

Practical 02: Javascript

The purpose of the checkpoints is to provide you with the skills necessary to complete the practical component of the web development assignment, which constitutes a significant portion of your marks for this topic. You are encouraged to seek assistance from your demonstrators on any aspect of the tasks that you do not understand — *we are here to help*.

While you are encouraged to discuss the concepts covered in this topic with your fellow students, please bear in mind that all work you produce and submit must be your own and that submissions will be subject to similarity checks. As a student, you should ensure you are familiar with the University's [Academic Integrity Policy](https://students.flinders.edu.au/my-course/academic-integrity) and the penalties that can result from academic dishonesty and plagiarism. To that end, you may, at any time, be asked to explain your task solutions and the process you went through to develop them.

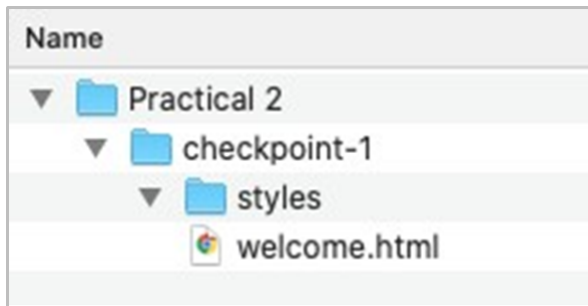
The files necessary to complete this practical can be found at this [link](https://canvas.flinders.edu.au/courses/5687/files/887338?wrap=1) [↓](https://canvas.flinders.edu.au/courses/5687/files/887338/download?download_frd=1) [. \(https://canvas.flinders.edu.au/courses/5687/files/887338/download?download_frd=1\)](https://canvas.flinders.edu.au/courses/5687/files/887338/download?download_frd=1) .

Checkpoint 1

For this practical, you will build a simple quiz using a combination of HTML, CSS, and JavaScript.

1. Similar to the previous practical, you should first establish a directory (folder) structure for the files you will be working on in the following checkpoints. Begin by creating a directory **prac-2** for Practical 2 somewhere on your machine.
2. Download **p2-checkpoint1-starter.zip** from the link above and extract the contents into your Practical 2 directory. You should now have a directory structure that looks like the following:





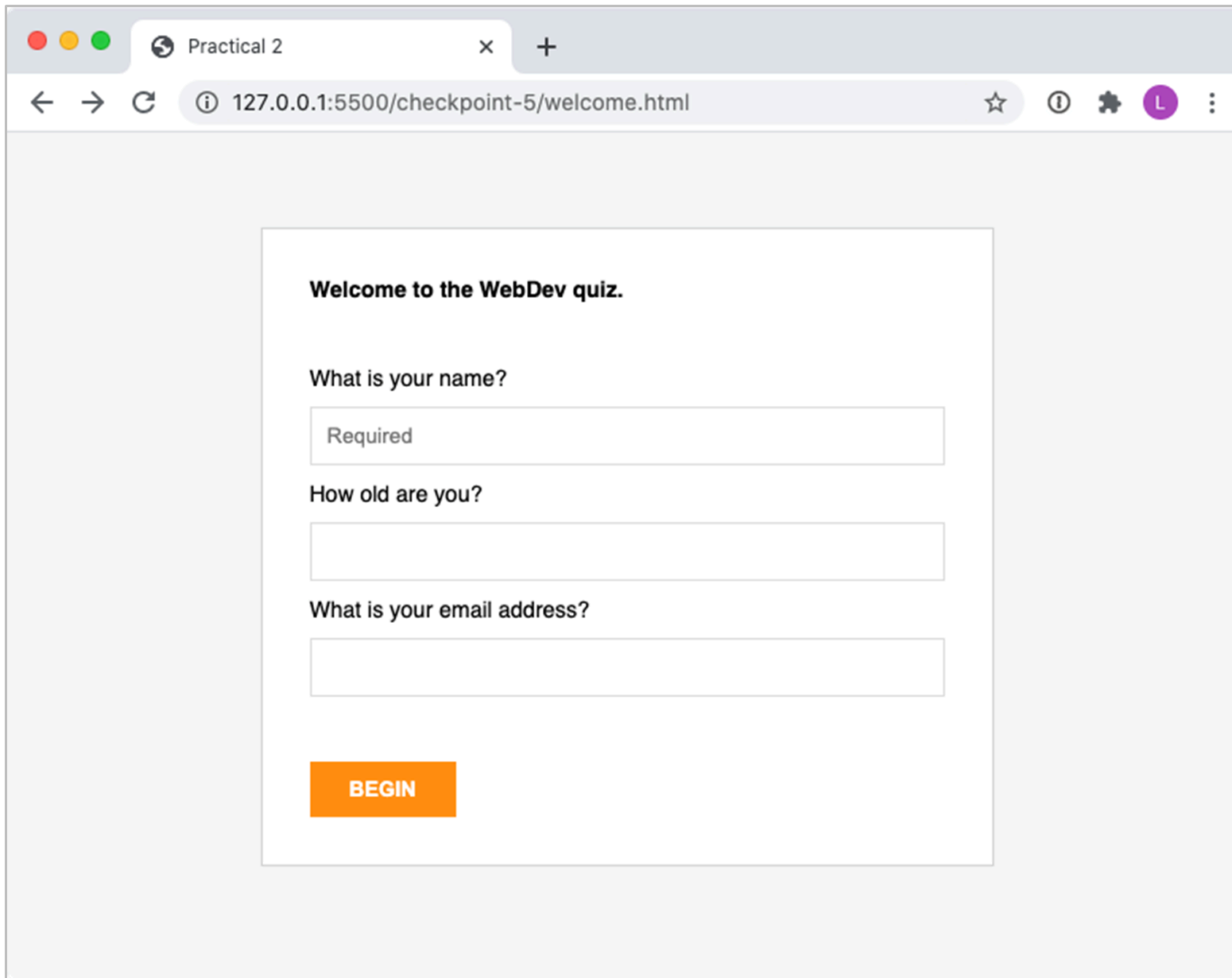
1. Within the **styles** subdirectory, create a new file called **style.css**. All CSS must be defined in this file. There should be no internal (via style tags) or inline (via a style attribute) CSS used.
2. Open the **checkpoint-1** directory in VS Code.
3. The **welcome.html** file contains the core structure of a welcome page that greets quiz 'players' and asks them to provide a few details before they begin. The first checkpoint will involve styling this page and adding HTML5 validation to the form fields.

Edit **welcome.html** and **style.css** accordingly to produce a page that looks similar to the image below.

Note that minor differences are fine as long as your page is a close replica.

- The page is expected to have **charset**, **author** and **description** metadata elements. The charset should be UTF-8, the author should be you, and the description should be 'Practical 2'.
- The **title** of the page should also be 'Practical 2'.
- Add a **link** element that refers to the **styles/style.css** file.
- Modify the form elements to add HTML5 form validation. You will need to modify existing attributes or add new attributes to accomplish this.
 - The **name** field should be of type **text**, have a **minlength** value of 2, and be set to **required**.
 - The **age** field should be of type **number**, have a **min** value of 10, and a **max** value of 100.
 - The **email** field should be of type **email**.
 - The **begin** button should be of type **submit**.
- Add **id** or **class** attributes as necessary to facilitate styling with CSS
 - You may wish to add further div and/or span elements to your page to help with forming document sections
 - Of course, you can also target elements directly with CSS selectors if this is appropriate





A screenshot of a web browser window. The title bar shows 'Practical 2' and a close button. The address bar shows '127.0.0.1:5500/checkpoint-5/welcome.html'. The page content is a white box on a light gray background. Inside the box, the text 'Welcome to the WebDev quiz.' is at the top. Below it are three questions: 'What is your name?', 'How old are you?', and 'What is your email address?'. Each question has a text input field. The first input field has the placeholder text 'Required'. At the bottom of the box is an orange button with the text 'BEGIN'.

