BEM

تُعدُّ منهجية BEM وهي اختصار لهذه الكلمات معًا Block و Element و (Modifier مصطلح تسمية شائع يستخدم من أجل الأصناف في HTML و CSS تم تطوير بواسطة فريق فيYandex ؛ هدفها هو مساعدة المطورين في فهم العلاقة بين HTML و CSS في مشروع معين بشكل أفضل.

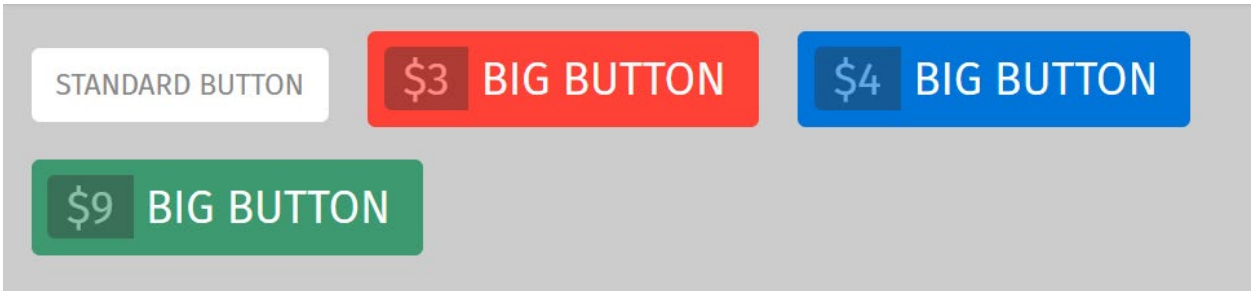
فيما يلي مثال حول ما يكتبه مطور CSS في نمط:BEM

```
/* مكون كتلة */
.btn {}

/* عنصر يعتمد على كتلة */
.btn__price {}

/* مُعدِّل يعدِّل تنسيق الكتلة */
.btn--orange {}
.btn--big {}
```

في شيفرة CSS السابقة، الكتلة (Block) عبارة عن عنصر جديد في أعلى مستوى من التجريد-(top level abstraction)على سبيل المثال زر .btn{}. :يجب اعتبار الجملة البرمجية هذه بمثابة أحد الأبوين، إذ يمكن وضع العناصر الأبناء (elements) في الداخل ويتم الإشارة إليها بواسطة شرطتين سفليتين يتبع اسم الكتلة أو الحاوية البرمجية تلك مثل .btn__price{}. أخيرًا، يمكن للمُعدلات (modifiers) معالجة الحاوية البرمجية (Block) بحيث يمكننا تمييز أو تصميم ذلك المكون المحدد دون إحداث تغييرات على بقية الحاويات البرمجية الأخرى يتم ذلك عن طريق إلحاق شرطتين باسم الحاوية البرمجية Block تمامًا مثل .btn--orange

```
<a class="btn btn--big btn--orange" href="http://css-
tricks.com">
  <span class="btn__price">$9.99</span>
  <span class="btn__text">Subscribe</span>
</a>
```

و كتب مطور آخر هذه الشيفرة ولم يكن لدينا خبرة فيCSS ، فيجب أن نأخذ فكرة جيدة عن أي الأصناف ومسؤوليتها وكيف تعتمد على بعضها بعضًا. يستطيع المطورون آنذاك بناء مكوناتهم الخاصة وتعديل الحاويات (الكتل) الموجودة إلى المحتوى الذي يريدونه. يحتمل بدون كتاب مثل شيفرة CSS هذه أن يتمكن المطورون من كتابة عدة مجموعات مختلفة من الأزرار ببساطة عبر تعيير صنف في الشيفرة:



في البداية، قد تبدو هذه الصياغة أبطأ من بناء صنف جديد لكل نوع من أنواع الأزرار، ولكن هذا غير صحيح لعدة أسباب سنذكرها.

لماذا علينا أخذ BEM في الحسبان؟

1. إذا كنا نريد إنشاء تصميم جديد لعنصر معين، يمكننا أولًا النظر في الحاويات البرمجية لمعرفة أي المُعدِّلات، والعناصرالفرعية موجودة بالفعل. ربما ندرك أننا لسنا بحاجة لكتابة أي تعبير CSS برمجي آخر لأن هناك معدّل موجود مسبقًا يفي بالغرض.
2. إذا كنا نقرأ الوسوم بدلاً من تعابير CSS البرمجية، يجب أن نكون قادرين على الحصول بسرعة على فكرة حول العنصر الذي يعتمد على عنصر آخر؛ في المثال السابق، يمكننا رؤية أن العنصر `btn__price` يعتمد على العنصر `.btn`، حتى لو لم نكن على دراية بعمل أي من تلك العناصر.
3. يمكن للمصممين والمطورين تسمية العناصر لتسهيل الاتصال بين فريق المطورين بمعنى آخر، يوفر BEM لكل مطور في المشروع تسمية صيغة إعلانية للعناصر يمكنه مشاركتها بحيث تكون في نفس الصفحة.

