A Browser You’ve Never Heard of Is Dethroning Google in Asia

"A mobile browser rarely used in the West has outflanked Google’s Chrome in some of Asia’s fastest-growing markets, giving owner Alibaba Group Holding Ltd. an advantage in the race among technology giants to capture the next generation of internet users.

“Hundreds of millions of people in India, Indonesia and other emerging markets getting online for the first time are picking UC Browser, owned by Chinese e-commerce giant Alibaba, over ones made by U.S. rivals. Users say UC Browser works better in countries dominated by low-end smartphones and spotty mobile service.

Administrivia

- Assignments:
  - Taijitu due before lab tomorrow (1/11)
  - Reading Check 1 due tomorrow (1/11)
  - Logo Design due Friday (1/12)

- Signing up for Piazza [demo]
Homework: Logo Design

drawing canvas: 400 x 220
colors: purple, gold, white (bg)
Homework: Logo Design

```plaintext
/* uw_logo.pde
   Created by Justin Hsia
   
   UW logo made out of rectangles in school colors.
   */

size(400,220);  // drawing canvas of 400x220
background(255);  // white background

// The letter 'U' in purple
fill( 75, 47, 131);  // purple fill
rect( 20, 20, 40, 180);  // left side of U
rect( 65, 140, 40, 60);  // middle base of U
rect(110, 20, 40, 180);  // right side of U

// The letter 'W' in gold
fill(183, 165, 122);  // gold fill
rect(160, 20, 40, 180);  // left segment of W
rect(205, 140, 40, 60);  // left base of W
rect(250, 100, 40, 90);  // middle segment of W
rect(295, 140, 40, 60);  // right base of W
rect(340, 20, 40, 180);  // right segment of W
```
Drawing a House

- One solution from worksheet:

```plaintext
triangle(7, 1, 3, 5, 11, 5);
rect(3, 5, 8, 8);
rect(8, 9, 2, 4);
```

- What if we wanted to move the house?
Variables

- Piece of your program that holds the value of something
  - Every variable must be given a name and a data type
  - The values of these variables can change (i.e. vary) during the execution of your program
    - Warning: Not like a variable in Algebra (i.e. an unknown)
    - $5x = 20 \implies x = 4$

- Assignment/Write: give a variable a specific value
  - e.g. $x \leftarrow 12$; “put value 12 into the box $x$”
Variables

- Piece of your program that holds the value of something
  - Every variable must be given a name and a data type
  - The values of these variables can change (i.e. vary) during the execution of your program
    - Warning: Not like a variable in Algebra (i.e. an unknown)

- Read: use the current value of a variable
  - e.g. $\text{ellipse}(x+1, 50, 20, 20)$; looks in box $x$ and use its current value
    draws an ellipse with center at $(13, 50)$
Datatypes

- `int` integers (e.g. 12)
- `float` decimal/real numbers (e.g. 3.14)
- `color` RGB (e.g. `color(0)`)
- `char` characters (e.g. 'J')
- `boolean` true or false (e.g. `true`)

- Many more exist and can be found in the Processing Reference:
Declarations

- We declare a variable by telling Processing the variable’s datatype, followed by the variable’s name:

```
1 int x;
2 float half;
3 color yellow;
```

- You can also give a variable a starting value (initialization) in the same line as the declaration:

```
1 int x = 4;
2 float half = 0.5;
3 color yellow = color(255, 255, 0);
```
Variable Manipulation

- Executed sequentially, just like other statements
- For variable assignments, compute right-hand side \textit{first}, then store result in variable
- \textbf{Example:}
  - \texttt{int x = 4;}
  - \texttt{x = x + 1;}

  1) Read the current value of \(x\) (4) for the right-hand side
  2) Add 1 to the current value of \(x\)
  3) Store the result (5) back into \(x\)
Variable Practice

1) \[ \text{int } x = 1; \]
   \[ \text{int } y = 2; \]
   \[ \text{int } z = 3; \]
   \[ x = x + 1; \]
   \[ y = y - 1; \]
   \[ z = z + 2; \]

2) \[ \text{int } x = 7; \]
   \[ \text{int } y = 2; \]
   \[ \text{int } z = 0; \]
   \[ x = x + 3; \]
   \[ y = y - 2; \]
   \[ z = x + y; \]

3) \[ \text{int } x = -1; \]
   \[ \text{int } y = 0; \]
   \[ \text{int } z = 5; \]
   \[ x = x + z; \]
   \[ y = y - x; \]
   \[ z = x + z; \]
Variable Rules & Guidelines

- Variable naming rules:
  - Variables are case-sensitive (e.g. myx vs. myX)
  - Numbers allowed, but not at beginning (e.g. k9 vs. 9k)
  - Generally avoid symbols other than underscore (e.g. my_x)