Practical 2

127.0.0.1:5500/checkpoint-5/welcome.html

Welcome to the WebDev quiz.

What is your name?

Required

How old are you?

What is your email address?

BEGIN

Key style details

- The font is Helvetica (fallback to Arial then sans-serif) and is 14px in size.



- The background colour is #5f5f6.
 - The content area...
 - is 400px wide with a white background and 1px border in the colour #d2d2d2
 - has a padding on all sides of 30px
 - is centred in the browser window
 - The **name**, **age**, and **email** input elements have...
 - a border of 1px in the colour #dddddd
 - a padding on all sides of 10px
 - a top and bottom margin of 10px
 - a width of 100%
 - a box-sizing of 'border-box' to include the padding in the browser's size calculations and prevent overflow
 - The **begin** button has...
 - no border
 - a background colour of #ff8f00 (with a hover background colour of #f57c00)
 - bold text in uppercase
 - a padding of 10px top/bottom and 25px left/right
1. Verify that your form validates the input as follows when you click begin:
- Requires a name value. If a name is not provided, you should see:



The screenshot shows a web form with two input fields. The first field is labeled "What is your name?" and contains the text "Required" in a purple box. The second field is labeled "How old are you?". A tooltip with an orange exclamation mark icon and the text "Please fill in this field." is pointing to the first field.

- Requires a name value of at least 2 characters. If the value is less than 2 characters, you should see:



What is your name?

! Please lengthen this text to 2 characters or more (you are currently using 1 character).

- Requires an age value between 10 and 100. If the value is outside this range, you should see a message similar to:

How old are you?

What is ! Value must be less than or equal to 100.

- Requires a valid email address value. If the email address is malformed, you should see:

What is your email address?

! Please include an '@' in the email address.
'invalid_email_address' is missing an '@'.

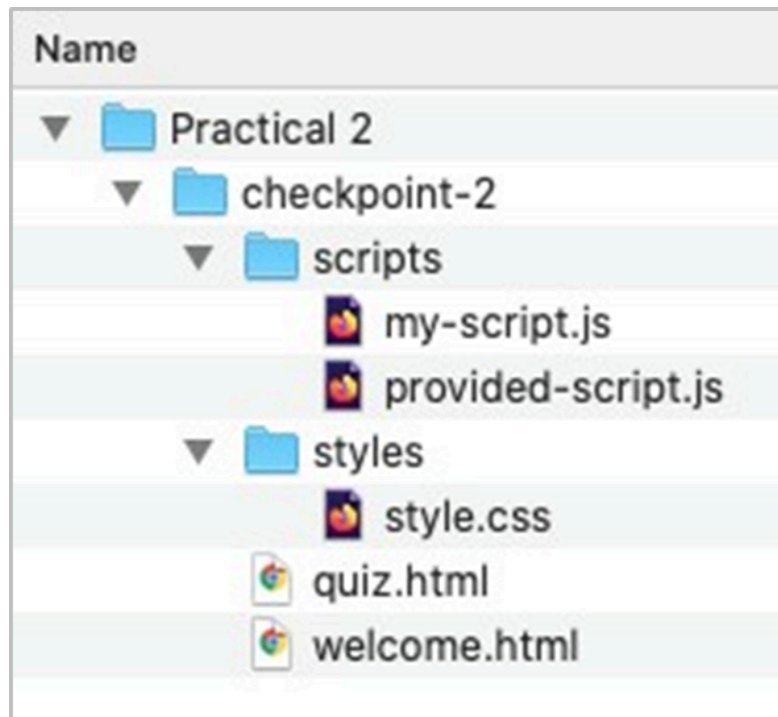
Checkpoint 2

This checkpoint will involve styling a second HTML page that will become the basis for the quiz. You will then configure the form you created in the previous checkpoint to submit to this page and finally add JavaScript to read the form values.

