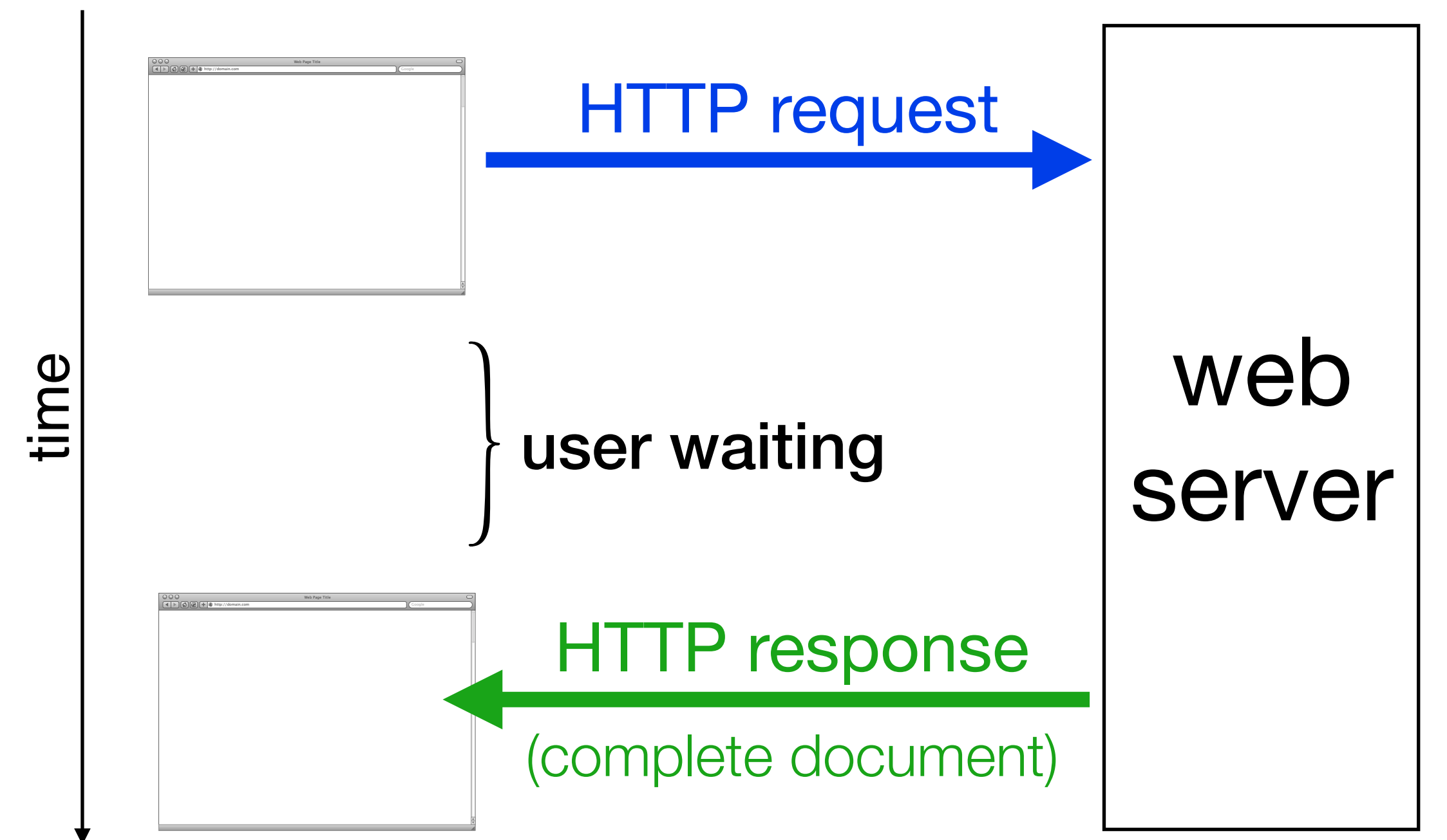


# Web Programming

## **AJAX**

# Traditional web interaction

- User requests a page = browser (client) sends HTTP request to server
- Browser is “blocked” from activity while it waits for the server to provide the document
- When the response arrives, the browser renders the document



*synchronous request-response communication*

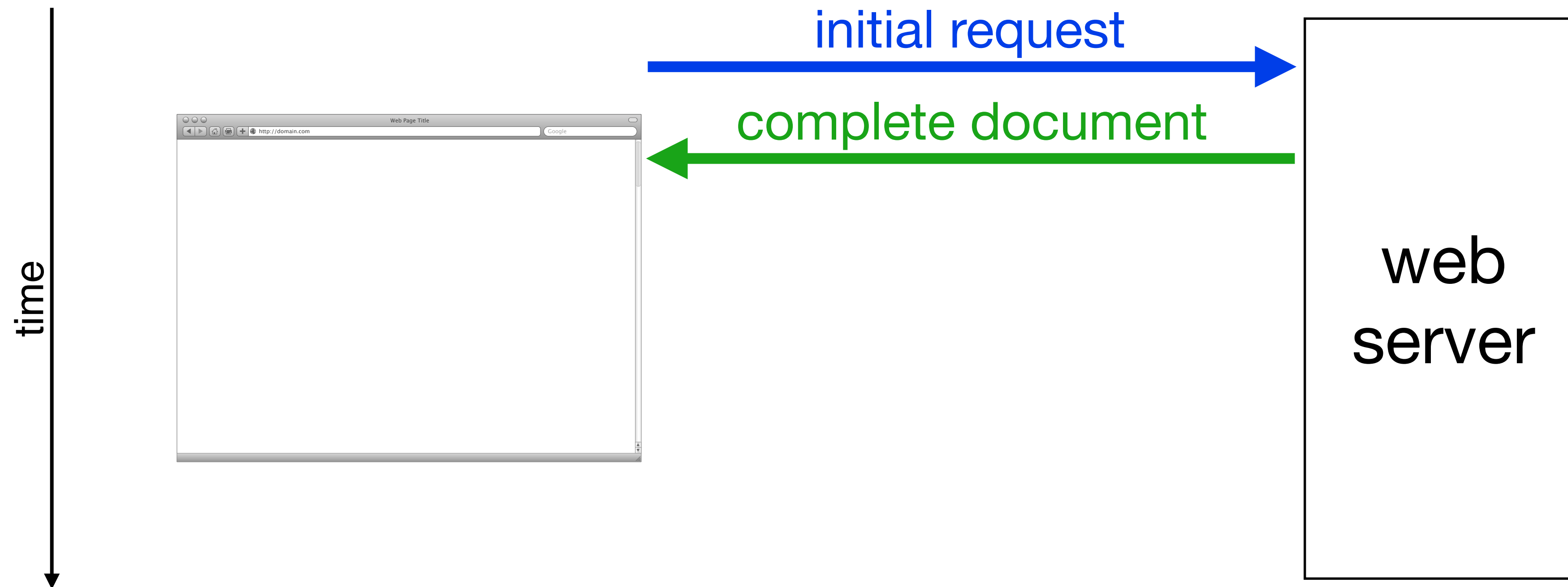
# Motivation

- Provide web-based applications with rich user interfaces and responsiveness
- This requires frequent interactions between the user and the server
  - Speed of interactions determines the usability of the application!
- Often, only (relatively small) parts of the documents are modified or updated. No need to reload the entire page
- Client might want to send data to the server in the background

