

#WWDC18

Getting and Using a MapKit JS Key

Session 508

Eric Gelin, MapKit JS

Agenda

Learn how MapKit JS authorization works

Create a MapKit JS Key

Create an authorization token

Make a map

Agenda

Learn how MapKit JS authorization works

Create a MapKit JS Key

Create an authorization token

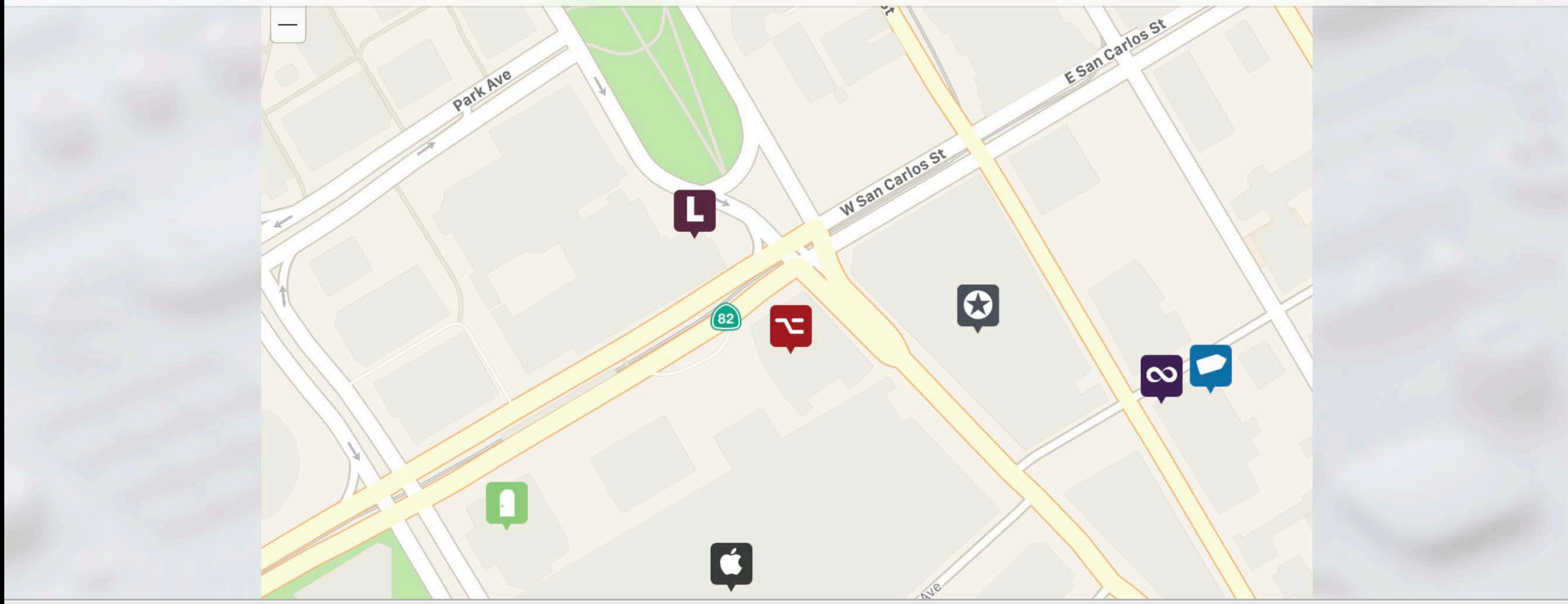
Make a map

The web is plaintext 🙌

apple.com

WWDC18

Overview Scholarships Attending More



Map showing the WWDC18 venue area, including streets like Park Ave, W San Carlos St, and E San Carlos St. The map features several icons: a purple 'L' for transit, a red shopping cart, a black star, a purple infinity symbol, a blue speech bubble, a green bell, and a black Apple logo. A yellow highlighted path winds through the area.

53

Search

Elements Debugger Storage Network Search Timelines Resources Console

html > body > script

```
<div class="center large-10 medium-11 small-12">
  ...</div>
</div>
</section>
<section class="map overflow">
  <div class="grid">...</div>
</section>
<div class="section-details">
  <div class="section-content">...</div>
  <div class="section-content">...</div>
  <div class="section-content">...</div>
</div>
```

Styles Computed Node Layers

Active Focus Hover Visited

Style Attribute { }

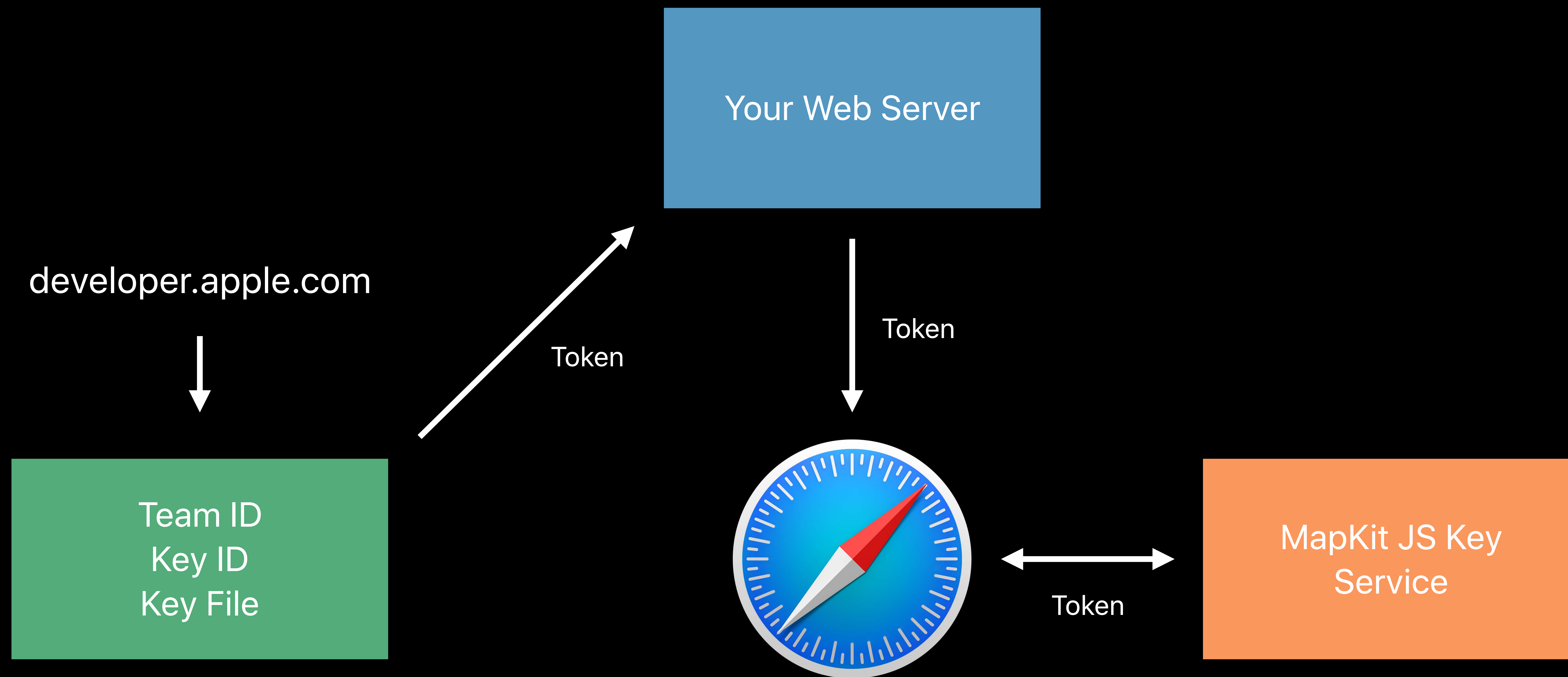
*, ::before, ::after { global-alt.dist.css:10428
~~-webkit-box-sizing: inherit;~~
~~box-sizing: inherit;~~

