IMPORTANT NOTICE

Next week, students will be required to hand in wireframes for their final projects. You can create wireframes using tools like Balsamiq Mockups, Sketch or just plain old pen and paper.

Previous FEWD projects from around the world are here:

https://gallery.generalassemb.ly/FEWD

FEWD - WEEK 4

WILL MYERS

Freelance Front End Developer

SLIDES

http://www.slideshare.net/wilkom/fewd-week4-slides

YOUR WEEKLY FEWD GITHUB REPOSITORY

- Use the '+' button in the top-left of GitHub Desktop (*Create* tab)
- Create a new repository called *'FEWD_Week4'*
- Choose the [home]/FEWD folder for the local path
- Open this repo folder in your editor
- Commit the changes and publish the *FEWD_Week4* repository to github.com

YOUR WEEKLY WORKING FILES FROM ME

To get the *week4_working_files* you should just be able to select the *ga-fewd-files* repository in GitHub Desktop and press 'Sync'. This should pull in this weeks folder from github.com.

If you any difficulties you should just re-clone the *ga-fewd-files* repository.

Copy the whole *week4_working_files* into your *FEWD_Week4* repository and commit and publish to github.com

AGENDA

- Introduction To Programming
- What JS Can Do
- Reading JS
- Lab

ASSIGNMENT FOR TODAY

Continue with Relaxr using JavaScript and jQuery

Also...

IMPORTANT NOTICE

Next week, students will be required to hand in wireframes for their final projects. You can create wireframes using tools like Balsamiq Mockups, Sketch or just plain old pen and paper.

Previous FEWD projects from around the world are here:

https://gallery.generalassemb.ly/FEWD

INTRODUCTION TO PROGRAMMING

So far we have looked at HTML as a **markup language**, and CSS as a **styling language**. We now want to think about a **programming language** so that we can get the computer to perform tasks.

INTRODUCTION TO PROGRAMMING

The computer will do what you tell it to do.

WHAT IS A PROGRAM

A **program** is a set of instructions that you write to tell a computer what to do

WHAT IS PROGRAMMING

Programming is the task of writing those instructions in a language that the computer can understand.

BECOMING A PROGRAMMER

At the beginning, it isn't about the programming language. It is about changing how you think.

We have to know how the computer thinks to change how we think.

HOW COMPUTERS 'THINK'

The short answer is that they don't think.

The slightly longer answer is that while computers donât think, they act as if they do, by sequentially executing simple instructions.

The only things a computer knows are the things we tell it.

PSEUDO CODE

Pseudocode is the process of writing a program without using the syntax of a programming language.

Pseudocode is a mixture of natural language and high-level programming constructs. For example,

If the door is closed and I want to exit the room, **then** open the door

We can pseudocode to **train our brain** for programming.

PSEUDO CODE

In pseudocode we can introduce simple bits of programming syntax and naming conventions.

- join **meaningful words** together with **underscores** to imply a computer task or some computer data e.g. `my_email_inbox`
- use **verbs** to imply a computer task (**action**): e.g. `get_my_email`
- use some high-level programming syntax like `if...then` or `repeat`
- use simple arithmetic **operators** like = (assignment), == (equals), < (less than), > (greater than)

THERMOSTAT PSEUDOCODE

Let's try and write the pseudocode for a thermostat that controls a heater

- what data do we need to know from the thermostat?
- what **actions** do we want to be able to do with the heater?
- how often should we be performing our actions?

THERMOSTAT PSEUDOCODE

```
get target_temperature
target_temperature = 72
repeat forever,
      current_temperature = get_sensor_reading
      if target_temperature > (current_temperature+5),
          turn_on_heater
      if target_temperature <= current_temperature,
          turn_off_heater
```

ROCK PAPER SCISSORS PSEUDOCODE

- What does each *thing* have in common with the other two?
- What happens each time a *turn* takes place?
- What happens each time that is different?
- What happens each time that is the same?

WHAT IS JAVASCRIPT

JavaScript is the **behaviour** of a web site.

- HTML - Document Structure
- CSS - Looks, Style
- JavaScript - Logic, Functionality, Behavior

It is historically seen as wrong to mix these things up. Putting things where they belong reduces complexity and increases maintainability. However some new JS frameworks are challenging this way of thinking.

JAVASCRIPT AND THE DOM

JavaScript in the browser has an API (application program interface) for accessing and editing the DOM (the document object model) of an HTML document. The DOM is a hierarchical tree representation of the structure of the HTML.

