Tampereen yliopisto
Tampere University

COMP.CS.510 Web Development 2 - Architecting

Web applications, Web as an application platform

David Hästbacka

Content

- Web applications (basics)
- Web as an application platform
- SPA

Course topics outline

- **Web as a platform and means of communication, web application components**
- **Single-page Applications**, software and systems architecture considerations
- Web Services (REST, SOAP, …)
- Service Oriented Architecture, API ecosystems, service composition
- Other web based communication (web sockets, asynchronous communication, message buses, DDS, OPC UA, …)
- Microservices, serverless computing and FaaS, event-driven scalability (implementing back end logic)
- Information security and ensuring Quality of Service in distributed web applications
- Data and semantics for interoperable web applications (beyond JSON and XML, information models, …)
- Applications of distributed web applications and as the means of communication
  - IoT, Industrial Internet, Industry4.0, Smart Cities …

Web Application Node.js example

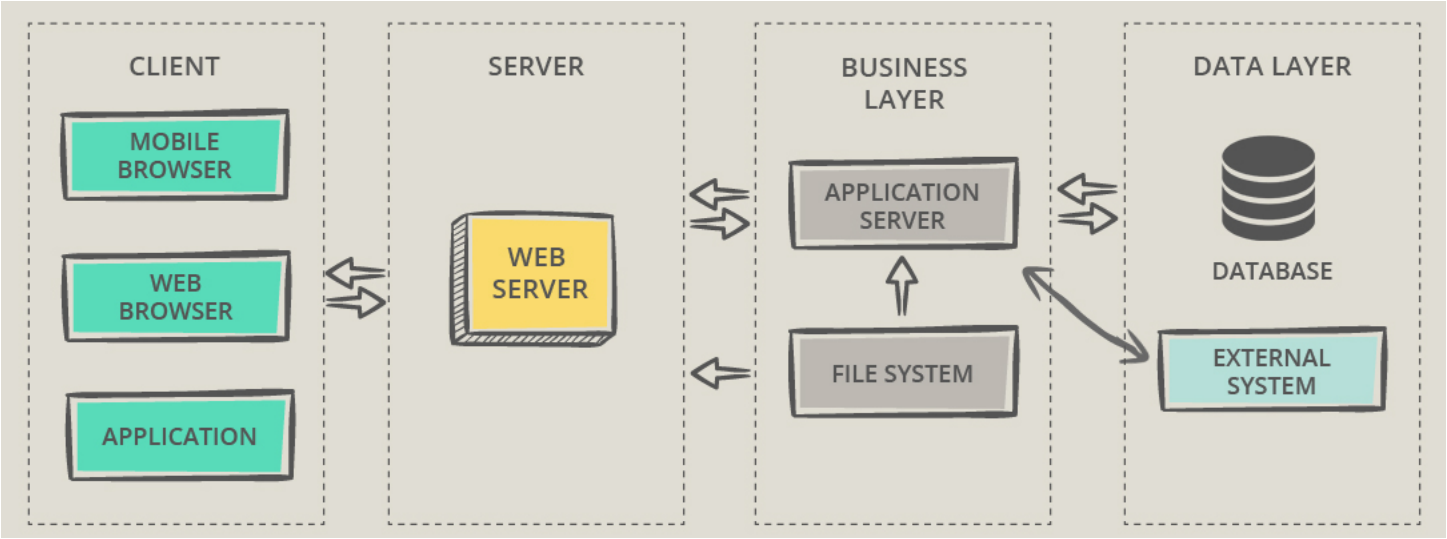


Figure: https://existek.com/blog/web-application-architecture/

https://ilovecoding.org/courses/nodejs/lessons/creating-a-simple-web-app-with-nodejs

webapp.js

```
var http = require('http');

http.createServer(onRequest).listen(8888);
console.log('Server has started');

function onRequest(request, response){
  response.writeHead(200);
  response.write('Hello Noders');
  response.end();
}
```

```
webapp >> node webapp
Server has started
```

localhost:8888/contact

```
Hello Noders
```

Hello Noders shows up, regardless of what was in the URL

https://ilovecoding.org/courses/nodejs/lessons/creating-a-simple-web-app-with-nodejs

