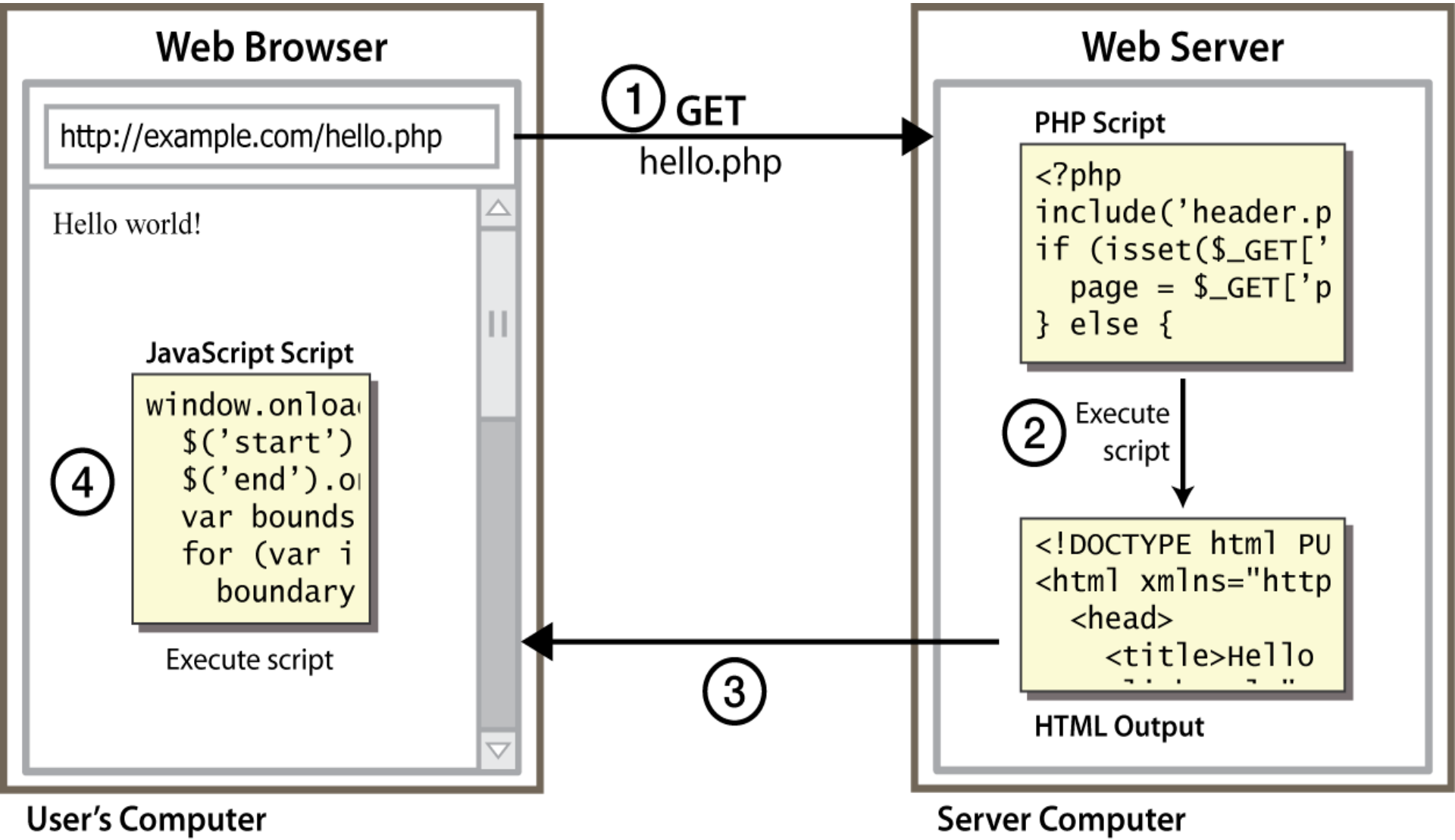
Brief Introduction to JavaScript & Data Visualization

2018.5.7

JavaScript

Client Side Scripting



Why use client-side programming?

PHP already allows us to create dynamic web pages. Why also use client-side scripting?