حدد Harry Robertsفائدة رئيسية أخرى لاستخدام صيغة برمجية مثل BEM حين كتب التالي عن تحسين ثقة المطور:

"هذا هو السبب الرئيسي الذي يجعلنا ننتهي من قواعد الشفرة المتضخمة المليئة بشيفرات برمجية قديمة وغير قابلة للتعديل من .CSS نحن نفتقر إلى الثقة في أن نكون قادرين على العمل مع الأنماط الحالية وتعديلها لأننا نخشى عواقب أن تعمل CSS عالميًا، إذ هي مشهورة بحد ذاتها. تتلخص كل المشكلات تقريبًا المتعلقة بـ CSS على نطاق واسع في الثقة (أو عدم وجودها)، لذلك لا يقومون بإجراء تغييرات لأنهم لا يعرفون ما هي الآثار لهذه التغييرات على المدى البعيد".

وبالمثل، يناقش [Philip Walton](#) حل لهذه المشكلة وهو إلتزام عدد كاف من المطورين بمبادئ BEM:

"على الرغم من أن الشيفرة القابلة للتنبؤ بنسبة 100٪ غير متوافرة، من المهم فهم المفاضلات التي تجريها مع الطريقة المتبعة في كتابة التعابير البرمجية التي تختارها. إذا كنت تتبع طريقة BEM الصارمة، فستكون قادرًا على تحديث تعابير CSS وإضافة الشيفرات البرمجية الخاصة بك إليها في المستقبل بثقة تامة بأن هذه التغيرات لن يكون لها أي آثار جانبية".

لذلك، إذا كان بإمكان المطورين العمل في مشروع بطريقة أكثر ثقة، فإنهم على يقين من اتخاذ قرارات أكثر ذكاءً حول كيفية استخدام هذه العناصر المرئية. قد لا تكون هذه الطريقة حلًا مثاليًا لجميع هذه المشاكل، لكنها بالتأكيد تمنح المطورين معيارًا لكتابة جمل برمجية أفضل وأكثر قابلية للصيانة في المستقبل.

هناك جانب آخر جيد في منهجيةBEM ، وهو أن كل مجموعة جمل برمجية خاصة بعنصر معين توجد داخل حاوية خاصة به ولا يوجد شيء متداخل مما يجعل خصوصية CSS سلسة جدًا ومنخفضة وهذا هو المطلوب. أي أنك لن تجهد نفسك فيما يتعلق بخصوصية تعابير CSS البرمجية.

دعونا نلقي نظرة على بعض المشاكل معBEM.

مشاكل مع CSS BEM

لن يقوم أحد بمعاقبتك بالطبع إذا خرجت عن قواعد .BEM لا يزال بإمكانك كتابة محدد CSS كالتالي:

```
.nav .nav__listItem .btn--orange {
  background-color: green;
}
```

يبدو أن المثال السابق يحتوي أجزاء من منهجيةBEM ، لكنه ليس كذلك فهو يحتوي على محددات متداخلة، ولا يصف المُعدِّل (modifier) بدقة ما يحصل في هذه الجمل البرمجية. إذا فعلنا ذلك، فسنكون فشلنا في تحقيق مبدأ الخصوصية وهذا مفيد جدا لمنهجيةBEM.

يجب ألا تَبطل كتلةٌ (مثل `nav`) عمل كتلة أخرى أو معدل آخر (مثل `btn --orange`).وإلا فإن هذا سيجعل من المستحيل تقريبًا قراءة ملف HTML وفهم عمل هذا العنصر. في هذه العملية، نحن ملزمون الى حد كبير أن نكون أهلًا لثقة مطور آخر في البرنامج، هذا ينطبق على HTML أيضًا.

ماذا تتوقع إذا رأيت الشيفرة التالية؟

```
<a class="btn" href="http://css-tricks.com">
  <div class="nav__listItem">Item one</div>
  <div class="nav__listItem">Item two</div>
</a>
```

ربما ما يحدث في المثال السابق، هو أن عنصرًا في الحاوية البرمجية block يشتمل على الشيفرة التي يحتاجها المطور، لكن العناصر الأبناء لا تحتاج إلى الصنف `nav`. على أنه عنصر أب. هذه المشكلة تجعل البرنامج استثنائيًّا وغير متسق ومتناقض مع نفسه وينبغي تجنبها بأي ثمن. لذلك، يمكننا تلخيص هذه المشكلات في:

1. عدم استخدام المُعدلات في الحاويات البرمجية block غير المترابطة.
2. تجنب صنع عناصر رئيسية غير ضرورية عندما يكون العنصر الفرعي موجودًا بشكل جيد بدون أي مشاكل.