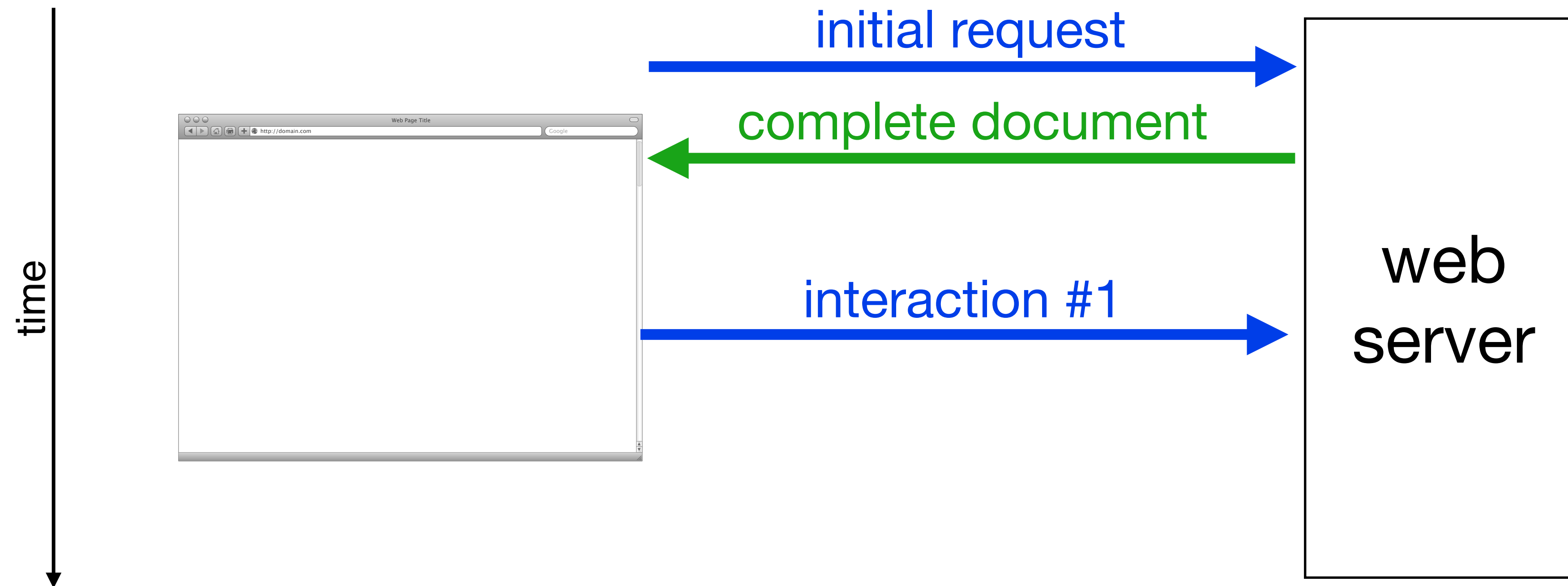
# AJAX

- **Asynchronous JavaScript and XML**
- **Combination of web technologies**
  - Client side: HTML, JavaScript
  - Server side: any programming language
  - Despite the name, XML is not required!
- **Two key features**
  - Update only parts of the page
  - Asynchronous, i.e., no need to "lock" the document while waiting for the response

