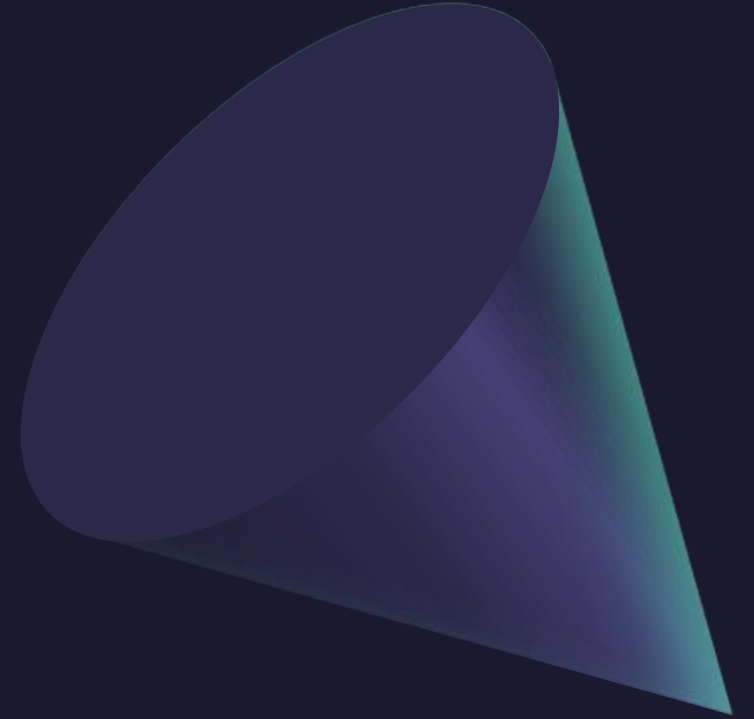


# Client-side Javascript

Software Systems Development

IIT Hyderabad



# Content

---

Meaning

---

HTTP Basics

---

Embedding JS in HTML

---

DOM

---

Event Handling

---

jQuery

---

AJAX

---

# Meaning

- JavaScript is used within web browsers, commonly called ***client-side JavaScript***.
- The **Window** object is the main entry point to all client-side JavaScript features and APIs that represents a web browser window or frame, and you can refer to it with the identifier window.
- The **Window** object defines properties like location , which refers to a Location object that specifies the URL currently displayed in the window and allows a script to load a new URL into the window :



```
window.location = http://www.oreilly.com/;
```

# Meaning



```
> window.document
< #document
  <!DOCTYPE html>
  <html lang="en">
    ► #shadow-root (open)
    ► <head>...</head>
    ► <body>...</body>
  </html>
```

- Document object under window contains the html code that the current window contains.
- DOM (Document Object Model) provides us the flexibility to handle events in js or attain dynamicity.

# HTTP Basics

## HyperText Transfer Protocol

- ✧ Based on request/response *stateless* protocol
  - Client opens connection to server
  - Client sends HTTP request for a resource
  - Server sends HTTP response to the client (w/resource)
  - Client closes connection to server

# Identifying Resources on Web

## URN: Uniform Resource Name

- ✧ **Uniquely identifies resource or name of resource**
- ✧ **Does not tell us how to get the resource**

example:

**“HTML/CSS/Javascript/Web Developers/Yaakov/Chaikin”**

# Identifying Resources on Web

## URI: Uniform Resource Identifier

- ✧ **Uniquely identifies resource or location of resource**
- ✧ **Does not necessarily tell us how to get the resource**

example:

**/official\_web\_site/index.html**



# Identifying Resources on Web

## URL: Uniform Resource Locator

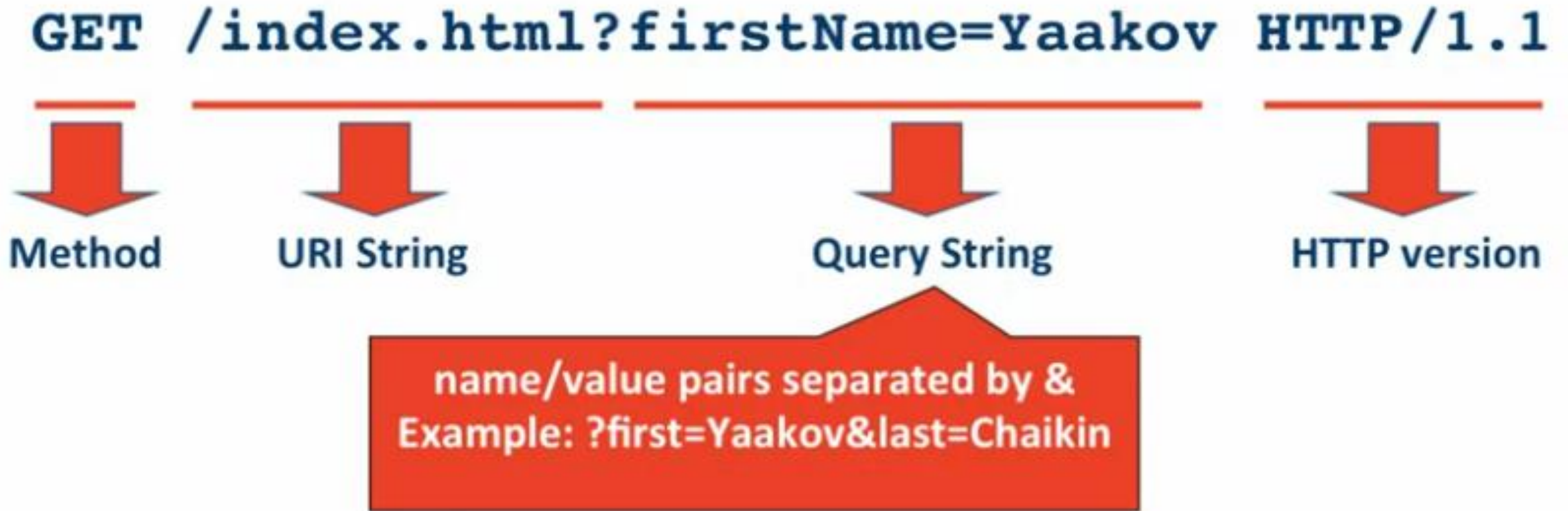
✧ **Form of URI that provides info on how to get resource**

example:

**`http://www.mysite.com/official_web_site/index.html`**



# HTTP Request structure (GET)



# HTTP Request GET and POST

## ✧ GET

- Retrieves the resource
- Data is passed to server as part of the URI
  - I.e., query string

## ✧ POST

- Sends data to server in order to be processed
- Data is sent in the message body

# HTTP Request Structure (POST)

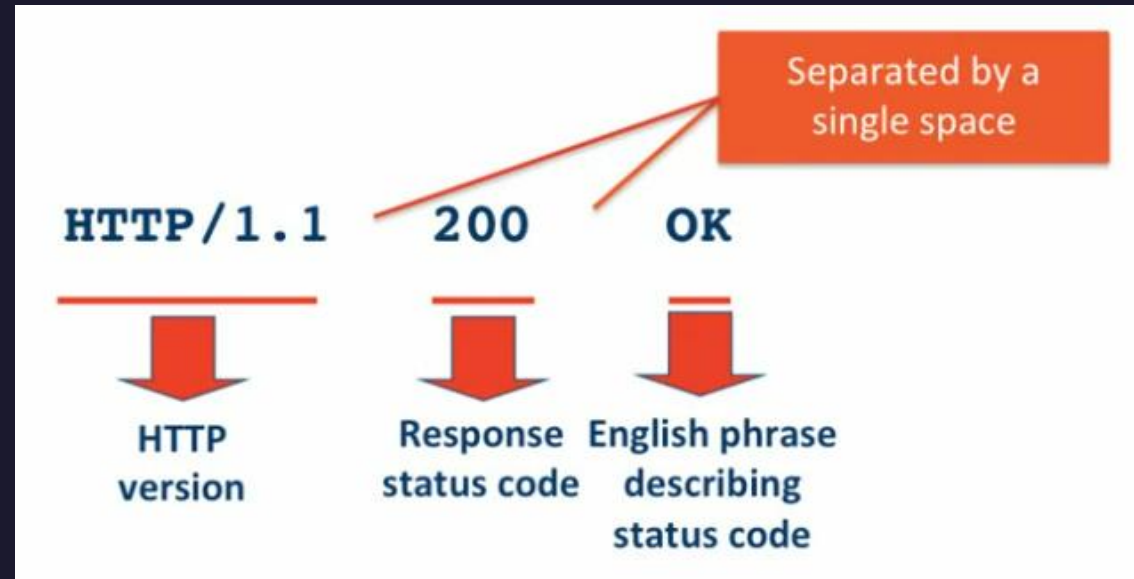
The diagram illustrates the structure of an HTTP POST request. It shows the request line and the request headers. Annotations with arrows point to specific parts of the request line: 'Http request method' points to 'POST', 'Path to source on Web Server' points to '/profile.jsp', and 'Protocol Version Browser support' points to 'HTTP/1.1'. A large curly bracket on the left side groups the headers under the label 'request header'. An arrow on the right points to the 'user=abhi&course=java' parameter, labeling it as 'parameter inside message body'.

```
POST/profile.jsp HTTP/1.1
Host: www.studytonight.com
User-Agent: Mozilla/5.0
Accept: text/xml,text/html,text/plain,image/jpeg
Accept-Language: en-us,en
Accept-Encoding: gzip
Keep-Alive: 300
Connection: keep-alive
user=abhi&course=java
```