JavaScript can target specific elements on an HTML page and show, hide, style, edit and animate them. It can also listen for events emitted by an HTML page (e.g mouse-clicks) and invoke functions when an event occurs.

WHAT CAN JAVASCRIPT DO?

- Front-end website logic - user input event handling, dynamically applying styles
- `<canvas>`, `<audio>`, `<video>`
- Server-side NodeJS
- WebGL

https://developer.mozilla.org/samples/video/chroma-key/index.xhtml (works in Firefox)

http://mdn.github.io/canvas-raycaster/

http://webaudiodemos.appspot.com/wubwubwub/index.html

http://threejs.org/examples/

COLOR SWITCHER

Open *week4_working_files/color_scheme_switcher*

Let's think about the **control flow** of a program.

We will think in a more *high-level* way about coding at first. JavaScript syntax will come later.

COLOR SWITCHER - CONTROL FLOW

In computer science, control flow (or alternatively, flow of control) is the order in which individual statements, instructions or function calls of an imperative program are executed or evaluated. *[Wikipedia]*

In `js/index.js` try moving the `<script>` link for the JavaScript out of the `<body>` and into the `<head>`.

What is happening in `js/index.js`?

EVENTS

An event is *something that happens* that *other things can respond to*. An object **triggers an event**, and an **event listener** (or **handler**) fires when the event is triggered.

E.g. the `click` event occurs when the user clicks on an element.

An event and event listener are an implementation of the **Observer pattern** - a pattern is a recognised and repeated way of programming. The Observer pattern involves an **observable** object (which triggers the event) and an **observer** listener (which responds to the event).

TRAFFIC LIGHT

Go to this CodePen: http://codepen.io/nevan/pen/shtLA

The green light does not work. Change the code so that the traffic light works correctly.

AGENDA AFTER LUNCH

- Intro To Programming Reivew
- Intro To jQuery
- jQuery Basics
  - File Structure
  - Syntax
- Adding Interactivity

INTRO TO PROGRAMMING REVIEW

Any questions?

INTRO TO JQUERY

jQuery is a cross-browser JavaScript **library**, designed to simplify front-end JavaScript web programming.

jQuery is written in JavaScript.

JQUERY

jQuery is designed to make the following things easier:

- Document traversal
- Modify the appearance of the page (CSS Manipulation)
- Edit the page content
- Respond to user interaction (Event Handling)
- Add animation
- Retrieve data from a server using AJAX (Asynchronous JavaScript and XML)
- Simplify common JavaScript tasks

JQUERY

jQuery also provides the following useful features:

- uses CSS selector syntax
- supports extensions
- abstracts away browser quirks
- allows multiple actions to be defined in one line with chaining

JQUERY VS VANILLA JS

Open *week4_working_files/styling-css-with-jquery* and *week4_working_files/styling-css-with-js*

The advantages of jQuery for a new programmer is that there is less code to right, it is somewhat easier to read and understand, and it is cross-browser compatible.

DISADVANTAGES OF JQUERY

The two main downsides to using the jQuery library are:

- it is an additional file download which will delay rendering your website for the first time
- it is an *abstraction* which has **slower** performance than using native functionality.

DO WE NEED JQUERY?

jQuery is a mature and robust library that provides cross-browser compatibility for a wide range of tasks. This has made it very popular over the years, particularly when supporting browsers like IE.

But browsers are evolving and it is now possible to use native JavaScript APIs that are much faster and have good support across newer browsers.

For example `document.querySelector` and `document.querySelectorAll` allow selecting elements from the DOM with **CSS selector syntax** and are now relatively mature.

DO WE NEED JQUERY?

jQuery is still an important library, particularly for a new web programmer. Its maturity and cross-browser compatibility are still essential for many web sites. So it is still worth learning.

But remember that you can and will eventually do more JavaScript **without** an abstraction like jQuery. jQuery will slowly become less important as native browser functionality matures over time.

http://youmightnotneedjquery.com/ (Keep this link open)

http://stackoverflow.com/questions/11503534/jquery-vs-document-queryselectorall

JS/JQUERY BASICS

Saying all that, this week we'll use jQuery to add JavaScript to our web pages. Because it is easier, we can use it as a stepping stone to using vanilla JavaScript.

JQUERY SCRIPT TAGS

Adding jQuery script tag to your website

```
// Adding the file.
<script src="js/jquery-1.8.3.min.js"></script>
```

```
// linking the library from a CDN
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.mir
```

SYNTAX

Syntax: Spelling and grammar rules of a programming language.

