Layout style guide

Inspired by Google's styleguide

TOP uses Markdown for the layout and formatting of lesson and project files to get properly formatted HTML for the TOP website.

The goals of this style guide are to help create Markdown that is:

- Readable for as many users as possible.
- Editable by any contributor.
- Consistent across the TOP website.

A note on language: TOP follows American English and American style punctuation. When adding content to the curriculum, be sure to follow this practice for consistency across lessons.

Table of Contents

Layouts

In general, the following layouts should be used for all lessons and projects. Text that should be replaced with the author's own content will be in all CAPS, with any additional information regarding a section listed at the end of the layout code block.

When adding new lessons or projects, make a copy of either the lesson template or the project template in the appropriate folder where the new lesson/project should be placed. Then begin editing the template copy.

The lesson example and project example files both show how this style guide can be put to use in an actual lesson/project. They don't cover every situation (the lesson example doesn't show a lesson with an assignment and one without, for example), but they should give you a better representation of how lessons/projects should look after this style guide is applied.

Lesson layout

```
### Introduction

A BRIEF INTRODUCTION.

### Lesson overview

This section contains a general overview of topics that you will learn in this
lesson.

- A LESSON OVERVIEW ITEM.

### CUSTOM SECTION HEADING

CUSTOM SECTION CONTENT.

### Assignment

<div class="lesson-content__panel" markdown="1">

#### OPTIONAL CUSTOM ASSIGNMENT HEADING

1. A RESOURCE OR EXERCISE ITEM
   - AN INSTRUCTION ITEM

</div>

### Knowledge check

This section contains questions for you to check your understanding of this lesson
on your own. If you're having trouble answering a question, click it and review
the material it links to.

- [A KNOWLEDGE CHECK QUESTION](A-KNOWLEDGE-CHECK-URL)

### Additional resources

This section contains helpful links to related content. It isn't required, so
consider it supplemental.

- It looks like this lesson doesn't have any additional resources yet. Help us
expand this section by contributing to our curriculum.
```

1. `### Introduction`: A brief summary on what the lesson is about and/or why the topics or concepts it covers are important. Replace the `A BRIEF INTRODUCTION.` text with your own lesson introduction.
2. `### Lesson overview`: A bulleted list of items that provide a general overview of what the user will learn about in the lesson. Lesson overviews should include general, higher level statements that cover the core concepts of the lesson. They should serve and be phrased as a list of key items that a user

should be expected to *learn about* throughout the lesson, rather than a list of things they should be able to *do* by the end of it.

Replace the `A LESSON OVERVIEW ITEM.` text with your own lesson overview item, then add any additional bulleted lesson overview items. The lesson should ideally have no more than 7 lesson overview items, but this number might vary by lesson. **If the lesson does not have a lesson overview, remove this entire section from the lesson.**

3. `### CUSTOM SECTION HEADING`: A custom section that contains some of the main content of the lesson. Replace the `CUSTOM SECTION HEADING` text with a proper section heading and the `CUSTOM SECTION CONTENT.` text with your own content, then add any additional custom sections. **If the lesson does not have any custom sections, remove this entire section from the lesson.**

4. `### Assignment`: A numbered list of external resources the user must read or watch, or practical exercises the user must complete (such as our exercise repos), in order to fully complete the lesson. If an assignment is intended to have multiple lists, each list should include a level 4 heading by replacing the `#### OPTIONAL CUSTOM ASSIGNMENT HEADING` with a proper level 4 heading, otherwise this custom heading can be omitted.

   Each assignment item should include some brief text that further informs the user on why it is included in the assignment or what purpose it serves. When necessary, an assignment item should also explicitly state any instructions that should be followed. Examples of instructions can include (but aren't limited to) a specific section the user should read, whether the user should complete any specific exercises, and whether the user should redirect themselves to additional links within the resource.

   Replace the `A RESOURCE OR EXERCISE ITEM.` text with your own text and a link to the resource or exercise (or any applicable instructions if an exercise isn't external), then add any additional numbered assignment items. The lesson should ideally have no more than 3-5 assignment items (reading several sections on a web page or completing a folder of 5 exercises would be considered a single assignment item). **If the lesson does not have an assignment, remove this entire section from the lesson.**

   If an assignment item includes any instructions, replace the `AN INSTRUCTION ITEM` text with a single instruction, then add any additional bulleted instruction items.

   If a user should only read specific sections within a resource (e.g. "Skip Chapter 7") or complete only specific exercises (e.g. "Complete the first two exercises in the repo"), each instruction item should be its own bullet.

   If an assignment item does not have any instructions, remove the bulleted `AN INSTRUCTION ITEM` text from it.

