DT Challenge Year 7/8 JavaScript – Cookie Clicker

https://groklearning.com/course/aca-dt-78-js-cookie/

About this activity

JavaScript is a programming language used to make the web interactive. It is a very popular language, with most modern websites having a JavaScript component.
In this course, students will learn about web technologies, and how JavaScript can be used to create interactive webpages.
Students will also learn basic HTML and CSS, and combining these, will create an interactive and addictive *incremental* game: Cookie Clicker.

Age

This course targets students in year 7 and 8, though it can also be used as an advanced course for students in earlier years, especially those who are ready to move beyond block based visual coding languages.

Language

JavaScript — a text based programming language used extensively for the web. The course also introduces HTML and CSS.

Time

The course is designed to be completed in 15 hours of class time.

Key Concepts

Key Concept	**Coverage**
Abstraction	
Data: collection, representation, interpretation	Data representation as integer and string
Specification, algorithms, implementation	Simple Algorithms
Digital Systems	
Interaction	Interaction (interactive webpage input/output)
Impact	

Objectives (Content Descriptions)

ACTDIK015	**Examine how whole numbers are used to represent all data in digital systems**
ACTDIP019	**Design, modify and follow simple algorithms involving sequences of steps, branching**, and iteration (repetition)

	Note: This course does not cover iteration.
ACTDIP030	**Implement and modify programs with user interfaces involving branching**, iteration and functions in a general-purpose programming language. *Note: This course covers programs with user interfaces involving branching, but not the other elements described in this curriculum outcome, nor a general purpose programming language.*
ACTDIP018	**Design a user interface for a digital system**.
ACTDIP028	**Design the user experience of a digital system**, generating, evaluating and communicating alternative designs.
ACTDIK035	**Analyse** simple compression of data and **how content data are separated from presentation**. *Note: This course covers only the second half of this curriculum outcome, and does not cover compression of data.*

What are we learning? (Abstract)

At the conclusion of this course students will be able to:

- Write programs using a JavaScript, a programming language used extensively online
- Recognise that breaking down a problem into smaller steps (deposition) makes it easier to solve problems
- Debug algorithms
- Recognise that steps in algorithms need to accurate and precise
- Recognise that problems can have multiple solutions
- Utilise user input
- Add text, images to webpages
- Modify the styles of websites, whilst separating content and style elements
- Use CSS to create a styled website, changing colours and font family
- Define the term *algorithm*
- Define the term *decomposing*
- Define the term *branching*
- Define the term *function*

Module outline

The course consists of seven modules:

1. Showing the cookie
   This module introduces webpages and web technologies. Students learn how to add paragraphs, headings and pictures.
2. Clicking the cookie
   This module introduces students to JavaScript, a powerful web programming language. Students learn how to make alerts, events and other dialog boxes and explore the basics of interacting with the user of a webpage. This module also introduces functions, which allow students to define code '*recipes*'.
3. Selecting the cookie
   This module introduces the concept of the DOM, the Document Object Model, which lets JavaScript interact with the HTML of a webpage. Students learn how to make more interactive and engaging webpages.

Funded by the Australian Government Department of Education and Training.

4. Counting cookies
   This module introduces variables, a way of storing information for use in a program or webpage. Students learn how to define, modify and use (as well as name!) variables to further improve their webpages. At this point, students have a fully working Cookie Clicker!
5. Extensions: Decorating the cookie
   This module introduces students to some extension tips and tricks for making great interactive webpages. The module introduces both the basics of CSS and some advanced CSS concepts like CSS Transitions, as well as introducing decision making with the if statement in JavaScript. The module also covers generating random colours. Students learn how to integrate these features into their webpages and specifically, into an engaging and even more exciting Cookie Clicker.

Lesson Preparation

Requirements
In order to complete this lesson you need an internet connection and some pencils and graph paper for activity 1, and pencils and paper for activity 2.

Introduction— what is programming?

What does it mean to program a computer?
Programming a computer means giving it instructions - a sequence of steps - to follow. Another word for this is an algorithm.

New Vocabulary

Algorithm: A set of rules or step by step instructions to solve a problem or achieve an objective.
Decomposing: Breaking down a problem into parts to then deal with individually. Imagine making a cake but you only had the final picture. The recipe is the instructions that break the problem up so it is easier to follow.
Function: A group of code instructions that are followed like a recipe. Functions are a good way to split up a program into chunks, each of which can be reused.
Branching: A programming instruction that directs the computer to another part of the program based on the results of a decision or comparison.

