AJAX

Asynchronous JavaScript and XML

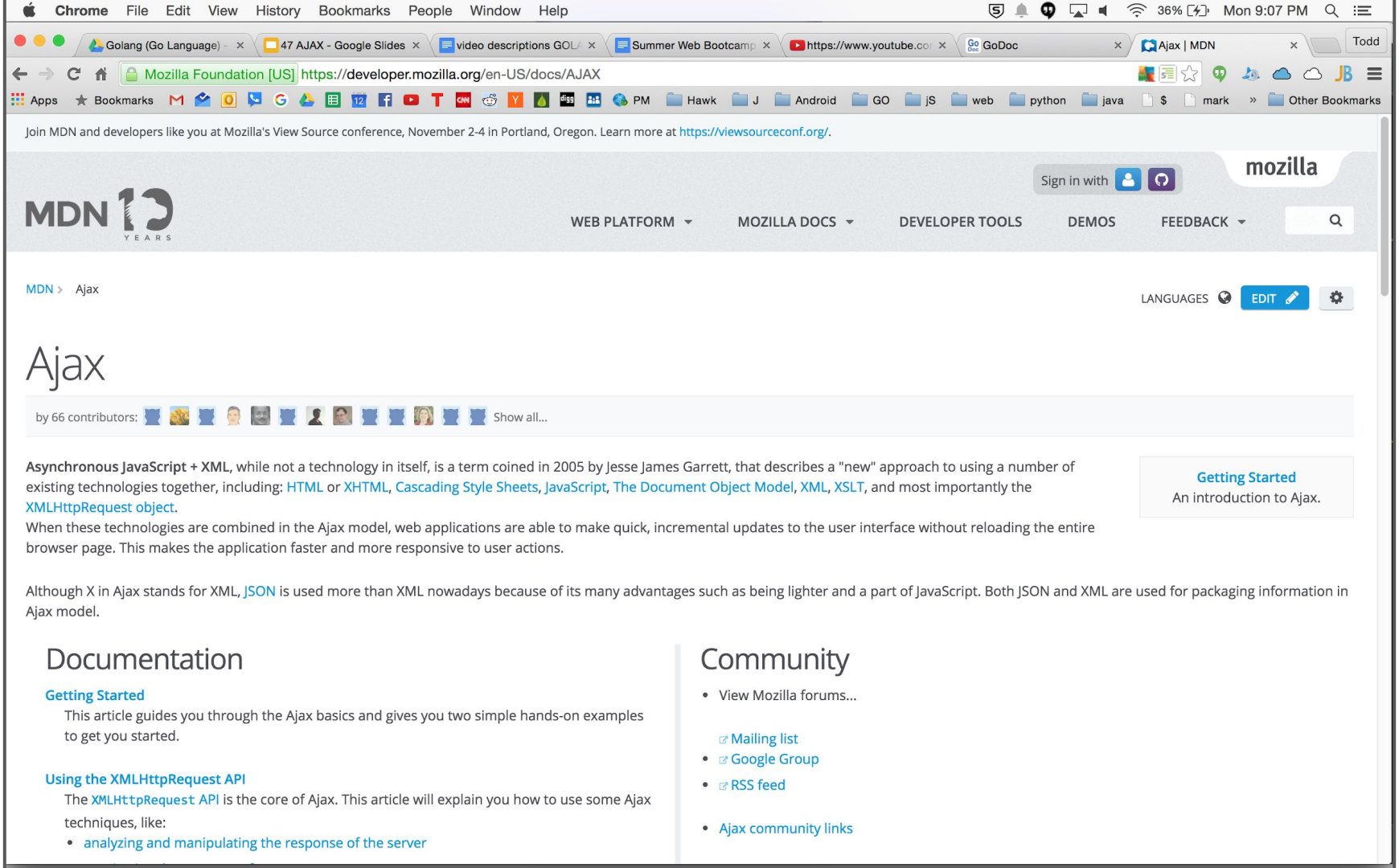
What's AJAX?

