CODE AVENGERS HTML/CSS 2 Instructions

1. Intro to level 2 HTML5/CSS3

1.1. doctype and title

The page labeled test2.html is not part of your travel plan, but is for you to experiment with.

This lesson reviews some HTML from level 1 as you write the index.html page.

For task 1, let's add the two **elements** that are needed to make a valid HTML page:

1. Add the HTML5 `doctype` to the top of index.html.
2. Add a `title` tag with the text:
   "Index, Code Avengers Travel"

Your web page will still be blank after adding these tags.

1.2. heading

This course follows the Google style guide, a set of rules for writing code that is easy to understand. Read our **style guide summary** for an overview.

For this task, do the following:

1. Choose a cool name for your 3 day holiday.
2. On the index page, add a level 1 **heading** with your holiday name.

1.3. logo

Now you'll put the Code Avengers logo above your heading.

1. On a new line between the title and heading elements, add an **image** with the source attribute set to: `/images/caLogo.png`
2. Set the image **alt** attribute to: "CA logo"
3. Add an image **tooltip** that says: "Code Avengers"

1.4. anchors and break

The index.html needs links to the other 4 HTML pages. The example page uses the following code for the 1st link: `<a href="plan.html">Plan</a>`

The link to the Code Avengers home page needs a / at the start because its in a different folder than your index page. A / is not needed when linking to a page in the same folder.

1. Add 4 **anchor** tags with the text shown on the example page.
2. Link each of the tags to the other page by setting the `href` attribute.
3. Add a **break** tag between the anchor elements to put them on separate lines.

1.5. doctype and title for each page

For the last task in this lesson you'll add some code to the other 4 HTML pages. Then you can check that your links work correctly.

1. Add a `doctype` to each HTML page.
2. Add a `title` tag to each page, with the text "Plan, Code Avengers Travel", "Day 1, Code Avengers Travel" etc.
3. Add a level 1 **heading** to each page and set the text to: "Plan", "Day 1 Activities" etc.
4. Click the links on index.html to check that they go to the correct pages. Click to go back to the index page.

2. External CSS stylesheet

2.1. external stylesheets

In lessons 1-10 of the level 1 course you added style to your pages by linking to an **external** css style sheet that we created for you. Then in lessons 11-20 you added style by putting CSS code directly in your HTML code inside `<style>` tags.

In this course you will write your own **external** CSS style sheet called styles.css, and link to it with the `link` tag.

1. Add a **stylesheet link** tag on line 3 of index.html and set the `href` attribute to styles.css.
2. Add a body tag rule to styles.css that sets the `background-color` of index.html to **lightGreen**.

2.2. heading style

All the CSS code for your travel guide will go in styles.css.

Now add style to the heading by adding a CSS rule for the `h1` tag with the following properties:

1. **Font size** of **25** pixels.
2. **Color** of **dimGray**.
3. Use the **BenchNine** Google font by following **these steps**.

2.3. background image

Now you will add a semi-transparent image to the background of your page.

There are **15 images to choose from**. The first image has the URL `/images/textures/light1a.png`, and the last one is named **light15a.png**.

1. Set the **background image** for your body tag to one of the semi-transparent images.
2. Add a property to the body tag rule to **align the text** to the center of the page.
3. Use the Verdana **web-safe font** in the body tag rule.

2.4. text style

Now let's style the text some more.

1. Add a white **text shadow** to the heading, shifted down and across **1 pixel** with a **2 pixel** blur.
2. Add a CSS rule to make the **font size** for the anchor tags **18 pixels**.
3. Add a property to the anchor tag rule to remove the **underline**.

2.5. external stylesheet link

One reason to put CSS in an external stylesheet is so you can use it on every page in your site.

1. Add the stylesheet **link** on index.html to each page in your website.
2. Feel free to change fonts on your site using our **web-safe fonts list** and **Google fonts instructions**.

3. Audio and video

3.1. audio controls

In this lesson let's add **audio** and **video** to our page.

One of the new tags in HTML5 is the `audio` tag. The audio tag has several attributes that change the way it functions.

The `controls` attribute is used to make the browser show controls that allow the user to control the audio volume and playback.

1. Add an audio player to index.html by adding the code `<audio controls></audio>` at the bottom of the page.
2. See what happens when you remove the attribute `controls` from the `audio` tag.

3.2. audio source

Now let's set the audio file that plays by putting `source` elements inside the audio element.

The following code adds the Code Avengers theme song to the home page.
```
<audio controls>
  <source src="/audio/CodeAvengersTheme.ogg"
    type="audio/ogg">
  <source src="/audio/CodeAvengersTheme.mp3"
    type="audio/mpeg">
</audio>
```

The `source` element tells the **web browser** which audio file to play.

You need 2 source elements, because no audio format is supported by all browsers, mainly due to legal reasons. The **Ogg** format is not supported by **IE** and **Safari**. **MP3** is not supported by **Opera**. Firefox has full support for Ogg and limited support for MP3.

The `type` attribute is **recommended but not required**.

1. Re-add the `controls` attribute to the `audio` tag.
2. Add 2 `source` elements (with `type` attributes) to play the files located at:
   /audio/background1.ogg
   /audio/background1.mp3

3.3. `autoplay` and `loop` attributes

In addition to the `controls` attribute, you can add the `loop` attribute to make the music re-play from the start as soon as it ends.

The `autoplay` attribute makes the music play as soon as the page loads. If you use this attribute, it's a good idea to make sure the music plays quietly in the background, so it does not annoy your visitors.

1. Add the `autoplay` and `loop` attributes to the `audio` tag.
2. There are **17 background music files** named **background1** to **background17**; choose the best music for your site by changing /audio/background1.ogg and /audio/background1.mp3.
3. In order to test the `autoplay` attribute, click the link at the top of the phone to open your web page in a new tab.

3.4. `iframe` element

The `iframe` element (short for **inline frame**) is used to put another web page onto your page. In this task you will use the `iframe` element to put a YouTube video player on your page.

The video in the example was added with the following code:

```
<iframe frameborder="1" height="140" width="340"
  src="http://www.youtube.com/embed/64qx95Ckrwc">
  </iframe>
```

1. Find a video on YouTube and add it to index.html; **click here** for instructions.
2. Set the `width` attribute of the `iframe` to **320** pixels.
3. Set the **height** to a value between **150** and **200** pixels.
4. Set the width of the **frameborder** to **0**.

Video can also be played using the HTML5 video element.

