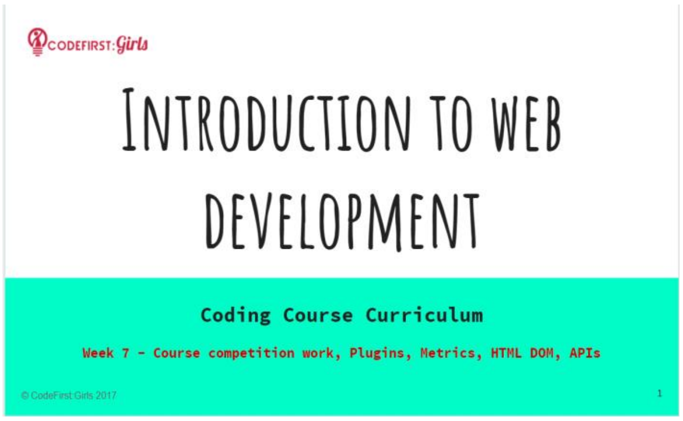
Code First:Girls Beginners Coding course – Front end Web development

Week 7 - Course competition work, Plugins, Metrics, HTML DOM, APIs


CODEFIRST:Girls

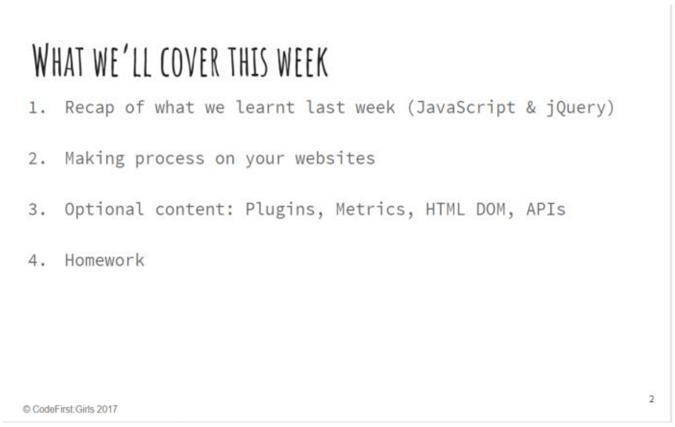
Introduction to web development

Coding Course Curriculum

Week 7 - Course competition work, Plugins, Metrics, HTML DOM, APIs

© CodeFirst:Girls 2017 1

This is the penultimate week of the course! Congratulations on making it this far!
This week is also the last session where we teach course learning materials. Next week is all about getting your projects finished, and presenting your team's website to the instructors and your classmates.


What we'll cover this week

1. Recap of what we learnt last week (JavaScript & jQuery)
2. Making process on your websites
3. Optional content: Plugins, Metrics, HTML DOM, APIs
4. Homework

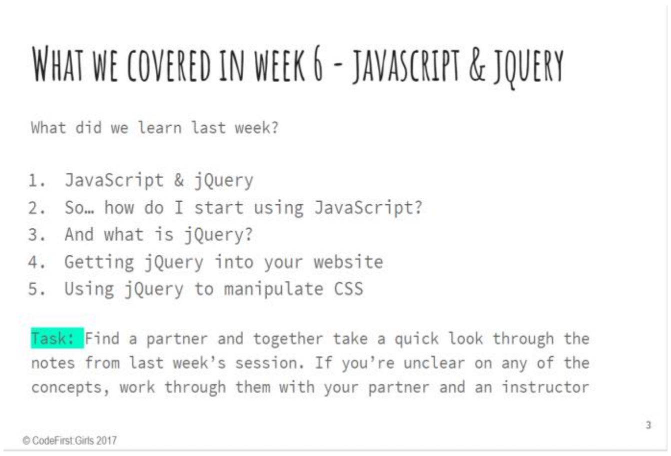
© CodeFirst:Girls 2017 2

This week we will run through what we learnt last week, ie JavaScript and jQuery, and we'll check in with each team to see how they are doing with their website. This will take up the first hour of the class.

After that, you have a choice. You can either spend the rest of the session working on your website, in your teams, or you can learn some other cool stuff, including:

- Plugins
- Website metrics (eg. Google Analytics)
- HTML DOM and
- APIs

Ask the learners what they would like to do. If there is a difference of opinion, and there are enough instructors, one instructor could go into another room or area and help with the website projects while another group



What we covered in week 6 - JavaScript & jQuery

What did we learn last week?

1. JavaScript & jQuery
2. So… how do I start using JavaScript?
3. And what is jQuery?
4. Getting jQuery into your website
5. Using jQuery to manipulate CSS

Task: Find a partner and together take a quick look through the notes from last week's session. If you're unclear on any of the concepts, work through them with your partner and an instructor

3

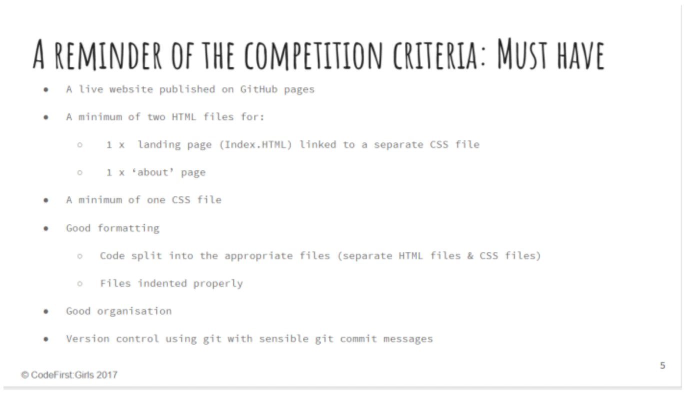
© CodeFirst:Girls 2017

It's nearly competition time!! Next week each team will present their website to the class. The instructors will pick a winner, which will instantly bestow on the winning team members huge riches and instant fame (or as least, Amazon vouchers and the possibility of being listed on the CFG website!)