+ Filter Classes

Page

The web is plaintext 🤞

Authorization Overview



Agenda

Learn how MapKit JS authorization works

Create a MapKit JS Key

Create an authorization token

Make a map

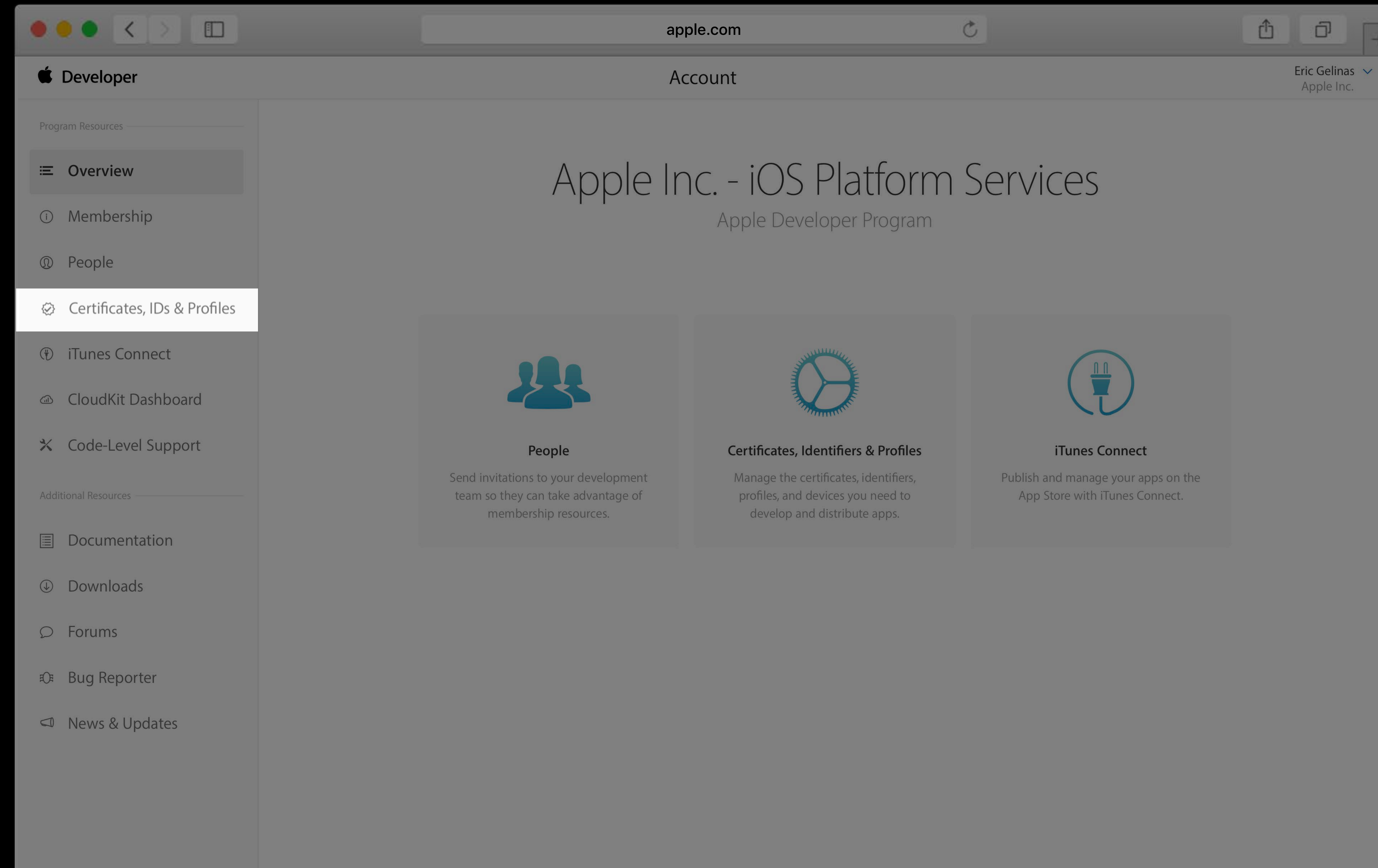
Create a MapKit JS Key

Create a Maps Identifier

Create a MapKit JS Key

Download your MapKit JS Key and saving it in a safe place

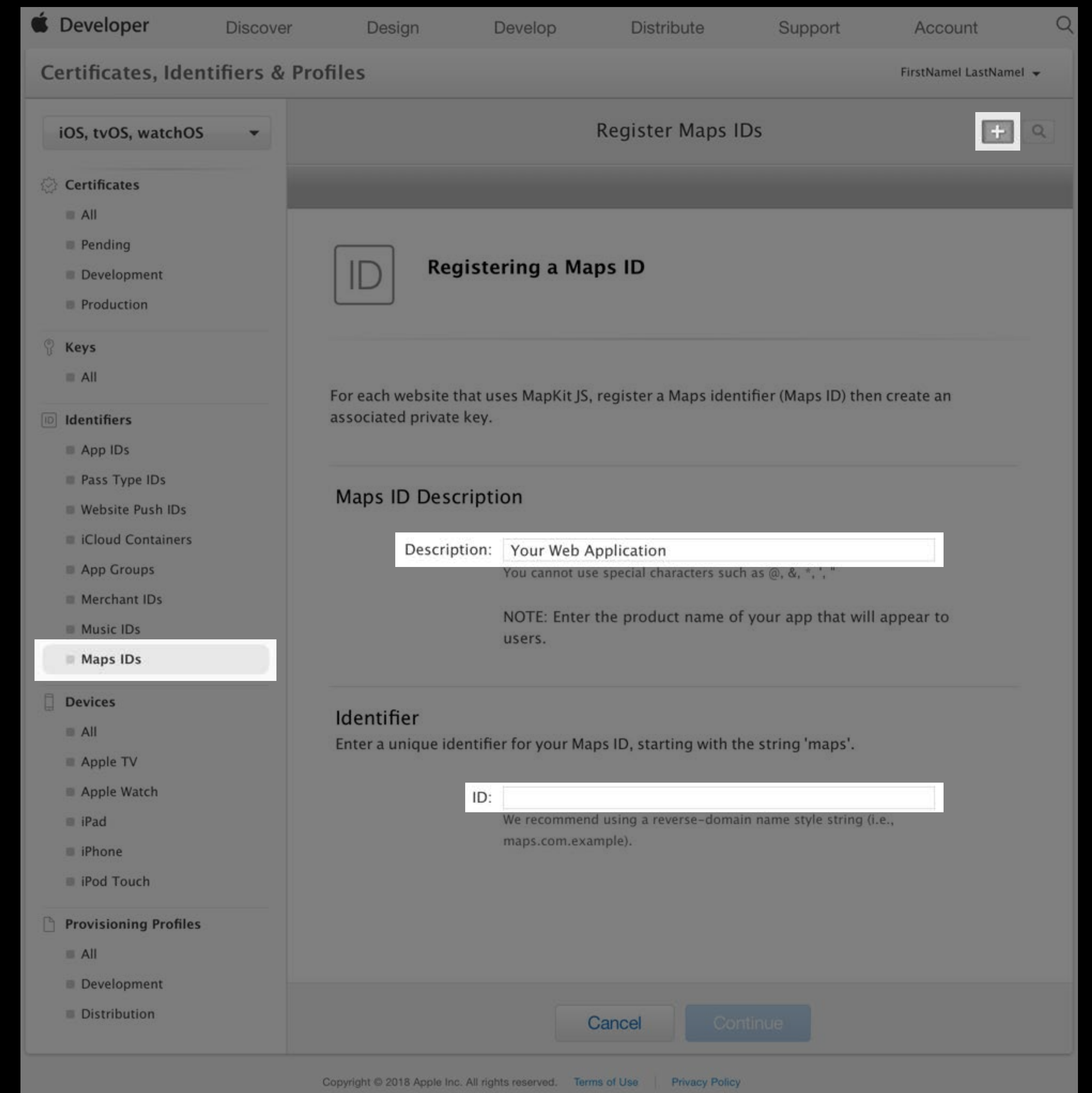
Create a MapKit JS Key



developer.apple.com/account

Create a Maps Identifier

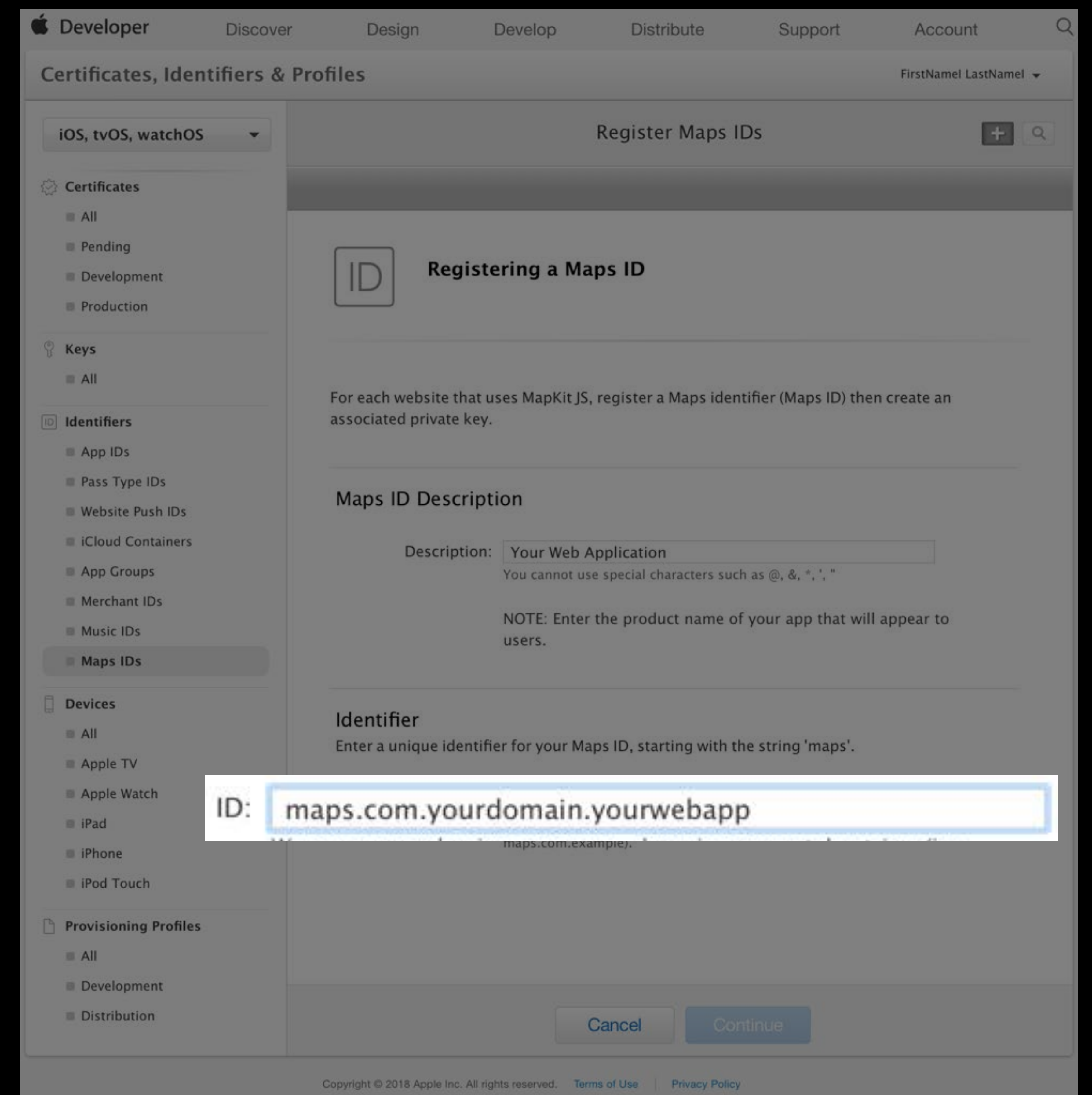
1. Select "Maps IDs" from the menu
2. Click the + button in the upper right corner
3. Give this identifier a description which should be the name of your website as it will appear to users
4. Give this identifier a unique ID starting with the string "maps"