Annotations:

- Http request method: `POST`
- Path to source on Web Server: `/profile.jsp`
- Protocol Version Browser support: `HTTP/1.1`
- request header (grouped):
  - `Host: www.studytonight.com`
  - `User-Agent: Mozilla/5.0`
  - `Accept: text/xml,text/html,text/plain,image/jpeg`
  - `Accept-Language: en-us,en`
  - `Accept-Encoding: gzip`
  - `Keep-Alive: 300`
  - `Connection: keep-alive`
- parameter inside message body: `user=abhi&course=java`

# HTTP Response Structure



**HTTP/1.1 200 OK**

**Date: Tue, 11 Aug 2004 19:00:01 GMT**

**Content-Type: text/html**

**<html>**

**<body>**

**<h1>Secret to gaining weight REVEALED!</h1>**

**<p>Develop Coursera courses after work at night while eating sweets to keep yourself awake!</p>**

**</body>**

**</html>**



# HTTP response status codes

## ✧ **200 OK**

- Ok, here is the content you requested

## ✧ **404 Not Found**

- Server can't find the resource requested

## ✧ **403 Forbidden**

- Unauthenticated client tried to access a secure resource

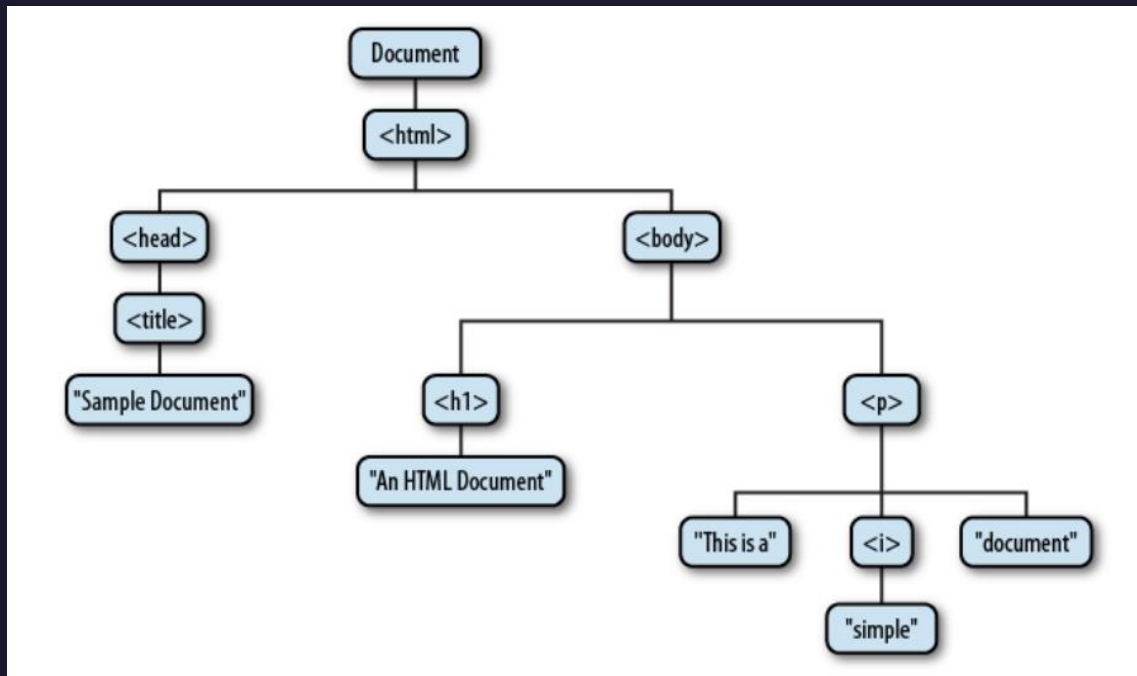
## ✧ **500 Internal Server Error**

- Some unhandled error was raised on the server

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>
</body>
</html>
```

# Embedding JS in HTML

```
<script src="../scripts/util.js"></script>
```



```
<html>
  <head>
    <title>Sample Document</title>
  </head>
  <body>
    <h1>An HTML Document</h1>
    <p>This is a <i>simple</i> document.
  </p>
</html>
```

# DOM

- The Document Object Model, or DOM, is the fundamental API for representing and manipulating the content of HTML and XML documents.





# Selecting Document elements

## By ID

- `var section1 = document.getElementById("section1");`

## By Name

- `var radiobuttons = document.getElementsByName("favorite_color");`

## By Type

- `var spans = document.getElementsByTagName("span");`

## By CSS class

- `var warnings = document.getElementsByClassName("warning");`

## By CSS Selectors

- `querySelectorAll()`
- `querySelector()`

# Event Handling

- A web browser generates an **event** whenever something interesting happens to the document or browser or to some element or object associated with it.
- For example, the web browser generates an event when it finishes loading a document, when the user moves the mouse over a hyperlink, or when the user strikes a key on the keyboard.
