Project Planning Tutorial

PREPARED FOR

Udacity Students

PREPARED BY

Pascal Meers

Apr 3, 2018

Hello Fellow Students

Recently it came to my and other student leaders attention that some of you struggle to start working on a project.

For this reason I wrote this short tutorial to give a better understanding of how you can fit all the pieces you know together.

For this tutorial I'm gonna take the memory game as a reference on how to structure this project.

This tutorial will not have any code examples since there are many ways to achieve your end goal.

I'm also not gonna explain the whole game, just the basics about how to break it down.

We will start by looking at the project specifications.

I know as a beginner programmer that this can be overwhelming and frustrating to do. But don't let this be an excuse. Everybody can do programming with the right guidance and tools you can do it too.

Project Specifications

When starting any project you start by reading what your end goal is. This means gather all the information about the product you need to create.

See it a bit as a lego game where you get a box with blocks without a drawing of what you need to build but a description. The description says ‘build a house’. Well that's great, but first see what is needed to build a house?

- Does my house need windows? Yes!
- Does my house need an entrance door? Yes!
- Does my house need an indoor swimming pool? Optional!
- Does my house need a bathroom? Yes!

This seems logical right? Well now you need to see if you have the right tools for that.

Before we look at the tools, we are going to look at the matching card project and try to create a list of all the functionalities your game needs to have based on the project requirements.

Down below I gathered all the information from the udacity course:

How The Game Works

The game board consists of sixteen "cards" arranged in a grid. The deck is made up of eight different pairs of cards, each with different symbols on one side. The cards are arranged randomly on the grid with the symbol face down. The gameplay rules are very simple: flip over two hidden cards at a time to locate the ones that match!

Each turn:

- *The player flips one card over to reveal its underlying symbol.*
- *The player then turns over a second card, trying to find the corresponding card with the same symbol.*
- *If the cards match, both cards stay flipped over.*
- *If the cards do not match, both cards are flipped face down.*

The game ends once all cards have been correctly matched.

Game Functionality

The real-life game, players flip over cards to locate the pairs that match. The goal is to recreate this effect in your project. There are a couple of interactions that you'll need to handle:

- *Flipping cards*
- *What happens when cards match*
- *What happens when cards do not match*
- *When the game finishes*

The above part is an overview of the project. Write down some of the parts from this overview. Try to see everything as a lego block.

Below I'll make an example list what I would look for in this overview:

- ***A grid with 16 cards***
- ***8 different pairs of cards***
- ***Cards are randomly placed***
- ***The symbols face down***
- ***On a click the card must turn and show the icon and stays turned***
- ***On a click on a different card the card must turn and show the icon***
- ***If the 2 turned cards match they stay turned***
- ***If the cards don't match they turn back***
- ***The game ends when all cards are flipped***

These steps are all found in the project overview. Udacity also provides a nice project rubric with specifications for the project. Here you can gather even more information and break down your project even in more bits and pieces.
Let's have a look at these specifications first.

Criteria	*Meets Specifications*
Memory Game Logic	The game randomly shuffles the cards. A user wins once all cards have successfully been matched.
Congratulations Popup	When a user wins the game, a modal appears to congratulate the player and ask if they want to play again. It should also tell the user how much time it took to win the game, and what the star rating was.
Restart Button	A restart button allows the player to reset the game board, the timer, and the star rating.
Star Rating	The game displays a star rating (from 1-3) that reflects the player's performance. At the beginning of a game, it should display 3 stars. After some number of moves, it should change to a 2 star rating. After a few more moves, it should change to a 1 star rating.The number of moves needed to change the rating is up to you, but it should happen at *some* point.
Timer	When the player starts a game, a displayed timer should also start. Once the player wins the game, the timer stops.
Move Counter	Game displays the current number of moves a user has made.

With this new information my list would look like this:

- ***A grid with 16 cards***
- ***8 different pairs of cards***
- ***Cards are randomly placed***
- ***The symbols face down***
- ***On a click the card must turn and show the icon and stays turned***
- ***On a click on a different card the card must turn and show the icon***
- ***If the 2 turned cards match they stay turned***
- ***If the cards don't match they turn back***
- ***The game ends when all cards are flipped***
- ***When the game ends show a modal***
- ***Reset button***
- ***Star rating***
- ***Timer***
- ***Move counter***

See how I narrowed down all the information provided by Udacity to a small list.

