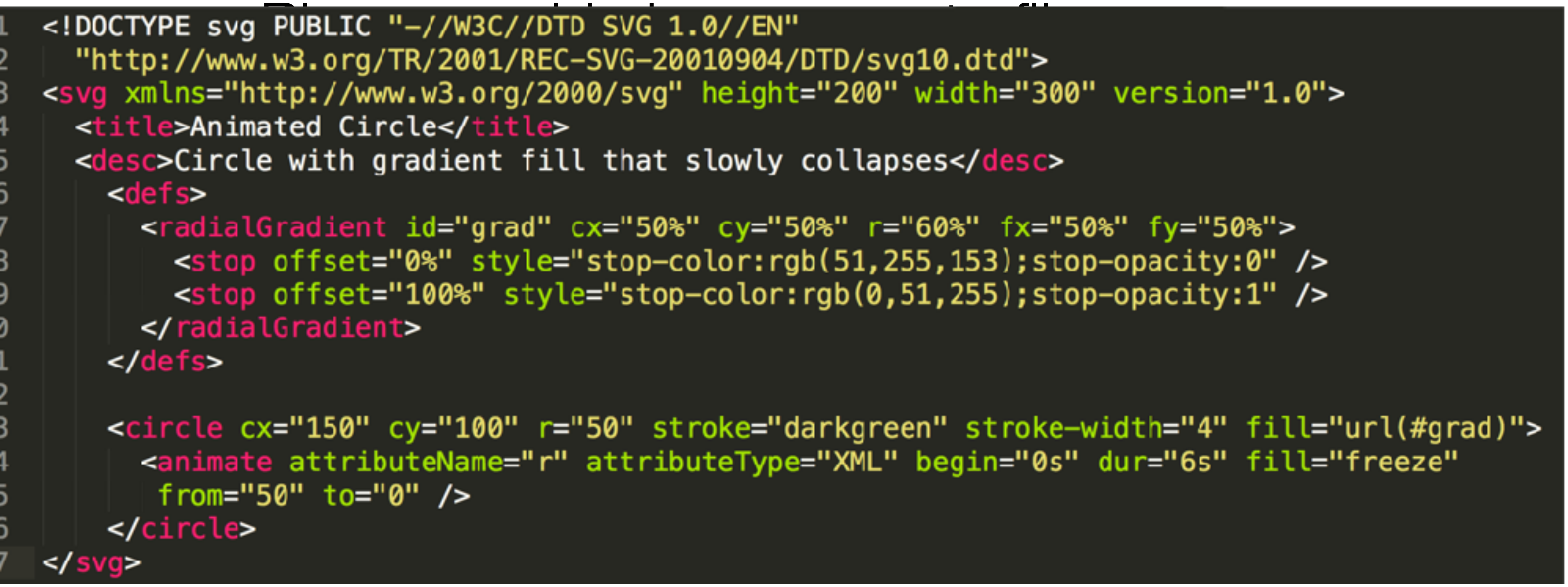
HTML5 and CSS3 for Mobile Applications

Prof. Paul Krause, University of Surrey
What did HTML5 ever do for us?

Why is HTML5 a game changer?

- Pros for HTML5 Web Apps
  - Easier to build and iterate
  - Use existing skills in HTML and CSS
  - Same technology across all devices
  - All mobile browsers support HTML5 Audio and Video
    - Control via the DOM, style via CSS
  - Geolocation API
  - Offline content and storage

Scalable Vector Graphics



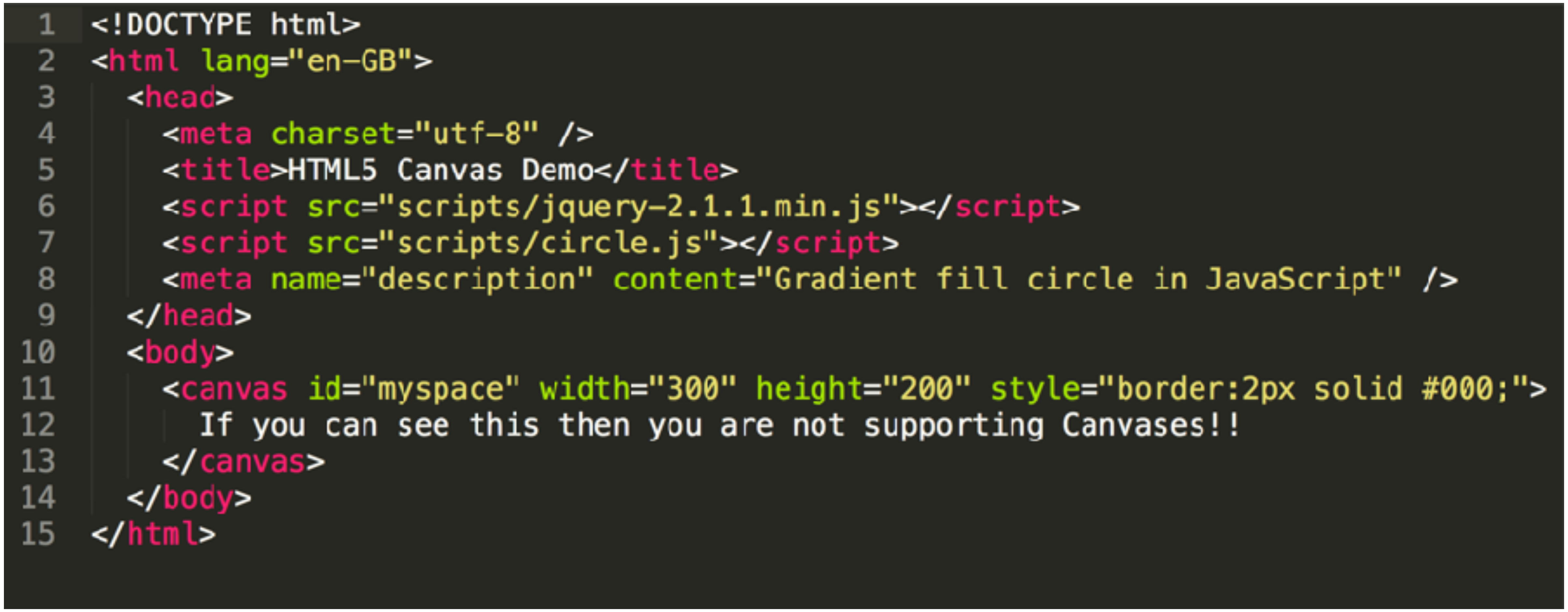
```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg" height="200" width="300" version="1.0">
  <title>Animated Circle</title>
  <desc>Circle with gradient fill that slowly collapses</desc>
    <defs>
      <radialGradient id="grad" cx="50%" cy="50%" r="60%" fx="50%" fy="50%">
        <stop offset="0%" style="stop-color:rgb(51,255,153);stop-opacity:0" />
        <stop offset="100%" style="stop-color:rgb(0,51,255);stop-opacity:1" />
      </radialGradient>
    </defs>

    <circle cx="150" cy="100" r="50" stroke="darkgreen" stroke-width="4" fill="url(#grad)">
      <animate attributeName="r" attributeType="XML" begin="0s" dur="6s" fill="freeze"
       from="50" to="0" />
    </circle>
</svg>
```