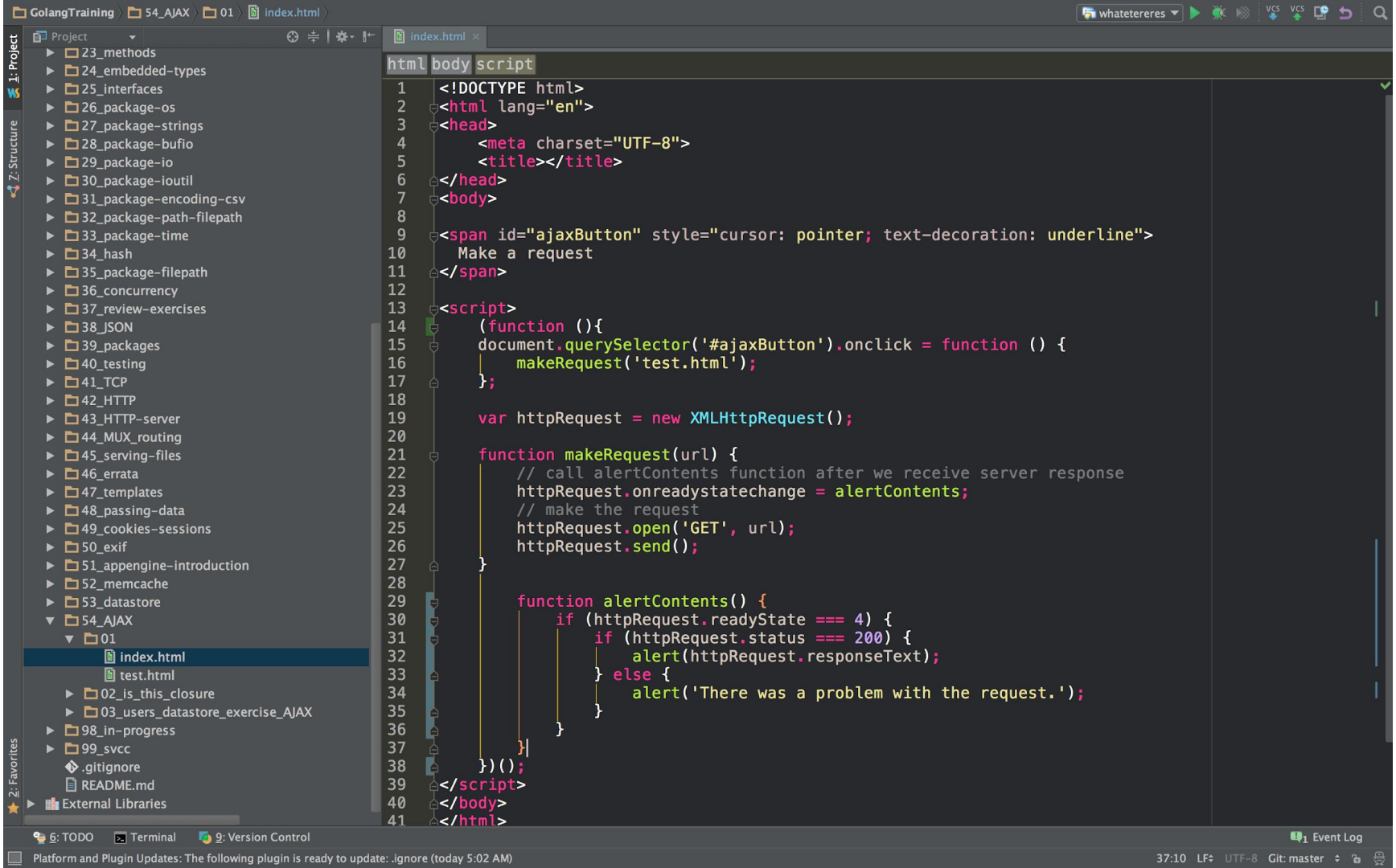
AJAX stands for Asynchronous JavaScript and XML. In a nutshell, it is the use of the `XMLHttpRequest` object to communicate with server-side scripts. It can send as well as receive information in a variety of formats, including JSON, XML, HTML, and even text files. AJAX's most appealing characteristic, however, is its "asynchronous" nature, which means it can do all of this without having to refresh the page. This lets you update portions of a page based upon user events.

The two features in question are that you can:

- Make requests to the server without reloading the page
- Receive and work with data from the server

https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started


Chrome File Edit View History Bookmarks People Window Help
36% Mon 9:07 PM
Golang (Go Language) -
47 AJAX - Google Slides
video descriptions GOL
Summer Web Bootcamp
https://www.youtube.co
GoDoc
Ajax | MDN
Todd
Mozilla Foundation [US] https://developer.mozilla.org/en-US/docs/AJAX
Apps Bookmarks PM Hawk J Android GO jS web python java $ mark Other Bookmarks
Join MDN and developers like you at Mozilla's View Source conference, November 2-4 in Portland, Oregon. Learn more at https://viewsourceconf.org/.
Sign in with
mozilla
MDN 10 YEARS
WEB PLATFORM
MOZILLA DOCS
DEVELOPER TOOLS
DEMOS
FEEDBACK
MDN › Ajax
LANGUAGES
EDIT
Ajax
by 66 contributors:
Show all...
Asynchronous JavaScript + XML, while not a technology in itself, is a term coined in 2005 by Jesse James Garrett, that describes a "new" approach to using a number of existing technologies together, including: HTML or XHTML, Cascading Style Sheets, JavaScript, The Document Object Model, XML, XSLT, and most importantly the XMLHttpRequest object.
When these technologies are combined in the Ajax model, web applications are able to make quick, incremental updates to the user interface without reloading the entire browser page. This makes the application faster and more responsive to user actions.
Getting Started
An introduction to Ajax.
Although X in Ajax stands for XML, JSON is used more than XML nowadays because of its many advantages such as being lighter and a part of JavaScript. Both JSON and XML are used for packaging information in Ajax model.
Documentation
Getting Started
This article guides you through the Ajax basics and gives you two simple hands-on examples to get you started.
Using the XMLHttpRequest API
The XMLHttpRequest API is the core of Ajax. This article will explain you how to use some Ajax techniques, like:
analyzing and manipulating the response of the server
Community
View Mozilla forums...
Mailing list
Google Group
RSS feed
Ajax community links