webapp.js

```
var http = require('http');
var url  = require('url');

http.createServer(onRequest).listen(8888);
console.log('Server has started');

function onRequest(request, response){
  var pathName = url.parse(request.url).pathname;
  console.log('pathname' + pathName);
  response.writeHead(200);
  response.write('Hello Noders');
  response.end();
}
```

webapp — node — 37×27

```
webapp >> node webapp
Server has started
^X^Cwebapp >> node webapp
Server has started
```

https://ilovecoding.org/courses/nodejs/lessons/creating-a-simple-web-app-with-nodejs

webapp.js

```
var http = require('http');
var url  = require('url');

http.createServer(onRequest).listen(8888);
console.log('Server has started');

function onRequest(request, response){
  var pathName = url.parse(request.url).pathname;
  console.log('pathname' + pathName);
  showPage(response)
}

function showPage(response){
  response.writeHead(200);
  response.write('Hello Noders');
  response.end();
}
```

webapp — node — 37×27

```
webapp >> node webapp
Server has started
^X^Cwebapp >> node webapp
Server has started
pathname/asdasasd
pathname/asdasasd
^Cwebapp >> node webapp
Server has started
pathname/asdasasd
```

https://ilovecoding.org/courses/nodejs/lessons/creating-a-simple-web-app-with-nodejs

webapp.js

```
var http = require('http');
var url  = require('url');

http.createServer(onRequest).listen(8888);
console.log('Server has started');

function onRequest(request, response){
  var pathName = url.parse(request.url).pathnam
  console.log('pathname' + pathName);
  showPage(response, pathName)
}

function showPage(response, pathName){
  if(pathName === '/'){
    response.writeHead(200, {'Content-Type': 't
    response.write(contentMap['/']);
    response.end();
  }
}

var contentMap = {
  '/' : '<h1>Welcome to the site</h1>',
  '/contact' : '<h1>Contact Page</h1>'
}
```

```
webapp >> node webapp
Server has started
^X^Cwebapp >> node webapp
Server has started
```

localhost:8888

Welcome to the site

https://ilovecoding.org/courses/nodejs/lessons/creating-a-simple-web-app-with-nodejs

Terminal Shell Edit View Window Help

webapp.js — webapp

webapp.js

```
8   var pathName = url.parse(request.url).pathname;
9   console.log('pathname' + pathName);
10  showPage(response, pathName)
11 }
12
13 function showPage(response, pathName){
14   if(contentMap[pathName]){
15     response.writeHead(200, {'Content-Type': 'tex
16     response.write(contentMap[pathName]);
17     response.end();
18   }else {
19     response.writeHead(404, {'Content-Type': 'tex
20     response.write('404 Page Not Found');
21     response.end();
22   }
23 }
24
25 var contentMap = {
26   '/' : '<h1>Welcome to the site</h1>',
27   '/contact' : '<h1>Contact Page</h1>',
28   '/about' : '<h1>About Page</h1>',
29   '/users' : '<h1>Users Page</h1>',
30   '/privacy' : '<h1>Privacy Page</h1>'
31 }
```

Line 29, Column 36 Spaces: 2 JavaScript

Chrome File Edit View History Bookmarks Window People Help

localhost:8888/about

iLoveCoding

localhost:8888/about

About Page

If you go to any page that you have created it will show in your browser, else an error 404 Page was not found will show.

Add commas at the end

Different types of web applications
Basic legacy web app

- The server outputs HTML shown in the browser, typically the entire page
- The data is more secure as all data is ”hidden”
- Increases data exchange, server work load, latencies, …

