ThoughtWorks®
GLOBAL SOFTWARE CONSULTANCY

MICROSERVICES AND FRONTENDS

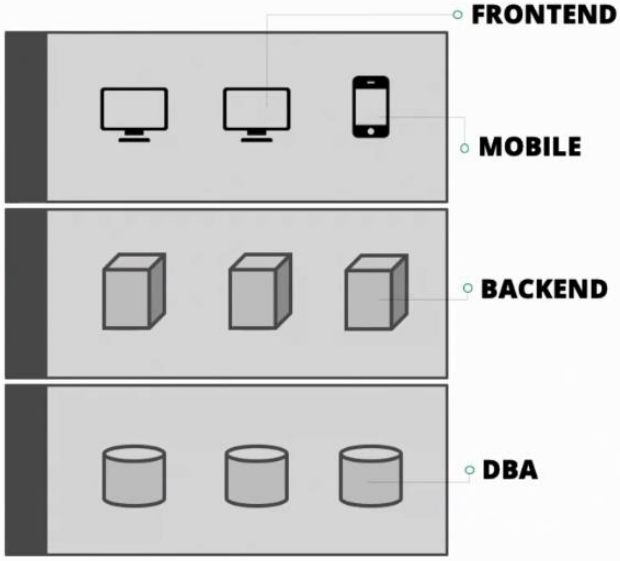
Erik Doernenburg | @erikdoe

PART 1

WHAT WERE THEY THINKING?

THE GOOD OLD LAYERED ARCHITECTURE

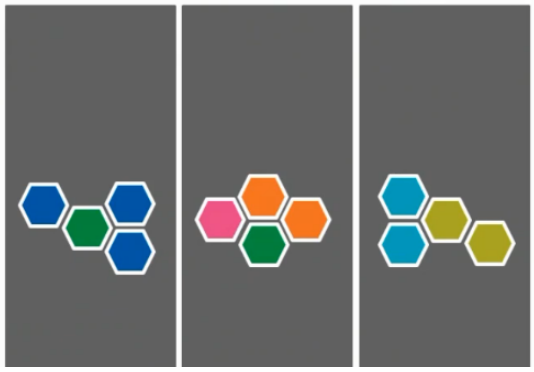
Tech Stacks



ARCHITECTURE BASED ON MICROSERVICES

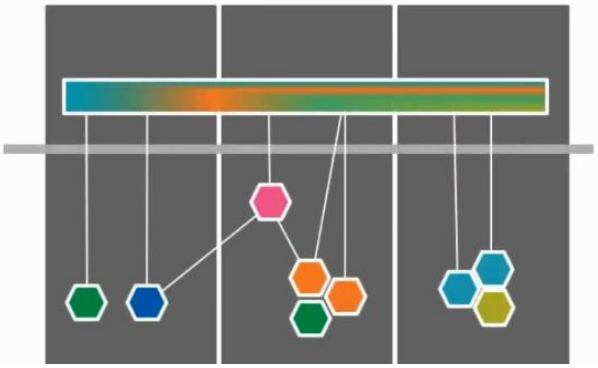
1) Independent Evolvability

2) The Last Rewrite



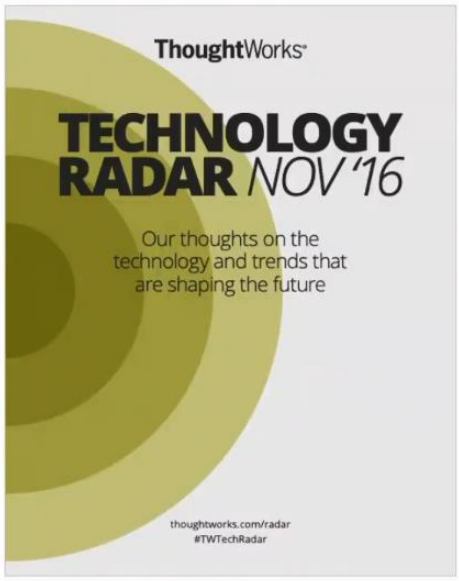
MORPHING INTO THE FRONTEND MONOLITH

What *were* they thinking?



A RELUCTANT NEOLOGISM MICRO FRONTENDS

"Teams have often struggled to avoid the creation of front-end monoliths—large and sprawling browser applications that are as difficult to maintain and evolve as the monolithic server-side applications we've abandoned."



BROWSER BULKS UP SERVER SLIMS DOWN

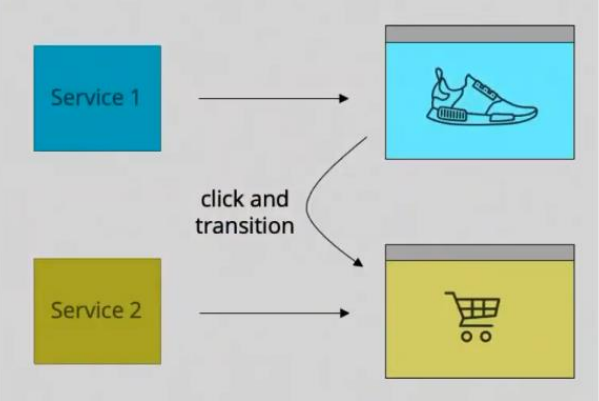
"The browser continues to expand its capabilities as a deployment target for application logic [and] we see a trend toward reduced complexity in back-end logic."