1. Make a copy of your **checkpoint-1** directory and name it **checkpoint-2**.



2. Download **p2-checkpoint2-starter.zip** from FLO and extract the contents into your **checkpoint-2** directory. You should now have a **quiz.html** file alongside your **welcome.html** file and a new **scripts** directory.



1. Open the **checkpoint-2** directory in VS Code.
2. The **quiz.html** file contains the core structure of the quiz page. Edit **quiz.html** and **style.css** accordingly to produce a page that looks similar to the image below.

Note that minor differences are fine as long as your page is a close replica.

- The page is expected to have **charset**, **author** and **description** metadata elements. The charset should be UTF-8, the author should be you, and the description should be 'Practical 2'.
- The **title** of the page should be 'Practical 2'.
- Add a **link** element that refers to the **styles/style.css** file.
- The **input** and **label** elements already contain appropriate attributes for this practical, so you should not need to remove any prov in the template page. You may wish to add attributes, however.
- Add **id** or **class** attributes as necessary to facilitate styling with CSS.



- You may wish to add further div and/or span elements to your page to help with forming document sections.
- Of course, you can also target elements directly with CSS selectors if this is appropriate.

Key style details (in addition to those from Checkpoint 1):

- The question number text is bold and coloured #ff8f00.
- The list item elements have the same appearance as the name, age, and email input fields from before, which is
 - a border of 1px in the colour #dddddd
 - a padding on all sides of 10px
 - a top and bottom margin of 10px
 - a width of 100%
 - a box-sizing of 'border-box' to include the padding in the browser's size calculations and prevent overflow
- The **results** div should be hidden (i.e., a display value of 'none') initially. A class would be useful for this.



The screenshot shows a web browser window with the title 'Practical 2' and a single tab. The address bar displays the URL '127.0.0.1:5500/checkpoint-2/quiz.html'. The main content area features a white rectangular box with a light gray border. Inside this box, the text 'Hello, **Player**. Let's begin.' is displayed. Below this, the heading 'Question N' is shown in orange, followed by the text 'Question text should appear here'. Three radio button options are listed: 'Choice A', 'Choice B', and 'Choice C'. At the bottom of the box is an orange button with the text 'NEXT' in white.

1. Modify the **form** element in **welcome.html** by adding **method** and **action** attributes:
 - The **action** attribute should have a value of **quiz.html**
 - The **method** attribute should have a value of **GET**




This will enable the form values (name, age, and email) to be passed to quiz.html as a series of name-value query parameters known as a query string. If you fill in the form and click begin, you should now be taken to quiz.html. You should also see the query string appended to the end of the URL.

1. The next step is to use JavaScript to read the query parameters from the query string. For this checkpoint, you will extract the player's name and update the greeting message on quiz.html.

Open

quiz.html and add a **script** tag to the head element that references **scripts/provided-script.js**. The tag should be defined as follows:

```
<script src="scripts/provided-script.js" defer></script>
```

The **defer**  (https://www.w3schools.com/tags/att_script_defer.asp) attribute tells the browser to defer loading the script until after the HTML document has been parsed. Add another **script** tag directly below that references **scripts/my-script.js**. Add the **defer** attribute to this as well. The **provided-script.js** file contains supporting JavaScript for the checkpoint tasks — you should *avoid modifying this file* and *write all of your JavaScript in the my-script.js file*. Following this guideline will make the marking process easier for tutors.

