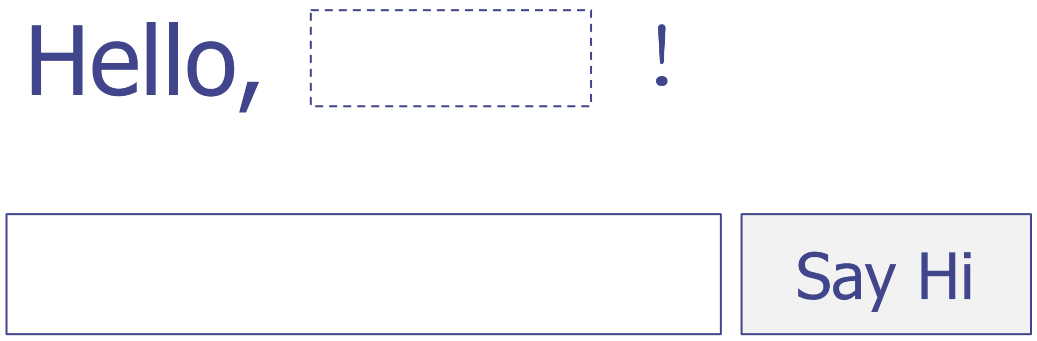
CS3220 Web and Internet Programming

Client-Side JavaScript and jQuery

Chengyu Sun
California State University, Los Angeles

Client-Side JavaScript Example



About The Example

- Event handler `onclick`
  - See [Section 6.1.5.2, HTML 5 Specification]
- Embed JavaScript in HTML using the `<script>` tag
  - Link to external file or enclose code directly
  - Can appear any number of times in both <head> and <body>
  - HTML 5 no longer requires the type attribute
- Run inside a browser: `document` is the object representing the current page

Processing an HTML Document

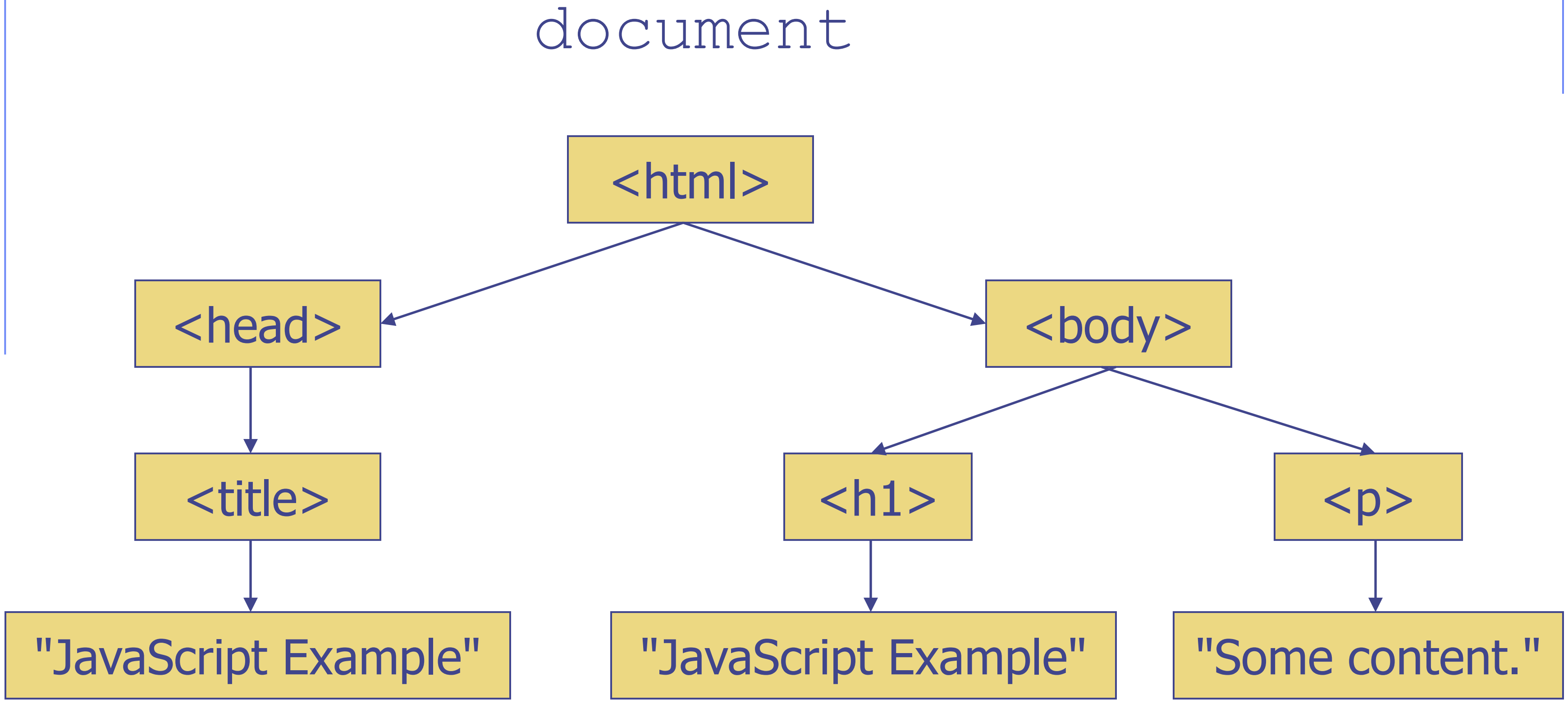
```
<html>
<head><title>JavaScript Example</title></head>
<body>
    <h1>JavaScript Example</h1>
    <p>Some content.</p>
</body>
</html>
```