developer.apple.com/account

Create a Maps Identifier

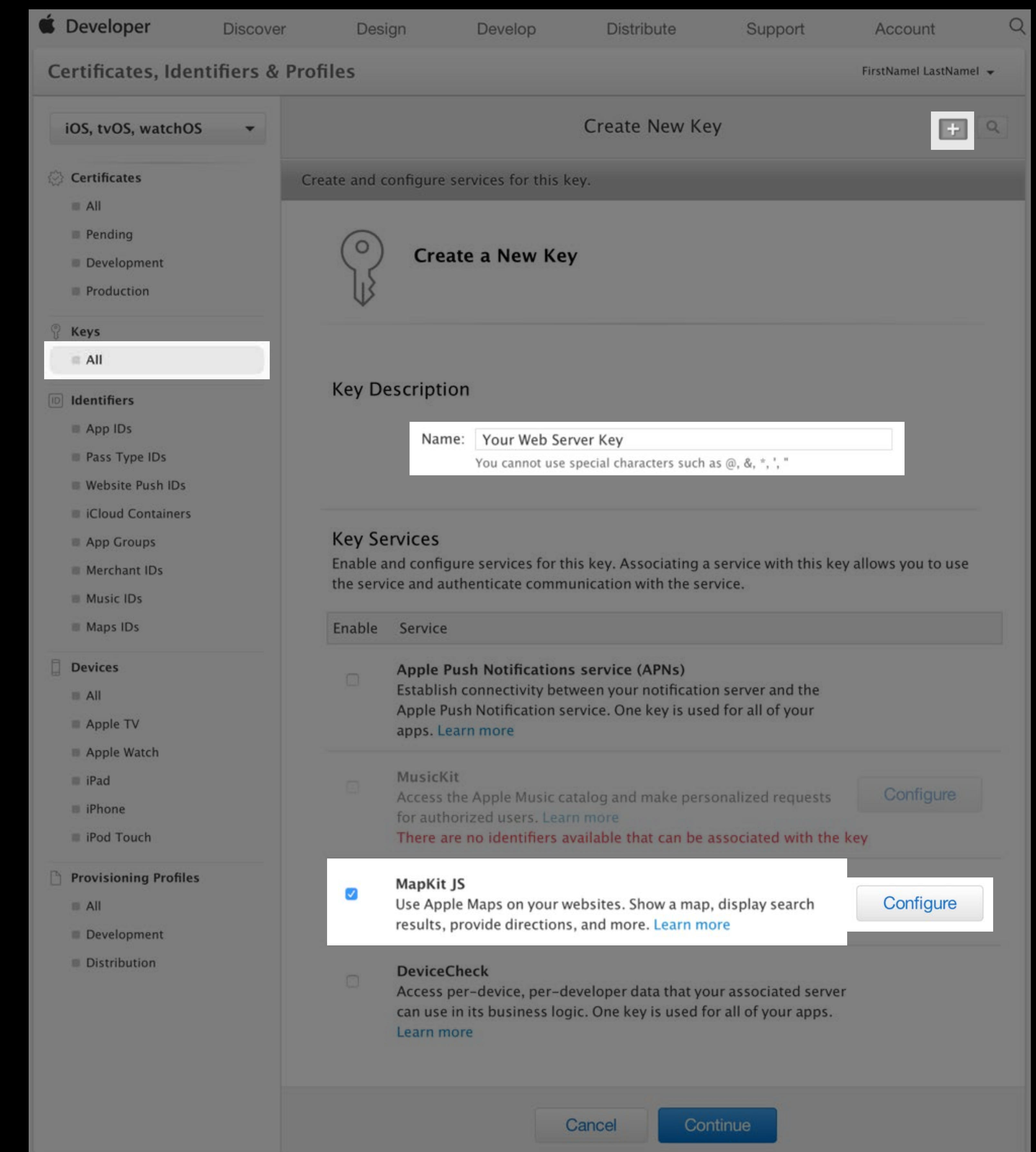
1. Select "Maps IDs" from the menu
2. Click the + button in the upper right
3. Give this identifier a description which should be the name of your website as it will appear to users
4. Give this identifier a unique ID starting with the string "maps"



developer.apple.com/account

Create a New Key

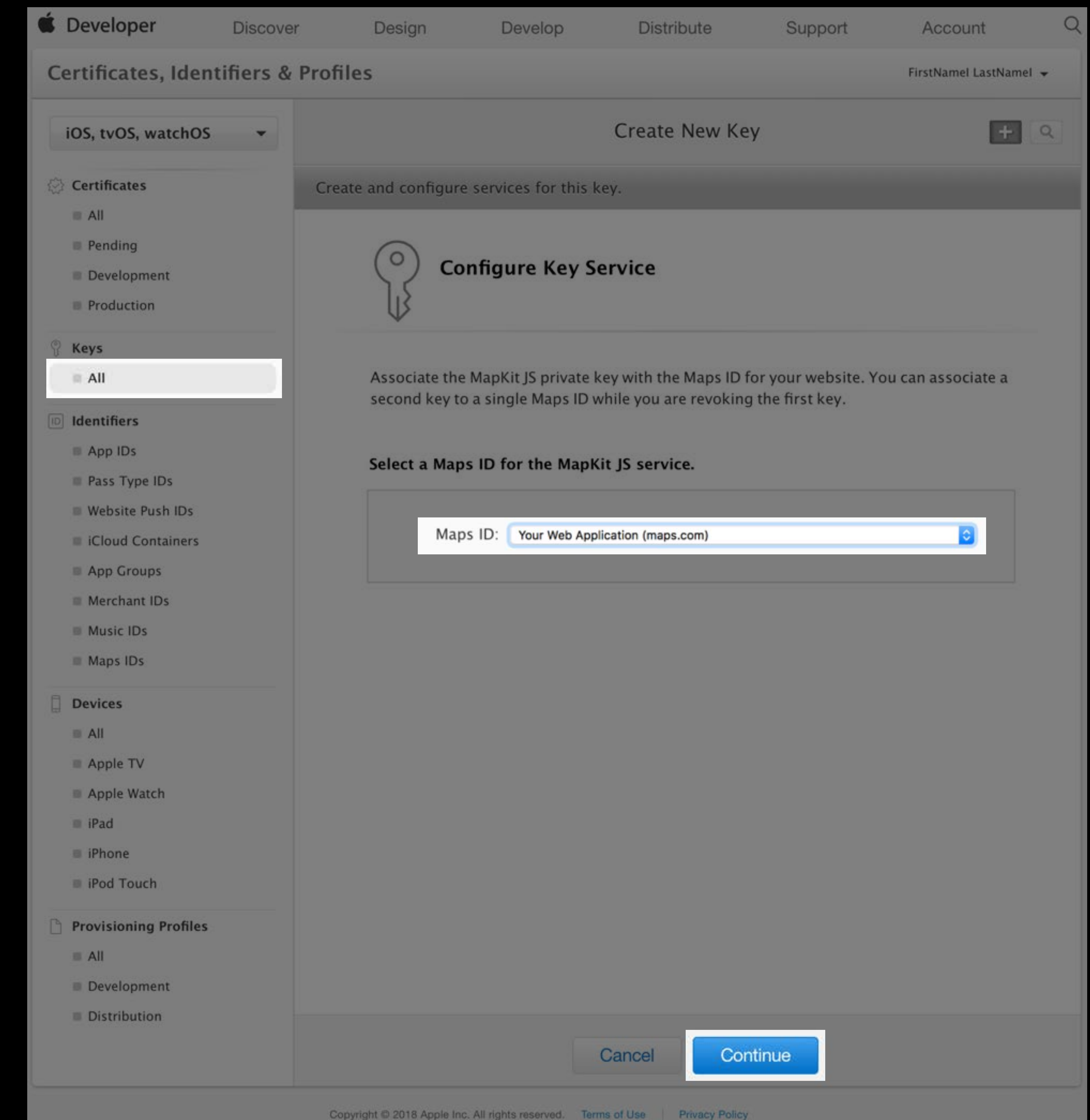
1. Click on "All" under the "Keys" subheading in the left-hand menu
2. Click the "+" icon in the header
3. Give this new key a name
4. Check the "MapKit JS" checkbox
5. Click the "Configure" button



