

# JAVASCRIPT DEVELOPMENT

*Sasha Vodnik, Instructor*

# HELLO!

1. Pull changes from the `svodnik/JS-SF-12-resources` repo to your computer
2. Open the `16-deploying` folder in your code editor

---

**JAVASCRIPT DEVELOPMENT**

---

# **DEPLOYING YOUR APP**

# **LEARNING OBJECTIVES**

At the end of this class, you will be able to

- › Understand what hosting is.
- › Identify a program's needs in terms of host providers.
- › Ensure backward compatibility by using Babel to transpile code.
- › Deploy to a web host.

# AGENDA

- Convert code to a module
- Transpile with Babel
- Deploy with Firebase

---

## DEPLOYING YOUR APP

---

# WEEKLY OVERVIEW

**WEEK 9**

Deploying your app / Final project lab

**WEEK 10**

(holiday) / React

**WEEK 11**

Final project presentations!

---

**DEPLOYING YOUR APP**

---

# **HOMework REvIEW**

---

# ACTIVITY

---



## **KEY OBJECTIVE**

---

- Review Feedr project and show off your work

## **TYPE OF EXERCISE**

---

- Groups of 2-4

## **TIMING**

---

*10 min*

1. Open Feedr sites on laptops and display them proudly!
2. Give feedback to your peers: "I like" and "I wish/wonder"
3. Share a challenge you ran into in your project and discuss how other group members may have worked with it.
4. Did you incorporate template literals in your project? Show your group how you did it!



---

# ACTIVITY

---



## **KEY OBJECTIVE**

---

- › Check in on final projects

## **TYPE OF EXERCISE**

---

- › Groups of 2-4

## **TIMING**

---

*6 min*

1. Share what you have done so far on your final project (notes/outline, wireframe, pseudocode, basic functionality...)
2. Share your next step. If you're not sure, share where you are right now and brainstorm with your group what next steps might look like.

# **EXIT TICKET QUESTIONS**

1. What are the advantages and disadvantages of using Firebase. Are there comparable offerings?
2. What is webpack?


---

**DEPLOYING YOUR APP**

---

# **CONVERTING CODE TO A MODULE**

# **BUILDING BLOCKS OF A MODULE**



**OBJECT-  
ORIENTED  
CODE**



**CLOSURES**



**IIFES**

## **THE MODULE PATTERN**

- Using an IIFE to return an object literal
- The methods of the returned object can access the private properties and methods of the IIFE (closures!), but other code cannot do this
- This means specific parts of the IIFE are not available in the global scope

# BUILDING A MODULE

```
let counter = function() {  
  let count = 0;  
  return {  
    reset: function() {  
      count = 0;  
    },  
    get: function() {  
      return count;  
    },  
    increment: function() {  
      count++;  
    }  
  };  
}();
```

returning an  
object literal

from an IIFE

containing closures

## **BENEFITS OF THE MODULE PATTERN**

- Keeps some functions and variables private
- Avoids polluting the global scope
- Organizes code into objects

## **CREATING A REVEALING MODULE**

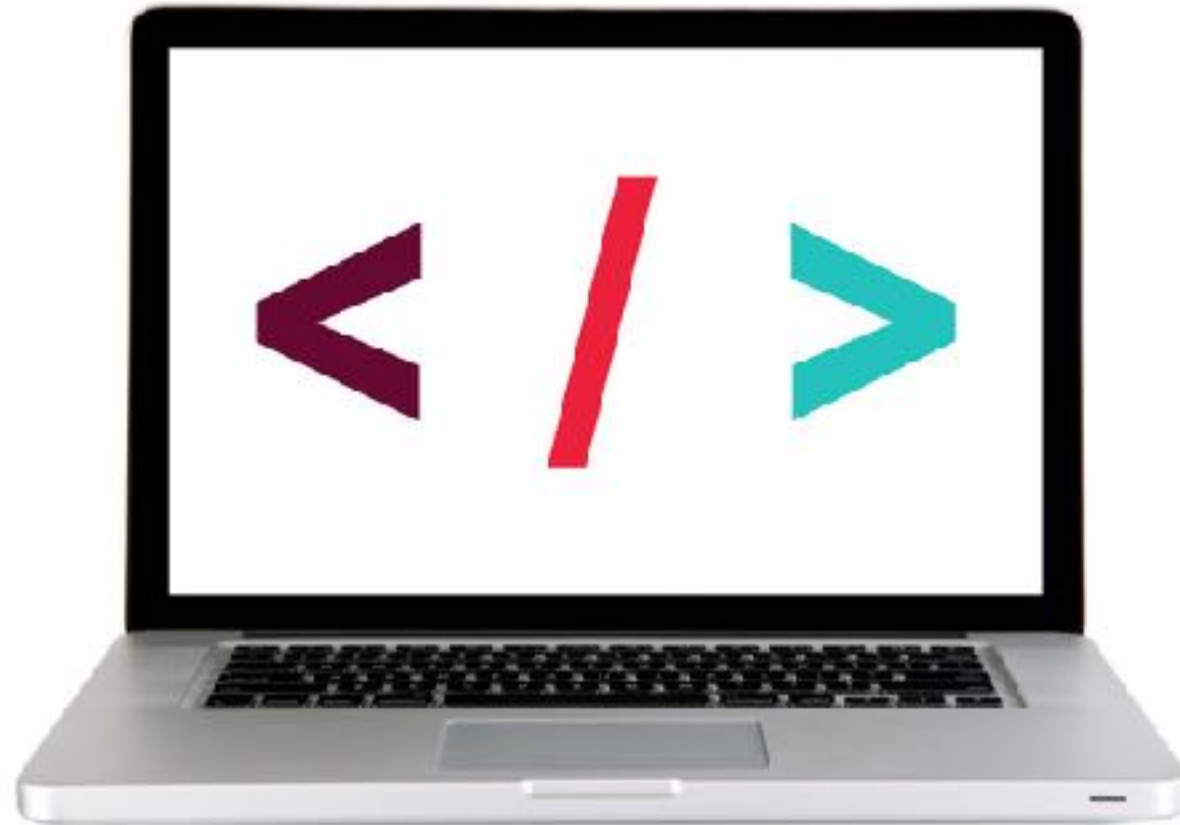
1. Group variables and functions within an IIFE.
2. Return an object from the IIFE containing properties and/or methods that are aliases for variables and/or functions within the IIFE.
3. Change any references to the variables and functions outside of the IIFE to use object notation and reference the aliases you defined.