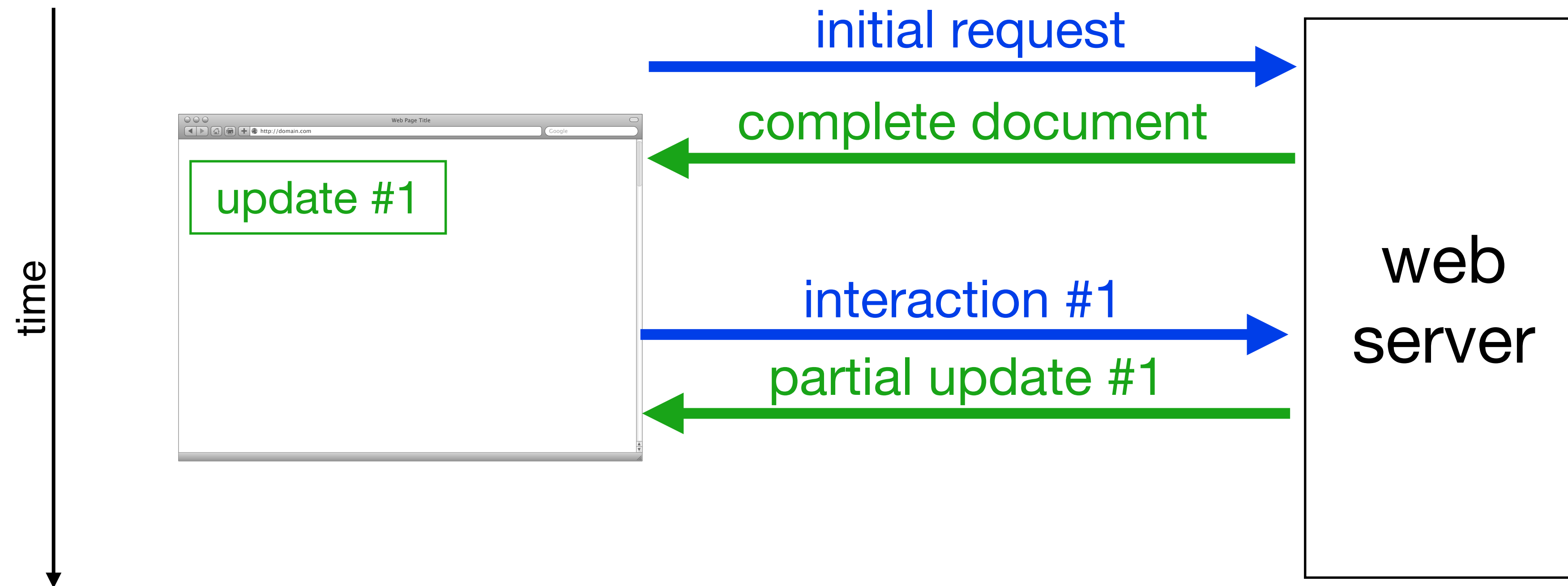
# AJAX interaction



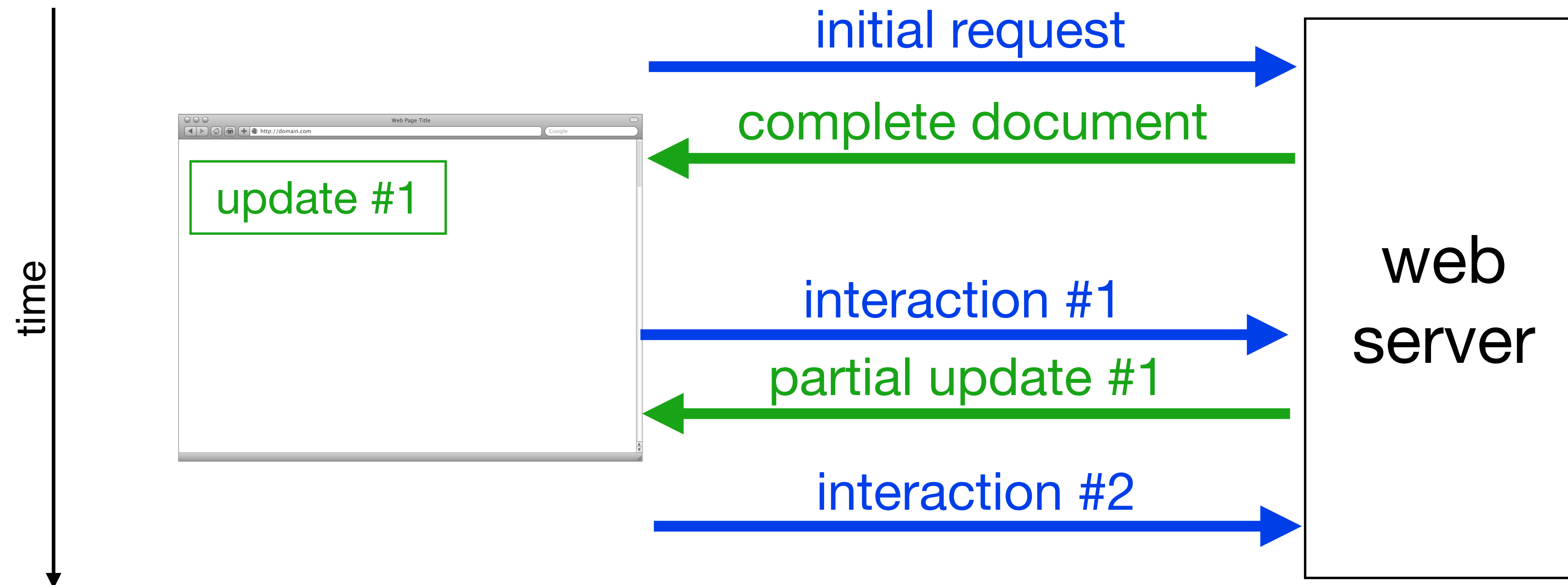
# AJAX interaction



# AJAX interaction

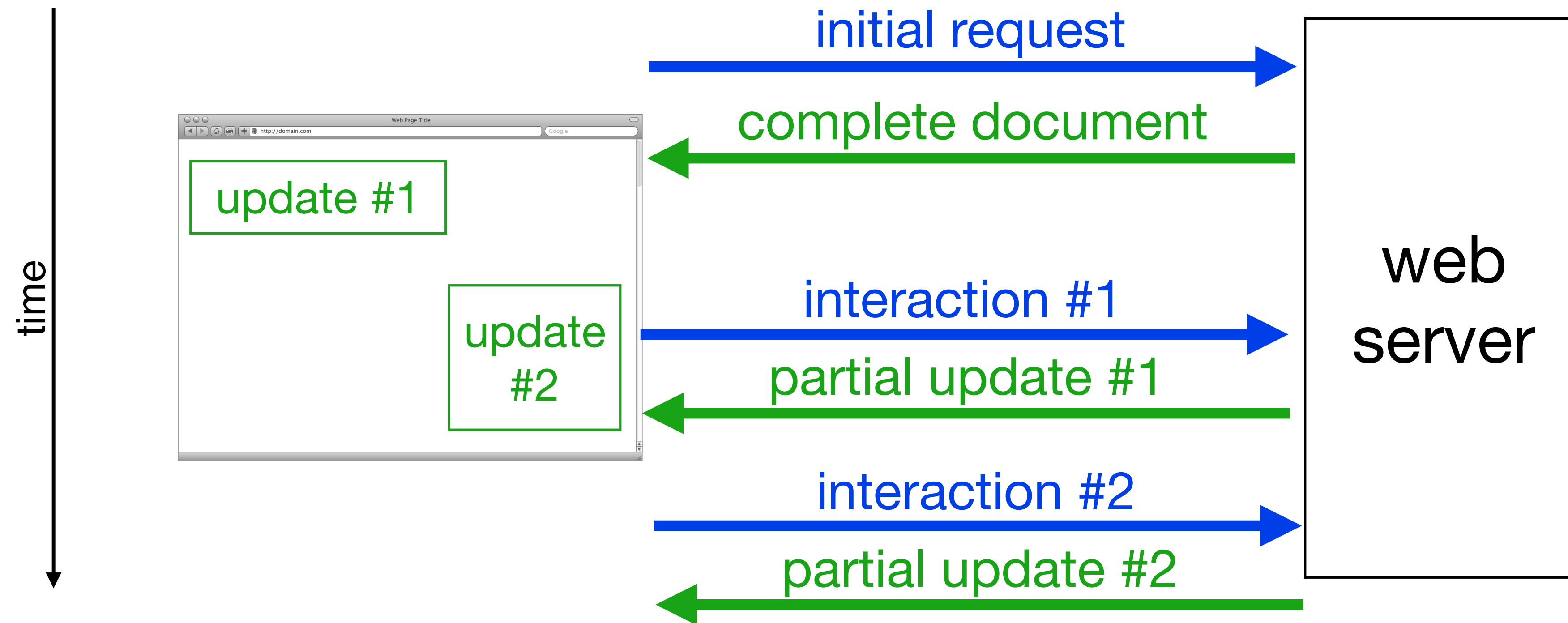


# AJAX interaction

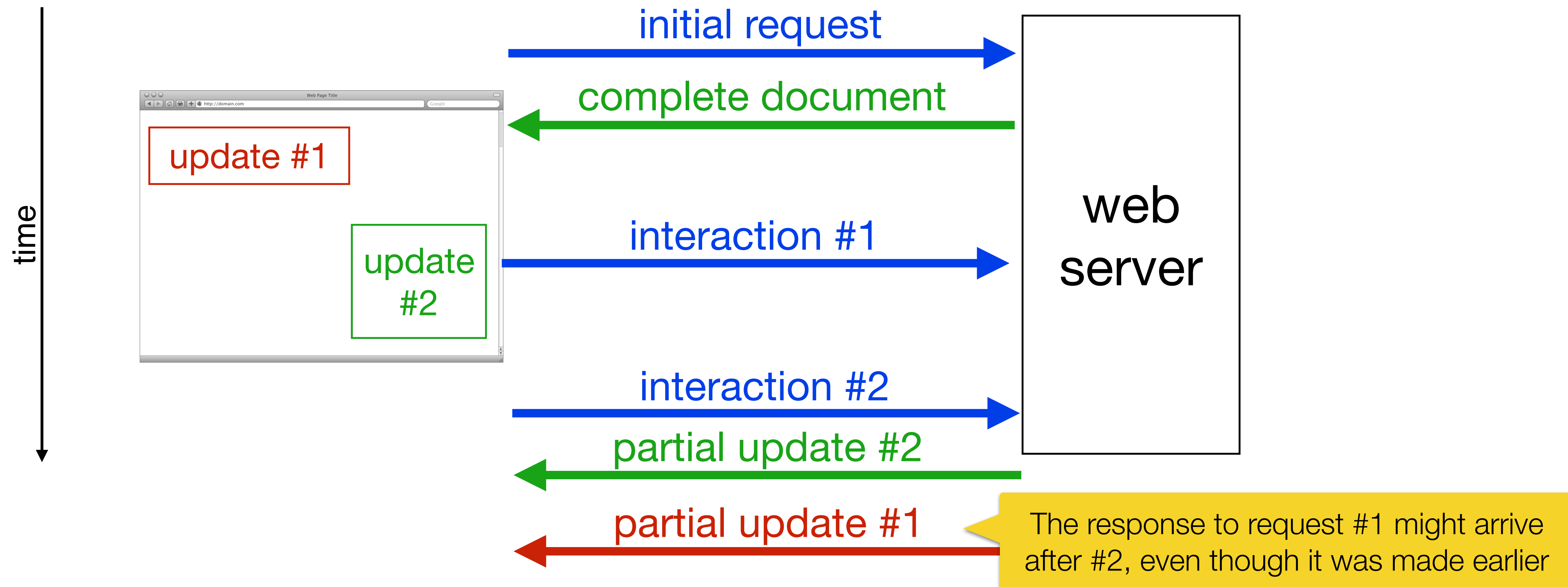




# AJAX interaction




# Note that responses are asynchronous



# **Where to use AJAX?**

# Where to use AJAX?



where to use ajax

where to use ajax


where to use ajax **in a website**


where to use ajax **in asp net**

where **not** to use ajax


Press Enter to search.


Klangnomad - I am what I am