3.5. Wikipedia `iframe`

Websites like Facebook and Google don't let you show their pages in an `iframe`. If you try to do it, you'll get an empty box on your page. However, most sites work with iframes.

For example, the following code is used to put a Wikipedia article about the **letter A** on your page:

```
<iframe
  src="http://wikipedia.org/wiki/A?printable=yes">
</iframe>
```

Putting `?printable=yes` at the end of a Wikipedia URL hides the logo and toolbars, so that only the article contents are shown.

1. Add a Wikipedia page to index.html using **these instructions**.
2. Use the `width` attribute to make it the same width as the video.
3. Use the `frameborder` attribute to turn off the border.
4. In styles.css add a rule to set the `iframe` background color to **white**.

4. Comments in HTML & CSS code

4.1. HTML comments

It is a good idea to write **comments** in your HTML code to describe and explain complicated code.

Put `<!--` and `-->` around comments so that they **don't** appear on the web page.

E.g. *`<!-- This comment is hidden -->`*

1. Add a comment above the code for the `iframe` YouTube video to explain what it is about. Start the comment with:
   "This video . . ."
2. Add another comment above the `audio` element to explain what type of music it is. Start the comment with:
   "This music . . ."

4.2. comment out code

Comments are also used to hide parts of your page that you want to show again later; this is called **commenting out** code.

The `iframe` elements slow the loading of the page, so for now you will hide them by surrounding each iframe element with `<!--` and `-->`

1. Comment out the video `iframe`.
2. Comment out the Wikipedia `iframe`.

4.3. todo comments

Comments are also used to add reminders of things **to do** later.

The Google style guide says that these comments should begin with `TODO:`

E.g. *`<!-- TODO: Make the links look cooler -->`*

1. Add the example **todo** comment above to index.html.
2. Add a 2nd **todo** comment that says: "unhide the video and article"

4.4. author comments

When you are working in a team, you can put your name and email address in a comment at the top of the page, so other team members know who created each page.

1. Add a comment with your name and email after the `doctype` in each html file.
2. Add a TODO comment to plan.html that reminds you to add a table with the daily schedule.
3. Think of a fun activity for each day of your travel plan. Describe the activity for each day in a comment (of at least 5 words) in day1.html, day2.html and day3.html.

4.5. CSS comments

Comments look different in **CSS** and **HTML**. A CSS comment starts with `/*` and ends with `*/`

E.g. *`/* This is a CSS comment`*
`that is split over two lines*/`

1. Add a comment to styles.css with your name and email address.
2. Add a 2nd comment to styles.css that says anything you like.

5. Spacing with [[padding]] and [[margin]]

5.1. styled images

In this lesson you will add space around HTML elements using the CSS `margin` and `padding` properties.

For your 1st task:

1. On separate lines of code, add 4 **images** with file paths: **/images/daughter10.jpg** to **/images/daughter13.jpg**
2. Add a `<style>` **element** (i.e. start and end tag) to put CSS code inside.
3. Add a CSS rule to set the image **width**s to **100** pixels.
4. Add a solid **blue border** with a thickness of 4 pixels to the images.

To remove the space between the images, put them on a single line of code.

5.2. padding and margin

The `padding` property adds space between an image and its border.

The `margin` property adds space outside the border of the image.

See what happens when you add the following properties to the image CSS rule:

1. `padding` of **8** pixels.
2. `margin` of **8** pixels.
3. `background-color` of **red**.

5.3. padding and margin longhand

The `padding` property is shorthand for setting `padding-top`, `padding-right`, `padding-bottom` and `padding-left` separately.

The `margin` property is shorthand for setting `margin-top`, `margin-right`, `margin-bottom` and `margin-left`.

1. Remove the `margin` and `padding` shorthand properties from the CSS image rule.
2. Add 4 properties to set `padding-top` to 0 pixels, `padding-right` to 2 pixels, `padding-bottom` to 4 pixels and `padding-left` to 8 pixels.
3. Add 4 properties to set `margin-top` and `margin-bottom` to 12 pixels; set `margin-right` and `margin-left` to 6 pixels.

When the value is 0 pixels you can use `0` instead of `0px` in your CSS code. The Google style guide recommends `0`.

5.4. padding and margin shorthand

There are 4 versions of the `padding` shorthand property. Here are examples of each:

`padding: 1px 2px 3px 4px;` set padding **top** to 1px; **right** 2px; **bottom** 3px; **left** 4px.

`padding: 1px 2px 3px;` top 1px; right & left 2px; bottom 3px.

`padding: 1px 2px;` top & bottom 1px; right & left 2px.

`padding: 1px;` top, right, bottom & left 1px.

The same rules apply to the `margin` property.

1. Replace the 4 `padding` rules with a shorthand rule that has the same values.
2. Replace the 4 `margin` rules with a shorthand rule that has the same values.

The order for the shorthand properties starts at the **top** and goes clockwise to **right**, **bottom**, **left**.

5.5. overriding rules

If you set 2 or more values for the same property, the one that appears last is used. For example:

```
img {
  padding: 4px;
  padding: 8px;
}
```

This code sets the image padding to 8 pixels **not** 4 pixels.

1. Set `padding-top` to **24** pixels on a line after the `padding` shorthand property from task 4.
2. Set `margin-top` to **24** pixels on the line after the `margin` shorthand property from task 4.

6. 3 display types of HTML elements

6.1. `display` type

This lesson continues from the previous lesson.

Each HTML element has a `display` type. So far you've used elements with 4 display types:

Block: `audio`, `footer`, `h1`, `header`, `hr`, `ol`, `p`, `section`, `ul`
Inline: `a`, `br`, `iframe`, `strong`
Inline-block: `img`
None: `link`, `meta`, `source`, `style`, `title`

Elements with a `display` type of none are **not** shown on the web page.

The rest of this lesson explains the difference between the other 3 types.

For your 1st task:

1. Above the images, add a level 2 heading that says: "Cute Photos"
2. Under the images add a **paragraph** tag followed by this text: "Pictures from FocusStudio"
3. Link the name "FocusStudio" to: http://focusstudio.co.nz
4. Make the word "Pictures" bold using `strong` tags.
5. Add a 3 pixel solid **blue** **border** around the `h2` and `p` elements.
6. Add a 3 pixel solid **lime** **border** around the `a` and `strong` elements.
7. Remove the `margin` CSS properties for the `img` tag.