5. `### Knowledge check`: A bulleted list of specific questions that a user should be able to answer on their own after reading the lesson and completing any assignment or practice. A knowledge check should only link either to a section within the lesson (either with a Heading 3 `###` or Heading 4 `####`, or by wrapping text in a `<span>` element with an `id` attribute), or a resource previously linked to in the lesson. This link should help users review the necessary material in order to answer the knowledge check without requiring them to re-read the entire lesson.

   Replace the `A KNOWLEDGE CHECK URL` text with the actual link to the section/resource and the `A KNOWLEDGE CHECK QUESTION.` text with your own question/problem that the user should be able to

answer/solve. Then add any additional bulleted knowledge check items. The lesson should ideally have no more than 7 knowledge checks, but this number might vary by lesson. **If the lesson does not have a knowledge check, remove this entire section from the lesson.**

In order to link to a Heading 3 `###` or Heading 4 `####` within the lesson, replace the value within the parenthesis for the knowledge check link with a hashtag `#` followed immediately by the section title in lowercase and any spaces replaced with a hyphen `-`. For example, a Heading 3 section titled `### Creating a method` would be linked to with `(#creating-a-method)`.

In order to link to a `<span>` element within the lesson, replace the value within the parenthesis with the exact `id` attribute of the `<span>` element (this will be case sensitive). For example, a `<span id="Knowledge-Check-3">` element would be linked to with `(#Knowledge-Check-3)`.

6. `### Additional resources`: A bulleted list of optional resources for the user to read. Additional resources should be related to the content of the lesson in some way, without being necessary to gain an understanding of the lesson content. An additional resource should include brief text that further informs the user on why it is included or what purpose it serves, and generally should stand out in some way from the lesson content and other additional resources. A good rule of thumb is to try and answer, "what purpose does this resource serve?"

   If the lesson doesn't include any additional resources, leave this section as-is. Otherwise, replace the default bulleted resource item with your own resource, then add any additional bulleted resource items. The lesson should ideally have no more than 3-5 additional resources.

Project layout

```
### Introduction

A BRIEF INTRODUCTION.

### OPTIONAL PRE-ASSIGNMENT SECTION HEADING

OPTIONAL PRE-ASSIGNMENT SECTION CONTENT.

### Assignment

<div class="lesson-content__panel" markdown="1">

#### OPTIONAL CUSTOM ASSIGNMENT HEADING

1. A REQUIREMENT/USER STORY.

#### Extra credit

- AN OPTIONAL ADD-ON/USER STORY.

</div>

### OPTIONAL POST-ASSIGNMENT SECTION HEADING
```

```
OPTIONAL POST-ASSIGNMENT SECTION CONTENT.
```

1. `### Introduction`: A brief summary on what the project is and an overview of what the user will be building. Replace the `A BRIEF INTRODUCTION.` text with your own project introduction.

2. `### OPTIONAL PRE-ASSIGNMENT SECTION HEADING`: A section that contains content that should come before the actual project assignment. This section will most likely not be needed for most projects, but when it is needed simply replace the `OPTIONAL PRE-ASSIGNMENT SECTION HEADING` text with a proper section heading and the `OPTIONAL PRE-ASSIGNMENT SECTION CONTENT.` text with your own content. Then add any additional pre-assignment sections. **If the project does not have a pre-assignment section, remove this entire section from the project.**

3. `### Assignment`: A numbered list of items that describe detailed requirements or user stories that must be followed in order to complete the project. Replace the `A REQUIREMENT/USER STORY.` with your own requirement, then add any additional numbered requirement items. If an assignment is intended to have multiple lists, each list should include a level 4 heading by replacing the `#### OPTIONAL CUSTOM ASSIGNMENT HEADING` with a proper level 4 heading, otherwise this custom heading can be omitted.

4. `#### Extra credit`: A bulleted list of items that describe any optional add-ons or user stories that might make a user's project stand out. Replace the `AN OPTIONAL ADD-ON/USER STORY.` text with your own add-on, then add any additional bulleted add-on items. **If the project does not have any extra credit items, remove the extra credit section from the assignment.**

5. `### OPTIONAL POST-ASSIGNMENT SECTION HEADING`: A section that contains content that should come after the actual project assignment. This section will most likely not be needed for most projects, but when it is needed simply replace the `OPTIONAL POST-ASSIGNMENT SECTION HEADING` text with a proper section heading and the `OPTIONAL POST-ASSIGNMENT SECTION CONTENT.` text with your own content. Then add any additional post-assignment sections. **If the project does not have a post-assignment section, remove this entire section from the project.**

Headings

Case

Headings should always use sentence case:

```
<!-- Incorrect -->
### This Is Not Sentence case

<!-- Correct -->
### This is sentence case

<!-- Correct -->
### This is also sentence case with HTML
```