1. Open **my-script.js**. Within the **init** function, create a new variable (or constant) and assign it the element representing the player's name in quiz.html. If you recall, the player's name was displayed in the following HTML:

```
<h1>Hello, <b>Player</b>. Let's begin.</h1>
```

There are various ways to accomplish this. One approach would be to add an **id** attribute to the **b** element and then use JavaScript's **getElementById** function to retrieve a reference to it. You could also add a different tag around the player's name (with an id attribute), recreate the h1 inner HTML entirely, or use an alternative JavaScript function such as **querySelector**. You are encouraged to experiment with different approaches so you get comfortable accessing HTML from JavaScript.

To use the first approach, for example, add an **id** attribute to the **b** element as follows:

```
<h1>Hello, <b id="player-name">Player</b>. Let's begin.</h1>
```

In the **init** function, get a reference to the element by writing:



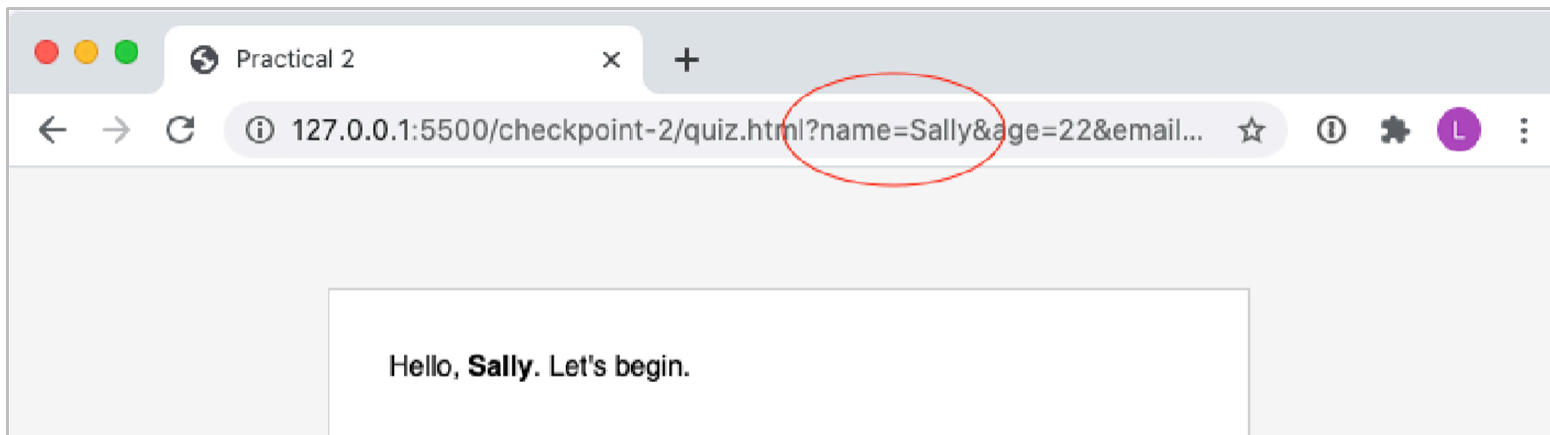
```
const player = document.getElementById("player-name");
```

You can set the content of this element to one of the query parameter values by using the **getUrlParam** function, which has been defined for you in **provided-script.js**. This function accepts the query parameter name as an argument and returns the corresponding value. Therefore, setting the player name can be accomplished with the statement:

```
player.innerText = getUrlParam("name");
```

The **innerText** property allows you to set the textual content of an element. The '**name**' value passed to the function corresponds to the query parameter for the player's name.

1. If you complete the form fields on **welcome.html** and click *begin*, you should now see the name you entered shown on the resulting **quiz.html** page.



Checkpoint 3

This checkpoint will involve displaying question information on the quiz.html page.

1. Make a copy of your **checkpoint-2** directory and name it **checkpoint-3**.
2. Open your **checkpoint-3** directory in VS Code.
3. Edit your **my-script.js** file as follows:



- Define a new function called **getQuestion**. The function has no formal parameters. Within this function, add code to update the contents of the elements in the **quiz** div with valid question information.

For this checkpoint, you only need to retrieve and display the first question. Support for moving between multiple questions will be added in the next checkpoint.

