



CS371

Web Application Development

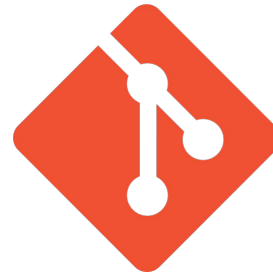
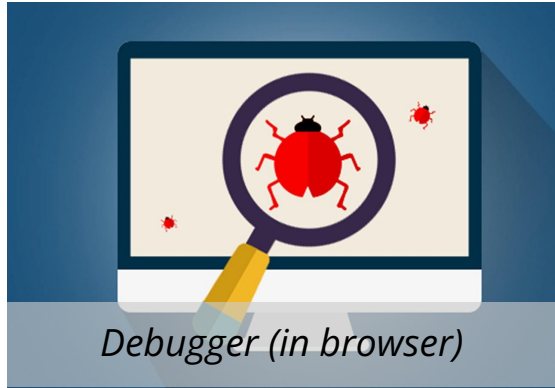
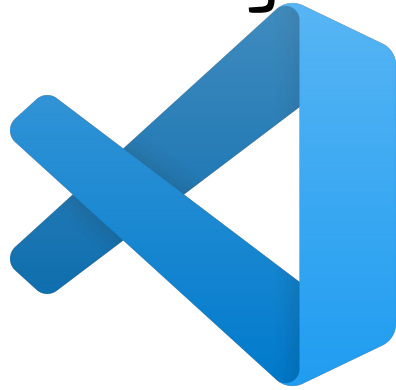
Hans Dulimarta



Course Logistics

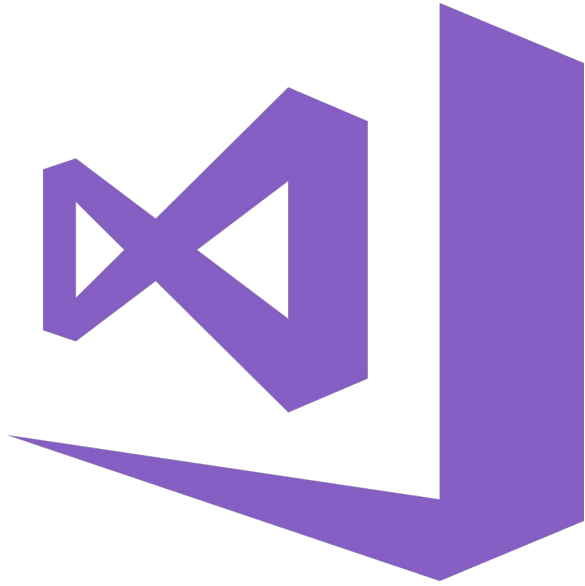
- No assigned textbook
 - most materials about web development is available online
- Bb and the instructor's teaching web site
 - Office hours (via Calendly)
 - Weekly Schedule
 - Assignments
- Zoom session (in case of snow days)
 - Login via <https://gvsu-edu.zoom.us>
 - Use the link and password posted on Bb

Programming Tools

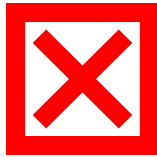


git





 Visual Studio



Visual Studio Code



Prereqs

- Fluent in Java (or other OO Languages)
 - You should be able to solve most problems at codingbat.com in a couple of minutes. *If you struggle in solving these problems, then **you are not ready** to take this course*
- Self Learner
 - Proficient in high-level programming **concepts**, and able to teach yourself the basics of other C-like languages (Java | Type)Script
- Good understanding of OO techniques: inheritance, methods, interface, ...

Expected Java Fluency

- Accessing object properties (without using a “getter” functions)
- Using loops on arrays of objects
- Writing own functions/methods
- Passing arguments into functions
- Function return value
 - Returning “result” from a function
 - Using a function “result”

Warming Up

- Brief Instructor introduction
- Individual introduction
 - Name and what do you want to be called
 - Background experience in web work
 - Specific topics you seek to learn from this class

What is Web Programming?

Group Discussions

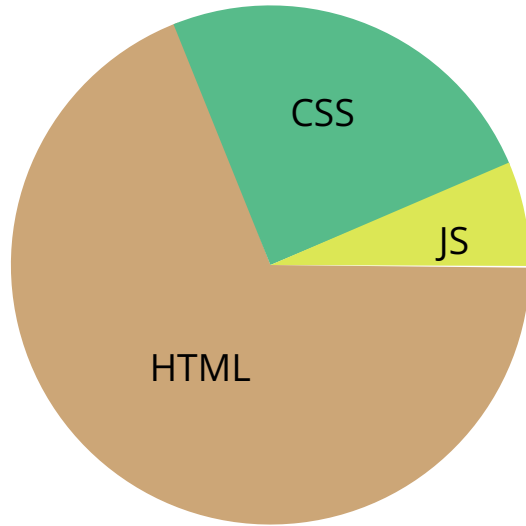
Unique characteristics of web apps?