GolangTraining
54_AJAX
01
index.html
whatetereres
Project
23_methods
24_embedded-types
25_interfaces
26_package-os
27_package-strings
28_package-bufio
29_package-io
30_package-ioutil
31_package-encoding-csv
32_package-path-filepath
33_package-time
34_hash
35_package-filepath
36_concurrency
37_review-exercises
38_JSON
39_packages
40_testing
41_TCP
42_HTTP
43_HTTP-server
44_MUX_routing
45_serving-files
46_errata
47_templates
48_passing-data
49_cookies-sessions
50_exif
51_appengine-introduction
52_memcache
53_datastore
54_AJAX
01
index.html
test.html
02_is_this_closure
03_users_datastore_exercise_AJAX
98_in-progress
99_svcc
.gitignore
README.md
External Libraries
index.html
html
body
script
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title></title>
6 </head>
7 <body>
8
9
10 Make a request
11
12
13 <script>
14 (function (){
15 document.querySelector('#ajaxButton').onclick = function () {
16 makeRequest('test.html');
17 };
18
19 var httpRequest = new XMLHttpRequest();
20
21 function makeRequest(url) {
22 // call alertContents function after we receive server response
23 httpRequest.onreadystatechange = alertContents;
24 // make the request
25 httpRequest.open('GET', url);
26 httpRequest.send();
27 }
28
29 function alertContents() {
30 if (httpRequest.readyState === 4) {
31 if (httpRequest.status === 200) {
32 alert(httpRequest.responseText);
33 } else {
34 alert('There was a problem with the request.');
35 }
36 }
37 }
38 })();
39 </script>
40 </body>
41 </html>
6: TODO
Terminal
9: Version Control
Event Log
Platform and Plugin Updates: The following plugin is ready to update: .ignore (today 5:02 AM)
37:10
LF
UTF-8
Git: master
1: Project
7: Structure
2: Favorites

```
<script>
    (function (){
    document.querySelector('#ajaxButton').onclick = function () {
        makeRequest('test.html');
    };

    var httpRequest = new XMLHttpRequest();

    function makeRequest(url) {
        // call alertContents function after we receive server response
        httpRequest.onreadystatechange = alertContents;
        // make the request
        httpRequest.open('GET', url);
        httpRequest.send();
    }

        function alertContents() {
            if (httpRequest.readyState === 4) {
                if (httpRequest.status === 200) {
                    alert(httpRequest.responseText);
                } else {
                    alert('There was a problem with the request.');
                }
            }
        }
    })();
</script>
```

Note 2: If you do not set header `Cache-Control: no-cache` the browser will cache the response and never re-submit the request, making debugging "challenging." You can also append an always-diferent aditional GET parameter, like the timestamp or a random number (see bypassing the cache)

Note 3: If the `httpRequest` variable is used globally, competing functions calling `makeRequest()` may overwrite each other, causing a race condition. Declaring the `httpRequest` variable local to a closure containing the AJAX functions prevents the race condition.

documentation

make an HTTP request

```
var xhr = new XMLHttpRequest();
```

In order to make an HTTP request to the server using JavaScript, you need an instance of a class that provides this functionality.

https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started

```
var xhr = new XMLHttpRequest();
```

xhr.onreadystatechange = nameOfTheFunction;

Tell the HTTP request object which JavaScript function will handle processing the response from the server. Set the **onreadystatechange** property of the object to the name of the JavaScript function that should be called when the state of the request changes.

https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started

```
var xhr = new XMLHttpRequest();
```

```
xhr.onreadystatechange = function(){
    // process the server response
};
```

Tell the HTTP request object which JavaScript function will handle processing the response from the server. Set the **onreadystatechange** property of the object to the name of the JavaScript function that should be called when the state of the request changes.

https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started

```
var xhr = new XMLHttpRequest();
```

```
  xhr.onreadystatechange = function(){
      // process the server response
};
xhr.open('GET','http://www.example.org/some.file', true);
xhr.send(null);
```

After you've declared what will happen as soon as you receive the response, you need to actually make the request. You need to call the **open()** and **send()** methods of the HTTP request class

https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started

```
var xhr = new XMLHttpRequest();
```

```
xhr.onreadystatechange = function(){
    // process the server response
};
xhr.open('GET','http://www.example.org/some.file', true);
xhr.send(null);
```