- Question data can be retrieved from the **questions** array, which has been defined for you (much in the same way as the **getUrlParam** function) in **provided-script.js**. This array consists of a collection of JavaScript objects that represent each question. Each object has the following properties:

```
question
choiceA
choiceB
choiceC
answer
```

All properties contain **string** values. The answer value matches the associated choice letter.

- Replace the placeholder text of the question text paragraph — *question text should appear here* — with the value of the first array element's **question** property. You can access this property as follows:

```
questions[0]["question"]
```

- Update the choice A label with the value of the first array element's **choiceA** property.
 - Update the choice B label with the value of the first array element's **choiceB** property.
 - Update the choice C label with the value of the first array element's **choiceC** property.
 - You can use a similar approach as before to target the HTML elements using a function such as **getElementById**.
 - Add a call to your **getQuestion** function at the end of your **init** function.
1. Verify you see information for the first question appear when the **quiz.html** page is loaded. Your page output should look similar to the following:



Hello, **Dave**. Let's begin.

Question N

A popular code editor for web development is...

☐ Word

☐ Visual Studio Code

☐ Photoshop




NEXT

Checkpoint 4

This checkpoint will involve adding functionality to support moving between different quiz questions.

1. Make a copy of your **checkpoint-3** directory and name it **checkpoint-4**.
2. Open your **checkpoint-4** directory in VS Code.
3. Edit your **my-script.js** file as follows:
 - Define a global variable to store the current question number. Initialise it to 0.
 - Modify your **getQuestion** function to fetch and display sequential questions.



- Update the statements that query the properties from the questions array (per Checkpoint 3) to use your question number variable as the array index instead of 0.
- Modify the placeholder text of the question number paragraph — *Question N* — to reflect the value of your question number variable. The first question should appear as *Question 1* so you will need to adjust the output to account for the fact that the variable value begins at 0.
- Define a new function called **next** with a single formal parameter called **event**.
- Add an **event listener**  (https://www.w3schools.com/js/js_htmlDOM_eventlistener.asp) for the **next** button to your **init** function. The event listener should register for **click** events and call the **next** function to handle the event.
 - You will need to get a reference to the **next** button before adding the event listener.
 - An **Event**  (https://www.w3schools.com/jsref/dom_obj_event.asp) object will be implicitly passed into your **next** function as the actual parameter value for **event**.
- Within your **next** function...
 - Increment your question number variable.
 - Check if there are remaining questions. You can query the number of questions in the questions array by reading the **length** property. If there are remaining questions to ask, call your **getQuestion** function to process the next question. If there are no more questions, hide the next button. You can access the next button element via the event parameter's **target**  (https://www.w3schools.com/jsref/event_target.asp) property.

Hint: to hide an element using JavaScript, look into functions for modifying the element's CSS style/classes.

- Finally, as we are using JavaScript to change the content of the question elements for each new question, the player's previous selection will be maintained when a new question is loaded. To reset the options (radio buttons), call the **clearSelection** function as follows either before or after you call **getQuestion**.

```
clearSelection("choices");
```

The **clearSelection** function has, again, been defined for you in **provided-script.js** alongside **getUrlParam**. The '**choices**' parameter passed to the function refers to the name of the radio button group in **quiz.html**.

- Verify the following:
 - You can move between questions when clicking the next button.
 - The question number text updates when moving to a new question.



- Any option selected by the player is cleared when a new question is loaded.
- When there are no more questions, the next button disappears.

Checkpoint 5

The final checkpoint will involve calculating the quiz score and displaying the final results.

