Assignment-1

1) Difference between HTTP1.1 vs 2

|  |  |
| --- | --- |
| HTTP1.1 | HTTP2 |
| keeps all requests and responses in plain text format | uses the binary framing layer to encapsulate all messages in binary format, while still maintaining HTTP semantics, such as verbs, methods, and headers |
| Has *head-of-line (HOL) blocking*, problem happens when multiple data packets cannot pass each other when traveling to the same destination, there are situations in which a request at the head of the queue that cannot retrieve its required resource will block all the requests behind it | Has Multiplexing which resolves the head-of-line blocking issue in HTTP/1.1 by ensuring that no message has to wait for another to finish. This results in better network and bandwidth utilization and thus decreases the overall operational cost. |
| HTTP1 has no weight assignment as there are no streams | By providing this facility of weight assignment, HTTP/2 enables developers to gain better control over web page rendering. The protocol also allows the client to change dependencies and reallocate weights at runtime in response to user interaction |
| To deal with with buffer overflow HTTP1 relies on TCP connection. Because HTTP/1.1 relies on the transport layer to avoid buffer overflow, each new TCP connection requires a separate flow control mechanism | HTTP/2 solves buffer overflow by allowing the client and server to implement their own flow controls, rather than relying on the transport layer. |
| In HTTP compression is allowed but header must not be compressed this will be problem when using cookies and resource intensive websites. | HTTP2 uses HPACK compression, this algorithm can encode the header metadata using Huffman coding, thereby greatly decreasing its size. Additionally, HPACK can keep track of previously conveyed metadata fields and further compress them according to a dynamically altered index shared between the client and the server |