Discussion Questions

- How do we make a computer do exactly what we want?
- How do we interact with computers? How do we interact with webpages?
- What makes a website engaging? What makes a website hard to use?

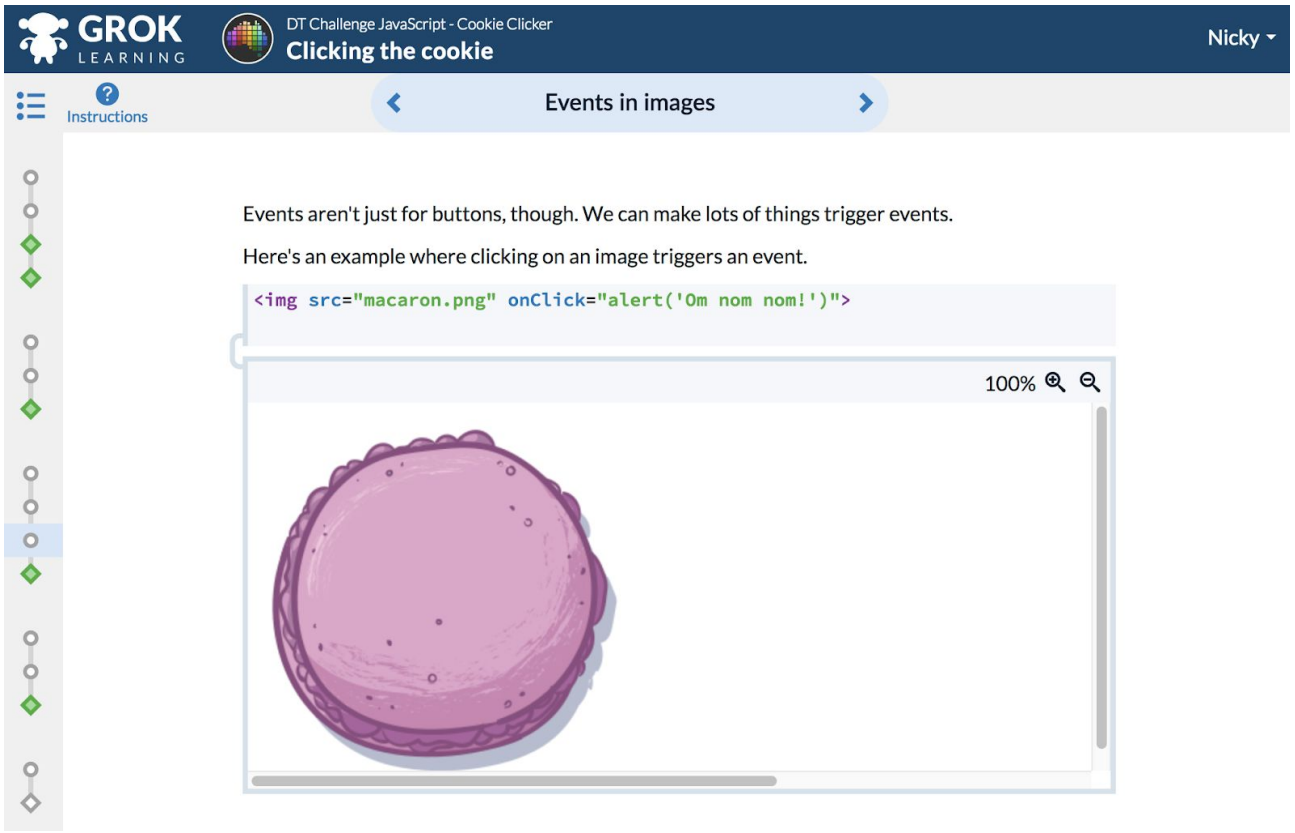
Funded by the Australian Government Department of Education and Training.