developer.apple.com/account

Configure your New Key

1. Select an identifier to associate with this key
2. Click continue

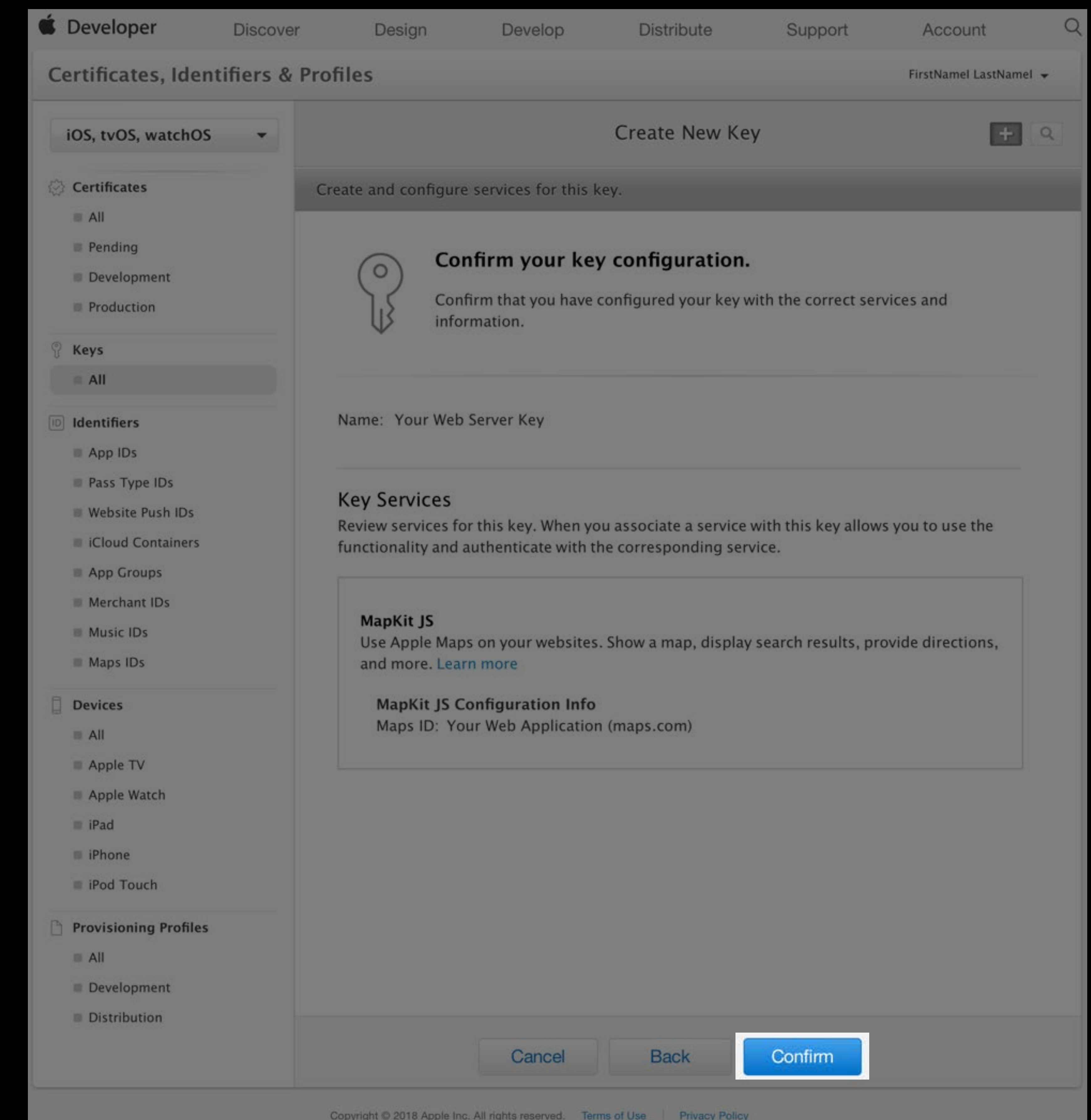


developer.apple.com/account

Confirm your New Key

1. Verify the information

2. Click "Confirm"



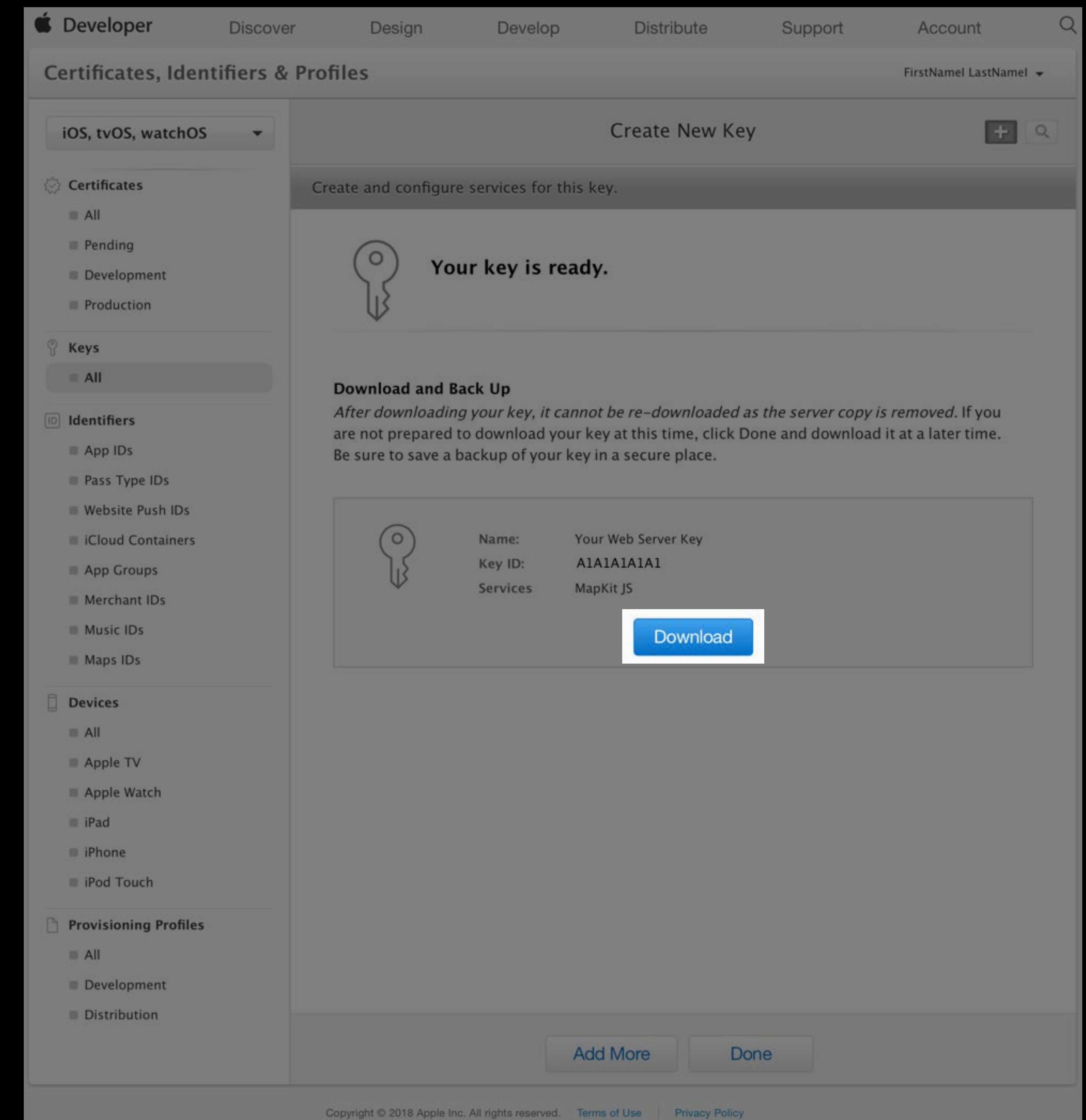
developer.apple.com/account

Download Your MapKit JS Key

Click the "Download" button

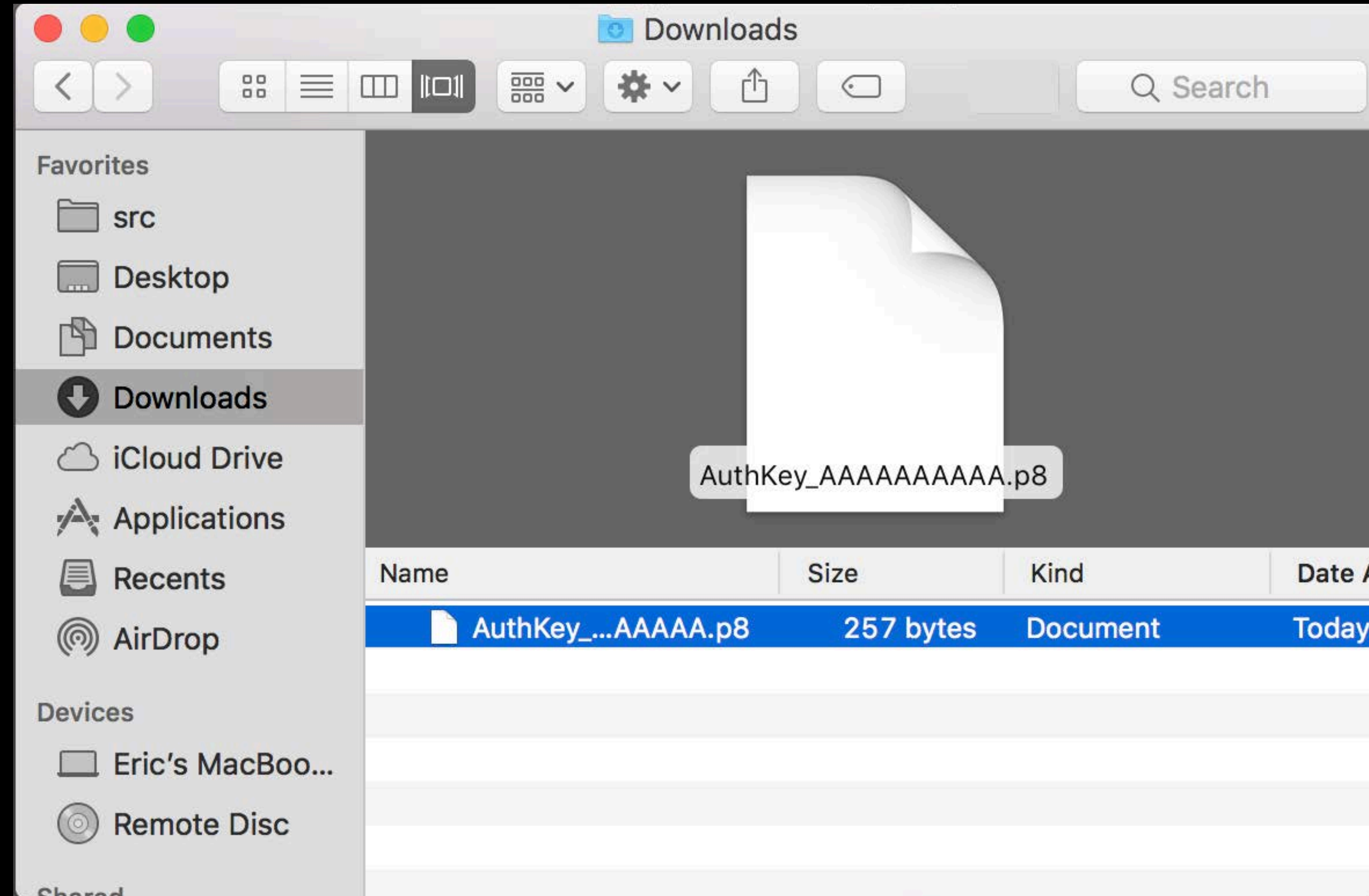
This file will be saved to your computer