2)Use typeof in all the datatypes and check the result

typeof(1) - number

typeof(1.1) - number

typeof("1.1") - string

typeof(true) - Boolean

typeof(null) - object

typeof(undefined) - undefined

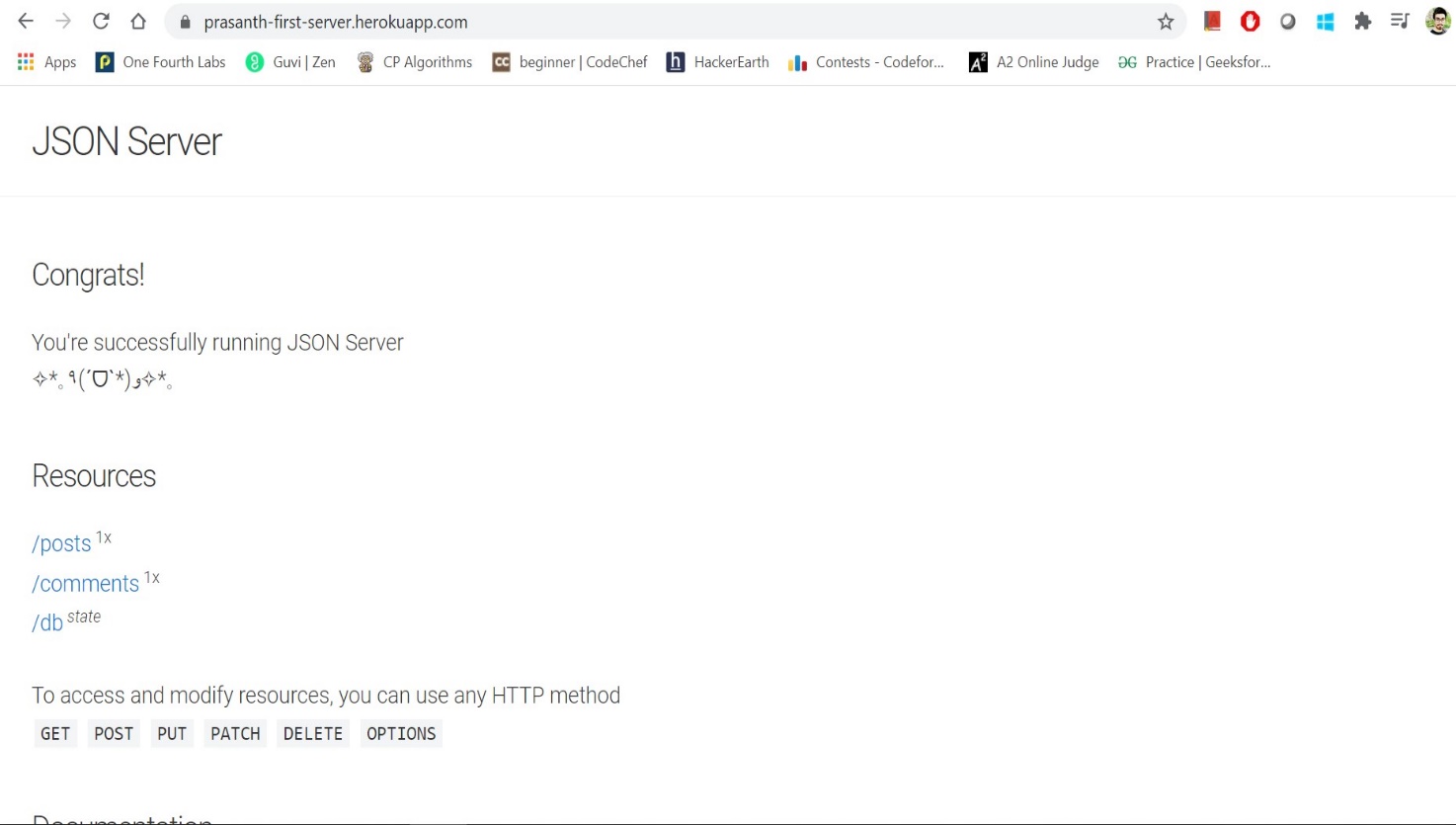
typeof([]) -object

typeof({}) – object

3) List 5 difference between Browser JS(console) v Nodejs

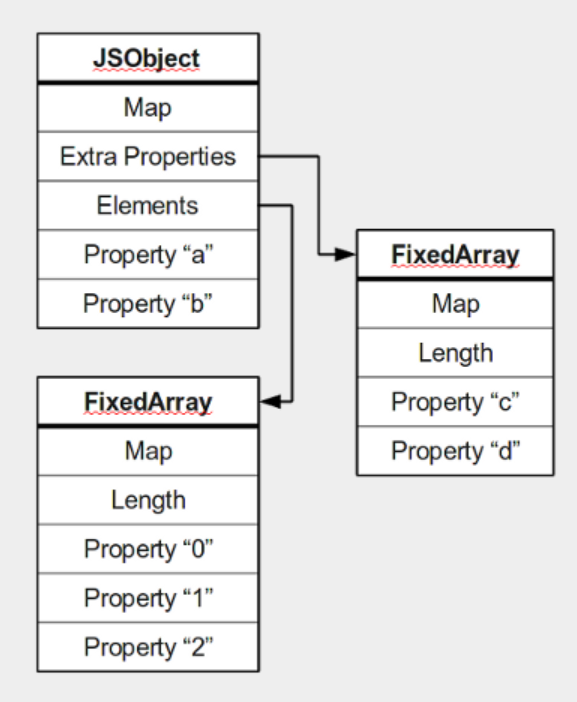
|  |  |
| --- | --- |
| Node | Browser |
| server side – here javascript can run anything OS offers | client side - Here JS is bounded by the Browser |
| Node doesn't have a predefined "window" object because it doesn't have a window to draw anything. | "window" is a predefined global object in browser which has functions and attributes, that must deal with window that has been drawn. |
| "location" object is related to a particular url, that means it is for page specific. So, node doesn't require that. | "location" is another predefined object in browsers, that has all the information about the url we have loaded. |
| In Node everything is a module. You must keep your code inside a module. | Moduling is not mandatory in client-side JavaScript, i.e. in browsers. |
| Node processes request object. | Browsers processes response objects. |
| Node has "global", which is a predefined global object. It contains several functions that are not available in browsers, because they are needed for server side works only. | Browsers may have an object named "global", but it will be the exact one as "window". |

4) Host JSON server in Heroku



5) Write a blog about objects and its internal representation in Javascript

A simple diagram is probably the best way to give a quick overview of the object representation in Javascript.



