**The problem**

JavaScript is a commonly used web-language to add client side behavior to a website. Because JavaScript runs locally on a user’s machine it responds instantly to user behavior. The experimenter took advantage of this feature to distinguish between actively engaging the website and being idle. However, the same feature can result in errors outside of the experimenter’s control as a result of a subject using outdated or unsupported software. One such error was when recording the amount of time a subject was active on the website. The JavaScript variable storing the information to be sent and stored on the experimenter’s server would not always reset to 0 upon successful transmission. As a result, the recorded time spent was reported larger than reality.

**The Solution**

**Method I**

To account for this “double counting”, we look at each user’s time data. Each user has thousands of time data objects that include meta data, such as a timestamp of when the JavaScript data was received and saved to the database. We ordered the objects by timestamp, with the most recent appearing first. Next, for each, we check that the current timestamp less the number of seconds recorded using JavaScript is not less than the timestamp of the next time data object. If it is, we remove the entry after adding any existing difference in timestamps between to the next time data object.

For each user

For each time data object (sorted by timestamp in descending order):

If (object.timestamp-object.value) < next object.timestamp:

Store object.timestamp – next object.timestamp

Remove object

Else:

Keep object

Add any stored differences to objects value.

**Method II**

To account for this “double counting”, we look at each user’s time data. Each user has thousands of time data objects that include meta data, such as a timestamp of when the JavaScript data was received and saved to the database. We ordered the objects by timestamp, with the most recent appearing first. Next, for each, we check that the current timestamp less the number of seconds recorded using JavaScript is not less than the timestamp of the next time data object. If it is, we replace the value recorded using JavaScript with the difference between the current timestamp and the next object’s timestamp.

For each user

For each time data object (sorted by timestamp in descending order):

If (object.timestamp-object.value) < next object.timestamp:

Replace time value with object.timestamp – next object.timestamp

Else:

Keep object (do nothing)

NOTE: The astute reader might recognize that adding time to next object might result in the next comparison to be erroneously flagged as an error. However, when making comparisons in the IF conditional, any added time from a previous object is not included.