---

**LET'S TAKE A CLOSER LOOK**

---



---

# EXERCISE — CONVERT CODE TO A MODULE

---



## **KEY OBJECTIVE**

---

- ▶ Convert the code for your CRUD app to use the module pattern

## **LOCATION**

---

- ▶ 1-module-exercise > js > app.js (or your completed app)

## **TIMING**

---

*5 min*

1. Enclose the `getPosts()`, `updateMessage()`, and `deleteMessage()` functions in an IIFE, and assign it the name `messageClass`.
2. Within the IIFE, add code to return an object containing a method that provides access to the `getPosts()` function.
3. In the `$(function)` code, change the `getPosts()` call to instead call the new method you created.
4. Test your app and make sure all its functionality still works.

---

**DEPLOYING YOUR APP**

---

# **FINALIZING YOUR CODE**

---

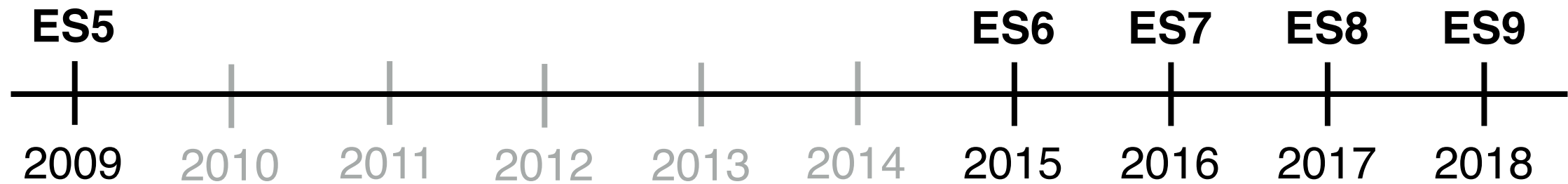
**DEPLOYING YOUR APP**

---

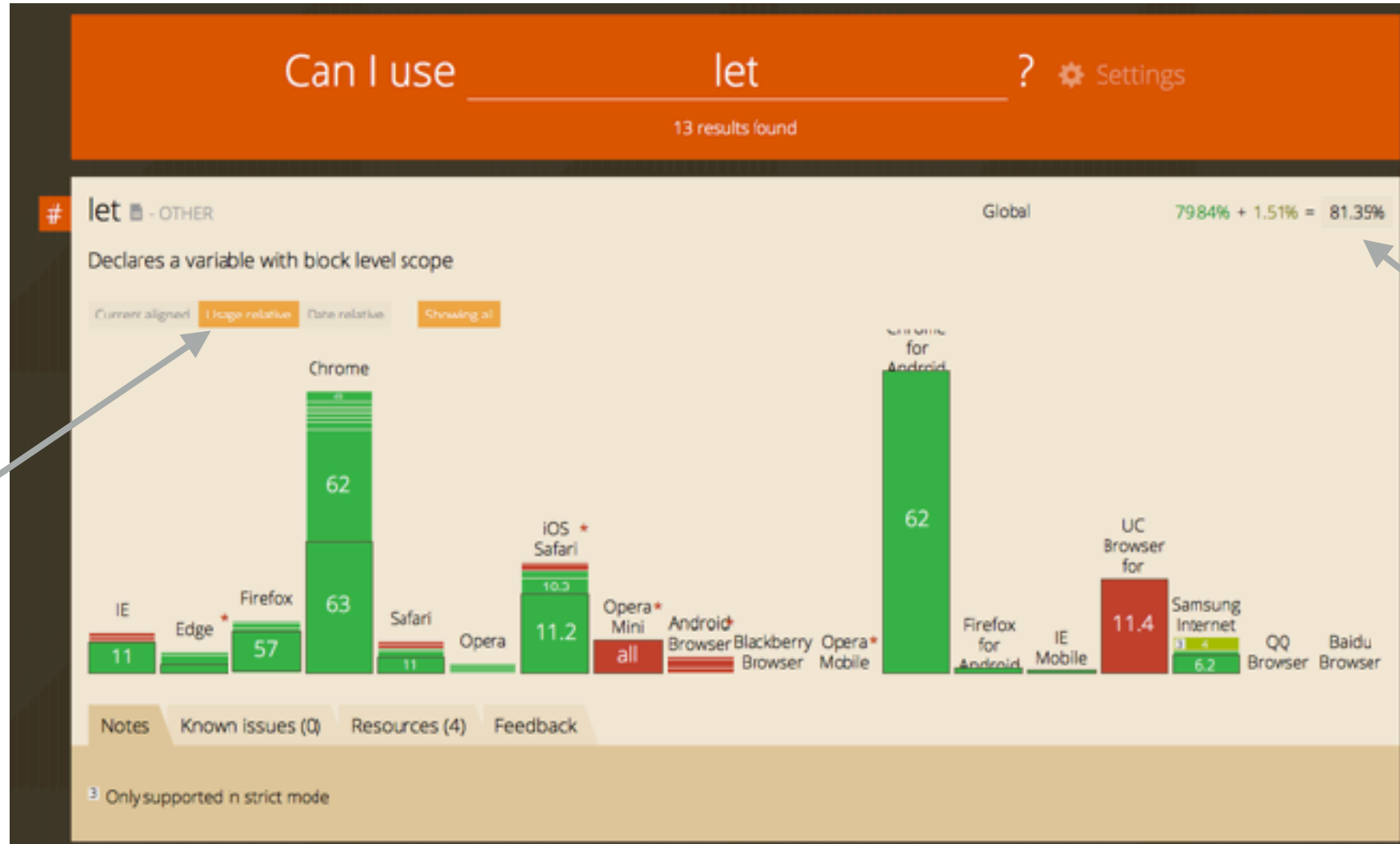
# **TRANSPILING**

virtually all browsers  
in use support ES5

only modern browsers  
support ES6+



caniuse.com



“Usage relative” option shows proportional graph

Estimated percent of global browser traffic that can parse this feature

**Transpiling** involves rewriting code that uses ES6+ features to produce the same result using ES5 code

ES6

```
const taxRate = 0.0875;  
let items = [];  
  
let addToCart = () => {  
  // do something  
}
```

transpiling



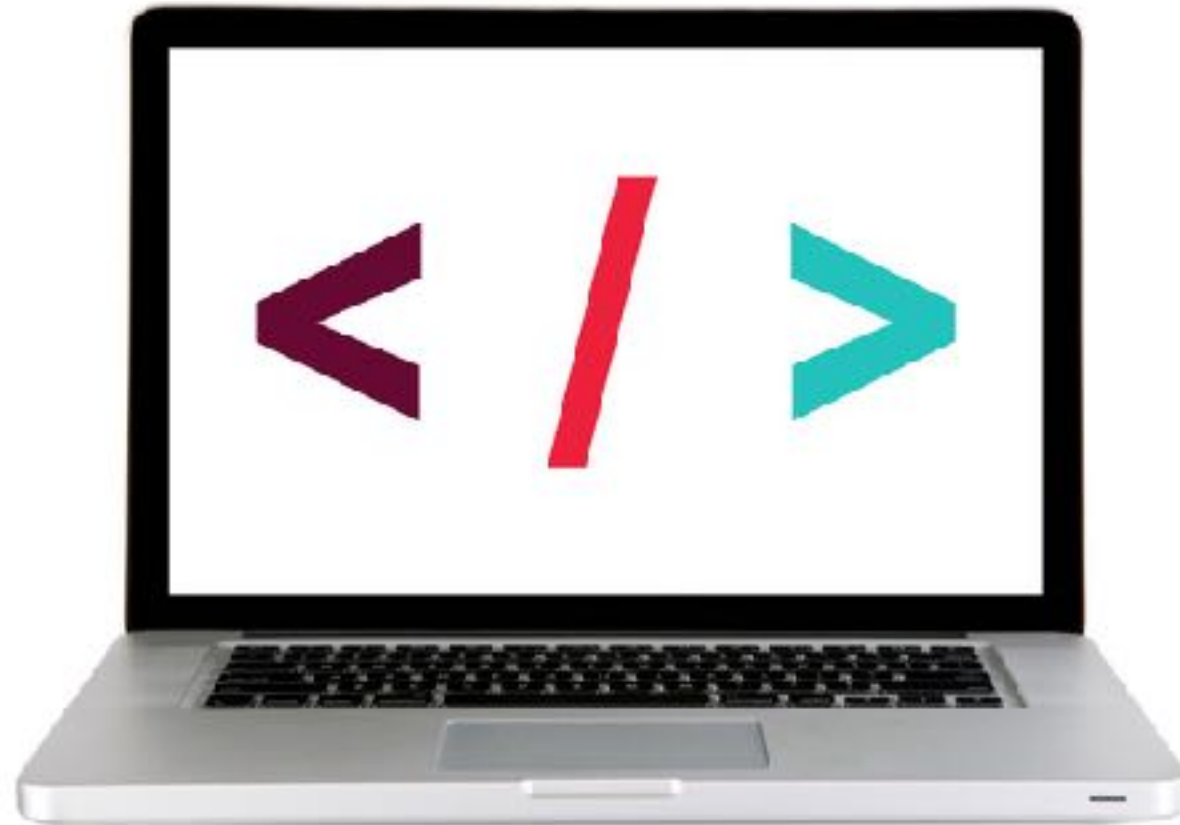
ES5

```
var taxRate = 0.0875;  
var items = [];  
  
function addToCart() {  
  // do something  
}
```

