Isomorphic Javascript

The Future of the Web

Markus Hederström

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

Contents

Abstract

A few years ago web sites consisted of only static pages with HTML and CSS for styling without any real interactive elements. The task of the server was only to send the web page to the users browser. A couple of years later Javascript became fairly popular, but it was mainly used to provide the user with effects rather than doing anything advanced. With the introduction of AJAX, Javascript became useful for the first time. With AJAX you could load content without the browser refreshing.

Even later it became popular with so called “Single page applications” where the site is dynamically loaded with the use of Javascript. Using this technology only a small part is fetched from the server and most of the operations is handled by the client.

However this technology is not without problems. In this study the problems of single applications will be walked through and compared with the use of Isomorphic Javascript.

Objectives

In this study we will investigate the uses for Isomorphic Javascript, to discover if there are any benefits or negative sides of the technology. We will compare the performance, security questions and other relevant aspects.

It will also look into the industry and determine whether or not the web development industry has been affected by the use of this technology. Does it increase the workflow? Can programmers write applications faster and get deploy websites in a faster pace?

In this study we will also dive into the frameworks that are currently using Isomorphic Javascript. Why have they chosen isomorphic javascript and how do they compare to each other frameworks?

Chapter 1

1. Introduction

Isomorphic Javascript is a fairly new technique that is increasing rapidly in popularity. In short, Isomorphic Applications are applications where the server and client shares all *or* some parts of the code. With this technology one can avoid having to write duplicate codes, but instead use the same code both backend and frontend.

There are many popular frameworks that are using this technology, such as Meteor and React which are two of the most popular frameworks using Isomorphic Javascript. We will discover more about this later in this study.

More and more companies are making the choice of using Node.js for their projects and the idea of sharing code between server and client are becoming more and more of a natural choice and might be the future of the web development business.

The idea is not all that new but it has taken a long time to development it into a working technology.

1.1 What is really an Isomorphic Application?

Traditionally Javascript were used in the client only but with the introduction of Node.js it has become a real robust server-side language. When one is talking about Isomorphic Javascript it means that the code is used both on the client and the server.

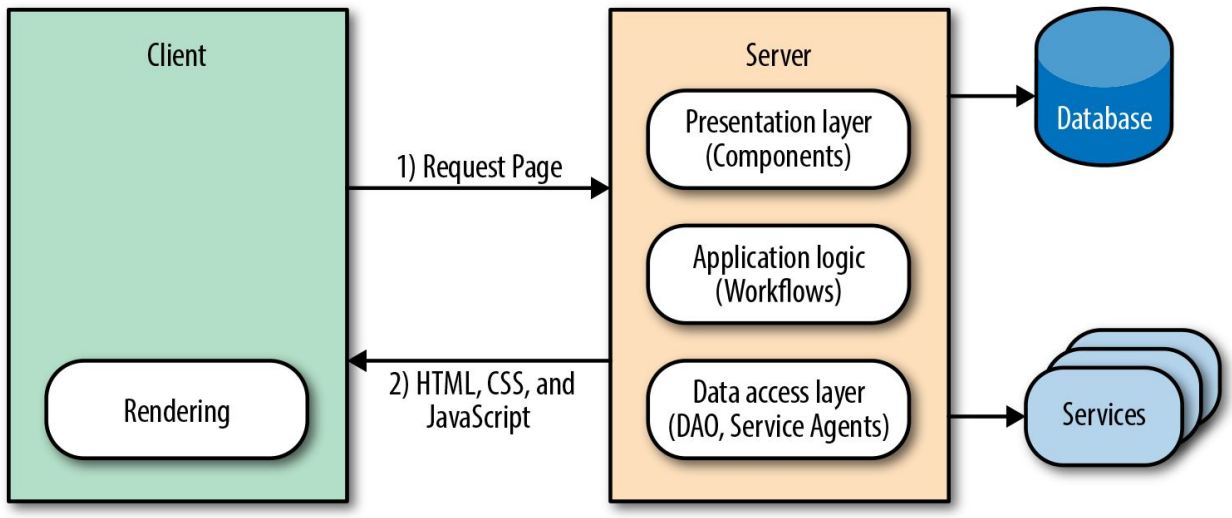

Figure 1.1. Classic web application flow

When sending a request to an isomorphic application the request is first processed by the server and then the request is send through to the application.