- As a text file – difficult and inefficient
- As an object

Document Object Model (DOM)

- Representing HTML documents as objects so they can be processed more easily by a programming language

DOM Representation



Common terminology:
parent, child, sibling, ancestor, descendant

About DOM

- Browser is responsible for parsing HTML document and creating the corresponding DOM object
- Changes to the DOM object are reflected on the page display

Manipulate a Document

- JavaScript Document API
  - Find Elements
  - Modify Elements
  - Create Elements

Find Elements

- document.getElementById()
- document.getElementsByName()
- document.getElementsByTagName()

Modify Elements …

- HTMLElement properites and methods
  - IE
    - innerHTML
    - innerText
    - insertAdjacentHTML()
    - insertAdjacentText()
  - Netscape/Mozilla
    - innerHTML
  - Element-specific

… Modify Elements

- node
  - setAttribute(), removeAttribute()
  - appendChild(), removeChild()
  - insertBefore(), replaceChild()

Create Elements

- document
  - createElement()
  - createTextNode()

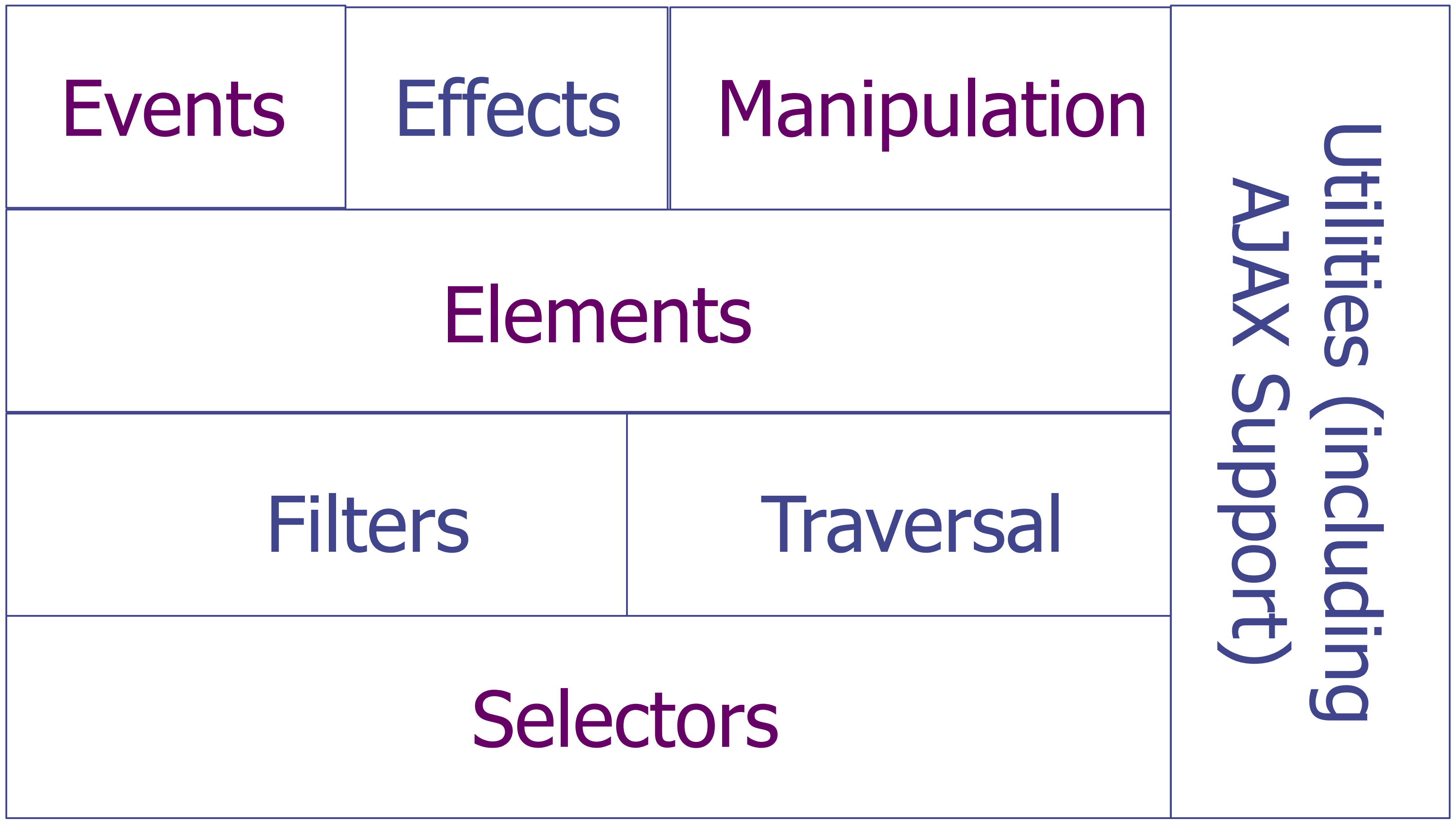
Problems with Plain Client-Side JavaScript