Most objects contain all their properties in a single block of memory (“a”, and “b”). All blocks of memory have a pointer to a map, which describes their structure.

Named properties that don’t fit in an object are usually stored in an overflow array (“c”, and “d”).

Numbered properties are stored separately, usually in a contiguous array.

The JavaScript standard allows developers to define objects in a very flexible way, and it is hard to come up with an efficient representation that works for everything. An object is essentially a collection of properties: basically key-value pairs. You can access properties using two different kinds of expressions:

🡺obj.prop

🡺obj[“prop”]

According to the spec, property names are always strings. If you use a name which is not a string, it is implicitly converted to a string. This may be a little surprising: if you use a number as a property name, it gets converted to a string as well (at least according to the spec). Because of this, you can store values at negative or fractional array indices. So a JavaScript object is basically a map from strings to values.

6) All tags in HTML Cheatsheet in your html with explanation

### Basic HTML

|  |  |
| --- | --- |
| Tag | Description |
| <!DOCTYPE> | Defines the document type |
| <html> | Defines an HTML document |
| <title> | Defines a title for the document |
| <body> | Defines the document’s body |
| <h1> to <h6> | Defines HTML headings |
| <p> | Defines a paragraph |
| <br> | Inserts a single line break |
| <hr> | Defines a thematic change in the content |
| <!–…–> | Defines a comment |

Formatting

|  |  |
| --- | --- |
| Tag | Description |
| <acronym> | Not supported in HTML5. Use <abbr> instead. |
|  | Defines an acronym |
| <abbr> | Defines an abbreviation or an acronym |
| <address> | Defines contact information for the author/owner of a document/article |
| <b> | Defines bold text |
| <bdi> | Isolates a part of text that might be formatted in a different direction from other text outside it |
| <bdo> | Overrides the current text direction |
| <big> | Not supported in HTML5. Use CSS instead. |
|  | Defines big text |
| <blockquote> | Defines a section that is quoted from another source |
| <center> | Not supported in HTML5. Use CSS instead. |
|  | Defines centered text |
| <cite> | Defines the title of a work |
| <code> | Defines a piece of computer code |
| <del> | Defines text that has been deleted from a document |
| <dfn> | Represents the defining instance of a term |
| <em> | Defines emphasized text |
| <font> | Not supported in HTML5. Use CSS instead. |
|  | Defines font, color, and size for text |
| <i> | Defines a part of text in an alternate voice or mood |
| <ins> | Defines a text that has been inserted into a document |
| <kbd> | Defines keyboard input |
| <mark> | Defines marked/highlighted text |
| <meter> | Defines a scalar measurement within a known range (a gauge) |
| <pre> | Defines preformatted text |
| <progress> | Represents the progress of a task |
| <q> | Defines a short quotation |
| <rp> | Defines what to show in browsers that do not support ruby annotations |
| <rt> | Defines an explanation/pronunciation of characters (for East Asian typography) |
| <ruby> | Defines a ruby annotation (for East Asian typography) |
| <s> | Defines text that is no longer correct |
| <samp> | Defines sample output from a computer program |
| <small> | Defines smaller text |
| <strike> | Not supported in HTML5. Use <del> or <s> instead. |
|  | Defines strikethrough text |
| <strong> | Defines important text |
| <sub> | Defines subscripted text |
| <sup> | Defines superscripted text |
| <time> | Defines a date/time |
| <tt> | Not supported in HTML5. Use CSS instead. |
|  | Defines teletype text |
| <u> | Defines text that should be stylistically different from normal text |
| <var> | Defines a variable |
| <wbr> | Defines a possible line-break |

Forms and Input

|  |  |
| --- | --- |
| Tag | Description |
| <form> | Defines an HTML form for user input |
| <input> | Defines an input control |
| <textarea> | Defines a multiline input control (text area) |
| <button> | Defines a clickable button |
| <select> | Defines a drop-down list |
| <optgroup> | Defines a group of related options in a drop-down list |
| <option> | Defines an option in a drop-down list |
| <label> | Defines a label for an <input> element |
| <fieldset> | Groups related elements in a form |
| <legend> | Defines a caption for a <fieldset> element |
| <datalist> | Specifies a list of pre-defined options for input controls |
| <keygen> | Defines a key-pair generator field (for forms) |
| <output> | Defines the result of a calculation |

