعبة الضامة

كيف تعمل؟ أنا ممتن لسؤالك. لن أريك كل الشيفرات هنا (يمكنك رؤيتها في [halma.js](#))، وسأختصر معظم الشيفرات التي تستعمل في برمجة آلية اللعب، لكنني سأركّز على بعض الأجزاء التي مهمتها هي الرسم في عنصر `canvas` والاستجابة إلى نقرات الفأرة. سننهيّئ اللعبة أثناء تحميل الصفحة بضبط أبعاد عنصر `<canvas>` ثم تخزين مرجع متعلق بلوحة الرسم.

```
gCanvasElement.width = kPixelWidth;
gCanvasElement.height = kPixelHeight;
gDrawingContext = gCanvasElement.getContext("2d");
```

ثم سنفعل شيئاً لم نفعله من قبل قط: سنضيف متابع أحداث إلى العنصر `<canvas>` لكي يتبع حدث النقر على العنصر.

```
gCanvasElement.addEventListener("click", halmaOnClick, false);
```

ستُستخدم الدالة `halmaOnClick()` عندما ينقر المستخدم في أي مكان ضمن العنصر `canvas`, والوسط الذي يُمَرِّر إليها هو كائن `MouseEvent` الذي يحتوي على معلومات حول مكان نقر المستخدم.

```
function halmaOnClick(e) {
    var cell = getCursorPosition(e);
    // بقية الشيفرات مرتبطة بالية اللعب
    for (var i = 0; i < gNumPieces; i++) {
        if ((gPieces[i].row == cell.row) &&
            (gPieces[i].column == cell.column)) {
            clickOnPiece(i);
            return;
        }
    }
    clickOnEmptyCell(cell);
}
```

الخطوة الآتية هيأخذ الكائن `MouseEvent` وحساب في أي مربع في رقعة الضامة قد تم النقر. تحتل رقعة الضامة كامل عنصر `canvas`, ولهذا تكون كل نقرة هي نقرة في مكان ما على الرقعة، وكل ما علينا هو معرفة المكان. هذا أمرٌ معقدٌ بعض الشيء، لأن الأحداث المولدة من الفأرة تختلف بشكلٍ أو آخر بين المتصفحات.

```
function getCursorPosition(e) {
    var x;
    var y;
    if (e.pageX != undefined && e.pageY != undefined) {
        x = e.pageX;
        y = e.pageY;
    }
    else {
        x = e.clientX + document.body.scrollLeft +
            document.documentElement.scrollLeft;
        y = e.clientY + document.body.scrollTop +
            document.documentElement.scrollTop;
    }
}
```

أصبح لدينا في هذه المرحلة إحداثيات x و y نسبةً إلى المستند (أي صفحة HTML كلها)، أي أنها ليس مفيدةً بعد؛ إذ نريد أن تكون الإحداثيات منسوبةً إلى لوحة الرسم.

```
x -= gCanvasElement.offsetLeft;
y -= gCanvasElement.offsetTop;
```

أصبح لدينا إحداثيات x و y منسوبةً إلى لوحة الرسم، هذا يعني أنه لو كانت قيمة x مساويةً للصفر و y مساوية للصفر، فهذا يعني أن المستخدم قد نقر على البكسل في أعلى وأيسير لوحة الرسم.

من هنا سنستطيع معرفة أي مربع قد نقر المستخدم عليه، ثم نتصرف بناءً على ذلك.

```
var cell = new Cell(Math.floor(y/kPieceHeight),
                    Math.floor(x/kPieceWidth));
return cell;
}
```

حسناً! ألم الأحداث المرتبطة بالفأرة معقدٌ. لكنك يمكنك استخدام نفس التسلسل المنطقي (أو بالأحرى، يمكنك استخدام الشيفرة نفسها) في جميع البرمجيات التي تعتمد على عنصر

. تذكر: نقرة بالفأرة ← إحداثيات منسوبة إلى المستند كله ← إحداثيات منسوبة إلى لوحة الرسم ← شيفرة خاصة بالتطبيق.

حسناً، لنلق نظرة على آلية الرسم الرئيسية. لما كانت الرسوميات بسيطةً جدًا، فقررت مسح محتويات لوحة الرسم وإعادة رسم الرقيقة في كل مرة يحصل فيها تغيير في اللعبة ولكن هذا ليس ضروريًا. سيحتفظ إطار الرسم في عنصر `canvas` بمحتواه الذي رسمته عليه حتى لو تخطاه المستخدم بتمريره في الصفحة، أو إذا غير إلى لسانٍ (tab) آخر ثم عاد إليه لاحقاً. إذا كنت تطّور تطبيقاً مبنّياً على عنصر `canvas` فيه رسوميات معقدة (مثل ألعاب الورق) فيمكنك تحسين الأداء بتتبع المناطق التي يجب مسحها من لوحة الرسم وإعادة الرسم داخل تلك المناطق فقط. لكن هذا خارج عن نطاق هذا الكتاب.

```
gDrawingContext.clearRect(0, 0, kPixelWidth, kPixelHeight);
```

يجب أن تكون الشيفرات الآتية مألوفةً لديك، لأنها شبيهة بتلك التي استعملناها لرسم مخطط إحداثيات `canvas` سابقًا.

```
gDrawingContext.beginPath();
```

الخطوط الرأسية //

```
for (var x = 0; x <= kPixelWidth; x += kPieceWidth) {
    gDrawingContext.moveTo(0.5 + x, 0);
    gDrawingContext.lineTo(0.5 + x, kPixelHeight);
}
```

الخطوط الأفقية //

```
for (var y = 0; y <= kPixelHeight; y += kPieceHeight) {
    gDrawingContext.moveTo(0, 0.5 + y);
```

```

gDrawingContext.lineTo(kPixelWidth, 0.5 + y);
}

// لرسم الرسمة!
gDrawingContext.strokeStyle = "#ccc";
gDrawingContext.stroke();

```

المتعة الحقيقة تبدأ عندما نهم برسم القطعة. القطعة هي دائرة، لكننا لم نرسم الدوائر من قبل، وعلاوةً على ذلك، لو اختار المستخدم قطعةً لكي يحركها، فسنحتاج إلى رسم تلك القطعة كدائرة ممولة. يُمثل المعامل `p` هنا قطعة، التي لها خاصيات `row` و `column` (سطر) و (عمود) التي تحدد المكان الحالي للقطعة. سنستخدم آلية لتحويل `(column, row)` إلى إحداثيات منسوبة إلى لوحة الرسم `(y, x)`، ثم سنرسم دائرة، ثم -إن كانت القطعة محددةً من المستخدم- سنملأ الدائرة بلونٍ معين.

```

function drawPiece(p, selected) {
    var column = p.column;
    var row = p.row;
    var x = (column * kPieceWidth) + (kPieceWidth/2);
    var y = (row * kPieceHeight) + (kPieceHeight/2);
    var radius = (kPieceWidth/2) - (kPieceWidth/10);
}

```

أصبح لدينا الآن إحداثيات `(y, x)` منسوبة إلى لوحة الرسم لمركز الدائرة التي نريد رسمها. للأسف لا توجد دالة باسم `circle()` في واجهة `canvas` البرمجية، لكن هناك الدالة `arc()`؛ لكن الدائرة ما هي إلا قوسٌ تتصل بدايته مع نهايته. هل تتذكر أساسيات الهندسة؟ تأخذ الدالة `arc()` نقطة المركز `(y, x)`، ونصف القطر، وزاوية البداية والنهاية (بالراديان)، وراية `flag` لتحديد اتجاه الدوران (`false` لاتجاه دوران عقارب الساعة، و `true` لعكس جهة دوران)

عقارب الساعة). يمكنك استخدام الكائن `Math` الموجود في لغة JavaScript لحساب الزوايا بالرadian.

```
gDrawingContext.beginPath();
gDrawingContext.arc(x, y, radius, 0, Math.PI * 2, false);
gDrawingContext.closePath();
```

انتظر برهةً! لم يرسم شيء. الدالة `arc()` تشبه عمل `moveTo()` و `lineTo()`. إذ ترسم «**بقلم الرصاص**». ولرسم الدائرة فعليًا، علينا ضبط خاصية `strokeStyle` ثم استدعاء الدالة `stroke()` «لتحبيرها».

```
gDrawingContext.strokeStyle = "#000";
gDrawingContext.stroke();
```

ماذا لو كانت القطعة محددةً؟ يمكننا إعادة استخدام نفس المسار الذي أنشأناه لرسم حدود القطعة لمئها بلونٍ معينٍ.

```
if (selected) {
    gDrawingContext.fillStyle = "#000";
    gDrawingContext.fill();
}
```

هذا كل ما في الأمر. تهتم بقية السكريبت بالبنية المنطقية للعبة، مثل التفريقي بين الحركات الصحيحة وغير الصحيحة، وإحصاء عدد الحركات، ومعرفة إذا انتهت اللعبة. مرحبًا! لقد أنشأنا لعبةً عبر عنصر `canvas` برسم 9 دوائر وبعض المستقيمات ودالة وحيدة `onclick` مربطة بالحدث.

9. مصادر إضافية

- Mozilla Developer Center في [Canvas tutorial](#) •
- Mihai Sucan من كتابة [HTML5 canvas – the basics](#) •
- موقع [CanvasDemos.com](#) الذي يحتوي على أدوات ونماذج ودوروس متعلقة بعنصر canvas •
- HTML5 في مواصفة [The canvas element](#) •
- [Internet Explorer 9 Guide for Developers: HTML5 canvas element](#) •

دورة إدارة تطوير المنتجات



مميزات الدورة

- ✓ شهادة معتمدة من أكاديمية حسوب
- ✓ إرشادات من المدربين على مدار الساعة
- ✓ من الصفر دون الحاجة لخبرة مسبقة
- ✓ بناء معرض أعمال قوي بمشاريع حقيقة
- ✓ وصول مدى الحياة لمحتويات الدورة
- ✓ تدريبات مستمرة على الدورة مجاناً

اشترك الآن



0

الفيديو

أي شخص زار موقع YouTube في الأعوام الماضية يعلم تمام العلم أن بالإمكان تضمين مقاطع الفيديو في صفحات الويب؛ لكن لم تكن هنالك طريقة معيارية لفعل ذلك قبل وجود HTML5. نظرًا كل مقاطع الفيديو التي سبق وأن شاهدتها على «الويب» تم تشغيلها عبر إضافة خارجية (ربما QuickTime أو Flash [كان YouTube يستخدم تقنية Flash]).

تتكامل تلك الإضافات مع متصفحك تكاملاً ممتازاً إلى درجة أنه لن تلحظ استخدامها إلى أن تحاول مشاهدة مقطع فيديو على منصة (أو جهاز) لا تدعم تلك الإضافة.

تعزز HTML5 طريقة معيارية لتضمين مقاطع الفيديو في صفحة الويب وذلك باستخدام العنصر <video>، ما يزال دعم العنصر <video> قيد التطوير، وهذه طريقة مهذبة للتصرير أنه لا يعمل بعد، أو على الأقل لا يعمل في كل مكان؛ لكن لا تقنط ولا تقلق، هنالك بدائل وخيارات أخرى كثيرة نستطيع اللجوء إليها.

يبين الجدول الآتي إصدارات مختلف المتصفحات التي تدعم العنصر <video>:

Android	iPhone	Opera	Chrome	Safari	Firefox	IE
+2.0	+1.0	+10.5	+3.0	+3.0	+3.5	+9.0

لكن الدعم لعنصر <video> نفسه ليس كافياً بل هو جزء صغير من الحكاية. لكن قبل أن نتحدث عن دعم الفيديو في HTML5، علينا أن نفهم بعض المعلومات حول مقاطع الفيديو نفسها (إن كنت تعرف تلك المعلومات، فيمكنك تخطي الفقرات الآتية والبدء من قسم «[الصيغ التي تعمل في الويب](#)»).

1. حاويات الفيديو

قد تتخيل ملفات الفيديو على أنها ملفات «AVI» أو «MP4». لكن «AVI» و «MP4» في الواقع ما هي إلا صيغ حاوية للفيديو. فهي تُشَبِّه ملفات ZIP التي يمكنها احتواء أي نوع من الملفات ضمنها، فصيغ حاويات الفيديو (video container formats) تُعْرَف طريقةً تخزين الأشياء ضمنها فقط، ولا تُحدِّد ما هي أنواع البيانات المخزَّنة (الأمر أكثر تعقيداً من هذا، لعدم توافق جميع أنواع مسارات [أو دفق] الفيديو [video streams] مع جميع صيغ الحاويات، لكن اغضض الطرف عنها الآن).

يحتوي ملف الفيديو عادةً على عدّة «مسارات» (tracks)، التي هي مسار الفيديو (دون صوت) بالإضافة إلى مسار صوتي أو أكثر (دون فيديو). تكون المسارات مترابطةً عادةً، فيحتوي مسار الصوت على مؤشرات أو علامات ضمنه للمساعدة في مزامنة الصوت مع الفيديو. يمكن لكل مسار أن يملك بيانات وصفية (metadata)، مثل نسبة العرض إلى الارتفاع في مسار الفيديو، أو لغة مسار الصوت. يمكن للحاويات أيضًا امتلاك بيانات وصفية، مثل عنوان (title) للفيديو نفسه، وغلاف (cover) للفيديو، وأرقام الحلقات (للمسلسلات)، وهلم جراً.

هناك العديد والعديد من صيغ حاويات الفيديو، هذه أشهرها:

- **MPEG 4**، التي تأتي عادةً مع اللاحقة `.mp4` أو `.m4v`. حاوية 4 MPEG مبنية على حاوية QuickTime القديمة (من Apple)، ما تزال تستخدم «الأفلام القصيرة» في موقع iTunes حاوية Apple القديمة، لكن الأفلام التي تأخذها من iTunes مبنية على حاوية 4 MPEG.

- Flash Video، تأتي عادةً مع اللاحقة `.flv`، صيغة Flash Video (بدهياً) تستعمل من Adobe Flash؛ وكانت هذه هي صيغة الحاويات الوحيدة التي كان يدعمها Flash قبل الإصدار 9.0.60.184 (أي مشغل Flash الإصدار 9 التحديث 3). الإصدارات الحديثة من Flash تدعم حاوية `MPEG 4` أيضاً.
- `Ogg`، تأتي عادةً مع اللاحقة `.ogg`؛ صيغة `Ogg` معيارها مفتوح المصدر، وغير مرتبطة بأية براءات اختراع. متصفحات `Firefox 3.5` و `Opera 10.5` و `Chrome 4` و `Firefox 3.5` و `Opera 10.5` و `Chrome 4` تدعم `Ogg`، دون إضافات خاصة-الحاوية `Ogg`، ومسار فيديو `Ogg` (المسمى «`Theora`»)، ومسار صوت `Ogg` (المسمى «`Vorbis`»). صيغة `Ogg` مدعومة تلقائياً في جميع توزيعات لينكس المشهورة، ويمكنك استخدامها على `Mac` و`Windows` بثبيت «`DirectShow filters`» أو «`QuickTime components`» على التوالي وبالترتيب. من الممكن أيضاً تشغيل هذه الصيغة ببرنامج `VLC` على جميع المنصات.
- `WebM`، هي صيغة حاويات جديدة، وهي -تقنياً- شبيهة بصيغة أخرى تسمى `Matroska`. أُعلنَ عن `WebM` في أيار/مايو من عام 2010، وهي مصممة لكي تستخدم حصرياً مع مرماز الفيديو `VP8` ومرماز الصوت `Vorbis` (سأتأتي على ذكرهما بعد قليل)، ثم حدّدت في عام 2013 لتدعم مرماز `VP9` للفيديو ومرماز `Opus` للصوت؛ وهذه الصيغة مدعومة داخلياً -دون إضافات خاصة- في آخر إصدار من `Chromium` `10.1` `Adobe Flash` و `Google Chrome` و `Firefox` و `Mozilla Opera`، وأعلنت أن `WebM` سيدعم صيغة `WebM`.

- **Audio Video Interleave** AVI من صيغة **avi**.. تم ابتكار صيغة AVI من مايكروسوفت منذ وقتٍ طويلاً، حين كانت إمكانية تشغيل الحواسيب لمقاطع الفيديو أمرًا عظيماً. لا تدعم هذه الصيغة -رسمياً- ميزات صيغ الحاويات الأحدث منها مثل البيانات الوصفية المدمجة، ولا تدعم أيضاً -رسمياً- أغليبية مرمازات الفيديو والصوت المستعملة حالياً. وقد حاولت الشركات مع مرور الوقت توسيع هذه الصيغة بطرق غير متوافقةٍ مع بعضها البعض لدعم بعض الميزات، ولكنها مع ذلك بقيت صيغة الحاوية الافتراضية لبعض المرمّزات (encoders) مثل **MEncoder**.

2. مرمازات الفيديو

عندما تتحدث عن «مشاهدة مقطع فيديو» فغالباً ستقصد مشاهدة تجميعية من مسار للفيديو ومسار للصوت؛ لكن ليس عندك ملفين منفصلين، فكل ما لديك هو ملف «فيديو» واحد، وربما يكون ملف AVI أو MP4؛ تلك هي **صيغ الحاويات**، مثل ملف ZIP الذي يحتوي على عدّة أنواع من الملفات داخله. تُعرّف صيغة الحاوية آلية تخزين مسارات الفيديو والصوت في ملفٍ واحد.

سيقوم **مُشغل الفيديو** بثلاثة أشياء على الأقل عندما «تشاهد مقطع فيديو»:

1. محاولة تفسير صيغة الحاوية لمعرفة ما هي مسارات الفيديو والصوت المتوفرة، وطريقة تخزينها ضمن الملف كي يعرف أين سيعثر على البيانات التي يجب عليه فك ترميزها لاحقاً
2. فك ترميز مسار الفيديو وعرض سلسلة من الصور على الشاشة
3. فك ترميز مسار الصوت وإرسال الأصوات إلى مكبرات الصوت في جهازك

رماز الفيديو (video codec) هو الخوارزمية التي يُرمّز (encode) فيها مقطع الفيديو، أي أنه يُحدّد طريقة القيام بالخطوة رقم 2 مما سبق (الكلمة «codec» - أي مرماز-آتية من دمج الكلمتين «coder» و «decoder»). يفك مشغل الفيديو ترميز (decode) مسار الفيديو وفقاً لرماز الفيديو المستخدم، ثم سيعرض سلسلة من الصور أو «الإطارات» على الشاشة، أغلبية مرمازات الفيديو الحديثة تستخدم حيلاً عديدة لتقليل حجم المعلومات الالزمة لعرض الإطار تلو الإطار. على سبيل المثال، بدلاً من تخزين كل إطار على حدة (كأنه لقطة شاشة)، فسيتم تخزين الاختلافات بين الإطارات. لا يحدث في أغلبية مقاطع الفيديو تغيرات كبيرة بين الإطارات المتتالية، مما يسمح بضغطها بدرجة أكبر، مما يؤدي إلى تقليل المساحة التخزينية للملف.

هناك مرمازات فيديو فقودة (lossy) وأخرى لا فقودة (lossless). المساحة التخزينية لملفات الفيديو غير الفقودة كبيرة جدًا ولن تكون ذات فائدة في الويب، لهذا سنركز على المرمازات الفقودة. المرماز الفقود يعني أن بعض المعلومات ستضيع دون إمكانية استعادتها أثناء عملية الترميز (encoding)، مثل عملية نسخ كاسيت صوتي إلى آخر. ست فقد في هذه العملية بعض المعلومات المتعلقة بالفيديو المصدري، وستقلل الجودة في كل مرة تعيد ترميز المقطع فيها. فبدلاً من صوت «التشويش» في شريط الكاسيت (إن كنت تستعمله في صغرك)، سيبدو مقطع الفيديو الذي تعيد ترميزه مراتٍ عدّة غير واضح، خصوصاً المشاهد التي فيها الكثير من الحركة (في الواقع، يمكن أن يحدث ذلك حتى لو قمت بالترميز من المصدر مباشرةً، فربما اخترت مرماز فيديو سيئاً أو مررت مجموعة وسائل خطأ). لكن في المقابل، تضغط مرمازات الفيديو الفقودة مقاطع الفيديو ضغطاً كبيراً ويصعب في الوقت نفسه ملاحظة فقدان البيانات بالعين المجردة.

هناك الكثير من مرمادات الفيديو، لكن أهم ثلاثة منها هي H.264، و Theora، و VP9.

. H.264

MPEG-4 معروف أيضًا باسم H.264 أو «MPEG-4 AVC» أو «MPEG-4 part 10».

أصبح معيارًا قياسيًا في Advanced Video Coding (AVC). طور مرماد H.264 من MPEG group وأصبح معيارًا قياسيًا في 2003، ويهدف إلى توفير مرماد واحد للأجهزة ذات التراسل الشبكي المحدود وقدرة المعالجة المحدودة (أي الهاتف المحمولة)، وللأجهزة التي يتوفّر لها تراسل شبكي كبير وقدرة معالجة ممتازة (مثل الحواسيب الحديثة)، وأي شيء يقع بينهما. ولكن يتم تحقيق ذلك، فـ H.264 يحدّد كلًّ منها مجموعةً من الميزات الاختيارية التي توازن بين تعقيد ترميز الملف ومساحته التخزينية. الأنماط العليا (Higher profiles) تستخدم ميزات اختيارية أكثر، وتتوفر جودةً أكبر بمساحةٍ تخزينية أقل، لكنها تأخذ وقتاً أطول لترمذ، وتستهلك طاقة معالجة أكبر لفك الترميز في الوقت الحقيقي.

لإعطائك فكرةً عامةً عن تنوع الأنماط: يدعم iPhone نمط Baseline، ويدعم AppleTV «Baseline» و «Main»، بينما يدعم Adobe Flash على الحواسيب أنماط «Baseline» و «Main» و «High». ويستعمل YouTube الآن مرماد H.264 لترميز مقاطع الفيديو عالية الدقة (HD) التي يمكن تشغيلها عبر YouTube؛ ويدعم Adobe Flash أيضًا إرسال مقاطع الفيديو المرمزة برماد H.264 إلى الهواتف المحمولة، بما في ذلك هواتف iPhone والهواتف العاملة بنظام أندرويد. إضافةً إلى ما سبق، مرماد H.264 هو المرماد الذي تستعمله مواصفة Blu-Ray، إذ تستخدم أقراص Blu-Ray نمط «High» في العموم.

أغلبية الأجهزة التي تشغّل فيديو H.264 عدا الحواسيب (بما في ذلك هواتف iPhone وقارئات Blu-Ray) تستعمل شريحة منفصلة لفك ترميز الفيديو، لأن معالجاتها المركزية ضعيفة نسبياً لفك ترميز الفيديو في الوقت الحقيقي. وأصبحت في هذه الأيام بطاقات العرض الرخيصة للحواسيب المكتبية تدعم فك ترميز فيديو H.264 عادةً. وهناك أيضاً [نافس بين مرمزات H.264](#)، بما في ذلك المكتبة مفتوحة المصدر [x264](#). معيار H.264 محمي ببراءات اختراع؛ ويتم الترخيص عبر [MPEG LA group](#).

يمكن تضمين فيديو H.264 في [أغلبية الحاويات الحديثة](#)، بما في ذلك MP4 (تستعملها Apple بشكلٍ أساسي في متجر iTunes) و MKV (التي تستعمل بشكلٍ أساسي من هواة الفيديو الذين ليس لهم هدف تجاري).

بـ. Theora

تم تطوير Theora من [رماز VP3](#) ثم طُور من مؤسسة Xiph.org مرماً مجاني الاستخدام (royalty-free) وليس محمياً بأية براءات اختراع عدا براءات اختراع مرماز VP3 الأصلي، الذي رُخص على أنه مجاني الاستخدام أيضاً. على الرغم من أنَّ المعيار قد «جُمد» (frozen) منذ عام 2004، إلا أنَّ Theora project (الذي يتضمن مكتبات ترميز وفك ترميز مفتوحة المصدر) قد أصدر النسخة 1.0 في تشرين الثاني/نوفمبر عام 2008، والإصدار 1.1 في أيلول/سبتمبر عام 2009.

يمكن تضمين الفيديو المرمز برمماز Theora في أية صيغة من صيغ الحاويات، لكن من الشائع أن نراه في حاوية Ogg. تدعم جميع توزيعات لينكس Theora، ويتضمن متصفح Mozilla Firefox 3.5 دعماً داخلياً للفيديو المرمز برمماز Theora؛ وأقصد بكلمة «داخلياً» أنه متوفراً على

جميع الأنظمة دون إضافات خاصة. يمكنك أيضًا تشغيل الفيديو المرمز برماز Theora على [Xiph.org](#) أو على Mac OS X بعد تثبيت برمجية فك الترميز المفتوحة المصدر من [Xiph.org](#). مرماز Theora قل استعماله في الفترة الماضية، وحلّ مرماز VP9 محلّه.

ج. VP9 و VP8

(Theora VP8 هو مرماز فيديو آخر من On2، الشركة التي طرأت VP3 (الذي أصبح لاحقًا [VP8](#)) بنفسها. بكلام تقني، يُنتج هذا المرماز فيديو بنفس جودة النمط «High» في H.264، في حين يحاول تقليل تعقيد عملية فك الترميز كما في نمط «Baseline» في H.264.

في عام 2010، اشتريت Google شركة On2 ونشرت مواصفات مرماز الفيديو وأصدرت Google برمجية ترميز وفك ترميز مفتوحة المصدر. وكان جزءاً من هذه العملية هو «فتح» جميع براءات الاختراع التي سجلتها شركة On2 لمرماز VP8، وذلك بجعلها مجانية الاستخدام أي royalty-free (وهذا أفضل ما يمكن فعله مع براءات الاختراع، فلا يمكنك حقاً «التخلص» عنها أو حذفها بعد تسجيلها. لكن لجعلها متوفقة مع البرمجيات مفتوحة المصدر فيمكن ترخيصها للاستخدام مجانيًّا، وبهذا يستطيع أي شخص استخدام التقنيات المغطاة من براءة الاختراع دون دفع أي شيء أو دون محاولة الحصول على ترخيص خاص). وبهذا أصبح VP8 مرمازًا عصريًّا مجاني الاستخدام ليس محميًّا بأية براءات اختراع عدا تلك التي سجلتها شركة On2 (وتملكها Google حالياً) والتي يمكن استعمالها مجاناً.

وفي نهاية عام 2012، أطلقت Google مرماز VP9 الذي طرأ على خطى مرماز VP8 وهو مرماز مفتوح المصدر ومحمي الإستخدام، وقد اشتهر كثيراً بعد استخدامه في موقع YouTube.

3. مرمazat al-sawt

ستحتاج إلى تضمين مسار صوتي في مقاطع الفيديو إلا إذا كنت ستنشر **الأفلام الصامتة** (قبل عام 1927) فقط. ومثل **رممازات الفيديو**، مرممازات الصوت هي الخوارزميات التي يُرْمَز بها مسار الصوت. وأيًّضاً مثل مرممازات الفيديو، هنالك مرممازات صوت غير فقدودة (encode) وأخرى فقدودة (lossy). وكما هو حال الفيديو غير الفقدود، ستكون المساحة التخزينية لمسارات الصوت غير الفقدود كبيرةً جدًا لكي تستفيد منها على الويب؛ لذلك سنرَكِّز على مرممازات الصوت التي تسبب فقدانًا لبعض البيانات.

وسأوضح المجال قليلاً، لوجود تصنيفاتٍ مختلفة لرممازات الصوت؛ إذ أنَّ الصوت يستعمل في أماكن لا يُستعمل فيها الفيديو (في الهاتف مثلاً)، وهنالك تصنيف كاملٌ من **رممازات الصوت المختخصة بترميز الكلام**، فلا تفكِّر في ترميز الموسيقى بهذه المرممازات، لأن النتيجة النهائية ستبدو وكأن طفلاً ذا أربعة أعوام يغني على الهاتف. لكنك تستطيع استخدامها في خدمة PBX من **Asterisk** لأن تراسل البيانات ثمين هناك، ولأن هذه المرممازات تضغط الصوت البشري ليأخذ مساحة تخزينية لا تشَكُّل إلا جزءاً بسيطاً من المساحة التخزينية التي كانت مستهلكها المرممازات ذات الغرض العام. لكن نتيجةً لعدم دعم تلك المرممازات في المتصفحات داخلياً أو عبر الإضافات الخارجية، فلم يسع نجم تلك المرممازات في الويب، ولهذا سنرَكِّز على مرممازات الصوت العامة الفقدودة.

وكما ذكرنا **سابقاً**، سيقوم حاسوبك بثلاثة أشياء على الأقل في وقتٍ واحد عندما «تشاهد مقطع فيديو»:

١. محاولة تفسير صيغة الحاوية

٢. فك ترميز مسار الفيديو

٣. فك ترميز مسار الصوت وإرسال الأصوات إلى مكبرات الصوت في جهازك

يُحدّد مرماز الصوت طريقة القيام بالخطوة رقم 3 مما سبق: فك ترميز مسار الصوت وتحويله إلى موجات رقمية يمكن لمكبرات الصوت عندك تحويلها إلى صوت. وكما في مرمازات الفيديو، هناك حيل عدّة تستعمل لتقليل حجم البيانات المخزّنة في مسار الصوت. وللّهَا كثا نتحدث عن مرمازات الصوت الفقدة، فسيفقد بعض المعلومات أثناء عملية «التسجيل → الترميز ← فك الترميز ← الاستماع». تحذف شتى مرمازات الصوت أشياءً مختلفة، لكنها تخدم كلّها الغرض نفسه: ألا تشعر بضياع أي شيء عندما تستمع إلى المسار.

هناك مفهوم في مسارات الصوت ليس موجوداً في مسارات الفيديو ألا وهو «القنوات» (channels). نحن نرسل الصوت إلى مكبرات الصوت، صحيح؟ حسناً، ما عدد مكبرات الصوت عندك؟ إذا كنت تجلس أمام حاسوب محمول فربما يكون عندك مكبران: أحدهما على اليمين والآخر على اليسار. لكن حاسوب المكتبي يملك ثلاثة مكبرات: واحد على اليمين وآخر على اليسار وثالث على الأرضية. أنظمة «**الصوت المحيطي**» (surround sound) السمعية تملك ستة مكبرات أو أكثر موزعة توزيعاً مدروساً في أنحاء الغرفة. مهمة كل مكبر أن يخرج قناة معينة من التسجيل الأصلي. الفكرة النظرية وراء ذاك النظام هي أنك ستجلس في منتصف المسافة الفاصلة بين تلك المكبرات محاطاً بست قنوات منفصلة من الصوت، وسيدرك مخك الصوت وستشعر كما لو أنك في منتصف الأحداث التي تسمع صوتها. لكن هل تعمل فعلًا كما شرحنا أعلاه؟ تقول الشركات ذات رؤوس الأموال الكبيرة أنها تعمل كما ينبغي.

تستطيع أغلبية مرمازات الصوت ذات الغرض العام التعامل مع قناتين من الصوت. يُقسم الصوت أثناء التسجيل إلى قناتين يمنى ويسرى؛ وستخزن كلا القناتين في المسار الصوتي نفسه أثناء الترميز؛ وسترسل محتويات كل قناة إلى مكبر الصوت الموافق لها أثناء فك الترميز. يمكن بعض مرمازات الصوت التعامل مع أكثر من قناتين، إذ يعرفون أي قناة يجب أن تُرسل لأي مكبر ثم سيتولى مُشغل الصوتيات عندك الأمر.

هناك الكثير من مرمازات الصوت، هل ذكرت لك سابقًا أنَّ هناك الكثير الكثير من مرمازات الفيديو؟ إنَّ ذلك! هناك **الكثير والكثير من مرمازات الصوت**، لكن يهمنا في الوبِ ثلَاثة منها:

.Vorbis و ACC و MP3

MPEG-1 Audio Layer 3 .

MPEG-1 Audio Layer 3 معروفة بالعامية على أنها «MP3»؛ لا أعرف من أي كوكب أتيت

إن لم تسمع من قبل عن MP3. تبيع شركة Walmart مشغلات موسيقى محمولة وتسميتها «MP3 players». وهي منتشرة انتشاراً كبيراً...

يمكن أن تحتوي ملفات MP3 على قناتين صوتيتين على الأكثُر، ويمكن ترميزها بمختلف معدلات البث (أي bitrates): 64 كيلوبت/ثانية، أو 128 كيلوبت/ثانية، أو 192 كيلوبت/ثانية، أو غيرها التي تتراوح قيمها من 32 إلى 320. قيم معدلات البث (bitrates) الأعلى تعني أنَّ مساحة الملفات أكبر وجودة الصوت أدق، لكن العلاقة بين جودة الصوت ومعدل البث ليست خطية (linear)، أي أنَّ جودة الأصوات المرمزة بمعدل بث 128 كيلوبت/ثانية أفضل بمقدار الضعفين من 64 كيلوبت/ثانية، لكن 256 كيلوبت/ثانية ليس أكثر جودة بمقدار الضعفين من 128 كيلوبت/ثانية). تسمح صيغة MP3 بالترميز ذي معدل البث المتغير (variable bitrate).

encoding، الذي يعني أنَّ أجزاءً من المسار مضغوطَة أكثر من غيرها). على سبيل المثال، يمكن ترميز السكت بين النotas الموسيقية بمعدل بث منخفض، ثم سيرتفع معدل البث عندما يُعرَف لحن معقدٌ على عدّة آلات موسيقية. يمكن أيضًا ترميز MP3 بمعدل بث ثابت، ولا غرابة أن يُسمى ذلك بالترميز ذي معدل البث الثابت (constant bitrate encoding).

لا يُعرَف معيار MP3 كيفية الترميز باستخدام MP3 تماماً (على الرغم من أنَّه يُعرَف تماماً كيفية فك الترميز)؛ تستخدم مختلف المرمّزات آلياتٍ مختلفةٍ تُنْتِج نتائج متنوعةٍ كثيرةً، لكن يمكن قراءتها جميئاً من المشغلات نفسها. مشروع LAME المفتوح المصدر هو أفضل مرّمز حر، ويمكن القول أنَّه أفضل المرّمزات لجميع معدلات البث إلَّا المنخفضة منها.

صيغة MP3 (التي وُضِعَ معيارها عام 1991) هي صيغةٌ محميةٌ ببراءة اختراع، وهذا هو السبب وراء عدم إمكانية تشغيل نظام لينكس لملفات MP3 مباشرةً (مع بعض الاستثناءات مثل توزيعة Fedora التي أضيف إليها دعم تشغيل ملفات MP3 في 2017). وبشكل عام، تدعم جميع مشغلات الموسيقى المحمولة ملفات MP3، ويمكن تضمين مسارات MP3 ضمن أي حاوية فيديو. ويمكن أن يُشَغِّل Adobe Flash ملفات MP3 لوحدها أو مسارات MP3 الموجودة ضمن حاوية MP4.

بـ Advanced Audio Coding

المعروفة أيضًا باسم «AAC» التي وُضِعَ معيارها عام 1997، Advanced Audio Coding والتي ذاع صيتها عندما اختارتتها Apple صيغةً افتراضيةً لمتجر iTunes. كانت جميع ملفات AAC التي تم «شراؤها» من متجر iTunes مشفرةً بحقوق رقمية مملوكة لشركة Apple (أي DRM) التي كانت تدعى FairPlay؛ أصبحت بعض الأغانيات متاحةً في متجر iTunes كملفات

غير محمية، التي تدعوها «iTunes Plus» لأن ذلك أفضل من تسمية جميع الأشياء AAC الأخرى باسم «iTunes Minus». صيغة AAC محمية ببراءات اختراع؛ وأسعار الترخيص متوفرة على الإنترت.

ضُمِّنت AAC لتوفير صوت بجودة أعلى من MP3 بنفس معدل البث (bitrate)، ويمكن ترميز الصوت بأي معدل بث (صيغة MP3 محدودة بعدد ثابت من معدلات البث، حدّها الأقصى هو 320 كيلوبت/ثانية). يمكن لصيغة AAC ترميز 48 قناة صوتية كحد أقصى، على الرغم من عدم وجود حالة لتطبيق ذلك عملياً. تختلف صيغة AAC عن MP3 في أنها تعرّف أنماطاً (profiles) بطريقة مشابهة لرماد 264 H. ولنفس الأسباب. فهناك «نط» بسيط مُصمّم لكي يشغّل في الوقت الحقيقي على الأجهزة ذات قدرة المعالجة المحدودة، بينما توجد «أنماط» توفر صوتاً أكثر جودةً بنفس معدل البث لكن ذلك على حساب البطء في الترميز وفك الترميز.

جميع منتجات Apple الحالية بما فيها iPod و AppleTV و QuickTime تدعم بعض «أنماط» صيغة AAC في ملفات الصوت المنفصلة وفي مسارات الصوت المدمجة في حاوية MP4. يدعم Adobe Flash جميع «أنماط» صيغة AAC في MP4، وكذلك المشغلات الحرة مثل VLC و MPlayer. أما للترميز، فمكتبة FAAC هي مكتبة شهيرة لترميز AAC؛ وهناك خياراً في وقت البناء (compile-time) في برمجية mencoder و ffmpeg لدعم ترميز AAC.

ج. Vorbis

Vorbis تُسمى عادةً «Ogg Vorbis» (على الرغم من أنَّ هذا خطأ تقبيلاً، لأنَّ «Ogg» هي صيغة حاويات ويمكن لمسارات صوت Vorbis أنْ تضمّن في حاويات أخرى). صيغة Vorbis ليست محمية بأية براءات اختراع ولهذا ستجدها مدعومةً في توزيعات لينكس مباشرةً وفي

جميع الأجهزة المحمولة التي تُشَغِّل نظام Mozilla Firefox 3.5 Rockbox. يدعم Mozilla Firefox 3.5 مسارات صوت في حاوية Ogg، أو فيديو Ogg مع مسار صوتي بصيغة Vorbis، ويمكن أيضًا للهواتف العاملة بنظام Android أن تُشَغِّل ملفات Vorbis الصوتية المستقلة. عادةً ما تُضمن مسارات MKV الصوتية في حاوية Ogg أو WebM، لكن يمكن أيضًا تضمينها في حاوية MP4 أو Vorbis (أو بعد تطبيق بعض الحيل: في حاوية AVI). تدعم صيغة Vorbis الصوتية أي عدد من القنوات الصوتية.

هناك عددٌ من برامجيات ترميز وفك ترميز صوت Vorbis المفتوحة المصدر، بما في ذلك [OggConvert](#) (للترميز)، و [ffmpege](#) (فك ترميز)، و [aoTuV](#) (للترميز)، و [libvorbis](#) (لفك الترميز). وهناك إضافات لبرمجة [QuickTime](#) لنظام Mac OS و [DirectShow](#) لنظام Windows.

د. Opus

صيغة Opus هي صيغة مفتوحة ومتحركة للاستخدام مجانًا، طورتها مؤسسة Xiph.org بدأة الأمر، ثم أصبحت معيارًا من IETF وذلك عام 2012، ومن ذاك الحين انتشرت انتشاراً كبيراً.

صممت هذه الصيغة لترميز الكلام البشري والأصوات العامة في صيغة وحيدة بكفاءة عالية، وهي ملائمة أيضًا لاستعمالها في التواصل في الوقت الحقيقي؛ ويمكن استخدامها في أغلبية حاويات الفيديو الحديثة مثل MP4 أو Ogg أو WebM (وحتى Firefox).

دعم هذه الصيغة يتسع باستمرار، وأغلبية المتصفحات الحديثة تدعمها (بدءًا من Chrome 15، و Safari 15، و Opera 33، و Edge 14، لكنها غير مدعومة في IE).

4. الصيغ التي تعمل في الويب

إن لم يؤلمك رأسك إلى الآن فأنت تبلي بلاءً حسناً. يمكنك أن تستنتج بسهولة أنَّ الفيديو (والصوت) هو موضوعٌ معقدٌ، وما أوردناه سابقاً هو النسخة المختصرة من القصة! أنا متأكد أنك تسأعل عن ارتباط HTML5 بما سبق. حسناً، يوجد في HTML5 عنصر اسمه `<video>` لتضمين مقاطع الفيديو في صفحة ويب، ولا توجد هنالك أيّة قيود على المرماز المستخدم لترميز الفيديو أو الصوت أو حتى صيغة الحاوية التي يمكنك استخدامها لمقاطع الفيديو. يمكن لعنصر `<video>` وحيد أنْ يُشير إلى عدّة ملفات فيديو، وسيختار المتصفح أول ملف فيديو يستطيع تشغيله. كل ما عليك فعله هو معرفة ما هي الحاويات والرموزات التي تدعمها المتصفحات.

هذا هو حال دعم صيغ الفيديو في HTML5 في الوقت الراهن:

- متصفح Mozilla Firefox (3.5 أو ما بعده) يدعم فيديو Theora وصوت Vorbis في حاوية .WebM. ويدعم Firefox 4 صيغة Ogg.
- متصفح Opera 10.5 (وما بعده) يدعم فيديو Theora وصوت Vorbis في حاوية .WebM. يدعم Opera 10.60 صيغة Ogg.
- متصفح Google Chrome 3.0 (وما بعده) يدعم فيديو Theora وصوت Vorbis في حاوية .WebM. يدعم Google Chrome 6.0 صيغة Ogg.
- متصفح Safari على نظامي Mac وويندوز (3.0 وما بعده) سيدعم أي شيء يدعمه QuickTime. نظرياً، هذا يعني أنَّك قد تطلب من مستخدميك تثبيت إضافات خارجية إلى مُشغِّل QuickTime؛ لكن عملياً لن يفعل ذلك إلا القليل. لذلك عليك أن تستعمل الصيغ التي يدعمها QuickTime مباشرةً. وهذه قائمة طويلة إلا أنها لا تحتوي WebM،

- أو Theora، أو حاوية Ogg. لكن QuickTime فيه دعم مدمج لفيديو H.264 أو AAC في حاوية MP4 (نط AAC) وصوت MP4 في حاوية .MP4.
- الهواتف المحمولة مثل iPhone والهواتف التي تعمل بنظام أندرويد تدعم فيديو H.264 وصوت AAC في حاوية MP4.
 - يدعم Adobe Flash (الإصدار 9.0.60.184 وما بعده) فيديو H.264 وصوت AAC في حاوية MP4.
 - يدعم Internet Explorer جميع «أنماط» فيديو H.264 وصوت AAC أو MP3 في حاوية MP4. ويمكنه أيضًا تشغيل فيديو WebM إن ثبت مرماً خارجيًا، الذي لا يكون مثبتًا افتراضيًا في أي إصدار من ويندوز. لا يدعم IE9 المرمادات الخارجية الأخرى (على عكس Safari، الذي سيشغل ما يستطيع QuickTime تشغيله).
 - لا يدعم متصفح Internet Explorer 8 عنصر video في HTML5 إطلاقًا، لكن جميع مستخدمي Internet Explorer تقريبًا يملكون إضافة Adobe Flash. وسأريك لاحقًا في هذا الفصل كيف يمكنك استخدام عنصر video في HTML5 لكن مع خيار احتياطي هو استخدام Flash.
- ربما من الأسهل تلخيص ما سبق في جدول.

Android	iPhone	Opera	Chrome	Safari	Firefox	IE	الترميز/الحاوية
+2.3	.	+10.5	+3.0	*	+3.5	.	Theora+Vorbis+Ogg
+2.0	+3.2	+25	****+3.0	+3.2	+21	+9.0	H.264+AAC+MP4
***2.3	.	+10.6	+6.0	*	+4.0	**+9.0	WebM

* متتصفح Safari سيُشغّل أي شيء يستطيع QuickTime تشغيله، لكن QuickTime لا يأتي مع دعم مسبق إلا لصيغة H.264/AAC/MP4.

