CAMBRIDGE
CODING ACADEMY

BUILD YOUR WEB APP_

Contents

There are five modules in the Web App Workshop, each introducing one or two core programming concepts. The learning activities in each allow you to build the web page in manageable steps while seeing the concepts in action as you code.

1 Set-up

Note that this is only applicable if you did not set up for the previous day's "Build a Game" course. If you already have running code from the previous day, skip these steps and start from Module 1.

- On Github, fork the project ***https://github.com/cambridgecoding/webapp-flappy*** to your own account.
- Clone your fork of the project on your computer.
- Open the folder in Atom.

Structuring with HTML 2

In this module, you will learn to create a simple web page and style it. You already have been given a page for your game, but it is better to create your own to fit your own game!

Your first web page

Learning Activity 1: Any text is a web page!

1. Create a text file in your project, inside the `pages/` directory, call `simple.txt`.
2. Inside this file, put a greeting for your game's page. For example we'll use the following (but you can customise it to your theme):

```
Flappy Birdy!

Welcome to Flappy Bird.

Here will be the game

How to play:
- Press SPACE to flap your wings.
- Avoid the pipes.
- If you crash, just try again!
```

3. Save the file and open it in a browser. To do that, use the "Packages" menu and choose "Live Server" and "Start on port 3000". Then, in the URL bar, type `localhost:3000/pages/simple.txt`.

Congratulations, you've created your first web page!

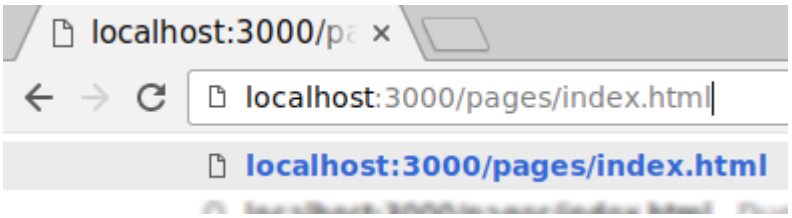
All web pages are at their core just text files telling the browser what to show. You can see the text file behind your favorite website right now by visiting it, right-clicking anywhere on the page, and clicking *View page source* (different browsers might have different names for it).

Learning Activity 2: Web pages need more structure

1. What happens if you put the same content in the file `pages/index.html`? Try and open it in your web browser. You can open change files in your browser by editing the URL directly.

FIGURE 2-1

Opening a file in the browser



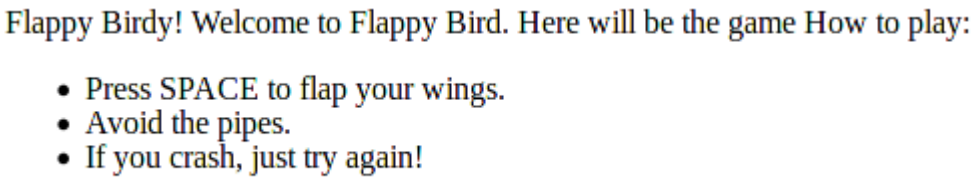
However, when you view the file, it seems to have lost all structure and appears in a continuous line. This is because the browser ignores some *whitespace* like new lines or tabs. Instead, it expects special pieces of code called *HTML tags* to provide structure.

2. Let's try telling the browser that we want the instructions shown as a bullet point list. Surround the list of instructions with `<ul>` and `</ul>`. Surround each instruction with `<li>` and `</li>`. Your code should now look like this:

```
Flappy Birdy!

Welcome to Flappy Bird.

Here will be the game

How to play:
<ul>
  <li>Press SPACE to flap your wings.</li>
  <li>Avoid the pipes.</li>
  <li>If you crash, just try again!</li>
</ul>
```

FIGURE 2-2

How lists are rendered by the browser



If the page does not change automatically, refresh it.

HTML TAGS

HTML files are composed of raw content (what appears in the browser) marked up with tags (which tell the browser how to display it).

The list of instructions is surrounded by the `ul` tag. It starts with `<ul>` and ends with `</ul>`. It is used for Unordered Lists (i.e., bullet point lists).

Most tags have the same form; e.g., `<li>` and `</li>` for List Items. The first part is called the *opening tag*, the second one is called the *closing tag*.

You will learn many more tags in this course.

Learning Activity 3: More HTML tags

Let's learn about some more HTML tags. Can you put the following tags in the right places in the code in order to obtain the result shown next?

```
<h1>This is a heading</h1>                                    ❶
<p>This is a paragraph. It can be as long or as short as you want.</p>  ❷
```

❶ Headings (in `h1` tags) are titles at the beginning of a document or a section.

❷ Paragraphs (in `p` tags) hold separate blocks of text.

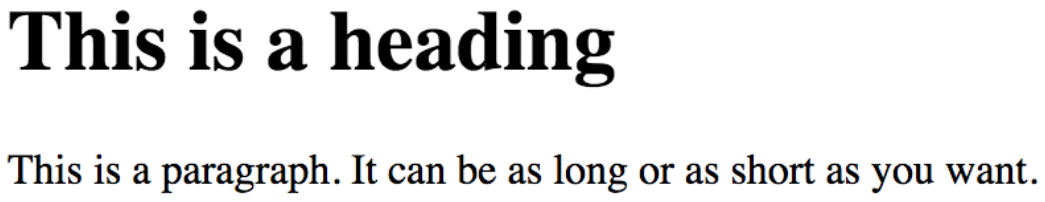


FIGURE 2-3

How the header and a paragraph are rendered by the browser

Learning Activity 4: Metadata

So far, we've been working directly on the content that gets shown to the user in the web page. However, pages often contain additional bits of information about what the document contains. These extra bits of information are called "metadata" (which means "data about the data"). Let's learn about them now.

1. Surround the entire contents of the file with `body` tags (`<body>` at the start and `</body>` at the end).
2. Surround again with `html` tags (`<html>` and `</html>`).
3. Between the `<html>` and `<body>` opening tags, place a pair of `head` tags (i.e., place `<head>` and `</head>`). The metadata goes within the `head` tag.
4. Within the `<head>` section, add a `<title>` tag with the title of the page inside.
5. Your code should look like this:

```
<html>

<head>
    <title>Flappy Birdy</title>
</head>

<body>
  <h1>Flappy Birdy!</h1>

  <p>Welcome to Flappy Bird.</p>

  <p>Here will be the game</p>
```

```
  <p>How to play:</p>
  <ul>
    <li>Press SPACE to flap your wings.</li>
    <li>Avoid the pipes.</li>
    <li>If you crash, just try again!</li>
  </ul>
</body>

</html>
```

Do you see what's changed when you refresh the page? (Hint: look at the browser tab.)

You may now be wondering what are the `head` and `body` tags? In a nutshell, the `head` tag provides general information (metadata) about the document: its title, references to other files (we will cover this in the next module), technical information about character encoding, the language the text is in, etc. The `body` tag represents the content of an HTML document. There is always only one `body` element in an HTML document.

Can you guess what the tag `html` is about? Quite simply it just indicates that the document is in HTML. Note that you can also see `<!DOCTYPE html>` at the top of the file. It's good practice to add that, it just tells the browser to use the latest current version of HTML (HTML 5).

Learning Activity 5: Going beyond text

Of course, web pages are rich media, and can hold more than text. Let's create some links, and include some images.

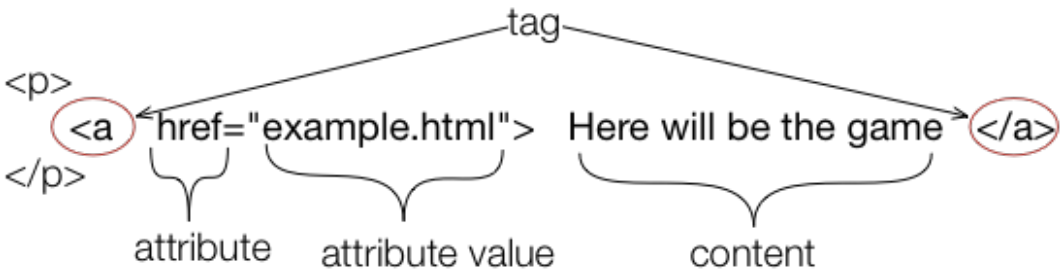
1. Replace the text `Here will be the game` with an `a` tag ("a" stands for "anchor"), and see what happens in the browser.

