Q1: Differences between Browser JS and Node JS

**Answer:**

|  |  |  |  |
| --- | --- | --- | --- |
| S. No | **Factor** | **Browser JS** | **Node JS** |
| 1 | **Application** | Mainly used for client-side (front-end) applications. | Mainly used for server-side(back-end) applications. |
| 2 | **System Access** | It is sandboxed in a browser and has limited access to the browser. | It has full system access |
| 3 | **Missing Objects** | Objects like global, require are missing. | Objects like Window, location, document are missing. |
| 4 | **Running Engine** | It runs on any engine like monkey (Firefox), JavaScript Core (Safari), V8 (Google Chrome). | It runs only on V8 |
| 5 | **GUI** | It is with GUI | No GUI |
| 6 | **Modularity** | Not mandatory | Everything is a module. |

Q2: Watch the video and summarize 5 points:

-<https://www.youtube.com/watch?v=SmE4OwHztCc&ab_channel=JSConf>

**Answer:**

**Components of a Browser:**

Bindings: Operating system, Network, etc.

Rendering: Parsing, Layout, Painting etc.

Platform, JS Virtual Machine

**Performance Insight 1:**

DOM + CSSOM:

Combines the two object models, style resolution

Multiple Trees:

* Render objects
* Render styles
* Render layers
* Line boxes

Not in render tree:

Non-visual elements head, script, title etc

Nodes hidden via display: none

Render Tree:

DOM Node to Render Object:

* Visual output
* Geometric Info
* Can layout and paint
* Holds style and computed metrics

Calculating visual properties:

* Combines all styles
* Defaults, external, style elements and inline
* Complexity around matching rules for each element
* Style computation

Layout:

Recursive process:

* Traverse render tree
* Nodes position and size
* Layout its children

Will batch layouts:

* Incremental layouts
* The browser will intelligently batch changes
* Render tree items will flag themselves as dirty
* The batch will transverse the tree and find all dirty trees
* Asynchronous

Immediate layout:

* Doing a font-size change will relay out the entire document
* Same with browser resize
* Accessing certain properties via JavaScript e.g. node.offsetHeight

**Performance insight 2:**

Take note from the browser and batch

* Act like the browser and batch your DOM changes
* Do all your reads in one pass
* Followed by writes

Real world

* FastDom, Preventing layout thrashing
* Most modern JS frameworks do this internally.

Paint:

* Will take the layedout render trees.
* Creates layers.
* Incremental process.
* Builds up over 12 phases.

Render layers:

* Creates layers from Render Objects
* Position nodes, transparency, overflow, canvas, video etc
* Many-to-I relationship a Render Layer could contain multiple renderojects

Painting:

* Produces a bitmap from each layer
* Bitmap is uploaded to the GPU as a texture
* Composites the textures into a final image to render to the screen.

**Performance Index 3:**

Inline critical CSS:

* The most important bits of your site/app
* Speeds up first paint times
* External JS and CSS can block
* Delta last bitmap

Q3: Code and its output

Answer:

|  |  |  |
| --- | --- | --- |
| S.No | Code | Output |
| 1. | typeof(1) | number |
| 2. | typeof(1.1) | number |
| 3. | typeof(“1.1”) | string |
| 4. | typeof(true) | Boolean |
| 5. | typeof(null) | object |
| 6. | typeof(undefined) | undefined |
| 7. | typeof([]) | object |
| 8. | typeof({}) | object |
| 9. | typeof(NaN) | number |

4. **Prototype:**

Answer: The prototype is an object that is associated with every functions and objects by default in JavaScript, where function's prototype property is accessible and modifiable and object's prototype property (aka attribute) is not visible. The prototype object is special type of enumerable object to which additional properties can be attached to it which will be shared across all the instances of its constructor function.