عشر قواعد بسيطة لتطوير الدروس بالتعاون

غابرييل أ. ديفيني، ريمي إيمونيه، راينا م. هاريس، كيت ل. هيرتويك، داميان إيرفينغ، إيان ميليغان، غريغ ويلسون



الملخص:

يقدّم هذا المقال عشر قواعد عملية لتطوير الدروس التعليمية تعاونيًا، استخلصها مؤلفوه من خبرتهم الطويلة في مشروعات مثل (Software Carpentry) و(Programming Historian)، التي تقوم على أن تكون الدروس مفتوحة ويسهل الوصول إليها، وأن يتعهدها مجتمع من المساهمين بالصيانة والتحسين المستمرين. وتشمل القواعد: تحديد الجمهور بكتابة ملفات للمتعلمين، وتقسيم الدروس إلى وحدات صغيرة، وتعليم المدربين أفضل الممارسات التربوية، وتيسير المساهمة وجعلها شفافة، وبناء مجتمع حول الدروس، ونشر إصدارات دورية تعترف بالمساهمين وتمنحها معرّفات (DOI)، وتقييم الدروس على مستويات متعددة، وإعادة استخدام الدروس القائمة مع مراعاة تراخيصها، والربط بمصادر مكمّلة، وإدراك أن أي درس لا يمكنه إرضاء الجميع. ويخلص المؤلفون إلى أن تبنّي نموذج تطوير البرمجيات تعاونيًا يتيح دروسًا أعلى جودة وأكثر استدامة، تبقى محدّثة وتتطلب وقتًا أقل من كل مؤلف.

الكلمات المفتاحية: الموارد التعليمية المفتوحة، تطوير الدروس تعاونيًا، مشروع (Software Carpentry)، بناء المجتمعات، تراخيص المشاع الإبداعي

مقدمة:

يتطلب بناء الدروس جهدًا كبيرًا، وتتطلب صيانتها جهدًا أكبر. ويؤدي معظم الأكاديميين هذا العمل بمفردهم، غير أن الاستفادة من نهج مجتمعي يمكن أن تجعل تطوير الموارد التعليمية أكثر استدامة ومتانة واستجابة. فالتعامل مع الدروس بوصفها موردًا مجتمعيًا يُحدَّث ويُكيَّف ويُحسَّن تدريجيًا يمكن أن يوفّر وقتًا ثمينًا ويرفع الجودة في آن واحد.

ورغم نجاح الانفتاح في تطوير البرمجيات وفي تحرير ويكيبيديا، فإنه نهج غير مألوف في بيئة التدريس الأكاديمي. ففي كل عام يدرّس آلاف المحاضرين الجامعيين موضوعات تتراوح بين علم الأحياء للسنة الأولى ومقررات الدراسات العليا في السينما الهندية. ويستخدم بعضهم كتبًا دراسية ألّفها مؤلف واحد أو بضعة مؤلفين، لكنهم فيما عدا ذلك يطوّرون موادهم الدراسية ويتعهدونها بمعزل عن الآخرين.

ونظرًا إلى أن البحث الأكاديمي يعتمد في الغالب على المشاركة، فإن هذا النهج المختلف في تطوير المواد التربوية لافت للنظر، غير أن الجوانب الاجتماعية والنفسية الكامنة وراء هذه النقطة العمياء تتجاوز نطاق هذه الورقة.

ويتمتع المؤلفون بسنوات طويلة من الخبرة في الدروس التي يطورها المجتمع، في سياق الحوسبة البحثية في العلوم والإنسانيات، من خلال منظمات مثل (Software Carpentry) و(Programming Historian) [1]. وقد تأسست (Software Carpentry) عام 1998 لتعليم العلماء مهارات الحوسبة الأساسية، ثم انبثقت عنها منظمتان شقيقتان هما (Data Carpentry) و(Library Carpentry). أما (Programming Historian) فتأسس عام 2008 وتطور ليصبح موقعًا يُحرَّر تعاونيًا ويقدّم دروسًا للباحثين في العلوم الإنسانية. ومن مبادئ هذه المشروعات التوجيهية أن تكون الدروس مفتوحة ويسهل الوصول إليها، وأن يتعهدها مجتمع من المساهمين بالصيانة والتنقيح والتحسين المستمرين.

