**Coding for GUIs (JavaScript edition)**

**LESSON 2: Controls for input**

**Summary:**

This is the second 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 buttons and other controls for data input, such as **range sliders** and **updowns**.

**Year level**: 7-8

## **Learning hook**

As part of a graphical user interface (GUI), **input controls** are the different elements available to your user for providing information and issuing commands, such as buttons, text fields or checkboxes.

Students should start by completing the worksheet ‘Choosing the best input control’. [Download the worksheet](http://www.digitaltechnologieshub.edu.au/docs/default-source/Lesson-Ideas/coding-for-guis/Lesson-2/worksheet_choosing-the-best-input-control_final.docx).

Now, discuss:

* When do you think a **dropdown** control is more useful than a **text field** control, and when is the reverse true?
* When is a **spinner/updown** control more useful than a **text field** control?
* Some of these controls have been around for decades. What other more recent GUI elements have you seen on websites or apps?

Note: The activity above touches on the first 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). For users to feel in control, a GUI should be:

* easy to navigate (with visual cues that make actions predictable)
* acknowledging (giving feedback and status)
* accommodating of different skill levels
* forgiving (e.g. include an ‘undo’ function)

## **Learning map and outcomes**

By the end of this lesson students will:

* access an online programming environment for **JavaScript** alongside **HTML** and **CSS**
* upgrade three text applications to GUI applications with input controls
* take on a fresh coding challenge to explore and incorporate an input control of their choice.

## **Learning input**

This [video](https://publish.viostream.com/player/download/bxixurbdb1my5y) introduces the first application for this lesson.

Test out the original JavaScript program without a GUI, so you can see our starting point:

**Starting point:** [JSFiddle environment](https://jsfiddle.net/nathanesa/5Lskqf6r/), [repl.it environment](https://repl.it/@digitechhub/Lesson-2-Learning-input#script.js)

## **Learning construction**

##### Options

**Step 1: Set-up**

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

View these videos on setting up environments: [CodePen](https://publish.viostream.com/player/download/bxixurbdnosebc) and [JSFiddle](https://publish.viostream.com/player/download/bxixurbdnosebm).

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: Upgrading the wi-fi checker to GUI interface**

This [video](https://publish.viostream.com/player/download/bxixurbdb1mgyc) demonstrates changing the wi-fi checker from a text-only program to one with a GUI.

The difference between HTML, CSS and JavaScript

Let's start with some terms. Read these carefully, then answer the questions below.

|  |  |
| --- | --- |
| **HTML:** HyperText Markup Language  HTML is one of the earliest languages of the internet. It defines the **structure** of a webpage, the static content that will appear on the page when it first loads. It also allows for links. You can make a webpage purely in HTML, but it's hard to make it look good!  You might think of HTML as the **skeleton, muscles and nerves** of a webpage. |  |
| **CSS:** Cascading Style Sheets  CSS is a script that defines the **style** of a webpage, allowing for webpages to have interesting and modern look-and-feel: colours, fonts, curved borders, etc.  You might think of CSS as the **skin and** **clothes** of a webpage. |  |
| **JavaScript**  JavaScript defines the **functionality** of a webpage, allowing interactivity beyond just links. It is the only true general-purpose programming language of the three. With JavaScript, webpages can have all the functionality of applications, with loops, decisions, functions and data structures.  You might think of JavaScript as the **brain** of a webpage. |  |

Try it yourself before checking the solution code.

**Solution code**: [JSFiddle environment](https://jsfiddle.net/nathanesa/fd6vj7zg/), [repl.it environment](https://repl.it/@digitechhub/Lesson-2-Learning-construction-STEP-2)

**Step 3: Adding a spinner/updown control**

This [video](https://publish.viostream.com/player/download/bxixurbdb1mngj) further improves the wi-fi checker by adding a spinner/updown control, which gives an instant response.

Try implementing this improvement before checking the solution code.

**Solution code**: [JSFiddle environment](https://jsfiddle.net/nathanesa/Lxgva4ws/), [repl.it environment](https://repl.it/@digitechhub/Lesson-2-Learning-construction-STEP-3)

**Step 4: Second application – temperature converter**

This [video](https://publish.viostream.com/player/download/bxixurbdb1mr3u) demonstrates upgrading a temperature converter to a GUI.

Try coding this GUI version yourself before checking the solution code.

**Starting point** (no GUI): [JSFiddle environment](https://jsfiddle.net/nathanesa/jdwnc7q3/), [repl.it environment](https://repl.it/@digitechhub/Lesson-2-STEP-4-starting-point), [original lesson](https://www.digitaltechnologieshub.edu.au/teachers/lesson-ideas/visual-to-text-coding-lesson-1-temperature-converter)

**Solution code**: [JSFiddle environment](https://jsfiddle.net/nathanesa/709wfbzq/), [repl.it environment](https://repl.it/@digitechhub/Lesson-2-STEP-4-solution)

**Step 5: Third application – password generator**

This [video](https://publish.viostream.com/player/download/bxixurbdb1mr3o) demonstrates upgrading a password generator application to a GUI.