Web Apps \neq Web Pages



CS 371 (this course)

Web Pages

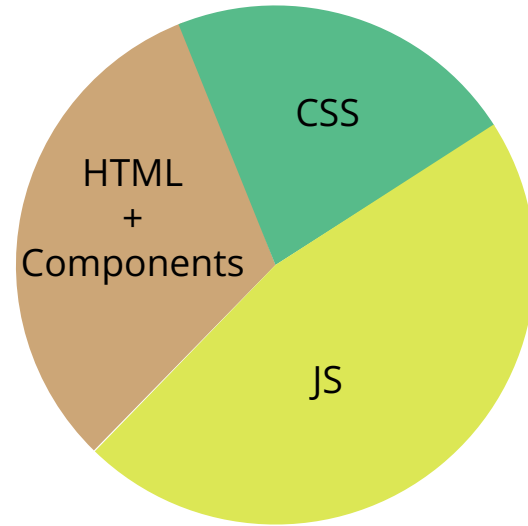


Static Web Pages

Web 1.0: read-only web

vs.

Web Apps



Web Apps

Web 2.0: dynamic read-write web

Desktop Apps

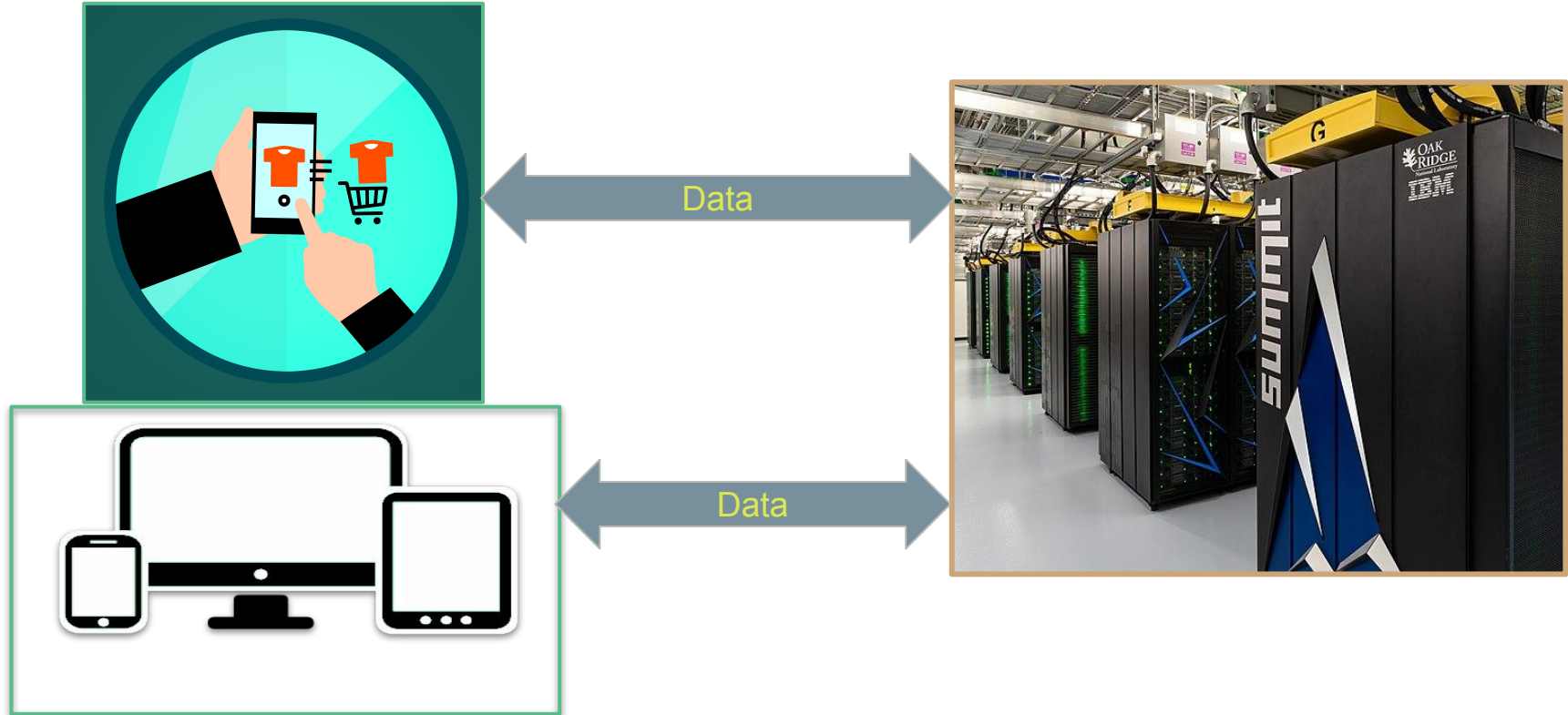
vs.