وتستوفي جميع مشروعات التعليم المفتوح (مثل المقررات الإلكترونية المفتوحة واسعة الالتحاق) المعيار الأول بحكم تعريفها، لكن قليلًا جدًا منها يستوفي المعيار الثاني. وبعبارة أخرى، فمع أنه من الشائع أن يحدّث فرد أو فريق صغير مشروعات التعليم المفتوح من حين إلى آخر (كما يحدث عند تحرير طبعة جديدة من كتاب ونشرها)، فإن ذلك يختلف عن التحسين المستمر الذي يقوم به مجتمع كبير من المساهمين. وتلخّص القواعد البسيطة العشر التالية ما تعلمناه عن القيام بذلك بوصفنا قائمين على صيانة دروس يستخدمها عشرات الآلاف من الناس ومحررين ومراجعين لها. ونرى أنه باتباع هذه القواعد يمكن إنشاء دروس أعلى جودة مما يستطيع فرد أو فريق صغير إنشاءه، من حيث الدقة ومن حيث الجانب التربوي معًا. ومن المزايا الإضافية أن الدروس تظل محدّثة دائمًا، وأنها تتطلب وقتًا أقل من كل مؤلف لتطويرها وصيانتها.

القاعدة 1: حدّد جمهورك:

الشرط الأول لبناء الدروس معًا هو معرفة من تُبنى له. فعبارة «طلاب الآثار» مبهمة إلى حد بعيد: هل تفكر أنت ومعاونوك في طلاب السنة الأولى الذين يحتاجون إلى مدخل إلى هذا المجال، أم في طلاب الدراسات العليا الذين ينوون التخصص في الفرع الذي يركز عليه الدرس، أم في فئة بين هؤلاء وأولئك؟ فإذا اختلفت تصورات المساهمين بشأن المعرفة المسبقة المطلوبة، أو المعدات أو البرمجيات اللازمة، أو مقدار الوقت المتاح للمتعلمين، فسيصعب عليهم العمل معًا.

وبدلًا من البدء بأهداف التعلم (القاعدة 3)، قد يكون من المفيد كتابة ملفات تعريفية للمتعلمين (learner profiles) لتوضيح الخلفية العامة للمتعلم، وما يعرفه مسبقًا، وما يظن أنه يريد فعله، وكيف ستساعده المادة، وأي احتياجات خاصة قد تكون لديه. وهذه التقنية مستعارة من تصميم واجهات المستخدم، ويُعرض ملف تعريفي نموذجي لمتعلم في الإطار 1.

الإطار 1: ملف تعريفي لمتعلم:

انتقل خورخي (Jorge) لتوّه من كوستاريكا إلى كندا لدراسة الهندسة الزراعية. ومع أنه يجيد الإسبانية والإنجليزية كلتيهما، فإنه يعاني إعاقة سمعية تجعل فهم المحاضرات صعبًا عليه أحيانًا، ولا سيما في البيئات الصاخبة. وفيما عدا استخدام (Excel) و(Word) والإنترنت، فإن أبرز خبرة سابقة لخورخي مع الحواسيب هي مساعدة أخته في بناء موقع على (WordPress) لمشروع العائلة التجاري.

ويحتاج خورخي إلى قياس خصائص التربة في المزارع المجاورة باستخدام جهاز محمول يرسل ملفات نصية إلى حاسوبه. وفي الوقت الحالي، يتعين على خورخي فتح كل ملف في (Excel)، وحذف أول قيمة وآخر قيمة، وحساب المتوسط. وستبيّن له هذه الورشة كيف يكتب برنامجًا صغيرًا بلغة (Python) لقراءة البيانات، واختيار القيم الصحيحة من كل ملف، وحساب الإحصاءات المطلوبة.

