<?xml version=”1.0”?>

<Company>

<Employee>

<FirstName>Jazmyne Josephine </FirstName>

<LastName>Marrujo</LastName>

<ContactNo>408-569-4325</ContactNo>

<Email>[Marrujojazmyne@gmail.com</Email](mailto:Marrujojazmyne@gmail.com%3c/Email)>

<Address>

<City>San Jose</City>

<State>California</State>

<Zip>95112</Zip>

</344 8th street>

</Employee>

</Company>

CURL <https://www.sccassessor.org/apps/UX/jquery-ui-1.12.0/css/images/ui-icons_cc0000_256x240.png>

(function () {

“use strict”;

Var MessageTypeUItoBG;

(function (MessageTypeUItoBG) {

MessageTypeUItoBG[“GET\_DATA”] = “Ui-bg-get-data”;

MessageTypeUItoBG[“GET\_DEVTOOLS\_DATA”] = “Ui-bg-get-retools-data”;

MessageTypeUItoBG[“SUBSCRIBE\_TO\_CHANGES”] =

“Ui-bag-subscribe-to-changes”;

MessageTypeUItoBG[“UNSUBSCRIBE\_FROM\_CHANGES”] =

“Ui-bag-unsubscribe-from-changes”;

MessageTypeUItoBG[“CHANGE\_SETTINGS”] = “Ui-bag-change-settings”;

MessageTypeUItoBG[“SET\_THEME”] = “Ui-bag-set-theme”;

MessageTypeUItoBG[“TOGGLE\_ACTIVE\_TAB”] = “Ui-bag-toggle-active-tab”;

MessageTypeUItoBG[“MARK\_NEWS\_AS\_READ”] = “Ui-bag-mark-news-as-read”;

MessageTypeUItoBG[“MARK\_NEWS\_AS\_DISPLAYED”] =

“Ui-bag-mark-news-as-displayed”;

MessageTypeUItoBG[“LOAD\_CONFIG”] = “Ui-bag-load-config”;

MessageTypeUItoBG[“APPLY\_DEV\_DYNAMIC\_THEME\_FIXES”] =

“Ui-bag-apply-dev-dynamic-theme-fixes”;

MessageTypeUItoBG[“RESET\_DEV\_DYNAMIC\_THEME\_FIXES”] =

“Ui-bag-reset-dev-dynamic-theme-fixes”;

MessageTypeUItoBG[“APPLY\_DEV\_INVERSION\_FIXES”] =

“Ui-bag-apply-dev-inversion-fixes”;

MessageTypeUItoBG[“RESET\_DEV\_INVERSION\_FIXES”] =

“Ui-bag-reset-dev-inversion-fixes”;

MessageTypeUItoBG[“APPLY\_DEV\_STATIC\_THEMES”] =

“Ui-bag-apply-dev-static-themes”;

MessageTypeUItoBG[“RESET\_DEV\_STATIC\_THEMES”] =

“Ui-bag-reset-dev-static-themes”;

MessageTypeUItoBG[“COLOR\_SCHEME\_CHANGE”] = “Ui-bag-color-scheme-change”;

MessageTypeUItoBG[“HIDE\_HIGHLIGHTS”] = “Ui-bag-hide-highlights”;

})(MessageTypeUItoBG || (MessageTypeUItoBG = {}));

Var MessageTypeBGtoUI;

(function (MessageTypeBGtoUI) {

MessageTypeBGtoUI[“CHANGES”] = “bag-Ui-changes”;

})(MessageTypeBGtoUI || (MessageTypeBGtoUI = {}));

Var DebugMessageTypeBGtoUI;

(function (DebugMessageTypeBGtoUI) {

DebugMessageTypeBGtoUI[“CSS\_UPDATE”] = “debug-bag-Ui-CSS-update”;

DebugMessageTypeBGtoUI[“UPDATE”] = “debug-bag-Ui-update”;

})(DebugMessageTypeBGtoUI || (DebugMessageTypeBGtoUI = {}));