Advantages of SVG

- Small file size
- Can be changed with scripting
- Scales without pixelated or jagged edges
- Easy to understand
- Can be animated

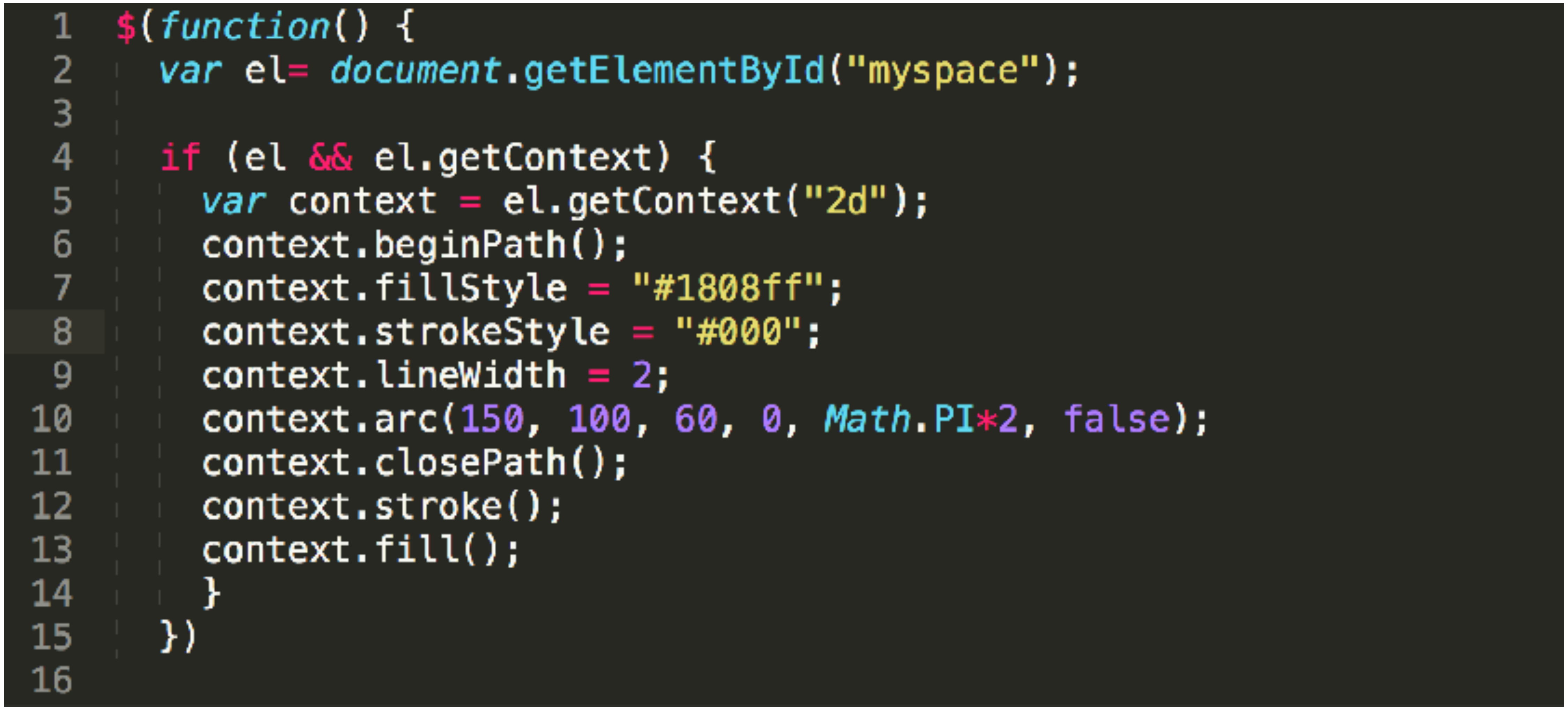
Canvases

Importing the JavaScript

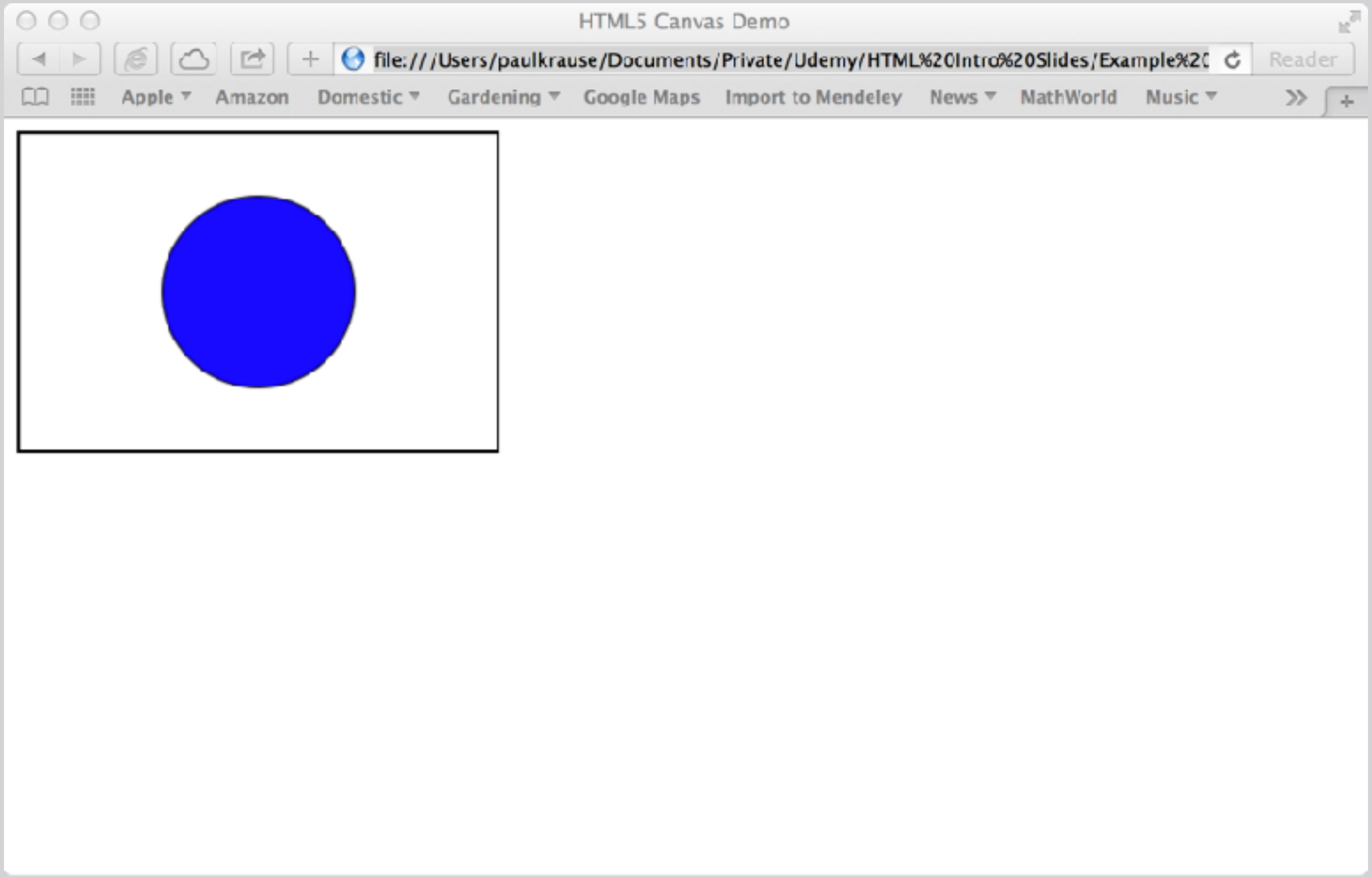


```
<!DOCTYPE html>
<html lang="en-GB">
  <head>
    <meta charset="utf-8" />
    <title>HTML5 Canvas Demo</title>
    <script src="scripts/jquery-2.1.1.min.js"></script>
    <script src="scripts/circle.js"></script>
    <meta name="description" content="Gradient fill circle in JavaScript" />
  </head>
  <body>
    <canvas id="myspace" width="300" height="200" style="border:2px solid #000;">
      If you can see this then you are not supporting Canvases!!
    </canvas>
  </body>
</html>
```

Add the fill



```javascript
$(function() {
  var el= document.getElementById("myspace");