- Variable names are meaningless to computers, but meaningful to humans
  - Choosing informative names improves readability and reduces confusion

- In this class, most of our variables will be declared and initialized at the very top of our programs
Drawing a House with Variables

```
int houseX = 7;
```

- Initial solution:
  ```
  triangle(7, 1, x-4, 3, 5, 11, 5);
  rect(3, 5, 8, 8);
  rect(8, 9, 2, 4);
  ```

- What properties might be useful to store in variables?
TMNT: Donatello

```plaintext
size(500, 500);
noStroke();
background(255, 245, 220);

// Donatello
fill(0, 100, 0);  // dark green
rect(230, 182, 40, 15);  // top of head

fill(88, 44, 141);  // purple
rect(230, 197, 40, 6);  // bandana mask

fill(0, 100, 0);  // dark green
rect(230, 203, 40, 20);  // bottom of head

fill(219, 136, 0);  // dark yellow
rect(230, 223, 40, 50);  // shell

fill(0, 100, 0);  // dark green
rect(230, 273, 40, 45);  // lower body
```
Donatello with a Variable

```java
int x_pos = 100;  // x-position
size(500,500);
noStroke();
background(255,245,220);

// Donatello
fill(0,100,0);    // dark green
rect(x_pos,182,40,15);  // top of head
fill(88,44,141);  // purple
rect(x_pos,197,40,6);  // bandana mask
fill(0,100,0);    // dark green
rect(x_pos,203,40,20);  // bottom of head
fill(219,136,0);  // dark yellow
rect(x_pos,223,40,50);  // shell
fill(0,100,0);    // dark green
rect(x_pos,273,40,45);  // lower body
```

`x_pos` moves entire drawing!
Donatello with Motion

- Program starts
  - Run `setup()` once
  - Run `draw()` infinitely

Actions:
- Declare and initialize variable (`x_pos`)
- Update variable (`x_pos = x_pos + 1`)

Action Explanation:
- Adding to `x_pos` moves Donatello's body to the right!
Stopping Motion

- Stop Donatello from running off the right side of the screen:
  \[ x_{\text{pos}} = \min(x_{\text{pos}} + 1, 460); \]

<table>
<thead>
<tr>
<th>old ( x_{\text{pos}} )</th>
<th>( x_{\text{pos}} + 1 )</th>
<th>( \min(x_{\text{pos}} + 1, 460) )</th>
<th>new ( x_{\text{pos}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>458</td>
<td>459</td>
<td>459</td>
<td>459</td>
</tr>
<tr>
<td>459</td>
<td>460</td>
<td>460</td>
<td>460</td>
</tr>
<tr>
<td>460</td>
<td>461</td>
<td>460</td>
<td>460</td>
</tr>
</tbody>
</table>

  returns minimum of these two numbers

- Stop Donatello from running off the left side of the screen:
  \[ x_{\text{pos}} = \max(x_{\text{pos}} - 1, 0); \]

  returns maximum of these two numbers

  \( \text{// sets minimum } x_{\text{pos}} \text{ of 0} \)
Falling Into Place

- Introduce variables for each body segment:

```java
3 int head_pos = 0; // head position
4 float mask_pos = 15; // mask position
5 int face_pos = 21; // face position
6 float body_pos = 41; // body position
7 int leg_pos = 91; // leg position
```

Initial y-positions for each body segment

- Update each variable at different speeds:

```java
33 head_pos = min(head_pos + 3, 364);
34 mask_pos = min(mask_pos + 3.5, 379);
35 face_pos = min(face_pos + 4, 385);
36 body_pos = min(body_pos + 4.5, 405);
37 leg_pos = min(leg_pos + 5, 455);
```

Variables that use decimals need to be declared as float

- Higher segments fall slower
- Lower segments fall faster
Falling Into Place

- Update y-positions of drawing based on new variables:

```c
// Donatello
fill(0,100,0); // dark green
rect(x_pos,head_pos,40,15); // top of head

fill(88,44,141); // purple
rect(x_pos,mask_pos,40,6); // bandana mask

fill(0,100,0); // dark green
rect(x_pos,face_pos,40,20); // bottom of head

fill(219,136,0); // dark yellow
rect(x_pos,body_pos,40,50); // shell

fill(0,100,0); // dark green
rect(x_pos,leg_pos,40,45); // lower body
```
Summary

- Variables are named quantities that can vary during the execution of a program.

- Datatypes specific different forms of data
  - e.g. `int`, `float`, `color`, `Boolean`

- Variable *declarations* specify a variable datatype and name to the program
  - Generally occurs at top of program

- `min()` and `max()` functions can be used to limit or stop change in a variable value