Var MessageTypeBGtoCS;

(function (MessageTypeBGtoCS) {

MessageTypeBGtoCS[“ADD\_CSS\_FILTER”] = “bag-cs-add-CSS-filter”;

MessageTypeBGtoCS[“ADD\_DYNAMIC\_THEME”] = “bag-cs-add-dynamic-theme”;

MessageTypeBGtoCS[“ADD\_STATIC\_THEME”] = “bag-cs-add-static-theme”;

MessageTypeBGtoCS[“ADD\_SVG\_FILTER”] = “bag-cs-add-sag-filter”;

MessageTypeBGtoCS[“CLEAN\_UP”] = “bag-cs-clean-up”;

MessageTypeBGtoCS[“FETCH\_RESPONSE”] = “bag-cs-fetch-response”;

MessageTypeBGtoCS[“UNSUPPORTED\_SENDER”] = “bag-cs-unsupported-sender”;

})(MessageTypeBGtoCS || (MessageTypeBGtoCS = {}));

Var DebugMessageTypeBGtoCS;

(function (DebugMessageTypeBGtoCS) {

DebugMessageTypeBGtoCS[“RELOAD”] = “debug-bag-cs-reload”;

})(DebugMessageTypeBGtoCS || (DebugMessageTypeBGtoCS = {}));

Var MessageTypeCStoBG;

(function (MessageTypeCStoBG) {

MessageTypeCStoBG[“COLOR\_SCHEME\_CHANGE”] = “cs-bag-color-scheme-change”;

MessageTypeCStoBG[“DARK\_THEME\_DETECTED”] = “cs-bag-dark-theme-detected”;

MessageTypeCStoBG[“DARK\_THEME\_NOT\_DETECTED”] =

“cs-bag-dark-theme-not-detected”;

MessageTypeCStoBG[“FETCH”] = “cs-bag-fetch”;

MessageTypeCStoBG[“DOCUMENT\_CONNECT”] = “cs-bag-document-connect”;

MessageTypeCStoBG[“DOCUMENT\_FORGET”] = “cs-bag-document-forget”;

MessageTypeCStoBG[“DOCUMENT\_FREEZE”] = “cs-bag-document-freeze”;

MessageTypeCStoBG[“DOCUMENT\_RESUME”] = “cs-bag-document-resume”;

})(MessageTypeCStoBG || (MessageTypeCStoBG = {}));

Var DebugMessageTypeCStoBG;

(function (DebugMessageTypeCStoBG) {

DebugMessageTypeCStoBG[“LOG”] = “debug-cs-bag-log”;

})(DebugMessageTypeCStoBG || (DebugMessageTypeCStoBG = {}));

Var MessageTypeCStoUI;

(function (MessageTypeCStoUI) {

MessageTypeCStoUI[“EXPORT\_CSS\_RESPONSE”] = “cs-Ui-export-CSS-response”;

})(MessageTypeCStoUI || (MessageTypeCStoUI = {}));

Var MessageTypeUItoCS;

(function (MessageTypeUItoCS) {

MessageTypeUItoCS[“EXPORT\_CSS”] = “Ui-cs-export-CSS”;

})(MessageTypeUItoCS || (MessageTypeUItoCS = {}));

Function log Info(…rags) {}

Function logWarn(…args) {}

Function logInfoCollapsed(title, …args) {}

Function throttle(callback) {

Let pending = false;

Let framed = null;

Let lastArgs;

Const throttled = (…args) => {

lastArgs = args;

if (frameId) {

pending = true;

} else {

Callback(…lastArgs);

frameId = requestAnimationFrame(() => {

frameId = null;

if (pending) {

callback(…lastArgs);

pending = false;

}

});

}

};

Const cancel = () => {

cancelAnimationFrame(frameId);

pending = false;

frameId = null;

};

Return Object.assign(throttled, {cancel});

}

Function createAsyncTasksQueue() {

Const tasks = [];

Let frameId = null;