- The **event type** is a string that specifies what kind of event occurred. The type "mouse-move", for example, means that the user moved the mouse. The type "keydown" means that a key on the keyboard was pushed down. And the type "load" means that a document (or some other resource) has finished loading from the network.
- The **event target** is the object on which the event occurred or with which the event is associated. When we speak of an event, we must specify both the type and the target. A load event on a Window, for example, or a click event on a <button> Element.

# Event Handling

Type	Description
click	A higher-level event fired when the user presses and releases a mouse button or otherwise “activates” an element.
contextmenu	A cancelable event fired when a contextmenu is about to be popped up. Current browsers display context menus on right mouse clicks, so this event can also be used like the click event.
dblclick	Fired when the user double-clicks the mouse
mousedown	Fired when the user presses a mouse button
mouseup	Fired when the user releases a mouse button
mousemove	Fired when the user moves the mouse.
mouseover	Fired when the mouse enters an element. <code>relatedTarget</code> (or <code>fromElement</code> in IE); specifies what element the mouse is coming from.
mouseout	Fired when the mouse leaves an element. <code>relatedTarget</code> (or <code>toElement</code> in IE); specifies what element the mouse is going to.
mouseenter	Like “mouseover”, but does not bubble. Introduced by IE and standardized in HTML5 but not yet widely implemented.
mouseleave	Like “mouseout”, but does not bubble. Introduced by IE and standardized in HTML5 but not yet widely implemented.

[MouseEvent.clientX](#)




Read only

The X coordinate of the mouse pointer in local (DOM content) coordinates.

[MouseEvent.clientY](#)

Read only

The Y coordinate of the mouse pointer in local (DOM content) coordinates.

- [KeyboardEvent](#)
- [MediaStreamEvent](#) 
- [MessageEvent](#)
- [MouseEvent](#)
- [MutationEvent](#) 
- [OfflineAudioCompletionEvent](#)
- [PageTransitionEvent](#)
- [PaymentRequestUpdateEvent](#)
- [PointerEvent](#)
- [PopStateEvent](#)
- [ProgressEvent](#)
- [RTCDataChannelEvent](#)
- [RTCPeerConnectionIceEvent](#)
- [StorageEvent](#)
- [SubmitEvent](#)
- [SVGEvent](#) 
- [TimeEvent](#)
- [TouchEvent](#)
- [TrackEvent](#)

<https://developer.mozilla.org/en-US/docs/web/api/event>

# Example of Event Handling

- Handling a click event

## HTML