- Verbose code for even simple tasks
- Browser compatibility issues

jQuery: Made Client-Side JavaScript Great Again

- Hide browser incompatibility
- Greatly simplify/improve DOM operations, event handling, and asynchronous request and response
- Usage statistics - https://trends.builtwith.com/javascript/javascript-library/traffic/Entire-Internet

jQuery Overview …



… jQuery Overview …

- **Selectors**: select elements in a document
- **Filters**: filter selected elements based on some conditions
- **Traversal**: traverse the DOM tree from selected elements (e.g. select their children, parents, etc.)
- **Elements**: get and set element attributes, properties, content, values, and/or CSS classes

… jQuery Overview

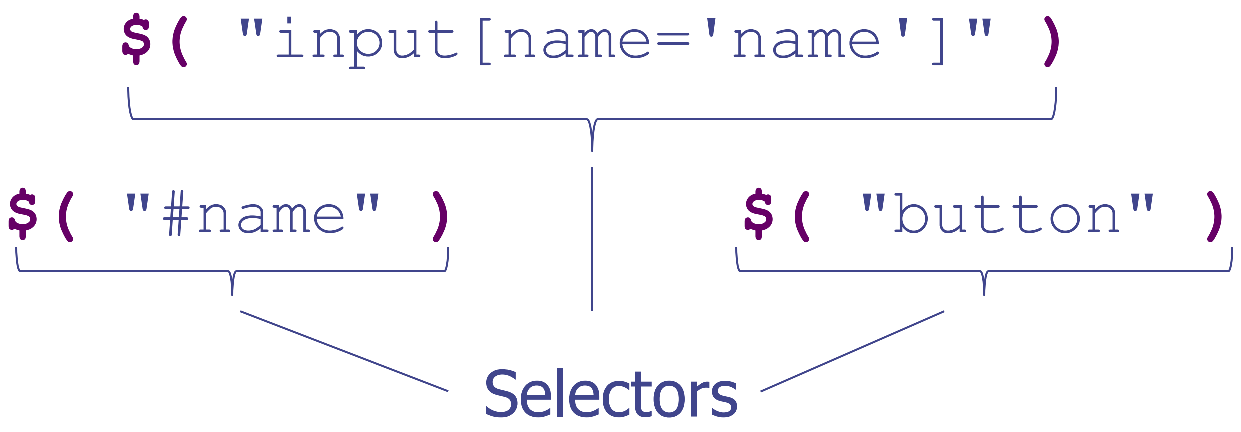
- **Events**: handle events
- **Effects**: basic effects like hiding, showing, fading and so on
- **DOM Manipulation**: add/remove elements
- **Utilities** (including support for AJAX operations)

jQuery Example

- Usage
- jQuery wrapper
- Selectors
- Elements
- Events and event handling
- DOM Manipulation

jQuery Wrapper

- `jQuery()` or `$()`
  - Return a collection of matched elements either found in the DOM based on passed argument(s) or created by passing an HTML string.



```
$( "input[name='name']" )
$( "#name" )
$( "button" )
```

Basic Selectors

- By id
  - `$("#foo")`
- By tag name
  - `$("div")`
- By CSS class
  - `$(".foo")`
- By attribute
  - `$("[name]")`
  - `$("[name='joe']")`