1. Make a copy of your **checkpoint-4** directory and name it **checkpoint-5**.
2. Open your **checkpoint-5** directory in VS Code.
3. Edit your **my-script.js** file as follows:
 - Define another global variable to store the number of questions the player answers correctly. Initialise it to 0.
 - Modify your **next** function to evaluate the player's selections (answers).
 - At the top of the function, add code to compare the player's selection with the current question answer. Much like the **clearSelection** function used in the last checkpoint, a **getSelection** function has also been defined for you in **provided-script.js** that you can use to help with this step. The function returns the value of the selected radio input element as a string. In this case, it will be a single letter: 'A', 'B', or 'C'. If no selection is made, an empty string will be returned instead. The **getSelection** function should be called as follows:

```
getSelection("choices");
```

As before, the '**choices**' parameter refers to the name of the radio button group in quiz.html. The answer for a question can be retrieved from the **answer** property.

- If the player answers the question correctly, increment your 'correct' variable.
- You may find JavaScript's **console.log** function useful when working through this step.
- Define a new function called **showResults**. The function has no formal parameters. Within this function, add code to calculate and report the final quiz score.
 - Calculate the player's overall score percentage and store the result in a temporary local variable.
 - Replace the placeholder text of the paragraph in the **results** div — your final score was ? — with a report of the player's final score. Your report output should be presented as follows:



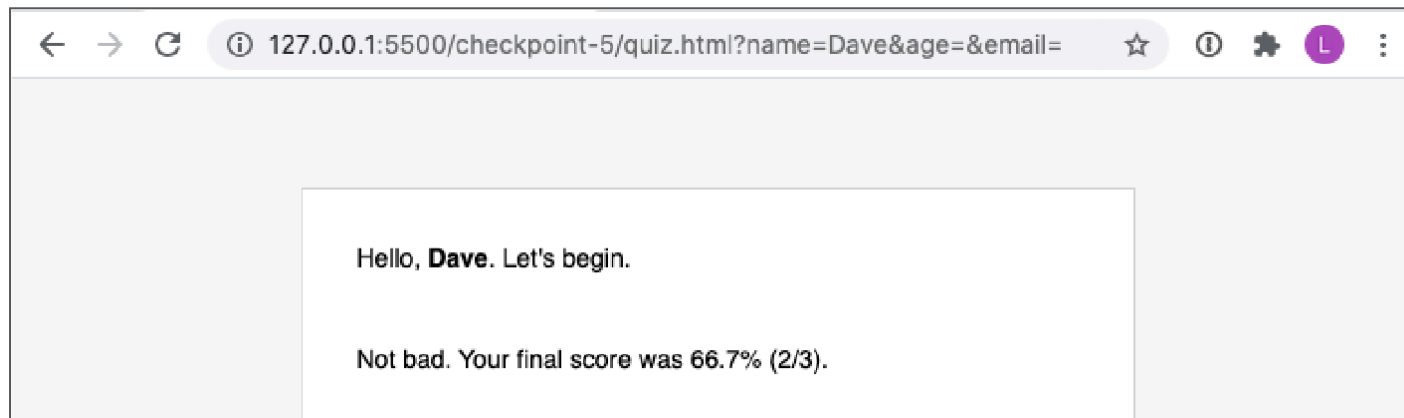
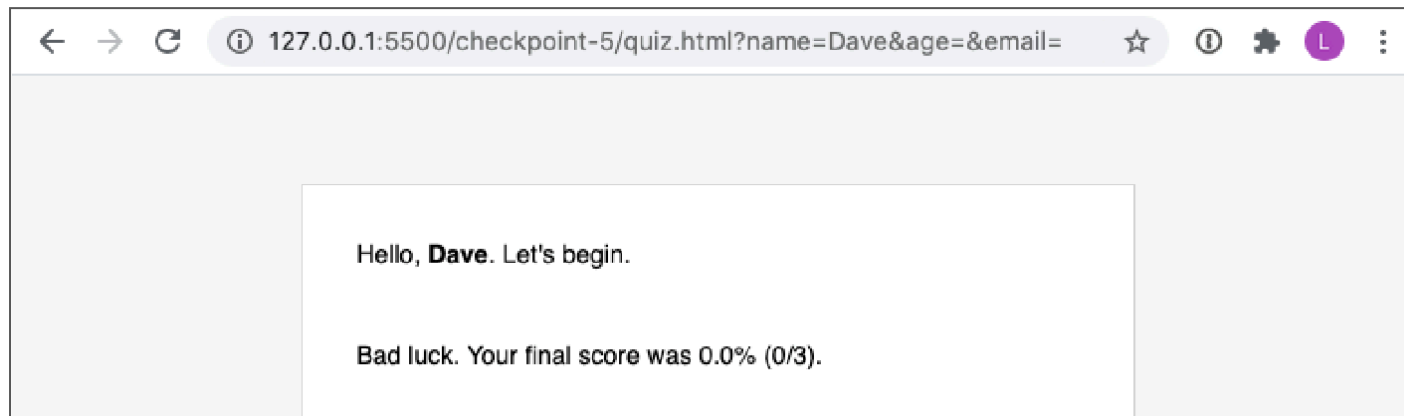
where **X** is the calculated percentage, formatted to one decimal place, **Y** is the number of questions the player answered correctly, and **Z** is the total number of questions asked.