   ```
   <p>
       <a href="example.html">Here will be the game</a>
   </p>
   ```

   The `a` tag has an *attribute* named `href`. Attributes go within the opening tag. They carry information (here `example.html`) which is used by the browser. In this case, the attribute indicates the link goes to the `example.html` page. Note that the browser displays the content of the tag (`Here will be the game`) on the page.

FIGURE 2-4

The general structure of an HTML tag: tag name, attribute and content.



2. Try clicking on the link. The browser loads and renders a new HTML page which contains yesterday's game. Let's take a look at the code for that new page (note that your version may be slightly different):

   ```
   <!DOCTYPE html>
   <html>
   ```

```
<head>
    <title>Cambridge Coding Academy Flappy Bird</title>
    <link rel="stylesheet" href="../assets/cca.css"> ❶
    <script type="text/javascript" src="../lib/phaser.min.js"></script> ❷
    <script type="text/javascript" src="../js/flappy.js"></script>
</head>

<body>
    <div id="header"> ❸
        <h1 id="flappy">Flappy Bird</h1>
        <div id="cca">
        <a href="http://cambridgecoding.com">
            <img src="http://cambridgecoding.com/assets/images/logos/logo_cca.png"
                id="logo"> ❹
            <p>My very own flappy thing</p>
        </a>
        </div>
    </div>

    <div id="game"> </div>

    <div id="footer">
        <p>Cambridge Coding Academy</p>
    </div>

</body>

</html>
```

You will notice that the code is not much longer than what you have written! It includes a few tags you have not seen yet:

❶ `<link>`

❷ `<script>`

❸ `<div>`

❹ `<img>`

You'll learn them all but first, let's focus on `<img>`. It is used to include pictures in your web page! The location of the image is indicated by the `src` (short for source) attribute.

- In your own document, `index.html`, add a new image with the picture of your game:

```
<!DOCTYPE html>
<html>

  <head>
    <title>Flappy Birdy</title>
  </head>

  <body>
    <h1>Flappy Birdy!</h1>

    <p>Welcome to Flappy Bird.</p>

    <p>
```

```
      <a href="example.html">Here will be the game</a>
    </p>
    <img src="../assets/screenshot.png" />  (1)

    <p>How to play:</p>
    <ul>
      <li>Press SPACE to flap your wings.</li>
      <li>Avoid the pipes.</li>
      <li>If you crash, just try again!</li>
    </ul>
  </body>

</html>
```

(1) An image tag with a local file (`../assets/screenshot.png`). The double dot (`..`) goes up one directory (from `pages` where the HTML document is), the `assets` goes down into the asset directory, the `screenshot.png` indicates the file to select.

- You can use different images: take your own screenshot, save it into the `assets` folder, and change the attribute to match your file name.
- Another peculiar aspect of the `<img>` tag is that it doesn't need to be closed but includes a `/`. You will see other similar special tags later.

Learning Activity 6: Using Developer tools

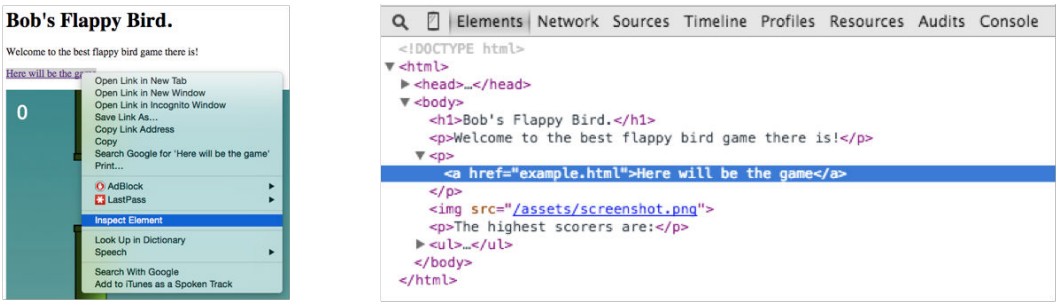


FIGURE 2-5

Inspecting an element on a web page using Developer Tools

1. Right click on any part of the web page, and click *Inspect Element*. (Different browsers might have different names for that function.) What do you see?
2. A new pane opens up with some developer tools. The code you've written laid out in a collapsible form. Hovering over any line highlights the corresponding part of the web page.
3. Play around with this, and try making changes to your page directly in the tool by double-clicking on any part of the code. You'll see your changes reflected live, but they won't be saved back to the code file when you refresh the page.

Wrap up

- HTML lets you structure a document with HTML tags.
- An HTML document can be rendered by a web browser.
- There are two kinds of HTML tags.
    - Tags that needs to be "opened" and "closed" like `<h1></h1>`, and `<p></p>`.

    - Special tags that don't need to be closed such as `<img src=".."/>` but includes a `/` at the end instead.
- HTML tags such as `<a></a>` and `<img>` can have attributes.
- Attributes can provide further details to an HTML tag such as `src` for the location (or source) of an image.

Final code for this module can be found in Appendix A

3 Styling with CSS

Your game website is beginning to take shape, but it still looks very bland. You'll now learn how to style a page with simple CSS files. CSS stands for Cascading Style Sheet and lets you customise the colours, fonts, spacing, etc. of your website. Where HTML is the language of content and structure, CSS is the language of style.

First steps with CSS

Learning Activity 7: Using a stylesheet

1. If it doesn't already exist, create a file called `style.css` inside the `assets` directory. Enter the following contents in this file.

   ```
   h1 {
       color: darkblue;
   }
   ```

   (Remember, most programming languages use American spelling)
   Like some other programming languages, CSS requires you to use semi-colons at the end of some lines.

2. Tell the browser to use these styles when rendering `index.html` by adding a `<link>` tag inside the `<head>` tag in the HTML:

   ```
   ...
   <head>
       <title>Flappy Birdy</title>
       <link rel="stylesheet" href="../assets/style.css"/>  ❶
   </head>
   ...
   ```

   `<link/>` is another example of special tags that doesn't need a closing tag, but instead ends with a `/`. It contains two attributes:

   - `rel` indicates to the browser the type of link it is: a stylesheet.
   - `href` indicates where to find the stylesheet.

3. When you refresh the web page, you should now see the heading appear in dark blue. This is because in the CSS, the `h1` *selects* all level-1 headers: any styles within the braces after the selector apply to all `h1` tags in the HTML (in this case we just have one).

FIGURE 3-1

Text changes its colour when you apply styles to specific elements, e.g. the heading turns dark blue

4. Add any other rules you like for the header, by adding these lines within the braces. Here are some examples:

```
font-size: 16px;
font-style: italic;
font-family: Montserrat, "Helvetica Neue", Helvetica, Arial, sans-serif;
```

Note that `px` means “pixels” and is a unit of measure for the size of things displayed on a screen.

MORE INFORMATION ON STYLING

Look up Mozilla Developer Network for documentation on how each of these styles work: *https://developer.mozilla.org/en-US/docs/Web/CSS/Reference*

Classes and IDs

The selector above (`h1`) selects all the `h1` tags. You can make another rule for all the paragraphs using the `p` selector. (And all the list items using `li` and so on.) But what if you want to apply the rule to some specific elements only.

Learning Activity 8: Using a CSS class

If you need to select more specific parts of your web page, you can use the `class` attribute.

1. Modify the list inside `index.html`:

```
<p>Best scores:</p>
<ul>
    <li class="gold">Me</li>
    <li class="score">Also me</li>
    <li class="score">Me again (I'm the best)</li>
</ul>
```

2. Add the selectors `.gold` and `.score` to your `style.css`:

```
h1 {
    color: darkblue;
}

.score {
    color: gray;
```

```
}

.gold {
    color: gold;
}
```

In CSS, a class selector is prefixed by a dot (`.`).

FIGURE 3-2

Styling a group of elements via a `class`

Learning Activity 9: Using a CSS id

The most specific way to target CSS is to assign an identifier to a specific element. While there can be multiple elements with the same class in an HTML file, there can only be one element with a specific ID.