  if (el && el.getContext) {
    var context = el.getContext("2d");
    context.beginPath();
    context.fillStyle = "#1808ff";
    context.strokeStyle = "#000";
    context.lineWidth = 2;
    context.arc(150, 100, 60, 0, Math.PI*2, false);
    context.closePath();
    context.stroke();
    context.fill();
    }
  })

```


HTML5 Canvas Demo
file:///Users/paulkrause/Documents/Private/Udemy/HTML%20Intro%20Slides/Example%2(
Reader
Apple
Amazon
Domestic
Gardening
Google Maps
Import to Mendeley
News
MathWorld
Music

Compare and contrast

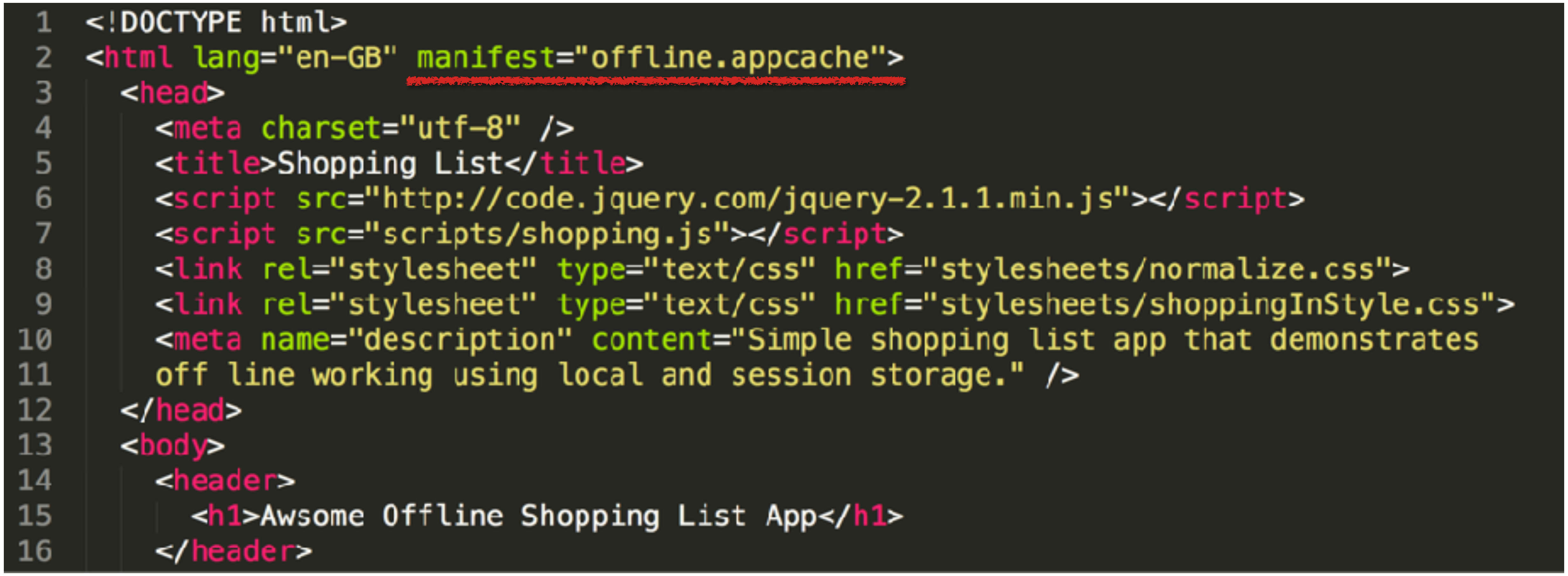
- SVG you draw in XML; with Canvas, you draw in JavaScript;
- SVG creates DOM nodes that can be manipulated, and is resolution independent;
- Canvas draws in pixels so pixilates on zooming;
- No animation API with Canvases
- But SVG can perform badly - especially an issue on mobile devices