** متتصفح Internet Explorer 9 سيُدعم WebM «فقط إذا ثبّت المستخدم المرماز يدوياً»، الذي يعني ضمنياً أنَّ مايكروسوفت لن تضيف المرماز بنفسها.

*** على الرغم من أنَّ متتصفح أندرويد 2.3 يدعم WebM، لكن لا توجد إمكانية لفك الترميز عادةً، وهذا يقلل من عمر البطارية.

**** قد أعلنَ أنَّ Google Chrome سيُسقط الدعم عن H.264 قريباً، وقد تم تبرير ذاك القرار؛ لكنه لم يُطبّق إلى الآن.

هذا هو مسار عملك إذا كنت تزيد توفير مقطع الفيديو بأكبر قدر من التوافقية بين

المتصفحات:

1. أنشئ نسخةً تستعمل .(VP9 + Vorbis) WebM
2. أنشئ نسخةً أخرى تستعمل فيديو H.264 baseline مع صوت AAC بنمط low complexity .MP4 في حاوية
3. اربط ملفَي الفيديو السابقين إلى عنصر <video> وحيد، واترك مجالاً لاستعمال مشغل video يعتمد على تقنية Flash إن لم يدعم المتتصفح العنصر video

5. مشاكل الترخيص مع فيديو H.264

قبل أن نكمل، علي أن أشير أنَّ هنالك تكلفة لترميز مقاطع الفيديو مرتين. حسناً، لا أقصد أنَّ عليك فعلًا ترميزها مرتين، وهذا يعني استهلاك أكبر للوقت ولطاقة المعالجة فيما لو كنت سترمزها مرَّةً واحدة. لكن هنالك تكلفة حقيقة مرتبطة بفيديو H.264: تكلفة الترخيص.

هل تذكرة عندما شرحت H.264 أول مرة، ومررُث مروراً سريعاً على أنَّ ذاك الترميز خاضع لبراءات اختراع ويمكن أخذ ترخيص بالاستخدام من LA؟ قد تبيّن أنَّ ما سبق مهم، ولكن

لتفهم لماذا، سأحيلك إلى مقتطف من مقالة «[The H.264 Licensing Labyrinth](#)»:

تقسام MPEG LA رخصة H.264 إلى رخصتين فرعيتين: واحدة لمصنعي برمجيات الترميز وفك الترميز، وأخرى لموزعي المحتوى. [...]

الرخصة الفرعية المتعلقة بتوزيع المحتوى تقسم بدورها إلى أربعة تصنيفات أساسية، اثنان منها (الاشتراك [subscription]، و «title-by-title» أي لكل مقطع على حدة) مرتبطان فيما إذا كان المستخدم النهائي سيدفع مباشرةً لقاء خدمة بث الفيديو، واثنان منها (البث «الم مجاني» للتلفاز وللإنترنت) مرتبطان بالربح القائم من مصادر أخرى بخلاف المشاهد.

تكلفة الرخصة للبث «الم مجاني» للتلفاز مبنية على خياري دفع. أول خيار هو دفع مبلغ \$2500 لكل مرمز AVC، الذي يعني تسليم مرمز AVC واحد فقط «مستخدم من قبل أو نيابة عن المُرخص له لبث فيديو AVC للمستخدم النهائي»، الذي سيفك الترميز ويشاهد الفيديو. إذا كنت تتتساعل فيما إذا كانت التكلفة مضاعفة هنا، فالجواب هو نعم: دفعت أجراً للرخصة من مصنع المرمز، وسيدفع الطرف الذي سيبث الفيديو أحد خياري الدفع.

ال الخيار الثاني للترخيص هو دفع اشتراك سنوي للبث. [...] وأجرة البث السنوي مُقسّمة إلى شرائح تختلف حسب عدد المشاهدين:

- \$2500 في كل سنة لكل سوق من أسواق البث (broadcast markets) لعدد

الأسر التي تشاهد البث بين 100,000-499,999

- \$5000 في كل سنة لكل سوق من أسواق البث (broadcast markets) لعدد

الأسر التي تشاهد البث بين 500,000-999,999

- \$10,000 في كل سنة لكل سوق من أسواق البث (broadcast markets) لعدد

الأسر التي تشاهد البث الأكبر من 1,000,000

[...] بعد استعراض جميع المشاكل التي تحفّ البث «المجاني» لل்தلفاز، لماذا يجب أن يهتم بها شخص لا يبيث الفيديو؟ كما ذكرت سابقاً، تكلفة الاشتراك تنطبق على أي عملية توصيل للمحتوى، لكن تعريف البث «المجاني» لل்தلفاز يعني أكثر من البث عبر الهوائيات (over-the-air). أضافت MPEG LA أجوراً اشتراكاً لبث الفيديو عبر الإنترنـت التي لن يدفع المستخدم النهائي أيّة تكاليف لمشاهدة المقطع، أي بصيغة أخرى: أيّ بثٍ عامٍ لفيديو سواءً كان عبر الهوائيات أو خدمة «الكبير» أو الأقمار الاصطناعية أو عبر الإنترنـت... سيُخضع إلى أجور الاشتراك.

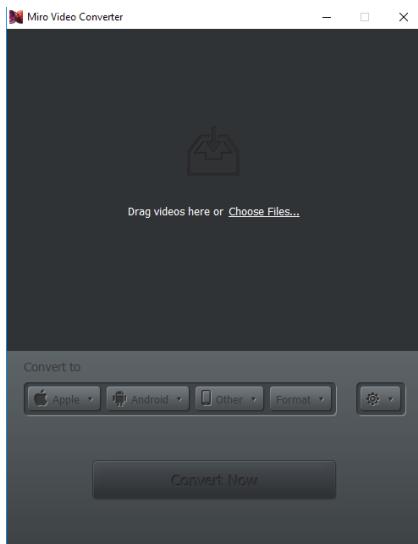
لكن MPEG-LA أعلـنا أنـهم لن يأخذـوا أجورـاً على البث عبر الإنترنـت، لكن هذا لا يعني أنـ مـرـماـز H.264 مـجاـني الاستـخدـام لـجمـيع المستـخدمـين؛ وبـشكل خـاص بـرمـجيـات التـرمـيز (ـكـالـتي تـعـالـج الفـيديـو المرـفـوع عـلـى موقع YouTube) وـبـرمـجيـات فـك التـرمـيز (ـمـثـل البرـمجـية المـوجـودـة فـي مـتصـفح Microsoft Internet Explorer 9)ـ التي ما زـالت تخـضع لأـجـور التـرـخيصـ.

6. ترميز الفيديو باستخدام Miro Video Converter

هنا لك أدوات عديدة لترميز الفيديو، وهناك خيارات كثيرة للترميز تؤثر على جودة الفيديو؛ فإن لم تكن تريده فهم كل شيء متعلق بترميز الفيديو فعليك بهذا القسم.

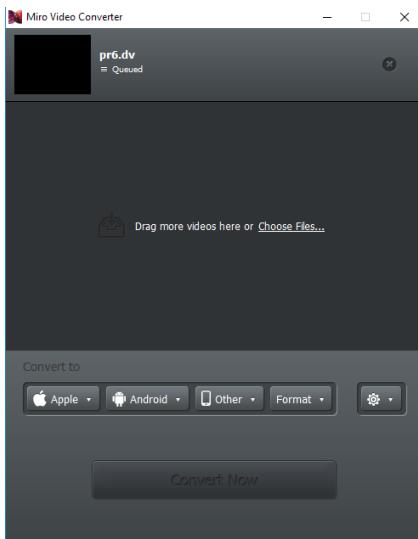
Miro Video Converter هو برنامج مفتوح المصدر مُرخص برخصة GPL لترميز الفيديو بعدة صيغ، يمكنك تنزيله لنظامي Mac OS أو ويندوز. وهو يدعم التحويل إلى جميع الصيغ التي ذكرناها سابقاً في هذا الفصل، لكنه لا يوفر خيارات سوى اختيار ملف الفيديو والصيغة التي سيحول إليها. يمكنه نظرياً قبول أي صيغة من صيغ الفيديو كملف مصدر، بما في ذلك فيديو DV الذي تُنتجه كاميرات الفيديو الرقمية من فئة المستهلكين. يُنتج البرنامج مقاطع فيديو ذات جودة مقبولة؛ ولأن هذا البرنامج لا يوفر لك خياراتٍ كثيرة، فليس عندك حل سوى تغيير البرنامج إن لم تعجبك مقاطع الفيديو الناتجة من البرنامج.

عليك أولاً تشغيل برنامج Miro Video Converter :



الشكل 17: واجهة برنامج Miro Video Converter الرئيسية

. انقر فوق «Choose file» واختر ملف الفيديو المصدرى الذى ترى ترميزه.



الشكل 18: اختيار مقطع فيديو

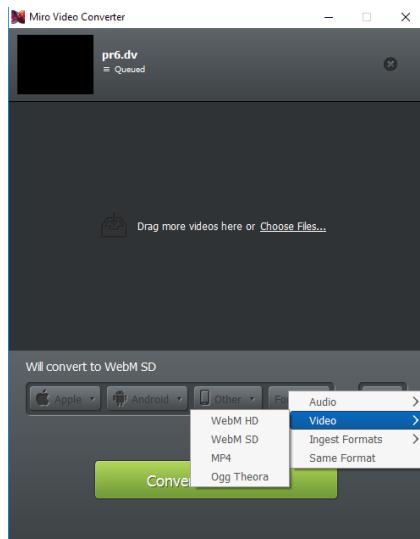
اضغط فوق القائمة المنسدلة «Pick a Device or Video Format» التي فيها قائمة بمختلف الأجهزة والصيغ. لكننا مهتمين بصيغتين منها:

.1. «WebM» هو فيديو WebM (أي فيديو VP8 وصوت **Vorbis** في حاوية WebM).

.2. «iPhone 5» هو فيديو H.264 ذو النمط Baseline وصوت AAC ذو النمط low.

.MP4 في حاوية **complexity**.

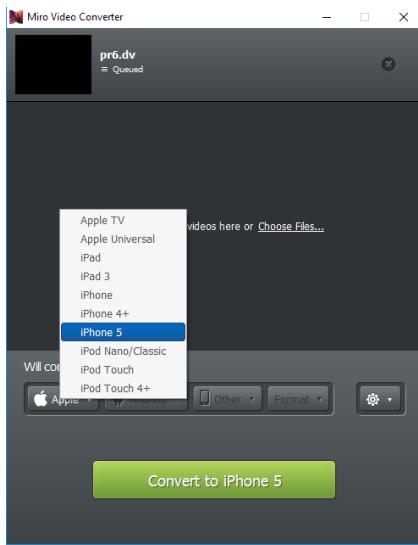
اختر أولاً «WebM» في حاوية:



الشكل 19: اختيار صيغة WebM

اضغط فوق زر «Convert» وسيبدأ برنامج Miro Video Converter بترميز الفيديو مباشرةً. سيسمى الملف الناتج باسم SOURCEFILE.webm وسيحفظ في مجلد الفيديو في نظام التشغيل (وهو في المسار C:\Users\USERNAME\Videos\Miro Video Converter في نظام ويندوز).

في النهاية، عليك ترميز المقطع بمرماز H.264 المتواافق مع iPhone وذلك باختيار «iPhone 5» من قائمة الأجهزة والصيغ.



الشكل 20: اختر iPhone 5 وليس iPhone 4

اضغط على زر «Convert» السحري وانتظر، سيسمى الناتج باسم SOURCENAME.iphone5.mp4 وسيحفظ في مجلد الفيديو في نظام التشغيل.

يجب أن يكون لديك ملفات فيديو بجانب ملفك الأصلي؛ إن أعجبتك جودة الفيديو فانتقل إلى القسم «**أخيراً: الشيفرات**» لترى كيف تجمعها مع بعضها في عنصر `<video>` وحيد الذي يعمل في جميع المتصفحات. أما إذا كنت تريدين تعلم المزيد حول الأدوات الأخرى للترميز فأكمل القراءة.

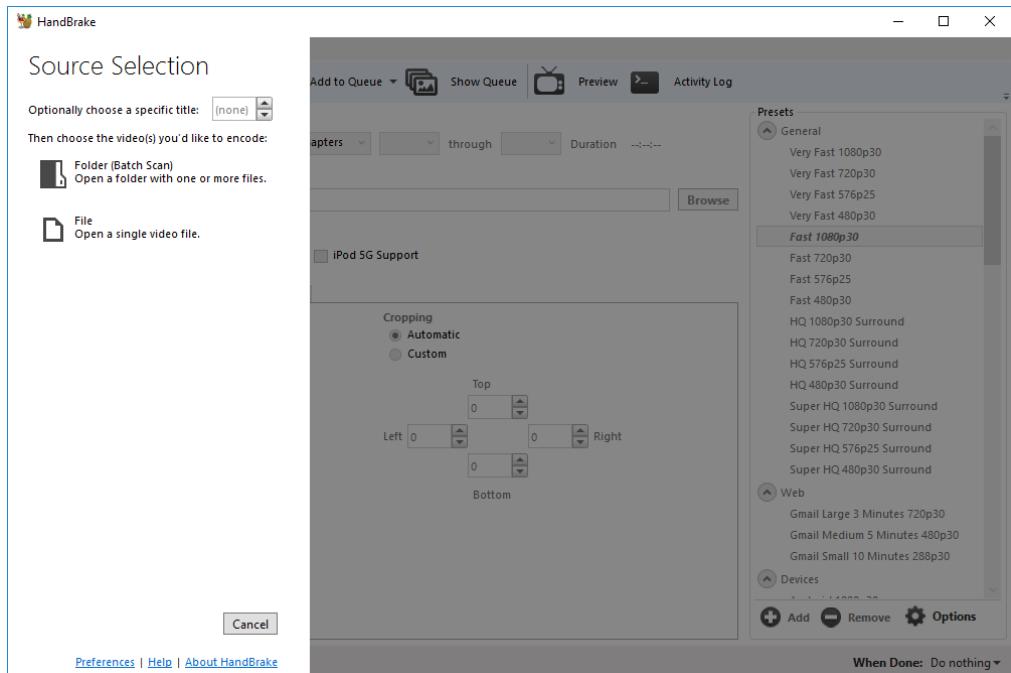
7. ترميز فيديو H.264 باستخدام HandBrake

ملاحظة: سأستخدم المصطلح «فيديو H.264» في هذا القسم اختصاراً للعبارة «فيديو H.264 بنمط Baseline مع صوت AAC بنمط low-complexity في حاوية MPEG-4»، وهذه تجمعية من المركبات+حاوية التي تعمل دون إضافات في Adobe Flash و Safari، وفي iPhone، والهواتف العاملة بنظام أندرويد.

إذا غضضنا النظر عن **المشكل في الترميز** فإن أسهل طريقة لترميز فيديو H.264 هي استخدام HandBrake. HandBrake هو برمجية مفتوحة المصدر مرخصة برخصة GPL لترميز فيديو H.264 (كانت تستطيع ترميز صيغ فيديو أخرى فيما سبق، لكن المطوريين أسقطوا الدعم عن الصيغ الأخرى لتركيز جهودهم على صيغة H.264). تتوفر ملفات تنفيذية لـ Windows و Mac OS X وتوزيعات لينكس الحديثة.

يأتي برنامج HandBrake بنسختين: رسومية، وسطرية (أي تعمل من سطر الأوامر). سأشرح لك الواجهة الرسومية أولاً، ثم سأريك كيف يمكن تحويل الإعدادات إلى نسخة سطر الأوامر.

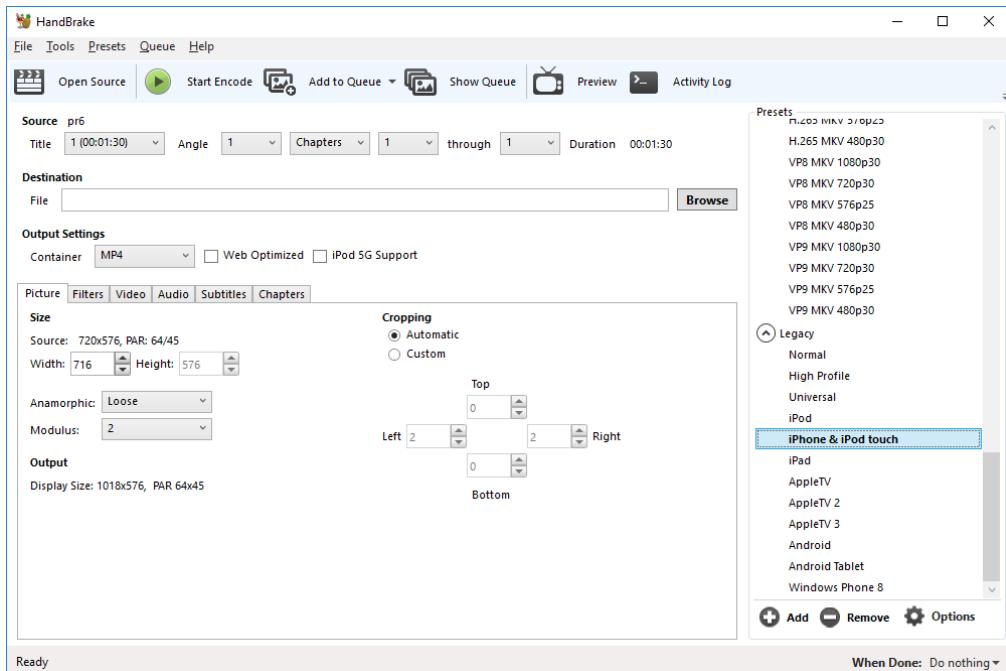
أول شيء ستفعله بعد أن تبدأ بـ HandBrake هو اختيار ملف الفيديو المصدر. اضغط على زر «Source» ثم «Video File» لكي تختار ملفاً. يمكن أن يقبل HandBrake نظرياً أي صيغة من صيغ الفيديو، بما في ذلك فيديو DV الذي تنتجه كاميرات الفيديو الرقمية القديمة من فئة المستهلكين.



الشكل 21: اختر ملف الفيديو الذي تريده ترميزه

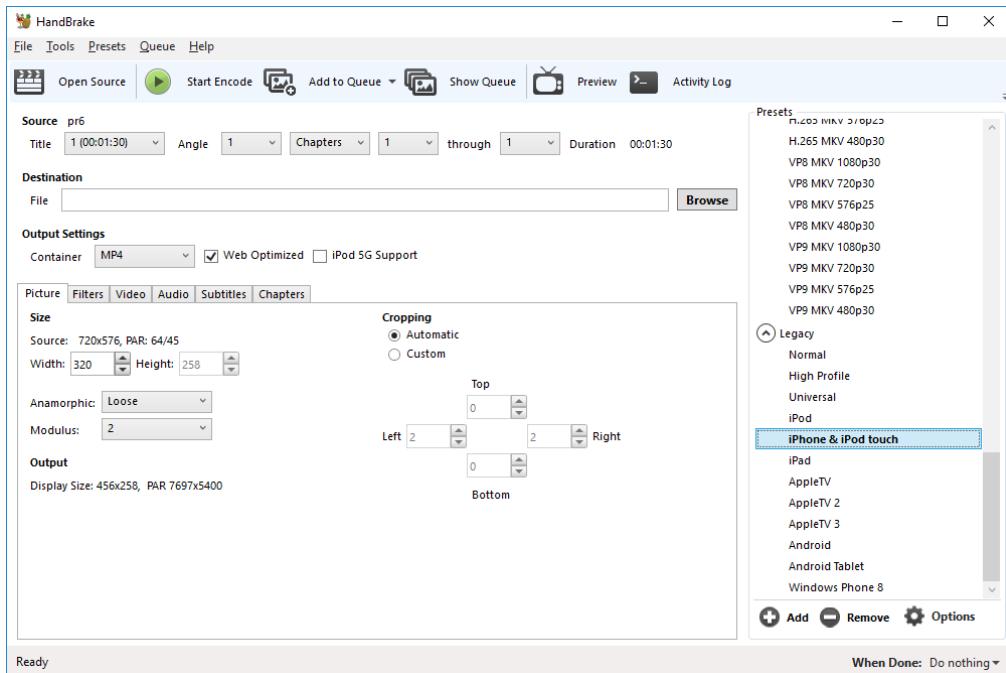
هناك قائمة بأنماط الضبط المسبق (presets) على الجانب الأيمن من البرنامج، اختر منها

نطء «iPhone & iPod Touch» الذي سيضبط أغلبية الخيارات التي ستحتاج لها.



الشكل 22: اختر نمط iPhone

أحد الخيارات المهمة المُعطلة افتراضياً هو الخيار «Web optimized»، تفعيل هذا الخيار سبأدي إلى إعادة ترتيب بعض البيانات الوصفية داخل الفيديو المرمّز لكي تستطيع مشاهدة بداية الفيديو أثناء تنزيل الباقي في الخلفية. أنسحـك - وبشـدة - أن تفعـل هذا الخيار دومـاً، إذ لا يؤثـر على جـودـة أو حـجم المـلـف النـاتـج، لـذـا لـن يـكون هـنـاك أي سـبـب لـعدـم تـفعـيلـه.



الشكل 23: فَقْلُ الْخِيَارِ «Web optimized» دوًماً

يمكنك أن تضبط العرض والارتفاع الأقصى لمقطع الفيديو المفرّمّز في لسان «Picture».

يمكنك أن تضبط أربعة خيارات مهمة في لسان «Video»:

- «Video Codec»: تأكّد أنّ قيمة هذا الخيار هي «H.264 (x264)».

«2-Pass Encoding»: إذا فَقَلتْ هذا الخيار، فسيُشَغِّلُ HandBrake مُرْمَزَ الفيديو

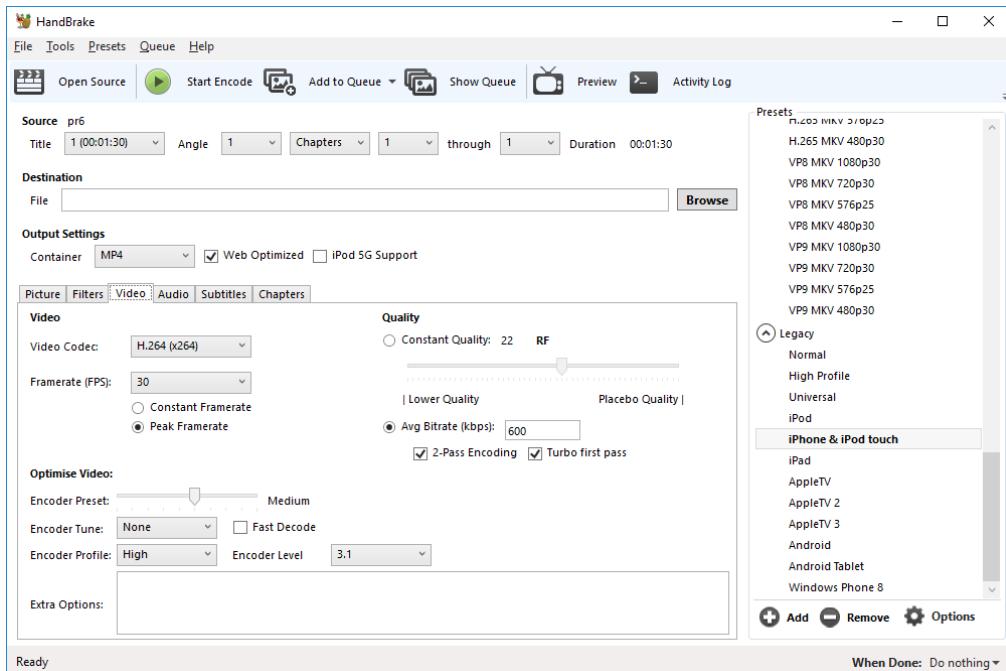
مرتين؛ سيحلّل الفيديو في أول مرة فقط باحثًا عن أشياء مثل تركيبات الألوان، والحركة، ومكان انتهاء المشاهد. وفي المرة الثانية سيرمز الفيديو فعلّيًّا مستخدماً المعلومات التي جمعها أول مرة. وكما توقعت، سيستغرق ذلك ضعف المدة الزمنية، لكنه يؤدي إلى دقة أفضل دون زيادة في مساحة الملف التخزينية. أفعّل هذا الخيار دوًماً

عند ترميز H.264. وعليك فعل ذلك ما لم تكن ترمّز مقاطع الفيديو 24 ساعة في اليوم على مدار الأسبوع.

- «Turbo First Pass»: بعد أن تفعّل «2-Pass Encoding» يمكنك توفير بعض الوقت بتفعيل الخيار «Turbo First Pass»، الذي يقلل حجم العمل المُنجز في أول مرور على الفيديو (أي مرحلة تحليل المقطع)، في حين أنه يقلل الجودة بشكل بسيط جدًا. أفعّل هذا الخيار عادةً، لكن إن كانت جودة الفيديو مهمةً جدًا لك، فعليك إيقاؤه معطلًا.
- «Quality»: هناك عدّة طرق لتحديد «جودة» مقاطع الفيديو: يمكنك تحديد المساحة التخزينية التي تتوقعها للملف الناتج، وسيحاول HandBrake جاهدًا أن يضمن أنَّ الملف الناتج ليس أكبر من المساحة المحددة. أو يمكنك أن تحدّد معدل «البث» (bitrate) الوسطي، الذي هو عدد البتات اللازمة لتخزين ثانية واحدة من الفيديو (ولقد سمي معدل البث «الوسطي» لأن بعض الثوانی تحتاج إلى بتات أكثر من غيرها). أو يمكنك أن تحدّد جودةً ثابتةً على مقاييس من 0 إلى 100%. كلما ازدادت النسبة ستزداد الجودة لكن الملفات ستستهلك مساحة تخزينية أكبر. لا يمكن أن أعطيك جوابًا واحدًا عن ضبط الجودة الذي عليك استخدامه.

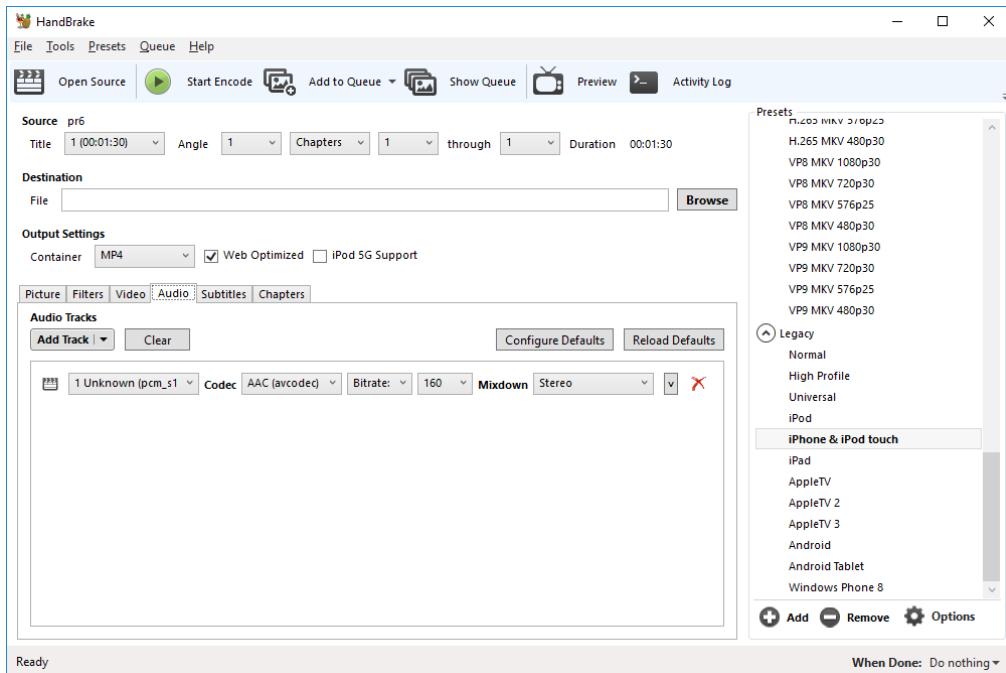
لقد اخترت في هذا المثال معدل بث وسطي هو 600 كيلوبت/ثانية، الذي يعتبر مرتفعًا بالنسبة إلى فيديو بأبعاد 320x240، واختارت أيضًا ترميز مع المرور مرتين، وأن يكون أول مرور

«سريعاً» (turbo).



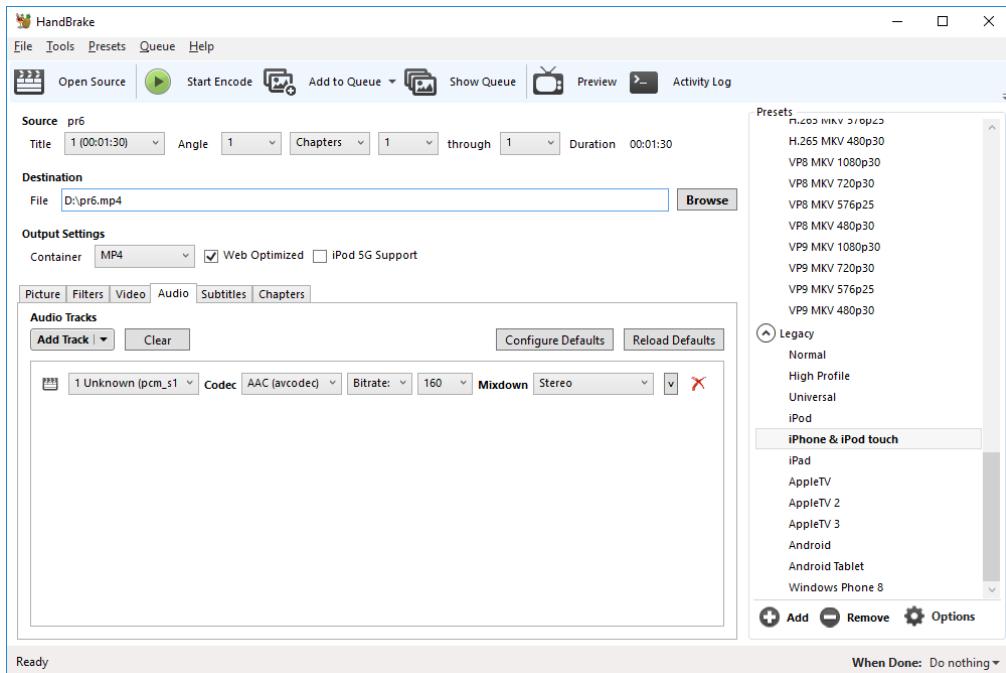
الشكل 24: خيارات جودة الفيديو

لا يفترض عليك تعديل أي شيء في لسان «Audio»؛ أما لو كان لمقطع الفيديو المصدر أكثر من مسار صوتي، فربما تريدين اختيار أي المسارات تريدها في الفيديو الناتج. إذا كان محتوى الفيديو في مجلمه عن شخصٍ يتحدث (على النقيض من الموسيقى أو الأصوات العامة)، فيمكنك تحفيض معدل البث للصوت إلى 96 كيلوبت/ثانية أو ما شابهه، فيما عدا الحالتين السابقتين، تكون القيم الافتراضية المأخوذة من نمط «iPhone» جيدةً عموماً.



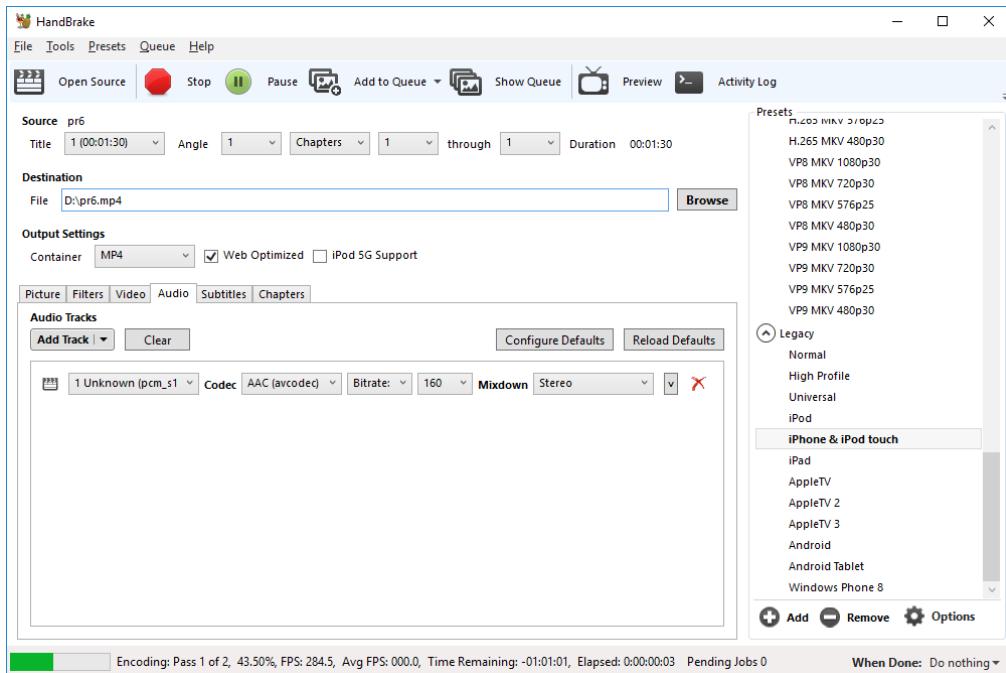
الشكل 25: خيارات جودة الصوت

اضغط الآن على زر «Browse» واختر مجلداً واسم ملف لتحفظ فيه مقطع الفيديو الناتج.



الشكل 26: اختار اسم الملف الناتج

في النهاية اضغط على «Start Encode» لبدء ترميز الفيديو.



الشكل 27: بدء عملية ترميز الفيديو

سيعرض HandBrake بعض الإحصائيات عن تقدم عملية ترميز مقطع الفيديو.

8. ترميز عدّة مقاطع إلى H.264 عبر HandBrake

ملاحظة: كما في القسم السابق، سأستخدم المصطلح «فيديو H.264» في هذا القسم اختصاراً للعبارة «فيديو H.264 بنمط AAC مع صوت Baseline low-complexity». وهذه تجميعية من المرمادات+حاوية التي تعمل دون إضافات في Safari، حاوية MPEG-4، وفي هواتف iPhone، والهواتف العاملة بنظام أندرويد.

يأتي HandBrake بنسخة سطحية (أي تعمل من سطر الأوامر)، نسخة سطر الأوامر من HandBrakeCLI --help توفر عدداً هائلاً من الخيارات (اكتتب HandBrakeCLI لتقرأ المزيد عنها).

لكنني سأرَّ على بعضها:

- .HandBrake --preset "X": حيث «X» هو اسم نمط (preset) من أنماط ضبط iPhone & iPod H.264 المُخصَّص للويب هو "iPhone & iPod Touch". ومن المهم أن تضع الاسم بأكمله ضمن علامتي اقتباس.
- HandBrake --width W: حيث «W» هو عرض الفيديو الذي تريده ترميزه، وسيُعادل الارتفاع تلقائياً ليحافظ على تناسب أبعاد المقطع الأصلي.
- .--vb Q: حيث «Q» هو معدل البث الوسطي (مقاساً بواحدة الكيلوبت/ثانية).
- .--two-pass: تفعيل المرور مرتين (2-pass) أثناء الترميز.
- .--turbo: تسريع المرور الأول عند تفعيل ميزة المرور مرتين أثناء الترميز.
- .--input F: حيث «F» هو مسار ملف الفيديو المصدر.
- .--output E: حيث «E» هو مسار ملف الفيديو الناتج.

هذا مثالٌ عن كيفية استدعاء HandBrake من سطر الأوامر، مع استخدام خيارات ثُطابِقِ

الإعدادات التي اخترناها في الواجهة الرسومية سابقًا.

```
you@localhost$ HandBrakeCLI --preset "iPhone & iPod Touch"
--width 320
--vb 600
--two-pass
--turbo
--input pr6.dv
--output pr6.mp4
```

شرح الأمر السابق من الأعلى إلى الأسفل: تشغيل HandBrake مع نمط «iPhone & iPod Touch»، وإعادة تحجيم المقطع إلى 320x240، وضبط معدل البث الوسطي إلى

600 كيلوبت/ثانية، وتفعيل خيار المروor مرتين مع تسريع أول مرور، وقراءة الملف المصدري .mrt6.mp4 وترميزه وإخراج الملف النهائي إلى .mrt6.dv

9. ترميز فيديو WebM باستخدام FFMPEG

دُعمَت صيغة WebM دعماً كاملاً في 0.6 ffmpeg وما بعده. نفذ الأمر ffmpeg في سطر الأوامر دون وسائل وتحقق أنه مبنيٌّ مع دعم مرماز VP9.

```
you@localhost$ ffmpeg
ffmpeg version 2.8.11-0ubuntu0.16.04.1 Copyright (c) 2000-2017
the FFmpeg developers
built with gcc 5.4.0 (Ubuntu 5.4.0-6ubuntu1~16.04.4) 20160609
  configuration: --enable-gpl --enable-shared --disable-
stripping --disable-decoder=libopenjpeg --disable-decoder=lib
schroedinger --enable-avresample --enable-avisynth --enable-
gnutls --enable-ladspa --enable-libass -
--enable-libbluray --enable-libbbs2b --enable-libcaca --enable-
libcdio --enable-libflite --enable-libf
ontconfig --enable-libfreetype --enable-libfribidi --enable-
libgme --enable-libgsm --enable-libmodpl
ug --enable-libmp3lame --enable-libopenjpeg --enable-libopus
--enable-libpulse --enable-librtmp --en
able-libschroedinger --enable-libshine --enable-libsnapy
--enable-libsoxr --enable-libspeex --enabl
e-libssh --enable-libtheora --enable-libtwolame --enable-
libvorbis --enable-libvpx --enable-libwavpa
ck --enable-libwebp --enable-libx265 --enable-libxvid --enable-
libzvbi --enable-openal --enable-open
gl --enable-x11grab --enable-libdc1394 --enable-libiec61883
--enable-libzmq --enable-frei0r --enable
-libx264 --enable-libopencv
```

إن لم تشاهد الكلمتين السحريتين «--enable-libvorbis» و «--enable-libvpx» فلا يدعم إصدار ffmpeg المثبت لديك WebM (إذا بنتيَّت ffmpeg من المصدر بنفسك، فتحقق إذا

كانت لديك نسختين منه. لا بأس في ذلك، فلن يتعارضا مع بعضهما، لكن عليك استخدام المسار الكامل لنسخة `ffmpeg` التي تدعم مرماز VP9.

سأستخدم ميزة المرور مررتين، سيمسح (scan) المرور الأول ملف الدخل (-i pr6.dv) وسيكتب بعض الإحصائيات إلى السجل، وسأحدّد مرماز الفيديو باستخدام الخيار -c:v، تجاهل الرسالة التي ستخبرك أنَّ ملف الخرج فارغ (Output file is empty, nothing was encoded):

```
you@localhost$ ffmpeg -i pr6.dv -c:v libvpx-vp9 -pass 1 -b:v 1000K -threads 1 -speed 4 -tile-columns 0 -frame-parallel 0 -auto-alt-ref 1 -lag-in-frames 25 -g 9999 -aq-mode 0 -an -s 320x240 -f webm /dev/null
```

غالبية خيارات الأمر `ffmpeg` السابقة ليس لها علاقة بمرماز VP9 أو صيغة WebM. تدعم مكتبة `libvpx` عدداً من الخيارات الخاصة بمرماز VP9 التي يمكنك تمريرها إلى `ffmpeg`، لكنني لم أفعل ذلك في الأمر السابق.

وعند المرور مرةً أخرى على الملف، فسيقرأ `ffmpeg` الإحصائيات التي كتبها أثناء المرور الأول وسيبدأ بترميز الفيديو والصوت، ثم سيكتب ملف ..webm

```
you@localhost$ ffmpeg -i pr6.dv -c:v libvpx-vp9 -pass 2 -b:v 600K -threads 1 -speed 0 -tile-columns 0 -frame-parallel 0 -auto-alt-ref 1 -lag-in-frames 25 -g 9999 -aq-mode 0 -c:a libvorbis -b:a 64k -s 320x240 -f webm out.webm
```

هناك خمسة خيارات مهمة للأمر السابق:

- `-c:v libvpx-vp9`: يُحدّد أننا نريد ترميز المقطع بمرماز VP9.
- `-b:v 600K`: تحديد معدل البث (bitrate)، فإذا أردت ترميز مقطع فيديو بمعدل بث 600 كيلوبت/ثانية، فعليك وضع الحرف K بعد الرقم 600.

- s -: تحديد أبعاد الفيديو الناتج، العرض ثم الارتفاع.
- c:-: تحديد أننا نريد ترميز الصوت بمرماز Vorbis. يمكنك هنا استخدام libvorbis
- .libopus مرماز Opus باستخدام مكتبة

10. وأخيراً، الشيفرات

حسناً، يفترض أنّ هذا الكتاب عن HTML، لذا أين الشيفرات؟

تمتلك HTML5 طريقتين لتضمين الفيديو في صفحة الويب، وكلا الطريقتين تستخدمان العنصر <video>. إذا كان لديك ملف فيديو وحيد، فيمكنك ببساطة إضافة رابط له في خاصية src، وهذا شبيه جدًا بطريقة إدراج صورة بالوسم

```
<video src="pr6.webm"></video>
```

تقنياً، هذا كل ما تحتاج له؛ ولكن كما في عنصر عليك دوماً أن تضيف الخصيّتين height و width في وسوم <video>. يمكن أن تكون الخصيّتان height و width مطابقتين لعرض وارتفاع الفيديو الذي رمّته. لا تقلق إن كان أحد بعدي الفيديو أصغر من القيمتين المحدّدين، لأن المتصفح سيضع الفيديو في منتصف المربع الناتج عن وسم <video>، ولهذا لن يتثنّه المقطع.

```
<video src="pr6.webm" width="320" height="240"></video>
```

افتراضياً، لن يظهر الوسم <video> أي نوع من عناصر التحكم بالتشغيل، يمكنك إنشاء عناصر التحكم الخاصة بك باستخدام HTML و CSS و JavaScript؛ فالعنصر <video> يملك دوال مثل () play و () pause ويمكن القراءة والكتابة إلى خاصيات مثل currentTime،

وهنالك أيضًا خصصيات أخرى يمكن القراءة منها والكتابة عليها مثل `volume` و `muted`: لذا تستطيع أن تبنيواجهة عناصر التحكم كيما تشاء.

إن لم ترغب ببناء واجهة عناصر التحكم يدوياً، فيمكنك أن تخبر المتصفح أن يظهر عناصر التحكم المدمجة فيه؛ وذلك بتضمين الخاصية `controls` في وسم `<video>`.

```
<video src="pr6.webm" width="320" height="240"
controls></video>
```

هناك خاصيتان اختياريتان إضافيتان أريد أن أذكرهما قبل الإكمال: `preload` و `autoplay`: دعني أشرح لك فائدتهما. تُخبر الخاصية `preload` المتصفح أنك تريد البدء بتنزيل ملف الفيديو في أقرب فرصة بعد انتهاء تحميل الصفحة، وهذا شيء منطقي إذا كان الغرض من الصفحة هو مشاهدة مقطع الفيديو؛ لكن في المقابل، إذا كان المقطع ثانويًا ولن يشاهده إلا القلة، فيمكنك أن تضبط الخاصية `preload` إلى `none` كي تخبر المتصفح بذلك لتقليل استهلاك التراسل الشبكي.