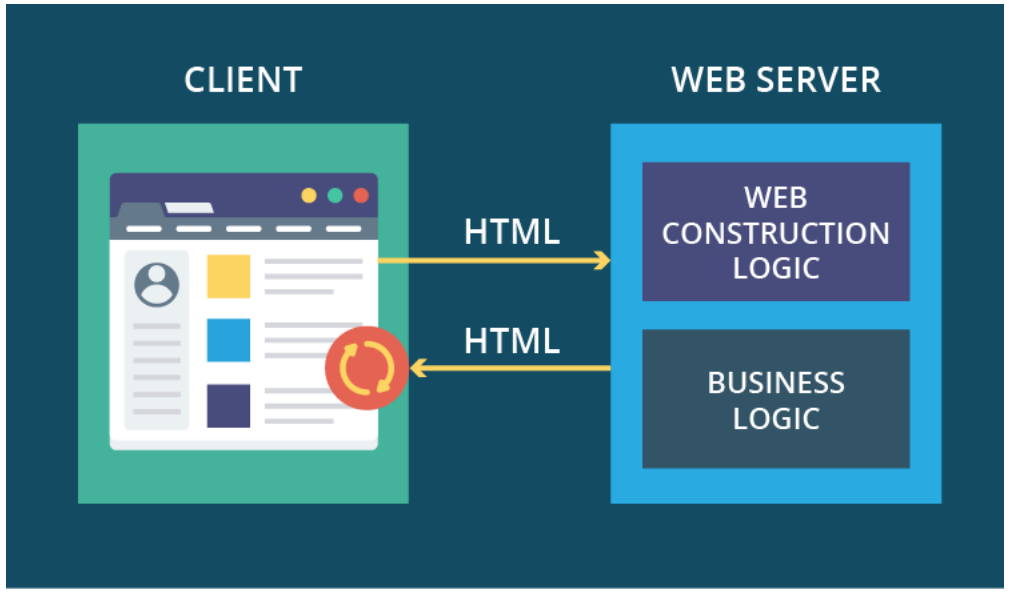


Figure: https://www.scnsoft.com/blog/web-application-architecture

Different types of web applications
Widget based web app

- Application is separated into sections as widgets that independently display HTML or render JSON/XML content
- No reload of entire page, more dynamic behavior, …
- Part of the application logic is exposed to the client side

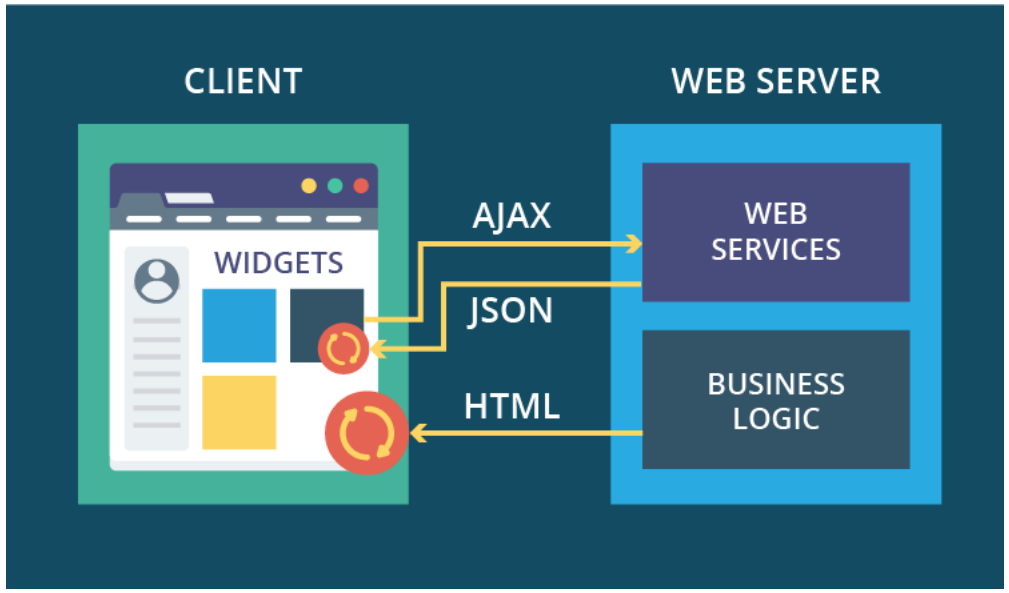


Figure: https://www.scnsoft.com/blog/web-application-architecture

Different types of web applications
Single-page web app

- UI Client is downloaded once
- The client has a JavaScript layer that communicates with the server
- Fast, efficient, reduces server work load but increases client (browser) work load
- All client application logic is visible, all data sources (on the server API) are visible