- client-side scripting (JavaScript) benefits:
  - **usability**: can modify a page without having to post back to the server (faster UI)
  - **efficiency**: can make small, quick changes to page without waiting for server
  - **event-driven**: can respond to user actions like clicks and key presses

Why use client-side programming?

- server-side programming (PHP) benefits:
  - **security**: has access to server's private data; client can't see source code
  - **compatibility**: not subject to browser compatibility issues
  - **power**: can write files, open connections to servers, connect to databases, ...

What is Javascript?

- a lightweight programming language ("scripting language")
  - used to make web pages interactive
  - insert dynamic text into HTML (ex: user name)
  - **react to events** (ex: page load user click)
  - get information about a user's computer (ex: browser type)
  - perform calculations on user's computer (ex: form validation)

What is Javascript?

- a web standard (but not supported identically by all browsers)
- NOT related to Java other than by name and some syntactic similarities

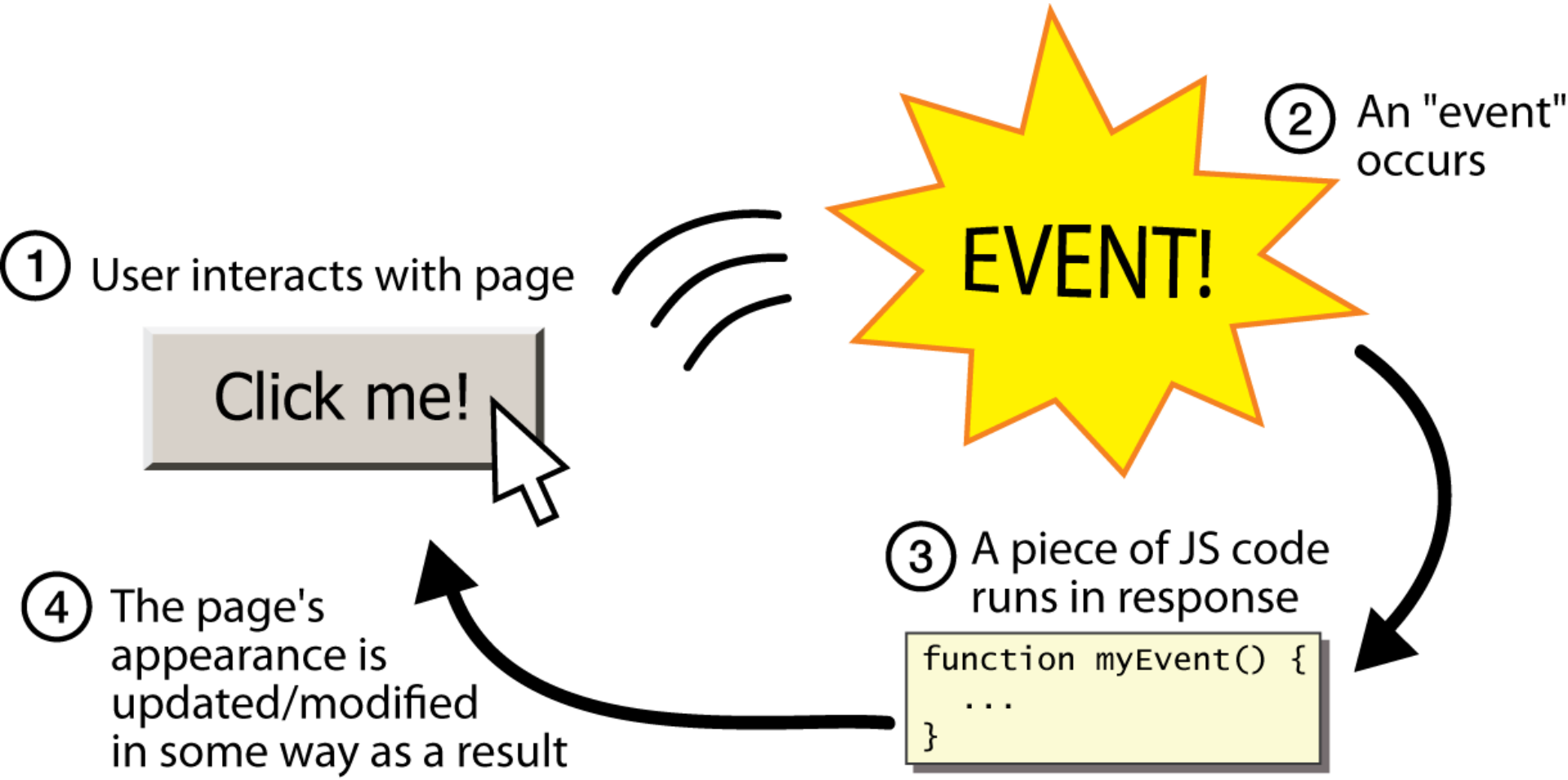
Javascript vs Java

Linking to a JavaScript file: `script`


```
<script src="filename" type="text/javascript"></script>
```
HTML

- script tag should be placed in HTML page's head
- script code is stored in a separate .js file
- JS code can be placed directly in the HTML file's body or head (like CSS)
  - but this is bad style (should separate content, presentation, and behavior