 Klangnomad Music


 38,919



10,061,407 views



 Add to


 Share


 Report


 More


 65,995  2,211


  112

5 Comments 4 Shares 

 Like

 Comment

 Share

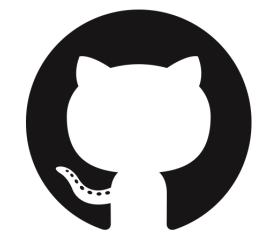
kbalog 

Username is already taken

# Four main parts

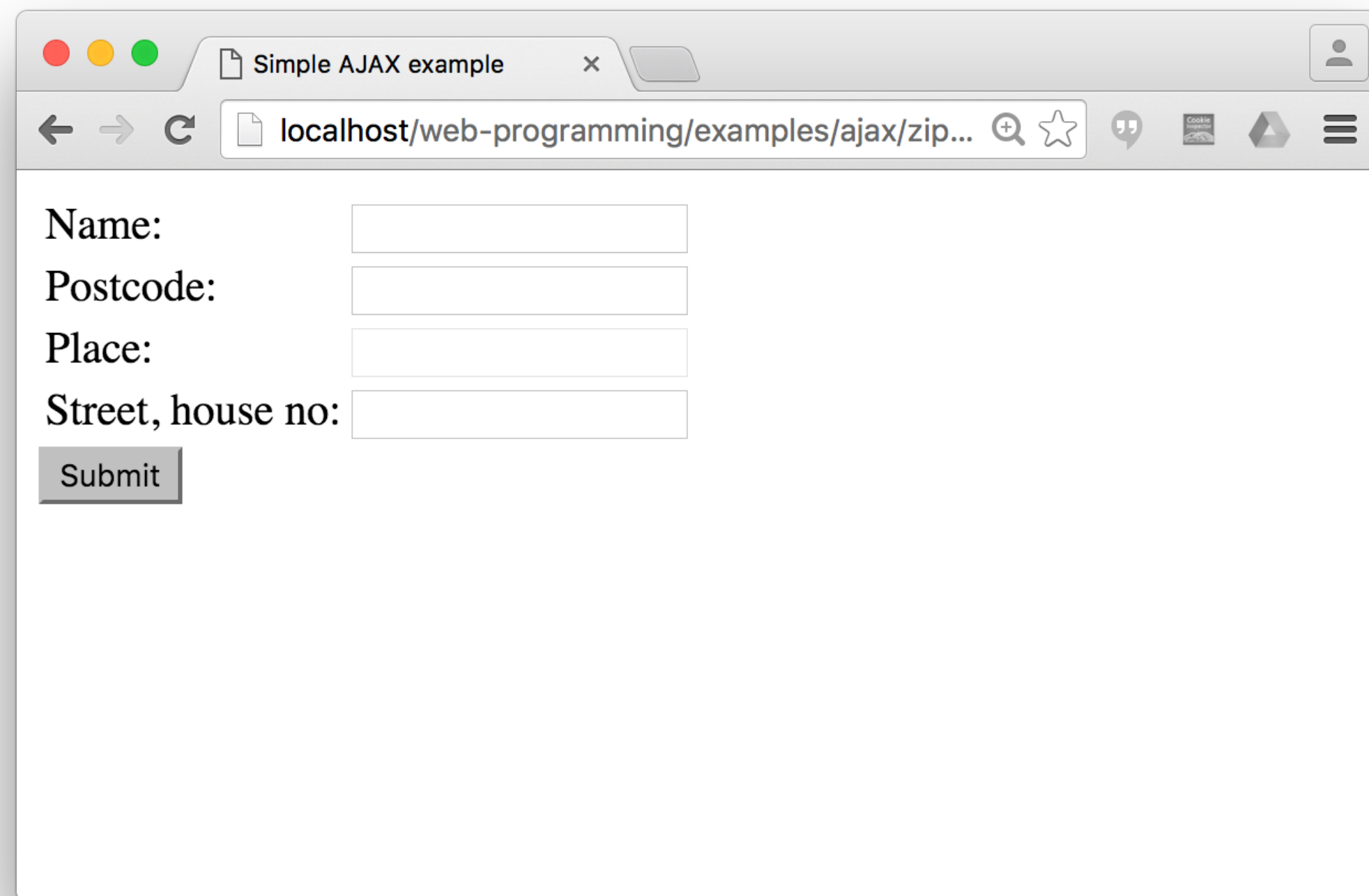
1. Initial HTML document (may be generated using Python)
2. JavaScript to send the AJAX request to the server
3. Server-side program to receive the request and produce the requested data
4. JavaScript to receive the new data and integrate it into the original document being displayed