```
<button>Change color</button>
```

## JS

```
const btn = document.querySelector("button");

function random(number) {
  return Math.floor(Math.random() * (number + 1));
}

btn.addEventListener("click", () => {
  const rndCol = `rgb(${random(255)}, ${random(255)}, ${random(255)})`;
  document.body.style.backgroundColor = rndCol;
});
```

# jQuery

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6
<script>
$(document).ready(function(){
  $("p").click(function(){
    $(this).hide();
  });
});
</script>
</head>
<body>

<p>If you click on me, I will disappear.</p>
<p>Click me away!</p>
<p>Click me too!</p>

</body>
</html>
```

- JavaScript has an intentionally simple core API and an overly complicated client-side API that is marred by major incompatibilities between browsers.
- jQuery makes it easy to find the elements of a document that you care about and then manipulate those elements by adding content, editing HTML attributes and CSS properties, defining event handlers, and performing animations.

# AJAX

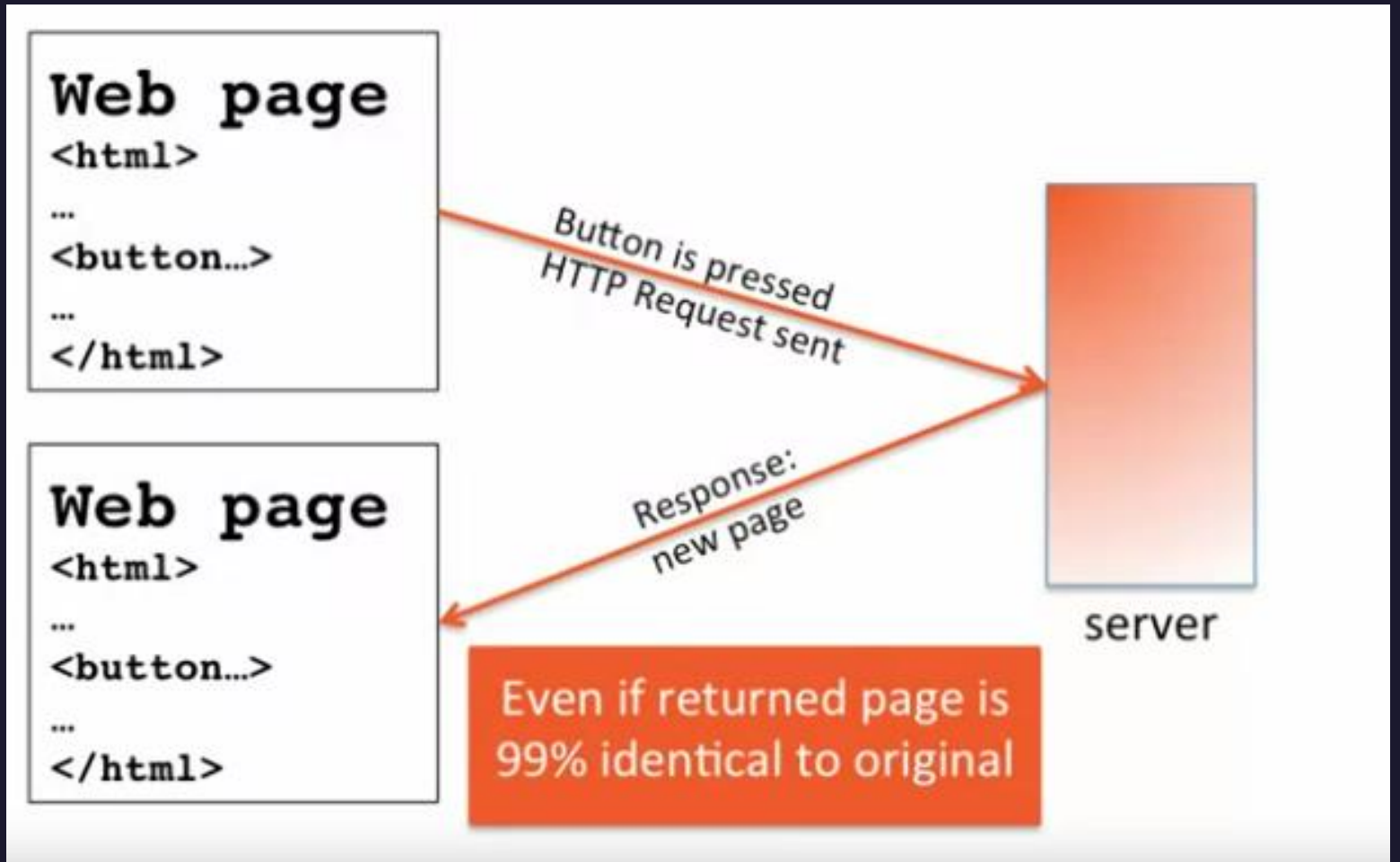
stands for

## Asynchronous Javascript And XML

- ✧ While Ajax started with XML, very few apps use it nowadays
  - Plain text (at times as html) and JSON is used instead

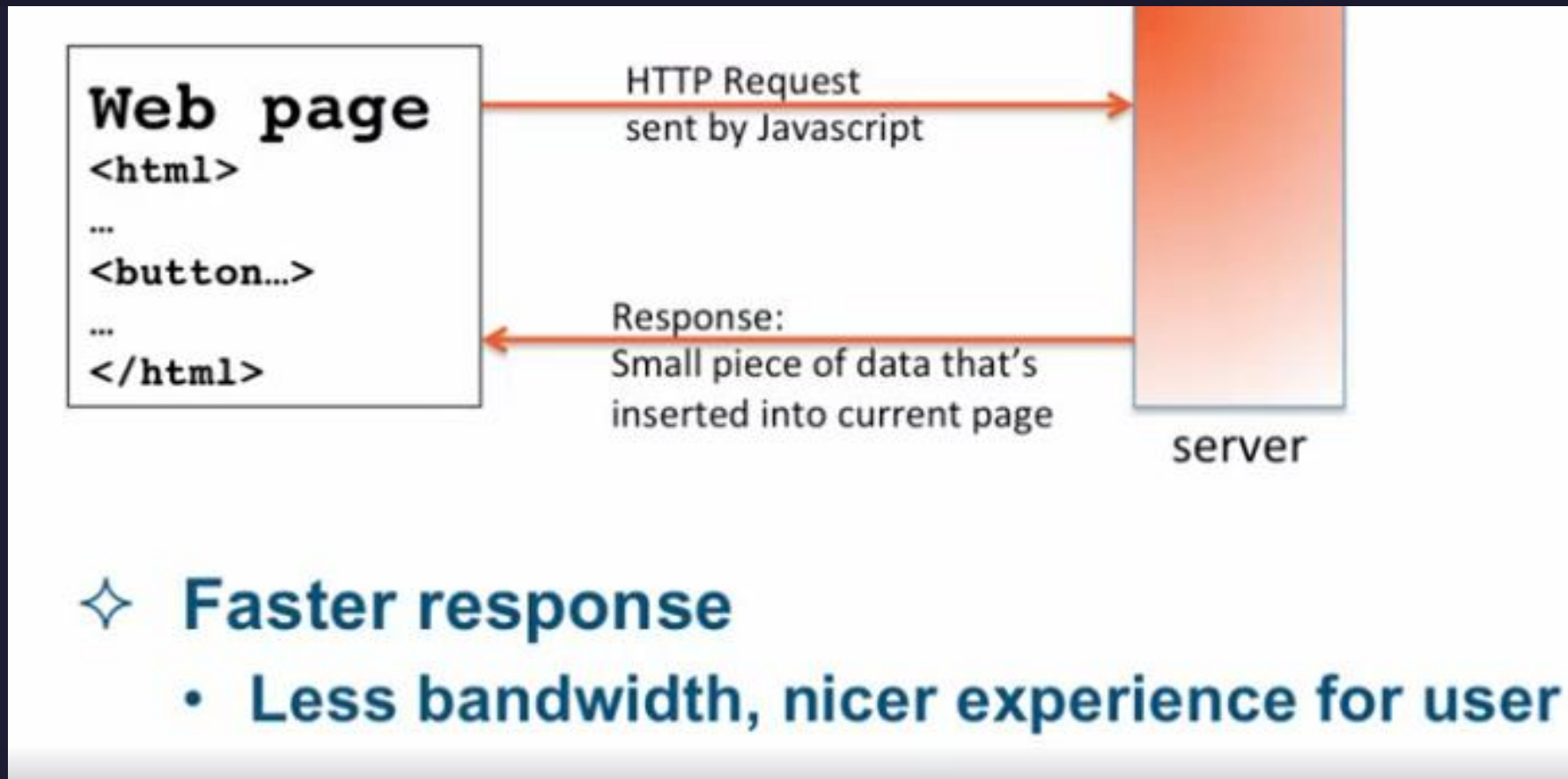
- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background

