Imports

Cory House
bitnative.com
Twitter: @housecor

pluralsight
hardcore developer training

HTML5 Imports Agenda

The Problem

The Solution

The Pitfalls

How Do I Import HTML?

Import CSS

```
<link rel="stylesheet" href="style.css" />
```

Import JS

```
<script src="jquery.min.js"></script>
```

Import HTML

Uh, let the hackery begin…

AJAX calls…
Iframes…
Script tags…
Hidden divs…

Imports vs Templates

Both are a way to store inert HTML

Only imports support:

1. Storing inert HTML in a separate file

2. Bundling

Packaging Matters

We Need Bundling

What if you want to bundle some CSS, JS, and HTML together?

Consider Bootstrap…

```
<script src="jquery.min.js"></script>
<script src="bootstrap.min.js"></script>
<link rel="stylesheet" href="bootstrap.min.css" />
<script src="script.js"></script>
<link rel="stylesheet" href="style.css" />
```
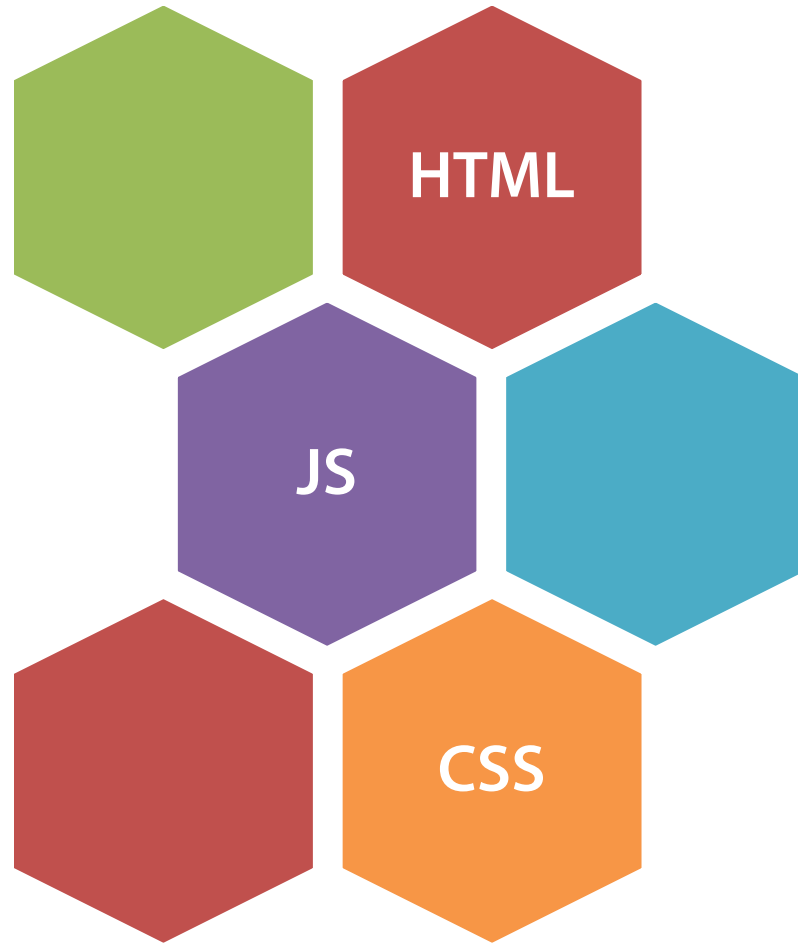
What if we could simply say:

```
<import rel="import" href="myBootstrapComponent.html" />
```

Bundle Up

HTML

JS

CSS

Import Example

```
<link rel="import" href="stuffToImport.html">

<link rel="import" href="http://othersite.com/import.html">
```

A Few Rules:
1. The value of the href is called the import location
2. Link must sit in <head>
3. HTML is initially inert
4. JavaScript and CSS run and apply immediately
5. Imports from separate domains must support CORS

Programmatic Access to Import Content

```javascript
var content = document.querySelector('#my-import').import;
```