هذا مثال عن مقطع فيديو يبدأ بالتنزيل (لكنه لن يُشغّل) بعد انتهاء تنزيل الصفحة:

```
<video src="pr6.webm" width="320" height="240" preload></video>
```

هذا مثال عن مقطع فيديو لن يبدأ بالتنزيل عند انتهاء تحميل الصفحة:

```
<video src="pr6.webm" width="320" height="240"
preload="none"></video>
```

وظيفة الخاصية `autoplay` واضحة من اسمها: تخبر المتصفح أنك تحبّذ تنزيل ملف الفيديو بعد انتهاء تحميل الصفحة، وترغب في أن يبدأ تشغيل المقطع تلقائياً في أقرب وقتٍ

ممكن. بعض الأشخاص يحبون هذا الأمر، وبعضهم يكرهونه؛ لكن دعني أشرح لماذا من المهم وجود هذه الخاصية في HTML5. بعض الأشخاص يريدون بــ بعد تشغيل مقاطع الفيديو في صفحاتهم تلقائياً حتى لو كان ذلك سيزعج زوار موقعهم. إن لم تُعرّف HTML5 طريقةً معياريةً لــ تشغيل مقاطع الفيديو فسيلجأ المطورون إلى استخدام JavaScript لــ ذلك (على سبيل المثال، عبر استدعاء الدالة `window.onload`)؛ وهذا يُصعّب مهمة تعطيل هذه الخاصية على الزوار، إذ يمكن استخدام إضافة إلى المتصفح (أو كتابة واحدة إن لزم الأمر) مهمتها هي تجاهل خاصية `autoplay`، مما يُعطل تشغيل الفيديو التلقائي.

هذا مثال عن مقطع فيديو سيبداً تنزيله وتشغيله في أقرب فرصة ممكنة بعد انتهاء

[تحميل الصفحة](#):

```
<video src="pr6.webm" width="320" height="240"
autoplay></video>
```

هذا هو سكريبت Greasemonkey الذي تستطيع تثبيته على متصفحك ليمنع التشغيل التلقائي لفيديو HTML5؛ إذ يستخدم خاصية `autoplay` في كائن DOM (المكافأة لخاصية `(disable_video_autoplay.user.js)` في شيفرة HTML) لــ تعطيل التشغيل `autoplay`

```
// ==UserScript==
// @name           Disable video autoplay
// @namespace
http://diveintomark.org/projects/greasemonkey/
// @description   Ensures that HTML5 video elements do not
autoplay
// @include        *
// ==/UserScript==

var arVideos = document.getElementsByTagName('video');
```

```
for (var i = arVideos.length - 1; i >= 0; i--) {
    var elmVideo = arVideos[i];
    elmVideo.autoplay = false;
}
```

تمهل قليلاً... إذا كنت تتبع معي منذ بداية هذا الفصل، فأنت تعلم أن لدينا ملقي فيديو بدلاً من واحد، فهناك ملف mp4. الذي أنشأته باستخدام HandBrake، وهناك ملف webm. الذي أنشأته عبر ffmpeg. توفر HTML5 طريقةً لإضافة روابط لجميع الملفات السابقة: العنصر <source>. يمكن أن يحتوي كل عنصر <video> على أكثر من عنصر <source>; وسيقرأ المتصفح قائمةً عناصر source بالترتيب وسيشغّل أول مقطع يستطيع تشغيله.

وهذا يطرح سؤالاً آخر: كيف يعلم المتصفح أي مقطع يستطيع تشغيله؟ حسناً، أسوأ الاحتمالات هي تحميل كل مقطع من تلك المقاطع ومحاولة تشغيله؛ وهذا إهدار لتراسل البيانات. تستطيع أن تُسهل الأمر على المتصفح (وتتوفر قدراً لا بأس به من تراسل البيانات) إذا أخبرت المتصفح معلوماتٍ عن كل مقطع، وذلك باستخدام الخاصية type في عنصر <source>.

```
<video width="320" height="240" controls>
    <source src="pr6.mp4" type='video/mp4; codecs="avc1.42E01E,
    mp4a.40.2'">
    <source src="pr6.webm" type='video/webm; codecs="vp9,
    vorbis"'>
</video>
```

لنُقسِّم الشيفرة السابقة ليسهل فهمها. يُحدّد العنصر <video> عرض وارتفاع مقطع الفيديو، لكنه لا يُحدّد رابطاً لملف الفيديو؛ أما داخل عنصر <video> فهناك ثلاثة عناصر <source>، كل عنصر <source> يُشير إلى ملف فيديو وحيد (باستخدام خاصية src)، وينتقل معلومات أيضاً حول صيغة الفيديو (في خاصية type).

تبعد خاصية `type` معتقدًّا، وهذا صحيح! فهي مجموعة من ثلاثة أقسام من المعلومات: **صيغة الحاوية، ومرماز الفيديو، ومرماز الصوت**. لنبدأ من الأسفل إلى الأعلى: تكون صيغة الحاوية لملفات `webm`. هي `WebM` الممثلة هنا بالعبارة `video/webm` (بكلامٍ تقني، هذا هو نوع `MIME` لملفات `WebM`)، ومرماز الفيديو هو `VP9`، ومرماز الصوت هو `Vorbis`. تبيّن أنَّ خاصية `type` بسيطة، عدا كون شكلها مشوهة قليلاً، لأن القيمة نفسها تتضمن علامات اقتباس، وهذا يعني أنَّ عليك استخدام نوع آخر من علامات الاقتباس لإحاطة القيمة كلها.

```
<source src="pr6.webm" type='video/webm; codecs="vp9, vorbis"'>
```

ملاحظة: لا أدرى إن انتبهت أنَّ برنامج `Miro Video Converter` يرمّز فيديو `WebM` بمرماز `VP8` وليس `VP9`، إذا إن كنت تستعمله للتحويل (ولم تستعمل برمجية `FFMPEG`) فعليك وضع `vp8` بدلاً من `vp9` في الأمثلة المذكورة هنا.

أما فيديو `H.264` فهو أكثر تعقيداً؛ هل تذكر ما قلته عند شرح **فيديو H.264 وصوت AAC** أنها تأتي بأنماط (`profiles`) مختلفة؟ لقد رمّزنا الفيديو بنمط «`baseline`» والصوت بنمط «`low-complexity`». كل هذه المعلومات موجودة في `MPEG-4`. ثم وضعناهما في حاوية `MP4`. خاصية `type`.

```
<source src="pr6.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
```

الفائدة التي نجنيها من وراء مشقة تحديد نوع مقطع الفيديو هي أنَّ المتصفح سيتحقق من خاصية `type` أولاً ليدرى إن كان بإمكانه تشغيل ملف فيديو معين؛ وإن فرر المتصفح أنَّه لا

يستطيع تشغيل فيديو معين، فلن ينزل الملف، ولا أي جزء منه؛ وبهذا لن تستهلك تراسل البيانات، وسيشاهد زوار موقعك الفيديو الذي يريدونه بشكل أسرع.

إذا كنت تتبع التعليمات الواردة سابقاً في هذا الفصل لترميز مقاطع الفيديو، فيمكنك أن تنسخ وتلصق قيم خاصية `type` من المثال السابق، وإلا فعليك أن **تكتبها يدوياً بنفسك**.

أ. أنواع MIME تكشف عن وجهها القبيح

هناك قطع كثيرة لأحجية تشغيل الفيديو على الويب، ولقد ترددت كثيراً في طرح هذا الموضوع، لكنه مهم لأن الضبط الخطا لخادوم الويب سيؤدي إلى مشاكل لا تنتهي عندما تحاول معرفة سبب تشغيل مقاطع الفيديو على جهازك المحلي بينما لا تستطيع ذلك على الخادوم الإنتاجي. إذا واجهتك هذه المشكلة فاعلم أن المسبب الرئيسي لها هو أنواع MIME في أغلبية الحالات.

ذكرت أنواع MIME في فصل «[تاریخ HTML5](#)»، لكن ربما تحطّيَت تلك الفقرة ولم تعرها اهتماماً، هذه خلاصة الموضوع:

يجب أن تُخَدِّم ملفات الفيديو بنوع MIME الملائم لها.

ما هو «نوع MIME الملائم»؟ لقد رأيته سابقاً: هو جزء من قيمة الخاصية `type` في العنصر `<source>`، لكن ضبط الخاصية `type` في شيفرات HTML ليس كافياً، فعليك أيضاً أن تتأكد أن خادوم الويب يُضْمِن نوع MIME المناسب في ترويسة `Content-Type` في `.HTTP`.

إذا كنت تستخدم خادوم ويب أباتشي (Apache) أو أي خادوم آخر مشتَق منه، فيمكنك استخدام تعليمية «[AddType](#)» في ملف `apache2.conf` الذي يُعَدِّل الضبط لكل الموقع، أو في ملف `htaccess`. الذي يُعَدِّل الضبط للمجلد الذي تخزن فيه مقاطع الفيديو (إذا كنت تستخدم

خادوم ويب آخر فانظر إلى توثيق ذاك الخادوم لترى كيف يمكنك ضبط ترويسة Content - في HTTP لأنواع ملفات معينة).

```
AddType video/mp4 .mp4
AddType video/webm .webm
```

أول سطر مما سبق لمقاطع الفيديو في حاوية MPEG-4، والسطر الثاني لمقاطع الفيديو في WebM. اضبطها مرّةً ودعها تعمل عملها؛ لكن إن نسيت ضبطها فلن تعمل جميع مقاطع الفيديو المحدّدة في بعض المتصفحات، حتى لو وضعت نوع MIME في خاصية type في شيفرات HTML.

لتفاصيل أكثر حول ضبط إعدادات خادوم الويب عندك، فأنا أُحيلك إلى هذه المقالة الرائعة (محتوى هذه المقالة ينطبق أيضًا على مقاطع [Configuring servers for Ogg media](#)) (WebM و MP4).

11. ماذا عن متصفح IE؟

يدعم متصفح Internet Explorer عنصر `<video>` في HTML5، ويدعم فيديو H.264 وصوت AAC في حاوية MPEG-4، مثل متصفح Safari وهاتف iPhone. لكن ماذا عن الإصدارات القديمة من Internet Explorer؟! مثل IE8 وما دونه؟! أغلبية الأشخاص الذين يستخدمون Internet Explorer لديهم إضافة Adobe Flash مثبتةً على جهازهم.

الإصدارات الحديثة من Adobe Flash (بدءاً من الإصدار 9.0.60.184) تدعم فيديو AAC في حاوية MPEG-4، مثل متصفح Safari وiPhone. فبمجرد [ترميزك لمقاطع](#)

الفيديو برماز H.264 لمتصفح Safari، ستستطيع تشغيله في مشغل فيديو يعتمد على تقنية HTML5، وذلك في حال اكتشفت أن أحد زوارك لا يملك متصفحًا متواافقًا مع فيديو Flash. **FlowPlayer** هو مشغل فيديو مبني على تقنية Flash مفتوح المصدر ومرخص برخصة GPL (توفر رخص تجارية له). لا يعُرف مشغل FlowPlayer أي شيء عن عنصر `<video>`، ولن يحُول وسم `<video>` بشكلٍ سحري إلى كائن Flash، لكن HTML5 مصممة تصميمًا جيدًا لكي تعامل مع هذه الحالات، لأنك تستطيع تضمين عنصر `<object>` ضمن عنصر `<video>` وستحمل العنصر `<object>` الموجود ضمنه بدلاً عنه؛ الذي سيشغل مقطع الفيديو باستخدام `source`: أما المتصفحات التي تدعم الفيديو في HTML5 العنصر `<video>` وستتجاهل العنصر `<object>` تستطيع تشغيله، ثم ستتجاهل العنصر `<object>` تماماً.

آخر جزء من مفتاح حل الأحجية هو أن HTML5 تقول بوجوب تجاهل جميع العناصر (ما عدا عناصر `<source>`) التي تكون «أبناء» (children) للعنصر `<video>` تماماً، وهذا يسمح لك باستخدام عنصر `<video>` في المتصفحات الحديثة مع توفير حل للمتصفحات القديمة عبر تقنية Flash دون الحاجة إلى كتابة سكريبتات JavaScript معقدة. يمكنك قراءة المزيد عن هذه التقنية هنا: [Video For Everybody](#).

12. مثال متكامل

هذا تطبيق للتقنيات التي تعلمناها سابقاً في هذا الفصل؛ عدّلْت قليلاً على شيفرة «Video For Everybody» لتضمين فيديو WebM، ولقد رمّزَ الفيديو المصدري إلى ثلاثة صيغ باستخدام هذه الأوامر:

```
## H.264/AAC/MP4
you@localhost$ HandBrakeCLI --preset "iPhone & iPod Touch" --vb
200 --width 320 --two-pass --turbo --optimize --input pr6.dv
--output pr6.mp4

## VP9/Vorbis/WebM
you@localhost$ ffmpeg -i pr6.dv -c:v libvpx-vp9 -pass 1 -b:v
1000K -threads 1 -speed 4 -tile-columns 0 -frame-parallel 0
-auto-alt-ref 1 -lag-in-frames 25 -g 9999 -aq-mode 0 -an -s
320x240 -f webm /dev/null
you@localhost$ ffmpeg -i pr6.dv -c:v libvpx-vp9 -pass 2 -b:v
600K -threads 1 -speed 0 -tile-columns 0 -frame-parallel 0
-auto-alt-ref 1 -lag-in-frames 25 -g 9999 -aq-mode 0 -c:a
libvorbis -b:a 64k -s 320x240 -f webm out.webm
```

سنسخدم العنصر `<video>` في الشيفرة النهائية، مع وجود عنصر `<object>` لمشغل

إن لم يدعم المتصفح العنصر `<video>` Flash

```
<video id="movie" width="320" height="240" preload controls>
  <source src="pr6.webm" type='video/webm; codecs="vp9,
vorbis"' />
  <source src="pr6.mp4" />
  <object width="320" height="240" type="application/x-
shockwave-flash"
    data="flowplayer-3.2.1.swf">
    <param name="movie" value="flowplayer-3.2.1.swf" />
    <param name="allowfullscreen" value="true" />
    <param name="flashvars" value='config={"clip": {"url":
"http://wearehugh.com/dih5/pr6.mp4", "autoPlay":false,
"autoBuffering":true}}' />
    <p>Download video as <a href="pr6.mp4">MP4</a>, <a
href="pr6.webm">WebM</a>.</p>
  </object>
</video>
```

ستتمكن من تشغيل مقطع الفيديو السابق على أي متصفح أو جهاز بدمج شيفرات HTML5 مع مشغل Flash كما في المثال السابق.

13. مصادر إضافية

- HTML5: The <video> element
- Video for Everybody
- Configuring servers for Ogg media
- Encoding with the x264 codec
- Video type parameters
- Everything you need to know about HTML5 audio and video
- Internet Explorer 9 Guide for Developers: HTML5 video and audio elements
- Encoding Video

عناصر تحكم بتشغيل الفيديو مُعدّة مسبقاً:

- VideoJS
- MediaElement.js
- Kaltura HTML5 Video & Media JavaScript Library



ادخل سوق العمل ونفذ المشاريع باحترافية
عبر أكبر منصة عمل حر بالعالم العربي

ابدأ الآن كمستقل

٧

تحديد الموضع الجغرافي

تحديد الموقع الجغرافي هو آلية معرفة مكان وجودك في هذا العالم ومشاركة تلك المعلومات (اختيارياً) مع الأشخاص الذين تثق بهم؛ وهنالك أكثر من طريقة لمعرفة أين أنت: إما باستخدام عنوان IP الخاص بك، وإما عبر اتصال الشبكة اللاسلكية، أو عبر برج التغطية الخلوية الذي يتصل به هاتفك، أو عبر شريحة GPS التي تحسب مكانك نسبةً إلى خطوط الطول (longitude) والعرض (latitude) من المعلومات التي ترسلها الأقمار الاصطناعية من السماء.

س: يبدو لي أنَّ تحديد الموقع الجغرافي مرعبٌ فقد أفزعني كثيراً؛ هل أستطيع تعطيله؟

ج: يقلل المستخدمون من انتهاك الخصوصية عندما نتحدث عن مشاركة موقعك الفيزيائي مع خادوم ويب بعيد. تقول واجهة تحديد الموقع الجغرافي البرمجية (API) «أنَّ على المتصفحات عدم إرسال معلومات الموقع الحالي إلى مواقع الويب دون إذن المستخدم»؛ أي بعبارة أخرى، السماح بمشاركة الموقع الجغرافي منوطُ بك، فإن شئت سمحت بمشاركته، وإنْ فلأ.

1. واجهة تحديد الموقع الجغرافي البرمجية

تسمح **واجهة تحديد الموقع الجغرافي البرمجية** (geolocation API) لك بمشاركة موقعك الحالي مع موقع الويب الموثوقة. ستتوفر إحداثيات الطول والعرض للصفحات عبر JavaScript، التي بدورها سترسل تلك المعلومات إلى خادوم الويب البعيد الذي سيجري عمليات عجيبة متعلقة بالموقع الجغرافي مثل العثور على شركة محلية أو إظهار موقعك على خريطة. كما ستري في الجدول الآتي، تدعم واجهة تحديد الموقع الجغرافي منأغلبية متصفحات الحاسوب والهواتف المحمولة؛ وهنالك دعمٌ لبعض المتصفحات القديمة باستخدام مكتبات خارجية، التي سنأتي على ذكرها لاحقاً في هذا الفصل.

Android	iPhone	Opera	Chrome	Safari	Firefox	IE
+2.0	+3.0	+10.6	+5.0	+5.0	+3.5	+9.0

بجانب دعم واجهة تحديد الموقع الجغرافي القياسية، هنالك عدد من الواجهات البرمجية الخاصة بهواتف معينة، التي سنتغطي شرحها لاحقاً في هذا الفصل.

2. أريني الشيفرة

تتمحور واجهة تحديد الموقع الجغرافي البرمجية حول خاصية جديدة في كائن

`navigator.geolocation`: `navigator`

أبسط استخدام لواجهة تحديد الموقع الجغرافي كما يلي:

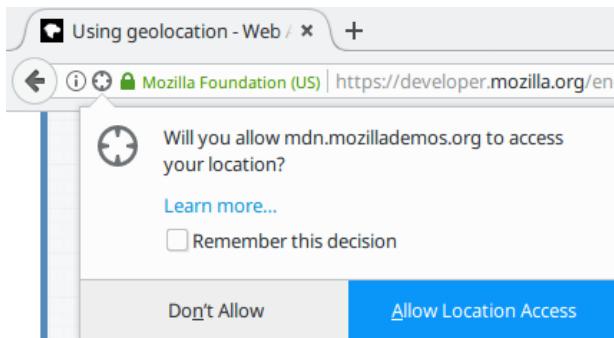
```
function get_location() {
    navigator.geolocation.getCurrentPosition(show_map);
}
```

لكن ليست في الشيفرة السابقة أية آليات للتحقق من دعم المتصفح أو التعامل مع الأخطاء أو خياراتٍ أخرى؛ ويجب عادةً أن يتضمن تطبيق الويب اثنين مما سبق.

يمكنك استخدام `Modernizr` للتحقق من دعم واجهة تحديد الموقع الجغرافي البرمجية:

```
function get_location() {
    if (Modernizr.geolocation) {
        navigator.geolocation.getCurrentPosition(show_map);
    } else {
        // ?Gears لا يوجد دعم لتحديد الموقع؛ ربما تجرب استخدام
    }
}
```

ما الذي تريده فعله إن لم يكن تحديد الموضع مدعوماً من وظيفة `getCurrentPosition()`? مكتبة Gears لاحقاً، لكنني سأتحدث عما يحدث عند استدعاء `getCurrentPosition()`. كما ذكرت سابقاً في بداية هذا الفصل، لن يُجبر المتصفح على إعطاء موقعك الفيزيائي إلى الخادم البعيد، ولكن تختلف طريقة فعل ذلك من متصفح إلى آخر؛ فسيؤدي استدعاء الدالة `getCurrentPosition()` في متصفح Firefox إلى إظهار «شريط معلومات» في أعلى نافذة المتصفح، الذي يبدو كالتالي:



الشكل 28: شريط المعلومات الذي يُظهره متصفح Firefox عند محاولة الوصول إلى الموقع الفيزيائي.

هناك الكثير من الأشياء المضمنة في ذاك الشريط؛ أنت كمستخدم للمتصفح:

- سيتم إخبارك أنّ موقع ويب يحاول معرفة موقعك الفيزيائي
- سيتم إخبارك ما هو موقع الويب الذي يحاول معرفة موقعك الفيزيائي
- ستتمكن من الذهاب إلى صفحة المساعدة «Location-Aware Browsing» التي تشرح لك ما الذي يجري (النسخة المختصرة من القصة هي أنّ Google ستتوفر الموقع وستخزن بياناتك بما يتواافق مع اتفاقية الخصوصية لخدمة تحديد الموضع الخاصة بها)

- ستستطيع أن تسمح بمشاركة موقعك الجغرافي
- ستتمكن من عدم السماح بمشاركة موقعك الجغرافي
- ستتمكن من إخبار المتصفح أن يتذكر اختيارك (سواءً كنت تزيد مشاركة موقعك الجغرافي أم لا) لكي لا تشاهد شريط المعلومات مرةً أخرى لكن هناك المزيد! هذا الشريط:
- لا يمنعك من التبديل بين ألسنة (tabs) المتصفح أو بين نوافذه
- خاص بالصفحة وسيختفي بمجرد تبديلك إلى لسان أو نافذة أخرى ثم سيظهر مرةً ثانية عند عودتك إلى اللسان الأصلي.
- لا يمكن لموقع الويب تجاوزه أو الالتفاف عليه
- يمنع مشاركة الموقع الجغرافي مع خادوم الويب أثناء انتظاره لجوابك (إن كنت تزيد المشاركة أم لا)

لقد رأيت شيفرة JavaScript التي تؤدي إلى إظهار شريط المعلومات السابق، وفيها دالة تؤدي إلى استدعاء دالةٍ أخرى (التي سميتها `show_map`)، وستُنفذ الدالة `getCurrentPosition()` مباشرةً لكن هذا لا يعني أنك تستطيع الوصول إلى بيانات موقع المستخدم؛ فأول مرة تضمن فيها حصولك على تلك البيانات هي داخل الدالة التي سُتستدعي؛ التي تبدو كالتالي:

```
function show_map(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    // لظهور خريطة أو شيئاً آخر مفيداً
}
```

تأتي الدالة السابقة مع معامل (parameter) وحيد، الذي هو كائن له خاصيتان: `coords` و `timestamp`. خاصية `timestamp` بسيطة، فهي الوقت والتاريخ الذي حُسب فيه الموقع (لا يمكنك توقع متى سيُحسب الموقع لأن ذلك يحدث بشكلٍ غير متزامن). فربما سيأخذ المستخدم بعض الوقت لقراءة شريط المعلومات والموافقة على مشاركة الموقع الجغرافي، وقد تستغرق الأجهزة ذات شريحة GPS بعض الوقت للاتصال بأقمار GPS الاصطناعية،... إلخ.). أما الكائن `coords` فلديه خاصيات مثل `longitude` و `latitude` الواضح من اسمها أنها إحداثيات الموقع الفيزيائي للمستخدم.

يوضح هذا الجدول خاصيات الكائن `:position`

الخاصية	النوع	ملاحظات
<code>coords.latitude</code>	<code>double</code>	عدد عشري
<code>coords.longitude</code>	<code>double</code>	عدد عشري
<code>coords.altitude</code>	<code>null</code> أو <code>double</code>	متراً فوق المجسم المرجعي للأرض (الإهليج)
<code>coords.accuracy</code>	<code>double</code>	بواحدة المتر
<code>coords.altitudeAccuracy</code>	<code>null</code> أو <code>double</code>	بواحدة المتر
<code>coords.heading</code>	<code>null</code> أو <code>double</code>	درجات باتجاه عقارب الساعة من الشمال الحقيقي
<code>coords.speed</code>	<code>null</code> أو <code>double</code>	بواحدة متر/ثانية
<code>timestamp</code>	<code>DOMTimeStamp</code>	مثل الكائن (<code>Date()</code>)

من المضمون وجود ثلاث خاصيات من الخواص السابقة (`coords.latitude`) و (`coords.accuracy`) أما البقية فيمكن أن يعيدها القيمة `null` اعتماداً على قدرات جهازك وعلى قدرات خادوم تحديد الموقع الذي تتعامل معه. ستحسب الخواص `speed` و `heading` اعتماداً على موقع المستخدم السابق إذا كان متوفراً.

3. التعامل مع الأخطاء

موضوع تحديد الموقع الجغرافي معقدٌ بعض الشيء، ويحتمل أن تأخذ الأمور منحى خطأ. ذكرت سابقاً ناحية «موافقة المستخدم»؛ فلو أراد تطبيق الويب الحصول على الموقع الفيزيائي للمستخدم لكن المستخدم لم يرغب في إعطائه للتطبيق، فلن تحصل عليه وسيربح المستخدم دائمًا.

كيف يبدو التعامل مع الأخطاء في الشيفرات؟ عليك أن تمرر وسيطًا ثانئاً إلى الدالة `getCurrentPosition()` هو الدالة التي ستشتدعى عند حدوث خطأ.

```
navigator.geolocation.getCurrentPosition(  
    show_map, handle_error)
```

إن حدث أي خطأً فستشتدعى الدالة المحددة مع تمرير الكائن `PositionError` إليها. يوضح الجدول الآتي خواص الكائن `:PositionError`

الخاصية	النوع	ملاحظات
<code>code</code>	<code>short</code>	قيمة عددية
<code>message</code>	<code>DOMString</code>	الرسالة الموجودة في هذه الخاصية ليست موجهة للمستخدم النهائي.

ستكون قيمة الخاصية `code` واحدة من القيم الآتية:

- (1): إذا ضغط المستخدم على زر «Don't Share» أو منع الوصول إلى موقعه بطريقةٍ أو بأخرى.
- (2): إذا توقفت الشبكة عن العمل أو في حال عدم التمكن من الوصول إلى الأقمار الصناعية.
- (3): إذا كانت الشبكة تعمل لكنها تأخذ وقتاً طويلاً لحساب موقع المستخدم الفيزيائي؛ لكن بكم يُقدر «الوقت الطويل»؟ سأريك كيفية تعريف تلك القيمة في القسم التالي.

```
function handle_error(err) {
  if (err.code == 1) {
    لم يسمح المستخدم بالحصول على الموقع الجغرافي! //
  }
}
```

س: هل تعمل واجهة تحديد الموقع الجغرافي في المحطة الفضائية الدولية، أو على القمر، أو على الكواكب الأخرى؟

ج: تقول مواصفة تحديد الموضع أنّ «نظام الإحداثيات الجغرافية المستخدم في هذا الصدد هو نظام الإحداثيات الجيوديزية العالمي [WGS84]. بقية أنظمة الإحداثيات غير مدعومة». تدور المحطة الفضائية الدولية حول الأرض، لذلك يمكن وصف موقع رواد الفضاء على المحطة بإحداثيات طول وعرض وارتفاع عن الأرض، لكن يتمحور نظام الإحداثيات الجيوديزية العالمي حول الأرض، ولا يمكن استخدامه لتعيين موقع على القمر أو بقية الكواكب.

4. الخيارات المتاحة أمامك

تدعى بعض الهواتف المحمولة - مثل iPhone وهواتف أندرويد - طرفيتين لتحديد مكانك.

تحسب أول طريقة موقعك بناءً على قربك من عدّة أبراج تغطية مملوكة من شركة الاتصالات الخلوية المشترك فيها؛ هذه الطريقة سريعة ولا تحتاج إلى شريحة GPS فيزيائية، لكنها تعطي فكرة عامة عن موقعك، ويمكن تعريف الدقة بناءً على عدد أبراج التغطية في موقعك، فقد تكون على مستوى المباني السكنية، أو على نطاق كيلومتر من مكانك.

تستعمل الطريقة الثانية شريحة GPS في هاتفك لتبادل المعلومات مع أقمار الأصطناعية التي تدور حول الأرض. يمكن تحديد موقعك عبر GPS بدقة كبيرة (عدّة أمتار)، لكن من سلبيات هذه الطريقة هي الاستهلاك الكبير للطاقة من شريحة GPS، لذا ستعطل الهاتف المحمولة هذه الشريحة إلى أن يتم الاحتياج إليها؛ وهذا يعني أنّ هنالك تأخير عند تشغيل الشريحة ريثما يُهيأ الاتصال مع أقمار GPS الأصطناعية. إذا سبق لك واستخدمت Google Maps على هاتف ذكي مثل iPhone أو هواتف أندرويد، فستشاهد تطبيقاً لكلا الطريقتين السابقتين: ستشاهد أولاً دائرة كبيرة تحدد موقعك تقربياً (وذلك بالبحث عن أقرب برج تغطية)، ثم دائرة أصغر (بحساب الموقع بناءً على عدّة أبراج تغطية)، ثم نقطة وحيدة دقيقة (التي هي إحداثيات موقعك الفيزيائي بناءً على المعلومات الآتية من أقمار GPS الأصطناعية).

السبب وراء ذكري لهذه المعلومات هي أنّك لا تحتاج دوماً إلى دقة عالية، فإن كنت تبحث عن قائمة بدور عرض الأفلام التي بالجوار، فلا تلزمك إلا معرفة الموقع العام للمستخدم؛ إذ لا توجد دور عرض سينمائية كثيرة، حتى في المدن المزدحمة، وستذكر - على أية حال - أكثر من دار عرض. أما على الكفة الأخرى، إذا كان تطبيقك يعطي توجيهات لسائق السيارة في الوقت

ال حقيقي، فيجب أن تعرف موقع المستخدم الفيزيائي بدقة لكي تستطيع أن تقول «انعطف نحو اليمين بعد 20 متراً» (أو ما شابه ذلك).

يمكن تمرير وسيط `getCurrentPosition()` ثالث اختياري إلى دالة `(argument)`؛ وهناك ثلاثة خاصيات يمكنك ضبطها في كائن `PositionOptions` وكل تلك الخاصيات اختيارية، إذ تستطيع أن تضبطها جميّاً أو أن لا تضبط أياً منها، وهي مبيّنة في الجدول الآتي:

الخاصية	النوع	القيمة الافتراضية	ملاحظات
<code>enableHighAccuracy</code>	<code>Boolean</code>	<code>false</code>	قد تسبّب القيمة <code>true</code> بطأً
<code>timeout</code>	<code>long</code>	(لا توجد قيمة افتراضية)	القيمة بواحدة الميلي ثانية
<code>maximumAge</code>	<code>long</code>	0	القيمة بواحدة الميلي ثانية

وظيفة خاصية `enableHighAccuracy` واضحة من اسمها، إن كانت قيمتها `true` وكان يدعمها الجهاز ووافق المستخدم على مشاركة موقعه الفيزيائي مع التطبيق، فسيحاول الجهاز توفير الموقع الفيزيائي بدقة. هناك أذونات منفصلة للتحديد الدقيق وغير الدقيق للموقع الجغرافي في هواتف iPhone وأندرويد؛ لهذا من الممكن أن يفشل استدعاء الدالة `getCurrentPosition()` مع ضبط الخاصية `enableHighAccuracy:true`، لكن قد ينجح استدعاؤها مع ضبط الخاصية `.enableHighAccuracy:false`.

تُحدّد خاصية `timeout` كم ملي ثانية على تطبيق الويب أن يتضرر الحصول على الموقع الفيزيائي، لكن لا يبدأ المؤقت إلا بعد موافقة المستخدم على إعطاء إحداثيات موقعك الفيزيائي؛ فليس الغرض من هذه الخاصية قياس سرعة ردة فعل المستخدم، وإنما قياس سرعة الشبكة.

تسمح خاصية `maximumAge` للجهاز بأن يُجيب مباشرةً بنسخة محفوظة من الإحداثيات. على سبيل المثال، لنقل أنك استدعيت `getCurrentPosition()` لأول مرة، ثم وافق المستخدم على إعطاء موقعه الجغرافي وانتهت عملية حساب الموقع الفيزيائي في الساعة AM 10:00 تماماً؛ وبعد دقيقة واحدة (أي 10:01 AM) استدعيت الدالة `getCurrentPosition()` مرة أخرى مع ضبط خاصية `maximumAge` إلى 75000.

```
navigator.geolocation.getCurrentPosition(  
    success_callback, error_callback, {maximumAge: 75000});
```

أنت تقول أنه لا يهمك موقع المستخدم في لحظة استدعاء الدالة، وإنما ستقبل بمعرفة أين كان المستخدم منذ 75 ثانية مضت (75 ملي ثانية)؛ لكن الجهاز يعرف أين كان المستخدم منذ 60 ثانية (60000 ملي ثانية)، لأنه حسب موقعه في أول مرة استدعيت فيها الدالة `getCurrentPosition()`؛ وبالتالي لن يحتاج الجهاز إلى إعادة حساب موقع المستخدم الحالي، إذ سيستخدم المعلومات نفسها التي أرسلها أول مرة: أي إحداثيات الطول والعرض نفسها، والدقة نفسها، وبصمة الوقت (أي timestamp 10:00 AM) نفسها.

عليك أن تفكّر في مدى الدقة المطلوبة قبل أن تسأل المستخدم عن موقعه، وتضبط الخاصية `enableHighAccuracy` وفقاً لذلك. وإذا كنت تريد معرفة موقع المستخدم أكثر من مرة، فعليك التفكير في العمر الأقصى للمعلومات التي تستطيع الاستفادة منها، وتضبط الخاصية `maximumAge` وفقاً لذلك. أما إن أردت معرفة موقع المستخدم بشكل دائم، فلن تكون الدالة `getCurrentPosition()` مناسبة لك، وعليك حينها استخدام `.watchPosition()`.

تملك دالة `watchPosition()` نفس بنيّة الدالة `getCurrentPosition()`، إذ يمكنها استدعاء دالتين، إحداهما ضرورية وتستخدم إن نجحت عملية الحصول على الموقع، وأخرى

اختيارية غرضها هو التعامل مع الأخطاء؛ ويمكنها -أي الدالة- أن تقبل تمرير كائن `PositionOptions` اختيارياً الذي يملك الخصائص ذاتها التي تعلمتها منذ قليل. الاختلاف في أن الدالة التي ستشتدعى ستثتفّذ في كل مرة يتغير فيها موقع المستخدم، ولن تحتاج إلى محاولة الحصول على الموقع يدوياً، فسيحذّد جهازك الفاصل الزمني الأمثل لتحديث الموقع وسيستدعي الدالة عند كل تغيير لموقع المستخدم. يمكنك الاستفادة من هذا لتحديث موضع مؤشر على الخريطة، أو توفير تعليمات عن المكان الذي عليك زيارته لاحقاً، أو أي شيء تريده.

تعيد الدالة `(watchPosition)` رقمًا عليك تخزينه في مكان ما، فلو أردت إيقاف عملية مراقبة تغيير موقع المستخدم، فعليك استدعاء الدالة `(clearWatch)` ممّرّزاً إليها ذاك الرقم، وسيتوقف الجهاز عن إرسال تحديثات الموقع إلى دالتك. يعمل ما سبق تماماً كالدالتين `(setInterval)` و `(clearInterval)` في JavaScript إن استخدمنهما من قبل.

5. ماذا عن متصفح IE؟

لم يكن يدعم متصفح Internet Explorer قبل الإصدار التاسع واجهة تحديد الموقع البرمجية من W3C التي شرحتها من قبل، لكن لا تقنط! Gears هي إضافة مفتوحة المصدر للمتصفحات من Google التي تعمل على ويندوز ولينكس و Mac OS X والهواتف العاملة بنظامي Windows Phone وأندرويد (يجدر بالذكر أنَّ Google أعلنت أنها توقفت عن دعم هذه المكتبة). إذ مهمتها هي توفير ميزات للمتصفحات القديمة، وإحدى الميزات التي توفرها Gears هي واجهة برمجية لتحديد الموقع إلا أنَّها ليس مماثلة لواجهة W3C البرمجية، لكنها تخدم الغرض نفسه. لما كثُر تحدث هنا عن المنصات القديمة، فمن الجدير بالذكر أنَّ عدداً من أنظمة الهواتف المحمولة القديمة لها واجهات برمجية خاصة بها لتحديد الموقع، إذ توفر هواتف BlackBerry

Gears و Nokia و palm و OMTP BONDI واجهات برمجية خاصة بها؛ التي تختلف بالطبع عن Gears والتي تختلف بدورها عن W3C.

6. مكتبة geo.js

geo.js هي مكتبة JavaScript مفتوحة المصدر مرخصة برخصة MIT التي تسهل التعامل مع واجهة W3C البرمجية وواجهة Gears البرمجية والواجهات البرمجية التي توفرها أنظمة الهاتف القديمة. عليك أن تضمن عنصري <script> في أسفل صفتاك لكي تستخدمها (يمكنك أن تضع العنصرين في أي مكان في الصفحة، لكن وضعهما في عنصر <head> سيبطيء من تحميل الصفحة، فلا تفعل ذلك).

أول سكريبت هو gears_init.js الذي يهيئ إضافة Gears إن وجدت، أما السكريبت الثاني

.geo.js فهو

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Dive Into HTML5</title>
</head>
<body>
  ...
  <script src="gears_init.js"></script>
  <script src="geo.js"></script>
</body>
</html>
```

ستتمكن من تحديد الموقع الآن بغض النظر عن الواجهة البرمجية المدعومة في المتصفح.

```
if (geo_position_js.init()) {
  geo_position_js.getCurrentPosition(geo_success, geo_error);
```

```
{
}
```

لُقِّبَ مَا سبق ونشر كل سطْرٍ على حدة. ستحتاج أولاً إلى استدعاء دالة `(init)`, التي تُعِيد `true` إن وجد دعم لإحدى واجهات تحديد الموضع البرمجية.

```
if (geo_position_js.init()) {
```

لن تعثر الدالة `(init)` على الموضع الجغرافي، وإنما تتحقق من أنَّ الوصول إلى الموضع ممكِّن. وعليك استدعاء الدالة `(getCurrentPosition)` للحصول على الموضع.

```
geo_position_js.getCurrentPosition(geo_success, geo_error);
```

ستؤدي الدالة `(getCurrentPosition)` إلى جعل المتصفح يطلب من المستخدم إذنه للحصول على موقعه الفيزيائي ومشاركته. إن كان الوصول إلى الموضع الجغرافي موفراً من إضافة `Gears` فسيظهر مربع حوار يسألك إن كنت تثق بموقع الويب لكي يحصل على موقعك. أما إذا كان المتصفح لا يدعم تحديد الموضع داخلياً، فسيظهر مربع حوار ذو شكلٍ مختلف. على سبيل المثال، يدعم `Firefox` واجهة تحديد الموضع الجغرافي البرمجية داخلياً، فلو حاولت الحصول على الموضع الجغرافي فيه، فسيظهر شريط معلومات في أعلى الصفحة يسأل المستخدم إن كان يريد مشاركة موقعه الجغرافي مع موقع الويب.

تأخذ الدالة `(getCurrentPosition)` وسيطين هما الدالتان اللتان ستُستدعيان، فإن نجحت الدالة `(getCurrentPosition)` في الحصول على موقع المستخدم -أي أنَّه أعطى إذناً للوصول إلى الموضع الجغرافي، واستطاعت واجهة تحديد الموضع الجغرافي البرمجية تعين الموقع- فستُستدعى الدالة الأولى، التي تكون في هذه المثال `geo_success`.

```
geo_position_js.getCurrentPosition(geo_success, geo_error);
```

تأخذ تلك الدالة وسيطًا وحيداً يحتوي على معلومات الموقع الفيزيائي:

```
function geo_success(p) {
    alert("Found you at latitude " + p.coords.latitude +
        ", longitude " + p.coords.longitude);
}
```

وإن لم تستطع الدالة `getCurrentPosition()` معرفة موقع المستخدم -إما أن يكون المستخدم قد رفض إعطاء الإذن، وإما لفشل تعين الموقع من الواجهة البرمجية بسبب من الأسباب- فستستدعي الدالة الثانية، التي تكون في مثالنا `.geo_error`.

```
geo_position_js.getCurrentPosition(geo_success, geo_error);
```

لا تأخذ تلك الدالة أية وسائل:

```
function geo_error() {
    alert("Could not find you!");
}
```

لاتدعم مكتبة `geo.js` الدالة `watchPosition()`، لذا عليك أن تطلب الدالة `getCurrentPosition()` بشكل متواصل إن أردت الحصول على تحديث فوري لموقع المستخدم.

7. مثال متكم

سأشرح لك مثلاً يستخدم مكتبة geo.js للوصول إلى موقعك وعرض خريطة لما حولك. ستستدعى الدالة (`geo_position_js.init()`) عند تحميل الصفحة لمعرفة فيما إذا كانت توفر ميزة تحديد الموقع الجغرافي بأي شكلٍ من الأشكال التي تدعمها js.geo. فإن كانت مدعومةً فسيظهر رابط يمكن للمستخدم النقر عليه لإظهار موقعه الجغرافي؛ يستدعي هذه الرابط الدالة (`lookup_location()`) الظاهرة هنا:

```
function lookup_location() {
    geo_position_js.getCurrentPosition(show_map, show_map_error);
}
```

إذا أعطى المستخدم موافقته على تحديد الموقع، وكانت الخدمة الخلفية (backend service) قادرةً على تحديد الموقع، فستستدعي مكتبة geo.js أول دالة التي هي (`show_map()`) مع وسيط وحيد الذي هو `loc` إذ يمثل خاصية `coords` التي تحتوي إحداثيات الطول والعرض ودقة القياس (لا يستخدم هذا المثال معلومات دقة القياس). تستعمل بقية الدالة (`show_map()`) وجهاً Google Maps البرمجية لإظهار الخريطة.