Web Apps

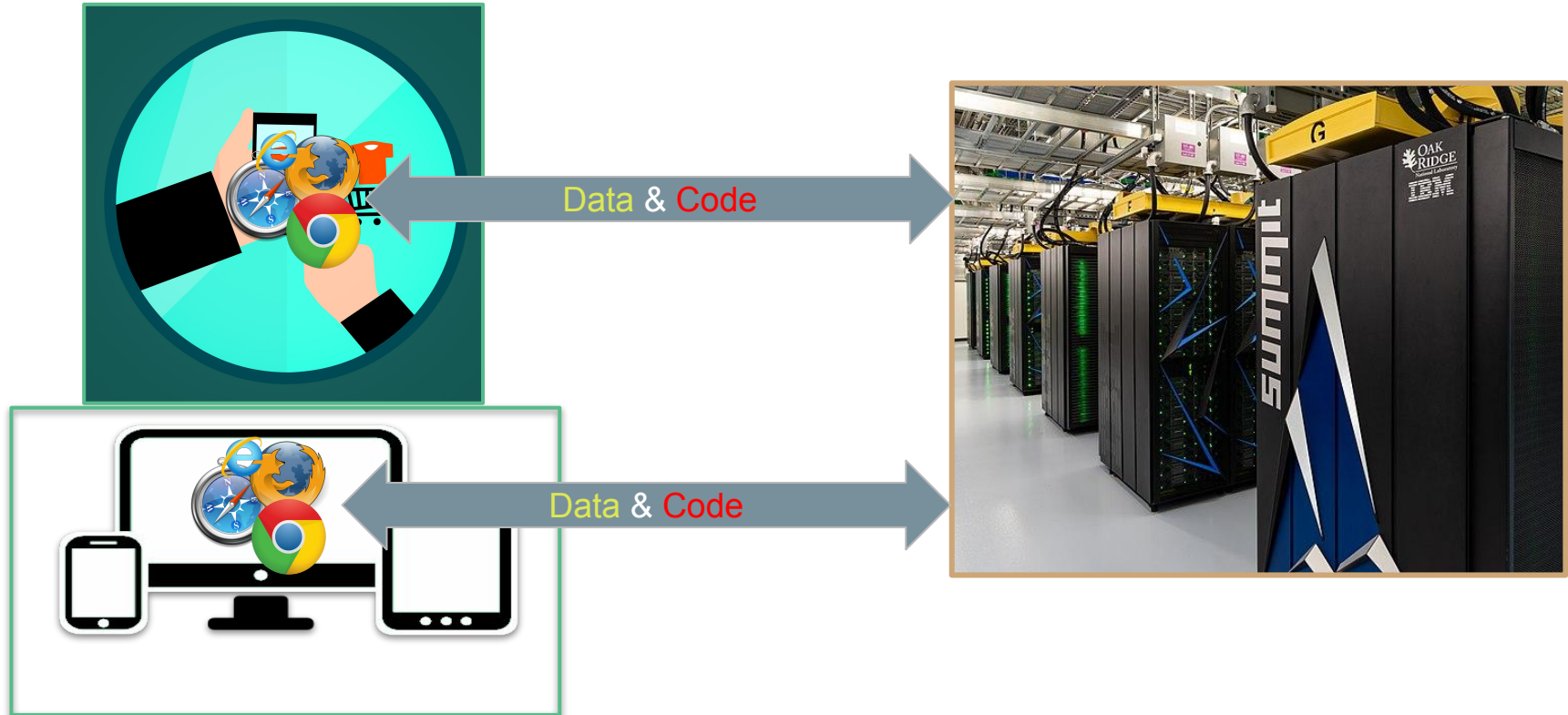
vs.