القاعدة 2: قسّم الدروس إلى وحدات مستقلة:

تختلف احتياجات كل مدرّس، لذا ابنِ أجزاء صغيرة يمكن إعادة توظيفها بطرق كثيرة. فمحاضر جامعي في علم الأرصاد الجوية، مثلًا، قد يبني مقررًا لطلابه بجمع دروس في المعادلات التفاضلية وميكانيكا الموائع والتحليل الطيفي بالامتصاص. وإنشاء المقررات على هذا النحو ينقل عبء المدرّس من كتابة المواد إلى البحث عنها والجمع بينها، وهذا أيسر إذا حددت الدروس بوضوح ما تغطيه (القاعدة 1)، وإذا صممها أشخاص يتشاركون رؤية واحدة (القاعدة 3).

ومن طرق تحقيق ذلك أخذ المقررات القائمة وتقسيمها إلى وحدات أصغر أحادية الغرض (وهو تغيير له في حد ذاته مزايا تربوية وإدارية). وعند القيام بذلك، يمكن تحسين قابلية هذه الوحدات للاكتشاف (discoverability) بربطها بنقاط محددة في المناهج النموذجية التي تصدرها كثير من الجمعيات المهنية (مثلًا بوصفها أهدافًا للتعلم). كما أن الوحدات الأصغر أيسر تناولًا للمساهمين الجدد (القاعدة 4).

القاعدة 3: علّم أفضل الممارسات في تطوير الدروس:

أسفرت عقود من البحث التربوي عن رؤى كثيرة بشأن أفضل السبل لبناء الدروس وتقديمها [2]. ولسوء الحظ، فإن كثيرًا من أعضاء هيئات التدريس في الكليات والجامعات لم يتلقوا إلا قليلًا من التدريب الرسمي في التعليم أو لم يتلقوا أي تدريب [3]، ولذلك نادرًا ما تُطبَّق هذه المعرفة في قاعات الدرس.

وتشير تجربتنا إلى أن تقديم مدخل موجز إلى عدد قليل من الممارسات الأساسية يساعد على تطوير الدروس تعاونيًا. فإذا كان لدى الناس فهم مشترك لكيفية تطوير الدروس، كان العمل معًا أيسر عليهم. ومما هو أقل وضوحًا أنه إذا كان لديهم تصور مشترك لكيفية استخدام الدروس، كانوا أكثر ميلًا إلى بناء مواد قابلة لإعادة الاستخدام. وأخيرًا، فإن تعليم الناس كيف يعلّمون طريقة رائعة لتعريف بعضهم ببعض وبناء المجتمع.

وعلى سبيل المثال، تشجع (Software Carpentry) متطوعيها على استخدام منهجية تطوير الدروس الشائعة التي قدمها ويغينز (Wiggins) وماكتاي (McTighe) [4]، والتي تُوضع فيها أهداف التعلم والتقييمات قبل تطوير أي مواد للدرس. وعلى وجه التحديد، تُعدّ «التقييمات الختامية» (summative assessments) لتحديد ما إذا كانت أهداف التعلم قد تحققت، وتُعدّ «التقييمات التكوينية» (formative assessments) لقياس تقدّم المتعلمين ومنحهم فرصة لممارسة المهارات الأساسية. ثم تُرتَّب التقييمات التكوينية المكتملة، وبعد ذلك فقط تُكتب الدروس بهدف ربط كل تقييم تكويني بالذي يليه. وهذه الطريقة فعالة في حد ذاتها، لكن أعظم فوائدها أنها تمنح الجميع إطارًا للتعاون.