Application cache

AppCache API

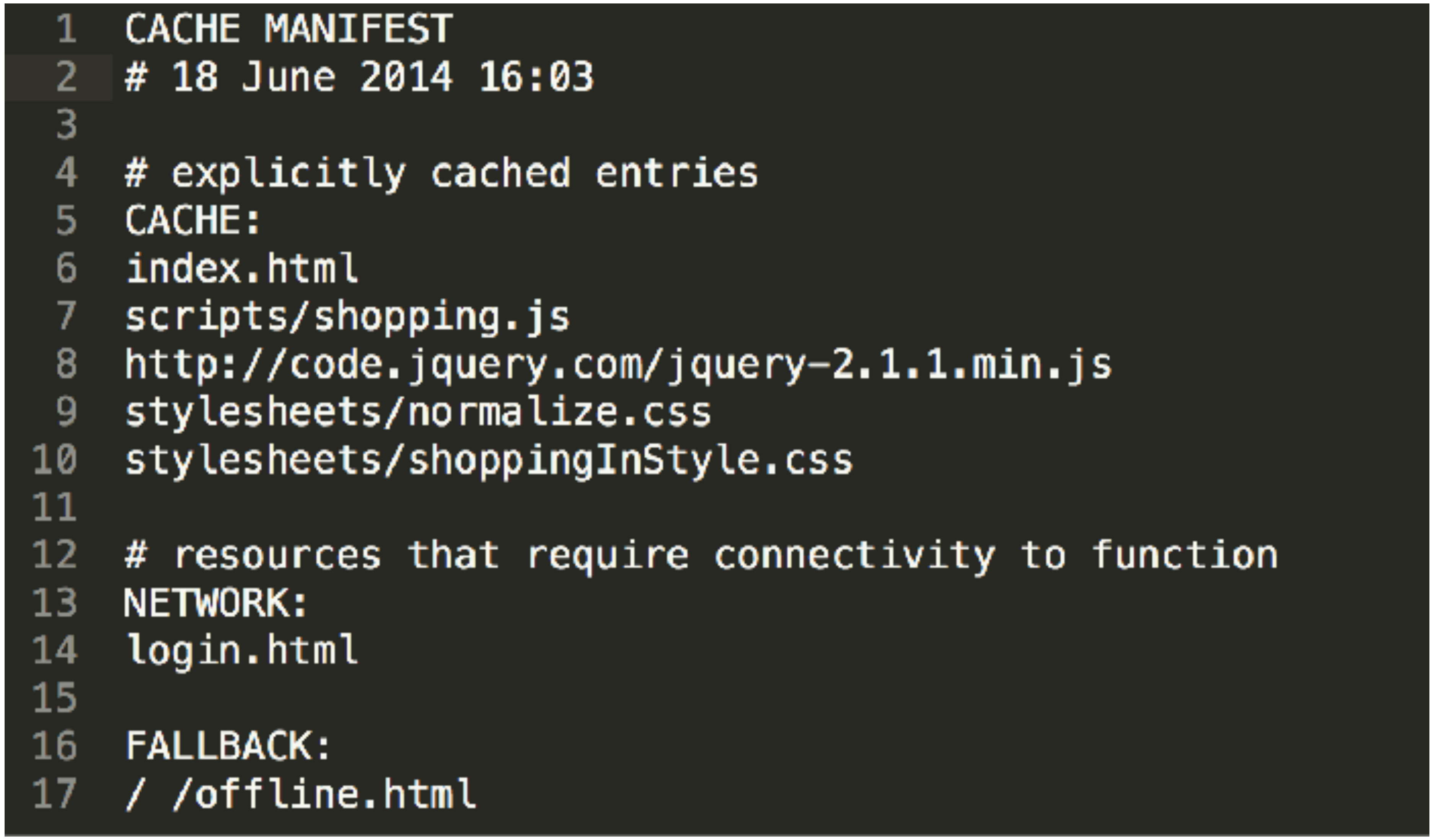
- specify which files should be cached
- Supports:
  - offline browsing/working/playing
  - faster reloads
  - reduced server load
- Max of 5MB local storage on most mobile browsers

Just add a manifest file and you’re ready to go…



```html
<!DOCTYPE html>
<html lang="en-GB" manifest="offline.appcache">
  <head>
    <meta charset="utf-8" />
    <title>Shopping List</title>
    <script src="http://code.jquery.com/jquery-2.1.1.min.js"></script>
    <script src="scripts/shopping.js"></script>
    <link rel="stylesheet" type="text/css" href="stylesheets/normalize.css">
    <link rel="stylesheet" type="text/css" href="stylesheets/shoppingInStyle.css">
    <meta name="description" content="Simple shopping list app that demonstrates
    off line working using local and session storage." />
  </head>
  <body>
    <header>
      <h1>Awsome Offline Shopping List App</h1>
    </header>
```

<filename>.appcache



```
CACHE MANIFEST
# 18 June 2014 16:03

# explicitly cached entries
CACHE:
index.html
scripts/shopping.js
http://code.jquery.com/jquery-2.1.1.min.js
stylesheets/normalize.css
stylesheets/shoppingInStyle.css

# resources that require connectivity to function
NETWORK:
login.html

FALLBACK:
/ /offline.html
```

Session storage

Session storage

- Session storage data is:
  - confined to the browser window it was created in
  - accessible to any page from the same origin within that window
  - is deleted when the session ends
  - stored as key:value pairs

LocalStorage

Local Storage

- Local Storage data is:
  - accessible across all windows in the same browser
  - is accessible to any page from the same origin within the browser
  - persists after the browser (window) is closed
  - stored as key:value pairs

Promises

What is a promise?