```
function show_map(loc) {
    $("#geo-wrapper").css({'width': '320px', 'height': '350px'});
    var map = new GMap2(document.getElementById("geo-wrapper"));
    var center = new GLatLng(loc.coords.latitude,
    loc.coords.longitude);
    map.setCenter(center, 14);
    map.addControl(new GSmallMapControl());
    map.addControl(new GMapTypeControl());
    map.addOverlay(new GMarker(center, {draggable: false, title:
    "You are here (more or less)"}));
}
```

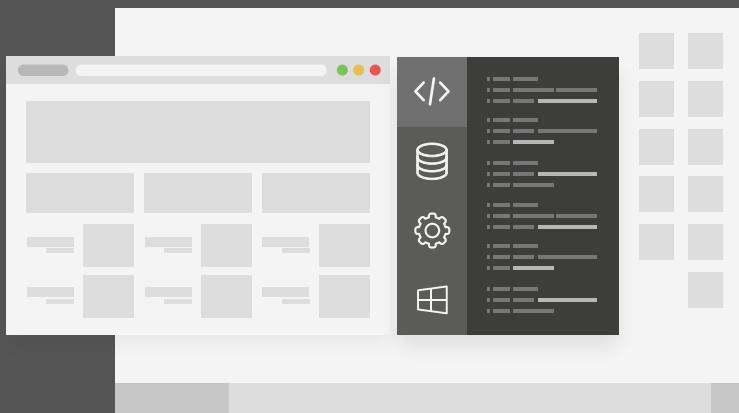
أما لو لم تستطع geo.js تحديد موقعك، فستستدعي الدالة ()
.show_map_error()

```
function show_map_error() {  
    $("#live-geolocation").html('Unable to determine your  
location.');//  
}
```

8. مصادر إضافية

- W3C geolocation API
- مكتبة geo.js
- Internet Explorer 9 Guide for Developers: Geolocation

دورة علوم الحاسوب



مميزات الدورة

- ✓ شهادة معتمدة من أكاديمية حسوب
- ✓ إرشادات من المدربين على مدار الساعة
- ✓ من الصفر دون الحاجة لخبرة مسبقة
- ✓ بناء معرض أعمال قوي بمشاريع حقيقة
- ✓ وصول مدى الحياة لمحتويات الدورة
- ✓ تدريبات مستمرة على الدورة مجاناً

اشترك الآن



التخزين المحلي

V

كانت البرمجيات المكتبية تتفوق على تطبيقات الويب بإمكانية تخزين المعلومات محلياً تخزيئاً دائماً؛ إذ يوفر نظام التشغيل عادةً طبقةً وسيطةً لتخزين وقراءة بيانات خاصة بالتطبيق مثل الإعدادات وحالة التشغيل، وقد تُخزن هذه القيم في سجل النظام (registry) أو ملفات ini أو ملفات XML أو في مكان آخر وفقاً للتقاليد المتبعة في نظام التشغيل؛ أما لو احتاج التطبيق المكتبي إلى تخزين محلٍّ أكثر تعقيداً من مجرد تخزين البيانات على شكل «المفتاح/القيمة»، فيمكنك أن تضمن قاعدة البيانات الخاصة بتطبيقك، أو أن تبتكر صيغة ملفات للتخزين، أو غيره ذلك من الحلول.

لكن على مرّ التاريخ، لم تملك تطبيقات الويب هذا الامتياز، وعلى الرغم من ابتكار الكعكات (Cookies) في بدايات الويب لكن كان الغرض منها هو التخزين المحلي لكميّة قليلةٍ من البيانات، إلا أنَّ هنالك ثلاثة أسباب تمنعنا من استخدامها لهذا الغرض:

- ستضمن الكعكات في كل طلبية HTTP، مما يؤدي إلى حدوث بطء في تطبيق الويب بسبب نقل البيانات نفسها مراراً وتكراراً دون داعٍ
- ستضمن الكعكات في كل طلبية HTTP، وهذا يعني إرسال البيانات دون تشفير عبر الإنترنت (إلا إذا كان يُخدم تطبيق الويب عندك عبر طبقة SSL)
- المساحة التخزينية للكعكات محدودة إلى حوالي 4 كيلوبايت من البيانات، وهي كافية لإبطاء تطبيقك (انظر أعلاه)، لكنها ليست كافية لتخزين شيءٍ مفيدٍ ما نحتاج له حقاً هو:
- مساحة تخزينية كبيرة
- موجودة على جهاز العميل

- يمكن أن تبقى حتى بعد تحديث الصفحة
- لن تُثقل طوال الوقت إلى الخادوم

جميع المحاولات -قبل HTML5- لتحقيق ما سبق كانت غير مرضية لمختلف الأسباب.

1. لمحة تاريخية عن التخزين المحلي قبل HTML5

لم يكن هناك سوى متصفح Internet Explorer في بدايات الويب، أو على الأقل هذا ما حاولت مايكروسوف特 إيهام العالم به، ولتحقيق هذه الغاية، وجزء من **الحرب الكبرى الأولى للمتصفحات**، ابتكرت مايكروسوف特 ميزات كثيرة ووضعتها في متصفحها Internet Explorer-Internet Explorer، الذي أنهى تلك الحرب. واحدة من تلك الميزات تُسمى «**DHTML Behaviors**» وكان أحد خصائصها يُدعى **userData**.

تسمح ميزة **userData** لصفحات الويب أن تخزن 64 كيلوبايت كحد أقصى لكل نطاق (domain)، وذلك عبر هيكلية تعتمد على XML (أما النطاقات الموثوقة، مثل موقع إنترانت [intranet]، فتستطيع تخزين 10 أضعاف الكمية؛ وكانت 640 كيلوبايت في ذاك الوقت أكثر من كافية). لم يوفر IE أي مربع حوار لأخذ إذن المستخدم، ولم تكن هناك إمكانية لزيادة كمية البيانات التي يمكن تخزينها محلياً.

في عام 2002، أضافت شركة Adobe ميزة في Flash 6 التي اكتسبت الاسم «**Local Shared Objects**»، لكن هذه الميزة كانت معروفةً ضمن بيئة Flash بالاسم **cookies**، باختصار، تسمح هذه الميزة لكائنات Flash أن تخزن 100 كيلوبايت من البيانات كحد أقصى لكل نطاق. طور Brad Neuberg نموذجاً أولياً لجسرٍ يربط تقنية Flash بلغة JavaScript وأسماه **AMASS** (اختصار للعبارة AJAX Massive Storage System)، لكنه كان محدوداً بسبب بعض

المشكلات في تصميم صيغة Flash. لكن في 2006، ومع مجيء `ExternalInterface` في Local 8، أصبح من الممكن بسهولة وسرعة الوصول إلى الكائنات المشتركة المخزنة محليًا (AMASS أو اختصارًا JavaScript Shared Objects LSOs) من Brad كتابة `dojox.storage` تحت الاسم `Dojo Toolkit` ودمجها مع كل نطاق 100 كيلوبايت من التخزين المحلي «مجانًا»، وستطلب موافقة المستخدم عند كل زيادة في تخزين البيانات (1 ميغابايت، 10 ميغابايت، وهكذا).

في عام 2007، أصدرت Google إضافة `Gears`، التي هي إضافة مفتوحة المصدر للمتصفحات غرضها هو توفير إمكانيات إضافية إليها (تحديثنا سابقاً عن `Gears` في سياق `Tوفيرواجهة برمجية لتحديد الموقع الجغرافي لمتصفح IE`). توفر `Gears` واجهة برمجية (API) للوصول إلى قاعدة بيانات SQL مدمجة فيها مبنية على محرك قواعد البيانات `SQLite`. يمكن لإضافة `Gears` تخزين كمية غير محدودة من البيانات لكل نطاق في جداول قاعدة بيانات SQL بعد أخذ إذن المستخدم.

في تلك الأثناء، أكمل Brad Neuberg `dojox.storage` وآخرون مشوارهم في تطوير لتوفير واجهة موحدة لمختلف الإضافات، وبحلول 2009 أصبح بمقدور `dojox.storage` أن تكتشف دعم (وتتوفر واجهة موحدة) لبرمجية Adobe Flash و `Gears` و `Adobe AIR` والنموذج الأولي من التخزين المحلي في HTML5 الذي كان مطابقاً في الإصدارات القديمة من Firefox.

عندما تنظر إلى تلك الحلول، فستكتشف أن جميعها كان خاصاً بمتصفح معين أو كان يتبع إضافة خارجية. وعلى الرغم من الجهود البطولية لتوحيد تلك الاختلافات (`dojox.storage`)

إلا أن تلك الحلول تملك واجهات برمجية مختلفة جزئياً عن بعضها، ولكل منها حدود قصوى لمقدار المساحة التخزينية المتوفرة، ولكل منها تجربة مستخدم مختلفة. هذه هي المشكلة التي أتت HTML5 لحلها: توفير واجهة برمجية معيارية، ومطبقة في جميع المتصفحات، دون الحاجة إلى استخدام إضافات خارجية.

2. تمهيد إلى التخزين المحلي في HTML5

ما أشير إليه على أنه «التخزين المحلي في HTML5» (HTML5 Storage) هو مواصفة باسم «Web Storage» التي كانت جزءاً من معيار HTML5، لكنها انقسمت وأصبحت معياراً مستقلاً لأسباب ليست مهمة. بعض الشركات المسؤولة عن المتصفحات تطلق عليها الاسم «التخزين المحلي» (Local Storage) أو «تخزين DOM» (DOM Storage). ازداد تعقيد موضوع التسميات خصوصاً بعد ظهور عدد من المعايير الجديدة التي سأناقشها في [نهاية هذا الفصل](#).

إذًا، ما هو التخزين المحلي في HTML؟ بشكل مبسط: هو طريقة تتمكن صفحات الويب من خلالها تخزين البيانات على شكل «المفتاح/القيمة» محلياً داخل متصفح الويب في حاسوب العميل. ومثل الكعكات، ستبقى البيانات موجودةً حتى بعد إغلاقك للسان الصفحة في المتصفح أو إغلاق المتصفح. لكن على عكس الكعكات، لن تُرسل البيانات تلقائياً إلى خادوم الويب البعيد؛ وعلى التقىض من كل المحاولات السابقة لتوفير ميزة التخزين المحلي، هذه الميزة موجودةً داخلياً في متصفحات الويب، لذلك ستكون متاحةً للاستخدام حتى لو لم تتوفر إضافات خارجيةٌ للمتصفح.

ما هي المتصفحات التي تدعمها؟ حسناً، التخزين المحلي في HTML5 مدعومٌ من أغلبية المتصفحات، وحتى القديمة منها.

Android	iPhone	Opera	Chrome	Safari	Firefox	IE
+2.0	+2.0	+10.5	+4.0	+4.0	+3.5	+8.0

تستطيع الوصول إلى التخزين المحلي في HTML5 في شيفرات JavaScript عبر الكائن `window.localStorage`; لكن قبل أن تستخدمها، عليك أن تكتشف دعم المتصفح لها.

```
function supports_html5_storage() {
    try {
        return 'localStorage' in window && window['localStorage'] !
== null;
    } catch (e) {
        return false;
    }
}
```

لكن بدلاً من كتابة الدالة السابقة يدوياً، يمكنك استخدام Modernizr لاكتشاف دعم التخزين المحلي في HTML5.

```
if (Modernizr.localstorage) {
    // window.localStorage متوفرة!
} else {
    // لا يوجد دعم للتخزين المحلي: (
    // ربما تجرب dojo.storage أو مكتبة أخرى
}
```

3. استخدام التخزين المحلي في HTML5

يعتمد التخزين المحلي في أساسه على تخزين البيانات على شكل «مفتاح/قيمة». أي أنك تُخزن البيانات في مفتاح له اسم مُميّز، ثم تستطيع الحصول على تلك البيانات مرةً أخرى باستخدام المفتاح نفسه. ذاك المفتاح هو سلسلة نصية، ويمكن أن تكون البيانات المخزنة من أي نوع تدعمه لغة JavaScript بما في ذلك السلاسل النصية والقيم المنطقية (`true` و `false`) أو الأعداد الصحيحة أو الأعداد العشرية؛ لكن في الواقع، سُتخزن البيانات كسلسلة نصية، وهذا يعني أنّه لو لم تكن القيمة المخزنة نصيةً فستحتاج إلى استعمال دوال مثل `parseInt()` أو `parseFloat()` لكي تحول البيانات التي حصلت عليها إلى نوع البيانات الذي تريده.

```
interface Storage {
    getter any getItem(in DOMString key);
    setter creator void.setItem(in DOMString key, in any data);
};
```

سيؤدي استدعاء الدالة `setItem()` مع تمرير مفتاح موجود مسبقاً إلى إعادة الكتابة فوق القيمة السابقة دون إشعار. وسيؤدي استدعاء الدالة `getItem()` مع تمرير مفتاح غير موجود إلى إعادة `null` بدلًا من رمي استثناء `(throw an exception)`.
وكما هو الحال مع بقية الكائنات في JavaScript، يمكنك أن تُعامل الكائن `localStorage` على أنه مصفوفة ترابطية `(associative array)`. فبدلاً من استخدام الدالتين `getItem()` و `setItem()`، تستطيع بكل بساطة أن تستعمل الأقواس المربعة (التي تستعملها للوصول إلى عناصر المصفوفات). يمكن على سبيل المثال أن تُعيد كتابة هذه الشيفرة:

```
var foo = localStorage.getItem("bar");
// ...
localStorage.setItem("bar", foo);
```

بهذا الشكل لاستخدام الأقواس المربعة:

```
var foo = localStorage["bar"];
// ...
localStorage["bar"] = foo;
```

هناك دوالٌ أخرى لحذف قيمة مرتبطة بمفتاح معين، ولحذف كل ما هو مُخزَّنٌ محليًّا (وهذا يعني حذف كل المفاتيح والقيم معًا).

```
interface Storage {
    deleter void removeItem(in DOMString key);
    void clear();
};
```

لن يؤدي استدعاء الدالة `removeItem()` مع تمرير مفتاح غير موجود إلى فعل أي شيء. وأخيرًا، هناك خاصية للحصول على العدد الكلي للقيم المُخزَّنة محليًّا، دالة للحصول على اسم كل مفتاح عبر تمرير فهرسه المكاني `(index)`.

```
interface Storage {
    readonly attribute unsigned long length;
    getter DOMString key(in unsigned long index);
};
```

لو استدعيت الدالة `(key)` مع فهرس لا يقع بين `0 - (length-1)` فستُعيد الدالة `null`.

١. تتبع التغييرات في مساحة التخزين المحلي

إذا أردت أن تتبع التغييرات في مساحة التخزين (storage area) برمجيًا، فعليك أن تستعمل الحدث storage، الذي يُفْعَل (fired) في الكائن العام window في كل مرة تُستدعي فيها الدالة () clear() أوsetItem() أو removeItem()، وتجري تلك الدالة تغييرًا ما. فعلى سبيل المثال، لو أعددت ضبط قيمة موجودة مسبقاً وكانت القيمة الجديدة مساويةً للقيمة القديمة، أو استدعيت الدالة clear() لكن لم تكن هنالك أية قيم في مساحة التخزين، فلن يُفْعَل الحدث storage، لعدم تغيير شيء في مساحة التخزين.

الحدث storage مدعومٌ في كل متصفح يدعم الكائن localStorage، وهذا يتضمن الحدث storage المدعوم من W3C لمراقبة الأحداث، لكن Internet Explorer 8 لا يدعم الدالة المعيارية addEventListener (لكتها أُضيفت في نهاية المطاف في IE 9)؛ ولهذا، إذا أردت مراقبة تفعيل الحدث storage فعليك أن تكتشف ما هي آلية الأحداث التي يدعمها المتصفح أولاً (إذا فعلت هذا من قبل مع الأحداث الأخرى، فيمكنك تخطي هذه الفقرة والانتقال إلى آخر القسم. مراقبة الحدث storage مماثلة تماماً لعملية مراقبة الأحداث الأخرى التي سبق وأن رأيتها؛ وإذا كنت تفضل استخدام jQuery أو مكتبة JavaScript أخرى لتسجيل دوال مراقبة الأحداث، فتستطيع فعل ذلك مع الحدث storage أيضًا).

```
if (window.addEventListener) {
    window.addEventListener("storage", handle_storage, false);
} else {
    window.attachEvent("onstorage", handle_storage);
};
```

ستُستدعي الدالة `handle_storage` مع تمرير كائن من نوع `StorageEvent`, عدا في متصفح Internet Explorer إذ يُخَرِّن الكائن في `window.event`.

```
function handle_storage(e) {
  if (!e) { e = window.event; }
}
```

سيكون المتغير `e` -عند هذه النقطة- كائناً من نوع `StorageEvent`, الذي لديه الخصائص

المبيّنة في الجدول الآتي:

الخاصية	النوع	الشرح
key	سلسلة نصية	مفتاح القيمة التي أضيفت أو حُذِفت أو عُدِّلت
oldValue	أي نوع	القيمة (التي كُتِبَ فوقها), أو <code>null</code> إذا أضيف عنصرٌ جديد
newValue	أي نوع	القيمة الجديدة, أو <code>null</code> إن حُذِفَ عنصرٌ ما
*url	سلسلة نصية	الصفحة التي تحتوي على الدالة التي أجرت هذا التغيير

* ملاحظة: كان اسم الخاصية `url` الأصلي هو `uri`, وذلك لأن بعض المتصفحات امتلكت هذه الخاصية قبل تغيير موصفة التخزين المحلي. لأكبر قدر من التوافقية، عليك أن تتحقق من وجود الخاصية `url`. فإن لم تكن موجوداً فتحقق من قيمة الخاصية `uri`.

لا يمكن إلغاء الأحداث في الحدث `storage`. فلا توجد طريقةٌ من داخل الدالة `handle_storage` تستطيع إيقاف تغييرٍ ما من الحدوث. بكل بساطة، هذه طريقةٌ لكي يخبرك المتصفح: «هذا ما حصل لتوه، لا يمكنك فعل أي شيء تجاهه؛ كل ما تستطيع فعله هو إخبارك ما الذي حدث».

ب. المحدوديات في المتصفحات الحالية

في حديثي عن [الملحة التاريخية عن محاولات تخزين البيانات محلياً](#) باستخدام إضافات خارجية، حرصت على ذكر محدوديات كل تقنية من تلك التقنيات، مثل محدودية المساحة التخزинية. لكنني لم أذكر شيئاً عن محدوديات التخزين المحلي في HTML5 المعياري. سأعطيك الأجرة أولاً ثم سأشرحها. الأجرة هي -بترتيبها حسب الأهمية:-

«5 ميغابايت»، و «QUOTA_EXCEEDED_ERR» و «لا».

«5 ميغابايت» هي المساحة التي يُسمح لكل موقع بالحصول عليها افتراضياً، وهذه القيمة متساوية -على غير العادة- بين المتصفحات، على الرغم من أنها مذكورة في مواصفة التخزين المحلي في HTML5 على أنها «اقتراح». أبقى في ذهنك أنك تُخزن سلسل نصية، ولا تُخزن البيانات بصيغتها الأصلية، فلو كنت تُخزن الكثير من الأعداد الصحيحة (integers) أو العشرية (floats)، فسيكون الفرق في طريقة تمثيل البيانات مؤثراً، إذ يُخزن كل رقم من عدد عشرى كحرف (character)، وليس بالتمثيل التقليدي لعدد عشري.

«QUOTA_EXCEEDED_ERR» هو الاستثناء (exception) الذي سيُرمى (thrown) عندما تتجاوز حد 5 ميغابايت. أما «لا» فهو الجواب على السؤال البدهي الذي سيخطر ببالك: «هل يمكنني طلب المزيد من المساحة التخزинية من المستخدم؟» على حد الآن، لا تدعم أيّة متصفحات أي آلية يتمكن خلالها مطورو الويب من طلب المزيد من المساحة التخزинية. لكن بعض المتصفحات (مثلاً Firefox أو Opera) تسمح للمستخدم أن يتحكم بالحد الأقصى للتخزين المحلي، لكن هذا منوط بالمستخدم تماماً، ولا يمكنك -كمطور ويب- الاعتماد على ذلك لبناء تطبيقك.

4. مثال عملي عن استخدام التخزين المحلي

لأخذ مثلاً عملياً عن التخزين المحلي في HTML. هل تتذكر لعبة الضامة التي بنيناها في الفصل الذي يتحدث عن **canvas**? هناك مشكلة صغيرة مع هذه اللعبة: ستخسر تقدّمك في اللعبة عندما تغلق نافذة المتصفح. لكن باستخدام التخزين في HTML5، سنستطيع حفظ التقدّم محلياً داخل المتصفح. هذا **مثال حي** للعبة بعد التعديل. حرك بعض القطع، ثمأغلق لسان الصفحة (أو المتصفح)، ثم أعد فتح الصفحة. فإذا كان يدعم متصفحك التخزين المحلي، فيجب أن تتذكر الصفحة السابقة خطواتك التي أجريتها في اللعبة، بما في ذلك عدد الخطوات التي تحركت بها، ومكان كل قطعة على رقعة اللعب، وحتى آخر قطعة قمت بتحديدها.

ما هي الآلية التي اتبعناها لفعل ذلك؟ سنستدعي الدالة الآتية في كل مرة يطرأ فيها تغيير داخل اللعبة:

```
function saveGameState() {
    if (!supportsLocalStorage()) { return false; }
    localStorage["halma.game.in.progress"] = gGameInProgress;
    for (var i = 0; i < kNumPieces; i++) {
        localStorage["halma.piece." + i + ".row"] =
gPieces[i].row;
        localStorage["halma.piece." + i + ".column"] =
gPieces[i].column;
    }
    localStorage["halma.selectedpiece"] = gSelectedPieceIndex;
    localStorage["halma.selectedpiecehasmoved"] =
gSelectedPieceHasMoved;
    localStorage["halma.movecount"] = gMoveCount;
    return true;
}
```

كما لاحظت، تستعمل الدالة السابقة الكائن `localStorage` لتخزين أنّ المستخدم قد بدأ اللعب (المفتاح `gGameInProgress`، الذي هو قيمة منطقية [Boolean]). ثم ستدور حلقة `for` على جميع القطع (المتغير `gPieces`، الذي هو مصفوفة في لغة JavaScript) ثم يحفظ رقم السطر والعمود لكل قطعة؛ ثم تحفظ الدالة بعض المعلومات الإضافية عن اللعبة، بما في ذلك القطعة التي تم تحديدها (القيمة `SelectedPieceIndex`، التي هي رقم صحيح [integer])، وفيما إذا كانت القطعة في منتصف سلسلة من القفزات (القيمة `SelectedPieceHasMoved`، التي هي قيمة منطقية)، والعدد الكلي للحركات التي قام بها اللاعب (القيمة `MoveCount`، التي هي عدد صحيح).

وعند تحميل الصفحة، وبدلًا من الاستدعاء التلقائي للدالة `(newGame()` التي ستعيد ضبط جميع المتغيرات إلى قيم مُحددة مسبقًا، فسنستدعي الدالة `(resumeGame()`. التي تتحقق -باستخدام التخزين المحلي في HTML5- فيما إذا كانت هناك نسخة محفوظة من اللعبة مُخزنًة محلًياً؛ فإن وُجدَت، فستستعاد تلك القيم باستخدام الكائن `localStorage`.

```
function resumeGame() {
    if (!supportsLocalStorage()) { return false; }
    gGameInProgress = (localStorage["halma.game.in.progress"] == "true");
    if (!gGameInProgress) { return false; }
    gPieces = new Array(kNumPieces);
    for (var i = 0; i < kNumPieces; i++) {
        var row = parseInt(localStorage["halma.piece." + i + ".row"]);
        var column = parseInt(localStorage["halma.piece." + i + ".column"]);
        gPieces[i] = new Cell(row, column);
    }
    gNumPieces = kNumPieces;
```

```

gSelectedPieceIndex =
parseInt(localStorage["halma.selectedpiece"]);
gSelectedPieceHasMoved =
localStorage["halma.selectedpiecehasmoved"] == "true";
gMoveCount = parseInt(localStorage["halma.movecount"]);
drawBoard();
return true;
}

```

أهم فكرة في هذه الدالة هي تطبيق التحذير الذي ذكرته لك سابقًا في هذا الفصل، وسأكرره هنا: «**سُتَخْرِّنُ** البيانات كسلسلة نصية، وهذا يعني أنّه لو لم تكن القيمة المُخْرَّنة نصيةً فستحتاج إلى تحويل البيانات التي حصلت عليها إلى نوع البيانات الذي تريده». فعلى سبيل المثال، القيمة التي تحدّد فيما إذا كانت هنالك لعبة قيد اللعب (`gGameInProgress`) هي قيمة منطقية، وفي الدالة `(saveGameState()` خرّنا القيمة دون أن نلقي بالاً لنوعها:

```
localStorage["halma.game.in.progress"] = gGameInProgress;
```

لكن في دالة `(resumeGame()` علينا أن نتعامل القيمة التي أحذناها من التخزين المحلي كسلسلة نصيةٍ كالأتي:

```
gGameInProgress = (localStorage["halma.game.in.progress"] ==
"true");
```

وبشكلٍ مشابه، يخّرّن عدد الخطوات في `gMoveCount` كعدد صحيحٍ؛ فقد خرّناها ببساطة في الدالة `(saveGameState()`:

```
localStorage["halma.movecount"] = gMoveCount;
```

لكن في دالة `resumeGame()` علينا أن نحول القيمة إلى عدد صحيح باستخدام الدالة `JavaScript` parseInt()` الموجودة في:

```
gMoveCount = parseInt(localStorage["halma.movecount"]);
```

5. مستقبل التخزين المحلي في تطبيقات الويب

على الرغم من أنّ الماضي كان مليئاً بالطرق الالتفافية، لكن الوضع الراهن للتخزين المحلي في HTML5 مشرق، فهناك واجهة برمجية (API) جديدة قد وضع لها معيار وطبق هذا المعيار في جميع المتصفحات الرئيسية على مختلف المنصات والأجهزة. فهذا أمرٌ لا تراه كل يوم كمطورو ويب، أليس كذلك؟ لكن ألا تتطلع إلى أكثر من «5 ميغابايت من الثنائيات على شكل مفتاح/قيمة»؟ حسناً، هنالك عدد من الرؤى التنافسية لمستقبل التخزين المحلي.

إحدى تلك الرؤى هي اختصار تعرفه بالتأكيد: SQL. أطلقت Google في عام 2007 إضافة **Gears** المفتوحة المصدر التي تعمل على مختلف المتصفحات والتي تحتوت على قاعدة بيانات **Web SQL** مضمونة فيها مبنية على **SQLite**. أثّر هذا النموذج الأولي لاحقاً على إنشاء مواصفة **Database**، والتي كانت تعرف رسمياً باسم «**WebDB**» التي توفر طبقةً للوصول إلى قاعدة بيانات SQL، سامحةً لك بالقيام بأشياءٍ شبيهةً بما يلي عبر JavaScript (لاحظ أنّ الشيفرة الآتية حقيقة وتعمل على أربعة متصفحات):

```
openDatabase('documents', '1.0', 'Local document storage',
5*1024*1024, function (db) {
  db.changeVersion('', '1.0', function (t) {
    t.executeSql('CREATE TABLE docids (id, name)');
  }, error);
});
```

كما لاحظت، ما يهم في الشيفرة السابقة هو السلسلة النصية التي مررتها إلى الدالة executeSql، ويمكن أن تحتوي تلك السلسلة النصية على أيّة تعليمات SQL مدعومة، بما في ذلك SELECT و UPDATE و INSERT و DELETE. الأمر هنا شبيه ببرمجة قواعد البيانات بلغة مثل JavaScript، إلا أنّك تقوم بذلك عبر PHP! طبّقت مواصفات Web SQL Database من أربعة متصفحات ومنصات.

Android	iPhone	Opera	Chrome	Safari	Firefox	IE
+2.0	+3.0	+10.5	+4.0	+4.0	.	.

وبكل تأكيد، لو سبق وأن استخدمت أكثر من محرك لقواعد البيانات في حياتك، فأنت تعلم أن «SQL» هي مصطلح تسويقي أكثر من كونها معياراً متكاملاً (قد يقول البعض أن HTML5 كذلك، لكن لا تأبه بقولهم). حسناً، هنالك معيار للغة SQL (يسمى SQL-92) لكن لا يوجد خادوم قواعد بيانات في العالم يتواافق تماماً مع ذاك المعيار. فهنالك نسخة SQL لقواعد بيانات Oracle، ونسخة أخرى لقواعد MySQL، ونسخة أخرى لقواعد بيانات MSSQL، وأخرى لقواعد بيانات SQLite، ولا ننس نسخة PostgreSQL لقواعد بيانات SQLite. وحتى كل منتج من تلك المنتجات يُضيف ميزات SQL جديدة على مرّ الزمن، وبهذا يكون قولنا «نسخة SQL لقواعد بيانات SQLite» ليس كافياً لتحديد ما نتحدث عنه بدقة. فعليك أن تقول «نسخة SQL التي تأتي مع قواعد بيانات SQLite ذات الإصدار Z.Y.X.».

كل ما سبق أدى إلى الإعلان الآتي، التي يقع الآن في أعلى [صفحة مواصفة](#)

:Web SQL Database

واجهت هذه الموصافة طريقاً مسدوداً: جميع المتصفحات التي طبقت هذه الموصافة استخدمت السند الخلفي (backend) نفسه لقواعد البيانات (الذي هو SQLite)، لكننا نحتاج إلى عدّة تطبيقات مختلفة إضافية للإكمال في مسار تحويلها إلى معيار. وريثما يحين ذاك الوقت، فستشير «نسخة» (dialect) SQL إلى المستخدمة في هذا المعيار إلى SQLite، لكن هذا ليس مقبولاً بالنسبة لمعايير قياسي.

وعلى ضوء هذا، سأعرّفك على رؤية تنافسية أخرى لتخزين محلٍ متقدم وثابت لتطبيقات الويب: [Indexed Database API](#) المعروفة سابقاً باسم «WebSimpleDB» التي اشتهرت باسم «IndexedDB».

تحتوي IndexedDB على ما يُسمى «مخزن الكائنات» (object store)، الذي يتشارك مع قاعدة بيانات SQL في الكثير من المفاهيم؛ فهناك «قواعد بيانات» (databases) فيها «سجلات» (records)، ويملك كل سجل عدداً من «الحقول» (fields)، وكل حقل له نوع بيانات معين، الذي يُعرف عند إنشاء قاعدة البيانات. تستطيع أيضاً تحديد مجموعة فرعية من السجلات، ثم تعرضها عبر «مؤشر» (cursor)، ويتم التعامل مع التغييرات على مخزن الكائنات عبر «التحويلات» (transactions).

إذا سبق وأن برمجت قليلاً لأي نوع من أنواع قواعد بيانات SQL، فمن المرجح أن تبدو المصطلحات السابقة مألوفةً لديك. الفرق الرئيسي هو أنَّ «مخزن الكائنات» ليس لديه «لغة استعلام بنبوية»، لا تستطيع كتابة عبارات مثل «`SELECT * from USERS where ACTIVE = '1'`» لكنك تستطيع استخدام الدوال التي يوفرها مخزن الكائنات لفتح «مؤشر» (cursor) في

قاعدة البيانات «USERS»، ثم تمر عبر السجلات، وتستبعد سجلات المستخدمين غير النشطين، ثم تستخدم دوّالاً للوصول إلى قيم كل حقل في السجلات المتبقية.

دعم IndexedDB موجود في Mozilla Firefox منذ الإصدار 4.0 (صرّحت Mozilla أنها لن تدعم Internet Explorer Web SQL Database في متصفحها)، و Chrome منذ الإصدار 11، وحتى أصبح يدعمIndexedDB من الإصدار 10.

6. مصادر إضافية

التخزين في HTML5 :

- مواصفة HTML5 Storage
- MSDN في Introduction to DOM Storage
- Opera Web Storage: easier, more powerful client-side data storage
- Developer Community
- درس تعليمي Unlock local storage for mobile Web applications with HTML5
- في IBM DeveloperWorks
- العمل الذي قام به Brad Neuberg (قبل وجود HTML5)
- userDate عن كائن Internet Explorer Has Native Support for Persistence?!?!
- في (IE)
- جزء من درس أكبر حول مكتبة Dojo Offline (التي لم تعد موجودة) Dojo Storage
- Subversion مستودع dojox.storage

:IndexedDB

مواصلة Indexed Database API •

Beyond HTML5: Database APIs and the Road to IndexedDB •

IndexedDB API •

دورة تطوير تطبيقات الويب باستخدام لغة Ruby



دورة تدريبية متكاملة من الصفر حتى الاحتراف
تمكّنك من التخصّص في هندسة الويب ودخول سوق العمل

التحق بالدورة الآن



تطبيقات الويب التي تعمل دون اتصال



ما هي تطبيقات الويب التي تعمل دون اتصال؟ يبدو الأمر من الوهلة الأولى كأنه هناك تضارياً في المفاهيم. فهل هي صفحات الويب التي تنزلها ثم تفتحها بعد ذلك؟ لكن التنزيل يتطلب اتصالاً شبكيّاً، فكيف ستستطيع تنزيل الصفحة عندما لا تكون متصلة؟ لن تستطيع فعل ذلك بكل تأكيد، لكنك تستطيع تنزيل الصفحة عندما تكون متصلة بالشبكة، وهذه هي آلية عمل تطبيقات HTML5 التي تعمل دون اتصال (offline web applications).

بأبسط الكلمات: تطبيق الويب الذي يعمل دون اتصال هو قائمةً بروابط URL التي تشير إلى ملفات HTML أو CSS أو JavaScript أو الصور أو أي مورد آخر. تُشير الصفحة الرئيسية إلى تلك القائمة التي تُسمى «ملف manifest» الذي هو ملف نصي موجودٌ في مكانٍ ما على خادوم الويب، وسيقرأ متصفح الويب الذي يدعم تشغيل تطبيقات الويب دون اتصال قائمةً روابط URL الموجودة في ملف manifest، ثم ينزل تلك الموارد (resources)، ثم يخزنُها تخزينًا مؤقتًا محليًا (local cache)، ثم سيحدث النسخ المحلية منها تلقائيًا في حال تغيرت. وعندما يحين وقت محاولتك الوصول إلى تطبيق الويب دون اتصالٍ شبكيٍّ، فسيحاول متصفحك عرض النسخة المخزنة محليًا تلقائيًا.

ومن هذه النقطة، سيلقى الحمل على عاتقك تماماً -كمطورو ويب-؛ فهناك راية (flag) في DOM تخبرك إذا كان المتصفح متصلًا بالشبكة أم لا، وهناك أحداث (events) تُفعَّل عندما تتغير حالة الوصول إلى الشبكة (أي لو كنت تعمل دون اتصال، ثم توفر لديك بعد دقيقةٍ اتصالٌ شبكيٌّ؛ أو بالعكس).

لكن إن كان يولد تطبيقك بياناتٍ أو يحفظها، فالامر متروك لك لتخزين البيانات محليًا عندما لا تكون متصلة بالشبكة ثم تزامنها مع الخادوم البعيد بعد أن تستعيد اتصالك به. بعبارة أخرى،

تتمكن HTML من جعل تطبيقك يعمل دون اتصال، لكن ما يفعله تطبيقك في هذه المرحلة عائدٌ إليك تماماً.

Android	iPhone	Opera	Chrome	Safari	Firefox	IE
+2.0	+2.1	+10.6	+5.0	+4.0	+3.5	+10

1. ملف Manifest

يتمحور تطبيق الويب الذي يعمل دون اتصال حول ملف manifest للتخزين المؤقت. إذاً ما هو ملف manifest؟ هو قائمة بكل الموارد (resources) التي يحتاج لها تطبيق الويب لكي يستطيع المستخدم الوصول إليه وهو غير متصل بالشبكة. عليك أن تشير إلى ملف manifest باستعمالك لخاصية manifest في عنصر <html> لتمهيد الطريق لعملية تنزيل وتخزين تلك الموارد تخزينياً مؤقتاً.

```
<!DOCTYPE HTML>
<html manifest="/cache.manifest">
<body>
...
</body>
</html>
```

يمكن أن يتواجد ملف manifest للتخزين المؤقت في أي مكان في خادوم الويب عندك، لكن يجب أن يُخدَّم بنوع text/cache-manifest؛ إذا كنت تستعمل خادوم ويب يعتمد على أباتشي (Apache)، فمن المرجح أن كل ما عليك فعله هو إضافة تعليمة AddType في ملف htaccess في المجلد الجذر الذي يُخدَّم منه تطبيق الويب:

```
AddType text/cache-manifest .manifest
```

تأكد أنَّ اسم ملف manifest للتخزين المؤقت ينتهي باللاحقة .manifest؛ إن كنت تستعمل خادوم ويب آخر أو ضبطاً مختلفاً لأباتشي، فراجع التوثيق الخاص بالخادوم لمزيدٍ من المعلومات حول التحكم في ترويسة Content-Type.

س: تطبيق الويب الخاص بي مُقسم إلى عدّة صفحات. هل أحتج إلى إضافة خاصية manifest في كل صفحة، أم علي وضعها في الصفحة الرئيسية فقط؟

ج: يجب وضع خاصية manifest في كل صفحة من صفحات موقعك، ويجب على تلك الخاصية أن تشير إلى ملف manifest للتخزين المؤقت لكامل تطبيق الويب.

حسناً، على كل صفحة من صفحات HTML أن تشير إلى ملف manifest للتخزين المؤقت، ويجب أن يُخدم ملف manifest بترويسة Content-Type مناسبة. لكن ماذا يوجد داخل ملف manifest؟

أول سطر من أي ملف manifest هو:

CACHE MANIFEST

وبعد هذا السطر، ستُقسم جميع ملفات manifest إلى ثلاثة أقسام: قسم «explicit»، وقسم «online whitelist»، وقسم «fallback». لدى كل قسم ترويسة مذكورة في سطرٍ خاصٍ بها؛ وإن لم يحتوي ملف manifest على أية ترويسات، فهذا يعني ضمنياً أنَّ كل الموارد المذكورة في الملف موجودة في القسم «explicit». لا تحاول أن تطيل التفكير في الاصطلاحات السابقة، خشية أن ينفجر رأسك من الصداع.

هذا ملف manifest سليم البنية، ويحتوي على ثلاثة موارد: ملف CSS، وملف JavaScript،

وصورة JPEG.

```
CACHE MANIFEST
/clock.css
/clock.js
/clock-face.jpg
```

لا يحتوي ملف manifest السابق على تزويدات للأقسام، لذا ستكون جميع الموارد المذكورة فيه موجودةً في قسم «explicit» افتراضياً. ستنزل الموارد المذكورة في قسم «explicit» وتحزن تخزينها مؤقتاً محلياً، وستستعمل مكان نظيراتها الموجودة على الشبكة في حال كان المستخدم غير متصل. وبهذا -وعند تحميل ملف manifest للتخزين المؤقت- سيُنزل متصفح الويب الملفات clock.css وclock.js وclock-face.jpg من المجلد الجذر لخادوم الويب، ثم إذا أزليت مقبس الشبكة وأعدت تحديث الصفحة، فستبقى كل تلك الموارد متوفرةً دون اتصال.

س: هل أحتج إلى ذكر جميع صفحات HTML في ملف manifest؟
ج: نعم ولا. إن كان تطبيق الويب محتوى في صفحةٍ وحيدة، فعليك أن تتأكد أن تلك الصفحة تشير إلى ملف manifest باستخدام خاصية manifest.html. فعندما تفتح صفحة HTML تحتوي على خاصية manifest فستعتبر الصفحة نفسها جزءاً من تطبيق الويب، ولست بحاجة إلى وضعها ضمن ملف manifest.
 أما لو كان تطبيق الويب يمتد على أكثر من صفحة، فعليك حينها أن تذكر جميع صفحات HTML في ملف manifest، وإلا فلن يعلم المتصفح أن هناك صفحات HTML أخرى عليه تنزيلها وتخزينها تخزينياً مؤقتاً.

أ. NETWORK

هذا مثال أكثر تعقيداً. نقل أنك تريد من تطبيق الساعة الذي أنشأته أن يتبع الزوار باستخدام سكريبت tracking.cgi الذي سيحمل ديناميكياً في خاصية img src، إلا أنَّ

تخزين هذا الملف تخزيًنا مؤقتاً سيؤثـر سلباً على عَرَض ذاك السكريبت (الذي هو تتبع المستخدمين آلياً)، لذا لا يجب أبداً تخزين هذا المورد مؤقتاً ولا يجوز أن يُتاح دون اتصال.

هذه هي طريقة فعل ذلك:

```
CACHE MANIFEST
NETWORK:
/tracking.cgi
CACHE:
/clock.css
/clock.js
/clock-face.jpg
```

ملف السابق يحتوي على ترويسات للأقسام، فالسطر الذي يحتوي على NETWORK هو بداية قسم «online whitelist»، والموارد المذكورة في هذا القسم لن تخزن محلياً ولن تتوفر دون اتصال (محاولة تحميلها عند عدم توفر اتصال ستؤدي إلى خطأ). أما السطر: CACHE فهو بداية قسم «explicit»، وبقية ملف manifest مماثلة للمثال السابق، حيث سينزل وسيخزن كل مورد من الموارد المذكورة محلياً وسيتاحة للاستعمال دون اتصال.

ب. قسم **FALLBACK**

هناك نوع آخر من الأقسام في ملف manifest: قسم fallback، الذي تعرّف فيه البديل عن الموارد الموجودة على الشبكة التي -لسببٍ من الأسباب- لا يمكن تخزينها مؤقتاً أو لم يتم ذلك بنجاح.

توفر مواصفة HTML5 حالاً ذكيًّا لطريقة استخدام قسم fallback:

```
CACHE MANIFEST
FALLBACK:
/ /offline.html
```

NETWORK:

*

ماذا يفعل المثال السابق؟ أوًّا، لنأخذ مثلاً موقعاً يحتوي ملابس الصفحات مثل ويكيبيديا؛ لا تستطيع تنزيل كامل الموقع، ومن المؤكد أنك لا ترغب بذلك. لكن لنقل أنك تريد أن تأخذ جزءاً منه وتجعله متواافقاً دون اتصال؛ لكن كيف ستقرر ما هي الصفحات التي تحتاج إلى تخزينها مؤقتاً؟ مثلاً عن: كل صفحة زرتها في موقع ويكيبيديا (الذي افترضنا جدأً أنه يدعم التشغيل دون اتصال) ستنزل وستخزن مؤقتاً. وهذا يتضمن كل مدخلة من مدخلات الموسوعة التي زرتها، وكل صفحة نقاش (أي المكان الذي تتناقش فيه عن مدخلة معينة)، وكل صفحة تحرير (أي تلك الصفحة التي تُجري فيها تعديلاتٍ إلى مدخلة ما).

هذا ما سيفعله ملف manifest السابق: لنفترض أن كل صفحة HTML (صفحة المدخلة أو النقاش أو التعديل أو تاريخ الصفحة) في ويكيبيديا تشير إلى ملف manifest السابق؛ فعندما تزور أي صفحة تشير إلى ملف manifest فسيقول متصفحك: «مهلاً، هذه الصفحة جزء من تطبيق الويب يعمل دون اتصال، لكن هل أعرف شيئاً عن هذا التطبيق؟» إن لم يُنزل متصفحك ملف manifest المحدد من قبل قط، فسيهييء «مخزن جديد للتطبيق» (appcache)، اختصار للعبارة «application cache»، وسيُنزل جميع الموارد المذكورة في ملف manifest، ثم سيضيف الصفحة الحالية إلى مخزن التطبيق (appcache).

أما إن كان يعرف متصفحك ملف manifest من قبل، فسيضيف الصفحة الحالية إلى مخزن التطبيق (appcache) الموجود مسبقاً. وفي كلا الحالتين ستضاف الصفحة التي زرتها إلى مخزن التطبيق. وهذا مهم لأنك تستطيع بناء تطبيق ويب يُضيف الصفحات التي يزورها المستخدم تلقائياً إلى مخزنه. فلست بحاجة إلى وضع كل صفحة من صفحات HTML في ملف

manifest للتخزين المؤقت.

انظر الآن إلى قسم fallback في ملف manifest السابق، الذي يحتوي على سطيرٍ واحد. القسم الأول من السطير (الذي يقع قبل الفراغ) ليس عنوان URL، وإنما نمط URL pattern ()، حيث سيُطابِق المحرف / أي صفحة في موقعك وليس الصفحة الرئيسية فقط. فعندما تحاول زيارة صفحة ما عندما تكون غير متصل بالشبكة، فسيبحث متصفحك عنها في مخزن التطبيق (appcache)، فإن وجد المتصفح الصفحة في مخزن التطبيق (لأنك زرّتها عندما كنت متصلًا بالشبكة، ومن ثم أضيَّقَت الصفحة إلى مخزن التطبيق [appcache] في ذات الوقت)، فسيعرض المتصفح النسخة المخزنة من الصفحة محليًّا؛ إما إذا لم يجد متصفحك الصفحة في مخزن التطبيق، فبدلاً من عرض رسالة الخطأ، فسيعرض الصفحة offline.html / التي تمثل القسم الثاني في السطير المذكور في قسم fallback.

في النهاية، لننظر إلى قسم الشبكة (network). يحتوي قسم الشبكة في ملف manifest السابق على سطير فيه محرف وحيد (*)، هذا المحرف له معنى خاص في قسم الشبكة. وهو يُدعى «online whitelist wildcard flag» وهي طريقة منمقة لقول أن كل شيء غير موجود في مخزن التطبيق (appcache) سيُنَزَّل من عنوان الويب الأصلي لطالما هنالك اتصال شبكي بالإنترنت. هذا مهم لتطبيقات الويب التي تعمل دون اتصال التي لا نريد تنزيلها كل صفحاتها ومواردها، وهذا يعني أنَّه أثناء تصفحك لموقع ويكيبيديا -الذي افترضنا جدلاً أنَّه يدعم العمل دون اتصال - مع اتصال بالإنترنت، فسيحمل المتصفح الصور ومقاطع الفيديو والموارد الأخرى المضمنة بشكلٍ اعتمادي، حتى ولو كانت في نطاقٍ (domain) مختلف (هذا الأمر شائع في الواقع الكبُرِي حتى لو لم تكن جزءاً من تطبيق ويبر يعمل دون اتصال. فتوَّد وثُخَدَّم صفحات

HTML محلّياً في تلك الخواديم، لكن الصور والفيديو تُخدَّم عبر **CDN** في نطاقٍ آخر). لكن دون هذا المحرف البديل (wildcard)، فسيتصرّف تطبيق ويكيبيديا الذي افترضنا أنّه يعمل دون اتصال تصرفاتٍ غريبة عندما تكون متصلًا بالشبكة. بشكلٍ خاص: لن يستطيع تنزيل أي صور أو مقاطع فيديو مخزنة على نطاقٍ خارجي.

هل هذا المثال كامل؟ لا، لأنّ ويكيبيديا أكثر من مجرد صفحات HTML. فهي تستعمل موارد CSS و JavaScript وصور مشتركة في كل صفحة. فيجب أن تذكّر تلك الموارد بشكلٍ صريح في قسم **CACHE** من ملف **manifest**، لكي تعرّض الصفحات عرضاً صحيحاً وتسلك سلوكاً سليماً عندما تُشغَّل دون اتصال. لكن الغاية من قسم **fallback** تكمن عندما لا تتنزّل كامل صفحات تطبيق الويب الذي يحتوي على موارد لم تذكرها بصراحة في ملف **manifest**.

2. تسلسل الأحداث

تحدثنا إلى الآن عن تطبيقات الويب التي تعمل دون اتصال، وعن ملف **manifest** للتخزين المؤقت، وعن مخزن التطبيق (**«appcache»**) بشكلٍ مبهم، وكأنّ الأمر يجري بطريقةٍ سحريةٍ فستنزل الملفات، وتقوم المتصفحات بمهامها على أتمّ وجه، وكل شيء يعمل عملاً سليماً! لكننا نتحدث هنا عن تطوير تطبيقات الويب، فلا يعمل أي شيءٍ من تلقاء نفسه.

بدايةً، لنتكلّم عن تسلسل الأحداث، تحديداً أحداث DOM. عندما يزور متصفحك صفحة **window.applicationCache**، أعلم أنّ هذا قد يبدو معقداً، لكن ثق بي، هذه أبسط نسخة تمكنت من كتابتها والتي لا تهمل أيّة معلومةٍ مهمة.

1. بعد أن يلاحظ المتصفح خاصية manifest في عنصر <html>, فسيُطلق الحدث checking مباشرةً (جميع الأحداث المذكورة هنا ستفعل في الكائن window.applicationCache), سيُفْعَل الحدث checking (وما، حتى لو زرت هذه الصفحة من قبل أو كانت تشير صفحات أخرى إلى ملف manifest نفسه).
2. إن لم يتعامل متصفحك مع ملف manifest المحدد من قبل...
 - سيُطلق الحدث downloading، ثم سيبدأ بتنزيل الموارد المذكورة في ملف manifest للتخزين المؤقت.
 - أثناء التنزيل، سيُطلق متصفحك الحدث progress بين الحين والآخر، الذي يحتوي على معلوماتٍ عن عدد الملفات التي تُرِكَت.
 - بعد إكمال تنزيل جميع الموارد المذكورة في ملف manifest بنجاح، سيُطلق المتصفح الحدث النهائي cached الذي يُشير إلى أنَّ تطبيق الويب قد خزن مؤقتًا بشكلٍ كامل، وهو جاهز لكي يستخدم دون اتصال.
3. في المقابل، إن زرت هذه الصفحة أو أي صفحة أخرى تُشير إلى نفس ملف manifest من قبل، فسيعلم متصفحك عن ملف manifest؛ فربما خزن بعض الموارد في مخزن التطبيق (appcache)، وربما خزن كامل التطبيق. إذاً السؤال الآن هو: هل تغيير ملف manifest منذ آخر مرة تحقق فيها المتصفح من ذلك؟
 - إذا كان الجواب: لا، لم يتغير ملف manifest؛ فسيُطلق متصفحك الحدث noupdate، ولن يحتاج إلى اتخاذ خطواتٍ إضافية.

- إذا كان الجواب: نعم، تغيير ملف manifest: فسيطّلّق متصفحك الحدث .manifest ويعيد تنزيل كل مورِي موجودٍ في ملف manifest downloading أثناء التنزيل، سيعطّل متصفحك الحدث progress بين الحين والآخر، الذي يحتوي على معلومات عن عدد الملفات التي تُرِكَتْ.
- بعد إعادة تنزيل كل الموارد الموجودة في ملف manifest بنجاح، سيعطّل المتصفح الحدث النهائي updateReady الذي يشير إلى أنَّ النسخة الجديدة من تطبيق الويب قد خُرِّبت مؤقّتاً بشكلٍ كامل، وهي جاهزة لكي تُستخدم دون اتصال. لكن انتبه إلى أنَّ النسخة الحديثة لن تستعمل فوراً، فلا تبدِّل النسخة القديمة آنئِياً، لكنه تستطيع استخدام النسخة الجديدة دون إجبار المستخدم على إعادة تحميل الصفحة باستدعاء الدالة (window.applicationCache.swapCache()) يدوياً.
- إذا حدث أي شيء بشكلٍ خاطئ في أي نقطة في هذه المرحلة، فسيطّلّق متصفحك الحدث error وسيتوقف. هذه هي قائمة مختصرة بالأشياء التي قد تسبِّب المشكلة:

 - أعاد ملف manifest رسالة الخطأ 404 HTTP (أي Page Not Found) أو 410 (أي (Permanently Gone).
 - غُرِّرَ على ملف manifest ولم يتغيّر، لكن فشل تنزيل صفحة HTML التي تشير إلى الملف.
 - تغيير ملف manifest أثناء التحديث.
 - غُرِّرَ على ملف manifest وقد تغيّر، لكن المتصفح قد فشل بتنزيل أحد الموارد المذكورة فيه.

3. تنقية الأخطاء

أريد أن أشير إلى نقطتين مهمتين هنا، أولهما هو شيءٌ قرأته لتتوّك لكنني متأكّد تماماً أنّك لم تعره اهتماماً، لذا دعني أكرره: إذا فشل تنزيل أحد الموارد الموجودة في ملف manifest تزييلاً سليماً، فستفشل عملية التخزين المؤقت لتطبيق الويب الذي يعمل دون اتصال بالكلية، وسيطّلق المتصفح الحدث `error`، لكن ليس هنالك أيّة إشارات إلى ماهية المشكلة. وهذا ما يجعل تنقية (debugging) تطبيقات الويب التي تعمل دون اتصال أمراً معقداً ومزعجاً أكثر من المعتاد.

النقطة الثانية هي ليست خطأً -إذا ابتعينا الدقة التقنية-، لكنها ستبدو وكأنها علّة خطيرة في المتصفح إلى أن تعلم ما الذي يجري. لهذه النقطة علاقةً بالآلية تحقق متّصفحك فيما إذا تغيير ملف manifest. وهي عملية تتّألف من ثلاثة خطواتٍ هذه العملية مملةً لكنها مهمةً، لذا انتبه جيداً لما سأتلوه عليك.

1. سيسأل متّصفحك -عبر البنى الهيكلية الاعتيادية في HTTP- إذا انتهت صلاحية التخزين المؤقت لملف manifest. وكما في بقية الملفات التي تُخدَّم عبر HTTP، سيُضمن خادوم الويب عندك بشكلٍ اعْتِيادي بعض المعلومات حول الملف في ترويسات HTTP. بعض تلك الترويسات (`Cache-Control` و `Expires`) تخبر متّصفحك كيف يُسمح له بتخزين الملف مؤقتاً دون سؤال الخادوم إذا تغيّر أم لا. لا علاقة لهذه النوع من التخزين بتطبيقات الويب التي تعمل دون اتصال؛ وهو يحدث تقريباً لكل صفحة HTML أو صفحة أنماط CSS أو سكريبتٍ ما أو صورةً أو أي موردٍ آخر على الويب.

2. إذا انتهت صلاحية ملف manifest المؤقت (وفقاً لترويisات HTTP الخاصة به)،

فسيسأل متصفحك الخادوم إذا كانت هنالك نسخة جديدة من الملف؛ فإن وجدت، فسيُنزلها المتصفح. وللقيام بهذا، سيرسل متصفحك طلبية HTTP التي تتضمن تاريخ آخر تعديل لملف manifest المخزن مؤقتاً، وهو نفسه الذي أرسله الخادوم في جواب HTTP (HTTP response) في آخر مرة نزل فيها متصفحك ملف manifest. إذا قال خادوم الويب أنَّ ملف manifest لم يتغير منذ ذاك الوقت، فسيعيد الخادوم الحالة 304 (أي Not Modified). أكرر أنَّ هذا لا علاقة له بتطبيقات الويب التي تعمل دون اتصال، وهو يحدث لكل نوع من أنواع الموارد الموجودة في الويب.

3. إذا ظنَّ خادوم الويب أنَّ ملف manifest قد تغير منذ ذاك التاريخ، فسيعيد الحالة

HTTP 200 (أي OK) متبوعةً بمحطويات الملف الجديد بالإضافة إلى ترويisات Cache-Control الجديدة وتاريخ جديد لآخر تعديل، لذا ستعمل الخطوتان 1 و 2 بشكلٍ سليم في المرة القادمة (بروتوكول HTTP جميل جداً؛ إذ تخطط خواديم الويب للمستقبل دائمًا، فإن كان على خادوم الويب إرسال ملفٍ إليك، فسيفعل ما بوسعه لكي يتأكد أنه لن يحتاج إلى إرساله لك مرتين دون داعٍ).

بعد تنزيل ملف manifest الجديد، سيقارن المتصفح المحطويات مع النسخة التي نزلها آخر مرة. إذا كانت محطويات ملف manifest مماثلةً لمحطويات آخر نسخة، فلن يُعيد متصفحك تنزيل أيٍّ من الموارد المذكورة في الملف.

قد تعرض إحدى الخطوات السابقة طريقة أثناء تطويرك واختبارك لتطبيق الويب الذي يعمل دون اتصال، على سبيل المثال، لنقل أنَّك أنشأت نسخةً من ملف manifest، ثم بعد 10

دقايق أدركت أنك تحتاج إلى إضافة مورد آخر إلى الملف. لا توجد مشكلة، صحيح؟ أضفت سطراً جديداً وجريت التطبيق، لكنه لم يعمل! إليك ما حدث: عندما أعددت تحميل الصفحة، لاحظ المتصفح خاصية manifest، وأطلق الحدث checking، لكنه لم يفعل شيئاً... أصرّ متصفحك بعنادٍ أن ملف manifest لم يتغير. لماذا؟ لأن خادوم الويب -افتراضياً- مضبوط للطلب من المتصفحات أن تخزن الملفات الثابتة مؤقتاً لعدة ساعات (وذلك عبر ترويسات Cache-Control في بروتوكول HTTP). وهذا يعني أن متصفحك سيبقى عالقاً في الخطوة رقم 1 من الخطوات الثلاث السابقة. من المؤكد أن خادوم الويب يعرف أن الملف قد تغير، لكن متصفحك لن يحاول سؤال خادوم الويب. لماذا؟ لأنه في آخر مرة نزل متصفحك فيها ملف manifest، طلب الخادوم منه أن يخزن الملف مؤقتاً لعدة ساعات، لكن متصفحك حاول إعادة طلب ذاك الملف بعد 10 دقائق.

دعني أوضح لك أن هذه ليست علة وإنما ميزة. إذ يعمل كل شيء كما يجب. إن لم تكن هنالك طريقة لتخزين الملفات مؤقتاً في خواديم الويب (والخواديم الوسيطة [proxies]), فسينهار الويب بين ليلة وضحاها. لكن هذا ليس عزاءً لك بعد أن قضيت عدة ساعات محاولاً معرفة لماذا لم يلاحظ متصفحك أن ملف manifest قد تعدل (من الطريف إن كنت قد انتظرت فترة كافيةً ثم أعددت تحميل الصفحة فسيعمل الملف بشكلٍ سحري! ذلك لأن فترة صلاحية الملف قد انتهت، كما قد حدّ لها تماماً).

هذا ما عليك فعله فورياً: إعادة ضبط خادوم الويب عندك لكي لا يخزن ملف manifest مؤقتاً عبر بروتوكول HTTP. إذا كنت تستعمل خادوم ويب مبني على أباتشي، فكل ما عليك فعله هو إضافة السطرين الآتيين إلى ملف htaccess:

```
ExpiresActive On
ExpiresDefault "access"
```

التعليمات السابقة ستعطل التخزين المؤقت لكل ملف في ذاك المجلد وفي جميع المجلدات الفرعية، ومن المرجح أنك لا تريده فعل هذا في خادوم إنتاجي؛ لذا عليك إما أن تضع التعليمات السابقة ضمن تعليمية `<Files manifest>` لكي تؤثر على ملف `manifest` فقط، أو تنشئ مجلداً فرعياً لا يحتوي إلا على ملف `manifest` وملف `.htaccess`. فقط. وكالمعتاد، تختلف تفاصيل الضبط من خادوم إلى آخر، لذا راجع توثيق خادوم الويب الذي تستعمله لتفاصيل حول كيفية التحكم بمترويسات التخزين المؤقت لبروتوكول HTTP.

بعد أن تعطل التخزين المؤقت الخاص ببروتوكول HTTP على ملف `manifest`، ربما تواجهك مشكلة أخرى في أنك قد عدلت في أحد الموارد التي ستُخزن في مخزن التطبيق (`appcache`) للاستعمال دون اتصال، لكنها بقيت تملك نفس عنوان URL في خادومك. هنا ستعرض الخطوة رقم 2 من الخطوات الثلاث السابقة طريقك. إذا لم يتغير محتوى ملف `manifest`، فلن يلاحظ المتصفح أن أحد الموارد المخزنة مسبقاً قد تغير. ألق نظرةً إلى المثال الآتي:

```
CACHE MANIFEST
# rev 42
clock.js
clock.css
```

إذا عدلت ملف الأنماط `clock.css` وجربت التطبيق، فلن تلاحظ وجود التعديلات التي أجريتها، وذلك لأن ملف `manifest` نفسه لم يُعدل.

لذا في كل مرة تجرب فيها تعديلاً لمورِّد ما في تطبيق الويب الذي يعمل دون اتصال، فعليك أن تعدل ملف manifest نفسه. وهذا بسيط جدًا، فلا يلزمك إلا تغيير حرفٍ واحدٍ. أسهل طريقة وجدتها لفعل هذا هو تضمين تعليق فيه رقم النسخة أو المراجعة (revision). غير رقم النسخة الموجود في التعليق، ثم سيلاحظ متصفحك أنَّ محتوى الملف قد تغير وسيعيد تنزيل كل الموارد المذكورة فيه.

```
CACHE MANIFEST
# rev 43
clock.js
clock.css
```

4. للنشيء واحدًا!

هل تتذكر لعبة الضامة التي برمجناها في [فصل canvas](#) ثم حسناًها لاحقاً بحفظنا للتحركات عبر التخزين المحلي؟ ما رأيك أن نجعل اللعبة تعمل دون اتصال. علينا أن ننشئ ملف manifest يحتوي على قائمة بجميع الموارد التي تحتاج لها اللعبة. حسناً، هناك صفحة HTML رئيسية، وملف JavaScript وحيد الذي يحتوي على شيفرة اللعبة. لا توجد صور، لأننا رسمنا كل شيء برمجياً عبر [واجهة البرمجة لعنصر canvas](#). وجميع أنماط CSS موجودة في عنصر `<style>` في أعلى صفحة HTML. ما هو ذا ملف manifest الخاص باللعبة:

```
CACHE MANIFEST
halma.html
../halma-localstorage.js
```

كلمة عن المسارات: أنشأ مجلداً فرعياً باسم `examples` في مجلد `offline`, وملف `manifest` السابق موجود هناك في المجلد الفرعى. ولأنَّ صفحة HTML ستحتاج إلى تعديلٍ بسيطٍ لكي تعمل دون اتصال (سأريك به بعد دقيقة)، فأنشأ نسخةً منفصلةً من ملف HTML، الموجودة أيضاً في المجلد الفرعى `offline`, ولعدم وجود أيّة تعديلات على شيفرة JavaScript منذ أن **أضفنا دعماً للتخزين المحلي**, فسأستعمل ملفات `js`. نفسها الموجودة في المجلد الأب (`/examples`). ستبدو المسارات كالتالي:

```
/examples/localstorage-halma.html
/examples/halma-localstorage.js
/examples/offline/halma.manifest
/examples/offline/halma.html
```

نرى——د الإشارة إلى ملفين في مجلد `examples` للتخزين المحلي `halma.manifest` (أولهما هو النسخة التي تعمل دون اتصال لملف `halma.html`) ولأنَّ كلا الملفين في المجلد نفسه، فمن الممكن ذكره في ملف `manifest` دون وجود سابقة للمسار. أما ثانيهما، فهو ملف JavaScript الموجود في المجلد الأب (`halma-localstorage.js`), ويجب استخدام المسارات النسبية في ملف `halma-localstorage.js` : `manifest` .. وهذا مشابه لاستعمالك لعنوان URL نسبي في خاصية `src` ``. يمكنك أيضًا استعمال المسارات المطلقة (`absolute`)، تلك التي تبدأ من المجلد الجذر للموقع) أو حتى عناوين URL المطلقة (التي تشير إلى موارد موجودة على نطاقات أخرى).

عليها إضافة خاصية `manifest` في ملف HTML لكي تشير إلى ملف `manifest` للتخزين المؤقت:

```
<!DOCTYPE html>
<html lang="en" manifest="halma.manifest">
```

هذا كل ما عليك فعله! عندما تزور **صفحة اللعبة التي تعمل دون اتصال** بمتناصٍ حديث، فسيُنزل ملف `manifest` المُشار إليه في خاصية `manifest` ثم سيبدأ بتنزيل جميع الموارد الموجودة فيه ويضعها في مخزن التطبيق (`appcache`). ومن هنا ستتولى شيفرة الصفحة الأمر في كل مرة تزور فيها الصفحة. يمكنك اللعب دون اتصال وستخزنَّ بيانات اللعبة محلياً، لذا تستطيع أن تُغلق اللعبة ثم تعود إليها متى شئت.

5. كلمة أخيرة

أعلنت W3C أخيراً أنها **ستنزل هذه الميزة من معيار HTML5**، إلا أنَّ هذه العملية ستستغرق وقتاً طويلاً. ونصحت W3C باستخدام Service Workers بدلاً منها. المشكلة أنَّ ميزة Workers غير مدرومة من الغالبية العظمى من المتصفحات، والمتصفحات التي تتواجد فيها ميزة Service Workers تدعمها دعماً جزئياً فقط.

تركَت W3C المطوريين بين المطرقة والسنдан، لكنني أرى أنَّ عليك الاستمرار في استخدام هذه الميزة، مع متابعة أخبار دعم Service Workers لتنقل إليها في المستقبل.

6. مصادر إضافية

المعايير:

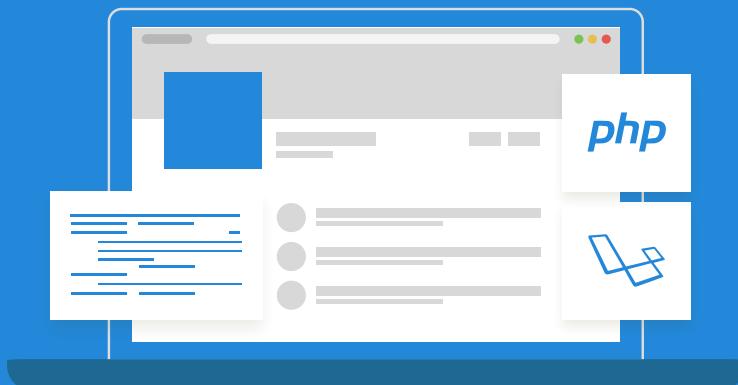
HTML5 في مواصفة **Offline web applications** •

توثيق من صانعي المتصفحات:

Offline resources in Firefox •

-
- Safari client-side storage، التي هي جزءٌ من HTML5 offline application cache •
and offline applications programming guide
الدروس التعليمية:
 - Gmail for mobile HTML5 series: using appcache to launch offline - part 1 •
 - Gmail for mobile HTML5 series: using appcache to launch offline - part 2 •
 - Gmail for mobile HTML5 series: using appcache to launch offline - part 3 •
Debugging HTML5 offline application cache •
an HTML5 offline image editor and uploader application •

دورة تطوير تطبيقات الويب باستخدام لغة PHP



مميزات الدورة

- ✓ شهادة معتمدة من أكاديمية حسوب
- ✓ إرشادات من المدربين على مدار الساعة
- ✓ من الصفر دون الحاجة لخبرة مسبقة
- ✓ بناء معرض أعمال قوي بمشاريع حقيقة
- ✓ وصول مدى الحياة لمحتويات الدورة
- ✓ تحديثات مستمرة على الدورة مجاناً

اشترك الآن



٩

النماذج

كلنا نعرف نماذج الويب، أليس كذلك؟ أنشئ `<form>`، وبعض عناصر `<input>` وأنه النموذج بزر `<input type="text">`، وربما عنصر `<input type="password">` .`<input type="submit">`

معلوماتك السابقة منقوصة، إذ تُعرّف HTML5 ثلاثة عشر نوع إدخالٍ جديِّدٍ التي تستطيع استعمالها في نماذجك. لاحظ أنني ذكرت كلمة «تستطيع» بصيغة المضارع، أي أنَّك تستطيع استخدامها الآن دون أيَّة أمور التفافية أو إضافات مُخصَّصة. لكن لا تتحمَس كثيراً؛ فلم أكن أقصد أنَّ جميع تلك الميزات الرائعة مدعومةٌ في كل متصفح؛ ما أقصده هو أنَّ تلك النماذج ستعمل بشكلٍ رائع في المتصفحات الحديثة، لكنها ستبقى تعمل في المتصفحات القديمة (بما في ذلك IE6) وإن لم يكن سلوكها مماثلاً لسلوكها في المتصفحات الحديثة.

1. النص البديل

Android	iPhone	Opera	Chrome	Safari	Firefox	IE
+2.3	+4.0	+11.0	+4.0	+4.0	+3.7	+10

أول تحسين لنماذج الويب أنت به HTML5 هو القدرة على ضبط نص بديل (`placeholder`) في حقل الإدخال. «النص البديل» هو النص الذي سيُعرض داخل حقل الإدخال لطالما كان حقل الإدخال فارغاً، وسيختفي النص البديل بعد بدء الكتابة في الحقل. من المرجح أنك شاهدت نصاً بديلاً من قبل، فمتصفح Firefox فيه نص بديل في شريط العنوان مكتوبٌ فيه «Search Bookmarks and History» (النص البديل في الإصدارات الحديثة هو «Search» فقط):



الشكل 29: النص البديل في Firefox

وعندما تهم بكتابية أي شيء فيه، فسيختفي النص البديل:



الشكل 30: سيختفي النص البديل عند بدء الكتابة

هذه آلية تضمين النص البديل في نماذج الويب الخاصة بك:

```
<form>
  <input name="q" placeholder="Search Bookmarks and History">
  <input type="submit" value="Search">
</form>
```

ستتجاهل المتصفحات التي لا تدعم النص البديل الخاصة `placeholder` ببساطة. انظر إلى الجدول أعلاه لمعرفة ما هي المتصفحات التي تدعم النص البديل.

-
- س:** هل يمكنني وضع وسوم HTML في خاصية `placeholder`? أريد أن أضيف صورةً أو أن أغير الألوان.
- ج:** لا يمكن أن تحتوي خاصية `placeholder` إلا على نص بسيط، ولا يسمح بوضع وسوم HTML فيها. لكن هناك إضافات CSS تسمح لك بتنسيق النص البديل في بعض المتصفحات.
-

2. التركيز التلقائي على الحقول

Android	iPhone	Opera	Chrome	Safari	Firefox	IE
+3.0	.	+10.1	+5.0	+5.0	+4.0	+10

يمكن لموقع الويب استخدام JavaScript للتركيز على حقلٍ للإدخال في نموذج الويب تلقائياً. على سبيل المثال، الصفحة الرئيسية لمحرك البحث Google سترُكِّز تلقائياً (auto-focus) على حقل البحث لكي تستطيع البدء في كتابة عبارة البحث مباشرةً؛ وعلى الرغم من أنَّ هذا الأمر ملائمً للكثيرين، لكنه قد يُزعج المستخدمين المتقدمين أو أولي الاحتياجات الخاصة؛ فلو ضغطت على زر المسافة (space) متوقعاً أنْ تُمَرَّر الصفحة إلى الأسفل، فستفاجأ بعدم التمرير، لأنَّ مؤشر الكتابة موجودٌ في حقلٍ من حقول النموذج (سيُكتَب فراغ في ذاك الحقل بدلاً من التمرير). وإن ضغطت على حقل إدخال مختلف أثناء تحميل الصفحة وبدأت الكتابة فيه، فسيأتي سكريبت التركيز التلقائي (بعد اكتمال تحميل الصفحة) و«يساعدك» وينقل التركيز إلى حقل الإدخال الأصلي، مما يجعلك تكتب في حقلٍ مختلف، ويقطع لك «سلسلة أفكارك». ولما كان التركيز التلقائي يُنْفَذ عبر JavaScript، فمن الصعب التعامل مع كل الحالات الغريبة؛ وليس هناك منفذٌ لمن يريدون تعطيل ميزة التركيز التلقائي.

لحل هذه المشكلة، وفرَّت HTML5 خاصية `autofocus` لجميع عناصر التحكم في نماذج الويب. عمل الخاصية `autofocus` واضحٌ من اسمها: نقل التركيز إلى حقل إدخال معين في HTML أقرب فرصة ممكنة عند تحميل الصفحة. ولكن لما كانت هذه الخاصية موجودة في JavaScript وليس مطبقة عبر متصفحات (أو مطورو الإضافات) من منح المستخدمين المتصفحات. وسيتمكن مطورو المتصفحات (أو مطورو الإضافات) من منع المستخدمين إمكانية تعطيل التركيز التلقائي تماماً.

هذا مثالٌ عن كيفية التركيز التلقائي عبر الخاصية `:autofocus`:

```
<form>
  <input name="q" autofocus>
  <input type="submit" value="Search">
</form>
```

المتصفحات التي لا تدعم الخاصية `autofocus` ستتجاهلها تماماً. انظر إلى الجدول أعلاه

لتعرف ما هي المتصفحات التي تدعم خاصية `autofocus`.

هل تريده أن تعمل ميزة التركيز التلقائي في جميع المتصفحات، وليس تلك التي تدعم `HTML5` فقط؟ يمكنك الاستمرار في استخدام سكريبت التركيز التلقائي، لكن عليك إجراء تعديلين بسيطين:

1. أضف الخاصية `autofocus` إلى شيفرة HTML

2. اكتشف إذا كان المتصفح يدعم الخاصية `autofocus`، وشغّل سكريبت التركيز التلقائي إن لم يكن يدعم المتصفح الخاصية `autofocus` داخلياً.

```
<form name="f">
  <input id="q" autofocus>
  <script>
    if (!("autofocus" in document.createElement("input"))) {
      document.getElementById("q").focus();
    }
  </script>
  <input type="submit" value="Go">
</form>
...
```

جرب الصفحة الآتية لترى مثلاً عملياً عن التركيز التلقائي.

١. ضبط التركيز التلقائي في أقرب فرصة ممكنة

تنتظر العديد من صفحات الويب وقوع الحدث `window.onload` لكي تضبط التركيز؛ لكن الحدث `window.onload` لا يُفعّل إلا بعد تحميل جميع الصور؛ وإذا حوت صفحتك على الكثير من الصور، فمن المحتمل أن السكريبت الذي ستستخدمه سيؤدي إلى إعادة التركيز على حقل الإدخال المُعيّن بعد أن يبدأ المستخدم تفاعله مع قسم آخر في صفحتك. هذا هو سبب كره المستخدمين المتقدمين لسكريبتات التركيز التلقائي.

وضعنا سكريبت التركيز التلقائي في المثال الموجود في القسم السابق بعد حقل الإدخال مباشرةً؛ وهذا حلٌّ مثالٍ لمشكلتنا، لكن قد ترى أنَّ وضع شيفرة JavaScript في منتصف الصفحة هو أمرٌ سيء (أو قد لا يسمح لك السندي الخلفي [back-end] بذلك)؛ فإن لم تستطع وضع السكريبت في منتصف الصفحة، فعليك أن تضبط التركيز أثناء وقوع حدث مخصص مثل `.window.onload` بدلاً من `jQuery $(document).ready()`

```
<head>
<script src=jquery.min.js></script>
<script>
$(document).ready(function() {
    if (!("autofocus" in document.createElement("input"))) {
        $("#q").focus();
    }
});
</script>
</head>
<body>
<form name="f">
    <input id="q" autofocus>
    <input type="submit" value="Go">
</form>
```

هذا **مثالٌ حيٌّ** عن استخدام `jQuery` للتركيز التلقائي.

تُفعّل مكتبة `jQuery` للحدث الخاص `ready` في أقرب فرصة تتوافر فيها شجرة DOM للصفحة، أي أنها تنتظر إلى أن ينتهي تحميل النص في الصفحة، لكنها لن تنتظر تحميل جميع الصور فيها. لكن هذا ليس حلاً مثالياً، فإن كانت الصفحة كبيرةً جداً أو كان الاتصال بطبيعةً للغاية، فقد يبدأ المستخدم تفاعله مع الصفحة قبل تنفيذ سكريبت التركيز التلقائي؛ إلا أنَّ هذا الحل أفضل بكثير من انتظار وقوع الحدث `window.onload`.

إذا كنت قادرًا على إضافة تعبير `JavaScript` وحيد في شيفرة صفحتك، فهنا لك حلٌّ وسطٌ بين الأمرين. يمكنك استخدام الأحداث الخاصة في `jQuery` لتعريف حدث خاص بك، ولنقل أنَّ اسمه هو `autofocus_ready`. ثم تستطيع تفعيل هذا الحدث يدوياً بعد أن ثناج خاصية `autofocus` مباشرةً.

```
<head>
<script src=jquery.min.js></script>
<script>
$(document).bind('autofocus_ready', function() {
    if (!("autofocus" in document.createElement("input")))
        $("#q").focus();
});
</script>
</head>
<body>
<form name="f">
    <input id="q" autofocus>
    <script>$(document).trigger('autofocus_ready');</script>
    <input type="submit" value="Go">
</form>
```

هذا **مثالٌ حيٌّ** عن استخدام الأحداث المخصصة في `jQuery` للتركيز التلقائي.

هذا الحل مثالٍ مثل الحل الأول، إذ يضبط التركيز إلى الحقل المُحدَّد في أقرب فرصة ممكنة، وذلك أثناء تحميل بقية نص الصفحة. لكنه ينقل تنفيذ التسلسل المنطقي لتطبيقك (التركيز على حقل الإدخال) من جسم الصفحة إلى ترويستها. يعتمد المثال السابق على مكتبة jQuery، لكن مفهوم الأحداث الخاصة ليس مقتصرًا على jQuery فحسب، فلدي مكتبات JavaScript الأخرى مثل YUI و Dojo إمكانيات مشابهة.

الخلاصة هي:

- من المهم ضبط التركيز التلقائي ضبطاً سليماً.
- يُفضل أن تدع المتصفح يضبط التركيز التلقائي عبر خاصية `autofocus` في حقل الإدخال الذي تريده تلقائياً عليه إذا كان ذلك ممكناً.
- إذا كنت تريدين حلاً للمتصفحات القديمة، فاكتشف أولًا دعم المتصفح للخاصية `autofocus` لكي تتأكد أن السكريبت الذي كتبته سينفذ على المتصفحات القديمة فقط.
- حاول ضبط التركيز التلقائي في أقرب فرصة ممكنة، حاول مثلاً أن تضع سكريبت التركيز في شيفرة HTML بعد حقل الإدخال مباشرةً. فإن لم تستطع، فاستعمل مكتبة JavaScript تدعم الأحداث المخصصة، وفُعل الحدث المخصص مباشرةً بعد شيفرة `jQuery` في مكتبة `$(document).ready()`.
- لا تنتظر الحدث `window.onload` لكي يضبط التركيز تحت أي ظرفٍ كان.

3. عناوين البريد الإلكتروني

لفترٍ تجاوزت العقد من الزمن، احتوت نماذج الويب على عدد قليلٍ من الحقول، وكان

أكثُرها شبيهًا:

نوع الحقل	شيفرة HTML	ملاحظات
مربع تأشير (checkbox)	<input type="checkbox">	يمكن تفعيله أو تعطيله
زر انتقاء (radio button)	<input type="radio">	يمكن تجميعه مع حقول أخرى
حقل كلمة مرور	<input type="password">	يُظهر نقطاً بدلاً من الحروف المكتوبة
قائمة منسدلة	<select><option>...	-
منتقى الملفات	<input type="file">	يُظهر مربع حوار «اختيار ملف»
زر الإرسال	<input type="submit">	-
نص عادي	<input type="text">	يمكن حذف الخاصية type

تعمل جميع أنواع الحقول السابقة في HTML5، فلو أردت «التحديث إلى HTML5» (ربما

بتغيير نوع المستند DOCTYPE)، فلن تحتاج إلى إجراء أيّة تعديلات على نماذج الويب عندك،

والفضل بذلك يعود إلى توافقية HTML5 مع الإصدارات التي تسبقها.

لكن HTML5 أضافت ثلاثة عشر نوعاً جديداً من الحقول، ولأسبابٍ ستتضاح لك بعد قليل، لا

يوجد سبب يمنعك من استعمالها الآن.

أول أنواع المدخلات الجديدة مخصص لعناوين البريد، ويبدو كما يلي:

```
<form>
  <input type="email">
  <input type="submit" value="Go">
```

```
</form>
```

أوشكث على كتابة جملة مطلعها: «أما في المتصفحات التي لا تدعم `type="email"` لكنني تداركُ نفسي وتوقفت. لكن ما السبب؟ لأنني لست متأكداً من ماذا يعني عدم دعم المتصفح للحقل `type="email"`، إذ «تدعم» جميع المتصفحات `type="email"`، لكنها قد لا تفعل شيئاً خاصاً لها (سترى بعد قليل بعض الأمثلة عن المعاملة الخاصة لهذا الحقل)؛ لكن المتصفحات التي لا تتعرف على `type="text"` ستعامله على أنه `type="text"` وسيظهر حقلٌ نصيٌ عاديٌ.

لا تسعني الكلمات للتعبير عن مدى أهمية ما سبق، لأنَّ في الويب ملايين النماذج التي تسألك أن تدخل عنوان بريدك الإلكتروني، وجميعها تستعمل `<input type="text">` وستشاهد مربعاً نصياً، ثم تدخل فيه عنوان بريدك، وانتهي. ثم أتت HTML5 التي أضافت `type="email"`، فهل ستترتبك المتصفحات؟ لا، يتعامل كل متصفح على وجه هذا الكوكب القيم غير المعرفة لخاصية `type` على أنها `type="text"`، بما في ذلك IE 6. لذلك تستطيع استعمال `type="email"` حالاً دون القلق حول دعم المتصفحات.

لكن ماذا يعني أنَّ المتصفح «يدعم» الحقل `type="email"`؟ حسناً، قد يعني هذا عدّة أشياء. لم تحدد مواصفة HTML5 أيّة توصية حول الواجهة الرسمية التي تظهر للمستخدم لأنواع المدخلات الجديدة. ستعرضه أغلبية متصفحات سطر المكتب مثل Chrome و Safari و Opera و Firefox كحقلٌ نصيٌ؛ ولهذا لن يلاحظ مستخدمو موقعك الفرق (إلى أن يحاولوا إرسال النموذج).

ثم ها قد أتت الهواتف المحمولة...

ليس للهاتف المحمولة لوحة مفاتيح فيزيائية، فكل «الكتابة» تتم بالضغط على لوحة مفاتيح ظاهرة على الشاشة التي تُعرض عند الحاجة لها (عند التركيز على حقل من حقول أحد النماذج في صفحة الويب على سبيل المثال). تتعرف متصفحات الهاتف المحمولة الذكية على العديد من أنواع المدخلات الجديدة في HTML5، وتُجري تعديلات ديناميكية على لوحة المفاتيح الظاهرة على الشاشة لكي تلائم نوع المدخلات.

مثلاً، عناوين البريد الإلكتروني هي نصوص، صحيح؟ بالطبع، لكنها نوع خاص من النصوص، فمثلاً يحتوي كل عنوان بريد إلكتروني (نظرياً) على رمز @ وعلى نقطة (.) واحدة على الأقل؛ ومن غير المحتمل أن يحتوي على فراغات؛ لذا لو كنت تستعمل هاتف iPhone وحاولت الكتابة في عنصر `<input type="email">`، فستظهر لوحة مفاتيح تحتوي على زر مسافة أصغر من المعتاد، بالإضافة إلى أزرار مخصصة لمحرفي @ و . وستُخصص هواتف أندرويد لوحة مفاتيحها بشكلٍ مشابه.



الشكل 31: تخصيص لوحة المفاتيح لتسهيل كتابة عناوين البريد الإلكتروني

الخلاصة: لا توجد سلبيات لتحويل جميع الحقول التي تمثل عناوين البريد الإلكتروني إلى `type="email"` في الحال. فلن يلاحظ ذلك أحدٌ إلا مستخدمي الهواتف المحمولة، الذين أظن أنهم لن يتبعوا لذلك أيضًا:-|. لكن من سيلاحظ ذلك سيبتسم بصمت ويشكرك في قلبك لأنك سهلت عليه الأمر قليلاً.

4. عناوين الويب

عناوين موقع الويب -التي يُسمّيها مهووسو المعايير «URLs»، إلا بعض المتخاذلين فأولئك يدعونها «URIs»- هي نوع آخر من النص المخصص؛ البنية العامة لعناوين الويب مرتبطة بمعايير الإنترن特 ذات الصلة. إذا طلب منك أحدهم كتابة عنوان لموقع ويب في نموذج،

فسيتوقعون منك كتابة شيء مثل «<http://www.google.com/>» وليس «125 Road»؛ ومن الشائع استخدام الخط المائل / في العناوين (الخط المائل مذكور ثلاث مرات في عنوان صفحة Google الرئيسية)؛ وينتشر استخدام النقط . أيضاً، لكن يمْعَن وضع فراغات في العنوان. لدى جميع عناوين الويب لاحقة للنطاق مثل «.com» أو «.org».

تعرض أغلبية متصفحات الويب لسطح المكتب الحديثة حقل "url" type="text" كحقلٍ نصيّ عادي، لذلك لن يلاحظ مستخدمو موقعك ذلك **إلى أن يرسلوا النموذج**. سُتعامل المتصفحات التي لا تدعم HTML5 الحقل "url" type="text" كحقلٍ تماماً، فلا بأس من استعمال هذا الحقل في صفحات الويب الحديثة عند الحاجة.

سُتعَدِّل الهواتف الذكية من طريقة عرض لوحة المفاتيح كما في **حقل عنوان البريد الإلكتروني**، لكن لوحة المفاتيح في هذه المرة سُتخصّص لتسهيل إدخال عناوين الويب. وفي هاتف iPhone سيُزال زر المسافة تماماً وسيُستعاوض عنه بنقطة وخط مائل وزر «.com». (يمكنك الضغط مطولاً على زر «.com» للاختيار بين اللاحقات الشهيرة الأخرى مثل «.org» أو «.net»)؛ وستُخصّص هواتف أندرويد لوحة مفاتيحيها بشكلٍ مشابه.



الشكل 32: تخصيص لوحة المفاتيح لتسهيل كتابة عناوين الويب

5. إدخال الأرقام

طلب إدخال الأرقام أصعب من طلب كتابة عنوان بريد إلكتروني أو موقع ويب؛ لأنَّ الأرقام معقدة أكثر مما تظن. اختر رقمًا ما بسرعة. -1؟ لا، كنت أقصد رقمًا بين 1 و 10 - 7½ لا، ليس رقمًا كسريًّا. π؟ لماذا تختار الأرقام العجيبة؟!

الفكرة التي أود إيصالها هي أنَّك لا تسأل عن «رقم ما»، فمن المحتمل أنَّك ستطلب من المستخدم إدخال رقم في مجال معين، ولا تريده إلا نوعًا محدَّدًا من الأرقام ضمن ذلك المجال، وقد تريده استبعاد الأعداد الكسرية أو العشرية، أو أن تسمح بإدخال الأرقام التي تقبل القسمة على 10. ستسمح لك HTML5 بكل هذا.