# Example walkthrough



[https://github.com/kbalog/web-programming/tree/master/](https://github.com/kbalog/web-programming/tree/master/examples/ajax/zipcode)  
**examples/ajax/zipcode**

# Example



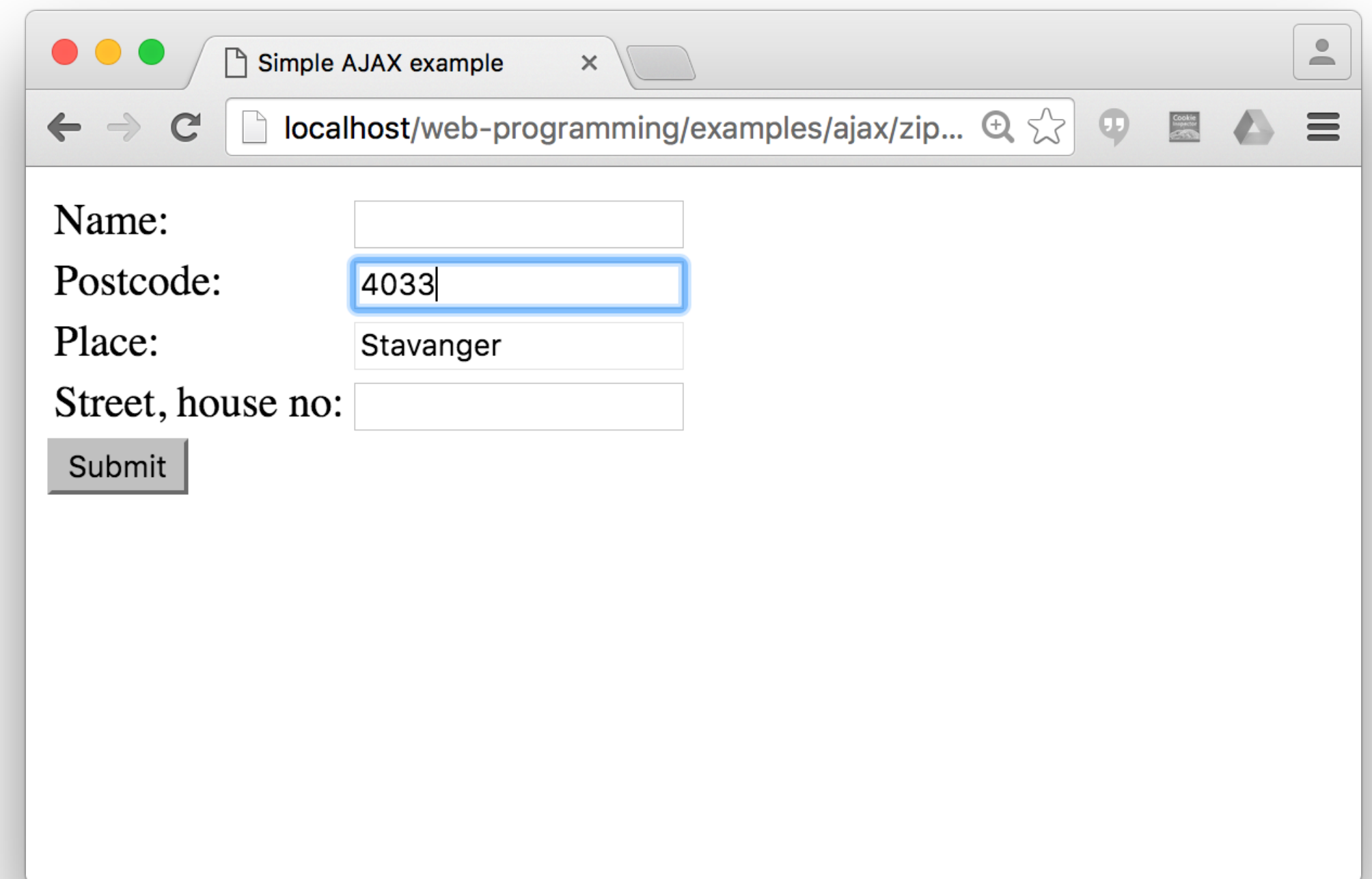
A screenshot of a web browser window titled "Simple AJAX example". The address bar shows the URL "localhost/web-programming/examples/ajax/zip...". The form contains four input fields: "Name:", "Postcode:", "Place:", and "Street, house no:". A "Submit" button is located below the "Street, house no:" field. All input fields are empty.

Name:

Postcode:

Place:

Street, house no:



A screenshot of a web browser window titled "Simple AJAX example". The address bar shows the URL "localhost/web-programming/examples/ajax/zip...". The form contains four input fields: "Name:", "Postcode:", "Place:", and "Street, house no:". A "Submit" button is located below the "Street, house no:" field. The "Postcode:" field is pre-filled with "4033" and has a blue border. The "Place:" field is pre-filled with "Stavanger".

Name:

Postcode:

Place:

Street, house no:

# 1. Initial HTML document

- Register JavaScript handler function on onkeyup event
  - I.e., whenever the user presses a key

zipcode.html

```
<input type="text" name="postcode" onkeyup="getPlace(this.value);"/>
```



## 2. Request phase

- Register callback function
- Make asynchronous call

zipcode.js

```
function getPlace(postcode) {  
    var xhr = new XMLHttpRequest();  
    /* register an embedded function as the handler */  
    xhr.onreadystatechange = function () {  
        [...]  
    };  
    /* send the request using GET */  
    xhr.open("GET", "/getplace?postcode=" + postcode, true);  
    xhr.send(null);  
}
```

setting this parameter to **true** means making an asynchronous request

# 3. Response document

- Flask app generates simple text response

app.py

```
@app.route("/getplace", methods=["GET"])
def getplace():
    POSTCODES = {
        "0107": "Oslo",
        "0506": "Oslo",
        "4090": "Hafrsfjord",
        ...
    }
    postcode = request.args.get("postcode", None)
    # look up corresponding place or return empty string
    if postcode and (postcode in POSTCODES):
        return POSTCODES[postcode]
    return ""
```