---

**LET'S TAKE A CLOSER LOOK**

---





---

## EXERCISE — TRANSPILE CODE USING BABEL

---



EXERCISE

### KEY OBJECTIVE

---

- › Ensure backward compatibility by using Babel to transpile code.

### TIMING

---

*5 min*

1. Configure Babel for the app you created in class. (If your code isn't quite working, use the code in the `starter-code > 3-transpiling-exercise` folder as a starting point.)
2. Run Babel to create an ES5-compatible version of your code.
3. Open the converted file in your editor and verify the code was transpiled.
4. Test your app in the browser and make sure it still works as it did previously.

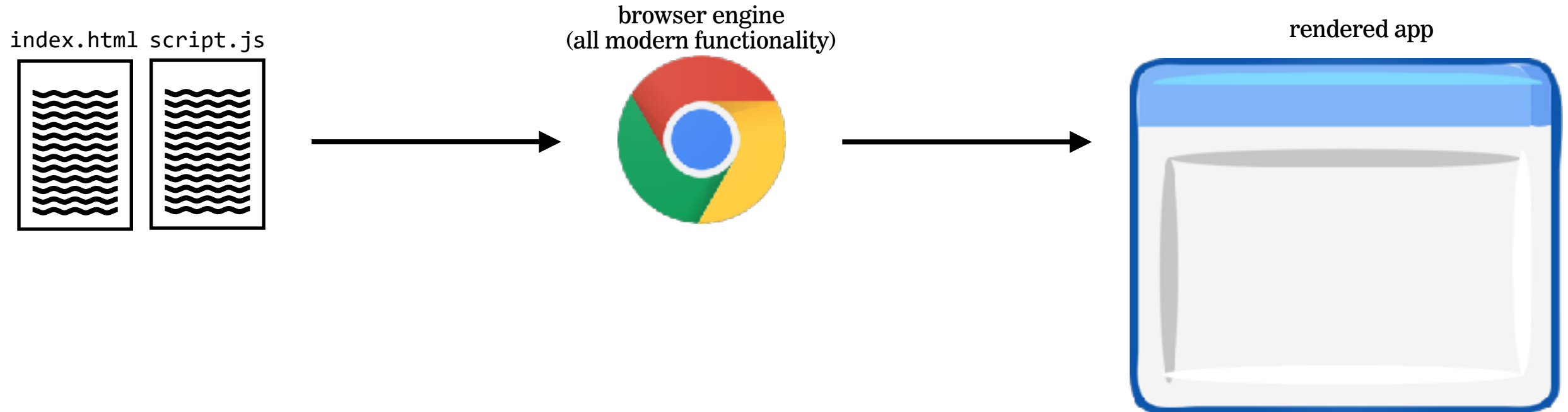
---

**DEPLOYING YOUR APP**

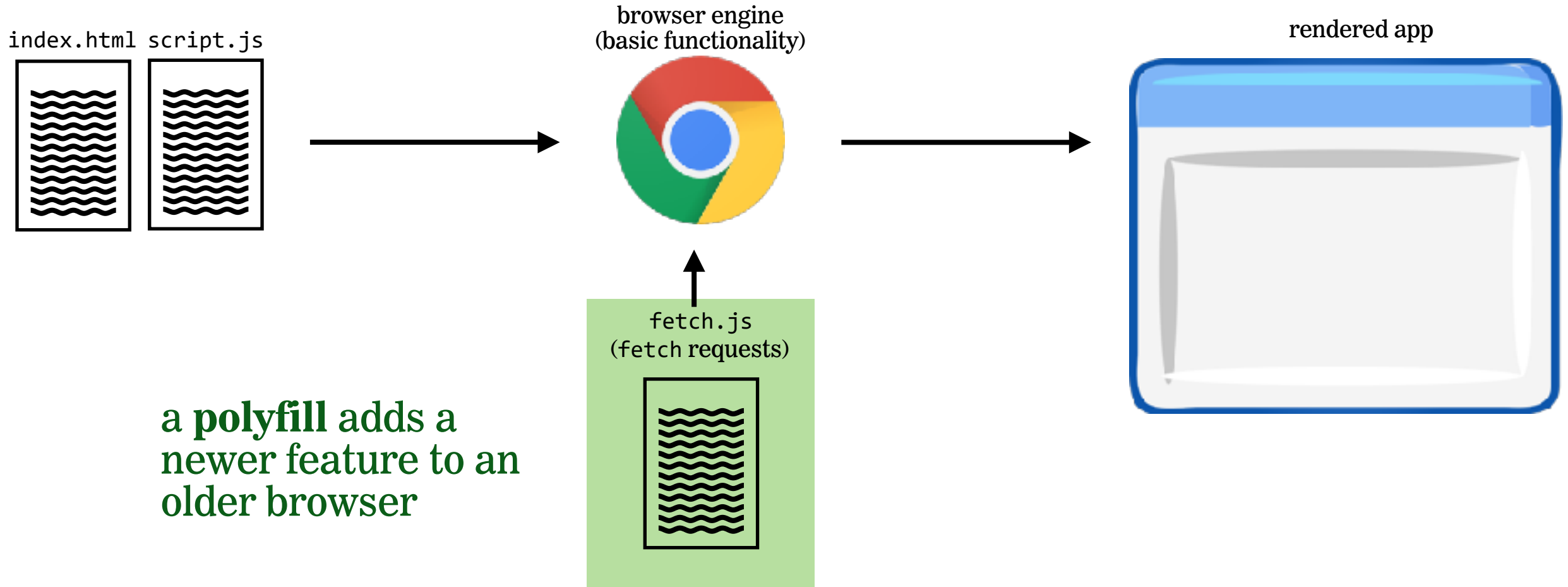
---

# **POLYFILLS**

# APP FUNCTIONALITY IN A MODERN BROWSER



# APP FUNCTIONALITY IN AN OLDER BROWSER

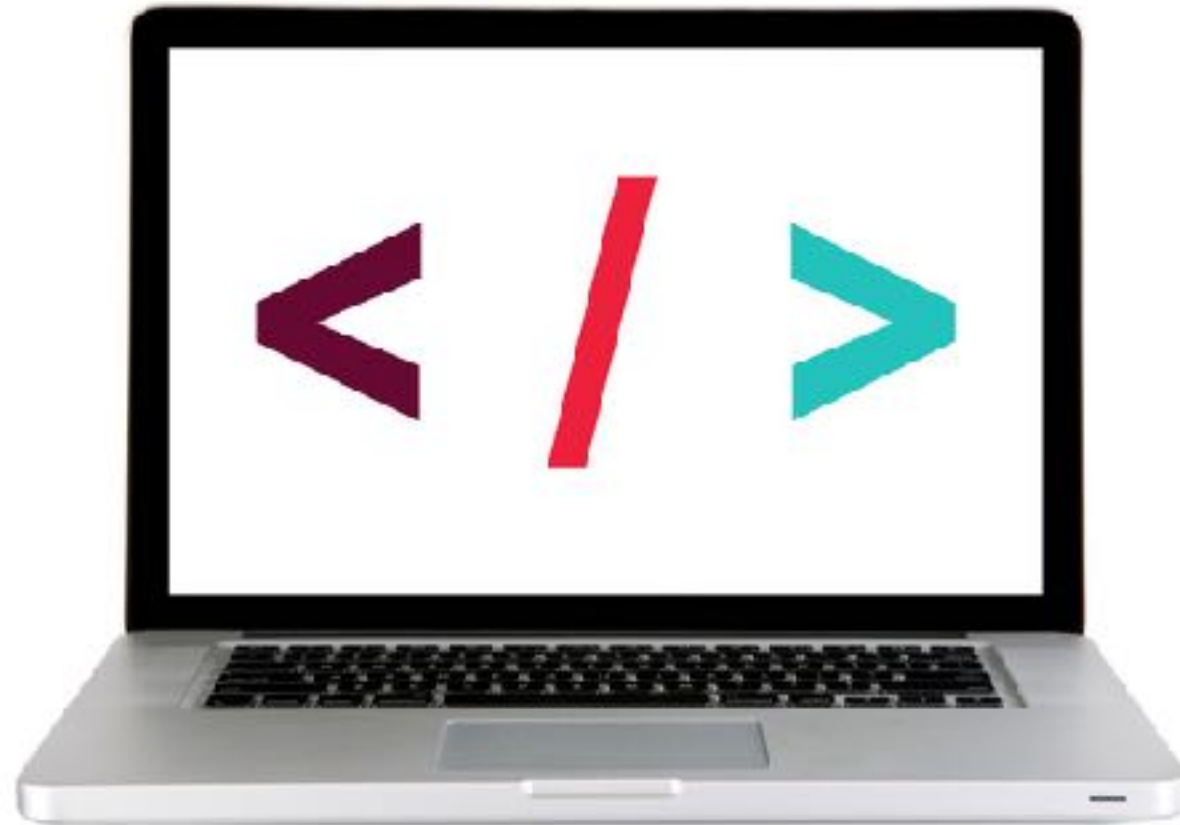


a **polyfill** adds a newer feature to an older browser

---

**LET'S TAKE A CLOSER LOOK**

---

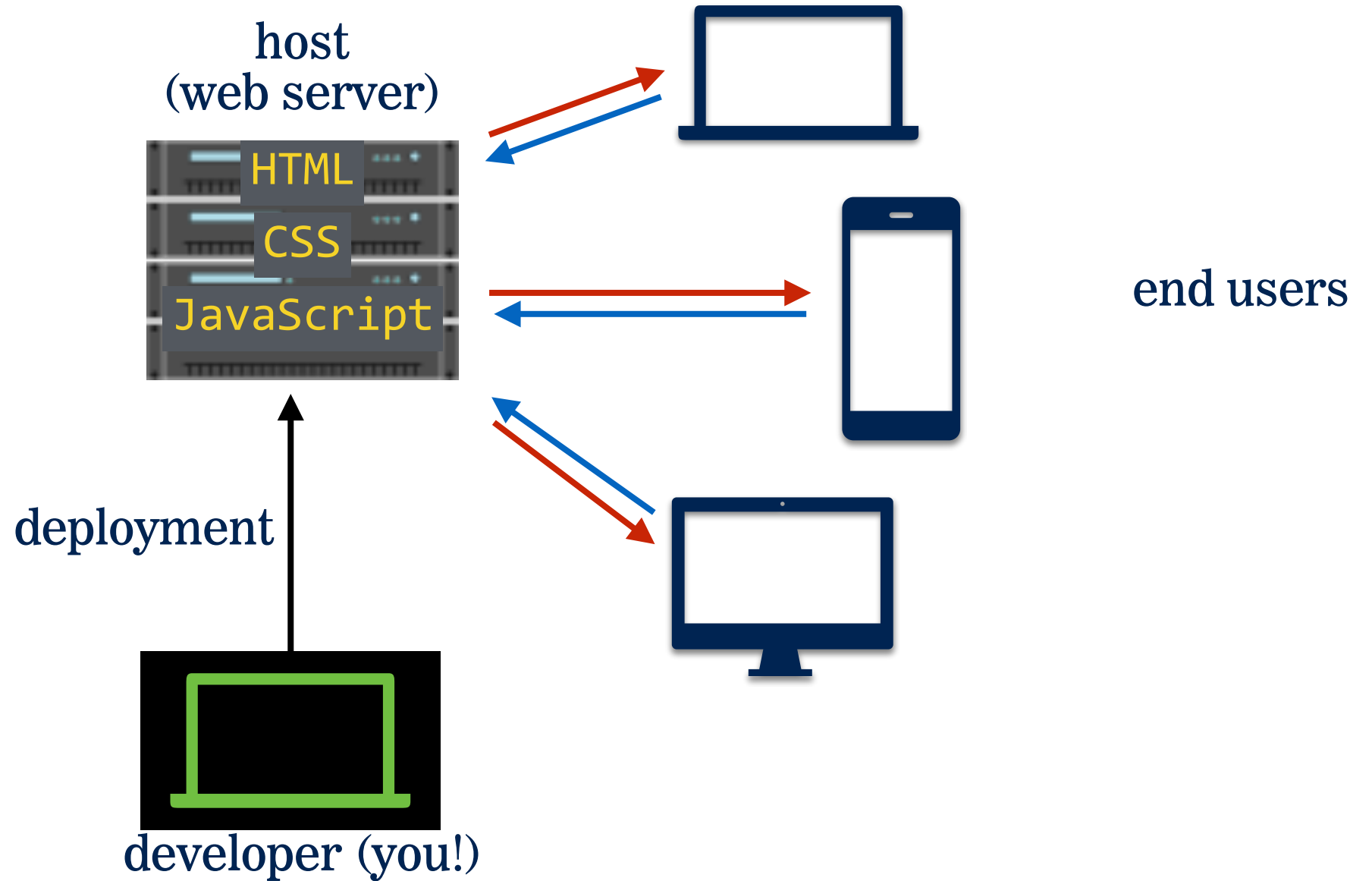


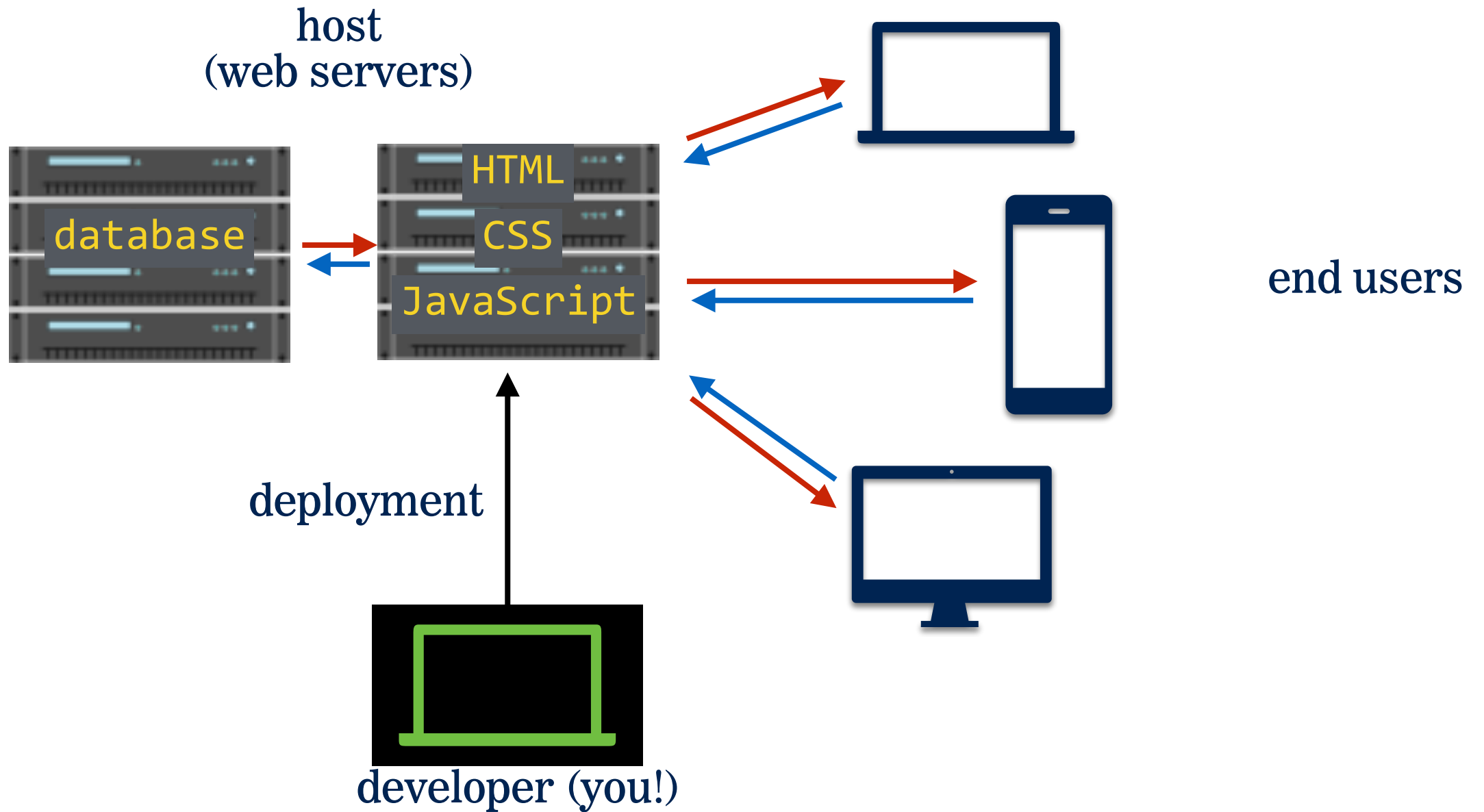
---

