Single-Page Application

- Now Web apps are becoming almost like a desktop app, except that it is downloaded from the Web
- Single-page Application (SPA)
  - An app where everything happens on a single page
  - No page reload and wait
    - Even when the browser needs to obtain data from server

Scripted HTTP

- Google suggest example interaction: http://oak.cs.ucla.edu/cs144/examples/google-suggest.html
  - Q: What is going on behind the scene? What events does it monitor? What does it do when the event is detected?
  - Q: When the "typing" event is detected, what does it have to do? How can it let users keep type while waiting for data from server?

XMLHttpRequest

`XMLHttpRequest`: JavaScript object for *asynchronous* communication with the server

- Sending a request to server

```
xmlHttp = new XMLHttpRequest();
xmlHttp.open("GET", URL); // method, url
xmlHttp.send(null);       // body of the request
```

  - `open(method, URL)` produces request line
  - `setRequestHeader(header-name, value)` adds request header
  - `send(body)` produces request body
  - Actual connection to the server is made only when `send()` is called
- Handling response:
  - Set `onreadystatechange` to our own event handler

```
xmlHttp.onreadystatechange = handlerfunction;
```

– `onreadystatechange` is called whenever a change occurs in the state of the request
  * The current state of the request is available at `readyState` property
    - 0: `UNSENT`. `open()` has not been called
    - 1: `OPENED`. `open()` has been called
    - 2: `HEADERS_RECEIVED`. Headers have been received
    - 3: `LOADING`. The response body is being received
    - 4: `DONE`. The response is complete

– Response from the server is available as `responseText`/`responseXML` properties
  * `responseText` is text, `responseXML` is XML DOM

– Server response in JSON
  * The server response used to be mainly in XML, but JSON has gained popularity
  * `response = JSON.parse(xmlHttp.responseText);`
  * Once parsed, `response` work just like a regular JavaScript object

XML (Extensible Markup Language)

- HTML was hugely successful due to
  – Simplicity -> can be learned easily
  – Text based -> can be edited by any text editor. No need for a special tool
- But, HTML is mainly for human consumption
  – HTML tags are for document structure, not for semantic meaning
  – e.g., <`table`>, <`ul`>, etc.
- XML: data representation standard with "semantic" tag
  – Show example in first XML slide

```
<?xml version="1.0"?>
<Book edition="1">
    <Title>Database systems</Title>
    <Author>Hector Garcia-Molina</Author>
    <ISBN>135-383-9038</ISBN>
    <Price>$100</Price>
</Book>
```

- XML consists of three things:
  1. Tagged elements, which may be nested within one another
  2. Attributes on elements
  3. Text
- Well-formed XML should have
  - Single root element
  - Matching tags
  - Unique attribute name

XML Namespaces

- A way to avoid "name conflict"
- XML namespace allows specifying where the tags "belong"
  - Very similar to C++ namespace and Java package specification

  e.g.) Show how we can add a "default namespace" to the first example (slide 2)

```
<?xml version="1.0"?>
<Book edition="1" xmlns="http://oak.cs.ucla.edu/cs144">
    <Title>Database systems</Title>
    <Author>Hector Garcia-Molina</Author>
    <ISBN>135-383-9038</ISBN>
    <Price>$100</Price>
</Book>
```

  - Note: The namespace URL does not have to point to any real page.
    * The URL is just the unique identifier of the namespace.
  - Q: What namespace does element `Title` belong to?
- Q: Is it possible to use different namespace for different elements?

Book, Title, Author, ISBN, Price: http://oak.cs.ucla.edu/cs144
Price: http://xml.com/shopping

```
<?xml version="1.0"?>
<Book edition="1" xmlns="http://oak.cs.ucla.edu/cs144"
                  xmlns:s="http://xml.com/shopping">
    <Title>Database systems</Title>
    <Author>Hector Garcia-Molina</Author>
    <ISBN>135-383-9038</ISBN>
    <s:Price>$100</s:Price>
</Book>
```

- Q: Do E1 and E2 belong to the same namespace?

```
<a:E1 xmlns:a="http://a.com/">
<b:E2 xmlns:b="http://a.com/">
```