You can only download this file once

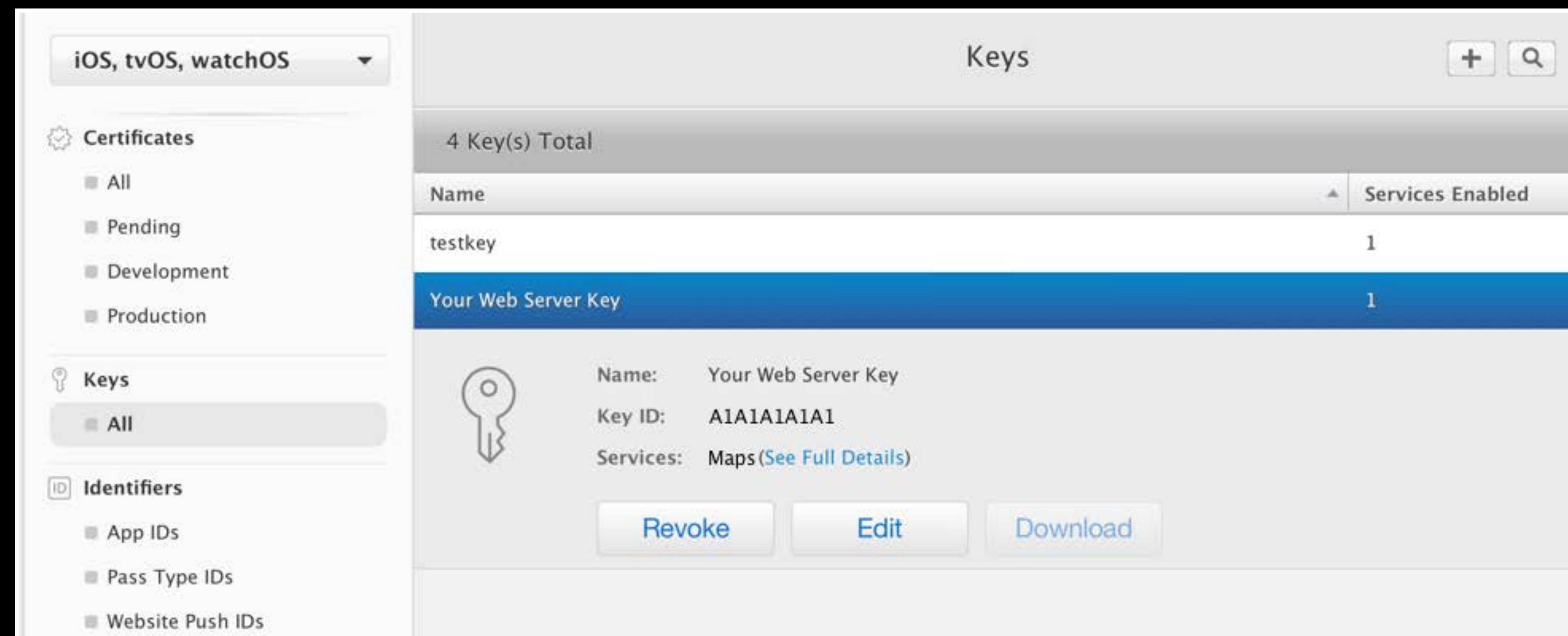


developer.apple.com/account

Your MapKit JS Key



Revoking a MapKit JS Key



developer.apple.com/account

Agenda

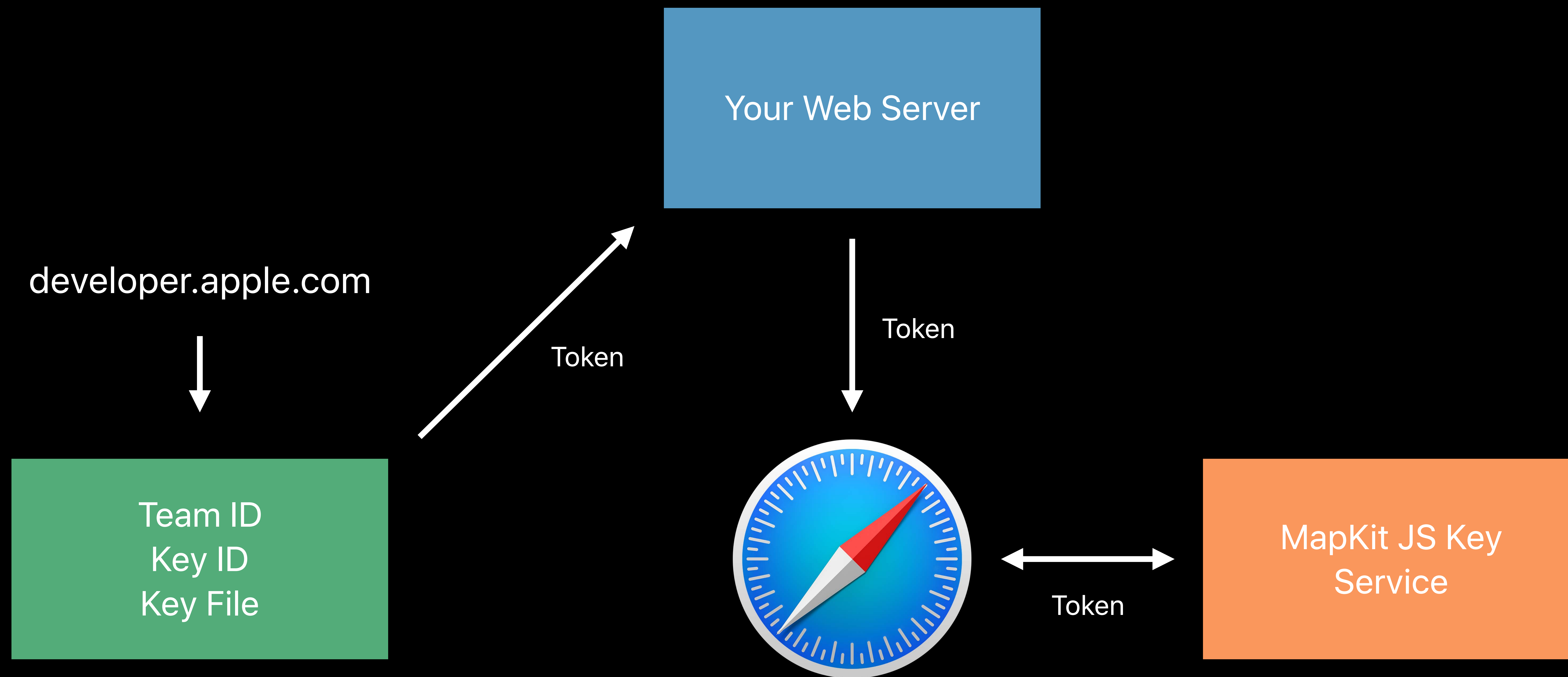
Learn how MapKit JS authorization works

Create a MapKit JS Key

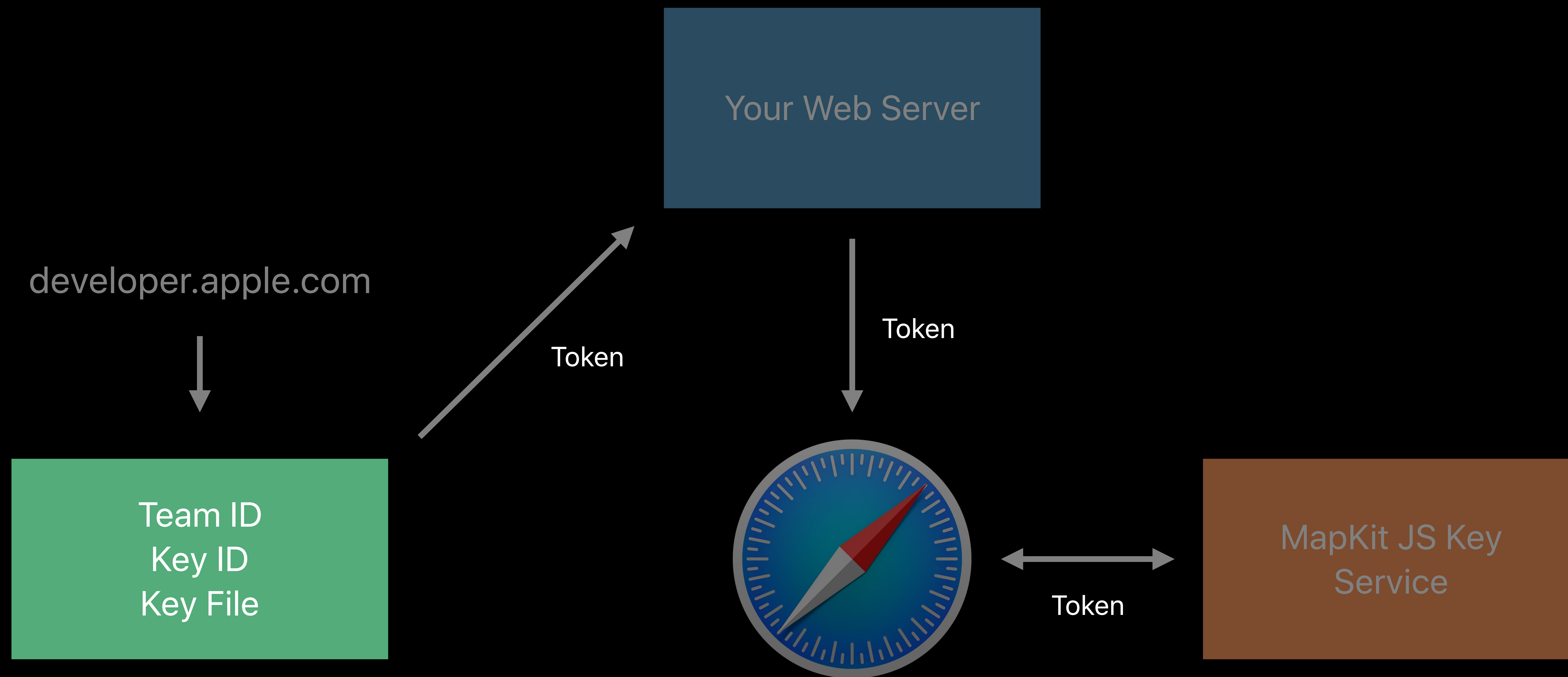
Create an authorization token

Make a map

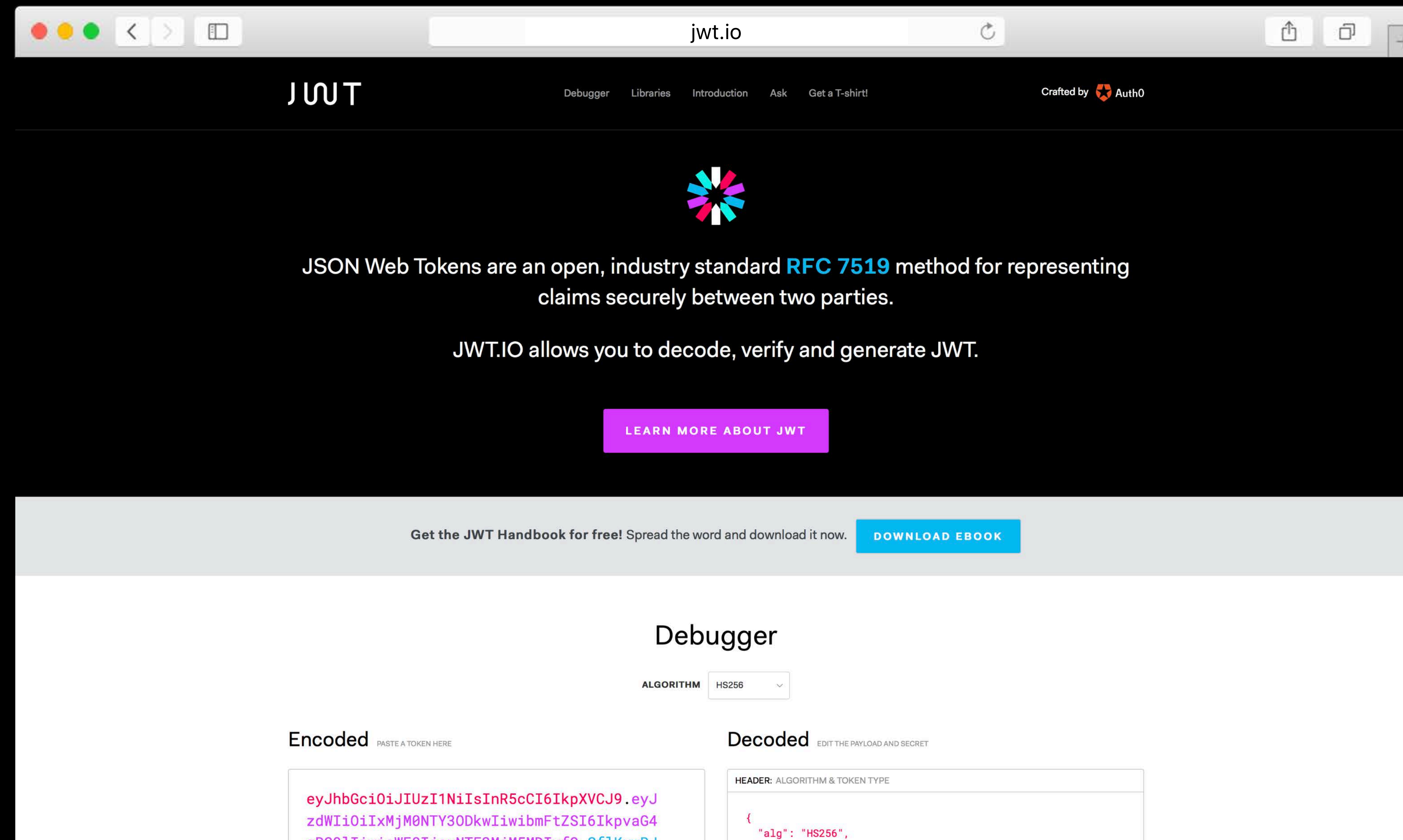
Authorization Overview



Authorization Overview



MapKit Authorization Tokens are based on JWT



The screenshot shows the jwt.io website in a browser window. The browser's address bar displays "jwt.io". The website's header includes the "JWT" logo, navigation links for "Debugger", "Libraries", "Introduction", "Ask", and "Get a T-shirt", and a note "Crafted by Auth0". The main content area features a colorful starburst logo and text explaining that JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties. It also states that JWT.IO allows users to decode, verify, and generate JWTs. A purple button labeled "LEARN MORE ABOUT JWT" is positioned below the text. A banner at the bottom of the main content area promotes a free "JWT Handbook" with a blue "DOWNLOAD EBOOK" button. The "Debugger" section is active, showing an "ALGORITHM" dropdown set to "HS256". It has two input fields: "Encoded" (with a "PASTE A TOKEN HERE" placeholder) and "Decoded" (with an "EDIT THE PAYLOAD AND SECRET" placeholder). The "Encoded" field contains a long alphanumeric string. The "Decoded" field shows a JSON object with the header: "alg": "HS256".

jwt.io

JWT Signing Libraries

The screenshot shows the jwt.io website interface. At the top, there's a navigation bar with the JWT logo and links for Debugger, Libraries, Introduction, Ask, and Get a T-shirt!. The main content area is divided into three columns representing different programming languages: Python, Node.js, and Java. Each column contains a list of supported signing algorithms and their corresponding checkmarks (green for supported, red for not supported). Below each list, there's a 'View Repo' link and a code block for installation instructions.

Language	Library	Author	Stars	Installation
Python	pyjwt	José Padilla	1925	<code>pip install pyjwt</code>
