**Coding for GUIs (JavaScript edition)**

**LESSON 4: Flipping images**

This is the fourth in a series of lessons to incorporate graphical user interfaces (GUIs) into your general-purpose programming. The series follows on from the [Visual to text coding lesson series](https://www.digitaltechnologieshub.edu.au/teachers/lesson-ideas/visual-to-text-coding-index-page).

Included videos can be used by a beginner teacher and/or students to see how to code each of the simple programs step by step with HTML, CSS and JavaScript.

This lesson may take two to three 45-minute periods. It introduces JavaScript code to add and change images on a webpage dynamically. This is useful for creating visual effects beyond changing text.

**Year levels**: 7–8

## **Learning hook**

In pairs, students should visit [userinyerface.com](https://userinyerface.com/) and try the challenge.

User Inyerface is an online ‘game’ where the player must survive working through a frustratingly designed sign-up form. The challenge is to see whether you can get through the whole form.

While working through, the student pair needs to write down as many poor user interface design choices as they can.

**Note:** Although the game does not use personal details, it is recommended that students enter fake details when filling out the form.

Finally, come together as a class and have student pairs call out each design problem as you proceed through. For each one, discuss what could have been done instead to improve the user experience.

The activity above touches on the third of four principles for good user interface design (distilled from [this article](https://xd.adobe.com/ideas/process/ui-design/4-golden-rules-ui-design/) hosted by Adobe): Reduce cognitive load.

A GUI should:

* organise elements visually for clarity, chunking or grouping elements where appropriate
* allow recognition through familiar symbols and tooltips
* minimise the number of actions required to complete a task (e g. the ‘[three click rule-of-thumb](https://en.wikipedia.org/wiki/Three-click_rule)’).

## **Learning map and outcomes**

By the end of this lesson students will:

1. access an online programming environment for **JavaScript** alongside **HTML** and **CSS**
2. create an interactive **dice roll simulation** with dice face images that change when you roll
3. create a dynamic data visualisation to show **teaspoons of sugar** in various drinks
4. take on a fresh coding challenge to create your own dynamic data visualisation.

## **Learning input**

The videos below introduce the two main applications we'll make in this lesson.

* Video 4.1: [Dice roll](https://publish.viostream.com/player/download/bxixurbdbw9ar7)
* Video 4.2: [How much sugar overview](https://publish.viostream.com/player/download/bxixurbdbw9tbb)

**Discuss as a class:** How can students ensure that images they are using in their GUIs are not restricted due to copyright?

**Suggested answer:**By understanding [Creative Commons](https://creativecommons.org.au/learn/education/) and Public Domain licenses, and learning about the places to find them, students can use images appropriately, giving attribution when necessary.

##  **Learning construction**

**STEP 1: Set-up**

In this course, different environments will be selected based on their suitability for each demonstrated project.

View this [video](https://publish.viostream.com/player/download/bxixurbdnosebo) on setting up.

For more on the set-up and environments used, see [Lesson 1](https://www.digitaltechnologieshub.edu.au/teachers/lesson-ideas/coding-for-guis-lesson-1-structure-style-and-function/#learning-construction).

**STEP 2: Constructing the dice roll simulator**

This [video](https://publish.viostream.com/player/download/bxixurbdbw95ya) builds the digital clock application from the ground up.

Try it yourself before checking the solution code.

You'll need these six images for the die sides:

* [1.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/1.png)
* [2.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/2.png)
* [3.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/3.png)
* [4.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/4.png)
* [5.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/5.png)
* [6.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/6.png)

*Solution code:* [*repl.it environment*](https://repl.it/%40digitechhub/Lesson-4-STEP-2-Solution)

This short [video](https://publish.viostream.com/player/download/bxixurbdbw95y5) demonstrates how the same application can optionally be developed offline on your own computer desktop.

Remember, you'll need a **plain text editor** to edit the code files (.html, .css and .js). Do not attempt to work in Microsoft Word or Google Docs.

SIDEBAR – Plain text files

A simple website is made up of files inside folders just like on your computer. To place the site **online** so that other people can access it, the files must be placed on a host web server. (While it is possible to make your own computer into a web server, this is generally unwise because it greatly increases your exposure to cyber attacks. Instead, web page content files are usually hosted by dedicated companies.)

The files for your HTML, CSS and JavaScript code are **plain text files**. Unlike Microsoft Word documents or Google Docs, plain text files do not have any formatting (bold, italics, etc.), fonts or page settings. They are simply made up of characters, spaces and new lines.

To work with plain text files on your own computer:

* Windows comes with **Notepad**.
* Mac comes with **TextEdit**.