Imports

Demo

Other Uses for Imports

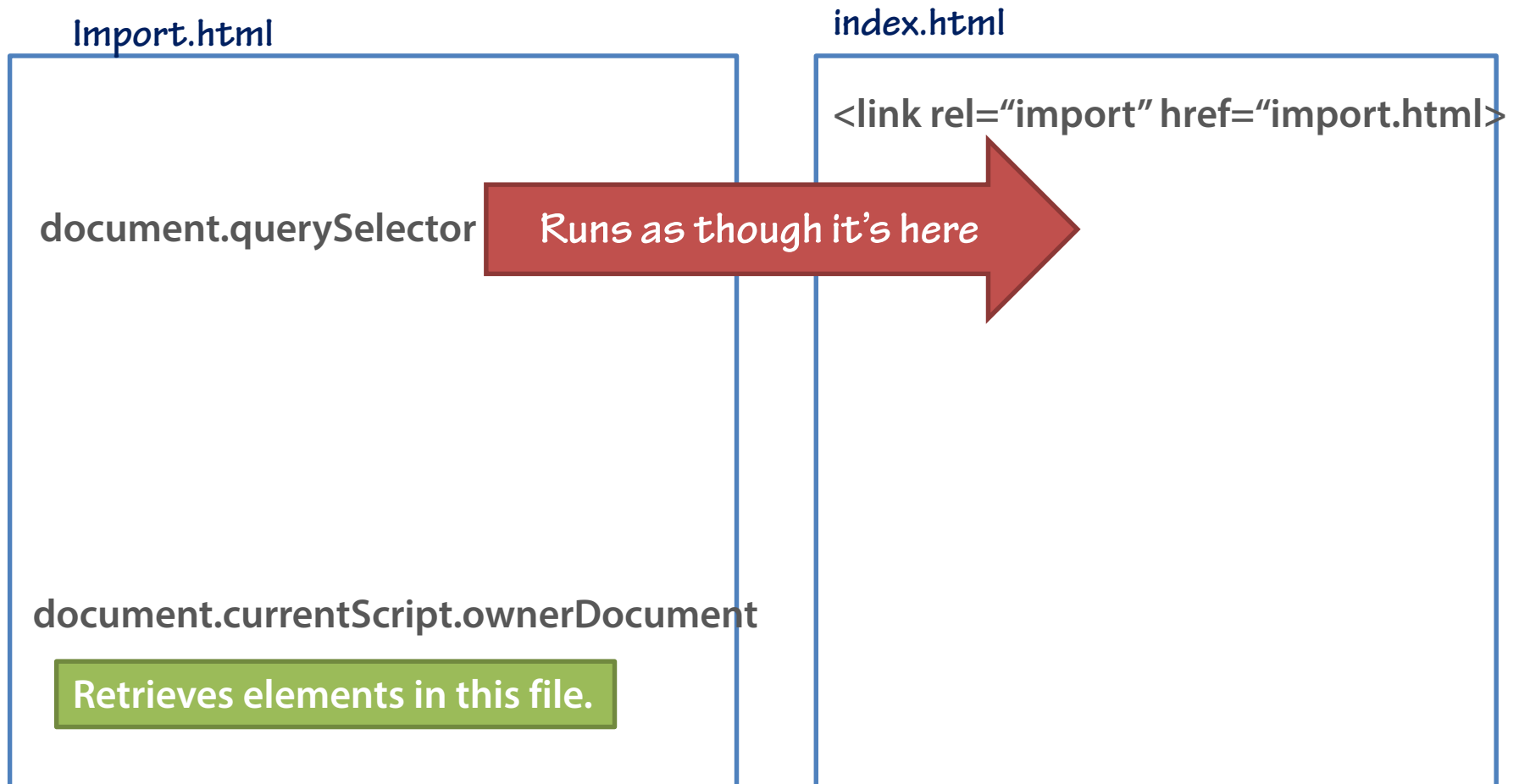| Themes	Component Library	App Sections	An entire app!
• Even change markup and behavior within themes	• Bundle related components together	• Bundle resources used in specific portions of your app	• Just an import and a single custom element…

Reference the Owner Document

document.currentScript.ownerDocument

Import.html

index.html

<link rel="import" href="import.html>

document.querySelector

Runs as though it's here

document.currentScript.ownerDocument

Retrieves elements in this file.

Referencing the Owner Document

Demo

When Does Content *in Imports* Apply?

	Outside Template	Inside Template
HTML	Inert until copied	Inert until copied
CSS	Applies immediately	Inert until copied
JavaScript	Applies immediately	Inert until copied

Imports Block Rendering by Default

```
<link rel="import" href="import.html">
```
← This blocks

```
<link rel="stylesheet" href="style.css">
```
← Just like this

```
<link rel="import" href="import.html" async>
```

Place <script> at the end of the main document to avoid blocking.

When is content in imports applied?

Demo

Handling Events

Two Events:

1. onload
2. onerror

```html
<script>
    function loaded(event) {
            console.log('Import Loaded');
    }

    function error(event) {
            console.log('Uh oh, couldn\'t load ' + event.target.href);
    }
</script>

<link rel="import" href="import.html" onload="loaded(event)"
        onerror="error(event)">
```

Handling Events on Imports

Demo

copies 6 of jQuery?!

Duplicate Requests

The browser handles identical requests automatically.
Differences in protocol, version, or domain? Duplicate.

Conflict Resolution

Web-Component-A references jQuery **1.10**

Web-Component-B references jQuery **1.11**

Both will be downloaded, and potentially conflict.

Avoiding Duplicate/Conflicting Resources

Eliminate dependencies

Don't bundle dependencies

Use centralized resource

Utilize a package manager

Run the latest version of third party libraries

Sub-Imports

Pluralsight-tabs.html

```
<link rel="import" href="jquery.html">
file continues…
```

Pluralsight-accordian.html

```
<link rel="import" href="jquery.html">
file continues…
```

Pluralsight-search.html

```
<link rel="import" href="jquery.html">
file continues…
```

Polymer's FAQ Suggests…

"If multiple libraries want to share a dependency, they will have to agree on a system. Feature detection, or an agreed upon common location for a 'jquery.html' file in a CDN, etc."

Sub-Imports

Demo

Summary

Store inert HTML

Imports bundle up and deliver your components

Programmatically access import contents

Avoid third party dependencies

Consider utilizing Bower

Agree on a standard

Congratulations!