Event-driven programming



Event-driven programming

- you are used to programs start with a main method (or implicit main like in PHP)
- JavaScript programs instead wait for user actions called *events* and respond to them
- event-driven programming: writing programs driven by user events
- Let's write a page with a clickable button that pops up a "Hello, World" window...

Buttons


```
<button>Click me!</button>
```
HTML

- button's text appears inside tag; can also contain images
- To make a responsive button or other UI control:
  1. choose the control (e.g. button) and event (e.g. mouse 1. click) of interest
  2. write a JavaScript function to run when the event occurs
  3. attach the function to the event on the control

JavaScript functions


```
function name() {
statement ;
statement ;
...
statement ;
}
```
JS

```
function myFunction() {
       alert("Hello World!");
}
```
JS

- the above could be the contents of example.js linked to our HTML page
- statements placed into functions can be evaluated in response to user events

Event handlers

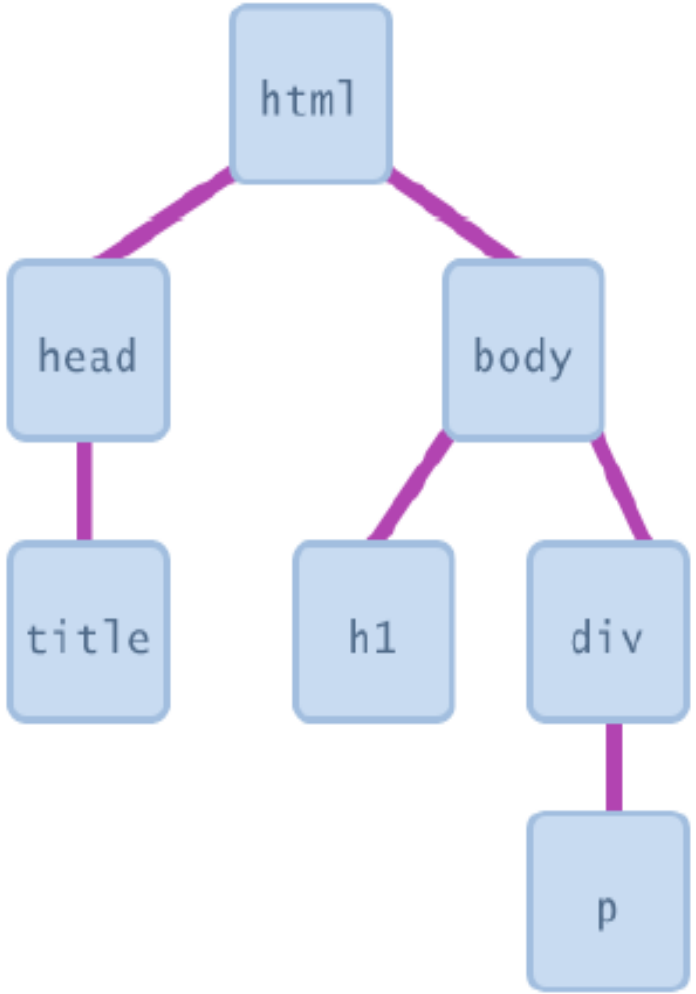

```HTML
<element attributes onclick="function();">...
```

```HTML
<button onclick="myFunction();">Click me!</button>
```

- JavaScript functions can be set as event handlers
  - when you interact with the element, the function will execute
- onclick is just one of many event HTML attributes we'll use
- but popping up an alert window is disruptive and annoying
  - A better user experience would be to have the message appear on the page...