**DEPLOYING YOUR APP**

---









# **DEPLOYMENT**







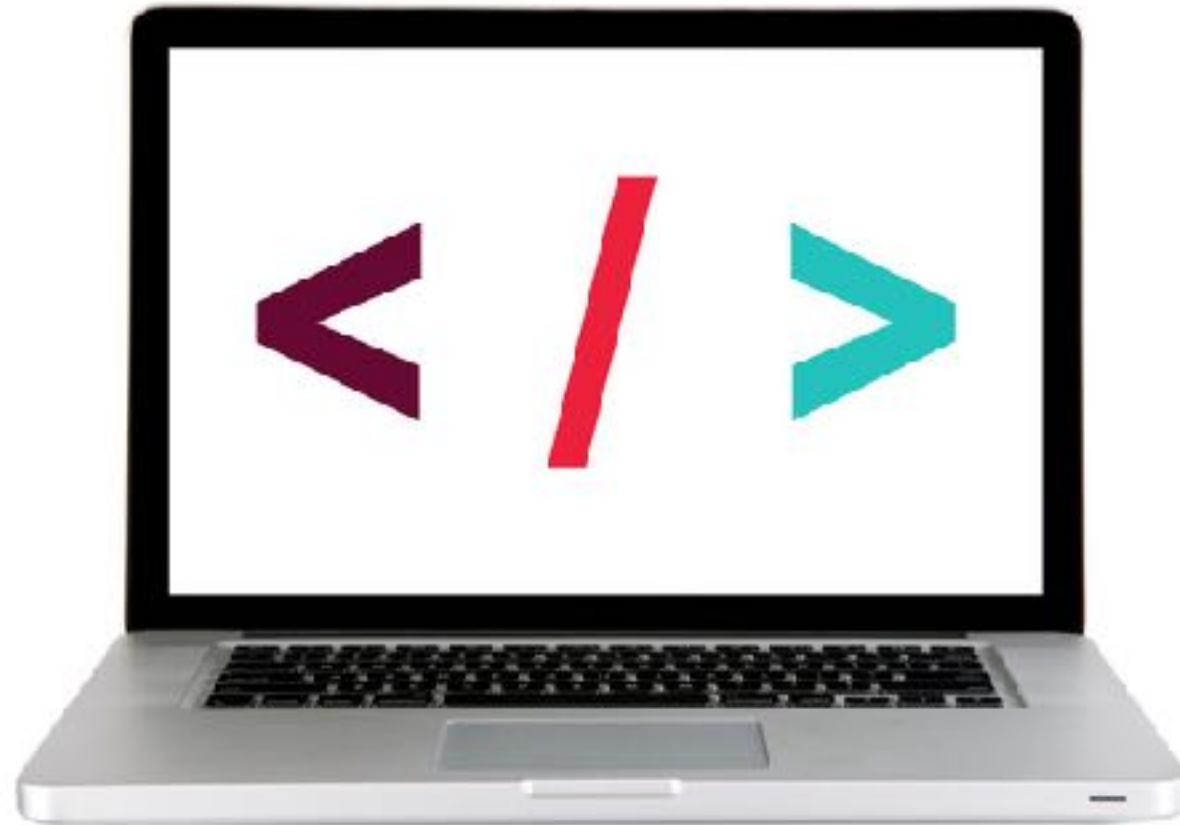
# ALTERNATIVE “SERVERLESS” SERVICES

 <p><b>Google Firebase</b></p> <p><b>Relevant Capabilities</b></p> <ul style="list-style-type: none"> <li>Auth</li> <li>Realtime Database</li> <li>Media Storage</li> <li>Cloud Functions</li> </ul> <p><b>Quick Overview</b></p> <p>Google Firebase is very powerful while being very easy to use. For example, you can run cloud functions, but you don't even need to for most data storage and retrieval or auth. It might be expensive to scale on though.</p>	 <p><b>Google Cloud Platform</b></p> <p><b>Relevant Capabilities</b></p> <ul style="list-style-type: none"> <li>Everything</li> </ul> <p><b>Quick Overview</b></p> <p>More of a major infrastructure provider in vein of Amazon Web Services than a toolkit for building out an app like Firebase is.</p>	 <p><b>Amazon Web Services</b></p> <p><b>Relevant Capabilities</b></p> <ul style="list-style-type: none"> <li>Everything</li> </ul> <p><b>Quick Overview</b></p> <p>Lambda, API Gateway, S3, and