1. Modify the paragraph where you link the game inside `index.html`:

```
<p id="gameLink">
    <a href="example.html">Here will be the game</a>
</p>
```

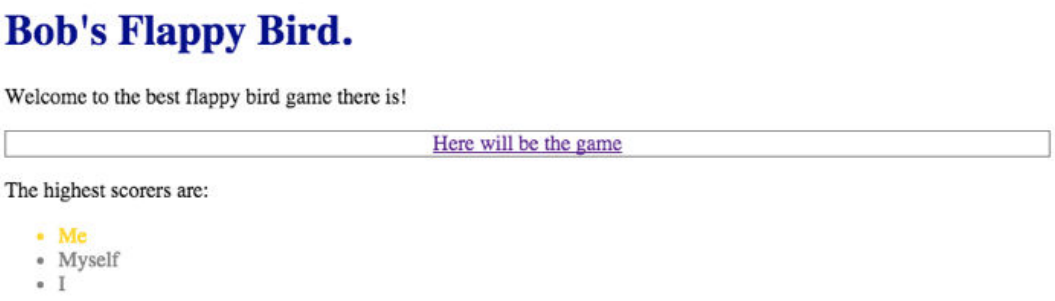
2. Add the identifier selector `#gameLink` to your `style.css`

```
#gameLink {
    text-align: center;
    border: 1px solid gray;
}
```

As you might have noticed, selectors for HTML tags in the stylesheet are the names of the tags themselves. For classes they are prefixed with a dot. For IDs, they are prefixed with a hash sign (`#`).

FIGURE 3-3

Styling a specific element via an `id`



COMBINING SELECTORS

You can also combine selectors together to refine your selection.

For example, the selector `.score` selects all elements that have the class name score. The selector `li + .score` selects only `<li>` elements that have the class name score. So for example, in the later case, if you added class score to some of your paragraphs, the rule would not apply!

There are many rules for combining selectors:

- An element has multiple classes and you want to find those that have some or all of the classes.
- An element can be selected based on your HTML document structure.
- Elements can have *pseudo classes*, e.g. to change its appearance when you hover over it with your mouse.

You can find more details here: *https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started/Selectors*

Learning Activity 10: Live editing with Developer Tools

Writing CSS blindly is quite hard, isn't it? You have to change the file, then go to your web browser, check the changes, modify the file some more, etc. Wouldn't it be great if you could tweak the CSS values for elements and see the results live?

1. Right-click on one of the best scores in the web page, and click *Inspect Element*. On the right hand side of the developer tools panel (the location of `developer tools` differs for different versions of chrome; in some versions it is under `More tools`), you should be able to see some style information.
2. Try double-clicking, in the developer tools panel, on the name of the colour `gray`. You can now edit it! Replace it with `#7CE8C0`. Experiment with various colour values, by trial and error or by finding a colour picker on the internet. In some browsers, the developer tools also have their own colour pickers:

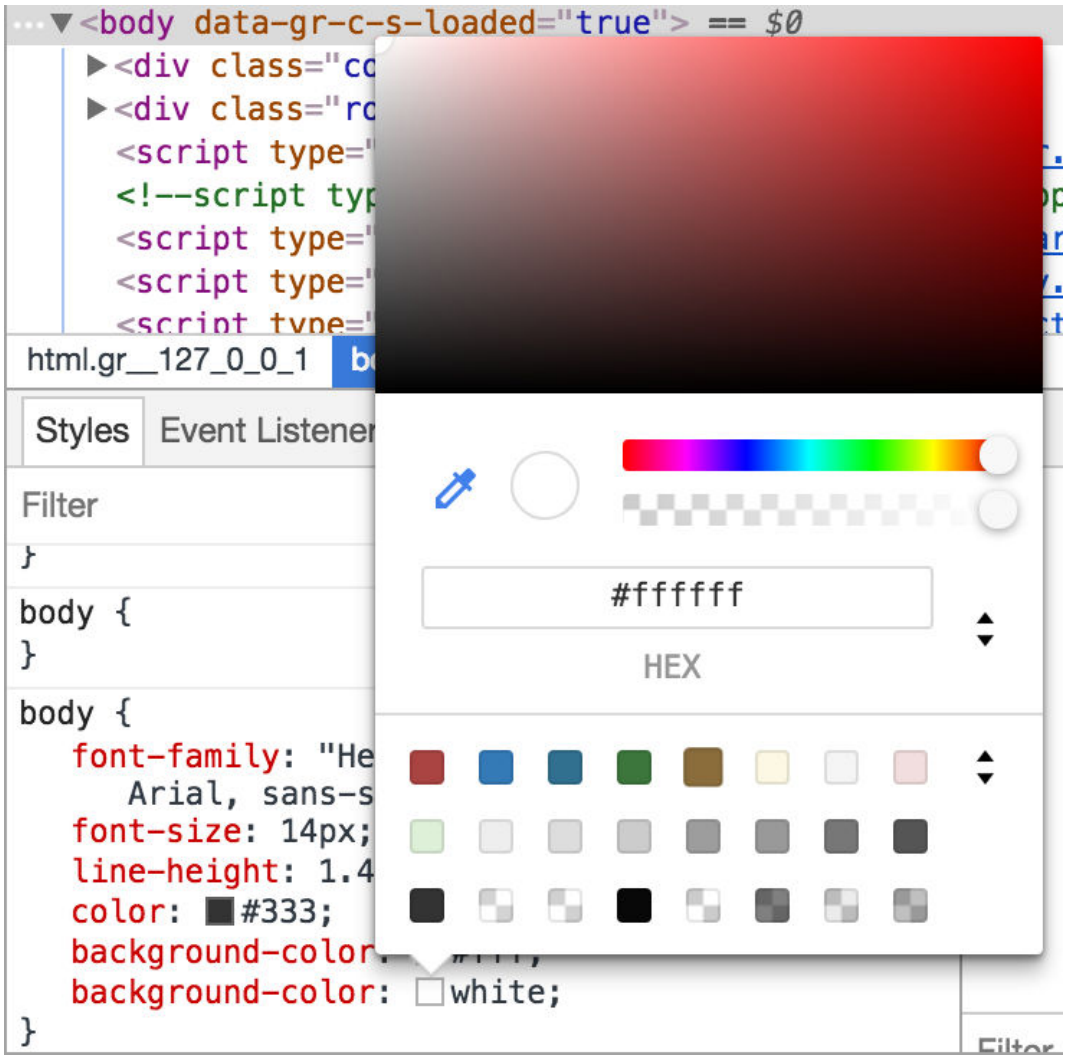


FIGURE 3-4

Using the browser's built-in color picker to select the colour you want

Note that any live editing with the Developer Tools are not reflected back to the original file. You can use this to try things out, but remember to copy the changes back into your CSS file.

Wrap up

- CSS is the language of style for your HTML document.
- You need to specify a stylesheet using the `<link>` tag.
- CSS works by letting you target specific HTML element and apply some style rules to them.
- You can target HTML elements in three ways.
    - By selecting HTML elements by tag names such as `h1` and `p`.
    - By selecting a class with the dot prefix (`.score`).
    - By selecting an ID with the hash prefix (`#gameLink`).

***Final code for this module can be found in** Appendix B*

Advanced layout and styling 4

In this module, you will look in more detail at how you can create compelling layouts for your website. You will learn how to use *Bootstrap*, a CSS library with many predefined CSS styles to make you more productive.

The grid system using bootstrap.css

Writing CSS and selectors can be tedious especially when you need to manage the position of the different parts of your web page. For example, you may want three columns side-by-side which is hard to do in CSS. Even more complicated you might want the number of columns to change depending on the size of the screen: fewer columns on a smart phone, more columns on a desktop computer. This kind of layout is called "responsive" because it responds (reacts) to the changes in size.

Fortunately there's a popular CSS library called *bootstrap* that makes it really easy to design responsive layouts. Bootstrap comes with a lot of pre-defined styles out of the box so you don't have to do a lot of work. In this learning activity, you will see how to use it to create advanced layouts very quickly.

BOOTSTRAP

When you need more details on Bootstrap, you can find comprehensive documentation at *http://getbootstrap.com/css/*

Learning Activity 11: First steps with bootstrap.css

For this learning activity you will need a new HTML file. In the `pages` directory, create a file named `game.html` with empty `<head>` and `<body>` tags.