After you've
need to

- The first parameter of the call to `open()` is the HTTP request method – GET, POST, HEAD or any other method you want to use and that is supported by your server. Keep the method capitalized as per the HTTP standard; otherwise some browsers (like Firefox) might not process the request. For more information on the possible HTTP request methods you can check the W3C specs.
- The second parameter is the URL of the page you're requesting. As a security feature, you cannot call pages on 3rd-party domains. Be sure to use the exact domain name on all of your pages or you will get a "permission denied" error when you call `open()`. A common pitfall is accessing your site by `domain.tld`, but attempting to call pages with `www.domain.tld`. If you really need to send a request to another domain, see HTTP access control.
- The optional third parameter sets whether the request is asynchronous. If `TRUE` (the default), the execution of the JavaScript function will continue while the response of the server has not yet arrived. This is the A in AJAX.

```
var xhr = new XMLHttpRequest();
```

```
xhr.onreadystatechange = function(){
    // process the server response
};
xhr.open('GET','http://www.example.org/some.file', true);
xhr.send(null);
```

The parameter to the `send()` method can be any data you want to send to the server if `POST`-ing the request. Form data should be sent in a format that the server can parse easily. This can be as a query string, like:

```
"name=value&anothername="+encodeURIComponent(myVar)+"&so=on"
```

or in several other formats, including JSON, SOAP, etc.

Note that if you want to `POST` data, you may have to set the MIME type of the request. For example, use the following line before calling `send()` for form data sent as a query string:

```
httpRequest.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
```

EAD or any other method you want to
TP standard; otherwise some browsers
TTP request methods you can check the

e, you cannot call pages on 3rd-party
a "permission denied" error when you
o call pages with `www.domain.tld`. If you

default), the execution of the JavaScript
e A in AJAX.

handle the server response

Remember that when you were sending the request, you provided the name of a JavaScript function that is designed to handle the response.

FYI, MDN calls their object httpRequest while I was calling it xhr

```
httpRequest.onreadystatechange = nameOfTheFunction;
```

Let's see what this function should do. First, the function needs to check for the state of the request. If the state has the value of 4, that means that the full server response has been received and it's OK for you to continue processing it.

```
if (httpRequest.readyState === 4) {
    // everything is good, the response is received
} else {
    // still not ready
}
```

The full list of the `readyState` values is as follows:

- 0 (uninitialized)
- 1 (loading)
- 2 (loaded)
- 3 (interactive)
- 4 (complete)

https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started

The next thing to check is the response code of the HTTP server response. All the possible codes are listed on the W3C site. In the following example, we differentiate between a successful or unsuccessful AJAX call by checking for a `200 OK` response code.

```
if (httpRequest.status === 200) {
    // perfect!
} else {
    // there was a problem with the request,
    // for example the response may contain a 404 (Not Found)
    // or 500 (Internal Server Error) response code
}
```