ومن الأمثلة على كيفية تعليم هذه الممارسات التربوية برنامج تدريب المدربين في (Software Carpentry). فقد قُدّم أول مرة عام 2012، وهو الآن مقرر مدته يومان يُقدَّم حضوريًا وعبر الإنترنت [5-7]. وإلى جانب تركيزه على الجانب التربوي، يعلّم المقرر لمن وُضعت دروس (Software Carpentry)، وكيف تُقدَّم، وكيف تُصان. وبفضل هذا التدريب إلى حد كبير، يساهم الآن عدة مئات من الأشخاص سنويًا في دروس (Software Carpentry).

القاعدة 4: شجّع المساهمين ومكّنهم:

جعل عملية المساهمة في الدرس بسيطة وشفافة هو مفتاح تلقّي المساهمات. فالترخيص، ومدونة السلوك، والحوكمة، وعملية المراجعة والنشر، كلها يجب أن تكون صريحة لا ضمنية، لخفض الحواجز الاجتماعية أمام المساهمة.

ويمكن للأدوات أن تساعد، ولا سيما إذا كانت تتيح عرض التغييرات المقترحة ومناقشتها قبل إدراجها في الدروس. (ويُعرف ذلك في تطوير البرمجيات باسم «المراجعة قبل الدمج» (premerge review).) غير أن بعض الأدوات الشائعة في تطوير البرمجيات مفتوحة المصدر تتطلب كلفة تعلّم أولية كبيرة. فمنصات مثل (GitHub)، على سبيل المثال، تدعم كل ما يحتاج إليه تطوير الدروس المفتوحة، لكنها تتطلب من المساهمين استخدام (Git)، المعروف بمنحنى تعلّمه شديد الانحدار [8].

ومما يزيد الأمور تعقيدًا أن بعض صيغ الملفات تجعل التعاون أيسر أو أصعب. فرغم انتشار أنظمة التحكم في الإصدارات (version control) مفتوحة المصدر، فإنها لا تدعم مباشرة مراجعة صيغ ملفات (Microsoft Office) أو (OpenDocument) أو دمجها، وهو ما يضيف عبئًا على القادمين الجدد [9]. ومع أن مستندات غوغل (Google Docs) والويكي تفتقر إلى بعض الإمكانات، مثل المراجعة الكاملة قبل الدمج (وإن كان «وضع الاقتراح» يخفف ذلك إلى حد ما)، فإن انخفاض عتبة الدخول إليها يجعلها أكثر ترحيبًا بالقادمين الجدد.

وأفضل طريقة لاختيار أدوات إدارة الدروس أن تسأل المساهمين المحتملين عما يرتاحون إليه، بدلًا من إلزامهم بالأدوات التي تفضّلها أنت. وتذكّر أيضًا أن المساهمة في درس ليست على الأرجح أولويتهم القصوى، وابحث عن سبل لتخفيف عبئهم المعرفي. فعلى سبيل المثال، يمكن لمنتديات النقاش المتسلسلة أن تحسّن نسبة المحتوى المفيد إلى الرسائل الزائدة بالحد من تبادلات البريد الإلكتروني الطويلة بخاصية «الرد على الجميع». وتتوافر أطر مفتوحة عدة لتيسير تطوير دروس جديدة، مثل (learnr)، و(Morea)، وقوالب (DataCamp).

القاعدة 5: ابنِ مجتمعًا حول الدروس:

تتغير إصدارات البرمجيات واعتمادياتها باستمرار، في حين يتقدم الإنتاج الأكاديمي بوتيرة متسارعة على الدوام. ونتيجة لذلك، فإن ما يكون في طليعة المستجدات في عام قد يصبح متقادمًا في العام التالي وخاطئًا ببساطة في العام الذي يليه. ولذلك يجب أن تركز مجموعات تطوير الدروس تعاونيًا على إنشاء مجتمع يدعم فيه المساهمون بعضهم بعضًا، بدلًا من الاعتماد على مجموعة صغيرة من القائمين على صيانة الدروس. فلا يمكن أن يُتوقَّع من المؤلفين أن يظلوا يقظين على الدوام تجاه درس ما، لكن ذلك ضروري لاستمرار استخدامه.