6.2. `block` elements

`block` elements (e.g. `h2`, `p`) expand to the width of the page; their height expands to fit the text and elements they contain.

Some `block` elements (e.g. `h2`, `p`) automatically have `margin` space above and below them. The size and spacing can be changed by setting the `width`, `height`, `margin` and `padding` CSS properties.

1. Remove the space above the heading and paragraph by setting `margin-top` to **0** pixels
2. Set the `padding` for the heading and paragraph to **10** pixels.
3. Set the `height` of the paragraph to **40** pixels.
4. Set the `width` of the heading to **120** pixels.

6.3. `inline` elements

`inline` elements (e.g. `a`, `strong`) expand to the width and height of their contents. The size of inline elements **can't** be changed with the `width` and `height` CSS properties.

The `margin` and `padding` CSS properties will add space around an inline element. However, only the left and right properties will change the position of the text.

1. Set the `width` and `height` of `a` and `strong` to **80** pixels; see that nothing changes.
2. Set the `padding-top` and `padding-bottom` to **20** pixels.
3. Set the `padding-left` and `padding-right` to **15** pixels.
4. Set the `margin-left` and `margin-right` to **10** pixels.

6.4. `inline-block` elements

The rules for **`inline-block`** elements (e.g. `img`) are a combination of `block` and `inline`.

As with `block` elements, you can change the size and spacing of an `inline-block` with the `width`, `height`, `margin` and `padding` properties.

One difference is that `block` elements naturally **drop below** other elements, whereas `inline` and `inline-block` elements are put next to previous elements if there is space.

1. Copy the heading code (`<h2>Cute Photos</h2>`) and put directly under the first `h2` heading. **What happened?**
2. Change the width of the images to 60 pixels. **What happened?**
3. You should see a small space between the images. Remove the space by putting all the image tags next to each other on 1 line of code. **What happened?**

6.5. changing display type

Each HTML **element** has a default display type. The `display` CSS property is used to change an elements type.

Make the following changes to display type. Before you make each change, see if you can guess what will happen.

1. Set the `display` property for the headings to `inline`. **What happened?**
2. Set the `display` property for the headings to `inline-block`. **What happened?**
3. Set the `display` property for the **images** to `block`. **What happened?**

7. Make an [[anchor]] look like a button

7.1. button style

In this lesson you will write CSS code in styles.css to make the links on index.html look like buttons.

For your 1st task set the following properties for the **anchor** tag:

1. **Background color** of **lawnGreen**.
2. Text **color** of **darkGreen**.
3. A solid **limeGreen** **border** with a thickness of 1 pixel.
4. A **white text shadow** shifted right 0 and down 1 pixel.

7.2. button `width`

To make each button the same width, you need set 2 attributes: `width` and `display`.

Before we use the `display` attribute, see what happens if you set `width` **without** setting `display`.

1. Set the **width** for the **anchor** tag to **250** pixels.
2. Set the **background color** of `h1` to **red**.
3. Set the **width** for the `h1` tag to **200** pixels.

7.3. `block` elements

Remember that each HTML element has a display type. For example, `h1` is a `block` element; **anchor** is an `inline` element.

The `width` property works for `block` elements, but not for `inline` elements. So to make the `width` property work, you need to set the anchor tags `display` property to `block`.

1. Remove the `background-color` and `width` properties from the `h1` tag.
2. Set the `display` property of the anchor tags to `block`.
3. Add rounded corners by setting `border-radius` property of the anchor tags to **5** pixels.

7.4. increase the button heights

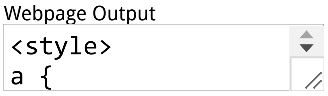
Now that the button widths are set correctly, let's increase their height.

One way to do this is by setting the `height` property, as shown in the following example:

HTML Code
```
<style>
a {
  border: 1px solid
blue;
  display: block;
  text-align: center;
  height: 60px;
  width: 120px;
}
</style>

<p><a>Button 1</a>
<p><a>Button 2</a>
```

Webpage Output


```
<style>
a {
```

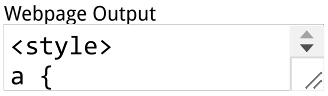
The problem with setting the height is that we want the text to be in the middle of the button, **not** at the top.

Instead of setting the `height` you can set the `padding` property to add space around the text inside the button. For example:

HTML Code
```
<style>
a {
  border: 1px solid
blue;
  display: block;
  text-align: center;
  padding: 16px;
  width: 120px;
}
</style>

<a href="/">Home</a>
```

Webpage Output


```
<style>
a {
```

For your next task:

1. Set the anchor tag `padding` to **10** pixels.
2. Set anchor tag `box-shadow` property value to:
   `0 1px 1px lightGreen, inset 0 1px 0 honeyDew`

This code adds 2 shadows to the button; the 1st is a **lightGreen** shadow **outside** the bottom of the button; the 2nd shadow uses the `inset` property to put a shadow **inside** the top of the button.

7.5. horizontally center buttons

To horizontally center your buttons (or any `block` element) set the `width` then set the `margin` properties.

Remember that the `margin` shorthand property uses the following order:

```
margin: top right bottom left;
```

Complete the following using either the shorthand or separate `margin` properties:

1. Remove the break tags from index.html.
2. In styles.css, set `margin-top` and `margin-bottom` to: `10px`
3. Set `margin-left` and `margin-right` to **auto**.

8. Use [[colorTable2|RGB color]] values

8.1. RGB colors

So far you have set CSS colors using the 140 **color names**.

In this lesson you will use the **RGB color values**, which use a mix of **red**, **green** and **blue** to make a specific color. The amount of red, green and blue are numbers from **0-255**, which gives a total of 16,777,216 possible colors!

The following code makes text **red**:
`color: rgb(255, 0, 0);`

The following code makes the background **green**:
`background-color: rgb(0, 255, 0);`

The value for [illegible] is: `rgb(0, 0, 0)`

The value for **white** is: `rgb(255, 255, 255)`

1. Add a level 1 **heading** that says: "RGB Color Test"
2. Add `style` tags and set the **background color** to the RGB value for **blue** in the body tag CSS rule.
3. In the body rule, set the text **color** to the RGB value for **white**.

8.2. RGB color tool

Imagine you are in a [illegible] with **red**, **green** and **blue** flashlights. If you shine all 3 lights at the same spot on the wall, you will see a white light.

Try our **RGB color mixer tool**. Slide the 3 sliders all the way to the right to turn on the 3 lights.