Course competition

4

A reminder about what the course competition criteria are. What makes a good website? What's need-to-have and what's nice to have?



A reminder of the competition criteria: Must have

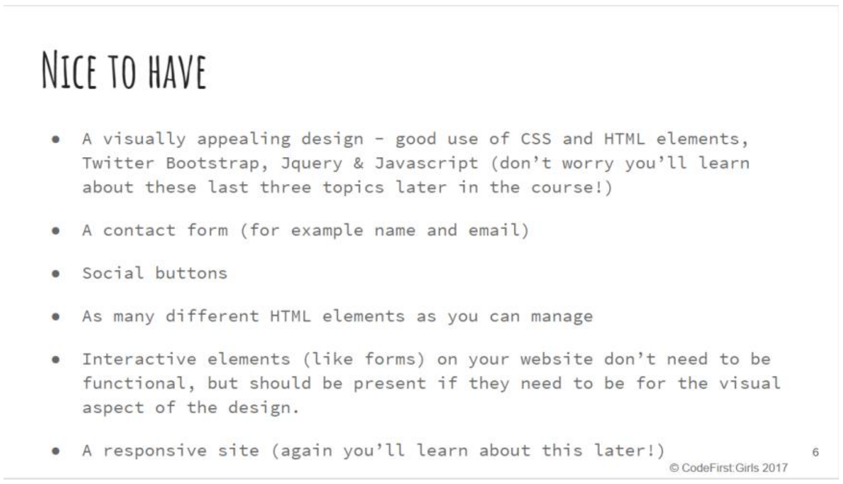
- A live website published on GitHub pages
- A minimum of two HTML files for:
  - 1 x landing page (Index.HTML) linked to a separate CSS file
  - 1 x 'about' page
- A minimum of one CSS file
- Good formatting
  - Code split into the appropriate files (separate HTML files & CSS files)
  - Files indented properly
- Good organisation
- Version control using git with sensible git commit messages

© CodeFirst:Girls 2017

5

These are the must-have features for your website:

1. A live website published on GitHub pages
2. A minimum of two HTML files for:
   a. 1 x landing page (Index.HTML) linked to a separate CSS file
   b. 1 x 'about' page
3. A minimum of one CSS file
4. Good formatting
   a. Code split into the appropriate files (separate HTML files & CSS files)
   b. Files indented properly
5. Good organisation
6. Version control using git with sensible git commit messages


Nice to have

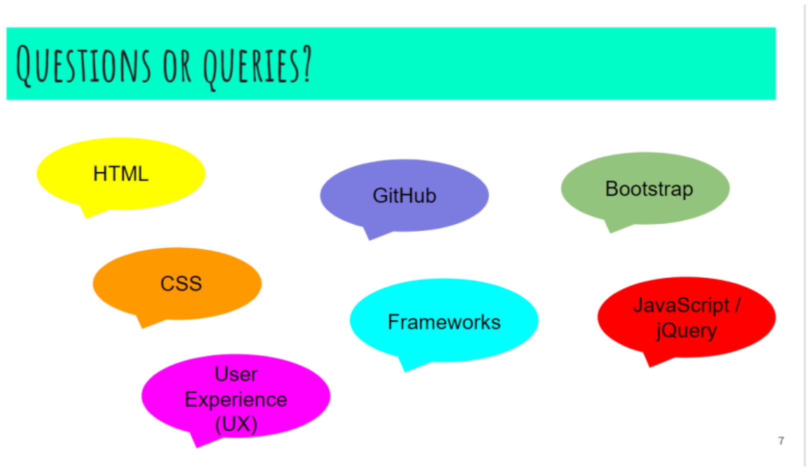
- A visually appealing design - good use of CSS and HTML elements, Twitter Bootstrap, Jquery & Javascript (don't worry you'll learn about these last three topics later in the course!)
- A contact form (for example name and email)
- Social buttons
- As many different HTML elements as you can manage
- Interactive elements (like forms) on your website don't need to be functional, but should be present if they need to be for the visual aspect of the design.
- A responsive site (again you'll learn about this later!)

6

© CodeFirst:Girls 2017

These are the nice to have features for your website:

1. A visually appealing design - good use of CSS and HTML elements, Twitter Bootstrap, Jquery & Javascript (don't worry you'll learn about these last three topics later in the course!)
2. A contact form (for example name and email)
3. Social buttons
4. As many different HTML elements as you can manage
5. Interactive elements (like forms) on your website don't need to be functional, but should be present if they need to be for the visual aspect of the design.
6. A responsive site (again you'll learn about this later!)


Questions or Queries?

Before we decide what to do in the remaining time, are there any questions?