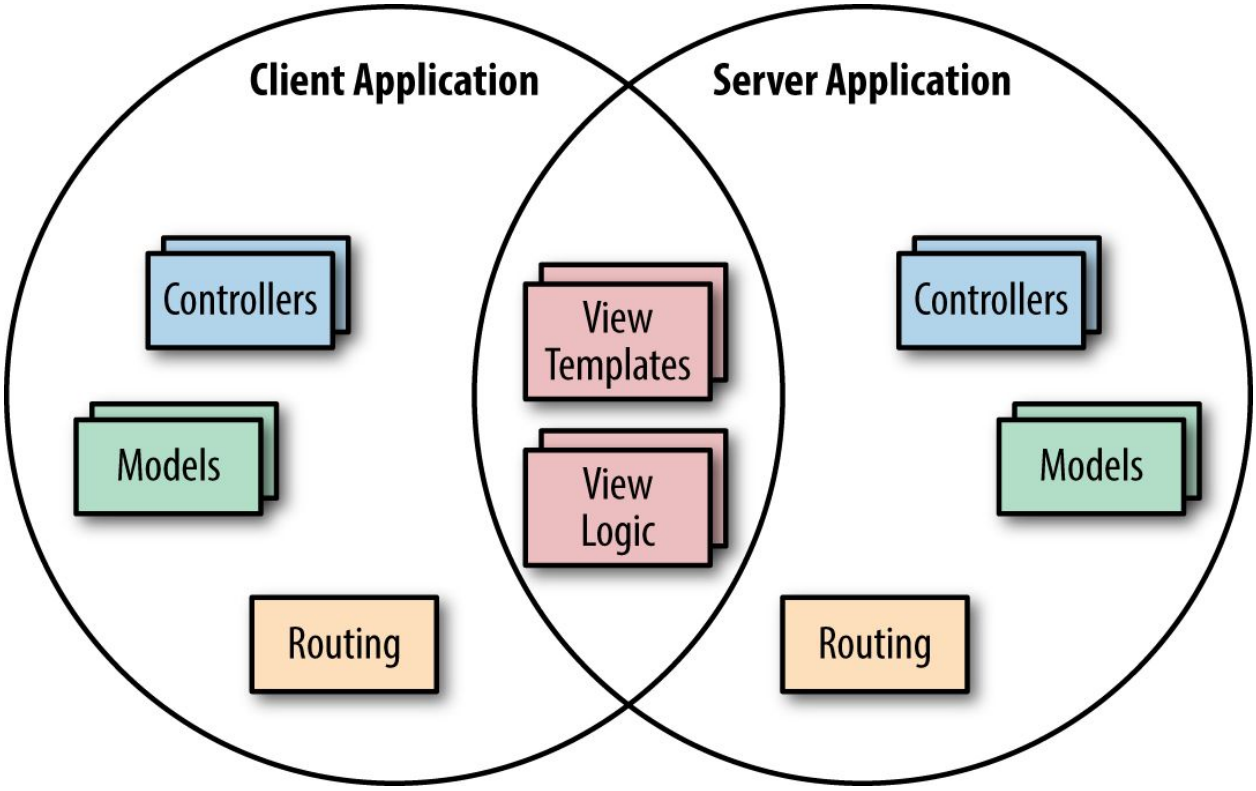

Figure 1.2. Sharing the view layer

In isomorphic applications one can either share all code between the client and server or decide which code that are shared.

In *figure 1.2* some parts of the applications is shared between the client and the server while some is specific to the environment. In *figure 1.3* (below) the whole application is shared. Generally it is a good idea to keep some parts only on the server side in order to not display information such as passwords to the public.