- "A promise is the eventual result of an asynchronous operation"

Promises/A+ specification

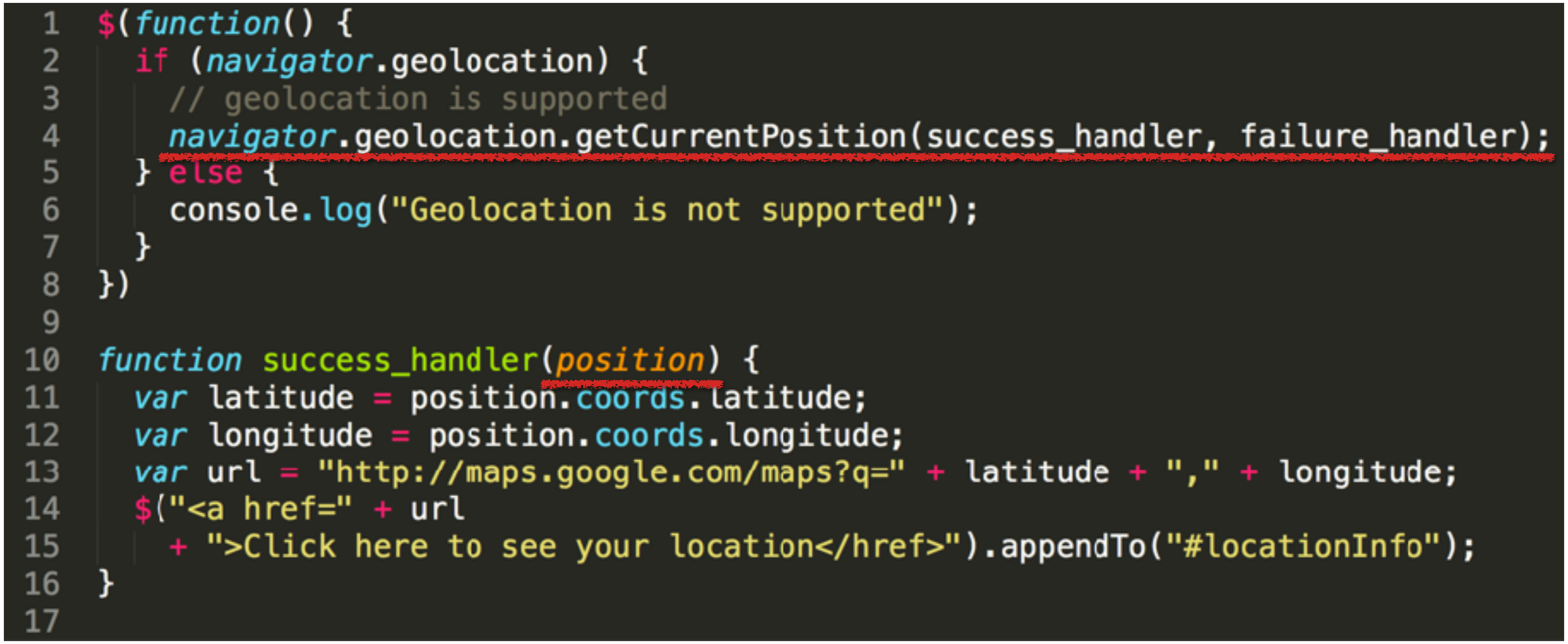
- A *promise* must have a *then* method
- The *then* method registers callbacks to (eventually):
  - Receive the <u>value</u> of the *promise*, or
  - Receive the <u>reason</u> why the promise cannot be fulfilled

Geolocation

Geolocation

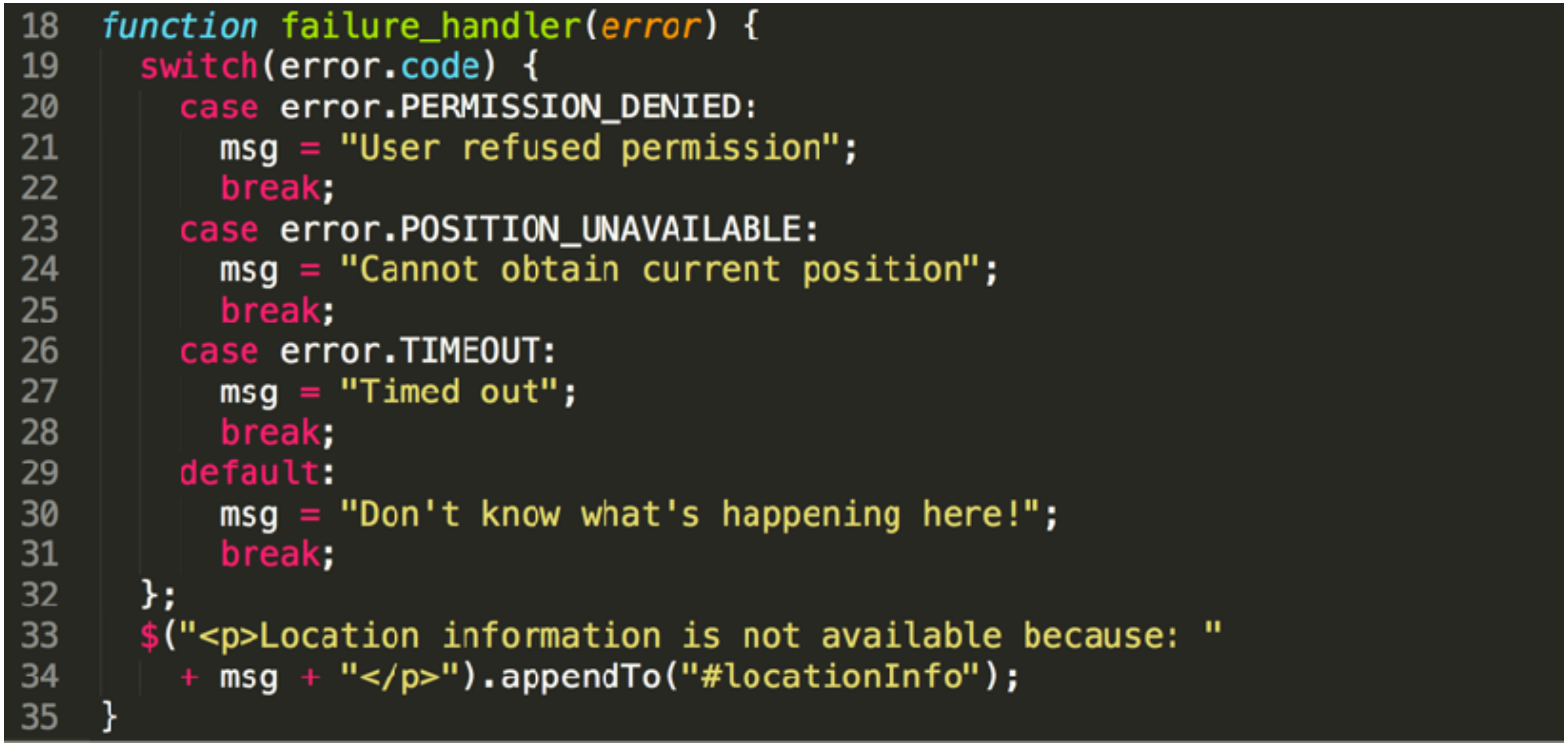
- An opt in feature
- Enables location to be determined and tracked
- Uses a range of tools to provide an “approximate” location:
  - IP addresses; cell towers; GPS; WiFi networks
- The geolocation API is asynchronous