ومن العناصر الأساسية في ذلك إتاحة فرص لـ«المشاركة الطرفية المشروعة» (legitimate peripheral participation). فإعداد قائمة بمهام صغيرة يمكن للقادمين الجدد إنجازها بسهولة، أو تشجيعهم على إبداء ملاحظاتهم على التغييرات المقترحة، أو دعوتهم إلى إضافة تمارين جديدة وتعديل الرسوم البيانية والمراجع، كلها يمكن أن تكون مدخلًا لأشخاص قد يشككون في أهليتهم أو قدرتهم على تغيير المتن الرئيسي لدرس ما. وبالمثل، فإن الاعتراف بجميع المساهمات، مهما صغرت، يمنح المساهمين الجدد مكافأة مبكرة على مشاركتهم.

وفي عام 2015، أنشأت (Software Carpentry) لجنة فرعية للإرشاد (Mentoring Subcommittee) لدعم المدربين في أثناء تقدمهم في التدريب والتدريس وتطوير المناهج. وقد عززت هذه اللجنة بناء المجتمع بتوفير فضاءات افتراضية يتبادل فيها مدربون من أنحاء العالم قصص النجاح ويناقشون استراتيجيات التغلب على التحديات. وقد ساعد ذلك على تقوية المجتمع وقدّم رؤى حول كيفية تحسين الدروس (القاعدة 7).

وأخيرًا، قد يكون العمل في العلن أمرًا رائعًا، لكنه قد يثبط بعض المشاركين عن التعبير دون قصد. ولذلك توفّر (Programming Historian) أمين مظالم (ombudsperson) متاحًا للمحادثات الخاصة والتيسير لضمان عدم إقصاء أحد. وتعمل (Software Carpentry) وفق مدونة سلوك (Code of Conduct) تحدد معايير السلوك المقبولة لأعضاء المجتمع ولمن يتعاملون مع منظمات (Carpentries) في الفعاليات وفي الفضاءات الافتراضية. ويتولى أعضاء من المجتمع في لجنة فرعية للسياسات (Policy Subcommittee) مناصرة مدونة السلوك والفصل في المخالفات المبلغ عنها.

القاعدة 6: انشر دوريًا واعترف بالمساهمات:

كما هو الحال في البرمجيات، ينبغي نشر إصدارات محددة من الدروس أو إطلاقها دوريًا، بحيث يكون لدى المتعلمين أو المدربين شيء مستقر يرجعون إليه طوال مدة استخدامهم له. كما تتيح الإصدارات الدورية فرصة للاعتراف بمساهمات المؤلفين والقائمين على الصيانة الجدد.

ولا يملك الوسط الأكاديمي إلا طرقًا قليلة للاعتراف بالمساهمات. وإلى أن تتسع هذه الطرق، من المهم نشر الدروس بأساليب تستطيع الأنظمة الأكاديمية التقليدية استيعابها. ومن ذلك منح الإصدارات معرّفات الكائنات الرقمية (DOIs) التي يقدمها مزودون مثل (Zenodo) أو (DataCite). ويمكن إدراج المساهمين مؤلفين، والقائمين على صيانة الدرس محررين، للتمييز في الاعتراف بمساهماتهم. وفي كل مرة يُنشر فيها الدرس، ينبغي جمع الأسماء والمعرّفات، مثل معرّفات (ORCID)، لجميع المساهمين.

وإصدار الدرس فرصة جيدة لإيصال المادة إلى صورة مستقرة بمعالجة المشكلات العالقة ودمج المساهمات. ويحتفظ نظام التحكم في الإصدارات تلقائيًا بقائمة المساهمين، ويمكن استخدامه أيضًا لتتبع المحتوى الموجود في كل إصدار (مثلًا باستخدام الفروع أو الوسوم). وينبغي أن تتبع إصدارات الدروس نظام تسمية متسقًا؛ وقد استخدمت (Software Carpentry) في إصداراتها سنة الإصدار وشهره (مثل «2017.05») [10، 11].