Make the following changes to the code:

1. Set the **background color** to the RGB value for **darkGreen**, using the numbers 0 and 100.
2. Set the text **color** to the RGB value for **yellow**, using the numbers 0 and 255.

8.3. RGB color practice

Now use the **color name chart** and the **color mixer** to complete the following task:

1. If there is not a level 1 heading, add one.
2. Set the **background color** in the body rule to the RGB value for **lightSalmon**, which uses the numbers 255, 160, 122; use the **color mixer** to get the correct order for the numbers.
3. Set the text **color** to the RGB value for **teal**, which uses the numbers 0 and 128.

8.4. semi-transparent colors

HTML5 has a way to set **semi-transparent** colors using RGBA (red, green, blue, alpha).

The alpha value is a decimal number between 0 and 1 that sets how transparent the color is. A value of 1 is a solid color; 0 is completely transparent; 0.5 is semi-transparent.

For example, a semi-transparent **yellow** is set using the code: `rgba(255, 255, 0, 0.5)`

1. Set the `background-color` for the `h1` tag to the RGBA value for **red** with an alpha value of **1**.
2. Change the alpha value to **0**, **0.5**, **0.75** and finally set it to **.25**.
3. Add a 1 pixel solid **red** **border** to the heading (use an RGBA value with alpha of 1).
4. Set the space between the border and text to 8 pixels.
5. Give the heading rounded corners with a radius of 4 pixels.

8.5. choose your colors

For your final task:

1. Change at least 3 of the colors in styles.css to RGB color values that match the theme of your travel plan.

Use our **color name chart**, **RGB color chart** and **color mixer** to choose colors you like.

9. Use new CSS3 properties with browser prefixes

9.1. `user-select`

The **W3C** organization decides the rules for CSS properties. **Web browsers** use browser prefixes to add CSS properties that are not part of the official **W3C** rules.

For example, the `user-select` property is not in the **W3C** rules, but was added by Firefox, Chrome, Safari and IE10. To use this property add these 3 lines of code:

```
-webkit-user-select: none;
-moz-user-select: none;
-ms-user-select: none;
```

The 1st part of these property names is called a **browser prefix**. Properties that start with:

- `-webkit-` only work with Chrome and Safari (the other browsers ignore it);
- `-moz-` is for Firefox;
- `-ms-` is for IE;
- `-o-` is for Opera.

For your next task:

1. Add the 3 lines of the `user-select` property to the body tag rule in styles.css.
2. If your browser supports `user-select` check that the text on index.html can't be selected.

9.2. `user-select`

If the **W3C** makes it an official CSS3 property, browsers will add support for `user-select` without a **browser prefix**.

1. To prepare for the future, add `user-select: none;` after the browser prefixed versions.
2. Opera may support `user-select` in a future version, so add a line that adds support for Opera: `-o-`

9.3. `article` element

For this task you will add some information to day1.html and use the `article` tag, which is new in HTML5.

`article` tags should be put around things like magazine or news articles and forum or blog posts.

1. Put **article** start and end tags after the level 1 heading on day1.html.
2. Inside the article tags, add 2 `h2` headings with the names of the main activities on day 1.
3. Under each heading write at least 2 sentences about the activity; put a **paragraph** tag at the start of each sentence.
4. In styles.css add an article tag rule to align the text to the left.
5. Add a rule to change the size of the `h2` headings to 18 pixels, and set the top margin to 0.

9.4. `column-count`

Another property new in CSS3 is `column-count`, which puts content in 2 or more columns.

Since it is a new property **browser prefixes** are required for Chrome, Safari and Firefox. It works without prefixes in IE10+ and Opera 11.1+.

1. Set the `column-count` property for the article tag to 2; use 3 lines of code to make sure it works in all 5 major **web browsers**.
2. Add information about at least 1 activity in day2.html and day3.html using the same format as in day1.html.

9.5. `linear-gradient`

Browser prefixes are also used with CSS3 property **values**.

For example, the **background image** property can be set to a **linear gradient** value.

This **demo** code works for IE10+, Opera 11.1+, Firefox 3.6+, Chrome 10+ and Safari 4+:

```
background-image: -webkit-linear-gradient(left, red, yellow);
background-image: -moz-linear-gradient(left, red, yellow);
background-image: -o-linear-gradient(left, red, yellow);
background-image: linear-gradient(to right, red, yellow);
```

1. Remove the `background-image` property in the body tag rule in styles.css.
2. Copy the 4 **background image** properties above and put them in the body rule.
3. Change **red** to **lightGreen**; confirm which line is used by your **web browser**.

10. Use [[gradient]] color fills

10.1. linear gradient

For this task you will add a **linear gradient** to the buttons on index.html.

Linear gradients are new in CSS3. However, IE used its own `filter` property to support gradients since 2000 (IE5.5).

Safari and Chrome added their own `-webkit-gradient` in 2008.

In 2010 **W3C** made the rules for `linear-gradient`, then a year later changed the rules. This standard form of gradient works in the following browser versions: Firefox 16+, Chrome 26+, IE10+, Opera 12.1+, Safari 6.1+.

For details about **all** the gradient versions, read this article.

In this course use these 4 versions, which work in IE10+, Opera 11.1+, Firefox 3.6+, Chrome 10+ and Safari 4+:

```
background-image: -webkit-linear-gradient(left, green,
yellow);
background-image: -moz-linear-gradient(left, green, yellow);
background-image: -o-linear-gradient(left, green, yellow);
background-image: linear-gradient(to right, green, yellow);
```

Notice that the standard form on the final line, is slightly different from the other forms.

1. Copy the 4 **background image** properties and put them in the **anchor** rule in styles.css.
2. Change **left** and **right** to **top** and **bottom** so that **yellow** is on top.

10.2. multi-colored gradients

Gradients can have a list of 2 or more colors separated by: **,**

The demo uses 5 colors: **blue**, **green**, **orange**, **red** and **yellow**.

If a **linear gradient** doesn't set a direction, it goes from top to bottom by default.

The following lines of code both create a gradient with green on top and yellow on bottom:

```
background-image:
  linear-gradient(green, yellow);
background-image:
  linear-gradient(to bottom, green, yellow);
```

In styles.css make the following changes:

1. Change the **linear gradient** in the body tag to use the rainbow colors in the demo; set the colors from top to bottom, without stating a direction.
2. Remove the directions from the gradient for the buttons, and change the colors to be **lawnGreen** on top and **greenYellow** on bottom.