Now you have an hour to continue with your projects, working in teams, to build your websites. If you have any questions or need help from an instructor, please let us know.

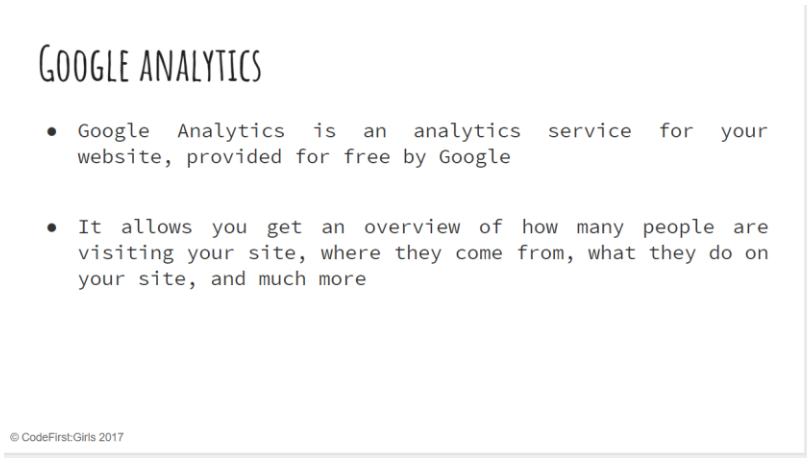
Optional content

(Google analytics, domain names for Github pages)

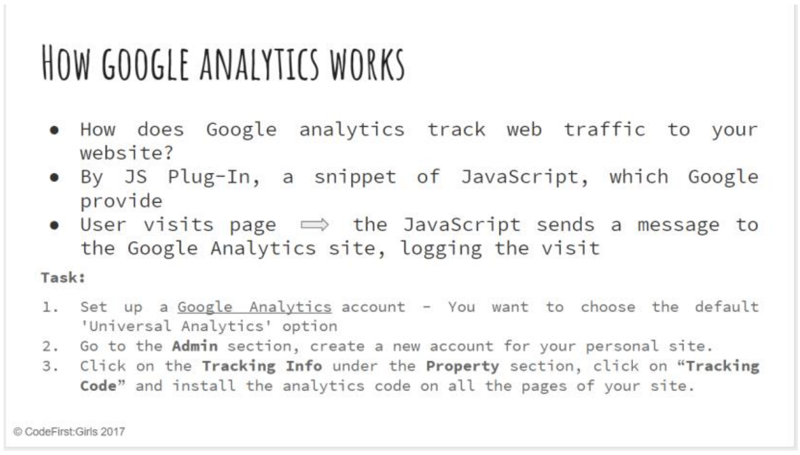
9

This is the last bit of taught materials. Ask students to choose one or two of these subjects and spend the last 30 mins going through them.

- Google analytics
- Google forms
- Domain names for GitHub pages
- Content delivery networks
- HTML DOM, Interacting with APIs, working with company APIs


GOOGLE ANALYTICS

- Google Analytics is an analytics service for your website, provided for free by Google
- It allows you get an overview of how many people are visiting your site, where they come from, what they do on your site, and much more


HOW GOOGLE ANALYTICS WORKS

- How does Google analytics track web traffic to your website?
- By JS Plug-In, a snippet of JavaScript, which Google provide
- User visits page ⇨ the JavaScript sends a message to the Google Analytics site, logging the visit

Task:

1. Set up a Google Analytics account - You want to choose the default 'Universal Analytics' option
2. Go to the **Admin** section, create a new account for your personal site.
3. Click on the **Tracking Info** under the **Property** section, click on "**Tracking Code**" and install the analytics code on all the pages of your site.

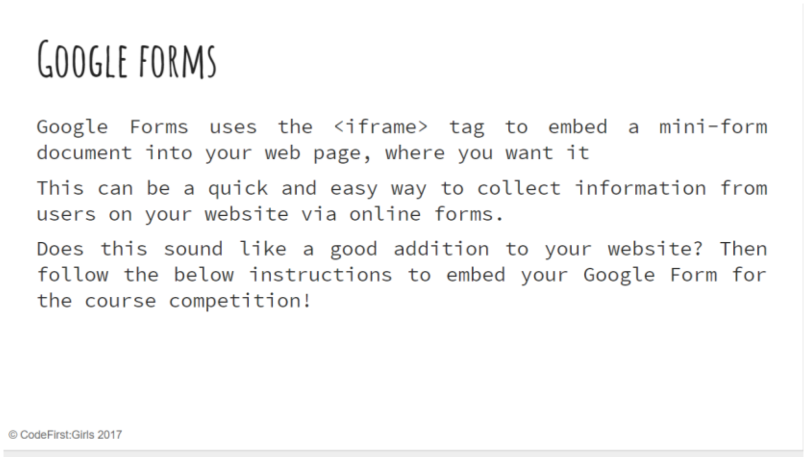
How does Google Analytics work?

To use Google Analytics you need to place JS Plug-In, a snippet of JavaScript, (which Google provide) on each of the HTML pages on your site.

When a user visits the page, the JavaScript sends a message to the Google Analytics site, logging the visit

Task (for students):

1. Set up a Google Analytics account - You want to choose the default 'Universal Analytics' option
2. Go to the **Admin** section, create a new account for your personal site.
3. Click on the **Tracking Info** under the **Property** section, click on "**Tracking Code**" and install the analytics code on all the pages of your site.


