What's New with KeyboardEvents? Keys and Codes!

By Jeff Posnick
Web DevRel @ Google

The past few versions of Chrome have seen two additions to `KeyboardEvents`, which are used as a parameter passed to **`keydown`**, **`keypress`**, and **`keyup`** event listeners. Both the **`code`** attribute (added in Chrome 48) and the **`key`** attribute (added in Chrome 51) give developers a straightforward way to get information that would otherwise be difficult using legacy attributes.

The code attribute

First up is the **`code`** attribute. This is set to a string representing the key that was pressed to generate the **`KeyboardEvent`**, *without* taking the current keyboard layout (for example, QWERTY vs. Dvorak), locale (for example, English vs. French), or any modifier keys into account. This is useful when you care about which *physical* key was pressed, rather thanwhich character it corresponds to. For example, if you're a writing a game, you might want a certain set of keys to move the player in different directions, and that mapping should ideally be independent of keyboard layout.

The key attribute

Next, we have the new **`key`** attribute. It's also set to a string, but while **`code`** returns information about the physical key that was pressed, **`key`** contains the character that is generated by that key, taking into account the current keyboard layout, locale and modifier keys. Looking at the **`key`** attribute's value comes in handy when you need to know what character would be displayed on the screen as if the user had typed into a text input.

What's this mean in practice?

To give a concrete example, let's assume your user is using a U.S. keyboard with a QWERTY keyboard layout. Pressing the physical `Q` key on that keyboard will result in a **`KeyboardEvent`** with a **`code`** attribute set to "**`KeyQ`**". This is true regardless of keyboard layout, and regardless of any other modifier keys. For comparison, on a French (AZERTY) keyboard this key would still have a **`code`** of "**`KeyQ`**" even though the letter printed on the keycap is an "a". Pressing the physical `Q` key on that same U.S. keyboard will typically generate a **`KeyboardEvent`** with **`key`** set to "**`q`**" (with no modifier keys), or "**`Q`**" (with Shift or CapsLock), or "**`œ`**" (on OS X, with Alt). On a French AZERTY keyboard, this same key would generate an "a" (or "A" with Shift or CapsLock). And for other keyboard layouts, the **`key`** value could be "й", "ض", "ㅂ", "た", or some other character. Revisiting our game example from earlier, if you want your game to use the WASD keys for movement, you can use the **`code`** attribute and check for "**`KeyW`**", "**`KeyA`**", "**`KeyS`**" and "**`KeyD`**". This will work for all keyboards and all layouts—even AZERTY keyboards that swap the position of the "w" and "z" keys.

Virtual keyboards

You'll notice that up until now, we've been focusing on the behavior assuming a physical keyboard. What about users who are typing on a virtual keyboard or an alternative input device? The specification offers official guidance for the **`code`** attribute. To summarize, a virtual keyboard that mimics the layout of a standard keyboard is expected to result in an appropriate **`code`** attribute being set, but virtual keyboards that adopt non-traditional layouts may result in **`code`** not being set at all.

Things are more straightforward for the **`key`** attribute, which you should expect to be set to a string based on which character the user (virtually) typed.

Try it out

Gary Kačmarčík has put together a fantastic demo for visualizing all the attributes associated with **`KeyboardEvent`**s:

Keyboard Event Viewer

UserAgent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2711.0 Safari/537.36

Input: Hello Clear Table Show Options

		Legacy			Modifiers				UI Events							Proposed	
#	Event type	charCode	keyCode	which	shift	ctrl	alt	meta	key	code	location	repeat	isComposing	inputType	data	locale	Input field
22	keyup	0	79 O	79	✗	✗	✗	✗	o	KeyO	0	✗	✗	-	-	-	'Hello'
21	keyup	0	76 L	76	✗	✗	✗	✗	l	KeyL	0	✗	✗	-	-	-	'Hello'
20	input	-	-	-	✗	✗	✗	✗	-	-	-	✗	✗	-	-	-	'Hello'
19	keypress	111 o	111	111	✗	✗	✗	✗	o	KeyO	0	✗	✗	-	-	-	'Hell'
18	keydown	0	79 O	79	✗	✗	✗	✗	o	KeyO	0	✗	✗	-	-	-	'Hell'
17	input	-	-	-	✗	✗	✗	✗	-	-	-	✗	✗	-	-	-	'Hell'
16	keypress	108 l	108	108	✗	✗	✗	✗	l	KeyL	0	✗	✗	-	-	-	'Hel'
15	keydown	0	76 L	76	✗	✗	✗	✗	l	KeyL	0	✗	✗	-	-	-	'Hel'
14	keyup	0	76 L	76	✗	✗	✗	✗	l	KeyL	0	✗	✗	-	-	-	'Hel'
13	input	-	-	-	✗	✗	✗	✗	-	-	-	✗	✗	-	-	-	'Hel'
12	keypress	108 l	108	108	✗	✗	✗	✗	l	KeyL	0	✗	✗	-	-	-	'He'
11	keydown	0	76 L	76	✗	✗	✗	✗	l	KeyL	0	✗	✗	-	-	-	'He'
10	keyup	0	69 E	69	✗	✗	✗	✗	e	KeyE	0	✗	✗	-	-	-	'He'
9	input	-	-	-	✗	✗	✗	✗	-	-	-	✗	✗	-	-	-	'He'
8	keypress	101 e	101	101	✗	✗	✗	✗	e	KeyE	0	✗	✗	-	-	-	'H'
7	keydown	0	69 E	69	✗	✗	✗	✗	e	KeyE	0	✗	✗	-	-	-	'H'
6	keyup	0	16 Shift	16	✗	✗	✗	✗	Shift	ShiftLeft	LEFT	✗	✗	-	-	-	'H'
5	keyup	0	72 H	72	✓	✗	✗	✗	H	KeyH	0	✗	✗	-	-	-	'H'
4	input	-	-	-	✗	✗	✗	✗	-	-	-	✗	✗	-	-	-	'H'
3	keypress	72 H	72	72	✓	✗	✗	✗	H	KeyH	0	✗	✗	-	-	-	''
2	keydown	0	72 H	72	✓	✗	✗	✗	H	KeyH	0	✗	✗	-	-	-	''
1	keydown	0	16 Shift	16	✓	✗	✗	✗	Shift	ShiftLeft	LEFT	✗	✗	-	-	-	''

Cross-browser support

Support for the **code** attribute is, as of this writing, limited to Chrome 48+, Opera 35+, and Firefox 44+. The **key** attribute is supported in Firefox 44+, Chrome 51+, and Opera 38+, with partial support in Internet Explorer 9+ and Edge 13+.

Last updated July 2, 2018.