عيوبBEM

ربما لا تحب استخدام شرطة مزدوجة. حسنًا، استخدم شيئًا آخر فريدًا تستخدمه باستمرار. هنا رأي آخر:

هذه الثلاثة محددات الأخيرة جميعها لها مستويات خصوصية مختلفة. تحتمل أن يكون لها عنصر رئيسي أو لا بدون أي قواعد معمول بها، هل من الممكن أن يكون هذا المثال الصغير جيد بالنسبة لك؟ ربما. ولكن كلما زاد عدد تعابير CSS البرمجية في المشروع، زاد عدد الأشياء الصغيرة مثل هذه، وبالتالي زادت مشاكل الخصوصية والتعقيد.

ليس بالضرورة اختيار رأي صموئيل هنا، ولكن آراءه شاركها الكثير من الناس لذلك فهو مثال جيد.
بخصوص من يرفضون BEM تمامًا فلا بأس بذلك، لكنني أعتقد أنه سيكون من الصعب القول بأن وجود مجموعة من القواعد التي تساعد في الفهم والحفاظ على CSS قابل للتعديل هو فكرة سيئة.

في منهجية SMACSS، من المحتمل أن تجد اسم صنف CSS متكون من ثلاثة أحرف. تتبع المُعدِّلات بعدئذٍ اسم الوحدة النمطية باستخدام شرطة:

```
/* مثال عن وحدة */
.btn { }

/* btn معدِّل الصنف */
.btn-primary { }

/* الوحدة Btn مع حالة */
.btn.is-collapsed { }
```

هذه مجرد طريقة تسمية مختلفة لنفس المشكلة. إنه مشابه إلى حد ما، لكن تكون أكثر تحديدًا بشأن التبعيات والحفاظ على خصوصية التفاصيل.

في OOCSS، الحاويات البرمجية هي عامة بالغالب.

```
/* مثال عن وحدة */
.mod { }

/* جزء من الوحدة */
.inner { }

/* Talk الوحدة */
.talk { }

/* تغيير جزء داخل الوحدة */
.talk .inner { }
```

لذلك، يمكنك استخدام أصناف متعددة في HTML للحالات المختلفة. لم يتم تسمية الجزء الداخلي مثل التابع له، لذلك فهو أقل وضوحًا ولكنه قابل لإعادة استخدامه. ستقوم BEM بعمل .mod__inner و.mod--talk و..mod-talk__inner

هذه مجرد اختلافات في المنهجية. تذكر أن لا أحد يجبرك على استخدامها، فهذه قواعد مفروضة ذاتيًا حيث تأتي القيمة من متابعتها.

BEMوSass

لأولئك الذين يستخدمون Sass ويستمتعون بتشعيب (nesting) العناصر كوسيلة لتحديد النطاقات لتنسيقات، لازال بإمكانك أن تكون المسؤول عن الصيغة المتشعبة ولكن يمكنك الحصول على CSS غير متشعب، باستخدام :@at-root

ملف:Scss

```
.block {
  @at-root #{&}__element {
  }
  @at-root #{&}--modifier {
  }
}
```

يولد ملف CSS التالي:

```
.block {
}
.block__element {
}
.block--modifier {
}
```

الخلاصة

أعتقد أنه من الإنصاف القول إنه على الرغم من أن BEM لن يحل جميع مشاكلنا، إلا أنه مفيد بشكل كبير في بناء واجهات قابلة للتطوير والصيانة حيث يجب أن يكون لدى كل فرد في الفريق فكرة واضحة عن كيفية تطوير تلك الأشياء. ذلك لأن الكثير من التطوير في الواجهة الأمامية لا يتعلق فقط بالخدع اللطيفة التي تحل مشكلة صغيرة واحدة على المدى القصير؛ نحتاج إلى اتفاقات وعقود ملزمة بين المطورين بحيث يمكن لبرنامجنا التكيف مع مرور الوقت وعلى المدى البعيد.

SEARCH 3PROPRTIS IN CSS

1 : Text Align

توفر CSS العديد من الخصائص التي يمكنك استخدامها لتحسين طريقة ظهور النصوص و جعلها تظهر بالشكل الذي تريده بالضبط، فمثلاً يمكنك تغيير لون النص، تغيير إتجاه النص، عكس إتجاه الأحرف و غيرها من الأمور التي سنتعرف عليها في هذا الدرس.

The text-align Property

text-align: center:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

text-align: left:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

text-align: right:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

text-align: justify:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

2 : Margin

هو المسافة الفارغة التي يمكن وضعها حول العنصر من الخارج لتحديد بعده عن العناصر الأخرى الموضوعة في الصفحة.

حجم الهامش الخارجي للعنصر يمكنك تحديده من كل الجهات أو من جهات محددة.

The margin Property

A paragraph with no specified margins.

This paragraph has a margin of 35 pixels on all four sides.

A paragraph with no specified margins.

3 : float

يقصد به تحديد ما إن كانت العناصر ستظهر من الجهة اليمنى أو اليسرى، الأمر الذي يجعلك قادر على التحكم بمكان ظهور العناصر الموضوعة في ذات المكان.

The float Property

In this example, the image will float to the right in the text, and the text in the paragraph will wrap around the image.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.