GOOGLE FORMS

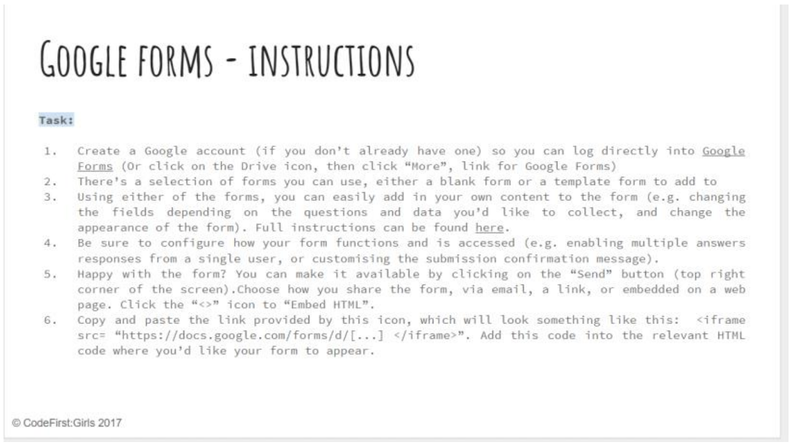
Google Forms uses the <iframe> tag to embed a mini-form document into your web page, where you want it

This can be a quick and easy way to collect information from users on your website via online forms.

Does this sound like a good addition to your website? Then follow the below instructions to embed your Google Form for the course competition!

Google Forms

Google Forms uses the <iframe> tag to embed a mini-form document into your web page, where you want it. This can be a quick and easy way to collect information from users on your website via online forms. Does this sound like a good addition to your website? Then follow the below instructions to embed your Google Form for the course competition!


GOOGLE FORMS - INSTRUCTIONS

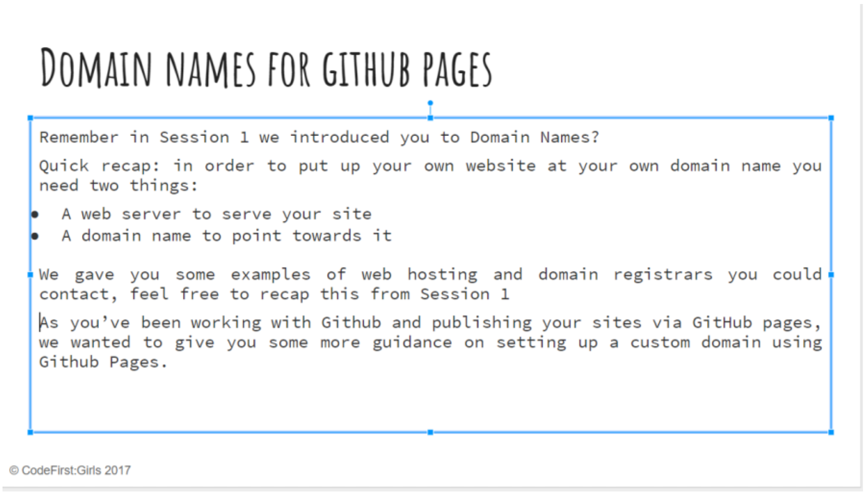
Task:

1. Create a Google account (if you don't already have one) so you can log directly into Google Forms (Or click on the Drive icon, then click "More", link for Google Forms)
2. There's a selection of forms you can use, either a blank form or a template form to add to
3. Using either of the forms, you can easily add in your own content to the form (e.g. changing the fields depending on the questions and data you'd like to collect, and change the appearance of the form). Full instructions can be found here.
4. Be sure to configure how your form functions and is accessed (e.g. enabling multiple answers responses from a single user, or customising the submission confirmation message).
5. Happy with the form? You can make it available by clicking on the "Send" button (top right corner of the screen).Choose how you share the form, via email, a link, or embedded on a web page. Click the "<>" icon to "Embed HTML".
6. Copy and paste the link provided by this icon, which will look something like this: <iframe src= "https://docs.google.com/forms/d/[...] </iframe>". Add this code into the relevant HTML code where you'd like your form to appear.

Task:

1. If you don't already have one create a Google account, so that you can then log directly into Google Forms. Or alternatively click on the Drive icon, then click more and you'll see a link for Google Forms.
2. You'll then be presented with a selection of forms you can use, either a blank form or a template form to add to.

3. Using either of the forms, you'll be able to easily add in your own content to the form (e.g. changing the fields depending on the questions and data you'd like to collect, and amend the appearance of the form). Full instructions can be found here.
4. Be sure to configure how your form functions and is accessed (e.g. enabling multiple answers responses from a single user and customising the submission confirmation message for example).
5. Once you are happy with the form you can make it available by clicking on the "Send" button in the top right corner of the screen. You can then choose how you share the form, via email, a link, or embedded on a web page. Click the "<>" icon to "Embed HTML".
6. Copy and paste the link provided by this icon, which will look something like this: <iframe src= "https://docs.google.com/forms/d/[...] </iframe>". Add this code into the relevant HTML code where you'd like your form to appear.


DOMAIN NAMES FOR GITHUB PAGES

Remember in Session 1 we introduced you to Domain Names?