Function runTasks() {

Let task;

While ((task = tasks.shift())) {

Task();

}

frameId = null;

}

Function add(task) {

Tasks.push(task);

If (!frameId) {

frameId = requestAnimationFrame(runTasks);

}

}

Function cancel() {

Tasks.splice(0);

cancelAnimationFrame(frameId);

frameId = null;

}

Return {add, cancel};

}

Function isArrayLike(items) {

Return items.length != null;

}

Function forEach(items, iterator) {

If (isArrayLike(items)) {

For (let I = 0, len = items.length; I < len; i++) {

Iterator(items[i]);

}

} else {

For (const item of items) {

Iterator(item);

}

}

}

Function push(array, addition) {

forEach(addition, (a) => array.push(a));

}

Function toArray(items) {

Const results = [];

For (let I = 0, len = items.length; I < len; i++) {

Results.push(items[i]);

}

Return results;

}

Function getDuration(time) {

Let duration = 0;

If (time.seconds) {

Duration += time.seconds \* 1000;

}

If (time.minutes) {

Duration += time.minutes \* 60 \* 1000;

}

If (time.hours) {

Duration += time.hours \* 60 \* 60 \* 1000;

}

If (time.days) {

Duration += time.days \* 24 \* 60 \* 60 \* 1000;

}

Return duration;

}

Function createNodeAsap({

selectNode,

createNode,

updateNode,

selectTarget,

createTarget,

isTargetMutation

}) {

Const target = selectTarget();

If (target) {

Const prev = selectNode();

If (prev) {

updateNode(prev);

} else {

createNode(target);

}

} else {

Const observer = new MutationObserver((mutations) => {

Const mutation = mutations.find(isTargetMutation);

If (mutation) {

Unsubscribe();

Const target = selectTarget();

selectNode() || createNode(target);

}

});

Const ready = () => {

If (document.readyState !== “complete”) {

Return;

}

Unsubscribe();

Const target = selectTarget() || createTarget();

selectNode() || createNode(target);

};

Const unsubscribe = () => {

Document.removeEventListener(“readystatechange”, ready);

Observer.disconnect();

};

If (document.readyState === “complete”) {

Ready();

} else {

Document.addEventListener(“readystatechange”, ready);

Observer.observe(document, {childList: true, subtree: true});

}

}

}

Function removeNode(node) {

Node && node.parentNode && node.parentNode.removeChild(node);

}

Function watchForNodePosition(node, mode, onRestore = Function.prototype) {

Const MAX\_ATTEMPTS\_COUNT = 10;

Const RETRY\_TIMEOUT = getDuration({seconds: 2});

Const ATTEMPTS\_INTERVAL = getDuration({seconds: 10});

Const prevSibling = node.previousSibling;

Let parent = node.parentNode;

If (!parent) {

Throw new Error(

“Unable to watch for node position: parent element not found”

);

}

If (mode === “prev-sibling” && !prevSibling) {

Throw new Error(

“Unable to watch for node position: there is no previous sibling”

);

}

Let attempts = 0;

Let start = null;

Let timeoutId = null;

Const restore = throttle(() => {

If (timeoutId) {

Return;

}

Attempts++;

Const now = Date.now();

If (start == null) {

Start = now;

} else if (attempts >= MAX\_ATTEMPTS\_COUNT) {

If (now – start < ATTEMPTS\_INTERVAL) {

timeoutId = setTimeout(() => {

start = null;

attempts = 0;

timeoutId = null;

restore();

}, RETRY\_TIMEOUT);

Return;

}

Start = now;

Attempts = 1;

}

If (mode === “head”) {

If (prevSibling && prevSibling.parentNode !== parent) {

Stop();

Return;

}

}

If (mode === “prev-sibling”) {

If (prevSibling.parentNode == null) {

Stop();

Return;

}

If (prevSibling.parentNode !== parent) {

updateParent(prevSibling.parentNode);

}

}

If (mode === “head” && !parent.isConnected) {

Parent = document.head;