1. The first thing you need is to import the bootstrap CSS file. Add the following line inside the `<head></head>` section.

   ```
   <link rel="stylesheet" href="../lib/bootstrap.css"/>
   ```

   This is exactly the same as you did for your own stylesheet. However, the bootstrap file is located in the `lib` directory (because it is a library). Make sure the code goes after your other stylesheet (style.css) as each style sheet overrides the previous one.
2. To use the layout facility of bootstrap you will need a tag to hold content. In bootstrap terminology, it is called a "container". Use a `div` tag with the `container` class:

```
<div class="container">
    ❶
</div>
```

❶ Your game's title and welcome message go here.

3. You may be wondering what the `<div>` tag is. It is nothing more than a block element that can group other HTML elements - it defines a division of a document. The `<div>` tags are used to provide additional structure to HTML content.
4. Inside a container you can define rows. They are used to create horizontal groups of columns. To create a row you use the class row:

```
<div class="container">
    <h1>Flappy Birdy!</h1>
    <p>The best Flappy Birdy game there is!</p>
    <div class="row">
        ❶
    </div>
</div>
```

❶ Columns go in there.

5. Let's look at an example to create a two column layout before explaining the bootstrap grid system in more details. Let's say you wanted a two column layout with equal sized columns, one for the game and one for the scores. You would have the following HTML structure:

```
<div class="container">
    <h1>Bob's Flappy Bird.</h1>
    <p>Welcome to the best flappy bird game there is!</p>
    <div class="row"> ❶

        <div class="col-md-6"> ❷
            <p id="gameLink">
                <a href="example.html">Here will be the game</a>
            </p>
        </div>                 ❷

        <div class="col-md-6"> ❸
            <p>The highest scorers are:</p>
            <ul>
                <li class="gold">Me</li>
                <li class="score">Myself</li>
                <li class="score">I</li>
            </ul>
        </div>                 ❸

    </div> ❹
</div>
```

❶ Open the container div.

❷ The first column is here.

❸ And the second column there.

❹ Close the container div.

6. Open the webpage in your browser. Try resizing the window. What happens when the screen gets smaller?

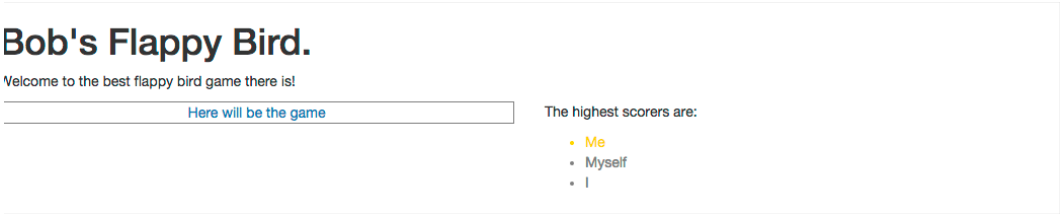


FIGURE 4-1

Two-column page with bootstrap structure on a desktop browser

FIGURE 4-2

Two-column page with bootstrap structure on a mobile browser or a smaller screen

Using `col-md-*` grid classes, you can create a basic grid system that starts out stacked on mobile and tablet devices (ones with a smaller screen) before becoming horizontal on desktop (medium) devices. There are 12 columns and the grid looks as illustrated in **Figure 4-3**.

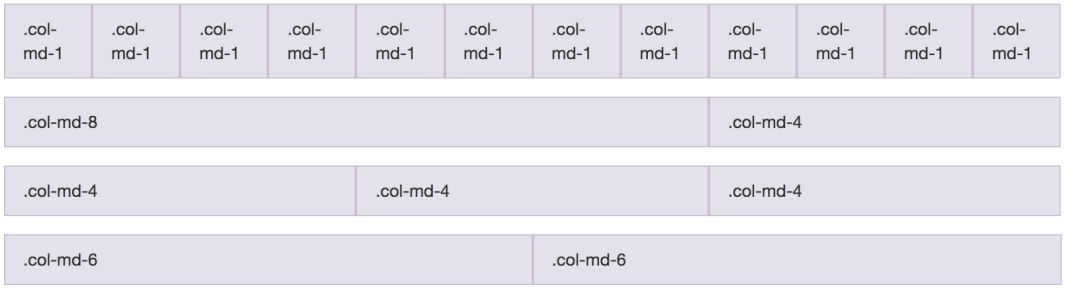


FIGURE 4-3

The grid layout of bootstrap — you can use columns of various sizes to create your own layout

You are free to mix match any of these CSS classes to create new column layouts but you can have maximum 12 columns.

Learning Activity 12: Complex column structures

1. Using the grid visual above, how would you have a three column layout with equal column width?
2. What about a layout with two rows with two columns?
3. What about a layout with three rows and three columns?

There are more classes available in bootstrap to get finer control on how the grid is rendered on mobile devices. For example, you may not want columns to be stacked. You can use the `col-xs-*` classes for this. For example, the following HTML will be displayed as two columns on mobile devices as well as on a desktop:

```
<div class="container">
    <h1>Bob's Flappy Bird.</h1>
    <p>Welcome to the best flappy bird game there is!</p>
    <div class="row">
        <div class="col-xs-8">
            <p id="gameLink">
                <a href="example.html">Here will be the game</a>
            </p>
        </div>
        <div class="col-xs-4">
            <p>The highest scorers are:</p>
            <ul>
                <li class="gold">Me</li>
                <li class="score">Myself</li>
                <li class="score">I</li>
            </ul>
        </div>
    </div>
</div>
```

FIGURE 4-4

Displaying content in two columns even on a small mobile device screen using Bootstrap layout

Other bootstrap components

The bootstrap library is not just about layout. It comes with several useful CSS classes and components such as navigation bar, rounded images and alert boxes etc. We will now explore a few of them. Later you can refer to this page to discover new components: ***http://getbootstrap.com/components/***

Learning Activity 13: Add a responsive game illustration

A page with only text is fairly boring, why don't you add an image instead of `Here will be the game` text? It can still link to your other page:

```
<div class="col-xs-8">
    <p id="gameLink">
        <a href="example.html">
            <img src="../assets/screenshot.png" class="img-responsive" />
        </a>
    </p>
</div>
```

❶ Notice the `.img-responsive` class - it is a Bootstrap utility that makes images resize automatically for different window sizes!

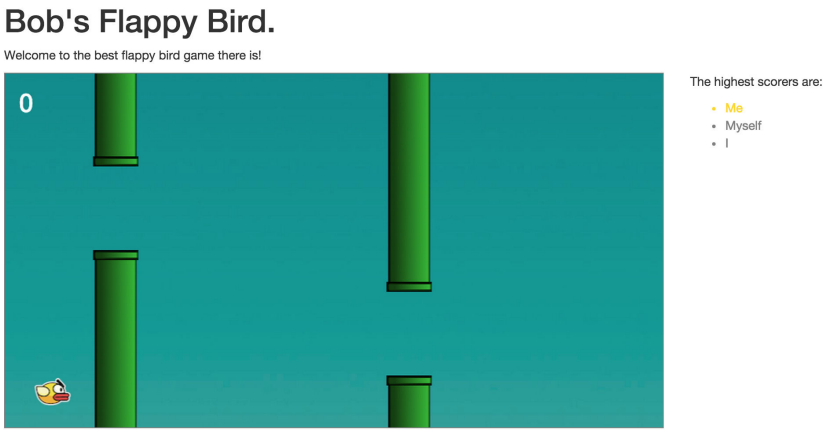


FIGURE 4-5

Responsive illustration instead of just text – resizing of the screen still works!