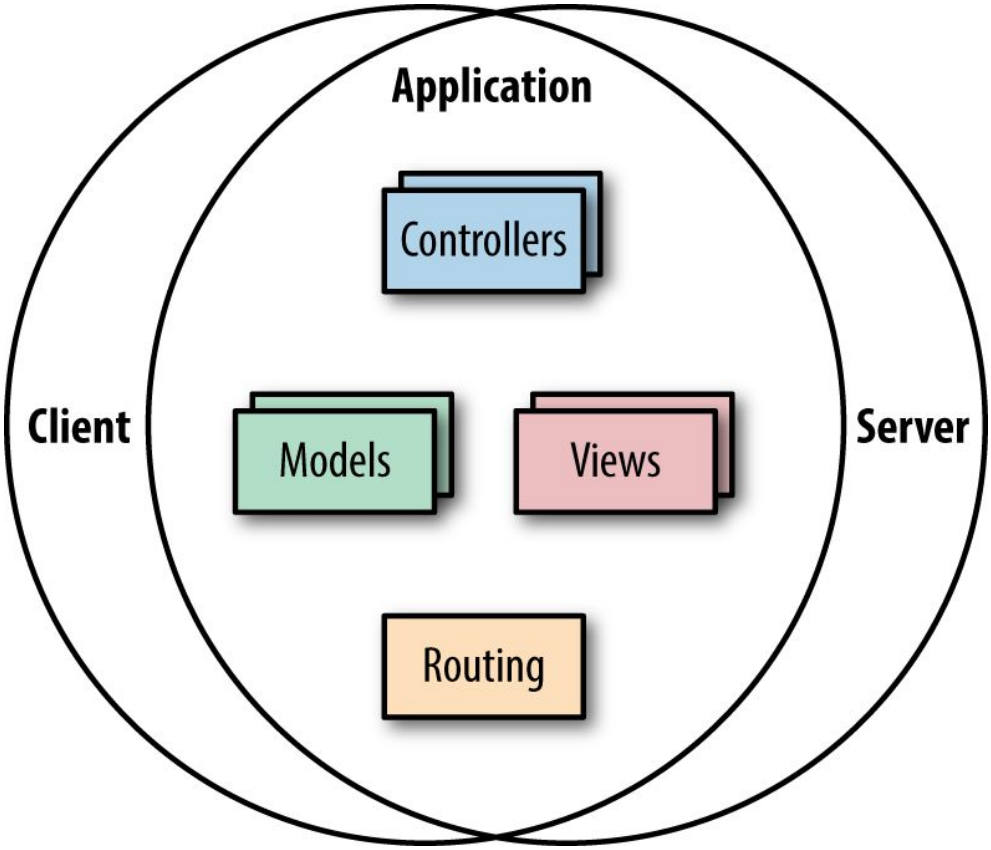

Figure 1.3. Sharing the whole application

1.2 Benefits of using this technology

If you compare the use of an application that is using isomorphic Javascript and one that is not there are many benefits in using the first. One huge benefit is that if you write for example a backend server using Python and a front end using Javascript you will sometime need to write duplicate code. In other words you will have to write the same code twice but with different syntax. If one, on the other hand would write it using the isomorphic technology you would share this code. This would mean that one could deliver web applications faster to the customers and do more work in less time.

Another big benefit is that once you have parsed the website the page loading will be much faster. [3]

But as mentioned before, since the code is shared to the client it is very important that one is careful in what parts of the code they share to ensure that no sensitive information is being accessible in the client.

Chapter 2

2. Research Questions

The purpose of this study is to dig a little deeper into the technology of isomorphic javascript.

The study will focus on the following questions:

Q1. What are the benefits of using isomorphic javascript and what restrictions and limitations is there?

The purpose is to find out if using this technology is worth it. Will it improve performance or workflow? Are there any restrictions such as style or modules? Will you be able to use all libraries that you would if not using isomorphic code?

Q2. What might we expect of the future in application development? Will there be some type of universal code that works on both client and server without using shims and such?

The motive for this question is to find out if this is the best alternative or will it be improved in the future? Maybe web pages won't render HTML in the future and there will be some kind of universal code that runs easy on both client and server.

Q3. How does this technology affect the industry?

To determine if this technology affects people that develop web applications as a profession. Has their workflow increased?

Q4. What does the term "Isomorphic" mean? Why is it called this and is it a good term?

Is the term "Isomorphic" really fitting for the technology? Are there any better terms to describe its functionality.

Q5. Is there a security risk in using Isomorphic Javascript?

Since the code is shared between client and server. Is there a risk that sensitive data is displayed to users?

Q6. What requirements is there to build an isomorphic application?

Does one need build tools to compile an isomorphic application? Does it depend on other modules in order to work? What kind of frameworks are available today?

Chapter 3

3. Method

The method used in this study is based on documentary analysis. The information used in the study is gathered from various sources. We have listed the references at the end of this study.