10.3. `background-attachment`

In the previous task, the gradient pattern restarts below the audio player.

The `background-attachment` property is used to make the background go from the top of the page to the bottom of the page.

1. Set the `background-attachment` for the body tag to `fixed`.
2. Change the colors for the background to go from **honeydew** on top to **lightGreen** on bottom.

Note: **honeydew** is a very pale green, and may appear as **white** on older screens.

10.4. diagonal gradients

As shown in the demo, the direction of gradients can also be diagonal.

```
background-image:
  -webkit-linear-gradient(left top, red, blue);
background-image:
  -moz-linear-gradient(left top, red, blue);
background-image:
  -o-linear-gradient(left top, red, blue);
background-image:
  linear-gradient(to right bottom, red, blue);
```

Note that `bottom right` and `right bottom` both work the same, however, for this course **please put left and right first**.

1. Add **direction** values to change the gradient for the body tag to be **lightGreen** in the bottom left and **honeydew** in the top right; don't change the order of the colors in the code.
2. Copy the gradient code for the buttons and put it in the **a**`:hover` rule; add a **direction** value so that when you hover over the buttons, **greenYellow** is on top and **lawnGreen** on bottom. Note: there are 2 possible answers.

10.5. semi-transparent gradients

Instead of **color names**, gradients can use **RGB values** or even **RGBA** to make semi-transparent gradients.

A texture image can be put behind the gradient as shown in the demo, using the following code:

```
background-image: linear-gradient(rgba(255, 0, 0, .8),
  rgba(0, 0, 255, .2)), url(/images/textures/light1.png);
```

1. Change the body tag gradient colors to **RGBA** by comparing the **color names** with the **RGB values**; set the **alpha** values to .75 for **lightGreen** and .25 for **honeydew**.
2. Add one of the texture images, as shown in the demo code.

11. Use validation to check the HTML correctness

11.1. HTML validation

An HTML validator is a tool that displays error messages if there are bugs in your code. You can click to run the HTML5 validator at https://validator.w3.org/nu.

Some mistakes cause 2, 3 or more messages. The messages show where the validator found a problem, which can be several lines **after** the actual mistake.

The validator does not detect errors in **attribute** values.

In this lesson you will use the validator to find and fix errors.

1. Click to open the HTML5 validator. Select "Check by text input". Copy and paste your code into the box, then click the **Check** button below the box.
2. Scroll down and read the messages, fix the error on line 2, then run the validator on your code again.
3. Fix the attribute value error on line 3 then click to complete the task.

11.2. HTML validation

If you want to validate HTML5 code, visit http://html5.validator.nu. With this tool you can choose a web page address, a file, or cut and paste some code to validate.

W3C has a validator at https://validator.w3.org, which validates **any** version of HTML. For HTML5 it uses http://html5.validator.nu in the background.

1. Click to run the HTML5 validator and read the error messages.
2. Fix the error on line 11 then re-validate.
3. Fix the error on line 12 then re-validate.
4. When there are no more errors, click to complete the task.

11.3. HTML validation

If a web page contains errors, your **browser** will read and display it as best it can. However, different browsers may display errors differently.

Writing valid code increases the chance that your web pages look the same on all browsers. Doing so also makes it easier for team members to understand your code.

There are several other reasons why some web professionals validate their code.

1. Click and read the error messages.
2. Fix the first error then re-validate.
3. Fix the second error then re-validate.
4. Click when there are no more errors.

11.4. HTML validation

In level 1, lesson 9 you learned that the `html`, `head` and `body` tags are optional, but are automatically added by your browser if left out. This lesson includes the optional tags so that the validation errors are clearer.

Remember the following:

`<head>` tags surround content that is not shown on the web page, e.g. `title`, `link` and `meta` tags.

`<body>` tags surround content that is visible on the web page.

1. Click and read the error messages.
2. Fix the first error then re-validate.
3. Fix the second error then re-validate.
4. Click when there are no more errors.

11.5. HTML validation

When you don't use the `html`, `head` and `body` tags, you should make sure you put the `doctype` on line 1 followed by `meta`, `link` and `title` tags, followed by the visible content of the web page.

In this task there are 3 errors with the images.

1. Click and read the error messages.
2. Fix the first error then re-validate.
3. Fix the second error then re-validate.
4. Fix the third error then re-validate.
5. Click when there are no more errors.

12. Use the {{figure}} tag

12.1. Google maps

In this lesson you will add maps, images and video to day1.html, day2.html and day3.html.

As with YouTube videos, you can use the `iframe` tag to put a Google Map on your page.

1. Read the instructions on how to include a Google map on your page. Add a map that shows the location of the day 1 activities.
2. Set the `width` attribute of the map's `iframe` tag to **320** pixels.
3. Set the `height` attribute of the map's `iframe` tag to any value you like.
4. Remove the `frameborder` attribute from the `iframe` tag because it is not valid in HTML5.

If you have a Google account, you can create a single map with multiple locations on it.

12.2. image and video

Now add an image and a video about the day's activities to day1.html.

1. **Click here for instructions** on how to add a YouTube video on your page; be sure to resize your video to a width of 320 and height of 240.
2. **Click here for instructions** on how to get an image from Google image search on your page. Resize your image to have maximum width of 320;

To use an image from another website on your own website you need permission from the image owner.

Visit Creative Commons Search to find images that you can use for free without permission.

12.3. `figure` and `figcaption`

Another new element in HTML5 is `figure`. This element is put around content like images, maps and videos that are important to the meaning of the page and can be positioned anywhere without changing the meaning.

An optional caption for the figure may be put inside a `figcaption` element, which is placed inside `figure` tags. For example:

```
<figure>
  <img src="http://....">
  <figcaption>
    Waitomo's most concentrated Action Adventure
  </figcaption>
</figure>
```

1. Surround the image, map and video on day1.html with `figure` tags.
2. Add captions to the image, map and video.

12.4. `figure` style

The default figure style has a large left and right margin.

Add a CSS rule for the `figure` tag with the following properties:

1. Set the left and right margins to 5 pixels and the top and bottom margins to 10 pixels using the `margin` shorthand property.
2. Set the left and right padding to 0 pixels and the top and bottom padding to 5 pixels using the `padding` shorthand property.
3. Add a 1 pixel dotted **gray** border to the top and bottom of the figure using `border-bottom` and `border-top`.
4. Set the text **color** to **gray**.
5. Horizontally **center the text** and image in the `figure`.
6. In the `iframe` CSS rule, set the `border` property to none.