No code snippets

Headings should never contain any code snippets.

```
<!-- Incorrect -->
### The `id` property

<!-- Correct -->
### The id property
```

ATX-style headings

Use Heading 3 `###` for main section titles ("Lesson overview", "Assignment", custom sections, etc):

```
### Section heading
```

Sub-heading

Use either Heading 4 `####` for sub-headings that are on their own line or Markdown's bold syntax, e.g. `**Sub-heading**`, for inline sub-headings:

```
...text before.

#### Sub-heading

Text after...

**Inline sub-heading:** Some text defining this sub-heading...
```

Newlines

Each Markdown file should have an empty newline at the very end, after all of the file's contents.

Always add a newline before and after a heading, a list, an Assignment panel, or any other content that is not strictly text:

```
Content before...

### Section heading

1. A list item

...content after.
```

Lists

Lazy numbering

Markdown is smart enough to let the resulting HTML render your numbered lists correctly. For longer lists that may change, especially long nested lists, use "lazy" numbering. The following Markdown:

```
1. Foo.
1. Bar.
1. Foofoo.
1. Barbar.
1. Baz.
```

Will result in the following output:

1. Foo.
2. Bar.
3. Foofoo.
4. Barbar.
5. Baz.

Nested lists

When nesting lists, use a 2 space indent when nesting inside a bulleted list and a 3 space indent when nesting inside a numbered list. The following Markdown:

```
1. The first item
2. A second item
   - A sub-item for the second item with 3 spaces before the hyphen

- A bulleted list item
  - A sub-bullet with a 2 space indent
- A new list item
```

Will result in the following output:

1. The first item
2. A second item
    - A sub-item for the second item with 3 spaces before the hyphen

- A bulleted list item
    - A sub-bullet with a 2 space indent
- A new list item

Multi-line list items

When list items should wrap onto multiple lines – such as to create a line break between a lengthy list item – insert an empty line before and after each wrapped line and use a 2 to 3 space indent on the wrapped line.

You should use a 2 space indent for bulleted lists and a 3 space indent for numbered lists. The following Markdown:

```
1. This is a lengthy list item.

   This is related information to the first item, but visually separated out.

2. A new list item

- This is a lengthy bulleted list item.

  This is related information to the first item, but visually separated out.

- A new bulleted list item
```

Will result in the following output:

1. This is a lengthy list item.

   This is related information to the first item, but visually separated out.

2. A new list item

- This is a lengthy bulleted list item.

  This is related information to the first item, but visually separated out.

- A new bulleted list item

Unordered lists

The preferred way to create unordered lists for The Odin Project is by using hyphens `-`. Both hyphens and asterisks give the same results, but sticking to one way keeps the source markdown consistent.

```
- This is a list item made using a hyphen.
- This is a list item made using a hyphen.
- This is a list item made using a hyphen.
```

Code

Inline

`Backticks` designate `inline code`, and will render all wrapped content literally. Use them for short code quotations, field names, or file names:

```
Write these in the `script` tag of a skeleton HTML file.

...which is why we can call `taco.printString()` but not
```

```
`taco.capitalizeString()`.

Create a new file named `styles.css` first.
```

Codeblocks

For code quotations longer than a single line, use a codeblock with 3 opening and closing tilde marks:

```
~~~javascript
const obj = {
  name: "object",
  marker: "X"
}
~~~
```

Declare the language

It is best practice to explicitly declare the language immediately after the opening tilde marks, so that neither the syntax highlighter nor the next editor must guess.

No extraneous characters

Codeblocks should only contain actual code snippets, terminal commands, or commented out text. Never include leading terminal content, such as the dollar sign `$` you might see preceding any commands you type in.

```
# Incorrect
$ cd Documents

# Correct
cd Documents
```

Nest codeblocks within lists

If you need a codeblock within a list, you should follow the same indenting rules for multi-line list items, with the codeblock being indented with 2 spaces for a bulleted list item and 3 spaces for a numbered list item. The following Markdown:

```
- Bullet.

  ~~~javascript
  // We start indenting with 2 space for the codeblock
  function tester() {
    const yay = 'From here we can indent like we normally would'
  }
```

```
  ~~~

- Next bullet.
```

Will result in the following output:

- Bullet.

  ```
  // We start indenting with 2 space for the codeblock
  function tester() {
    const yay = 'From here we can indent like we normally would'
  }
  ```

- Next bullet.

Note boxes

Note boxes can be added by wrapping the content in a `div` with the class `lesson-note`. This will add styling to make the note stand out visually to users.

A heading can be added to a note by using an `h4` element. When adding a heading, be sure to provide text that helps describe the note rather than "A note" or "Warning".