https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started

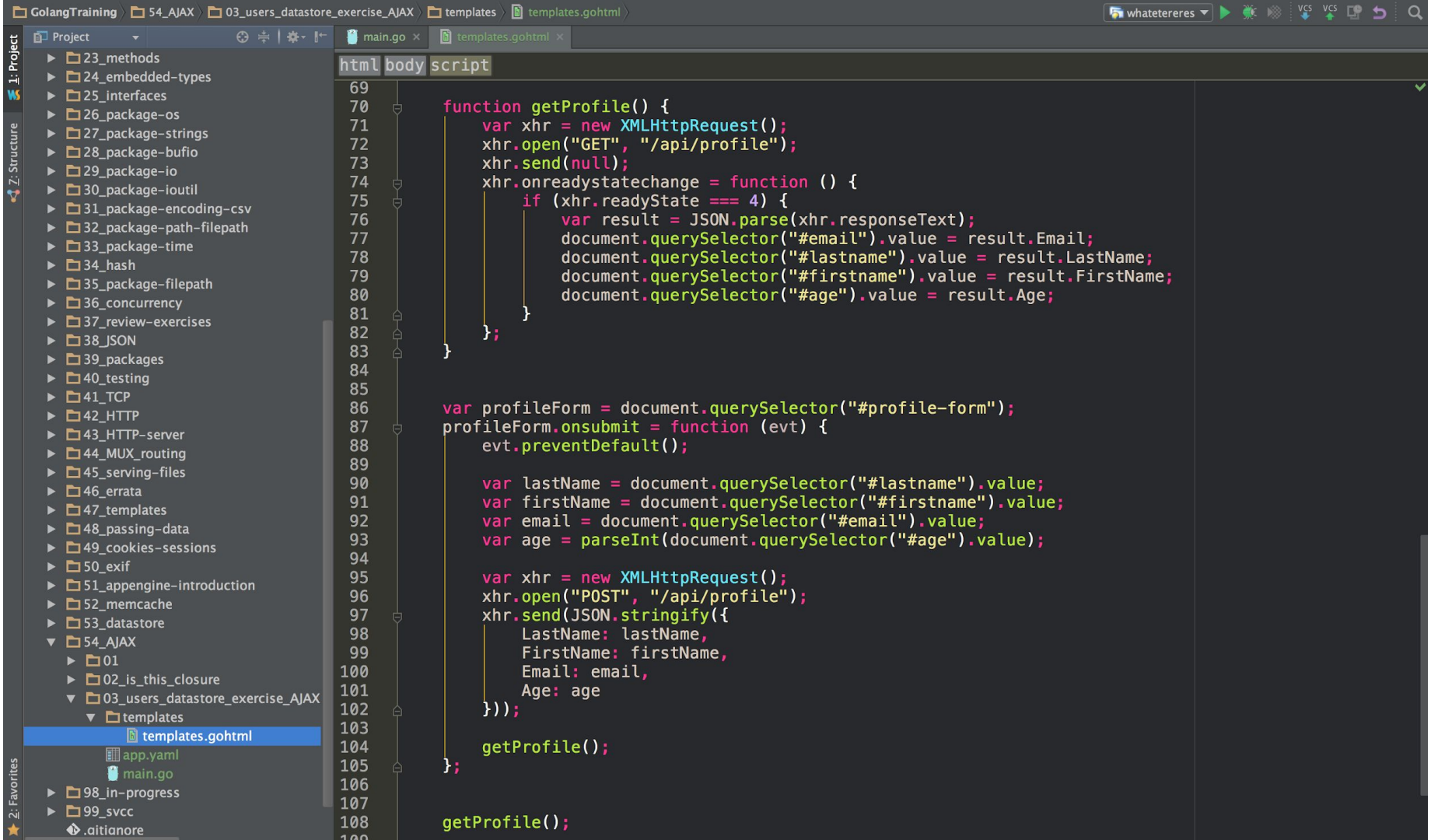
Now after you've checked the state of the request and the HTTP status code of the response, it's up to you to do whatever you want with the data the server has sent to you. You have two options to access that data:

- `httpRequest.responseText` – returns the server response as a string of text
- `httpRequest.responseXML` – returns the response as an `XMLDocument` object you can traverse using the JavaScript DOM functions

Note that the steps above are only valid if you used an asynchronous request (third parameter of `open()` was set to `true`). If you used an **synchronous** request you don't need to specify a function, you can access the data return by the server right after calling `send()`, because the script will stop and wait for the server answer.

