Getting Stuff Done

digIT web workshop - winter 2019

digit-winter.neocities.org

Here’s a bunny

(I know you were all waiting…)

Today’s workshop

- How to use CodePen
- HTML inputs
- JS basics & syntax
- Revision - the DOM tree
- Using JS to manipulate the DOM tree
- JS exercise OR design exercise (handout)
- Really cool HTML/CSS/JS demos

CodePen

CodePen

- Frontend code playground
- Free to use! (You don’t even have to sign up.)
- Write HTML, CSS and JavaScript in one page in your browser and preview the results
- View and “fork” other people’s pens to figure out how they work and remix them
- (A great way to experiment with different libraries and pre-processors without having to install anything yourself!)
- Built-in JavaScript console where you can try lines of code and see what they do

HTML Inputs

The Input Element

- Just another HTML tag!
- Traditionally used as part of a form, but you can just whack them randomly on a page, too.
- Lets you take user input on a web page
- P.S. You can use CSS to style inputs in cool ways, too

Text input

- The default input type: if you don’t specify another type, the browser will assume one of these
- `<input type="text">`
- `<input type="text" value="default value">`
- A single-line text input field

Checkbox

- A checkbox that can be ticked or unticked
- `<input type="checkbox">`
- `<input type="checkbox" checked>`
- You can check its `checked` attribute to work out if it’s ticked

Other types of `inputs`

- Radio buttons, password fields, buttons, hidden fields etc…
- We won’t be needing these today.
- See the MDN documentation for full details
- Also see `select`, `textarea`, etc - these are yet more types of inputs that don’t fall under the `input` tag, but we won’t be using these today either

Buttons

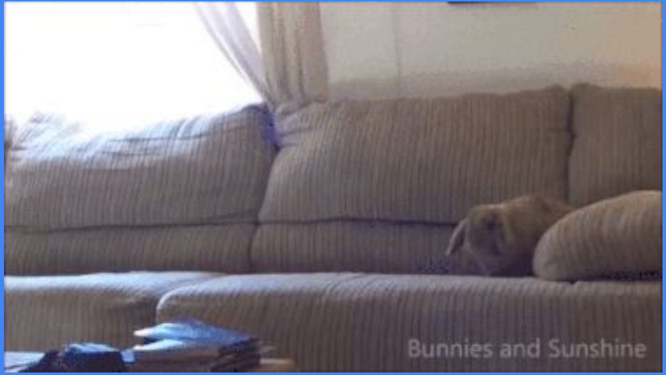
- A clickable button
- `<button>Button Text</button>`
- You need to tell it what you want it to do when you click it
- `<button onClick="doSomething()">Button Text</button>`

Labels

- A caption for an input field
- `<label>Label text</label>`
- You can associate the label with an input field that has an `id`

-
  ```
  <input type="checkbox" id="bun">
  <label for="bun">Tick for bunnies</label>
  ```

JavaScript Crash Course



What is JavaScript?