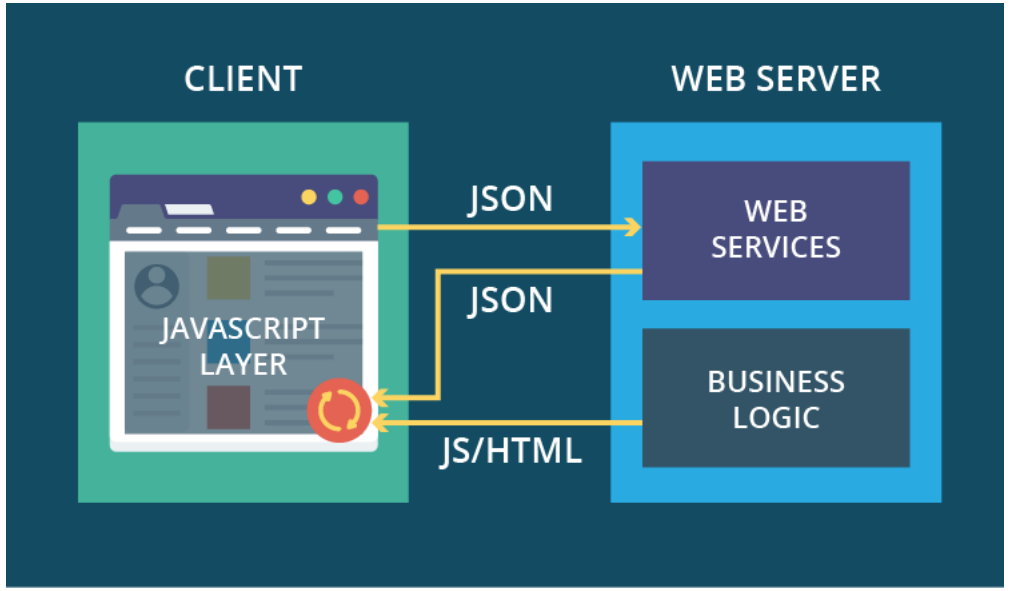
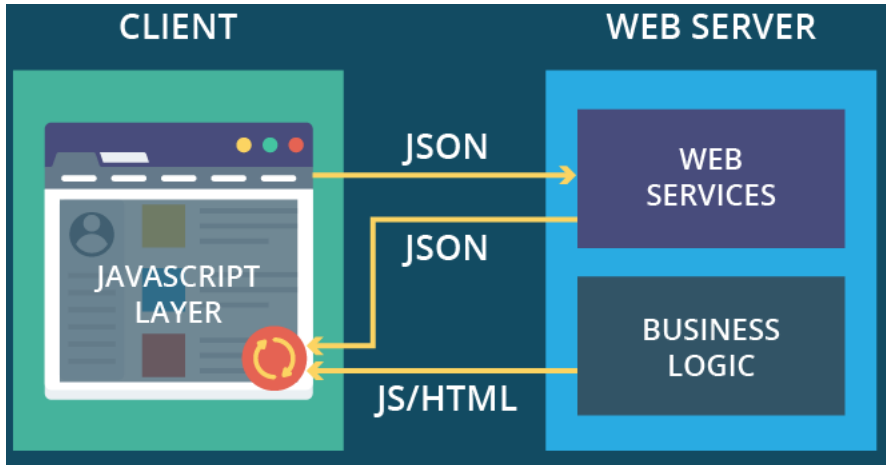