Quick recap: in order to put up your own website at your own domain name you need two things:

- A web server to serve your site
- A domain name to point towards it

We gave you some examples of web hosting and domain registrars you could contact, feel free to recap this from Session 1

As you've been working with Github and publishing your sites via GitHub pages, we wanted to give you some more guidance on setting up a custom domain using Github Pages.

© CodeFirst:Girls 2017

Domain Names for Github Pages

Remember in Session 1 we previously introduced you to Domain Names? In case you need a quick recap, in order to put up your own website at your own domain name you need two things:

1. A web server to serve your site
2. A domain name to point towards it

We previously gave you some examples of web hosting and domain registrars you could contact, feel free to recap this from Session 1. As you've been working with Github and publishing your sites via Github pages we wanted to give you some more guidance on setting up a custom domain using Github Pages.

HOW TO CREATE A DOMAIN NAME FOR GITHUB PAGES

Task (if you have bought a personal domain name):

For Github pages there are three main stages to setting up a custom domain:

1. Pick a custom domain and register it with a DNS provider/Domain Registrar/DNS host (3 names for the same service). In Session 1 we provided some examples of these: 123-reg.co.uk, godaddy.com and namecheap.com.
2. Add your custom domain to your Github Pages site. Follow these instructions here.
3. Set up your custom domain with your DNS provider. For more information on how this should look find some examples from Github pages here and here.

© CodeFirst:Girls 2017

Task (if you have bought a personal domain name):

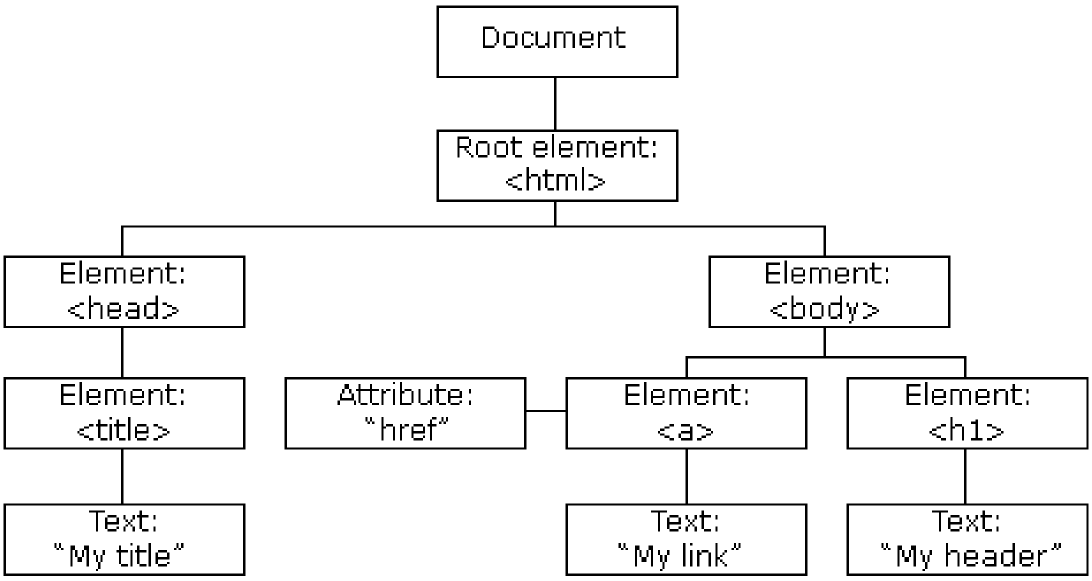
For Github pages there are three main stages to setting up a custom domain:

1. Pick a custom domain and register it with a DNS provider/Domain Registrar/DNS host (three names for the same service). In Session 1 we provided some examples of these: 123-reg.co.uk, godaddy.com and namecheap.com.
2. Add your custom domain to your Github Pages site. Follow these instructions here.
3. Set up your custom domain with your DNS provider. For more information on how this should look find some examples from Github pages here and here.

The HTML DOM

As we have covered in this course, HTML, CSS and JavaScript are the three main components that make almost every website. How do browsers comprehend our code?

When they receive code, browsers start building websites by converting all of the HTML



they receive into a JavaScript object called the Document Object Model (DOM).

In fact, from the Developer Tools console, you can examine the DOM for any website.

The DOM is demonstrated by the node-tree visual representation in your developer tools, identifying relevant attributes (and their respective values), CSS styles, JavaScript event listeners, breakpoints, and properties of each HTML element. This is important because this is how we create dynamic websites; Through the DOM, JavaScript is able to manipulate all aspects of a web page, from HTML elements and their attributes, CSS styles, adding or removing new HTML elements & their attributes, reacting to existing HTML events on the page, to finally, creating new HTML events on the page.

Formally, the HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as **objects**
- The **properties** (values you can obtain or set) of all HTML elements
- The **methods** (actions you can take) to access all HTML elements
- The **events** (scenarios which actions lead to) for all HTML elements

It is a W3C Standard for how to manipulate HTML elements. A lot of functions we have used so far in jQuery have made use of the DOM to tell the computer what to do when users interact with the page.

What is an API?