Document Object Model (DOM)

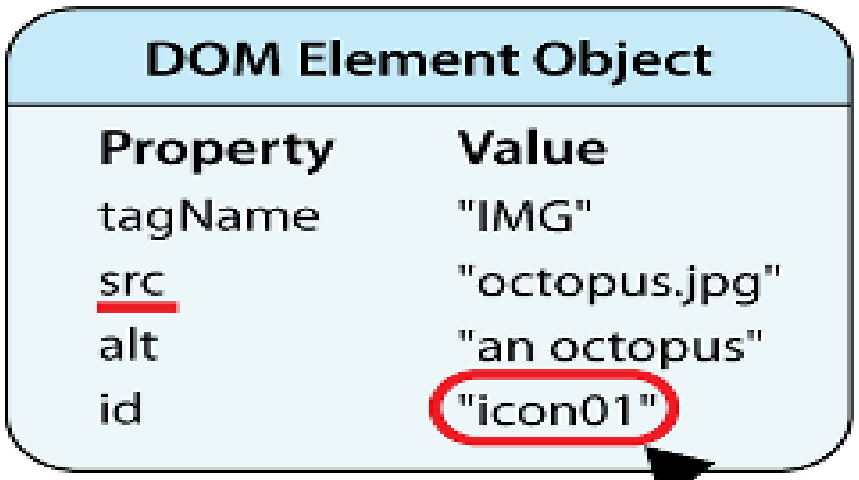
- most JS code manipulates elements on an HTML page
- we can examine elements' state
  - e.g. see whether a box is checked
- we can change state
  - e.g. insert some new text into a div
- we can change styles
  - e.g. make a paragraph red



DOM element objects

HTML

```
<p>
  Look at this octopus:
  <img src="octopus.jpg" alt="an octopus" id="icon01" />
  Cute, huh?
</p>
```



JavaScript

```
var icon = document.getElementById("icon01");
icon.src = "kitty.gif";
```

Accessing elements: document.getElementById


```JS
var name = document.getElementById("id");
```

```HTML
<button onclick="changeText();">Click me!</button>
<span id="output">replace me</span>
<input id="textbox" type="text" />
```

```JS
function changeText() {
	var span = document.getElementById("output");
	var textBox = document.getElementById("textbox");

	 textbox.style.color = "red";

}
```

More Javascript Syntax

Variables


```
var name = expression;                                    JS
```

```
var clientName = "Connie Client";
var age = 32;
var weight = 127.4;                                       JS
```

- variables are declared with the var keyword (case sensitive)
- types are not specified, but JS does have types ("loosely typed")
  - `Number, Boolean, String, Array, Object, Function, Null, Undefined`
  - can find out a variable's type by calling `typeof`

Number type


```
var enrollment = 99;
var medianGrade = 2.8;
var credits = 5 + 4 + (2 * 3);
```

JS

- integers and real numbers are the same type (no int vs. double)
- same operators: + - * / % ++ -- = += -= *= /= %=
- similar precedence to Java
- many operators auto-convert types: "2" * 3 is 6

Math object


```
var rand1to10 = Math.floor(Math.random() * 10 + 1);
var three = Math.floor(Math.PI);
```

JS

- **methods:** `abs, ceil, cos, floor, log, max, min, pow, random, round, sin, sqrt, tan`
- **properties:** `E, PI`

Special values: null and undefined


```
var ned = null;
var benson = 9;
// at this point in the code,
// ned is null
// benson's 9
// caroline is undefined
```

JS

- `undefined` : has not been declared, does not exist
- `null` : exists, but was specifically assigned an empty or null value
- Why does JavaScript have both of these?