وإذا كانت الدروس تصدر بانتظام، فأتمت العملية وأرشف الإصدارات القديمة في موقع يسهل اكتشافه. واحرص أيضًا على أن يعرف كل المعنيين ما يعنيه «الإنجاز»، أي المشكلات العالقة التي يجب معالجتها وكيفية تنسيقها حتى يصدر الإصدار التالي. وتكفي في البداية قائمة مراجعة بسيطة تُحفظ مع مواد الدرس، لكن مع مرور الوقت قد يرغب المجتمع في استخدام نظام ما لتتبع المشكلات، بحيث يمكن إسناد بنود العمل إلى أشخاص بعينهم ثم التأشير عليها عند إنجازها.

القاعدة 7: قيّم الدروس على مستويات متعددة:

ما يرى المنغمسون في تطوير الدروس أنه يحتاج إلى إصلاح قد يختلف بسهولة عما يراه المتعلمون. ولذلك من الضروري جمع الملاحظات على مستويات متعددة والعمل بموجبها، للتحقق من الافتراضات والبقاء على المسار الصحيح.

ويمكن للمدرّس جمع الملاحظات على المستوى الجزئي في أثناء تدريس درس معين. فبإمكان المتعلمين إبداء ملاحظاتهم على كل شيء، من الأخطاء المطبعية ووضوح أسئلة الاختبارات القصيرة إلى ترتيب عرض الموضوعات، وينبغي للمدرّس أن يسجل ذلك كله في نهاية كل حصة في موضع مشترك ما (مثل مستند (Google) أو قائمة المشكلات في (GitHub)). وإلى جانب تشجيع الملاحظات الشفهية المباشرة، من المستحسن أن تُتاح للمتعلمين وسيلة لإبداء ملاحظاتهم دون الكشف عن هويتهم في أثناء الحصة (مثلًا على قصاصات ورق صغيرة كالأوراق اللاصقة أو من خلال استبانات مجهولة الهوية).

وينبغي استخدام الاستبانات والمقابلات قبل الحصة وبعدها للكشف عن المشكلات الأكبر، ولا سيما تلك الناشئة عن عدم فهم المطورين لجمهورهم فهمًا كاملًا، كافتراض معرفة مسبقة لا يملكها المتعلمون. وتكون هذه الاستبانات أكثر فاعلية إذا أُجريت بعد 30-90 يومًا من الحصة؛ إذ يمنح ذلك الناس وقتًا للتأمل، فتعكس ملاحظاتهم بدقة أكبر ما تعلموه لا مقدار ما استمتعوا به. وأهداف التعلم المصوغة بوضوح (القاعدة 3) ضرورية هنا، لأنها تخبر القائمين على التقييم بما ينبغي لهم قياسه.

القاعدة 8: قلّل، وأعد الاستخدام، وأعد التدوير:

كما أن الباحث لا يكتب ورقة دون مراجعة للأدبيات، فإن المدرّس لا ينبغي له أن ينشئ درسًا جديدًا إذا كان ثمة درس قائم يمكنه استخدامه أو المساهمة فيه. فبحث قصير عبر الإنترنت يمكن أن يكشف ما إذا كان أحدهم قد كتب ما تحتاج إليه، وما إذا كان مكمّلًا لأهدافك، وما إذا كان يمكن تعديله أو تحويره لتلبية احتياجاتك.

وقبل إعادة استخدام المحتوى، تأكد من التحقق من ترخيصه. فكل من (Programming Historian) ومشروعات (Carpentry) تستخدم رخصة المشاع الإبداعي - نسب المصنف (Creative Commons-Attribution)، التي تسمح للناس بمشاركة المادة وتكييفها لأي غرض شريطة الاستشهاد بالمصدر الأصلي. وقد تقيّد تراخيص المشاع الإبداعي الأخرى الاستخدام التجاري و/أو إنشاء المواد المشتقة.