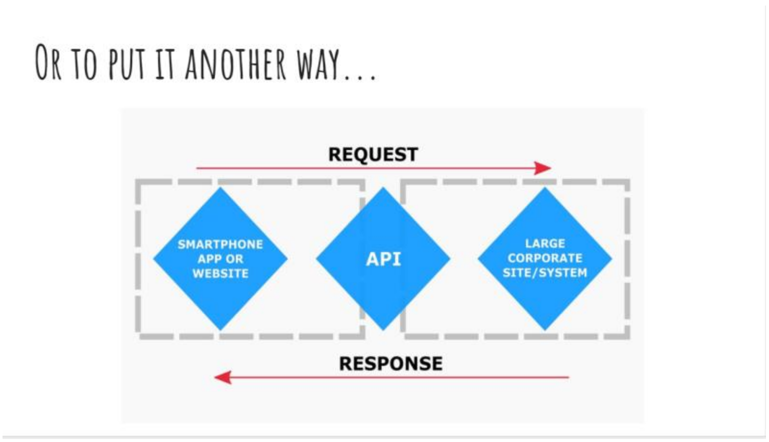
The web is made up of a large network of servers, all connected together. Every web page on the internet is stored on a remote server. A remote server is part of a remotely located computer that is "optimized" to process requests

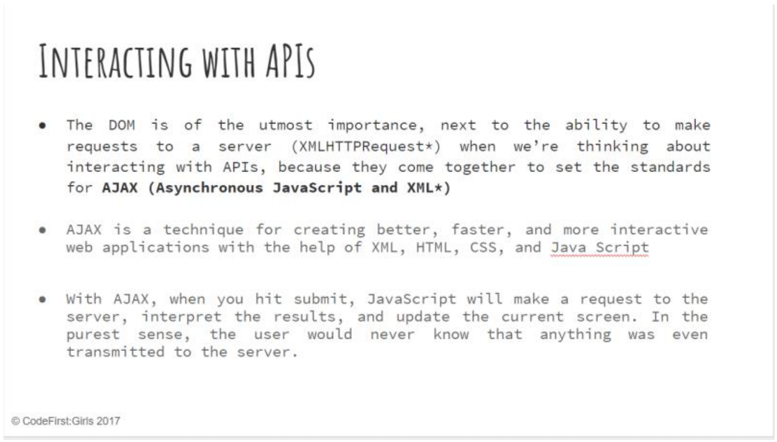
When you type www.facebook.com into your browser, a request goes to Facebook's remote server. Once your browser receives the response, it interprets the code and displays the page

To the browser, also known as the *client*, Facebook's server is an API. This means that every time you visit a page on the Web, you interact with some remote server's API

An API isn't the same as the remote server — rather **it is the part of the server that receives requests and sends responses**.

Or to put it another way...


Or to put it another way...
REQUEST
SMARTPHONE APP OR WEBSITE
API
LARGE CORPORATE SITE/SYSTEM
RESPONSE



Interacting with APIs

- The DOM is of the utmost importance, next to the ability to make requests to a server (XMLHTTPRequest*) when we're thinking about interacting with APIs, because they come together to set the standards for **AJAX (Asynchronous JavaScript and XML*)**
- AJAX is a technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script
- With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.

Interacting with an API

The DOM is of the utmost importance, next to the ability to make requests to a server (XMLHTTPRequest*) when we're thinking about interacting with APIs, because they come together to set the standards for **AJAX (Asynchronous JavaScript and XML*)**.

AJAX is a technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.

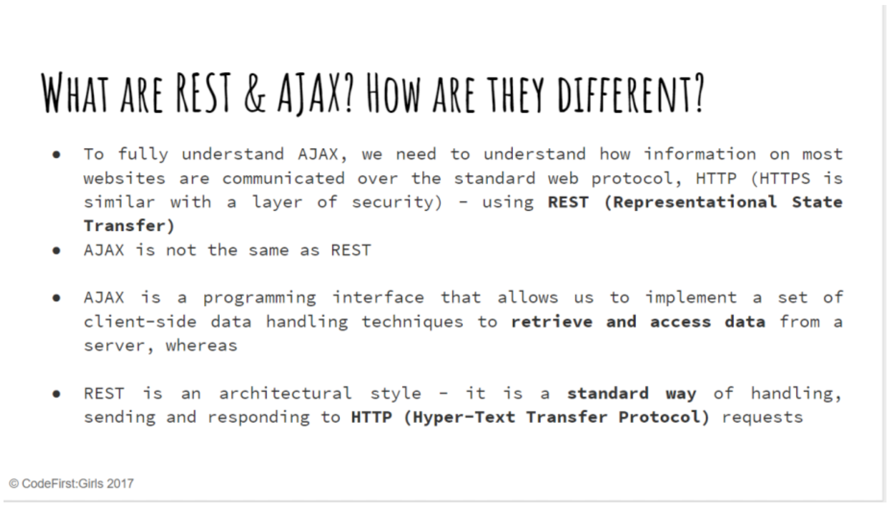
Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.

- With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.

AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

We will explain how AJAX is used to ask for and receive data from servers, and we can use the super straightforward jQuery way here.

** You don't have to understand XML to use AJAX - it's a historic and somewhat misleading name*



WHAT ARE REST & AJAX? HOW ARE THEY DIFFERENT?

