Introduction to web development

By Alex Robic

Main objective of this course

- Make YOUR website about things YOU like
  - Don’t make it too simple but also don’t over complicate it
- Ex: I like Pokemon so my site is all about pokemon
- **The real objective is to give you enough basics for you to make a simple page, if you want to make it more complicated, I can give you references**
- How I like to structure my courses:
  - Tutorial part (1h-1h30)
  - Independent work (30min-1h) → during this time, you can consult me for help (debugging) while you work on your website

By the end of the week you should have a simple website that you can then edit **as you wish**

Legal disclaimer: Alex does not condone nor condemn the content on your website. You have total freedom to put whatever you want but do keep in mind that your parents may want to look at it

Class rules

Important note: demonstrations are important, not everything is easily understandable in theory so sometimes, it's best to see it live and understand why it's doing such a thing

- Name format:
  - Instructor: Alex Robic
  - Student:
    - Alex R
    - Alex
- Chat use: unlimited but DO NOT:
  - Share contact info (phone #, email, social media etc)

  You can share your website links with others (after all you *own* it)
- **FEEL FREE TO ASK AS MANY QUESTIONS AS YOU WANT AND/OR REQUEST A DEMONSTRATION (this kind of stuff is not always easy to understand and some of you are more visual learners like me)**

Good things to have

- A web browser (I think everyone has one) → software that lets you explore the web
- Visual studio code → free software that lets you edit (with colors ☺)
- A github account (that's how you'll be hosting your website)

By the end of the course…

You should be able to:

1. Place elements on a webpage using HTML (hypertext markup language)

2. Style these elements (placement, alignment, color, border etc)

3. Add interaction with these elements using javascript (+ use a programming language as opposed to a scripting one)

Objective 1: build a simple web page

- Build a simple web page with HTML
  - Header types → (<h> and <p>)
  - Title
  - Page structure
  - Div / span
  - Links
  - Images
  - Videos
  - Iframe
  - Br / hr
  - Table / tr / td
  - Ol / ul
  - Favicon

Objective 2: give it style with CSS

- Give it style with CSS
  - Color
  - Font
  - Placement
  - Highlight
  - Decoration
  - Float
  - Border / background
  - Style markers VS external file

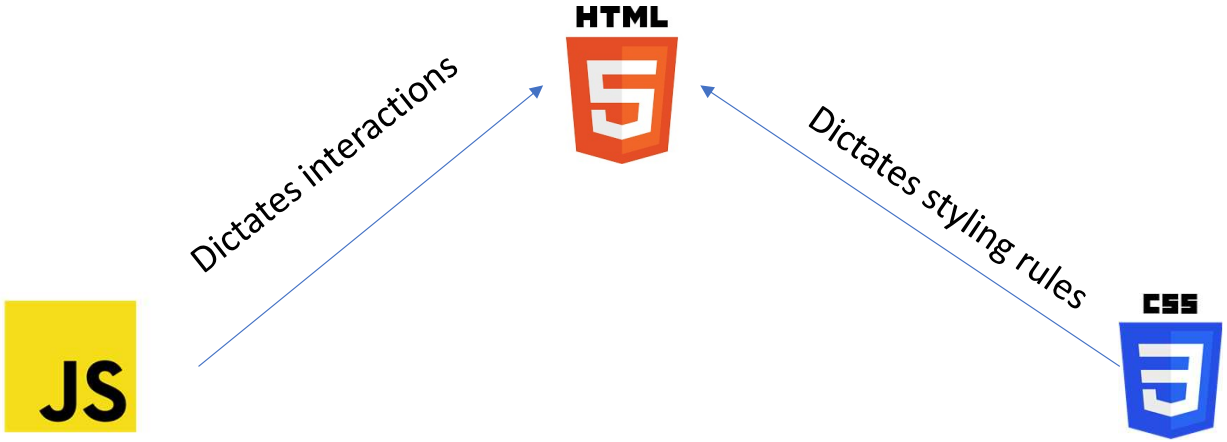
Objective 3: make it interactive with javascript

- Click, double click, drag and mouse over
- Alert and confirm
- Variables, loops etc
- Animations
- Keyboard events
- Mouse events