https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started

App Engine & AJAX


GolangTraining
54_AJAX
03_users_datastore_exercise_AJAX
templates
templates.gohtml
whatetereres
Project
main.go
templates.gohtml
html
body
script
23_methods
24_embedded-types
25_interfaces
26_package-os
27_package-strings
28_package-bufio
29_package-io
30_package-ioutil
31_package-encoding-csv
32_package-path-filepath
33_package-time
34_hash
35_package-filepath
36_concurrency
37_review-exercises
38_JSON
39_packages
40_testing
41_TCP
42_HTTP
43_HTTP-server
44_MUX_routing
45_serving-files
46_errata
47_templates
48_passing-data
49_cookies-sessions
50_exif
51_appengine-introduction
52_memcache
53_datastore
54_AJAX
01
02_is_this_closure
03_users_datastore_exercise_AJAX
templates
templates.gohtml
app.yaml
main.go
98_in-progress
99_svcc
.gitignore
1: Project
Z: Structure
2: Favorites
69
70 function getProfile() {
71 var xhr = new XMLHttpRequest();
72 xhr.open("GET", "/api/profile");
73 xhr.send(null);
74 xhr.onreadystatechange = function () {
75 if (xhr.readyState === 4) {
76 var result = JSON.parse(xhr.responseText);
77 document.querySelector("#email").value = result.Email;
78 document.querySelector("#lastname").value = result.LastName;
79 document.querySelector("#firstname").value = result.FirstName;
80 document.querySelector("#age").value = result.Age;
81 }
82 };
83 }
84
85
86 var profileForm = document.querySelector("#profile-form");
87 profileForm.onsubmit = function (evt) {
88 evt.preventDefault();
89
90 var lastName = document.querySelector("#lastname").value;
91 var firstName = document.querySelector("#firstname").value;
92 var email = document.querySelector("#email").value;
93 var age = parseInt(document.querySelector("#age").value);
94
95 var xhr = new XMLHttpRequest();
96 xhr.open("POST", "/api/profile");
97 xhr.send(JSON.stringify({
98 LastName: lastName,
99 FirstName: firstName,
100 Email: email,
101 Age: age
102 }));
103
104 getProfile();
105 };
106
107
108 getProfile();

```
69
70      function getProfile() {
71          var xhr = new XMLHttpRequest();
72          xhr.open("GET", "/api/profile");
73          xhr.send(null);
74          xhr.onreadystatechange = function () {
75              if (xhr.readyState === 4) {
76                  var result = JSON.parse(xhr.responseText);
77                  document.querySelector("#email").value = result.Email;
78                  document.querySelector("#lastname").value = result.LastName;
79                  document.querySelector("#firstname").value = result.FirstName;
80                  document.querySelector("#age").value = result.Age;
81              }
82          };
83      }
84
```

```
85
86      var profileForm = document.querySelector("#profile-form");
87      profileForm.onsubmit = function (evt) {
88          evt.preventDefault();
89
90          var lastName = document.querySelector("#lastname").value;
91          var firstName = document.querySelector("#firstname").value;
92          var email = document.querySelector("#email").value;
93          var age = parseInt(document.querySelector("#age").value);
94
95          var xhr = new XMLHttpRequest();
96          xhr.open("POST", "/api/profile");
97          xhr.send(JSON.stringify({
98              LastName: lastName,
99              FirstName: firstName,
100             Email: email,
101             Age: age
102         }));
103
104         getProfile();
105     };
106
107
108     getProfile();
```

```go
type Profile struct {
    Email     string
    FirstName string
    LastName  string
    Age       int
}

func init() {
    router := httprouter.New()
    router.GET("/", showIndex)
    router.GET("/profile", showProfile)
    router.GET("/api/profile", getAPIProfile)
    router.POST("/api/profile", updateAPIProfile)
    http.Handle("/", router)
}

func getAPIProfile(res http.ResponseWriter, req *http.Request, params httprouter.Params) {
    ctx := appengine.NewContext(req)
    u := user.Current(ctx)
    key := datastore.NewKey(ctx, "Profile", u.Email, 0, nil)
    var profile Profile
    err := datastore.Get(ctx, key, &profile)
    if err != nil {
        profile.Email = u.Email
    }
    json.NewEncoder(res).Encode(profile)
}
```

```
41
42  func updateAPIProfile(res http.ResponseWriter, req *http.Request, params httprouter.Params) {
43      var profile Profile
44      json.NewDecoder(req.Body).Decode(&profile)
45
46      ctx := appengine.NewContext(req)
47      u := user.Current(ctx)
48      key := datastore.NewKey(ctx, "Profile", u.Email, 0, nil)
49      _, err := datastore.Put(ctx, key, &profile)
50      if err != nil {
51          http.Error(res, err.Error(), 500)
52      }
53  }
```

```go
func showIndex(res http.ResponseWriter, req *http.Request, params httprouter.Params) {
    http.Redirect(res, req, "/profile", 302)
}

func showProfile(res http.ResponseWriter, req *http.Request, params httprouter.Params) {
    tpl, err := template.ParseFiles("templates/templates.gohtml")
    if err != nil {
        panic(err)
    }

    err = tpl.ExecuteTemplate(res, "templates.gohtml", nil)
    if err != nil {
        http.Error(res, err.Error(), 500)
    }
}
```

exercise solution AJAX

Introducing the Accelerated Mobile Pages Project, for a faster, open mobile web

October 7, 2015

Smartphones and tablets have revolutionized the way we access information, and today people consume a tremendous amount of news on their phones. Publishers around the world use the mobile web to reach these readers, but the experience can often leave a lot to be desired. Every time a webpage takes too long to load, they lose a reader—and the opportunity to earn revenue through advertising or subscriptions. That's because advertisers on these websites have a hard time getting consumers to pay attention to their ads when the pages load so slowly that people abandon them entirely.

Today, after discussions with publishers and technology companies around the world, we're announcing a new open source initiative called Accelerated Mobile Pages, which

ACCELERATED
MOBILE
PAGES
PROJECT

CONTACT US

VIEW FAQ

Instant. Everywhere.

For many, reading on the mobile web is a slow, clunky and frustrating experience - but it doesn't have to be that way. The Accelerated Mobile Pages (AMP) Project is an open source initiative that embodies the vision that publishers can create mobile optimized content once and have it load instantly everywhere.