Handling the success callback



```
$(function() {
  if (navigator.geolocation) {
    // geolocation is supported
    navigator.geolocation.getCurrentPosition(success_handler, failure_handler);
  } else {
    console.log("Geolocation is not supported");
  }
})

function success_handler(position) {
  var latitude = position.coords.latitude;
  var longitude = position.coords.longitude;
  var url = "http://maps.google.com/maps?q=" + latitude + "," + longitude;
  $("<a href=" + url
    + ">Click here to see your location</href>").appendTo("#locationInfo");
}

```

Handling the failure callback



```
function failure_handler(error) {
  switch(error.code) {
    case error.PERMISSION_DENIED:
      msg = "User refused permission";
      break;
    case error.POSITION_UNAVAILABLE:
      msg = "Cannot obtain current position";
      break;
    case error.TIMEOUT:
      msg = "Timed out";
      break;
    default:
      msg = "Don't know what's happening here!";
      break;
  };
  $("<p>Location information is not available because: "
    + msg + "</p>").appendTo("#locationInfo");
}
```

JavaScript

Executable content in Web Pages

- Control document content and appearance
- Control the browser
- Control the content of forms that appear in a browser
- Supports two-way interaction through a Web Page
    - A basis of “Rich Client Interfaces” for Web Applications

Event Handling

- Enables a *JavaScript* program to respond in some way when you:
  - click on a button
  - mouse over an item
  - enter a value in an input field
  - click on a image
  - ...

Summary of JavaScript object

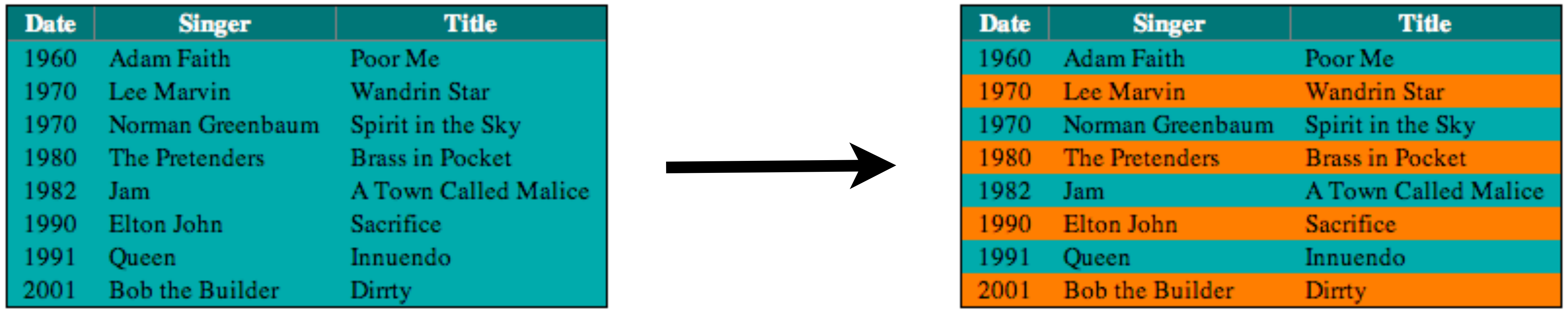
- A JavaScript object is an unordered collection of properties
- Properties consist of a name and a value
- Objects can be declared using object literals
- Top level *variables* are properties of *window*

Functions are also objects in JavaScript

- They are constructed with the JavaScript constructor: *function*
- JavaScript functions can be:
  - Assigned to variables;
  - Assigned as a property of an object;
  - Passed as a parameter
  - Returned as a function result;
  - Created using literals.

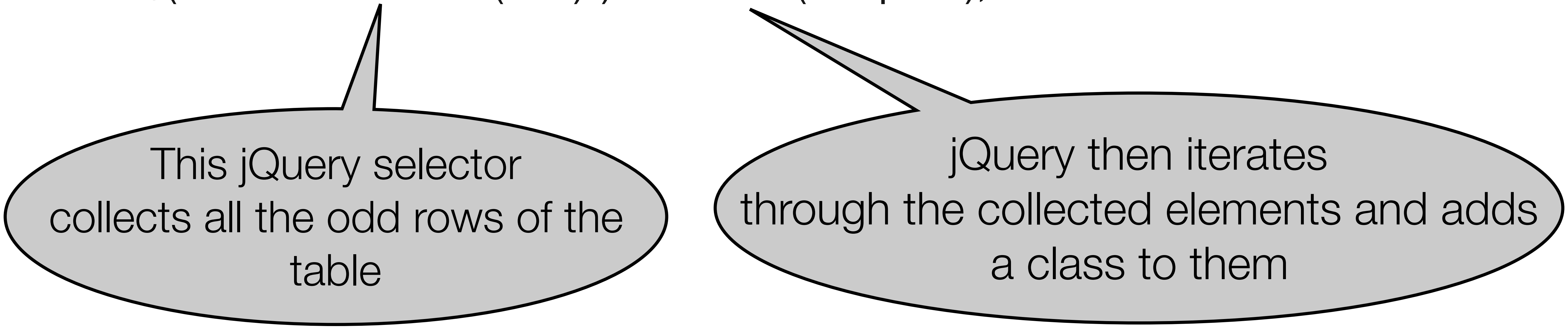
jQuery

Using jQuery to add dynamic functionality



Date	Singer	Title
1960	Adam Faith	Poor Me
1970	Lee Marvin	Wandrin Star
1970	Norman Greenbaum	Spirit in the Sky
1980	The Pretenders	Brass in Pocket
1982	Jam	A Town Called Malice
1990	Elton John	Sacrifice
1991	Queen	Innuendo
2001	Bob the Builder	Dirrty

- Dynamically updating the appearance of an html document is simple:
  - $("table tr:nth-child(odd)").addClass("striped");


This jQuery selector collects all the odd rows of the table
jQuery then iterates through the collected elements and adds a class to them

Look at our example again

```
<html>
    <head>
        <meta charset="UTF-8">
        <title>Some jQuery Examples</title>
        <script src="jquery-1.8.2.min.js"></script>
        <script>
            $(function(){
                $('#myTitle')
                .bind('mouseover', function(event){
                    $("<p class='bear'>I have a bear!</p><p>I don't</p>")
                    .filter(".bear").click(function(){
                        alert("I'm a bear!");
                    }).end().appendTo("#myParentDiv");
                })
            })
        </script>
    </head>
    <body>
        <h1 id="myTitle">Some jQuery examples</h1>
        <p id="myParentDiv">The fun starts here:</p>
        <p>And ends here!</p>
    </body>
</html>
```