Now that you know the functionality of your game you can start looking in your toolbox. So lets make a few examples here. You want to keep your functions as small as possible and it is even better if your function does only one thing. By writing down how you could solve a problem you get a better understanding of the whole project in the end.

I'm gonna start with the grid here. Why? Because all the other functionalities depend on the grid. Without a grid you can not have a card to click on.

Toolbox - how to solve your problem

After creating a nice list with the core requirements, You can start with creating solutions. I made a few examples below where I write down how I would like to handle a certain problem. You can use Google if your not sure how to solve a problem. Just Google it and write down what you found.

A grid with 16 cards

You previously learned about array's these are great to store data like the card symbols.

8 different pairs of cards

Google searches

- https://stackoverflow.com/a/11128791/8498100 - create an ul from an array.

Tip: Don't just copy and paste the outcome of your Google searches. Try to understand what they do and use those same tools to solve your problem.

Your just created array must hold pairs. Meaning that you have to add your icon / images double to the array. This is the easiest way. You now created your array with all the cards. Now the question is how can I show them on the html.
For this you have a few tools, just to list a few:

- Document - this is your html.
- createElement - with this you can create an element like an or .
- getElementById - with this you can get an element by it's id. How? Say you added id="deck" to your html you could now get this element in your JavaScript by using getElementById.
- querySelector - this works roughly the same as getElementById.
- A loop like for of - to loop over each element in the array.
- appendChild - to append your elements to your deck

Before you move on to the next element try to make this work. Don't work on all the part's at the same time because this is going to make it confusing and you lose track of what you were doing.

Cards are randomly placed

Now the shuffle part. This function is already provided by Udacity, so make use of it. Try to understand what this function does and what it returns.
In this case it returns your recently created cards array shuffled. You can store this in a new array for later reference. A good name for this new array would be shuffledArray. You now have a freshly shuffled array. To check if it works, you can console.log it and refresh the page a few times. when you know it works it's time to connect this functionality to your front-end by replacing simply the cards array you loop over to display your shuffledArray cards now.

The symbols face down

For this you can use css with opacity or transform.

On a click the card must turn and show the icon and stays turned

You now should have the cards turned and shuffled. By doing this step by step you have an overview.
This part can be broken up in pieces too:

- You need a list of your cards.
- You need to listen to user events, so you are going to have to use an eventListener.
- You need a new css class that gets added after the click and shows the icon
- You don't want the user to click the same card again.
- You want to store your open card temporary somewhere to check later if it's a match with the next clicked card.

On a small project like this, this all might seem a lot of work for a game but it really makes your life easier if think about your project first. Don't start coding immediately. First go over the project specifications and start stripping them down to small pieces. Some pieces can be even stripped down into even smaller pieces. Don't try to solve the whole picture at once and write down your steps. You could make a nice table like the example below.

Step	Description	Done
1.0.1	Create an array	no
1.1.1	Create a reference to the deck	no
1.2.1	Create a list and append it to the deck	no
1.2.2	Add a class to the list elements	no
1.3.1	Create an icon element	no
1.3.2	Add the class to the icon element from the array	no