Frames

|  |  |
| --- | --- |
| Tag | Description |
| <frame> | Not supported in HTML5. |
|  | Defines a window (a frame) in a frameset |
| <frameset> | Not supported in HTML5. |
|  | Defines a set of frames |
| <noframes> | Not supported in HTML5. |
|  | Defines an alternate content for users that do not support frames |
| <iframe> | Defines an inline frame |

Images

|  |  |
| --- | --- |
| Tag | Description |
| <img> | Defines an image |
| <map> | Defines a client-side image-map |
| <area> | Defines an area inside an image-map |
| <canvas> | Used to draw graphics, on the fly, via scripting (usually JavaScript) |
| <figcaption> | Defines a caption for a <figure> element |
| <figure> | Specifies self-contained content |

Audio / Video

|  |  |
| --- | --- |
| Tag | Description |
| <audio> | Defines sound content |
| <source> | Defines multiple media resources for media elements (<video> and <audio>) |
| <track> | Defines text tracks for media elements (<video> and <audio>) |
| <video> | Defines a video or movie |

Links

|  |  |
| --- | --- |
| Tag | Description |
| <a> | Defines a hyperlink |
| <link> | Defines the relationship between a document and an external resource (most used to link to style sheets) |
| <nav> | Defines navigation links |

Lists

|  |  |
| --- | --- |
| Tag | Description |
| <ul> | Defines an unordered list |
| <ol> | Defines an ordered list |
| <li> | Defines a list item |
| <dir> | Not supported in HTML5. Use <ul> instead. |
|  | Defines a directory list |
| <dl> | Defines a description list |
| <dt> | Defines a term/name in a description list |
| <dd> | Defines a description of a term/name in a description list |
| <menu> | Defines a list/menu of commands |
| <menuitem> | Defines a command/menu item that the user can invoke from a popup menu |

Tables

|  |  |
| --- | --- |
| Tag | Description |
| <table> | Defines a table |
| <caption> | Defines a table caption |
| <th> | Defines a header cell in a table |
| <tr> | Defines a row in a table |
| <td> | Defines a cell in a table |
| <thead> | Groups the header content in a table |
| <tbody> | Groups the body content in a table |
| <tfoot> | Groups the footer content in a table |
| <col> | Specifies column properties for each column within a <colgroup> element |
| <colgroup> | Specifies a group of one or more columns in a table for formatting |

Styles and Semantics

|  |  |
| --- | --- |
| Tag | Description |
| <style> | Defines style information for a document |
| <div> | Defines a section in a document |
| <span> | Defines a section in a document |
| <header> | Defines a header for a document or section |
| <footer> | Defines a footer for a document or section |
| <main> | Specifies the main content of a document |
| <section> | Defines a section in a document |
| <article> | Defines an article |
| <aside> | Defines content aside from the page content |
| <details> | Defines additional details that the user can view or hide |
| <dialog> | Defines a dialog box or window |
| <summary> | Defines a visible heading for a <details> element |

Meta Info

|  |  |
| --- | --- |
| Tag | Description |
| <head> | Defines information about the document |
| <meta> | Defines metadata about an HTML document |
| <base> | Specifies the base URL/target for all relative URLs in a document |
| <basefont> | Not supported in HTML5. Use CSS instead. |
|  | Specifies a default color, size, and font for all text in a document |

Programming

|  |  |
| --- | --- |
| Tag | Description |
| <script> | Defines a client-side script |
| <noscript> | Defines an alternate content for users that do not support client-side scripts |
| <applet> | Not supported in HTML5. Use <embed> or <object> instead. |
|  | Defines an embedded applet |
| <embed> | Defines a container for an external (non-HTML) application |
| <object> | Defines an embedded object |
| <param> | Defines a parameter for an object |