

Getting on key

Iván Sánchez Ortega

Consultant productivity engineer
at Collabora Productivity

<http://ivan.sanchezortega.es>



Getting on key

Iván Sánchez Ortega

Javascript person working remotely

<http://ivan.sanchezortega.es>



This talk is a **love letter**.

2014

- Indoor maps for Mazemap.com

2014

- Indoor maps for Mazemap.com
- Debugging touch interactions on Leaflet

2014

- Indoor maps for Mazemap.com
- Debugging touch interactions on Leaflet
- Saw the "Getting Touchy" presentation by Patrick H. Lauke

2014

- Indoor maps for Mazemap.com
- Debugging touch interactions on Leaflet
- Saw the "Getting Touchy" presentation by Patrick H. Lauke
- Best resource for JS people to understand touchscreens

A close-up photograph of a hand interacting with a large, glowing blue touch screen. The screen is illuminated with a bright blue light, and the hand is positioned over it, with fingers touching the surface. The background is dark, making the glowing screen stand out.

getting **touchy**

EVERYTHING YOU (N)EVER WANTED TO KNOW ABOUT **TOUCH AND POINTER** EVENTS

Patrick H. Lauke / Last major changes: 8 May 2017

This talk is my love letter to "**getting touchy**".

I can only hope this helps somebody in a similar way than "getting touchy" helped me.

<https://patrickhlauke.github.io/getting-touchy-presentation/>
<https://invidio.us/watch?v=bBdQkhm1DvI>

Getting on key

Everything you (n)ever wanted to know
about keyboard and input events



Press a key → application gets notified

Press a key → application gets notified

Press a key **down**, release the key **up**.

Press a key → application gets notified

Press a key **down**, release the key **up**.

Hold a key **down**, OS will generate several **key presses** (based on the keyboard repeat speed), release the key **up**.

A

- `keydown` (phys key A)
- `keypress` (ASCII lowercase "a")
- `keyup` (phys key A)

Shift+S

- `keydown` (phys key LeftShift)
- `keydown` (phys key S)
- `keypress` (ASCII uppercase "S")
- `keyup` (phys key S)
- `keyup` (phys key LeftShift)

Is a keyboard event writing something (**Shift**+**S**) or
not (**Ctrl**+**S**)?

Is a keyboard event writing something (**Shift+S**) or
not (**Ctrl+S**)?

Enter the **textInput** events. These **deprecate**
keypress.

What's the state of the system *during* event handling?

What's the state of the system *during* event handling?

Undefined

What's the state of the system *during* event handling?

Undefined

Enter the `beforeinput` and `input` events. These **deprecate** both `textInput` and `keypress`.

A

- `keydown` (phys key A)
- `beforeinput` (ASCII lowercase "a")
- ~~`keypress` (ASCII lowercase "a")~~
- `input` (ASCII lowercase "a")
- `keyup` (phys key A)

Shift+S

- `keydown` (phys key LeftShift)
- `keydown` (phys key S)
- `beforeinput` (ASCII uppercase "S")
- ~~`keypress` (ASCII uppercase "S")~~
- `input` (ASCII uppercase "S")
- `keyup` (phys key S)
- `keyup` (phys key LeftShift)

But then, not all people use QWERTY

~ , `	! 1	@ 2	# 3	\$ 4	% 5	^ 6	& 7	* 8	(9) 0	{ [}]	← Backspace
Tab ↔	" '	< ,	> .	P	Y	F	G	C	R	L	? /	+ =	 \ _
Caps Lock ↑	A	O	E	U	I	D	H	T	N	S	- _	Enter ↵	
Shift ↑	:	Q	J	K	X	B	M	W	V	Z	Shift ↑		
Ctrl	Win Key	Alt							Alt Gr	Win Key	Menu	Ctrl	

KeyboardEvents in Javascript have:

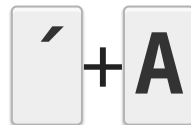
- `code` (hardware scancode, layout-independant)
- `location` (hardware scancode, layout-independant)
- `keyCode` (implementation-dependant)
- `which` (implementation-dependant)
- `keyIdentifier` (transitional)
- `keyLocation` (transitional)
- `char` (printable representation of key)
- `charCode` (number for printable representation of key)
- `location` (tell apart e.g. leftCtrl & rightCtrl)
- **`key` (name of key)**

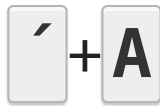
KeyboardEvents in Javascript have:

- `code` (hardware scancode, layout-independant)
- `location` (hardware scancode, layout-independant)
- `keyCode` (implementation-dependant)
- `which` (implementation-dependant)
- `keyIdentifier` (transitional)
- `keyLocation` (transitional)
- `char` (printable representation of key)
- `charCode` (number for printable representation of key)
- `location` (tell apart e.g. leftCtrl & rightCtrl)
- **`key`** (name of key)