Mobile Apps

Traditional Client/Server architecture



Web Apps: Client/Server architecture



Roles of Web Browsers in Web apps

- **Present data**

- HTML + Text + Audio + Video + Image
- Content animation (CSS)
- 2D Graphics or 3D Graphics (WebGL) on <canvas>

- **Receiver user input**

- Textual input
- Mouse clicks / screen taps
- *Screen orientation (gyroscope on smartphones) ⇒ WebVR*

- **Run code**

- JavaScript (engines: Google **V8**, Mozilla **SpiderMonkey**, Apple **JSCore**, Microsoft **Chakra**)
- *Web Assembly (proposal since 2017)*

Web Client/Server Architecture

(3) Send "contents" (HTML + CSS + JS & **other data**)

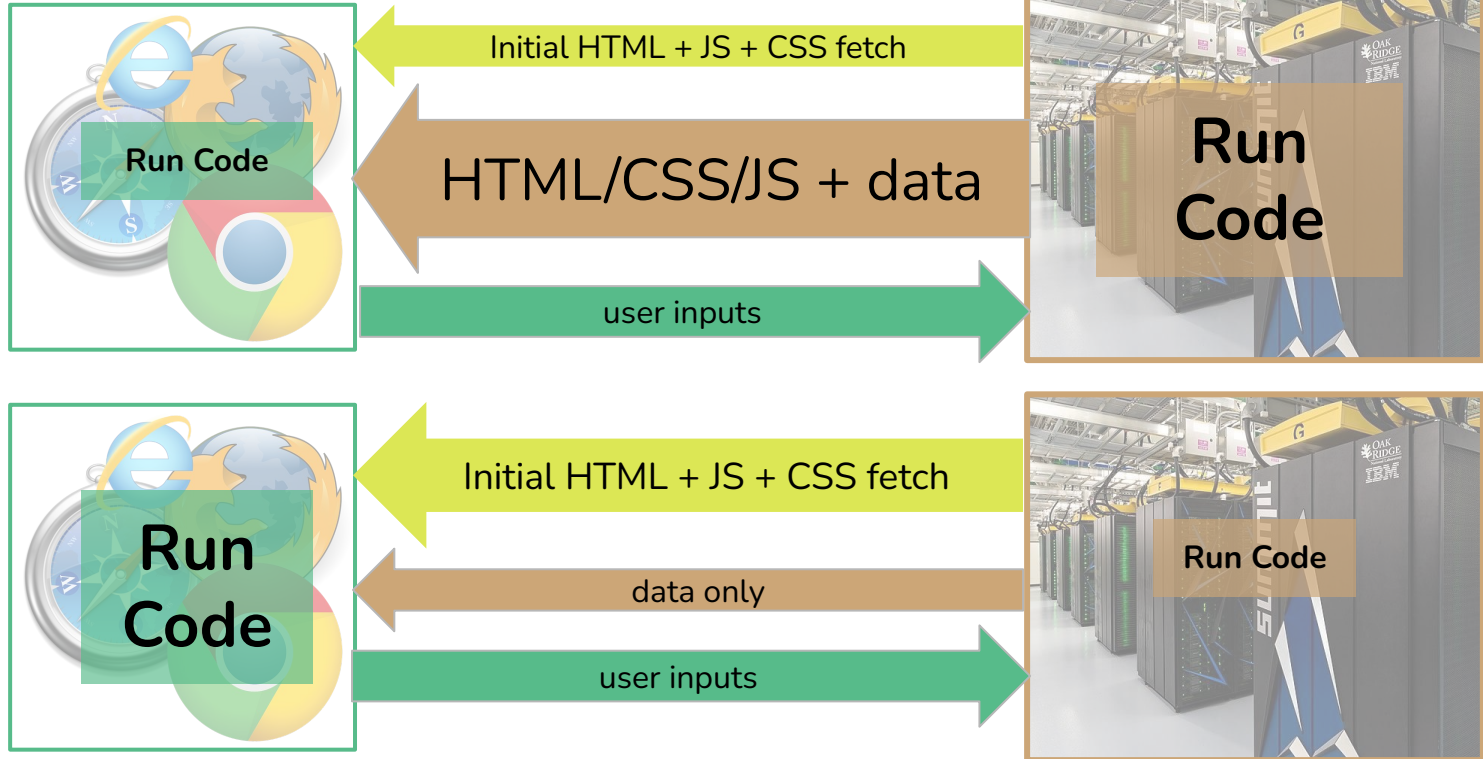
(4) Present "contents"