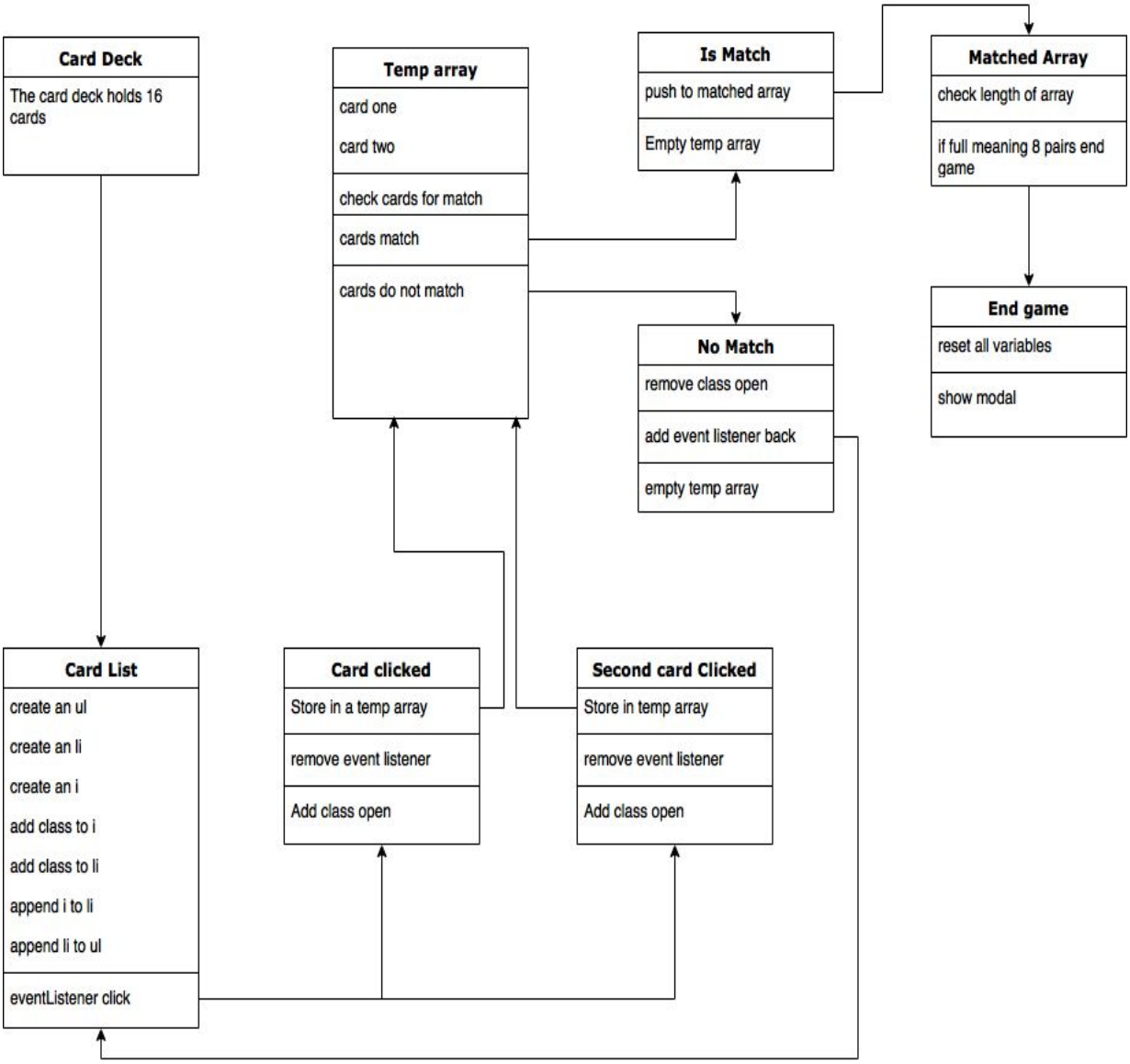
After writing down all the functionalities and how you would tackle the problem, you can make a nice table like the one above.
For me this works great. In the beginning you will struggle a bit with figuring out what to use for what. To solve this I suggest you google the problem.
Example: how to shuffle an array Javascript -jquery

Why -jquery? This will tell google to filter out all jquery related answers and focus the search on JavaScript.

If you need more information about this use the handbook created by @asher https://sites.google.com/knowlabs.com/gdnd2017/faqs/how-to-find-answers as a reference.
This guide can be used on any kind of project. It may sound a bit redundant to produce a document like this, but keep in mind the document can contain pure keywords. I wrote a lot to explain certain things, but the basics should be:

- A list of project specifications
- A list with tools you can use - if you used google to look up certain things write them down.
- And a sort of checklist
- A flowchart - This is pure optional, but can help you to put all the pieces together.

And one more time break your project down in pieces. And build it block by block.
I'm not great at making diagrams and flowcharts but below a little example of how i would make one for my project.


Card Deck
The card deck holds 16 cards
Temp array
card one
card two
check cards for match
cards match
cards do not match
Is Match
push to matched array
Empty temp array
Matched Array
check length of array
if full meaning 8 pairs end game
End game
reset all variables
show modal
No Match
remove class open
add event listener back
empty temp array
Card List
create an ul
create an li
create an i
add class to i
add class to li
append i to li
append li to ul
eventListener click
Card clicked
Store in a temp array
remove event listener
Add class open
Second card Clicked
Store in temp array
remove event listener
Add class open