12.5. day 2 and day 3

For this task finish day2.html and day3.html.

1. Add a Google map to day2.html and day3.html (see how to add Google maps).
2. Add a YouTube video or image to day2.html and day3.html; (see **YouTube video instructions**, and **Google image instructions**).
3. Use figure and figure caption tags with the maps, images and videos.

13. Add {{meta}} tags

13.1. `meta` viewport

If you view your web page on a real phone the phone will zoom out and your page will be small.

In the level 1 course you used the `meta` viewport tag to make your pages look better on mobile phones.

In this lesson you will use a few more `meta` tags.

1. On each HTML page, add the `meta` viewport tag on a new line after the `doctype`.
2. Try viewing your page on a real mobile phone.

Use the reference if you need help.

13.2. other `meta` tags

`meta` tags contain information about a page, that is not visible to human viewers. These tags are put near the top of the code and do not have an end tag, i.e. you don't use **`</meta>`**.

There is a long list of registered meta tags. Most of the common ones have `name` and `content` attributes. For example, the **author** `meta` tag:

```
<meta name="author" content="Author's Name">
```

1. Add the author `meta` tag to each html page. Use your name as the `content` value.
2. On index.html, add the `geo.country` `meta` tag and set the `content` to the 2 digit ISO code for the country of your travel plan (see list of country codes).
3. On index.html, add a `geo.placename` `meta` tag that gives the location of your travel plan; the region can be a state, province or area.

13.3. description and keywords

Some `meta` tags give information about your web pages to search engines like Google.

The **description** `meta` tag gives a short sentence describing your page, which is often displayed in search engine results.

The **keywords** `meta` tag gives a list of keywords related to your site. This is not used by Google, but is used by Yahoo and some small search engines.

```
<meta name="keywords"
content="Waitomo,caves,abseiling,adventure">
```

1. Add the **description** `meta` tag to each page with at least 4 words describing its contents.
2. Add the **keywords** `meta` tag to each page with at least 2 keywords separated by commas.

13.4. robot `meta` tags

A **robot** is a computer program that is used by search engines like Google to create a list of all the pages on the world wide web. When a robot finds your home page, it looks at the contents of your page, then uses the links (i.e. anchor tags) on the page to find the rest of your site.

The **robots** `meta` tag tells robots how to search your site and which pages to add to its list.

`noindex` tells the robot not to add the page to the list. `<meta name="robots" content="noindex">`

`nofollow` tells the robot not to search pages that the current page links to. `<meta name="robots" content="nofollow">`

The other possible values are `Instructs the search engine not to store an archived copy of the page.|noarchive` and `Asks the search engine not to include a snippet from the page in its search results.|nosnippet`.

The **robot** `content` value can include one or more of these values.

1. On test2.html add the robots tag with content of: `noindex,nofollow`
2. On index.html add the robots tag with content of: `nofollow`

13.5. meta charset

There are many ways the computer can store the characters in your code. The character encoding `meta` tag tells the computer how to store the characters. The Google style guide recommends using a character encoding called **utf-8**.

Set the character encoding by putting the following code on the line below the doctype tag: `<meta charset="utf-8">`

Using this tag makes sure that text like "コードアベンジャーズ" doesn't end up looking like "?????".

1. Add the charset `meta` tag to each html page, on the line after the `doctype`.
2. **(Optional)** Read more about character encodings and declaring character encodings in HTML.

14. Create a [[table]]

14.1. `table` element

In this lesson you'll add a basic travel timetable to plan.html. In the next lesson you will make it look nice.

Here is the code for a simple **table**:

```
<table>
  <tr><th>Header 1<th>Header 2
  <tr><td>Data 1<td>Data 2
  <tr><td>Data 3<td>Data 4
</table>
```

1. Copy the code for the basic table, and paste it into plan.html.
2. In styles.css add a rule that gives the `table` a 1 pixel solid **red border**.
3. Add another rule that gives the `td` and `th` a 1 pixel solid **blue border**.

14.2. table rows and header

Each **t**able **r**ow starts with `<tr>`; putting `</tr>` at the end of the row is optional.

Each row contains 1 or more **cells**.

There are two types of cells: **d**ata cells start with `<td>` and optionally end with `</td>`; **h**eader cells start with `<th>` and optionally end with `</th>`.

Make your table look like the example plan.html page.

1. The table needs 15 rows.
2. Row 1 needs 4 header cells with the text **Time**, **Day 1**, **Day 2** and **Day 3**.
3. Rows 2-15 need 1 data cell each with 24 hour times from **0800–2100**.

14.3. table data

Add your plan for day 1 as shown in the example.

1. Add times for lunch and dinner.
2. Add at least 2 activities that take at least 2 hours each.

14.4. `rowspan` attribute

The activities that take more than 1 hour should fill multiple cells. This is done using the `rowspan` attribute.

For example the following code makes the **Lost World Tour** cover 4 hours: `<tr><td>1500<td rowspan="4">Lost World Tour`

1. Use the `rowspan` attribute to make your activites fill at least 2 hours.
2. Fill all the empty slots on day 1 with **travel** and **free time** (leave day 2 and 3 empty).

14.5. `colspan` attribute

The `colspan` attribute makes a cell cover multiple columns.

1. Use the `colspan` attribute to make dinner at **1900** for all 3 days (as in the example).
2. Fill in all the cells for the rest of days 2 and 3.

15. Style a [[table]]

15.1. `border-collapse` property

In this lesson you will make your table look better by adding CSS to styles.css.

For your first task you will remove the space between table cells using the `border-collapse` css property.

1. Set the `border-collapse` property for the `table` rule to a value of `collapse`.
2. Remove the border for the `table` rule.
3. Change the color of the table cell borders to an **RGB color** with the values 191, 187 and 128.
4. Add 3 pixels of space inside each cell using the `padding` property.

15.2. `class` attribute and selector

Now you will change the background color of the activity cells. To change the color of a specific activity cell you will edit HTML code and CSS code.

1. On plan.html, set the `class` attribute for one of the `td` table cells to the value: "activity"
2. Add the following code to styles.css:
```
.activity {
  background-color: rgb(242, 255, 145);
}
```

In CSS code, any **selector** that begins with a period (`.`) is called a **class selector**. The CSS code above sets the background color for all HTML elements with a class attribute set to "activity".

15.3. activity class

Now make all the activities the same color.