Use `key` and **forget about everything else.**

And then we have dead keys, as the á in Iván





- `keydown` (phys key composing acute)
- `keyup` (phys key composing acute)
- `keydown` (phys key A)
- `beforeinput` (lowercase "á")
- ~~`keypress` (lowercase "á")~~
- `input` (lowercase "á")
- `keyup` (phys key A)

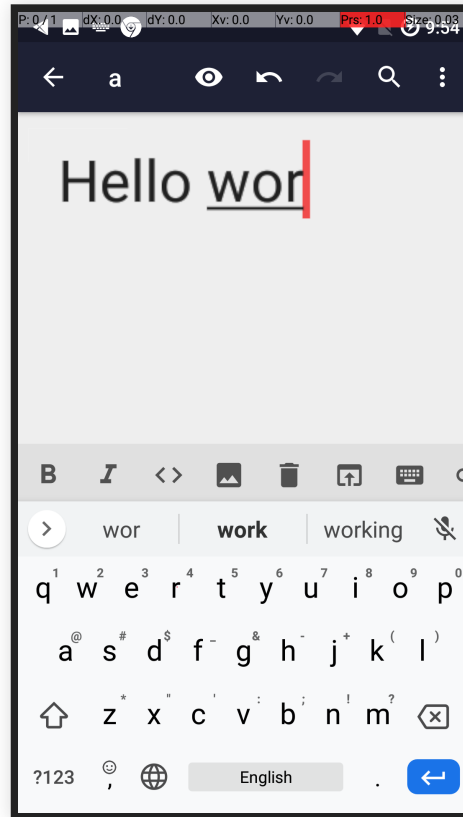


- `keydown` (phys key A)
- `beforeinput` (lowercase "á")
- ~~`keypress` (lowercase "á")~~
- `input` (lowercase "á")
- `keyup` (phys key A)

And then the western (i.e. english-writing) world forgot about IME or text composition

And then the western (i.e. english-writing) world forgot about IME or text composition until autocorrect on on-screen keyboards happened.

If the word you're writing is **underlined**, then you're using text composition.



H then choose an autocomplete suggestion

- `keydown` (phys key Unidentified)
- `compositionstart`
- `beforeinput` (uppercase "H")
- `compositionupdate` ("H")
- ~~`keypress` (uppercase "H" or Unidentified)~~
- `input` (uppercase "H")
- `keyup` (phys key Unidentified)
- `keydown` (phys key Unidentified)
- `compositionupdate` ("Hello")
- `beforeinput` ("Hello")
- `compositionupdate` ("Hello")
- ~~`keypress` (uppercase "H" or Unidentified)~~
- `compositionend` ("Hello")
- `input` (uppercase "Hello")
- `keyup` (phys key Unidentified)

Not-really-intuitive stuff about text composition:

- The word being composed is part of the input field
- `selectionstart` and `selectionend` span the word being composed during *some* event handlers
- `selectionstart` and `selectionend` are both the position of the cursor caret during any other event handler
- Behaviour across platforms is **amazingly inconsistent**

The life of a Javascript developer testing stuff:

- Firefox (gecko)
- Chrome (blink)
- Safari (webkit)
- IE / Edge (trident/EdgeHTML)

The life of a Javascript developer testing stuff **on
keyboards:**

- Firefox (gecko)
- Chrome (blink)
- Safari (webkit)
- IE / Edge (trident/EdgeHTML)
- USB keyboard (and PS/2 and laptop)
- GBoard
- AnySoftKeyboard
- SwiftKey
- AOSP keyboard (android)
- Samsung keyboard (android)
- iOS
- Win10 on-screen-keyboard
- iOS + Bluetooth
- Android + Bluetooth
- Speech recognition

How to approach this problem?

How to approach this problem?