Logical operators

- > < >= <= && || ! == != === !==
- most logical operators automatically convert types:
  - 5 < "7" is true
  - 42 == 42.0 is true
  - "5.0" == 5 is true
- === and !== are strict equality tests; checks both type and value
  - "5.0" === 5 is false

if/else statement (same as Java)


```
if (condition) {
        statements;
} else if (condition) {
        statements;
} else {
        statements;
}
```

JS

- identical structure to Java's if/else statement
- JavaScript allows almost anything as a condition

Boolean type


```
var iLike190M = true;
var ieIsGood = "IE6" > 0; // false
if ("web devevelopment is great") { /* true */ }
if (0) { /* false */ }
```

JS

- any value can be used as a Boolean
  - "falsey" values: 0, 0.0, NaN, "", null, and undefined
  - "truthy" values: anything else
- converting a value into a Boolean explicitly:
  - `var boolValue = Boolean(otherValue);`
  - `var boolValue = !!(otherValue);`

for loop (same as Java)


```
var sum = 0;
for (var i = 0; i < 100; i++) {
        sum = sum + i;
}
```
JS

```
var s1 = "hello";
var s2 = "";
for (var i = 0; i < s.length; i++) {
        s2 += s1.charAt(i) + s1.charAt(i);
}
// s2 stores "hheellllloo"
```
JS

while loops (same as Java)


```
while (condition) {
        statements;
}
```
JS

```
do {
    statements;
} while (condition);
```
JS

- break and continue keywords also behave as in Java

Arrays


```
var name = []; // empty array
var name = [value, value, ..., value]; // pre-filled
name[index] = value; // store element
```

JS

```
var ducks = ["Huey", "Dewey", "Louie"];
var stooges = []; // stooges.length is 0
stooges[0] = "Larry"; // stooges.length is 1
stooges[1] = "Moe"; // stooges.length is 2
stooges[4] = "Curly"; // stooges.length is 5
stooges[4] = "Shemp"; // stooges.length is 5
```

JS

Array methods


```
var a = ["Stef", "Jason"]; // Stef, Jason
a.push("Brian"); // Stef, Jason, Brian
a.unshift("Kelly"); // Kelly, Stef, Jason, Brian
a.pop(); // Kelly, Stef, Jason
a.shift(); // Stef, Jason
a.sort(); // Jason, Stef
```

JS

- array serves as many data structures: list, queue, stack, ...
- methods: `concat, join, pop, push, reverse, shift, slice, sort, splice, toString, unshift`
  - push and pop add / remove from back
  - unshift and shift add / remove from front
  - shift and pop return the element that is removed

String type


```
var s = "Connie Client";
var fName = s.substring(0, s.indexOf(" ")); // "Connie"
var len = s.length; // 13
var s2 = 'Melvin Merchant';
```
JS

- **methods:** `charAt, charCodeAt, fromCharCode, indexOf, lastIndexOf, replace, split, substring, toLowerCase, toUpperCase`
  - charAt returns a one-letter String (there is no char type)
- length property (not a method as in Java)
- Strings can be specified with "" or ''
- concatenation with + :
  - 1 + 1 is 2, but "1" + 1 is "11"

Splitting strings: split and join


```
var s = "the quick brown fox";
var a = s.split(" "); // ["the", "quick", "brown", "fox"]
a.reverse(); // ["fox", "brown", "quick", "the"]
s = a.join("!"); // "fox!brown!quick!the"
```

JS

- split breaks apart a string into an array using a delimiter
  - can also be used with regular expressions (seen later)
- join merges an array into a single string, placing a delimiter between them

jQuery and AJAX

What is jQuery?

- jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. (jQuery.com)

Dynamic HTML (DHTML)

- Manipulating the web page's structure is essential for creating a highly responsive UI

- Two main approaches
  - Manipulate page via plain JS
  - Manipulate page using JS + library (e.g., jQuery)

Document Object Model (DOM)

- Fancy name for the web page's structure

- Web page is basically a tree structure
  - One node per HTML element
  - Each node can have attributes

Rewriting using innerHTML attribute

```
<span id="stuff"></span>
<form><input id="inpt" onchange="doit()"></form>
<script>
function doit() {
   document.getElementById("stuff").innerHTML =
   document.getElementById("inpt").value;
}
</script>
```

Rewriting the contents of a span. NOTE: There is a browser-compatibility problem in the code above. See next slides.