The project started with a literature study to research what has been discovered in the area before to get a grasp on what is important with this technology. To complete the litterature study the method of snowballing is used. The aspect of this method is to search for articles and when an article that may be a subject of this study is found one goes through the sources references in the article, in the purpose of getting the main source but also to find other related information that the article may have missed or purposely decided to leave out.

The sources used are documented in the references chapter at the end of the study.

Chapter 4

4. Literature Review

The purpose of literature review is to justify the relevance of the references used when conducting the study. It ensures that the study has not already been done and that it shares some new insight on the topic.

The main source for information for this study is being gathered from the book *"Building Isomorphic JavaScript Apps"*. It is published September 2016 by *Jason Strimpel, Maxime Najim.* Since the book is published so recently and during the time of this study the information is fresh and up to date. All the illustrations used in this study are gathered from this book.

The book had a good illustration of how one can describe the isomorphic state. See the figure below.

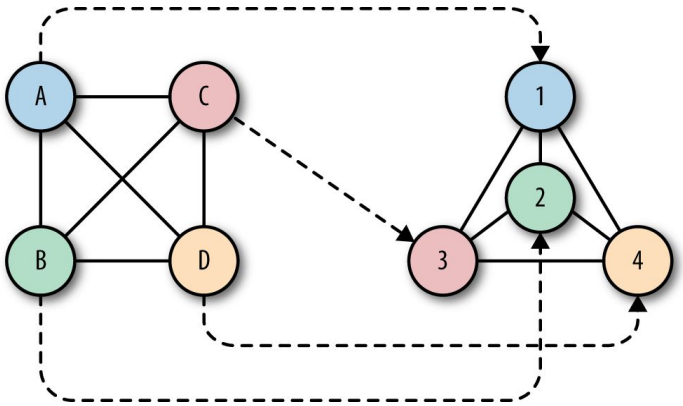

Figure 4.1. Isomorphic graphs

The graphs in figure 4.1 are isomorphic. They are isomorphic in the way that each node have the same number of nodes, with each node having the same number of edges. But the important part is that there is a connection from each node in the first graph to a corresponding node in the second graph.

Chapter 5

5. Analysis and discussion

After studying articles on the subject some interesting points and information of interest have come to light in regards to the questions for this study.

5.1 The term “isomorphic”

The term isomorphic comes from the idea of that they take on equal (iso) shape (morphosis) regardless of what environment they are used in. [4]

The subject whether or not it is a good term is widely discussed. When writing this study it showed that people who are against the term often refer to it as “universal javascript”.

The reasoning behind the term *isomorphic* is that it highlights two mathematical objects that have related or similar forms in their specific environment.

One of the developers of the *react-router* project, *Michael Jackson* suggested the term “Universal Javascript” and argues that this term describes that it is javascript that can be run “not only on servers and browsers, but on native devices and embedded architectures as well”. He also states that “isomorphism” is a mathematical term [4].

Looking at the *figure 4.1,* we get a better understanding of how the Javascript code to be run on both client and server the environments have to be isomorphic. Javascript objects that are not specific to its environment can easily be executed both on the server and client. When they are environment specific such as *req.path* on the server does the same as *window.location.pathname* does on the client side, one needs to map this together. This is called shimming. We will cover the different categories in the next part of this chapter.

5.2 Different categories of Isomorphic Javascript

There are two important categories when talking about isomorphic javascript. *Environment agnostic,* being the first and *shimmed for environment* the second.

Environment agnostic
In this category there are the pure javascript modules that only use pure javascript without any environment specific properties such as *window.* The modules in this categories will work straight away in an isomorphic application [9]. There are many modules that fit into this category such as Lodash, Moment and Handlebars. The only requirement is that they are loaded through Node’s *require()* module

loader. In order to use the module loader client side one need to use a bundler, such as Browserify or Webpack [8]. This will enable the browser to use the *require()* loader from Node. *For more information about bundlers see chapter 6*.

Shimmed for environment
In this category the modules that doesn't meet the requirements to be in the *Environment Agnostic* category, that is that they are not pure javascript. If one were to run the following command on the client:

window.location.href = 'http://google.com';

One would be redirected to Google.com, but if executing the same code on the server one would get an error stating *"window is not defined"*.

The solution to this is use *shimming*. In order to do this you extract the redirect logic to a separate module that is aware of it's environment. That way you can call the redirect module in the same code and the module would evaluate whether to use, for this example *window.location.href* or *res.writeHead.*

5.3 More trouble than it's worth

Some applications are not suited for isomorphic Javascript. When creating a smaller web application that does not require good SEO support and fast performance it might be more work than it's worth. [4]
The time spent on creating the server is unfortunately a lot more than creating a single page application.