	Firefox 67.0.0 / Linux 0.0.0 / usb	Chrome Mobile 75.0.3770 / Android 0.0.0 / gboard	Firefox 60.0.0 / Linux 0.0.0 / usb	Chrome 75.0.3770 / Linux 0.0.0 / usb	IE 11.0.0 / Windows 8.1.0.0 / usb	Edge 18.17763.0 / Windows 10.0.0 / usb	Edge 18.17763.0 / Windows 10.0.0 / winosk	Chrome 77.0.3862 / Windows 10.0.0 / usb	Chrome 77.0.3862 / Windows 10.0.0 / winosk	Firefox 68.0.0 / Windows 10.0.0 / usb	Firefox 68.0.0 / Windows 10.0.0 / winosk	Chrome Mobile 77.0.3857 / Android 6.0.1 / gboard	Chrome Mobile 77.0.3857 / Android 6.0.1 / anysoftkeyboard	Chrome Mobile 77.0.3857 / Android 6.0.1 / aosp	Chrome Mobile 51.0.2704 / Android 6.0.1 / aosp
ta-hello	[view]	[view]	[view]	[view]	[view]	[view]		[view]		[view]		[view]	[view]	[view]	[view]
ce-hello	[view]	[view]	[view]	[view]	[view]	[view]		[view]		[view]		[view]	[view]	[view]	[view]
ta-hello-space	[view]	[view]	[view]	[view]	[view]	[view]		[view]		[view]		[view]	[view]	[view]	[view]
ce-hello-space	[view]	[view]	[view]	[view]	[view]	[view]		[view]		[view]		[view]	[view]	[view]	[view]
ta-hello-dot	[view]	[view]	[view]	[view]	[view]	[view]		[view]		[view]		[view]	[view]	[view]	[view]
ce-hello-dot	[view]	[view]	[view]	[view]	[view]	[view]		[view]		[view]		[view]	[view]	[view]	[view]
ta-backspace	[view]	[view]	[view]	[view]	[view]	[view]		[view]	[view]	[view]		[view]	[view]	[view]	[view]
ce-backspace	[view]	[view]	[view]	[view]	[view]	[view]		[view]		[view]		[view]	[view]	[view]	[view]
ta-delete	[view]		[view]	[view]	[view]	[view]		[view]		[view]			[view]		
ce-delete	[view]		[view]	[view]	[view]	[view]		[view]		[view]			[view]		
ta-backspace-stxetx	[view]	[view]	[view]	[view]	[view]	[view]		[view]	[view]	[view]		[view]	[view]	[view]	[view]
ce-backspace-stxetx	[view]	[view]	[view]	[view]	[view]	[view]		[view]		[view]		[view]	[view]	[view]	[view]
ta-delete-stxetx	[view]		[view]	[view]	[view]	[view]		[view]		[view]			[view]		
ce-delete-stxetx	[view]		[view]	[view]	[view]	[view]		[view]		[view]			[view]		
ta-backspace-andorra	[view]	[view]	[view]	[view]	[view]	[view]		[view]	[view]	[view]		[view]	[view]	[view]	[view]
ce-backspace-andorra	[view]	[view]	[view]	[view]	[view]	[view]		[view]	[view]	[view]		[view]	[view]	[view]	[view]
ta-delete-andorra	[view]		[view]	[view]	[view]	[view]		[view]		[view]			[view]		
ce-delete-andorra	[view]		[view]	[view]	[view]	[view]		[view]		[view]			[view]		
ta-move-left	[view]	[view]	[view]	[view]	[view]	[view]		[view]		[view]		[view]	[view]	[view]	
ce-move-left	[view]	[view]	[view]	[view]	[view]	[view]		[view]	[view]	[view]		[view]	[view]	[view]	
ta-move-right	[view]	[view]	[view]	[view]	[view]	[view]		[view]		[view]		[view]	[view]	[view]	
ce-move-right	[view]	[view]	[view]	[view]	[view]	[view]		[view]		[view]		[view]	[view]	[view]	
ta-move-left-andorra	[view]	[view]	[view]	[view]	[view]	[view]		[view]		[view]		[view]	[view]	[view]	
ce-move-left-andorra	[view]	[view]	[view]	[view]	[view]	[view]		[view]		[view]		[view]	[view]	[view]	

More than 1000 (!!) event traces

Major takeaway:

The same sequence of user actions produce different results **depending only on the keyboard:**

- Space after autosuggestion?
- Accept autocorrect suggestion by pressing space?
- Press **.** after spellcheck and automatic space. Is space removed?
- How are word capitalization, spellcheck-disable and rich text hints handled?

Keyboard behaviour **depends on the what's already written** and is **browser-independant**.

Major takeaway:

Browsers can be catalogued into three categories:

- `inputType` support (field in `input` and `beforeinput` events)
- `textInput` support (deprecated event)
- Legacy (no `inputType`, no `textInput` and no `beforeinput`)

inputType

Good idea, ok-ish spec, subpar implementations (so far)

<https://developer.mozilla.org/en-US/docs/Web/API/InputEvent/inputType>

<https://w3c.github.io/input-events/#dom-inputevent-inputtype>

inputType	User's expression of intention
"insertText"	insert typed plain text
"insertReplacementText"	replace existing text by means of a spell checker, auto-correct or similar
"insertLineBreak"	insert a line break
"insertParagraph"	insert a paragraph break
"insertOrderedList"	insert a numbered list
"insertUnorderedList"	insert a bulleted list
"insertHorizontalRule"	insert a horizontal rule
"insertFromYank"	replace the current selection with content stored in a kill buffer
"insertFromDrop"	insert content into the DOM by means of drop
"insertFromPaste"	paste
"insertFromPasteAsQuotation"	paste content as a quotation
"insertTranspose"	transpose the last two characters that were entered
"insertCompositionText"	replace the current composition string
"insertFromComposition"	insert into the DOM a finalized composed string that will not form part of the next composition string
"insertLink"	insert a link
"deleteByComposition"	remove a part of the DOM in order to recompose this part using IME
"deleteCompositionText"	delete the current composition string before committing a finalized string to the DOM
"deleteWordBackward"	delete a word directly before the caret position
"deleteWordForward"	delete a word directly after the caret position
"deleteSoftLineBackward"	delete from the caret to the nearest visual line break before the caret position
"deleteSoftLineForward"	delete from the caret to the nearest visual line break after the caret position
"deleteEntireSoftLine"	delete from to the nearest visual line break before the caret position to the nearest visual line break after the caret position
"deleteHardLineBackward"	delete from the caret to the nearest beginning of a block element or br element before the caret position