An architectural framework built for speed

ACCELERATED
MOBILE
PAGES
PROJECT

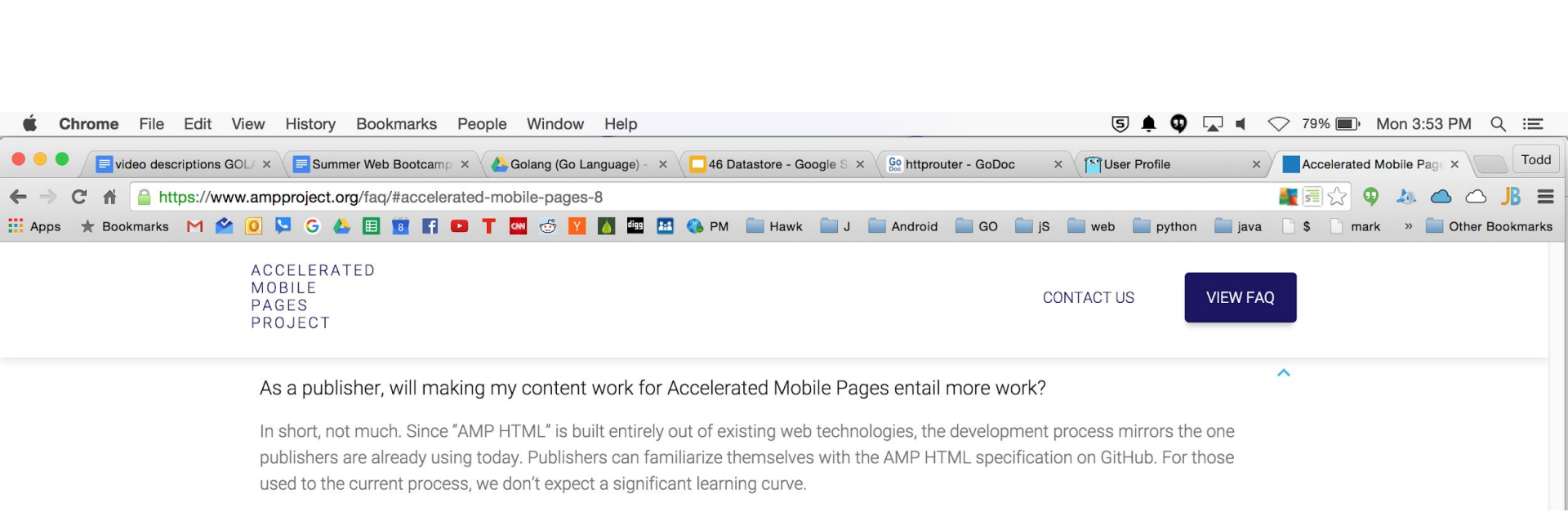
CONTACT US VIEW FAQ

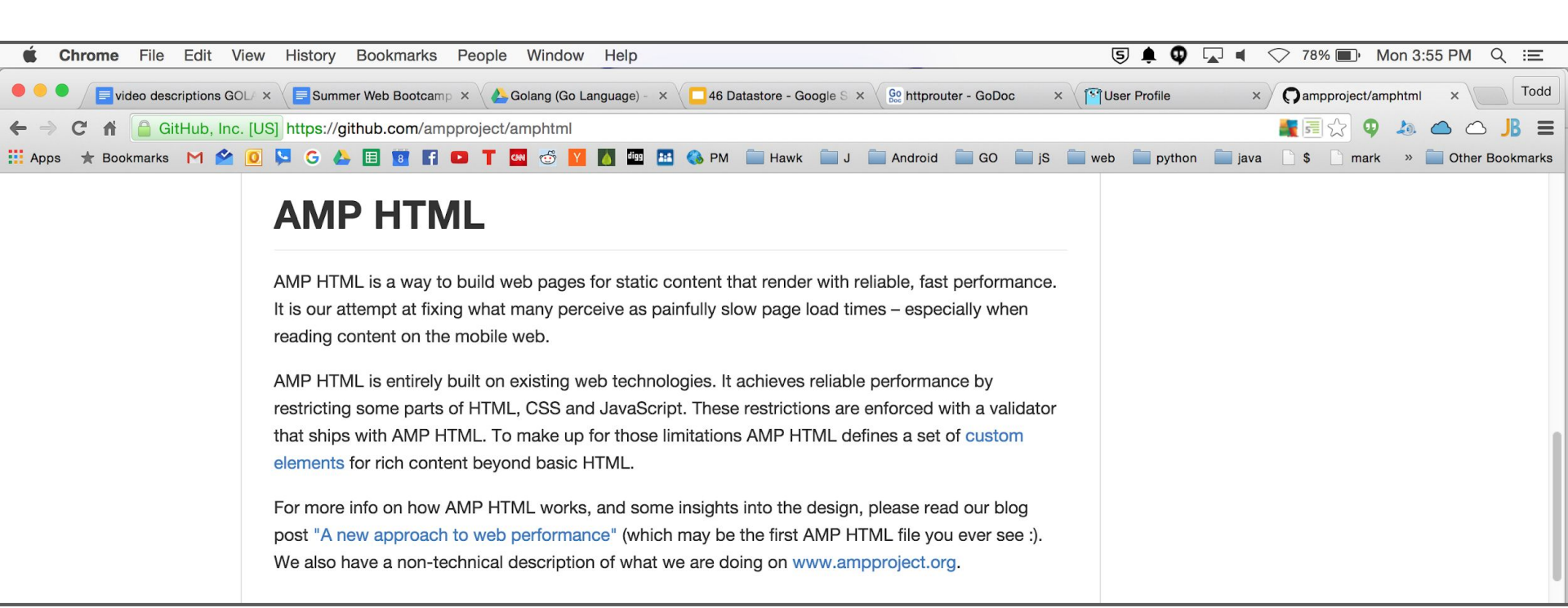
How do Accelerated Mobile Pages work?