Define the document ready handler

```
<script>
   $(function(){
      $('#myTitle')
      .bind('mouseover', function(event){
         $("<p class='bear'>I have a bear!</p><p>I don't</p>")
         .filter(".bear").click(function(){
             alert("I'm a bear!");
         }).end().appendTo("#myParentDiv");
      })
   })
</script>
```

When the document is ready bind a mouseover event handler to myTitle

```
<script>
   $(function(){
      $('#myTitle')
      .bind('mouseover', function(event){
         $("<p class='bear'>I have a bear!</p><p>I don't</p>")
         .filter(".bear").click(function(){
             alert("I'm a bear!");
         }).end().appendTo("#myParentDiv");
      })
   })
</script>
```

On “mouseover” create two new paragraphs One has class ‘bear’

```
<script>
   $(function(){
      $('#myTitle')
      .bind('mouseover', function(event){
         $("<p class='bear'>I have a bear!</p><p>I don't</p>")
         .filter(".bear").click(function(){
             alert("I'm a bear!");
         }).end().appendTo("#myParentDiv");
      })
   })
</script>
```

Filter out from that element set only those with class ‘bear’

```
<script>
   $(function(){
      $('#myTitle')
      .bind('mouseover', function(event){
         $("<p class='bear'>I have a bear!</p><p>I don't</p>")
         .filter(".bear").click(function(){
             alert("I'm a bear!");
         }).end().appendTo("#myParentDiv");
      })
   })
</script>
```

Bind a “click” event handler to the new elements of class ‘bear’

```
<script>
   $(function(){
      $('#myTitle')
      .bind('mouseover', function(event){
         $("<p class='bear'>I have a bear!</p><p>I don't</p>")
         .filter(".bear").click(function(){
             alert("I'm a bear!");
         }).end().appendTo("#myParentDiv");
      })
   })
</script>
```

Revert back to the full set of new elements

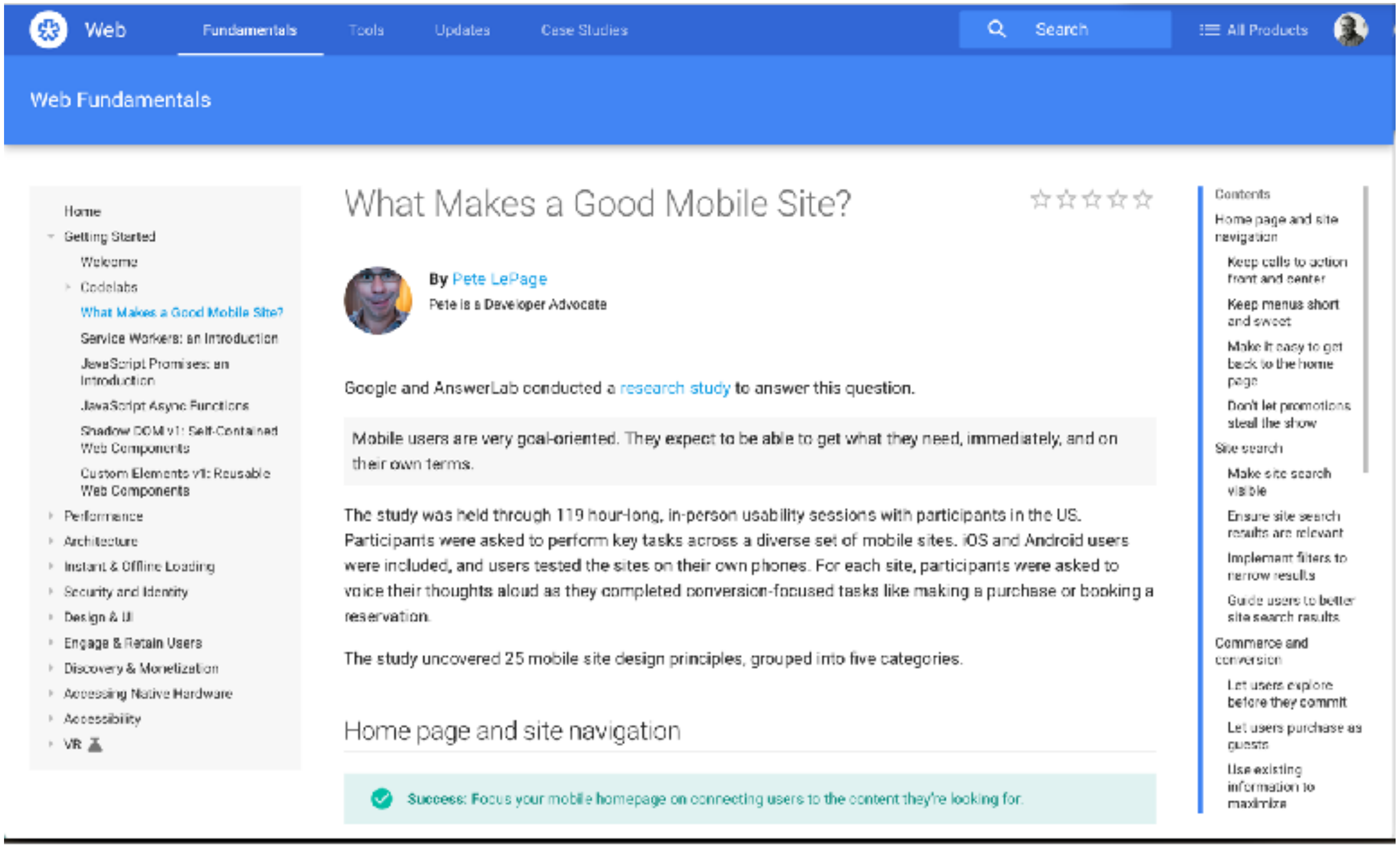
```
<script>
   $(function(){
      $('#myTitle')
      .bind('mouseover', function(event){
         $("<p class='bear'>I have a bear!</p><p>I don't</p>")
         .filter(".bear").click(function(){
             alert("I'm a bear!");
         }).end().appendTo("#myParentDiv");
      })
   })
</script>
```

And append them to myParentDiv

```
<script>
   $(function(){
      $('#myTitle')
      .bind('mouseover', function(event){
         $("<p class='bear'>I have a bear!</p><p>I don't</p>")
         .filter(".bear").click(function(){
             alert("I'm a bear!");
         }).end().appendTo("#myParentDiv");
      })
   })
</script>
```

Mobile First

developers.google.com



Web | Fundamentals | Tools | Updates | Case Studies | Search | All Products

Web Fundamentals

- Home
- Getting Started
  - Welcome
  - Codelabs
  - What Makes a Good Mobile Site?
  - Service Workers: an Introduction
  - JavaScript Promises: an Introduction
  - JavaScript Async Functions
  - Shadow DOM v1: Self-Contained Web Components
  - Custom Elements v1: Reusable Web Components
- Performance
- Architecture
- Instant & Offline Loading
- Security and Identity
- Design & UI
- Engage & Retain Users
- Discovery & Monetization
- Accessing Native Hardware
- Accessibility
- VR

What Makes a Good Mobile Site?