`insertCompositionText` means "*replace* current composition string"

WTF?

Theory

- `compositionstart`
 - `compositionupdate + input/insertCompositionText`
 - `compositionupdate + input/insertCompositionText`
 - `compositionupdate + input/insertCompositionText`
 - `compositionend + input/insertFromComposition`
-

Common reality

`compositionend + (input/deleteContentBackwards) + input/insertText`
or `input/insertCompositionText`

No browser ever fires

`input/insertFromComposition`

Event	Key/Data/Mutations	Selection/ranges	Value/InnerHTML
compositionstart		forward[0-0]	0[]
beforeinput/insertCompositionText	data[H]	forward[0-0]	0[]
compositionupdate	data[H]	forward[0-0]	0[]
input/insertCompositionText	data[H]	forward[1-1]	1[H]
beforeinput/insertCompositionText	data[He]	forward[0-1]	1[H]
compositionupdate	data[He]	forward[0-1]	1[H]
input/insertCompositionText	data[He]	forward[2-2]	2[He]
beforeinput/insertCompositionText	data[Hel]	forward[0-2]	2[He]
compositionupdate	data[Hel]	forward[0-2]	2[He]
input/insertCompositionText	data[Hel]	forward[3-3]	3[Hel]
beforeinput/insertCompositionText	data[Hell]	forward[0-3]	3[Hel]
compositionupdate	data[Hell]	forward[0-3]	3[Hel]
input/insertCompositionText	data[Hell]	forward[4-4]	4[Hell]
beforeinput/insertCompositionText	data[Hello]	forward[0-4]	4[Hell]
compositionupdate	data[Hello]	forward[0-4]	4[Hell]
input/insertCompositionText	data[Hello]	forward[5-5]	5[Hello]
compositionend	data[Hello]	forward[5-5]	5[Hello]
beforeinput/insertText	data[]	forward[5-5]	5[Hello]
input/insertText	data[]	forward[6-6]	6[Hello]
change		forward[6-6]	6[Hello]

`input` events have a `isComposing` property.

Don't trust that property. Rely on `compositionstart/compositionend` events instead.

Don't assume that `compositionstart` and `compositionend` events appear in well-formed pairs.

Chrome 51 on Android + AOSP keyboard when spellchecking a previously written word.

Event	Key/Data/Mutations	Selection/ranges	Value/InnerHTML
compositionstart		forward[0-0]	0[]
compositionupdate	data[K]	forward[0-0]	0[]
input		forward[0-1]	1[K]
compositionupdate	data[Ko]	forward[0-1]	1[K]
input		forward[0-2]	2[Ko]
compositionupdate	data[Kov]	forward[0-2]	2[Ko]
input		forward[0-3]	3[Kov]
compositionupdate	data[Kove]	forward[0-3]	3[Kov]
input		forward[0-4]	4[Kove]
compositionupdate	data[Kove]	forward[0-4]	4[Kove]
input		forward[0-4]	4[Kove]
compositionupdate	data[Kovel]	forward[0-4]	4[Kove]
input		forward[0-5]	5[Kovel]
compositionupdate	data[Kovely]	forward[0-5]	5[Kovel]
input		forward[0-6]	6[Kovely]
compositionend	data[Lovely]	forward[0-6]	6[Kovely]
input		forward[6-6]	6[Lovely]
input		forward[7-7]	7[Lovely]
input		forward[0-0]	0[]
input		forward[6-6]	6[Kovely]
compositionend	data[]	forward[0-6]	6[Kovely]
input		forward[1-1]	1[]
input		forward[7-7]	7[Lovely]
change		forward[7-7]	7[Lovely]

Don't assume that composition for latin script happens
only in Android/iOS

Safari on desktop Mac with ~~dead keys~~ composing diacritics.

Don't assume that the order of `textInput` and `beforeinput` and `input` is consistent.

Chrome 77 on Android 6 + bluetooth: `beforeinput`→`textInput`→`input`.

Chrome 74 on iOS 12.3: `textInput`→`beforeinput`→`input`.