```
<input type="number"
       min="0"
       max="10"
       step="2"
       value="6">
```

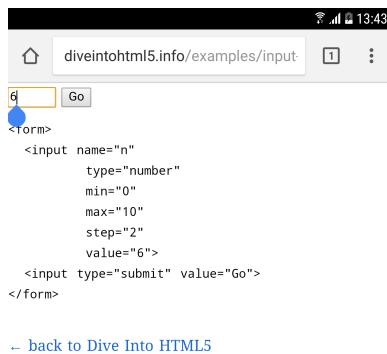
لنتحدث عن الخواص السابقة كلاً على حدة (يمكنك المتابعة مع [المثال الحي](#) إن شئت).

- الخواص "type="number" تعني أنَّ الحقل يقبل الأرقام.
- الخواص "min="0" تُحدِّد القيمة الدنيا المقبولة لهذا الحقل.
- الخواص "max="10" تُحدِّد القيمة القصوى المقبولة.
- الخواص "step="2" مجتمعةً مع قيمة الخاصية min ستعُرِّف ما هي الأرقام المسموحة في المجال: 0 و 2 و 4 وهكذا إلى أن تصل إلى قيمة الخاصية max.
- الخاصية "value="6" تُحدِّد القيمة الافتراضية. يفترض أن تكون هذه الخاصية مألوفةً لديك، فهي نفس الخاصية التي تستعملها دومًا لتحديد قيم حقول النموذج (ذكرت هذه النقطة هنا لكي أذكرك أنَّ HTML5 مبنية على إصدارات HTML السابقة، فلا حاجة أن تعيد تعلم كل الأمور التي تعرفها من قبل!).

هذه هي الشيفرة الخاصة بحقل الأرقام. ابقي في ذهنك أنَّ جميع الخواص السابقة اختيارية. فإذا كانت لديك قيمة دنيا للمجال المقبول لكن دون وجود حد أقصى للأرقام، فيمكنك ضبط خاصية min وعدم ضبط الخاصية max. الخطوة الافتراضية هي 1، ويمكنك عدم ذكر الخاصية step إلا إذا كانت قيمة الخطوة عندك مختلفةً عن 1. تستطيع إسناد سلسلة نصية فارغة إلى الخاصية value إن لم تكن هنالك قيمة افتراضية، أو بإمكانك حذف الخاصية تماماً. لكن HTML5 لا تقف عند هذا الحد، إذ توفر لك دوال JavaScript للتحكم بهذا الحقل:

- الدالة `n.input.stepUp()` تزيد قيمة الحقل مقدار `n`.
- الدالة `n.input.stepDown()` تنقص من قيمة الحقل مقدار `n`.
- الخاصية `n.input.valueAsNumber` تُعيد القيمة الحالية للحقل كعدد ذي فاصلٍ عشرية (تذكّر أنَّ الخاصية `n.input.value` تُعيد سلسلةً نصيّةً دوّماً).

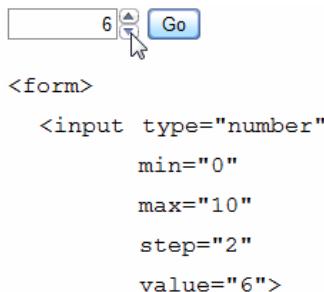
هل صعب عليك تخيل شكل هذا الحقل؟ حسناً، طريقة عرض هذا الحقل عائدة تماماً لمتصفحك، ويدعم مختلف مُصنّعي المتصفحات هذا الحقل بطراائق مختلفة. وفي الهواتف التي يصعب فيها كتابة المدخلات عموماً - سيحسن المتصفح من لوحة المفاتيح مجدداً لكتابة الأرقام.



1	2	3	-
4	5	6	←
7	8	9	✖
,	0	.	→

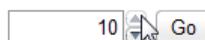
الشكل 33: تخصيص لوحة المفاتيح لإدخال الأرقام

أما في نسخة سطح المكتب من المتصفحات، سيظهر نفس الحقل `type="number"` كعنصر «`spinbox`» الذي يملك أسهماً صغيرةً للأعلى والأسفل تستطيع الضغط عليها لتغيير القيمة.



الشكل 34: حقل `spinbox` في متصفحات سطح المكتب

تحترم المتصفحات قيم الخصائص `min` و `max` و `step`، لذلك ستكون قيمة ذاك الحقل مقبولة دوماً، فلو وصلت إلى القيمة القصوى، فسيُعَطَّل زر السهم العلوي ولن تستطيع زيادة الرقم الموجود.



الشكل 35: تعطيل زر السهم العلوي عندما تبلغ قيمة الحقل الحد الأقصى

وكما في بقية حقول الإدخال التي شرحناها سابقاً في هذا الفصل، المتصفحات التي لا تدعم `type="number"` ستتعامل الحقل وكأنه `type="text"`، وستظهر القيمة الافتراضية في الحقل النصي (لأنها مخزنة في الخاصية `value`)، لكن ستجاهل المتصفح الخاصيات الأخرى مثل `min` و `max`؛ لكنك تستطيع إنشاء `spinbox` بنفسك، أو قد تستعمل مكتبة JavaScript تحتوي على هذا العنصر؛ لكن تذَّكر أن تتحقق من دعم المتصفح لهذا الحقل أولاً على سبيل المثال:

```

if (!Modernizr.inputtypes.number) {
    // لا يوجد دعم لحقول type=number
    // ربما تجرب JavaScript أو مكتبة Dojo أخرى
}

```

6. تحديد الأرقام عبر المزلاج

هناك آلية أخرى لتمثيل المدخلات الرقمية، فمن المحتمل أنك رأيك «مزلاجاً» (slider) من

قبل يُشِّيه:



الشكل 36: المزلاج

يمكنك وضع المزلاج في نماذج HTML5 أيضًا، والشيفرة الخاصة به شبيهة جدًا

:`spinbox` بحقل

```
<input type="range"
       min="0"
       max="10"
       step="2"
       value="6">
```

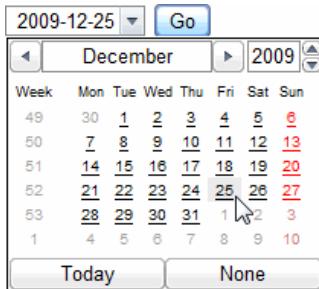
جميع الخصيـة المتـوفـرة مـاـثـلة لـحـقـل "type="number" (أـي min و max و step)، ولـها نفس المعـنىـ. الفـرقـ الـوحـيدـ هوـ فيـ وـاجـهـةـ الـاستـخدـامـ؛ فـبـدـلاـ منـ وـجـودـ حـقـلـ valueـ)، ولـها نفس المعـنىـ. الـفـرقـ الـوحـيدـ هوـ فيـ وـاجـهـةـ الـاستـخدـامـ؛ فـبـدـلاـ منـ وـجـودـ حـقـلـ type="range" كـمـزـلاـجـ، بيـنـماـ سـتـعرـضـهـ لـكـتابـةـ الرـقـمـ، سـتـعرـضـ الـمـتصـفحـاتـ الـحـديثـةـ الـحـقـلـ type="text"ـ كـمـزـلاـجـ، بيـنـماـ سـتـعرـضـهـ الـمـتصـفحـاتـ الـقـديـمةـ الـتيـ لاـ تـدعـمـ HTML5ـ كـحـقـلـ "type="text"ـ، لـذـاـ لـاـ يـوجـدـ سـبـبـ يـمـنـعـكـ منـ الـبـدـءـ باـسـتـخدـامـهـ مـباـشـرـاـًـ.

7. مُنتقي التاريخ

لم تتضمن HTML 4 حقولاً لاختيار التاريخ، لكن مكتبات JavaScript تدارك الأمر (**Dojo**) لكن هذه الحلول كانت تتطلب «الخوض في» مكتبة **UI** و **jQuery** و **YUI** و **Closure Library** (date picker). أضافت HTML5 حقولاً لانتقاء التاريخ دون الحاجة إلى كتابته يدوياً عبر JavaScript وفي الواقع، أضافت ستة حقول: واحد للتاريخ (date) وآخر للشهر (month) وآخر للأسبوع (week) وآخر للوقت (time) وآخر للتاريخ والوقت (date + time) وآخر للتاريخ والوقت لكن دون ذكر المنطقة الزمنية (date + time – timezone).

لكن للأسف، هذا الحقل غير مدعوم من أغلبية المتصفحات. إذ يدعمه متصفح Opera منذ الإصدار التاسع و Chrome من الإصدار 20.

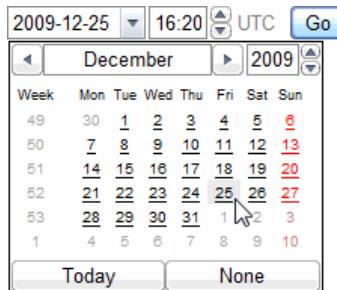
هذه هي طريقة عرض متصفح Opera لحقل `<input type="date">`:



الشكل 37: مُنتقي التاريخ

وإذا أردت من المستخدم انتقاء الوقت والتاريخ، فهناك حقل

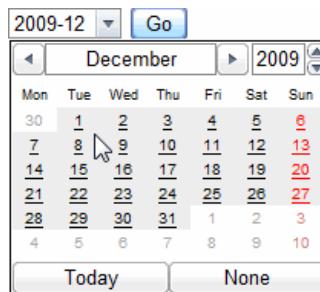
`<input type="datetime">`



الشكل 38: مُنتقي الوقت والتاريخ

أما لو كنت تحتاج إلى الشهر والسنة فقط (ربما تريد إدخال تاريخ انتهاء البطاقة

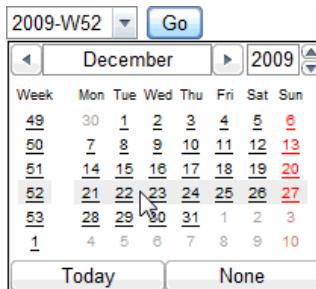
:< >



الشكل 39: مُنتقي الشهر والسنة

ويتوفر أيضًا حقل لانتقاء أسبوع معين في السنة - وإن لم يكن ذلك شائعاً - عبر الحقل

:< >



الشكل 40: مُنتقي الأسبوع

أخيرًا وليس آخرًا، يمكنك اختيار الوقت عبر الحقل <`input type="time">`:



الشكل 41: مُنتقي الوقت

من المحتمل أن تدعم المتصفحات حقول الإدخال السابقة تباعًا، لكن كما في حقل `type="email"` وغيرها، ستعرض هذه الحقول نصيّة بسيطةٍ في المتصفحات التي لا تدعم الحقل `type="date"` وأخوته. يمكنك ببساطة أن تستعمل الحقل `type="date"` وأخوته لتوفّر مُنتقي التاريخ لمستخدمي متصفحٍ Opera وـ Chrome وتنتظر دعم بقية المتصفحات. أو أن تعتمد حلاً عمليًا هو استعمال `input type="date"` ثم تكتشف إن كان المتصفح يدعم مُنتقي التاريخ، ثم تستعمل حلاً برمجيًا إن لم يكن يدعمه (مثل Dojo و UI و YUI و jQuery و Closure Library أو حلاً آخر).

```
<form>
  <input type="date">
</form>
...
<script>
  var i = document.createElement("input");
  i.setAttribute("type", "date");
```

```

if (i.type == "text") {
    // لا يوجد دعم لمنتقي التاريخ :(
    // استخدام مكتبة Dojo/jQueryUI/YUI/Closure لإنشاء واحد
    // ثم استبدل حقل <input> دينا ميكينا
}
</script>

```

8. حقول البحث

حسناً، وظيفة هذا الحقل واضحة من اسمه، لكن قد نحتاج إلى شرح آلية تطبيقه في المتصفحات.

البحث لا يكون فقط في محركات البحث مثل Google أو Yahoo، فمن الممكن أن يكون حقل البحث في أي صفحة وفي أي موقع؛ فهناك واحد في موقع أمازون، وآخر في موقع CNN، ويتوارد أيضاً في أغلبية المدونات. لكن ما هو الوسم المستخدم لتلك الحقول؟ مثل بقية حقول النص الموجودة في الويب. لنجاول `<input type="text">` تصحيح الأمر...

```

<form>
    <input name="q" type="search">
    <input type="submit" value="Find">
</form>

```

ما رأيك بتجربة حقل `<input type="search">` في متصفحك. قد لا تلاحظ أية اختلافات بينه وبين الحقل النصي العادي؛ لكن إن كنت تستعمل Safari على نظام Mac OS X، فسيبدو الحقل كما يلي:



```
<form>
  <input type="search">
  <input type="submit" value="Go">
</form>
```

الشكل 42: حقل البحث

هل لاحظت الفرق؟ لدى حقل البحث زوايا مدوره! أعلم أنّك لا تستطيع احتواء نفسك من الفرح، لكن انتظر، فهناك المزيد! عندما تبدأ الكتابة في حقل `type="search"`، فسيوضع المتصفح زر «`x`» صغير في الجانب الأيمن من الحقل؛ وبؤدي الضغط عليه إلى حذف محتويات الحقل (متصفح Chrome، الذي يشارك مع Safari في البنية الداخلية، له السلوك السابق نفسه). الغرض من التعديلات البسيطة السابقة هي إعطاء حقول البحث شكلًا وسلوگًّا شبّهها بحقول البحث في تطبيقات Mac OS X المكتبية مثل iTunes.



```
<form>
  <input type="search" value="foo">
  <input type="submit" value="Go">
</form>
```

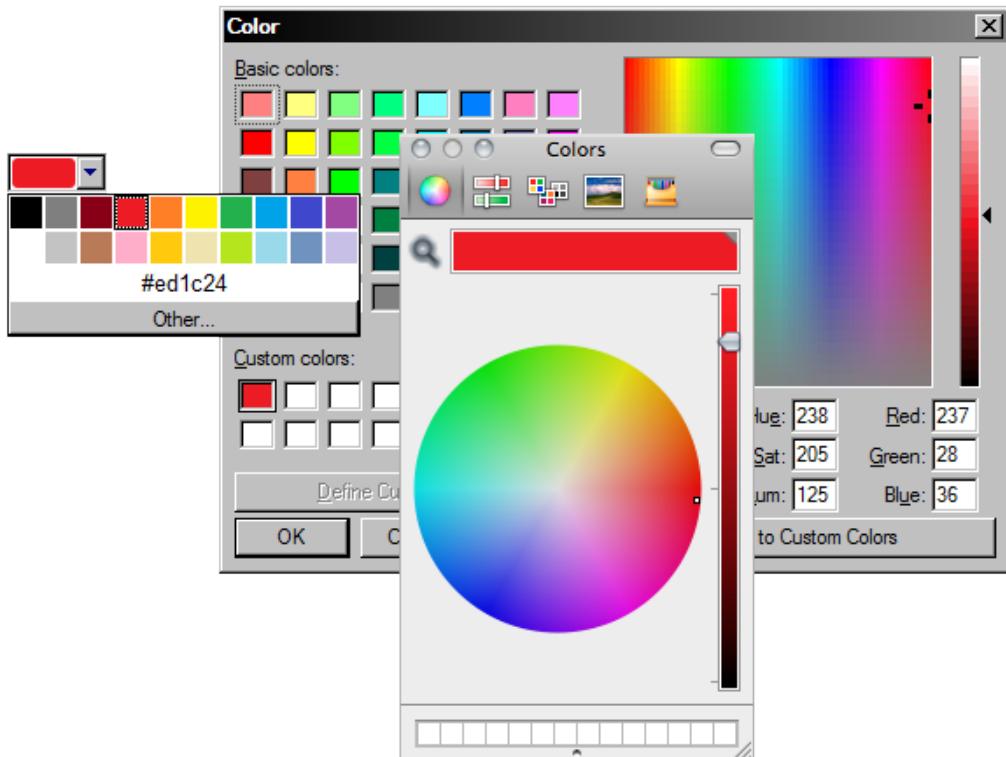
الشكل 43: حقل البحث بعد الكتابة فيه

يُستعمل موقع Apple.com العنصر `<input type="search">` لحقل البحث في الموقع لإعطائه شكلًا مألوفًا لمستخدمي Mac، لكن ذلك ليس خاصًا بنظام Mac فقط؛ فهو شيفرة HTML فحسب، وبهذا يُستطيع كل متصفح على كل منصة (أو نظام تشغيل) أن يعرض الحقل بشكلٍ مشابه لعناصر الواجهة الرسومية الخاصة بالمنصة. وكما هو الحال في بقية أنواع الحقول،

المتصفحات التي لا تعرف على حقل "type="text" ستعامله كأنه "type="search"؛ فلا يوجد سبب يمنعك من استخدام "type="search" في حقول البحث حالاً.

9. مُنتقي الألوان

تُعرف HTML5 حقل `<input type="color">` الذي يسمح لك باختيار لونٍ ما ويعيد التمثيل الست عشري لللون المختار؛ تأخرت المتصفحات في دعم هذا الحقل، إذ يدعمه Opera منذ الإصدار 17، و Firefox منذ الإصدار 29، و Chrome منذ الإصدار 20، وما زلنا في انتظار دعم بقية المتصفحات له. يندمج هذا الحقل جيداً مع مُنتقي الألوان الموجود في نظامي ويندوز و Mac، أما في لينكس فهو يعرض مُنتقي ألوان أساسي. وهو يعيد قيمة ست عشرية لللون RGB الذي يمكن استخدامه في أي مكان يقبل ألوان CSS (جرب مُنتقي الألوان في متصفحك).



الشكل 44: مُنْتَقِي الْأَلْوَانِ فِي وِينْدُوز وَ Mac

10. التحقق من صحة مدخلات المستخدم

Android	iPhone	Opera	Chrome	Safari	Firefox	IE
+4.0	+4.1	+9.0	+10	+5.0	+4.0	+10

تحديث في هذا الفصل عن عددٍ من حقول الإدخال الجديدة وبعض الميزات المحدثة مثل التركيز التلقائي لحقلٍ من حقول النموذج، لكنني لم أذكر ما أعتبره أهم جزءٌ من النماذج الحديثة في HTML5: التتحقق التلقائي من صحة مدخلات المستخدم. خذ على سبيل المثال مشكلةً شائعةً

هي إدخال عنوان بريد إلكتروني في نموذج ويب؛ ربما ستجري تحققًا من مدخلات المستخدم من طرف العميل عبر JavaScript، متبعًا بتحققٍ من جهة الخادم عبر PHP أو Python أو أيًا كانت لغة البرمجة التي تستعملها. لن يُشكّل التحقق من مدخلات المستخدم عبر HTML5 بديلاً عن التتحقق من جهة الخادم، لكن من المرجح أن تكون بديلاً عن سكريبتات JavaScript التي تستعملها.

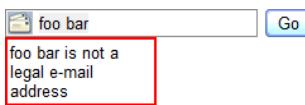
هناك مشكلتين كبيرتين في التتحقق من البريد الإلكتروني عبر JavaScript:

1. عدد كبير جدًا من زوار موقعك (حوالي 10% تقريبًا) يُعطّلون JavaScript

في متصفحهم

2. ستفشل في التتحقق من صحة البريد الإلكتروني تحققًا سليماً

أنا آسف لقول هذا، لكنك ستفشل في ذلك. عملية تحديد فيما إذا كانت سلسلة نصية ما هي عنوان بريد إلكتروني **معقدة بشكل لا يصدق**. فكلما أمعنت النظر في الأمر، لوجدت **مدى تعقيده**. هل ذكرت لتوي أنَّ الأمر **معقد جدًا**؟ أليس من الأسهل إلقاء ذاك الحمل والصداع الناتج عنه على عاتق المتصفح؟

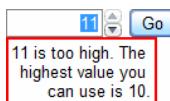


الشكل 45: التتحقق من عنوان البريد الإلكتروني

لقطة الشاشة السابقة مأخوذة من متصفح Opera 10، إلا أنَّ إمكانية التتحقق من حقول النماذج متوافرة منذ الإصدار 9 - ولدى Firefox 4 و Chrome 10 آالية مشابهة. كل ما عليك فعله هو ضبط الخاصية **type** إلى "email". وعندما يحاول المستخدم إرسال (submit) نموذج فيه حقل <input type="email">, فسيتحقق المتصفح تلقائيًا من عنوان البريد الإلكتروني حتى

لو كانت JavaScript معطلة في المتصفح.

توفر HTML5 أيضًا تحققًا من عناوين الويب المدخلة في حقول `<input type="url">`، والأرقام المدخلة في حقول `<input type="number">`; ستؤخذ قيم الخاصية `min` و `max` بالحسبان عند التتحقق من الأرقام، فلن تسمح لك المتصفحات بإرسال النموذج إذا أدخلت رقمًا كبيرًا أكبر من الحد الأقصى.



الشكل 46: التتحقق من الأرقام

لا حاجة إلى وضع شيفرات لتفعيل التتحقق من المدخلات في HTML5؛ إذ تكون مفعّلة افتراضياً، إلا أنك تستطيع تعطيلها عبر وضع الخاصية `.novalidate`.

```
<form novalidate>
  <input type="email" id="addr">
  <input type="submit" value="Subscribe">
</form>
```

11. الحقول المطلوبة

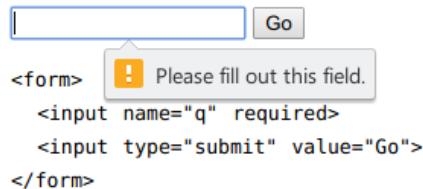
Android	iPhone	Opera	Chrome	Safari	Firefox	IE
+4.0	+4.1	+9.0	+10	+5.0	+4.0	+10

التتحقق من مدخلات المستخدم في HTML5 ليس محدوداً بنوع الحقل، إذ تستطيع أيضاً أن تُشير إلى أن بعض الحقول «مطلوبه»؛ يجب توفير قيم للحقول المطلوبة قبل أن ترسل النموذج. شيفرة الحقول المطلوبة بسيطة جدًا:

```
<form>
  <input id="q" required>
  <input type="submit" value="Search">
</form>
```

يمكنك تجربة حقل `<input required>` في متصفحك.

قد تُغيّر بعض المتصفحات الشكل الافتراضي للحقول المطلوبة. وإذا حاول المستخدم إرسال النموذج دون تعبئة الحقل المطلوب، فسيظهر إشعار يخبر المستخدم أنَّ من الضروري إدخال قيمة في الحقل وعدم تركه فارغاً (الصورة الآتية من متصفح Chrome):



الشكل 47: حقل مطلوب

12. مصادر إضافية

المواصفات والمعايير:

- `<input>` أنواع
- `<input placeholder>` خاصية
- `<input autofocus>` خاصية
- `<form novalidate>` خاصية
- `<input required>` خاصية

مكتبات JavaScript:

HTML5, مكتبة لاكتشاف دعم Modernizr •

مقالات مفيدة:

Forward Thinking Form Validation •

Mozilla Developer Center: Forms in HTML5 •

HTML5 Forms in Mozilla Firefox •

HTML5 Form Validation •



أكبر موقع توظيف عن بعد في العالم العربي

ابحث عن الوظيفة التي تحقق أهدافك وطموحاتك
المهنية في أكبر موقع توظيف عن بعد

تصفح الوظائف الآن

البيانات الوصفية



هناك أكثر من 100 عنصر في HTML5، بعضها هيكلٌ تماماً وبعضها مجرد حاويةٌ لواجهةٍ برمجية. عبر تاريخ HTML، تجادل كاتبو المعايير حول العناصر التي يجب تضمينها في اللغة، فهل يجب أن تحتوي HTML على عنصر <figure>؟ أو عنصر <person>؟ مثلاً عن عنصر <rant>؟ أتُخذَت القرارات، وكتبت المعايير، وطور المطوروون تطبيقاً لهم، وأضاف صانعو المتصفحات الميزات لمتصفحاتهم، ودفعَت عجلة تطوير الويب إلى الأمام. من المؤكَّد أنَّ HTML لن تستطيع إرضاء الجميع، إذ لا يستطيع أيٌّ معيارٌ فعل هذا. لم تصل بعض الأفكار المقترحة إلى المستوى المطلوب، فمثلاً، لا يوجد عنصر <person> في HTML5 (وكذلك الأمر لعنصر <rant>)؛ لا يوجد شيءٌ يمنعك من إضافة عنصر <person> إلى صفحات الويب التي تكتبها، لكنها لن تكون سليمةً بنحوًياً، ولن تعمل بشكلٍ متماثل في جميع المتصفحات، وقد تتعارض مع معايير HTML المستقبلية إنْ أضافتها لاحقاً.

حسناً، إن لم يكن الحل كاملاً في إنشاء عناصر جديدة، فماذا على مطُور الويب الذي يحب اتباع القواعد الهيكلية أن يفعل؟ كانت هناك محاولات لتتوسيع الإصدارات القديمة من HTML. أشهر طريقة هي microformats، التي تستعمل الخاصيتين class و rel في HTML. خيار آخر هو RDFa، التي صُممَت لتعمل في XHTML لكنها خُذلت لاحقاً إلى HTML أيضاً. لدى Microformats و RDFa نقاط قوةٌ وضعف، إذ تأخذان طريقاً مختلفاً تماماً لتحقيق الهدف نفسه: توسيعة صفحات الويب بإضافة بنى هيكلية جديدة لا تمثل جزءاً من أساس لغة HTML. لا أنوي أن أقلب هذا الفصل إلى حربٍ بين الصيغ؛ وإنما أريد أن أرتكز على خيار ثالثٍ طُور بعد تعلم الدروس من RDFa و microformats، وصُمم ليندمج جيداً مع HTML5: إنه «البيانات الوصفية» (microdata).

1. ما هي البيانات الوصفية؟

كل كلمة في الجملة الآتية مهمة، لذا انتبه جيداً إليها:

توضّف البيانات الوصفية شجرة DOM بثنائياتٍ على شكل «الاسم/القيمة» آتية من أنواع اصطلاحاتٍ مُخصصة.

حسناً، ماذا تعني الجملة السابقة العجيبة؟ لنبدأ من نهايتها إلى بدايتها. تتمحور البيانات الوصفية (microdata) حول «أنواع الاصطلاحات المخصصة» (custom vocabularies). تخيل أن جميع عناصر HTML5 هي نوعٌ وحيدٌ من الاصطلاحات. وهذا النوع يستطيع تمثيل «قسم» (section) أو «مقالة» (article)، لكنه لا يستطيع تضمين عناصر تمثيل «شخص» أو «حدث». فإذا أردت تمثيل «شخص» في صفحة الويب، فعليك تعريف نوع اصطلاحاتٍ خاص بك. تسمح لك البيانات الوصفية (microdata) بذلك، إذ يستطيع أيُّ شخصٍ أن يُعرّف اصطلاحات microdata خاصة به ويبدأ بتضمين خصائصه (properties) في صفحات الويب التي يطورها. النقطة الثانية التي عليك أن تعرفها عن البيانات الوصفية أنَّه تعمل وفق ثنائيةات «الاسم/القيمة». فكل نوع اصطلاحاتٍ يُعرّف مجموعةً من الخصائص التي لها أسماءً معينة. على سبيل المثال، قد يتضمن نوع الاصطلاحات «person» خصيّاتٍ مثل name و image. وللتضمين خاصية من خصائص البيانات الوصفية (microdata) في صفحة الويب، عليك وضع اسم الخاصية في مكانٍ معين. واعتماداً على العنصر الذي تضع فيه خاصيتك، هنالك قواعد حول كيفية استخلاص قيمة الخاصية (سنأتي على ذكرها في القسم التالي).

بالإضافة إلى الخصائص التي لها أسماء، تعتمد البيانات الوصفية كثيراً على مفهوم «المجال» (scope). أبسط طريقة تستطيع تخيل المجالات في البيانات الوصفية هي أن تخيل

علاقة «الأب-الابن» بين عناصر DOM. العنصر `<head>` يحتوي عادةً على عنصرين: `<html>` و `<body>`. العنصر `<body>` يحتوي عادةً على عدّة أبناء وقد يحتوي كلًّ منها على أبناء خاصة به. فمثلاً قد تحتوي صفحة الويب على عنصر `<h1>` موجودٍ ضمن عنصر `<hgroup>` موجودٍ داخل عنصر `<header>` الموجود داخل عنصر `<body>`. وقد يحتوي جدولً ما على خلية `<td>` موجودة ضمن `<tr>` الموجود ضمن `<table>` (الذي يتفرّع من `<body>`). تُعيّد البيانات الوصفية استخدام هذه البنية الهيكلية لشجرة DOM لتوفير طريقة لقول «جميع الخصائص ضمن هذا العنصر مأخوذة من نوع الاصطلاحات المُحدّد». وهذا يسمح لك باستخدام أكثر من نوع من أنواع الاصطلاحات في البيانات الوصفية في الصفحة نفسها. تستطيع أيضًا أن تضع نوعًا من أنواع اصطلاحات البيانات الوصفية ضمن نوعٍ آخر، وذلك عبر إعادة استخدام البنية الهيكلية لشجرة DOM (سأريك عدّة أمثلة عن تداخل أنواع الاصطلاحات في هذا الفصل).

تحدثت سابقًا عن موضوع DOM، لكن دعني أستفيض قليلاً فيه. مهمة البيانات الوصفية هي إضافة مزيدٍ من الهيكلية إلى البيانات الظاهرة في صفحة الويب. ليس الغرض من البيانات الوصفية أن تكون صيغة بياناتٍ تعمل بمفرداتها، وإنما هي مكملةٌ لغة HTML وتعتمد عليها. وكما سترى في القسم التالي، تعمل البيانات الوصفية بأفضل صورة ممكنة عندما تستعمل HTML استعمالاً سليماً، لكن اصطلاحات HTML غير كافية للتعبير عن كل ما نريده، لذلك أتت البيانات الوصفية (microdata) للتحكم الدقيق في الثنائي الهيكلية للبيانات الموجودة في شجرة DOM. إذا كانت البيانات التي تحاول توصيفها غير موجودةٍ في شجرة DOM، فربما عليك أن تتراجع وتعيد التفكير فيما إذا كانت البيانات الوصفية هي الحل الصحيح لمشكلتك.

هل أصبحت هذه الجملة واضحةً الآن؟ «تُوَضِّفُ البيانات الوصفية شجرة DOM بثنائيات على شكل «الاسم/القيمة» آتيةً من أنواع اصطلاحاتٍ مُخْصَصة». أرجو ذلك، وسنرى تطبيقات عمليةً عليها في بقية هذا الفصل.

2. النموذج الهيكلي للبيانات الوصفية

تعريف نوع اصطلاحات البيانات الوصفية الخاص بك سهلٌ. تحتاج أولاً إلى مجال أسماء (namespace) الذي هو رابط URL. رابط URL لمجال الأسماء يمكن أن يُشير إلى صفحة ويب موجودة، لكن هذا ليس ضروريًا. نقل أنني أريد إنشاء نوع اصطلاحات لوصف «شخص ما». إذا كنت أملك النطاق <http://schema.org/Person> فسأستعمل رابط URL الآتي كمجال أسماء لنوع اصطلاحات البيانات الوصفية. هذه طريقةٌ سهلةٌ لإنشاء معرفٍ فريدٍ عالمي: اختر رابط URL في نطاقٍ تملكه.

سأحتاج إلى تعريف بعض الخصائص في نوع الاصطلاحات، لنبدأ بثلاث خصائص أساسية:

- name (اسمك الكامل)
- image (رابطٌ لصورةٍ لك)
- url (رابطٌ لموقعٍ يتعلّق بك، مثل مدونتك أو حسابك على Google+)

بعض الخصائص السابقة هي روابط URL، وبعضاًها الآخر مجرد نص بسيط. وترتبط كلٌ واحدةٌ منها نفسها بنوعٍ معينٍ من الشيفرات، وحتى قبل أن تبدأ بالتفكير عن البيانات الوصفية أو الاصطلاحات أو إلى ما هنالك... تخيل أنّ لديك صفحة لحساب المستخدم أو صفحة «About»، من المحتمل أنّك ستضع اسمك كترونيسة (ربما في عنصر `<h1>`)، أما صورتك فهي عنصر `` ذلك لأنّك تريده أن يراها زوار صفحتك، وستوضع أيّة روابط URL مرتبطة بك في عناصر

`<a>` ذلك لأنك تريد أن يتمكن زوار صفحتك من النقر عليها. ولنقل أيضًا أنَّ كامل معلوماتك الشخصية موجودة في عنصر `<section>` لفصلها عن بقية محتويات الصفحة. إِذًا:

```
<section>
  <h1>Mark Pilgrim</h1>
  <p></p>
  <p><a href="http://diveintomark.org/">weblog</a></p>
</section>
```

النموذج الهيكلی للبيانات الوصفية هو ثنائيات على شكل «الاسم/القيمة». يُعرَّف اسم الخاصية التي تتبع لبيانات الوصفية (مثل `name` أو `src` في مثالنا) دومًا ضمن عنصر HTML. ثم تؤخذ قيمة تلك الخاصية من شجرة DOM للعنصر. ولأغلبية عناصر HTML، تكون قيمة الخاصية هي المحتوى النصي للعنصر، لكن هناك عدد لا يأس به من الاستثناءات.

العنصر	القيمة	الخاصية
<code><meta></code>		<code>content</code>
<code><audio></code>		<code>src</code>
<code><embed></code>		
<code><iframe></code>		
<code></code>		
<code><source></code>		
<code><video></code>		
<code><a></code>		<code>href</code>
<code><area></code>		
<code><link></code>		
<code><object></code>		<code>data</code>

الخاصية <code>datetime</code>	<code><time></code>
المحتوى النصي	جميع العناصر الأخرى

«إضافة البيانات الوصفية» إلى صفحتك هي مسألة إضافة بعض الخاصية إلى عناصر HTML التي لديك. أول شيء عليك فعله هو التصريح عن نوع الاصطلاحات الذي ستستخدمه، وذلك عبر الخاصية `itemtype`، أما الشيء الثاني الذي عليك دائمًا فعله هو التصريح عن مجال (scope) نوع الاصطلاحات، وذلك عبر الخاصية `itemscope`. جميع البيانات التي نريد هيكلتها في المثال الآتي موجودة في عنصر `<section>`، لهذا سنُعرّف الخاصيتين `itemtype` و `itemscope` في العنصر `<section>`.

```
<section itemscope itemtype="http://schema.org/Person">
```

اسمك هو أول المعلومات الموجودة ضمن عنصر `<section>`، وهو موجود ضمن عنصر `<h1>`، وعنصر `<h1>` لا يملك أي معنى خاص في **النموذج الهيكلی للبيانات الوصفية في HTML5**، لذلك سُيصنَّف تحت بند «جميع العناصر الأخرى» حيث تكون قيمة الخاصية هي المحتوى النصي الموجود ضمن العنصر (سيعمل ما سبق بشكلٍ مماثل إن كان اسمك موجوداً ضمن عنصر `` أو `<div>` أو `<p>`).

```
<h1 itemprop="name">Mark Pilgrim</h1>
```

السطر السابق يقول بالعربية: «هذه هي خاصية `name` لنوع الاصطلاحات `http://schema.org/Person`، وقيمة تلك الخاصية هي «Mark Pilgrim».

الخاصية التالية هي خاصية `image`, التي من المفترض أن تكون رابط URL, ووفقاً للنموذج **الهيكل للبيانات الوصفية في HTML5**, «قيمة» خاصية البيانات الوصفية الموجودة في العنصر `` هي قيمة الخاصية `src`, لكن لاحظ أنَّ رابط URL لصورتك الشخصية موجودٌ في خاصية ``, فكل ما علينا فعله هو التصريح أنَّ العنصر `` يُمثل خاصية `image`.

```
<p></p>
```

السطر السابق يقول بالعربية: «هذه هي قيمة خاصية `image` لنوع الاصطلاحات `http://schema.org/Person`, وقيمة تلك الخاصية هي `http://www.example.com/photo.jpg`

في النهاية, الخاصية `url` هي رابط URL أيضًا, ووفقاً للنموذج **الهيكل للبيانات الوصفية في HTML5**, «قيمة» خاصية البيانات الوصفية الموجودة في العنصر `<a>` هي قيمة الخاصية `href`, وهذا يتواافق تואفعًا مثالياً مع الشيفرة الموجودة عندك؛ فكل ما تحتاج له هو أن تقول أنَّ عنصر `<a>` الموجود مسبقًا يُمثل الخاصية `url`:

```
<a itemprop="url" href="http://diveintomark.org/">dive into
mark</a>
```

السطر السابق يقول بالعربية: «هذه هي قيمة خاصية `url` لنوع الاصطلاحات `http://schema.org/Person`, وقيمة الخاصية هي `http://diveintomark.org/`». إذا كانت شيفراتك مختلفةً قليلاً فلن تُشكّل لك مشكلةً بكل تأكيد. يمكنك إضافة خاصيات البيانات الوصفية وقيمتها إلى أي شيفرة من شيفرات HTML, حتى الشيفرات من القرن العشرين

التي كانت تستعمل الجداول لإنشاء تخطيط الصفحة. وبغض النظر عن أنني لا أُنصح بتاتاً بكتابه مثل هذه الشيفرات، لكنها للأسف ما تزال شائعةً، وما يزال بإمكاننا إضافة خصائص البيانات الوصفية إليها.

```
<TABLE>
<TR><TD>Name<TD>Mark Pilgrim
<TR><TD>Link<TD>
<A href="#
onclick=goExternalLink()>http://diveintomark.org/</A>
</TABLE>
```

أضف خاصية `itemprop` في خلية الجدول التي تحتوي على الاسم لإنشاء الخاصية `name`. خلايا الجدول لا تملك أي معنى خاص في [النموذج الهيكل للبيانات الوصفية في HTML5](#). لذلك ستكون قيمة الخاصية هي المحتوى النصي الموجود ضمن الخلية.

```
<TR><TD>Name<TD itemprop="name"

```

إضافة الخاصية `url` أصعب بقليل، إذ لا تستعمل الشيفرة السابقة العنصر `a` استعمالاً سليماً، فبدلاً من وضع رابط للصفحة الهدف في خاصية `href`، فستستعمل JavaScript والخاصية `onclick` لاستدعاء دالة (غير معروضة هنا) التي تستخلص رابط URL ثم تنتقل إليه. ولكن ثساب بالغثيان، نقل أن تلك الدالة ستفتح الرابط في نافذة منبثقة صغيرة دون شريط تمرير. ألم يكن الويب مسلياً في القرن الماضي؟

على أية حال، ما يزال متوجهاً عليك إضافة خاصية البيانات الوصفية، لكن عليك أن تكون مبدعاً قليلاً. لا يمكنك استخدام عنصر `a` إذ أنَّ الرابط الهدف ليس موجوداً في خاصية `href`، ولا توجد طريقة تستطيع فيها تجاوز القاعدة التي تقول «في العنصر `a`، ابحث عن قيمة

خاصية البيانات الوصفية في `href`، لكنك تستطيع إضافة عنصر ليحتوي الفوقي السابقة، وثضيف الخاصية `url` إليه.

```
<TABLE itemscope itemtype="http://schema.org/Person">
  <TR><TD>Name<TD>Mark Pilgrim
  <TR><TD>Link<TD>
    <span itemprop="url">
      <A href="#" onclick=goExternalLink()>http://diveintomark.org/</A>
    </span>
  </TABLE>
```

ولعدم وجود قاعدة خاصة تتطبق على العنصر `span`، فستُستعمل القاعدة الافتراضية «قيمة الخاصية هي المحتوى النصي الموجود ضمن العنصر». «المحتوى النصي» لا يعني «جميع الشيفرات داخل العنصر» (كالتي تحصل عليها عبر خاصية `innerHTML` في DOM على سبيل المثال)، وإنما تعني «النص فقط». وفي هذه الحالة يكون المحتوى النصي لعنصر `a` الموجود ضمن العنصر `span` هو `http://diveintomark.org/`.

الخلاصة: يمكنك إضافة خاصية البيانات الوصفية إلى أي شيفرة. وإذا كنت تستعمل HTML بشكلٍ صحيح، فستجد أنَّ إضافة البيانات الوصفية أسهل وأيسر فيما لو كانت شيفرة مشوهَةً، لكنك تستطيع إضافة البيانات الوصفية إليها على أية حال.

3. توصيف الأشخاص

بالمناسبة، لم أختبر الأمثلة السابقة من عندي تماماً، فهناك اصطلاحات للبيانات الوصفية للتوصيف المعلومات الخاصة بالأشخاص، ومن السهل فعل ذلك. لنلقي نظرةً أقرب. أسهل طريقة لدمج البيانات الوصفية في موقعك الشخصي تكون في صفحة «About»،

من المرجح وجود صفحة «About» لديك، أليس كذلك؟ إن لم تكن لديك صفحة، في يمكنك المتابعة معي في توسيعة صفحة About الآتية ببعض البنى الهيكيلية.

لننظر أولاً إلى الشيفرة المصدرية، قبل إضافة أية خاصيات لها علاقة بالبيانات الوصفية:

```
<section>
  

  <h1>Contact Information</h1>
  <dl>
    <dt>Name</dt>
    <dd>Mark Pilgrim</dd>

    <dt>Position</dt>
    <dd>Developer advocate for Google, Inc.</dd>

    <dt>Mailing address</dt>
    <dd>
      100 Main Street<br>
      Anytown, PA 19999<br>
      USA
    </dd>
  </dl>
  <h1>My Digital Footprints</h1>
  <ul>
    <li><a href="http://diveintomark.org/">weblog</a></li>
    <li><a href="http://www.google.com/profiles/pilgrim">Google
profile</a></li>
    <li><a
      href="http://www.reddit.com/user/MarkPilgrim">Reddit.com
profile</a></li>
    <li><a
      href="http://www.twitter.com/diveintomark">Twitter</a></li>
  </ul>
</section>
```

أول شيء عليك فعله دائمًا هو التصريح عن نوع الاصطلاحات الذي ستستعمله، ومجال `itemtype` (scope) الخاصيات التي تريد إضافتها. يمكنك القيام بذلك عبر إضافة خاصيّة `itemtype` إلى العنصر الأب الذي يحتوي على بقية العناصر التي تريد توصيف البيانات فيها، وهو في حالتنا العنصر `<section>`.

```
<section itemscope itemtype="http://schema.org/Person">
```

يمكنك الآن البدء بتعريف خاصيات البيانات الوصفية من نوع الاصطلاحات، لكن ما هي هذه الخاصيات؟ كما هو واضح، تستطيع رؤية كامل قائمة الخاصيات بزيارة الصفحة <http://schema.org/Person> في متصفحك. لا تتطلب مواصفة البيانات الوصفية أن توضع قائمة الخاصيات في تلك الصفحة، لكنني أرى أن ذلك مستحسن. فلو أردت مثلاً أن يجعل المطورين يستعملون نوع اصطلاحات البيانات الوصفية الذي أنشأته، فستحتاج إلى توثيقه. ولا يوجد مكان أفضل لوضع التوثيق فيه من رابط نوع الاصطلاحات نفسه، أليس كذلك؟

الخاصية	الشرح
name	الاسم
additionalName	الاسم الإضافي، قد يكون الاسم الأوسط أو اللقب
image	رابط لصورة له
jobTitle	المُسَفِّـي الوظيفي (مثلاً، مدير مالي [Financial Manager])
url	رابط URL لصفحة ويب، مثل الصفحة الرئيسية لمدونة ذاك الشخص
affiliation	المنظمة التي يرتبط بها هذا الشخص (أن يكون مثلاً موظفاً أو طالباً فيها)

العنوان الفيزيائي للشخص. يمكن أن يحتوي على خاصيات أخرى مثل postalCode و addressRegion و addressLocality و streetAddress و addressCountry و	address
علاقة اجتماعية بين الشخص الموصوف وشخص آخر	knows

أول شيء نصادفه في صفحة «About» السابقة هي صورةً موضوعةً ضمن عنصر ``، ولكن نُصرّح أنَّ الصورة الموجودة هي صورة الشخص الموصوف، فكل ما نحتاج له هو إضافة `.`.

```

```

أين هي قيمة خاصية البيانات الوصفية؟ إنها موجودةً في خاصية `src`، وإذا كنت تتذكر من **النموذج الهيكلي للبيانات الوصفية في HTML5**، «قيمة» خاصية البيانات الوصفية في عنصر `` هي محتوى الخاصية `src`. وكل عنصر `` خاصية `src` -وإلا فلن تُعرض الصورة- وخاصية `src` تحتوي على رابط URL دائمًا، أترى؟ إذا كنت تكتب HTML بشكلٍ صحيح، فاستعمال البيانات الوصفية سهلٌ جدًا.

علاوةً على ذلك، العنصر `` ليس موجودًا لوحده في الصفحة، فهو عنصر ابن للعنصر `<section>`، الذي عُرِفناه مع الخاصية `itemscope`. تُعيَّد البيانات الوصفية استعمال علاقَة `«الأب-الابن»` بين العناصر في الصفحة لتعريف مجال (`scope`) خاصيات البيانات الوصفية. أي أننا نقول بالعربية: «العنصر `` يُمثل شخصًا، وأية خاصيات للبيانات الوصفية التي تجدها في العناصر التي تكون أباً للعنصر `` هي خاصياتٌ تابعةً لذاك الشخص».

يمكنك تخيل الأمر على أنَّ العنصر `<section>` هو الفاعل في الجملة، والخاصية `itemprop` تمثِّل الفعل (وهو يُشَبِّه: صُورَ في»)، وقيمة خاصية البيانات الوصفية هي المفعول به في الجملة.

هذا الشخص [من <`<section itemscope itemtype="...>`

صُورَ في [من <``

الصورة `http://diveintohtml5.org/examples/2000_05_mark.jpg`

[من خاصية ``

يجب تعريف «الفاعل» مرةً واحدةً فقط، وذلك بوضع الخاصيتين `itemscope` و `itemtype` في عنصر `<section>` الأب. أما «ال فعل» فيُعرَف بوضع الخاصية `itemprop="image"` في عنصر ``. أما «المفعول به» فلا يحتاج إلى أيَّة شيفرات خاصة، لأنَّ **النموذج الهيكلي** يقول أنَّ قيمة خاصية البيانات الوصفية في عنصر `` في خاصية `src`. سنتنتقل الآن إلى القسم التالي من الشيفرة، سنشاهد ترويسة `<h1>` وبداية قائمة `<dl>`. ليس من الضروري إضافة خاصيات البيانات الوصفية إلى عنصري `<h1>` و `<dl>`، فلا حاجة إلى وضع خاصية من خاصيات البيانات الوصفية في كل عنصر من عناصر HTML. الفرض من البيانات الوصفية هو «توصيف» البيانات وليس الشيفرات أو الترويسات التي تحيط بها. ترويسة `<h1>` لا تحتوي على قيمة، فهي مجرد ترويسة. وكذلك الأمر لعنصر `<dt>` الذي يحتوي على السلسلة النصية «Name» التي لا تمثل خاصية، وإنما لافتة (`label`) فقط.