- To fully understand AJAX, we need to understand how information on most websites are communicated over the standard web protocol, HTTP (HTTPS is similar with a layer of security) - using **REST (Representational State Transfer)**
- AJAX is not the same as REST
- AJAX is a programming interface that allows us to implement a set of client-side data handling techniques to **retrieve and access data** from a server, whereas
- REST is an architectural style - it is a **standard way** of handling, sending and responding to **HTTP (Hyper-Text Transfer Protocol)** requests

© CodeFirst:Girls 2017

REST & AJAX

To fully understand AJAX, we need to understand how information on most websites are communicated over the standard web protocol, HTTP (HTTPS is similar with a layer of security) - using **REST (Representational State Transfer)**.

AJAX is not REST. AJAX is a programming interface that allows us to implement a set of client-side data handling techniques to **retrieve and access data** from a server, whereas REST is an architectural style - it is a **standard way** of handling, sending and responding to **HTTP (Hyper-Text Transfer Protocol)** requests.

REST is an architectural standard for HTTP requests and defines how applications are built to respond to requests for data,. AJAX is the code in our website calling for data.

The details of REST would be something you would learn in a back-end course, but what we need to know is the types of standard request methods one can make to a RESTful API: GET - retrieve, POST - update, PUT - create, and DELETE - does what it says it will.

Applications built using REST allow clients to retrieve and manipulate data using AJAX and an authentication layer.

JSON

- So now we know (roughly)
  - how clients fetch data (AJAX), and
  - how applications are built on the server-side to listen for our requests for data (REST architectures), so…

how is the data received?

© CodeFirst:Girls 2017

JSON

Okay, so now we know roughly how clients fetch data (AJAX), and how applications are built on the server-side to listen for our requests for data (REST architectures), how is the data received?

For JavaScript, we refer to the data that is sent back to us by servers as **JSON - JavaScript Object Notation**. JSON is a simple key-value mapping of data which is very quick to manipulate.

For example, let's use Facebook's example which demonstrates how their Graph API is structured: This GET request:

```
GET                                                    graph.facebook.com
  /facebook/picture?
    redirect=false
```

Returns this JSON:

```
{
       data:
        {
                is_silhouette: false,
                url:                                       "https://scontent.xx.fbcdn.net/v/t1.0-
        1/p100x100/12006203_10154088276211729_2432197377106462187_n.png?oh=d1b6b18e1846c1ad
        efd157a45c4d384d&oe=58226457"
        }
}
```
Facebook’s Graph API was built to handle requests with specific parameters, in order to return their logo to us in JSON format.

Working with company APIs

What are some good APIs for us to try out on our websites?

Let's go with the well-established, public facing APIs of some big tech cos:

- Twitter for Websites
- Google Maps
- Facebook Social Sharing Plugins
- Facebook Graph API

As Developers, we love tools that make our lives easier, so let's get a useful tool for interacting and testing interactions with APIs. Postman is a Chrome extension that does just what we need it to.

Some companies provide their own consoles, which allow us to try out HTTP requests in browser too; pretty handy!

JSON request example (from Facebook)

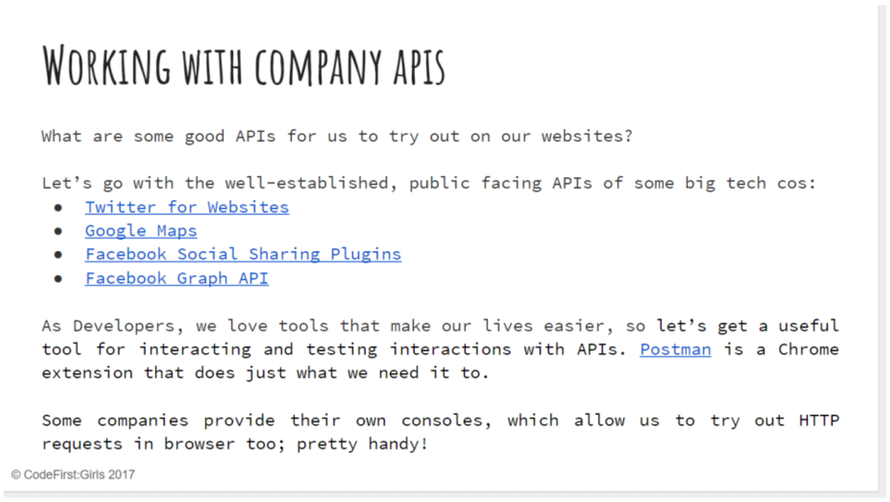
- This Facebook example demonstrates how their Graph API is structured: This GET request:

```
GET graph.facebook.com
  /facebook/picture?
       redirect=false
```

Returns this JSON:

```
{
      data:
      {
            is_silhouette: false,
            url:
      "https://scontent.xx.fbcdn.net/v/t1.0-1/p100x100/12006203_10154088276211729_2432197377106462187_n.png?oh=d1b6b18e1846c1adefd1
      57a45c4d384d&oe=58226457"
      }
}
```

Facebook's Graph API was built to handle requests with specific parameters, in order to return their logo to us in JSON form


Working with company APIs

What are some good APIs for us to try out on our websites?

Let's go with the well-established, public facing APIs of some big tech cos:

- Twitter for Websites
- Google Maps
- Facebook Social Sharing Plugins
- Facebook Graph API

As Developers, we love tools that make our lives easier, so let's get a useful tool for interacting and testing interactions with APIs. Postman is a Chrome extension that does just what we need it to.