Try coding this GUI version yourself before checking the solution code.

**Starting point** (no GUI): [JSFiddle environment](https://jsfiddle.net/nathanesa/dqr3shp5/), [repl.it environment](https://repl.it/@digitechhub/Lesson-2-STEP-5-starting-point), [original lesson](https://www.digitaltechnologieshub.edu.au/teachers/lesson-ideas/visual-to-text-coding-lesson-5-password-generator/)

**Solution code**: [JSFiddle environment](https://jsfiddle.net/nathanesa/gcrj3obm/), [repl.it environment](https://repl.it/@digitechhub/Lesson-2-STEP-5-solution)

## **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. The links below are a simple text-only application for converting currency. Test it out before continuing.

**Starting point** (no GUI): [JSFiddle environment](https://jsfiddle.net/nathanesa/feqdam0b/), [repl.it environment](https://repl.it/@digitechhub/Lesson-2-Challenge-1-Starter-program#script.js)

Your task is to upgrade this application to have a GUI using HTML, JavaScript and CSS. You can choose the best way to receive the user input and to present the result. (Note: your application does not have to use live currency conversion rates.)

* 1. Start by sketching a rough GUI mock-up to show what the user will see and interact with on the webpage. Include text labels and fields, buttons and other input controls.

GUI mock-ups

Mock-ups or wireframes are used during user interface design to visually plan or sketch the basic look and functionality.

Sometimes multiple ‘frames’ will be drawn, to show how the interface changes when the user navigates and makes choices within it.



Importantly, mock-ups can be **annotated** with notes and arrows to explain details about the look and feel (e.g. style, fonts, colours) or to explain functionality (e.g. this option appears when this button is pressed).

Tools for wireframing range from free and simple to highly complex paid tools intended for user experience (UX) professionals.

Some free options for school students:

* Draw on paper. Use one colour for the interface elements and another colour for annotations. Sometimes grid paper can help.
* Free online tool [diagrams.net](https://app.diagrams.net/) (select **+ more shapes** to add GUI mock-up elements).
* Presentation software (such as Microsoft PowerPoint or Google Slides) tends to allow more freedom of element placement than word-processing software.
  1. Develop the webpage from the text-only starting point, using your preferred environment for HTML, CSS and JavaScript.

**Sample solution**

This [video](https://publish.viostream.com/player/download/bxixurbdb1p856) gives an overview of one sample solution. This [video](https://publish.viostream.com/player/download/bxixurbdb1pekb) demonstrates building the same solution. Could you improve it further by removing the need for pop-up prompts?

1. This challenge is to create a GUI to allow you to experiment with an input control you haven't tried yet. For example, if you are trying **radio buttons/option buttons**, you might create a simple application that constructs a movie idea as a sentence based on your selections (e.g. gender of main character, setting, genre).
   1. Choose an input control to try. You might want to consider one of the controls from the [worksheet](http://www.digitaltechnologieshub.edu.au/docs/default-source/Lesson-Ideas/coding-for-guis/Lesson-2/worksheet_choosing-the-best-input-control_final.docx) at the start of this lesson. See GUI side bar from Challenge 1.
   2. Next, consider what sort of simple application could use that control, and prepare a simple mock-up of the full interface, including title, any images, the input controls and output area.
   3. Develop the webpage from scratch, using your preferred environment for HTML, CSS and JavaScript. Or you can ‘fork’ and modify one of the basic starting points below.

HINT: As in the lesson examples, you can use the HTML <input> tag to make your input control(s). The **oninput** attribute activates your JavaScript function, which will then get the current value from the input control and use it. Alternatively, you can have a <button> tag to make a button with an **onclick** attribute pointing to your JavaScript function*.*

[This W3Schools webpage](https://www.w3schools.com/tags/att_input_type.asp) lists the many **type** attributes that allow different input controls to be made with the <input> tag.

*Starting point:* [*JSFiddle environment*](https://jsfiddle.net/nathanesa/egrt1fja/)*,* [*repl.it environment*](https://repl.it/@digitechhub/Lesson-2-Challenge-2-Starting-point)

## **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-2/self-assessment---coding-for-guis---lesson-2.docx) or [PDF](http://www.digitaltechnologieshub.edu.au/docs/default-source/Lesson-Ideas/coding-for-guis/Lesson-2/self-assessment---coding-for-guis---lesson-221c14c9809f96792a599ff0000f327dd.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 curriculum area

| **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) |