Accelerated Mobile Pages are just like any other HTML page, but with a limited set of allowed technical functionality that is defined and governed by the open source AMP spec. Just like all web pages, Accelerated Mobile Pages will load in any modern browser or app webview. AMP files take advantage of various technical and architectural approaches that prioritize speed to provide a faster experience for users. The goal is not to homogenize how content looks and feels, but instead to build a more common technical core between pages that speeds up load times.

In addition, AMP files can be cached in the cloud in order to reduce the time content takes to get a user's mobile device. Under this type of framework, publishers continue to control their content, but platforms can easily cache or mirror the content for optimal delivery speed users. Google has stated that it will provide a cache that can be used by anyone at no cost, though the cache (Google's or otherwise) is not required. Other companies may build their own cache as well.

In summary, the goal is that that the combination of limited technical functionality with a distribution system built around caching will lead to better performing pages, and increased audience development for publishers.


Chrome File Edit View History Bookmarks People Window Help
79% Mon 3:53 PM
video descriptions GOL
Summer Web Bootcamp
Golang (Go Language)
46 Datastore - Google S
httprouter - GoDoc
User Profile
Accelerated Mobile Pag
Todd
https://www.ampproject.org/faq/#accelerated-mobile-pages-8
Apps
Bookmarks
PM
Hawk
J
Android
GO
jS
web
python
java
$
mark
Other Bookmarks
ACCELERATED
MOBILE
PAGES
PROJECT
CONTACT US
VIEW FAQ
As a publisher, will making my content work for Accelerated Mobile Pages entail more work?
In short, not much. Since "AMP HTML" is built entirely out of existing web technologies, the development process mirrors the one publishers are already using today. Publishers can familiarize themselves with the AMP HTML specification on GitHub. For those used to the current process, we don't expect a significant learning curve.


Chrome File Edit View History Bookmarks People Window Help
78% Mon 3:55 PM
video descriptions GOL
Summer Web Bootcamp
Golang (Go Language)
46 Datastore - Google S
httprouter - GoDoc
User Profile
ampproject/amphtml
Todd
GitHub, Inc. [US] https://github.com/ampproject/amphtml
Apps Bookmarks PM Hawk J Android GO jS web python java $ mark Other Bookmarks
AMP HTML
AMP HTML is a way to build web pages for static content that render with reliable, fast performance. It is our attempt at fixing what many perceive as painfully slow page load times – especially when reading content on the mobile web.
AMP HTML is entirely built on existing web technologies. It achieves reliable performance by restricting some parts of HTML, CSS and JavaScript. These restrictions are enforced with a validator that ships with AMP HTML. To make up for those limitations AMP HTML defines a set of custom elements for rich content beyond basic HTML.
For more info on how AMP HTML works, and some insights into the design, please read our blog post "A new approach to web performance" (which may be the first AMP HTML file you ever see :). We also have a non-technical description of what we are doing on www.ampproject.org.

https://www.ampproject.org/how-it-works/

One thing we realized early on is that many performance issues are caused by the integration of multiple JavaScript libraries, tools, embeds, etc. into a page. This isn't saying that JavaScript immediately leads to bad performance, but once arbitrary JavaScript is in play, most bets are off because anything could happen at any time and it is hard to make any type of performance guarantee. With this in mind we made the tough decision that AMP HTML documents would not include any author-written JavaScript, nor any third-party scripts.

One thing we realized early on is that many performance issues are caused by the integration of multiple JavaScript libraries, tools, embeds, etc. into a page. This isn't saying that JavaScript immediately leads to bad performance, but once arbitrary JavaScript is in play, most bets are off because anything could happen at any time and it is hard to make any type of performance guarantee. With this in mind we made the tough decision that AMP HTML documents would not include any author-written JavaScript, nor any third-party scripts.

JavaScript is the core building block for advanced web apps, but for static content it may not always be required: for a headline, some text and an image you do not need JS. Looking further into the content being created on the web nowadays, there are, however, things like lightboxes, various embeds, polls, quizzes and other interactive features that cannot easily be implemented without JavaScript. But the web platform has a great solution: custom elements and web components. AMP components may have JavaScript under the hood, but it is coordinated with other AMP components, so its composition into the page doesn't cause performance degradation. If AMP HTML provided the right custom elements, we might be able to get rid of arbitrary JavaScript for these documents altogether.

So, AMP HTML comes with strong limitations on JS. What about CSS? AMP HTML loves CSS! We do add some best practice enforcement, but we do want AMP HTML documents to look like their authors intended them to, and so allowing extensive styling is core to the platform.