By Pete LePage

Pete is a Developer Advocate

Google and AnswerLab conducted a research study to answer this question.

> Mobile users are very goal-oriented. They expect to be able to get what they need, immediately, and on their own terms.

The study was held through 119 hour-long, in-person usability sessions with participants in the US. Participants were asked to perform key tasks across a diverse set of mobile sites. iOS and Android users were included, and users tested the sites on their own phones. For each site, participants were asked to voice their thoughts aloud as they completed conversion-focused tasks like making a purchase or booking a reservation.

The study uncovered 25 mobile site design principles, grouped into five categories.

Home page and site navigation

Success: Focus your mobile homepage on connecting users to the content they're looking for.

Contents

- Home page and site navigation
  - Keep calls to action front and center
  - Keep menus short and sweet
  - Make it easy to get back to the home page
  - Don't let promotions steal the show
- Site search
  - Make site search visible
  - Ensure site search results are relevant
  - Implement filters to narrow results
  - Guide users to better site search results
- Commerce and conversion
  - Let users explore before they commit
  - Let users purchase as guests
  - Use existing information to maximize

What do we want?

Mobile users are very goal-oriented. They expect to be able to get what they need, immediately, and on their own terms.

Googlebot Mobile Friendly Test

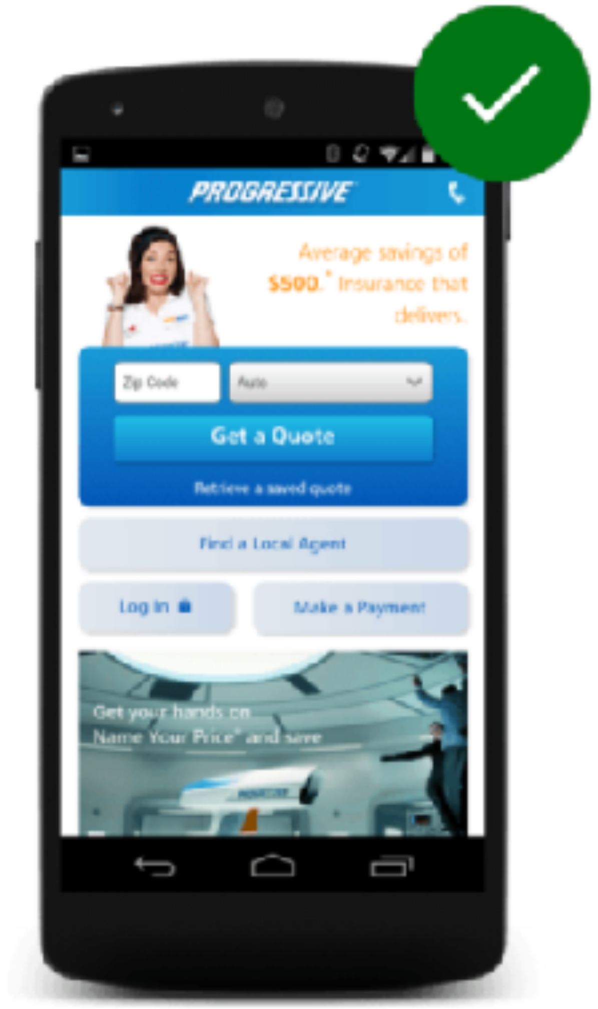
https://search.google.com/search-console/mobile-friendly

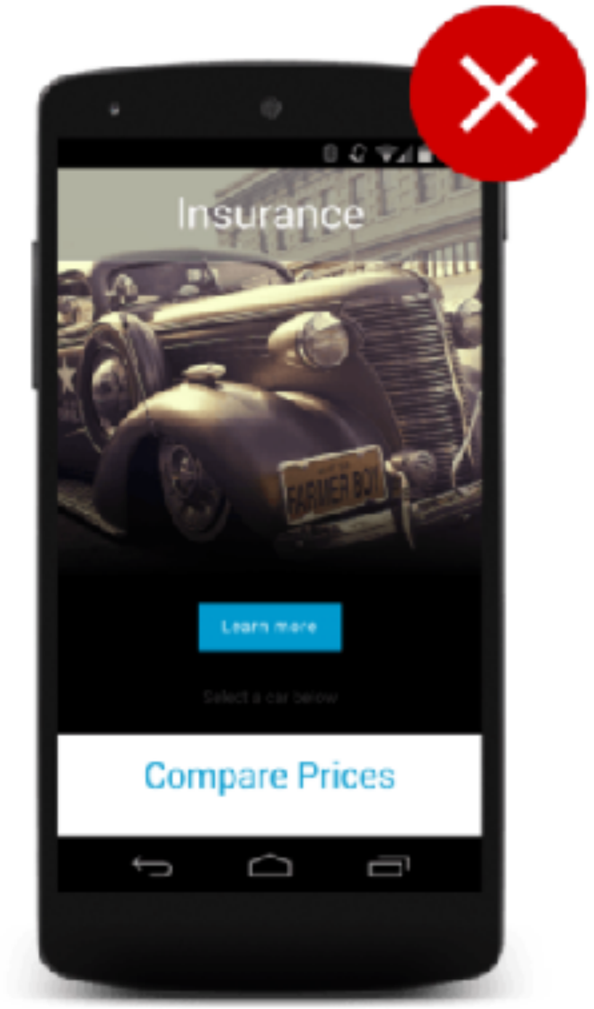
Checks to make sure your site:

- Avoids software that is not common on mobile devices, like Flash
- Uses text that is readable without zooming
- Sizes content to the screen so users don’t have to scroll horizontally or zoom
- Places links far enough apart so that the correct one can be easily tapped

1.1 Home Page and Site Navigation

- Keep Calls to Action Front and Centre





Minimising the number of navigation options on each screen helps users focus on what they need to do and helps to prevent errors such as accidentally tapping through to another task.

Work Hard & Good Luck!