MONDAY

Objective 1: a simple page with a few things on it

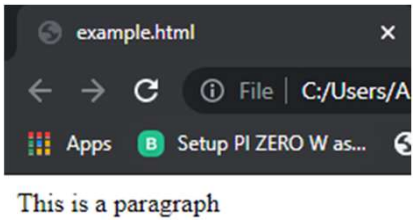
- File type: .html → HTML = HyperText Markup Language
  - Other file extensions exist (.php, .xml) but those will not be covered in this course



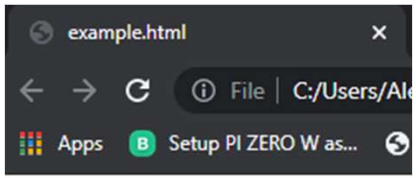
Objective 1: a simple page with a few things on it

- Ex:

```
1  <html>
2      <link rel="stylesheet" href="style.css">
3      <script type="text/javascript" src="script.js"></script>
4      <p id="paragraph" onclick="disappear(this)"> This is a paragraph </p>
5  </html>
```



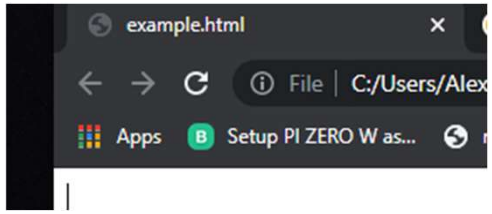
- HTML: This is a paragraph →
- CSS: p { color:red; } →
- Javascript:




```
1  function disappear (x) {
2      x.style.display = "none";
3  }
4
5  var p = document.getElementById("paragraph");
```

AR1

The code



After clicking on the paragraph

Slide 11

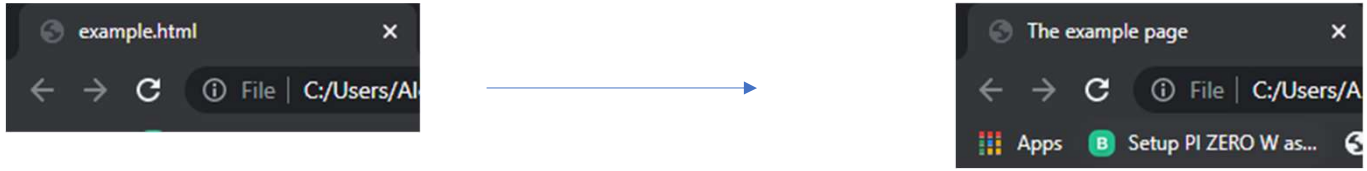
AR1 Alexandre Robic, 6/20/2021

Objective 1: a simple page with a few things on it

- Today: HTML
- **What is written first, gets loaded first**
- Rule: in most cases, if you open a tag, you have to close it
  - Ex: opens a paragraph and closes it
  - Exceptions: <img>
- See it as a big sandwich but you have to have 2 of each thing (hope that makes sense)
- Your document starts with a <html> tag so which should it end with?
- Page structured like a human (header, body, footer)

Objective 1: a simple page with a few things on it

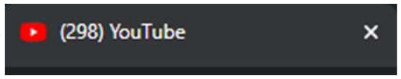
- Title: by default uses the file name (here, example.html)




```
<html>
    <header>
    <link rel="stylesheet" href="style.css">
    <script type="text/javascript" src="script.js"></script>
    <title> The example page </title>
    </header>
    <body>

    </body>
    <footer>

    </footer>

</html>
```

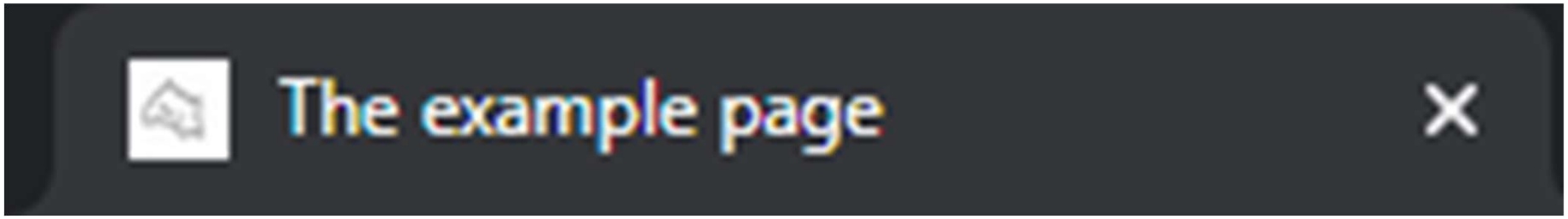
<title> Your page title here </title>
→ **Put this in your header**