Learning Activity 14: Add a marketing message

Now you can add a marketing message using the `jumbotron` class right at the beginning of the page, to make your claim that yours really is the best game stand out even more:

```
<div class="container">
    <div class="jumbotron">
        <h1>Bob's Flappy Bird.</h1>
        <p>Welcome to the best flappy bird game there is!</p>
    </div>
    ...
</div>
```

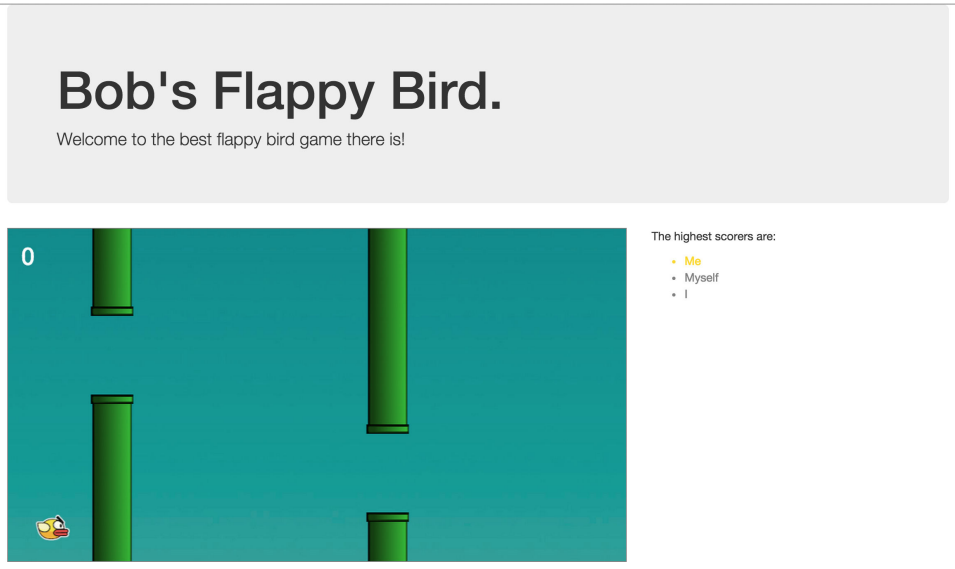


FIGURE 4-6

Use the `jumbotron` class for your marketing message

Learning Activity 15: Rounded borders

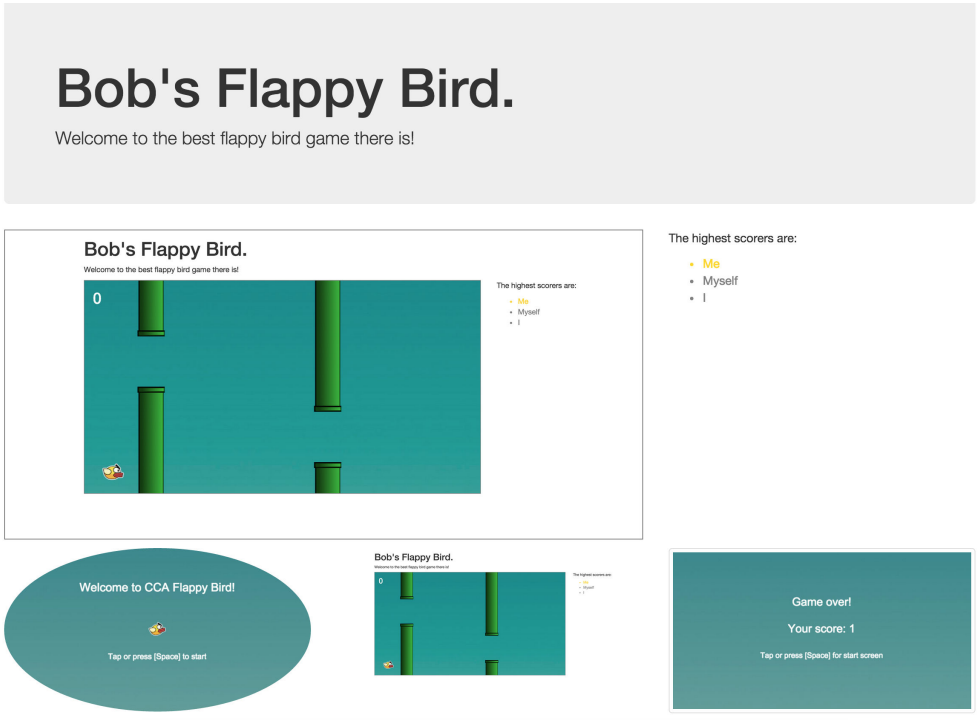
Pictures with rounded borders are very trendy because they add a friendly touch. Bootstrap offers three classes for images that you can use: img-rounded, img-circle and img-thumbnail. You will again find the class img-responsive useful so the image scales automatically to the parent element.

Add multiple screenshots of your game below the main example to make your game page look nicer still!

```
<div class="row">
    <div class="col-md-4">
        <img src="../assets/screenshotStart.png" class="img-circle img-responsive" />
    </div>
    <div class="col-md-4">
        <img src="../assets/screenshot.png" class="img-rounded img-responsive" />
    </div>
    <div class="col-md-4">
        <img src="../assets/screenshotOver.png" class="img-thumbnail img-responsive" />
    </div>
</div>
```

FIGURE 4-7

Use different classes to add borders to images and to scale images with the changing screen size



But stick to one style for most of your images — you will have to admit that our example looks rather odd!

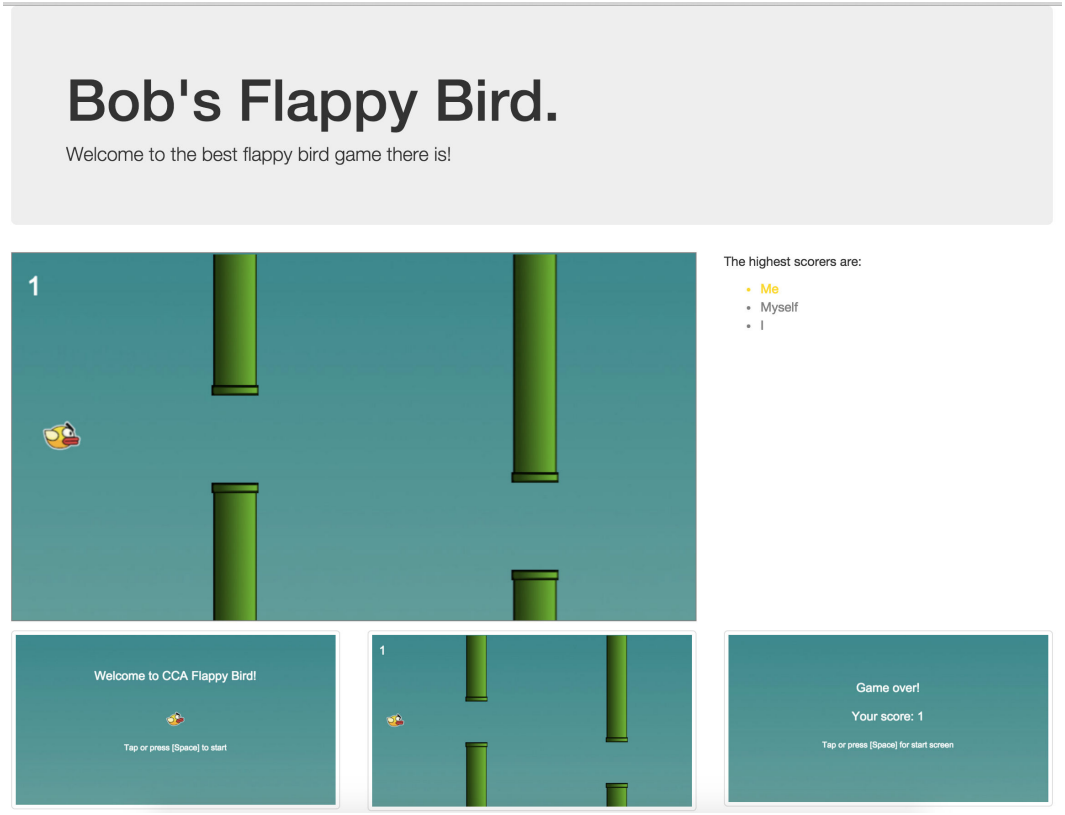


FIGURE 4-8

Our complete example of a page — yours may well look different!

Learning Activity 16: Bonus: Integrate your game into the page

Put your own game within the new page instead of simply linking to it.