```
<h1>Contact Information</h1>
<dl>
  <dt>Name</dt>
  <dd>Mark Pilgrim</dd>
```

أين توجد المعلومات الحقيقية؟ في عنصر `<dd>`، وهناك ستحتاج إلى وضع خاصية `itemprop`، لكن أي خاصية منها؟ إنها خاصية `name`، وأين قيمة الخاصية؟ هي النص الموجود ضمن العنصر `<dd>`، لكن لا تحتاج إلى وضع القيمة في شيفرة خاصة؟ **المونوج الهيكلية للبيانات الوصفية في HTML5** يقول لا، فلا يوجد معنى خاص لعناصر `<dd>`، وستكون قيمة الخاصية هي النص الموجود ضمن العنصر.

```
<dd itemprop="name">Mark Pilgrim</dd>
```

كيف نستطيع التعبير عما سبق بالعربية؟ «اسم هذا الشخص هو Mark Pilgrim». حسناً، لنتابع.

إضافة الخصائص التاليتين صعب قليلاً، هذه هي الشيفرة قبل إضافة البيانات الوصفية:

```
<dt>Position</dt>
<dd>Developer advocate for Google, Inc.</dd>
```

إذا نظرت إلى نوع اصطلاحات Person، فستجد أن النص «Developer advocate for Google, Inc.» يحتوي على خصائصتين: `jobTitle` (قيمتها `Developer advocate`) و `affiliation` (قيمتها `Google, Inc.`)، لكن كيف تستطيع أن تُعبر عن ذلك عبر البيانات الوصفية؟

الجواب المختصر: لا يمكنك فعل ذلك. لا توجد طريقة في البيانات الوصفية تمكّنك من تقسيم سلسلة نصية إلى عدّة خاصيات. لا يمكنك القول «أول 18 حرفاً من هذه السلسلة النصية هي خاصية بياناتٍ وصفية، وآخر 12 حرفاً هي خاصية أخرى».

لكن هذا لا يعني أنَّ الأمر مستحيل. تخيل أنك تريد أن تُنسق النص «Developer» بنوع خطيٍ مختلف عن النص «Google, Inc.». حسناً، CSS لا تستطيع فعل ذلك أيضاً، لكن ماذا كنت ستفعل؟ ستحتاج أولاً إلى وضع كل قسم من السلسلة النصية في حاويات مختلفة، مثل ``، ثم تطبيق أنماط CSS على كل عنصر `` على حدة. يمكنك تطبيق هذه التقنية أيضاً على البيانات الوصفية، فهناك معلوماتان منفصلتان هنا: إذا وضعت كل معلومة في عنصر ``، فستستطيع القول أنَّ كل عنصر `` هو خاصيةٌ مستقلةٌ من خاصيات البيانات الوصفية.

```
<dt>Position</dt>
<dd><span itemprop="jobTitle">Developer advocate</span> for
      <span itemprop="affiliation">Google, Inc.</span></dd>
```

هذا يعني: «وظيفة هذا الشخص هي "Developer advocate". هذا الشخص يعمل لدى Google, Inc.». تلك جملتان، وخاصيتاً بياناتٍ وصفية. صحيح أننا وضعنا مزيداً من الشيفرات، لكننا استفدنا منها خيرً استفادة.

سنستفيد أيضاً من نفس التقنية لتوصيف معلومات العنوان، يُعرّف نوع الاصطلاحات `address`، التي هي بدورها عنصراً من عناصر البيانات الوصفية، وهذا يعني أنَّ `Person` للعنوان نوع اصطلاحاتٍ خاصٍ به (`http://schema.org/PostalAddress`) ، وله خاصيات متعلقةٌ به.

يُعرّف نوع الاصطلاحات `PostalAddress` خمس خاصياتٍ:

`.addressCountry` و `postalCode` و `addressRegion` و `addressLocality` و

إذا كنت مبرمجاً، فمن المرجح أنك تعرف كيف تفصل النقطة بين الكائنات وخاصياتها،

تخيل أن العلاقة كالتالي:

- `Person`
- `Person.PostalAddress`
- `Person.PostalAddress.streetAddress`
- `Person.PostalAddress.addressLocality`
- `Person.PostalAddress.addressRegion`
- `Person.PostalAddress.postalCode`
- `Person.PostalAddress.addressCountry`

لنعد إلى مثالنا. العنوان بأكمله موجود في عنصر `<dd>` وحيد (أكّر مرّة أخرى أن العنصر

`<dt>` هو لافتة، ولا يلعب دوراً في إضافة معلومات إلى البيانات الوصفية). من السهل الإشارة إلى خاصية `address`، كل ما عليك فعله هو إضافة الخاصية `itemprop="address"` في عنصر `<dd>`.

```
<dt>Mailing address</dt>
<dd itemprop="address">
```

لكن تذَّكر أنَّ خاصية `address` هي بدورها عنصرٌ من عناصر البيانات الوصفية، هذا يعني

أننا نحتاج إلى وضع الخصيتيين `itemscope` و `itemtype` أيضاً.

```
<dt>Mailing address</dt>
<dd itemprop="address" itemscope
    itemtype="http://schema.org/PostalAddress">
```

لقد رأينا هذا من قبل، لكن للعناصر من المستوى الأول (top-level). عنصر `<section>`

يحتوي على `itemscope` و `itemtype`، وجميع العناصر الموجودة ضمن العنصر `<section>`

التي لديها خاصيات للبيانات الوصفية هي ضمن «مجال» (scope) نوع اصطلاحات البيانات الوصفية. لكن هذه هي أول مرة نرى فيها «تشقّب» المجالات، أي تعريف itemtype و itemscope (في عنصر `<dd>`) داخل مجال موجود مسبقاً (في عنصر `<section>`). المجالات المتشعبة تعمل تماماً كما تعمل شجرة DOM في HTML. العنصر `<dd>` يحتوي على عددٍ معينٍ من العناصر الأبناء، ويكون مجالها هو نوع الاصطلاحات المعرف في العنصر `<dd>`، وبعد أن ينتهي العنصر `<dd>` عبر وسم الإغلاق `</dd>` فسيرجع المجال إلى نوع الاصطلاحات المعرف في العنصر الأب (الذي هو `<section>` في حالتنا).

تعاني خاصيات العنوان من المشكلة نفسها التي واجهناها عند تعريف الخصيتيين affiliation و jobTitle، لكن السلسلة النصية للعنوان أطول قليلاً، وعليينا تقسيمها إلى خمس خاصيات للبيانات الوصفية. وسنستعمل الآلية نفسها التي اتبناها سابقاً: وضع كل قطعة من المعلومات في عنصر ``، ثم توصيف تلك المعلومات عبر خاصيات البيانات الوصفية.

```
<dd itemprop="address" itemscope
      itemtype="http://schema.org/PostalAddress">
    <span itemprop="streetAddress">100 Main Street</span><br>
    <span itemprop="addressLocality">Anytown</span>,
    <span itemprop="addressRegion">PA</span>
    <span itemprop="postalCode">19999</span>
    <span itemprop="addressCountry">USA</span>
  </dd>
</dl>
```

بالعربية: «هذا الشخص يملك عنواناً بريدياً. اسم الشارع لذاك العنوان البريدي هو "Main Street"، أما البلدة (locality) فهي "Anytown" ، والإقليم (region) هو "PA" ، والرمز البريدي (postal code) هو "19999" ، واسم الدولة هو "USA" .»

س: هل صيغة العنوان البريدي خاصةً بالولايات المتحدة؟

ج: لا. خاصيات نوع الاصطلاحات `PostalAddress` عامةً لتمكن من وصف أي عنوان بريدي في العالم. لكن لن يكون لجميع العناوين قيمةً لكل خاصية من الخصائص، ولا بأس بهذا؛ وقد تتطلب بعض العناوين وضع أكثر من «سطر» واحد في خاصية معينة، ولا بأس بهذا أيضًا. فمثلاً، لو كان يحتوي العنوان البريدي على عنوان الشارع `:streetAddress` ورقم البناء، فسيتمثل كلاهما الخاصية

```
<p itemprop="address" itemscope
    itemtype="http://schema.org/PostalAddress">
<span itemprop="streetAddress">
    100 Main Street
    Suite 415
</span>
...
</p>
```

بقي شيءٌ آخرٌ في صفحة «About»: قائمةُ بروابط URL. لدى نوع الاصطلاحات `Person` خاصيةً لهذا الغرض اسمها `url`، التي يمكن أن تحتوي على أي نوعٍ من أنواع الروابط (المهم أن يكون «رابطًا»).

ما أقصده هو أنَّ تعريف الخاصية `url` غير محدد، ويمكن أن تحتوي على أيّة روابط متعلقة بالشخص: مدونة، أو معرض صور، أو حساب شخصي على موقعٍ آخر مثل فيسبوك أو تويتر. من المهم أن تلاحظ أنَّ الشخص الواحد قد يمتلك أكثر من خاصية `url`. تقلياً، يمكن لأي خاصية أن تتكرر، لكن إلى الآن لم تستفد من هذا. فمثلاً قد يكون لديك أكثر من خاصية `image` تشير إلى روابط URL لصورتين مختلفتين. أريد هنا أن أذكر أربعة روابط URL مختلفة: المدونة، وحساب Google، وحساب Reddit، وحساب تويتر. هناك قائمة في HTML فيها أربعة روابط موجودة في أربعة عناصر `<a>`، كلُّ واحدٍ منها موجودٌ في عنصر `` خاص به. سُتضيف الخاصية "url" إلى كل عنصر من عناصر `<a>`.

```

<h1>My Digital Footprints</h1>
<ul>
  <li><a href="http://diveintomark.org/">
    itemprop="url">weblog</a></li>
  <li><a href="http://www.google.com/profiles/pilgrim">
    itemprop="url">Google profile</a></li>
  <li><a href="http://www.reddit.com/user/MarkPilgrim">
    itemprop="url">Reddit.com profile</a></li>
  <li><a href="http://www.twitter.com/diveintomark">
    itemprop="url">Twitter</a></li>
</ul>

```

وفقاً للنموذج الهيكلي للبيانات الوصفية في HTML5، سيُعامل العنصر <a> معاملة خاصةً.

فقيمة خاصية البيانات الوصفية تؤخذ من الخاصية href، وليس من المحتوى النصي للعنصر.

وسيتم تجاهل المحتوى النصي لكل رابط من قبل مفسّر البيانات الوصفية. وهذا يعني

-بالعربية-: «هذا الشخص لديه رابط URL في http://diveintomark.org وهذا الشخص

لديه رابط URL آخر في http://www.google.com/profiles/pilgrim وهذا الشخص لديه

رابط URL آخر في http://www.reddit.com/user/MarkPilgrim وهذا الشخص لديه

رابط URL آخر في http://www.twitter.com/diveintomark.

١. التعرف على المقتطفات المنسقة

ربما تتساءل : «لماذا نفعل هذا؟» هل نضيف البنى الهيكلية عبّا؟ لماذا نأبه للبيانات

الوصفية ونستعملها؟

هناك نوعان رئيسيان من التطبيقات التي تستخدم HTML، وبطريقها تستخدم البيانات

الوصفية أيضاً:

١. متصفحات الويب

٢. محركات البحث

أما للمتصفحات، فهناك واجهة برمجية في DOM لاستخلاص عناصر البيانات الوصفية وخاصياتها وقيم تلك الخاصيات من صفحة الويب، لكن للأسف هذه الواجهة البرمجية غير مدعومة إلا من الإصدارات الحديثة لبعض المتصفحات، لهذا اعتبر أن هذا الطريق مسدود إلى أن تدعم جميع المتصفحات هذه الواجهة البرمجية.

مستهلك آخر لشيفرات HTML هو محركات البحث. ماذا يمكن لمحركات البحث فعله مع خاصيات البيانات الوصفية التي تتحدث عن شخص ما؟ تخيل هذا: بدلاً من عرض عنوان الصفحة ومُلخص عن محتواها، فسيعرض محرك البحث بعض المعلومات الهيكيلية الموجودة فيها، مثل الاسم الكامل، والمسمى الوظيفي، والشركة التي يعمل بها، والعناوين، وربما سيعرض أيضاً صورةً مصغرّةً له. هل جذب ذلك انتباحك؟

يدعم محرك البحث Google البيانات الوصفية كجزء من برنامج «المقتطفات المنسقة» (Rich Snippets)، فعندما يُفسر عنobot البحث في Google صفحتك ويجد خاصيات للبيانات الوصفية التي تتطابق مع نوع الاصطلاحات <http://schema.org/Person>، فسيحاول تفسير تلك الخاصيات ويُخزن قيمها بجانب بقية بيانات الصفحة. لدى Google أداة رائعة لكي ترى كيف «يرى» Google خاصيات البيانات الوصفية في صفحتك، واختبارها على صفحة About التي نعمل عليها سيعطي النتيجة:

▼ Person (1)

All good ✓

Person

image:	http://diveintohtml5.org/examples/2000_05_mark.jpg
---------------	--

name:	Mark Pilgrim
--------------	--------------

jobTitle:	Developer advocate
------------------	--------------------

url:	http://diveintomark.org/
-------------	--------------------------

url:	http://www.google.com/profiles/pilgrim
-------------	--

url:	http://www.reddit.com/user/MarkPilgrim
-------------	--

url:	http://www.twitter.com/diveintomark
-------------	-------------------------------------

affiliation [Organization]:

name:	Google, Inc.
--------------	--------------

address [PostalAddress]:

streetAddress:	100 Main Street
-----------------------	-----------------

addressLocality:	Anytown
-------------------------	---------

addressRegion:	PA
-----------------------	----

postalCode:	19999
--------------------	-------

addressCountry [Country]:

name:	USA
--------------	-----

الشكل 48: معلومات البيانات الوصفية كما تُظهرها أداة اختبار البيانات المنظمة

كل البيانات الوصفية موجودة هنا: خاصية `image` من `<img src=`, جميع روابط URL من قائمة عناصر `a`, وحتى كائن العنوان (مذكور في «address») والخاصيات الخمس المتعلقة به.

الآن، كيف يستعمل Google كل هذه المعلومات؟ الأمر نسبي، فلا توجد قواعد ملزمة لكيفية عرض خاصيات البيانات الوصفية، ولا أيّها سيُعرض، وحتى لا توجد قواعد تحكم إذا كانت ستُعرض هذه الخاصيات أم لا. إذا بحث أحدهم عن «Mark Pilgrim» ورأى Google صفحة «About» تستحق الظهور في نتائج البحث، وقرر Google أنّ خاصيات البيانات الوصفية الموجودة في تلك الصفحة تستحق أن تُعرض، فعندها ستبدو نتيجة البحث مشابهةً لما يلي:

About Mark Pilgrim

Anytown PA - Developer advocate - Google, Inc.

Excerpt from the page will show up here.

Excerpt from the page will show up here.

diveintohtml5.org/examples/person-plus-microdata.html - [Cached](#) - [Similar pages](#)

الشكل 49: مثالٌ عن نتائج البحث عن صفحة فيها بياناتٌ وصفيةٌ تصف شخصاً

أول سطر «About Mark Pilgrim» هو عنوان الصفحة الموجود في عنصر `<title>`، ولكن هذا ليس أمراً مثيراً للاهتمام؛ لأن محرك Google يفعل هذا لكل صفحة، لكن السطر الثاني مليء بالمعلومات المأخوذة مباشرةً من البيانات الوصفية التي أضفناها إلى الصفحة. «Anytown PA» هو جزءٌ من العنوان البريدي، الموصوف عبر نوع الاصطلاحات «Google, Inc.» و «Developer advocate». أما `http://schema.org/PostalAddress` هما الخاصيتان من نوع الاصطلاحات `http://schema.org/Person` (الخاصية `http://schema.org/jobTitle`) على التوالي وبالترتيب).

هذا رائع! لا تحتاج إلى أن تكون شركةً كبيرةً تبرم اتفاقيات خاصةً مع شركات محركات البحث لتخصيص طريقة عرض نتائج البحث. كل ما تحتاج له هو عشر دقائق وبعض خصائص HTML لكي توضح فيها بياناتك التي ستنشرها في صفحتك.

س: فعلت كل ما قلته لي، لكن لم تغير طريقة عرض نتائج البحث عن صفحتي في Google، ما الخطأ؟
ج: «لا تضمن Google أن الشيفرة الموجودة في أيّة صفحة أو موقع سُتستخدم في نتائج البحث»، لكن بعض النظر عن قرار محرك Google لا يستعمل البيانات الوصفية في صفحتك، فقد يستعملها محرك بحث آخر. فمعيار البيانات الوصفية (Microdata) هو معيارٌ مفتوحٌ يستطيع أيّ شخصٍ توظيفه - كما في بقية أجزاء تقنية HTML5-. من واجبك توفير أكبر قدرٍ من البيانات تستطيع تقديمها. ثم اترك الأمر للآخرين لكي يقرّروا ماذا يفعلون معها. ربما يفاجئوك!

4. توصيف المنظمات

البيانات الوصفية ليست محدودةً لنوع اصطلاحاتٍ وحيد. صفحة «About» جيدة، لكن من المرجح أنَّ لديك صفحةً واحدةً اسمها «About»، ولكنك متغطش للمزيد؟ لنتعلم كيف نوَّضف المنظمات والشركات.

هذه نموذجٌ لصفحة شركةٍ ما، لتلقِ نظرةً على شيفرة HTML دون البيانات الوصفية.

```
<article>
  <h1>Google, Inc.</h1>
  <p>
    1600 Amphitheatre Parkway<br>
    Mountain View, CA 94043<br>
    USA
  </p>
  <p>650-253-0000</p>
  <p><a href="http://www.google.com/">Google.com</a></p>
</article>
```

وتصُّرْفُ قصيرٌ ومنظمٌ، فجميع المعلومات حول هذه المنظمة موجودةٌ ضمن عنصر `<article>`.

```
<article itemscope itemtype="http://schema.org/Organization">
```

وكما فعلنا عند توصيف الأشخاص، سنحتاج إلى ضبط الخصيَّتين `itemscope` و `itemtype` في العنصر الحاوي لبقية العناصر، وهو في حالتنا العنصر `<article>`. خاصية `itemtype` تُصرِّحُ ما هو نوع الاصطلاحات التي تستعملها (في هذه الحالة `http://schema.org/Organization`)، وخاصية `itemscope` تُصرِّحُ أنَّ كل الخصيَّات التي تضبطها للعناصر الأبناء للعنصر الحالي ترتبط بنوع الاصطلاحات هذا.

إذًا، ماذا يوجد في نوع الاصطلاحات Organization؟ بعض خاصياتٍ بسيطة، التي يكون بعضها مأولاًً لديك.

الخاصية	الشرح
name	اسم المنظمة (على سبيل المثال: «Initech»)
url	رابط URL لصفحة ويب، مثل الصفحة الرئيسية للمنظمة
address	العنوان الفيزيائي للمنظمة، يمكن أن يحتوي على خاصيات أخرى مثل streetAddress و addressCountry و postalCode و addressRegion و addressLocality و
telephone	رقم هاتف المنظمة
geo	تحديد الإحداثيات الجغرافية لموقع المنظمة، ويملك خاصيتين دائمًا: latitude و longitude .

أول قطعة من الشيفرات في عنصر `<h1>` هي `<article>`. يحتوي عنصر `<h1>` على اسم الشركة، ولهذا سأضيف خاصية `itemprop="name"` إليه مباشرةً.

```
<h1 itemprop="name">Google, Inc.</h1>
```

ووفقًا للنموذج الهيكلي للبيانات الوصفية في HTML5، عناصر `<h1>` ليس لها معالجة خاصة، وستكون قيمة خاصية البيانات الوصفية هي المحتوى النصي للعنصر. أي أننا قلنا بالعربية: «اسم المنظمة هو "Google, inc."».

المعلومة التالية التي نريد توصيفها هي العنوان. توصيف عنوان المنظمة مماثل تماماً لـ **توصيف عنوان شخص ما**. أضف أولًا خاصية `itemprop="address"` إلى العنصر الذي يحتوي على العنوان (العنصر `<p>` في هذه الحالة). وهذا يُصرّح أنَّ هذه هي خاصية `address` للمنظمة، لكن ماذا عن خاصيات العنوان نفسه؟ سنحتاج إلى الخاصيتين `itemtype` و `itemscope` لكي

نقول أنَّ عنصر العنوان هذا له خاصيَّاتٌ تابعةٌ له.

```
<p itemprop="address" itemscope
    itemtype="http://schema.org/PostalAddress">
```

في النهاية، علينا وضع كل قطعة من المعلومات في عنصر `` لكي نتمكن من وضع `addressRegion` و `addressLocality` و `streetAddress` و `addressCountry` و `postalCode` في كل عنصر من تلك العناصر.

```
<p itemprop="address" itemscope
    itemtype="http://schema.org/PostalAddress">
    <span itemprop="streetAddress">1600 Amphitheatre
    Parkway</span><br>
    <span itemprop="addressLocality">Mountain View</span>,
    <span itemprop="addressRegion">CA</span>
    <span itemprop="postalCode">94043</span><br>
    <span itemprop="addressCountry">USA</span>
</p>
```

بالعربية: «هذه المنظمة تملك عنواناً بريدياً. اسم الشارع لذاك العنوان البريدي هو "1600 Amphitheatre Parkway" ، أما البلدة (locality) فهي "Mountain View" ، والإقليم (region) هو "CA" ، والرمز البريدي (postal code) هو "94043" ، واسم الدولة هو "USA" .»

الخطوة التالية هي توصيف رقم الهاتف للمنظمة. بُنيةُ أرقام الهواتف معقدةٌ بعض الشيء، والصيغة المحددة لها خاصةً بكل دولة (والأمر أسوأ إذا أردت الاتصال بدولةٍ أخرى). لدينا في مثاناً رقم هاتفي من الولايات المتحدة، وهو مكتوب بصيغةٍ ملائمةٍ للاتصال من أي مكان داخل الولايات المتحدة.

```
<p itemprop="telephone">650-253-0000</p>
```

(في حال لم تنتبه، لقد خرج نوع الاصطلاحات `Address` من المجال [scope] عندماأغلق العنصر `<p>`). لقد عدنا الآن إلى تعريف الخصيّات لنوع الاصطلاحات `Organization`.)

إذا كنت تريده وضع أكثر من رقم هاتف - ربما واحد للزبائن في الولايات المتحدة وآخر للزبائن من بقية دول العالم - فيمكنك فعل ذلك. إذ يمكن تكرار أي خاصية من خصيّات البيانات الوصفية أكثر من مرة. كل ما عليك فعله هو التأكد أنّ لكل رقمٍ عنصر HTML خاص به مفصولٌ عن أيّة لافتة وضعتها له.

```
<p>
    US customers: <span itemprop="telephone">650-253-
    0000</span><br>
    UK customers: <span itemprop="telephone">00 + 1* +
    6502530000</span>
</p>
```

ووفقًا للنموذج الهيكلي للبيانات الوصفية في HTML5، عناصر `<p>` أو عناصر `` ليس لها معالجةٌ خاصة، وستكون قيمة خاصية `telephone` هي المحتوى النصي للعنصر. لا يحاول نوع الاصطلاحات `Organization` أن يقسّم ويصنّف مختلف أجزاء رقم الهاتف، فخاصية `telephone` هي نصّ عادي. حيث تستطيع وضع رمز المنطقة بين قوسين، أو أن تستعمل الفراغات بدلاً من الشرطات للفصل بين الأرقام. وإذا حاول العميل الذي يفسّر البيانات الوصفية أن يفسّر رقم الهاتف، فآلية ذلك عائدَةً تماماً إليه.

لدينا بعد ذلك خاصيةً تألفها: `url`. ومثل روابط URL المتعلقة بشخصٍ ما، يمكن لروابط URL أن تتعلق بمنظمة. وقد تكون هذه الروابط هي الصفحة الرئيسية للشركة، أو صفحة التواصل، أو صفحة المنتجات، أو أيّة صفحة أخرى. بتعبيرٍ آخر: إذا كان لديك رابط URL عن أو من أو يتعلّق بالمنظمة، فوّضه بخاصية `url`.

```
<p><a itemprop="url" href="http://www.google.com/">Google.com</a></p>
```

ووفقًا للنموذج الهيكلی للبيانات الوصفية في HTML5، عنصر `a` له معالجة خاصة، فقيمة خاصية البيانات الوصفية هي قيمة الخاصية `href`، وليس المحتوى النصي للرابط. بالعربيّة: «لدى هذه المنظمة رابط URL الآتي `http://www.google.com/`. لكن الخاصية لا تحدّد مزيدًا من المعلومات عن هذا الارتباط، ولا تضمّن السلسلة النصية للرابط `Google.com`». في النهاية، أريد الحديث عن الموقع الجغرافي. لا أقصد هنا **واجهة البرمجية** لتحديد **الموقع الجغرافي**؛ وإنما أقصد كيفية توصيف الموقع الجغرافي للمنظمات باستخدام البيانات الوصفية.

لحد هذه اللحظة، كل الأمثلة التي رأيناها ركّزت على توصيف البيانات المرئية. بعبارة أخرى، لديك عنصر `h1` فيه اسم الشركة، لذا سُتضيّف خاصية `itemprop` إلى عنصر `h1` لتصرّح أنَّ النص (المرئي) الموجود في تلك الترويسة هو اسم المنظمة. أو ربما لديك عنصر `img` الذي يُشير إلى صورةٍ ما، وتستطيع إضافة الخاصية `itemprop` لعنصر `img` لكي تصرّح أنَّ تلك الصورة (المرئية) هي صورةً لشخصٍ ما.

أما في هذا المثال، لن تكون معلومات الموقع الجغرافي على هذا النحو؛ فلا يوجد نصٌّ مرئيٌ يعطي إحداثيات الطول والعرض (بدقة أربعة أرقام عشرية!) لموقع المنظمة. وفي الواقع، **الصفحة التي نعمل عليها** لا تحتوي على معلومات عن الموقع الجغرافي إطلاقاً. وحتى لو وجد رابطُ إلى خرائط Google، لكنه لا يحتوي على إحداثيات الطول والعرض (يحتوي على معلومات مشابهة في صيغةٍ خاصةٍ بخرائط Google). لكن حتى لو افترضنا وجود رابط URL لإحدى خدمات الخرائط التي تضع إحداثيات خطوط الطول والعرض كوسائل في رابط URL، فلا توجد

طريقة في البيانات الوصفية لفصل أجزاء URL عن بعضها؛ فلا تستطيع أن تقول أن «أول وسيط في طلبية URL هو إحداثيات الطول والوسط الثاني هو إحداثيات العرض وبقية وسائل الطلبية غير مهمة بالنسبة لنا».

توفر HTML5 طريقة لتوصيف البيانات غير المرئية للتعامل مع الحالات الخاصة مثل هذه الحالة. لا تستعمل هذه التقنية إلا ملائماً أخيراً لك. فلو كانت هنالك طريقة لعرض البيانات التي تريد توصيفها، فافعل ذلك. أرى أن البيانات المخفية التي تستطيع «الآلات» قراءتها فقط «ستتعفن» بسرعة. وهذا يعني أن أحدhem سيأتي عاجلاً أم آجلاً وسيحدث النص المرئي الموجود في الصفحة لكنه سينسى تحديث البيانات غير المرئية. وهذا يحدث أكثر مما تتوقع. لكن مع هذا، هنالك حالات لا نستطيع فيها تجنب استخدام البيانات المخفية. ربما يريد رئيسك في العمل وضع معلومات للموقع الجغرافي لكنه لا يريد أن تعم الفوضى في الواجهة بوجود زوجين من الأرقام غير المفهومة ذات ست خانات. الخيار الوحيد الذي أمامك هنا هو البيانات المخفية. كل ما تستطيع فعله هنا هو وضع البيانات المخفية بعد النص المرئي الذي يصفها مباشرةً، لعل ذلك يذكر الشخص الذي أتى ليحدث النص المرئي لكي يحدث البيانات المخفية بعده مباشرةً.

أنشأنا في هذا المثال عنصر `` ضمن عنصر `<article>` الذي يحوي بقية خاصيات المنظمة، ثم وضعنا بيانات الموقع الجغرافي المخفية داخل عنصر `` (هذه هي الطريقة التنظيمية لنوع الاصطلاحات Organization، راجع صفحة <http://schema.org/Organization> لمزيد من المعلومات).

```
<span itemprop="areaServed" itemscope
itemtype="http://schema.org/Place">
```

```
<span itemprop="geo" itemscope
  itemtype="http://schema.org/GeoCoordinates">
  <meta itemprop="latitude" content="37.4149" />
  <meta itemprop="longitude" content="-122.078" />
</span>
</span>
</article>
```

معلومات الموقع الجغرافي مُعرَّفة في نوع الاصطلاحات الخاص بها، مثل «عنوان»

(الشخص أو المنظمة، وبالتالي، سيحتاج عنصر `` إلى ثلاثة خصائص:

.1. يُقال أن هذا العنصر يُمثل خاصية `geo` للمنظمة التي يتبع لها

.2. يُحدد أي نوع اصطلاحات `itemtype="http://schema.org/GeoCoordinates"`

ستخضع له خصائص هذا العنصر

.3. يقول أن هذا العنصر هو عنصر حاوٍ يملك نوع اصطلاحات خاص به (مُحدّد)

في خاصية `itemtype`). جميع الخصائص الموجودة ضمن هذا العنصر هي خصائص

نوع الاصطلاحات `http://schema.org/GeoCoordinates`، وليس لنوع

الاصطلاحات للعنصر الرئيسي `http://schema.org/Organization`

السؤال المهم في هذا المثال هو: «كيف تستطيع توصيف البيانات غير المرئية؟» يمكنك

استخدام العنصر `<meta>`. لم تكن في السابق تستطيع استخدام العنصر `<meta>` إلا داخل

ترويسة صفتكم; أما في HTML5، فتستطيع استخدام العنصر `<meta>` في أي مكان:

```
<meta itemprop="latitude" content="37.4149" />
```

ووفقاً **للموج الهيكلي للبيانات الوصفية في HTML5**، عنصر `<meta>` له معالجة خاصة،

فقيمة خاصية البيانات الوصفية هي قيمة الخاصية `content`; ولأن هذه الخاصية لا تُعرض أبداً،

فهي مثالية لضبط عدد غير محدود من البيانات المخفية. لكن ستزداد المسؤولية الملقاة على عاتقك هنا، إذ عليك الحرص على أن المعلومات المخفية ستبقى متوافقةً مع ما حولها من النص المرئي.

لا يوجد دعم مباشر لنوع الاصطلاحات Organization في المقتطفات المنسقة في Google، لذا لا أملك نتيجة بحث جميلة لعرضها لك. لكن لها تأثير على الأحداث (events) والمراجعات (reviews)، اللتان تدعهما المقتطفات المنسقة في Google (تقبل بعض خاصياتهما أن يكون نوع الخاصية Organization).

Organization (1)		All good ✓
Organization		
name:	Google, Inc.	
telephone:	650-253-0000	
telephone:	00 + 1* + 6502530000	
url:	http://www.google.com/	
address [PostalAddress]:		
streetAddress:	1600 Amphitheatre Parkway	
addressLocality:	Mountain View	
addressRegion:	CA	
postalCode:	94043	
addressCountry [Country]:		
name:	USA	
areaServed [Place]:		
geo [GeoCoordinates]:		
latitude:	37.4149	
longitude:	-122.078	

الشكل 50: معلومات البيانات الوصفية كما تُظهرها أداة اختبار البيانات المنظمة

5. توصيف الأحداث

تحدث بعض المناسبات في أوقاتٍ معينة، ألن يكون من الجميل أن تستطيع إخبار محركات البحث متى ستقع تلك المناسبات تحديداً؟ هناك طريقةً لفعل هذا في HTML5 لنبدأ بإلقاء نظرة على الجدول الزمني الخاص بالمحاضرات والكلمات التي سألقاها.

```
<article>
  <h1>Google Developer Day 2009</h1>
  
  <p>
```

```

Google Developer Days are a chance to learn about Google
developer products from the engineers who built them. This
one-day conference includes seminars and “office hours”
on web technologies like Google Maps, OpenSocial, Android,
AJAX APIs, Chrome, and Google Web Toolkit.

</p>
<p>
<time datetime="2009-11-06T08:30+01:00">2009 November 6,
8:30</time>
    &ndash;
<time datetime="2009-11-06T20:30+01:00">20:30</time>
</p>
<p>
    Congress Center<br>
    5th května 65<br>
    140 21 Praha 4<br>
    Czech Republic
</p>
<p><a
href="http://code.google.com/intl/cs/events/developerday/2009/h
ome.html">GDD/Prague home page</a></p>
</article>
```

جميع المعلومات التي تخص الحدث موجودة في عنصر `<article>`، وهو المكان الذي نريد وضع خاصيتي `itemscope` و `itemtype` فيه.

```
<article itemscope itemtype="http://schema.org/Event">
```

رابط URL لنوع الاصطلاحات `Event` هو <http://schema.org/Event>، الذي يحتوي أيضاً على جدولٍ يصف خصائص هذه النوع من الاصطلاحات؛ التي ذكرت بعضها في هذا الجدول.

الخاصية	الشرح
name	اسم الحدث
url	رابط لصفحة تفاصيل الحدث
location	الموقع الذي سيجري فيه الحدث الذي يمكن أن يمثل بنوع الاصطلاحات Place أو PostalAddress
description	وصف قصيري للحدث
startDate	تاريخ بدء فعاليات الحدث بصيغة ISO
endDate	تاريخ انتهاء فعاليات الحدث بصيغة ISO
duration	المدة الزمنية لفعاليات الحدث بصيغة ISO
image	صورة تعبيرية متعلقة بالحدث

اسم الحدث موجود في عنصر `<h1>`, ووفقاً [للمодèle الهيكلي للبيانات الوصفية](#) في HTML5، عنصر `<h1>` ليس له معالجة خاصة، ويستكون قيمة خاصية البيانات الوصفية هي المحتوى النصي للعنصر `<h1>`, لذا كل ما نحتاج له هو إضافة الخاصية `itemprop` لكي نُصرّح أنَّ هذا العنصر يحتوي على اسم الحدث.

```
<h1 itemprop="name">Google Developer Day 2009</h1>
```

نقول بالعربية: «اسم هذا الحدث هو Google Developer Day 2009».

يحتوي الحدث على صورة، التي يمكن توصيفها باستخدام الخاصية `image`, وكما تتوقع، ثُرَّض الصورة عبر عنصر ``, وكما في [خاصية image في نوع الاصطلاحات Person](#), الصورة في نوع الاصطلاحات Event هي رابط URL, ولأن [المودèle الهيكلي للبيانات الوصفية](#)

يقول أنَّ قيمة الخاصية الموجودة في العنصر `` هي قيمة خاصية `src`, وكل ما علينا فعله هو إضافة الخاصية `itemprop` إلى العنصر ``.

```

```

نقول بالعربية: «إحدى الصور المتعلقة بهذا الحدث موجودة في الرابط <http://diveintohtml5.org/examples/gdd-2009-prague-pilgrim.jpg>.» يأتي بعد الصورة وصفٌ مختصٌ للحدث، وهو فقرةٌ نصيةٌ من الكلام العادي.

```
<p itemprop="description">Google Developer Days are a chance
to
learn about Google developer products from the engineers who
built
them. This one-day conference includes seminars and “office
hours” on web technologies like Google Maps, OpenSocial,
Android, AJAX APIs, Chrome, and Google Web Toolkit.</p>
```

المعلومة التي تلي الوصف جديدةً علينا. تقع الأحداث عموماً في تواريخ محددة وتبعد عن انتهاء في أوقاتٍ معينة. يجب أن نضع الأوقات والتاريخ في **عنصر `time`** HTML5 وهذا ما فعلناه. لذا سيعطي السؤال الآن: كيف سنتمكن من إضافة خصائص البيانات الوصفية إلى عناصر `time` السابقة؟ بالنظر مرةً أخرى إلى **النموذج الهيكلي للبيانات الوصفية** في **HTML5**, سنلاحظ أنَّ هنالك معالجةٌ خاصةً للعنصر `time`. فقيمة خاصية البيانات الوصفية هي قيمة خاصية `datetime` في العنصر `time`. لكن مهلاً، أليست الصيغة المعيارية لخاصيات البيانات الوصفية `startDate` و `endDate` هي ISO، ممثلها كمثيل خاصية `datetime` في العنصر

. وأذكرك مجدداً كيف تتوافق وتناغم البنية الهيكيلية من أساس HTML مع البنية `<time>` الهيكيلية التي تضيفها من نوع الاصطلاحات الخاص بالبيانات الوصفية. عملية توصيف تواريخ بداية ونهاية الحدث عبر البيانات الوصفية سهلةٌ وتم كالآتي:

1. استخدام HTML استخداماً صحيحاً في المقام الأول (باستخدام عناصر `<time>` للوقت

وال تاريخ)، ومن ثم

2. إضافة خاصية `itemprop`

```
<p>
  <time itemprop="startDate" datetime="2009-11-
  06T08:30+01:00">2009 November 6, 8:30</time>
  &ndash;
  <time itemprop="endDate" datetime="2009-11-
  06T20:30+01:00">20:30</time>
</p>
```

بالعربية: «هذا الحديث يبدأ في November 6, 2009 الساعة 8:30 صباحاً، ويستمر إلى November 6, 2009 الساعة 20:30 (بتوقيت مدينة براغ المحلي، GMT+1)».

ثم ستأتي خاصية `location`, التي تقول عنها **صفحة تعريف نوع الاصطلاحات Event** أنها قد تكون من نوع `Place` أو `PostalAddress`. وفي حالتنا، سيُقام الحدث في مكانٍ متخصص بالمؤتمرات، وهو `Congress Center` في مدينة براغ. وإمكانية توصيفه على أنه «مكان» (Place) ستسمح لنا بتضمين اسمه بالإضافة إلى عنوانه.

لنبدأ بالتصريح أنّ العنصر `<p>` الذي يحتوي على العنوان هو خاصية `location` لنوع الاصطلاحات `Event`, وهذا العنصر يحتوي على خصائصٍ تابعةٍ لنوع الاصطلاحات

<http://schema.org/Place>

```
<p itemprop="location" itemscope
    itemtype="http://schema.org/Place">
```

ثم سُتَعْرِفُ اسم المكان بوضعه في عنصر `` وإضافة الخاصية `itemprop` إليه.

```
<span itemprop="name">Congress Center</span><br>
```

وبسبب قواعد المجالات (scoping rules) في البيانات الوصفية، خاصية `itemprop="name"` السابقة تُعرَّف اسم المكان في نوع الاصطلاحات `Place` وليس في نوع الاصطلاحات `Event`. وذلك لأنَّ العنصر `<p>` السابق قد صرَّح عن بداية خصائص المكان (`Place`)، ولماً يُغلَق عنصر `<p>` بعدَ عبَرَ وسم `</p>`. أيُّهُ خصائصٍ تُعرَّفُها والتي تَتَبعُ للبيانات الوصفية تكونُ من خصائص آخرِ نوع اصطلاحاتٍ دخلَ المجال. يمكن أن تتدَلَّلُ أنواع الاصطلاحات مثل **المكدس** (`stack`). لم نحذف إلى الآن آخر عنصر من المكدس، وهذا يعني أننا ما زلنا ضمن مجال نوع الاصطلاحات `Place`.

في الحقيقة، سيلزمنا إضافة نوع اصطلاحات ثالث إلى المكدس: عنوان (`Address`) للمكان

(`Event`) الذي سيُقام به الحدث (`Place`)

```
<span itemprop="address" itemscope
    itemtype="http://schema.org/PostalAddress">
```

مرةً أخرى، علينا أن نضع كل جزء من العنوان كخاصية بيانات وصفية مستقلة، لذلك علينا أن نُضيف عدًّا من عناصر `` لكي نضع خصائص `itemprop` فيها (إذا رأيتني أشرح الأمور بسرعة هنا، فأناصحك بالعودة إلى الأقسام السابقة وقراءة **كيفية توصيف العنوان للأفراد وكيفية توصيف العنوان للمنظمات**).

```
<span itemprop="address" itemscope
      itemtype="http://schema.org/PostalAddress">
  <span itemprop="streetAddress">5th května 65</span><br>
  <span itemprop="postalCode">140 21</span>
  <span itemprop="addressLocality">Praha 4</span><br>
  <span itemprop="addressCountry">Czech Republic</span>
</span>
```

لا توجد خصيّات إضافية للعنوان، لذا لنغلق عنصر `` الذي بدأ المجال الخاص بنوع

الاصطلاحات `.Address`

```
</span>
```

عليينا الآن إضافة الخاصيّة `geo` لتمثيل الموقع الفيزيائي للحدث. سنستخدم نفس نوع الاصطلاحات (`GeoCoordinates`) الذي استعملناه في القسم السابق لتحديد موقع المنظمة. ستحتاج إلى عنصر `` لكي يعمل كحاوية، والذي يملك الخاصيّتين `itemtype` و `itemscope`. و ضمن العنصر `` سنضع عنصري `<meta>`، واحدٌ لتحديد إحداثيات العرض (الخاصيّة `latitude`) والآخر لتحديد إحداثيات الطول (الخاصيّة `longitude`).

```
<span itemprop="geo" itemscope
itemtype="http://schema.org/GeoCoordinates">
  <meta itemprop="latitude" content="50.047893" />
  <meta itemprop="longitude" content="14.4491" />
</span>
```

بعد إغلاقنا لعنصر `` الذي يحوي خصيّات الموقع الجغرافي، سنعود إلى مجال `Place`، لكن لم تعد هنالك أيّة خصيّات لإضافتها إلى `Place`، لذا سُنُغلق العنصر `<p>` الذي بدأ مجال نوع الاصطلاحات `Place`، وبهذا سنعود إلى تعريف الخصيّات التابعة لنوع الاصطلاحات `.Event`.

</p>

وأخيراً وليس آخرًا، بقيت الخاصية `url`، التي يجب أن تكون مألوفةً لك. تضاف روابط URL المتعلقة بالأحداث بنفس طريقة **إضافة الروابط للأشخاص** وطريقة **إضافة روابط للمنظمات**. إذا كنت تستعمل HTML استعملاً صحيحاً (أي وضع الروابط في عنصر ``، فالآية التصريح أنَّ الروابط تمثِّلُ خاصية `url` في البيانات الوصفية بسيطةً جدًا وذلك بإضافة خاصية `itemprop="url"` إلى تلك العناصر فقط.

