Cookie Clicker - A workbook for Event Driven programming in JavaScript

Adam Chapman-Dodd and Christopher Dalziel

April 2025

Let's get started:

Cookie Clicker is a web-based incremental video game written in JavaScript by French developer Julien "Orteil" Thiennot with some assistance from their friend Opti. The game has a simple progression loop revolving around the collection and spending of cookies, which serve both as points and as a kind of currency. Cookies can be spent to purchase upgrades which in turn allow for the collection of more cookies.

While on paper this sounds boring and a little pointless, these simple mechanics combine to make for a compelling game. Although in the twelve years since its original release it has received updates with progressively complicated and esoteric mechanics — with many experienced players suggesting that it would take between a 1.5–3 years to complete entirely — it is these simple features which characterise it.

With this workbook, you can create your own simplified version of this cookie clicker game with the core functionalities. You are provided a template of the HTML, CSS and Javascript files you will need to make this however this is just a shell: test, fix and add all the missing features and code to make a working cookie clicker game.

If, at any point, you are unsure on which direction you should be heading in, your teacher has been provided with some hints and ideas to keep you on the right track.

Task Overview:

1. The `Cookie` Button
   (a) Connect the button to the `cookies` value and make it so clicking the button increases the counter.
   (b) Make the cookie pulse when clicked using CSS animations.
2. CPS Tracker
   (a) Track how many cookies were gained in the last second and update the CPS value every second and display it.
3. Upgrades
   (a) Buying *upgrade 1* should increase how many cookies each click adds: but buying it currently does nothing. Fix this!
   (b) Buying *upgrade 2* should generate some cookies over time. Use `setInterval` to add cookies passively.
   (c) Using the logic of the previous upgrade, create a new passive *upgrade 3* with its own prices and scaling.
4. Golden Cookies
   (a) Make a `golden cookie` appear randomly on the screen every so often. When clicked, it gives bonus cookies. Remove the golden cookie after a few seconds if it is not clicked in time.
   (b) The amount of bonus cookies for each golden cookie should be random and the de-spawn time be based on this value.

1 The Cookie Clicker Button

Learn

Event listeners are pieces of code which respond to an HTML event occurring for a given element of the page. To make the cookie *clickable* we want an event listener for the cookie to run the cookie update function.

```
let score = 0;
let clickPower = 1;

// ------- Cookie clicking -------
let cookie = document.getElementById('cookie');
let scoreDisplay = document.getElementById('score');

// Event listener for cookie click
// Increment score on click
function cookieClick() {
    // YOUR CODE HERE
}

cookie.addEventListener(???, cookieClick); // YOUR CODE HERE

let refreshCookieCount = function() {
    scoreDisplay.textContent = score;
};
```

At the moment, the event listener is present for the cookie — but the `cookieClick()` function doesn't do anything and the event listener isn't listening for an event! The player's score should *increase* by the `clickPower` whenever the cookie is clicked!

Do

- ☐ Add the event name to the event listener.
- ☐ Add code to increase the score and call `refreshCookieCount()` when clicked.
- ☐ **Bonus:** Add a CSS animation that makes the cookie "pulse" when clicked.

Discuss

- Why do we separate logic (JS), styling (CSS) and structure (HTML) rather than keep it all in the same file?
- How might you use different types of events (like `mouseenter`, `dblclick` or `keydown`) to change how players increase the score?
- Why is `refreshCookieCount()` called after increasing the score? What would happen if we forgot to call it?
- How could you make this click interaction more engaging or apparent for a user?

2 Cookies Per Second (CPS)

Learn

`setInterval(func, delay)` runs the function `func()` every time the provided `delay` has passed, allowing for periodically repeating functions

To implement the Cookies per Second (CPS) counter we need to work out how many cookies have been clicked in the last second, which we can do by updating the variable `clickedLastSecond` and by using `setInterval`.

```
// ------- Cookies Per Second -------
let clickedLastSecond = 0;

// Function to update the cookies per second display
function updateCPS() {
    let cpsDisplay = document.getElementById('cps');
    // YOUR CODE HERE
    cpsDisplay.textContent = ???;
}

// Update the cookies per second display every second
setInterval(updateCPS, 1000);
```

You should add the logic to this function for updating the cookies per second to represent the number of cookies received in the last second. This will include both user clicks and autoclickers!

Do

- □ Use `setInterval()` to update the CPS display every second.
- □ Make sure that the `clickedLastSecond` counter is reset every second.

Discuss

- Why use `setInterval()` instead of updating CPS on each click? What problems could occur if CPS was updated on every frame?
- Why do we need to reset the counter each second? What happens if you simply don't do this?

3 Upgrades

Learn

To avoid repeating logic, we can abstract upgrade logic into a function: `createUpgrade()` sets up buttons, scaling, prices, and effects.

We can also use `console.log()` to see values in the console while running the game.

```
// A function that says hello in the console when called
function sayHello() {
    console.log("Hello!");
}
```

Do

- □ Complete Upgrade 1 (Click Power):
  - Increases cookies per click.
  - Scales cost over time.
- □ Complete Upgrade 2 (Auto Clicker):
  - Generates cookies passively using `setInterval()`.
- □ Create a new passive Upgrade (3):
  - Use the functions and the idea of upgrade 2 to create another auto clicker upgrade, making sure to add it to the shop (HTML)

Discuss

- Why is it a good idea to have one function like `createUpgrade()` handle upgrade setup?
- What is the issue with upgrade prices not scaling with level?
- How do HTML element IDs help JavaScript interact with the correct parts of a web page? What would happen if IDs are reused?

4 Golden Cookies

Learn

When we want to perform an action based on time instead of any user interactions and only want to do it (once rather than periodically) we look at using timed functions and delays. Take the below which, after 60 seconds (60000ms) calls the `showSecretMessage` function.

```
function showSecretMessage() {
    showMessage("Secret message!");
}

// Show secret message after 60 seconds
setTimeout(showSecretMessage, 60000);
```

Do

- □ Fix `spawnGoldenCookie()`. The intended functionality is:
  - A golden cookie spawns in a random location on the screen after a certain delay. A random number is used for each golden cookie value.
  - On click, add cookie value to score and despawn that golden cookie
  - The cookie despawns after a time based on its value. It then calls the spawning of the next golden cookie.
  - An initial spawn call is necessary to start the cycle.
- □ Add some chips to the golden cookie design using CSS and HTML. Use the `cookie` design as inspiration.

Discuss

- Why do we use `Math.random()` here? How could you skew that to be weighted towards lower numbers?
- Why do we use `setTimeout()` instead of `setInterval()` for controlling how long a golden cookie stays on screen?
- How could we make golden cookies rarer or more powerful? How would you start implementing that?

Extension

Bonus Features

Have a go at adding some more features. Here are some ideas of things you can add:

- Add sound effects
- Add achievements and new upgrade types
- Save game progress using local storage

Further Discussion Prompts

- What part of this project did you find the most difficult? *Is there something you could now add or improve in your game that would show you've overcome that challenge?*
- Which coding techniques did you think is most useful or reusable? *Can you think of ways you might apply them in other projects or programming languages?*
- If you had more time to explore, test and build, what features would you add? *How would you make a start in adding these?*

Completion Checklist

- ☐ Cookie button increases score.
- ☐ CPS tracker works correctly.
- ☐ At least two upgrade types are implemented.
- ☐ Golden cookies appear and give bonus cookies.
- ☐ Code is well-structured and documented with comments.
- ☐ At least one original improvement or feature added.