XMLHttpRequest Example

```
<!DOCTYPE html>
<html>
<head>
</head>
<body onload="init();">
    <b>Your query:</b> <input type="text" id="queryBox"><br>
    <b>Suggestion</b>: <div id="suggestion"></div>
</body>
<script type="text/javascript">
let xmlHttp;

// initialization
function init() {
    xmlHttp = new XMLHttpRequest();
    document.getElementById("queryBox").oninput = sendAjaxRequest;
}

// send Google suggest request based on the user input
```

```
function sendAjaxRequest(event)
{
    let request = "google-suggest.php?q="+encodeURI(this.value);

    xmlHttp.open("GET", request);
    xmlHttp.onreadystatechange = showSuggestion;
    xmlHttp.send(null);
}

// update the page with the response
function showSuggestion() {
    // if the request is complete, display the response from the
       server
    if (xmlHttp.readyState == 4) {
        htmlCode = xmlHttp.responseText.replace(/</g,'&lt;').
           replace(/>/g,'&gt;');
        document.getElementById("suggestion").innerHTML = htmlCode;
    }
}
</script>
</html>
```

- **Q:** What events does the code monitor?
- **Q:** What does it do when the event is detected?
- **Q:** When it receives the response from the server, what does it do?
- **Q:** What URL does it send the request to? Why is not the Google server?

Same-origin policy

- `XMLHttpRequest` can send a request *only to the same host* of the page
  - Due to this policy, a third-party site cannot be contacted through `XMLHttpRequest`
  - Run a "proxy" on the same host, which takes a request and forwards it to the third-party Web site
  - Cross-Origin Resource Sharing (CORS) and JSONP have been developed to get around this restriction

- Cross-Origin Resource Sharing (CORS)
  - The browser can inquire server-approved cross-request domains through `Origin:` header
  - The server replies the list of allowed domains with `Access-Control-Allow-Origin:` header
  - Example:

```
In request to server
Origin: http://oak.cs.ucla.edu

In response from the server
Access-Control-Allow-Origin: http://www.google.com
```

  - This allows Javascript code running on a page from oak.cs.ucla.edu to issue a cross-site request to www.google.com
  - CORS is automatically taken care of by modern browsers, so there is nothing JavaScript programmer has to do, as long as the servers are configured to allow CORS

JSONP (JSON with Padding)

- A "hack" to get around same-origin policy restriction
- Using JavaScript, set `src` to the URL to which a request should be sent
  - Same origin policy is not applied to `src` in `<script src="url">`!
- The response is considered as a JavaScript by the browser and gets executed
  - If the response is in JSON, a JavaScript object is created!
- To be able to use the object inside our code, the JSON response should be "wrapped" with a function call like `myFunc({"x": 10, "y": 20});`
  - `myFunc` is called with the object as the parameter.
- JSONP requires support from the third-party Web site
  - The callback function name is often provided as part of the request

Animation Effects

- Q: How can we create dynamic animation on a Web page?
  - e.g., scrolling news tickers, flying boxes, . . .

- Two approaches
  - Use JavaScript!
  - CSS animation

JavaScript Animation

- Key functions and properties for animation
  - `setInterval("event_handler", interval)`: time-based event generator
  - `element.style`: allows modifying CSS style properties
    - e.g., `div.style.left`, `div.style.right`, `div.style.top`, ...
- Example: http://oak.cs.ucla.edu/classes/cs144/examples/ticker.html
  - Show the page and animation and then code.

```
<!DOCTYPE html>
<html>
<head><title>Ticker</title></head>
<body onload="tickerStart();">
<div id="ticker" style="position: absolute; left: 0px;">Hello,
    there...</div>
</body>
<script type="text/javascript">
let ticker;
let loc = 0;
let timer;

function tickerStart() {
  ticker = document.getElementById("ticker");
  timer = setInterval(tickerSlide, 100);
}

function tickerSlide() {
  loc += 10;
  ticker.style.left = String(loc) + "px";

  // stop sliding after 300px
  if (loc > 300) clearInterval(timer);
}
</script>
</html>
```

  - Q: Why does the text move? What sequence of function calls?
  - Q: Why does it stop moving?
  - Q: What if we set ticker variable when we declare it first? Is it necessary to set the variable inside startTicker?