Python	python-jose	Michael Davis	403	<code>pip install python-jose</code>
Python	jwcrypto	Simo Sorce	74	<code>pip install jwcrypto</code>
Node.js	jsonwebtoken	Auth0	7495	<code>npm install jsonwebtoken</code>
Java	java-jwt	Auth0	1755	<code>maven: com.auth0 / java-jwt / 3.3.0</code>

Supported algorithms and checks for each library:

- pyjwt:** iat check (green), jti check (red), ES384 (green), ES512 (green)
- python-jose:** iat check (green), jti check (green), ES384 (green), ES512 (green)
- jwcrypto:** iat check (green), jti check (green), ES384 (green), ES512 (green)
- jsonwebtoken:** Sign (green), Verify (green), iss check (red), sub check (red), aud check (green), exp check (green), nbf check (red), iat check (red), jti check (red), HS256 (green), HS384 (green), HS512 (green), RS256 (green), RS384 (green), RS512 (green), ES256 (green), ES384 (green), ES512 (green)
- java-jwt:** Sign (green), Verify (green), iss check (green), sub check (green), aud check (green), exp check (green), nbf check (green), iat check (green), jti check (green), HS256 (green), HS384 (green), HS512 (green), RS256 (green), RS384 (green), RS512 (green), ES256 (green), ES384 (green), ES512 (green)

A JWT Token

```
eyJhbGciOiJIUzI1NiIsImtpZCI6ImZha2UifQ.eyJpc3MiOiJmYWt1IiwiaWF0IjoxNjI3MjE1NTk5LCJleHAiOjE1Mjc1Mjk1NzksIm9yaWdpbiI6Imh0dHBzOi8vZmFrZWRvbWFpbi5YWt1In0.a5-xFadXFUQ1cqL7I7AwzhquwxfeBKmdRWd-o00r1-XeI9N3V7YtNN-0UJK_7ItMk62174U-hphu1FnI71HfUW
```


JWT Sections

```
eyJhbGciOiJIUzI1NiIsImtpZCI6ImZha2UifQ.eyJpc3MiOiJmYWt1IiwiaWF0IjoxNjI3MjE1NTk1LCJleHAiOjE1Mjc1Mjk1NzksIm9yaWdpbiI6Imh0dHBzOi8vZmFrZWRvbWFpbi5Ywt1In0.a5-xFadXFUQ1cqL7I7AwzhquwxfeBKmdRWd-o00r1-XeI9N3V7YtNN-0UJK_7ItMk62174U-hphu1FnI71HfUW
```


JWT Sections

```
eyJhbGciOiJIUzI1NiIsImtpZCI6ImZha2UifQ.eyJpc3MiOiJmYWt1IiwiaWF0IjoxNjI3MjE1NTk5LCJleHAiOjE1Mjc1Mjk1NzksIm9yaWdpbiI6Imh0dHBzOi8vZmFrZWRvbWFpbj5Yw1In0.a5-xFadXFUQ1cqL7I7AwzhquwxfeBKmdRWd-o00r1-XeI9N3V7YtNN-0UJK_7ItMk62174U-hphu1FnI71HfUW
```

JWT Sections

```
eyJhbGciOiJIUzI1NiIsImtpZCI6ImZha2UifQ.eyJpc3MiOiJmYWt1IiwiaWF0IjoxNjI3MjE1NTk1LCJleHAiOjE1Mjc1Mjk1NzksIm9yaWdpbiI6Imh0dHBzOi8vZmFrZWRvbWFpbjUyYWt1In0.a5-xFadXFUQ1cqL7I7AwzhquwxfeBKmdRWd-o00r1-XeI9N3V7YtNN-0UJK_7ItMk62174U-hphu1FnI71HfUW
```