(6) Handle user input

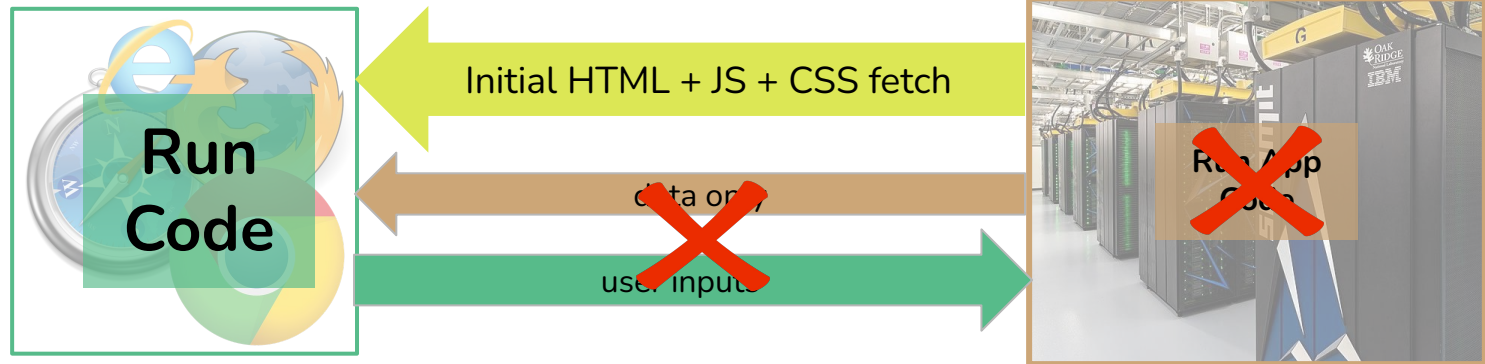


(1) Send "user input"