Some companies provide their own consoles, which allow us to try out HTTP requests in browser too; pretty handy!

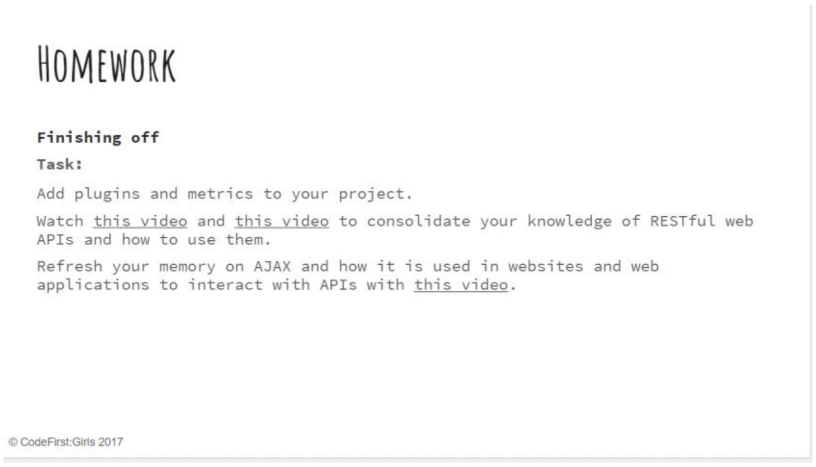
Working with company APIs

What are some good APIs for us to try out on our websites? Let's go with the well-established, public facing APIs of some big tech cos:

- Twitter for Websites
- Google Maps
- Facebook Social Sharing Plugins
- Facebook Graph API

As Developers, we love tools that make our lives easier, so let's get ourselves a useful tool for interacting and testing interactions with APIs. Postman is a chrome extension that does just what we need it to.

Some companies provide their own consoles which allow us to try out HTTP requests in browser too, which is pretty handy.


HOMEWORK

Finishing off

Task:

Add plugins and metrics to your project.

Watch this video and this video to consolidate your knowledge of RESTful web APIs and how to use them.

Refresh your memory on AJAX and how it is used in websites and web applications to interact with APIs with this video.

Homework

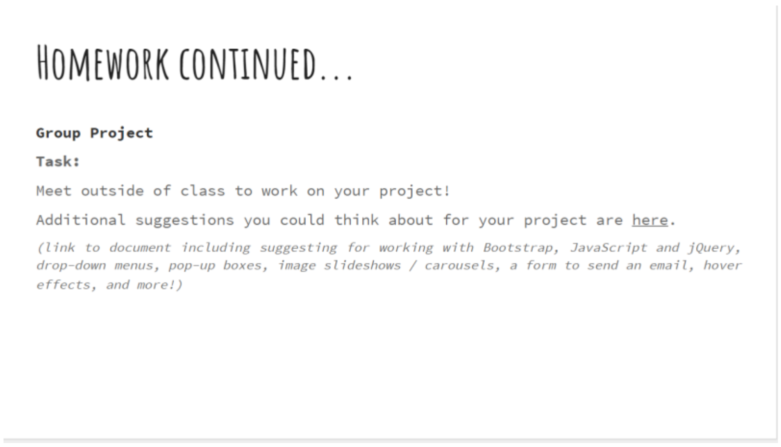
Finishing off

Task:

Add plugins and metrics to your project.

Watch [this video]() and [this video]() to consolidate your knowledge of RESTful web APIs and how to use them.

Refresh your memory on AJAX and how it is used in websites and web applications to interact with APIs with [this video]().


HOMEWORK CONTINUED...

Group Project

Task:

Meet outside of class to work on your project!

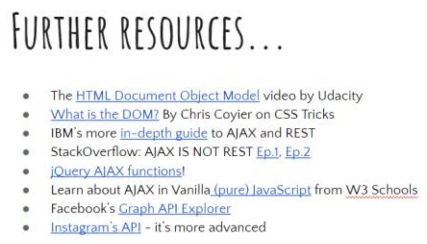
Additional suggestions you could think about for your project are here.

(link to document including suggesting for working with Bootstrap, JavaScript and jQuery, drop-down menus, pop-up boxes, image slideshows / carousels, a form to send an email, hover effects, and more!)

Homework: Group Project

Task:

Meet outside of class to work on your project!



FURTHER RESOURCES…

- The HTML Document Object Model video by Udacity
- What is the DOM? By Chris Coyier on CSS Tricks
- IBM's more in-depth guide to AJAX and REST
- StackOverflow: AJAX IS NOT REST Ep.1, Ep.2
- jQuery AJAX functions!
- Learn about AJAX in Vanilla (pure) JavaScript from W3 Schools
- Facebook's Graph API Explorer
- Instagram's API – it's more advanced

Further Resources

The HTML Document Object Model video by Udacity
What is the DOM? By Chris Coyier on CSS Tricks
IBM's more in-depth guide to AJAX and REST
StackOverflow: AJAX IS NOT REST Ep.1, Ep.2
jQuery AJAX functions!
Learn about AJAX in Vanilla (pure) JavaScript from W3 Schools
Facebook's Graph API Explorer
Instagram's API – it's more advanced