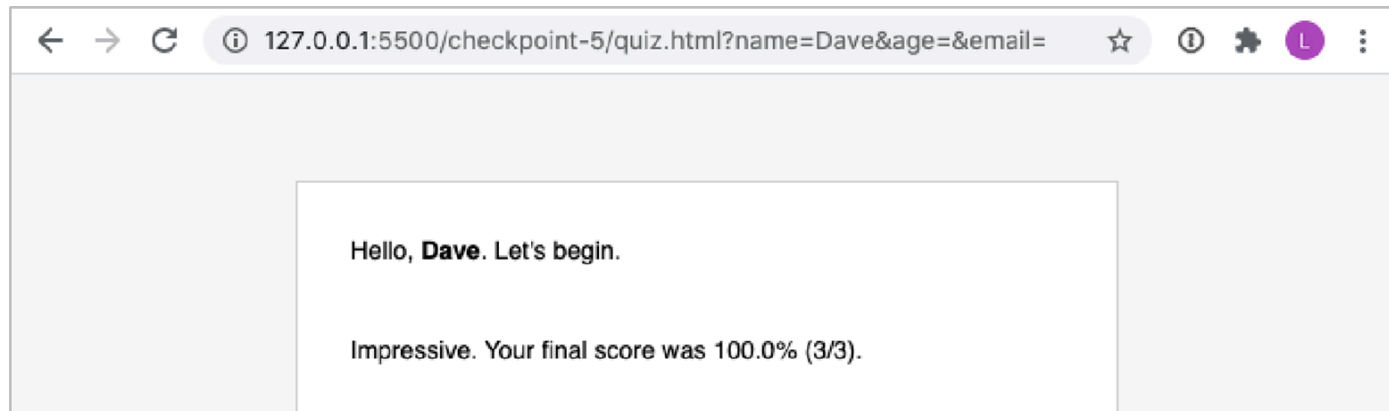
If X is less than 30, display: 'Bad luck. Your final score was X.X% (Y/Z).'

If X is between 30 and 75, display: 'Not bad. Your final score was X.X% (Y/Z).'

If X is greater than 75, display: 'Impressive. Your final score was X.X% (Y/Z).'

- Finally, hide the **quiz** div and show the **results** div.





1. Add a call to your **showResults** function after the statement in your **next** function that hides the next button (after all questions have been asked).
2. Verify that your quiz correctly evaluates each answer and presents the appropriate final report output upon completion.

Demonstration of final quiz for Practical 2



Practical 2

127.0.0.1:5500/checkpoint-5/welcome.html

Welcome to the WebDev quiz.

What is your name?

Arnold

How old are you?

37

What is your email address?

arnold@reddwarf.com

BEGIN

0:00 / 0:30