We Recommend Two JWT Configurations

Short lived tokens

Long lived tokens

Short Lived Tokens

Short (we recommend 30 min) time to live

If copied, will stop working after expiry

Requires you to provide a server endpoint

JWT Sections: Payload

```
let payload = {  
  iss: "A1A1A1A1A1", /* Issuer: Your Apple Developer Team ID */  
  iat: Date.now() / 1000, /* Issued at: Current time in seconds */  
  exp: (Date.now() / 1000) + 1800, /* Expiration: Time to expire in seconds */  
  origin: "https://yourdomain.com" /* (recommended - a domain restriction) */  
};
```

JWT Sections: Auth Key

```
let authKey = fs.readFile("./AuthKey_B1B1B1B1B1.p8");
```


JWT Sections: Header

```
let header = {  
  kid: "B1B1B1B1B1", /* Key Id: Your MapKit JS Key ID */  
  typ: "JWT" /* Type of token */  
  alg: "ES256" /* The hashing algorithm being used */  
};
```

JWT Sections: Pull It All Together

```
let authorizationToken = jwt.sign(payload, authKey, { header: header });
```

JWT Sections: Pull It All Together

```
let authorizatonToken = jwt.sign(payload, authKey, { header: header });
```

```
eyJhbGciOiJIUzI1NiIsImtpZCI6ImZha2UifQ.eyJpc3MiOiJmYWt1IiwiaWF0IjoxNjI3MjE1NTU5LCJleHAiOjE1Mjc1Mjk1NzksIm9yaWdpbiI6Imh0dHBzOi8vZmFrZWRvbWFpbi5YWt1In0.a5-xFadXFUQ1cqL7I7AwzhquwxfeBKmdRWd-o00r1-XeI9N3V7YtNN-0UJK_7ItMk62174U-hphu1FnI71HfUW
```

Create a Server Endpoint

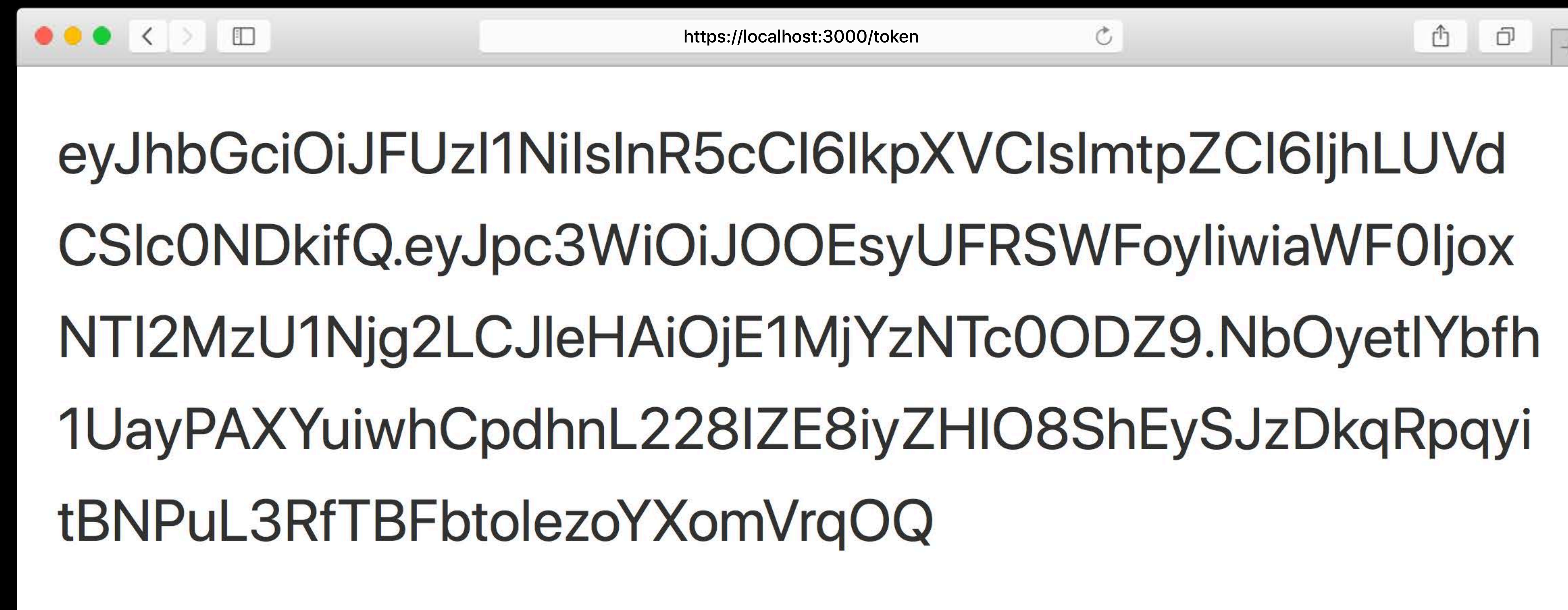
```
... /* Set up express server */  
  
app.get("/token", (req, res) => {  
  res.send(  
    jwt.sign(payload, keyFile, { header: header })  
  );  
});
```

Create a Server Endpoint

```
... /* Set up express server */

app.get("/token", (req, res, next) => {
  res.header("Cache-Control", "no-cache, no-store, must-revalidate");
  next();
}, (req, res) => {
  res.send(
    jwt.sign(payload, authKey, { header: header })
  );
});
```

Create a Server Endpoint



localhost:3000/token

Long Lived Token

Does not require a server

Can be used on static websites

Can be synced to your release cycle

"origin" claim strongly recommended

Creating a Long Lived Token

```
let payload = {  
  iss: "A1A1A1A1A1", /* Issuer: Your Apple Developer Team ID */  
  iat: Date.now() / 1000, /* Issued at: Current time in seconds */  
  exp: (Date.now() / 1000) + 15778800, /* Expiration time: Time to expire in seconds */  
  origin: "https://yourdomain.com" /* (recommended - a domain restriction) */  
};
```


Creating a Long Lived Token

```
let payload = {  
  iss: "A1A1A1A1A1", /* Issuer: Your Apple Developer Team ID */  
  iat: Date.now() / 1000, /* Issued at: Current time in seconds */  
  exp: (Date.now() / 1000) + 15778800, /* Expiration time: Time to expire in seconds */  
  origin: "https://yourdomain.com" /* (recommended - a domain restriction) */  
};
```

Now, we are ready to make a 

Make a Map

```
<!DOCTYPE html>
<html>
...
<div id="map"></div>
<script src="https://cdn.apple-mapkit.com/mk/5.0.x/mapkit.js"></script>
<script>
mapkit.init({ authorizationCallback: function(done) {
    fetch("/token")
      .then(res => res.text())
      .then(token => done(token)) /* If successful, return your token to MapKit JS */
      .catch(error => { /* Handle error */ });
}});
let map = new mapkit.Map("map", { center: new mapkit.Coordinate(37.32, -121.88) });
</script>
...
</html>
```

Make a Map

```
<!DOCTYPE html>
<html>
...
<div id="map"></div>
<script src="https://cdn.apple-mapkit.com/mk/5.0.x/mapkit.js"></script>
<script>
mapkit.init({ authorizationCallback: function(done) {
    fetch("/token")
      .then(res => res.text())
      .then(token => done(token)) /* If successful, return your token to MapKit JS */
      .catch(error => { /* Handle error */ });
  }});
let map = new mapkit.Map("map", { center: new mapkit.Coordinate(37.32, -121.88) });
</script>
...
</html>
```

Make a Map

```
<!DOCTYPE html>
<html>
...
<div id="map"></div>
<script src="https://cdn.apple-mapkit.com/mk/5.0.x/mapkit.js"></script>
<script>
mapkit.init({ authorizationCallback: function(done) {
    fetch("/token")
      .then(res => res.text())
      .then(token => done(token)) /* If successful, return your token to MapKit JS */
      .catch(error => { /* Handle error */ });
  }});
let map = new mapkit.Map("map", { center: new mapkit.Coordinate(37.32, -121.88) });
</script>
...
</html>
```

Make a Map

```
<!DOCTYPE html>
<html>
...
<div id="map"></div>
<script src="https://cdn.apple-mapkit.com/mk/5.0.x/mapkit.js"></script>
<script>
mapkit.init({ authorizationCallback: function(done) {
    fetch("/token")
      .then(res => res.text())
      .then(token => done(token)) /* If successful, return your token to MapKit JS */
      .catch(error => { /* Handle error */ });
  }});
let map = new mapkit.Map("map", { center: new mapkit.Coordinate(37.32, -121.88) });
</script>
...
</html>
```

Long Lived Token in MapKit JS

```
<!DOCTYPE html>
<html>
...
<div id="map"></div>
<script src="https://cdn.apple-mapkit.com/mk/5.0.x/mapkit.js"></script>
<script>
mapkit.init({ authorizationCallback: function(done) {
    done("eyJhbGciOiJIUzI1NiIsImtpZCI6ImZha2UifQ.eyJpc3MiOiJmYWt1IiwiaWF0IjoxN
    TI3NTI5NTk5LCJleHAiOjE1Mjc1Mjk1NzksIm9yaWdpbiI6Imh0dHBzOi8vZmFrZWRvb
    WFpbi5YWt1In0.a5-xFadXFUQ1cqL7I7AwzhquwxfeBKmdRWd-o00r1-XeI9N3V7YtNN
    -0UJK_7ItMk62174U-hphu1FnI71HfUW");
}});
let map = new mapkit.Map("map", { center: new mapkit.Coordinate(37.32, -121.88) });
</script>
...
</html>
```