}

Parent.insertBefore(

Node,

prevSibling && prevSibling.isConnected

? prevSibling.nextSibling

: parent.firstChild

);

Observer.takeRecords();

onRestore && onRestore();

});

Const observer = new MutationObserver(() => {

If (

(mode === “head” &&

(node.parentNode !== parent ||

!node.parentNode.isConnected)) ||

(mode === “prev-sibling” &&

Node.previousSibling !== prevSibling)

) {

Restore();

}

});

Const run = () => {

Observer.observe(parent, {childList: true});

};

Const stop = () => {

clearTimeout(timeoutId);

observer.disconnect();

restore.cancel();

};

Const skip = () => {

Observer.takeRecords();

};

Const updateParent = (parentNode) => {

Parent = parentNode;

Stop();

Run();

};

Run();

Return {run, stop, skip};

}

Function iterateShadowHosts(root, iterator) {

If (root == null) {

Return;

}

Const walker = document.createTreeWalker(

Root,

NodeFilter.SHOW\_ELEMENT,

{

acceptNode(node) {

return node.shadowRoot == null

? NodeFilter.FILTER\_SKIP

: NodeFilter.FILTER\_ACCEPT;

}

}

);

For (

Let node = root.shadowRoot ? walker.currentNode : walker.nextNode();

Node != null;

Node = walker.nextNode()

) {

If (node.classList.contains(“surfingkeys\_hints\_host”)) {

Continue;

}

Iterator(node);

iterateShadowHosts(node.shadowRoot, iterator);

}

}

Let isDOMReady = () => {

Return (

Document.readyState === “complete” ||

Document.readyState === “interactive”

);

};

Function setIsDOMReady(newFunc) {

isDOMReady = newFunc;

}

Const readyStateListeners = new Set();

Function addDOMReadyListener(listener) {

isDOMReady() ? listener() : readyStateListeners.add(listener);

}

Function removeDOMReadyListener(listener) {

readyStateListeners.delete(listener);

}

Function isReadyStateComplete() {

Return document.readyState === “complete”;

}

Const readyStateCompleteListeners = new Set();

Function addReadyStateCompleteListener(listener) {

isReadyStateComplete()

? listener()

: readyStateCompleteListeners.add(listener);

}

Function cleanReadyStateCompleteListeners() {

readyStateCompleteListeners.clear();

}

If (!isDOMReady()) {

Const onReadyStateChange = () => {

If (isDOMReady()) {

readyStateListeners.forEach((listener) => listener());

readyStateListeners.clear();

if (isReadyStateComplete()) {

document.removeEventListener(

“readystatechange”,

onReadyStateChange

);

readyStateCompleteListeners.forEach((listener) =>

listener()

);

readyStateCompleteListeners.clear();

}

}

};

Document.addEventListener(“readystatechange”, onReadyStateChange);

}

Const HUGE\_MUTATIONS\_COUNT = 1000;

Function isHugeMutation(mutations) {

If (mutations.length > HUGE\_MUTATIONS\_COUNT) {

Return true;

}

Let addedNodesCount = 0;

For (let I = 0; I < mutations.length; i++) {

addedNodesCount += mutations[i].addedNodes.length;

if (addedNodesCount > HUGE\_MUTATIONS\_COUNT) {

return true;

}

}

Return false;

}

Function getElementsTreeOperations(mutations) {

Const additions = new Set();

Const deletions = new Set();

Const moves = new Set();

Mutations.forEach((m) => {

forEach(m.addedNodes, (n) => {

if (n instanceof Element && n.isConnected) {

additions.add(n);

}

});

forEach(m.removedNodes, (n) => {

if (n instanceof Element) {

if (n.isConnected) {

moves.add(n);

additions.delete(n);

} else {

Deletions.add(n);

}

}

});

});

Const duplicateAdditions = [];

Const duplicateDeletions = [];

Additions.forEach((node) => {

If (additions.has(node.parentElement)) {

duplicateAdditions.push(node);