Getting started with jQuery

- Download a copy of the jquery JS file and store it on your hard drive
- Reference the JS file in your HTML
- Access the jQuery functions via the $ object

jQuery Selector

语法	描述
$(this)	当前 HTML 元素
$("p")	所有 <p> 元素
$("p.intro")	所有 class="intro" 的 <p> 元素
$(".intro")	所有 class="intro" 的元素
$("#intro")	id="intro" 的元素
$("ul li:first")	每个 的第一个 元素
$("[href$='.jpg']")	所有带有以 ".jpg" 结尾的属性值的 href 属性
$("div#intro .head")	id="intro" 的 元素中的所有 class="head" 的元素

Simple example

```
<script src="jquery-1.8.2.min.js"></script>
<span id="stuff"></span>
<form><input id="inpt" onchange="doit()"></form>
<script>
function doit() {
   $("#stuff").text($("#inpt").val());
}
</script>
```

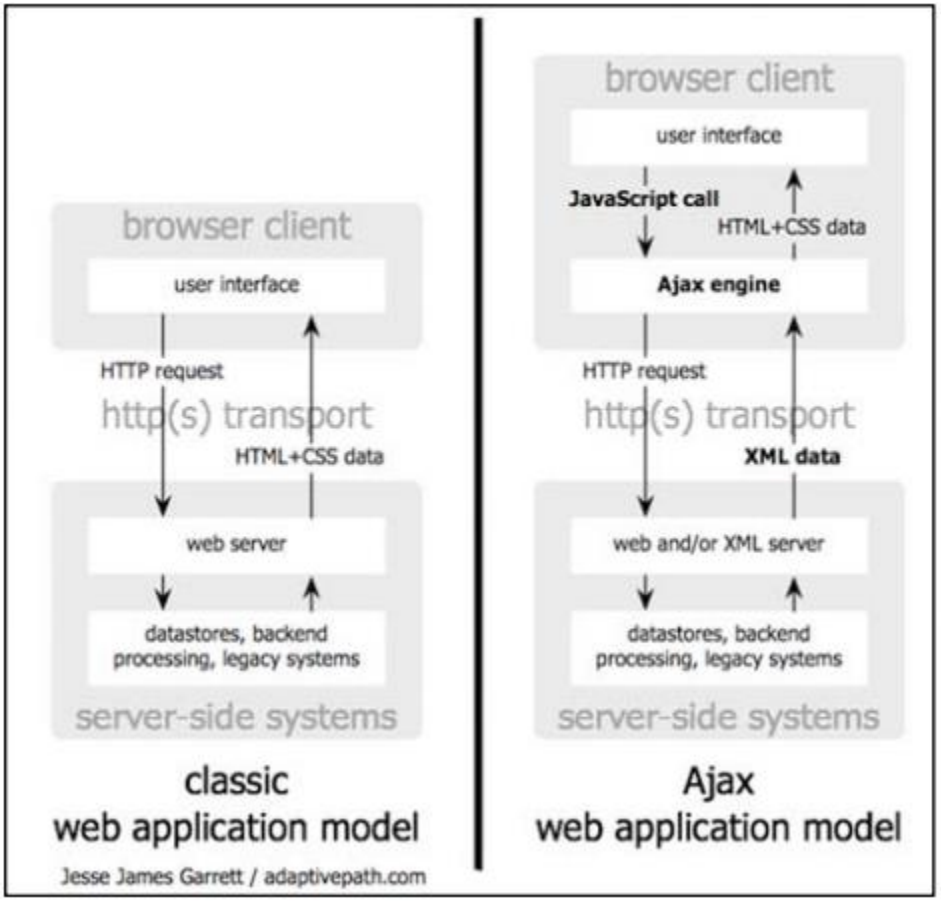
Rewriting the contents of a span. No security problems or cross-browser compatibility problems.

