Theory: Keyboard events handling

1072 users solved this topic. Latest completion was about 2 hours ago.

15 minutes 0 / 5 problems solved

Skip this topic Start practicing

You already know about the existence of keyboard events and one of the ways to handle them. You can write code that handles the pressing of absolutely any key by the user, but how to choose a specific one? That is the question we're going to address in this topic.

§1. Key Codes

In JS you can use `event.code`, which allows you to recognize a pressed key. It does not return the key itself, but its code. A distinction is made between codes for letter keys, numbers and codes for special keys (Enter, Tab). Different syntax is used for each code type. Letter codes are made according to the `Key + Letter` scheme, with the letter name always in upper case. Numeric key codes are built on the principle `Digit + Number`. The name of the special keys is at the same time its code, for instance, `Enter` for the Enter key.

For a more detailed understanding of the syntax, see the table:

key	event.code
A	KeyA
B	KeyB
C	KeyC
1	Digit1
Enter	Enter

The codes for uppercase letters and lowercase letters will be the same. For example, `KeyD` is code for both `D` and `d`. For situations when the case is important, there is `event.key` . `event.key` is a part of the web API offered by the browser:

character	event.key
e	e
E	E

If you are an expert in your keyboard, you probably noticed that some keys are duplicated: on the left and the right sides there are *Alt*, *Ctrl*, *Shift*. You can also work with them. For them, `event.key` remains the same as the key name, but `event.code` is made according to another scheme: `key name` + `Right` or `Left` depending on the location of the desired key.

key	event.code	event.key
Alt	AltRight	Alt
Ctrl	ControlLeft	Control

In this table, you can see examples for *the right Alt key* and *left Ctrl key*.

§2. Event handling

Learning by example is always easier, so let's take a look at actual code and understand how to process pressing certain keys:

```
document.addEventListener("keydown", function(event) {
  if (event.code == "AltRight") {
    console.log(event);
  }
});
```

Current topic:

Keyboard events handling

Topic depends on:

- ✕ Conditional operators
- ✕ The "addEventListener" method

Table of contents:

In this code, we have written a handler for the event `keydown`, which should take place when the *right Alt key* is pressed. For our event handler to work correctly, the `event` parameter was passed to the function (sometimes it is simply designated by the letter `e`). This parameter is a reference to a global object, it is necessary for the browser to place all the data of the current event into it. The result of executing these lines of code will be the output to the console of information that was passed to the `event`. You can verify this by running this code in your browser and pressing the right Alt key:

```
▶ KeyboardEvent {isTrusted: true, key: "Alt", code: "AltRight", location: 2, ctrlKey: false, …}
>
```

You can write a script that allows you to see the code of the pressed key in the console:

```
document.addEventListener("keydown", function(event) {
  console.log(event.code);
});
```

This can help you remember the key code or find an error in your program.

For the full picture, let's look at working with `event.key`. Suppose that we only want to display the console message `"W pressed"` when *only* the capital letter `W` has been pressed. In that case, the code will look like this:

```
document.addEventListener("keydown", function(event) {
  if (event.key == "W") {
    console.log("W Pressed");
  }
});
```

If you try to press the `w` symbol, the code will not be executed.

Use `event.code` when you do not care about case and vice versa, use `event.key` when you care about case. `event.key` is also useful if your application is used by multi-lingual users, because with some layouts `event.code` can be an unexpected character. You can read more about this on the World Wide Web Consortium website.

§3. Conclusion

Learning how to cope with more complicated events with the keyboard is another step forward in learning the basics of the language! Now you can write more complex and user-friendly programs. Just think how many possibilities key event processing opens up for people with disabilities, such as the visually impaired.

Report a typo

92 users liked this theory. **3** didn't like it. **What about you?**

Start practicing

Comments (6) Hints (0) Useful links (0) Show discussion