Figure: https://www.scnsoft.com/blog/web-application-architecture

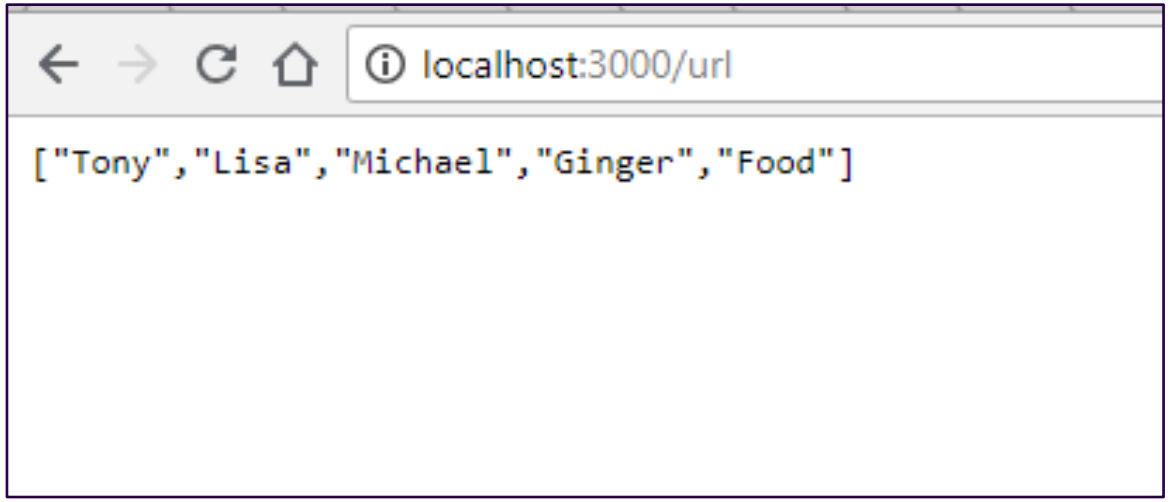
Figure: https://www.scnsoft.com/blog/web-application-architecture