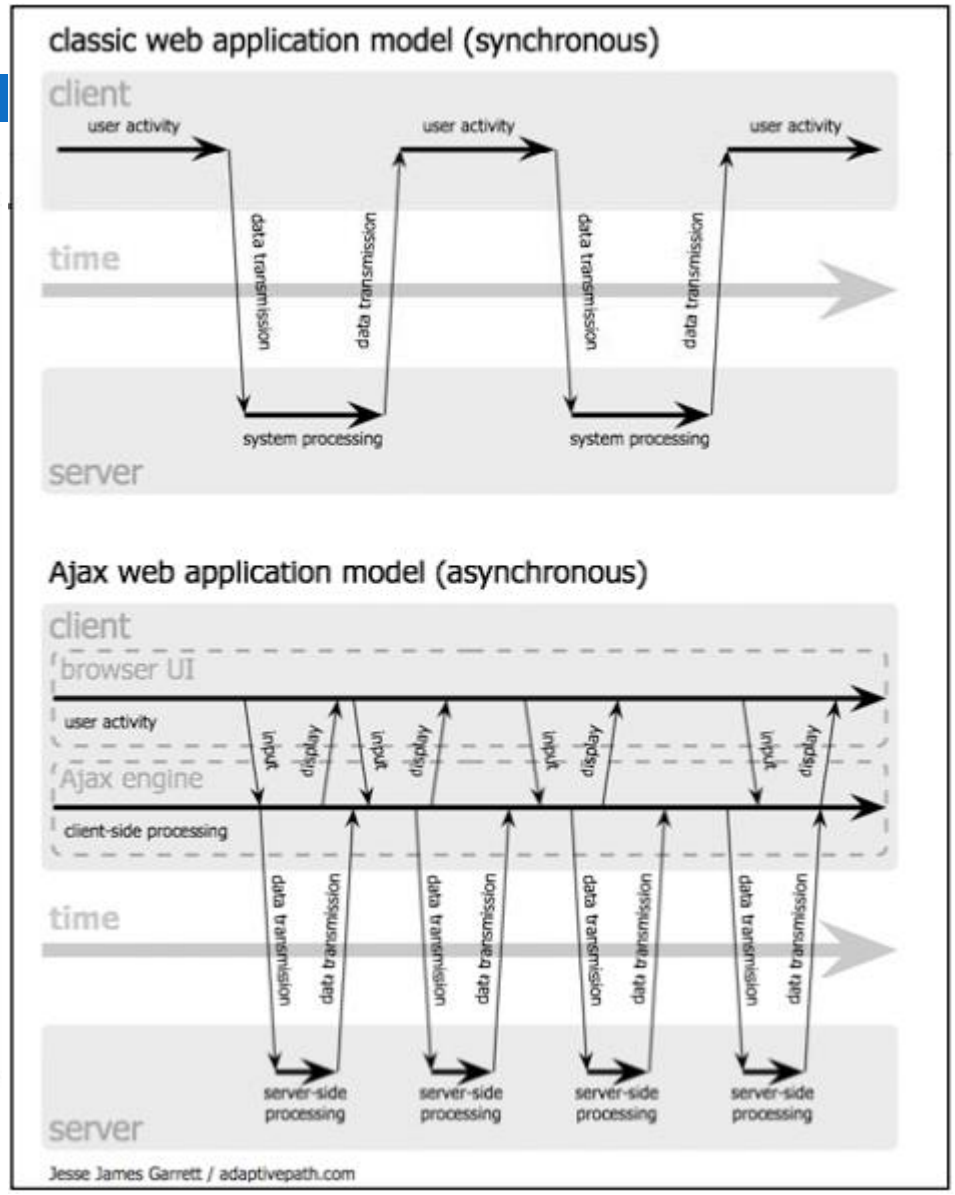
Examples of things you can do with jQuery

- Read the contents of DOM nodes (tag)
- Modify the contents of DOM nodes
- Modify the appearance of DOM nodes
- Create and attach new DOM nodes
- Remove DOM nodes
- Run a function right when the page is ready
- Add and remove event handlers
- Retrieve content from a web server
- Send content to a web server

What is AJAX?

