Front End Visualization Notes:

Sean al-Baroudi

January 19, 2018

aaa

- When a webpage is loaded, there is a difference between the **HTML document**, and the **DOM model**. The page that you see loaded is an instance of the DOM model, that was interpreted from the HTML document.
- Changes in the DOM done dynamically have nothing to do with HTML
- Hiearchy of Objects in DOM Model:
  1. Window
  2. Document
  3. HTML
  4. BODY...and all its nested objects.
-
-
-
-
-
-
-
-
-
-
- JQuery is a JavaScript Framework for implementing advanced functionality, and hiding common boilerplate that occurs from basic JavaScript.
- JQuery centers around *Selectors*, which are search operations that find DOM elements and return a set like structure.
- Given a set of elements (or single one), **Actions** can be performed on such elements. Mutation, or filtering of the elements (changing the set being operated on) are common things a developer might do.
- Selector Syntax:

  ```
  $(selector1,selector2,...).action()
  ```

  The $ is a symbol that indicates a jQuery call is about to be defined.
  (selector) can be anything - types of DOM tags to unique IDs for elements in our DOM tree. For every selector expression, jQuery searches the DOM tree for matching elements, and appends set results to a chronological list.
  (actions) involve callbacks or actual functions, that affect the selected set in some way.
- You can of course nest and chain jQueries. As actions also return sets, this can be done many times in a row to crush code into one line.
- Basic JQuery Start up Code:

  ```
  $(document).ready(function() {

  <put code here>

  });
  ```

  Here, the entire document was selected. an Event Listener has been added, with the condition that it triggers when the DOM model is fully loaded. A anonymous call back function is launched, that executes whatever jQuery commands you specify.
- **Attributes vs Properties:** The former refers to meta-data in the HTML document. The latter refers to meta-data in the DOM tree; Properties are mutable; attributes are generally not. Use the **prop()** function to alter node properties.
-
-
-
-
-

Understanding Event Handling and Listening (W3 Specification):

- The model is very complex. For the dynamic DOM tree, javascript can setup event listeners for specific nodes in the tree. **Every node in the tree can have multiple registered event listeners.**
- **Events per node are registered in order** (so earlier registered events are hit first).
- You can also have multiple event handlers for the same event type, on the same node; again, they are executed relative to registration order.
- When a user causes an interaction event on a webpage in the browser, this causes the following to occur:
  1. A user interaction/browser signal occurs; an event with a possible target node is created.
  2. This event is put into the browser's Event Queue.
  3. The browser will eventually act on the next element in the Event Queue (pop). The following set of actions occur:
     (a) From the DOM root, a Propogation Path to the target node is created (a list of nodes).
     (b) **Capture Phase:** For every ancestor in the Propogation Path, the de-queued event is checked against each ancestor's event listeners. Ancestors who are capturers may act in this phase.
     (c) **At Target Phase:** The actual element that was interacted with is found. The element's listeners are checked with the event. Some may trigger and execute.
     (d) **Bubbling Phase:** Propogation path is backtracked. And each ancestor's event listeners are again checked with the event. They may or may not act again, depending on their configuration.
  4. Bubbling Phase can be turned off in jQuery; and registering anscestor event triggers is also optional. For most things, we have event triggers at the actual node the user has interacted with.
  5.
  6. Low Level Code to add an Event Listener (in Javascript):

     ```
     target.addEventListener(EventType, listener, T/F)
     ```

     jQuery wraps these calls in its library. Target is the element in question, EventType is a user action (click, hover..), listener is our function that responds to the event in a predefined way. True means we have a Capture phase. False means we have Bubbling Phase (the *default* in jQuery).

Implementation of W3 model in jQuery:

- In jQuery, Bubbling is the default, but Capturing is not.
- For more fine grained control, there are two stop methods to stop propogation of events.
  1. event.stopPropogation(): This method will allow all registered events at the target to finish executing; but any bubbled ancestor event handlers will be prevented from firing.
     [Image]
  2. event.stopImmediatePropogation(): Event handlers on the current target will be stopped, in addition to bubbling events with target ancestors.
     [Image]
-
-
-
-
-
-

-
-
-
-
-

Dealing with 1st Class Functions in JS:

- Functions are mutable objects in JS. You can add properties to them after they are made.
- First Class Function: a function that can be used the same way as variables in a program. For example, a 1st class function can:
  1. ...be generated at run time, and not interp/compile time.
  2. ...be stored in a data structure.
  3. ...be passed as an argument.
  4. ...returned as a value to another function.
  5. ...be assigned to another reference name.
- **Higher Order Function:** One that accepts and returns other functions.
- **Invoking:** is another name for calling a function.
- Function Declaration/Statement: A block of code with a specified function name. This is stored in memory after being read.
- Function Expression: is a block of code (anonymous - no name) that returns a value when executed.
- When a function name is specified by itself in a statement, this references the body of code (not invoked).
- To invoke the function, we state the name with Parens () and arguments after it. This runs the block of code and returns a value.
-
-
-
-
-
-

Glossary:

- **Event Bubbling:** Refers to passing the event back up the propogation path to ancestors, **after** the target event listener(s) have activated.
- **Event Capturing:** When an event matches an event handler that is registered to one of our DOM elements.
- **Event Type:** What kind of action has occured on the page - "click", "blur", "load", "mouseover", etc.
- **Bubbling:** A property an event can have. If it bubbles, it will go back up the propogation path after the target handler(s) have compelted.
- **Capturing:** A property an event can have; this means that ancestors of the target may operate on the event, with suitable event handlers.
-

References

[1] https://www.dashingd3js.com

[2] https://bost.ocks.org/mike/join/

[3] https://timmknight.github.io/2015/first-class-functions-javascript/

[4] https://stackoverflow.com/questions/4728073/what-is-the-difference-between-an-expression-and-a-statement-in-python

[5] https://hackernoon.com/javascript-and-functional-programming-an-introduction-286aa625e26d

[6] https://hackernoon.com/javascript-and-functional-programming-pt-2-first-class-functions-4437a1aec217