Spellcheck that affects the **first letter** is weird

Theory

- `compositionupdate + input/insertCompositionText "koveley"`
 - `compositionupdate + input/insertCompositionText "lovely"`
 - `compositionend + input/insertFromComposition "lovely"`
-

Common reality

- `input/insertText "koveley"`
- `input/deleteContentBackward ""`
- `input/insertText "lovely"`

Timing issues!

No browser ever fires

`input/insertReplacementText`

Event	Key/Data/Mutations	Selection/ranges	Value/InnerHTML
beforeinput/insertText	data[k]	forward[0-0]	0[]
input/insertText	data[k]	forward[1-1]	1[k]
beforeinput/insertText	data[o]	forward[1-1]	1[k]
input/insertText	data[o]	forward[2-2]	2[ko]
beforeinput/insertText	data[v]	forward[2-2]	2[ko]
input/insertText	data[v]	forward[3-3]	3[kov]
beforeinput/insertText	data[e]	forward[3-3]	3[kov]
input/insertText	data[e]	forward[4-4]	4[kove]
beforeinput/insertText	data[l]	forward[4-4]	4[kove]
input/insertText	data[l]	forward[5-5]	5[kovel]
beforeinput/insertText	data[y]	forward[5-5]	5[kovel]
input/insertText	data[y]	forward[6-6]	6[kovely]
beforeinput/insertText	data[]	forward[6-6]	6[kovely]
input/insertText	data[]	forward[7-7]	7[kovely]
beforeinput/deleteContentBackward		forward[0-7]	7[kovely]
input/deleteContentBackward		forward[0-0]	0[]
beforeinput/insertText	data[lovely]	forward[0-0]	0[]
input/insertText	data[lovely]	forward[7-7]	7[lovely]

Don't assume that text composition is needed for
spellcheck.

Win8 / Win10 on-screen keyboard.

Don't assume that the DOM tree doesn't change.

IE and Edge mutate the DOM even when editing a `<textarea>`.

Don't assume that the strings are always valid Unicode.

Win10 + on-screen keyboard + anything but IE/Edge + emojis + UTF16 code units.

One (possibly invalid) UTF16 code unit per keystroke.

Edge 18:

Event	Key/Data/Mutations	Selection/ranges	Value/InnerHTML
input		undefined[2-2]	2[😄]

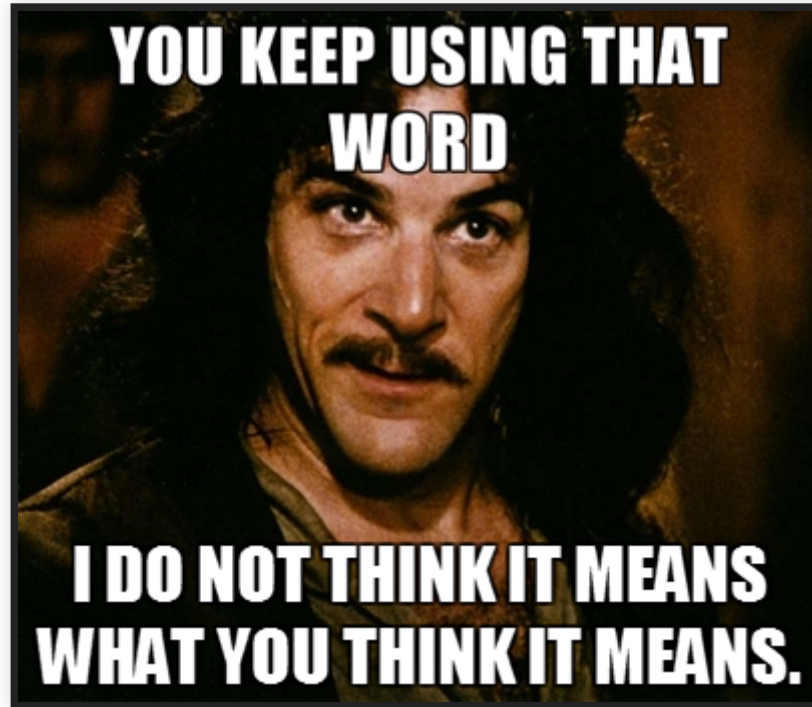
Firefox 68:

Event	Key/Data/Mutations	Selection/ranges	Value/InnerHTML
input/insertText	data[0]	forward[1-1]	1[0]
input/insertText	data[0]	forward[2-2]	2[😄]

Chrome 77:

Event	Key/Data/Mutations	Selection/ranges	Value/InnerHTML
beforeinput/insertText	data[0]	forward[0-0]	0[]
input/insertText	data[0]	forward[1-1]	1[0]
beforeinput/insertText	data[0]	forward[1-1]	1[0]
input/insertText	data[0]	forward[2-2]	2[😄]

Backspace.



Backspace.

- Delete the last character
- Delete the last grapheme
- Delete the last grapheme cluster
- Delete the last ligature
- Delete the last codepoint
- Delete the last codepoint and adjacent zero-width joiners
- Delete the last code unit
- Nothing (if there's nothing to delete)

Family with Edge 18.

