More Events

Review

- Last class we discussed
  - onclick
  - onload
  - Two ways create event handlers
- Bank Account Example

Exercise

- 1 mile = 1.60934km
- 1 km = 0.621371mi

- toKilometers(miles) that accepts a positive number of miles as a parameter and returns the equivalent number of kilometers
- toMiles(kms) that accepts a positive number of miles as a parameter a returns the equivalent number of miles
- Create two buttons such that when they are clicked each button triggers one of the above functions

Another Way to Handle Events

```
addEventListener ("click" ,
function(){
   console.log("You clicked!");
});
```

- This function registers its second argument to be called whenever the event described by its first argument occurs

The event object

- event is an *object*
  - It holds additional information about the event
  - The info stored in the event object will differ, depending on what event just took place
  - We will talk about objects more in 2-3 weeks

- event.type will always hold a string of info identifying what event occurred (like “click” or “mousedown”)
- if we want to know what key was pressed after a keyboard event, we can use event.keyCode

Event Propagation

- Event handlers that are set on elements with children will also receive some events that happen in the children
  - Ex. If a button inside a paragraph is clicked, event handlers on the paragraph will also receive the click event
- In this example the outer element, the paragraph, is considered the parent element
- The inner element is considered the child element
- if both the paragraph and the button have a handler, the more specific handler, the one on the button, gets to go first

Event Propagation

- The event is then said to **propagate** outward, from the element where it happened to that element's parent element and so on until there are no more parent elements
- At any point, an event handler can call the stopPropagation method on the event object to prevent handlers "further up" from receiving the event
  - event.stopPropagation();
- This can be useful when, for example, you have a button inside another clickable element and you don't want clicks on the button to activate the outer element's click behavior

Keyboard Events

- There are different kinds of events that we can detect from the keyboard
- When a key is pressed on the keyboard, a keydown event occurs
  - Keydown continuously occurs when a key is held down
- When a key is released on the keyboard, a keyup event occurs

Keycodes

- We can identify which key is being pressed/released using the **keycode property** of the event object
- Unfortunately, sometimes it can be tricky to convert keycode to specific keys
- Letter and numbers on the keyboard have a numeric code associated with the pressed key
- You can figure out what a character's keycode is yourself
  - http://keycode.info/
  - The keycode for the V key is 86

Example

```
<p> This page turns violet when you hold the V key.
</p>


<script>
addEventListener("keydown", function(event){
    if (event.keyCode == 86)
        document.body.style.background = "violet";
});

addEventListener("keyup", function(event){
    if (event.keyCode == 86)
        document.body.style.background = "";
}) ;
</ script >
```

Modifier Keys

- Modifier keys such as Shift, Ctrl, Alt generate key events just like normal keys
- When looking for key combinations, you can also find out whether these keys are held down by looking at the shiftKey, ctrlKey, altKey properties of keyboard and mouse events
- Ex.

```
if(event.keyCode==86 && event.ctrlKey)
   console.log("You are pressing ctrl and V");
```

What about keypress?

- keyup and keydown tell us the information about the **physical key being pressed**
- If you're interested in figuring out the actual text being typed, that's where the keypress event becomes useful
- keypress events occur after keydown events (and also fire repeatedly like keydown does when you hold the key)
- **But** keypress **only** fires for keys that produce character input
  - ex. A-Z, 1-0
- We can then look at event.charCode instead of keycode and interpret this value into a Unicode character code

Keypress Example

```
addEventListener("keypress", function(event){
  var ch = event.charCode;
  var letter = String.fromCharCode(ch);
  console.log(letter);
});
```

- In this example, the fromCharCode function turns the character code into an actual single character

IMPORTANT

- Almost always use keyup or keydown
- Only use keypress if you are trying to get the characters being typed

Quiz

- For each of the scenarios below should you use keyup/keydown or keypress?
  - a hangman game where the user types in letters for guesses
  - a game where the user controls the character’s movement with the arrow keys
  - a tower defense game where the user presses space to shoot a cannon
  - a game where the user uses W, A, S, and D to control the character’s direction of movement

Example

- As an exercise in doing ridiculous things with technology let’s try to program a text field that the letters Q, W and X cannot be typed into
- Create the HTML

```
<input type = “text”
id=“censored”>
```

- What event do we want to trigger?

```
<input type = “text”
id=“censored” onkeydown=“???”>
```

A censor() function

- The keycodes for q,w,x are 81, 87, and 88 respectively

```
function censor(event){
if(event.keyCode==81 || event.keyCode==88 ||
event.keyCode==87)
   return false;
}
```

- Remember - return false stops the event from propagating

So…

```
<input type = “text”
id=“censored”
onkeydown=“???”>
```

- What should we put in ???

So…

```
<input type = “text” id=“censored”
onkeydown=“???”>
```

- What should we put in ???

```
onkeydown=“return censor(event);”
```

- We must continue to propagate the event (or not propagate)
- Let’s see if it works!
  - keyboard.html

Exercise