1. From the file `example.html`, copy the `div` tag with id `"game"` into `index.html`. The game canvas will appear where that `div` is.
2. From the file `example.html`, copy the `script` tags. In `index.html`, place these tags at the end of the page, just before the closing `</body>` tag. Placing the scripts at the end ensures that the rest of the page is ready when the scripts are loaded.

Wrap up

- *Media queries* let you tailor the presentation of your page to different devices without having to change the content itself.
- Bootstrap is a CSS library that provides many useful styles out of the box
- You can create rich column layout using a combination of `row` and `col-*` classes
- You can nest rows and columns to create arbitrarily complex structures.
- The `<div>` tag is used to provide additional structure to HTML content.
- Use the `jumbotron` class if you want some text to *really* stand out.
- Use the class `img-responsive` to automatically resize images for different screen sizes.
- Use the classes `img-rounded`, `img-circle`, `img-thumbnail` on images to give a nice touch

Final code for this module can be found in Appendix C

5 Adding Interactivity with JavaScript

So far, you've been describing mostly "static" behaviour - how the page should be structured and how it should look. Now we'll look at describing how the page should act and react to user actions. For example, you may want to ask for information from the players of your game and display a confirmation message when they send it. You will use JavaScript to make your website interactive. JavaScript is another programming language understood by all web browsers. Any web page is almost always a combination of HTML, CSS and JavaScript.

In summary:

- HTML is the language of content and structure,
- CSS is the language of styling,
- JavaScript is the language of interactivity.

JavaScript is also used in other context (to make games and other kinds of applications). In this module, you will use JavaScript to make an interactive webpage.

Credits

Let's add a button that lets player know who created the game.

Learning Activity 17: A first interaction

1. Add a div tag somewhere in your webpage with the ID `credits`. At first, it will display the text "Contributors": `<div id="credits">Contributors</div>`. Later, you will make this content change automatically when the user clicks it.
2. In the `js` directory, create a file named `interactivity.js`. You can do this in atom by right-clicking on the `js` folder icon and selecting `New File`.
3. In your HTML file, at the end, just before the closing `</body>` tag, indicate to the web browser that it needs to load the script file:

```
        ...
        <script type="text/javascript" src="../lib/jquery.js"></script>
        <script type="text/javascript" src="../js/interactivity.js"></script>
    </body>
</html>
```

❶ jQuery is a JavaScript library. You've used other libraries in the past: Phaser (for coding the game) and Bootstrap (for layout of your web page). This one helps you add interactivity to HTML document.

❷ The file `interactivity.js` (in the `js` directory) will contain your own code!

4. In the `interactivity.js` file, add the following code:

```
jQuery("#credits").on("click", function() {
    var message = "Game created by Bob!";  ❶
    alert(message);                        ❷
});
```

❶ A variable to hold the contributors message. Change the name to yours!

❷ The function `alert` displays a pop-up message. Try to click on “Contributors” in your web page to see.

Understanding the code

Once you’ve tried it, let’s have a closer look at that code. We’ll add line-breaks to make it easier to discuss the different parts. (Remember that whitespace does not matter in JavaScript if it doesn’t break identifiers.)

```
jQuery("#credits")  ❶
    .on(  ❷
        "click",  ❸
        function() {  ❹
            var message = "Game created by Bob!";  ❺
            alert(message);                        ❺
        }  ❻
    );  ❼
```

❶ The `jQuery` function is provided by the jQuery library. The function takes one argument: a selector. The selectors in jQuery are the same as in CSS. This one is an ID selector. It selects the div you added to your web page. The name jQuery can be shortened to a $ sign.

❷ The method `on` adds event handler to the selected elements.

❸ The first argument to the `on` method specifies what events are considered. Here, the clicks are monitored and will trigger an action. You can use other events such as `"dblclick"`, `"mouseenter"` or `"mouseleave"`.

❹ This callback is triggered whenever the specified event happen. The style is similar to the one you used in your game but it uses an anonymous function.

❺ The code that will be executed when the event happens and the handler is triggered.

❻ Closes the handler function.

❼ Closes the list of arguments for the `on` method.

Learning Activity 18: More advanced interaction

In its current form, the interaction is crude: it shows a bland pop-up box that interrupts the user. Instead, let’s change the content of the web page.

1. In the `interactivity.js` file, change the event handler like so:

```
jQuery("#credits").on("click", function() {
    var message = "Game created by Bob!";  ❶
    jQuery("#credits").append(             ❷
        "<p>" + message + "</p>"           ❸
    );                                     ❹
});
```

❶ The same message, don't forget to customise it.

❷ When handling the event, select the credit div again. This is the same selector than above. However, it uses the `append` method (instead of `on`) which adds content right in the selected element.

❸ Build HTML content (here a paragraph) to insert in the page.

❹ Close the argument list of the `append` function.

2. Run your example (by going to the web page and clicking on "Contributors". Then inspect the page using the browser's developer tools.
3. Change the appended content (in the argument to `append`) and watch the result.

Tabs to select content

In this challenge you will place tabs which allow the user to switch between different content on the web page. You can place the whole thing instead of the score list. The result should look something like this (but don't hesitate to add more buttons or change some of them):

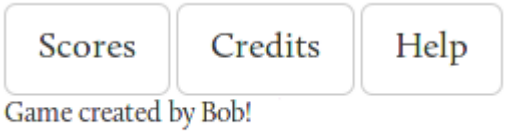


FIGURE 5-1

Tabs: Credits

And when the user clicks on the "Scores" button, it should look something like:

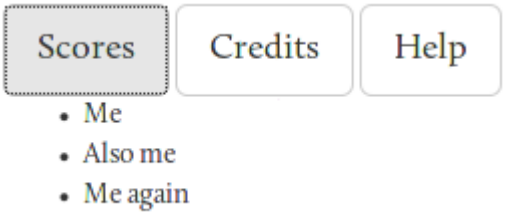


FIGURE 5-2

Tabs: Scores

What you need for this challenge:

1. In jQuery, you can use the `empty` method to remove all the content within a tag. E.g., `jQuery("#that").empty()`.
2. For styling, use Bootstrap's buttons. The full documentation is at this address: ***http://getbootstrap.com/css/#buttons*** but you can also simply use `<button type="button" class="btn btn-default">Scores</button>`.

Advice:

1. Use a div to hold the different contents. Add an ID attribute to this div so you can select it in jQuery.
2. Place three buttons on the page. Add different IDs for each button.
3. In the `interactivity.js` file, add some code to manage the first button. The code should select the button (use an ID selector), add an event handler for click. The event handler should empty the content div and add some different content.

Solution

If you are stuck, you can check the solution below. Note that multiple solutions are valid: yours might look different.

```
<div>
    <button type="button" class="btn btn-default" id="scoresbtn">Scores</button>
    <button type="button" class="btn btn-default" id="creditsbtn">Credits</button>
    <button type="button" class="btn btn-default" id="helpbtn">Help</button>
</div>

<div>
    <div id="content">
    </div>
</div>
```

```
jQuery("#scoresbtn").on("click", function() {
    jQuery("#content").empty();
    jQuery("#content").append(
        "<ul>" +
            "<li>" + "Me" + "</li>" +
            "<li>" + "Also me" + "</li>" +
            "<li>" + "Me again" + "</li>" +
        "</ul>"
    );
});

jQuery("#creditsbtn").on("click", function() {
    jQuery("#content").empty();
    jQuery("#content").append(
        "<div>" + "Game created by Bob!" + "</div>"
    );
});

jQuery("#helpbtn").on("click", function() {
    jQuery("#content").empty();
    jQuery("#content").append(
        "<ul>"
            + "<li>" + "Press SPACE to flap your wings" + "</li>"
            + "<li>" + "Avoid the incoming pipes" + "</li>"
        + "</ul>"
```

```
    );
});
```

Optional improvements

You can improve on the challenge above. Here are several changes you can make:

1. Use tabs instead of buttons. The documentation is on ***http://getbootstrap.com/components/#nav-tabs***
2. Change the look of the buttons/tabs when they are clicked. Check the `active` class. You can add and remove classes in jQuery: ***http://api.jquery.com/category/attributes/***

A real score board

Your page is interactive, but it doesn't yet have a real score board. (Just pretending you have all three top scores is real enough.) Let's change that!