Event	Key/Data/Mutations	Selection/ranges	Value/InnerHTML
focus		undefined[0-0]	0[]
keydown	key[👉]	undefined[0-0]	0[]
keypress	key[👉]	undefined[0-0]	0[]
input		undefined[2-2]	2[👉]
mutation/childList	null → 👉	winsel[][11-11]	
keydown	key[]	undefined[2-2]	2[👉]
keypress	key[]	undefined[2-2]	2[👉]
textInput	data[]	undefined[2-2]	2[👉]
input		undefined[3-3]	3[👉]
keydown	key[👈]	undefined[3-3]	3[👉]
keypress	key[👈]	undefined[3-3]	3[👉]
input		undefined[5-5]	5[👉👈]
keydown	key[]	undefined[5-5]	5[👉👈]
keypress	key[]	undefined[5-5]	5[👉👈]
textInput	data[]	undefined[5-5]	5[👉👈]
input		undefined[6-6]	6[👉👈]
keydown	key[👇]	undefined[6-6]	6[👉👈]
keypress	key[👇]	undefined[6-6]	6[👉👈]
input		undefined[8-8]	8[👉👈👇]
keydown	key[]	undefined[8-8]	8[👉👈👇]
keypress	key[]	undefined[8-8]	8[👉👈👇]
textInput	data[]	undefined[8-8]	8[👉👈👇]
input		undefined[9-9]	9[👉👈👇]
keydown	key[👉]	undefined[9-9]	9[👉👈👇]
keypress	key[👉]	undefined[9-9]	9[👉👈👇]
input		undefined[11-11]	11[👉👈]
keydown	key[Backspace]	undefined[11-11]	11[👉👈]
input		undefined[9-9]	9[👉👈]
keyup	key[Backspace]	undefined[9-9]	9[👉👈]
change		undefined[9-9]	9[👉👈]
blur		undefined[9-9]	9[👉👈]

Family codepoints with Edge 18.

Event	Key/Data/Mutations	Selection/ranges	Value/InnerHTML
focus		undefined[0-0]	0[]
keydown	key[0x1f468]	undefined[0-0]	0[]
keypress	key[0x1f468]	undefined[0-0]	0[]
input		undefined[2-2]	2[0x1f468]
mutation/childList	null → 🍌	winserl[][11-11]	
keydown	key[0x200d]	undefined[2-2]	2[0x1f468]
keypress	key[0x200d]	undefined[2-2]	2[0x1f468]
textInput	data[0x200d]	undefined[2-2]	2[0x1f468]
input		undefined[3-3]	3[0x1f468 0x200d]
keydown	key[0x1f469]	undefined[3-3]	3[0x1f468 0x200d]
keypress	key[0x1f469]	undefined[3-3]	3[0x1f468 0x200d]
input		undefined[5-5]	5[0x1f468 0x200d 0x1f469]
keydown	key[0x200d]	undefined[5-5]	5[0x1f468 0x200d 0x1f469]
keypress	key[0x200d]	undefined[5-5]	5[0x1f468 0x200d 0x1f469]
textInput	data[0x200d]	undefined[5-5]	5[0x1f468 0x200d 0x1f469]
input		undefined[6-6]	6[0x1f468 0x200d 0x1f469 0x200d]
keydown	key[0x1f467]	undefined[6-6]	6[0x1f468 0x200d 0x1f469 0x200d]
keypress	key[0x1f467]	undefined[6-6]	6[0x1f468 0x200d 0x1f469 0x200d]
input		undefined[8-8]	8[0x1f468 0x200d 0x1f469 0x200d 0x1f467]
keydown	key[0x200d]	undefined[8-8]	8[0x1f468 0x200d 0x1f469 0x200d 0x1f467]
keypress	key[0x200d]	undefined[8-8]	8[0x1f468 0x200d 0x1f469 0x200d 0x1f467]
textInput	data[0x200d]	undefined[8-8]	8[0x1f468 0x200d 0x1f469 0x200d 0x1f467]
input		undefined[9-9]	9[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d]
keydown	key[0x1f466]	undefined[9-9]	9[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d]
keypress	key[0x1f466]	undefined[9-9]	9[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d]
input		undefined[11-11]	11[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d 0x1f466]
keydown	key[0x42 0x61 0x63 0x6b 0x73 0x70 0x61 0x63 0x65]	undefined[11-11]	11[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d 0x1f466]
input		undefined[9-9]	9[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d]
keyup	key[0x42 0x61 0x63 0x6b 0x73 0x70 0x61 0x63 0x65]	undefined[9-9]	9[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d]
change		undefined[9-9]	9[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d]
blur		undefined[9-9]	9[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d]

Family with Chrome 77 + Android.