Cognito (auth) are probably the most relevant things to front-end developers. <a href="#">AACSync</a> is a ht like Firebase. There are frameworks that help you deploy to Lambda, like <a href="#">Craffie</a> and <a href="#">Functional Fleet</a>.</p>	 <p><b>Microsoft Azure</b></p> <p><b>Relevant Capabilities</b></p> <ul style="list-style-type: none"> <li>Everything</li> </ul> <p><b>Quick Overview</b></p> <p>A major infrastructure provider with solutions for about just everything, and generally considered the cheapest. For working with cloud functions, there is an online editor, but it also allows Git-hub sync and <a href="#">integrates directly with VS Code</a>. Data storage is through Cosmos DB.</p>	 <p><b>StdLib</b></p> <p><b>Relevant Capabilities</b></p> <ul style="list-style-type: none"> <li>Cloud Functions</li> </ul> <p><b>Quick Overview</b></p> <p>StdLib is based on Function as a Service ("server-less") architecture, popularized by AWS Lambda. You can use StdLib to build modular, scalable APIs for yourself and other developers in minutes without having to manage servers, gateways, domains, write documentation, or build SDKs. They also offer <a href="#">Code.xyz</a>, and online code editor for working with the APIs.</p>	 <p><b>Webtask</b></p> <p><b>Relevant Capabilities</b></p> <ul style="list-style-type: none"> <li>Cloud Functions</li> <li>Basic JSON data store</li> </ul> <p><b>Quick Overview</b></p> <p>An in-browser editor for creating and testing cloud functions. Seems like the nicest experience for this particular job. It's kinda of an elaborate demonstration of <a href="#">Auth0 Extend</a>, which is essentially a way to take Webtask and put it in your own app.</p>	 <p><b>IBM Cloud Functions</b></p> <p><b>Relevant Capabilities</b></p> <ul style="list-style-type: none"> <li>Cloud Functions</li> </ul> <p><b>Quick Overview</b></p> <p>Based on <a href="#">Apache OpenWhisk</a>.</p>	 <p><b>Backendless</b></p> <p><b>Relevant Capabilities</b></p> <ul style="list-style-type: none"> <li>Realtime Database</li> <li>Auth</li> </ul> <p><b>Quick Overview</b></p> <p>All-in-one kind of service similar to Firebase, including the realtime database. Has a PaaS version you can host yourself if you're, ya know, into running servers.</p>
--	--	---	---	---	--	--	---

---

**LET'S TAKE A CLOSER LOOK**

---



---

# EXERCISE — PUSH CHANGES TO FIREBASE

---



EXERCISE

## KEY OBJECTIVE

---

- › Deploy to a web host.

## TIMING

---

*5 min*

1. Make a change to the HTML, CSS, and/or JavaScript for the project you deployed to Firebase.
2. Push your changes to Firebase and verify that your updated code is what you see in your browser at `appname.firebaseio.com`

# **Exit Tickets!**

**(Class #16)**

---

# **LEARNING OBJECTIVES – REVIEW**

- Understand what hosting is.
- Identify a program's needs in terms of host providers.
- Ensure backward compatibility by using Babel to transpile code.
- Deploy to a web host.

# **NEXT MONDAY PREVIEW**

## **Intro to React**

- Understand the roles of model, view, and controller
- Describe the difference between frameworks and libraries
- Recognize the primary uses of React
- Create a component hierarchy
- Build a React component

# **Q&A**