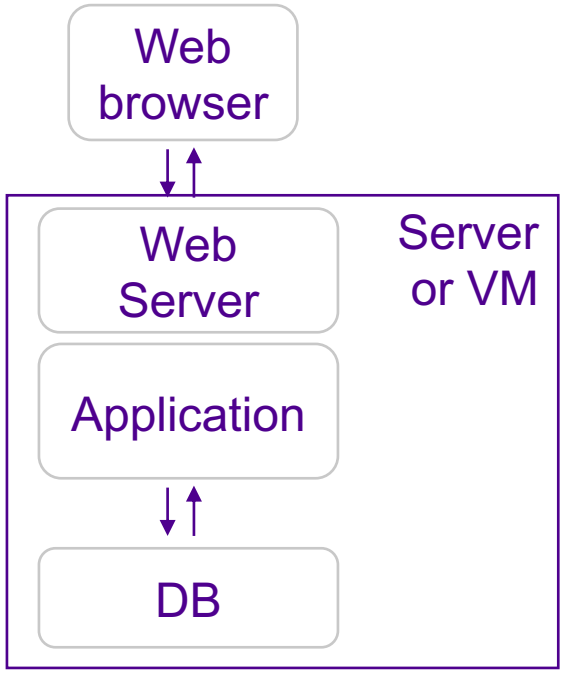
Single page application

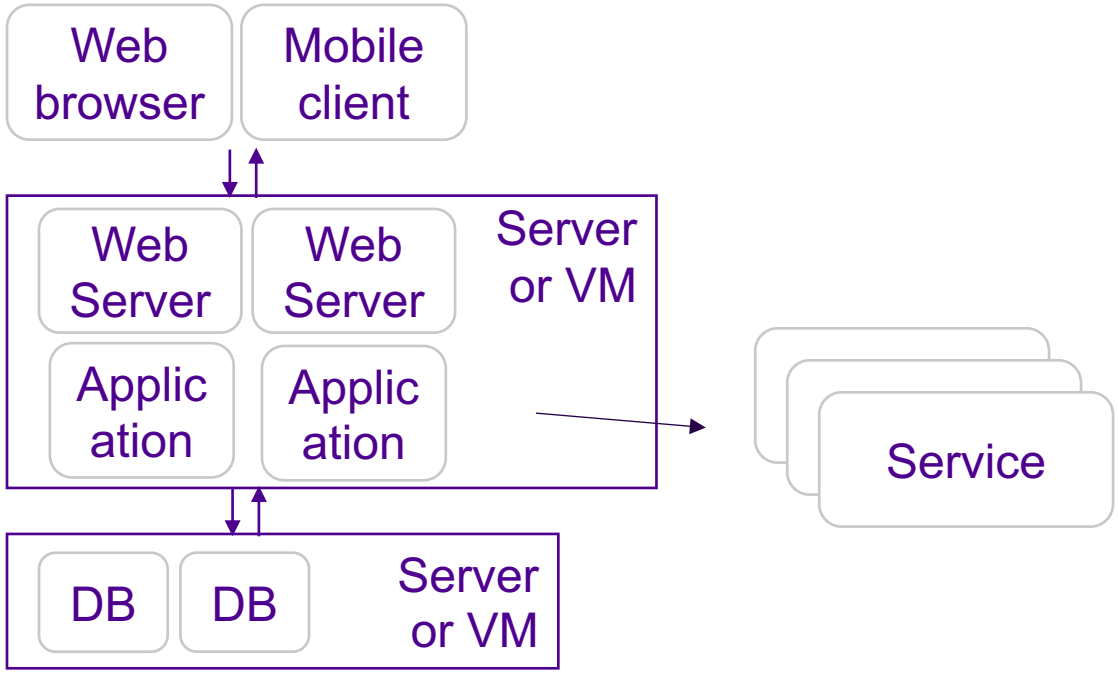
- Load HTML, static content, JS-files, … and start client
- Next, do mainly content calls
- E.g. in Node.JS

```
var express = require("express");
var app = express();

app.listen(3000, () => {
  console.log("Server running on port 3000");
});

app.get("/url", (req, res, next) => {
  res.json(["Tony", "Lisa", "Michael", "Ginger", "Food"]);
});
```



Web application components?