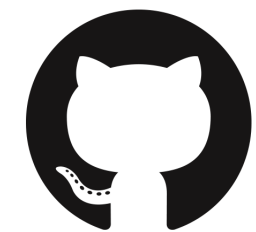
## 4. Receiver phase

- Callback is called multiple times, readyState indicates the progress (0..4)
- Status is 200 if the request was successfully completed

zipcode.js

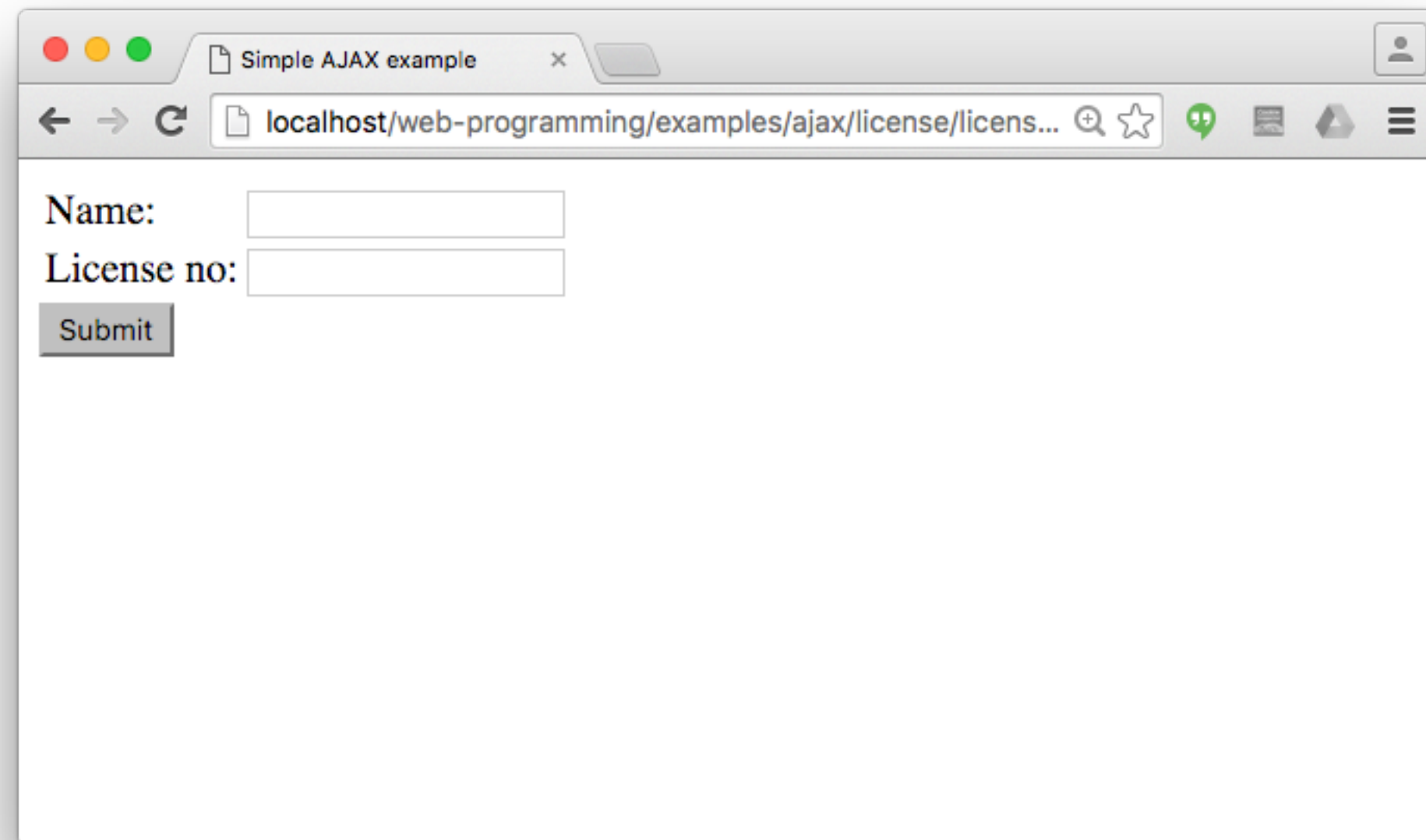
```
xhr.onreadystatechange = function () {  
    /* readyState = 4 means that the response has been completed  
     * status = 200 indicates that the request was successfully completed */  
    if (xhr.readyState == 4 && xhr.status == 200) {  
        var result = xhr.responseText;  
        document.getElementById("place").value = result;  
    }  
};
```

# Example walkthrough #2

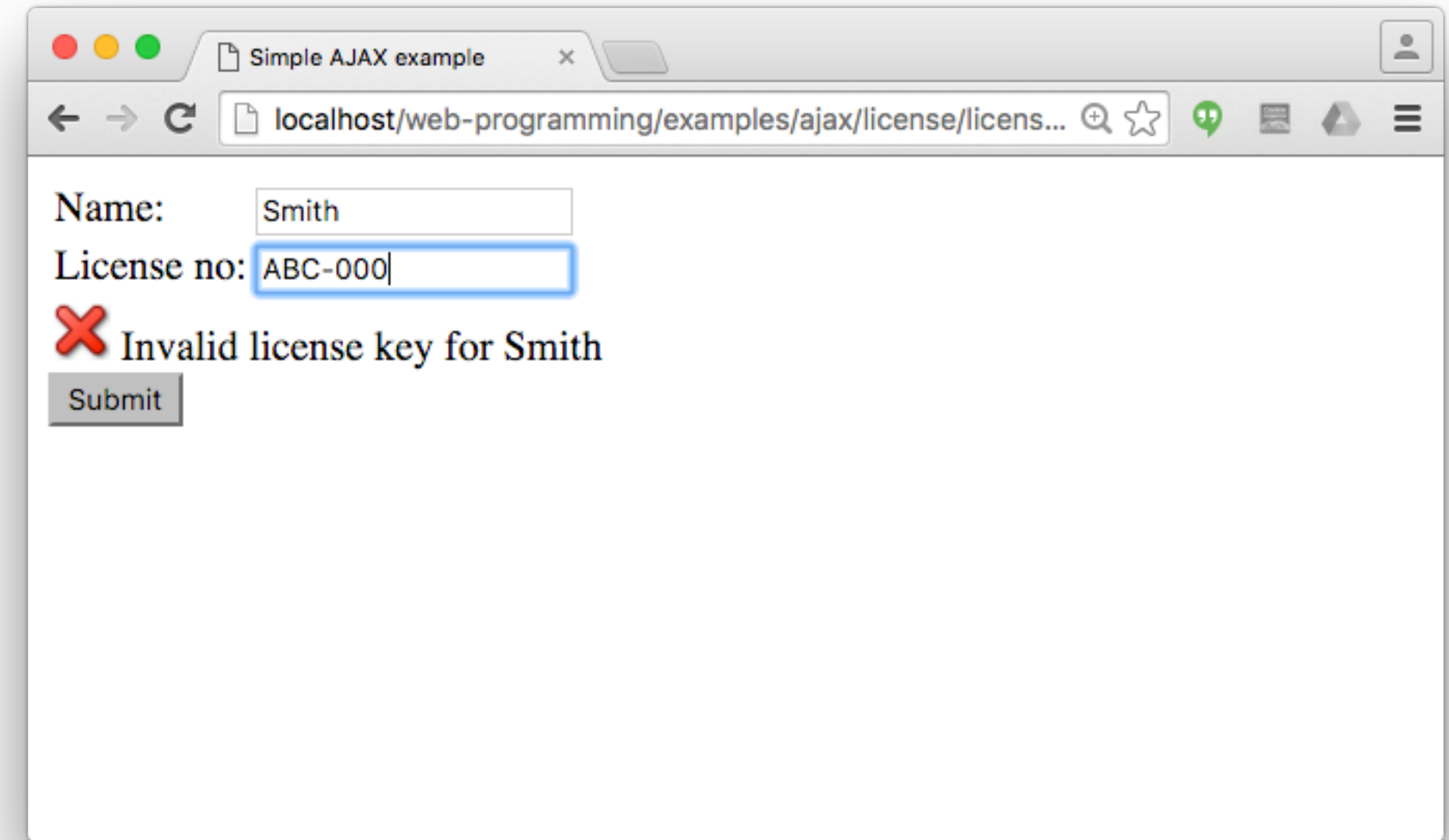


[https://github.com/kbalog/web-programming/tree/master/](https://github.com/kbalog/web-programming/tree/master/examples/ajax/license)  
**examples/ajax/license**