Client-Side vs. / Server-Side Programming

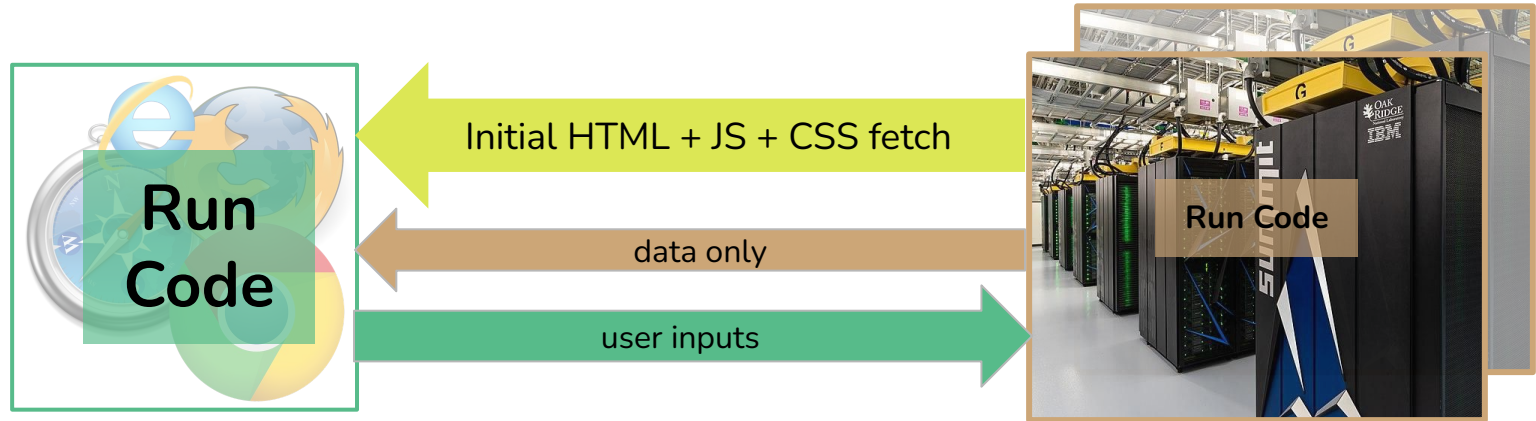


Web 1.0: Static web pages



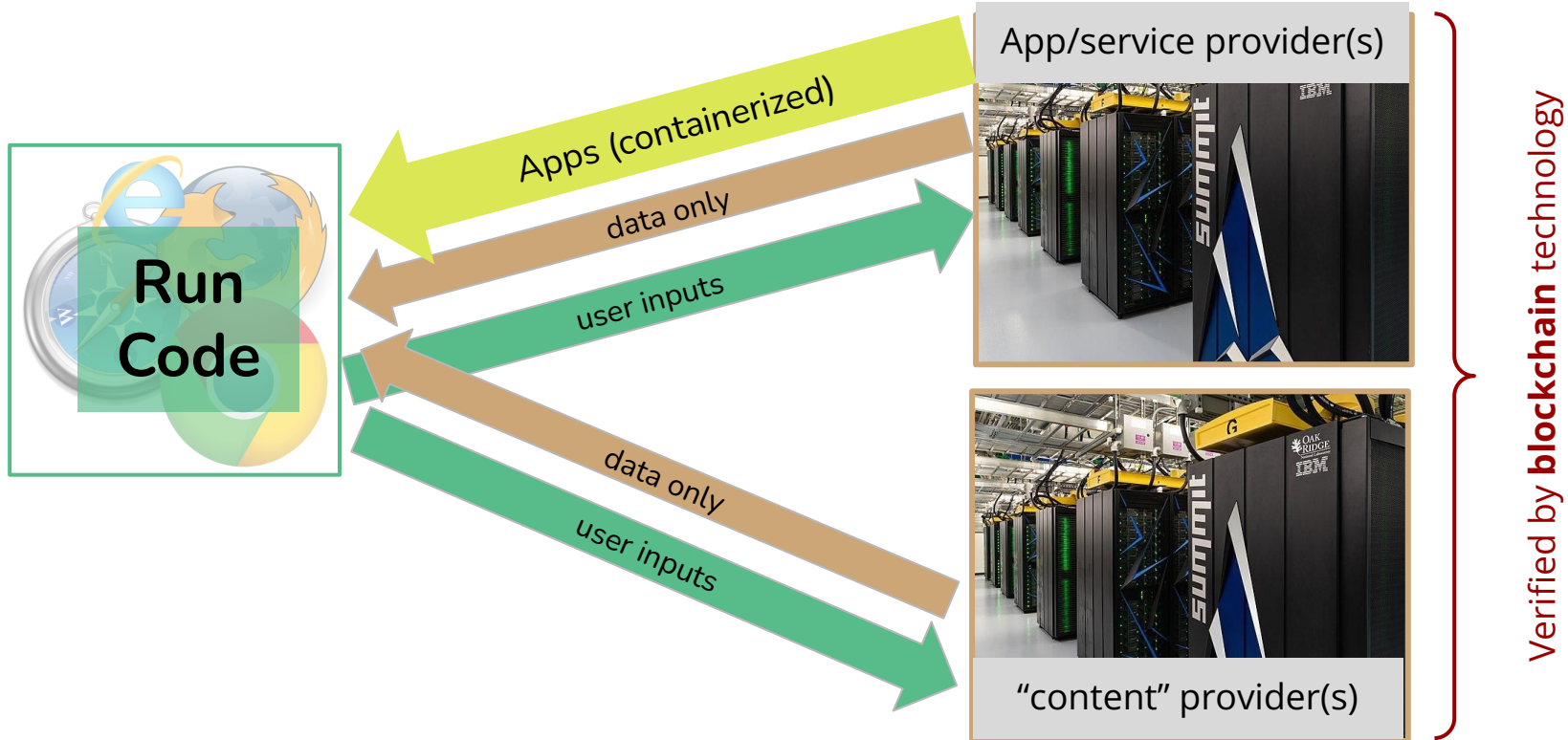
static web pages
hosting services


Web 2.0: Client/Server (dynamic R/W web apps)




Centralized
app hosting platforms

Web 3.0: Peer-to-Peer (decentralized providers)





Web App 2.0 \Rightarrow Web DApps 3.0

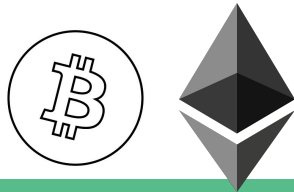




Interested in building dApps?

Try hardhat.org and [Ethereum Scaffold](#)





Block chain of transactions



heroku



Block chain of data

Block chain of apps (code) & services

	Web 2.0 (centralized)	Web 3.0 (decentralized)
Computing Engine	AWS, Heroku, Netlify	Solidity (smart contracts), App containers (“Docker”)
Data Storage	Amazon S3, Azure, Google Cloud	IPFS + Blockchain technology
Data Source	3rd party API	same 3rd party API
Monetization	Advertising	NFTs (proof of digital ownership)
Payments	PayPal, Visa, ...	Cryptocurrency