```
<p>
  <a itemprop="url"
    href="http://code.google.com/intl/cs/events/developerday/2009/home.html">
    GDD/Prague home page
  </a>
</p>
</article>
```

يجدر بالذكر أنَّك تستطيع توصيف أكثر من حدث في الصفحة نفسها.

ا. المقتطفات المنسقة من جديد!

وفقاً **لأداة اختبار البيانات المنظمة من Google**، المعلومات التي تحصل عليها عناكب محرك البحث من صفحة الحدث السابقة هي:

▼ Event (1)

All good ✓

Event	
name:	Google Developer Day 2009
image:	http://diveintohtml5.org/examples/gdd-2009-prague-pilgrim.jpg
description:	Google Developer Days are a chance to learn about Google developer products from the engineers who built them. This one-day conference includes seminars and “office hours” on web technologies like Google Maps, OpenSocial, Android, AJAX APIs, Chrome, and Google Web Toolkit.
startDate:	2009-11-06T08:30:00+01:00
endDate:	2009-11-06T20:30:00+01:00
url:	http://code.google.com/intl/cs/events/developerday/2009/home.html
location [Place]:	
name:	Congress Center
address [PostalAddress]:	
streetAddress:	5th května 65
postalCode:	140 21
addressLocality:	Praha 4
addressCountry [Country]:	
name:	Czech Republic
geo [GeoCoordinates]:	
latitude:	50.047893
longitude:	14.4491

الشكل 51: معلومات البيانات الوصفية كما ظهرت في أدلة اختبار البيانات المنظمة

كما لاحظت، جميع البيانات التي وضفتها موجودة هنا. لاحظ كيف ظهرت أدلة اختبار البيانات الهيكلية تشتمل أنواع الاصطلاحات؛ هذا التمثيل الرسومي سيساعدك كثيراً على تخيل الوضع.

هذه رسم توضيحي للطريقة المتوقعة لتمثيل الصفحة في نتائج بحث Google (أكرر مرةً أخرى أنَّ هذا مجرد مثال، فقد يغير Google طريقة تنسيق نتائج البحث في أي وقت، ولا توجد

هناك ضمانة أنَّ Google سُتُّفسِر البيانات الوصفية في صفحتك من الأساس!).

[Mark Pilgrim's event calendar](#)

Excerpt from the page will show up here.
Excerpt from the page will show up here.

[Google Developer Day 2009](#) Fri, Nov 6 Congress Center, Praha 4, Czech Republic
[ConFoo.ca 2010](#) Wed, Mar 10 Hilton Montreal Bonaventure, Montréal, Québec, Canada
diveintomarkt.org/examples/event-plus-microdata.html - [Cached](#) - [Similar pages](#)

الشكل 52: مثالٌ عن نتيجة البحث عن صفحة فيها بيانات وصفية تصف حدثاً

بعد عنوان الصفحة والمقططف المولَّد تلقائياً، سيبدأ محرك Google باستعمال البيانات الوصفية التي أضفناها إلى الصفحة لعرض جدول المحتويات. لاحظ طريقة تنسيق التاريخ «Fri, Nov 6» لم ترد هذه السلسلة النصية في أي مكانٍ في عناصر HTML أو خاصيات البيانات الوصفية. لكننا استخدمنا سلسلتين نصيتين هما التاريخ بصيغةٍ معياريةٍ («2009-11-06T08:30+01:00» و «2009-11-06T20:30+01:00»). أخذ Google هذين التاريخين وعَرِفَ أنَّهما في اليوم نفسه، وقرر عرض تاريخٍ وحيدٍ بصيغةٍ قراءتها أسهل.

انظر الآن إلى العنوان الفيزيائي. اختار Google أن يعرض اسم المكان + البلدة (locality) + الدولة، لكنه لم يعرض عنوان الشارع المُفضَّل. أصبح هذا ممكناً لأننا قسمنا العنوان إلى خمس خاصيات مُستقلةٍ addressRegion و addressLocality و streetAddress و addressCountry و postalCode و -استفاد Google من هذا لعرض عنوانٍ مختصرٍ للمكان. قد يختار مستهلكوَّن آخرونَ للمعلوماتِ التي توفرُها عبر البيانات الوصفية استخدامها بطريقَةٍ مختلفةٍ، إذ سيقررون ما الذي سيعرضوه وما الذي لن يعرضوه. لا يوجد خيارٌ صالحٌ وخيارٌ خطأ هنا. عليك أن توفرَ أكبرَ قدرٍ من البيانات الهيكليَّة، وبأكبر دقةٍ ممكنةٍ، واترك الباقي على الآخرين ليفسروه كيفما شاؤوا.

6. توصيف المراجعات

هذا مثالٌ آخر عن كيفية جعل الويب (وربما نتائج البحث) أفضل عبر استخدام البيانات الوصفية: مراجعات (reviews) الشركات والمنتجات. هذه مراجعة قصيرة كتبُتها لأحد مطاعم البيتزا (هذا المطعم حقيقي بالمناسبة). لننظر أولاً إلى الشيفرة الأصلية قبل إضافة البيانات الوصفية:

```
<article>
  <h1>Anna's Pizzeria</h1>
  <p>★★★☆ (4 stars out of 5)</p>
  <p>New York-style pizza right in historic downtown Apex</p>
  <p>
    Food is top-notch. Atmosphere is just right for a
    “neighborhood
    pizza joint.” The restaurant itself is a bit cramped; if
    you’re
    overweight, you may have difficulty getting in and out of
    your
    seat and navigating between other tables. Used to give free
    garlic knots when you sat down; now they give you plain
    bread
    and you have to pay for the good stuff. Overall, it’s a
    winner.
  </p>
  <p>
    100 North Salem Street<br>
    Apex, NC 27502<br>
    USA
  </p>
  <p>— reviewed by Mark Pilgrim, last updated March 31,
    2010</p>
</article>
```

هذه المراجعة موجودة ضمن عنصر `<article>`, الذي علينا وضع خاصيتي `itemtype` و `itemscope` فيه. رابط URL لمجال أسماء نوع الاصطلاحات الذي سنستعمل هو <http://schema.org/Review>

```
<article itemscope itemtype="http://schema.org/Review">
```

ما هي الخصيّات الموجودة في نوع الاصطلاحات `Review`? أنا ممتن لسؤالك.

الخاصية	الشرح
<code>itemReviewed</code>	اسم العنصر الذي ستتم مراجعته. يمكن أن يكون منتجًا أو خدمةً أو شركةً ...
<code>reviewRating</code>	التقييم الذي أعطاه المراجع للعنصر. يُمثّل بنوع الاصطلاحات <code>Rating</code> (http://schema.org/Rating)
<code>author</code>	اسم الشخص الذي كتب المراجعة
<code>datePublished</code>	تاريخ نشر المراجعة بصيغة ISO
<code>name</code>	عنوان مختصر للمراجعة
<code>reviewBody</code>	الوصف الذي كتبه المراجع للعنصر

أول خاصيّة بسيطة: `itemReviewed` هي نص عادي، وقيمته موجودة في عنصر `<h1>`، لذا سنضع تلك الخاصيّة في ذاك العنصر.

```
<h1 itemprop="itemReviewed">Anna's Pizzeria</h1>
```

سأتجاوز حالياً قسم التقييم وسأعود إليه لاحقاً في النهاية.
الخاصيّات التاليتان سهلتان وبسيطتان. خاصيّة `name` هي عنوان مختصر للمراجعة،
وخاصيّة `reviewBody` هي النص الذي كتبه المراجع لوصف العنصر المراجع.

```

<p itemprop="name">New York-style pizza right in historic
downtown Apex</p>
<p itemprop="reviewBody">
    Food is top-notch. Atmosphere is just right for a
    “neighborhood
    pizza joint.” The restaurant itself is a bit cramped; if
    you’re overweight, you may have difficulty getting in and out
    of your seat and navigating between other tables. Used to give
    free garlic knots when you sat down; now they give you plain
    bread and you have to pay for the good stuff. Overall, it’s a
    winner.
</p>
```

يجب أن يكون توصيف العنوان وإحداثيات الموقع الجغرافي مألفًا لديك الآن (إذا لم تكن تتبع معنا منذ بداية الفصل، فراجع [آلية توصيف عنوان شخص](#)، [آلية توصيف عنوان منظمة](#)، [وآلية توصيف إحداثيات الموقع الجغرافي](#) من الأقسام السابقة في هذا الفصل).

```

<p itemprop="contentLocation" itemscope
    itemtype="http://schema.org/Place">
    <span itemprop="address" itemscope
        itemtype="http://schema.org/PostalAddress">
        <span itemprop="streetAddress">100 North Salem
        Street</span><br>
        <span itemprop="addressLocality">Apex</span>,
        <span itemprop="addressRegion">NC</span>
        <span itemprop="postalCode">27502</span><br>
        <span itemprop="addressCountry">USA</span>
    </span>
    <span itemprop="geo" itemscope
        itemtype="http://schema.org/GeoCoordinates">
        <meta itemprop="latitude" content="50.047893" />
        <meta itemprop="longitude" content="14.4491" />
    </span>
</p>
```

في آخر سطر مشكلة شائعة: يحتوي على معلوماتين في عنصرٍ واحدٍ. اسم الشخص الذي كتب المراجعة هو `Mark Pilgrim`, وتاريخ المراجعة هو `March 31, 2010`. كيف نستطيع توصيف هاتين الخاصيتين المستقلتين؟ ضعهما في عنصرين منفصلين وضع خاصية `itemprop` في كلِّ عنصرٍ يجدر بنا في هذا المثال أن نضع التاريخ في عنصر `<time>`, لكي يوفر لنا طريقةً سليمةً لوضع خاصية `itemprop` المناسبة. لكن اسم المُراجع يمكن أن يُحتوى في عنصرٍ ليس له قيمة دلائلية مثل ``.

```
<p>— <span itemprop="author">Mark Pilgrim</span>, last
updated
    <time itemprop="datePublished" datetime="2010-03-31">
        March 31, 2010
    </time>
</p>
</article>
```

حسناً، لنتحدث عن التقييمات. أصعب جزء في توصيف المراجعة هو التقييم. افتراضياً، التقييمات في نوع الاصطلاحات `Rating` تكون على مقياس من 1 إلى 5، إذ 1 هو «سيء جداً» و 5 هو « رائع ». لكنك ما تزال قادرًا على توصيف التقييم إذا كنت تستخدم مقياساً مختلفاً، لكن دعنا نناقش المقياس الافتراضي أولاً.

```
<p>★★★☆☆ (<span itemprop="reviewRating">4</span> stars out of
5)</p>
```

إذا كنت تستعمل مقياساً افتراضياً من 1 إلى 5، فالخاصية الوحيدة التي عليك توصيفها هي التقييم نفسه (4 في مثانا). لكن ماذا لو كنت تستعمل مقياساً مختلفاً؟ كل ما عليك فعله هو التصريح عن حدود المقياس الذي تستعمل. على سبيل المثال، إذا أردت أن تستعمل مقياساً من

إلى 10، فما يزال عليك التصريح عن خاصية `itemprop="rating"` لكن بدلاً من إعطاء قيمة التقييم مباشرةً، سيتوجب عليك استخدام نوع `http://schema.org/Rating` لتعيين أقل قيمة وأعلى قيمة في مقياس المُخْصَص بالإضافة إلى تحديد قيمة للتقييم الفعلي ضمن ذاك المقياس.

```
<p itemprop="reviewRating" itemscope
    itemtype="http://schema.org/Rating">
    ★★★★★★☆
    (<span itemprop="ratingValue">9</span> on a scale of
     <span itemprop="worstRating">0</span> to
     <span itemprop="bestRating">10</span>)
</p>
```

بالعربية: «هذا المنتج الذي أكتب مراجعةً عنه يملك تقييماً قيمته 9 في مقياس من 0 إلى 10.».

هل ذكرت لك أن البيانات الوصفية المضافة للمراجعات قد تؤثر على نتائج البحث؟ هذه هي البيانات التي تستطيع **أداة اختبار البيانات المنظمة** استخلاصها من مراجعتنا:

Review (1)		All good														
Review																
name:	New York-style pizza right in historic downtown Apex															
reviewBody:	Food is top-notch. Atmosphere is just right for a "neighborhood pizza joint." The restaurant itself is a bit cramped; if you're overweight, you may have difficulty getting in and out of your seat and navigating between other tables. Used to give free garlic knots when you sat down; now they give you plain bread and you have to pay for the good stuff. Overall, it's a winner.															
datePublished:	2010-03-31															
itemReviewed [Thing]:	<table border="1"> <tr> <td>name:</td><td>Anna's Pizzeria</td></tr> <tr> <td>reviewRating [Rating]:</td><td></td></tr> <tr> <td>ratingValue:</td><td>9</td></tr> <tr> <td>worstRating:</td><td>0</td></tr> <tr> <td>bestRating:</td><td>10</td></tr> </table>		name:	Anna's Pizzeria	reviewRating [Rating]:		ratingValue:	9	worstRating:	0	bestRating:	10				
name:	Anna's Pizzeria															
reviewRating [Rating]:																
ratingValue:	9															
worstRating:	0															
bestRating:	10															
contentLocation [Place]:	<table border="1"> <tr> <td>address [PostalAddress]:</td><td></td></tr> <tr> <td>streetAddress:</td><td>100 North Salem Street</td></tr> <tr> <td>addressLocality:</td><td>Apex</td></tr> <tr> <td>addressRegion:</td><td>NC</td></tr> <tr> <td>postalCode:</td><td>27502</td></tr> <tr> <td>addressCountry [Country]:</td><td></td></tr> <tr> <td>name:</td><td>USA</td></tr> </table>		address [PostalAddress]:		streetAddress:	100 North Salem Street	addressLocality:	Apex	addressRegion:	NC	postalCode:	27502	addressCountry [Country]:		name:	USA
address [PostalAddress]:																
streetAddress:	100 North Salem Street															
addressLocality:	Apex															
addressRegion:	NC															
postalCode:	27502															
addressCountry [Country]:																
name:	USA															
geo [GeoCoordinates]:	<table border="1"> <tr> <td>latitude:</td><td>50.047893</td></tr> <tr> <td>longitude:</td><td>14.4491</td></tr> </table>		latitude:	50.047893	longitude:	14.4491										
latitude:	50.047893															
longitude:	14.4491															
author [Thing]:	<table border="1"> <tr> <td>name:</td><td>Mark Pilgrim</td></tr> </table>		name:	Mark Pilgrim												
name:	Mark Pilgrim															

الشكل 53: معلومات البيانات الوصفية كما تُظهرها أداة اختبار البيانات المنظمة

وهذه صورةً توضيحيةً لكيف يمكن أن تبدو المراجعة عند عرضها في نتائج البحث:

[Anna's Pizzeria: review](#)

★★★★★ Review by Mark Pilgrim - Mar 31, 2010

Excerpt from the page will show up here.

Excerpt from the page will show up here.

diveintohtml5.org/examples/review-plus-microdata.html - [Cached](#) - [Similar pages](#)

الشكل 54: مثالٌ عن نتيجة البحث عن صفحة فيها بياناتٌ وصفيةٌ تصف مراجعةً

أليس هذا مُبهراً؟

7. مصادر إضافية

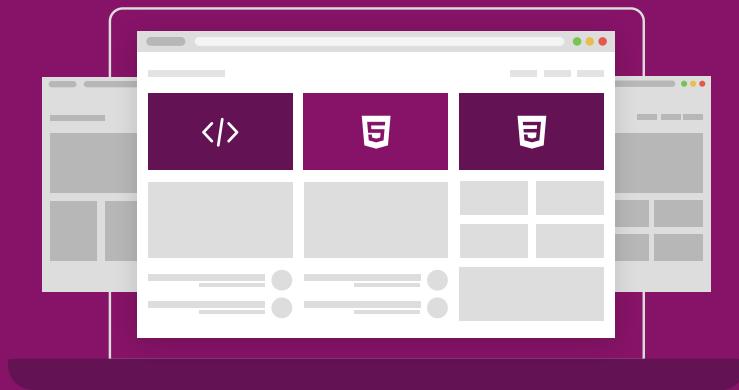
مصادر عن البيانات الوصفية (Microdata):

- مواصفة Microdata في HTML5
- صفحة Microdata في ويكيبيديا

مصادر عن المقتطفات المنسقة:

- صفحة Rich Snippets
- أداة اختبار البيانات المنظمة
- موقع schema.org الذي يحتوي معلوماتٍ عن أنواع الاصطلاحات المختلفة التي ذكرناها في هذا الفصل.

دورة تطوير واجهات المستخدم



مميزات الدورة

- ✓ شهادة معتمدة من أكاديمية حسوب
- ✓ إرشادات من المدربين على مدار الساعة
- ✓ من الصفر دون الحاجة لخبرة مسبقة
- ✓ بناء معرض أعمال قوي بمشاريع حقيقة
- ✓ وصول مدى الحياة لمحتويات الدورة
- ✓ تدريبات مستمرة على الدورة مجاناً

اشترك الآن



التعامل مع التأريخ



اعتبر أن شريط العنوان في المتصفح هو أشهر عنصرٍ من عناصر الواجهات الرسومية في العالم، إذ أصبحت تُعرض روابط URL على اللوحات الإعلانية، وعلى جوانب الطرق، وحتى في الكتابات على الجدران؛ مجتمعاً مع زر الرجوع إلى الخلف - أحد أهم الأزرار في المتصفح - ستحصل على مقدرةٍ على التنقل إلى الأمام وإلى الخلف في شبكة المعلومات الكبيرة التي نسميها الويب.

الواجهة البرمجية للتعامل مع التاريخ في HTML5 هي طريقةٌ معياريةٌ لتعديل تاريخ (history) المتصفح باستخدام السكريبتات. جزءٌ من هذه الواجهة البرمجية (التنقل في التاريخ) موجودٌ في الإصدارات السابقة من HTML؛ أما الأجزاء الجديدة في HTML5 تضمنَت طرائق إضافة مدخلات إلى تاريخ المتصفح، وتغيير رابط URL الظاهر في شريط المتصفح (دون الحاجة إلى تحديث الصفحة)، وإضافة حدث سيفُعل عندما تُحذف تلك المدخلات من المكدس stack، وهذه هي آلية التعامل الداخلية مع التاريخ) بوساطة المستخدم عند ضغطه لزر الرجوع في المتصفح. وهذا يعني أن رابط URL في شريط العنوان في المتصفح سيستمر بأداء عمله كمعرف فريد للمورد الحالي (current resource)، حتى في التطبيقات التي تعتمد اعتماداً كبيراً على السكريبتات التي لن تُجري تحديثاً كاملاً للصفحة.

1. السبب وراء تعديل التاريخ

لماذا تريد تعديل تاريخ المتصفح يدوياً؟ بكل بساطة، يمكن لرابط بسيطٍ أن ينقلك إلى URL جديد؛ وهذه هي الطريقة التي عمل بها الويب لأكثر من 25 سنة، وسيستمر بذلك. هذه الواجهة البرمجية لن تحاول تقويض الويب، وإنما العكس. وفي السنوات الأخيرة، وجّه مطورو الويب طرائق جديدة ومثيرة لتخريب الويب دون أي مساعدة من المعايير الناشئة. صُممَت

واجهة التاريخ البرمجية في HTML5 للتأكد من أن روابط URL مستمرة بأداء وظيفتها وأن تبقى مفيدةً حتى في تطبيقات الويب التي تعتمد تماماً على السكريبتات.

بالعودة إلى المفاهيم الأساسية، ما وظيفة رابط URL؟ إنه يُعرف مورداً فريداً (unique). يمكنك إضافة رابط مباشر إليه، أو وضع علامة مرجعية إليه، ويمكن لمحركات البحث فهرسته، ويمكنك نسخه ولصقه في رسالة عبر البريد الإلكتروني لشخص ما الذي يستطيع أن يضغط عليه ويرى نفس المورد الذي رأيته أنت. هذه الميزات الرائعة تُريك أهمية روابط URL. حسناً، نحن نريد أن تكون للموارد المختلفة روابط URL فريدة. لكن المتصفحات تعاني من قصورٍ أساسي: إذا غيرت رابط URL، حتى باستخدام السكريبتات، فستطلب من خادوم الويب البعيد تحديداً لـكامل الصفحة. وهذا يأخذ وقتاً وموارداً وسيبدو لك كم أن ذلك مُبِّدٌ للموارد إذا كانت الصفحة التي ستنتقل إليها شبيهةً جداً بالصفحة الحالية. ستُنزل كل أجزاء الصفحة الجديدة، حتى الأجزاء المماثلة للصفحة الحالية، فلا توجد هنالك طريقةٌ لتغيير رابط URL وتتنزيل نصف الصفحة فقط.

أنت الواجهة البرمجية للتعامل مع التاريخ في HTML5 لحل هذه الإشكالية. بدلًا من تحديث كامل الصفحة، يمكنك أن تستخدم سكريبتاً لتنزيل «نصف صفحة». هذه الخدعة صعبة قليلاً وتحتاج إلى قليلٍ من العمل...

لننقل أن لديك صفحتان، الصفحة A و الصفحة B. كلا الصفحتين متماثلة بنسبة 90%؛ وهناك 10% فقط من المحتوى يختلف بين الصفحتين. زار المستخدم الصفحة A، ثم حاول الانتقال إلى الصفحة B، لكن بدلاً من تحديث الصفحة تحديداً كاملاً، حاولَ اعتراف هذه العملية وإجراء الخطوات الآتية يدوياً:

- .1. تحميل 10% من الصفحة B المختلفة عن A (ربما باستخدام XMLHttpRequest)، وهذا سيتطلب بعض التعديلات من جهة الخادم لتطبيق الويب الخاص بك. إذ ستحتاج إلى كتابة شيفرة لكي تُعيد 10% فقط من الصفحة B المختلفة عن A، وربما تفعل ذلك عبر رابط URL مخفى أو عبر ترمير وسيط في الطلبيّة الذي لا يستطيع المستخدم النهائي رؤيّته في الحالات العاديّة.
 - .2. تبديل المحتوى الذي تغيّر (عبر استخدامinnerHTML أو دوال DOM الأخرى)، قد تحتاج أيضًا إلى إعادة ضبط أية دوال لمعالجة الأحداث المرتبطة بالعناصر الموجودة في المحتوى المُبدَّل.
 - .3. تحديث شريط العنوان في المتصفح لكي يحتوي على رابط URL للصفحة B، وذلك عبر دالة خاصة من الواجهة البرمجية للتعامل مع التأريخ في HTML5 التي سأريك إياها بعد قليل.
- في نهاية هذه الخدعة (إذا طبقتها تطبيقًّا صحيحًّا)، سينتهي الأمر بحصول المتصفح على شجرة DOM مماثلة لتلك التي سيحصل عليها لو انتقل إلى الصفحة B مباشرةً. وسيصبح العنوان في شريط المتصفح مساوًياً لرابط URL للصفحة B، كما لو أنك انتقلت إلى الصفحة B مباشرةً. لكنك في الحقيقة لم تتنقل إلى الصفحة B ولم تُجر تحديثًا كاملاً للصفحة. وهذه هي الخدعة التي كنت أتحدث عنها. لأن الصفحة النهائية مماثلةً تمامًا للصفحة B ولها نفس رابط URL للصفحة B، لكن المستخدم لم يلاحظ الفرق (ولم يكن ممنوعًّا لك على جهودك لتحسين تجربته مع التطبيق).

2. طريقة تعديل التاريخ

يوجد في واجهة التاريخ البرمجية عددٌ من الدوال في كائن `window.history`, بالإضافة إلى حدٍّ وحيد في الكائن `window`. يمكنك استخدامها لكتشاف الدعم لواجهة التاريخ البرمجية، وهناك دعم لا يُبَلِّغ به منأغلبية المتصفحات الحديثة لهذه الواجهة البرمجية.

Android	iPhone	Opera	Chrome	Safari	Firefox	IE
*+4.3	+4.2.1	+11.5	+8.0	+5.0	+4.0	+10

* من المفترض أنَّ هناك دعم لواجهة التاريخ البرمجية في الإصدارات السابقة من متصفح أندرويد، لكنَّ وجود عدد كبير من العلل البرمجية في تطبيق هذا الدعم في إصدار 4.0.4، فسنعتبر أنَّ الدعم «ال حقيقي » لواجهة التاريخ البرمجية قد بدأ منذ الإصدار 4.3.

« Dive Into Dogs » هو مثالٌ بسيطٌ وعمليٌّ لكيفية الاستفادة من الواجهة البرمجية للتاريخ في HTML5. إذ يُبَرِّز مشكلةً شائعةً: مقالةً طويلةً يرتبط بها معرض صورٍ سيؤدي الضغط على رابط « Previous » أو « Next » في معرض الصور إلى تحديث الصورة آنيًا وتحديث رابط URL في شريط العنوان في المتصفح دون الحاجة إلى تحديث كامل الصفحة. أما في المتصفحات التي لا تدعم واجهة التاريخ البرمجية -أو في المتصفحات الداعمة لها، لكن المستخدم عَذَّل السكريبتات- سيعمل الرابطان « Next » و « Previous » كرابطين اعتياديين، ويأخذانك إلى الصفحة التالية بعد تحديث كامل الصفحة.

وهذا يشير النقطة المهمة الآتية:

إذا لم يتمكن تطبيقك من العمل في حال تعطيل المستخدم للسكريبتات، فيجب نفيك إلى كوكب آخر على الفور!

لننفع في مثال «Dive Into Dogs» ونرى كيف يعمل. هذه هي الشيفرة التي تُستعمل لعرض صورة وحيدة:

```
<aside id="gallery">
  <p class="photonav">
    <a id="photonext" href="casey.html">Next &gt;</a>
    <a id="photoprev" href="adagio.html">&lt; Previous</a>
  </p>
  <figure id="photo">
    
    <figcaption>Fer, 1972</figcaption>
  </figure>
</aside>
```

لا يوجد شيء غير اعتيادي فيما سبق. الصورة نفسها هي عنصر `` موجود ضمن عنصر `<figure>`، جميع الروابط هي عناصر `<a>`، وكل شيء محتوى في عنصر `<aside>`، من المهم أن نلاحظ أن هذه الروابط العاديّة تعمل عملاً صحيحاً. جميع الشيفرات التي تحكم بالتأريخ موجودة بعد سكريبت **لاكتشاف الدعم**، فإن لم يكن متصفح المستخدم داعماً للواجهة البرمجية للتاريخ، فلا حاجة إلى تنفيذ الشيفرات المتعلقة بها، وكذلك الأمر للمستخدمين الذي عطلوا تنفيذ السكريبتات بالمجمل.

الدالة الرئيسية تحصل على كل رابط من تلك الروابط وتمرّره إلى

`.addClicker()`

```
function setupHistoryClicks() {
  addClicker(document.getElementById("photonext"));
  addClicker(document.getElementById("photoprev"));
}
```

هذه هي دالة `addClicker()` التي تأخذ عنصر `<a>` وتحضير إليه دالة للتعامل مع الحدث `click`، التي يحدث فيها أمرٌ مثيرٌ للاهتمام.

```
function addClicker(link) {
    link.addEventListener("click", function(e) {
        swapPhoto(link.href);
        history.pushState(null, null, link.href);
        e.preventDefault();
    }, false);
}
```

الدالة `(swapPhoto()` تقوم بأول خطوتين من **الخطوات الثلاث** التي تحدثنا عنها. أول قسم من الدالة `(swapPhoto()` يأخذ جزءاً من URL لرابط التنقل -أي `adegio.html` و `casey.html` وهكذا...- ويبني رابط URL جديد يشير إلى صفحةٍ مخفيةٍ التي لا تحتوي إلا على الشيفرة اللازمة لعرض الصورة التالية.

```
function swapPhoto(href) {
    var req = new XMLHttpRequest();
    req.open("GET",
        "http://diveintohtml5.org/examples/history/gallery/" +
        + href.split("/").pop(),
        false);
    req.send(null);
```

هذا مثالٌ عن **الشيفرات التي تُتَّبِعُها تلك الصفحات** (يمكنك التأكد من ذلك في متصفحك بزيارة الرابط السابق [الذي هو رابط URL] مباشرةً).

```
<p class="photonav">
    <a id="photonext" href="brandy.html">Next &gt;</a>
    <a id="photoprev" href="fer.html">&lt; Previous</a>
</p>
<figure id="photo">
```

```

<figcaption>Casey, 1984</figcaption>
</figure>
```

هل تبدو الشيفرة السابقة مألوفةً لديك؟ هذا طبيعي، لأنها **نفس الشيفرة التي استخدمناها في صفحتنا الرئيسية** لعرض أول صورة.

القسم الثاني من دالة swapPhoto() يجري الخطوة الثانية من **الخطوات الثلاث** التي تحدثنا عنها: وضع الشيفرة الجديدة التي نُزّلت في الصفحة الحالية، تذَكَّر أنَّ هنالك عنصر `<aside>` محيطٌ بالصورة والشرح التوضيحي لها، لذلك تكون عملية إضافة شيفرة الصورة الجديدة بسيطةً عبر ضبط خاصية `innerHTML` لعنصر `<aside>` إلى خاصية `responseText` من كائن XMLHttpRequest.

```
if (req.status == 200) {
  document.getElementById("gallery").innerHTML =
  req.responseText;
  setupHistoryClicks();
  return true;
}
return false;
}
```

لاحظ أيضًا استدعاء الدالة `setupHistoryClicks()`، وهذا ضروري لإعادة ضبط دوال التعامل مع الحدث `click` في روابط التنقل الجديدة، لأن ضبط المحتوى باستخدام `innerHTML` سيمسح كل آثار الروابط القديمة مع دوال التعامل مع أحداثها.

لنعد الآن إلى الدالة addClicker()، وبعد النجاح بتغيير الصورة، هنالك خطوة إضافية علينا عملها من **الخطوات الثلاث**: ضبط رابط URL في شريط العنوان دون تحديث الصفحة.

```
history.pushState(null, null, link.href);
```

تأخذ الدالة (`history.pushState()`) ثلاثة وسائط:

1. `state`: يمكن أن يكون أي بنية من بُنى بيانات JSON، وستُمَرَّر إلى الدالة التي تتعامل مع الحدث `popstate`، الذي ستتعلم المزيد من المعلومات عنه بعد قليل. لا يحتاج إلى تتبع أي حالة في هذا المثال، لذا سأترك القيمة مساويةً إلى `null`.
2. `title`: يمكن أن يكون أي سلسلة نصية. لكن هذا الوسيط غير مدعاوم من أغلبية المتصفحات الرئيسية، فإذا أردت ضبط عنوان الصفحة، فعليك تخزينه في الوسيط `state` ثم ضبطه يدوياً في الدالة التي تتعامل مع الحدث `popstate`.
3. `url`: يمكن أن يكون أي رابط URL، وهو الرابط الذي سيظهر في شريط العنوان في متصفحك.

استدعاء الدالة `history.pushState` سيؤدي إلى تغيير رابط URL الظاهر في شريط العنوان في المتصفح على الفور، لكن أليست هذه نهاية القصة؟ ليس تماماً، بقي علينا أن نتحدث عما سيحصل إذاضغط المستخدم على الزر المهم «الرجوع».

الحالة العادية عندما يزور المستخدم صفحة جديدة (بتحميلها كلها) هي إضافة المتصفح لرابط URL الجديد في «مكدس» التاريخ (history stack) ثم يُنْزَل ويعرض الصفحة الجديدة. وعندما يضغط المستخدم على زر الرجوع إلى الخلف، فسيزيل المتصفح الصفحة الحالية من مكدس التاريخ ويعرض الصفحة التي تسبقها، لكن ماذا سيحدث عندما عبّثت بهذه الآلية لكي تتفادى تحديداً لـكامل الصفحة؟

حسناً، لقد زيفت «التقدم إلى الأمام» إلى رابط URL جديد، وحان الوقت الآن لتزييف «الرجوع إلى الخلف» إلى رابط URL السابق. والمفتاح نحو تزييف «الرجوع إلى الخلف» هو `.popstate` الحدث.

```
window.addEventListener("popstate", function(e) {
    swapPhoto(location.pathname);
})
```

بعد أن استخدمنا الدالة `history.pushState()` لإضافة رابط URL مزيف في مكدس التاريخ الخاص بالمتصفح، سيُطلق المتصفح الحدث `popstate` في الكائن `window` عندما يضغط المستخدم على زر الرجوع. وهذه هي فرصتك لإكمال عملك في إيهام المستخدم أنه انتقل فعلاً إلى تلك الصفحة.

في هذا المثال، عملية إعادة الصفحة السابقة بسيطة جدًا، فكل ما عليك فعله هو إعادة الصورة الأصلية، وذلك باستدعاء الدالة `swapPhoto()` مع تمرير رابط URL الحالي لها. لأنّه عند استدعاء الدالة التي تعالج الحدث `popstate`، يكون رابط URL الظاهر في المتصفح قد تغير إلى رابط URL السابق. وهذا يعني أيضًا أنّ قيمة الخاصية العامة `location` قد حدّثت لرابط URL السابق.

لكي أساعدك في تخيل الوضع، دعني أتلّو عليك العملية «السحرية» من بدايتها إلى نهايتها:

- يُحمل المستخدم الصفحة <http://diveintohtml5.org/examples/history/fer.html> ويشاهد القصة وصورةً ل الكلب Fer.
- يضغط المستخدم على الرابط المعنون «Next»، الذي هو عنصر `<a>` تكون قيمة `href` فيه هي <http://diveintohtml5.org/examples/history/casey.html>.

- بدلاً من الانتقال إلى <http://diveintohtml5.org/examples/history/casey.html> •

مباشرةً وتحديث الصفحة تحدثاً كاملاً، فستعرض دالة خاصةً للحدث `click` عملية الضغط على عنصر `<a>` وتنفذ شيفرة خاصةً بها.
- تستدعي تلك الدالة التي تعالج الحدث `click` الدالة `swapPhoto()`، التي تنشئ كائن `XMLHttpRequest` لكي يثّل جزء HTML الموجود في الملف <http://diveintohtml5.org/examples/history/gallery/casey.html> بشكل تزامني. •

الدالة `swapPhoto()` تضبط الخاصية `innerHTML` للعنصر الذي يحوي الصورة (عنصر ``)، وبهذا ستبدل صورة `Casey` بصورة `Fer`.
- في النهاية، تستدعي الدالة التي تعامل مع الحدث `click` الدالة `history.pushState()` لتغيير رابط URL يدوياً في شريط عنوان المتصفح إلى <http://diveintohtml5.org/examples/history/casey.html>. •

يضغط المستخدم على زر الرجوع في المتصفح.
- يلاحظ المتصفح أنَّ رابط URL قد تغير يدوياً وأضيف إلى مكدس التاريخ (عبر الدالة `history.pushState()`) وبذلاً من الانتقال إلى رابط URL السابق وإعادة تحديث الصفحة، فسيحدث المتصفح الرابط الموجود في شريط العنوان إلى رابط URL للصفحة السابقة (<http://diveintohtml5.org/examples/history/fer.html>) ثم يطلق الحدث `popstate`. •

الدالة التي تعالج الحدث `popstate` تستدعي الدالة `swapPhoto()` مرةً أخرى، لكن هذه المرة مع تمرير الرابط القديم إليها الذي أصبح موجوداً الآن في شريط العنوان.

- ثم باستخدام XMLHttpRequest مرأة أخرى، ستنزل الدالة swapPhoto() جزءاً من صفحة HTML الموجودة في <http://diveintohtml5.org/examples/history/gallery>
- ثم ستضبط الخاصية innerHTML للعنصر الذي يحوي الصورة (عنصر `/fer.html`)، وبهذا ستبدل صورة Fer بصورة Casey.
- اكتملت خدعتنا، جميع الأدلة الظاهرة (محتوى الصفحة، وعنوان URL في المتصفح) تشير إلى أنَّ المستخدم قد انتقل إلى الأمام صفحةً وإلى الخلف صفحةً. لكن لم يحصل تحديث كاملٌ للصفحة.

3. مصادر إضافية

- HTML5 في معيار Session history and navigation
- Mozilla Developer Center على Manipulating the browser history
- استعراض بسيطة لواجهة التاريخ البرمجية
- Using HTML5 today التي تشرح كيف يستعمل موقع فيسبوك الواجهة البرمجية للتاريخ
- The Tree Slider التي تشرح استعمال موقع GitHub للواجهة البرمجية للتاريخ
- History.js هي مكتبةٌ للتعامل مع التاريخ في المتصفحات الحديثة والقديمة

بيكاليكا



هل تطمح لبيع منتجاتك الرقمية عبر الإنترنٌت؟

استثمر مهاراتك التقنية وأطلق منتجًا رقميًّا يحقق لك دخلًا عبر بيعه على متجر بيكاليكا

أطلق منتجك الآن

الملحق الأول: دليل اكتشاف دعم المتصفح لميزات HTML5

اقرأ **الفصل الثاني** لتتعرف على قواعد اكتشاف دعم المتصفح للميزات؛ أو استعمل مكتبة

إذا أردت حلاً جاهزاً لاكتشاف جميع الميزات التي تدعمها المتصفحات.

<audio> •

```
return !!document.createElement('audio').canPlayType;
```

MP3 بصيغة <audio> •

```
var a = document.createElement('audio');
return !(a.canPlayType &&
a.canPlayType('audio/mpeg;').replace(/no/, ''));
```

Vorbis بصيغة <audio> •

```
var a = document.createElement('audio');
return !(a.canPlayType && a.canPlayType('audio/ogg;
codecs="vorbis"').replace(/no/, ''));
```

WAV في حاوية <audio> •

```
var a = document.createElement('audio');
return !(a.canPlayType && a.canPlayType('audio/wav;
codecs="1"').replace(/no/, ''));
```

AAC بصيغة <audio> •

```
var a = document.createElement('audio');
return !(a.canPlayType && a.canPlayType('audio/mp4;
codecs="mp4a.40.2"').replace(/no/, ''));
```

<canvas> •

```
return !!document.createElement('canvas').getContext;
```

<canvas> في النصوص •

```
var c = document.createElement('canvas');
return c.getContext && typeof c.getContext('2d').fillText ==
'function';
```

<command> •

```
return 'type' in document.createElement('command');
```

<datalist> •

```
return 'options' in document.createElement('datalist');
```

<details> •

```
return 'open' in document.createElement('details');
```

<device> •

```
return 'type' in document.createElement('device');
```

التحقق من المدخلات في عنصر <form> •

```
return 'noValidate' in document.createElement('form');
```

<iframe sandbox> •

```
return 'sandbox' in document.createElement('iframe');
```

<iframe srcdoc> •

```
return 'srcdoc' in document.createElement('iframe');
```

<input autofocus> •

```
return 'autofocus' in document.createElement('input');
```

<input placeholder> •

```
return 'placeholder' in document.createElement('input');
```

<textarea placeholder> •

```
return 'placeholder' in document.createElement('textarea');
```

<input type="color"> •

```
var i = document.createElement('input');
i.setAttribute('type', 'color');
return i.type !== 'text';
```

<input type="email"> •

```
var i = document.createElement('input');
i.setAttribute('type', 'email');
return i.type !== 'text';
```

<input type="number"> •

```
var i = document.createElement('input');
i.setAttribute('type', 'number');
return i.type !== 'text';
```

<input type="range"> •

```
var i = document.createElement('input');
i.setAttribute('type', 'range');
return i.type !== 'text';
```

<input type="search"> •

```
var i = document.createElement('input');
i.setAttribute('type', 'search');
return i.type !== 'text';
```

<input type="tel"> •

```
var i = document.createElement('input');
i.setAttribute('type', 'tel');
return i.type !== 'text';
```

<input type="url"> •

```
var i = document.createElement('input');
i.setAttribute('type', 'url');
return i.type !== 'text';
```

<input type="date"> •

```
var i = document.createElement('input');
i.setAttribute('type', 'date');
```

```
return i.type !== 'text';
```

```
<input type="time"> •
```

```
var i = document.createElement('input');
i.setAttribute('type', 'time');
return i.type !== 'text';
```

```
<input type="datetime"> •
```

```
var i = document.createElement('input');
i.setAttribute('type', 'datetime');
return i.type !== 'text';
```

```
<input type="datetime-local"> •
```

```
var i = document.createElement('input');
i.setAttribute('type', 'datetime-local');
return i.type !== 'text';
```

```
<input type="month"> •
```

```
var i = document.createElement('input');
i.setAttribute('type', 'month');
return i.type !== 'text';
```

```
<input type="week"> •
```

```
var i = document.createElement('input');
i.setAttribute('type', 'week');
return i.type !== 'text';
```

<meter> •

```
return 'value' in document.createElement('meter');
```

<output> •

```
return 'value' in document.createElement('output');
```

<progress> •

```
return 'value' in document.createElement('progress');
```

<time> •

```
return 'valueAsDate' in document.createElement('time');
```

<video> •

```
return !!document.createElement('video').canPlayType;
```

التراجمات النصية في <video> •

```
return 'src' in document.createElement('track');
```

<video poster> •

```
return 'poster' in document.createElement('video');
```

WebM بصيغة <video> •

```
var v = document.createElement('video');
```

```
return !(v.canPlayType && v.canPlayType('video/webm;  
codecs="vp9, vorbis')).replace(/no/, ''));
```

H.264 بصيغة <video> •

```
var v = document.createElement('video');  
return !(v.canPlayType && v.canPlayType('video/mp4;  
codecs="avc1.42E01E, mp4a.40.2')).replace(/no/, ''));
```

contentEditable •

```
return 'isContentEditable' in document.createElement('span');
```

Cross-document messaging •

```
return !!window.postMessage;
```

السحب والإفلات (Drag-and-drop) •

```
return 'draggable' in document.createElement('span');
```

File •

```
return typeof FileReader != 'undefined';
```

الموقع الجغرافي (Geolocation) •

```
return !!navigator.geolocation;
```

التاريخ (History) •

```
return !(window.history && window.history.pushState);
```

- التخزين المحلي (Local storage)

```
try {
    return 'localStorage' in window && window['localStorage'] !== null;
} catch(e) {
    return false;
}
```

- البيانات الوصفية (Microdata)

```
return !!document.getItems;
```

- تطبيقات الويب التي تعمل دون اتصال (Offline web applications)

```
return !!window.applicationCache;
```

- الأحداث المرسلة من الخادم (Server-sent events)

```
return typeof EventSource !== 'undefined';
```

- تخزين الجلسة (Session storage)

```
try {
    return 'sessionStorage' in window && window['sessionStorage'] !== null;
} catch(e) {
    return false;
}
```

{

SVG •

```
return !(document.createElementNS &&
document.createElementNS('http://www.w3.org/2000/svg',
'svg').createSVGRect);
```

صور SVG في text/html •

```
var e = document.createElement('div');
e.innerHTML = '<svg></svg>';
return !(window.SVGSVGElement && e.firstChild instanceof
window.SVGSVGElement);
```

(Undo) التراجع •

```
return typeof UndoManager !== 'undefined';
```

IndexedDB •

```
return !window.indexedDB;
```

Web Sockets •

```
return !window.WebSocket;
```

Web SQL Database •

```
return !window.openDatabase;
```

Web Workers •

```
return !!window.Worker;
```

هل أنا في واحدي منها؟: **Widgets** •

```
return typeof widget !== 'undefined';
```

طلبيات عابرة للنطاقات: **XMLHttpRequest** •

```
return "withCredentials" in new XMLHttpRequest;
```

إرسال كبيانات نموذج: **XMLHttpRequest** •

```
return !!window.FormData;
```

الأحداث الناتجة عن رفع ملف: **XMLHttpRequest** •

```
return "upload" in new XMLHttpRequest;
```

1. مصادر إضافية

المعايير والمواصفات:

Geolocation •

Server-Sent Events •

WebSimpleDB •

Web Sockets •

Web SQL Database •

Web Storage •

Web Workers •

Widgets •

XMLHttpRequest Level 2 •

مكتبات JavaScript:

HTML5: مكتبة لاكتشاف دعم ميزات Modernizr •