PART 2

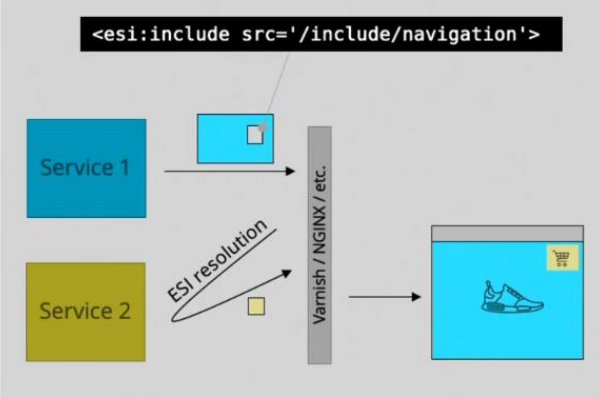
FRONTEND INTEGRATION PATTERNS

from simple to complex

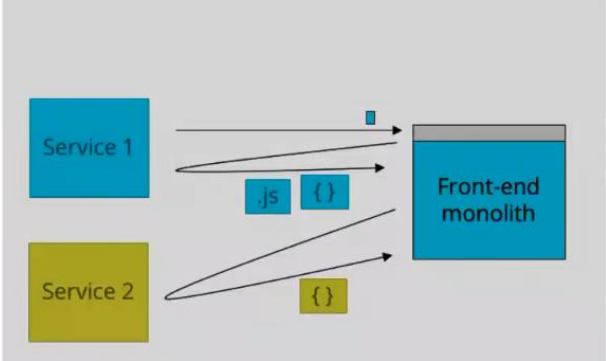
PATTERN 1:
THE WEB APPROACH



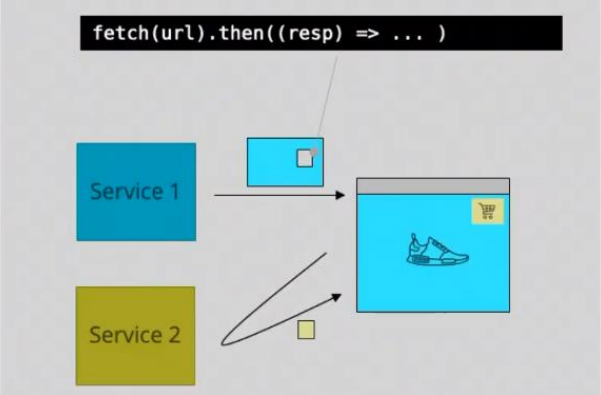
PATTERN 2:
SERVER-SIDE COMPOSITION



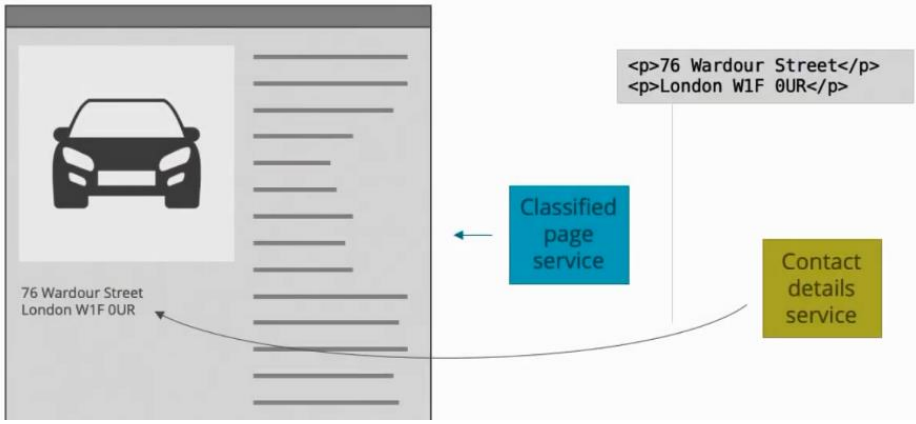
ANTIPATTERN #1:
CROSS-SERVICE DATA LOADING



PATTERN 3:
CLIENT-SIDE COMPOSITION

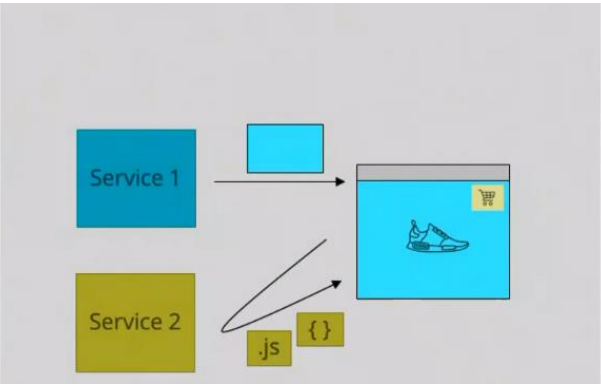


DETOUR
INDEPENDENT EVOLVABILITY



PATTERN 4:
CLIENT-SIDE RENDERING

Service provides minimal JS to consume its own API.