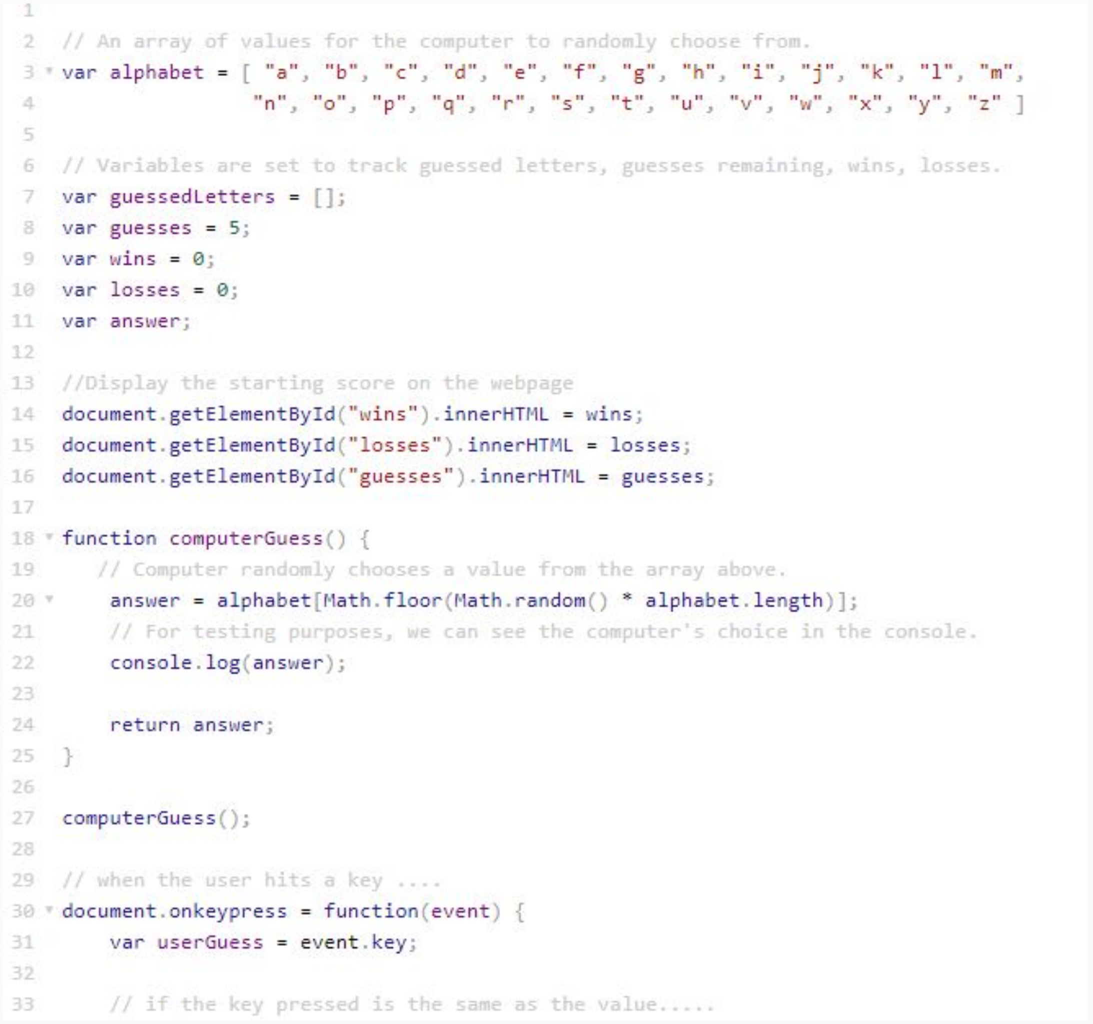
- A programming language that runs in your browser
- Powers the interactivity on most websites you see today
- Can be used for tiny things on a page right up to huge web-apps
- Since JS is usually used in a browser, a lot of code manipulates or interacts with elements on a web page, or the browser itself
- (... but you can also use JavaScript for backend programming!)

Programming in a different language

- Don’t be scared!
- Break down problems in the same way
- The same kind of constructs you’re used to from Python are available (eg. variables, functions, if/else, loops, …)
- The main thing that’s different is the *syntax*
- Python uses whitespace to divide up parts of a program. JavaScript uses punctuation

