CODEFIRST: Professionals

Web Fundamentals - Session 7

Plugins, Metrics, and more!

Google Analytics

Google Analytics is an analytics service 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 it works

To use Google Analytics you need to place JS Plug-In, a snippet of JavaScript, (that they 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:

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!:

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 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.

Task:

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.

Content Delivery Networks

When you're serving a web application to a large audience at scale across different countries, you need a service that delivers your web content to users based on where they are located on visiting your website. This is where a Content Delivery Network (CDN) comes in. You've heard about CDNs when using Bootstrap and jQuery, so it's not a new concept.

CDNs speed the delivery of content to users when websites are facing high traffic across a global reach. Companies that provide CDN services look after a networked system of distributed servers which deliver content to users faster by owning CDN servers that are geographically closer to users.

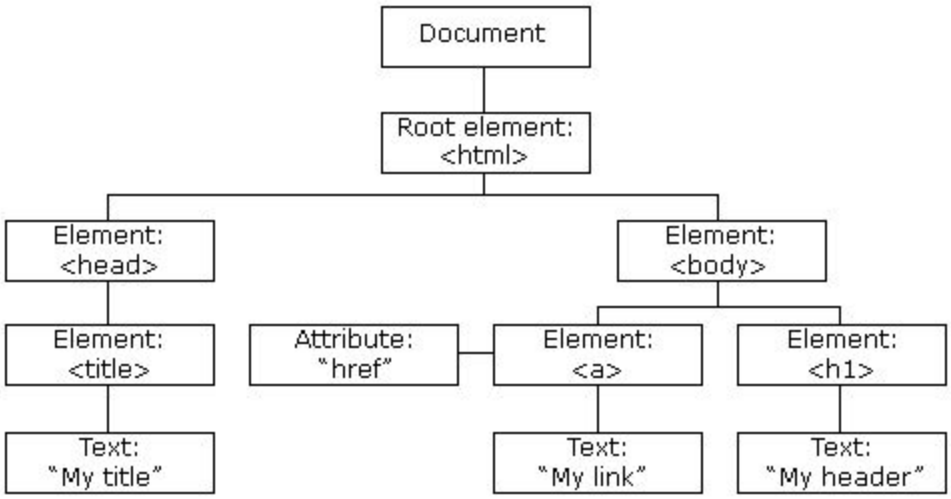
CDN servers communicate with the originating server to make sure all content is up to date, and cache information so that if one server goes down, another one can pick up quickly. A popular CDN provider, which also provides DNS and security services is CloudFlare.

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.

CODEFIRST: Professionals

Web Fundamentals - Session 7

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 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.

To refresh your memory from the last lesson, we spoke about 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*

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)**. (There is another alternative called SOAP, but that is getting out of date now REST has taken over most popular websites).

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.

CODEFIRST: Professionals

Web Fundamentals - Session 7

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

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=d1b6b18e1846c1adefd157a45c4d384d&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.

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.

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