Integrate the game with your page

So far, the game is played on a separate page (`+example.html`). Place the game within your `index.html` page instead.

You do this in two steps:

1. Indicate to the web browser to load the Phaser library and the game code. Add two `script` tags at the end of your web page to load the necessary files:

```
    ...
    <script type="text/javascript" src="../lib/phaser.min.js"></script>
    <script type="text/javascript" src="../js/flappy.js"></script>
</body>
```

2. Place a `div` tag with and ID attribute `game` in the web page, where your game goes.

```
    <div id="game"></div>
```

The Phaser library will replace the content of the `div` with the game canvas and start the game.

Events in your game trigger change on the web page

For the next step, you need functions in your game to change the page outside of the game. This is simple to do.

Learning Activity 19: Calling functions outside the game

1. In `interactivity.js` prepare a function named `registerScore` which takes one argument (`score`) and adds it in a list. Remember you can add content to a web page with the jQuery, using the `append` or `prepend` methods.
2. When you add the score to the list, you can ask the player for their name. Use the `prompt` function for that purpose. It takes one argument (some text to display to the user), it displays the given text and a input box, and it returns what the player inputs.

```
...
var playerName = prompt("What's your name?");
var scoreEntry = "<li>" + playerName + ":" + score.toString() + "</li>";
...
```

3. In `flappy.js` change your `gameOver` function. It has to call `registerScore` and then restart the game (without reloading the page, otherwise the previous scores will be erased).

```
function gameOver(){
    registerScore(score);
    game.state.restart();
}
```

Better score board

With the code as it is now, each score gets added to the score board. You can improve on this. Try the following changes:

1. Only add a score to the board if it is over a certain threshold (say more than 10). You can use a conditional inside the `registerScore` function.
2. Only add a score if it is better than the best score on the board. Use a global variable to remember the best score. Change the condition in `registerScore` function. Remember to update the best score when a better one arrives.

Rehauling

When you add features one by one to a web page, you get side-tracked by this cool addition and that change, etc. It's now time to redo your website almost from scratch. Here are some guidelines.

1. Start with a mock-up on paper. Choose where you want each element to go (will the game be dead centre in the page or on a side? will the additional content (score board, etc.) be on the side or below the game?). Choose a colour palette that fits your theme (for flappy bird: light blue with white highlights for a cloud on sky effect; for flappy fish: dark blue with green highlight for sea weed in the water). Find colour codes that correspond to your colours.
2. Place the different elements of your website, but leave the game canvas out for now. Just put an image of the same size so you can visualise the final result.
3. Tweak the CSS values in the web browser and copy them back to your own CSS file.
4. Add the scripts (including the game one) to your page.
5. Test and improve your design until you're happy with it.

If you get stuck, ask around! You can also check the internet for advice and documentation.

Wrap up

- JavaScript is the language for interactivity.
- User inputs (e.g., mouse clicks and keyboard presses) are events that you can handle in a web page.
- `jQuery` is a useful JavaScript library to react to events, access and update HTML elements dynamically.

Final code for this module can be found in Appendix D

6 Publishing

In this module, you will make your game available to the whole world! You can then share the URL with your friends and challenge them to beat your score.

Publish to Github pages

Github lets you publish web pages. To make use of this functionality, we included a script in the repository. Here's how to run it.

1. Open the `publish.js` script.
2. In the menus, select "Packages" > "Script" > "Run Script".

That's it! You can find your `index.html` web page at the address ***`http://<github-name>.github.io/webapp-flappy/pages/index.html`*** (you need to replace `<github-name>` with your Github login name). Share that URL with your friends. If you received error messages about your device not being configured. Check that Git Plus is located in the packages section of atom. Otherwise ask a teacher or look at the bottom of this section for more information about using git.

Share button

Now that all your friends can play the game, you can add a "Share" button that lets them advertise your game. You'll add a Twitter button.

Learning Activity 20: A simple Share button

Adding a twitter sharing button is easy and only takes one step. Simply add the following in your HTML document wherever you want the Share button to appear.

```
<a href="https://twitter.com/share"  ❶
    target="_blank"                  ❷
>
    Share on Twitter
</a>
```

❶ The target of the link.

❷ This attribute indicates the link should open in a new tab.

If you want a fancier button, use the code provided by Twitter instead: ***https://about.twitter.com/resources/buttons#tweet*** . Alternatively, you can customise it yourself using CSS. Just give the tag an ID attribute and add rule in your style file.

Learning Activity 21: Sharing a score

The Button above is good: it can get some advertisement for your website. But you can do better: let your players boast about their high scores!

1. Change the text displayed in the button (i.e., the content of the `a` tag).

```
<a href="https://twitter.com/share" target="_blank">
    Share your score on Twitter
</a>
```

2. Add an ID attribute to the `a` tag. (Note that if you added one for styling, you can use that ID instead.)

```
<a href="https://twitter.com/share"
    id="sharing"
    target="_blank"
>
    Share your score on Twitter
</a>
```

3. Change the `href` attribute to include a message about the score. The format of the `href` attribute for sharing on Twitter is `"https://twitter.com/share?text=<some-text>"` where `<some-text>` is the content you encourage people to share.
    1. Use jQuery to select the button. You can rely on the ID `sharing`.
    2. Add a handler for the click event.
    3. In the handler, select the button again.
    4. Prepare some text and store it in a variable.
    5. Form the complete URL. You can't use special characters in URL (e.g., space, `?`, `#`, `/`) because they have a special meaning. So you need to escape your text using the `encodeURIComponent` function.
    6. Set the value of the attribute `href` using jQuery's `attr` method. It takes two parameters: the attribute name and the attribute value.
    Here's the finished code:

```
jQuery("#sharing").on("click", function(){
    var text =                                          ❶
        "I scored " +
        score.toString() +
        " in Flappy Birdy! Can you do better?";
    var escapedText = encodeURIComponent(text);         ❷
    var url =                                           ❸
        "https:twitter.com/share?text=" +
        escapedText
    jQuery("#sharing").attr("href", url);               ❹
});
```

❶ Prepare text.

❷ Escape the text so it can be used in the url.

❸ Prepare the url.

❹ Change the target of the link.

Aside: escaping

In a URL, some characters have a special meaning. For example, `/` separates sections of the url. Even spaces are not allowed.

To get around that limitation, characters can be "escaped" (we also say "encoded"). The space for example is replaced by `%20`. Fortunately, you don't have to memorise the code of all the characters and substitute them by hand. In JavaScript, there's a function to do that automatically: `encodeURIComponent`.

Escaping is used in other context. For example, in JavaScript, strings are delimited by double quotes `"like this"`. But then, how can you use the double-quote character inside a string? You escape it with a backslash prefix `"like \"that \"!"`. (But then, backslash is a special character… Which you can use `"like so: \\"`.)

>

Other social media

Many websites let you share pages and messages with your friends. But each uses a different method for sharing. You have to check them individually and adapt the code they give you.

Extra information about git and publishing content to github.

7

There is a method that you can use in the future if you are trying to publish a different project:

First you need to commit all your changes in the given project. Committing changes is marking them as "ready". You normally do that every time you have finished implementing some feature and your files are not "work in progress" any more. You can do this via the terminal or in the case of Windows, Git Cmd. Simply type in the commands:

```
> git add --all ❶
```

This tells git to include all the files in the directory.

Then:

```
> git commit -m "website is ready, preparing to publish." ❷
```

The `-m` flag indicates that there is a message included with the commit. This message explains what the changes are.

Finally, you need to push the committed changes to Github.

```
> git push ❸
```

Your code should now be on your github page online. Simply add a branch called gh-pages to your repository and your new project will be located at the same URL as above.

Aside: `git` and Github

Git is a program that helps you collaborate with other people when you write code. It is also useful for projects you do by yourself: it records a history of changes, lets you roll them back, etc.

Git is complicated to use because it is a very powerful tool with many options to control different behaviours.