Choosing platforms

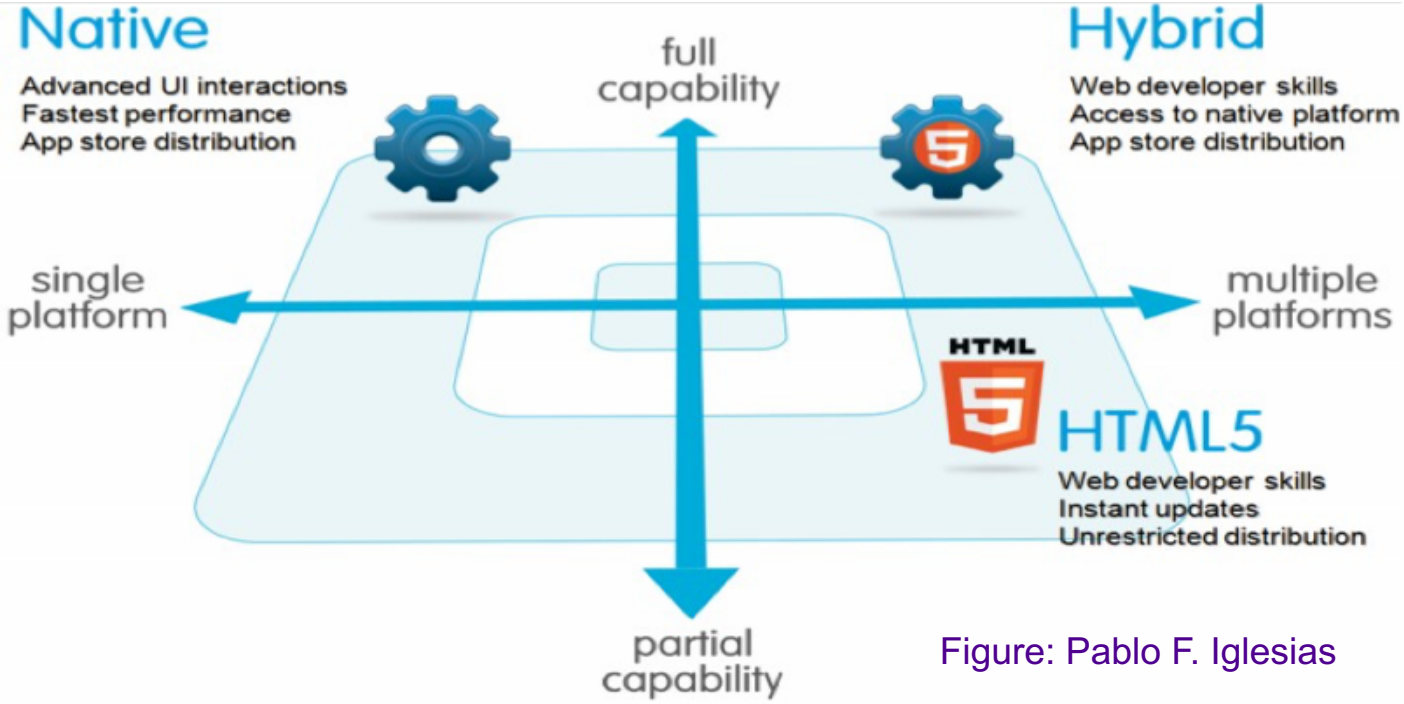


Figure: Pablo F. Iglesias

Progressive Web Apps
(Mohan Ram.H.R)

- What is PWA?
  - A Progressive Web App uses modern web capabilities to deliver an app-like user experience.
  - They are installable and live on the user's **home screen**, without the need for an app store.
- Why PWA?
  - App like experience
  - Push notifications
  - Working offline
  - Adding as icon in home screen

Progressive Web Apps
(Mohan Ram.H.R)

- Core tenets of PWA
  - Service Workers
    - They power offline functionality, push notifications, background content updating, content caching etc
  - App Shell
    - Simple design concept to define the application initial load to initiate the application.
  - App Manifest
    - Helps to install/add application to home screen via install banners

HTML5 add power to web applications

(Programming HTML5 Applications by Zachary Kessin)

- Local data storage - It can store up to 5 MB of data, referenced with a key-value system.
- Databases - Originally a SQLite-based API, the tide seems to have shifted to IndexedDB, a NoSQL system that is natively JavaScript.
- Files - While applications still can’t freely access the filesystem (for obvious security reasons), they can now work with files the user specifies and are starting to be able to create files as well.
- Taking it offline - When a laptop or phone is in airplane mode, web applications are not able to communicate with the server. Manifest files help developers work around that by caching files for later use.
- Web Workers - Threads and forks have always been problematic, but JavaScript simply didn’t offer them. Web Workers provide a way to put application processes into separate spaces where they can work without blocking other code.
- Web sockets - Hypertext Transfer Protocol (HTTP) has been the foundation of the Web, despite a few updates over time. Web sockets transform the request-response approach to create much more flexible communication systems.

Ionic as an example framework

- ”The open source Ionic Framework features a rich library of front-end building blocks and UI components that make it easy to design beautiful, high-performance mobile and Progressive Web Apps (or PWAs) using web technologies like HTML, CSS, and JavaScript.”
- ”Our universal web components pair with any JavaScript framework, including Angular, React, Vue, or no framework at all (just add a script tag!). Ionic apps are backend agnostic, with connections to AWS, Azure, and Firebase.”