Combine Selectors

- Select all the `<div>` elements with CSS class `foo` and an attribute `bar`

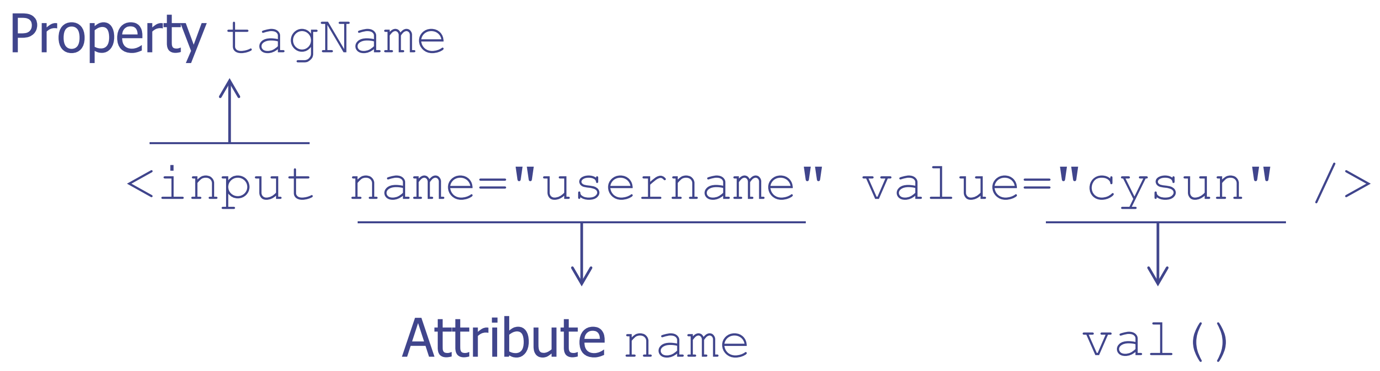
```
$("div.foo[bar]")
```

- Select all the `<div>` elements, and all the elements with CSS class `foo`, and all the elements with an attribute `bar`

```
$("div,.foo,[bar]")
```

What Can We Do With An Element

- Get and set
  - `html()`, `text()`
  - `attr()`
  - `prop()`
  - `val()`
- Manipulate CSS
  - `addClass()`
  - `removeClass()`
  - `toggleClass()`
  - `hasClass()`



```
<input name="username" value="cysun" />
```

Typical Event and Event Handling in jQuery

Event Event Handler

$$\begin{tikzcd}
\text{Event} \arrow[d] & \text{Event Handler} \arrow[d] \\
\texttt{.click(} & \texttt{function()\{}
\end{tikzcd}$$

```
$("#click").click( function(){
        ... ...
});
```

Unobtrusive JavaScript:
separate style, behavior, and structure.

↓

```
<button id="click" ~~onclick="display();"~~>
Click Me</button>
```

Other Events

- Mouse events
  - `.click()`
  - `.dbclick()`
  - …
- Keyboard events
  - `.keyup()`
  - `.keydown()`
  - `.keypress()`
  - …
- Form events
  - `.change()`
  - `.submit()`
  - …
- Browser events
  - `.resize()`
  - …
- Document events

Document Ready Event

- Triggered when the DOM hierarchy of the HTML document is fully constructed

```
$( document ).ready( handler )
```

⬇

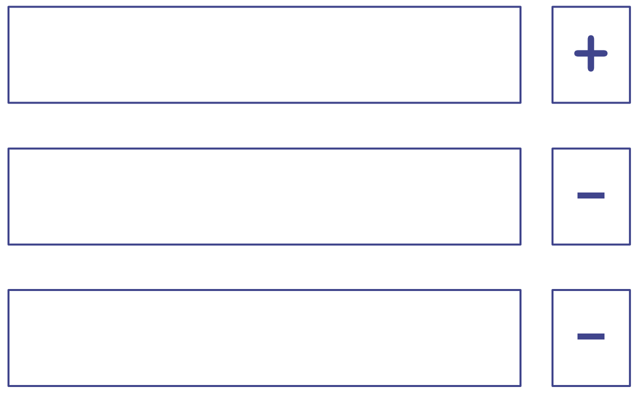
```
$().ready( handler )
```
(not recommended)

⬇

`$( handler )`

Example: Language Question

What program languages do you know?



DOM Manipulation

- Insertion
  - Around (i.e. parent)
  - Inside (i.e. children)
    - `append()`, …
  - Outside (i.e. sibling)
- Removal, e.g. `remove()`, …
- Replacement

Example: Tic Tac

o	x	
	x	x
	o	

Count

- Place `x` and `o` alternately on the board
- Count the number of `x`'s and `o`'s on the board using `.each()` method

About jQuery Object …

- `$(selector)` → jQuery object
- A jQuery object may contain multiple HMTL elements, e.g. `$("td")`
- Most of jQuery object methods apply to each of the selected elements, e.g.
  - `$("td").`**`click`**`(function(){})`
  - `$("tr").`**`append`**`("<td></td>")`
- `.each(function)` can be used to apply a user-defined function to each of the selected element

… About jQuery Object

- Distinguish between HTML elements and jQuery objects
  - **`this`** in an event handler or `.each(function)` is an HTML element
  - Wrapping **`$()`** around an HTML element turns it into a jQuery object (so we can use jQuery object methods like `.click()`, `.append()`, and so on)

References

- jQuery API