1. In styles.css set the `color` for the `table` rule to: `191, 187, 128`
2. Set the `background-color` for the table to: `30, 38, 12`
3. Add the "activity" class to all of the activity cells in the table.
4. Set the `color` for the activity class CSS rule to the RGB value for ⬛.

15.4. other classes

Now let's change the color of the travel, meal and free time cells in the table.

1. Add the class "eat" to cells for lunch and dinner.
2. Add the class "free" to cells that are free time.
3. Add the class "other" to cells for events such as travel, which don't have a class yet.
4. Use a single CSS rule to set the background color for the **eat**, **free** and **other** classes to the RGB value for **white**.

15.5. table border

For the final task do the following:

1. Add a solid 3 pixel **border** around the outside of the table, and make it the same color as the cell borders.
2. Change the colors of the table to match your theme using the **color mixer** and **color table**.

16. Learn about class selectors

16.1. table review

In this lesson you will learn more about using classes.

For your 1st task:

1. Create a table with 3 rows, 3 columns, and cells that are numbered 1-9 (i.e. the top row contains the numbers 1-3).
2. Add CSS `style` tags on line 2.
3. Inside the `style` tags add a table CSS rule and set the width and height to 300 pixels.
4. Horizontally center the text by adding the correct property to the table rule.
5. Add a 2 pixel solid border colored with the RGB value for **blue** (use the numbers 0 and 255).

16.2. even class

For this task you will make the table look like a checker board.

1. Add the class "even" to every 2nd table cell.
2. Add a CSS rule for the "even" class that sets the background color to **black** and text color to **white** using RGB values.
3. Set the table background color to **red** using an RGB value with the numbers 200 and 0.
4. Use the `border-collapse` property to make the corners of the ⬛ squares touch each other.

16.3. class names

Class names almost always use only lowercase letters, numbers, _ and `-`. Also the name can't start with a number or double dash (`--`).

For example, these are valid class names: `name` `-name` `n4m3` `_a_e`
These are invalid class names: `9ame` `--name` `*name` `.name`

The complete rules for class names are a little more complex.

When an element belongs to more than 1 class, the classes are separated by a space.
E.g. `<td class="even multiple-of-4">8`

1. Add the class `multiple-of-3` to the cells 3, 6 and 9.
2. Add a CSS rule to increase the font-size of those 3 cells to 48 pixels.
3. Increase the font-size of the other 6 cells to 24 pixels by adding a property to the table rule.

16.4. css rule priority

Cell number 6 has 2 classes, `even` and `multiple-of-3`. If you have CSS rules for both classes that set the `font-size`, the 2nd rule takes priority.

Test this out by doing the following:

1. In the `even` rule, set font size to 10 pixels and move the rule to the line after the `multiple-of-3` rule.
2. Then see what happens to cell 6 when you move the `multiple-of-3` rule to the line after the `even` rule.

16.5. CSS rules with multiple classes

CSS rules for elements with **2** or more classes are created by listing the classes, without a space between them.

For example, the following rule changes the color of elements with the `odd` and `multiple-of-4` classes:

```
.odd.multiple-of-4 {
  color: rgb(255, 255, 0);
}
```

1. Add a CSS rule that uses 2 classes to make the text for cell 6 bold and 64 pixels in size.
2. When cell contents have different widths, the web browser automatically changes the table column widths. Try to set the width and height of all cells to 100 pixels.

17. Learn about descendant selectors

17.1. selectors review

In this lesson you will learn to use more complex CSS selectors.

1. Create an **ordered list** with these vehicles (in alphabetical order): airplane, bicycle, canoe, hovercraft, jetboat, truck.
2. Use the **class attribute** to set the type of each vehicle to **air**, **land** or **water** (1 of the vehicles travels on land and water).
3. Set the **background color** for the water vehicles to the RGB value for **lightCyan** which has a red value of 224 and green and blue of 255.

17.2. list selectors

There are several ways to set the **background color** of an ordered list.

1. Create an `ol` CSS rule and set **background color** to: `rgb(255, 192, 203)`
2. Create a `li` CSS rule and set **background color** to the RGB value for **lightYellow** (use the numbers 224 and 255).

Properties set with a `li` rule always take priority over the `ol` rule.

Properties set with a class selector (e.g. `.water`) take priority over element selectors (e.g. `li` and `ol`).

17.3. selectors with multiple classes

Now add a 2nd list.

1. Create an **unordered** list of these 6 animals (in alphabetic order): dog, eagle, frog, seal, shark, whale.
2. Use the **class attribute** to set the type of each animal to **air**, **land** or **water** (2 of the animals can live in water and land).
3. Create a CSS rule to set the background color for the unordered list to the RGB value for **lightGray** using the number **211**.

Notice that the `li` and `.water` CSS rules work for both lists.

17.4. selectors with multiple elements.

In this task you will use different colors for the items on each list. This is done using **descendant selectors**, which are `style` elements that are inside other elements.

For example, the following code sets the color of items inside an ordered list:

```
ol li {
  background-color: red;
}
```

1. Add a CSS rule that sets the **background color** of items inside an unordered list to: `rgb(210, 180, 140)`
2. Add a CSS rule that underlines items that have both land and water classes.

Note: when using descendant selectors, the selectors are separated by a space.

17.5. descendant selectors with classes

Descendant selectors can also use classes.

For example, this rule will `style` elements with the class **water** that are inside an element with the class **animal**: `.animal .water {`
```
  background-color: rgb(210, 180, 140);
}
```

Look carefully to notice that this rule is slightly different to CSS rules with multiple classes.

1. Change the animal list to an ordered list.
2. Add the classes **animal** and **vehicle** to the `ol` tags.
3. Remove the `ul li` CSS rule.
4. Change the `.water` CSS rule so that it only sets the background color for **water vehicles**.
5. Add a CSS rule that gives **land animals** a background color of the RGB value for **lightGreen**, which uses 144 and 238.

18. Create a [[header]] and [[footer]]

18.1. `header` and `section`

In this lesson you'll add a fixed `header` and `footer` to index.html.

1. Put `header` tags around the Code Avengers logo near the top of the page.
2. Put a `section` start tag just before the page heading.
3. Put a `section` end tag at the end of your page.
4. In between `</header>` and `<section>` add a `footer` element with the text: "Created by ???". Replace `???` with your name.

You could put the footer at the end of the code, but for this lesson just follow instruction 4.

18.2. `fixed` position

The next 2 tasks add CSS rules to move the `footer` to the bottom of the screen.