Objective 1: a simple page with a few things on it

- Favicon: the little icon next to the page title
- → 

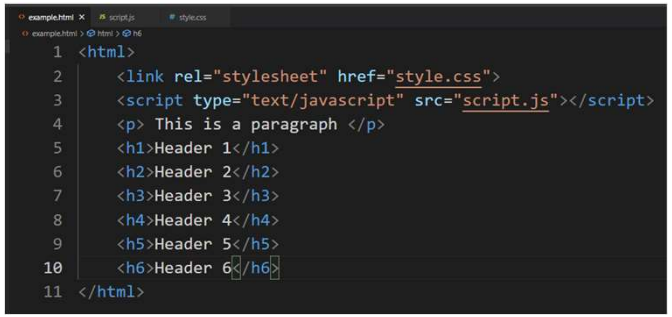
- You can set your own!
- <link rel="shortcut icon" href="bongo.jpg" type="image/jpg">

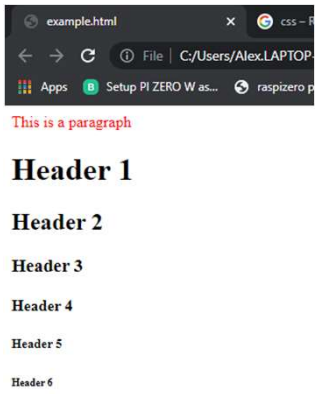
Opens a link tag | Define the type of link | Give it the link to your image | In this case, it's a good thing to mention the type of image



Objective 1: a simple page with a few things on it

- Header types:
- <h> tags are used for titles and <p> for paragraphs
- <h> tags can be accompanied by a number to give a certain order of importance
  - Ex: <h1> is your most important (page title perhaps) and <h6> is your least important (end credits?)





Objective 1: a simple page with a few things on it

- Images: One of the only tags that does not have a closing tag
- <img src='path/to/image' alt='alt text' width='chosen width' height='chosen height'>
- Height and width can be written in px or % (although other units exist, these are the 2 you will most frequently find, pt is another story)
  - % is % of window
  - Px is fixed (when you change the size of the window, you DO NOT change the size of the image)
- Demonstration with pikachu image

Objective 1: a simple page with a few things on it

-
 and <hr>
-
 used to break line (like pressing enter when you're typing)
  - Use case: You have content you want on the next line or you want to space content out
- <hr> used to draw a horizontal line on page (useful to separate content and can be customized)

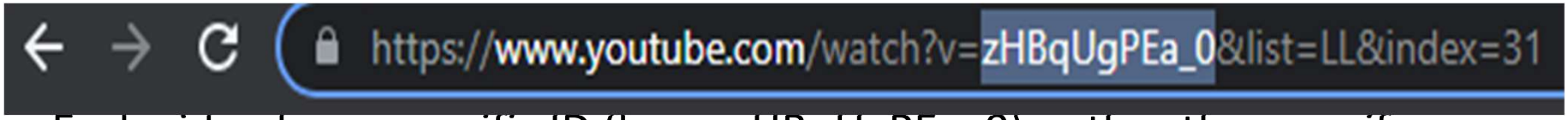
Objective 1: a simple page with a few things on it

```
<video width="50%" controls muted>
    <source src="Microwave Sugar Cookie _ Em's Kitchen.mp4" type="video/mp4">
    Your web browser does not support videos
</video>
```

- Video tag (closed at the end)
- Source (you can put several and if the first one does not work or is not supported, it will switch to the second one)
- The text at the bottom is displayed if none of the sources work (like if for some reason your browser does not support video playback)
- Controls, controls muted, autoplay muted (chrome does not allow autoplay)
- It's good to set a width (and/or height) to avoid flickering during page load (and avoid too large of a video)
- Poster → thumbnail