We will look at jQuery syntax and some basic JS syntax first. We will look further at vanilla JavaScript syntax next week.

Like with any language, there are formal rules around how to write it. This is the syntax.

BASIC JAVASCRIPT SYNTAX - PUNCTUATION

- Semicolon `;` - should come at the end of an **expression**
- Curly Braces `{ }` - denotes a **block** of code e.g a **function** or an **object literal**
- Parentheses `( )` - used to **invoke** (call) a function, or to **evaluate** part of an expression
- Quotation Marks `" "` - contains a **string** of text, e.g. "Hello World!"

BASIC JAVASCRIPT SYNTAX - COMMENTS

```
//Single Line Comments

/* Multi line comments */
```

Use **Cmd +** / in your text editor to toggle comments, same as for CSS and HTML

JQUERY SYNTAX

$ DOLLAR SYMBOL

When you import the jQuery library into your web page the dollar $ symbol by default becomes a global identifier (global variable) for jQuery.

It is possible to change this, as the dollar symbol is used widely in different programming languages and libraries.

But most jQuery implementations on the front-end stick with the $, and we will too.

JQUERY SYNTAX

$ DOLLAR SYMBOL

With jQuery firstly you **select** a DOM element (DOM traversal) and it gets *wrapped* in a jQuery object.

Then you **invoke** a **method** on this object which does something to the selected element, e.g. changes what it looks like, or adds some user interaction logic.

```
$("selector").method(argument)
```

JQUERY SYNTAX - SELECTORS

Selectors are just like CSS

For selecting an element in the DOM, just use the $ followed by parentheses (invocation). The parentheses contain a **string** as an **argument**. This string is a **CSS selector**.

```
$("#id")
$(".class")
$("main")
```

JQUERY SYNTAX - METHODS

To invoke a method on your jQuery-selected element you use **dot syntax**, following the **selector expression**.

You pass further arguments into your **method invocation**.

E.g. You can change the CSS style on a selected item with the `$.css()` method.

```
$("main").css("border", "10px solid black");
```

JQUERY CLICK EVENT

https://api.jquery.com/click/

$(myElement).click(someFunction)

This is used for triggering a function call when a user clicks on an element.

It is a short-form of a more generic jQuery syntax: `.on( "click", handler )`.

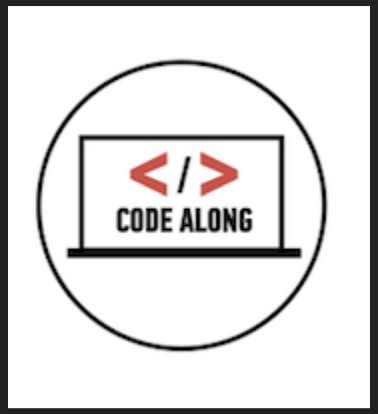
```
$('selector').click(doSomething);
function doSomething() {
    // make something happen here
}
```

What's the syntax for a click event handler in vanilla JS?

JQUERY SYNTAX - SOME MORE METHODS

- .click
- .slideToggle()
- .hide()
- .show()
- .slideUp()
- .slideDown()
- .children()
- .attr()

You can look them up at https://api.jquery.com/



JQUERY TRAFFIC LIGHT

Let's 'jQueryify' our traffic light example. Open up *week4_working_files/jquery_traffic_light_exercise*

We can also try and 'jQueryify' the *color_scheme_switcher* we looked at earlier. Create a new `index_jq.js` file in *week4_working_files/color_scheme_switcher/js* and link this into the html file instead of `index.js`.

JQUERY BREAKDOWN

Let's have a look at what was actually happening in the scrolling page anchor links example from Week 2.

Open up *week4_working_files/jquery_scrolling_link*

SYNTAX DRILL

Fork this CodePen:
http://codepen.io/GeneralAssembly/pen/EAubl

Try and do the following:

- Use jQuery syntax to change all p tags to blue.
- Use jQuery to change the size of the boxes etc.

ADDING INTERACTIVITY

http://codepen.io/nevan/pen/mKzvs

Play around with this code, so you understand what is happening.

TRIGGERING ANIMATIONS IN VANILLA JS

Open up *week4_working_files/scroll_animated_header*.

This is vanilla JavaScript, can you understand what is going on?

A SIMPLE GAME ROCK PAPER SCISSORS GAME IN VANILLA JS

Open up *week4_working_files/rock_paper_scissors*

Can you make the *game result* display in some text in the page rather than in an alert box?