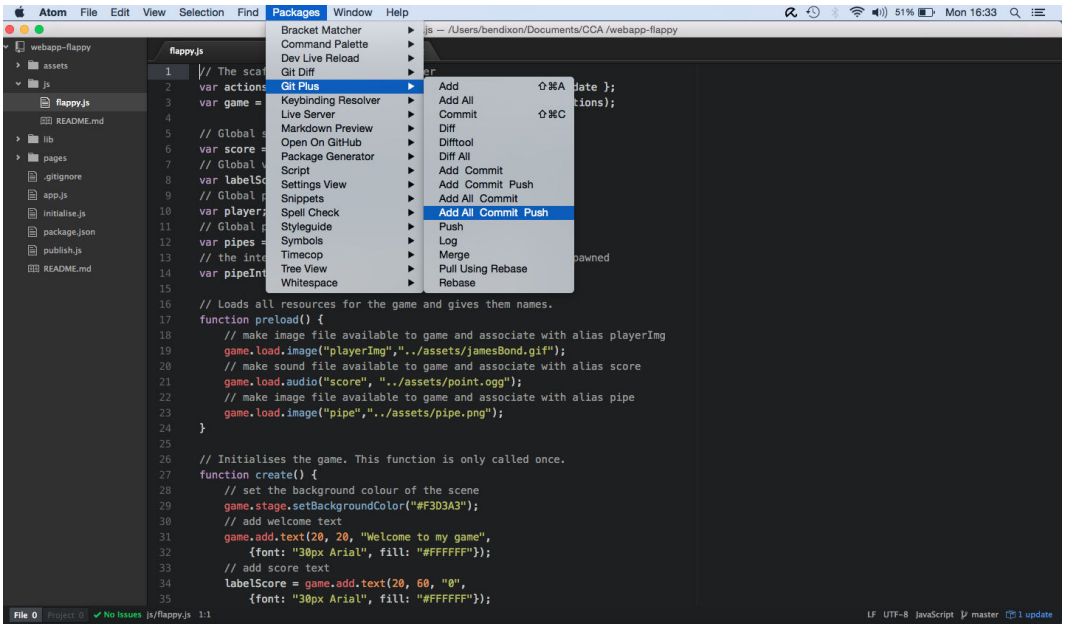
Github is a website that hosts code. It lets you create repository, share code, collaborate to fix bugs, etc.

If you do not like using the terminal or the command prompt, you can do the same process with atom:

- Git add can be achieved by going selecting "Packages" > "Git Plus" > "Add All Commit Push"

FIGURE 7-1

Complete the git process in one step — The updated code should then be on github.



Final code for Module 1

A

HTML (index.html)

```
<!DOCTYPE html>
<html>

<head>
    <title>Bob's Flappy Bird</title>
</head>

<body>
    <h1>Bob's Flappy Bird.</h1>
    <p>Welcome to the best flappy bird game there is!</p>
    <p>
        <a href="example.html">Here will be the game</a>
    </p>

    <img src="../assets/screenshot.png" />

    <p>The highest scorers are:</p>
    <ul>
        <li>Me</li>
        <li>Myself</li>
        <li>I</li>
    </ul>
</body>

</html>
```

Final code for Module 2

B

```
<!DOCTYPE html>
<html>

<head>
    <title>Bob's Flappy Bird</title>
    <link rel="stylesheet" href="../assets/style.css"/>
</head>

<body>
    <h1>Bob's Flappy Bird.</h1>
    <p>Welcome to the best flappy bird game there is!</p>
    <p id="gameLink">
        <a href="example.html">Here will be the game</a>
    </p>

    <p>The highest scorers are:</p>
    <ul>
        <li class="gold">Me</li>
        <li class="score">Myself</li>
        <li class="score">I</li>
    </ul>
</body>

</html>
```

CSS (style.css)

```
h1 {
    color: darkblue;
}

.score {
        color: gray;
}

.gold {
        color: gold;
}

#gameLink {
    text-align: center;
    border: 1px solid gray;
}
```

Final code for Module 3

C

HTML (index.html)

```
<!DOCTYPE html>
<html>

<head>
    <title>Bob's Flappy Bird</title>
    <link rel="stylesheet" href="../assets/style.css" />
    <link rel="stylesheet" href="../lib/bootstrap.css" />
</head>

<body>
    <div class="container">
        <div class="jumbotron">
            <h1>Bob's Flappy Bird.</h1>
            <p>Welcome to the best flappy bird game there is!</p>
        </div>
        <div class="row">
            <div class="col-xs-8">
                <p id="gameLink">
                    <a href="example.html">
                        <img src="../assets/screenshot.png"
                            class="img-responsive"/>
                    </a>
                </p>
            </div>
            <div class="col-xs-4">
                <p>The highest scorers are:</p>
                <ul>
                    <li class="gold">Me</li>
                    <li class="score">Myself</li>
                    <li class="score">I</li>
                </ul>
            </div>
        </div>

        <div class="row">
            <div class="col-md-4">
                <img src="../assets/screenshotStart.png"
                    class="img-thumbnail img-responsive" />
            </div>
            <div class="col-md-4">
                <img src="../assets/screenshot.png"
                    class="img-thumbnail img-responsive" />
            </div>
            <div class="col-md-4">
                <img src="../assets/screenshotOver.png"
                    class="img-thumbnail img-responsive" />
```

```
            </div>
        </div>
    </div>
</body>

</html>
```

Final code for Module 4

D

HTML (index.html)

```
<!DOCTYPE html>
<html>

<head>
    <title>Bob's Flappy Bird</title>
    <link rel="stylesheet" href="../assets/style.css" />
    <link rel="stylesheet" href="../lib/bootstrap.css" />
</head>

<body>
    <div class="container">
        <div class="jumbotron">
            <h1>Bob's Flappy Bird.</h1>
            <p>Welcome to the best flappy bird game there is!</p>
        </div>
        <div class="row">
            <div class="col-xs-8">
                <div id="game"> </div>
            </div>
            <div class="col-xs-4">
                <p>The highest scorers are:</p>
                <ul>
                    <li class="gold">Me</li>
                    <li class="score">Myself</li>
                    <li class="score">I</li>
                </ul>
            </div>
        </div>

        <div class="row">
            <div class="col-md-4">
                <img src="../assets/screenshotStart.png"
                    class="img-thumbnail img-responsive" />
            </div>
            <div class="col-md-4">
                <img src="../assets/screenshot.png"
                    class="img-thumbnail img-responsive" />
            </div>
            <div class="col-md-4">
                <img src="../assets/screenshotOver.png"
                    class="img-thumbnail img-responsive" />
            </div>
        </div>
    </div>

    <div id="greeting">
```

```
        <h3>Save your score</h3>
        <form id="greeting-form" class="form-inline">
            <div class="form-group">
                <label for="fullName">What's your name?</label>
                <input type="text" id="fullName"
                    name="fullName" class="form-control" placeholder="Bob">

                <label for="email">What's your email?</label>
                <input type="email" id="email"
                    name="email" class="form-control">

                <input type="hidden" id="score"
                    name="score" value="0">
              </div>
            <button class="btn btn-default" type="submit">OK</button>
        </form>
    </div>
    <script src="../lib/jquery.js"></script>
    <script src="../lib/phaser.min.js"></script>
    <script src="../js/flappy.js"></script>
</body>

</html>
```

Game code (js/flappy.js)

At the begining of the file, with the variable initialisation:

```
jQuery("#greeting-form").on("submit", function(event_details) {
    var greeting = "Hello ";
    var name = jQuery("#fullName").val();
    var greeting_message = greeting + name;
    jQuery("#greeting-form").hide();
    jQuery("#greeting").append("<p>" + greeting_message + " (" +
        jQuery("#email").val() + "): " + jQuery("#score").val() + "</p>");
    event_details.preventDefault();
});
```

In the `gameOver` function:

```
function game_over() {
    game.destroy();
    $("#score").val(score);
    $("#greeting").show();
}
```

Form styling (assets/style.css)

```
h1 {
    color: darkblue;
}

.score {
        color: gray;
}

.gold {
        color: gold;
```

```
}

#gameLink {
    text-align: center;
    border: 1px solid gray;
}

#greeting {
        display: none;
        position: absolute;
        left: 40%;
        top: 40%;
        background: #fff;
        width: 350px;
        height: 200px;
        border: 2px solid gray;
        border-radius: 5px;
        padding: 20px;
}
```


CAMBRIDGE
CODING ACADEMY