وتُطرح مسألة الترخيص أيضًا عند إعادة تدوير مكونات الدروس، مثل الصور أو البيانات أو الأشكال أو الشفرات البرمجية. فإذا لم تشملها الرخصة صراحة، فاطلب الإذن كما تفعل مع أي مادة أكاديمية أخرى.

والجانب الآخر من هذه القاعدة أن تجعل رخصة دروسك صريحة ويسهل اكتشافها. فعلى سبيل المثال، عند نشر الدروس (القاعدة 6)، تأكد من ظهور كلمات مفتاحية مثل «(CC-BY)» في مداخلها الببليوغرافية وفي ترويسات صفحات (HTML) الخاصة بها.

القاعدة 9: اربط بمصادر أخرى:

من غير المرجح أن يستوعب المتعلمون كل ما يحتاجون إلى معرفته عن موضوع ما من درسك وحده. ويرجع ذلك جزئيًا إلى مسألة النطاق، فأي موضوع مثير للاهتمام أكبر من أن يتسع له درس واحد، لكنه يرجع أيضًا إلى مسألة المستوى والاتجاه. وكما حاجّ كولفيلد (Caulfield) [12]، فإن أفضل طريقة لاستخدام الإنترنت هي تقديم مجموعة متنوعة من الشروح تعرض زوايا ومقاربات كثيرة لموضوع معين، قد يكون كل منها الأنسب لمجموعة مختلفة من الاحتياجات.

وينبغي للدروس المطوَّرة تعاونيًا أن توجّه المتعلمين إلى هذه المصادر في مواضع مناسبة، ومنها الكتب الدراسية والتوثيق التقني ومقاطع الفيديو وصفحات الويب. وإذا وُجد مجتمع أو منتدى نقاش للموضوع، كسلاسل النقاش على (Quora) أو القوائم البريدية، فمن المفيد أيضًا الإحالة إليه.

والقيام بذلك عمل كبير، وصيانته أكبر، وهو ما يزيد أهمية بناء المجتمع حول الدروس (القاعدة 5). ومن الحيوي على وجه الخصوص إشراك المتعلمين مشاركين متكافئين في ذلك المجتمع. فينبغي أن يتمكنوا من اقتراح التحديثات والتصويبات والإضافات على الدروس، وأن يعرفوا أنهم يُشجَّعون على ذلك (القاعدة 4).

القاعدة 10: لا يمكنك إرضاء الجميع:

لا يمكن لدرس واحد أن يناسب كل متعلم. فشخصان لا يملكان أي معرفة مسبقة بموضوع معين قد يتقدمان مع ذلك بسرعتين مختلفتين بسبب تفاوت مستوى معرفتهما العامة. وبالمثل، فإن دروس علم البيئة لمتعلمين في ولاية يوتا وفي فيتنام ستكون على الأرجح أقرب إليهم إذا استخدمت أمثلة مختلفة. ولذلك قد يتعهد مجتمع ما عدة دروس في موضوع واحد تختلف في توجهاتها أو في وتيرتها، تمامًا كما توفر لغات البرمجة عدة مكتبات مختلفة لأداء الوظيفة العامة نفسها بمستويات مختلفة من الأداء والتعقيد.

وبالمثل، لا يمكن لمجتمع تطوير دروس أن يخدم جميع الأغراض. فبعض المجموعات قد تعطي الأولوية للتطور السريع، في حين قد يفضل غيرها نهج «قِس مرتين واقطع مرة واحدة». وإذا كانت هناك طرق متكاملة لشرح أمر ما، أو وجهات نظر يمكن أن تتعايش باحترام، فقد يكون من الممكن عرضها جنبًا إلى جنب. وثمة أسباب تربوية وجيهة لذلك حتى إن لم يختلف المساهمون، إذ إن الموازنة بين البدائل تعزز مهارات التفكير العليا.