# Traditional Web app flow





# AJAX web app flow



# Synchronous Vs Asynchronous Execution

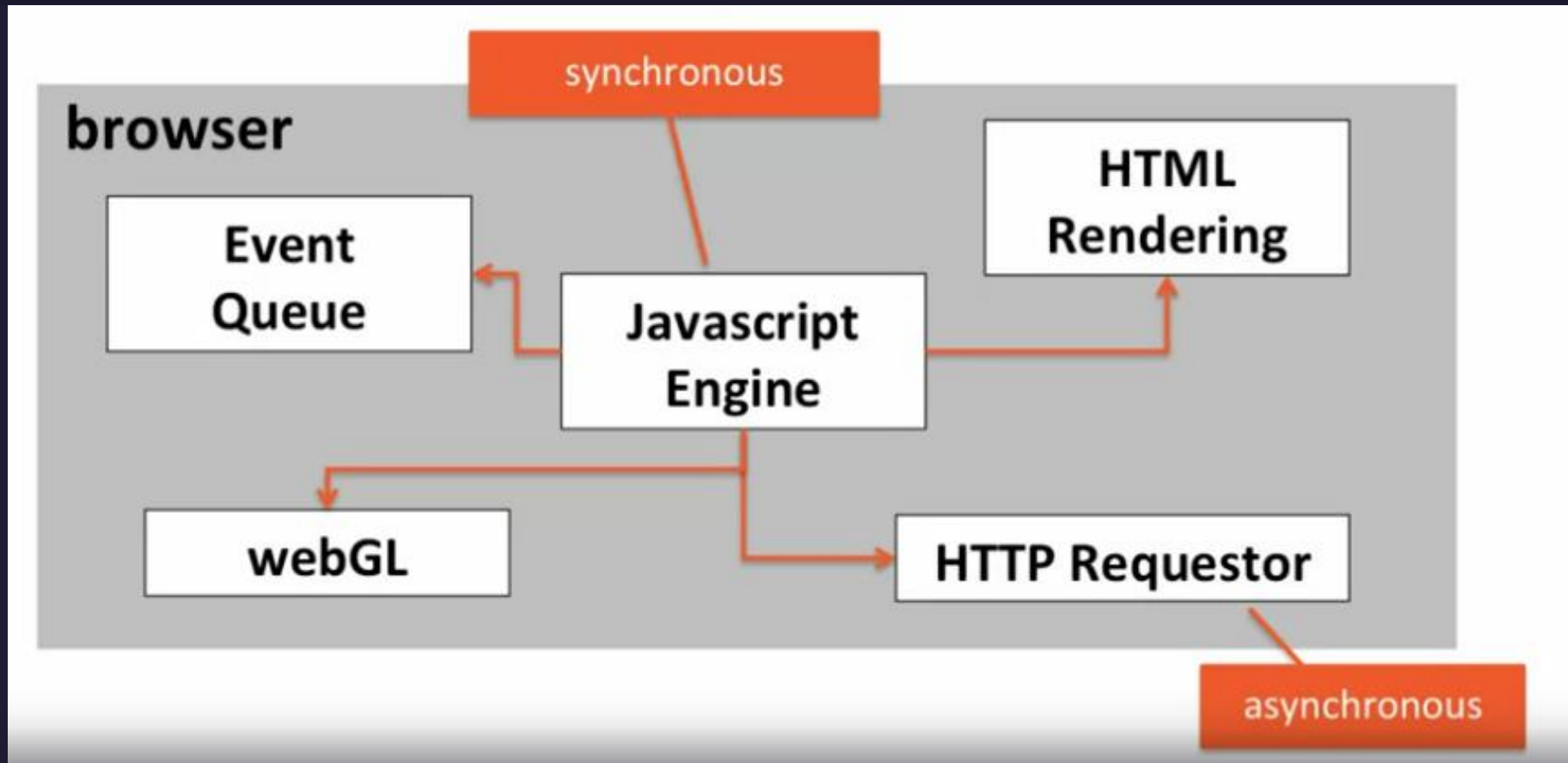
## Synchronous

- Execution of one instruction at a time.
- Can't start the execution of another instruction until previous one has completed execution.