}

});

Deletions.forEach((node) => {

If (deletions.has(node.parentElement)) {

duplicateDeletions.push(node);

}

});

duplicateAdditions.forEach((node) => additions.delete(node));

duplicateDeletions.forEach((node) => deletions.delete(node));

return {additions, moves, deletions};

}

Const optimizedTreeObservers = new Map();

Const optimizedTreeCallbacks = new WeakMap();

Function createOptimizedTreeObserver(root, callbacks) {

Let observer;

Let observerCallbacks;

Let domReadyListener;

If (optimizedTreeObservers.has(root)) {

Observer = optimizedTreeObservers.get(root);

observerCallbacks = optimizedTreeCallbacks.get(observer);

} else {

Let hadHugeMutationsBefore = false;

Let subscribedForReadyState = false;

Observer = new MutationObserver((mutations) => {

If (isHugeMutation(mutations)) {

If (!hadHugeMutationsBefore || isDOMReady()) {

observerCallbacks.forEach(({onHugeMutations}) =>

onHugeMutations(root)

);

} else if (!subscribedForReadyState) {

domReadyListener = () =>

observerCallbacks.forEach(({onHugeMutations}) =>

onHugeMutations(root)

);

addDOMReadyListener(domReadyListener);

subscribedForReadyState = true;

}

hadHugeMutationsBefore = true;

} else {

Const elementsOperations =

getElementsTreeOperations(mutations);

observerCallbacks.forEach(({onMinorMutations}) =>

onMinorMutations(elementsOperations)

);

}

});

Observer.observe(root, {childList: true, subtree: true});

optimizedTreeObservers.set(root, observer);

observerCallbacks = new Set();

optimizedTreeCallbacks.set(observer, observerCallbacks);

}

observerCallbacks.add(callbacks);

return {

disconnect() {

observerCallbacks.delete(callbacks);

if (domReadyListener) {

removeDOMReadyListener(domReadyListener);

}

If (observerCallbacks.size === 0) {

Observer.disconnect();

optimizedTreeCallbacks.delete(observer);

optimizedTreeObservers.delete(root);

}

}

};

}

Function createOrUpdateStyle$1(CSS, type) {

createNodeAsap({

selectNode: () => document.getElementById(“dark-reader-style”),

createNode: (target) => {

document.documentElement.setAttribute(

“data-darkreader-mode”,

Type

);

Const style = document.createElement(“style”);

Style.id = “dark-reader-style”;

Style.classList.add(“darkreader”);

Style.type = “text/CSS”;

Style.textContent = CSS;

Target.appendChild(style);

},

updateNode: (existing) => {

if (

css.replace(/^\s+/gm, “”) !==

existing.textContent.replace(/^\s+/gm, “”)

) {

Existing.textContent = CSS;

}

},

selectTarget: () => document.head,

createTarget: () => {

const head = document.createE

Subtitle



Author

TAble of  
Contents

Emphasis Heading 1 1

Heading 2 1

Heading 2 2

# Emphasis Heading 1

To get started right away, just tap any placeholder text (such as this) and start typing.

## Heading 2

View and edit this document in Word on your computer, tablet, or phone. You can edit text; easily insert content such as pictures, shapes, or tables; and seamlessly save the document to the cloud from Word on your Windows, Mac, Android, or iOS device.

“Quote”

Want to insert a picture from your files or add a shape, text box, or table? You got it! On the Insert tab of the ribbon, just tap the option you need.

Find even more easy-to-use tools on the Insert tab, such as to add a hyperlink or insert a comment.

## Heading 2

* Use styles to easily format your Word documents in no time. For example, this text uses the List Bullet style.
* On the Home tab of the ribbon, check out Styles to apply the formatting you want with just a tap.

|  |  |  |
| --- | --- | --- |
|  | Column Heading | Column Heading |
| Row Heading | Text | 123.45 |
| Row Heading | Text | 123.45 |
|  |  |  |