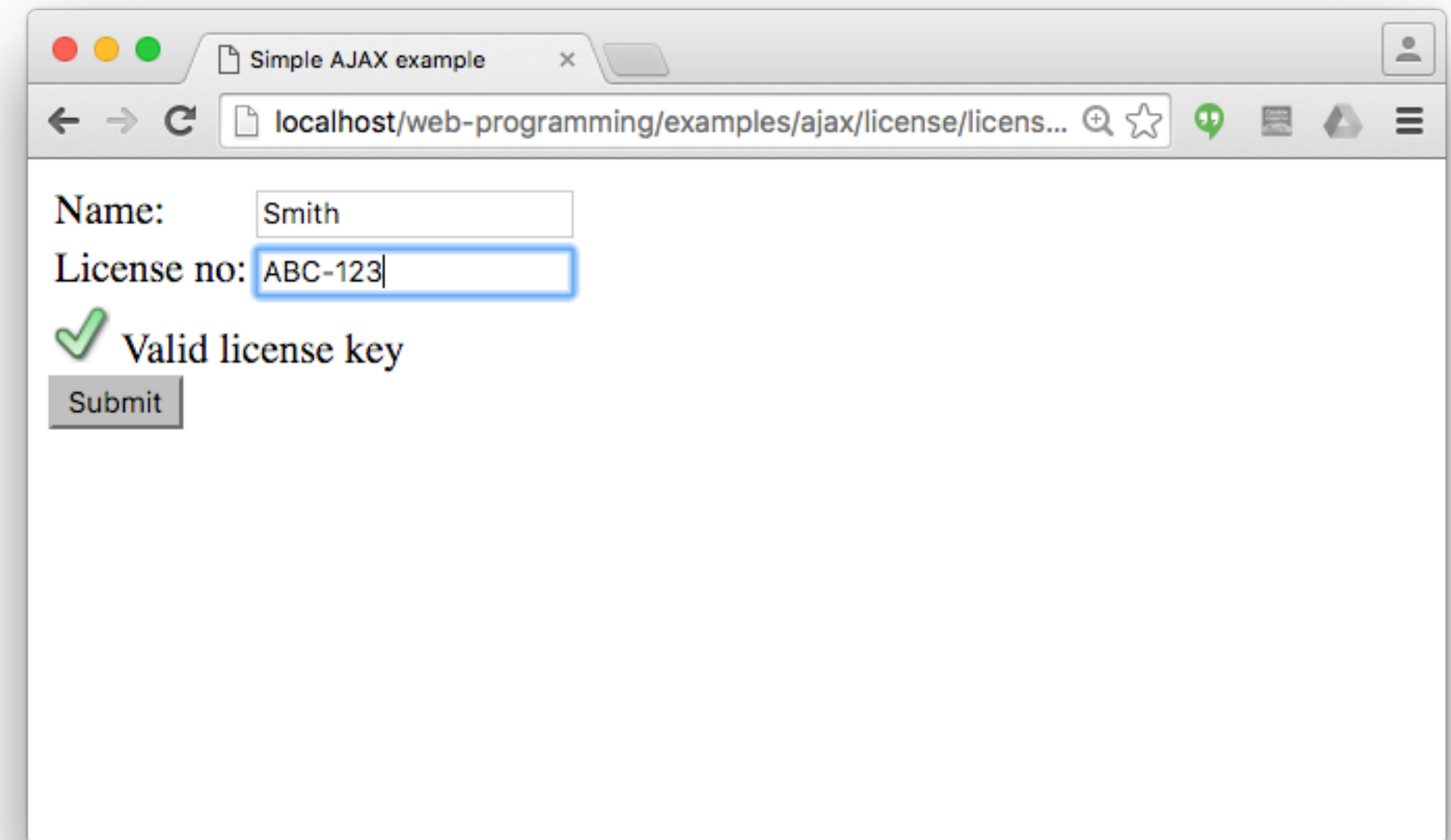
# Example #2



A screenshot of a web browser window titled "Simple AJAX example". The address bar shows the URL "localhost/web-programming/examples/ajax/license/licens...". The form contains two input fields: "Name:" and "License no:". Below the "License no:" field is a "Submit" button.



A screenshot of a web browser window titled "Simple AJAX example". The address bar shows the URL "localhost/web-programming/examples/ajax/license/licens...". The form contains two input fields: "Name:" with the value "Smith" and "License no:" with the value "ABC-000". Below the "License no:" field is a red "X" icon and the text "Invalid license key for Smith". Below this message is a "Submit" button.



A screenshot of a web browser window titled "Simple AJAX example". The address bar shows the URL "localhost/web-programming/examples/ajax/license/licens...". The form contains two input fields: "Name:" with the value "Smith" and "License no:" with the value "ABC-123". Below the "License no:" field is a green checkmark icon and the text "Valid license key". Below this message is a "Submit" button.

# Example #2

- Request can be POST as well
- It is also possible for the server to send back a HTML snippet
- The client updates part of the page (i.e., the DOM) with the received snippet

# 1. Initial HTML document

- Register JavaScript handler function on onkeyup events
  - I.e., whenever the user presses a key in the name or license fields

license.html

```
<input type="text" name="name" id="name" onkeyup="checkLicense();" />
```

```
<input type="text" name="license" id="license" onkeyup="checkLicense();" />
```

## 2. Request phase

- Make asynchronous call using POST
  - Need to add a HTTP header to make it as if it was a form submission

license.js

```
function checkLicense() {  
    [...]  
  
    /* send the request using POST */  
    xhr.open("POST", "/check_license", true);  
    /* To POST data like an HTML form, add an HTTP header */  
    xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  
    /* variables go in the request body */  
    xhr.send("name=" + name + "&license=" + license);  
  
    [...]  
}
```



# 3. Response document

- Flask app generates a HTML snippet

app.py

```
@app.route("/check_license", methods=["POST"])
def check_license():
    VALID_LICENSES = {...}
    name = request.form.get("name", None)
    license = request.form.get("license", None)
    # check if name and license match
    if name and license:
        if VALID_LICENSES.get(name, None) == license:
            return "<img src='/static/images/yes.png' /> Valid license key"
        else:
            return "<img src='/static/images/no.png' /> Invalid license key for {}".format(name)
    return ""
```

