CS472 WAP

Tree Manipulation

DOM, jQuery

Main Point Preview

- An efficient and clear style is to use object literals to create complex DOM elements in jQuery. Lists of elements should be created as a single list element before insertion for efficiency.
- *Science of Consciousness: Actions arising from quiet levels of awareness are naturally efficient.*

Aspects of the DOM and jQuery

- **Identification:**
  - how do I obtain a reference to the node that I want.
  - using css-like selectors to get target nodes
- **Traversal:**
  - how do I move around the DOM tree.
  - using children, sibling, parent, etc links to get target nodes
- **Node Manipulation:**
  - how do I get or set aspects of a DOM node.
  - e.g., style, attributes, innerHTML
- **Tree Manipulation:**
  - how do I change the structure of the page.

DOM innerHTML hacking

- Why not just code this way?

```
document.getElementById("myid").innerHTML +=
  "<p>A paragraph!</p>";
```

- Imagine that the new node is more complex

```
document.getElementById("myid").innerHTML +=
  "<p style='color: red; " +
  "margin-left: 50px;' " +
  "onclick='myOnClick();'>" +
  "A paragraph!</p>";
```

  - ugly
  - bad style on many levels
  - HTML (and CSS and JS!) code embedded within JS
- error-prone: must carefully distinguish " and '

Create nodes in jQuery

- jQuery creates a DOM node if the argument is an HTML element

```
const newElement = $("<div>");
```

- creating element does not add it to the page
- can be added as child of an existing element
  - *append* for last child, *prepend* for first child

```
$(“#myid”).append(newElement);
```

- jQuery programmers typically - 1 line

```
$("#myid").append($("<div>"));
```

- Can remove any matched elements

```
$("li:contains('kangeroo')").remove();
```

Creating complex nodes in jQuery

- The terrible way, this is no better than innerHTML hacking

```
$("<p id='myid' class='special'>My paragraph is awesome!</p>")
```

- The bad way, decent jQuery, but we can do better

```
$("<p>")
  .attr("id", "myid")
  .addClass("special")
  .text("My paragraph is awesome!");
```

- The good way

```
$("<p>", { "id": "myid", "class": "special", "text": "My paragraph is
  awesome!" });
```

Poor jQuery example

```
$("#ex2 span.special").each(function(i, elem) {
  const img = $("<img>")
    .attr("src", "../images/laughing_man.jpg")
    .attr("alt", "laughing man")
    .css("vertical-align", "middle")
    .css("border", "2px solid black")
    .click(function() {
      alert("clicked");
    });
  $(elem).prepend(img);
});
```

Good jQuery example

```
$("#ex3 span.special").prepend($("<img>", {
  "src": "../images/laughing_man.jpg",
  "alt": "laughing man",
  "css": {
    "vertical-align": "middle",
    "border": "2px solid black"
  },
  "click": function() {
    alert("clicked");
  }
}));
```

Insert via .append or .prepend

- if creating list of elements
  - best practice is to create entire list before inserting
  - entire DOM redraws whenever modified

- .append
  - make last child of matched elements

- .prepend
  - make first child of matched elements

*

Insert via .before and .after

- .after, .before
  - insert new (or move selected) content after or before matched elements

jQuery $ function signatures

- Responding to the page ready event

```
$(function);
```

- Identifying elements

```
$("selector", [context]);
```

- Upgrading DOM elements

```
$(elements);
```

- Creating new elements

```
$("<html>", [properties]);
```

Main Point

- An efficient and clear style is to use object literals to create complex DOM elements in jQuery. Lists of elements should be created as a single list element before insertion for efficiency.
- *Science of Consciousness: Actions arising from quiet levels of awareness are naturally efficient.*

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

The TM Technique and Pure Consciousness

1. jQuery is a powerful and widely used JavaScript API for working with the DOM that provides cross-browser compatibility.
2. The $() function is heavily overloaded. It will be a different function depending on the arguments it has, including running a callback function on page load, selecting DOM elements, wrapping DOM elements, and creating new DOM elements.

---

3. **Transcendental consciousness**. The TM Technique is a convenient and cross-platform API for experiencing transcendental consciousness.
4. **Impulses within the transcendental field:** Thoughts and actions at quiet levels of awareness are simple, natural, and effective because they are in accord with all the laws of nature that reside at these deep levels.
5. **Wholeness moving within itself:** Advanced TM Techniques and the TM-Sidhi Programs are powerful APIs for bringing the calm dynamism and bliss of pure consciousness into the experiences of daily activity.


MAHARISHI
HIGHER EDUCATION
FOR HIGHER CONSCIOUSNESS
INTERNATIONAL UNIVERSITY