These programs do not highlight your code with colours. For this functionality other software is required, like [Notepad++](https://notepad-plus-plus.org/) for Windows or [Brackets](http://brackets.io/) for Mac.

**STEP 3: Tinkering with the dice roll simulator**

By editing the HTML, CSS and JavaScript, upgrade the dice roll application as follows:

* Instead of two 6-sided dice, the program now rolls two 12-sided dice.
* You'll need these additional twelve images for the new die's sides:
	+ [BigDie01.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/bigdie01.png)
	+ [BigDie02.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/bigdie02.png)
	+ [BigDie03.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/bigdie03.png)
	+ [BigDie04.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/bigdie04.png)
	+ [BigDie05.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/bigdie05.png)
	+ [BigDie06.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/bigdie06.png)
	+ [BigDie07.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/bigdie07.png)
	+ [BigDie08.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/bigdie08.png)
	+ [BigDie09.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/bigdie09.png)
	+ [BigDie10.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/bigdie10.png)
	+ [BigDie11.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/bigdie11.png)
	+ [BigDie12.png](https://www.digitaltechnologieshub.edu.au/images/default-source/Lesson-ideas-details/coding-for-guis/lesson-4/bigdie12.png)

(Images screen-captured from [teal 3D dice roller](http://a.teall.info/dice/) website.)

*Solution code:* [*repl.it environment*](https://repl.it/%40digitechhub/Lesson-4-STEP-3-Solution-A)

* *Extra extension task:* The user now has a choice whether to roll 6-sided dice or 12-sided dice.



*Solution code:* [*repl.it environment*](https://repl.it/%40digitechhub/Lesson-4-STEP-3-Solution-B)

**STEP 4: Second program – how much sugar?**

This [video](https://publish.viostream.com/player/download/bxixurbdbw9tbn) demonstrates how to create a dynamic data visualisation showing how much sugar is in various drinks.

Try it yourself before checking the solution code.

*Solution code:* [*repl.it environment*](https://repl.it/%40digitechhub/Lesson-4-STEP-4-Solution)

**STEP 5: Tinkering with 'how much sugar?'**

Make the following adjustments to the solution:

* Research one more drink of your choice. You'll need an image of the drink as well as the number of teaspoons of sugar in it.
* By editing the HTML, CSS and JavaScript, add the new drink to the solution so that it behaves like the others.
* Also add a link to your source below the first source in the footer.

**Challenge**

These challenges use the skills covered so far. By writing or modifying their own programs, students have an opportunity to demonstrate Application and Creation.

1. This challenge is to create a new dynamic data visualisation, using a small amount of data sourced from an online repository or a class survey.
	1. Start by deciding what data you will try to display. A simple set of 10 or fewer values is enough.
	2. Prepare an annotated mock-up to show how the solution will look and work. When the user clicks or mouses over a category, your solution might:
		1. repeat an image to represent a whole number, similar to the 'How much sugar?' approach
		2. scale a single image to represent different relative amounts (see the sample solution below)
		3. perform some other visualisation of data.
	3. Find or create the image(s) you'll need for your solution.
	4. Use HTML, CSS and JavaScript to create the data visualisation. You might begin by forking the 'How much sugar?' solution in this lesson.

***Sample solution:***[*Agricultural area by region*](https://repl.it/%40digitechhub/Lesson-4-Challenge-1-SOLUTION)When the user mouses over a region name, a single barn image is resized to represent the relative area given over to agricultural purposes in that region in 2016. A little bit of maths is used to get a linear scale value so that the image is resized according to the desired area.

1. This second challenge is more open-ended. Now that you know how to switch images, see if you can create a simple game that employs buttons to take actions and a single image to represent a location or status.

For example:

* 1. An adventure game with buttons to move North, South, East and West. A text area explains what is happening in the story, and an image gives an illustration.
	2. A combat game with buttons to attack, defend or use magic/items. The enemy is shown as an image that changes as the enemy is wounded or defeated.

Games like the above will involve significantly more JavaScript code than previous challenges in this course. Consider organising your code with functions for the buttons and a main function called to update the game once a button has its effect.