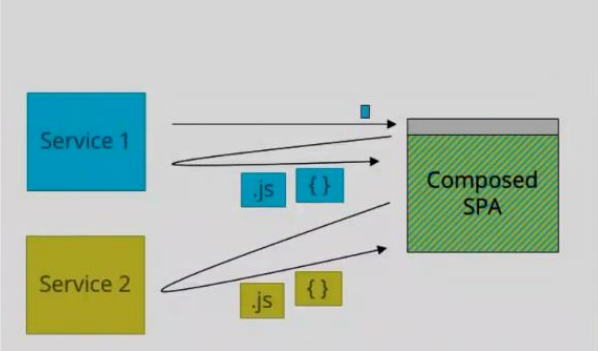


PATTERN 4: CLIENT-SIDE RENDERING

Service provides minimal JS to consume its own API.

```
<h2>Active Github repositories</h2>
<div id="personalrepos" class="githubrepos">
  <p class="loading">Loading...
</div>
<script type="text/javascript">
  $(document).ready(function(){
    github.showRepos({
      type: 'users',
      user: 'erikdoe',
      title: 'Personal',
      count: 5,
      target: '#personalrepos'
    });
  });
</script>
```

PATTERN 5: SPA/PWA COMPOSITION




```
const externals = {
  CartPreviewTile: DefaultComponent,
  cartServiceRoute: DefaultComponent,
  /* ... */
};
```

```
export function initExternals() {
  return loadFrontEndConfig().then(loadFrontEnds);
}

export default externals;
```

```
const externals = {
  CartPreviewTile: DefaultComponent,
  cartServiceRoute: DefaultComponent,
  /* ... */
};
```

```
function loadFrontEndConfig() {
  return fetch('/api/config', { credentials: 'same-origin' })
    .then(response => (response.ok ?
      response.json() :
      Promise.reject('Failed to load frontend config from /api/config')));
}
```

```
export function initExternals() {
  return loadFrontEndConfig().then(loadFrontEnds);
}

export default externals;
```


```
const externals = {
  CartPreviewTile: DefaultComponent,
  cartServiceRoute: DefaultComponent,
  /* ... */
};
```

```
function loadFrontEndConfig() {
  return fetch('/api/config', { credentials: 'same-origin' })
```


```
function loadFrontEnds(config) {
  loadStylesheet(config['cart-css']);
  const cartPromise = loadScript(config['cart-js'], 'cart')
    .then((exports) => {
      Object.assign(externals, exports);
      store.addReducer('cart', exports.cartReducers);
    });
  /* ... */
  return Promise.all([cartPromise /* ... */]);
}
```


```
  return loadFrontEndConfig().then(loadFrontEnds);
}

export default externals;
```

COUPLED
which components to load from other services

DECOUPLED
where to load the assets from

GENERIC
how to load the components' assets

```
const externals = {
  CartPreviewTile: DefaultComponent,
  cartServiceRoute: DefaultComponent,
  /* ... */
};

function loadFrontEndConfig() {
  return fetch('/api/config', { credentials: 'same-origin' })
    .then(response => (response.ok ?
      response.json() :
      Promise.reject('Failed to load frontend config from /api/config')));
}

function loadFrontEnds(config) {
  loadStylesheet(config['cart-css']);
  const cartPromise = loadScript(config['cart-js'], 'cart')
    .then((exports) => {
      Object.assign(externals, exports);
      store.addReducer('cart', exports.cartReducers);
    });
  /* ... */
  return Promise.all([cartPromise /* ... */]);
}

export function initExternals() {
  return loadFrontEndConfig().then(loadFrontEnds);
}

export default externals;
```

COUPLED
route to the component

```
<Provider store={store.instance()} >
  <HashRouter>
    <Switch>
      <Route path="/cart" component={externals.cartServiceRoute} />
```

COUPLED
displaying the component

```
<div className={styles.tile} >
  <externals.CartPreviewTile />
</div>
```

```
/* Components.js */

export {
  CartRoute, cartReducers,
  CartPreviewTile,
};
```

```
module.exports = {
  output: {
    path: paths.appLib,
    filename: "app.js",
    library: "cart",
    libraryTarget: "umd",
  },
```

DEFINITIONS
describing the contract

=> All coupling is one-off and components can evolve independently

ThoughtWorks®

ERIK DOERNENBURG
HEAD OF TECHNOLOGY
erik@thoughtworks.com

web approach | server-side composition | client-side composition | client-side rendering | SPA/PWA composition