Make a Map

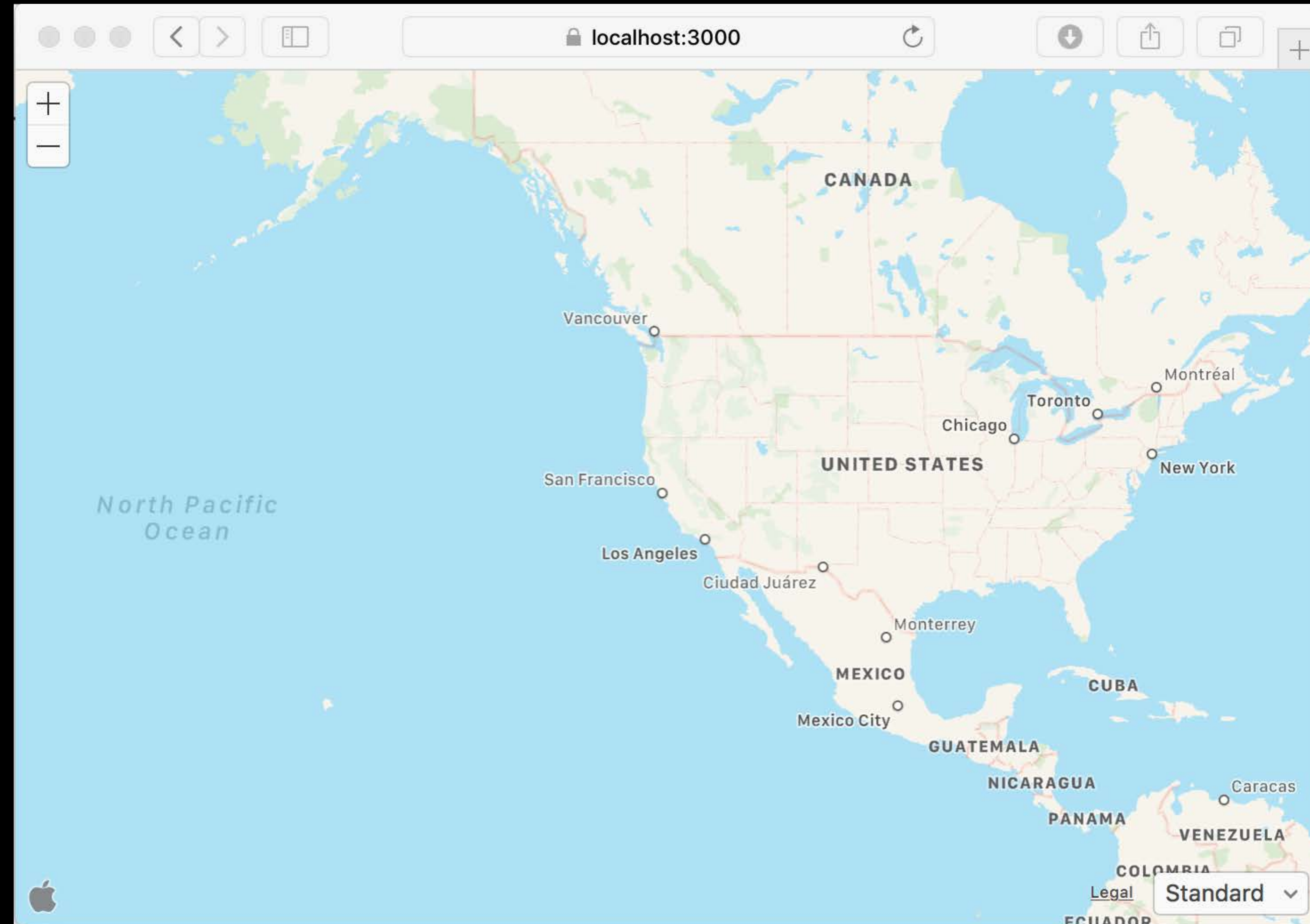
```
<!DOCTYPE html>
<html>
...
<div id="map"></div>
<script src="https://cdn.apple-mapkit.com/mk/5.0.x/mapkit.js"></script>
<script>
mapkit.init({ authorizationCallback: function(done) {
    done("eyJhbGciOiJIUzI1NiIsImtpZCI6ImZha2UifQ.eyJpc3MiOiJmYWt1IiwiaWF0IjoxN
    TI3NTI5NTk5LCJleHAiOjE1Mjc1Mjk1NzksIm9yaWdpbiI6Imh0dHBzOi8vZmFrZWRvb
    WFpbi5YWt1In0.a5-xFadXFUQ1cqL7I7AwzhquwxfeBKmdRWd-o00r1-XeI9N3V7YtNN
    -0UJK_7ItMk62174U-hphu1FnI71HfUW");
}});
let map = new mapkit.Map("map", { center: new mapkit.Coordinate(37.32, -121.88) });
</script>
...
</html>
```


Start Your Server

A terminal window with a title bar that reads "remarkableproduct — node server — 80x24". The terminal content shows the command "node server" being executed, followed by the output "Your server is running on port 3000". A cursor is visible on the line following the output.

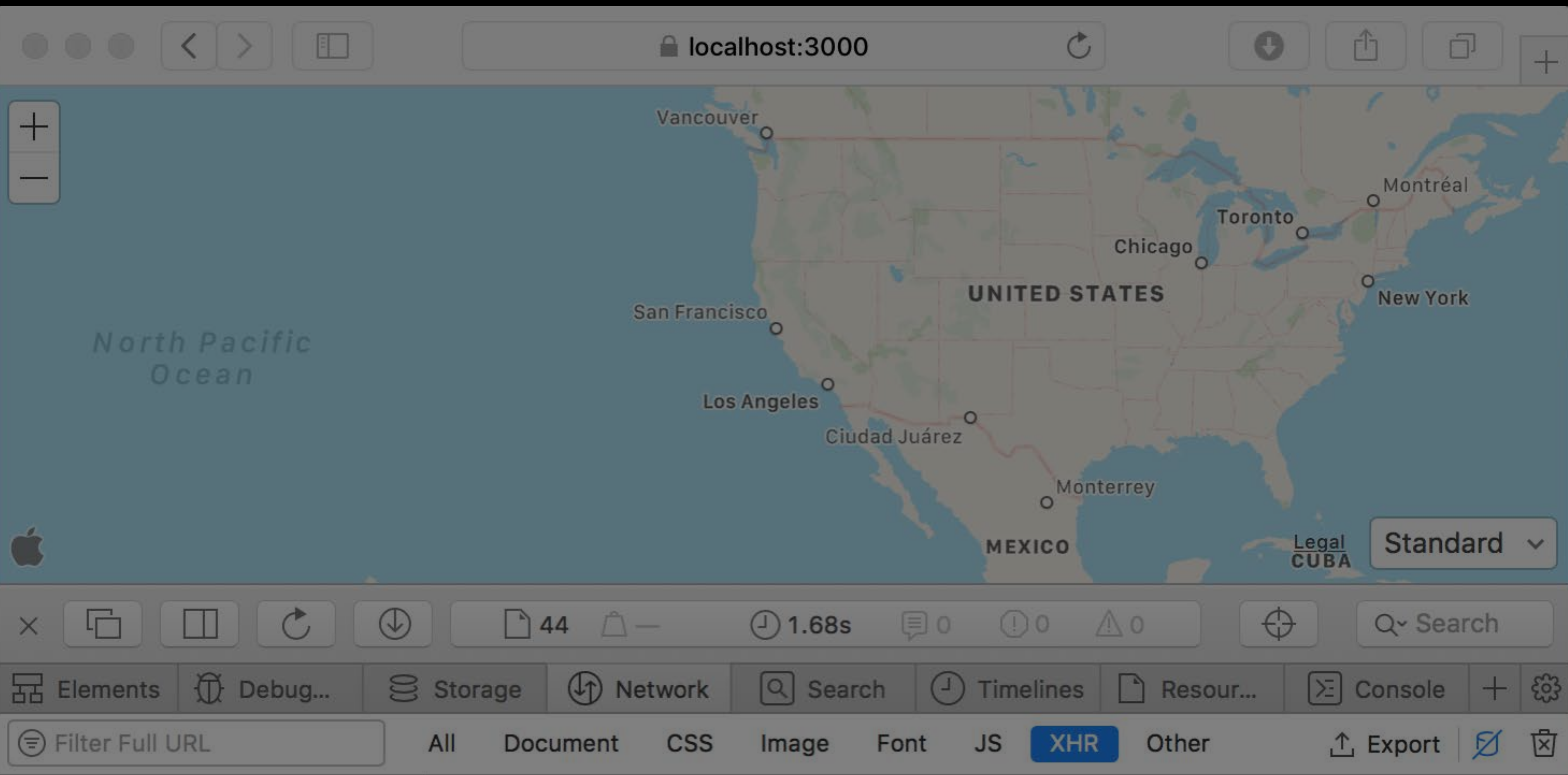
```
[Eric's-MacBook-Pro-3:remarkableproduct eric$ node server  
Your server is running on port 3000  
█
```

Take a Look in Safari



localhost:3000

Authorization Header Containing Your Token



The screenshot shows a web browser at localhost:3000 displaying a map of the United States and Mexico. The browser's developer tools are open, showing the XHR (XMLHttpRequest) tab. A request for a 'token' is highlighted, and its response is shown as a long alphanumeric string.

```
1 eyJhbGciOiJFUzI1NiIsImtpZCI6ImZha2UifQ.eyJpc3MiOiJmYWtlliwiaWF0IjoxNTI3NTI5NTk5LCJleHAiOiJlMjc1Mjk1NzksIm9yaWdpbil6Imh0dHBzOi8vZmFrZWRvbWpbi5YWtlln0.a5-xFadXFUQlcqL7I7AwzhquwxfeBKmdRWd-oOOr1-Xel9N3V7YtNN-0UJK_7ItMk62I74U-hphu1FnI7IHfUW
```

localhost:3000

Agenda

Learn how MapKit JS authorization works

Create a MapKit JS key

Create an authorization token

Make a map

Summary

Get your MapKit JS Key and store it in a safe place

Only send authorization tokens over the web.

If a key becomes compromised, revoke it on the [Apple Developer Website](#) as soon as you have created a replacement

More Information

<https://developer.apple.com/wwdc18/508>

Introducing MapKit JS

Executive Ballroom

Tuesday, 5:00pm

 **WWDC18**