Since the engineers who have knowledge of the technology is low compared to the ones who are experience of more common backend systems such as PHP, Python or Java it may be more trouble to maintain the server and may result in more costs.

5.4 Security risks using the technology

As mentioned in the introduction using isomorphic javascript is not without risks. One important risk to have in mind when writing the application is to avoid publishing sensitive information such as passwords to databases or API-keys in the code that would get shared with the client. Doing this may have big consequences [8]. This seems however to be the only real security issue associated with isomorphic javascript, and if one keep track of the sensitive information and keep it inaccessible to the public one should not have any security concerns.

Chapter 6

6. Results

In regard to the benefits of using isomorphic javascript this study found that there are many benefits to the technology. By using isomorphic javascript old browsers can parse the website as the application returns HTML while most single page application returns Javascript [3].

As to performance we found that the first initial load is fast and the next is even faster when comparing to systems that are not isomorphic. [3]

6.1 Search engine optimization

A big issue with single page applications is that the are not SEO-friendly, while isomorphic applications are. [1] Depending on content, most web pages today is in need of good SEO support as often you want visitors to reach your page. That is a big negative impact on single page applications. By using isomorphic javascript users can get good search engine optimization while still having both backend and frontend running Javascript.

6.2 Same functionality use the same code

Often you use the same functionality both on the front end and the backend. For example form validation. With a single page application one would first validate the form values on the client side and then validate them on the backend before completing the request. A benefit of using isomorphic javascript is that you can handle the this type of issues using the same code. [1]

6.3 Implementation

In order to implement the technology you need to have some sort of bundler. [10] Webpack is one of the most common bundler. This is used to bundle all the assets, such as javascript files into static assets.

In regards to the downside of using the technology this study found that it has a steep learning curve [4]. It is not as quick and easy to understand and learn as learning a new programming language might be.

6.4 Effect on the industry

A company that uses the technology for their web server will find that it may reduce their operating costs, since the engineers have full control of the UI. It gives a clearer view on the separation of back- and

frontend [4]. It will decrease the maintenance time since maintaining a code that is shared between the server and client only needs to be changed on one place. The risk of breaking one component is lower since it all uses the same API.

It will give the the company more exposure to the customers since search engine optimization is vastly improved. That is when comparing to a single page application built on javascript which is rendered in the client.

Many companies today want a web framework that can power large websites [4]. They also want good SEO support with an optimized page load. Isomorphic Javascript meets all of these requirements and they are becoming more and more a natural choice.

6.5 The requirements

In order to run an isomorphic javascript application there are several dependencies one needs for its project. First and foremost one need to install Node.js. Node.js is an extremely fast and open source Javascript environment which enables you to program a server using Javascript. [1]

Another requirement is to have a server that node can use. One of the most common is Express. You will also need a bundler to bundle all your files into a static file, such as Webpack. [10]

When all this is setup you will find that you need an isomorphic framework. *For more information about frameworks see chapter 6.6*.

6.6 Frameworks

Writing an isomorphic application without using a framework is a lot of work. Luckily there are many good frameworks available to choose from. Some of the common frameworks that this study will dive into are *React*, *Meteor* and *Rendr*.

React

React is one of the most popular frameworks. An important note about React is that one can use it to build an isomorphic application or to not. Choosing React for a project can be a good idea as the team behind it is Facebook [2], which would warrant some kind of quality. Given that it is very popular it is also frequently updated. It is also open source.

React is a component driven frameworks and typically renders HTML through custom HTML tags. Making an application isomorphic with React is easy. There is a method called *renderToString()* [5] where you pass in your HTML template to the server.

Netflix is using an isomorphic application for their system and it is powered by the React framework.

Meteor

Meteor is another common framework to use when building isomorphic applications. This framework differ from React in a way that it is only isomorphic [6]. One can actually use Meteor for their project and have React to render the views. Meteor is also like React, open source.

Rendr

Rendr is a small framework developed by Airbnb [7]. Rendr enables you to run backbone.js in an isomorphic way. When Airbnb developed Rendr and started using it for their application it decreased the initial page load drastically [7]. The reason for this is because instead of waiting for the browser to download the javascript to render the HTML, the HTML is instead being rendered directly from the server. As mentioned in the beginning of this chapter, isomorphic Javascript has the benefit of decreasing the load time and improves the performance.