Event	Key/Data/Mutations	Selection/ranges	Value/InnerHTML
keydown	key[Unidentified]	forward[0-0]	0[]
beforeinput/insertText	data[👉👈]	forward[0-0]	0[]
input/insertText	data[👉👈]	forward[11-11]	11[👉👈]
keyup	key[Unidentified]	forward[11-11]	11[👉👈]
keydown	key[Unidentified]	forward[11-11]	11[👉👈]
beforeinput/deleteContentBackward		forward[0-11]	11[👉👈]
input/deleteContentBackward		forward[0-0]	0[]
keyup	key[Unidentified]	forward[0-0]	0[]

Family codepoints with Chrome 77 + Android.

Event	Key/Data/Mutations	Selection/ranges	Value/InnerHTML
keydown	key[0x55 0x6e 0x69 0x64 0x65 0x6e 0x74 0x69 0x66 0x69 0x65 0x64]	forward[0-0]	0[]
beforeinput/insertText	data[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d 0x1f466]	forward[0-0]	0[]
input/insertText	data[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d 0x1f466]	forward[11-11]	11[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d 0x1f466]
keyup	key[0x55 0x6e 0x69 0x64 0x65 0x6e 0x74 0x69 0x66 0x69 0x65 0x64]	forward[11-11]	11[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d 0x1f466]
keydown	key[0x55 0x6e 0x69 0x64 0x65 0x6e 0x74 0x69 0x66 0x69 0x65 0x64]	forward[11-11]	11[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d 0x1f466]
beforeinput/deleteContentBackward		forward[0-11]	11[0x1f468 0x200d 0x1f469 0x200d 0x1f467 0x200d 0x1f466]
input/deleteContentBackward		forward[0-0]	0[]
keyup	key[0x55 0x6e 0x69 0x64 0x65 0x6e 0x74 0x69 0x66 0x69 0x65 0x64]	forward[0-0]	0[]





A flag is:

- 1 grapheme cluster
- 1 grapheme
- 2 unicode codepoints
- 4 UTF16 code units
- *kinda* like a ligature

...and even though the 2 codepoints can be represented individually, *most* keyboards do delete both at the same time.

Event	Key/Data/Mutations	Selection/ranges	Value/InnerHTML
keydown	key[Unidentified]	forward[0-0]	0[]
beforeinput/insertText	data[🇧🇪]	forward[0-0]	0[]
input/insertText	data[🇧🇪]	forward[4-4]	4[🇧🇪]
keyup	key[Unidentified]	forward[4-4]	4[🇧🇪]
compositionstart		forward[4-4]	4[🇧🇪]
compositionupdate	data[🇧🇪]	forward[4-4]	4[🇧🇪]
keydown	key[Unidentified]	forward[4-4]	4[🇧🇪]
beforeinput/insertCompositionText		forward[0-4]	4[🇧🇪]
compositionupdate		forward[0-4]	4[🇧🇪]
input/deleteContentBackward		forward[0-0]	0[]
compositionend		forward[0-0]	0[]
keyup	key[Unidentified]	forward[0-0]	0[]

Takeaway:

- **What** "backspace" deletes (how many graphemes/codepoints/code units) is decided by the keyboard and the context
- **How** that deletion is translated into events is decided by the web browser

Idem for "delete", idem for cursor keys.

Major takeaway:

Be explicit in what your API handles: Codepoints? UTF8?
UTF16? Graphemes clusters? Keystrokes?

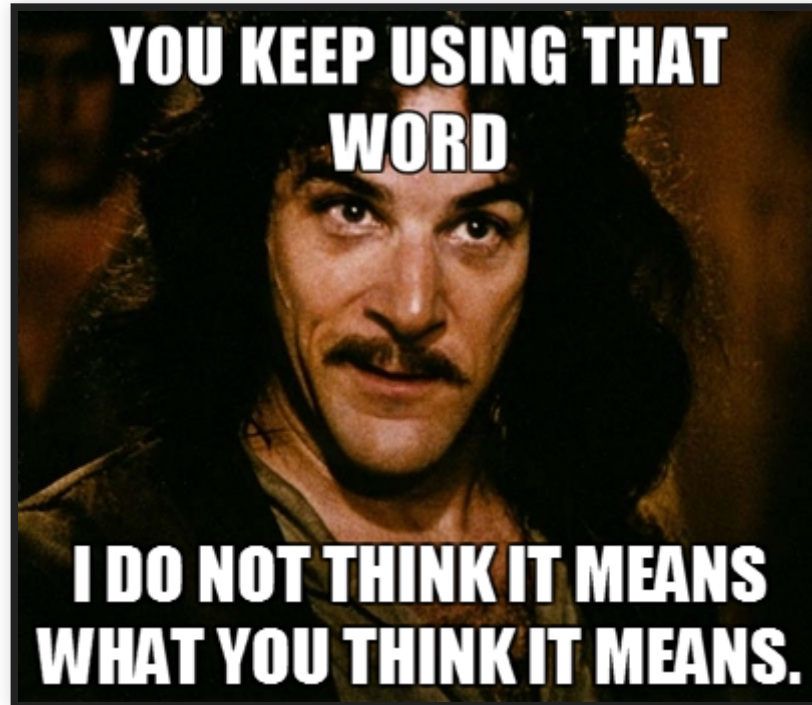
If you can handle inserting/deleting/moving through families and flags, you'll have no problem handling difficult cases like combining diacritics on seemingly obscure scripts.