Activity 1 - Cookie Clicker

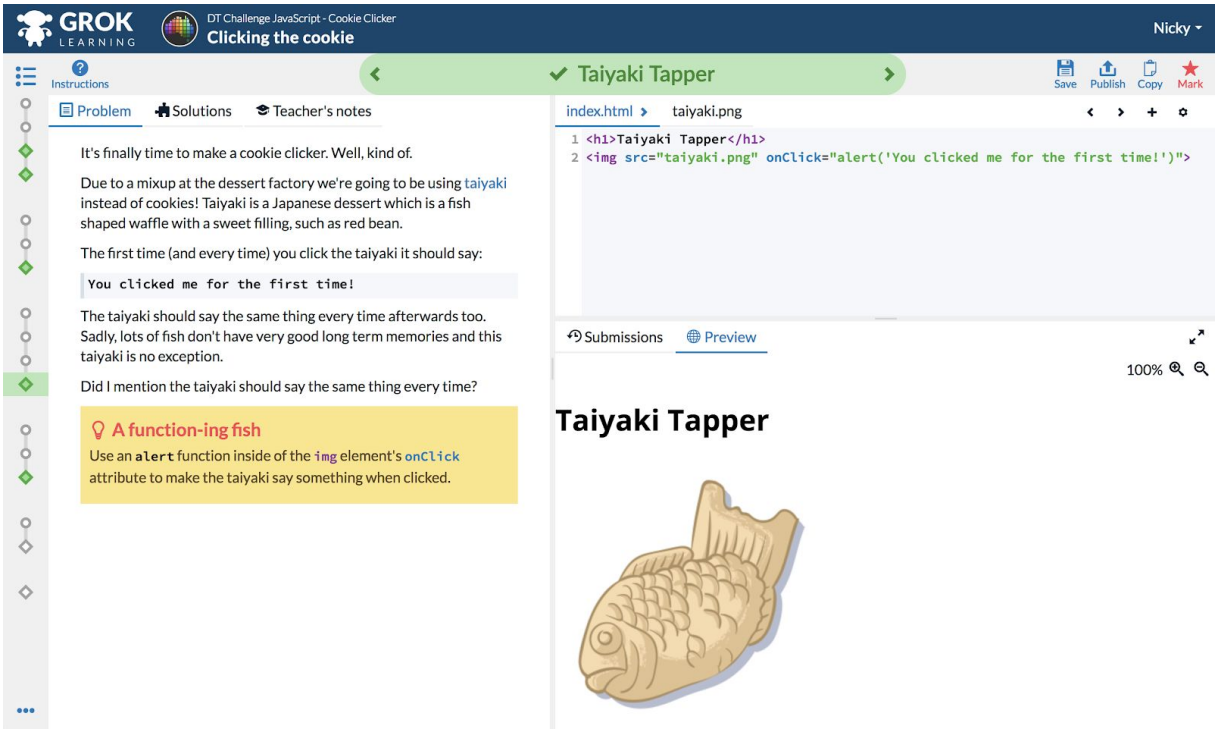
Activity overview

- Students should read interactive notes and complete the coding challenges.
- The students should write their code in JavaScript, referring to the notes as required.
- Click the **Run** button to check your code by running it. Then press the **Mark** button to see if your code is correct and to see feedback for your solution.

Example Notes slide:



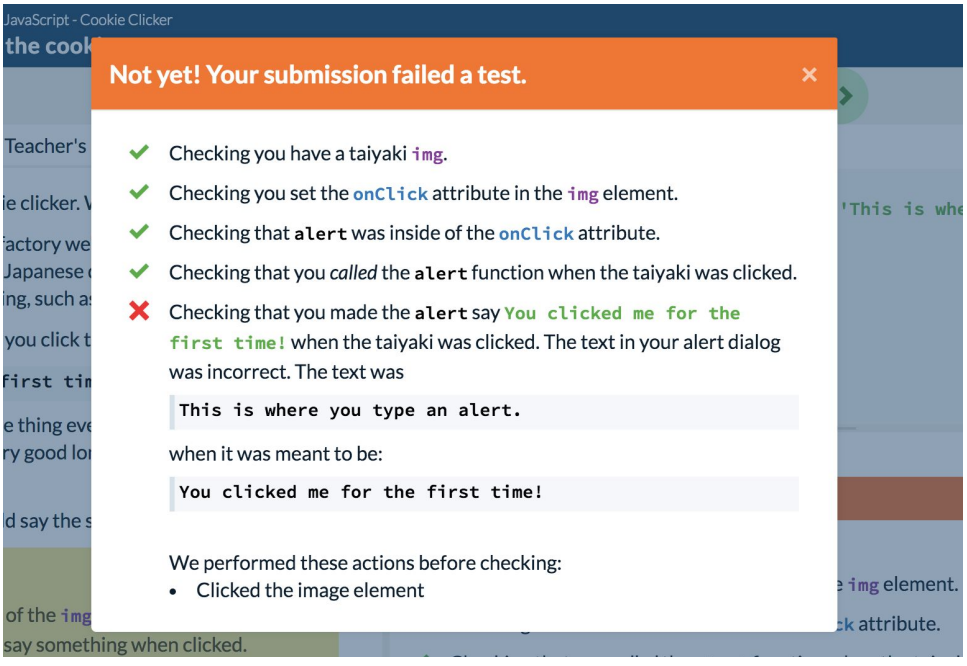
Example Question to be marked:



Students should read the question text on the left, then write their solution in the box on the top right of the page. The code updates the live preview on the bottom right of the page as students type. This lets students immediately see what changes their code has had on the page, and they can test whether they think their solution correctly solves the challenge posed.

Once they are confident in their answer (or if they are stuck and need a hint) they can click ‘Mark’ in the top right corner of the page to submit their code to our automarker for help.

Example of marking feedback:



Activity 2 - User Interface Design

Activity Overview

- Ask students to design an app that tells you the weather for the day, and draw how it would look.
- Ask students to explain how a user would interact with it. What are the aims of these users, what are they trying to achieve and how can they use the app to achieve these aims.
- Have students pair up and “use” their partner’s design, - see if they can understand how it works by, for example, seeing if they could check today’s weather.
- Ask the students how they might change their app if they were designing it for young children, elderly people, or people from a different language background.
- Have students choose one different ‘target audience’ for their app, and redesign (redraw) how the app would look now, highlighting and explaining the main differences.
- Have students pair up again, and see if it was now easier or harder for them to understand how the app worked.

Discussion questions:

- Were there many parts of the interface design that were more complicated to understand than necessary?
- Was it easier or harder to understand the app after it had been redesigned?

Extension ideas:

- Introduce aspects of accessibility. Would students change the app design for a user who was vision impaired?
- Would the ideal design be different for someone who worked at the Bureau of Meteorology?

Activity 3 - Blind Pictures

This activity will demonstrate the importance of precise instructions for the computer to follow when coding or designing an algorithm.

Resources:

Participants will need two pieces of paper and a pencil or pen to draw an object.

Activity objectives

- Follow a simple sequence of steps algorithm carefully to solve a problem
- Experience how difficult it is to describe an algorithm unambiguously and succinctly
- Practice using lengths, angles, proportions and spatial language to describe a design
- Practice communicating and collaborating to solve a problem

Funded by the Australian Government Department of Education and Training.

Instructions

Discuss: In this activity you will use instructions to communicate an image. This is how digital technologies represent scalable images (such as SVGs or EPS) and how printers produce high-quality output.

Instructions:

1. Students should work in pairs: a drawer and an instructor
    a. the drawer from each pair needs close their eyes
    b. the instructor needs to look to the screen for what they must draw
2. Jump to the next slide so only instructors can see the house
3. Instructors should draw their version of the house and hide their own version
4. Cover up the slide again and everyone should open their eyes
5. Instructors give their drawer step-by-step instructions to replicate their image
6. After finishing their drawings, compare the images

Optional: Have partners swap roles, and draw a car, but this time the instruction givers cannot see what is being drawn, and therefore can't give adaptive feedback

Note: Hiding the target image on the screen discourages specific vocabulary, e.g. house, window, door --- after the activity have a discussion about whether using this vocabulary would have helped.

Additional Resources

The Digital Technologies Hub topic User Interface Design provides various additional resources and lesson ideas.

A user interface is a way that people interact with a computer system, including how they exchange information and instructions with it. When developers create information systems, they often design them with the user in mind, taking into account the target audience for their resource or application. In the review process of a new system, users are often asked to test and evaluate the system to help ensure it meets their needs.

See: https://www.digitaltechnologieshub.edu.au/teachers/topics/user-interface

Review:

 **Reflection**

Computers are good at following instructions, but those instructions need to be very accurate! What happens if the instructions aren't quite right?

Loops and decisions are useful to automate tasks. Can you think of a tasks at home or at school that you would want to automate? How would you approach this automation?

Funded by the Australian Government Department of Education and Training.