Chapter 7

7. Conclusion

In this study we have evaluated and compared the use of isomorphic javascript and single page applications. We have found that using isomorphic javascript has proved to be beneficial in many ways. It improves performance and requires less code to be written. It is however a new technology and there are still a lot of bugs and problems with it. There are many frameworks using the technology but even they are not as bug free as one might have expected. Hopefully in the near future we will see isomorphic application frameworks that are easily implemented and without any issues.

To summarize this study, we found that:

Using this technology is well worth it. It may take a bit longer to set up than creating an application without the technology. However when it is setup it will require less time on the developing part. It will also increase the performance, especially the load time and it will improve SEO performance drastically.

In regard to question *Q1*, we found some limitations in using the technology, the main one being that it is very dependant on other modules to work and takes time to set up. Sometimes it is simply not worth the effort it takes to set it up (*see chapter 5.3)*.

For *Q2* we did not find any real answers so the question would remain unanswered and a possible question for future works. However taking account the references we have used in this study, all of them agrees on the benefits of using isomorphic javascript and our opinion is that in the near future the usage will increase.

A motive for this study was to find out how it affected the industry *(see question Q3)*. In this study when taking to account the ability to reuse code and not having to write the same functionality twice, we conclude that deploying web applications will be quicker using the technology and thus affects the workflow. This, in turn will reduce the time spent on each project. We did however not find any real concrete studies to validate this results, and it seems that no studies on this question have been made.

We also wanted to find what the term isomorphic means (*see question Q4)*. We found that the term means that the application is (iso) shape (morphosis) regardless of the environment. *(see chapter 5.1)*

We uncovered some security risks associated with the technology (*see chapter 5.4).* But according to the findings discovered there is not an immediate security risk in using Isomorphic applications, while keeping in mind the issues referred to in this study. So to sum up and as an answer to *Q5*, we conclude that there are no real security threats apart from sharing sensitive information to the public.

For the last question we wanted to know what the requirements are for using an isomorphic application. Our research found that it is pretty advanced to set up and takes more time than an application not using it. It also requires a bundler *(see chapter 6.3)*.

Chapter 8

8. Future work

During this study I discovered some more interesting topics related to the technology. In this study I have looked mainly into the uses of isomorphic applications and their benefits. For future work it would be interesting with more information about the bundlers and how they work, as they are an important part of creating an isomorphic application.

The purpose being to discover the differences between the most common bundlers and how they work “behind the scenes”.

Since the technology is so new there is updates coming out very often and for future work there may be changes that have big impact on the study. It is also likely that there are new features to write and study about.

A good purpose for future work is a follow up study done in a few years to evaluate what kind of impact this has had on the industry after this study was done. If it is still being used and if it is, what changes have been made.

It might also be interesting to get statistical data for how the use various from different countries, regions or if there is more use in big companies than small.

References

[1] http://xbsoftware.com/blog/isomorphic-react-spa-benefits-seo/

[2] http://www.infoworld.com/article/2608181/javascript/react--making-faster--smoother-uis-for-data-driven-web-apps.html

[3] https://www.lullabot.com/articles/what-is-an-isomorphic-application

[4] Jason Strimpel, Maxime Najim, “Building Isomorphic JavaScript Apps: From Concept to Implementation to Real-World Solutions”

[5] https://facebook.github.io/react/docs/top-level-api.html#reactdomserver

[6] https://guide.meteor.com/structure.html

[7] http://nerds.airbnb.com/weve-launched-our-first-nodejs-app-to-product/

[8] https://www.sumologic.com/blog-devops/what-are-isomorphic-applications/

[9] http://www.oreilly.com/pub/e/3009

[10] The Pain and the Joy of creating isomorphic apps in ReactJS, https://reactjsnews.com/isomorphic-react-in-real-life

Wordlist

SEO
Search engine optimization is the process of improving the numbers of visitors a site get by getting a high appearance on the search engines.

Ajax
Ajax stands for *Asynchronous Javascript and XML*. It is used to communicate between the browser and the server.

Bundler
A tool used for combining all files and dependencies into one file.

Shimming
Shimming is when you use a library to intercept API calls and redirect them to another.

UI
User interface.