Objective 1: a simple page with a few things on it

- Youtube: the exception + iframes
- Iframe: a special tag to see a window INSIDE another window



- Each video has a specific ID (here, zHBqUgPEa_0), rather than use iframe on whole link (and get video suggestions, ads on side etc), use embedded youtube (youtube.com/embed/{video id})
- Ex:
- https:/www.youtube.com/embed/zHBqUgPEa_0 as opposed to
- https:/www.youtube.com/watch?v=zHBqUgPEa_0

Objective 1: a simple page with a few things on it

- Talking about getting stuff from other websites…
- Links or the secret of the internet's success
-

<!—Whatever you put in here, becomes what is clicked on to access your site -->

Ex:

<img src="bongo.jpg" width="25%">

This places the bongo cat image on screen and if you click it, takes you to a youtube video

Objective 1: a simple page with a few things on it

- Tables (starts with a <table> tag) (see it as a wall of boxes, each table is a wall of shelves, each tr is a shelf and each td is a box)
- <tr> → opens a new row
- <td> → opens a new data entry (or column)
- (stuff in tables doesn't have to be numbers btw)
- Ex:

<table>

<tr> <td> A </td> <td> L </td> </tr>

<tr> <td> F </td> <td> H </td> </tr>

</table>

1	**2**
3	**4**

Fill in the table

Objective 1: a simple page with a few things on it

- Lists:
- Unordered (ul) → by default, displayed with disks
- Ordered (ol) → by default, displayed with numbers (Arabic)
- List item (li) → an item in the list (by default, goes to next line at the end)

```
<ol>
<li>Tomatoes</li>
<li>Bread</li>
<li>Cheese</li>
</ol>
```

What will it look like?

Objective 1: a simple page with a few things on it

- Div / span → useful for styling (tomorrow's topic)
- Div: a certain division of the page
- Span: a certain portion of text
- We'll look at this more tomorrow when we do styling

Objective 1: a simple page with a few things on it

- A few extra tags to start to add a bit of style:
- <u>Text</u> → <u>Text</u>
- Text → *Text*
- Text → **Text**
-   → puts content in center of screen
- 1st → 1^{st}
- C₁₂H₂₂O₁₁ → $C_{12}H_{22}O_{11}$

(For those of you wondering, it's the chemical formula for sugar, like your everyday table sugar)

Objective 1: a simple page with a few things on it

- Get to work!!!!!!
- Your task: put everything (or almost everything) you want on your website on the page, we'll worry about style tomorrow (+ build the other pages to your website)
- Feel free to consult me (just give me a few minutes to get coffee while you work)
- You have total design freedom (just do note that the staff or your parents may want to look at it)
- When you're done if you don't already have a github account, create one, if you do, create a repository called *username.github.io* and upload your page(s) **with the main one being index.html**

Tuesday

Objective 2: give it style with CSS

- CSS: Cascade StyleSheet → the last styling option is the one kept
- → if you put p {color:red;} then body {color:blue; font-style:italic;}

And you have:

<body>

 Paragraph

</body>

In your html: what will you get?
Paragraph or *Paragraph?*

Objective 2: give it style with CSS

- 3 ways to include style:
  - Either between style tags → <style></style>
  - In an external file (my preferred method):
    - <link rel="stylesheet" href="style.css">
    - This goes in the head
  - Style each item individually:
    - Paragraph

Objective 2: give it style with CSS

- Generally:

type {

attribute-to-change: value;
}
Ex:

h1 { → Changes EVERY h1

text-align:center; → Sets the alignment to center (like tags) of division it is in

}

Objective 2: give it style with CSS

- This poses a problem: what if I wanted to change 1 h1 but not another? (or a p, div, img etc)
- Solution: classes and ids
- Difference: class should be used for several objects (see it as a category) and id should be used for a single object (will be seen again when doing javascript)

Objective 2: give it style with CSS

- When styling IDs:

#id-name {attributes…}

- When styling classes:

.class-name {attributes…}

Ex: see example page

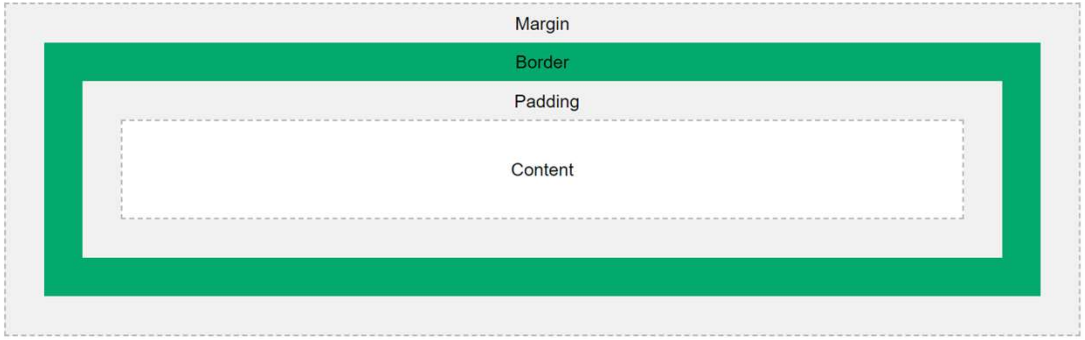
Objective 2: give it style with CSS

- Selectors:

tag.class → selects all tags that have that specific class

tag1, tag2, tag3… → apply this to all of tag1 and tag2 and tag3

Objective 2:



- A few widely used attributes:

border: [size] [color] [style]

Ex: border: 2px outset lightcoral (see example page h1)

padding, margin: [size]

padding is space between content and border, margin is space between border and other item

background-color: [color]

background-image: [see example page]

Objective 2:

- Color: [font-color]
- Opacity: [opacity percentage, 100% is all normal, 0% is transparent]
- Font-style: [italic, oblique, normal, inherit]
- Text-decoration: [dashed, dotted, double, line-through, none, overline]
- ::selection {attributes} → apply style to text when it's selected
- Item:hover {attributes} → apply style to item when the mouse is over it (similar types of selectors include ::first-letter, :link, :active, :visited, :first-line etc)
- Display: {attributes} → display an inline as a block or vice versa, you can use none to not show and object (make it disappear)

Objective 2:

- Setting 2 (or more) items next to each other (the pikachu images)
- Best solution:
  - Create a div to store everything (don't need to give it a border or background color)
  - Set both items with a combined width + margin + padding + borders < 100%
  - Give them each a class
  - In CSS (tags or external stylesheet): set float to either right or left
  - Give the div an overflow: auto or hidden

  Float: left [or right] → send the next item to the left [or right]

Wednesday

Objective 3: make it interactive with Javascript (should take more than one class)

- 2 ways to include javascript code into your web page:

<script type="text/javascript" src="script.js"></script> → gets code from extern script called script .js

<script>Your code here </script> → put your code between 2 script tags

Objective 3: make it interactive with Javascript

Refresher from Python course + intro to syntax:

Variable → var a; OR var a = {value};

Array → var array = new Array(); (creates an empty array)

OR var array = [1,2,3,4,5]; (like python)

For(var i = 0; i < 10; i++) {

…

}

While(cond) {}

if(cond) else if(cond2) {} else {}

NEVER FORGET ;

Objective 3: make it interactive with Javascript

Functions exclusive to javascript:

-console.log() equivalent to Python's print

(note to self, don't forget to show them the console)

-alert(message) → alerts the user (displays a window with your message and an OK button)

-confirm(message) → if the user clicks "OK", the function returns True, otherwise returns False (a good way to get users to confirm an action) Bulbusaur example

-prompt(question, placeholder) → creates a prompt window, puts the placeholder text in the input bar and once the user clicks OK, returns the value in the input bar (Squirtle example)

Objective 3: make it interactive with Javascript

Get an object to interact with it:

var element = document.getElementById(id); → collects the element from the page with the ID id

Change its style:

element.style.border = “2px red solid”; → see example

Writing to page: document.write(message)

This, combined with events allows for interaction with your page

(memory break? → show them project from college web dev class)

Objective 3: make it interactive with Javascript

Events: when something happens (for example, you clicked, doubleclicked, mouseover, pressed a key, released a key)

Mouse events: tied to an object

Keyboard events: tied to the page

Objective 3: make it interactive with Javascript

Mouse events:

Onclick, ondoubleclick, onmouseover etc

Best way to integrate it is through tag declaration:

1. Declare a function in your javascript script (the external one)
2. In your tag declaration: onclick="function_name()"

   If you want to use the function on your current element, add "this" between the () → function_name(this) (Give more details + demo)

   this → function only executed on element it is in

Objective 3: make it interactive with Javascript

Get to work:

Today, start adding more things to your website, make things interactive

Tomorrow, we'll look at the canvas, keyboard events and animations

Thursday

Today: more javascript + working on your websites

Canvas

Keyboard events

Animations + timed events

Canvas: drawing with javascript

Involves using a certain tag (canvas) to create a blank piece of paper to draw on

```
<canvas id="mycanvas" height="500" width="1000" styl
e="border:2px green solid"></canvas>
```

Canvas: drawing with javascript

To start drawing:

Get canvas as element

```
var canvas = document.getElementById("mycanvas");
```

And since we're drawing in 2d, set the context to 2d

```
var context = canvas.getContext("2d");
```

We're ready to draw!

Canvas: let's draw

Note: the image isn't displayed until `context.stroke();` is called

(imagine drawing it on a separate piece of paper and stamping it onto the canvas after)

Clearing the screen:

This can be accomplished by drawing a white rectangle (or whatever background color you chose) with the width and height of your canvas:

```
function clear() {
    context.fillStyle = "white";
    context.fillRect(0,0,1000,1000);
}
//I recommend creating a clear function
```

Canvas: let's draw

Images: context.drawImg(img, x, y);

Img: the image, video to use

X: x coordinate

Y: y coordinate

Circles and arcs:

Context.arc(radius, x, y, beginning angle, end angle);

→ Both angles should be specified in radians (special unit used in trigonometry as opposed to angles, use the following code for the conversion)

```
function radian(a) {
    var rad = Math.PI/180;
    return rad * a;
}
```

Canvas: let's write

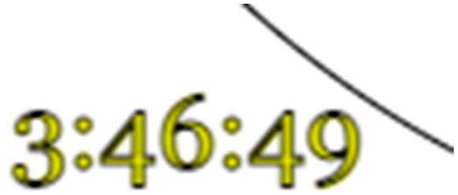
Text is like shapes: set the font attribute then stroke the text onto the canvas:

```
context.fillText("Clock", 25, 25);
context.strokeText("Clock", 25, 25);
```

fillText draws a filled text

strokeText draws just the outline of the letters

Combining both (along with setting the fillStyle and strokeStyle) can give the following →



Which strokeStyle and fillStyle was used?

Keyboard events: getting key presses

Best way to do it and how it's done in the gaming industry:

Event listener

Basically, listen for all events and associate functions to certain events

Ex: the pokeball animation

Keyboard events: getting key presses

Keys are associated to keycodes (numbers)

→Down arrow = 37, up arrow = 39, space = 32 etc

→Full list at http://gcctech.org/csc/javascript/javascript_keycodes.htm

Key code can be captured either on key:

Up (After the key has been pressed, the event is triggered)

Down (event is triggered when the key is down)

The pokeball code: animation through example

Let's have a quick look at it:

1st part: the keyCodes (how to control the animation, speed it up, slow it down etc)

2nd part: moving the pokeball

3rd part: making sure it's repeated (setInterval VS setTimeout)

Bonus content: the sound player

Look at the pikachu code:

```
var pikasound = new Audio("pikachu\ sounds.mp3");
//Create a new Audio object (with the source)
pikasound.play();
//play
//There's also a pause function and a paused
attribute (bool)
```

Rest of the session + tomorrow

For the rest of the session + for tomorrow's session, you're tasked with working on your website. Add animations, games etc.

MAKE IT YOUR OWN!!!!!

(note to self, if not already done and if you have enough time, show the students how to host the website)

Fun games:

Memory (show them the example you made for web dev class)

Tic tac toe

Snake

Friday

Today: no meme on the front page (sorry)

-Work on your website

-Show it off

-Let me know if you have questions

-We could possibly do a show and tell at the end of the class…

Think it's done? Try adding games like memory, tic tac toe and other easy games or even a clock (use the canvas)

Useful links:

https://www.w3schools.com/tags/default.asp

https://www.w3schools.com/cssref/default.asp

https://www.w3schools.com/jsref/default.asp