1. Add a single CSS rule that sets the `position` property of the `footer`, `header` and `section` elements to `fixed` (the header and footer will overlap in the top left corner).
2. Add another CSS rule that sets the `background-color` of the header and footer to a **light green** using an RGB color value that uses the numbers 255 and 120.
3. Set the `padding` for the `footer`, `header` and `section` elements to 5 pixels.

To create a single CSS rule that sets the style for 2 tag types, list the tags separated by commas.
```
h1,
h2 {
  color: blue;
}
```

E.g., this rule sets the **color** of level 1 and 2 headings to **blue**.

18.3. `fixed` position `footer`

Elements with `position` set to `fixed` can be moved to a specific location on the screen by setting the `top`, `right`, `bottom` and `left` properties.

1. Add a rule that sets the `top` property of the `footer` tag to 40 pixels.
2. Experiment with setting the following values for the `footer` tag: left to 10 pixels, right to 0 pixels, bottom to 20 pixels.
3. Modify the `footer` rule so that its left, bottom and right sides touch the edge of the screen.

18.4. `fixed` position header and section

Now put the **header** and **section** in the correct position.

1. Add a CSS rule for the `header` tag that makes its top, left and right sides touch the edge of the screen.
2. Add a CSS rule for the `section` tag that sets **top** to 50 pixels, **bottom** to 30 pixels and **left** and **right** to 0.

18.5. combining css rules

Did you notice that the left and right properties are set to 0 for footer, header and section?

1. Change the code so that `right: 0;` and `left: 0;` are only in your CSS code once.
2. Add a 2 pixel solid border to the header and footer.
3. Set the color of the header and footer background and border to **any color you like**.

19. Create a navigation bar

19.1. navigation element

In this lesson you'll use an HTML5 element called `nav` (short for navigation). This tag is put around a group of links (or **anchor**s) to other web pages in your site.

1. Put `nav` tags around the 4 links on index.html.
2. Put a copy of the `nav` tags and the 4 links after the logo inside the `header` element.

19.2. header style

Remember that the order of CSS rules is important. With the code below, if an `<a>` element is inside both a `<header>` and `<nav>` it is colored **red**, because the 2nd rule takes priority. Click here to read more about the priority rules for CSS.

```
nav a {
  color: blue;
}

header a {
  color: red;
}
```

Now add a CSS rule that makes the links in the header smaller.

1. Create a CSS rule with `header a` as the selector so that the rule only styles links inside the header. In order to give it priority over rules we will add later in this lesson, position the rule at the bottom of styles.css.
2. Set the **width** to **50** pixels.
3. Set the **display** to `inline-block`.
4. Set the **font size** to **14** pixels.
5. Set the **margin** to **0** pixels.

19.3. navigation buttons

Now put the navigation buttons on the same line as the logo by following these steps:

1. In the code for index.html remove all spaces between the `</a>` and `<a>` tags inside the **header**.
2. Create a CSS rule that only styles the **nav** tags inside the **header**.
3. Set its `display` property to `inline-block`.
4. Set its `vertical-align` property to `top` (you'll learn about the this property later).

19.4. header and footer opacity

Opacity is a new CSS3 property that makes an element semi-transparent; opaque means the opposite of **transparent**.

The value for the `opacity` property is a decimal number between 0 (completely transparent) and 1 (completely opaque).

1. In the header and footer rule, set the `opacity` property with a value of **0.7**.
2. Add a rule that sets the header `opacity` to **1** when the user hovers over it.

19.5. header style

Most websites link the logo at the top of the page to the home page.

1. Link the logo in the top left corner to index.html.
2. Change the logo from a button back to normal by modifying styles.css. Change the selector for the a and `a:hover` rules to `nav a` and `nav a:hover`. This makes only `<a>`s inside the `<nav>` look like buttons.
3. Surround the content on plan.html, day1.html, day2.html, day3.html with `section` tags.
4. Copy the `header` and `footer` from index.html and paste at the bottom of plan.html, day1.html, day2.html, day3.html.
5. In styles.css set the `overflow` property for the `section` rule to `auto`; this makes scroll bars appear when all the content can't fit on the screen.

20. Style your pages for printing

20.1. print media queries

This final lesson will use CSS **media queries** to make your pages look nice when it is printed.

If you put `@media print { }` around CSS rules, those rules are only used when printing. The following code makes headings on the page red when printed:

```
@media print {
  h1 {
    color: red;
  }
}
```

1. Copy the example code above and paste it at the bottom of styles.css.
2. Go to day1.html and click the link on the phone to open the page in a new tab.
3. Select print preview from your browser menu to see what your page looks like.

Don't be surprised if the print preview looks bad!

20.2. screen media queries

When you print web pages the background is made **white**. Firefox has an option to print the background colors of a website in it's print set-up options.

Each web browser prints differently. For example, Chrome does weird things when the `position` property of section, header and footer are set to `fixed`.

1. To display the page content properly when printing, add a `section` CSS rule in the print media query that sets the `position` property to `static`.
2. Hide the header and footer during printing by adding a CSS rule with `display` set to none.
3. Open your page in a new tab and check the print preview.
4. For testing purposes, change `@media print` to `@media screen` before clicking .

20.3. print versus screen

`@media print` is a media query used to write rules that only apply to printing.

`@media screen` is a media query used to write rules that only apply when the page is displayed on screen.

For testing purposes, continue to use `@media screen` in this task and the next one (task 4). You'll switch to `@media print` in the final task of this lesson.

1. Remove the `h1` rule from the media query code.
2. Add a rule to the media query code that sets the article column count to 1.
3. Add a rule to the media query code that hides the audio control on index.html.

20.4. hide videos from print

YouTube videos display as an empty space during printing, so we will hide them.

1. Add a **class attribute** with the value **video** to `figure` tags that contain YouTube videos in day1.html, day2.html and day3.html.
2. Add a CSS rule to the media query that hides the figures with YouTube videos.

Review lesson 16 on `class` selectors for help.

20.5. change to print media query

Congratulations!

You have reached the end of the level 2 Code Avengers HTML5/CSS3 course!

For your final task:

1. Change `@media screen` back to `@media print`
2. Open your page in a new tab and check the print preview.
3. **(Optional)** Remove the comments on index.html to show the YouTube and Wikipedia pages and add `figure` and `figcaption` tags.
4. **(Optional)** Link the cells in the table header row on plan.html to day1.html, day2.html and day3.html.