- Example: http://oak.cs.ucla.edu/classes/cs144/examples/box.html

  Q: What will the following page do?

```
<!DOCTYPE html>
<html>
<head><title>Box</title></head>
<body onload="start();">
    <div id="box" style="width: 200px; height: 200px; border:
       solid 5px black;"></div>
</body>
<script type="text/javascript">
let box;
let size = 200;

function start() {
    box = document.getElementById("box");
    setInterval(shrinkBox, 100);
}

function shrinkBox() {
    size -= 10;
    if (size < 0) size = 200;
    box.style.width = String(size) + "px";
    box.style.height = String(size) + "px";
}
</script>
</html>
```

CSS Animation

- `transition` property
  - CSS transition creates a "transition effect" whenever an element's property changes
  - e.g., `transition: height 1s;`: whenever height changes, animate the change over 1s
  - Example: http://oak.cs.ucla.edu/classes/cs144/examples/css-transition.html

```
<!DOCTYPE html>
<html>
<head>
<title>CSS animation</title>
<style>
    div {
        height: 1em;
        color: lightgray;
        background: blue;
        transition: height 1s;
    }
    div:hover {
        height: 10em;
    }
</style>
</head>
<body>
    <div>CSS Transition</div>
</body>
</html>
```

  - `transform` property: transform the shape of the element
    - e.g., `transform: rotate(45deg)scale(1.5);`
- General animation using `@keyframes` rule
  - `@keyframes` rule describes the sequence of animation
  - Apply a keyframe rule to an element by the `animation` property
  - Example: http://oak.cs.ucla.edu/classes/cs144/examples/css-animation.

html

```html
<!DOCTYPE html>
<html>
<head>
<title>CSS animation</title>
<style>
    @keyframes css3animation {
        0%    { background: red; }
        50%   { background: yellow; }
        100%  { background: green; }
    }

    #header1 {
        animation: css3animation 3s;
    }
</style>
</head>
<body>
    <h1 id="header1">CSS Animation</h1>
</body>
</html>
```

- Other relevant CSS3 properties
  * `animation-delay`: when the animation will start
  * `animation-play-state: paused|running` whether the animation is running or paused
  * `animation-iteration-count`: # of times animation is played (or `infinite`)

Web Storage

- HTML5 provides `localStorage`: a persistent "storage" to store data locally
- Example

```javascript
// store and retrieve data
localStorage["username"] = "John";
localStorage["object"] = JSON.stringify(obj);
```

```
let name = localStorage["username"];

// iterate over all stored keys
for(let key in localStorage) {
    let value = localStorage[key];
}

localStorage.removeItem("username");
localStorage.clear();  // delete everything
```

- `localStorage` and `sessionStorage`
  - Associative key-value store
  - HTML5 standard allows storing any object, but most browsers support only string
  - `localStorage` persists over multiple browser sessions
    - Separate storage is allocated per each server
  - `sessionStorage` persists only within the current browser tab
    - Data disappears once the browser tab is closed
    - If two tabs from the same server is opened, they get separate storage

HTML5 Session history

- Example: http://mail.yahoo.com
  - Q: Open an email. When a user presses back button, what will the user expect?
  - Q: Knowing what we know, what is likely to happen?
    - Q: What would have happened if "opening the email" was pointing to a new URL?
    - Q: What would happen if "opening the email" was simple JavaScript code without changes in the URL?
- *Back button*
  - Browser's back button behavior may cause a serious usability issue unless handled well
  - User expects the previous app state *within* the SPA

  - But browser may actually unload the app and show the previous page in the history
- *Deep link*
  - Q: When a user "saves" a URL, what does the user expect to see when the user visits the URL later?
    - e.g., when the user saved the URL, an email message was open.

- Pre-HTML5 solution: URL fragment identifier
  - Change in URL fragment identifier does not reload a page
    - Navigation within the same page
  - Associate each "state" of the app with a unique URL fragment identifier
  - Back button navigates within the same page without reload
- Session history API in HTML5
  - Save and update app state in response to the back and forward buttons without reloading the page
  - `window.location.hash`: standardized API for pre-HTML5 solution
    - Update `window.location.hash` to change fragment identifier of the URL
    - The new URL value is appended to the browser history
    - Set `window.onhashchange` to a custom handler function to change what happens when the fragment identifier changes as a result of history navigation
  - `history.pushState(object, title, url)` and `history.replaceState(object, title, url)`
    - Allows saving an "object" as part of browser
    - Appends (`pushState`) or replaces (`replaceState`) browser history
    - Set `window.onpopstate` to a custom handler function to update the app using the "popped object" as a result of history navigation

References

- XMLHttpRequest: https://xhr.spec.whatwg.org/
- XML 1.1 specification: https://www.w3.org/TR/2006/REC-xml11-20060816/
- XML namespace 1.1 specification: https://www.w3.org/TR/2006/REC-xml11-20060816/

- Cross-Origin Resource Sharing: https://www.w3.org/TR/cors/
- CSS Animation: https://www.w3.org/TR/css-animations-1/
- Web Storage: https://www.w3.org/TR/webstorage/