Variations

Note boxes come in two variations, which can be set by adding an extra class together with `lesson-note`:

- `lesson-note--tip` for tips or general information
- `lesson-note--warning` for warnings about potential issues/pitfalls, and are more severe than a tip

Example

```
<div class="lesson-note">
<h4>An optional title</h4>
A sample note box.
</div>
```

```
<div class="lesson-note lesson-note--tip">
<h4>An optional title</h4>
A sample note box, variation: tip.
</div>
```

Links

Long links make source Markdown difficult to read and break the 80 character wrapping. **Wherever possible, shorten your links**.

Use informative Markdown link titles

Markdown link syntax allows you to set a link title, just as HTML does. Use it wisely.

Titling your links as "link" or "here" tells the reader precisely nothing when quickly scanning your doc and is a waste of space. Instead, write the sentence naturally, then go back and wrap the most appropriate phrase with the link:

```
See the [lesson template](./templates/lesson-template.md) for a more easily
copyable lesson file.
Or, check out the [project template](./templates/project-template.md) for a more
easily copyable project file.
```

Don't scatter links throughout lessons

Links to required reading should not be scattered throughout a lesson, and should instead be placed in either the `### Assignment` or `### Additional resources` section. Links that refer a user to a previous lesson as a refresher, or a link to a Wikipedia page that offers a definition/explanation of a term are fine to place outside of these two sections.

Images

Images in Markdown follow the same syntax as links, except they begin with an exclamation point `!`:

```
![flex shorthand]
(https://cdn.statically.io/gh/TheOdinProject/curriculum/495704c6eb6bf33bc927534f23
1533a82b27b2ac/html_css/v2/foundations/flexbox/imgs/10.png)
```

The text in square brackets will be included as the image's alt text. Similar to link titles, the alt text should be informative, but shouldn't be overly verbose.

In order to properly add images to a lesson, follow the instructions in our Adding Images to the Curriculum Wiki page to get a statically URL as seen in the codeblock above.

Keyboard shortcuts

For keyboard shortcuts we use the HTML keyboard input element `<kbd>`.

Example code which will be rendered as: `Ctrl + Shift + ?`

```
<kbd>Ctrl</kbd> + <kbd>Shift</kbd> + <kbd>?</kbd>
```

Style standardization

- Use separate `<kbd>` elements for individual keys:

```
<kbd>Ctrl</kbd> + <kbd>Shift</kbd>
```

- Use capitalized common abbreviations for the keys and avoid using symbols like `⌘`:

```
<kbd>Cmd</kbd>
<kbd>Alt</kbd>
<kbd>B</kbd>
<kbd>Opt</kbd>
```

- Use symbols for character keys instead of spelling out the symbol like `period`:

```
<kbd>.</kbd>
<kbd>,</kbd>
<kbd>Ctrl</kbd> + <kbd>Shift</kbd> + <kbd>?</kbd>
```

Codepen embeds

In order to embed a Codepen example into a lesson, you must be in the editor view for the Codepen you wish to embed and then click the `Embed` button at the bottom right of the page.

The following options should be selected when creating a Codepen embed:

- **Default Tabs**: The "Result" tab must be selected in addition to one of the other three options (HTML, CSS, or JavaScript), depending on the main purpose of the Codepen. If the purpose is to show an HTML concept then the "HTML" option must also be selected, for example.
- **Theme**: "Dark"
- **Use Click-to-Load**: "Off"
- **Make Code Editable**: "On"

Finally, the **HTML (Recommended)** code option must be the one that is copy + pasted into the lesson.

Maintainer instructions

When a user adds a Codepen embed to a lesson, a maintainer should fork the embed to the official TOP Codepen account. When necessary, the name of new pens should be updated to better reflect their purpose, e.g. `Simple SVG Example` for a pen showing a simple SVG or `max-width | CSS Responsiveness` for a pen about the `max-width` property.

After forking a pen to the TOP account and ensuring the embeds options from above are selected, the lesson the original embed is from should be updated to include the forked, TOP version instead.

Mermaid diagrams

To add a Mermaid diagram to a lesson, visit the Mermaid docs to learn the diagram syntax for the specific type of diagram you want to add. After you've figured out the content you want in the diagram, you can add it to a lesson's markdown by surrounding the content with `<pre>` tags with a `class="mermaid"` ie:

```
<pre class="mermaid">
  mermaid diagram content here
</pre>
```

This has full support in the Lesson Preview tool, so be sure to check that the diagram renders correctly with the lesson content before contributing.

Markdown styling

While Markdown supports the use of both asterisks `*` and underscores `_` to make text bold or italic, asterisks should always be used.

```
**This** is how *we* do things.

__This__ is _not_ how we do things.
```