## **Resources**

* Online environments for creating webpages with HTML, CSS and JavaScript:
	+ [JSFiddle](https://jsfiddle.net) – simple interface that hides linking HTML code, also used in [Visual to text coding lesson series](https://www.digitaltechnologieshub.edu.au/teachers/lesson-ideas/visual-to-text-coding-index-page)
	+ [repl.it](https://repl.it) – shows complete HTML to reflect offline approach, and allows uploading of images and other files for use in webpages
* [Visual to text coding lesson series](https://www.digitaltechnologieshub.edu.au/teachers/lesson-ideas/visual-to-text-coding-index-page) – The predecessor to this learning sequence introduces JavaScript as well as Python.
* JavaScript [CheatSheet](https://htmlcheatsheet.com/js/) (Tip: Press the little blue tabs to move **Variables**, **Basics**, **Strings** and **Data Types** to the top.)
* Articles and lessons on user interface principles:
	+ [UI Design principles distilled](https://xd.adobe.com/ideas/process/ui-design/4-golden-rules-ui-design/) (Adobe)
	+ [Topic page](http://www.digitaltechnologieshub.edu.au/teachers/topics/user-interface) on Digital Technologies Hub
	+ [Classic CS Unplugged lesson](http://www.classic.csunplugged.org/human-interface-design/)
	+ [code.org lesson](https://curriculum.code.org/pwc/ayp/6/)

# **Assessment**

Students undertake a self-reflection of the programming task. The teacher uses the completed self-assessments to assist in summative assessment.

* Download the self-assessment sheet in [Word](http://www.digitaltechnologieshub.edu.au/docs/default-source/Lesson-Ideas/coding-for-guis/lesson-4/self-assessment---coding-for-guis---lesson-4.docx) or [PDF](http://www.digitaltechnologieshub.edu.au/docs/default-source/Lesson-Ideas/coding-for-guis/lesson-4/self-assessment---coding-for-guis---lesson-4e6c54c9809f96792a599ff0000f327dd.pdf) format.

In assessing code in languages like JavaScript, consider a rubric that brings in important skills for general-purpose programming.

* Download a sample rubric in [Word](https://www.digitaltechnologieshub.edu.au/docs/default-source/Lesson-Ideas/visual-to-text-coding/rubric-example---software-design-and-development-with-general-purpose-language6baf4a9809f96792a599ff0000f327dd.docx) or [PDF](https://www.digitaltechnologieshub.edu.au/docs/default-source/Lesson-Ideas/visual-to-text-coding/rubric-example---software-design-and-development-with-general-purpose-language.pdf) format.

# **Australian Curriculum**

Links with Australian Curriculum: Digital Technologies

| **Strand** | **Year** | **Content description** |
| --- | --- | --- |
| Processes and Production Skills | 7-8 | Design the user experience of a digital system [(AC9TDI8P07)](https://v9.australiancurriculum.edu.au/f-10-curriculum.html/learning-areas/digital-technologies/year-7_year-8/content-description?subject-identifier=TECTDIY78&content-description-code=AC9TDI8P07&detailed-content-descriptions=0&hide-ccp=0&hide-gc=0&side-by-side=1&strands-start-index=0&subjects-start-index=0&view=quick)Generate, modify, communicate and evaluate alternative designs [(AC9TDI8P08)](https://v9.australiancurriculum.edu.au/f-10-curriculum.html/learning-areas/digital-technologies/year-7_year-8/content-description?subject-identifier=TECTDIY78&content-description-code=AC9TDI8P08&detailed-content-descriptions=0&hide-ccp=0&hide-gc=0&side-by-side=1&strands-start-index=0&subjects-start-index=0&view=quick)Design algorithms involving nested control structures and represent them using flowcharts and pseudocode [(AC9TDI8P05)](https://v9.australiancurriculum.edu.au/f-10-curriculum.html/learning-areas/digital-technologies/year-7_year-8/content-description?subject-identifier=TECTDIY78&content-description-code=AC9TDI8P05&detailed-content-descriptions=0&hide-ccp=0&hide-gc=0&side-by-side=1&strands-start-index=0&subjects-start-index=0&view=quick)Trace algorithms to predict output for a given input and to identify errors [(AC9TDI8P06)](https://v9.australiancurriculum.edu.au/f-10-curriculum.html/learning-areas/digital-technologies/year-7_year-8/content-description?subject-identifier=TECTDIY78&content-description-code=AC9TDI8P06&detailed-content-descriptions=0&hide-ccp=0&hide-gc=0&side-by-side=1&strands-start-index=0&subjects-start-index=0&view=quick)Implement, modify and debug programs involving control structures and functions in a general-purpose programming language [(AC9TDI8P09)](https://v9.australiancurriculum.edu.au/f-10-curriculum.html/learning-areas/digital-technologies/year-7_year-8/content-description?subject-identifier=TECTDIY78&content-description-code=AC9TDI8P09&detailed-content-descriptions=0&hide-ccp=0&hide-gc=0&side-by-side=1&strands-start-index=0&subjects-start-index=0&view=quick?subject-identifier=TECTDIY78&content-description-code=AC9TDI8P09&detailed-content-descriptions=0&hide-ccp=0&hide-gc=0&side-by-side=1&strands-start-index=0&subjects-start-index=0&view=quick) |