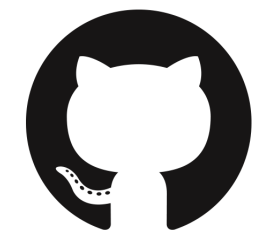
## 4. Receiver phase

- Callback is called multiple times, readyState indicates the progress (0..4)
- Status is 200 if the request was successfully completed

license.js

```
xhr.onreadystatechange = function () {  
    /* readyState = 4 means that the response has been completed  
     * status = 200 indicates that the request was successfully completed */  
    if (xhr.readyState == 4 && xhr.status == 200) {  
        var result = xhr.responseText;  
        document.getElementById("license_check").innerHTML = result;  
    }  
};
```

# Exercises #1, #1b



[https://github.com/kbalog/web-programming/tree/master/](https://github.com/kbalog/web-programming/tree/master/exercises/ajax)  
**exercises/ajax**

# What can be the response document?

- Data as a simple string
- HTML snippet
- Data as "object"
  - Both the client and the server need to speak the same language, i.e., how to *encode* and *decode* the object

# JSON

- JavaScript Object Notation
- Lightweight data-interchange format
- Language independent
- Two structures
  - Collection of name-value pairs (object)
    - a.k.a. record, struct, dictionary, hash table, associative array
  - Ordered list of values (array)
    - a.k.a. vector, list

# JSON

- Values can be
  - string (in between "...")
  - number
  - object
  - array
  - boolean (true/false)
  - null

# Example JSON

```
{  
  "name": "John Smith",  
  "age": 32,  
  "married": true,  
  "interests": [1, 2, 3],  
  "other": {  
    "city": "Stavanger",  
    "postcode": 4041  
  }  
}
```

# JSON with Python

🔗 [examples/ajax/json/json\\_python.php](https://www.w3schools.com/python/examples/ajax/json/json_python.php)

- **json** is a standard module
- **json.dumps(data)**
  - returns JSON representation of the data
- **json.loads(json\_value)**
  - decodes a JSON value
- **json.dumps()** and **json.loads()** work with strings
- **json.dump()** and **json.load()** work with file streams



# JSON with JavaScript

🔗 [examples/ajax/json/json\\_js.html](examples/ajax/json/json_js.html)

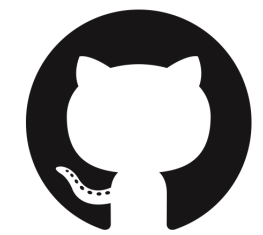
- **JSON.stringify(value)**

- returns JSON representation of a value (encode)

- **JSON.parse(json)**

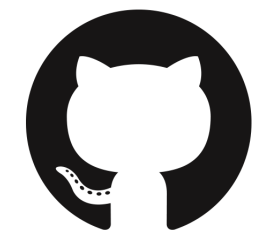
- parses a JSON value into a JavaScript object (decode)

# Exercise #2



[https://github.com/kbalog/web-programming/tree/master/](https://github.com/kbalog/web-programming/tree/master/exercises/ajax)  
**exercises/ajax**

# Example



[https://github.com/kbalog/web-programming/tree/master/](https://github.com/kbalog/web-programming/tree/master/examples/ajax/loading)  
**examples/ajax/loading**

# Indicating waiting


- An animated gif is displayed until the response arrives
- In this example there is an artificial delay of 1sec is added to the Python code

Password:

Submit



Password:



Password:

MD5: 5ebe2294ecd0e0f08eab7690d2a6ee69

Submit

# **AJAX in jQuery**

# AJAX controls

- **`$.ajax()`** — global function
- Shorthand AJAX methods: **`$.get()`** and **`$.post()`**
- **`load()`** method
  - replaces the HTML content of the matched elements with the content returned from a remote file
  - (does not work with form input fields!)
- Full reference:  
[http://www.w3schools.com/jquery/jquery\\_ref\\_ajax.asp](http://www.w3schools.com/jquery/jquery_ref_ajax.asp)

# \$.ajax()

- \$.ajax(*params*)
- where *params* is a settings map object

```
var params = {  
  type: "GET",  
  url: "requestUrl",  
  dataType: "text", // html, xml, json  
  success: successCallbackFunction,  
  error: errorCallbackFunction  
};
```

# **\$.get(), \$.post()**

- Full syntax:
  - **\$.get(*url*, *data*, *function(data, status, xhr)*, *dataType*)**
  - **\$.post(*url*, *data*, *function(data, status, xhr)*, *dataType*)**
- Where:
  - ***url*** where the request is sent
  - ***data*** (optional) data to be sent (map with variables and values)
  - ***function(...)*** callback function to run if the request succeeds
  - ***dataType*** (optional) data type setting (xml, html, text, ...)

```
$.post("ajax.php", {"var1":"value"}, function (data) {  
    $("#bar").html(data);  
});
```



# Example (zipcode) using JavaScript

🔗 [examples/ajax/zipcode/](#)

```
function getPlace(postcode) {  
    var xhr = new XMLHttpRequest();  
    /* register an embedded function as the handler */  
    xhr.onreadystatechange = function () {  
        /* readyState = 4 means that the response has been completed  
        * status = 200 indicates that the request was successfully completed */  
        if (xhr.readyState == 4 && xhr.status == 200) {  
            var result = xhr.responseText;  
            document.getElementById("place").value = result;  
        }  
    };  
    /* send the request using GET */  
    xhr.open("GET", "/getplace?postcode=" + postcode, true);  
    xhr.send(null);  
}
```

```
<input type="text" name="postcode" onkeyup="getPlace(this.value);"/>
```

# Example (zipcode) using jQuery

🔗 [examples/jquery/zipcode2/](#)

```
$(document).ready(function() {  
    $("input[name=postcode]").blur(function() {  
        $.get("/getplace", {postcode: $(this).val()}, function (data) {  
            $("#place").val(data);  
        });  
    });  
});
```

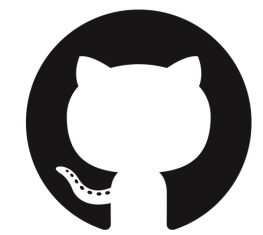
```
<input type="text" name="postcode"/>
```

# load()

- Loads data from a server and puts the returned data into the selected element
  - `$load(url, data, function(data, status, xhr))`
- Where:
  - ***url*** where the request is sent
  - ***data*** (optional) data to be sent to the server along with the request
  - ***function(...)*** (optional) callback function to run when the load() method is completed

```
$("button").click(function(){  
    $("#div1").load("demo_test.txt");  
});
```

# Exercises #3, #4



[https://github.com/kbalog/web-programming/tree/master/](https://github.com/kbalog/web-programming/tree/master/exercises/ajax)  
**exercises/ajax**