Syntax

- A JavaScript program looks a bit different from a Python program: it’s full of punctuation.
- Every statement ends with a semicolon.
- Variables need to be ***declared*** before use.
- Blocks are indicated with ***curly brackets***.



```javascript

// An array of values for the computer to randomly choose from.
var alphabet = [ "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m",
                 "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z" ]

// Variables are set to track guessed letters, guesses remaining, wins, losses.
var guessedLetters = [];
var guesses = 5;
var wins = 0;
var losses = 0;
var answer;

//Display the starting score on the webpage
document.getElementById("wins").innerHTML = wins;
document.getElementById("losses").innerHTML = losses;
document.getElementById("guesses").innerHTML = guesses;

function computerGuess() {
   // Computer randomly chooses a value from the array above.
    answer = alphabet[Math.floor(Math.random() * alphabet.length)];
    // For testing purposes, we can see the computer's choice in the console.
    console.log(answer);

    return answer;
}

computerGuess();

// when the user hits a key ....
document.onkeypress = function(event) {
    var userGuess = event.key;

    // if the key pressed is the same as the value.....
```

Variables in JavaScript

- Unlike Python, you need to *declare* a variable in JS before you use it. If you try to refer to a variable before it's been declared, your code will error and be sad. You use the `let` keyword to declare a variable.
- *Declare*: tell your computer you want this variable to exist and you want to be able to refer to it. *Define:* give the variable a value.
- You can declare and define a variable at the same time, or declare it first and define it later.
- `let x = 0;`
- `let y;`

Constants in JavaScript

- Constants are just like variables, except their values don't change.
- You use the keyword `const` (instead of `let`) to define a constant.
- If in doubt, you can just use `let` all the time. The only difference is that the value is allowed to change when you use `let` - you don't *have* to change it!
- (You can also use `var` in place of `let` - you'll see a lot of code that uses `var`, and in older versions of JavaScript it was the only way to declare variables and constants - `let` and `const` didn't exist. It's better code style to use `let`, since `var` has some weird quirks, but it doesn't *really* matter.)

Conditional statements

Python:

```python
if x > 0:
    print("hello")
elif x < 0:
    print("goodbye")
else:
    print("rabbit")
```

JavaScript:

```javascript
if (x > 0) {
    console.log("hello");
} else if (x < 0) {
    console.log("goodbye");
} else {
    console.log("rabbit");
}
```

Functions

Python:

```python
def foo(bar):
    x = 5
    x = x + 1
    return x
```

JavaScript:

```javascript
function foo(bar) {
    let x = 5;
    x = x + 1;
    return x;
}
```

We'll get to some more JavaScript in a minute.

The DOM Tree

Recall...

- Your browser parses HTML tags into *elements* and turns these into *nodes* of a document *tree*. (The handout uses these a bit interchangeably sometimes. Sorry.)
- Everything you see on a web page has its own place in the tree. So do some things that might not be visible when looking at the page in a browser.
- Nodes have *parents* and *children* depending on where they are in the tree.
- You can draw a picture of the tree from the HTML to help you visualise it.

Manipulating the DOM tree

- We can use JavaScript to…
  - **Add** nodes to the DOM tree
  - **Remove** nodes from the DOM tree
  - **Move** nodes around in the DOM tree
- But first we need to know how to get a reference to a node so we can manipulate it!
- **Important:** in JavaScript, `document` refers to the *root node* of the DOM tree. You can work downwards from here.

Getting an element on a page

- The function `getElementById` lets you **search** inside a document for an element that has the given `id` attribute. You shouldn't have more than one element with the same `id` attribute in a page.
- How to call the function: `document.getElementById("myid")`
- You can also find elements by tag name, CSS class name, or CSS selectors (but we won't be using that today)

Creating new nodes

- The function `createElement` creates a new element for your page. The new element isn't in the DOM tree until you put it somewhere, though.
- How to call the function: `document.createElement("tag")`
- How to attach the new element to the DOM tree:
  - Create the new element and give it a name:
    `let newNode = document.createElement("li")`
  - Find the node you want to attach the new element as a **child** of:
    `let parentNode = document.getElementById("myList")`
  - *Append* the new node to the **parent's** list of children:
    `parentNode.appendChild(newNode)`

Removing nodes

- `someNode.remove()`
- Rips it out of the DOM tree. Goodbye, node.