- AJAX is a technology used to facilitate real-time data changes and updates on a page without requiring a page reload.
- AJAX stands for Asynchronous Javascript And XML.
- Let's break that down:
  - Asynchronous: The response from the server doesn't have to be immediate, like a page load does. Other stuff can happen in-between.
  - Javascript: The client-side language which you use to make requests to and handle responses from the server
  - XML: The format often used to pass data between Javascript and the server.


browser client
user interface
HTTP request
http(s) transport
HTML+CSS data
web server
datastores, backend processing, legacy systems
server-side systems
classic web application model
browser client
user interface
JavaScript call
HTML+CSS data
Ajax engine
HTTP request
http(s) transport
XML data
web and/or XML server
datastores, backend processing, legacy systems
server-side systems
Ajax web application model
Jesse James Garrett / adaptivepath.com


classic web application model (synchronous)
client
user activity
user activity
user activity
data transmission
data transmission
data transmission
data transmission
time
system processing
system processing
server
Ajax web application model (asynchronous)
client
browser UI
user activity
input
display
Ajax engine
client-side processing
data transmission
server-side processing
time
server
Jesse James Garrett / adaptivepath.com

AJAX in jQuery

jQuery $.get() 方法

$.get() 方法通过 HTTP GET 请求从服务器上请求数据。

语法：

```
$.get(URL,callback);
```

必需的 *URL* 参数规定您希望请求的 URL。

可选的 *callback* 参数是请求成功后所执行的函数名。

```
$("button").click(function(){
  $.get("demo_test.asp",function(data,status){
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

$.getJSON()
$.post()
…

Data Visualization

- D3.js (https://d3js.org/#introduction)
- Echarts (http://echarts.baidu.com/tutorial.html#5%20%E5%88%86%E9%92%9F%E4%B8%8A%E6%89%8B%20ECharts)
- Chart.js, Gephi, Processing…

D3.js


```
//高宽
var w = 500;
var h = 100;

var dataset = [
    [5, 20], [480, 90], [250, 50], [100, 33], [330, 95],
    [410, 12], [475, 44], [25, 67], [85, 21], [220, 88]
    ];

//创建SVG
var svg = d3.select("body")
    .append("svg")
    .attr("width", w)
    .attr("height", h);

    svg.selectAll("circle")
    .data(dataset)
    .enter()
    .append("circle")
    .attr("cx", function(d) {
        return d[0];
    })
    .attr("cy", function(d) {
        return d[1];
    })
    .attr("r", 5);
```



```
.attr("r", function(d) {
    return Math.sqrt(h - d[1]);
});
```

D3.js


```
svg.selectAll("text")
    .data(dataset)
    .enter()
    .append("text")
    .text(function(d) {
        return d[0] + "," + d[1];
    })
    .attr("x", function(d) {
        return d[0];
    })
    .attr("y", function(d) {
        return d[1];
    })
    .attr("font-family", "sans-serif")
    .attr("font-size", "11px")
    .attr("fill", "red");
```

Echarts


```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>ECharts</title>
    <!-- 引入 echarts.js -->
    <script src="echarts.min.js"></script>
</head>
<body>
    <!-- 为ECharts准备一个具备大小（宽高）的Dom -->
    <div id="main" style="width: 600px;height:400px;"></div>
    <script type="text/javascript">
        // 基于准备好的dom，初始化echarts实例
        var myChart = echarts.init(document.getElementById('main'));
```

Echarts


```
        // 指定图表的配置项和数据
        var option = {
            title: {
                text: 'ECharts 入门示例'
            },
            tooltip: {},
            legend: {
                data:['销量']
            },
            xAxis: {
                data: ["衬衫","羊毛衫","雪纺衫","裤子","高跟鞋","袜子"]
            },
            yAxis: {},
            series: [{
                name: '销量',
                type: 'bar',
                data: [5, 20, 36, 10, 10, 20]
            }]
        };

        // 使用刚指定的配置项和数据显示图表。
        myChart.setOption(option);
    </script>
</body>
</html>
```

- JS: http://www.w3school.com.cn/js/
- jQuery(AJAX): http://www.w3school.com.cn/jquery/
- D3.js: https://d3js.org/#introduction
- Echarts: http://echarts.baidu.com/