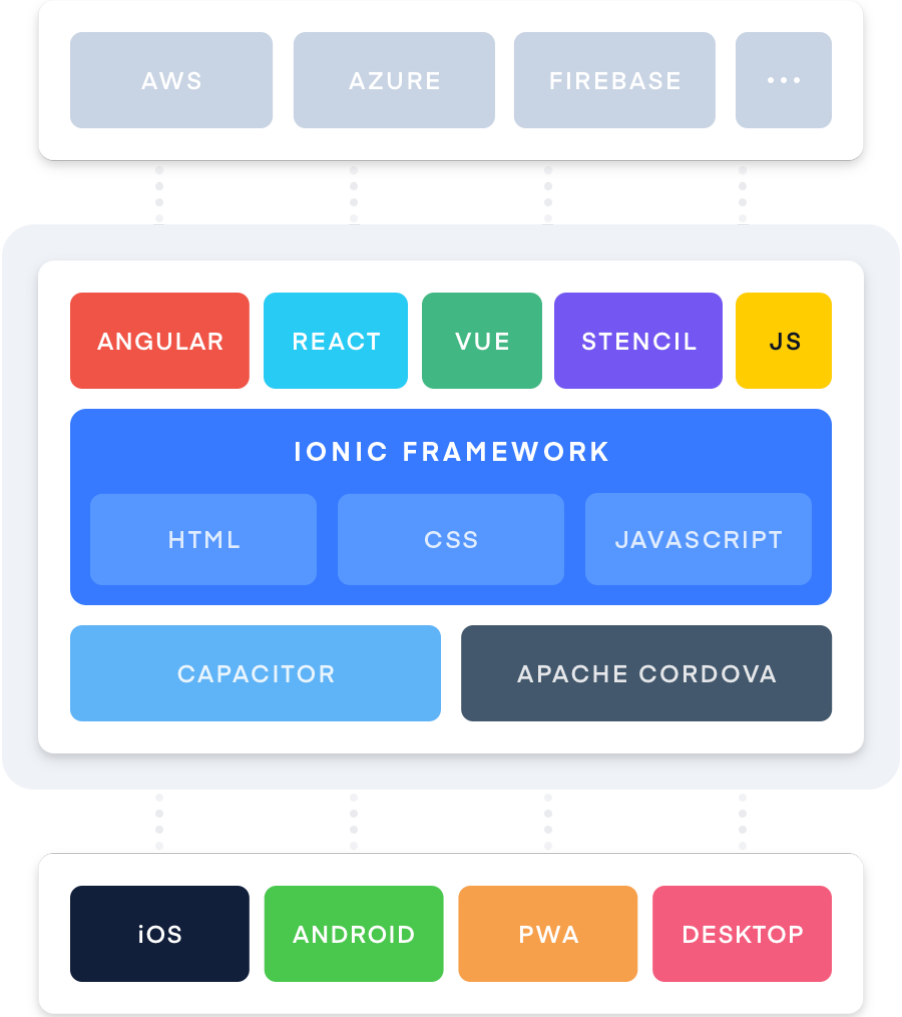


Figure: Ionic

Web as a Platform

- What can be done?
- https://whatwebcando.today

- Almost a native application like experience

Camera & Microphone

- AUDIO & VIDEO CAPTURE ✓
- ADVANCED CAMERA CONTROLS ✗
- RECORDING MEDIA ✗
- REAL-TIME COMMUNICATION ✓

Advertisement

Surroundings

- BLUETOOTH ✗
- USB ✗
- NFC ✗
- AMBIENT LIGHT ✗

Device Features

- NETWORK TYPE & SPEED ✗
- ONLINE STATE ✓
- VIBRATION ✗
- BATTERY STATUS ✗
- DEVICE MEMORY ✗

Native Behaviors

- LOCAL NOTIFICATIONS ✓
- PUSH MESSAGES ✓
- HOME SCREEN INSTALLATION ✗
- FOREGROUND DETECTION ✓
- PERMISSIONS ✗

Operating System

- OFFLINE STORAGE ✓
- FILE ACCESS ✓
- CONTACTS ✗
- SMS ✗
- STORAGE QUOTAS ✗
- TASK SCHEDULING ✗

Input

- TOUCH GESTURES ✗
- SPEECH RECOGNITION ✗
- CLIPBOARD (COPY & PASTE) ✓
- POINTING DEVICE ADAPTATION ✓

Seamless Experience

- OFFLINE MODE ✓
- BACKGROUND SYNC ✗
- INTER-APP COMMUNICATION ✗
- PAYMENTS ✓
- CREDENTIALS ✗

Location & Position

- GEOLOCATION ✓
- GEOFENCING ✗
- DEVICE POSITION ✗
- DEVICE MOTION ✗
- PROXIMITY SENSORS ✗

Screen & Output

- VIRTUAL & AUGMENTED REALITY ✗
- FULLSCREEN ✓
- SCREEN ORIENTATION & LOCK ✗
- WAKE LOCK ✗
- PRESENTATION FEATURES ✗

https://whatwebcando.today