لكن لا بد أحيانًا من اتخاذ قرارات. وقد صارع مجتمع البرمجيات مفتوحة المصدر هذه القضايا طوال ثلاثة عقود، وطوّر بعض أفضل الممارسات لمعالجتها [13]. وكما نوقش في القاعدة 4، فإن الخطوة الأولى هي وجود هيكل حوكمة واضح ورخصة واضحة متساهلة. وينبغي مناقشة الخلافات الصغيرة علنًا وباحترام. فإذا اتضح أنها جوهرية في نهاية المطاف، فينبغي للمساهمين أن ينفصلوا ويطوروا الدرس على النحو الذي يرونه الأفضل. (وهذا أحد أسباب اعتماد رخصة متساهلة.)

ونادرًا ما تحدث هذه الانقسامات في الواقع العملي. وعندما تحدث، من المهم أن نتذكر أننا جميعًا نتشارك الرؤية نفسها: دروس أفضل نبنيها معًا.

الخاتمة:

يقضي المدرّسون في أنحاء العالم كل يوم ساعات لا تُحصى في تكرار العمل الذي يؤديه زملاؤهم. وتقدم هذه القواعد العشر بديلًا: تبنّي نموذج تطوير البرمجيات تعاونيًا لإنشاء دروس أكثر متانة واستدامة في جميع المجالات، يمكن أن يحسّنها باستمرار من يستخدمونها. ونأمل أن تساعد تجاربنا الآخرين على أن يعلّموا أكثر، بأثر أكبر وجهد أقل.

المراجع:

- https://rstudio.github.io/learnr
- https://morea-framework.github.io
- https://www.datacamp.com/teach/documentation
- https://zenodo.org/
- https://www.datacite.org/
- https://orcid.org
- https://creativecommons.org/licenses/by/4.0/

1. Afanador-Llach MJ, Baker J, Crymble A, Gayol V, Gibbs F, Lincoln M, McDaniel C, Milligan I, Parr J, Castro AR, Knuppel AS, Visconti A, Walsh B, Wieringa J. Programming Historian. Accessed: 8 February 2018. https://programminghistorian.org/.
2. Ambrose SA, Bridges MW, DiPietro M, Lovett MC, Norman MK. How Learning Works. Jossey-Bass; 2010.
3. Brownell S. E. and Tanner K. D.: Barriers to faculty pedagogical change: Lack of training, time, incentives, and...tensions with professional identity? CBE Life Sciences Education, 11(4), 2012, pmid:23222828
4. Wiggins G, McTighe J. Understanding by Design. 2nd ed. Association for Supervision and Curriculum Development; 2005.
5. Wilson G. Software Carpentry: Lessons Learned. F1000Research, 3(62), 2016, pmid: 24715981
6. Koch C. and Wilson G. (eds.) Software Carpentry: Instructor Training; 2016. https://zenodo.org/record/57571#.WS8huDOZPdQ.
7. Wilson G. How to Teach Programming (And Other Things). Lulu.com; 2017.
8. GitLab Global Developer Survey. Accessed: 8 February 2018. http://get.gitlab.com/global-developer-survey/, 2016.

9. Jacobs C. T., Gorman G. J., Rees H. E., and Craig L. E.: Experiences With Efficient Methodologies for Teaching Computer Programming to Geoscientists Journal of Geoscience Education 64(3), 2016,
10. Devenyi G. A. and Koch C. (eds.): Software Carpentry: The Unix Shell; 2015. https://zenodo.org/record/27355#.WS8lajOZPdQ.
11. Devenyi G. A. and Srinath A. (eds.): Software Carpentry: The Unix Shell; 2017. https://zenodo.org/record/278226#.WS74tTOZPdQ.
12. Caulfield M. Choral Explanations; 2016. Accessed: 8 February 2018. https://hapgood.us/2016/05/13/choral-explanations/.
13. Fogel K. Producing Open Source Software. O'Reilly; 2005.