Major takeaway:

Keeping context in sync in LibreOffice Online is **hard**.

- No way to fetch text content from lowsd
- Workaround: Always keep two blank spaces and the cursor caret in-between to always be able to backspace/delete

Enter.



Enter.

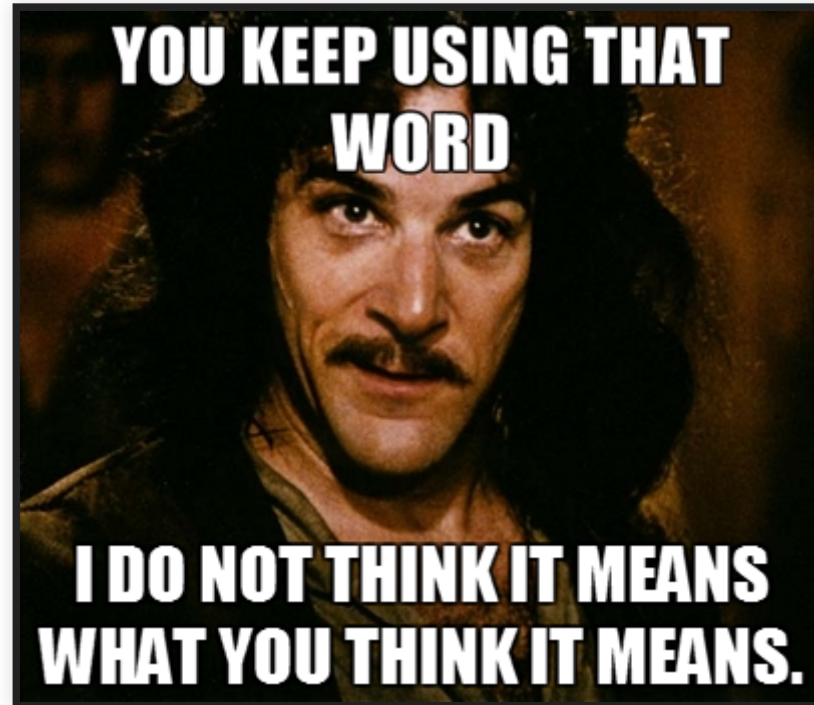
- Create a new paragraph `<p></p>`
- Insert Line Feed + Carriage Return `CRLF`
- Insert a newline `\n`
- Insert a line break `
`
- Send message / commit text
- Accept dialog
- "Click" on focused element

- **What** "enter" does (newline/paragraph/click) is decided by the keyboard and the context
- **How** that is translated into events is decided by the web browser

`input/insertParagraph` happens when on a `<foo contenteditable>`

`input/insertLineBreak` happens when on a `<textarea>`

Word.



Word.

- Any set of adjacent printable characters
- Any set of adjacent graphemes (what about dangling combining diacritics?)
- Any set of non-blank-space characters
- Any set of non-space, non-punctuation characters
- Stuff in-between ASCII regexp "word boundaries"
- Stuff in-between Unicode regexp "word boundaries"
- With/without leading/trailing blank/punctuation
- Is a URL one word or several? And a FQDN?

- **What** "word" means is decided by the keyboard and the context
- **How** that is translated into events (e.g. when **Ctrl+Backspace**) is decided by the web browser

Maybe `input/deleteContentBackward` / `input/deleteWordBackward`.

Idem for `*Forward`.

**Who can fix this
mess?**

Google:

- Google Docs
- Chrome
- GBoard
- Android
- Pixel

Microsoft:

- Office 365
- Edge
- Win10 On-Screen Keyboard
- Win10
- Surface

Who *should* fix this
mess?

- Standard editing software
- Standard web browsers
- Standard IMEs
- Standard OS
- Standard hardware

Where to go from
here?

Actionable

- Diff algorithm to support old (non-`inputType`) browsers
- Abstract events into ("delete" + "insert" + "update composition") bunches
- Apply timeouts and batch at a higher abstraction level
- **Work with browser vendors on the specs!!**

Wishful thinking

- Complete equivalency between `inputType` and `lowSd` interface
- Complete context transfer from `lowSd` to browser
- **Work with browser vendors on the specs!!**

Wishful thinking

- Serverless

Getting on key

Iván Sánchez Ortega

Javascript person working remotely

<http://ivan.sanchezortega.es>