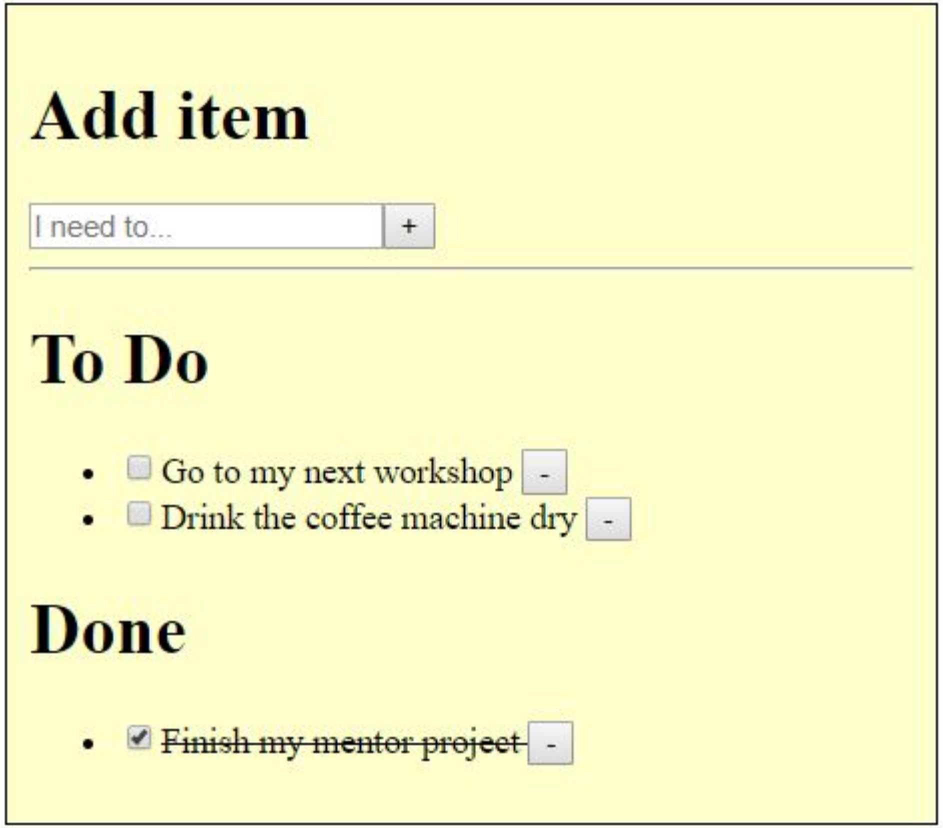
Moving nodes

- Get a reference to the node you want to move:
  ```
  let someNode = document.getElementById("myListItem")
  ```
- Find the node you want to move the element under:
  ```
  let parentNode = document.getElementById("myOtherList")
  ```
- Append the target node to the new parent’s list of children:
  ```
  parentNode.appendChild(someNode)
  ```
- This will rip it unceremoniously from its current place in the tree (💔) and attach it to its new parent.

Exercise Time

To-Do List

- It helps you "get stuff done"
- (get it? … sorry)



Add item

I need to... +

To Do

- ☐ Go to my next workshop -
- ☐ Drink the coffee machine dry -

Done

- ☑ ~~Finish my mentor project~~ -

Your options

- To-do list exercise in your handout
    - Coding exercise (JavaScript) *and/or*
    - Design exercise (CSS)
- Grok Learning course
    - Continue with the HTML/CSS course we worked on in the summer workshop
- Work on your own websites and ask questions if you have any
    - Try something new, continue the websites you made in summer, or continue with a website you made for your mentor project?
    - You could make a website *about* your summer project

Cool Demo Time!

Cool Demos

- I didn’t make any of these demos.
- I’m not *that* cool.

Ha! You were expecting buns?

(Actually, these are Jess’s piggies)

Okay. Fine. Here’s another bunny.