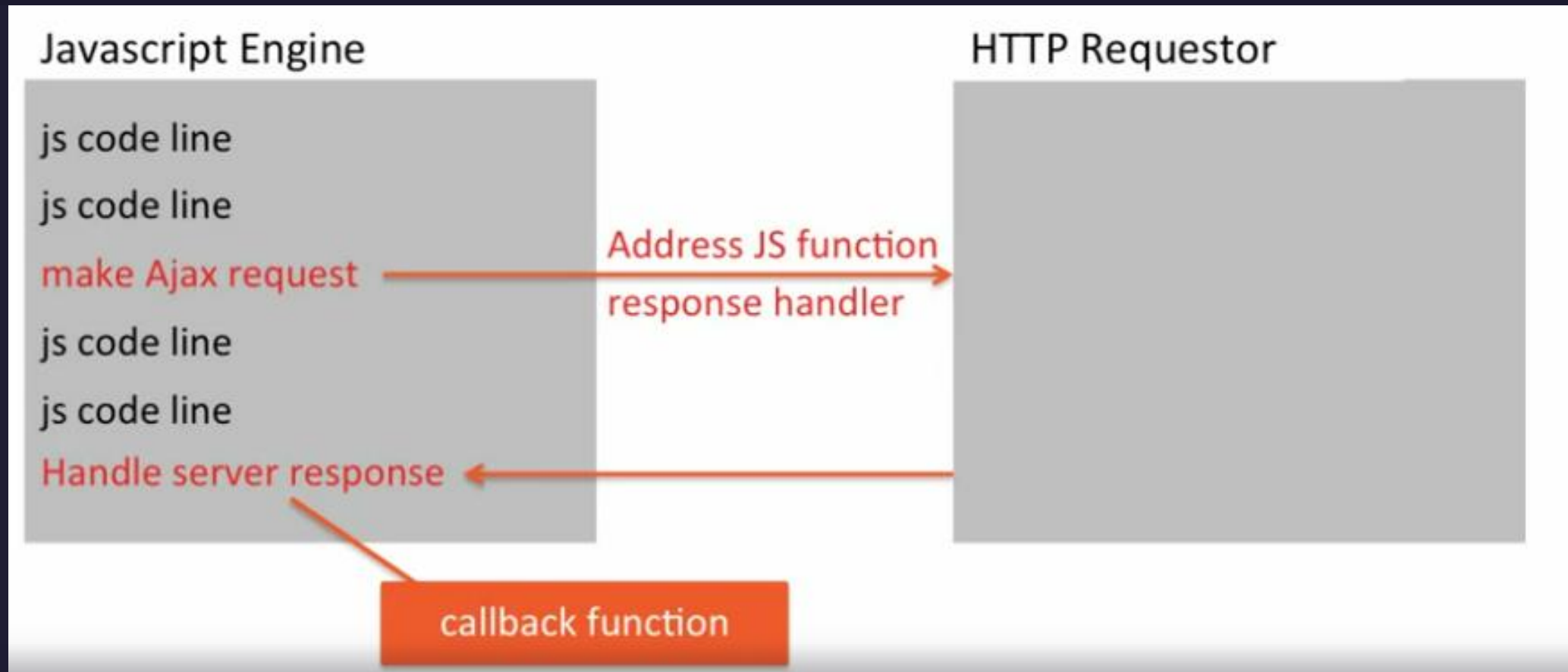
## Asynchronous

- Execution of more than one instruction at a time.
- It returns right away.
- The actual execution is done in a separate thread or process.

# How does Ajax work?



# Ajax Process



```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
<h1>The XMLHttpRequest Object</h1>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
        this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```

## The XMLHttpRequest Object

Change Content

## AJAX

AJAX is not a programming language.

AJAX is a technique for accessing web servers from a web page.

AJAX stands for Asynchronous JavaScript And XML.

Thank you

